INDUSTRIAL AND SYSTEMS ENGINEERING

ISE

A Gradient-Based Optimization Method Using the Koopman Operator

MENGQI HU[1], BIAN LI[2], YI-AN MA[3], YIFEI LOU[1], AND XIU YANG[2]

[1]Department of Mathematics and School of Data Science and Society, University of North Carolina at Chapel Hill, NC USA

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

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

ISE Technical Report 23T-030

LEHIGH UNIVERSITY

A Gradient-Based Optimization Method Using the Koopman Operator *

Mengqi Hu†, Bian Li‡, Yi-An Ma§, Yifei Lou†, and Xiu Yang*

Abstract. In this paper, we propose a novel approach to solving optimization problems by reformulating the optimization problem into a dynamical system, followed by the adaptive spectral Koopman (ASK) method. The Koopman operator, employed in our approach, approximates the evolution of an ordinary differential equation (ODE) using a finite number of eigenfunctions and eigenvalues. We begin by providing a brief overview of the Koopman operator and the ASK method. Subsequently, we adapt the ASK method for solving a general optimization problem. Moreover, we provide an error bound to aid in understanding the performance of the proposed approach, marking the initial step in a more comprehensive numerical analysis. Experimentally, we demonstrate the applicability and accuracy of our method across a diverse range of optimization problems, including min-max problems. Our approach consistently yields smaller gradient norms and higher success rates in finding critical points compared to state-of-the-art gradient-based methods. We also observe the proposed method works particularly well when the dynamical properties of the system can be effectively modeled by the system's behaviors in a neighborhood of critical points.

Key words. Dynamical systems, Koopman operator, spectral-collocation method, gradient flow, min-max optimization

AMS subject classifications. 37N30, 37N40, 37Mxx, 46N10, 47N10

1. Introduction.

Optimization plays a fundamental role in many areas of science, engineering, and machine learning. One of the most fundamental optimization methods is based on the gradient of an objective function since the gradient at a given point provides information about the direction of the steepest increase of the function at that point. By moving along the opposite direction of the gradient, one can iteratively minimize the function, while navigating the search space until a certain type of optimal solution is reached. This technique is particularly effective when dealing with continuously differentiable functions. There are four major categories of gradient-based minimization algorithms, each with its own characteristics and advantages:

- *Gradient Descent* (GD). This is the most basic form of gradient-based optimization. It involves iteratively taking a small step along the opposite direction of the gradient at the current point. The step size is determined by a hyperparameter, which is referred to as a learning rate in deep learning.
- *Gradient Descent with Momentum* (GDM). This technique accelerates the standard

*Submitted to the editors December 20, 2023.

Funding: XY was supported by National Science Foundation (NSF) CAREER DMS-2143915. YL was partially supported by NSF CAREER DMS-1846690. YM was supported in part by the National Science Foundation Grants NSF-SCALE MoDL(2134209) and NSF-CCF-2112665.

†Department of Mathematics & School of Data Science and Society, University of North Carolina at Chapel Hill, NC (humengqi@unc.edu, yflou@unc.edu)

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

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

GD by incorporating a momentum term when advancing to a new iterate. This addition helps to avoid local minima and often leads to faster convergence [38].

- *Stochastic Gradient Descent* (SGD). As a variant of gradient descent, SGD computes the gradient using a random subset of the data at each iteration. The additional randomness enhances the algorithm's robustness to initial conditions and noisy data, facilitating convergence to a critical point [6, 7, 41].
- *Adaptive Gradient Methods* (AGM). To deal with noisy data and non-convex objective functions, AGM chooses the learning rate for each parameter individually based on their historical gradients. It has been demonstrated that this type of algorithm, including Adaptive subgradient methods (AdaGrad) [15] and Adaptive Moment Estimation (Adam) [24], is particularly effective for problems with sparse gradients or ill-conditioned objective functions.

As we explore the realm of optimization using gradient-based methods, we are also intrigued by a specific type of optimization problem that involves the simultaneous minimization of one set of variables and the maximization of another set. This type of optimization is commonly referred to as a min-max problem, however, the duality adds a layer of complexity, akin to fundamental concepts found in game theory. The von Neumann minimax Theorem, a cornerstone in game theory and optimization [46], forms a bridge between min-max optimization problems and zero-sum two-player games. Specifically with finite strategy spaces, this theorem asserts that in a zero-sum two-player game with perfect information, both players' optimal strategies can be determined by solving a min-max optimization problem. This connection highlights the intriguing interactions between optimization and game theory in the pursuit of efficient solutions.

In this work, we develop an innovative gradient-based technique for optimization problems using the adaptive spectral Koopman (ASK) method [29], originally designed for solving ordinary differential equations (ODEs). The ASK method relies on the Koopman operator [25, 26], an infinite-dimensional linear operator for capturing the intricate nonlinearity in dynamical systems described by differential equations. Our approach shifts the perspective on optimization by interpreting it as a dynamical system rather than a standalone problem. This reinterpretation allows us to seek an equilibrium of the system through a finite-dimensional approximation achieved by combining various techniques, including eigen-decomposition, polynomial interpolation, and the Koopman operator. In contrast to traditional optimization methods that typically depend on fixed or adaptively chosen step sizes, the ASK method continuously evolves the system, bringing in the adaptability of the optimization process. This transition can be understood as a shift from time-domain discretization, as found in traditional gradient methods, to spatial-domain discretization. This innovative interpretation of gradient-based algorithms, elucidated by the ASK method, opens the door to potential improvements in the performance and convergence properties of optimization techniques. The main contributions of this paper are threefold.

(1) We propose an efficient workflow using the Koopman operator to obtain an equilibrium of a dynamical system, which coincides with a critical point of the corresponding optimization problem.
(2) By further integrating the idea of sparse grids [30] into our method, we significantly reduce the computational costs of the proposed method. Additionally, the implemen-

tation of an adaptive scheme allows our approach to handle complex problems with increased efficiency. We further analyze an error bound for a special case.

(3) We conduct extensive experiments on standard testing functions in both minimization and min-max problems, demonstrating that the proposed algorithm achieves significant improvements in accuracy and success rates compared with some existing gradient-based optimization techniques.

The rest of the paper is organized as follows. Section 2 provides a concise review of the relevant literature in optimization, the Koopman operator, and the ASK method. In Section 3 we describe the proposed approach with a particular focus on the modifications made to the ASK method. Section 4 is devoted to numerous examples including three bowl-shaped testing functions, four valley-shared functions, two high-dimensional functions, and four min-max problems to showcase the efficiency of the proposed algorithm in comparison to various baseline algorithms. Finally, Section 5 concludes the paper.

2. Literature Review.

We present the formulations of a general minimization and a min-max problem in Subsection 2.1, along with gradient-based optimization algorithms. In Subsection 2.2, we provide a brief review of the concepts of the Koopman operator. Subsection 2.3 is devoted to its applications in solving an ODE via an adaptive spectral method.

2.1. Optimization.

Consider a general unconstrained minimization problem, defined by

$$\min_{\boldsymbol{x}\in\mathbb{R}^d} f(\boldsymbol{x}), \tag{2.1}$$

where $f : \mathbb{R}^d \to \mathbb{R}$ is a continuously differentiable function to be optimized, preferably a smooth C^∞ function. Traditionally, gradient-based methods are employed to solve (2.1) by gradually decreasing the objective function f following the negative gradient direction. The most straightforward approach to solving the problem as a dynamical system is through a generic formula of gradient descent (GD) algorithms, which aims to minimize $f(\boldsymbol{x})$. This formula can be expressed as

$$\begin{cases} \boldsymbol{p}^{(k+1)} &= -\nabla f(\boldsymbol{x}^{(k)}) \\ \boldsymbol{x}^{(k+1)} &= \boldsymbol{x}^{(k)} + \alpha^{(k+1)}\boldsymbol{p}^{(k+1)}, \end{cases} \tag{2.2}$$

where k denotes the iteration number and $\alpha^{(k+1)} > 0$ represents a positive step size that can either be fixed or updated iteratively. Numerous variants of gradient descent exist, differentiated by how the step size is determined. For instance, steepest descent (SD) involves an exact line search to achieve the maximum descent along the gradient direction, making the descent the steepest. Its procedure can be outlined as follows,

$$\begin{cases} \boldsymbol{p}^{(k+1)} &= -\nabla f(\boldsymbol{x}^{(k)}) \\ \alpha^{(k+1)} &= \arg\min_\alpha f(\boldsymbol{x}^{(k)} + \alpha\boldsymbol{p}^{(k+1)}) \\ \boldsymbol{x}^{(k+1)} &= \boldsymbol{x}^{(k)} + \alpha^{(k+1)}\boldsymbol{p}^{(k+1)}. \end{cases} \tag{2.3}$$

However, SD does not work well empirically in most cases, since such a local descending property does not necessarily coincide with the overall descending of the original function.

Noting that the search direction in each iteration of the scheme (2.3) solely relies on the information at the current step $\boldsymbol{x}^{(k)}$, one can also incorporate previous steps into the iteration, which gives rise to momentum-based algorithms [8, 22]. The term "momentum" draws an analogy to a massive ball rolling on the surface of the objective function, where the update of each step is retained and utilized during the process. Consequently, the following iteration

$$\begin{cases} \boldsymbol{p}^{(k+1)} &= -\nabla f(\boldsymbol{x}^{(k)}) + \beta^{(k+1)}\boldsymbol{p}^{(k)} \\ \boldsymbol{x}^{(k+1)} &= \boldsymbol{x}^{(k)} + \alpha^{(k+1)}\boldsymbol{p}^{(k+1)}, \end{cases} \tag{2.4}$$

is referred to as a heavy ball (HB) algorithm [38]. Both $\alpha^{(k+1)}$ and $\beta^{(k+1)}$ in Equation (2.4) can be either fixed or adaptively selected based on a specific scheme. A distinct category of momentum-based algorithms was developed by Yurii Nesterov [36, 37]. Beginning with $t^{(0)} = 1$, Nesterov's accelerated gradient (NAG) is formulated by,

$$\begin{cases} t^{(k+1)} &= \frac{1+\sqrt{4(t^{(k)})^2+1}}{2} \\ \boldsymbol{p}^{(k+1)} &= -\nabla f\left(\boldsymbol{x}^{(k)}\right) \\ \boldsymbol{y}^{(k+1)} &= \boldsymbol{x}^{(k)} + \alpha^{(k+1)}\boldsymbol{p}^{(k+1)} \\ \boldsymbol{x}^{(k+1)} &= \boldsymbol{y}^{(k+1)} + \frac{t^{(k)}-1}{t^{(k+1)}}(\boldsymbol{y}^{(k+1)} - \boldsymbol{y}^{(k)}). \end{cases} \tag{2.5}$$

Similar to other gradient-based algorithms, the step size $\alpha^{(k+1)}$ in NAG can be constant or updated during the iteration. For any convex objective function $f(\cdot)$, NAG achieves a convergence rate of $O(\frac{1}{k^2})$, which is an improvement over the rate of $O(\frac{1}{k})$ obtained by standard gradient-based methods. This momentum scheme can be further accelerated by employing a suitable restart strategy with provable guarantees under certain conditions [35, 18, 44].

