# Learning-Based Optimization: Stability and Convergence in Explicit ODE-Driven Algorithms

Chris Junchi Li[◇]

Department of Electrical Engineering and Computer Sciences[◇]
University of California, Berkeley

**Abstract**

We propose a framework that translates ordinary differential equation (ODE) properties into discrete-time iterative optimization algorithms using explicit Euler discretization. Building on the recent advances in learning to optimize (L2O), we develop a methodology that automatically tunes the damping coefficients of ODEs to guarantee stability and convergence of the corresponding discretized methods. Our framework bridges the gap between continuous-time optimization inspired by ODEs and their discrete-time counterparts. Theoretical guarantees for convergence and stability are provided under explicit conditions, and extensive numerical experiments demonstrate the efficacy of the learned optimization algorithms.

**Keywords:** ODE, Optimization, Euler Discretization, Learning-to-Optimize, Stability, Convergence.

## 1 Introduction

Recent years have witnessed substantial efforts to understand the connection between continuous-time optimization techniques and their discrete-time analogs, especially through the lens of ordinary differential equations (ODEs). Several seminal works, such as Su, Boyd, and Candes [55], have drawn direct parallels between optimization algorithms like Nesterov's Accelerated Gradient (NAG) method and second-order ODEs. This framework has provided rich theoretical insights, but practical gaps remain when translating continuous-time convergence rates to discrete-time iterative methods. Specifically, while the continuous trajectory converges at a fast rate, the discrete versions often lose these benefits due to discretization errors.

This paper focuses on bridging this gap by developing a systematic approach to translate ODE-based optimization methods into practical, stable, and convergent discrete algorithms. Our approach utilizes the explicit Euler discretization scheme to construct a general framework for converting continuous-time models into discrete-time algorithms. We emphasize using explicit discretization methods to avoid the complexities and case-specific designs associated with implicit schemes. Furthermore, we introduce a learning-based optimization framework (L2O) to automatically tune the ODE coefficients, ensuring convergence and stability in the resulting discrete algorithms.

Our study offers both theoretical and practical contributions. Theoretically, we provide conditions for the convergence of the Euler-discretized sequences. Practically, we present an L2O-based method that tunes the ODE parameters for specific optimization tasks, leading to a family of efficient iterative optimization methods. Numerical experiments demonstrate the robustness of the learned algorithms across various benchmarks.

**Backgrounds** In recent years, there has been substantial work aimed at understanding the nature of acceleration methods using ordinary differential equations (ODEs). The seminal work of Su,

Boyd, and Candes [55] proposes a second-order differential equation as the continuous-time counterpart of the Nesterov accelerated gradient method (NAG) [49]. This work provides theoretical insights into the nature of NAG. By combining the continuous-time model of NAG with Hessian-driven damping, a differential equation called the inertial system with Hessian-driven damping (ISHD) has been investigated in [5, 7]:

$$\ddot{x}(t) + \frac{\alpha}{t}\dot{x}(t) + \beta(t)\nabla^2 f(x(t))\dot{x}(t) + \gamma(t)\nabla f(x(t)) = 0 \tag{1}$$

where $x(t)$ belongs to $\mathbb{R}^n$ and $f$ is twice differentiable and convex. Here, $t_0, \alpha > 0$, and $\beta, \gamma$ are non-negative continuous functions defined on $[t_0, +\infty)$. By choosing different values for $\alpha, \beta$, and $\gamma$, the equation (1) can coincide with many other optimization-inspired ODEs, providing a unified framework for understanding various optimization methods.

Although the ODE viewpoint has been successful in understanding optimization methods, significant practical gaps exist between iterative optimization methods and their continuous-time counterparts. The fast convergence rate of $x(t)$ in continuous-time cases may not be translated directly to the sequence $\{x_k\}_{k=0}^{\infty}$ obtained via discretizing the corresponding ODE. Designing iterative optimization methods using the ODE viewpoint often requires a case-by-case discussion and equation-specific discretization schemes.

For example, in [59], it focuses on a subfamily of (1) with $\alpha = p + 1, \beta(t) \equiv 0$, and $\gamma(t) = Cp^2t^{p-2}$, where $p, C > 0$. According to [59], the convergence rate of the trajectory $x(t)$ is given by $f(x(t)) - f_\star \leq \mathcal{O}(1/t^p)$. However, the arbitrary nature of the parameter $p$ suggests that the convergence rate of the trajectory cannot be bounded by *any inverse polynomial function* for any convex differentiable functions. This finding starkly contrasts with the well-known lower bound of $\mathcal{O}(1/k^2)$ for first-order methods when the function $f$ is $L$-smooth convex, as discussed in [48]. The underlying issue is attributed to the notation $x(t)$, which glosses over the challenges posed by differential equations with varying curvatures. In numerical computations, achieving a solution of comparable accuracy significantly depends on the curvature of the objective functions. It is misleading to directly compare the convergence rates of continuous-time models with their discrete-time counterparts. The empirical results show that a naive explicit Euler discretization scheme diverges even for a quadratic objective function. A stable discretization that recovers the $\mathcal{O}(1/k^p)$ rate requires an implicit update using extra higher-order gradient. Hence, a core problem in this field remains: (**P1**) ***"Can we translate the fast convergence properties of the trajectory of ODEs into the discretized sequences using explicit discretization schemes?"***

We provide a positive answer to this problem by giving conditions for discretizing the ODE (1) stably using the explicit Euler scheme. Our emphasis on explicit schemes, such as the explicit Euler discretization, stems from the desire to develop a general framework for translating continuous-time ODE properties into discrete-time algorithms. By doing so, we can avoid the need for ad hoc, equation-specific designs and case-by-case analyses, which are often required for implicit schemes. This exploration of explicit Euler discretization serves as a foundational step towards adapting more sophisticated explicit schemes, such as Runge-Kutta methods, to preserve the properties of continuous trajectories in discrete iterative algorithms.

Given the flexibility of the coefficients $\alpha, \beta(t)$, and $\gamma(t)$ in the ODE (1), the choice of these coefficients can significantly impact the convergence rate and stability of the ODE solution, which in turn affects the performance of the discretized iterative algorithm. While the ODE (1) provides a unifying framework, finding the optimal coefficients for a given problem is a non-trivial task. This leads to a crucial question: (**P2**) ***"What are the optimal coefficients of the ODE*** (1) ***for specific problems?"***

This issue, referred to as the "best tuning of the coefficients", has also been highlighted by [3]. We demonstrate that the optimal coefficients can be determined numerically using learning to optimize.

## 1.1 Related works

### 1.1.1 ODE viewpoint of optimizetion methods

The ODE viewpoint of optimization methods have a long history. The work by Su, Boyd, and Candes [55] has revived interest in using ODEs to understand acceleration methods. The concept of gradient correction, which uncovers the mechanism behind acceleration, is found within the high-resolution differential framework [53]. This framework notably explains the linear convergence of FISTA in composite optimization problems when the smooth function are strongly convex [38]. These advancements pave the way for further research and development in optimization algorithms. It is crucial to recognize that, despite their similar forms, the high-resolution differential equation and ISHD have different motivations. The high-resolution differential equation is designed for NAG, whereas the second-order information in ISHD is derived from Newton's method [6], which is used to analyze the forward-backward algorithm.

Besides the high-resolution framework and the ISHD, the acceleration methods have also been explained through the lens of numerical stability [43, 67]. Additionally, some studies consider closed-loop ODEs as analogous to adaptive optimization methods. For instance, a closed-loop dynamical system has been proposed to analyze high-order tensor algorithms for unconstrained smooth optimization from a control-theoretic perspective [40]. Another study explores the development of fast optimization methods through inertial continuous dynamics with nonlinear damping [4].

There are several works focused on designing iterative optimization methods by discretizing ODEs. In [65], the authors demonstrate that to ensure the $l$-th Runge-Kutta integrator applied to a subfamily of (1) with $\alpha = 2p+1$, $\beta(t) \equiv 0$, and $\gamma(t) = Cp^2t^{p-2}$, where $p, C > 0$, is convergent, the objective function must be sufficiently flat and the step size must be diminishing. Other methods that develop iterative optimization techniques by discretizing corresponding ODEs typically only consider strongly convex cases [56, 57, 66], which are not general enough and may difficult to generalize to other scenarios.

Some research, such as [46], considers the optimal selection of ODEs through the performance estimation problem (PEP). However, this approach does not focus on designing practical iterative optimization methods. Another straightforward approach is to tune the coefficients analytically. By exploiting the geometric properties of the function $f$ in (1), several studies focus on tuning coefficients analytically with provable fast convergence rate [9, 10, 11, 12]. Two main properties used in these works are the growth condition and flatness condition. When these conditions are absent, we show that the coefficients can be tuned numerically using machine learning techniques.

### 1.1.2 Learning to optimize

Learning to optimize (L2O) is a paradigm that leverages machine learning to automate the design of optimization algorithms. This approach aims to uncover the underlying structure of a collection of optimization problems through machine learning [19]. L2O can be broadly categorized into two classes: model-based methods and model-free methods. Model-based methods, which are derived from iterative optimization methods, involve identifying learnable parameters. Two prominent classes of model-based methods, algorithm unrolling [45] and plug-and-play [1, 36], have been successfully applied to various tasks in signal processing and image processing.

Seminal work of model-based methods has demonstrated that the iterative shrinkage-thresholding algorithm (ISTA) can be viewed as a recurrent neural network (RNN), and by introducing learnable parameters, significant improvements in sparse coding can be achieved [34]. This concept has been extended by unrolling the alternating direction method of multipliers (ADMM) algorithm to construct a neural network with prior knowledge of compressive sensing problems [62]. Further, by designing an efficient algorithm to compute the proximal operator induced by sparsity regularity, ISTA has been reformulated as a deep neural network in [63], which demonstrates substantial improvements in compressed sensing performance.

In contrast, model-free methods do not rely on existing models. By treating designing an optimization algorithm as a policy search within a reinforcement learning framework, the improved optimization algorithms has been discovered [39]. The generalization of the L2O paradigm within a derivative-free black-box optimization context was realized through the use of RNNs [22]. Further advancements were made by [2], who employed long short-term memory networks (LSTMs) to implement learned optimization methods, achieving superior performance compared to generic algorithms. In [20], the design of optimization algorithms is approached as a program search problem. It utilizes evolutionary search to develop the LION algorithm, which exhibits performance comparable to Adam. More recently, a transformer-based neural network has been proposed to represent the update step, incorporating a preconditioning matrix to enhance efficiency, as reported in [32].

In spite of the significant performance improvement, only a few works establish a non-trivial convergence guarantee for learned optimization methods. For instance, the learned ISTA has been thoroughly analyzed, with [21] showing that there exist weights that achieve a theoretical linear convergence rate on the LASSO problem, surpassing ISTA theoretically. An explicit formula to calculate the optimal learnable weights is further provided [41], effectively removing most of the computational overhead of training, and numerical experiments verify the theoretical results. The necessary condition for a learned optimization method to converge is investigated in [42]. The result partially addresses the primary difficulty in designing learned optimization algorithms with convergence guarantee. Other methods exist that search for parameters while ensuring convergence. For example, a general formulation with learnable parameters for solving nonsmooth composite optimization problems is presented [14]. The conditions for establishing the convergence property of the learned methods are then derived, and unsupervised learning is employed to find more efficient methods under these conditions. The effectiveness of this methodology in ensuring the convergence of learned optimizers on smooth unconstrained optimization and composite optimization problems is also demonstrated [15].

### 1.2 Our contributions

The framework of this paper is presented in Figure 1.

We address the first problem (**P1**) by providing conditions that ensure the convergence of the sequence $\{x_k\}_{k=0}^{\infty}$, which is generated by applying the explicit Euler discretization to equation (1). These conditions comprises a convergence condition and a stability condition. The convergence condition ensures that the continuous-time trajectory $x(t)$ converges. The stability condition ensures that when applying the explicit Euler scheme to the ODE (1), which satisfies the convergence condition, the discretization remains stable. The requirements of these conditions on the coefficients are relaxed, allowing us to generate numerous iterative optimization methods.

For the second problem (**P2**), we utilize a L2O framework and develop a corresponding algorithm to solve the L2O problem numerically. L2O is a paradigm that automatically designs

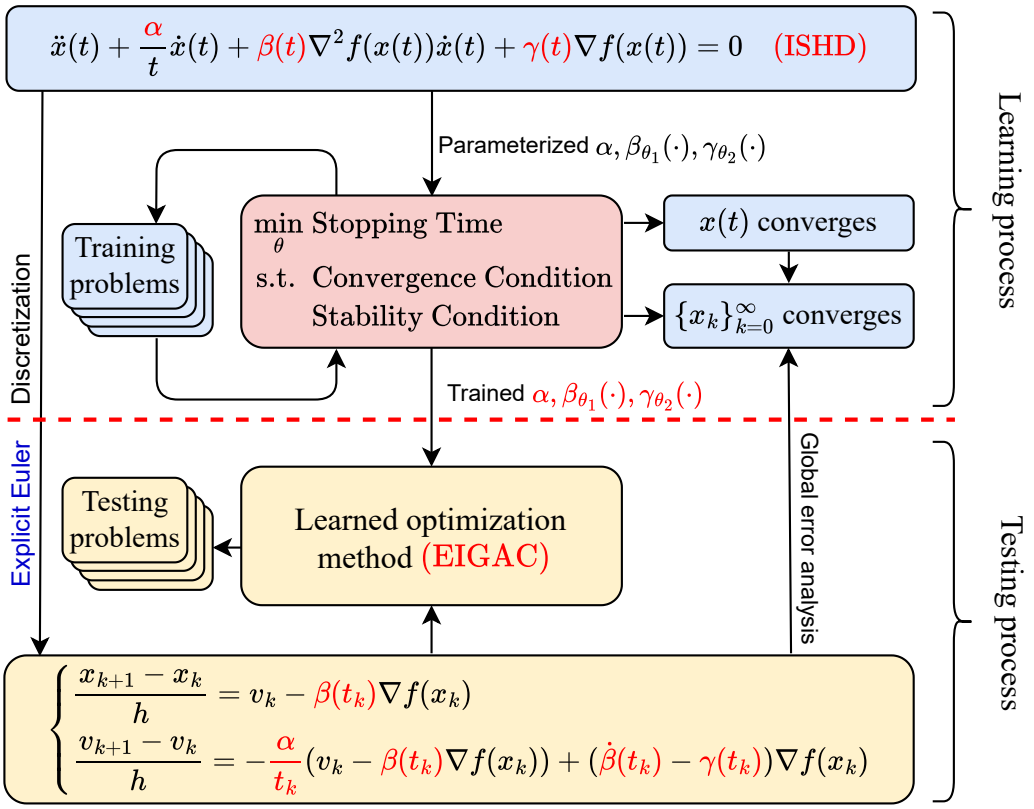



**Figure 1:** Our learning and testing framework.

optimization methods using machine learning techniques. The core idea is to leverage a training dataset of diverse optimization problems to tune the parameters of optimization methods, in order to improve their performance while ensuring convergence. L2O uses a metric of the efficiency of the optimization methods as the training loss. In our framework, we take the "stopping time" as the training loss, which measures the number of iterations that the the algorithms generated by discretizing the ODE (1) need to achieve a predefined suboptimality. This measure generalizes complexity from discrete-time to continuous-time cases. We then define the probability distribution of a parameterized function family by establishing equivalence with corresponding parameters. The training formulation of our L2O framework is posed as a problem that minimizes the expected stopping time under the expectation constraints that ensure the convergence and stability conditions hold for each function. Due to the existence of the expectation constraints, we combine the stochastic optimization methods with the penalty function method to solve the training formulation. Additionally, we derive conservative gradients of the stopping time and constraint functions, making our algorithm more robust and general. We also provide convergence guarantees for the training algorithm under the sufficient decrease assumption, using only the conservative gradients. Finally, we get the learned optimization methods by applying the explicit Euler scheme to the ODE (1) with learned coefficients.

Our contributions can be divided into two major parts. The first part involves an ODE methodology that derives the conditions necessary to ensure the convergence of the sequence obtained through discretization. The second part utilizes the L2O approach to find the optimal coefficients under these conditions. Although the convergence rate of the sequence is $\mathcal{O}(1/k)$ under these

conditions, we argue that these conditions allow for more flexible coefficient selection and facilitate the design of various practical iterative optimization methods. We also provide default values that satisfy these conditions. Numerical experiments demonstrate that the sequence with these default values exhibits comparable performance to other optimization methods with a theoretical $\mathcal{O}(1/k^2)$ convergence rate. In contrast to most L2O methods, our framework has a solid theoretical foundation, which is a crucial step towards designing practical optimization methods. Compared to other methods that ensure the convergence of learned optimization methods [14, 15], our result also includes an explicit convergence rate, whereas their results do not. Furthermore, the theoretical linear rate in [21] relies on the specific form of LASSO, whereas our result can be applied to various unconstrained convex smooth minimization problems. We summarize the key contributions of this paper as follows:

- We establish conditions under which the sequence $\{x_k\}_{k=0}^{\infty}$, obtained by applying the explicit Euler discretization to equation (1), is guaranteed to converge. This analysis is grounded in global error analysis from numerical solutions of differential equations.

- We introduce a metric named "stopping time", which serves as the continuous-time counterpart of complexity. We demonstrate that "stopping time" is differentiable under mild conditions and can be numerically optimized to improve algorithm performance.

- We propose a general learning-to-optimize (L2O) framework that automatically tunes the ODE coefficients, addressing the challenge of selecting optimal damping coefficients. The framework includes an algorithm with theoretical convergence guarantees, ensuring practical efficiency and stability in discrete optimization algorithms.

We conduct extensive numerical experiments to evaluate our algorithms. The optimization problems include the logistic regression and $\ell_p^p$ minimization on real world datasets, e.g. `a5a`, `mushrooms`, `w3a`, `covtype`, and `phishing`. The training process demonstrates that our algorithm for the training formulation converges to a feasible stationary point of the training formulation, verifying the effectiveness of our theory. The numerical experiments on the testing process highlight the advantages of the learned algorithms.

### 1.3 Organization of the Paper

The rest of the paper is organized as follows. In Section 2, we analyze the conditions for stabilizing the explicit Euler discretization of the inertial system with Hessian-driven damping (ISHD). Section 3 presents our methodology for deriving efficient optimization methods that guarantee worst-case convergence, leading to a stochastic optimization problem with expectation constraints. In Section 4, we derive expressions for the conservative gradients of the stopping time and constraint functions. Section 5 provides theoretical analyses of the proposed model. The effectiveness of our training algorithm is demonstrated through numerical experiments in Section 6, along with further experiments to showcase the efficiency and theoretical consistency of the learned optimization algorithms. Finally, we conclude and discuss future directions in Section 7.

## 2 Conditions that ensure a stable discretization of ISHD

### 2.1 Preliminaries

To simplify the presentation, we bundle the coefficients $\alpha$, $\beta$, and $\gamma$ into a single collection denoted as $\Xi$. Considering the coefficients $\Xi$, the function $f$, and the given time $t$, we define the trajectory associated with the system (1) as follows:

$$\begin{aligned} X\colon \mathbb{R}_+ \times \mathcal{C}(\mathbb{R},\mathbb{R}_+) \times \mathcal{C}(\mathbb{R},\mathbb{R}_+) \times \mathbb{R} \times \mathcal{C}(\mathbb{R}^n,\mathbb{R}) &\to \mathbb{R}^n, \\ (\Xi, t, f) &\to X(\Xi, t, f) \end{aligned} \tag{2}$$

We set the initial conditions for system (1) as follows: the initial time $t_0 > 0$, initial position $x(t_0)$, and initial velocity $\dot{x}(t_0)$ are all fixed.

For our notation, $\mathbb{S}^n$ and $\mathbb{S}^n_+$ represent the sets of all $n \times n$ symmetric matrices and symmetric positive semidefinite matrices, respectively. We denote the convex hull of a set $S$ by $\mathrm{co}(S)$. The notation $[\cdot]_+$ represents the maximum of a value and zero, defined as $[\cdot]_+ := \max\{\cdot, 0\}$. The set $\{0, 1, \ldots, M\}$, where $M$ is a non-negative integer, is denoted by $[M]$.

### 2.2 A condition that ensures the trajectory of ISHD converges

Throughout this paper, we make the following basic assumptions about (1).

**Assumption 1.** *$\alpha > 1, t_0 > 0, \varepsilon > 0$ are real numbers, $\beta$ and $\gamma$ are nonnegative continuously differentiable functions defined on $[t_0, +\infty)$. The function $f$ is twice differentiable convex with its domain $\mathrm{dom}\, f = \mathbb{R}^n$. The set of minimizers of $\min_x f(x)$ is not empty and the corresponding optimal value is $f_\star$.*

Assumption 1 also appears in the literature [3, 53] and is not restrictive. Given $\kappa \in (0, 1], \lambda \in (0, \alpha - 1]$, we define

$$\begin{aligned} \delta(t) &= t^2(\gamma(t) - \kappa\dot{\beta}(t) - \kappa\beta(t)/t) + (\kappa(\alpha - 1 - \lambda) - \lambda(1 - \kappa))t\beta(t), \\ w(t) &= \gamma(t) - \dot{\beta}(t) - \beta(t)/t \end{aligned} \tag{3}$$

Then, we provide a condition that ensures the convergence of the solution trajectory of (1).

**Theorem 1.** *Suppose that Assumption 1 and the following conditions hold true:*

$$\delta(t) > 0, \quad \textit{and} \quad \dot{\delta}(t) \le \lambda t w(t) \tag{4}$$

*Then, the solution trajectory of (1), $x(t)$, is bounded and the following inequalities can be derived:*

$$f(x(t)) - f_\star \le \mathcal{O}\left(\frac{1}{\delta(t)}\right),\ \|\nabla f(x(t))\| \le \mathcal{O}\left(\frac{1}{t\beta(t)}\right),\ \|\dot{x}(t)\| \le \mathcal{O}\left(\frac{1}{t}\right), \tag{5}$$

$$\int_{t_0}^{\infty} (\lambda t w(t) - \dot{\delta}(t))(f(x(t)) - f_\star)\,\mathrm{d}t < \infty, \tag{6}$$

$$\int_{t_0}^{\infty} t(\alpha - 1 - \lambda)\|\dot{x}(t)\|^2\,\mathrm{d}t < \infty, \tag{7}$$

$$\int_{t_0}^{\infty} t^2\beta(t)w(t)\|\nabla f(x)\|^2\,\mathrm{d}t < \infty, \tag{8}$$

$$\int_{t_0}^{\infty} t^2\beta(t)\langle\nabla^2 f(x(t))\dot{x}(t), \dot{x}(t)\rangle\,\mathrm{d}t < \infty. \tag{9}$$

The proof of Theorem 1 is presented in sec. 5. This theorem extends [5, Theorem 1] by providing estimations (6)-(9) for $f(x(t)) - f_\star$, $\|\dot{x}(t)\|$, $\|\nabla f(x(t))\|$, and $\|\dot{x}(t)\|^2_{\nabla^2 f(x(t))}$. They guarantee that these quantities are integrable when coupling with certain coefficients. The results on (5) imply that the convergence rate of $f(x(t)) - f_\star$, $\nabla f(x(t))$ can be controlled by the coefficients $\alpha, \beta(\cdot)$, and $\gamma(\cdot)$. Using the integrability with the convergence rate in (5), we can analyze the long-time behavior of the solution trajectory $x(t)$. This provides the guidance for deriving the conditions that guarantee the convergence of the sequence discretized form the equation (1).

### 2.3 A condition that ensures the stability of the explicit Euler discretization

Let $v(t_0) = x(t_0) + \beta(t_0)\nabla f(x(t_0))$ and

$$\psi_\Xi(x(t), v(t), t) = \begin{pmatrix} v(t) - \beta(t)\nabla f(x(t)) \\ -\dfrac{\alpha}{t}(v(t) - \beta(t)\nabla f(x(t))) + (\dot{\beta}(t) - \gamma(t))\nabla f(x(t)) \end{pmatrix} \tag{10}$$

The equation (1) can be reformulated as the first-order system

$$\begin{pmatrix} \dot{x}(t) \\ \dot{v}(t) \end{pmatrix} = \psi_\Xi(x(t), v(t), t) \tag{11}$$

We denote the flow associated with this system as $s(t, s_0, \Xi, f)$ with $s_0 = (x_0, v_0)$. Let $h$ be the step size, $t_k = t_0 + kh, k \geq 0$. The explicit Euler scheme of (1) writes

$$\begin{cases} \dfrac{x_{k+1} - x_k}{h} = v_k - \beta(t_k)\nabla f(x_k), \\ \dfrac{v_{k+1} - v_k}{h} = -\dfrac{\alpha}{t_k}\left(v_k - \beta(t_k)\nabla f(x_k)\right) + (\dot{\beta}(t_k) - \gamma(t_k))\nabla f(x_k) \end{cases} \tag{12}$$

The sequence $\{x_k\}_{k=0}^\infty$ denotes the position, while $\{v_k\}_{k=0}^\infty$ corresponds to the auxiliary variable sequence. Eliminating the auxiliary variable sequence $\{v_k\}_{k=0}^\infty$ in the equation (12) gives

$$\begin{aligned} x_{k+2} =& x_{k+1} - h^2\left(\gamma(t_k) - \dot{\beta}(t_k) + \frac{\beta(t_{k+1}) - \beta(t_k)}{h}\right)\nabla f(x_k) \\ &+ \left(1 - \frac{\alpha h}{t_k}\right)\underbrace{(x_{k+1} - x_k)}_{\text{inertia}} - h\beta(t_{k+1})\underbrace{(\nabla f(x_{k+1}) - \nabla f(x_k))}_{\text{gradient correction}} \end{aligned} \tag{13}$$

For clarity in our discussion, we reformulate the equation (12) in Algorithm 1. Since this algorithm is derived through the explicit Euler discretization of the ISHD, it includes an inertial term from the temporal discretization of the damping $\dot{x}(t)$ and a gradient correction term from the temporal discretization of the Hessian-driven damping $\nabla^2 f(x(t))\dot{x}(t)$. We name it the Explicit Inertial Gradient Algorithm with Correction (EIGAC). This algorithm is intended for the unconstrained smooth convex minimization problems:

$$\min_{x \in \mathbb{R}^n} \quad f(x) \tag{14}$$

where $f$ satisfies Assumption 1. Importantly, the temporal discretization of the Hessian-driven damping $\nabla^2 f(x(t))\dot{x}(t)$ only contains first-order information, which makes EIGAC a first-order algorithm. Through the selection of various coefficients, Algorithm 1 offers a diverse range of optimization methods.

