Adversarial Tracking Control via Strongly Adaptive Online Learning with Memory

Zhiyu Zhang
Boston University

Ashok Cutkosky
Boston University

Ioannis Ch. Paschalidis
Boston University

Abstract

We consider the problem of tracking an adversarial state sequence in a linear dynamical system subject to adversarial disturbances and loss functions, generalizing earlier settings in the literature. To this end, we develop three techniques, each of independent interest. First, we propose a comparator-adaptive algorithm for *online linear optimization with movement cost*. Without tuning, it nearly matches the performance of the optimally tuned gradient descent in hindsight. Next, considering a related problem called *online learning with memory*, we construct a novel strongly adaptive algorithm that uses our first contribution as a building block. Finally, we present the first reduction from adversarial tracking control to strongly adaptive online learning with memory. Summarizing these individual techniques, we obtain an adversarial tracking controller with a strong performance guarantee even when the reference trajectory has a large range of movement.[1]

1 INTRODUCTION

Regulation and tracking are two iconic branches of linear control problems based on the system equation

$$x_{t+1} = A_t x_t + B_t u_t + w_t.$$

By designing the action u_t, a *regulation* controller rejects the disturbance w_t such that the state x_t remains close to the origin. In comparison, a *tracking* controller aims at steering the state x_t to follow a reference trajectory x^*_t. Recently, there have been growing efforts applying online learning ideas to linear control, including the online *Linear Quadratic Regulator* (LQR) (Abbasi-Yadkori and Szepesvári, 2011; Cohen et al., 2018; Dean et al., 2019), its adversarial generalizations (Agarwal et al., 2019a,b) and model-predictive control (Li et al., 2019; Yu et al., 2020). However, most of these advances are based on the regulation problem, and their application to tracking requires that the controller already knows the reference trajectory.

In this paper, we address this gap by first solving a general online learning problem: strongly adaptive online learning with movement cost. Ordinary (non-adaptive) algorithms aim to produce actions whose performance is strong on average over the entire operation of the algorithm. In contrast, a strongly-adaptive algorithm's performance must be strong *over any time interval* of operation. This additional requirement significantly complicates the algorithm design. In fact, standard approaches to achieving strong adaptivity fail to account for movement costs, and cannot be easily modified to incorporate this extra performance metric.

Subsequently, we come back to control and consider a general *adversarial tracking* problem with the following challenges.

1. The system dynamics (A_t, B_t) are time-varying.
2. The reference trajectory is fully adversarial. That is, x^*_t can freely adapt to past actions of the controller, and we do not impose any assumption on its movement speed $\|x^*_t - x^*_{t-1}\|$.
3. The loss function l^*_t that quantifies the tracking performance is adversarial, and we do not require its minimizer to be unique. This generalizes the quadratic loss from existing works on adversarial tracking, and allows the modeling of *target regions*.
4. The disturbance w_t is adversarial, possibly combining noise, modeling error and (minor) nonlinearity.

Such a setting is useful for many practical problems, especially when the target to be tracked is hard to

[1]Future versions available at `https://arxiv.org/abs/2102.01623`.

Proceedings of the 25th International Conference on Artificial Intelligence and Statistics (AISTATS) 2022, Valencia, Spain. PMLR: Volume 151. Copyright 2022 by the author(s).

model and predict. However, due to the confluence of these challenges, existing controllers either cannot be applied, or cannot produce a regret bound that competes with a strong enough baseline. Taking a conceptual leap, we will provide a solution by exploiting a novel connection between adversarial tracking control and strongly adaptive online learning.

1.1 Our contribution

In this paper, we develop three techniques, each using the previous one as its building block.

1. We propose the first comparator-adaptive algorithm for *Online Linear Optimization (OLO) with movement cost*. This is nontrivial as the per-step movement of existing comparator-adaptive OLO algorithms can be exponentially large in T. (Section 2)
2. We propose a novel strongly adaptive algorithm for *Online Convex Optimization with Memory (OCOM)*, and the obtained bound further adapts to the observed gradients. (Section 3)
3. We propose the first reduction from adversarial tracking control to strongly adaptive OCOM. Our approach establishes a connection between two separate notions of tracking from online learning and linear control, which could facilitate the application of online learning ideas in a wider range of control problems. (Section 4)

Combining these individual techniques, we design a strongly adaptive adversarial tracking controller: on any time interval $\mathcal{I}$ contained in the time horizon $[1:T]$, the proposed controller suffers $\tilde{O}(\sqrt{|\mathcal{I}|})$ regret against the best $\mathcal{I}$-dependent static controller, where $|\mathcal{I}|$ is the length of this time interval. More intuitively, on any time interval $\mathcal{I}$, the proposed controller always pursues *the best fixed action for* $\mathcal{I}$. Such a performance guarantee significantly improves existing results, especially when the reference trajectory has a large range of movement. Finally, our theoretical results are supported by experiments.