We are also interested in a min-max problem formulated as follows,

$$\min_{\boldsymbol{x}\in\mathbb{R}^m}\max_{\boldsymbol{y}\in\mathbb{R}^n} f(\boldsymbol{x},\boldsymbol{y}), \tag{2.6}$$

for a continuously differentiable function $f(\cdot,\cdot)$. Solving (2.6) by gradient-based methods amounts to iterating between gradient descend on $\boldsymbol{x}$ and gradient ascend on $\boldsymbol{y}$. In particular, a gradient descend/ascend (GDA) algorithm [41] is expressed as follows,

$$\begin{cases} \boldsymbol{x}^{(k+1)} &= \boldsymbol{x}^{(k)} - \alpha^{(k+1)}\nabla_{\boldsymbol{x}} f(\boldsymbol{x}^{(k)},\boldsymbol{y}^{(k)}) \\ \boldsymbol{y}^{(k+1)} &= \boldsymbol{y}^{(k)} + \alpha^{(k+1)}\nabla_{\boldsymbol{y}} f(\boldsymbol{x}^{(k)},\boldsymbol{y}^{(k)}), \end{cases} \tag{2.7}$$

where $\alpha^{(k+1)} > 0$ represents a positive step size that can either be fixed or updated iteratively. Unfortunately, there exist instances where the system of equations in (2.7) exhibits a cyclic behavior. Motivated by a specific function f with an initial point that leads to such cycles, Daskalakis et al. [13] proposed an optimistic gradient descent/ascent (OGDA) approach,

$$\begin{cases} \boldsymbol{x}^{(k+1)} &= \boldsymbol{x}^{(k)} - \alpha^{(k+1)}\nabla_{\boldsymbol{x}} f(\boldsymbol{x}^{(k)},\boldsymbol{y}^{(k)}) - \alpha^{(k+1)}(\nabla_{\boldsymbol{x}} f(\boldsymbol{x}^{(k)},\boldsymbol{y}^{(k)}) - \nabla_{\boldsymbol{x}} f(\boldsymbol{x}^{(k-1)},\boldsymbol{y}^{(k-1)})) \\ \boldsymbol{y}^{(k+1)} &= \boldsymbol{y}^{(k)} + \alpha^{(k+1)}\nabla_{\boldsymbol{y}} f(\boldsymbol{x}^{(k)},\boldsymbol{y}^{(k)}) + \alpha^{(k+1)}(\nabla_{\boldsymbol{y}} f(\boldsymbol{x}^{(k)},\boldsymbol{y}^{(k)}) - \nabla_{\boldsymbol{y}} f(\boldsymbol{x}^{(k-1)},\boldsymbol{y}^{(k-1)})), \end{cases} \tag{2.8}$$

and established its convergence for bilinear objective functions, i.e., $f(\boldsymbol{x},\boldsymbol{y}) = \boldsymbol{x}^{\mathsf{T}} A\boldsymbol{y}$ [13]. It is worth noting that OGDA resembles the HB method [38, 8], but a key distinction lies in the presence of the "negative momentum" in OGDA, while the HB method employs a "positive momentum." In addition to gradient-based methods, alternative methods for solving the min-max problem (2.6) are documented in [8, 4, 28, 39].

All the aforementioned algorithms can be understood in the framework of solving a certain ODE defined by,

$$\begin{cases} \mathcal{D}\boldsymbol{x}(t) = \boldsymbol{u}(\boldsymbol{x}) \\ \boldsymbol{x}(t_0) = \boldsymbol{x}_0, \end{cases} \tag{2.9}$$

where $\mathcal{D}$ is a linear differential operator, $\boldsymbol{u}$ is a continuous function which is usually chosen as $\boldsymbol{u}(\boldsymbol{x}) = -\nabla f(\boldsymbol{x})$ for minimization and $\boldsymbol{u}(\boldsymbol{x},\boldsymbol{y}) = -\nabla f(\boldsymbol{x},-\boldsymbol{y})$ for min-max problems. It is straightforward that the gradient descend method (2.2) can be formulated by the forward Euler scheme. As for the momentum acceleration methods, the HB formula (2.4) applied the Euler's method to the following ODE [52],

$$\ddot{\boldsymbol{x}} + \epsilon\dot{\boldsymbol{x}} + \nabla f(\boldsymbol{x}) = \boldsymbol{0}, \tag{2.10}$$

where ϵ can be viewed as a viscosity coefficient in a damping system. From an optimization perspective, the ideal value for ϵ is associated with the Lipschitz and strong convexity properties of the function f [43]. Additionally, NAG is related to the ODE [44]

$$\ddot{\boldsymbol{x}} + \frac{3}{t}\dot{\boldsymbol{x}} + \nabla f(\boldsymbol{x}) = \boldsymbol{0}. \tag{2.11}$$

Note that it is necessary for $\nabla f(\mathbf{x})$ to be Lipschitz continuous with a Lipschitz constant L, and the step size should satisfy $\alpha < \frac{1}{cL}$ for some constant c to ensure the guaranteed acceleration achieved by these methods. This requirement implies that the step size α should be carefully tuned, and its optimal value may vary from one problem to another [44]. There is a wealth of research on continuous gradient flows and their acceleration schemes, such as (2.10) (2.11), using higher-order ODEs [42, 47]. However, most of them still adopt traditional numerical ODE methods that require careful tuning of the step size.

2.2. Koopman operator. We review the Koopman operator [26, 31], which maps a finite-dimensional nonlinear system to a linear one. This mapping enables us to utilize the solution of an infinite-dimensional linear system to find an equilibrium of the nonlinear system. Define a family of the Koopman operators by

$$\mathcal{K}_t g(\boldsymbol{x}(t_0)) = g\Big(\boldsymbol{x}(t_0 + t)\Big), \tag{2.12}$$

where g is any (scalar) observable function of the state variables $\boldsymbol{x}(t)$ at time t. The "derivative" of $\mathcal{K}_t$, referred to as the *infinitesimal generator* (or simply generator) of the Koopman operators, is defined by

$$\mathcal{K}g = \lim_{t\to 0} \frac{\mathcal{K}_t g - g}{t}. \tag{2.13}$$

As analogous to the chain rule in calculus, one has

$$\mathcal{K}g(\boldsymbol{x}) = \nabla g(\boldsymbol{x}) \cdot \frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = \frac{\mathrm{d}g(\boldsymbol{x})}{\mathrm{d}t}. \tag{2.14}$$

Suppose φ is an eigenfunction of $\mathcal{K}$ with the associated eigenvalue λ, i.e., $\mathcal{K}\varphi(\boldsymbol{x}) = \lambda\varphi(\boldsymbol{x})$. It follows from (2.14) that $\lambda\varphi(\boldsymbol{x}) = \mathcal{K}\varphi(\boldsymbol{x}) = \frac{d\varphi(\boldsymbol{x})}{dt}$, which implies that $\varphi(\boldsymbol{x}(t_0 + t)) = \exp(\lambda t)\varphi(\boldsymbol{x}(t_0))$. It follows from the definition of the Koopman operator (2.12) that

$$\mathcal{K}_t\varphi(\boldsymbol{x}(t_0)) = \varphi(\boldsymbol{x}(t_0 + t)) = \exp(\lambda t)\varphi(\boldsymbol{x}(t_0)). \tag{2.15}$$

If g belongs to the function space spanned by all the eigenfunctions φ_j of $\mathcal{K}_t$, i.e., $g(\boldsymbol{x}) = \sum_j^\infty c_j\varphi_j(\boldsymbol{x})$ with coefficients $\{c_j\}$, the linearity of $\mathcal{K}_t$ gives

$$\mathcal{K}_t[g(\boldsymbol{x}(t_0))] = \mathcal{K}_t[\sum_j^\infty c_j\varphi_j(\boldsymbol{x}(t_0))] = \sum_j^\infty c_j\mathcal{K}_t[\varphi_j(\boldsymbol{x}(t_0))]. \tag{2.16}$$

The above equation implies that

$$g(\boldsymbol{x}(t_0 + t)) = \sum_j^\infty c_j\varphi_j(\boldsymbol{x}(t_0))\exp(\lambda_j t), \tag{2.17}$$

where $\{(\varphi_j, \lambda_j)\}_j^\infty$ are eigenpairs of the Koopman operator $\mathcal{K}_t$, each satisfying (2.15).

2.3. Adaptive Spectral Koopman (ASK) methods. Li et al. [29] proposed a truncated version of (2.17) for solving the dynamical system governed by $\dot{\boldsymbol{x}}(t) = \boldsymbol{u}(\boldsymbol{x})$, where $\boldsymbol{u} \in \mathbb{R}^d$ is often called the dynamics of the state variable $\boldsymbol{x} \in \mathbb{R}^d$. Given an integer N, suppose each eigenfunction φ_j of $\mathcal{K}_t$ can be approximated by interpolation polynomials, denoted as φ_j^N. Then, a finite-dimensional approximation of g in (2.17) can be expressed by

$$g(\boldsymbol{x}(t_0 + t)) \approx g^N(\boldsymbol{x}(t_0 + t)) := \sum_{j=0}^N \tilde{c}_j\varphi_j^N(\boldsymbol{x}(t_0))\exp(\tilde{\lambda}_j t), \tag{2.18}$$

where λ_j, c_j are approximated by $\tilde{\lambda}_j$ and $\tilde{c}_j$, respectively. For each eigenfunction φ_j^N, Li et al. [30] further considered a finite interpolation on a set of (sparse) grid points. Specifically, a finite approximation of an arbitrary function φ is given by

$$\varphi(\boldsymbol{x}) \approx \varphi^N(\boldsymbol{x}) = \sum_{l=1}^N w_l\Psi_l(\boldsymbol{x}), \tag{2.19}$$

where $\{\Psi_l\}$ are Chebyshev polynomials. When evaluating $\boldsymbol{x}$ on a set of sparse grid points, denoted by $\{\boldsymbol{\xi}_l\}_{l=1}^N$ with $\boldsymbol{\xi}_l := (\xi_{l_1}, \xi_{l_2}, \ldots, \xi_{l_d}) \in \mathbb{R}^d$, (2.19) can be written by a matrix-vector multiplication, i.e.,

$$\begin{bmatrix} \varphi^N(\boldsymbol{\xi}_1) \\ \varphi^N(\boldsymbol{\xi}_2) \\ \vdots \\ \varphi^N(\boldsymbol{\xi}_N) \end{bmatrix} = \begin{bmatrix} \Psi_1(\boldsymbol{\xi}_1) & \Psi_2(\boldsymbol{\xi}_1) & \ldots & \Psi_N(\boldsymbol{\xi}_1) \\ \Psi_1(\boldsymbol{\xi}_2) & \Psi_2(\boldsymbol{\xi}_2) & \ldots & \Psi_N(\boldsymbol{\xi}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \Psi_1(\boldsymbol{\xi}_N) & \Psi_2(\boldsymbol{\xi}_N) & \ldots & \Psi_N(\boldsymbol{\xi}_N) \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \end{bmatrix}, \tag{2.20}$$

or $\boldsymbol{\varphi}^N = \mathbf{M}\boldsymbol{w}$ for short, where each element of $\mathbf{M}$ is given by $M_{ij} = \Psi_j(\boldsymbol{\xi}_i)$ and $\boldsymbol{w} = (w_1, \cdots, w_N)^\mathsf{T}$. Note that φ denotes a function, while the bold-faced version $\boldsymbol{\varphi}$ denotes the function evaluated at a set of points.

Taking partial derivatives of (2.19) with respect to x_i yields

$$\frac{\partial \varphi^N(\boldsymbol{x})}{\partial x_i} = \sum_{l=1}^{N} w_l \frac{\partial \Psi_l(\boldsymbol{x})}{\partial x_i}. \tag{2.21}$$

Denote the matrix $\mathbf{G}_i$ as $\partial_{x_i}\Psi_l(\boldsymbol{x})$ evaluated at $\{\boldsymbol{\xi}_l\}_{l=1}^N$, i.e.,

$$\mathbf{G}_i = \begin{bmatrix} \frac{\partial \Psi_1}{\partial x_i}(\boldsymbol{\xi}_1) & \frac{\partial \Psi_2}{\partial x_i}(\boldsymbol{\xi}_1) & \cdots & \frac{\partial \Psi_N}{\partial x_i}(\boldsymbol{\xi}_1) \\ \frac{\partial \Psi_1}{\partial x_i}(\boldsymbol{\xi}_2) & \frac{\partial \Psi_2}{\partial x_i}(\boldsymbol{\xi}_2) & \cdots & \frac{\partial \Psi_N}{\partial x_i}(\boldsymbol{\xi}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \Psi_1}{\partial x_i}(\boldsymbol{\xi}_N) & \frac{\partial \Psi_2}{\partial x_i}(\boldsymbol{\xi}_N) & \cdots & \frac{\partial \Psi_N}{\partial x_i}(\boldsymbol{\xi}_N) \end{bmatrix} \in \mathbb{R}^{N\times N}, \tag{2.22}$$

then (2.21) can be expressed by

$$\frac{\partial \varphi^N(\boldsymbol{x})}{\partial x_i} = \mathbf{G}_i \boldsymbol{w}. \tag{2.23}$$

