

## 715 A Proof of Theorem 1

716 *Proof.* Consider a function  $G(\theta, x_0, t) = J(x(t; \theta, x_0)) - \epsilon$ . By the definition of  $T = T_J(\theta, x_0, \epsilon)$ ,  
 717 we have

$$G(\theta, x_0, T) = G(\theta, x_0, T_J(\theta, x_0, \epsilon)) = 0.$$

718 Computing the partial derivatives yields

$$\frac{\partial G}{\partial \theta} = \nabla J(x)^\top \frac{\partial x}{\partial \theta},$$

719 and

$$\frac{\partial G}{\partial t} = \nabla J(x)^\top \dot{x}(t).$$

720 Using the implicit function theorem, we conclude that  $T$  can be expressed locally as a continuously  
 721 differentiable function of  $\theta$  or  $x_0$ . We now differentiate  $G$  with respect to  $\theta$  and  $x_0$ , which yields

$$0 = \frac{d}{d\theta} (G(\theta, x_0, T)) = \frac{d}{d\theta} J(x(T(\theta, x_0, \epsilon); \theta, x_0)) = \nabla J(x)^\top \left( \frac{\partial x}{\partial \theta} + \dot{x}(T) \frac{\partial T}{\partial \theta} \right),$$

722 and

$$0 = \frac{d}{dx_0} (G(\theta, x_0, T)) = \frac{d}{dx_0} J(x(T(\theta, x_0, \epsilon); \theta, x_0)) = \nabla J(x)^\top \left( \frac{\partial x}{\partial x_0} + \dot{x}(T) \frac{\partial T}{\partial x_0} \right).$$

723 Rearranging these equations leads to

$$\nabla J(x)^\top \dot{x}(T) \frac{\partial T}{\partial \theta} = -\nabla J(x)^\top \frac{\partial x}{\partial \theta}, \quad \text{and hence} \quad \frac{\partial T}{\partial \theta} = (\nabla J(x)^\top \dot{x}(T))^{-1} \nabla J(x)^\top \frac{\partial x}{\partial \theta},$$

724 as well as

$$\nabla J(x)^\top \dot{x}(T) \frac{\partial T}{\partial x_0} = -\nabla J(x)^\top \frac{\partial x}{\partial x_0}, \quad \text{and hence} \quad \frac{\partial T}{\partial x_0} = (\nabla J(x)^\top \dot{x}(T))^{-1} \nabla J(x)^\top \frac{\partial x}{\partial x_0}.$$

725 The above equations complete the proof.  $\square$

## 726 B Proof of Theorem 2

727 We first present a basic analysis in numerical ODEs.

728 **Proposition 2** (Error analysis of the forward Euler method). *Let  $f : \mathbb{R}^n \times \mathbb{R} \rightarrow \mathbb{R}$  be a function*  
 729 *defined by  $(x, t) \mapsto f(x, t)$ . Suppose the following assumptions hold*

730 1. *There exists a constant  $L_x > 0$  such that  $\|f(x_1, t) - f(x_2, t)\| \leq L_x \|x_1 - x_2\|$  for all*  
 731  *$x_1, x_2$ , and  $t$ .*

732 2. *There exists a constant  $L_t > 0$  such that  $\|f(x, t_1) - f(x, t_2)\| \leq L_t |t_1 - t_2|$  for all  $x, t_1$ ,*  
 733 *and  $t_2$ .*

734 3. *There exists a constant  $M > 0$  such that  $\|f(x, t)\| < M$  for all  $x$  and  $t$ .*

735 *Given an initial condition  $x(t_0) = x_0$  and a fixed stepsize  $h$ , we consider the sequence generated by*  
 736 *the forward Euler method as*

$$x_{k+1} = x_k + hf(x_k, t_k), \quad t_k = t_0 + kh.$$

737 *Then, for any positive integer  $k$ , the error  $e_k = x_k - x(t_k)$  satisfies*

$$\|e_k\| \leq \frac{h}{2} \left( M + \frac{L_t}{L_x} \right) (e^{L_x h k} - 1).$$

738 *Proof.* We begin by expressing the error at step  $k + 1$  as

$$e_{k+1} = x_{k+1} - x(t_{k+1}) = e_k + h(f(x_k, t_k) - f(x(t_k), t_k)) + x(t_k) + hf(x(t_k), t_k) - x(t_{k+1}).$$