1.2 Background and notation

Linear tracking control Tracking control is a decades old problem in linear control theory. Despite the empirical success of heuristic approaches (e.g., the PID controller), classical theoretical analysis typically requires strong assumptions on the reference trajectory: either (i) the reference trajectory is generated by a known linear system (Astolfi, 2015); or (ii) predictions are available (Limon and Alamo, 2015).

For us, the most relevant works are the learning-based approaches with regret guarantees of the form

$$L(\text{alg}) - \min_{C\in\mathbb{C}} L(C) \leq \text{Regret bound}.$$

$\mathbb{C}$ is a set of baseline controllers called *comparator class*. $L(\text{alg})$ and $L(C)$ are the cumulative loss of the proposed algorithm and a comparator $C \in \mathbb{C}$, respectively. A strong guarantee requires not only a small regret bound, but also a comparator class that contains a good tracking baseline. From this perspective, we discuss the limitation of existing works as follows.

1. Abbasi-Yadkori et al. (2014); Foster and Simchowitz (2020) proposed algorithms for tracking fully adversarial targets, and a nonconstructive minimax guarantee was proposed by Bhatia and Sridharan (2020). However, regret bounds are only established on the entire time horizon $[1:T]$, and the comparator controllers are static and affine in the state ($u_t = -Kx_t + c$) which only perform well if the reference trajectory is roughly constant (on $[1:T]$).
2. Another line of research (Agarwal et al., 2019a,b; Simchowitz, 2020; Simchowitz et al., 2020; Minasyan et al., 2021) considered nonstochastic control, a general control setting with adversarial disturbances and loss functions. The comparator class is a collection of stabilizing linear controllers, therefore the implicitly assumed goal is *disturbance rejection* (i.e., regulation) rather than tracking. We will provide a detailed discussion in Section 4.2.

In summary, designing an adversarial tracking controller with a strong theoretical guarantee remains an open problem. Next, we review classical settings of online learning and a special *tracking* concept therein.

Basic online learning models There are two standard online learning models (Zinkevich, 2003) relevant to our purpose: *Online Convex Optimization* (OCO) and *Online Linear Optimization* (OLO). OCO is a two-person game: in each round, a player makes a prediction x_t in a convex set $\mathcal{V}$, observes a convex loss function l_t selected by an adversary and suffers the loss $l_t(x_t)$. If l_t is linear, then the problem is also called OLO. The standard performance metric is the static regret: $\text{Regret}_{[1:T]} = \sum_{t=1}^{T} l_t(x_t) - \min_{u\in\mathcal{V}} l_t(u)$. In general, OCO can be converted into OLO through the inequality $\text{Regret}_{[1:T]} \leq \max_{u\in\mathcal{V}} \sum_{t=1}^{T} \langle g_t, x_t - u\rangle$ where $g_t \in \partial l_t(x_t)$, so it suffices to only consider OLO.

Adaptive online learning In this paper, we call adaptivity the property of an OLO algorithm such that on any time interval $\mathcal{I} \subset [1:T]$, it guarantees *small* regret bound $\text{Regret}_{\mathcal{I}}$ against the best $\mathcal{I}$-dependent static comparator. Early works (Hazan and Seshadhri, 2009; Adamskiy et al., 2016) studied *weakly* *adaptive algorithms* where $\mathrm{Regret}_{\mathcal{I}} = \tilde{O}(\sqrt{T})$. Improving on those, recent advances (Daniely et al., 2015; Jun et al., 2017; Zhang et al., 2019a,b) focused on a more powerful concept called *strong adaptivity*: an algorithm is strongly adaptive if for all $\mathcal{I} \subset [1:T]$, $\mathrm{Regret}_{\mathcal{I}} = O(\mathrm{poly}(\log T) \cdot \sqrt{|\mathcal{I}|})$. This is much stronger than weak adaptivity, especially on short time intervals.

To associate adaptivity with adversarial tracking control, let us consider the *tracking regret* (Herbster and Warmuth, 1998; Bousquet and Warmuth, 2002) in online learning as an intermediate step, where an OLO algorithm is compared to all sequences with bounded amount of switching. This generalizes the static regret, and interestingly, Daniely et al. (2015) showed that near-optimal tracking regret can be derived from strong adaptivity. The key idea is that strongly adaptive OLO algorithms can quickly respond to the incoming losses, resulting in a near-optimal regret on the entire time horizon compared to *nonstationary comparators*. This bears an intriguing similarity to tracking *nonstationary targets* in linear control, which we exploit later.