Now we regard each component of the dynamic $\boldsymbol{u}$ as an observable that the Koopman operator operates on, i.e., $\boldsymbol{u} := (g_1(\boldsymbol{x}), g_2(\boldsymbol{x}), ..., g_d(\boldsymbol{x}))^\mathsf{T}$, then a system of observables becomes

$$\frac{\mathrm{d}\boldsymbol{u}(\boldsymbol{x})}{\mathrm{d}t} = \mathcal{K}\boldsymbol{u}(\boldsymbol{x}) = \begin{bmatrix} \mathcal{K}g_1(\boldsymbol{x}) \\ \mathcal{K}g_2(\boldsymbol{x}) \\ \cdots \\ \mathcal{K}g_d(\boldsymbol{x}) \end{bmatrix} = \sum_{j}^{\infty} \lambda_j \varphi_j(\boldsymbol{x}) \boldsymbol{c}_j, \tag{2.24}$$

where $\boldsymbol{c}_j = (c_j^1, \cdots, c_j^d)^\mathsf{T} \in \mathbb{C}^d$ is called the j-th Koopman mode of the system. It follows from (2.14) that $\mathcal{K}\varphi(\boldsymbol{x}) = \frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} \cdot \nabla\varphi(\boldsymbol{x})$ for any function φ. Since $\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = \boldsymbol{u}(\boldsymbol{x})$, we have

$$\mathcal{K}\varphi = \boldsymbol{u} \cdot \nabla\varphi = g_1 \frac{\partial \varphi}{\partial x_1} + g_2 \frac{\partial \varphi}{\partial x_2} + \ldots + g_d \frac{\partial \varphi}{\partial x_d}. \tag{2.25}$$

It follows from (2.23) that the Koopman operator $\mathcal{K}$ given in (2.25) has a finite approximation by $\mathbf{K} \in \mathbb{R}^{N\times N}$, that is,

$$\mathbf{K}\varphi^N = \sum_{i=1}^{d} \mathrm{diag}\big(g_i(\boldsymbol{\xi}_1), \ldots, g_i(\boldsymbol{\xi}_N)\big)\, \mathbf{G}_i \boldsymbol{w}, \tag{2.26}$$

which holds for the function φ^N as an approximation to the function φ as in (2.19). The choice of φ^N determines the coefficient vector $\boldsymbol{w}$.

Suppose the discretized Koopman operator $\mathbf{K}$ has an eigenpair $(\varphi_j^N, \tilde{\lambda}_j)$, i.e., $\mathbf{K}\varphi_j^N = \tilde{\lambda}_j \varphi_j^N$. Applying (2.26) on φ_j^N yields

$$\sum_{i}^{d} \mathrm{diag}\big(g_i(\boldsymbol{\xi}_1), \ldots, g_i(\boldsymbol{\xi}_N)\big)\, \mathbf{G}_i \boldsymbol{w}_j = \mathbf{K}\varphi_j^N = \tilde{\lambda}_j \mathbf{M}\boldsymbol{w}_j, \tag{2.27}$$

or shortly $\mathbf{U}\boldsymbol{w}_j = \tilde{\lambda}_j \mathbf{M}\boldsymbol{w}_j$ with $\mathbf{U} := \sum_i^d \operatorname{diag}\big(g_i(\boldsymbol{\xi}_1), \ldots, g_i(\boldsymbol{\xi}_N)\big)\, \mathbf{G}_i$. Given $\mathbf{U}$ and $\mathbf{M}$, the coefficients $\boldsymbol{w}_j$ can be solved by a generalized eigenvalue problem. For compactness, we write it in a matrix form

$$\mathbf{U}\mathbf{W} = \mathbf{M}\mathbf{W}\boldsymbol{\Lambda}, \tag{2.28}$$

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

Setting $t = 0$ in (2.18) with a particular component g_i in the dynamics $\boldsymbol{u}$, we can compute Koopman modes $\tilde{c}_{ji}$ from

$$g_i(\boldsymbol{x}_0) \approx \sum_{j=1}^{N} \tilde{c}_{ji} \varphi_j^N(\boldsymbol{x}_0) =: g_i^N(\boldsymbol{x}_0),$$

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

$$g_i^N(\boldsymbol{\xi}_l) = \sum_{j=1}^{N} \tilde{c}_{ji} \varphi_j^N(\boldsymbol{\xi}_l), \quad l = 1, 2, \ldots, N, \tag{2.29}$$

which can be expressed in a matrix form $\boldsymbol{\Phi}^N \tilde{\boldsymbol{c}}_i = \boldsymbol{\Xi}_i$ with

$$\boldsymbol{\Xi}_i := \begin{bmatrix} g_i(\boldsymbol{\xi}_1) \\ \vdots \\ g_i(\boldsymbol{\xi}_l) \\ \vdots \\ g_i(\boldsymbol{\xi}_N) \end{bmatrix} \quad \text{and} \quad \tilde{\boldsymbol{c}}_i = \begin{bmatrix} \tilde{c}_{1i} \\ \vdots \\ \tilde{c}_{li} \\ \vdots \\ \tilde{c}_{Ni} \end{bmatrix}. \tag{2.30}$$

Putting $\boldsymbol{\Xi}_i$ and $\tilde{\boldsymbol{c}}_i$ as the ith column of the matrices $\boldsymbol{\Xi}$ and $\tilde{\mathbf{C}}$, respectively, we can solve $\tilde{\mathbf{C}}$ from $\boldsymbol{\Phi}^N \tilde{\mathbf{C}} = \boldsymbol{\Xi}$. As $\boldsymbol{\xi}_1 = \boldsymbol{x}_0$ by construction, $\varphi_j^N(\boldsymbol{x}_0)$ is the first element of vector $\boldsymbol{\varphi}_j^N$, denoted by $(\boldsymbol{\varphi}_j^N)_1$. Thus, the solution of the dynamical system $\dot{\boldsymbol{x}}(t) = \boldsymbol{u}(\boldsymbol{x})$ can be approximated by

$$\boldsymbol{x}(t) = \sum_{j=1}^{N} \tilde{\boldsymbol{c}}_j \varphi_j^N(\boldsymbol{x}_0) e^{\tilde{\lambda}_j t} = \sum_{j=1}^{N} \tilde{\boldsymbol{c}}_j (\boldsymbol{\varphi}_j^N)_1 e^{\tilde{\lambda}_j t}. \tag{2.31}$$

3. The proposed approach.

We regard a critical point of the optimization problem (2.1) as a steady state of the following dynamical system,

$$\begin{cases} \dot{\boldsymbol{x}}(t) = \boldsymbol{u}(\boldsymbol{x}) \\ \boldsymbol{x}(t_0) = \boldsymbol{x}_0, \end{cases} \tag{3.1}$$

where $\boldsymbol{u}(\boldsymbol{x}) := -\nabla f(\boldsymbol{x})$ and $\boldsymbol{x}_0$ is an initial point. In other words, we find a critical point of (2.1) by evolving the corresponding dynamical system (3.1) for a sufficiently long time to

arrive at the equilibrium that satisfies $\nabla f(\boldsymbol{x}) = \mathbf{0}$. As for the min-max problem (2.6), its corresponding dynamical system is

$$\begin{cases} \dot{\boldsymbol{x}}(t) = -\nabla_{\boldsymbol{x}} f(\boldsymbol{x}, \boldsymbol{y}) \\ \dot{\boldsymbol{y}}(t) = \nabla_{\boldsymbol{y}} f(\boldsymbol{x}, \boldsymbol{y}) \\ \boldsymbol{x}(t_0) = \boldsymbol{x}_0 \\ \boldsymbol{y}(t_0) = \boldsymbol{y}_0, \end{cases} \tag{3.2}$$

which can be written in terms of (3.1) with $\bar{\boldsymbol{x}} = (\boldsymbol{x}, \boldsymbol{y})^\mathsf{T}$ and $\boldsymbol{u}(\bar{\boldsymbol{x}}) = (-\nabla_{\boldsymbol{x}} f(\boldsymbol{x}, \boldsymbol{y}), \nabla_{\boldsymbol{y}} f(\boldsymbol{x}, \boldsymbol{y}))^\mathsf{T}$. Consequently, we only focus on the general optimization problem (2.1) and its corresponding dynamical system (3.1).

If the dynamical system (3.1) is linear, one can express it as

$$\begin{cases} \dot{\boldsymbol{x}}(t) = A\boldsymbol{x}(t) \\ \boldsymbol{x}(0) = \boldsymbol{x}_0, \end{cases} \tag{3.3}$$

where A is a finite-dimensional time invariant matrix. There is a closed-form solution of (3.3) given by Duhamel's formula [45], i.e.,

$$\boldsymbol{x}(t) = \exp(tA)\boldsymbol{x}_0. \tag{3.4}$$

Motivated by the fact that a linear dynamical system has a closed-form solution, we propose to employ the adaptive spectral Koopman (ASK) method [29] that robustly linearizes an arbitrary dynamical system with its steady state serving as a critical point of the corresponding optimization problem.

Before presenting our approach, we highlight the differences between solving ODEs and optimization problems. First, unlike ODE evolution, which closely tracks trajectories, optimization algorithms target the steady state—local minima for minimization problems and saddle points in min-max problems. This distinction affects parameter selection, such as the order/level of approximation (i.e., N) and length of evolution time (i.e., T). The choice of these parameters in optimization is more flexible than in ODEs, as the optimization focuses on the long-term behaviors of the system rather than precise trajectory tracking. This flexibility allows for a rough approximation of the dynamics in a neighborhood of the current state using mostly linear functions instead of higher-order polynomials. Second, it enables us to extend the time window for numerical integration without being overly concerned about errors introduced by the truncation in the finite-dimensional approximation. Our approach takes a distinct perspective, aiming to stably evolve the system towards equilibrium in the long term.

In Subsection 3.1, we describe our workflow of finding a steady state of the dynamical system (3.1) via ASK, highlighting important steps discussed in Subsection 2.3 and our modifications. Theoretical analysis on error bounds is conducted in Subsection 3.2.

3.1. Workflow. We consider each component of the dynamics $\boldsymbol{u}$ in (3.1) as an observable g that defines the Koopman operator (2.12), thus leading to a system of observables (2.24).

To leverage the properties of the Koopman operator, we consider its finite-dimensional approximation (2.18) and hence the solution of (3.1) can be obtained by a linear combination of the approximated eigenfunctions $\{\varphi_j^N\}$ with corresponding eigenvalues $\{\tilde{\lambda}_j\}$. As $t \to \infty$, the solution $\boldsymbol{x}(t)$ given by (2.31) approaches a critical point to the optimization problem (2.1). We choose $t = T$ for a sufficiently large value of T.

However, the accuracy of the solution (2.31) may deteriorate due to the finite-dimensional approximation of $\mathcal{K}$ and the local approximation of φ in the vicinity of $\boldsymbol{x}_0$, which is notably pronounced in systems characterized by highly nonlinear dynamics. To mitigate this issue, we periodically update an acceptable range for each component

$$R_i := [L_i, U_i], \tag{3.5}$$

where L_i and U_i symbolize the lower and upper bounds, respectively, and r denotes a flexible radius parameter. The interval $[L_i, U_i]$ determines the construction of grid points $\{\boldsymbol{\xi}_l\}_{l=1}^N$ used in (2.20). In particular, we construct the *sparse* grid points [30] on the reference domain $[-1, 1]$ to define the matrices $\{\mathbf{G}_i\}$ in (2.22) and $\{\boldsymbol{\Xi}_i\}$ in (2.30), which can then be rescaled to match the real domain $[L_i, U_i]$ by $\frac{2\mathbf{G}_i}{U_i - L_i}$ and $\frac{U_i - L_i}{2}(\boldsymbol{\Xi}_i + 1) + L_i$, respectively, for $i = 1, \cdots, d$. By construction, the central point of the domain is the first generated sparse grid point, e.g., $(0, 0, \ldots, 0) = \boldsymbol{\xi}_1$ for a multidimensional domain $[-1, 1]^d$. In order to avoid the situation of $\boldsymbol{x}_0 \notin \{\boldsymbol{\xi}_l\}$, we consider a neighborhood of $\boldsymbol{x}_0$ defined by $[\boldsymbol{x}_0 - \boldsymbol{r}, \boldsymbol{x}_0 + \boldsymbol{r}]$, where $\boldsymbol{r} = (r_1, r_2, \ldots, r_d)^\mathsf{T}$ is the radius of the neighbor so that the center of the neighborhood is always within the sparse grid. For simplicity, we adopt the isotropic setting, i.e., $r_1 = r_2 = \cdots = r_d = r$.