739 Applying Lipschitz continuity, we obtain the inequality

$$\|e_{k+1}\| \leq (1 + L_x h) \|e_k\| + \|x(t_k) + hf(x(t_k), t_k) - x(t_{k+1})\|.$$

740 The second term on the right-hand side can be expressed in integral form as

$$\|x(t_k) + hf(x(t_k), t_k) - x(t_{k+1})\| = \left\| \int_{t_k}^{t_{k+1}} [f(x(t), t) - f(x(t_k), t_k)] dt \right\|,$$

741 which is bounded above by

$$\left\| \int_{t_k}^{t_{k+1}} [f(x(t), t_k) - f(x(t_k), t_k)] dt \right\| + \left\| \int_{t_k}^{t_{k+1}} [f(x(t), t) - f(x(t), t_k)] dt \right\|.$$

742 Substituting the assumptions, we estimate the first integral as

$$\left\| \int_{t_k}^{t_{k+1}} [f(x(t), t_k) - f(x(t_k), t_k)] dt \right\| \leq L_x \int_{t_k}^{t_{k+1}} \|x(t) - x(t_k)\| dt \leq \frac{1}{2} M L_x h^2,$$

743 where the last inequality follows from the Lagrange mean value theorem, which implies that

$$\begin{aligned} \int_{t_k}^{t_{k+1}} \|x(t) - x(t_k)\| dt &= \int_{t_k}^{t_{k+1}} \|\dot{x}(\xi)\| |t - t_k| dt \\ &= \int_{t_k}^{t_{k+1}} \|f(x(\xi), \xi)\| |t - t_k| dt \\ &\leq M \int_{t_k}^{t_{k+1}} |t - t_k| dt = \frac{1}{2} M h^2. \end{aligned}$$

744 Similarly, for the second integral, we derive the bound as

$$\left\| \int_{t_k}^{t_{k+1}} [f(x(t), t) - f(x(t), t_k)] dt \right\| \leq L_t \int_{t_k}^{t_{k+1}} |t - t_k| dt = \frac{1}{2} L_t h^2.$$

745 Combining these inequalities, we obtain

$$\|e_{k+1}\| \leq (1 + L_x h) \|e_k\| + \frac{h^2}{2} (L_t + M L_x).$$

746 Finally, using the initial error  $e_0 = 0$ , we conclude that the global error satisfies

$$\|e_k\| \leq \frac{h}{2} \left( M + \frac{L_t}{L_x} \right) (e^{L_x h k} - 1).$$

747 This completes the proof.  $\square$

748 *Proof of the Theorem.* The following conditions are assumed throughout our analysis. First, the  
 749 function  $\mathcal{A}$  is twice continuously differentiable, i.e.,  $\mathcal{A} \in C^2$ . Second,  $\mathcal{A}$  itself, together with all  
 750 partial derivatives of  $\mathcal{A}$  (such as  $\frac{\partial}{\partial x} \mathcal{A}$ ,  $\frac{\partial^2 \mathcal{A}}{\partial \theta \partial t}$ ) and the gradient and Hessian of  $J$  (i.e.,  $\nabla J$  and  $\nabla^2 J$ ),  
 751 are uniformly bounded by constants  $A, A_x, A_\theta, A_t, A_{\theta,x}, A_{x,x}, A_{x,t}, A_{\theta,t}, J_1$ , and  $J_2$ , respectively.  
 752 Third, we assume the boundary condition  $|\nabla J(x(T))^\top \dot{x}(T)| = \delta$ .

753 For clarity and brevity, our main theorem states the regularity and boundedness assumptions using  
 754 Sobolev norms; specifically, we require that  $\mathcal{A}(\theta, x(t), t)$ , regarded as a function of  $(\theta, t)$ , has  
 755 uniformly bounded  $W^{2,\infty}$  norms with respect to  $(\theta, t)$  in a neighborhood of  $\theta \times [t_0, t_0 + N_J h]$ , and  
 756 that  $J$ , regarded as a function of  $x$ , has a uniformly bounded  $W^{2,\infty}$  norm in a neighborhood of  $x(T_J)$ .  
 757 In this proof, we equivalently expand these assumptions by explicitly introducing uniform bounds  
 758 for  $\mathcal{A}$ , its partial derivatives (with respect to  $x, \theta, t$ , etc.), and for the gradient  $\nabla J$  and Hessian  $\nabla^2 J$ ,  
 759 denoted by  $A, A_x, A_\theta, A_t, A_{\theta,x}, A_{x,x}, A_{x,t}, A_{\theta,t}, J_1$ , and  $J_2$ , respectively. This explicit formulation  
 760 is purely for notational convenience in the analysis, as it allows us to refer directly to these quantities