Building upon the foundation established by Theorem 1, we introduce a criterion termed the *stability condition*. This condition ensures the convergence of the Algorithm 1.

**Algorithm 1** Explicit Inertial Gradient Algorithm with Correction (EIGAC)

1: **Input:** the function $f$, the initial time $t_0$, the initial values $x_0$ and $v_0$, the step size $h$, the coefficients $\Xi = \{\alpha, \beta(\cdot), \gamma(\cdot)\}$.
2: **for** $k = 1, 2, \ldots$ **do**
3: Update the time: $t_k = t_0 + kh$.
4: Update $x_k$ by (12).
5: Update $v_k$ by (12).
6: **end for**
7: **Output:** the solution $x_\star$ reaching the given accuracy.

**Theorem 2** (Convergence rate)**.** *Suppose Assumption 1 and the convergence condition* (4) *hold. Given an initial time $t_0$, an initial value $s_0$, and a step size $h$, the sequence $\{x_k\}_{k=0}^{\infty}$ is generated by Algorithm 1. We denote the continuous time interpolation $\overline{x}(t)$ as*

$$\overline{x}(t) = x_k + \frac{x_{k+1} - x_k}{h}(t - t_k), \qquad t \in [t_k, t_{k+1}) \tag{15}$$

*Assume three constants $0 \le C_1$, $0 < C_2 \le 1/h - 1/t_0$, and $0 < C_3$ fulfill the growth condition:*

$$|\dot{\beta}(t)| \le C_1\beta(t), \quad |\dot{\gamma}(t) - \ddot{\beta}(t)| \le C_2(\gamma(t) - \dot{\beta}(t)), \quad \beta(t) \le C_3 w(t). \tag{16}$$

*Then, it holds $f(x_k) - f_\star \le \mathcal{O}(1/k)$ under the following stability condition:*

$$\Lambda(x, f) \ge \|\nabla^2 f(x)\|, \quad \alpha\beta(t)/t \le \gamma(t) - \dot{\beta}(t) \le \beta(t)/h, \tag{17}$$

$$\sqrt{\int_0^1 \Lambda((1-\tau)X(t, \Xi, f) + \tau\overline{x}(t), f)\,\mathrm{d}\tau} \le \frac{\sqrt{\gamma(t) - \dot{\beta}(t)} + \sqrt{\gamma(t) - \dot{\beta}(t) - \frac{\alpha}{t}\beta(t)}}{\beta(t)}. \tag{18}$$

**Remark.** Theorem 2 demonstrates that for an $L$-smooth convex function $f$, selecting $\Lambda(x, f) \equiv L$, $\alpha > 3$, $\beta(t) = (4 - 2\alpha h/t)/L$, and $\gamma(t) = \beta(t)/h$ guarantees the convergence of Algorithm 1. This encompasses a fairly large family of algorithms.

**Remark.** The condition (16) is not restrictive, since it only requires the growth rate of the coefficients does not excess the exponential rate. In practice, they are often dominated by some polynomials. Our stability condition shows that the Hessian-driven term $\nabla^2 f(x(t))\dot{x}(t)$ is crucial for achieving the stable discretization. When $\beta \equiv 0$, which corresponding to the vanishing Hessian-driven damping, the condition (17) cannot be satisfied.

**Remark.** When $\nabla^2 f$ is $L_H$-Lipschitz continuous, the condition (18) can be simplified as

$$\beta(t)\sqrt{\Lambda(X(t, \Xi, f), f) + \frac{L_H M_3}{2\sqrt{t}}} \le \sqrt{\gamma(t) - \dot{\beta}(t)} + \sqrt{\gamma(t) - \dot{\beta}(t) - \frac{\alpha}{t}\beta(t)}, \tag{19}$$

where $M_3$ is a constant determined by $f, \Xi, x_0, t_0, \kappa$, and $\lambda$. We prove this conclusion as a corollary of Theorem 2 in sec. 5. The simplified condition (19) is important for reducing the computation overhead.

The proof of Theorem 2 is deferred in sec. 5. We establish this result by controlling the deviation between the sequences $\{x_k\}_{k=0}^{\infty}$ and $\{X(\Xi, t_k, f)\}_{k=0}^{\infty}$. The key step involves decomposing the global discretization error and adjusting the coefficients to prevent it from exploding during

the propagation process. This results in the stability condition that aligns with the conventional wisdom in numerical solutions of ODEs: the curvature of the trajectory cannot be excessively large.

The observation in sec. 1 underscores that the convergence rate in Theorem 2 cannot be directly analogous to its continuous-time counterpart, $\mathcal{O}(1/\delta(t))$, as presented in Theorem 1. A stable discretization, achievable over a broad range of coefficients, comes at the cost of a reduced convergence rate. The primary factor in this reduction is the error introduced during the discretization process. To the best of our knowledge, in comparison with other conditions that ensure the stable discretization of (1), our results offer distinct advantages: 1) They allow for more flexible requirements on the coefficients, whereas other approaches may mandate specific values for the coefficients; 2) They introduce tools from the numerical solution of differential equations to analyze iterative optimization algorithms, which we believe is a valuable effort in bridging these two fields; 3) They theoretically establish the stability of an explicit discretization with a constant step size applied to optimization-inspired ODEs in a general convex setting, contrasting with other convergence results that necessitate modifying the setting to strong convexity or diminishing step size [56, 57, 65, 66].

# 3 Selecting the best coefficients of ISHD using L2O

## 3.1 The problem formulation of L2O

Before introducing the loss function used in the problem formulation of L2O, we briefly review *oracle complexity*. It assesses the efficiency of optimization methods by the computational effort required to achieve a specified level of suboptimality. Suppose $\mathcal{F}$ is a collection that contains a family of functions with certain structure, e.g., convex functions, $L$-smooth functions. The iterative algorithm $\mathcal{M}(\cdot,\cdot,\cdot)$ maps the triplet $(f, x_0, N)$ to the point $x_N = \mathcal{M}(f, x_0, N)$. The computational cost for obtaining the output $\mathcal{M}(f, x_0, N)$ is proportional to $N$. Let $m(f, x_n)$ be an optimality measure. For example, a popular choice in unconstrained smooth optimization is $m(f, x_n) = \|\nabla f(x_n)\|$. Then, the complexity for method $\mathcal{M}$ applied to function class $\mathcal{F}$ is defined as

$$N_{\mathcal{F}} := \inf\{N \colon m(f, \mathcal{M}(f, x_0, N)) \leq \varepsilon, \text{ for all } f \in \mathcal{F}\} \tag{20}$$

The analysis of complexity has motivated the development of numerous efficient optimization algorithms, as comprehensively discussed in [47]. Inspired by this paradigm of developing efficient algorithms through complexity analysis, we define the continuous-time counterpart of complexity using the trajectory of (1) as a surrogate for optimization methods in the continuous-time case, which serves as the loss function in the problem formulation of L2O.

**Definition 1** (Stopping Time)**.** *Given the initial time $t_0$, the initial value $x_0$, the initial velocity $\dot{x}(t_0)$, the trajectory $X(\Xi, t, f)$ of the ISHD* (1)*, and a tolerance $\varepsilon$, the stopping time of the criterion $\|\nabla f(x)\| \leq \varepsilon$ is*

$$T(\Xi, f) = \inf\{t \mid \|\nabla f(X(\Xi, t, f))\| \leq \varepsilon, t \geq t_0\} \tag{21}$$

From the definition, we have $T = \infty$ if $\inf_{t \geq t_0} \|\nabla f(X(\Xi, t, f))\| \geq \varepsilon$. We mention that while $x(t)$ glosses over the challenges in numerical computation, the stopping time measures the efficiency of Algorithm 1 when the corresponding ODE can be stably discretized with a fixed step size. The stopping time measures the efficiency of Algorithm 1, when the sequence converges to the optimal value.

As we will see in sec. 4, the stopping time is differentiable with respect to $\Xi$. This fact gives us the hope of integrating the stopping time into L2O. In L2O, a Bayesian approach is used to

describe a (parametric) function class $\mathcal{F}$. Consider a mapping from a set of parameters to a set of functions:

$$\mathcal{H} : \Omega \to \mathcal{F}, \quad \zeta \to f(\cdot; \zeta) \tag{22}$$

To ensure measurability, we assume that $\mathcal{H}$ is a bijection, which allows us to define a probability distribution on $\mathcal{F}$ through the probability distribution on $\Omega$.

**Definition 2** (Induced Probability Space)**.** *Given the probability space of the parameter $\zeta$, $(\Omega, \mathscr{A}, \mathbb{P})$, where $\Omega$ is the sample space, $\mathscr{A}$ is the $\sigma$-algebra, and $\mathbb{P}$ is the corresponding probability. We define a $\sigma$-algebra $\mathcal{H}(\mathscr{A})$ as*

$$\mathcal{H}(\mathscr{A}) = \{\mathcal{H}(A) \mid A \in \mathscr{A}\} \tag{23}$$

*The probability for an event $\mathcal{B} \in \mathcal{H}(\mathscr{A})$ is given by $\mathbb{P}_{\mathcal{H}}(\mathcal{B}) = \mathbb{P}(\{\Omega \mid \mathcal{H}(\Omega) \in \mathcal{B}\})$. Consequently, the space $(\mathcal{F}, \mathcal{H}(\mathscr{A}), \mathbb{P}_{\mathcal{H}})$ is isomorphic to $(\Omega, \mathscr{A}, \mathbb{P})$, and it is referred to as the induced probability space of $\mathcal{F}$.*

In this paper, to support intuitive understanding, we assume that when discussing the distribution of functions, integrability over this distribution is maintained, and that the differentiation operator with respect to variables and the expectation operator with respect to the functions are interchangeable. A thorough examination of the interchangeability between the integral and differentiation operators will be left for future work.

Using this definition, we obtain an explicit expression for the expectation of $f$. This expression is exemplified by using the stopping time:

$$\mathbb{E}_f[T(\Xi, f)] = \int_f T(\cdot, f)\, \mathrm{d}\mathbb{P}_{\mathcal{H}}(f) = \int_\zeta T(\cdot, \mathcal{H}^{-1}(f))\, \mathrm{d}\mathbb{P}(\zeta) = \mathbb{E}_\zeta[T(\cdot, f(\cdot, \zeta))]$$

In this way, we can directly calculate the expectation with respect to a probability distribution of functions. Given that the spaces $(\mathcal{F}, \mathcal{H}(\mathscr{A}), \mathbb{P}_{\mathcal{H}})$ and $(\Omega, \mathscr{A}, \mathbb{P})$ are isomorphic, we frequently interchange their notation without causing confusion.

Now, we present the problem formulation for selecting the best coefficients of the equation (1). Given the probability space of the parameterized functions $(\mathcal{F}, \mathcal{H}(\mathscr{A}), \mathbb{P}_{\mathcal{H}})$, we use the expectation of stopping time over $\mathbb{P}$ to measure the efficiency of the trajectory of the equation (1) with coefficients $(\alpha, \beta(\cdot), \gamma(\cdot))$. To make the stopping time also be a reasonable metric of the efficiency of Algorithm 1, we add the stability condition (17), (18) and convergence condition (4) for each $f \in \mathcal{F}$ as constraints. These constraints ensure that the sequence $\{x_k\}_{k=0}^{\infty}$ generated by Algorithm 1 with the learned coefficients converges for each $f \in \mathcal{F}$.

To simplify the notations used in our L2O problem, with $w(t), \delta(t)$ defined in (4), and an integrable function satisfying $\Lambda(x, f) \geq \|\nabla^2 f(x)\|$, we introduce

$$p(x, \overline{x}, \Xi, t, f) = \Bigg[\beta(t)\sqrt{\int_0^1 \Lambda((1-\tau)x + \tau\overline{x}, f)\, \mathrm{d}\tau} - \sqrt{\gamma(t) - \dot{\beta}(t)} - \sqrt{\gamma(t) - \dot{\beta}(t) - \frac{\alpha}{t}\beta(t)}\Bigg]_+, \tag{24}$$

$$q(\Xi, t) = \Big[\gamma(t) - \dot{\beta}(t) - \beta(t)/h\Big]_+ + \Big[\dot{\beta}(t) + \alpha\beta(t)/t - \gamma(t)\Big]_+ + \Big[\dot{\delta}(t) - \lambda t w(t)\Big]_+ + [-\delta(t)]_+ . \tag{25}$$

Using notations (24) and (25), we set

$$P(\Xi, f) = \int_{t_0}^{T(\Xi,f)} p(X(t, \Xi, f), \overline{x}(t), \Xi, t, f)\,\mathrm{d}t, \quad Q(\Xi, f) = \int_{t_0}^{T(\Xi,f)} q(\Xi, t)\,\mathrm{d}t. \tag{26}$$

Here $\Xi$ represents the triplet of coefficients $(\alpha, \beta(\cdot), \gamma(\cdot))$. Using the property of integration and notations in (26), simply setting $P(\Xi, f) \leq 0$ and $Q(\Xi, f) \leq 0$ ensures that the conditions (4), (17), and (18) hold almost surely in $[t_0, T(\Xi, f)]$.

The L2O problem writes

$$\min_{\Xi} \quad \mathbb{E}_f[T(\Xi, f)], \tag{27a}$$

$$\text{s.t.} \quad \mathbb{E}_f[P(\Xi, f)] \leq 0, \tag{27b}$$

$$\mathbb{E}_f[Q(\Xi, f)] \leq 0. \tag{27c}$$

Using the property of the expectation, constraints (27b) and (27c) guarantee that $P(\Xi, f) \leq 0$ and $Q(\Xi, f) \leq 0$ hold almost surely for functions in $\mathcal{F}$. As we have argued, $P(\Xi, f) \leq 0$ and $Q(\Xi, f) \leq 0$ are sufficient to guarantee that the convergence condition (4) and stability condition (17), (18) hold almost surely in interval $[t_0, T(\Xi, f)]$. Hence, given the integrability of (27b), (27c), these constraints ensure that the learned ODE is convergent and can be stably discretized until it achieves the specific suboptimality metric $\varepsilon$, for every function that can be sampled from the distribution $\mathbb{P}$.

In a word, the L2O problem (27) selects the best ODE by minimizing the expectation of stopping time over a distribution of functions while keeping the stability of the discretization using scheme (12) for every function that can be sampled from the distribution. We argue that the constraints of the problem are indispensable. Without the guarantee for the stability, one may achieve arbitrary fast convergence rate in continuous time. However, this rate can not be translated into discrete-time cases, and does not derive a practical optimization method.

### 3.2 Solving the L2O problem using penalty method and stochastic optimization

To solve problem (27) numerically, we parameterize $\beta(\cdot)$ as $\beta_{\theta_1}(\cdot)$ and $\gamma(\cdot)$ as $\gamma_{\theta_2}(\cdot)$, and collectively denote the parameters $(a, \theta_1, \theta_2)$ as $\theta$, the total count of all parameters as $n_\theta$. For instance, when $\beta$ and $\gamma$ are parameterized through neural networks, $\theta_1$ and $\theta_2$ represent the trainable weights. Alternatively, one might opt for different models, such as polynomials, for parameterization. To streamline the notation, we substitute the symbol $\Xi$ with $\theta$ to distinguish between models with and without parameterization. Consequently, the mapping $\psi_\Xi$, delineated in (10), is rearticulated as $\psi_\theta$. The functions $P$ and $Q$, initially introduced in (26), are now expressed as $P(\theta, f)$ and $Q(\theta, f)$, respectively.

In problem (27), both the objective function and the constraint functions are expectations of the same probability distribution of functions $f$. Based on the linearity of expectation, it is appealing to use an exact penalty method for solving problem (27). Given the penalty parameter $\rho$, the $\ell_1$ exact penalty problem for framework (27) writes

$$\begin{aligned} \min_{\theta} \ \Upsilon(\theta) &= \mathbb{E}_f[T(\theta, f)] + \rho\left(\mathbb{E}_f[P(\theta, f)] + \mathbb{E}_f[Q(\theta, f)]\right) \\ &= \mathbb{E}_f\left[T(\theta, f) + \rho\left(P(\theta, f) + Q(\theta, f)\right)\right] \end{aligned} \tag{28}$$

We omit the $\ell_1$ norm of the constraints, since they are nonnegative functions. The penalty function has an expectation form. Hence, we can solve problem (28) using a suitable stochastic optimization method.

Before approaching the algorithm, we give the formulas for calculating the gradients of different components of $\Upsilon(\theta)$ directly. We implicitly assume that the required level of smoothness is satisfied. When the parameterization is based on neural networks, this can be achieved via using differentiable activation functions like SoftPlus [68]. The formula for of the stopping time writes

$$\frac{\mathrm{d}T}{\mathrm{d}\theta} = \Big(\nabla f(X)^\top \nabla^2 f(X)\,(v(T) - X - \beta(T)\nabla f(X))\Big)^{-1} \int_{t=t_0}^{T} r^\top \frac{\partial \psi_\theta}{\partial \theta}\,\mathrm{d}t \tag{29}$$

The notation $X$ represents the position of the trajectory at time $T$ and $v$ is calculated through the equation (11). The function $r$ is a solution of the following ODE

$$\begin{cases} \dot{r}(t) = -r(t)^\top \dfrac{\partial \psi_\theta}{\partial s}(s(t), t) \text{ from } t_1 \text{ to } t_0, \\ r(t_1) = (\nabla f(X)^\top \nabla^2 f(X), \mathbf{0}^\top)^\top \end{cases} \tag{30}$$

Applying the formula of the gradient of the stopping time and the chain rule to (26), we have

$$\begin{aligned} \frac{\mathrm{d}P(\theta, f)}{\mathrm{d}\theta} &= \int_{t_0}^{T(\theta,f)} \frac{\partial \psi_\theta(s(t), t, f)}{\partial \theta} w(t) + \frac{\partial p(x(t), \overline{x}(t), \theta, t, f)}{\partial \theta}\,\mathrm{d}t \\ &\qquad + p(x(T), \overline{x}(T), \theta, T, f)\frac{\mathrm{d}T(\theta, f)}{\mathrm{d}\theta}, \\ \frac{\mathrm{d}Q(\theta, f)}{\mathrm{d}\theta} &= \int_{t_0}^{T(\theta,f)} \frac{\mathrm{d}q(\theta, t)}{\mathrm{d}\theta}\,\mathrm{d}t + q(\theta, T)\frac{\mathrm{d}T(\theta, f)}{\mathrm{d}\theta} \end{aligned} \tag{31}$$

where $\overline{x}(\cdot)$ is the interpolation defined in (15), $w(\cdot)$ is the solution of the following differential equation

$$-\begin{pmatrix} \frac{\partial p(x(t), \overline{x}(t), \theta, t, f)}{\partial x} \\ 0_{n\times 1} \end{pmatrix} - \frac{\partial \psi_\theta(s(t), t, f)}{\partial s} w(t) = \dot{w}(t), \quad w(T(\theta, f)) = 0_{2n\times 1} \tag{32}$$

The gradient $\partial p/\partial x$ are contingent on the estimator $\Lambda(x, f)$ for the local Lipschitz constant of $\nabla f$. A detailed discussion is reserved for sec. 4.4.

The derivation of these formulas are postponed in sec. 4. All of them can be computed using the automatic differentiation, which largely reduces the complexity in practical implementation. We also offer a version for cases where the parameterized coefficients lack smoothness. This alternative is grounded in a recently developed concept known as the conservative gradient, a generalized derivative that accommodates the use of non-smooth activation functions in neural network parameterization. Using conservative gradients, we can also confirm that our implementation based on the automatic differentiation is reasonable and has a sound theoretical foundation.

Given the formula for the gradients with respect to the objective function and constraint functions, we use stochastic gradient descent to minimize the exact $\ell_1$ penalty function (28). The procedure is outlined in Algorithm 2. In iteration $k$, a function $f_k$ is sampled from the distribution of functions, and the sampled penalty function is constructed. Then, the gradient is calculated, followed by a stochastic gradient update. This algorithm could be enhanced with mini-batch training, variance reduction, and adaptive optimization methods. For simplicity, we implement the basic form of SGD.

We note that step 4 in Algorithm 2 allows for the use of conservative gradients in nonsmooth cases. While the conservative gradient does not possess an explicit formula, it can be computed through automatic differentiation. This approach, albeit a bit indirect, is viable and practical.

**Algorithm 2** Stochastic Penalty Method (StoPM) for Problem (27)

**Input:** step size for Algorithm 1: $h$, step size for stochastic gradient descent: $\eta$, initial weight: $\theta_0$, penalty coefficient: $\rho$, a probability space of functions: $(\Omega, \mathscr{A}, \mathbb{P})$.
**while** Not`(Stopping Condition)` **do**
  Sample a function in $f_k \in \Omega$ according to the probability $\mathbb{P}$.
  Computing the (conservative) gradients $\frac{\mathrm{d}T}{\mathrm{d}\theta}$, $\frac{\mathrm{d}P}{\mathrm{d}\theta}$ and $\frac{\mathrm{d}Q}{\mathrm{d}\theta}$ using (29) and (31) ((43), (50), and (48)).
  Update the parameter: $\theta_{k+1} \leftarrow \theta_k - \eta \left( \frac{\mathrm{d}T}{\mathrm{d}\theta} + \rho \left( \frac{\mathrm{d}P}{\mathrm{d}\theta} + \frac{\mathrm{d}Q}{\mathrm{d}\theta} \right) \right)$.
  Update index: $k \leftarrow k + 1$.
**end while**
**Output:** the trained weight $\theta_\star$.

# 4 Deriving the conservative gradient of the penalty function

In this section, our primary goal is to enhance the theoretical rigor of Algorithm 2. We utilize the conservative gradient to characterize the outputs obtained from applying automatic differentiation to $\Upsilon(\theta)$. This characterization becomes essential when the parameterization includes nonsmooth components such as the ReLU and max operators, which are prevalent in neural network architectures. The conservative Jacobian offers a generalized approach to derivatives in scenarios where traditional differentiability does not apply. This is particularly pertinent to both forward and backward nonsmooth automatic differentiation techniques crucial in machine learning, especially within backpropagation algorithms. The parameterization discussed here aligns with that introduced in sec. 3.2. For readers more interested in the computational and implementation aspects of the algorithm, it is sufficient to directly apply the corollaries following each theorem, as they are intuitive and straightforward.

## 4.1 The conservative Jacobian of the flow $X(t, \theta, f)$

This subsection investigates the conservative Jacobian for the flow $X(t, \Xi, f)$, starting with basic concepts in set-valued maps. The double arrow notation $\rightrightarrows$ used in set-valued maps like $D : \mathbb{R}^{d_1} \rightrightarrows \mathbb{R}^{d_2}$ indicates that each element in the domain maps to a subset of the codomain, highlighting the multi-valued aspect of these mappings. A set-valued map is termed *locally bounded* if, for any $x \in \mathbb{R}^{d_1}$, there exists a neighborhood $V_x$ such that the union $\bigcup_{y \in V_x} D(y)$ remains a bounded subset of $\mathbb{R}^{d_2}$. Furthermore, such a map is *graph closed* if for any converging sequences $(x_k)_{k\in\mathbb{N}} \subseteq \mathbb{R}^{d_1}$ and $(v_k)_{k\in\mathbb{N}} \subseteq \mathbb{R}^{d_2}$, with $v_k \in D(x_k)$, the limit $\lim_{k\to\infty} v_k$ belongs to $D(\lim_{k\to\infty} x_k)$. These properties are critical in defining and understanding the conservative Jacobian, as detailed below.

**Definition 3** (Conservative Jacobian)**.** *Consider a nonempty, locally bounded, and graph-closed set-valued map $D : \mathbb{R}^{d_1} \rightrightarrows \mathbb{R}^{m\times d_1}$, and a locally Lipschitz continuous function $F : \mathbb{R}^{d_1} \to \mathbb{R}^m$. The map $D$ is termed the conservative Jacobian of $F$ if and only if*

$$\frac{\mathrm{d}}{\mathrm{d}t} F(r(t)) = A\dot{r}(t), \quad \textit{for all } A \in D(r(t)),$$

*holds for any absolutely continuous curve $r : [0, 1] \to \mathbb{R}^{d_1}$ and almost every $t \in [0, 1]$. When $m = 1$, we refer to $D$ as a conservative gradient.*

Based on conservative Jacobian, we give the definition of path differentiability. This concept is introduced in [18] and enables the application of the chain rule and various calculus operations on nonsmooth functions.

**Definition 4** (Path differentiability)**.** *A function $F : \mathbb{R}^d \to \mathbb{R}^m$ is termed path differentiable if there exists a set-valued map $D$ that serves as a conservative Jacobian for $F$.*

To investigate the relationship between the path differentiability of the vector field and the flow in a general ODE, we consider the ordinary differential equation for $t_1 \geq t_0 > 0$

$$\dot{Y}(t) = F(Y(t)), \quad Y(0) = y_0, \quad \forall t \in [t_0, t_1] \tag{33}$$

where $F\colon \mathbb{R}^d \to \mathbb{R}^d$ is a path differentiable Lipschitz function and $y_0 \in \mathbb{R}^d$. Let $\phi(y_0, t)$ be the flow associated with (33) and $D^F$ be a uniformly bounded convex valued conservative Jacobian of function $F$, i.e., $\sup_{y\in\mathbb{R}^d, J\in D^F(y)} \|J\|_{\text{op}} \leq K$ for certain $K > 0$. Then, the adjoint sensitivity equation of (33) based on conservative Jacobian writes

$$\begin{aligned} &\dot{A}(t) \in D^F(\phi(y_0, t))A(t), \quad \text{for almost all } t \in [t_0, t_1], \\ &A(t_0) = I \in \mathbb{R}^{d\times d} \end{aligned} \tag{34}$$

Since $D^F$ is uniformly bounded, the existence of the solutions of (34) is established by [8, Theorem 4, p. 101].

Using these notations, we now present a fundamental theorem on the path differentiability of ODE flows. For a comprehensive proof, which is both lengthy and complex, the reader is referred to Theorem 1 of [44].

**Theorem 3** (Path differentiability of ODE flows)**.** *Define the set-valued map $U : \mathbb{R}^d \times [t_0, t_1] \to \mathbb{R}^{d\times d}$ as:*

$$\begin{aligned} U\colon \mathbb{R}^d \times [t_0, t_1] &\to \mathbb{R}^{d\times d}, \\ (y_0, t) &\rightrightarrows U(y_0, t) := A(t), \quad A(\cdot) \text{ is a solution to (34)} \end{aligned} \tag{35}$$

*The map $y_0 \rightrightarrows U(y_0, t_1)$ constitutes a conservative Jacobian for the mapping $y_0 \mapsto \phi(y_0, t_1)$.*

This theorem demonstrates that a path differentiable vector field ensures the path differentiability of the corresponding ODE flow. We have adapted the notation for clarity and to maintain consistency with the conventions of this paper. To apply it in (11), we need the following projection lemma, adopted from [44], to focus on the part of the conservative Jacobian we are interested in.