Initially, we set $L_i = x_i - r$ and $U_i = x_i + r$. For the current state, denoted as $\boldsymbol{x}(t) = (x_1(t), x_2(t), \ldots, x_d(t))^\mathsf{T}$, we define the neighborhood $R_1 \times R_2 \times \ldots \times R_d$ as valid if all components $x_i(t)$ fall within their respective ranges R_i. However, should any component lie outside its range, i.e., $x_i(t) \notin R_i$, we implement a straightforward retraction procedure by halving t i.e. $t = \frac{t}{2}$. Our choice for the retraction by $t = \frac{t}{2}$ is primarily motivated by its simple implementation, but other methods are worth exploring. For instance, we can leverage bifurcation concepts to extend T until it approaches the boundary of one of the intervals R_i. This approach harnesses the full potential of the dynamics in the neighborhood, but it does not guarantee improved performance compared to a straightforward bifurcation approach. This is because the error introduced by numerical integration tends to be larger at the boundaries of the interval, and the range of feasible parameters is narrower in such cases.

By setting the end state of the current iteration as the initial condition to the next one, we rescale $\{\mathbf{G}_i\}$ and $\{\boldsymbol{\Xi}_i\}$ that are defined on $[-1, 1]$ by $\frac{2\mathbf{G}_i}{U_i - L_i}$ and $\frac{U_i - L_i}{2}(\boldsymbol{\Xi}_i + 1) + L_i$ to deal with the new interval $[L_i, U_i]$, thus getting a new solution (2.31) after eigendecomposition and adjustment to make the obtained solution within the bounds $[L_i, U_i]$. We stop the iterations until a stopping criterion is met or the maximum iteration number is achieved. The overall algorithm is summarized in Algorithm 3.1.

Note that we approximate the eigenfunctions using multi-dimensional orthogonal polynomials through the spectral method. Consequently, the Koopman operator's generator is discretized as a differentiation matrix. Similar approaches have been proposed for studying the behavior of differential equations [9, 10, 16]. Furthermore, these eigenfunctions can be constructed using various approximation approaches. For example, they can be built based on the kernel of a reproducing kernel Hilbert space [12, 17] in the data-driven setting. A

Algorithm 3.1 Sparse Grid Adaptive Spectral Koopman Method

Require: $n, T, \boldsymbol{x}_0, r, \kappa, \epsilon$

Set $k = 0, \boldsymbol{x}^{(0)} = \boldsymbol{x}_0$.
Generate sparse grid points $\{\boldsymbol{\xi}_l\}_{l=1}^N$
while $k < n$ or $\|\boldsymbol{u}(\boldsymbol{x}^{(k)})\| > \epsilon$ **do**
  Let $L_i = x_i - r, U_i = x_i + r$, set neighborhood R_i as $R_i = [L_i, U_i]$ for $i = 1, 2, ..., d$.
  Compute $\mathbf{G}_i$ at $\boldsymbol{x}^{(k)}$ in (2.22) to obtain $\mathbf{U} = \sum_i^d \operatorname{diag}\big(g_i(\boldsymbol{\xi}_1), \ldots, g_i(\boldsymbol{\xi}_N)\big)\, \mathbf{G}_i$.
  Given $\mathbf{M}$ in (2.20) and $\mathbf{U}$, apply eigen-decomposition $\mathbf{UW} = \mathbf{MW\Lambda}$ to get $\mathbf{\Phi}^N = \mathbf{MW}$.
  Solve linear system $\mathbf{\Phi}^N \mathbf{C} = \mathbf{\Xi}$, where $\mathbf{\Xi}$ is defined in (2.30).
  Let $t = T$.
  Let ν_j be the first element of the jth column of $\mathbf{\Phi}$. Construct solution at t as

$$\boldsymbol{x}(t) = \sum_j \mathbf{C}(j,:)\nu_j \exp\left(\tilde{\lambda}_j(t)\right),$$

  where $\mathbf{C}(j,:)$ is the jth row of $\mathbf{C}$ and $\tilde{\lambda}_j$ is the jth eigenvalue of $\mathbf{\Lambda}$.
  while $x_i(t) \notin R_i$ for any i **do**
    Set t = t/2.
    Re-evaluate $\boldsymbol{x}(t) = \sum_j \mathbf{C}(j,:)\nu_j \exp\left(\tilde{\lambda}_j(t)\right)$.
  end while
  Evolve the current solution to the time t, i.e., $\boldsymbol{x}^{(k)} = \boldsymbol{x}(t)$
  $k = k + 1$.
end while
return $\boldsymbol{x}^{(k)}$

more in-depth exploration of the properties of the Koopman operator and its applications to differential equations can be found, e.g., in [31, 1, 27, 33, 34, 11, 51, 50, 14, 2]. Certain methods discussed therein hold the potential to enhance the efficiency and accuracy of our approach, and these will be incorporated into our future work.

3.2. Numerical Analysis. In general, conducting a comprehensive numerical analysis of the ASK-based optimization algorithm is a complex task. The error in the ASK expansion for a dynamical system includes the following three components:

1. Errors in discretizing the Koopman generator $\mathcal{K}$. Despite being a linear operator, it is noteworthy that the eigenvalues of $\mathcal{K}$ only constitute a discrete spectrum. Consequently, when we employ a matrix $\mathbf{K}$ to approximate $\mathcal{K}$, we may lose information in the continuous spectrum of $\mathcal{K}$ beyond the errors introduced in the approximation of eigenvalues.
2. Errors in approximating global functions locally. The eigenfunction φ_j is defined on the entire state space, but in computation, we approximate it within a bounded domain. Consequently, employing local information to approximate global functions, such as eigenfunctions and Koopman modes, can result in inaccuracies. This is the rationale

behind the introduction of the adaptivity step in the design of ASK. Through this step, we aim at a more accurate approximation of global functions φ_j across various bounded domains within the state space.

3. Errors in approximating eigenfunctions φ_j using polynomials. We employ the standard spectral method to approximate eigenfunctions φ_j using polynomials φ_j^N within a bounded domain, a practice that may result in errors in the approximation. Similarly, alternative approximation approaches, such as radial basis function approximation, Fourier series approximation, and wavelet approximation, can introduce various numerical errors.

In this work, we focus on addressing the third type of error mentioned earlier, specifically investigating the approximation in the vicinity of a critical point by using our algorithm. Our objective is to leverage numerical analysis tools in both the spectral method and the sparse grids method to provide error estimates, which allows us to gain a preliminary understanding of the convergence of the proposed method, as outlined in Theorem 3.1.

Theorem 3.1. *Consider a (truncated) Koopman expansion of a scalar function $g \in \mathbb{R}^d \to \mathbb{R}$, i.e.,*

$$g(\boldsymbol{x}(t_0+t)) = \sum_{j=0}^{N} c_j \varphi_j(\boldsymbol{x}(t_0)) \exp(\lambda_j t),$$

and its ASK approximation $\sum_{j=0}^{N} \tilde{c}_j \varphi_j^N(\boldsymbol{x}(t_0)) \exp(\tilde{\lambda}_j t)$ in a bounded neighborhood of $\boldsymbol{x}(t_0)$, denoted as $B_{\boldsymbol{x}(t_0)}(r) := \{\boldsymbol{y} \in \mathbb{R}^d | \|\boldsymbol{x}(t_0) - \boldsymbol{y}\|_\infty \le r\}$, where eigenpairs (φ_j, λ_j) are approximated by $(\varphi_j^N, \tilde{\lambda}_j)$. Define $\varepsilon_\varphi = \max_{0\le j\le N}\{\|\varphi_j - \varphi_j^N\|_{B_{\boldsymbol{x}(t_0)}(r),\infty}\}$, $\varepsilon_\lambda = \max_{0\le j\le N}\{|\lambda_j - \tilde{\lambda}_j|\}$, and the matrix $\boldsymbol{\Phi}$ with each element $\Phi_{ij} = \varphi_j(\boldsymbol{\xi}_i)$ with its condition number denoted by κ. If λ_j and $\tilde{\lambda}_j$ have nonpositive real part, $\mathrm{Re}(\tilde{\lambda}_j) < 0$ for $\mathrm{Re}(\lambda_j) < 0$, and $\kappa(N+1)\varepsilon_\varphi < \|\boldsymbol{\Phi}\|_\infty$, we have:

$$\left\| \sum_{j=0}^{N} c_j \varphi_j(\boldsymbol{x}(t_0)) \exp(\lambda_j t) - \sum_{j=0}^{N} \tilde{c}_j \varphi_j^N(\boldsymbol{x}(t_0)) \exp(\tilde{\lambda}_j t) \right\| \lesssim C_1 \varepsilon_\lambda t + C_2 \varepsilon_\varphi, \tag{3.6}$$

where both C_1 and C_2 depends on $\|\boldsymbol{c}\|_\infty$ and $\max_j |\varphi_j(x(t_0))|$ with C_2 additionally dependent on condition number of $\boldsymbol{\Phi}$ and N.

Proof. The Koopman modes c_j satisfy $\boldsymbol{\Phi}\boldsymbol{c} = \boldsymbol{g}$ with $g_i = g(\boldsymbol{\xi}_i)$. Since φ_j are approximated by φ_j^N, the perturbation of $\boldsymbol{\Phi}$, defined as $\delta\boldsymbol{\Phi} = \boldsymbol{\Phi} - \boldsymbol{\Phi}^N$, satisfies $\|\delta\boldsymbol{\Phi}\|_\infty \le (N+1)\varepsilon_\varphi$. Recall that $\boldsymbol{\Phi}^N\tilde{\boldsymbol{c}} = \boldsymbol{g}$, and hence we get

$$\frac{\|\boldsymbol{c} - \tilde{\boldsymbol{c}}\|_\infty}{\|\boldsymbol{c}\|_\infty} \le \frac{\kappa\|\delta\boldsymbol{\Phi}\|_\infty}{\|\boldsymbol{\Phi}\|_\infty - \kappa\|\delta\boldsymbol{\Phi}\|_\infty} \le \frac{\kappa(N+1)\varepsilon_\varphi}{\|\boldsymbol{\Phi}\|_\infty - \kappa(N+1)\varepsilon_\varphi} := \varepsilon_\infty, \tag{3.7}$$

where we use the assumption that $\kappa(N+1)\varepsilon_\varphi < \|\boldsymbol{\Phi}\|_\infty$. If ε_φ is sufficiently small such that $\kappa(N+1)\varepsilon_\varphi \ll \|\boldsymbol{\Phi}\|_\infty$, then $\varepsilon_\infty \approx \frac{\kappa(N+1)}{\|\boldsymbol{\Phi}\|_\infty}\varepsilon_\varphi$. Define a set $S = \{j \in \{0,1,\cdots,N\} | \mathrm{Re}(\lambda_j) < 0\}$ and

$S^c = \{0, 1, \cdots, N\} \backslash S$. By the assumption that $\mathrm{Re}(\lambda_j)$ and $\mathrm{Re}(\tilde{\lambda}_j)$ are nonpositive, we have

$$
\begin{aligned}
&\left\| \sum_{j \in S} c_j \varphi_j(\boldsymbol{x}(t_0)) \exp(\lambda_j t) - \sum_{j \in S} \tilde{c}_j \varphi_j^N(\boldsymbol{x}(t_0)) \exp(\tilde{\lambda}_j t) \right\| \\
\leq & (N+1) \left(\|\boldsymbol{c}\|_\infty \exp(\lambda_{\max} t) + \|\tilde{\boldsymbol{c}}\|_\infty \exp(\tilde{\lambda}_{\max} t) \right) \\
\leq & (N+1) \|\boldsymbol{c}\|_\infty \left(\exp(\lambda_{\max} t) + (1 + \varepsilon_\infty) \exp(\tilde{\lambda}_{\max} t) \right),
\end{aligned} \tag{3.8}
$$

where $\lambda_{\max} = \max_{j \in S}\{\mathrm{Re}(\lambda_j)\} < 0$ and $\tilde{\lambda}_{\max} = \max_{j \in S}\{\mathrm{Re}(\tilde{\lambda}_j)\} < 0$. Therefore, the magnitude of the difference in (3.8) converges to zero exponentially as $t \to \infty$.