in the derivations, especially during Taylor expansions and error estimates. We emphasize that these detailed bounds can be derived from the  $W^{2,\infty}$  norm boundedness assumed in the theorem statement. Without loss of generality, we only prove the case for the  $L^2$  norm. We first recall the form and the definition of the derivative. They are given by

$$\begin{aligned}\nabla_\theta T &= \nabla_\theta T_J(\theta, x_0, \epsilon) = -\frac{\nabla J(x(T))^\top \frac{\partial x(T)}{\partial \theta}}{\nabla J(x(T))^\top \dot{x}(T)}, \\ \nabla_\theta N &= \nabla_\theta N_J(\theta, x_0, \epsilon) = -\frac{h \nabla J(x_N)^\top \frac{\partial x_N}{\partial \theta}}{J(x_N) - J(x_{N-1})}.\end{aligned}$$

We consider the iteration

$$x_{k+1} = x_k - h\mathcal{A}(\theta, x_k, t_k).$$

Differentiating with respect to  $\theta$ , we obtain

$$\frac{\partial x_{k+1}}{\partial \theta} = \left( I - h \frac{\partial}{\partial x} \mathcal{A}(\theta, x_k, t_k) \right) \frac{\partial x_k}{\partial \theta} - h \frac{\partial}{\partial \theta} \mathcal{A}(\theta, x_k, t_k),$$

where the initial condition  $\frac{\partial x_0}{\partial \theta}$  holds.

Also, we consider the flow

$$\dot{x}(t) = -\mathcal{A}(\theta, x(t), t).$$

Differentiating with respect to  $\theta$ , we obtain

$$\frac{d}{dt} \frac{\partial x(t)}{\partial \theta} = -\frac{\partial}{\partial \theta} \mathcal{A}(\theta, x(t), t) - \frac{\partial}{\partial x} \mathcal{A}(\theta, x(t), t) \frac{\partial x(t)}{\partial \theta}, \quad (16)$$

where the initial condition  $\frac{\partial x(t_0)}{\partial \theta} = 0$  holds. Let  $u(t) = \frac{\partial x(t)}{\partial \theta}$ . It is easy to observe that the iteration above corresponds to the forward Euler method for solving the ODE

$$\frac{d}{dt} u(t) = -\frac{\partial}{\partial \theta} \mathcal{A}(\theta, x(t), t) - \frac{\partial}{\partial x} \mathcal{A}(\theta, x(t), t) u(t).$$

We now proceed to show that  $u(t)$ , for  $t \in [t_0, T]$ , is bounded by some constant  $M > 0$ . Let  $v = u^\top u$ ,  $B(t) = -\frac{\partial}{\partial \theta} \mathcal{A}(\theta, x(t), t)$ , and  $C(t) = -\frac{\partial}{\partial x} \mathcal{A}(\theta, x(t), t)$ . Then we can derive that

$$\frac{d}{dt} v = 2u^\top \frac{d}{dt} u = 2u^\top B + 2u^\top C u.$$

Therefore, we have the bound

$$\left| \frac{d}{dt} v \right| \leq 2\|B\|\sqrt{v} + 2\|C\|v \leq \|B\| + (\|B\| + 2\|C\|)v \leq A_\theta + (A_\theta + 2A_x)v.$$

Applying the Gronwall inequality, for every  $t \in [t_0, T]$ , we obtain the following estimate

$$\|u(t)\| = \sqrt{v(t)} \leq \sqrt{\frac{A_\theta}{A_\theta + 2A_x} (e^{(A_\theta + 2A_x)(T-t_0)} - 1)} \triangleq M. \quad (17)$$

Employing Proposition 2, we obtain that  $\left\| \frac{\partial x_N}{\partial \theta} - \frac{\partial x(Nh)}{\partial \theta} \right\|$  is bounded by

$$\frac{h}{2} \left( MA_x + A_\theta + \frac{M(A_{x,t} + AA_{x,x}) + A_{\theta,t} + AA_{\theta,x}}{A_x} \right) (e^{A_x(T+1-t_0)} - 1).$$

Let

$$c_1 \triangleq \frac{1}{2} \left( MA_x + A_\theta + \frac{M(A_{x,t} + AA_{x,x}) + A_{\theta,t} + AA_{\theta,x}}{A_x} \right) (e^{A_x(T+1-t_0)} - 1). \quad (18)$$