**Lemma 1** (The projection preserves the conservativity)**.** *Let $G(y, z) : \mathbb{R}^{d_y+d_z} \to \mathbb{R}^m$ be a path differentiable function whose conservative Jacobian is denoted by $D^G : \mathbb{R}^{d_y+d_z} \to \mathbb{R}^{m\times(d_y+d_z)}$. Consider*

$$\Pi_z D^G(y, z) := \{M_2 \in \mathbb{R}^{m\times d_z}, \exists M_1 \in \mathbb{R}^{m\times d_y}, (M_1, M_2) \in D^G(y, z)\}.$$

*Then, for all $y \in \mathbb{R}^{d_y}$, $\Pi_z D^G(y, z)$ is a conservative Jacobian for the function $z \mapsto G(y, z)$.*

For clarity, we denote the conservative Jacobian of function $G$ with respect to (projected onto) $z$ as $D^G_z(y, z) = \Pi_z D^G(y, z)$. Note that we may omit some variables when they are clear from the context. Applying Theorem 3 and Lemma 1, we confirm the path differentiability of the flow $X(t, \theta, f)$ associated with (11). We consider the parameterized vector field $\psi_\theta$ in equation (11) as a mapping from $\mathbb{R}^{2n+1+n_\theta}$ to $\mathbb{R}^{2n}$, defined as:

$$\psi\colon (s, t, \theta) \mapsto \psi_\theta(s, t)$$

Here, we use $\psi$ to highlight that the parameter $\theta$, previously considered fixed, is now treated as a variable alongside $s$ and $t$. This allows us to analyze the behavior of the vector field as both the input variables and the parameter vary.

**Theorem 4.** *Assume that $\psi$ is path differentiable, its corresponding conservative Jacobian $D^\psi \colon \mathbb{R}^{2n+1+n_\theta} \rightrightarrows \mathbb{R}^{2n\times(2n+1+n_\theta)}$ exists and $(x(t), v(t))$ is the flow of the equation (11). Let $J_s^\psi(t)$ and $J_\theta^\psi(t)$ be measurable selections such that for all $t \in [t_0, t_1]$, $J_s^\psi(t) \in D_s^\psi(x(t), v(t), t, \theta)$ and $J_\theta^\psi(t) \in D_\theta^\psi(x(t), v(t), t, \theta)$ (existence ensured by [44, Lemma 2]). Consider $A(\cdot)$ as the solution to the matrix differential inclusion*

$$\begin{aligned} &\dot{A}(t) = J_s^\psi(t)A(t) + J_\theta^\psi(t), \\ &A(t_1) = 0_{2n\times n_\theta} \quad \textit{for all } t \in [t_0, t_1] \end{aligned} \tag{36}$$

*Then, the map $\theta \mapsto A(t_0)$ is a conservative Jacobian of $\theta \to s(t, s_0, \theta, f)$, where $s$ is the flow associated with the equation (11).*

*Proof.* Concatenate the vector as $Y(t) = (x(t), v(t), t, \theta)$, giving the autonomous form of (11) as $\dot{Y}(t) = F(Y(t))$ where

$$F(Y(t)) = \begin{pmatrix} \psi_\theta(x(t), v(t), t) \\ 1 \\ 0_{n_\theta\times 1} \end{pmatrix} \tag{37}$$

With $\psi$ being path differentiable, the function $F$ also exhibits path differentiability. A conservative Jacobian of $F$ writes

$$\begin{aligned} D^F \colon \mathbb{R}^{2n} \times \mathbb{R} \times \mathbb{R}^p &\rightrightarrows \mathbb{R}^{(2n+1+n_\theta)\times(2n+1+n_\theta)} \\ (\widehat{x}, \widehat{v}, t, \theta) &\rightrightarrows \begin{pmatrix} J_s & J_t & J_\theta \\ 0_{1\times 2n} & 1 & 0_{1\times n_\theta} \\ 0_{n_\theta\times 2n} & 0_{n_\theta\times 1} & 0_{n_\theta\times n_\theta} \end{pmatrix}, \quad \forall (J_s, J_t, J_\theta) \in D^\psi \end{aligned}$$

Invoking Theorem 3, we partition the matrix $A(t)$ appearing in the sensitivity equation (34) as

$$A(t) = \begin{pmatrix} A_1(t) & A_2(t) & A_3(t) \\ A_4(t) & A_5(t) & A_6(t) \\ A_7(t) & A_8(t) & A_9(t) \end{pmatrix}$$

where $A_1(t) \in \mathbb{R}^{2n\times 2n}, A_5(t) \in \mathbb{R}, A_9(t) \in \mathbb{R}^{n_\theta\times n_\theta}$. Combining the partition above with the boundary condition $A(t_1) = I_{(2n+1+n_\theta)\times(2n+1+n_\theta)}$ yields

$$\begin{aligned} &A_4(t) \equiv 0_{1\times 2n}, \quad A_5(t) = \exp(t - t_1), \quad A_6(t) \equiv 0_{1\times n_\theta}, \\ &A_7(t) \equiv 0_{n_\theta\times 2n}, \quad A_8(t) \equiv 0_{n_\theta\times 1}, \quad A_9 \equiv I_{n_\theta\times n_\theta} \end{aligned}$$

The matrix differential inclusion (34) resolves to

$$\dot{A}_1(t) = J_s(Y(t))A_1(t), \qquad A_1(t_0) = I_{2n\times 2n}, \tag{38}$$

$$\begin{aligned} &\dot{A}_2(t) = J_s(Y(t))A_2(t) + J_t(Y(t))\exp(t - t_1), && A_2(t_0) = 0_{2n\times 1}, \\ &\dot{A}_3(t) = J_s(Y(t))A_3(t) + J_\theta(Y(t)), && A_3(t_0) = 0_{2n\times n_\theta}, \end{aligned} \tag{39}$$

$$\text{for all } (J_s, J_t, J_\theta) \in D^\psi, t \in [t_0, t_1].$$

Applying Lemma 1, we know $A_3$ is a conservative Jacobian of $\theta \to X(t_1, \theta, f)$. Omitting the subscription of $A_3$ leads to (36). □ □

Theorem 4 demonstrates that the flow $s$ maintains the path differentiability of the vector field $\psi_\theta$ and gives the form of the conservative gradient $D^s_\theta(t, s_0, \theta, f)$. By considering the part associated with $x$ of $D^s_\theta$, say the first $n$ rows, we get $D^X_\theta(t, \theta, f)$, the conservative gradient of $X(t, \theta, f)$, where the influence of $x_0$ and $v_0$ omitted as they are fixed. Consequently, we primarily need to verify the conditions ensuring the path differentiability of $\psi_\theta$. Notably, when parameterization employs neural networks, the vector field $\psi_\theta$ is almost invariably path differentiable. This follows from the observation that each element of $\psi_\theta$ is derived through finite operations of addition, subtraction, multiplication, and division applied to $t$, $\nabla f(x)$, $v$, $\alpha$, and the neural networks $\dot{\beta}_{\theta_1}$, $\beta_{\theta_1}$, and $\gamma_{\theta_2}$. Assuming the path differentiability of $\nabla f$ with respect to $x$, and based on the principle that the product and composition of path differentiable functions are also path differentiable, the remaining task is to ensure the path differentiability of the parameterized $\dot{\beta}\theta 1$, $\beta_{\theta_1}$, and $\gamma_{\theta_2}$. Indeed, the path differentiability of most neural networks can be substantiated using [18, Theorem 8].

We now give a definition of the *o*-minimal structures, following [24, 29].

**Definition 5** (*o*-minimal structure)**.** *An o-minimal structure is a collection $\{\mathcal{O}_d\}_{d\in\mathbb{N}}$, where $\mathcal{O}_d$ is a family of subsets of $\mathbb{R}^d$ such that for each $d \in \mathbb{N}$:*

1. *$\mathcal{O}_d$ contains $\mathbb{R}_d$ and is stable under the operations of complementation, finite union, and finite intersection, meaning that any set resulting from these operations on elements of $\mathcal{O}_p$ will also be a member of $\mathcal{O}_p$.*
2. *if $A$ belongs to $\mathcal{O}_d$, then $A \times \mathbb{R}$ and $\mathbb{R} \times A$ belong to $\mathcal{O}_{d+1}$;*
3. *if $\Pi\colon \mathbb{R}^{d+1} \to \mathbb{R}^d$ denotes the coordinate projection on the first $d$ coordinates, then for any $A \in \mathcal{O}_{d+1}$, we have $\Pi(A) \in \mathcal{O}_d$;*
4. *$\mathcal{O}_d$ contains all sets of the form $\{x \in \mathbb{R}^d\colon p(x) = 0\}$, where $p$ is a polynomial on $\mathbb{R}^d$;*
5. *the elements of $\mathcal{O}_1$ are exactly the finite unions of intervals (possibly infinite) and points.*

The sets $A$ belonging to $\mathcal{O}_d$, for some $d \in \mathbb{N}$, are called definable in the *o*-minimal structure. A set valued mapping (or a function) is said to be definable in $\mathcal{O}$ whenever its graph is definable in $\mathcal{O}$. In algorithmic aspect, it has been proved in [18] that the forward and backward automatic differentiation outputs a conservative Jacobian for the corresponding definable function [18, Theorem 8].

A typical example of the *o*-minimal structures is given by semialgebraic sets. A set $A \subseteq \mathbb{R}^d$ is called *semialgebraic* if it is a finite union of sets of the form

$$\{x \in \mathbb{R}^d : p_i(x) = 0, i = 1, \ldots, k; p_i(x) < 0, i = k+1, \ldots, m\} \tag{40}$$

where $p_f$ are real polynomial functions and $k \geq 1$. We remark that the property 3 of Definition 5 for semialgebraic sets is not trivial and can be obtained from the Tarski-Seidenberg theorem. Another profound result by Wilkie [60] shows that there exists an *o*-minimal structure that contains both the exponential function $x \mapsto e^x$ and all semialgebraic functions. According to the inverse function theorem of definable functions appearing in Chapter 7.3 of [28], the logarithm function $x \mapsto \log(x)$ is also definable to Wilkie's *o*-minimal structure. The Proposition 1.6 of [24] ensures the composition of two definable maps is definable.

Using these results, we conclude that the following functions are definable using Wilkie's *o*-minimal structure.

$$t \mapsto t, \qquad t \mapsto t^2, \qquad t \mapsto \log(1 + \exp(t)),$$

$$t \mapsto \max\{0, t\}, \qquad t \mapsto \tanh(t), \qquad t \mapsto \frac{1}{1+\exp(-t)}$$

Additionally, all polynomials and functions whose domains can be segmented into finitely many intervals, coinciding with the previously mentioned functions, are definable. These results confirm that almost all activation functions utilized in deep learning are definable. Consequently, [18, Theorem 8] validates the existence of conservative fields for the parameterized $\psi_\theta$ within neural networks.

### 4.2 The conservative gradient of the stopping time $T(\theta, f)$

We present a characterization of the conservative gradient of the stopping time, which is grounded in the integration by parts formula for absolutely continuous functions.

**Lemma 2** (Integration by parts)**.** *Suppose $a, b\colon [t_0, t_1] \to \mathbb{R}^d$ are absolutely continuous functions. We have*

$$\int_{t_0}^{t_1} a(t)^\top \dot{b}(t)\,\mathrm{d}t = a(t_1)^\top b(t_1) - a(0)^\top b(0) - \int_{t_0}^{t_1} \dot{a}(t)^\top b(t)\,\mathrm{d}t$$

The proof of Lemma 2 is provided in Theorem 10 of Chapter 6 in [52]. Utilizing this lemma in conjunction with Theorem 4, we derive the conservative gradient of $\|\nabla f(X(T, \theta, f))\|^2$, which serves as a fundamental component in the characterization of the conservative gradient of the stopping time.

**Proposition 1.** *Adopting the notations from Theorem 4, let $D^s_{s_0}$ and $D^s_\theta$ denote the conservative Jacobians of $s(t, \theta, s_0, f)$ with respect to $s_0$ and $\theta$, respectively. Assume $J^s_{s_0}\colon [t_0, T] \to \mathbb{R}^{2n\times 2n}$ and $J^s_\theta\colon [t_0, T] \to \mathbb{R}^{2n\times n_\theta}$ are measurable selections satisfying $J^s_{s_0}(t) \in D^s_{s_0}(t, \theta, s_0, f)$ and $J^s_\theta(t) \in D^s_\theta(t, \theta, s_0, f)$ for all $t \in [t_0, T]$. According to [31, Chapter 0, Theorem 2], there exists a unique absolutely continuous solution $\omega\colon [t_0, T] \to \mathbb{R}^{2n}$ to the differential equation*

$$\begin{aligned} \dot{\omega}(t) &= -J^s_{s_0}(t)\omega(t), \\ \omega(T) &= (\nabla f(X)^\top \nabla^2 f(X), 0_{1\times n})^\top \end{aligned} \tag{41}$$

*Then, an element of $D^{\|\nabla f(X(T,\cdot,f))\|^2}$ is given by $\int_{t=t_0}^{T} \omega(t)^\top J^\psi_\theta(t)\,\mathrm{d}t$.*

*Proof.* Consider the solution $A(t)$ of the sensitivity equation (36) for computing the conservative Jacobian of $X(t, \theta, f)$ in Theorem 4. For any absolutely continuous function $\omega\colon [t_0, T] \to \mathbb{R}^{2n}$, applying Lemma 2 and the initial condition $A(t_0) = 0_{2n\times n_\theta}$ gives

$$\begin{aligned} \int_{t_0}^{T} \omega(t)^\top \dot{A}(t)\,\mathrm{d}t &= \omega(T)^\top A(T) - \int_{t_0}^{T} A(t)^\top \dot{\omega}(t)\,\mathrm{d}t \\ &= \int_{t_0}^{T} \omega(t)^\top J^\psi_s(t)A(t) + \omega(t)^\top J^\psi_\theta(t)\,\mathrm{d}t \end{aligned}$$

Setting $\omega$ as the solution of (41) results in:

$$\nabla f(X)^\top \nabla^2 f(X) A(T) = \int_{t_0}^{T} \omega(t)^\top J^\psi_\theta(t)\,\mathrm{d}t$$

Given that the product of conservative gradients maintains the conservative gradient property [18, Lemma 5], the expression on the left hand side of the equation qualifies as an element of the conservative gradient in question. This completes the proof. □ □

The next theorem derives the conservative gradient of $T(\theta, f)$ with respect to $\theta$, applying the formal non-smooth implicit differentiation via the conservative Jacobian, as detailed in [17, Theorem 2].

**Proposition 2.** *Let the squared gradient norm $\|\nabla f(X(\cdot,\cdot,f))\|^2 \colon \mathbb{R}\times\mathbb{R}^{n_\theta} \to \mathbb{R}$ be path differentiable on an open set $\mathcal{U}\times\mathcal{V} \subset \mathbb{R}\times\mathbb{R}^{n_\theta}$ and $T(\theta, f)\colon \mathcal{V} \to \mathcal{U}$ a locally Lipschitz function such that, for each $\theta \in \mathcal{V}$*

$$\|\nabla f(X(T(\theta, f), \theta, f))\|^2 = \varepsilon^2 \tag{42}$$

*Assume that for each $\theta \in \mathcal{V}$ and for each $B \in D_t^{\|\nabla f(X(\cdot,\cdot,f))\|^2}(T(\theta, f), \theta)$, the matrix $B$ is invertible. Then, $T : \mathcal{V} \to \mathcal{U}$ is path differentiable, with its conservative gradient given by:*

$$D^T(\theta) := \{-B^{-1}A : \forall A \in D_\theta^{\|\nabla f(X(\cdot,\cdot,f))\|^2}, B \in D_t^{\|\nabla f(X(\cdot,\cdot,f))\|^2}\} \tag{43}$$

*where $D_\theta^{\|\nabla f(X(\cdot,\cdot,f))\|^2}$ is the conservative gradient of $\|\nabla f(X(T(\theta, f),\cdot, f))\|^2$, as specified by Proposition 1.*

*Proof.* Let $\theta(\iota)\colon [0, 1] \to \mathcal{U}$ be an absolutely continuous loop. Since $T$ is locally Lipschitz, the composition function $T\circ\theta$ must also be absolutely continuous. Consequently, $\|f(X(\theta(\iota), T(\theta(\iota), f), f))\|^2$ is differentiable with respect to $\iota$ almost everywhere. By differentiating it and applying the chain rule, for almost every $\iota \in [0, 1]$ and for any $A \in D_\theta^{\|\nabla f(X(\cdot,\cdot,f))\|^2}$, $B \in D_t^{\|\nabla f(X(\cdot,\cdot,f))\|^2}$, we have:

$$A\dot{\theta}(\iota) + B\frac{\mathrm{d}T}{\mathrm{d}\theta}\dot{\theta}(\iota) = 0 \tag{44}$$

Thus, according to the definition of a conservative gradient 3, we conclude that $-B^{-1}A$ is an element of the conservative gradient of $T(\theta, f)$. □ □

While the conservative gradient of $T$ can indeed be computed using automatic differentiation, the process is not inherently intuitive or straightforward. To facilitate a clearer grasp of the underlying theories, we give a direct derivation of the gradient of $T$ in smooth cases, starting with a corollary of Proposition 1. This corollary calculates the gradient of $\|\nabla f(X(t_1, \theta, f))\|^2$ in smooth cases.

**Corollary 1.** *Given $\psi_\theta$ is continuously differentiable with respect to $\theta$, $r \in \mathcal{C}^1([t_0, t_1], \mathbb{R}^{2n})$ is the solution of (30), we have*

$$\nabla f(X)^\top \nabla^2 f(X)\frac{\partial X}{\partial \theta} = \int_{t=t_0}^{t_1} r^\top \frac{\partial \psi_\theta}{\partial \theta}\,\mathrm{d}t \tag{45}$$

*Proof.* Consider the ODE system

$$\frac{\mathrm{d}s}{\mathrm{d}t}(t) = \psi_\theta(s(t), t), \quad t \in [t_0, t_1], \quad s(t_0) = (x_0^\top, v_0^\top)^\top$$

For any continuously differentiable function $r(t) \in \mathcal{C}^1([t_0, t_1], \mathbb{R}^n)$, we have

$$\int_{t=t_0}^{t_1} r(t)^\top\,\mathrm{d}s(t) = \int_{t=t_0}^{t_1} r(t)^\top \psi_\theta(s(t), t)\,\mathrm{d}t$$

Applying $\mathrm{d}/\mathrm{d}\theta$ to above equation and using $\frac{\mathrm{d}}{\mathrm{d}\theta}\frac{\mathrm{d}}{\mathrm{d}t}s(t) = \frac{\mathrm{d}}{\mathrm{d}t}\frac{\mathrm{d}}{\mathrm{d}\theta}s(t)$ yields

$$\int_{t=t_0}^{t_1} r^\top \mathrm{d}\left(\frac{\mathrm{d}s}{\mathrm{d}\theta}\right) = \int_{t=t_0}^{t_1} r^\top \left(\frac{\partial \psi_\theta}{\partial s}\frac{\mathrm{d}s}{\mathrm{d}\theta} + \frac{\partial \psi_\theta}{\partial \theta}\right) \mathrm{d}t$$

Here we omit the variables for simplicity. Integrating by parts and using the fact that $\mathrm{d}s(t_0)/\mathrm{d}\theta \equiv 0$ gives

$$r(t_1)^\top \frac{\mathrm{d}s}{\mathrm{d}\theta}(t_1) = \int_{t=t_0}^{t_1} \left(\dot{r}^\top + r^\top \frac{\partial \psi_\theta}{\partial s}\right)\frac{\mathrm{d}s}{\mathrm{d}\theta} + r^\top \frac{\partial \psi_\theta}{\partial \theta}\,\mathrm{d}t \tag{46}$$

It should be noted that $\mathrm{d}s/\mathrm{d}\theta$ includes the term $\mathrm{d}x/\mathrm{d}\theta$. To evaluate the value of $\nabla f(X)^\top \nabla^2 f(X)\partial X/\partial\theta$, we solve the backward ODE (30). Plugging the solution into equation (46) gives the formula (45). □ □

**Remark.** Using the same technique in Corollary 1, we can give a direct derivation of Theorem 4 in smooth case through choosing $r(t_1) = I_{:,i}$ for $i = 1, 2, \ldots, n$.

Similar to the process for deriving conservative gradients, once the gradient of $\|\nabla f(X(t,\theta,f))\|^2$ is established, the gradient of $T$ with respect to $\theta$ can be computed using implicit function theorem. This calculation serves as a corollary to Proposition 2.

**Corollary 2.** *Let $T = T(\theta, f)$ and $X = X(T(\theta, f), \theta, f)$. Assuming that $\psi_\theta$ is continuously differentiable with respect to $\theta$ and that $\nabla f(X)^\top \nabla^2 f(X)\frac{\partial X}{\partial \theta}$ is non-zero, the gradient of $T$ is then given by the formula* (29).

*Proof.* From the definition of the stopping time and the continuity of $\|\nabla f(\cdot)\|$, we have $\|\nabla f(X(T(\theta,f),\theta,f))\|^2 - \varepsilon^2 \equiv 0$. The implicit function theorem implies

$$\left.\frac{\partial \|\nabla f(X(t,\theta,f))\|^2}{\partial t}\right|_{t=T} \frac{\mathrm{d}T}{\mathrm{d}\theta} + \nabla f(X)^\top \nabla^2 f(X)\frac{\partial X}{\partial \theta} = 0 \tag{47}$$

The second term can be calculated using Proposition 1. The chain rule gives

$$\left.\frac{\partial \|\nabla f(X(t,\theta,f))\|^2}{\partial t}\right|_{t=T} = \nabla f(X)^\top \nabla^2 f(X)\left.\frac{\partial X}{\partial t}\right|_{t=T}$$

Using equation (11), we have

$$\left.\frac{\partial X}{\partial t}\right|_{t=T} = \dot{x}(T) = v(T) - X - \beta(T)\nabla f(X)$$

where $x(t) = X(t, \theta, f)$. Combining these results we get the formula (29). □ □

### 4.3 The conservative gradients of the constraint functions $P(\theta, f)$ and $Q(\theta, f)$

In this subsection, we present the conservative gradients of $P(\theta, f)$ and $Q(\theta, f)$. These functions can be expressed as definite integrals of the functions $p(\theta, t, f)$ and $q(\theta, t)$ over the interval $[t_0, T(\theta, f)]$. Assuming that both $p$ and $q$ are path differentiable, our results rely on a characterization of the conservative gradient of the integral of a path differentiable function. For the sake of completeness, we present this characterization here, while the proof can be found in [44, Theorem 2]. The notations used throughout this section are consistent with those introduced in Theorem 3.

**Theorem 5.** *Let $\delta\colon \mathbb{R}^d \to \mathbb{R}$ be a locally Lipschitz continuous and path differentiable function. Let $D^\delta\colon \mathbb{R}^d \to \mathbb{R}^d$ be a conservative gradient for $\delta$ with convex values. For $t_1 > 0$, set*

$$\Delta(y_0) = \int_{t_0}^{t_1} \delta(\phi(y_0, t))\,\mathrm{d}t$$

*where $\phi$ is the flow of the equation (33). Then, given any solution of (34), denoted by $A(t)$, the following set valued field is a conservative gradient for $\Delta$,*

$$D^\Delta\colon y_0 \rightrightarrows \left\{ \int_{t_0}^{t_1} A(t)^\top w(t)\,\mathrm{d}t, w \in \mathcal{W}(y_0) \right\}$$

*where $\mathcal{W}(y_0)$ is the set of measurable selections $w(t) \in D^\delta(\phi(y_0, t))$.*

Based on Theorem 5, we provide the conservative gradients of $P$ and $Q$, given that the conservative gradients $D^p_\theta$ and $D^q_\theta$ can be readily evaluated.

**Proposition 3.** *Assume that the function $q(\theta, t)$ is path differentiable and $J^q_\theta(t)$ is a measurable selection of the conservative gradient $D^q_\theta(\theta, t)$, satisfying $J^q_\theta(t) \in D^q_\theta(\theta, t)$ for $t \in [t_0, T(\theta, f)]$. Then, the following holds:*

$$\int_{t_0}^{T(\theta,f)} J^q_\theta(t)\,\mathrm{d}t + q(\theta, T(\theta, f))J^T_\theta \in D^Q \tag{48}$$

*where $D^Q$ is the conservative gradient of $Q(\theta, f)$.*

*Proof.* The conservative gradient of $Q$ is determined by substituting $F$ from equation (37) into Theorem 5. Notably, since $q$ does not involve $x(t)$ or $v(t)$ as variables, this substitution leads directly to equation (48). □ □

Next, we explore the conservative gradient of $P$. The steps here are similar to those in Proposition 3. However, a key difference is that $p$ involves the estimator $\Lambda(x, f)$, with $X(t, \theta, f)$ as an input. This estimator is significant both theoretically and practically. We will discuss this term and its corresponding conservative gradient in the next subsection and directly apply $D^q_x$ in the following proposition.