By the assumption that $\mathrm{Re}(\tilde{\lambda}_j) < 0$ for $\mathrm{Re}(\lambda_j) < 0$, we have $\mathrm{Re}\,\tilde{\lambda}_j \leq \mathrm{Re}\,\lambda_j = 0$ for $j \in S^c$, which implies that

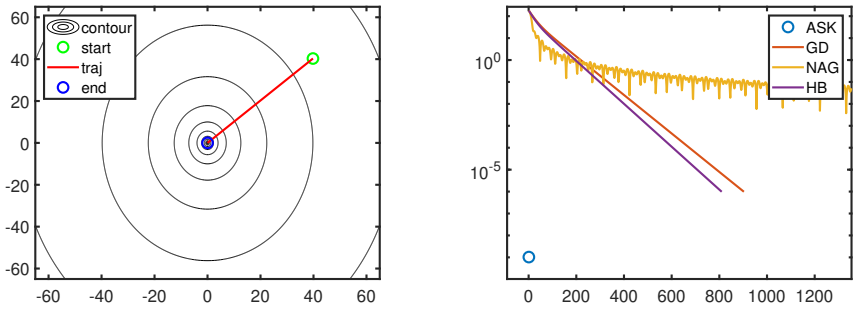
$$
\begin{aligned}
&\left| c_j \varphi_j(\boldsymbol{x}(t_0)) \exp(\lambda_j t) - \tilde{c}_j \varphi_j^N(\boldsymbol{x}(t_0)) \exp(\tilde{\lambda}_j t) \right| \\
\leq & \left| c_j \varphi_j(\boldsymbol{x}(t_0)) \exp(\lambda_j t) - c_j \varphi_j(\boldsymbol{x}(t_0)) \exp(\lambda_j t) \exp((\tilde{\lambda}_j - \lambda_j) t) \right. \\
& \left. + (c_j \varphi_j(\boldsymbol{x}(t_0)) - \tilde{c}_j \varphi_j^N(\boldsymbol{x}(t_0))) \exp(\tilde{\lambda}_j t) \right| \\
\leq & |c_j \varphi_j(\boldsymbol{x}(t_0)) \exp(\lambda_j t)| \cdot \left| 1 - \exp((\tilde{\lambda}_j - \lambda_j) t) \right| + |c_j \varphi_j(t_0) - c_j \varphi_j^N(t_0)| + |c_j \varphi_j^N(t_0) - \tilde{c}_j \varphi_j^N(t_0)| \\
\leq & |c_j \varphi_j(\boldsymbol{x}(t_0))| \cdot \left| 1 - \exp((\tilde{\lambda}_j - \lambda_j) t) \right| + \|\boldsymbol{c}\|_\infty \varepsilon_\varphi + \|\boldsymbol{c}\|_\infty \varepsilon_\infty (\max\{\varphi_j(t_0)\} + \varepsilon_\varphi).
\end{aligned}
$$

Therefore, when $|\tilde{\lambda}_j - \lambda_j|$ is sufficiently small, we have $\left| 1 - \exp((\tilde{\lambda}_j - \lambda_j) t) \right| = \mathcal{O}(\varepsilon_\lambda t)$. In practice, for a given small number ϵ, we identify t such that the difference in (3.8) is upper bounded by ϵ, thereby establishing the validity of the desired bound in (3.6). ∎

Of note, in Theorem 3.1, it is a requirement that $(\varphi_j^N, \tilde{\lambda}_j)$ serves as an accurate approximation of (φ_j, λ_j). Utilizing the spectral method, we generally observe that when eigenfunctions φ_j exhibit good regularity—meaning high-order (mixed) derivatives exist and are bounded—we can anticipate a high level of accuracy in the approximation, as shown in the study of eigenvalue problems by using the spectral method. However, the majority of such works are associated with self-adjoint operators under specific boundary conditions, a setup that differs from the one in ASK. Therefore, a comprehensive error estimate requires careful investigation and will be addressed in future work. In the ideal case, we may expect ε_λ and ε_φ to be at the order of $\mathcal{O}(n^{-k})$ with the accuracy of interpolation based on sparse grid [3], where n is the total number of sparse grids points and $\varphi_j \in C^k(B_{\boldsymbol{x}(t_0)}(r))$.

4. Numerical Results.

We present extensive numerical examples to demonstrate the performance of the proposed ASK method for solving minimization problems (2.1) in Section 4.1 and min-max problems (2.6) in Section 4.2. All of our numerical experiments are conducted using 100 randomly chosen initial points. We assess the efficiency of each competing algorithm based on both $\|\nabla f(\boldsymbol{x})\|$ and time consumption. For convex test functions, we include the comparison to the CVX package [19, 20], using the error to the global minimum as an additional evaluation metric. All the experiments are performed in MATLAB 2022a. We refer to a textbook [48] to implement these baseline algorithms in MATLAB. No other software or public libraries were used in our experiments. The computing platform is a personal computer with Ryzen 7 5800x, 32 GB of RAM, and the operating system of Windows 11.



(a) Trajectory of ASK method

(b) $\|\nabla f(\boldsymbol{x}^{(k)})\|$ v.s. k.

Figure 1: Algorithmic behaviors on Rotated Hyper-Ellipsoid function. As a semi-continuous algorithm, the proposed ASK method is able to reach the global optimum in one iteration.

Methods	$\|\nabla f\|$	Error	Time
ASK	1.0303e-09	3.6223e-10	4.8347e-03
GD	9.8228e-07	4.8131e-07	8.0462e-03
NAG	2.5607e-03	1.0887e-03	6.6670e-02
HB	9.8054e-07	4.7935e-07	6.4078e-03
CVX	3.8696e-28	9.6740e-29	2.3438e-01

Table 1: Rotated Hyper-Ellipsoid, init=$130 \times \mathrm{rand}(2,1) - 65$.

For the proposed method, we adopt the maximum iteration (i.e., adaptivity steps) number of 5×10^4. There are two major parameters that need to be tuned: one is the radius r of the acceptable range defined in (3.5), and the other is the maximum degree of basis polynomial function Ψ, which we refer to as the level of approximation. The acceptable range r is set to 10^{-1} and the level of approximation in the sparse grids method is set to 1 if not mentioned otherwise. In general, for level l, we generate $2^l + 1$ one-dimensional quadrature points and then use them to construct high-dimensional sparse grids points based the Smolyak structure. A method is considered to have successfully converged if it achieves a gradient norm $\|\nabla f(\boldsymbol{x})\|$ less than 10^{-6}. The reported results reflect the average outcomes over all such successful trials.

4.1. Minimization problems. We investigate bowl-shaped functions in Subsection 4.1.1 and valley-shaped functions in Subsection 4.1.2 focusing on 2-dimensional functions. In addition, two higher-dimensional examples are discussed in Subsection 4.1.3. All these functions are widely used in optimization for testing purposes. We compare the proposed ASK method with three baseline approaches: gradient descent (GD) in (2.2), heavy ball (HB) in (2.4), and Nesterov's accelerated gradient (NAG) in (2.5).

Methods	$\|\nabla f\|$	Error	Time
ASK	9.9966e-07	5.7725e-04	7.8031e-02
GD	3.2950e-05	3.3138e-03	1.0769e-01
NAG	2.2653e-03	3.9177e-03	7.0358e-02
HB	2.6712e-05	2.9836e-03	7.0009e-02
CVX	0.0000e+00	0.0000e+00	4.3750e-01

Table 2: The sum of different powers function, init=$2 \times \text{rand}(2,1) - 1$.

4.1.1. Bowl-shaped testing functions. We start with the rotated Hyper-Ellipsoid function [32], which is defined by

$$f(x_1, x_2) = \sum_{i=1}^{2} \sum_{j=1}^{i} x_j^2. \tag{4.1}$$

This function is convex and has a unique minimum of $f(\boldsymbol{x}^*) = 0$ at $\boldsymbol{x}^* = (0,0)$. An initial point is randomly chosen within the grid $[-65, 65] \times [-65, 65]$. Figure 1(a) illustrates a notable advantage of the proposed algorithm: it converges in just one iteration. This example showcases a scenario where local behaviours of the dynamics closely align with the global dynamics, enabling the identification of the global minimum in a single iteration. It also emphasizes the remarkable potential of our proposed method when applied in such circumstances. Figure 1(b) depicts the norm of the gradient at each iteration, confirming once again that the ASK algorithm converges in a single iteration. In addition, NAG demonstrates faster convergence than GD and HB initially but slows down as it approaches the target location. The quantitative comparison Table 1 indicates that all competing algorithms successfully find the global minimum. Our proposed algorithm converges closer to the global minimum than the three gradient-based algorithms with the least amount of computational time. While the CVX package achieves the highest accuracy, it is also the slowest.

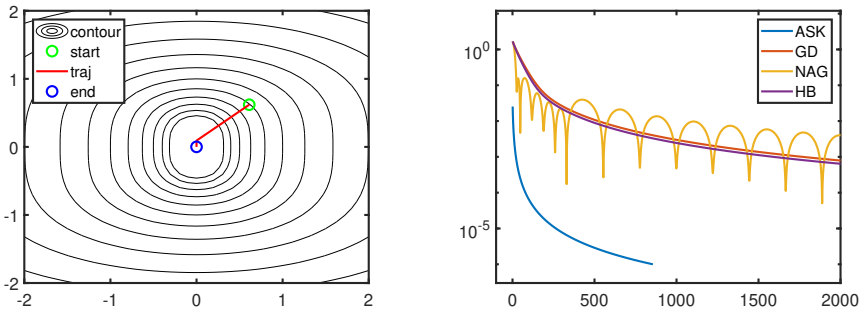
The second testing function is the sum of different powers [32], given by

$$f(x_1, x_2) = \sum_{i=1}^{2} |x_i|^{i+1}. \tag{4.2}$$

It has a unique minimum of $f(\boldsymbol{x}^*) = 0$ at $\boldsymbol{x}^* = (0,0)$. Due to the presence of $|x_2|^3$, the function is not differentiable. We choose an initial point randomly from $[-1, 1] \times [-1, 1]$. Figure 2(a) illustrates that the evolution of our proposed algorithm is relatively smooth, despite the function itself not being smooth, specifically, not differentiable. Both Figure 2 and Table 2 demonstrate that the proposed algorithm achieves the highest accuracy in terms of the norm of the gradient and error to the global solution with the least amount of computation time. The result obtained by CVX is included, showing that CVX yields the exact solution but at the cost of a longer runtime.

Lastly, we consider the second Bohachevsky function [5], labeled as Bohachevsky 2. It is defined by

$$f(x_1, x_2) = x_1^2 + x_2^2 - 0.3\cos(3\pi x_1)\cos(4\pi x_2) + 0.3. \tag{4.3}$$



(a) Trajectory of ASK method

(b) $\|\nabla f(\boldsymbol{x}^{(k)})\|$ v.s. k.

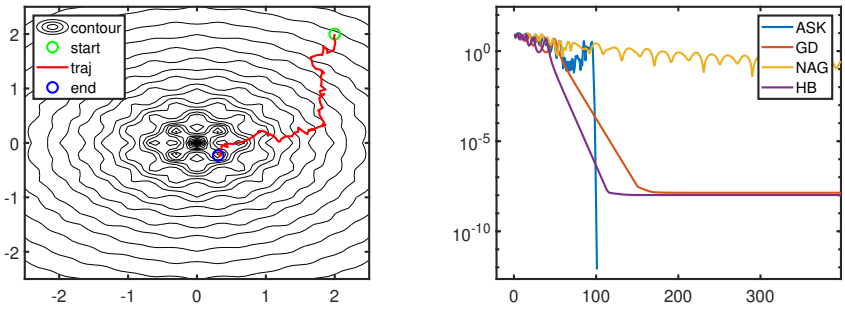
Figure 2: Algorithmic behaviors on the sum of different powers function. The proposed algorithm converges the fastest to the solution with the smallest norm of the gradient. Despite the objective function is not smooth, the evolution of ASK is rather smooth.

Methods	$\|\nabla f\|$	Time
ASK	**3.7616e-14**	1.1111e-01
GD	5.2897e-09	2.1634e-03
NAG	1.4387e-01	6.2574e-03
HB	9.5994e-10	**1.1356e-03**

Table 3: Bohachevsky 2 function, init$=4\times\text{rand}(2,1)-2$.