Noticing that  $\frac{d}{dt} \frac{\partial x(t)}{\partial \theta}$  is bounded by  $A_\theta + A_x M$  according to (16), and that  $|T - Nh| \leq h$ , we deduce that

$$\left\| \frac{\partial x(Nh)}{\partial \theta} - \frac{\partial x(T)}{\partial \theta} \right\| \leq (A_\theta + A_x M)h.$$

781 Let  $e_1 = \frac{\partial x_N}{\partial \theta} - \frac{\partial x(T)}{\partial \theta}$ . Then we obtain the estimate

$$\|e_1\| \leq (A_\theta + A_x M + c_1)h. \quad (19)$$

782 Similarly, by Proposition 2, we know that

$$\|x_N - x(Nh)\| \leq \frac{h}{2} \left( A + \frac{A_t}{A_x} \right) \left( e^{A_x(T+1-t_0)} - 1 \right).$$

783 Let

$$c_2 \triangleq \frac{1}{2} \left( A + \frac{A_t}{A_x} \right) \left( e^{A_x(T+1-t_0)} - 1 \right). \quad (20)$$

784 Since  $\frac{d}{dt}x(t) = -\mathcal{A}(\theta, x(t), t)$  is bounded by  $A$  and  $|T - Nh| \leq h$ , it follows that

$$\|x_N - x(T)\| \leq (A + c_2)h.$$

785 Let  $e_2 = \nabla J(x_N) - \nabla J(x(T))$ . Since  $\|\nabla^2 J\|$  is bounded by  $J_2$ , we obtain the estimate

$$|e_2| \leq J_2(A + c_2)h. \quad (21)$$

786 The Taylor expansion yields

$$J(x_N) = J(x_{N-1}) - \nabla J(x_{N-1})^\top (x_N - x_{N-1}) + \frac{1}{2} (x_N - x_{N-1})^\top \nabla^2 J(\xi) (x_N - x_{N-1}).$$

787 Combining this with the fact that  $x_N - x_{N-1} = -h\mathcal{A}(\theta, x_{N-1}, t_{N-1})$  and that  $\|\nabla^2 J\|$  is bounded  
788 by  $J_2$ , we obtain

$$\left| \frac{J(x_N) - J(x_{N-1})}{h} + \nabla J(x_{N-1})^\top \mathcal{A}(\theta, x_{N-1}, t_{N-1}) \right| \leq \frac{1}{2} h A^2 J_2.$$

789 Let

$$e_5 = \frac{J(x_N) - J(x_{N-1})}{h} + \nabla J(x_{N-1})^\top \mathcal{A}(\theta, x_{N-1}, t_{N-1})$$

790 and  $e_3 = \nabla J(x_{N-1}) - \nabla J(x(T))$ ,  $e_4 = \mathcal{A}(\theta, x_{N-1}, t_{N-1}) + \dot{x}(T)$ . We have just derived

$$|e_5| \leq \frac{1}{2} h A^2 J_2 \quad (22)$$

791 As in the previous estimate for  $e_2$ , we obtain

$$\|e_3\| \leq J_2(A + c_2)h. \quad (23)$$

792 Furthermore, we have

$$\begin{aligned} \|e_4\| &\leq \|\mathcal{A}(\theta, x_{N-1}, t_{N-1}) - \mathcal{A}(\theta, x(T), t_{N-1})\| + \|\mathcal{A}(\theta, x(T), t_{N-1}) - \mathcal{A}(\theta, x(T), T)\| \\ &\leq A_x(A + c_2)h + A_t h. \end{aligned} \quad (24)$$

793 Substituting the definitions of these error terms into the expression for  $\nabla_\theta N$ , we obtain

$$\nabla_\theta N = - \frac{(\nabla J(x(T)) + e_2)^\top \left( \frac{\partial x(T)}{\partial \theta} + e_1 \right)}{(\nabla J(x(T)) + e_3)^\top (\dot{x}(T) - e_4) + e_5}.$$

794 Recall that

$$\nabla_\theta T = - \frac{\nabla J(x(T))^\top \frac{\partial x(T)}{\partial \theta}}{\nabla J(x(T))^\top \dot{x}(T)}.$$