**Proposition 4.** *Assume that the functions $p(x, \overline{x}, \theta, t, f)$ is path differentiable, $J^p_x(t)$ is a measurable selection of $D^p_x(X(t, \theta, f), \overline{x}(t), \theta, t, f)$, $J^X_\theta(t)$ is a measurable selection of $D^X_\theta(t, \theta, f)$, and $J^p_\theta(t)$ is a measurable selection of the conservative gradient $D^p_\theta(X(t, \theta, f), \overline{x}(t), \theta, t, f)$, $\omega$ is the unique solution to the following equation*

$$-J^p_s(t) - J^\psi_s(t)\omega(t) = \dot{\omega}(t), \qquad \omega(T(\theta, f)) = 0_{2n\times 1} \tag{49}$$

*We have $\int_{t_0}^{T(\theta,f)} J^s_\theta(t)^\top J^p_s(t)\,\mathrm{d}t = -\int_{t_0}^{T(\theta,f)} J^\psi_\theta(t)^\top w(t)\,\mathrm{d}t$ and*

$$\int_{t_0}^{T} J^\psi_\theta(t)^\top w(t) + J^p_\theta(t)\,\mathrm{d}t + p(X(T, \theta, f), \overline{x}(T), \theta, T, f)J^T_\theta \in D^P \tag{50}$$

*For simplicity, we abbreviate $T(\theta, f)$ as $T$.*

*Proof.* The existence of the conservative gradients of $P, Q$ are ensured by Theorem 4 through augmenting the function $F$ with two additional dimensions $(p, q)$. Since the flow inherits the path differentiability of the corresponding vector field, we must have $P, Q$ are path differentiable given $p, q$ are path differentiable. Substituting the $F$ defined at the equation (37) in Theorem 5, the conservative gradient of $P$ writes

$$\int_{t_0}^{T} J_\theta^X(t)^\top J_x^p(t) + J_\theta^p(t)\,\mathrm{d}t + p(X(T,\theta,f),\overline{x}(T),\theta,T,f)J^T \in D^P(\theta) \tag{51}$$

Proposition 1 gives the method for obtaining an element $J^T$ from $D^T(\theta)$. However, a direct evaluation of $J_\theta^X(t)$ requires to solve a differential equation with the variable of the size $(2n+1+n_\theta)\times(2n+1+n_\theta)$. We derive a computational tractable way that only need solve a differential equation with size $(2n+1+n_\theta)\times 1$ to evaluate the integral in (51). Noticing that $J_s^p(t) = (J_x^p(t)^\top, 0)^\top$, we know

$$J_\theta^X(t)^\top J_x^p(t) = J_\theta^s(t)^\top J_s^p(t)$$

holds for any $t$. We use the fact that $J_\theta^s(t)$ satisfies the equation (36), say

$$\frac{\mathrm{d}}{\mathrm{d}t}J_\theta^s(t) = J_s^\psi(t)^\top J_\theta^s(t) + J_\theta^\psi(t), \quad J_\theta^s(t_1) = 0_{2n\times p}$$

Setting $t_1 = T(\theta, f)$, for any absolutely continuous function $\omega(t)\colon [t_0, t_1] \to \mathbb{R}^n$, we have

$$\begin{aligned}
&\int_{t_0}^{t_1} J_\theta^s(t)^\top J_s^p(t)\,\mathrm{d}t \\
=&\int_{t_0}^{t_1} J_\theta^s(t)^\top J_s^p(t) + \left(J_s^\psi(t)^\top J_\theta^s(t) + J_\theta^\psi(t) - J_s^\psi(t)^\top J_\theta^s(t) - J_\theta^\psi(t)\right)^\top w(t)\,\mathrm{d}t \\
=&\int_{t_0}^{t_1} J_\theta^s(t)^\top J_s^p(t) + \left(J_s^\psi(t)^\top J_\theta^s(t) + J_\theta^\psi(t)\right)^\top w(t)\,\mathrm{d}t - \int_{t_0}^{t_1}\left(\frac{\mathrm{d}}{\mathrm{d}t}J_\theta^s(t)\right)^\top w(t)\,\mathrm{d}t.
\end{aligned} \tag{52}$$

Using Lemma 2, we get

$$\begin{aligned}
&\int_{t_0}^{t_1} J_\theta^s(t)^\top J_s^p(t)\,\mathrm{d}t \\
=&\int_{t_0}^{t_1} J_\theta^s(t)^\top J_s^p(t) + \left(J_s^\psi(t)^\top J_\theta^s(t) + J_\theta^\psi(t)\right)^\top w(t)\,\mathrm{d}t \\
&\qquad - J_\theta^s(t_1)w(t_1) + J_\theta^s(t_0)w(t_0) + \int_{t_0}^{t_1} J_\theta^s(t)^\top \dot{w}(t)\,\mathrm{d}t \\
=&\int_{t_0}^{t_1} J_\theta^\psi(t)^\top w(t)\,\mathrm{d}t + \int_{t_0}^{t_1} J_\theta^s(t)^\top\left(J_s^p(t) + J_s^\psi(t)w(t) + \dot{w}(t)\right)\mathrm{d}t \\
&\qquad - J_\theta^s(t_1)w(t_1) + J_\theta^s(t_0)w(t_0)
\end{aligned} \tag{53}$$

Given $J_\theta^s(t_0) = 0_{2n\times 1}$, setting $\omega$ to the solution of (49) gives

$$\int_{t_0}^{T(\theta,f)} J_\theta^s(t)^\top J_s^p(t)\,\mathrm{d}t = -\int_{t_0}^{T(\theta,f)} J_\theta^\psi(t)^\top w(t)\,\mathrm{d}t$$

Combining this conclusion with (51) completes our proof. □ □

Using Remark  and the same argument as Proposition 4, we get a direct derivation of it in the smooth case. This is because the key steps (51) and (52) can be similarly derived by replacing the conservative gradient in the argument with the corresponding partial derivatives. The only challenge lies in deriving a smooth version of Theorem 4, which can be addressed using the approach of Proposition 1 and the technique mentioned in Remark .

**Corollary 3.** *Suppose the functions $p, q$, and $T$ are differentiable with respect to $\theta$. Then, the gradients of $P$ and $Q$ are given by the equation* (31)*.*

*Proof.* Applying the chain rule and the gradient of stopping time derived in Corollary 2 gives

$$\begin{aligned}\frac{\mathrm{d}P(\theta,f)}{\mathrm{d}\theta} &= \int_{t_0}^{T(\theta,f)} \frac{\partial p(x(t),\overline{x}(t),\theta,t,f)}{\partial x}\frac{\partial X(t,\theta,f)}{\partial \theta} + \frac{\partial p(x(t),\overline{x}(t),\theta,t,f)}{\partial \theta}\,\mathrm{d}t \\ &\qquad + p(x(T),\overline{x}(T),\theta,T,f)\frac{\mathrm{d}T(\theta,f)}{\mathrm{d}\theta}, \\ \frac{\mathrm{d}Q(\theta,f)}{\mathrm{d}\theta} &= \int_{t_0}^{T(\theta,f)} \frac{\mathrm{d}q(\theta,t)}{\mathrm{d}\theta}\,\mathrm{d}t + q(\theta,T)\frac{\mathrm{d}T(\theta,f)}{\mathrm{d}\theta}\end{aligned}$$

Then, the equation (31) can be proved verbosely like Lemma 4. One simply needs to replace the conservative gradient $J_s^{\psi}$ with the partial gradient $\partial\psi/\partial s$ and so forth. □ □

### 4.4 Choices of the estimators $\Lambda(x, f)$ for the local Lipschitz constant of $\nabla f$

This subsection discusses the computation of $D_x^p(X(t,\theta,f),\overline{x}(t),\theta,t,f)$. The following proposition characterizes the structure of the conservative gradient $D_x^p$.

**Proposition 5.** *Assume $\Lambda\colon \mathbb{R}^n \to \mathbb{R}$ is path differentiable. Then, an element of the conservative gradient $D_x^p(X(t,\theta,f),\overline{x}(t),\theta,t,f)$ is given by*

$$\frac{\beta_{\theta_1}(t)}{2\sqrt{\int_0^1 \Lambda((1-\tau)X(t,\theta,f)+\tau\overline{x}(t),f)\,\mathrm{d}\tau}}\int_0^1 J_x^{\Lambda}(\tau)\,\mathrm{d}\tau \tag{54}$$

*where $J_x^{\Lambda}(\tau)$ is a measurable selection satisfying*

$$J_x^{\Lambda}(\tau) \in D_x^{\Lambda}((1-\tau)X(t,\theta,f)+\tau\overline{x}(t),f), \qquad \forall \tau \in [0,1]$$

*Proof.* The equation (54) follows from the definition of conservative gradients and Theorem 5. □ □

As indicated by the proposition, the main challenge lies in computing the conservative gradient of the local Lipschitz constant estimator $\Lambda(x, f)$. We provide three choices of the estimator and derive the corresponding conservative gradients, with each choice determined by the underlying assumptions on the function $f$. We first consider the most general case: functions with local Lipschitz continuous gradient.

**Proposition 6.** *Suppose $f$ is twice differentiable and $\nabla^2 f(x)$ is path differentiable. Denote $\Lambda(x,f) = \lambda_{\max}(\nabla^2 f(x))$ as the largest eigenvalue of $f$ on point $x$. Then, $\Lambda(x,f)$ is path differentiable and*

$$\frac{\mathrm{d}}{\mathrm{d}\eta_2}\left(\frac{\mathrm{d}}{\mathrm{d}\eta_1}\nabla f(x+\eta_1 z+\eta_2 z)\Big|_{\eta_1=0}\right)\Big|_{\eta_2=0} \in D^{\Lambda}(x) \tag{55}$$

*where $\|z\| = 1$ and $z^\top \nabla^2 f(x) z = \lambda_{\max}(\nabla^2 f(x))$.*

*Proof.* A well known formula for the largest eigenvalue is

$$\lambda_{\max}(A) = \sup_{\|z\|=1} z^\top A z, \text{ for all } A \in \mathbb{S}^n \tag{56}$$

Using the fact that the point-wise maximum of a family of convex functions is convex, we know $\lambda_{\max}$ is convex. Furthermore, the subgradient of the supremum function at any point is the convex hull of the gradients of those functions in the family that attain the supremum at that point. This property gives the following characterization of the subgradient of $\lambda_{\max}$:

$$\partial^C \lambda_{\max}(A) = D^{\lambda_{\max}}(A) = \operatorname{co}\left(\bigcup_{z^\top A z = \lambda_{\max}(A)} \{zz^\top\}\right) \tag{57}$$

This helpful formula also appears in Corollary 2.4 of [54]. Here $\partial^C$ denotes the Clarke subdifferential as defined in Definition 10. The first equality arises from the property of conservative gradients, as outlined in [18], where the conservative gradient $D$ coincides with $\partial^C$ for convex functions. Combining the equation (57) and the chain rule for conservative gradients [18, Lemma 5], we get the following characterization of the conservative gradient:

$$\begin{aligned} &D^\Lambda(x, f) := \left\{ J^{\lambda_{\max}} \circ J^{\nabla^2 f} \mid J^{\lambda_{\max}} \in D^{\lambda_{\max}}(\nabla^2 f(x)),\ J^{\nabla^2 f} \in D^{\nabla^2 f}(x) \right\}, \\ &\text{where } D^{\lambda_{\max}}(\nabla^2 f(x)) = \operatorname{co}\left(\bigcup_{z^\top \nabla^2 f(x) z = \lambda_{\max}(\nabla^2 f(x))} \{zz^\top\}\right) \end{aligned} \tag{58}$$

The conservative gradient for $\nabla^2 f$ is a tensor of the size $\mathbb{R}^{n\times n\times n}$ and the operator $\circ$ is defined as

$$\left(J^{\lambda_{\max}} \circ J^{\nabla^2 f}\right)_k = \left\langle J^{\lambda_{\max}}, J_k^{\nabla^2 f} \right\rangle \tag{59}$$

where the subscription $k$ denotes the projection onto the $k$-th coordinate of $x$.

Next, we give a computationally tractable method to get an element of (58). Given the definition of the conservative gradients, we have

$$\frac{\mathrm{d}}{\mathrm{d}\eta} z^\top \nabla^2 f(x + \eta z)\bigg|_{\eta=0} = zz^\top \circ J^{\nabla^2 f}, \quad \forall z \in \mathbb{R}^n,\ J^{\nabla^2 f} \in D^{\nabla^2 f}(x)$$

Setting $\|z\| = 1$ and $z^\top \nabla^2 f(x) z = \lambda_{\max}(\nabla^2 f(x))$ gives $zz^\top \in D^{\lambda_{\max}}(\nabla^2 f(x))$. Hence, combining the equation above and (58) gives the equation (55). □ □

Note that the argument in Proposition 6 remains valid when the assumptions on $f$ are more relaxed. Therefore, we do not repeat the derivation for the corollary in the smooth case. Next, we demonstrate that the computational complexity of $zz^\top \circ J^{\nabla^2 f}$ is effectively just a constant multiple of the cost of evaluating $f(x)$.

Suppose the function $f(x)$ is automatically differentiable and can be evaluated with a computational complexity of $\mathcal{O}(N)$. According to [35], the gradient $\nabla f(x)$ can also be computed with the same complexity, $\mathcal{O}(N)$, via backward differentiation. Furthermore, using forward differentiation for $\nabla f(x + \eta_1 z)$, we can compute the directional derivative $z^\top \nabla^2 f(x)$ for a vector $z$ and a small perturbation $\eta_1$ with complexity $\mathcal{O}(N)$. This method applies equally to $z^\top \nabla^2 f(x + \eta_2 z)$, affirming that the complexity for evaluating the specified equation (55) remains $\mathcal{O}(N)$. In cases where the

function $f$ is not automatically differentiable, the gradient $\nabla f$ must be calculated symbolically, but this does not change the overall computational complexity for these operations.

Using power iteration [33] combined with forward automatic differentiation, the eigenvector $z$ referenced in equation (55) can be computed efficiently. The iterative process is defined as follows:

$$z_{k+1} = \frac{z_{k+1/2}}{\|z_{k+1/2}\|}, \quad z_{k+1/2} = \frac{\mathrm{d}}{\mathrm{d}\eta_1} \nabla f(x + \eta_1 z_k)\Big|_{\eta_1=0}$$

In our numerical experiments, iterating this process between 6 and 10 times typically yields a sufficiently accurate eigenvector $z$. Thus, the computational complexity remains at a manageable level of $\mathcal{O}(N)$, a constant multiple of the complexity for evaluating $f$ and its derivatives.

To enhance the computational efficiency of the framework outlined in equation (27), we propose two alternatives for calculating the conservative gradient $D^{\Lambda}$. In Theorem 1, we established that $x(t)$ remains bounded under the convergence condition specified in equation (4). This boundedness justifies the assumption that $\nabla f$ has a global Lipschitz constant, particularly since $\nabla^2 f(x)$ is continuous over the bounded set. One straightforward approach to estimating this constant is to use $\|\nabla^2 f(x_0)\|$, under the assumption that the local Lipschitz constant decreases as the iterations progress. This estimation method does not rely on the changing values of $x(t)$, thus simplifying the computational process. Additionally, under this assumption, the conservative gradient $D_x^p$ consistently equals zero.

Using a global Lipschitz constant for $\nabla f$ may limit the flexibility of our framework as represented in equation (27). As illustrated in Figure 2, the local Lipschitz constant can vary significantly during iterations. This variability is crucial for enhancing the performance of our algorithm, and our framework (27) is designed to capture such variations. The concept of $(L_0, L_1)$-smoothness, which has recently gained significant attention [64], offers a useful compromise. A function is defined as $(L_0, L_1)$-smooth if it satisfies the condition:

$$\|\nabla^2 f(x)\| \leq L_1 \|\nabla f(x)\| + L_0$$

This definition can be extended to include:

$$\|\nabla f(x) - \nabla f(y)\| \leq (L_1 \|\nabla f(x)\| + L_0)\|x - y\|, \quad \text{for all } \|y - x\| \leq 1$$

This class of functions encompasses classic $L$-smooth functions when $L_1 = 0$ and $L_0$ is the global Lipschitz constant of $\nabla f$. By defining $\Lambda(x, f) = L_1 \|\nabla f(x)\| + L_0$ and assuming $\nabla f$ is path differentiable, we obtain:

$$L_1 \frac{\nabla^2 f(x) J^{\nabla f}}{\|\nabla f(x)\|} \in D^{\Lambda}(x), \qquad \forall J^{\nabla f} \in D^{\nabla f}(x)$$

This approach leverages first-order information to obtain higher-order information efficiently. It significantly reduces the computational effort by simplifying the estimation of $\|\nabla f(x)\|$ to determining the upper bounds for constants $L_1$ and $L_0$.

# 5 Theoretical analysis

## 5.1 Proof of Theorem 1

This subsection explores the convergence properties of the trajectory of the ISHD. We give a proof of Theorem 1 by constructing a Lyapunov function. This function includes the terms $\|x(t) - x_\star\|^2$,

$\|t\beta(t)\nabla f(x(t))\|^2$, and $t\beta(t)\langle\nabla f(x(t)), x(t)-x_\star\rangle$, which are rarely seen in other Lyapunov function-based proofs. We add these terms to get a point-wise estimation of $\|\dot{x}(t)\|$ and $\|\nabla f(x(t))\|$.

*Proof.* Consider the Lyapunov function

$$\begin{aligned} E(t) =&\delta(t)\left(f(x(t)) - f_\star\right) + \frac{1}{2}\|\lambda(x(t)-x_\star) + t(\dot{x}(t) + \kappa\beta(t)\nabla f(x(t)))\|^2 \\ &+ \lambda(1-\kappa)t\beta(t)\langle\nabla f(x(t)), x(t)-x_\star\rangle + \frac{\kappa(1-\kappa)}{2}\|t\beta(t)\nabla f(x)\|^2 \\ &+ \frac{\lambda(\alpha-1-\lambda)}{2}\|x(t)-x_\star\|^2 \end{aligned} \tag{60}$$

We show that (60) is a decreasing function in the remaining part of this proof. Let $u(t) = \gamma(t) - \kappa\dot{\beta}(t) - \kappa\beta(t)/t$. Notice that

$$\begin{aligned} &\frac{\mathrm{d}}{\mathrm{d}t}\Big(\lambda(x(t)-x_\star) + t(\dot{x}(t) + \kappa\beta(t)\nabla f(x(t)))\Big) \\ &\qquad = (\lambda+1-\alpha)\dot{x}(t) - tu(t)\nabla f(x(t)) - (1-\kappa)t\beta(t)\nabla^2 f(x(t))\dot{x}(t) \end{aligned}$$

Plugging in the formula above and expanding the inner product give

$$\begin{aligned} &\frac{1}{2}\frac{\mathrm{d}}{\mathrm{d}t}\|\lambda(x(t)-x_\star) + t(\dot{x}(t) + \kappa\beta(t)\nabla f(x(t)))\|^2 \\ &\quad = -\big\langle(\alpha-1-\lambda)\dot{x}(t) + (1-\kappa)t\beta(t)\nabla^2 f(x(t))\dot{x}(t) + tu(t)\nabla f(x(t)), \\ &\qquad \lambda(x(t)-x_\star) + t(\dot{x}(t) + \kappa\beta(t)\nabla f(x(t)))\big\rangle \\ &\quad = -\lambda(\alpha-1-\lambda)\langle\dot{x}(t), x(t)-x_\star\rangle - \lambda(1-\kappa)t\beta(t)\langle\nabla^2 f(x(t))\dot{x}(t), x(t)-x_\star\rangle \\ &\qquad - \kappa(\alpha-1-\lambda)t\beta(t)\langle\nabla f(x(t)), \dot{x}(t)\rangle - \kappa t^2\beta(t)u(t)\|\nabla f(x(t))\|^2 \\ &\qquad - (1-\kappa)t^2\beta(t)\langle\nabla^2 f(x(t))\dot{x}(t), \dot{x}(t)\rangle - t^2u(t)\langle\nabla f(x(t)), \dot{x}(t)\rangle \\ &\qquad - \lambda tu(t)\langle\nabla f(x(t)), x(t)-x_\star\rangle - (\alpha-1-\lambda)t\|\dot{x}(t)\|^2 \\ &\qquad - \kappa(1-\kappa)t^2\beta(t)^2\langle\nabla^2 f(x(t))\dot{x}(t), \nabla f(x(t))\rangle \end{aligned}$$

Direct calculation gives $\frac{\lambda(\alpha-1-\lambda)}{2}\frac{\mathrm{d}}{\mathrm{d}t}\|x(t)-x_\star\|^2 = \lambda(\alpha-1-\lambda)\langle x(t)-x_\star, \dot{x}(t)\rangle$ and

$$\begin{aligned} &\frac{\kappa(1-\kappa)}{2}\frac{\mathrm{d}}{\mathrm{d}t}\|t\beta(t)\nabla f(x)\|^2 \\ &= \kappa(1-\kappa)\Big(t^2\beta(t)^2\langle\nabla^2 f(x(t))\dot{x}(t), \nabla f(x(t))\rangle + t\beta(t)(\beta(t) + t\dot{\beta}(t))\|\nabla f(x(t))\|^2\Big) \end{aligned}$$

We also have

$$\begin{aligned} &\lambda(1-\kappa)\frac{\mathrm{d}}{\mathrm{d}t}\left(t\beta(t)\langle\nabla f(x(t)), x(t)-x_\star\rangle\right) \\ &\quad = \lambda(1-\kappa)t\beta(t)\langle\nabla f(x(t)), \dot{x}(t)\rangle + \lambda(1-\kappa)t\beta(t)\langle\nabla^2 f(x(t))\dot{x}(t), x(t)-x_\star\rangle \\ &\qquad + \lambda(1-\kappa)(\beta(t) + t\dot{\beta}(t))\langle\nabla f(x(t)), x(t)-x_\star\rangle \end{aligned}$$

Using these results and combining similar terms yield

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}E(t) =&\dot{\delta}(t)(f(x(t)) - f_\star) - \lambda tw(t)\langle\nabla f(x(t)), x(t)-x_\star\rangle - (\alpha-1-\lambda)t\|\dot{x}(t)\|^2 \\ &+ \Big(\delta(t) - \big(t^2u(t) + (\kappa(\alpha-1-\lambda) - \lambda(1-\kappa))t\beta(t)\big)\Big)\langle\nabla f(x(t)), \dot{x}(t)\rangle \\ &- \kappa t^2\beta(t)w(t)\|\nabla f(x(t))\|^2 - (1-\kappa)t^2\beta(t)\langle\nabla^2 f(x(t))\dot{x}(t), \dot{x}(t)\rangle \end{aligned}$$

According to (4) and $\nabla^2 f(x(t)) \succeq 0$, we have $\dot{E}(t) \leq 0$ and further

$$\begin{aligned}\dot{E}(t) + (\lambda t w(t) - \dot{\delta}(t))(f(x(t)) - f_\star) + (\alpha - 1 - \lambda)t\|\dot{x}(t)\|^2 \\ + (1-\kappa)t^2\beta(t)\langle\nabla^2 f(x(t))\dot{x}(t), \dot{x}(t)\rangle + \kappa t^2\beta(t)w(t)\|\nabla f(x(t))\|^2 \leq 0\end{aligned}$$

Integrating the inequality above from $t_0$ to $t$ gives

$$\begin{aligned}&E(t) + \int_{t_0}^{t} (\lambda s w(s) - \dot{\delta}(s))(f(x(s)) - f_\star)\,\mathrm{d}s + \int_{t_0}^{t} \kappa s^2\beta(s)w(s)\|\nabla f(x(s))\|^2\,\mathrm{d}s \\ &+ \int_{t_0}^{t} (\alpha - 1 - \lambda)s\|\dot{x}(s)\|^2\,\mathrm{d}s + \int_{t_0}^{t} (1-\kappa)s^2\beta(s)\langle\nabla^2 f(x(s))\dot{x}(s), \dot{x}(s)\rangle\,\mathrm{d}s \leq E(t_0)\end{aligned} \tag{61}$$

The nonnegativity of $E(t)$ and the arbitrary of $t$ imply (6), (7), (8), and (9). Plugging in the expression (60) to (61) gives the first two terms of (5) and $\|x(t) - x_\star\| \leq \sqrt{2E(t_0)}/\sqrt{\lambda(\alpha - 1 - \lambda)}$. A direct calculation gives

$$\|\dot{x}(t)\| \leq \frac{1}{t}\left(\sqrt{2E(t_0)} + \lambda\|x(t) - x_0\|\right) + \kappa\beta(t)\|\nabla f(x(t))\| \leq \mathcal{O}\left(\frac{1}{t}\right)$$

Therefore, $x(t)$ is bounded and the third term in (5) are proved, which completes our proof. □ □

Theorem 1 ensures that the stopping time $T$ is well-defined when $\beta$ is lower bounded. It also enlightens a way to analyze the sequence $\{x_k\}_{k=0}^{\infty}$ generated by (12) with the help of the continuous-time trajectory, as we will show in the next subsection.

### 5.2 Proof of Theorem 2: Estimating the summation of the local truncated errors

To measure the deviation introduced at step $k$ and the cumulative deviation at step $k$ between the discretization $x_k$ and the continuous-time trajectory $x(t_k)$, we introduce two quantities before proceeding to the analysis.

**Definition 6** (Local Truncated Error)**.** *Given the coefficients $\alpha, \beta$, and $\gamma$, the local truncated error of* (12) *associated with a differentiable function $f$ using a stepsize $h$ is*

$$\varphi(t) = \begin{pmatrix} x(t+h) - x(t) \\ v(t+h) - v(t) \end{pmatrix} - h\begin{pmatrix} v(t) - \beta(t)\nabla f(x(t)) \\ -\frac{\alpha}{t}v(t) + \left(\frac{\alpha}{t}\beta(t) + \dot{\beta}(t) - \gamma(t)\right)\nabla f(x(t)) \end{pmatrix} \tag{62}$$

The local truncated error computes the difference between the exact solution of the differential equation at time $t + h$ and the approximate solution obtained using the explicit Euler method starting at time $t$. As pointed by its name, it only contains the information obtained at one step. To study how them accumulate in the proceeding iterations, we introduce the global truncated error, the difference between the exact solution of (11) and the approximate solution obtained using (12).

**Definition 7** (Global Truncated Error)**.** *Let $x(t_0) = x_0, v(t_0) = v_0$. Given the differentiable function $f$, the coefficients $\alpha, \beta$, $\gamma$, and the stepsize $h$, if $x(t)$ and $x_k$ are the solution of* (11) *and* (12)*, respectively, the vector*

$$e_k = \begin{pmatrix} r_k \\ s_k \end{pmatrix} = \begin{pmatrix} x(t_k) - x_k \\ v(t_k) - v_k \end{pmatrix} \tag{63}$$

*is the global truncated error at time $t_k$.*

Before proceeding, we present a lemma derived from the conditions (17) and (18), which provides an estimate for the growth rate of $\beta(t)$.

**Lemma 3** (Growth rate of $\beta(t)$)**.** *Under the Assumption 1 and given that the inequalities in* (17) *and* (18) *are satisfied, we have*

$$\beta(t)\|\nabla^2 f(x(t))\| \leq 4/h \tag{64}$$

*Proof.* The non-negativity of $\alpha, \beta(\cdot)$, and $t$ ensures that $\gamma(t)-\dot{\beta}(t)-\alpha\beta(t)/t \leq \gamma(t)-\dot{\beta}(t)$. Therefore, combining (17) and (18) leads to

$$\beta(t)\sqrt{\Lambda(f, X(t, \Xi, f))} \leq 2\sqrt{\gamma(t) - \dot{\beta}(t)} \leq 2\sqrt{\beta(t)/h}$$

Squaring both sides completes this proof. □ □

We are now at the position to present the analysis using a two-step procedure. We begin our analysis of $\{x_k\}_{k=0}^{\infty}$ by investigating the local truncated error. In the first step, we provide an enhanced estimation of the sequence $\{\|\varphi(t_k)\|\}_{k=0}^{\infty}$ using Theorem 1. A byproduct of this estimation shows that under some mild conditions the local truncated error vanishing with $\mathcal{O}(1/t)$ rate. In the second step, we show that the global truncated error inherits the convergence properties of the local truncated error under some mild conditions.