As illustrated in Figure 3(a), the function is non-convex and takes the form of a bowl with oscillatory level sets near the global minimum. Specifically, it features numerous local minima and a global minimum of $f(\boldsymbol{x}^*)=0$ at $\boldsymbol{x}^*=(0,0)$. For this function, the level of approximation is set to 3 rather than the default 1 due to the intricate dynamics around the critical points. We randomly choose initial points in the domain $[-5,5]\times[-5,5]$. When an initial point is far from a local extreme cluster, the proposed algorithm converges to a local minimum.

Figure 3(a) displays the trace of the proposed ASK method on the contour plot, revealing a zigzag path when approaching a critical point. Figure 3(b) compares the competing algorithms in terms of the norm of the gradient at each iteration, indicating that all the methods converge to a critical point except NAG. This is anticipated, as the NAG method may have adverse effects when applied to a non-convex function [49]. The proposed ASK achieves the highest accuracy among the competing methods. Table 3 confirms that ASK results in the smallest gradient norm, yet, unfortunately, it is the slowest due to the intricate dynamic in the neighbourhood that necessitate numerous eigen-decomposition calculations. This behavior is observed in Figure 3(b), where the magnitude of $\|\nabla f\|$ oscillates between 10^{-1} and 10^{1} before rapidly dropping below 10^{-10} after approximately 100 iterations.



(a) Trajectory of ASK method (b) $\|\nabla f(\boldsymbol{x}^{(k)})\|$ v.s. k.

Figure 3: Algorithmic behaviors on Bohachevsky 2 function. All the methods converge to a critical point except for NAG, and ASK achieves the highest accuracy in terms of the gradient norm.

Methods	$\|\nabla f\|$	Time
ASK	**7.9837e-09**	**8.0243e-03**
GD	6.6493e-08	7.9303e-02
NAG	4.5331e-04	5.6113e-02
HB	1.1336e-08	5.5693e-02

Table 4: Three-hump Camel function, init=$10 \times \text{rand}(2,1) - 5$.

4.1.2. Valley-Shaped functions. The first valley-shaped testing function we consider is called the three-hump Camel function [32], defined by

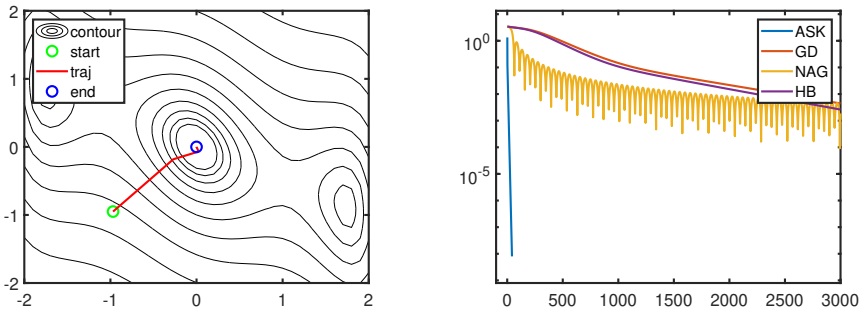
$$f(x_1, x_2) = 2x_1^2 - 1.05x_1^4 + \frac{x_1^6}{6} + x_1x_2 + x_2^2. \tag{4.4}$$

This nonconvex function has a global minimum of $f(\boldsymbol{x}^*) = 0$ at $\boldsymbol{x}^* = (0,0)$, two local minima, and two saddle points. The initial point is chosen randomly from $[-5,5] \times [-5,5]$. As shown in Figure 4 and Table 4, the proposed algorithm efficiently approaches a critical point in just a few iterations, making it the most time-effective. Additionally, it outperforms other competing methods in terms of the gradient norm.

We then consider a six-hump Camel function [21], defined by

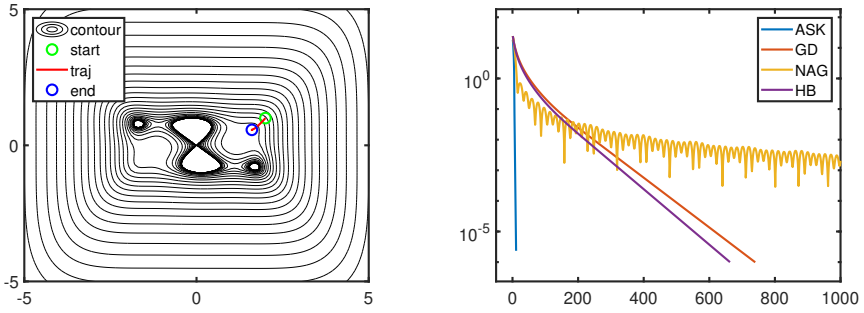
$$f(x_1, x_2) = \left(4 - 2.1x_1^2 + \frac{x_1^4}{3}\right)x_1^2 + x_1x_2 + (-4 + 4x_2^2)x_2^2, \tag{4.5}$$

as a testing function. This function is non-convex and has a global minimum of approximately $f(\boldsymbol{x}^*) = -1.0316$ at $(0.0898, -0.7126)$ and $(-0.0898, 0.7126)$, along with four other local minima. We select random initial points from $[-3,3] \times [-2,2]$ and present the results in Figure 5



(a) Trajectory of ASK method (b) $\|\nabla f(\boldsymbol{x}^{(k)})\|$ v.s. k.

Figure 4: Algorithmic behaviors on the three-hump Camel function. The proposed algorithm converges the fastest to a critical point.



(a) Trajectory of ASK method (b) $\|\nabla f(\boldsymbol{x}^{(k)})\|$ v.s. k.

Figure 5: Algorithmic behaviors on the six-hump Camel function. The proposed algorithm converges the fastest to a critical point.

and Table 5. Figure 5 illustrates that the proposed algorithm requires only a few iterations to converge. Table 5 indicates that ASK slightly outperforms other methods in terms of the gradient norm. While all the methods have comparable computation times, as listed in Table 5, the complexity per ASK iteration is rather high due to the eigen-decomposition.

The third valley-shaped testing function is the Dixon-Price function [21, 23], defined by

$$f(\boldsymbol{x}) = (x_1 - 1)^2 + \sum_{j=2}^{d} j(2x_j^2 - x_{j-1})^2. \tag{4.6}$$

The function has a global minimum of $f(\boldsymbol{x}^*) = 0$ at $x_j^* = 2^{-\frac{2^j-2}{2^j}}$ for $j = 1, \cdots, d$. We examine the case of $d = 2$ here, $d = 10$ in Subsection 4.1.3, and set $d = 100$ as a min-max problem in Subsec-

Methods	$\|\nabla f\|$	Time
ASK	5.3550e-07	9.3056e-03
GD	9.9561e-07	7.4152e-03
NAG	2.9041e-03	6.5248e-03
HB	9.8639e-07	5.1908e-03

Table 5: Six-hump Camel Function, init=$6 \times \text{rand}(2,1) - 3$.

Methods	$\|\nabla f\|$	Time
ASK	**9.4133e-08**	**6.7219e-03**
GD	1.5514e-04	1.1269e+00
NAG	3.1235e-01	4.3161e-02
HB	5.6322e-07	9.7755e-02

Table 6: Dixon-Price function, init=$20 \times \text{rand}(2,1) - 10$.

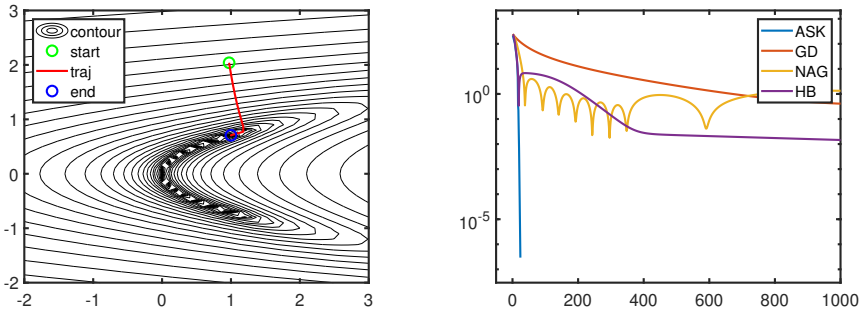
tion 4.2. We randomly choose initial points from $[-10,10] \times [-10,10]$. Table 6 demonstrates that our proposed method achieves nearly three orders of magnitude improvement over GD, while being three orders of magnitude faster. NAG does not converge to any critical point, as indicated by a large gradient norm. The HB method achieves a result similar to ASK but takes ten times longer. Figure 6 aligns with Table 6, showing that ASK converges in a few iterations, while NAG diverges.

The last valley-shaped testing function is the Rosenbrock function [40, 21] given as

$$f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (x_1 - 1)^2. \tag{4.7}$$

This function has a global minimum of $f(\boldsymbol{x}^*) = 0$ at $\boldsymbol{x}^* = (1,1)$. We start with a random initial point in $[-2,2] \times [-2,2]$. The quantitative behavior is presented in Table 7, showing that ASK yields the highest accuracy in terms of the gradient norm with the least amount of time. Figure 7(a) illustrates the trajectory of the ASK method on a contour plot, while Figure 7(b) reveals a barrier in the path where the gradient's norm is between 10^{-1} and 10^{-2}, signifying that all the competing methods significantly slow down. It takes ASK about 100 iterations to overcome this barrier. Naive GD and HB methods are able to pass through this barrier, while NAG fails in this case. We postulate that this phenomenon might stem from the intricate dynamics surrounding the crescent-shaped non-convex set where local minima are located. Additionally, we note that we set the level of approximation to 3 for this example, rather than the default choice of 1, as we believe a higher-order approximation is required to achieve better results.

4.1.3. High-dimensional examples. We investigate two high-dimensional testing functions for minimization. The first testing function is a 10-dimensional Dixon-Price function defined in (4.6) with $d = 10$. The comparison results are summarized in Table 8, indicating that the proposed algorithm returns the gradient norm closest to 0 with orders of magnitude

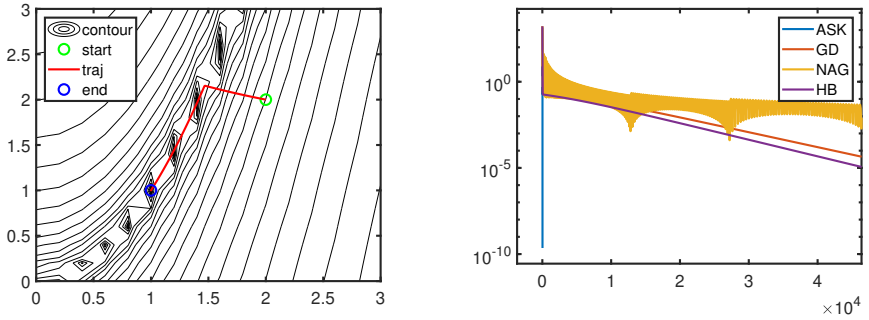


(a) Trajectory of ASK method (b) $\|\nabla f(\boldsymbol{x}^{(k)})\|$ v.s. k.

Figure 6: Algorithmic behaviors on the Dixon-Price function. The proposed algorithm converges to a critical point with a few iterations, while NAG seems diverges in the end.

Methods	$\|\nabla f\|$	Time
ASK	**3.0836e-07**	**2.6915e-02**
GD	5.4417e-05	7.4911e-01
NAG	5.9793e-02	1.0775e-01
HB	1.2757e-05	2.9323e-01

Table 7: Rosenbrock function, init=$4 \times \text{rand}(2,1) - 2$.



(a) Trajectory of ASK method (b) $\|\nabla f(\boldsymbol{x}^{(k)})\|$ v.s. k.

Figure 7: Algorithmic behaviors on the Rosenbrock function. All the gradient-based methods (GD, HB, and NAG) encounter difficulties or substantial slowdowns within the asymmetric valley that houses all local minima. In contrast, our proposed algorithm (ASK) manages to traverse efficiently within the valley, pinpointing a local minimum, attributed to its built-in adaptive mechanism.

faster convergence than other methods.

Methods	$\|\nabla f\|$	Time
ASK	**2.2497e-06**	**6.7327e-02**
GD	3.5192e-04	2.2789e+00
NAG	1.9299e-03	1.5885e-01
HB	1.3572e-04	6.1230e-01

Table 8: 10-d Dixon-Price Function, init=$20 \times \text{rand}(2,1) - 10$.