795 Comparing these two expressions and combining the estimates from (19), (21), (23), and (24),  
796 together with the assumptions, we arrive at the final estimate that

$$\|\nabla_\theta T - \nabla_\theta N\| \leq Rh + \mathcal{O}(h^2),$$

797 where

$$\begin{aligned} R &= \frac{J_1 M}{\delta^2} \left( J_1(A_t + A_x(A + c_2)) + \frac{3}{2} A^2 J_2 + A J_2 c_2 \right) \\ &\quad + \frac{1}{\delta} (J_1(A_0 + A_x M + c_1) + M J_2(A + c_2)). \end{aligned}$$

798 Here, the constants refer to those defined in (17), (18), (20), and the assumptions stated earlier. This  
799 completes the proof.  $\square$

## C Proof of Proposition 1

*Proof.* We aim to compute  $S_{\theta_j} = \nabla J(x_N)^\top \frac{\partial x_N}{\partial \theta_j}$  for each component  $\theta_j$  of  $\theta$ , and  $S_{x_0} = \nabla J(x_N)^\top \frac{\partial x_N}{\partial x_0}$ . Let  $L(\theta, x_0) = J(x_N(\theta, x_0))$ . We are interested in  $\nabla_\theta L$  and  $\nabla_{x_0} L$ . Define the adjoint (co-state) vectors  $\lambda_k \in \mathbb{R}^d$  for  $k = 0, \dots, N$  such that  $\lambda_k^\top = \frac{\partial J(x_N)}{\partial x_k} = \nabla J(x_N)^\top \frac{\partial x_N}{\partial x_k}$ . The base case is at  $k = N$ ,

$$\lambda_N = \frac{\partial J(x_N)}{\partial x_N} = \nabla J(x_N). \quad (25)$$

For  $k < N$ ,  $x_N$  depends on  $x_k$  through  $x_{k+1}$ . Using the chain rule

$$\frac{\partial J(x_N)}{\partial x_k} = \frac{\partial J(x_N)}{\partial x_{k+1}} \frac{\partial x_{k+1}}{\partial x_k}.$$

In terms of our adjoints, we have

$$\lambda_k^\top = \lambda_{k+1}^\top \frac{\partial x_{k+1}}{\partial x_k}.$$

Given  $x_{k+1} = x_k - h\mathcal{A}(\theta, x_k, t_k)$ , the Jacobian is  $\frac{\partial x_{k+1}}{\partial x_k} = I - h \frac{\partial \mathcal{A}(\theta, x_k, t_k)}{\partial x_k}$ . Thus, the backward recursion for the adjoints is

$$\lambda_k^\top = \lambda_{k+1}^\top \left( I - h \frac{\partial \mathcal{A}(\theta, x_k, t_k)}{\partial x_k} \right), \quad (26)$$

or  $\lambda_k = \left( I - h \frac{\partial \mathcal{A}(\theta, x_k, t_k)}{\partial x_k} \right)^\top \lambda_{k+1}$ . The loop in Algorithm 1 implements this recursion. At the beginning of iteration  $k$  (loop index in algorithm, representing the step from  $x_k$  to  $x_{k+1}$ ), the variable  $\lambda$  in the algorithm holds  $\lambda_{k+1}$  from our derivation.

Now consider the derivative with respect to a parameter  $\theta_j$ .  $J(x_N)$  depends on  $\theta_j$  through all  $x_m$  for  $m \leq N$  where  $x_m$  is influenced by  $\theta_j$ . Hence,

$$\frac{\partial J(x_N)}{\partial \theta_j} = \sum_{m=0}^{N-1} \frac{\partial J(x_N)}{\partial x_{m+1}} \left( \frac{\partial x_{m+1}}{\partial \theta_j} \right)_{\text{explicit}},$$

where  $(\frac{\partial x_{m+1}}{\partial \theta_j})_{\text{explicit}}$  means differentiating  $x_{m+1} = x_m - h\mathcal{A}(\theta, x_m, t_m)$  with respect to  $\theta_j$  while holding  $x_m$  fixed

$$\left( \frac{\partial x_{m+1}}{\partial \theta_j} \right)_{\text{explicit}} = -h \frac{\partial \mathcal{A}(\theta, x_m, t_m)}{\partial \theta_j}.$$

Thus, it holds

$$\frac{\partial J(x_N)}{\partial \theta_j} = \sum_{m=0}^{N-1} \lambda_{m+1}^\top \left( -h \frac{\partial \mathcal{A}(\theta, x_m, t_m)}{\partial \theta_j} \right). \quad (27)$$

The loop runs from  $k = N - 1$  down to 0. For each  $k$  in the loop, the term added is  $-h \left( \frac{\partial \mathcal{A}(\theta, x_k, t_k)}{\partial \theta} \right)^\top \lambda_{k+1}$ . Summing these terms gives  $(\nabla J(x_N)^\top \frac{\partial x_N}{\partial \theta})_j$ .