**Lemma 4** (Estimating the Summation of the Local Truncated Error)**.** *Assume the conditions* (16)*,* (17)*, and* (18) *hold. Using the notations in Theorem 1 and the energy function defined in* (60)*, we set*

$$M_1 = \max\{1 + (\alpha+1)/t_0, C_2/h + (1+\alpha/t_0)(1/h + C_1), \alpha/t_0 + 1/h + 1\}, \tag{65}$$

$$M_2 = \max\left\{\sqrt{\frac{E(t_0)}{2(\alpha-1)(\alpha-1-\lambda)t_0}}, \sqrt{\frac{C_3 E(t_0)}{\kappa(2\alpha-1)}}, \sqrt{\frac{4E(t_0)}{(1-\kappa)(2\alpha-1)h}}\right\}. \tag{66}$$

*Let* $t_k = t_0 + (k-1)h$ *and* $M_3 = M_1 M_2$*. Then,*

$$\|\varphi(t)\| \leq o(1/t) \quad \textit{and} \quad \sum_{k=0}^{n} t_k^{\alpha}\|\varphi(t_k)\| \leq M_3 t_n^{\alpha-1/2} \tag{67}$$

*Proof.* In order to get (67), we first bound the local truncated error as follows:

$$\|\varphi(t)\| \leq \int_t^{t+h}\int_t^s (\|\ddot{x}(\tau)\| + \|\ddot{v}(\tau)\|)\,\mathrm{d}\tau\,\mathrm{d}s \leq h\int_t^{t+h}(\|\ddot{x}(\tau)\| + \|\ddot{v}(\tau)\|)\,\mathrm{d}\tau \tag{68}$$

Using (1), the second term in (17), and the condition (16), we have the following estimation:

$$\begin{aligned}\int_t^{t+h}\|\ddot{x}(\tau)\|\,\mathrm{d}\tau \leq & \frac{\alpha}{t}\int_t^{t+h}\|\dot{x}(\tau)\|\,\mathrm{d}\tau + \int_t^{t+h}\beta(\tau)\|\nabla^2 f(x(\tau))\dot{x}(\tau)\|\,\mathrm{d}\tau \\ & + \left(\frac{1}{h} + C_1\right)\int_t^{t+h}\|\beta(\tau)\nabla f(x(\tau))\|\,\mathrm{d}\tau\end{aligned} \tag{69}$$

Invoking (12), we obtain

$$\ddot{v}(t) = \alpha\dot{x}(t)/t^2 - \alpha\ddot{x}(t)/t - (\gamma(t) - \dot{\beta}(t))\nabla^2 f(x(t))\dot{x}(t) - (\dot{\gamma}(t) - \ddot{\beta}(t))\nabla f(x(t)) \tag{70}$$

Using (69) and (70) yields

$$\begin{aligned}\int_t^{t+h} \|\ddot{v}(\tau)\| \,\mathrm{d}\tau \le \int_t^{t+h} \Big(&\frac{\alpha}{\tau}\|\ddot{x}(\tau)\| + \frac{\alpha}{\tau^2}\|\dot{x}(\tau)\| + |\dot{\gamma}(\tau) - \ddot{\beta}(\tau)|\|\nabla f(x(\tau))\| \\ &+ (\gamma(\tau) - \dot{\beta}(\tau))\|\nabla^2 f(x(\tau))\dot{x}(\tau)\|\Big) \,\mathrm{d}\tau \\ \le &\left(\frac{C_2}{h} + \frac{\alpha}{t}\left(\frac{1}{h} + C_1\right)\right)\int_t^{t+h} \|\beta(\tau)\nabla f(x(\tau))\| \,\mathrm{d}\tau \\ &+ \left(\frac{\alpha}{t} + \frac{1}{h}\right)\int_t^{t+h} \beta(\tau)\|\nabla^2 f(x(\tau))\dot{x}(\tau)\| \,\mathrm{d}\tau \\ &+ \frac{\alpha(\alpha+1)}{t^2}\int_t^{t+h} \|\dot{x}(\tau)\| \,\mathrm{d}\tau\end{aligned} \tag{71}$$

Combining (65), (68), (69), and (71) gives

$$\|\varphi(t)\| \le M_1 \int_t^{t+h} \left(\frac{\alpha}{t}\|\dot{x}(\tau)\| + \|\beta(\tau)\nabla f(x(\tau))\| + \beta(\tau)\|\nabla^2 f(x(\tau))\dot{x}(\tau)\|\right) \,\mathrm{d}\tau \tag{72}$$

The inequalities (8) and (9) imply

$$\sqrt{\beta(t)w(t)}\|\nabla f(x(t))\| = o\left(\frac{1}{t}\right), \quad \|\sqrt{\beta(t)\nabla^2 f(x(t))}\dot{x}(t)\| = o\left(\frac{1}{t}\right)$$

Using Lemma 3, the third inequality in (5), and the third inequality in (16), we get $\|\varphi(t)\| \le o(1/t)$, which is the first inequality in (67).

For the second inequality in (67), we transform the summation in (67) to integral by

$$\begin{aligned}&\sum_{k=0}^{n} t_k^{\alpha}\|\varphi(t_k)\| \\ &\le M_1 \sum_{k=0}^{n} t_k^{\alpha} \int_{t_k}^{t_{k+1}} \left(\frac{1}{t_k}\|\dot{x}(\tau)\| + \|\beta(\tau)\nabla f(x(\tau))\| + \beta(\tau)\|\nabla^2 f(x(\tau))\dot{x}(\tau)\|\right) \,\mathrm{d}\tau \\ &\le M_1 \int_{t_0}^{t_{n+1}} \left(\tau^{\alpha-1}\|\dot{x}(\tau)\| + \tau^{\alpha}\|\beta(\tau)\nabla f(x(\tau))\| + \tau^{\alpha}\beta(\tau)\|\nabla^2 f(x(\tau))\dot{x}(\tau)\|\right) \,\mathrm{d}\tau\end{aligned}$$

Applying the Cauchy inequality with (61) gives

$$\begin{aligned}&\int_{t_0}^{t_{n+1}} \tau^{\alpha-1}\|\dot{x}(\tau)\| \,\mathrm{d}\tau \\ &\qquad \le \sqrt{\int_{t_0}^{t_{n+1}} \tau^{2\alpha-3} \,\mathrm{d}\tau \int_{t_0}^{t_{n+1}} \tau\|\dot{x}(\tau)\|^2 \,\mathrm{d}\tau} \le \sqrt{\frac{E(t_0)}{2(\alpha-1)(\alpha-1-\lambda)}} t_{n+1}^{\alpha-1}\end{aligned} \tag{73}$$

Lemma 3 and the equation (61) imply

$$\begin{aligned}&\int_{t_0}^{t_{n+1}} \tau^{\alpha}\|\beta(\tau)\nabla f(x(\tau))\| \,\mathrm{d}\tau \\ &\qquad \le \sqrt{\int_{t_0}^{t_{n+1}} \tau^{2\alpha-2} \,\mathrm{d}\tau \int_{t_0}^{t_{n+1}} \tau^2\|\beta(\tau)\nabla f(x(\tau))\|^2 \,\mathrm{d}\tau} \le \sqrt{\frac{C_3 E(t_0)}{\kappa(2\alpha-1)}} t_{n+1}^{\alpha-1/2}\end{aligned} \tag{74}$$

Finally, using Lemma 3, $\beta(t) \leq C_3 w(t)$, and (61), we have

$$\begin{aligned}
&\int_{t_0}^{t_{k+1}} \tau^{\alpha}\beta(\tau)\|\nabla^2 f(x(\tau))\dot{x}(\tau)\|\,\mathrm{d}\tau \\
&\qquad \leq \sqrt{\frac{4}{h}\int_{t_0}^{t_{n+1}} \tau^{2\alpha-2}\,\mathrm{d}\tau \int_{t_0}^{t_{n+1}} \tau^2\beta(\tau)\|\dot{x}(\tau)\|^2_{\nabla^2 f(x(\tau))}\,\mathrm{d}\tau} \\
&\qquad \leq \sqrt{\frac{4E(t_0)}{(1-\kappa)(2\alpha-1)h}}\, t_{n+1}^{\alpha-\frac{1}{2}}
\end{aligned} \tag{75}$$

Combining (66), (73), (74), and (75) yields (67). □ □

### 5.3 Proof of Theorem 2

This subsection gives the convergence analysis of Algorithm 1. As the discretization scheme (12) is repeatedly applied, the local truncation errors $\{\|\varphi(t_k)\|\}_{k=0}^{\infty}$ from each step accumulate and interact in a complex manner, causing the global error to grow. The contraction factor helps characterize how the local errors propagate and interact.

**Definition 8** (Contraction Factor)**.** *Let $G \in \mathbb{S}_+^n$ be an approximation matrix of $\nabla^2 f(x(t))$. Given the coefficients $\alpha, \beta$, and $\gamma$, the contraction factor for scheme* (12) *associated with at time $t$ is*

$$\rho(t, G) := \left\| \begin{pmatrix} I - h\beta(t)G & hI \\ (\alpha\beta(t)/t + \dot{\beta}(t) - \gamma(t))G & (1 - \alpha h/t)I \end{pmatrix} \right\| \tag{76}$$

The contraction factor establishes a relationship between the amplification of global errors and the magnitude of local errors through a process of linearization. A smaller contraction factor leads to a more gradual increase in global errors, as it results in a stronger damping of local errors during their propagation. This argument is demonstrated in sec. 6.5. Following this, we present an explicit formula for calculating the contraction factor, grounded in the following lemma.

**Lemma 5** (Root Location)**.** *Given $\mu, \nu \in \mathbb{R}$ and $0 < \varrho < 1$, the $\varrho$-strong root condition is formulated for the roots of the equation*

$$r^2 + \mu r + \nu = 0 \tag{77}$$

*to lie within $|r| \leq \varrho$. The condition is expressed as:*

$$\nu \leq \varrho^2, \quad \varrho\mu - \varrho^2 \leq \nu, \quad -\varrho\mu - \varrho^2 \leq \nu \tag{78}$$

*Proof.* Based on the properties of Schur polynomials [37, Lemma 1.5], the necessary and sufficient condition for the roots (which can be complex) of (77) to lie within the unit circle is given by

$$\nu \leq 1, \quad \nu \geq \mu - 1, \quad \nu \geq -\mu - 1 \tag{79}$$

Ensuring that the roots lie within the circle $|r| \leq \varrho$ is equivalent to the roots of $\widetilde{r}^2 + \frac{\mu}{\varrho}\widetilde{r} + \frac{\nu}{\varrho^2} = 0$ being within the unit circle, where $\widetilde{r} = \frac{r}{\varrho}$. Applying (79) to this condition yields

$$\nu \leq \varrho^2, \quad \varrho\mu - \varrho^2 \leq \nu, \quad -\varrho\mu - \varrho^2 \leq \nu \tag{80}$$

which is the desired result. □ □

As an application of Lemma 5, we give the explicit formula for the contraction factor.

**Proposition 7.** *Given the matrix $G \in \mathbb{S}_+^n$, coefficients $\alpha, \beta(\cdot)$, and $\gamma(\cdot)$ in (76), we denote the eigenvalues of $G$ as $\lambda_1 \geq \ldots \geq \lambda_n \geq 0$ and introduce*

$$A(t,L) = h^2(\gamma(t) - \dot{\beta}(t))L,\ B(t,L) = \frac{h}{2}(\beta(t)L + \alpha/t),\ C(t,L) = B(t,L)^2 - A(t,L) \tag{81}$$

*Assuming the condition (17) holds, the contraction factor is*

$$\begin{aligned}\rho(t,G) = \max\{&\sqrt{[1-2B(t,\lambda_n)+A(t,\lambda_n)]_+}, B(t,\lambda_1) - 1 + \sqrt{[C(t,\lambda_1)]_+},\\ &1 - B(t,\lambda_1) + \sqrt{[C(t,\lambda_1)]_+}\}\end{aligned} \tag{82}$$

*Proof.* Since $G$ is symmetric semi-positive definite, it has the decomposition $G = Q\Lambda Q^\top$, where $Q$ is orthogonal and $\Lambda$ is a non-negative diagonal matrix. Using a block diagonal matrix $\mathrm{Diag}(Q,Q)$ to perform the similar transform to $G$, it is sufficient to consider $\mathrm{Diag}(G_1,\ldots,G_n)$ with

$$G_i = \begin{pmatrix} 1 - h\beta(t)\lambda_i & h \\ (\alpha\beta(t)/t + \dot{\beta}(t) - \gamma(t))h\lambda_i & 1 - \alpha h/t \end{pmatrix} \tag{83}$$

This fact can be derived by switching the columns and rows to form a block diagonal matrix with $2 \times 2$ sub-matrices.

The characteristic equation of $G_i$ is

$$\lambda^2 - (2 - 2B(t,\lambda_i))\,\lambda + 1 - 2B(t,\lambda_i) + A(t,\lambda_i) = 0 \tag{84}$$

Denote the spectral radius of (84) as $\rho_i$. Using Lemma 5 and elementary algebra, $\rho_i$ must satisfies

$$1 - 2B(t,\lambda_i) + A(t,\lambda_i) \leq \rho_i^2, \tag{85}$$

$$C(t,\lambda_i) \leq (1 + \rho_i - B(t,\lambda_i))^2, \tag{86}$$

$$C(t,\lambda_i) \leq (1 - \rho_i - B(t,\lambda_i))^2. \tag{87}$$

Notice that

$$1 - 2B(t,\lambda_i) + A(t,\lambda_i) + C(t,\lambda_i) = (1 - B(t,\lambda_i))^2 \tag{88}$$

When $C(t,\lambda_i) < 0$, only (85) takes effect. We have $1 - 2B(t,\lambda_i) + A(t,\lambda_i) > 0$ and

$$\sqrt{1 - 2B(t,\lambda_i) + A(t,\lambda_i)} \geq |1 - B(t,\lambda_i)| \geq \max\{1 - B(t,\lambda_i), B(t,\lambda_i) - 1\} \tag{89}$$

When $C(t,\lambda_i) \geq 0$, we have

$$\sqrt{1 - 2B(t,\lambda_i) + A(t,\lambda_i)} \leq |1 - B(t,\lambda_i)| \leq \max\{1 - B(t,\lambda_i), B(t,\lambda_i) - 1\} \tag{90}$$

This inequality excludes the possibility of $\rho_i \leq 1 - B(t,L) - \sqrt{[C(t,L)]_+}$ and $\rho_i \leq B(t,L) - 1 - \sqrt{[C(t,L)]_+}$. Combining the equations (89) and (90), the spectral radius of $G_i$ is

$$\begin{aligned}\rho_i = \max\{&\sqrt{[1-2B(t,\lambda_i)+A(t,\lambda_i)]_+}, B(t,\lambda_i) - 1 + \sqrt{[C(t,\lambda_i)]_+},\\ &1 - B(t,\lambda_i) + \sqrt{[C(t,\lambda_i)]_+}\}\end{aligned} \tag{91}$$

Since $\gamma(t)-\dot\beta(t)\le\beta(t)/h$, it is easy to verify that $\sqrt{[1-2B(t,L)+A(t,L)]_+}$ is an decreasing function with respect to $L$. Meanwhile, the derivative of $B(t,L)-\sqrt{C(t,L)}$ with respect to $L$ is

$$\begin{aligned}
&\frac{h}{2}\bigg(\beta(t)-\frac{2\beta(t)(\beta(t)L+\alpha/t)-4(\gamma(t)-\dot\beta(t))}{2\sqrt{\frac{4}{h^2}C(t,L)}}\bigg)\\
&=\frac{h\beta(t)}{2\sqrt{\frac{4}{h^2}C(t,L)}}\bigg(2\sqrt{(\beta(t)L+\alpha/t)^2-4(\gamma(t)-\dot\beta(t))L}\\
&\qquad\qquad-\Big(2(\beta(t)L+\alpha/t)-4(\gamma(t)-\dot\beta(t))/\beta(t)\Big)\bigg)
\end{aligned}$$

Elementary calculation gives

$$\begin{aligned}
&4(\beta(t)L+\alpha/t)^2-16(\gamma(t)-\dot\beta(t))L-4\Big((\beta(t)L+\alpha/t)-2(\gamma(t)-\dot\beta(t))/\beta(t)\Big)^2\\
&=-16(\gamma(t)-\dot\beta(t))L+16\frac{(\gamma(t)-\dot\beta(t))}{\beta(t)}(\beta(t)L+\alpha/t)-16\frac{(\gamma(t)-\dot\beta(t))^2}{\beta(t)^2}\\
&=16\frac{(\gamma(t)-\dot\beta(t))}{\beta(t)^2}\Big(\frac{\alpha}{t}\beta(t)-(\gamma(t)-\dot\beta(t))\Big)\le 0
\end{aligned}$$

We have $\frac{\mathrm{d}}{\mathrm{d}L}\Big(B(t,L)-\sqrt{C(t,L)}\Big)\le 0$. Considering the following relationships:

$$\begin{aligned}
&B(t,L)-1+\sqrt{[C(t,L)]_+}=2B(t,L)-1-(B(t,L)-\sqrt{C(t,L)}),\\
&1-B(t,L)+\sqrt{[C(t,L)]_+}=1-(B(t,L)-\sqrt{C(t,L)})
\end{aligned}$$

we observe that the functions $B(t,L)-1+\sqrt{[C(t,L)]_+}$ and $1-B(t,L)+\sqrt{[C(t,L)]_+}$ are increasing. By using $\rho(t,G)=\max_i\rho_i$ and the monotonicity of each component in (91), we derive the formula (82). □ □

A useful estimation of the contraction factor can be obtained through a more detailed investigation of Proposition 7. We use the same notations as Proposition 7 in the following corollary.

**Corollary 4.** *Given the condition* (17)*, if the matrix* $G\in\mathbb{S}_+^n$ *satisfies*

$$\beta(t)\sqrt{\|G\|}\le\sqrt{\gamma(t)-\dot\beta(t)}+\sqrt{\gamma(t)-\dot\beta(t)-\frac{\alpha}{t}\beta(t)} \tag{92}$$

*then* $\rho(t,G)\le 1-\alpha h/(2t)$.

*Proof.* The roots of $C(t,L)=0$ are

$$\begin{aligned}
L_\pm&=\frac{-\Big(\frac{\alpha}{t}\beta(t)-2(\gamma(t)-\dot\beta(t))\Big)\pm 2\sqrt{(\gamma(t)-\dot\beta(t))^2-\frac{\alpha}{t}\beta(t)(\gamma(t)-\dot\beta(t))}}{\beta(t)^2}\\
&=\left(\frac{\sqrt{\gamma(t)-\dot\beta(t)}\pm\sqrt{\gamma(t)-\dot\beta(t)-\frac{\alpha}{t}\beta(t)}}{\beta(t)}\right)^2
\end{aligned}$$

When $L_- \le \lambda_1 \le L_+$, we know $C(t,\lambda_1) \le 0$. Hence, the formula (89) implies

$$\begin{aligned} 1 - B(t,\lambda_1) + \sqrt{[C(t,\lambda_1)]_+} &\le \sqrt{1 - 2B(t,\lambda_1) + A(t,\lambda_1)} \\ &\le \sqrt{1 - 2B(t,\lambda_n) + A(t,\lambda_n)} \end{aligned}$$

When $L \le L_-$, we have $C(t,L) \ge 0$. The monotonicity of $B(t,L) - \sqrt{C(t,L)}$ implies

$$\begin{aligned} B(t,L) - 1 + \sqrt{C(t,L)} &\le B(t,L_-) - 1 + \sqrt{C(t,L_-)} \\ &= \sqrt{1 - 2B(t,L_-) + A(t,L_-)} \\ &\le \sqrt{1 - 2B(t,\lambda_n) + A(t,\lambda_n)} \end{aligned}$$

where the last inequality comes from $\lambda_n \le \lambda_1 \le L_-$ and the monotonicity of $\sqrt{1 - 2B(t,L) + A(t,L)}$.

Combining these results, we know $\lambda_1 \le L_+$ ensures that

$$\rho(t,G) \le \sqrt{1 - 2B(t,\lambda_n) + A(t,\lambda_n)} \le 1 - \frac{\alpha h}{2t}$$

The second inequality comes from $0 \le \lambda_n$, $\gamma(t) - \dot{\beta}(t) \le \beta(t)/h$, and $\sqrt{1-\alpha h/t} \le 1 - \alpha h/2t$. Simplifying $\lambda_1 \le L_+$ gives the equation (92). □ □

Next, we give the proof of Theorem 2 by analyzing the propagation of the global truncated error.

*Proof.* To investigate the global truncated error, we consider the relationship between it and the local truncated errors. Subtracting (62) with (12) at time $t_k$ and using the definition (63) yield

$$\begin{pmatrix} r_{k+1} \\ s_{k+1} \end{pmatrix} = \begin{pmatrix} I - h\beta(t_k)G & hI \\ (\alpha\beta(t_k)/t_k + \dot{\beta}(t_k) - \gamma(t_k))G & (1 - \alpha h/t_k)I \end{pmatrix} \begin{pmatrix} r_k \\ s_k \end{pmatrix} + h\varphi(t_k) \tag{93}$$

where $G = \int_0^1 \nabla^2 f(x(t_k) + \tau r_k)\,\mathrm{d}\tau$. Our proof is based on the error propagation equation (93). Denote $\rho(t_k) = \rho(t_k, G)$ with $G = \int_0^1 \nabla^2 f(x(t_k) + \tau r_k)\,\mathrm{d}\tau$. We enhance the theorem and prove the following conclusions for each $k$:

$$\rho(t_k) \le 1 - \frac{\alpha h}{2t_k}, \quad \|e_k\| \le \frac{M_3}{\sqrt{t_k}}, \quad \text{and} \quad f(x_k) - f_\star \le \mathcal{O}\left(\frac{1}{k}\right) \tag{94}$$

The constant $M_3$ come from Lemma 4. To facilitate our presentation, we designate the first inequality in (94) with index $k$ as $H_0(k)$, and the second as $H_1(k)$. We employ a mathematical induction on $k$ to demonstrate the theorem.

We begin by confirming the base case when $k = 0$. The restriction on $\alpha, \beta(t_0)$, and $\gamma(t_0)$ ensures $\rho(t_0) \le 1 - \alpha h/(2t_0)$. Since, $x(t_0) = x_0, v(t_0) = v_0$, we have $\|e_0\| = 0$. These results guarantee that both propositions $H_0(0)$ and $H_1(0)$ are satisfied.

Assume as our induction hypothesis that both $H_0(l)$ and $H_1(l)$ are valid for all indices less than or equal to $k$. We will now prove that they remain true for the index $k+1$. For $k \le K$, a useful estimation of $\prod_{l=k}^K \rho(t_l)$ is

$$\begin{aligned} \prod_{l=k}^{K} \rho(t_l) &= \exp\left(\sum_{l=k}^{K} \log\left(\rho(t_l) - 1 + 1\right)\right) \le \exp\left(\sum_{l=k}^{K} (\rho(t_l) - 1)\right) \\ &\le \exp\left(-\sum_{l=k}^{K} \frac{\alpha h}{2t_l}\right) \le \exp\left(-\frac{\alpha}{2}\int_{t_k}^{t_{K+1}} \frac{1}{t}\,\mathrm{d}t\right) = \left(\frac{t_k}{t_{K+1}}\right)^{\alpha/2} \end{aligned} \tag{95}$$

We first prove that $H_1(k+1)$ holds. Applying (93) successively, we have

$$\begin{aligned}\|e_{n+1}\| \leq & \left(1-\frac{\alpha h}{2t_k}\right)\|e_k\| + \|\varphi(t_k)\| \\ \leq & \prod_{k=0}^{n}(1-\rho(t_k))\|e_0\| + \|\varphi(t_n)\| + \sum_{k=0}^{n-1}\prod_{j=k+1}^{n}(1-\rho(t_j))\|\varphi(t_k)\| \\ \leq & \sum_{k=0}^{n}\left(\frac{t_{k+1}}{t_{n+1}}\right)^{\alpha/2}\|\varphi(t_k)\| \leq M_3\frac{1}{\sqrt{t_{n+1}}}\end{aligned}$$

The last inequality are derived from Lemma 4.

For $H_0(k+1)$, we note that the condition (18) ensures that

$$\begin{aligned}\beta(t)\sqrt{\left\|\int_0^1 \nabla^2 f(x(t_k)+\tau r_k)\,\mathrm{d}\tau\right\|} &\leq \beta(t)\sqrt{\int_0^1 \Lambda(X(t_k,\Xi,f)+\tau\overline{x}(t_k,\Xi,f),f)\,\mathrm{d}\tau} \\ &\leq \sqrt{\gamma(t)-\dot{\beta}(t)} + \sqrt{\gamma(t)-\dot{\beta}(t)-\frac{\alpha}{t}\beta(t)}\end{aligned}$$

Using Corollary 4, we get $\rho(t_{n+1}) \leq 1/(2t_{n+1})$.

Combining these results, we know $H_0(k)$ and $H_1(k)$ hold for all $k$. The function value minimization rate is

$$\begin{aligned}f(x_k)-f_\star \leq & |f(x_k)-f(x(t_k))| + |f(x(t_k))-f_\star| \\ \leq & \|\nabla f(x(t_k))\|\|e_k\| + \frac{1}{2}\left\|\int_0^1 \nabla^2 f(x(t_k)+\tau e_k)\,\mathrm{d}\tau\right\|\|e_k\|^2 + \frac{E(t_0)}{t_k^2 w(t_k)}\end{aligned} \tag{96}$$

which means the convergence rate is at least $\mathcal{O}(1/k)$. □ □

Using the same assumptions and notations as in Theorem 2, we present a simplified version of the stability condition under the Lipschitz continuity of the Hessian matrix.

**Corollary 5.** *Suppose the Hessian matrix satisfies the Lipschitz continuity condition*

$$\|\nabla^2 f(x)-\nabla^2 f(y)\| \leq L_H\|x-y\|$$

*Using the same notations and conditions as in Theorem 2, but replacing the condition* (18) *with the simplified condition* (19)*, we obtain that* $f(x_k)-f_\star \leq \mathcal{O}(1/k)$.