The second testing function is the least-squares minimization, which can be expressed as

$$f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T A\boldsymbol{x} - \boldsymbol{x}^T\boldsymbol{b}, \tag{4.8}$$

where A is a $d \times d$ matrix of full rank and $\boldsymbol{x}$, $\boldsymbol{b}$ are d-dimensional vector satisfying $A\boldsymbol{x} = \boldsymbol{b}$. We vary d by $200, 400, 600$ and record the corresponding results in Table 9. In all the three cases, our proposed method finds a solution with its gradient norm several orders of magnitude smaller than the other methods with comparable run time.

Methods	$\|\nabla f\|$	Time
dim = 200		
ASK	1.8745e-09	6.8912e+00
GD	4.2493e-03	4.0453e+01
NAG	1.4150e-05	9.6620e+00
HB	1.5566e-03	4.8435e+01
dim = 400		
ASK	5.6230e-11	9.6911e+01
GD	1.5588e-02	1.4123e+02
NAG	2.3108e-05	4.5453e+01
HB	8.7132e-06	1.3917e+02
dim = 600		
ASK	2.9439e-09	4.5376e+02
GD	1.3730e-03	2.9747e+02
NAG	1.1905e-06	1.7545e+02
HB	5.4395e-05	3.6776e+02

Table 9: Least-squares minimization under with dimensions of $200 \times 200, 400 \times 400, 600 \times 600$. For each case, ASK achieves the highest accuracy with comparable time.

4.2. Min-Max problems. In this section, we examine a series of numerical experiments designed to solve the min-max problem (2.6). Deliberately selecting four illustrative examples, we aim to conduct an in-depth exploration of the dynamics and efficacy of our proposed methodology:

(i) A challenging saddle point: we consider a system characterized by a particularly intricate saddle point, a scenario known to pose considerable challenges for traditional gradient-based algorithms.
(ii) Unique solution. While simpler in nature, this case underscores the effectiveness of our approach in handling well-defined optima.
(iii) Multiple solutions: we examine situations with multiple solutions that introduce complexity during the evolution of an algorithm. This case emphasizes the versatility of our proposed method in addressing diverse problem structures.
(iv) High-Dimensional case. Using a 100-dimensional testing function, we aim to demonstrate the scalability and adaptability of our approach.

Collectively, these examples comprehensively evaluate our proposed methodology, showcasing its performance and adaptability in addressing a spectrum of min-max optimization challenges.

We select a set of well-established gradient-based optimization techniques as benchmarks for comparison, including gradient descent ascent (GDA) (2.7), optimistic GDA with momentum (OGDA) (2.8), Nesterov's Accelerated Gradient (NAG) (2.5), and the heavy ball (HB) gradient method (2.4). OGDA, NAG, and HB introduce momentum terms in the optimization process to expedite the convergence of the gradient-based method GDA. Due to the complex nature of min-max problems, there are cases when an algorithm fails to converge. Consequently, we report the success rates (i.e., $\|f(\boldsymbol{x})\| < 10^{-6}$ along with the gradient norm and time consumption averaged over all successful trials.

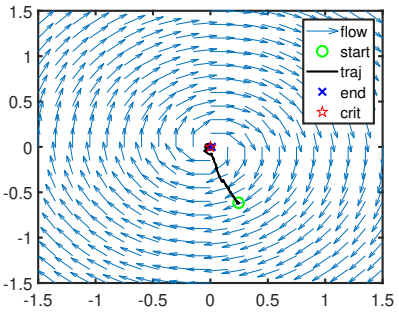
Case (i). We start with a simple function defined by

$$f(x_1, x_2) = x_1 x_2. \tag{4.9}$$

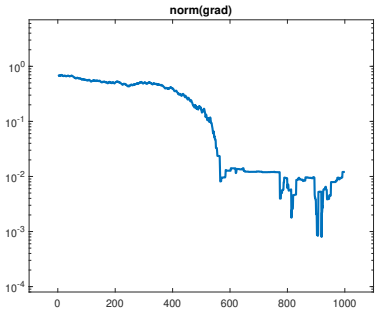
The Hessian matrix of this system is given by $H = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$. The determinant of the Hessian matrix is $\det(H) = -1$, which implies that $(0,0)$ is the unique saddle point to the problem:

$$\min_{x_1} \max_{x_2} x_1 x_2.$$

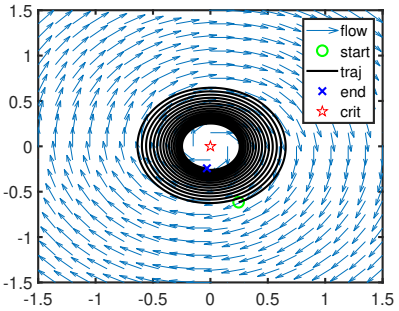
Unfortunately, the saddle point is not stable, as the eigenvalues of this matrix are imaginary. Starting from an initial point within the domain $[-1,1]\times[-1,1]$, we present the ASK trajectory in Figure 8(a) alongside the gradient norm plotted in Figure 8(b), illustrating plateaus and oscillations in the evolution process. As illustrated in Figure 8(c) and Figure 8(d), the trajectory of OGDA circles around $(0,0)$ and the reduction of $\|\nabla f\|$ slows down with each iteration. Table 10 reports the gradient norm of the solution, the run time, and the success rate of each method over 100 random initial points. It is evident that ASK is the only method capable of converging to the unique saddle point within a reasonable time frame, whereas both GDA and NAG diverge. The reason ASK succeeds in this example lies in the fact that the Koopman operator maps a finite-dimensional non-linear dynamical system to an infinitely dimensional linear system, which can be approximated and truncated by ASK. By carefully selecting the acceptable range radius r, our truncated approximation of the Koopman operator effectively tracks the descending factor in this problem. In contrast, traditional methods that strictly follow the exact gradient are unable to adapt to the problem's nuances, leading to their slower



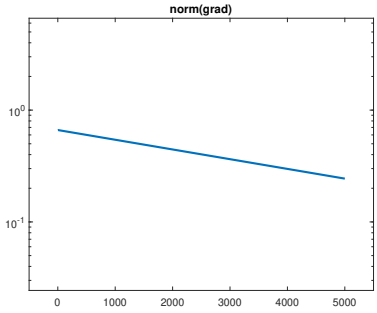
(a) ASK trajectory



(b) $\|\nabla f(\boldsymbol{x}^{(k)}))\|$ of ASK



(c) OGDA trajectory



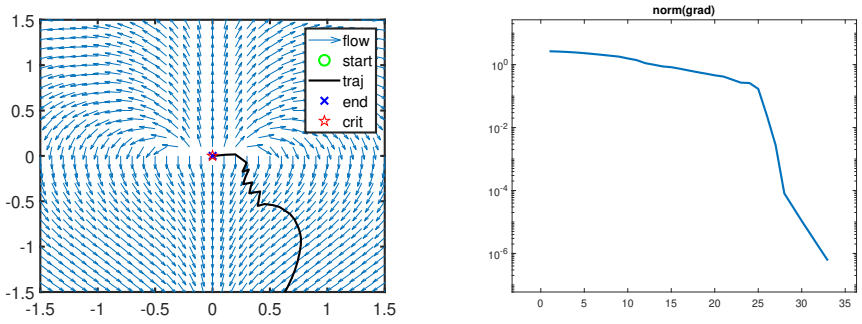
(d) $\|\nabla f(\boldsymbol{x}^{(k)}))\|$ of OGDA

Figure 8: The trajectory of ASK and OGDA evolution from a particular initial point for solving $\min_{x_1}\max_{x_2} f(x_1,x_2) = x_1x_2$. OGDA does not work well due to the lack of stability.

Methods	$\|\nabla f\|$	Time(Avg.)	Success rate
ASK	5.4423e-03	2.3591e+00	1.00
OGDA	6.5660e-01	6.9367e+01	0.00
GDA	1.3918e+00	9.8197e-03	0.00
NAG	5.2703e+01	6.2550e-03	0.00
HB	1.8026e+00	6.1683e-03	0.00

Table 10: $f(x_1,x_2) = x_1x_2$, init=$2\times\text{rand}(2,1)-1$. The proposed ASK is the only method that can converge to the unique saddle point with 100% success rates.

convergence or in some cases, failure to converge. In summary, ASK's success can be attributed to its adaptability to the underlying structure of the optimization landscape, enhancing its performance, especially in situations where traditional methods encounter difficulties in navigating complex, non-linear dynamics.



(a) ASK trajectory (b) $\|\nabla f(\boldsymbol{x}^{(k)}))\|$ of ASK

Figure 9: (a) A zig-zag trajectory of the ASK evolution for solving $\min_{x_1}\max_{x_2} f(x_1,x_2) = -x_1^2x_2+0.5x_2^2$ from a particular initial point. (b) The evolution of $\|\nabla f\left(\boldsymbol{x}^{(n)}\right)\|$ illustrates that the proposed method encounters challenges in the first 25 iterations. Here, $x_1 = 0$ is near the border of the neighborhood and the dynamic along that line is degenerated along the x-axis, which corresponds to a singular matrix. Consequently, the gradient drops rapidly when the trajectory gets close enough to this line.

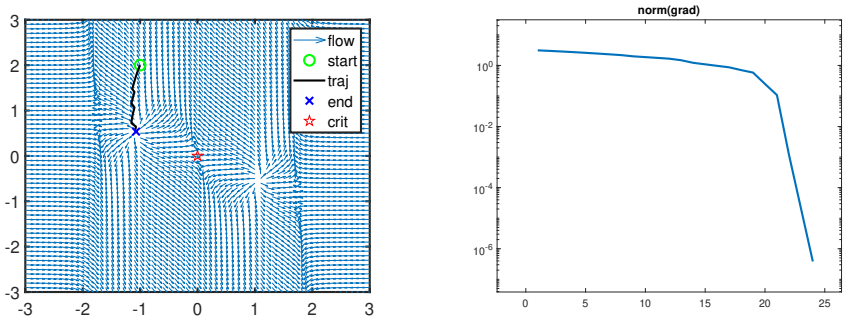
Table 11: $f(x_1,x_2) = -x_1^2x_2+0.5x_2^2$, init=$2\times\text{rand}(2,1)-1$

Methods	$\|\nabla f\|$	Time	Success rate
ASK	5.5890e-07	1.3220e-02	1.00
GDA	4.5703e-04	9.1963e-02	0.99
OGDA	7.6267e-05	2.0828e-01	1.00
NAG	1.3524e-3	5.8679e-02	0.51
HB	3.9246e-04	5.8969e-02	1.00

Case (ii). We consider the following function

$$f(x_1,x_2) = -x_1^2x_2^2+0.5x_2^2, \tag{4.10}$$

which has the unique saddle point of $f(\boldsymbol{x}^*) = 0$ at $\boldsymbol{x}^* = (0,0)$. We choose the initial points from $[-1,1]\times[-1,1]$. For a specific initial point, Figure 9(a) shows the trajectory of the ASK evolution, revealing a zig-zag behavior caused by local complexities where the system's dynamics shift. Consequently, these localized complexities necessitate multiple reevaluations of SVD to closely track an equilibrium. In Figure 9(b), the norm of its gradient at each iteration is plotted. It appears that the proposed method encounters challenges with in the first 25 iterations. Here, $x_1 = 0$ is near the border of the neighborhood and the dynamic along that line is degenerated along the x-axis, which corresponds to a singular matrix. Consequently, the gradient drops rapidly when the trajectory gets close enough to this line. Table 11 records the gradient norm of the final solution returned by each method, computation time, and suc-



(a) ASK trajectory (b) $\|\nabla f(\boldsymbol{x}^{(k)}))\|$ of ASK

Figure 10: (a) The dynamic of the 3-Hump Camel function and the trajectory of the ASK evolution from a specific initial point. (b) The plot of $\|\nabla f(\boldsymbol{x}^{(k)}))\|$ shows a gradual decrease for the first 20 iterations, followed by a rapid decay to 10^{-6}.