Finally, for the sensitivity with respect to  $x_0$ ,

$$\frac{\partial J(x_N)}{\partial x_0} = \lambda_0^\top.$$

After the loop in Algorithm 1 finishes (i.e., after the iteration for  $k = 0$ ), the variable  $\lambda$  will have been updated using  $\lambda_1$  and  $\frac{\partial \mathcal{A}(\theta, x_0, t_0)}{\partial x_0}$ , thus holding  $\lambda_0$ .  $\square$

## D Details of Experiments

**Implementation Details.** We adopt the official implementation of [33]<sup>1</sup> for the online learning rate adaptation experiments, and the codebase from [34]<sup>2</sup> for L2O experiments. They all follow the

<sup>1</sup><https://github.com/udellgroup/hypergrad>

<sup>2</sup><https://github.com/xhchrn/MS4L2O>

MIT License as specified in their respective GitHub repositories. All experiments are conducted on a workstation running Ubuntu with a 12-core Intel Xeon Platinum 8458P CPU (2.7GHz, 44 threads), one NVIDIA RTX 4090 GPU with 24GB memory, and 60GB of RAM. We note that, for both experimental setups, we have made moderate modifications to the original implementations to better align with the goals of our study. However, as the focus of this work is to explore the potential applications of stopping time in optimization rather than to achieve state-of-the-art performance across all settings, we did not perform extensive hyperparameter tuning for the stopping time-based algorithms under different configurations. This choice may explain why our method does not reach SOTA performance in some scenarios.

**NFEs of different solvers.** Figure 5 shows that the NFE for an adaptive solver is mainly influenced by the stopping criterion. Since it does not accept a prespecified time step size, all of the statistics remain the same for different  $h$ .

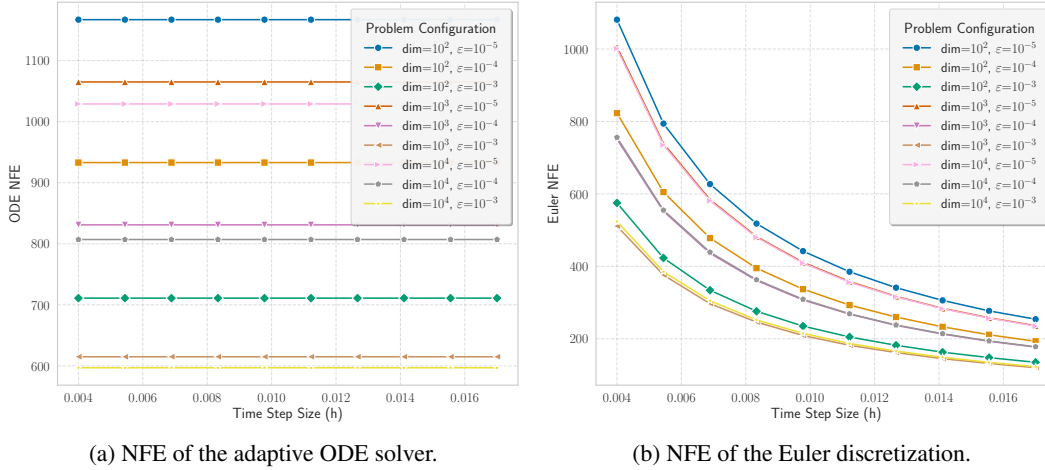


Figure 5: NFEs of different solvers.

836

**Hyperparameters of Baselines.** Adagrad is an adaptive gradient algorithm that adjusts learning rates per coordinate based on historical gradient information. The learning rate is set  $\beta \in \{10^{-3}, 10^{-2}, 10^{-1}, 1.0, 10.0, 1/L\}$  with  $\epsilon = 10^{-8}$ . For Heavy-Ball method (HB), the momentum parameter is selected from the set  $\{0.1, 0.5, 0.9, 1.0\}$ . Adam-HD is a notable variant of Adam [32], which employs a hypergradient-based scheme to adaptively update the base learning rate at each iteration in an online fashion. For Adam-HD, the hyperparameter  $\beta$  used to update the learning rate is chosen from the set  $\{10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$ . All other abbreviations follow their previously defined roles within the L2O framework. Adam-OLA and Adam-HD are all based on the classical Adam, where  $(\beta_1, \beta_2) = (0.9, 0.999)$  and  $\epsilon = 10^{-8}$ . The initial learning rate for Adam is selected from the set  $\alpha \in \{10^{-3}, 10^{-2}, 10^{-1}, 1.0, 10.0, 1/L\}$ .  $L$  is the Lipschitz constant of  $\nabla f(x)$ , estimated at the initial point  $x_0$ . The maximum number of iterations is set to 1000, with a stopping criterion tolerance of  $10^{-4}$ .