*Proof.* We use the same arguments as in the proof of Theorem 2. The parts besides $H_0(k+1)$ can be proven verbosely as Theorem 2. For $H_0(k+1)$, assuming $H_1(k)$ holds, the Hessian continuity and the condition (19) implies

$$\begin{aligned}&\beta(t)\sqrt{\left\|\int_0^1 \nabla^2 f(x(t_k)+\tau r_k)\,\mathrm{d}\tau\right\|} \\ &\leq \beta(t)\sqrt{\nabla^2 f(x(t_k)) + \frac{L_T}{2}\|r_k\|} \leq \beta(t)\sqrt{\Lambda(X(t_k),\Xi,f) + \frac{L_T M_3}{2\sqrt{t_k}}} \\ &\leq \sqrt{\gamma(t)-\dot{\beta}(t)} + \sqrt{\gamma(t)-\dot{\beta}(t)-\frac{\alpha}{t}\beta(t)}\end{aligned}$$

The second inequality follows from $H_1(k)$. Corollary 4 establishes $H_0(k+1)$. □ □

The convergence rate given in Theorem 2 is only $\mathcal{O}(1/k)$ rather than the accelerated rate $\mathcal{O}(1/k^2)$. The reason is that our technique in the proof makes the discretization error accumulates and eventually gives a slow convergence rate. However, the flexibility of the coefficients make it suitable for deriving a large range of optimization methods. We also show that the empirical performance of (12) is comparable to and can even be better than the discretized scheme with $\mathcal{O}(1/k^2)$ rate in theory after some tuning of the coefficients.

### 5.4 Convergence analysis of Algorithm 2

Before presenting the main result, we first give several useful definitions.

**Definition 9** (Clarke directional derivative)**.** *Let $\varkappa$ be Lipschitz continuous near $\overline{\theta}$. The Clarke directional derivative of $\varkappa$ at $\overline{\theta}$ along a nonzero vector $\vartheta$ is given by*

$$\varkappa^{\circ}(\overline{\theta};\vartheta) \triangleq \limsup_{\substack{\theta\to\overline{\theta}\\ \tau\downarrow 0}} \frac{\varkappa(\theta+\tau\vartheta)-\varkappa(\theta)}{\tau} \tag{97}$$

Based on Definition 9, we define the Clarke subdifferential.

**Definition 10** (Clarke subdifferential)**.** *For a function $\varkappa$ that is Lipschitz continuous near $\theta$, the Clarke subdifferential of $\varkappa$ at $\theta$ is given by*

$$\partial\varkappa(\theta) \triangleq \left\{a\in\mathbb{R}^{d_\theta} : \varkappa^{\circ}(\theta;\vartheta)\geq a^{\top}\vartheta, \quad \forall\vartheta\in\mathbb{R}^{d_\theta}\right\}$$

Another definition of the Clarke subdifferential is

$$\partial\varkappa(\theta) = \operatorname{co}\left\{\vartheta\in\mathbb{R}^{d_\theta} : \exists\,\{\theta_n\}_{n\in\mathbb{N}}\subset\operatorname{dom}\varkappa,\ \text{s.t.}\ \ (\theta_n,\nabla\varkappa(\theta_n))\to(\theta,\vartheta)\right\}$$

For the proof of the equivalence, one may refer to [23, Theorem 2.5.1]. Using the Clarke subdifferential, we define the Clarke stationarity. Besides, we also define the $D$-stationarity using conservative gradient.

**Definition 11** (Clarke stationary point)**.** *$\theta$ is a Clarke stationary point of $\varkappa$ if $0\in\partial\varkappa(\theta)$.*

**Definition 12** ($D$-stationary point)**.** *$\theta$ is a $D$-stationary point of $\varkappa$ if $0\in D^{\varkappa}(\theta)$.*

We apply Definition 9 and get the following criterion for Clarke stationary point based on Clarke directional derivative.

**Proposition 8** (Criterion of Clarke stationary point)**.** *$\theta$ is a Clarke stationary point of $\varkappa$ if and only if $\varkappa^{\circ}(\theta;\vartheta)\geq 0$ for all $\vartheta\in\mathbb{R}^{d_\theta}$.*

After addressing these preliminaries, we begin our proof, which is divided into two parts. In the first part, we establish a condition that ensures the feasibility of each stationary point of the penalty function (28). Within the context of the probability space for parameterized functions $(\mathcal{F},\mathcal{H}(\mathscr{A}),\mathbb{P}_{\mathcal{H}})$, we define the residual function as:

$$R(\theta) = \mathbb{E}_f\left[P\left(X(T(\theta,f),\theta,T(\theta,f),f)\right)\right] + \mathbb{E}_f\left[Q(\theta,T(\theta,f))\right]$$

This function quantifies the violation of constraints. The feasible set is defined by the equation:

$$S = \{\theta \mid R(\theta)\leq 0\} \tag{98}$$

With these notations, we proceed to introduce the sufficient decrease condition based on Clarke directional derivatives.

**Assumption 2** (Uniform sufficient decrease condition)**.** *For each infeasible point $\theta \in \mathbb{R}^{d_\theta}$, i.e. $\theta \notin S$, there exists a nonzero vector $\vartheta$, such that $R^\circ(\theta;\vartheta) \le -c\|\vartheta\|$. Here the constant $c$ is uniform uniform across all $\theta$.*

This condition is crucial for ensuring the effectiveness of the penalty function method as noted in the literature [13, 26]. It guarantees that for every infeasible point, there is a direction in which the point can move toward the feasible set. This intuitive notion ensures that the solutions generated by Algorithm 2 are feasible. We now formally demonstrates that the sufficient decrease condition effectively precludes any infeasible stationary points.

**Theorem 6.** *Suppose $\mathbb{E}_f[T(\theta,f)]$ is globally Lipschitz continuous with Lipschitz constant $L_T$. Let Assumption 2 hold. Given the penalty parameter $\rho > L_T/c$, any infeasible point of the penalty function* (28) *can not be a $D$-stationary point.*

*Proof.* Rewrite the penalty function as $\Upsilon(\theta) = \mathbb{E}_f[T(\theta,f)] + \rho R(\theta)$. For any infeasible point $\theta$, using Assumption 2, there must exists a direction $\vartheta$, such that

$$\Upsilon^\circ(\theta;\vartheta) = \mathbb{E}_f[T(\cdot,f)]^\circ(\theta;\vartheta) + \rho R^\circ(\theta,\vartheta) \le L_T\|\vartheta\| - c\rho\|\vartheta\| < 0$$

Invoking proposition 8, the criteria of Clarke stationary point, we know $\theta$ cannot be a Clarke stationary point of $\Upsilon$. Meanwhile, according to [18, Corollary 1], Clarke subdifferential is a minimal conservative gradient. For any conservative gradient $D^\Upsilon$ of $\Upsilon$, we have $\partial\Upsilon \subset D^\Upsilon$. Hence, $\theta$ can not be a $D$-stationary point of $\Upsilon$. □ □

Next, we show that Algorithm 2 converges a $D$-stationary point of the penalty function (28). We take the viewpoint of differential inclusion and utilize some fundamental results in this field [27, 16]. Consider the following differential inclusion

$$\frac{\mathrm{d}\theta(t)}{\mathrm{d}t} = -D^\Upsilon(\theta(t)), \quad \text{for a.e. } t \ge t_0 \tag{99}$$

Then, the update scheme in Algorithm 2 can be viewed as a noisy discretization of the differential inclusion (99)

$$\theta_{k+1} = \theta_k - \eta_k(\varrho_k + \xi_k), \quad \varrho_k \in D^\Upsilon(\theta_k) \tag{100}$$

The existence and uniqueness of solutions to equation (99) are established in [8]. The following lemma, also found in [27] but presented here using the conventions of conservative gradients, guarantees a descent property for $\Upsilon(\zeta(t))$ when $\zeta$ is a trajectory of (99) with $\mathcal{G} = D^\Upsilon$. This property directly stems from the outer-semicontinuity of the conservative gradient.

**Lemma 6.** *Set $\mathcal{G} = D^\Upsilon$ in* (99)*. Then the differential inclusion* (99) *satisfies the descent property: For any trajectory $\zeta : \mathbb{R}_+ \to \mathbb{R}^d$ of the differential inclusion* (99) *where $0 \notin \mathcal{G}(\zeta(0))$, there exists a real $t_1 > 0$ such that*

$$\varkappa(\zeta(t_1)) < \sup_{t\in[0,t_1]} \varkappa(\zeta(t)) \le \varkappa(\zeta(0))$$

*Proof.* We first verify the outer-semicontinuity of $D^\Upsilon$. Employing the definition from [51, sec. 5B], the outer limit of a mapping $S$ is

$$\limsup_{\theta\to\overline{\theta}} S(\theta) := \bigcup_{\theta^\nu\to\overline{\theta}} \limsup_{\nu\to\infty} S\left(\theta^\nu\right) = \left\{u \mid \exists \theta^\nu \to \overline{\theta}, \exists u^\nu \to u \text{ with } u^\nu \in S\left(\theta^\nu\right)\right\}$$

A set-valued mapping $S$ is outer-semicontinuous (osc) at $\overline{\theta}$ if $\limsup_{\theta \to \overline{\theta}} S(\theta) \subset S(\overline{\theta})$. The graph-closed property ensures that the conservative gradient $D^{\Upsilon}$ is outer-semicontinuous.

Suppose $\zeta$ is a solution of the differential inclusion (99) with $\mathcal{G} = D^{\Upsilon}$ and $0 \notin D^{\Upsilon}(\zeta(0))$, the outer-semicontinuity of $D^{\Upsilon}$ guarantees that there exist $\delta > 0$ and $t_1 > 0$, such that

$$\|D^{\Upsilon}(\zeta(t))\| := \min_{\varrho \in D^{\Upsilon}(\zeta(t))} \|\varrho\| \geq \delta, \quad \text{for all} \quad t \in [0, t_1]$$

Using the fact $\|\dot{\zeta}(t)\| = \|D^{\Upsilon}(\zeta(t))\|$ [27, Lemma 5.2] and the chain rule for conservative gradient we get

$$\frac{\mathrm{d}\Upsilon(\zeta(t))}{\mathrm{d}t} = \langle \varrho, \dot{\zeta}(t) \rangle = -\|\dot{\zeta}(t)\|^2 = -\|D^{\Upsilon}(\zeta(t))\|^2, \quad \varrho \in D^{\Upsilon}(\zeta(t)) \quad \text{a.e.}$$

Combining both results we get the descent property. □ □

To derive a convergence guarantee for Algorithm 2, we make the following assumptions of (100).

**Assumption 3** (Assumptions of the SGD)**.**

1. *The step sizes* $\{\eta_k\}_{k \geq 1}$ *satisfy*

$$\eta_k \geq 0, \quad \sum_{k=1}^{\infty} \eta_k = \infty, \quad \text{and} \quad \sum_{k=1}^{\infty} \eta_k^2 < \infty$$

2. *Almost surely, the iterates* $\{\theta_k\}_{k \geq 1}$ *are bounded, i.e.,* $\sup_{k \geq 1} \|\theta_k\| < \infty$.

3. $\{\xi_k\}_{k \geq 1}$ *is a uniformly bounded difference martingale sequence with respect to the increasing* $\sigma$*-fields*

$$\mathcal{F}_k = \sigma(\theta_j, \varrho_j, \xi_j \colon j \leq k)$$

*In other words, there exists a constant* $M_\xi > 0$ *such that*

$$\mathbb{E}[\xi_k \mid \mathcal{F}_k] = 0 \quad \text{and} \quad \mathbb{E}[\|\xi_k\|^2 \mid \mathcal{F}_k] \leq M_\xi \quad \text{for all} \quad k \geq 1$$

4. *The complementary of* $\{\Upsilon(\theta) \mid 0 \in D^{\Upsilon}(\theta)\}$ *is dense in* $\mathbb{R}$.

Assumption 3 (4) is a technical assumption termed the weak Sard property. It is necessary since there exist a famous counter example [58]. Using [27, Lemma 4.1], we know Assumption 3 (1-3) implies [27, Assumption A] almost surely. One difference is that they assume there exists a function $\varkappa$ that bounded on bounded sets such that

$$\mathbb{E}[\|\xi_k\|^2 \mid \mathcal{F}_k] \leq \varkappa(\theta_k)$$

Given the boundedness of $\{\theta_k\}$, we know that this equals to Assumption 3 (3). We are now at the position to give the convergence result for Algorithm 2.

**Theorem 7** (Convergence guarantee for Algorithm 2)**.** *Suppose Assumption 2 and 3 hold,* $\mathbb{E}_f[T(\theta, f)]$ *is local Lipschitz smooth, and* $\{\theta_k\}_{k \geq 1}$ *is generated by Algorithm 2. Then almost surely, every limit point* $\theta_\star$ *of* $\{\theta_k\}_{k \geq 1}$ *satisfies* $\theta_\star \in S$, $0 \in D^{\Upsilon}(\theta_\star)$ *and the sequence* $\{\Upsilon(\theta_k)\}_{k \geq 1}$ *converges.*

*Proof.* When Assumption 3 (1-3) hold, using our comment we know that [27, Assumption A] holds almost surely. Invoking Assumption 3 (4) and Lemma 6, we know Assumption [27, Assumption B] holds. Then, applying [27, Theorem 3.2], we get $0 \in D^{\Upsilon}(\theta_\star)$ and the convergence of $\{\Upsilon(\theta_k)\}_{k\geq 1}$.

Assumption 3 (2) and the local Lipschitz smoothness of $\mathbb{E}_f[T(\theta, f)]$ justify treating $\mathbb{E}_f[T(\theta, f)]$ as globally Lipschitz smooth. By integrating this observation with Assumption 2, we invoke Theorem 6 to eliminate the possibility of any infeasible points being stationary. This line of reasoning ensures that all limiting points must be feasible, that is, $\theta_\star \in S$, thereby completing the proof. □ □

To conclude this section, we note that it's possible to establish a connection between the stationary points obtained using finite sample approximations and the true stationary points at the population level. Specifically, as the number of samples increases indefinitely, the limit points of the finite-sample stationary points converge to the population-level stationary points. For a detailed discussion of this result, one can refer to Proposition 17 in [25].

# 6 Numerical results on the training process

This section presents the results obtained from running Algorithm 2 and verifies Theorem 7 through numerical experiments. We provide a detailed description of the dataset, the approach used to construct the training problems, and the implementation details of the training process. The numerical experiments were implemented using the PyTorch platform on a Lenovo workstation equipped with an Intel i9 processor, 64 GB of RAM, and an NVIDIA RTX 4090 GPU, running the Windows Subsystem for Linux.

## 6.1 Methodology for constructing minimization problems

In this subsection, we describe the methods used to construct the minimization problems referenced in (14). Given a dataset $\Sigma$, in $k$-th iteration of Algorithm 2, we draw finite instances from $\Sigma$ and construct the set $\mathscr{D}_k$. Then, the function $f_k := f_{\mathscr{D}_k}$ is constructed and can be used in Algorithm 2.

We test Algorithm 2 in two type minimization problems. Consider a finite set of instances, $\mathscr{D}$, consisting of data pairs $\{a_i, b_i\} \in \mathbb{R}^n \times \{0, 1\}, i \in [|\mathscr{D}|]$. The first type of minimization problem is a logistic regression problem defined by:

$$\min_{x\in\mathbb{R}^n} f_{\mathscr{D}}(x) = \frac{1}{|\mathscr{D}|} \sum_{(a_i,b_i)\in\mathscr{D}} \log(1 + \exp(-b_i\langle a_i, x\rangle))$$

Let $\sigma(t) = \frac{1}{1+\exp(-t)}$ belong to the interval (0,1). The Hessian matrix of $f_{\mathscr{D}}$ is given by:

$$\nabla^2 f_{\mathscr{D}}(x) = \frac{1}{|\mathscr{D}|} \sum_{(a_i,b_i)\in\mathscr{D}} b_i^2 a_i a_i^\top \sigma(b_i\langle a_i, x\rangle)(1 - \sigma(b_i\langle a_i, x\rangle))$$

Let $A = (a_1, \ldots, a_{|\mathscr{D}|})$. The Lipschitz constant of $\nabla f_{\mathscr{D}}$ is bounded by $L = \|AA^\top\|/|\mathscr{D}|$.

Given an even integer $p \geq 4$, the second type of minimization problem is the $\ell_p^p$ minimization, defined as follows:

$$\min_{x\in\mathbb{R}^n} f_{\mathscr{D}}(x) = \frac{1}{|\mathscr{D}|} \sum_{(a_i,b_i)\in\mathscr{D}} \frac{1}{p}(\langle a_i, x\rangle - b_i)^p$$

**Table 1:** A summary of the datasets used in experiments.

| Dataset | $n$ | $N_{\text{train}}$ | $N_{\text{test}}$ | Separable | References |
|---|---|---|---|---|---|
| `a5a` | 123 | $6,414$ | $26,147$ | No | [30] |
| `w3a` | 300 | $4,912$ | $44,837$ | No | [50] |
| `mushrooms` | 112 | $3,200$ | $4,924$ | Yes | [30] |
| `covtype` | 54 | $102,400$ | $478,612$ | No | [30] |
| `phishing` | 68 | $8,192$ | $2,863$ | No | [30] |
| `separable` | 101 | $20,480$ | $20,480$ | Yes | [61] |

The Hessian matrix of $f_{\mathscr{D}}$ is expressed as:

$$\nabla^2 f_{\mathscr{D}}(x) = \frac{1}{|\mathscr{D}|} \sum_{(a_i,b_i)\in\mathscr{D}} (p-1)(\langle a_i, x\rangle - b_i)^{p-2} a_i a_i^\top$$

As $(\langle a_i, x\rangle - b_i)^{p-2}$ is unbounded for each $i$, the Lipschitz constant for $\nabla f_{\mathscr{D}}$ cannot be globally bounded.

### 6.2 Datasets for constructing minimization problems

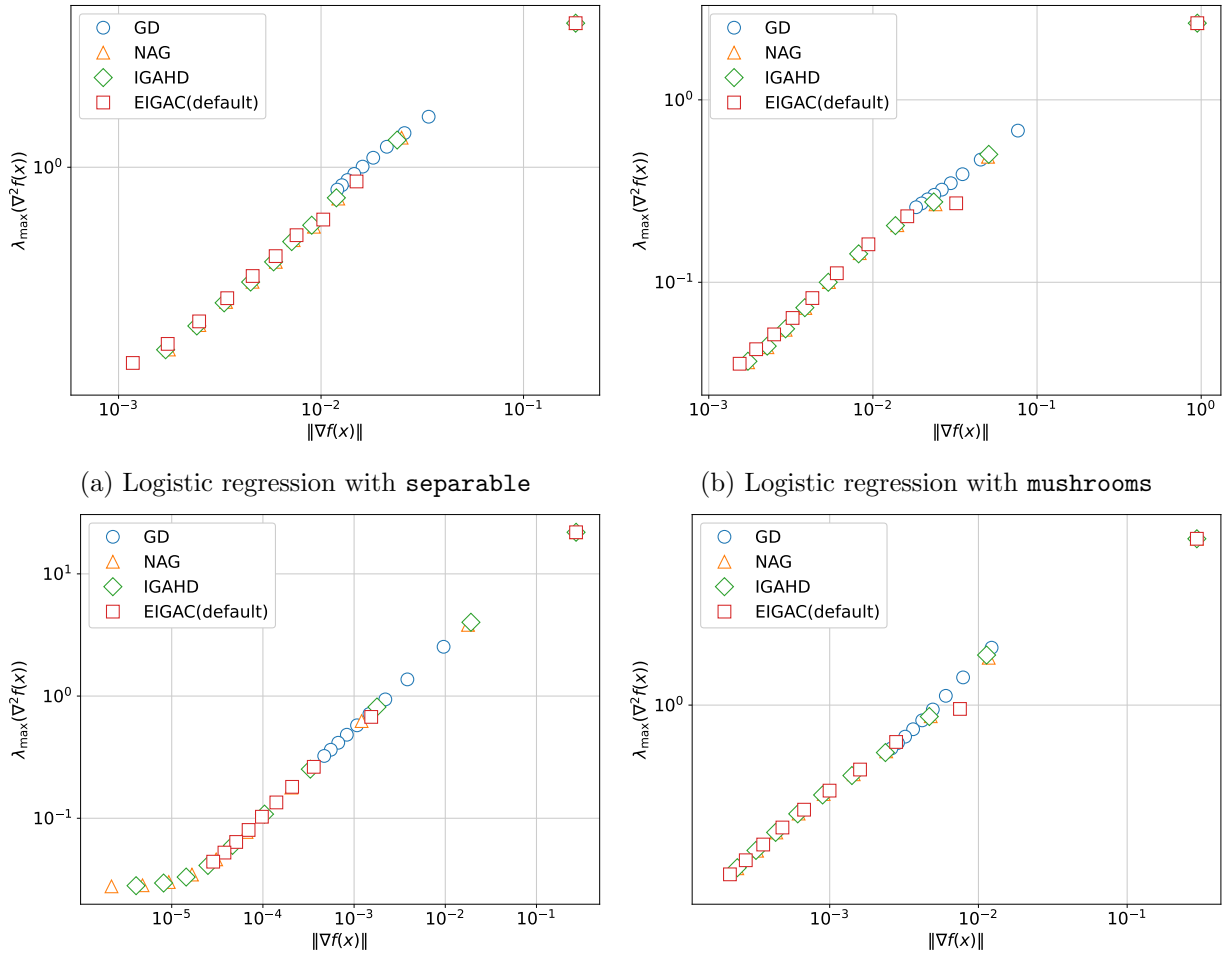
The datasets used in our experiments are summarized in Table 1. In this table, $n, N_{\text{train}}$, and $N_{\text{test}}$ represent the dimension of the variable, the number of instances in the training dataset, and the number of instances in the test dataset, respectively.

All the datasets are designed for binary classification problems, and downloaded from the LIBSVM data, except the `separable` dataset. We construct the `separable` dataset using the code snippet downloaded from [61]. They are generated by sampling 10240 instances from $\mathcal{N}(\mu, I_d)$ with label $b_i = 1$ and $\mathcal{N}(\mu + \nu, I_d)$ with label $b_i = 0$, respectively. Here, $I_d \in \mathbb{R}^{d\times d}$ denotes the identity matrix. Each element of the vector $\mu \in \mathbb{R}^d$ is sampled from $\{0, 1, \ldots, 19\}$ uniformly, while the elements of the margin vector $\nu$ are drawn from $\{0, 0.1, \ldots, 0.9\}$ uniformly.

For each dataset, the label of each sample belongs to $\{0, 1\}$. The value of each attribute are normalized to $[-1, 1]$ by dividing the data-matrix $(a_1, a_2, \ldots, a_N)$ with the max absolute value of each attribute. The training and testing sets are pre-specified for `a5a` and `w3a`. For datasets that do not specify the testing set and training set, we divide them manually.

### 6.3 Verifying the $(L_0, L_1)$-smoothness assumption

Both the objective function used in logistic regression and the $\ell_p^p$ minimization problem exhibit $(L_0, L_1)$-smoothness. This characteristic remains consistent across various datasets, with the constants $L_0$ and $L_1$ depending on the specific dataset $\mathscr{D}$. To illustrate this, we applied four different algorithms to two datasets, the details of which will be provided in subsection 6.6. Each algorithm was executed for 300 steps, and every 30 steps, we plotted the point $(\|\nabla f(x_k)\|, \Lambda(x_k, f))$ on a log-log scale scatter plot. Despite the diversity in the methods used, all points aligned along the same line, underscoring the $(L_0, L_1)$-smoothness as an intrinsic property of the function $f_{\mathscr{D}}$. Figure 2 shows a pronounced decline in $\Lambda(x_k, f)$ corresponding to decreases in $\|\nabla f(x_k)\|$, confirming the arguments in sec. 4.4.




(a) Logistic regression with `separable`

(b) Logistic regression with `mushrooms`

(c) $\ell_p^p$ minimization with `separable`

(d) $\ell_p^p$ minimization with `mushrooms`

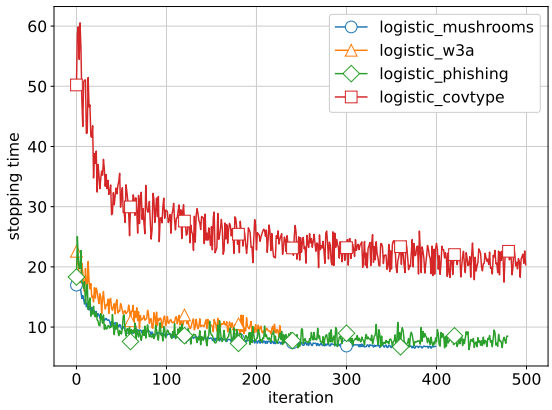
**Figure 2:** Numerical verification of the $(L_0, L_1)$-smoothness.