Table 12: 3-hump Camel function, init $= 6 \times \text{rand}(2,1) - 3$

Methods	$\|\nabla f\|$	Time	Sucess rate
ASK	3.8458e-07	7.8893e-03	0.82
GDA	9.9916e-07	4.7122e-02	0.84
OGDA	9.9755e-07	4.4569e-02	0.86
NAG	N/A	N/A	0.00
HB	9.9745e-07	3.3747e-02	0.80

cess rate over 100 random initial conditions. All the methods, except NAG, demonstrate a high success rate in finding the unique saddle point. ASK achieves the best results in terms of $\|\nabla f(\boldsymbol{x})\|$ and run time. In addition, OGDA achieves higher success rates than GDA and HB but at a cost of approximately twice the time consumption.

Case (iii). We investigate the three-hump Camel function, defined in (4.4). Figure 10 presents the ASK evolution and the gradient norm. Our algorithm exhibits a gradual decrease for the first 20 iterations, followed by a rapid decay to 10^{-6}. The comparisons with other methods are documented in Table 12, with each recorded value representing the average over 100 random initial points in the domain of $[-3,3] \times [-3,3]$. The proposed method achieves comparable success rates to GDA, OGDA, and HB but it is an order of magnitude faster.

Case (iv). The last testing function is the 100-d Dixon-Price function, defined in (4.6) with $d = 100$. The comparison results are presented in Table 13 averaged over 100 initial points chosen randomly from $[-10,10]^{100}$. The proposed algorithm gives the highest success rates while being several orders of magnitude faster than other methods.

5. Conclusions. In this paper, we have harnessed the power of the Adaptive Spectral Koopman (ASK) method [29] with sparse grids [30] that was originally designed for dynami-

Table 13: 100d Dixon-Price, init = $20 \times \text{rand}(100, 1) - 10$

Methods	$\|\nabla f\|$	Time	Sucess rate
ASK	6.5132e-07	7.8893e-03	0.85
GDA	N/A	N/A	0.00
OGDA	N/A	N/A	0.00
NAG	N/A	N/A	0.00
HB	5.9745e-02	4.1235e+01	0.20

cal systems to general optimization problems, including a min-max problem. Instead of closely tracking a single trajectory, we adapted the ASK method in such a way that it focuses on the dynamics of the system and identifies its equilibrium, which corresponds to a (local) critical point of the relevant optimization problem. We provided an error estimate of a special case, which connects the accuracy of the spectral methods based on sparse grid points and the proposed method. We conducted extensive experiments on various testing cases for a thorough assessment of ASK's capabilities. Our numerical findings consistently highlight the advantage of accuracy obtained by the ASK method, indicating its superior precision in finding critical points over some popular gradient-based methods. Remarkably, this elevated accuracy does not incur a substantial computational cost, as ASK's computational time remains comparable to, or marginally higher than, that of traditional optimization methods. The experimental results on min-max problems suggested an improved capability of ASK to identify critical points, particularly in scenarios where system's dynamics in the neighborhood at the current state effectively approximate the system behaviors globally. Collectively, this research demonstrates the efficacy and versatility of the ASK method as a powerful tool for a wide spectrum of optimization problems. Its remarkable accuracy, coupled with reasonable computational costs, positions it as a promising candidate for real-world applications.

REFERENCES

[1] H. Arbabi and I. Mezic, *Ergodic theory, dynamic mode decomposition, and computation of spectral properties of the koopman operator*, SIAM Journal on Applied Dynamical Systems, 16 (2017), pp. 2096–2126.

[2] M. Balabane, M. A. Mendez, and S. Najem, *Koopman operator for Burgers's equation*, Physical Review Fluids, 6 (2021), p. 064401.

[3] V. Barthelmann, E. Novak, and K. Ritter, *High dimensional polynomial interpolation on sparse grids*, Advances in Computational Mathematics, 12 (2000), pp. 273–288.

[4] A. Ben-Tal, L. El Ghaoui, and A. Nemirovski, *Robust optimization*, vol. 28, Princeton university press, 2009.

[5] I. O. Bohachevsky, M. E. Johnson, and M. L. Stein, *Generalized simulated annealing for function optimization*, Technometrics, 28 (1986), pp. 209–217.

[6] L. Bottou, *Large-scale machine learning with stochastic gradient descent*, in Proceedings of COMPSTAT'2010: 19th International Conference on Computational StatisticsParis France, August 22-27, 2010 Keynote, Invited and Contributed Papers, Springer, 2010, pp. 177–186.

[7] ———, *Stochastic gradient descent tricks*, Neural Networks: Tricks of the Trade: Second Edition, (2012), pp. 421–436.

[8] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex optimization*, Cambridge university press, 2004.
[9] D. Breda, S. Maset, and R. Vermiglio, *Pseudospectral approximation of eigenvalues of derivative operators with non-local boundary conditions*, Applied numerical mathematics, 56 (2006), pp. 318–331.
[10] ———, *Approximation of eigenvalues of evolution operators for linear retarded functional differential equations*, SIAM Journal on Numerical Analysis, 50 (2012), pp. 1456–1483.
[11] S. L. Brunton, M. Budišić, E. Kaiser, and J. N. Kutz, *Modern Koopman theory for dynamical systems*, SIAM Review, 64 (2022), pp. 229–340.
[12] S. Das and D. Giannakis, *Koopman spectra in reproducing kernel Hilbert spaces*, Applied and Computational Harmonic Analysis, 49 (2020), pp. 573–607.
[13] C. Daskalakis, A. Ilyas, V. Syrgkanis, and H. Zeng, *Training gans with optimism*, arXiv preprint arXiv:1711.00141, (2017).
[14] F. Dietrich, T. N. Thiem, and I. G. Kevrekidis, *On the Koopman operator of algorithms*, SIAM Journal on Applied Dynamical Systems, 19 (2020), pp. 860–885.
[15] J. Duchi, E. Hazan, and Y. Singer, *Adaptive subgradient methods for online learning and stochastic optimization*, Journal of machine learning research, 12 (2011).
[16] G. Froyland, O. Junge, and P. Koltai, *Estimating long-term behavior of flows without trajectory integration: The infinitesimal generator approach*, SIAM Journal on Numerical Analysis, 51 (2013), pp. 223–247.
[17] D. Giannakis, *Data-driven spectral decomposition and forecasting of ergodic dynamical systems*, Applied and Computational Harmonic Analysis, 47 (2019), pp. 338–396.
[18] P. Giselsson and S. Boyd, *Monotonicity and restart in fast gradient methods*, in IEEE Conference on Decision and Control, 2014, pp. 5058–5063.
[19] M. Grant and S. Boyd, *Graph implementations for nonsmooth convex programs*, in Recent Advances in Learning and Control, V. Blondel, S. Boyd, and H. Kimura, eds., Lecture Notes in Control and Information Sciences, Springer-Verlag Limited, 2008, pp. 95–110. http://stanford.edu/~boyd/graph_dcp.html.
[20] ———, *CVX: Matlab software for disciplined convex programming, version 2.1*. http://cvxr.com/cvx, Mar. 2014.
[21] A.-R. Hedar, *Global optimization test problems*, 2007.
[22] M. Hu, Y. Lou, B. Wang, M. Yan, X. Yang, and Q. Ye, *Accelerated sparse recovery via gradient descent with nonlinear conjugate gradient momentum*, Journal of Scientific Computing, 95 (2023), p. 33.
[23] M. Jamil and X.-S. Yang, *A literature survey of benchmark functions for global optimisation problems*, International Journal of Mathematical Modelling and Numerical Optimisation, 4 (2013), pp. 150–194.
[24] D. Kinga, J. B. Adam, et al., *A method for stochastic optimization*, in International conference on learning representations (ICLR), vol. 5, 2015, p. 6.
[25] B. O. Koopman, *Hamiltonian systems and transformation in Hilbert space*, Proceedings of the National Academy of Sciences of the United States of America, 17 (1931), p. 315.
[26] B. O. Koopman and J. v. Neumann, *Dynamical systems of continuous spectra*, Proceedings of the National Academy of Sciences, 18 (1932), pp. 255–263.
[27] M. Korda, M. Putinar, and I. Mezić, *Data-driven spectral analysis of the Koopman operator*, Applied and Computational Harmonic Analysis, 48 (2020), pp. 599–629.
[28] R. Laraki and J. B. Lasserre, *Semidefinite programming for min-max problems and games*, Mathematical programming, 131 (2012), pp. 305–332.
[29] B. Li, Y. Ma, J. N. Kutz, and X. Yang, *The adaptive spectral Koopman method for dynamical systems*, SIAM Journal on Applied Dynamical Systems, 22 (2023), pp. 1523–1551.
[30] B. Li, Y. Yu, and X. Yang, *The sparse-grid-based adaptive spectral koopman method*, arXiv preprint arXiv:2206.09955, (2022).
[31] I. Mezić, *Spectral properties of dynamical systems, model reduction and decompositions*, Nonlinear Dynamics, 41 (2005), pp. 309–325.
[32] M. Molga and C. Smutnicki, *Test functions for optimization needs*. https://robertmarks.org/Classes/ENGR5358/Papers/functions.pdf, 2005. Accessed on 12/18/2023.
[33] H. Nakao and I. Mezić, *Spectral analysis of the Koopman operator for partial differential equations*,

Chaos: An Interdisciplinary Journal of Nonlinear Science, 30 (2020), p. 113131.
[34] J. Nathan Kutz, J. L. Proctor, and S. L. Brunton, *Applied Koopman theory for partial differential equations and data-driven modeling of spatio-temporal systems*, Complexity, 2018 (2018).
[35] A. S. Nemirovski and Y. E. Nesterov, *Optimal methods of smooth convex minimization*, Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki, 25 (1985), pp. 356–369.
[36] Y. Nesterov, *A method of solving a convex programming problem with convergence rate o (1/k2)*, in Soviet Mathematics Doklady, vol. 27, 1983, pp. 372–376.
[37] ———, *Introductory lectures on convex optimization: A basic course*, vol. 87, Springer Science & Business Media, 2003.
[38] B. Polyak, *Some methods of speeding up the convergence of iteration methods*, USSR Computational Mathematics and Mathematical Physics, 4 (1964), pp. 1–17.
[39] A. Raghunathan, J. Steinhardt, and P. S. Liang, *Semidefinite relaxations for certifying robustness to adversarial examples*, Advances in neural information processing systems, 31 (2018).
[40] H. Rosenbrock, *An automatic method for finding the greatest or least value of a function*, The computer journal, 3 (1960), pp. 175–184.
[41] S. Ruder, *An overview of gradient descent optimization algorithms*, arXiv preprint arXiv:1609.04747, (2016).
[42] F. Santambrogio, *Euclidean, metric, and Wasserstein gradient flows: an overview*, Bulletin of Mathematical Sciences, 7 (2017), pp. 87–154.
[43] J. W. Siegel, *Accelerated first-order methods: Differential equations and Lyapunov functions*, arXiv preprint arXiv:1903.05671, (2019).
[44] W. Su, S. Boyd, and E. Candes, *A differential equation for modeling Nesterov's accelerated gradient method: Theory and insights*, Advances in neural information processing systems, 27 (2014), pp. 2510–2518.
[45] G. Teschl, *Ordinary differential equations and dynamical systems*, vol. 140, American Mathematical Soc., 2012.
[46] J. v. Neumann, *Zur theorie der gesellschaftsspiele*, Mathematische annalen, 100 (1928), pp. 295–320.
[47] Y. Wang and W. Li, *Accelerated information gradient flow*, Journal of Scientific Computing, 90 (2022), pp. 1–47.
[48] D. S. Watkins, *Fundamentals of matrix computations*, John Wiley & Sons, 2004.
[49] A. Wibisono, A. C. Wilson, and M. I. Jordan, *A variational perspective on accelerated methods in optimization*, proceedings of the National Academy of Sciences, 113 (2016), pp. E7351–E7358.
[50] M. O. Williams, M. S. Hemati, S. T. Dawson, I. G. Kevrekidis, and C. W. Rowley, *Extending data-driven Koopman analysis to actuated systems*, IFAC-PapersOnLine, 49 (2016), pp. 704–709.
[51] M. O. Williams, I. G. Kevrekidis, and C. W. Rowley, *A data-driven approximation of the Koopman operator: Extending dynamic mode decomposition*, Journal of Nonlinear Science, 25 (2015), pp. 1307–1346.
[52] A. C. Wilson, B. Recht, and M. I. Jordan, *A Lyapunov analysis of accelerated methods in optimization*, Journal of Machine Learning Research, 22 (2021), pp. 1–34.