Table 1: Hyperparameter settings for Adam-OLA on different datasets. The parameter  $\beta$  controls the learning rate adaptation magnitude, and  $\epsilon$  specifies the sufficient decrease threshold for triggering a learning rate update.

Dataset (Experiment)	$\beta$ (Learning Rate Update)	$\epsilon$ (Descent Threshold)
a1a ( <i>exp_svm</i> )	$1 \times 10^{-2}$	$1 \times 10^{-5}$
a2a ( <i>exp_svm</i> )	$1 \times 10^{-3}$	$1 \times 10^{-3}$
a3a ( <i>exp_svm</i> )	$5 \times 10^{-5}$	$5 \times 10^{-4}$
w3a ( <i>exp_svm</i> )	0.005	$5 \times 10^{-9}$

**Formulation of the Smooth SVM.** In this work, we consider the problem of binary classification using a smooth variant of the SVM, where the non-smooth hinge loss is replaced by its squared

counterpart to enable efficient gradient-based optimization. Given a dataset  $\{(x_i, y_i)\}_{i=1}^n$  with feature vectors  $x_i \in \mathbb{R}^d$  and binary labels  $y_i \in \{-1, +1\}$ , the objective function takes the form

$$f(w) = \frac{1}{2} \sum_{i=1}^n [\max(0, 1 - y_i w^\top x_i)]^2 + \frac{\lambda}{2} \|w\|^2,$$

where  $\lambda > 0$  is a regularization parameter. This formulation preserves the margin-maximizing behavior of the original SVM while allowing for stable and differentiable optimization. We further incorporate an intercept term into the model by appending a constant feature to each input vector. The resulting problem is solved using first-order methods with step size determined via an estimate of the gradient’s Lipschitz constant.

**More Examples of Online Learning Rate Adaptation.** We report the performance of Algorithm 2 and other baseline methods. Our method shows consistent improvement in the later stage of the convergence.

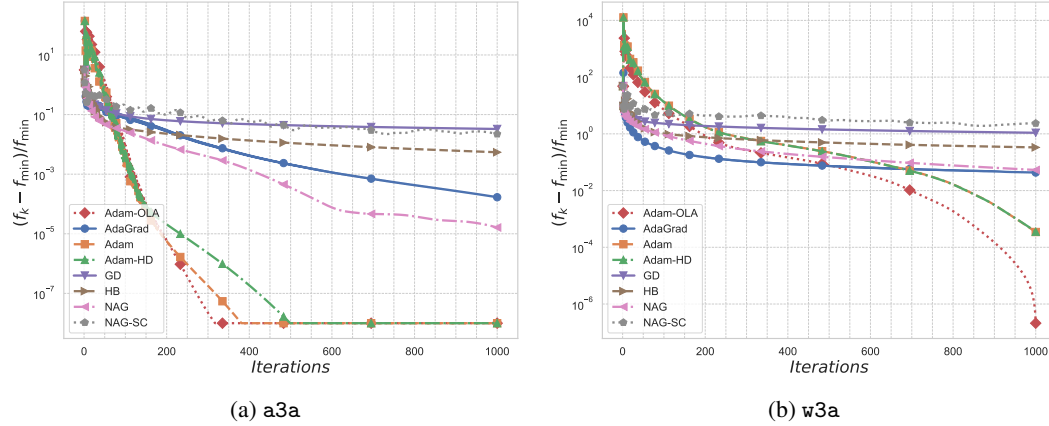


Figure 6: Comparison of different optimizers on smooth SVM: Function value versus iteration. Here,  $f_{\min}$  denotes the minimum function value achieved across all iterations for each optimizer.

**Data Synthetic Setting for L2O.** The data is synthetically generated. We first sample a sparse ground truth vector  $x^* \in \mathbb{R}^d$  with a prescribed sparsity level  $s$ , and then sample  $W \in \mathbb{R}^{n \times d}$  with standard normal entries. The binary labels are generated via

$$y_i = \mathbf{1}_{\{w_i^\top x^* \geq 0\}}, \quad i = 1, \dots, n.$$

A small proportion of labels are flipped to simulate noise.