### 6.4 Implementation details and the training process

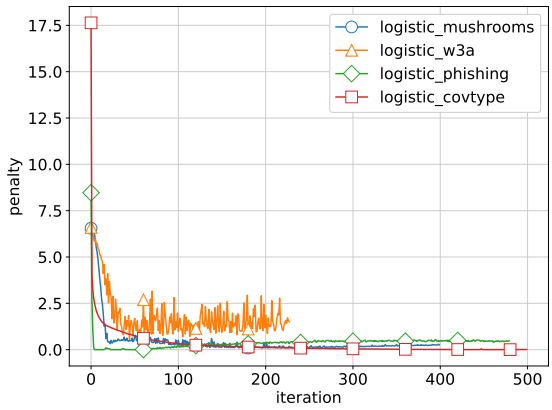
Let $\sigma(\cdot)$ represent applying the SoftPlus function element-wise, i.e., $[\sigma(x)]_i = \log(1+\exp(20x_i))/20$. Given the dimension of the hidden state $d_{\mathrm{h}}$, we parameterize $\beta$ and $\gamma$ as follows:

$$\beta_{\theta_1}(t) = W_3^\top \sigma(W_2\sigma(W_1 t + b_1) + b_2) + b_3, \quad \gamma_{\theta_2}(t) = V_3^\top \sigma(V_2\sigma(V_1 t + c_1) + c_2) + c_3$$

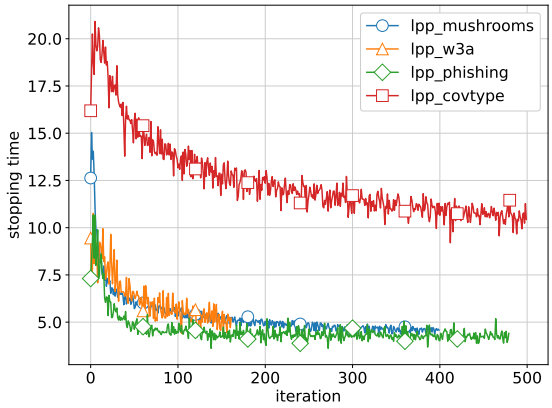
where $W_1, V_1, W_3, V_3, b_1, c_1, b_2, c_2 \in \mathbb{R}^{d_{\mathrm{h}}}, W_2, V_2 \in \mathbb{R}^{d_{\mathrm{h}}\times d_{\mathrm{h}}}$ and $b_3, c_3 \in \mathbb{R}$. In this context, the SoftPlus function ensures the differentiability of $\beta_{\theta_1}$ and $\gamma_{\theta_2}$ with respect to $t$. We define $\Lambda(x, f) := \lambda_{\max}(\nabla^2 f(x))$ to leverage the variability of the local Lipschitz constant of $\nabla f$. To control the computational complexity for evaluating $\Lambda(x, f)$, we combine the power iteration with the forward automatic differentiation. We initiate the algorithm with a randomly generated vector $u_0$. In each iteration, $z_{k+1}$ is computed successively using the formula $z_{k+1} = \nabla^2 f(x) z_k / \|\nabla^2 f(x) z_k\|$. The algorithm terminates when either the Euclidean norm of the difference between two successive iterations is less than or equal to $10^{-6}$, i.e., $\|z_{k+1} - z_k\| \le 10^{-6}$, or the number of iterations reaches or exceeds 10, i.e., $k \ge 10$. The output of the algorithm is denoted as $u_\star$. We restore it for the backpropagation and use the approximation $\Lambda(x, f) \approx u_\star^\top \nabla^2 f(x) u_\star$. The penalty terms $P$ and $Q$



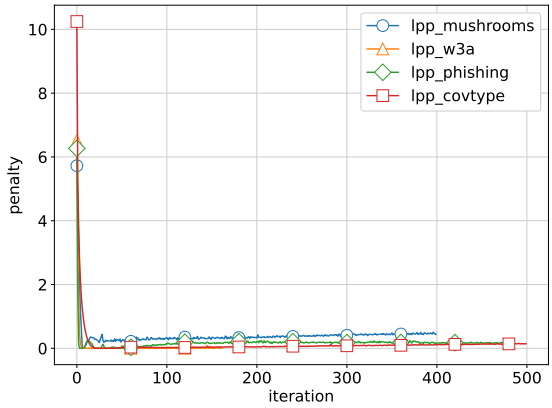

(a) Stopping time on logistic regression




(b) Penalty on logistic regression




(c) Stopping time on $\ell_p^p$ minimization




(d) Penalty on $\ell_p^p$ minimization

**Figure 3:** The training process in different tasks.

can be calculated by augmenting (11) as

$$(\dot{s}(t), \dot{P}, \dot{Q}) = (\psi_\theta(s(t), t), p(t, f, \theta), q(t, f, \theta))$$

Integrating this system from $t_0$ to $T(f, \theta)$ gives the value of $P$ and $Q$. The package `torchdiffeq` implemented using `PyTroch` provides the implementation of the adjoint sensitivity method. We integrate it with our implementation of the backpropagation of $T(\theta, f)$ and $\Lambda(x, f)$. This combination enables the automatic differentiation through $\Upsilon(\theta)$.

The initial point is chosen as $x_0 = x_1 = \mathbf{1}/n - \nabla f_{\mathscr{D}}(\mathbf{1}/n)/L$, and $v_0 = x_0 + \beta(t_0)\nabla f(x_0)$. All elements of the $n$-dimensional vector $\mathbf{1}$ are equal to 1. We set $h = 0.04$, $t_0 = 1$, and $L = \min\{\|A^\top A\|/N, 4\Lambda(\mathbf{1}/n, f_{\mathscr{D}})\}$. Here $L$ is a tight estimate of the Lipschitz constant of $\nabla f_{\mathscr{D}}$. For each dataset, the stopping criterion is $\|\nabla f(x)\| = \varepsilon$ with $\varepsilon = 3 \times 10^{-4}$, and the penalty coefficient $\rho$ of (28) is 0.1. We adopt the `SGD` optimizer in `PyTorch` with the learning rate 0.001 and the momentum 0.9. The number of epochs is $n_{\text{epoch}} = 60$. We initialize the coefficient functions as $\alpha = 6$, $\beta(t) = (4 - 2\alpha h/t)/L$, and $\gamma(t) = \beta(t)/h$. In $k$-th step of Algorithm 2, we construct a problem $f_k := f_{\mathscr{D}_k}$ by sampling $n_{\text{sp}} = 10240$ instances from the training dataset for `mushrooms` dataset and $n_{\text{sp}} = 1024$ instances for other datasets. Then, we perform one step of the Algorithm 2 to update $\theta$.

Figure 3 illustrates the training process, where `logistic_mushrooms` signifies logistic regression on the `mushrooms` dataset, `lpp_mushrooms` refers to $\ell_p^p$ minimization on the same dataset, and so forth. The training loss is decomposed into the stopping time and penalty term. The subfigures within Figure 3 clearly demonstrate a decrease in both constraint violation and stopping time across all experiments. This aligns with Theorem 7, confirming that Algorithm 2 converges to a feasible $D$-stationary point of problem (27).

**Numerical results on the testing process** In the next four subsections, we present a series of numerical experiments to illustrate the advantages of our L2O framework (27). We will demonstrate that by discretizing the learned ODE, our approach achieves acceleration compared to baseline methods.

### 6.5 Effects of the contraction factor and the relay EIGAC

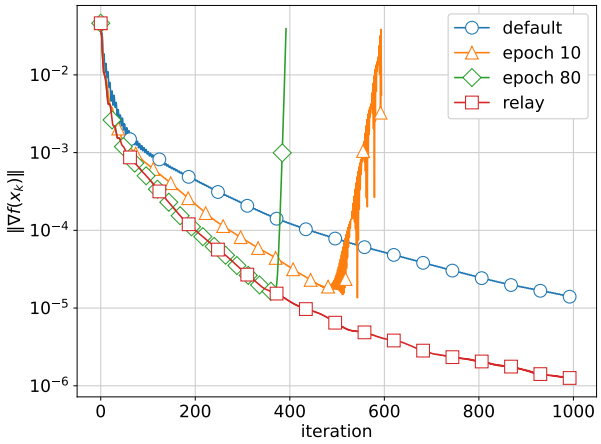
In this subsection, we investigate the effects of the contraction factor using the `lpp_a5a` problem. A relay strategy for EIGAC is proposed to generalize the learned EIGAC to unseen scenarios while maintaining convergence. The parameterized ODE is trained for 80 epochs using Algorithm 2. As shown in Figure 4, the performance of the learning-based method is compared with different training epochs on the same `lpp_a5a` testing problem. The corresponding contraction factor at each step is plotted with the following legend descriptions:

- `default` represents the EIGAC method with initial coefficients $\alpha = 6$, $\beta(t) = (4/h - 2\alpha/t)/L$, and $\beta(t) = h\gamma(t)$, where $L = \min\left\{\|A^\top A\|/N, 4\Lambda\left(\mathbf{1}/n, f\right)\right\}$;
- `epoch 10` represents the EIGAC method with coefficients trained for 10 epochs, and similarly for `epoch 80`;
- `relay` combines the coefficients trained for 80 epochs with a safeguard strategy to extrapolate the method to untrained scenarios. When $\|\nabla f(x_k)\| \geq 5\|\nabla f(x_0)\|$ and $\|\nabla f(x_k)\| < 3 \times 10^{-4}$, the coefficients $\gamma(t)$ are replaced with $(4/h - 2\alpha/t)/\Lambda(x_k, f)$ and $\beta(t) = h\gamma(t)$.

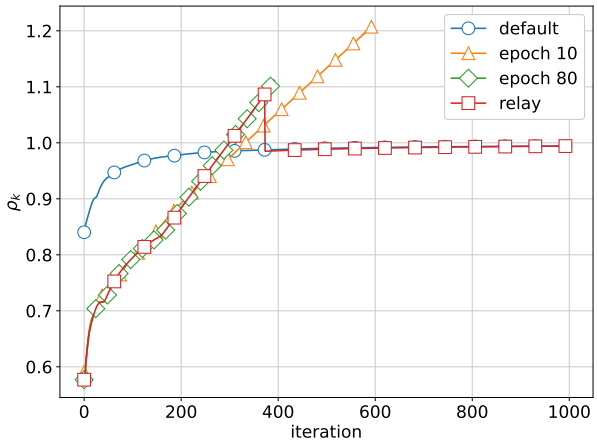
Here the estimator is chosen as $\Lambda(x, f) = \lambda_{\max}(\nabla^2 f(x))$. Besides the gradient norm, we also provide the figures of the contraction factor $\rho_k = \rho(t_k, G_k)$, the determination $C_k = C(t_k, \|G_k\|)$, and $B_k = B(t_k, \|G_k\|)$ with $G_k = \int_0^1 \nabla^2 f((1-\tau)x(t_k) + \tau x_k)\,\mathrm{d}\tau$. The functions $B, C$ are defined in Proposition 7.

The results demonstrate that the contraction factor is a reliable predictor for the behavior of the sequence $\{x_k\}_{k=0}^\infty$ generated by Algorithm 1, supporting the intuition behind the proof of Theorem 2. Despite initial oscillations, the `default` method does not diverge, as the contraction factor remains strictly less than 1.

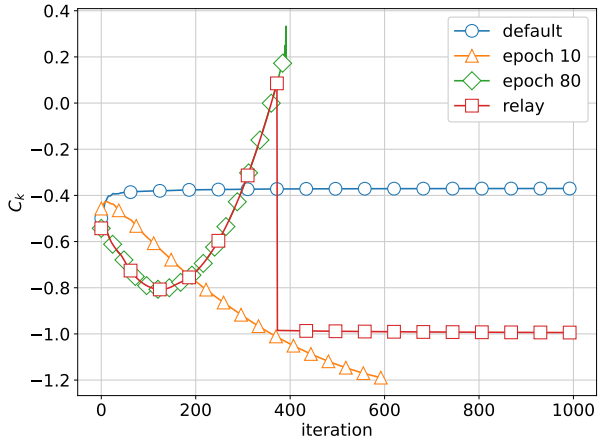
The `epoch 10` method, trained for 10 epochs, converges faster than the `default` method but diverges quickly after reaching a threshold $\|\nabla f(x)\| = 3 \times 10^{-5}$. This divergence occurs because its contraction factor is greater than 1 after 330 iterations, leading to an exponential accumulation of the global truncated error. The `epoch 80` method, which converges even faster, also faces divergence. However, the divergence patterns differ: the contraction factor for the `epoch 10` method increases gradually above 1, causing a gradual divergence with oscillation. In contrast, the `epoch 80` method diverges rapidly due to $C_k$ becoming greater than 0, forcing $\max\{B_k - 1, 1 - B_k\}$ to quickly increase the contraction factor far above 1, leading to immediate divergence without gradual error accumulation.



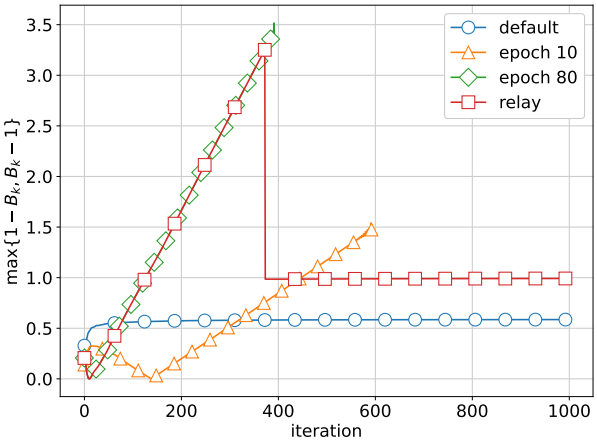

(a) $\ell_p^p$ minimization on `a5a`




(b) The contraction factor $\rho$ on `a5a`




(c) Determination versus iterations




(d) $\max\{B_k - 1, 1 - B_k\}$ versus iterations

**Figure 4:** Different indicators of $\ell_p^p$ minimization problem on `a5a` dataset.

The `relay` method remains stable after the `epoch 80` method diverges. This is due to the relay strategy, which reduces the contraction factor to below 1, maintaining stability. Upon detecting divergence, relay is applied to the coefficients, resulting in a quick decrease in $C_k$ to ensure it stays below 0. This relay strategy method will be applied for comparison with other methods.

## 6.6 Compared methods

In testing process, we adopt the same implementation details and convention used in the training process 6.4. Suppose the minimization problem used for testing is constructed from the set $\mathscr{D}$. we set $L = \min\{\|A^\top A\|/N, 4\Lambda(\mathbf{1}/n, f_{\mathscr{D}})\}$, $x_0 = x_1 = \mathbf{1}/n - \nabla f_{\mathscr{D}}(\mathbf{1}/n)/L$, and $v_0 = x_0 + \beta(t_0)\nabla f_{\mathscr{D}}(x_0)$. The compared methods in our experiments are listed below, where we abbreviate the $f_{\mathscr{D}}$ as $f$ for simplicity.

- **GD**. The vanilla gradient descent GD is the standard method in optimization. We set the stepsize as $s = 1/L$ and perform $x_{k+1} = x_k - s\nabla f(x_k)$ in each iteration.
- **NAG**. Nesterov's accelerated gradient descent method NAG is a milestone of the acceleration

methods. We employ the version for convex functions

$$y_{k+1} = x_k - s\nabla f(x_k), \quad x_{k+1} = y_{k+1} + \frac{k-1}{k+2}(y_{k+1} - y_k)$$

where the stepsize is chosen as $s = 1/L$.

- **IGAHD**. Inertial gradient algorithm with Hessian-driven damping. This method is obtained by applying a NAG inspired time discretization of

$$\ddot{x}(t) + \frac{\alpha}{t}\dot{x}(t) + \beta\nabla^2 f(x(t))\dot{x}(t) + \left(1 + \frac{\beta}{t}\right)\nabla f(x(t)) = 0 \tag{101}$$

Let $s = 1/L$. In each iteration, setting $\alpha_k = 1 - \alpha/k$, the method performs

$$\begin{cases} y_k = x_k + \alpha_k\left(x_k - x_{k-1}\right) - \beta\sqrt{s}\left(\nabla f\left(x_k\right) - \nabla f\left(x_{k-1}\right)\right) - \dfrac{\beta\sqrt{s}}{k}\nabla f\left(x_{k-1}\right), \\ x_{k+1} = y_k - s\nabla f\left(y_k\right) \end{cases} \tag{102}$$

In [5], it has been show that IGAHD owns $\mathcal{O}(1/k^2)$ convergence rate when $0 \leq \beta < 2/\sqrt{s}$ and $s \leq 1/L$. Its performance may not coincide with NAG due to the existence of the gradient correction term. In our experiments, IGAHD serves as a baseline of the optimization methods derived from the ODE viewpoint without learning.

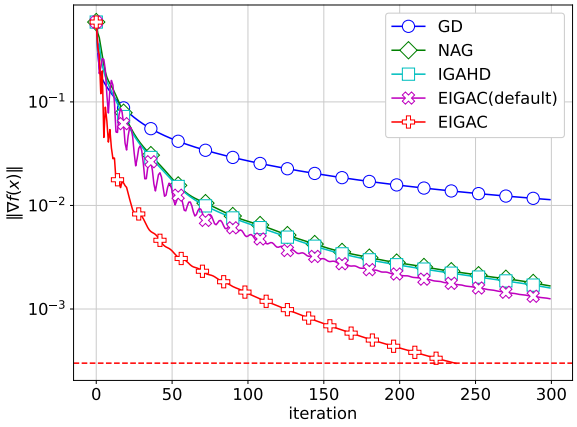
- **EIGAC**. Explicit inertial gradient algorithm with correction, i.e. Algorithm 1. We provide two versions of EIGAC with default coefficients as described in sec. 6.5 and the coefficients learned by Algorithm 2. The numerical experiments effectively show that the EIGAC with default coefficients are sufficient to converge and the performance is comparable with NAG, while EIGAC with learned coefficients is superior over other methods.

### 6.7 Exemplary cases

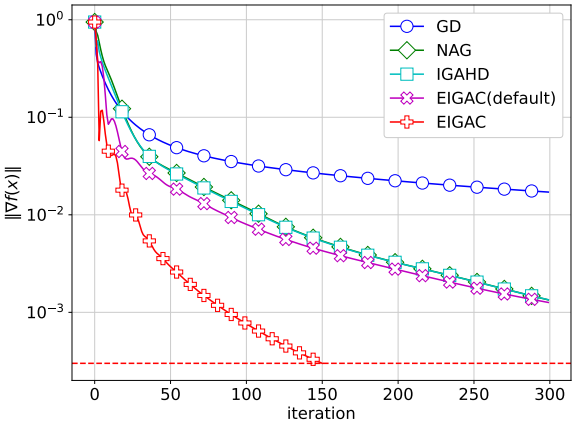
To get a detailed observation of the behavior of each method, we show the performance of them in exemplary cases. For each problem, we generate one testing minimization problem using the method described in sec. 6.1.

In Figures 5 and 6, the gradient norm versus the number of iterations for logistic regression and $\ell_p^p$ minimization are plotted, respectively. To emphasize the improvement obtained from the training process, these methods are terminated when the gradient norm is smaller than $3 \times 10^{-4}$ or the number of iterations is larger than 300. The threshold $3 \times 10^{-4}$ is plotted using a red dash line. In the hard cases `lpp_phishing` and `logistic_phishing`, EIGAC also saves at least half of the iterations. These exemplary cases suggest that EIGAC is general enough and has strong potential to be used in practice.

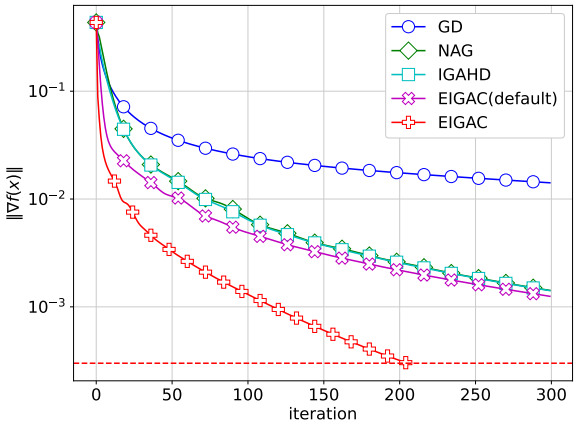
Figure 7 presents the function value versus the number of iterations of four different minimization problems. In these problems, we do not terminate the methods and show how our relay strategy helps stabilize the behavior of relay EIGAC. The results demonstrate that EIGAC surpasses other methods, even in the stage that not specifically trained for it. In the easiest case `logistic_phishing`, EIGAC achieves the optimal value much faster than others. In Figure 7c, after achieving the optimal value, EIGAC does not diverge and oscillate. This shows the effectiveness of our relay strategy.



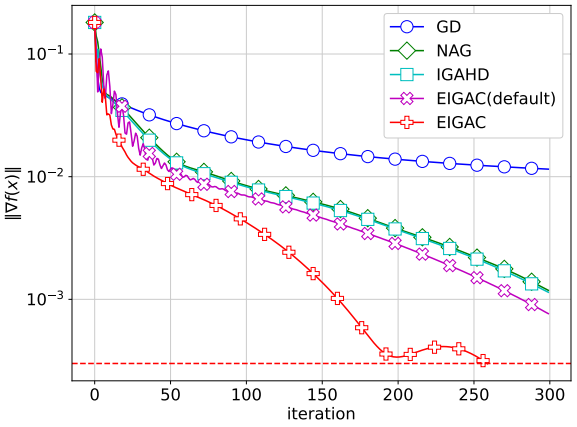

(a) `a5a`



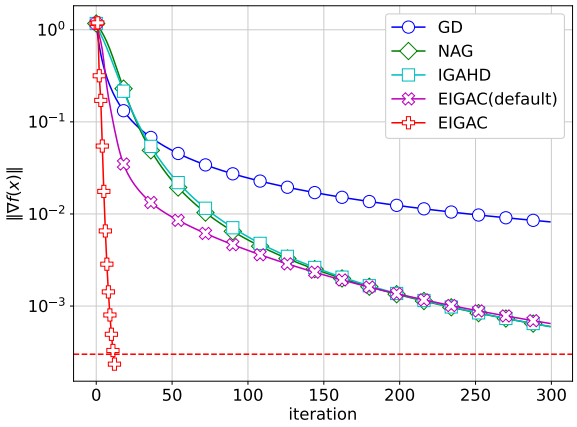

(b) `mushrooms`



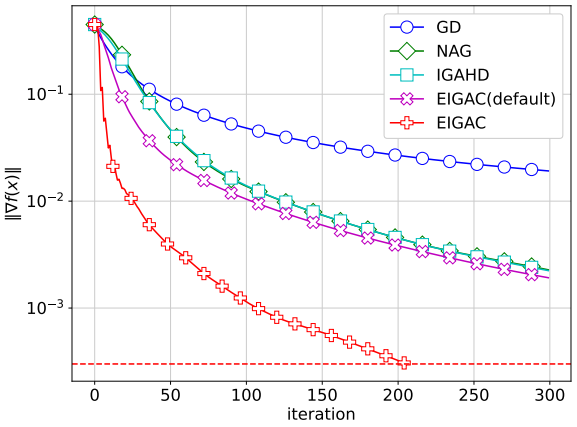

(c) `w3a`




(d) `covtype`

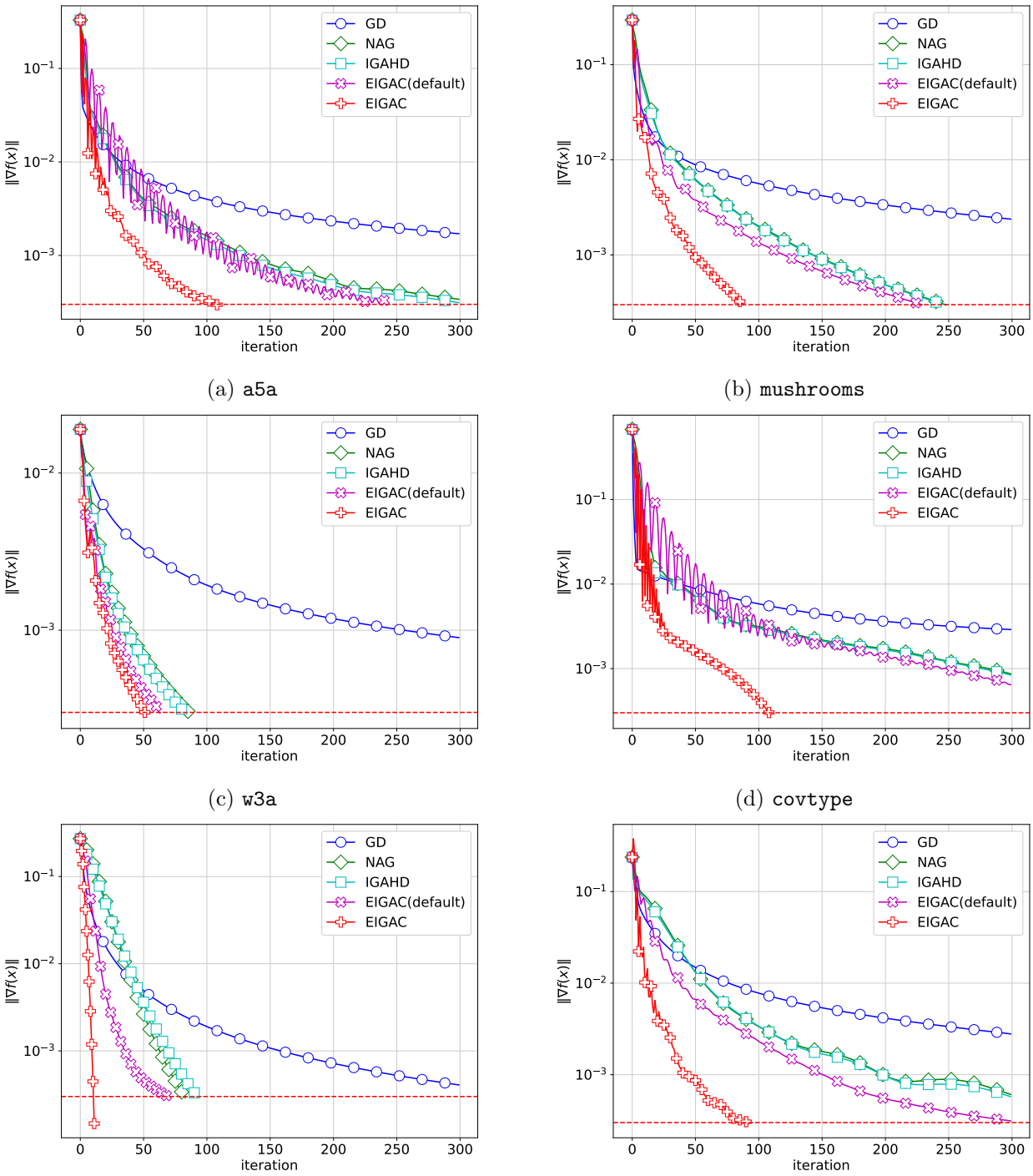



(e) `separable`




(f) `phishing`

**Figure 5:** Comparison on logistic regression.




(a) `a5a` (b) `mushrooms`

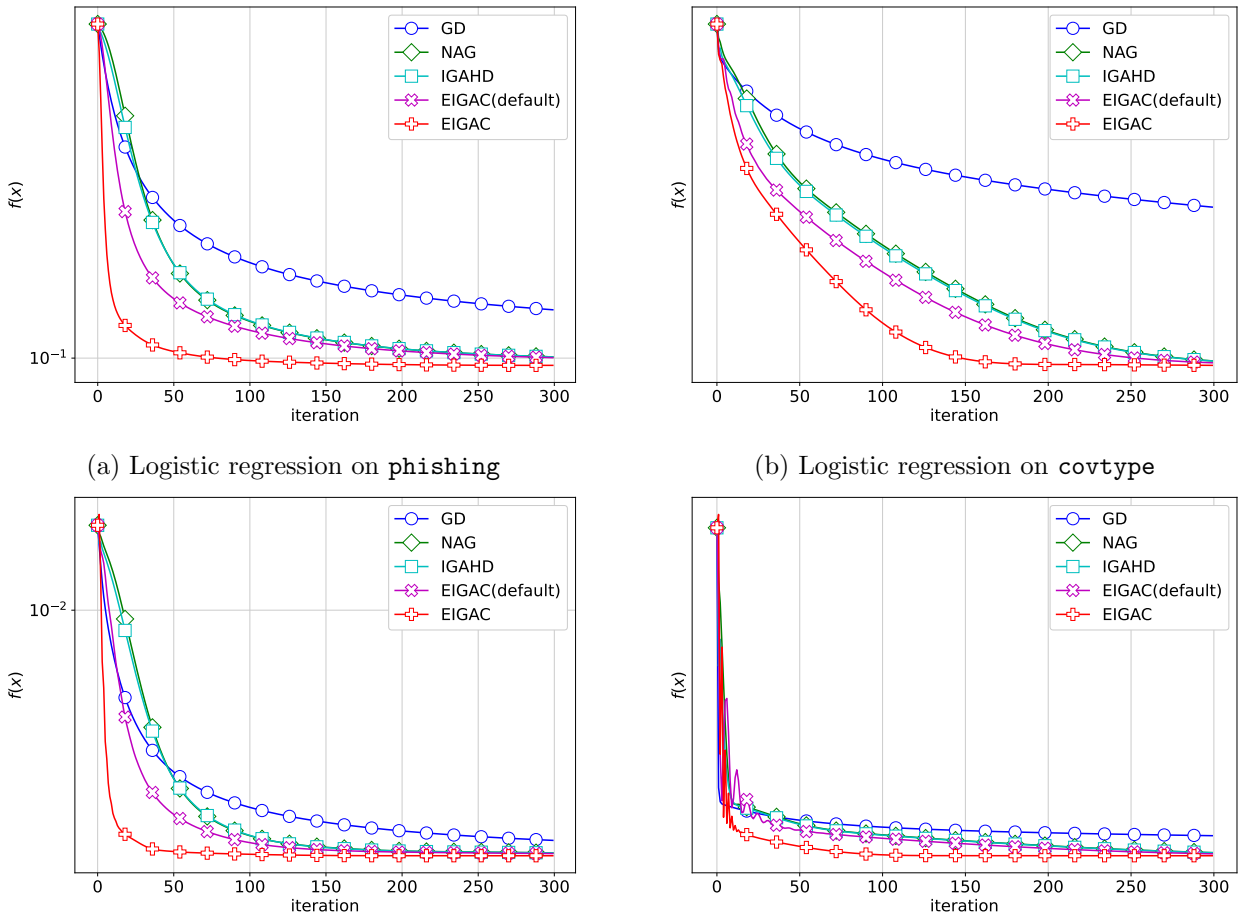
(c) `w3a` (d) `covtype`

(e) `separable` (f) `phishing`

**Figure 6:** Comparison on $\ell_p^p$ Minimization.

## 6.8 Averaged statistics

In this subsection, we randomly generate 100 testing minimization problems, denoted as $\mathcal{F}_{\text{test}} = \{f_i\}_{i=1}^{100}$, for each type of minimization problem using different datasets. The construction method-




(a) Logistic regression on `phishing`

(b) Logistic regression on `covtype`

(c) $\ell_p^p$ minimization on `phishing`

(d) $\ell_p^p$ minimization on `covtype`

**Figure 7:** Comparison on logistic regression.

ology follows that described in sec. 6.1. A problem is defined by the dataset and the type of the minimization problem. For example, the problem `lpp_a5a` refers to the `a5a` dataset with the $\ell_p^p$ minimization formulation. Multiple test functions for the same problem can be generated by varying the instances from the dataset used.