**Architectures of L2O Optimizers.** We now provide two examples of learned optimizers formulated within this framework, drawing from seminal works in the field. These learned optimizers typically output a direct parameter update  $U_k$  such that  $x_{k+1} = x_k + U_k$ . To fit the continuous-time dynamical system framework where  $x_{k+1} = x_k - h\mathcal{A}(\mathbf{w}_{opt}, x_k, t_k)$ , we define  $\mathcal{A}(\mathbf{w}_{opt}, x_k, t_k) = -U_k/h$ . Here,  $\mathbf{w}_{opt}$  denotes the parameters of the learned optimizer itself,  $x_k$  are the parameters being optimized, and  $h$  is the discretization step size from the underlying ODE.

**LSTM-based Optimizer.** The influential work by Andrychowicz et al. [28] introduced an optimizer based on a Long Short-Term Memory (LSTM) network, which we denote as  $m_{\mathbf{w}_{opt}}$ . This optimizer operates coordinate-wise, meaning a small, shared-weight LSTM is applied to each parameter (coordinate) of the function  $f(x)$  being optimized. For each coordinate, the LSTM takes the corresponding component of the gradient  $\nabla f(x(t))$  and its own previous state,  $\text{state}(t)$ , as input to compute the parameter update component  $U(t) = m_{\mathbf{w}_{opt}}(\nabla f(x(t)), \text{state}(t))$ . The term  $\text{state}(t)$  for each coordinate’s LSTM, typically a multi-layer LSTM (e.g., two layers as used in the paper), consists of a tuple of (cell state, hidden state) pairs for each layer, i.e.,  $((c_{t,1}, h_{t,1}), (c_{t,2}, h_{t,2}))$  for a two-layer LSTM. These states allow the optimizer to accumulate information over the optimization trajectory, akin to momentum. The function  $\mathcal{A}$  is then defined as

$$\mathcal{A}(\mathbf{w}_{opt}, x(t), t) = -\frac{1}{h} m_{\mathbf{w}_{opt}}(\nabla f(x(t)), \text{state}(t)). \quad (28)$$

877 Here,  $\mathbf{w}_{opt}$  are the learnable weights of the shared LSTM optimizer.

878 **RNNprop Optimizer.** Building on similar principles, Lv et al. [29] proposed the RNNprop optimizer.  
879 This optimizer also typically uses a coordinate-wise multi-layer LSTM (e.g., two-layer) as its  
880 core recurrent unit. Before the gradient information  $\nabla f(x(t))$  is fed to the RNN, it undergoes  
881 a preprocessing step,  $\mathcal{P}$ . This preprocessing involves calculating Adam-like statistics, such as  
882 estimates of the first and second moments of the gradients,  $s(t) = (\hat{m}(t), \hat{v}(t))$ , which are then  
883 used to normalize the current gradient and provide historical context. The preprocessed features,  
884  $\mathcal{P}(\nabla f(x(t)), s(t))$ , along with the RNN’s previous state,  $\text{state}(t)$ , are input to the RNN. Similar  
885 to the LSTM-optimizer described above,  $\text{state}(t)$  for each coordinate’s RNN consists of the (cell  
886 state, hidden state) tuples for each of its layers. The output of the RNN is then passed through a  
887 scaled hyperbolic tangent function to produce the final update  $U(t)$ . Let this entire update-generating  
888 function be  $U_{\mathbf{w}_{opt}}(\nabla f(x(t)), s(t), \text{state}(t))$ . The corresponding  $\mathcal{A}$  function is

$$\mathcal{A}(\mathbf{w}_{opt}, x(t), t) = -\frac{1}{h} U_{\mathbf{w}_{opt}}(\nabla f(x(t)), s(t), \text{state}(t)), \quad (29)$$

889 where  $U_{\mathbf{w}_{opt}}(\cdot)$  can be more specifically written as  $\alpha \tanh(\text{RNN}(\mathcal{P}(\nabla f(x(t)), s(t)), \text{state}(t); \mathbf{w}_{opt}))$ .  
890 The parameters  $\mathbf{w}_{opt}$  encompass those for the preprocessing module  $\mathcal{P}$  and the RNN, and  $\alpha$  is a  
891 scaling hyperparameter.