Given the type of the minimization problem and the dataset, for each method, we provide two statistics. The first one is the averaged performance measure at the $N$-th iteration:

$$\overline{m}(\mathcal{F}_{\text{test}}) = \frac{1}{|\mathcal{F}_{\text{test}}|} \sum_{f \in \mathcal{F}_{\text{test}}} \log \|\nabla f(x_N)\|$$

We set $N = 500$ for comparison. The second one is the averaged complexity:

$$\overline{N}(\mathcal{F}_{\text{test}}) = \frac{1}{|\mathcal{F}_{\text{test}}|} \sum_{f \in \mathcal{F}_{\text{test}}} \inf\{n \mid \|f(x_n)\| \leq \varepsilon\}$$

with $\varepsilon = 3 \times 10^{-4}$. If the method does not reaching the threshold, we denote its complexity in this test as 500.

**Table 2:** The averaged performance measure in logistic regression problems.

| | mushrooms | a5a | w3a | phishing | covtype | separable |
|---|---|---|---|---|---|---|
| GD | -1.55(0.016) | -1.81(0.031) | -1.90(0.022) | -1.35(0.007) | -1.89(0.008) | -1.56(0.000) |
| NAG | -3.37(0.027) | -3.11(0.059) | -3.26(0.061) | -3.01(0.074) | -3.07(0.012) | -3.66(0.000) |
| EIGAC(default) | -3.02(0.020) | -2.97(0.065) | -3.02(0.044) | -2.80(0.064) | -3.48(0.059) | -3.32(0.000) |
| IGAHD | -3.02(0.020) | -2.97(0.065) | -3.02(0.043) | -2.80(0.064) | -3.48(0.059) | -3.31(0.000) |
| EIGAC | -4.83(0.017) | -4.38(0.062) | -4.46(0.093) | -4.82(0.120) | -4.37(0.055) | -5.49(0.090) |
| Win Rate | 100.00% | 100.00% | 100.00% | 100.00% | 100.00% | 100.00% |

**Table 3:** The averaged performance measure in $\ell_p^p$ minimization problems.

| | mushrooms | a5a | w3a | phishing | covtype | separable |
|---|---|---|---|---|---|---|
| GD | -2.49(0.018) | -2.79(0.042) | -3.18(0.053) | -2.36(0.015) | -2.65(0.008) | -2.95(0.001) |
| NAG | -4.35(0.033) | -4.19(0.058) | -4.72(0.085) | -4.06(0.052) | -4.43(0.091) | -6.11(0.077) |
| EIGAC(default) | -4.16(0.022) | -3.99(0.069) | -4.66(0.085) | -4.37(0.147) | -4.47(0.043) | -6.15(0.048) |
| IGAHD | -4.16(0.022) | -4.05(0.068) | -4.66(0.085) | -4.37(0.150) | -4.51(0.037) | -6.14(0.049) |
| EIGAC | -5.27(0.049) | -5.11(0.081) | -5.71(0.145) | -5.65(0.091) | -5.14(0.110) | -7.55(0.044) |
| Win Rate | 100.00% | 100.00% | 100.00% | 100.00% | 100.00% | 100.00% |

**Table 4:** The averaged complexity in logistic regression problems.

| | mushrooms | a5a | w3a | phishing | covtype | separable |
|---|---|---|---|---|---|---|
| GD | 500.00(0.000) | 500.00(0.000) | 500.00(0.000) | 500.00(0.000) | 500.00(0.000) | 500.00(0.000) |
| NAG | 500.00(0.000) | 500.00(0.000) | 500.00(0.000) | 500.00(0.000) | 500.00(0.000) | 424.71(0.456) |
| EIGAC(default) | 500.00(0.000) | 500.00(0.000) | 500.00(0.000) | 500.00(0.000) | 497.12(5.741) | 500.00(0.000) |
| IGAHD | 500.00(0.000) | 500.00(0.000) | 500.00(0.000) | 500.00(0.000) | 497.36(5.509) | 500.00(0.000) |
| EIGAC | 153.48(2.525) | 227.32(17.759) | 216.42(16.462) | 182.15(21.568) | 237.88(34.565) | 11.49(0.502) |
| Win Rate | 100.00% | 100.00% | 100.00% | 100.00% | 100.00% | 100.00% |

The averaged performance measure with the stand error in the parenthesis is reported in Table 2 and 3. The `Win Rate` row represents the rate of EIGAC outperforms other methods. In these experiments, we observed that the performance of NAG, IGAHD, and EIGAC(default) is similar, both in performance measure and complexity. The non divergence of EIGAC also verifies the effectiveness of the relay strategy. The result shows that EIGAC outperforms other methods with at least a magnitude in most cases. In the hard cases `logistic_covtype` and `lpp_covtype`, the improvement is still significant and shows the potential of the learning-based method.

In Tables 4 and 5, we present the averaged complexity for logistic regression and $\ell_p^p$ minimization problems, respectively. Since the value of $\log_{10}(3\times10^{-4})$ is approximately $-3.52$, GD, EIGAC, and IGAHD do not reach the threshold in most logistic regression problems within 500 iterations, as indicated by the performance measure statistics. In most logistic regression problems, the averaged complexity of EIGAC is only half that of other methods. In the easiest case, `logistic_separable`, EIGAC requires just $\frac{1}{40}$ of the iterations needed by NAG to reach the threshold. In the relatively easier $\ell_p^p$ minimization problems, EIGAC also shows a consistent improvement in complexity. While other methods achieve the threshold quickly, the averaged complexity of EIGAC is approximately half that of the others in each problem.

**Table 5:** The averaged complexity in $\ell_p^p$ minimization problems.

| | mushrooms | a5a | w3a | phishing | covtype | separable |
|---|---|---|---|---|---|---|
| GD | 500.00(0.000) | 500.00(0.000) | 500.00(0.000) | 500.00(0.000) | 500.00(0.000) | 500.00(0.000) |
| NAG | 183.77(4.583) | 211.87(17.323) | 92.61(7.590) | 252.43(16.373) | 167.31(1.650) | 52.15(0.359) |
| EIGAC(default) | 235.07(6.331) | 245.68(26.391) | 96.17(9.527) | 224.36(21.168) | 203.02(2.040) | 22.10(0.302) |
| IGAHD | 235.84(6.419) | 239.49(27.857) | 96.03(9.682) | 224.53(21.373) | 204.12(1.996) | 29.98(0.141) |
| EIGAC | 93.12(2.124) | 122.16(11.176) | 50.93(4.529) | 85.57(5.385) | 109.15(0.999) | 11.00(0.000) |
| Win Rate | 100.00% | 100.00% | 100.00% | 100.00% | 100.00% | 100.00% |

# 7 Conclusion and future directions

In this paper, we have introduced a comprehensive framework that integrates the inertial system with Hessian-driven damping (ISHD) and learning-based optimization (L2O) to develop efficient optimization methods. By establishing convergence conditions and analyzing the stability of explicit Euler discretization schemes, we derive a large family of practical optimization algorithms with a solid theoretical foundation. The formulation of a learning problem that minimizes the stopping time, subject to convergence and stability constraints, marks a significant step forward in L2O. Our approach, which combines penalty methods, stochastic optimization, and conservative gradients, effectively addresses this learning problem, with proven convergence guarantees for the learning process.

Extensive numerical experiments validate the superior performance of the learned optimization methods, highlighting the robustness and practical potential of our framework. However, the theoretical results may not fully explain the observed empirical performance, as methods with default coefficients, like EIGAC, outperform traditional algorithms like NAG despite having only an $\mathcal{O}(1/k)$ theoretical convergence rate. Future work will focus on refining the theoretical analysis to better align with the empirical results and extending the framework to handle more challenging optimization problems, such as non-smooth, constrained, and non-convex optimization tasks.

# References


[1] Ahmad, R., Bouman, C.A., Buzzard, G.T., Chan, S., Liu, S., Reehorst, E.T., Schniter, P.: Plug-and-play methods for magnetic resonance imaging: Using denoisers for image recovery. IEEE Signal Processing Magazine **37**(1), 105–116 (2020). DOI 10.1109/MSP.2019.2949470

[2] Andrychowicz, M., Denil, M., Colmenarejo, S.G., Hoffman, M.W., Pfau, D., Schaul, T., de Freitas, N.: Learning to learn by gradient descent by gradient descent. In: D.D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, R. Garnett (eds.) Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain, pp. 3981–3989 (2016). URL `https://proceedings.neurips.cc/paper/2016/hash/fb87582825f9d28a8d42c5e5e5e8b23d-Abstract.html`

[3] Attouch, H.: Fast algorithmic methods for optimization and learning. Recent trends. (2022)

[4] Attouch, H., Boţ, R.I., Csetnek, E.R.: Fast optimization via inertial dynamics with closed-loop damping. J. Eur. Math. Soc. (JEMS) **25**(5), 1985–2056 (2023)

[5] Attouch, H., Chbani, Z., Fadili, J., Riahi, H.: First-order optimization algorithms via inertial systems with Hessian-driven damping. Math. Program. **193**(1), 113–155 (2022)

[6] Attouch, H., Peypouquet, J., Redont, P.: A dynamical approach to an inertial forward-backward algorithm for convex minimization. SIAM J. Optim. **24**(1), 232–256 (2014)

[7] Attouch, H., Peypouquet, J., Redont, P.: Fast convex optimization via inertial dynamics with Hessian driven damping. J. Differential Equations **261**(10), 5734–5783 (2016). DOI 10.1016/j.jde.2016.08.020

[8] Aubin, J.P., Cellina, A.: Differential inclusions, *Grundlehren der mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]*, vol. 264. Springer-Verlag, Berlin (1984). Set-valued maps and viability theory

[9] Aujol, J.F., Dossal, C., Hoàng, V.H., Labarrière, H., Rondepierre, A.: Fast convergence of inertial dynamics with Hessian-driven damping under geometry assumptions. Appl. Math. Optim. **88**(3), 81 (2023)

[10] Aujol, J.F., Dossal, C., Rondepierre, A.: Optimal convergence rates for Nesterov acceleration. SIAM J. Optim. **29**(4), 3131–3153 (2019)

[11] Aujol, J.F., Dossal, C., Rondepierre, A.: Convergence rates of the heavy-ball method under the Lojasiewicz property. Math. Program. **198**(1), 195–254 (2023)

[12] Aujol, J.F., Dossal, C., Rondepierre, A.: FISTA is an automatic geometrically optimized algorithm for strongly convex functions. Math. Program. pp. 1–43 (2023)

[13] Ba, Q., Pang, J.S.: Exact penalization of generalized Nash equilibrium problems. Oper. Res. **70**(3), 1448–1464 (2022). DOI 10.1287/opre.2019.1942. URL `https://doi.org/10.1287/opre.2019.1942`

[14] Banert, S., Ringh, A., Adler, J., Karlsson, J., Öktem, O.: Data-driven nonsmooth optimization. SIAM J. Optim. **30**(1), 102–131 (2020)

[15] Banert, S., Rudzusika, J., Öktem, O., Adler, J.: Accelerated forward-backward optimization using deep learning. SIAM Journal on Optimization **34**(2), 1236–1263 (2024). DOI 10.1137/22M1532548. URL `https://doi.org/10.1137/22M1532548`

[16] Benaïm, M., Hofbauer, J., Sorin, S.: Stochastic approximations and differential inclusions. SIAM J. Control Optim. **44**(1), 328–348 (2005). DOI 10.1137/S0363012904439301. URL `https://doi.org/10.1137/S0363012904439301`

[17] Bolte, J., Le, T., Pauwels, E., Silveti-Falls, T.: Nonsmooth implicit differentiation for machine-learning and optimization. In: M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, J.W. Vaughan (eds.) Advances in Neural Information Processing Systems, vol. 34, pp. 13537–13549. Curran Associates, Inc. (2021)

[18] Bolte, J., Pauwels, E.: Conservative set valued fields, automatic differentiation, stochastic gradient methods and deep learning. Math. Program. **188**(1), 19–51 (2021)

[19] Chen, T., Chen, X., Chen, W., Wang, Z., Heaton, H., Liu, J., Yin, W.: Learning to optimize: a primer and a benchmark. J. Mach. Learn. Res. **23**, Paper No. [189], 59 (2022)

[20] Chen, X., Liang, C., Huang, D., Real, E., Wang, K., Pham, H., Dong, X., Luong, T., Hsieh, C., Lu, Y., Le, Q.V.: Symbolic discovery of optimization algorithms. In: A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, S. Levine (eds.) Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023 (2023). URL `http://papers.nips.cc/paper_files/paper/2023/hash/9a39b4925e35cf447ccba8757137d84f-Abstract-Conference.html`

[21] Chen, X., Liu, J., Wang, Z., Yin, W.: Theoretical linear convergence of unfolded ISTA and its practical weights and thresholds. In: S. Bengio, H.M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, R. Garnett (eds.) Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada, pp. 9079–9089 (2018). URL `https://proceedings.neurips.cc/paper/2018/hash/cf8c9be2a4508a24ae92c9d3d379131d-Abstract.html`

[22] Chen, Y., Hoffman, M.W., Colmenarejo, S.G., Denil, M., Lillicrap, T.P., Botvinick, M.M., de Freitas, N.: Learning to learn without gradient descent by gradient descent. In: D. Precup, Y.W. Teh (eds.) Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017, *Proceedings of Machine Learning Research*, vol. 70, pp. 748–756. PMLR (2017). URL `http://proceedings.mlr.press/v70/chen17e.html`

[23] Clarke, F.H.: Optimization and nonsmooth analysis, *Classics in Applied Mathematics*, vol. 5, second edn. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA (1990). DOI 10.1137/1.9781611971309. URL `https://doi.org/10.1137/1.9781611971309`

[24] Coste, M.: An introduction to o-minimal geometry. Istituti editoriali e poligrafici internazionali Pisa (2000)

[25] Cui, Y., Liu, J., Pang, J.S.: Nonconvex and nonsmooth approaches for affine chance-constrained stochastic programs. Set-Valued Var. Anal. **30**(3), 1149–1211 (2022). DOI 10.1007/s11228-022-00639-y. URL `https://doi.org/10.1007/s11228-022-00639-y`

[26] Cui, Y., Pang, J.S.: Modern nonconvex nondifferentiable optimization, *MOS-SIAM Series on Optimization*, vol. 29. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA; Mathematical Optimization Society, Philadelphia, PA (2022)

[27] Davis, D., Drusvyatskiy, D., Kakade, S., Lee, J.D.: Stochastic subgradient method converges on tame functions. Found. Comput. Math. **20**(1), 119–154 (2020)

[28] van den Dries, L.: Tame topology and o-minimal structures, *London Mathematical Society Lecture Note Series*, vol. 248. Cambridge University Press, Cambridge (1998)

[29] van den Dries, L., Miller, C.: Geometric categories and o-minimal structures. Duke Mathematical Journal **84**(2), 497 – 540 (1996)

[30] Dua, D., Graff, C.: UCI machine learning repository (2017)

[31] Filippov, A.F.: Differential equations with discontinuous righthand sides, *Mathematics and its Applications (Soviet Series)*, vol. 18. Kluwer Academic Publishers Group, Dordrecht (1988). Translated from the Russian

[32] Gärtner, E., Metz, L., Andriluka, M., Freeman, C.D., Sminchisescu, C.: Transformer-based learned optimization. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023, pp. 11970–11979. IEEE (2023). DOI 10.1109/CVPR52729.2023.01152. URL `https://doi.org/10.1109/CVPR52729.2023.01152`

[33] Golub, G.H., Van Loan, C.F.: Matrix computations, fourth edn. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, Baltimore, MD (2013)

[34] Gregor, K., LeCun, Y.: Learning fast approximations of sparse coding. In: J. Fürnkranz, T. Joachims (eds.) Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel, pp. 399–406. Omnipress (2010). URL `https://icml.cc/Conferences/2010/papers/449.pdf`

[35] Griewank, A., Walther, A.: Evaluating derivatives: Principles and techniques of algorithmic differentiation, second edn. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA (2008)

[36] Kamilov, U.S., Bouman, C.A., Buzzard, G.T., Wohlberg, B.: Plug-and-play methods for integrating physical and learned models in computational imaging: Theory, algorithms, and applications. IEEE Signal Processing Magazine **40**(1), 85–97 (2023). DOI 10.1109/MSP.2022.3199595

[37] Lambert, J.D.: Numerical methods for ordinary differential systems: The initial value problem. John Wiley & Sons, Ltd., Chichester (1991)

[38] Li, B., Shi, B., xiang Yuan, Y.: Linear convergence of forward-backward accelerated algorithms without knowledge of the modulus of strong convexity (2024)

[39] Li, K., Malik, J.: Learning to optimize. In: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. OpenReview.net (2017). URL https://openreview.net/forum?id=ry4Vrt5gl

[40] Lin, T., Jordan, M.I.: A control-theoretic perspective on optimal high-order optimization. Math. Program. **195**(1-2), 929–975 (2022)

[41] Liu, J., Chen, X., Wang, Z., Yin, W.: ALISTA: analytic weights are as good as learned weights in LISTA. In: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019. OpenReview.net (2019). URL https://openreview.net/forum?id=B1lnzn0ctQ

[42] Liu, J., Chen, X., Wang, Z., Yin, W., Cai, H.: Towards constituting mathematical structures for learning to optimize. In: A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, J. Scarlett (eds.) International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA, *Proceedings of Machine Learning Research*, vol. 202, pp. 21426–21449. PMLR (2023). URL https://proceedings.mlr.press/v202/liu23e.html

[43] Luo, H., Chen, L.: From differential equation solvers to accelerated first-order methods for convex optimization. Math. Program. **195**(1-2), 735–781 (2022)

[44] Marx, S., Pauwels, E.: Path differentiability of ODE flows. J. Differential Equations **338**, 321–351 (2022)

[45] Monga, V., Li, Y., Eldar, Y.C.: Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing. IEEE Signal Processing Magazine **38**(2), 18–44 (2021). DOI 10.1109/MSP.2020.3016905

[46] Moucer, C., Taylor, A., Bach, F.: A systematic approach to Lyapunov analyses of continuous-time models in convex optimization. SIAM J. Optim. **33**(3), 1558–1586 (2023)

[47] Nemirovsky, A.S., Yudin, D.B.: Problem complexity and method efficiency in optimization. Wiley-Interscience Series in Discrete Mathematics. John Wiley & Sons, Inc., New York (1983)

[48] Nesterov, Y.: Lectures on convex optimization, *Springer Optimization and Its Applications*, vol. 137, second edn. Springer, Cham (2018)

[49] Nesterov, Y.E.: A method for solving the convex programming problem with convergence rate $O(1/k^2)$. Dokl. Akad. Nauk SSSR **269**(3), 543–547 (1983)

[50] Platt, J.: Fast training of support vector machines using sequential minimal optimization. In: Advances in Kernel Methods - Support Vector Learning. MIT Press (1998)

[51] Rockafellar, R.T., Wets, R.J.B.: Variational analysis, *Grundlehren der mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]*, vol. 317. Springer-Verlag, Berlin (1998). DOI 10.1007/978-3-642-02431-3. URL https://doi.org/10.1007/978-3-642-02431-3

[52] Royden, H.L., Fitzpatrick, P.: Real analysis, vol. 2. Macmillan New York (1968)

[53] Shi, B., Du, S.S., Jordan, M.I., Su, W.J.: Understanding the acceleration phenomenon via high-resolution differential equations. Math. Program. **195**(1-2), 79–148 (2022)

[54] Stewart, G.W., Sun, J.G.: Matrix perturbation theory. Computer Science and Scientific Computing. Academic Press, Inc., Boston, MA (1990)

[55] Su, W., Boyd, S., Candès, E.J.: A differential equation for modeling Nesterov's accelerated gradient method: theory and insights. J. Mach. Learn. Res. **17**, Paper No. 153, 43 (2016)

[56] Vaquero, M., Cortes, J.: Convergence-rate-matching discretization of accelerated optimization flows through opportunistic state-triggered control. In: H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, R. Garnett (eds.) Advances in Neural Information Processing Systems, vol. 32. Curran Associates, Inc. (2019)

[57] Vaquero, M., Mestres, P., Cortés, J.: Resource-aware discretization of accelerated optimization flows: The heavy-ball dynamics case. IEEE Transactions on Automatic Control **68**(4), 2109–2124 (2023). DOI 10.1109/TAC.2022.3171307

[58] Whitney, H.: A function not constant on a connected set of critical points. Duke Mathematical Journal **1**(4), 514 – 517 (1935). DOI 10.1215/S0012-7094-35-00138-7. URL `https://doi.org/10.1215/S0012-7094-35-00138-7`

[59] Wibisono, A., Wilson, A.C., Jordan, M.I.: A variational perspective on accelerated methods in optimization. Proc. Natl. Acad. Sci. USA **113**(47), E7351–E7358 (2016)

[60] Wilkie, A.J.: Model completeness results for expansions of the ordered field of real numbers by restricted Pfaffian functions and the exponential function. J. Amer. Math. Soc. **9**(4), 1051–1094 (1996)

[61] Wilson, A.C., Mackey, L., Wibisono, A.: Accelerating rescaled gradient descent: Fast optimization of smooth functions. In: Advances in Neural Information Processing Systems, vol. 32. Curran Associates, Inc. (2019)

[62] Yang, Y., Sun, J., Li, H., Xu, Z.: Admm-csnet: A deep learning approach for image compressive sensing. IEEE Transactions on Pattern Analysis and Machine Intelligence **42**(3), 521–538 (2020). DOI 10.1109/TPAMI.2018.2883941

[63] Zhang, J., Ghanem, B.: Ista-net: Interpretable optimization-inspired deep network for image compressive sensing. In: 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018, pp. 1828–1837. Computer Vision Foundation / IEEE Computer Society (2018). DOI 10.1109/CVPR.2018.00196. URL `http://openaccess.thecvf.com/content_cvpr_2018/html/Zhang_ISTA-Net_Interpretable_Optimization-Inspired_CVPR_2018_paper.html`

[64] Zhang, J., He, T., Sra, S., Jadbabaie, A.: Why gradient clipping accelerates training: A theoretical justification for adaptivity. In: 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020. OpenReview.net (2020). URL `https://openreview.net/forum?id=BJgnXpVYwS`

[65] Zhang, J., Mokhtari, A., Sra, S., Jadbabaie, A.: Direct Runge-Kutta discretization achieves acceleration. In: Advances in Neural Information Processing Systems, vol. 31. Curran Associates, Inc. (2018)

[66] Zhang, J., Sra, S., Jadbabaie, A.: Acceleration in first order quasi-strongly convex optimization by ode discretization. In: 2019 IEEE 58th Conference on Decision and Control (CDC), pp. 1501–1506 (2019). DOI 10.1109/CDC40024.2019.9030046

[67] Zhang, P., Orvieto, A., Daneshmand, H., Hofmann, T., Smith, R.S.: Revisiting the role of euler numerical integration on acceleration and stability in convex optimization. In: A. Banerjee, K. Fukumizu (eds.) The 24th International Conference on Artificial Intelligence and Statistics, AISTATS 2021, April 13-15, 2021, Virtual Event, *Proceedings of Machine Learning Research*, vol. 130, pp. 3979–3987. PMLR (2021)

[68] Zheng, H., Yang, Z., Liu, W., Liang, J., Li, Y.: Improving deep neural networks using softplus units. In: 2015 International Joint Conference on Neural Networks (IJCNN), pp. 1–4. IEEE (2015)