As for the design of adaptive OLO algorithms, the predominant approach is a two-level composition pioneered by Hazan and Seshadhri (2009). Notably, Cutkosky (2020) proposed an alternative framework based on *comparator-adaptive* online learning (McMahan and Orabona, 2014; Orabona and Pál, 2016; Foster et al., 2018; van der Hoeven, 2019; Mhammedi and Koolen, 2020). Our construction will incorporate movement cost into the latter, which is a highly nontrivial task.

Strongly adaptive OCOM The performance of control suffers from past mistakes, therefore when we reduce it to online learning the resulting setting should also model this behavior, leading to a popular problem called *Online Convex Optimization with Memory* (OCOM) (Anava et al., 2015). A weakly adaptive OCOM algorithm was proposed in (Gradu et al., 2020), but achieving strong adaptivity is a much more challenging task due to two contradictory requirements: (i) strong adaptivity requires the predictions to move (i.e., respond to incoming information) very quickly; but (ii) movement cost requires the predictions to move slowly.

Recently, Daniely and Mansour (2019) proposed a strongly adaptive algorithm for OCOM with one-step memory, and its key component is an asymmetrical expert algorithm from Kapralov and Panigrahy (2010). In comparison, our approach (Contribution 2) is based on a fundamentally different mechanism and analysis. Our obtained bound adapts to the observed gradients, and more importantly provides an alternative line of intuition to the regret-movement trade-off in strongly adaptive online learning.

For conciseness, further discussion on existing works is deferred to Appendix A, including a series of related but incomparable works on *linear control with prediction*.

Notation We use $\|\cdot\|$ for the Euclidean norm of vectors and the spectral norm of matrices. These are the default norms throughout this paper. Let 0 be a zero vector or matrix. Let $\Pi_{\mathcal{V}}(x)$ be the Euclidean projection of x to a set $\mathcal{V}$. $\mathsf{B}^d(x, r)$ denotes the Euclidean norm ball centered at $x \in \mathbb{R}^d$ with radius r.

For two integers $a \leq b$, $[a:b]$ is the set of all integers c such that $a \leq c \leq b$; the brackets are removed when on the subscript, denoting a finite sequence with indices in $[a:b]$. Let $|\cdot|$ be the cardinality of a finite set. Given square matrices $M_{a:b}$, define their product as $\prod_{i=a}^{b} M_i = M_b \cdots M_a$. (When $b < a$, the product is the identity matrix.) Finally, log denotes natural logarithm when the base is omitted.

2 COMPARATOR-ADAPTIVE OLO WITH MOVEMENT COST

Starting with our first contribution, we introduce a comparator-adaptive algorithm for a variant of OLO called *OLO with movement cost*. The difference from standard OLO is that in each round, besides suffering the instantaneous loss $l_t(x_t)$, the player also suffers a movement cost $\lambda |x_t - x_{t-1}|$ where λ is a known constant. Movement penalties have been studied in online learning in various forms (Kalai and Vempala, 2005; Cesa-Bianchi et al., 2013; Gofer, 2014; Bhaskara et al., 2021; Sherman and Koren, 2021), sometimes under the name *switching cost* originated from the bandit problems. Since this paper focuses on the continuous domain, we name it as movement cost to avoid confusion. Notably, our setting is different from another classical problem called *Smoothed OCO* (Chen et al., 2018; Goel et al., 2019), where the loss function is observed *before* making the prediction.

Our algorithm is first developed on a one-dimensional domain $[0, \bar{R}]$, and then extended to higher dimensions.

2.1 The one-dimensional algorithm

We present the one-dimensional version in Algorithm 1. It critically relies on a duality between OLO and the *coin-betting game* (Orabona and Pál, 2016), which we summarize in Appendix B.1. Four hyperparameters are required: λ is the weight of movement costs, γ is a regularization weight, G is the Lipschitz constant (assumed known) of the OLO losses, and ε is the "budget" for the cumulative cost and movement.

To get the gist of this algorithm, let us briefly ignore the surrogate loss $\tilde{g}_t$ from Line 3 and the projection of $\tilde{x}_{t+1}$

from Line 6 (i.e., assume $\bar{R} = \infty$). With $g_t = \tilde{g}_t$ and $x_{t+1} = \tilde{x}_{t+1}$, Algorithm 1 becomes an unconstrained OLO algorithm with predictions recommended by the following betting scheme: A bettor has money Wealth_t in the t-th round. After choosing a betting fraction β_{t+1}, he bets money $x_{t+1} = \beta_{t+1}\mathrm{Wealth}_t$ on the next loss gradient g_{t+1}. The favorable outcome is $g_{t+1}x_{t+1}$ being negative which means the OLO algorithm suffers *negative loss*. Therefore, after observing g_{t+1}, the bettor treats $-g_{t+1}x_{t+1}$ as the money he gains and updates his wealth accordingly. Since large movement is also undesirable, the bettor further loses money proportional to the change of his betting amount; this is an important and novel step in our approach. Using this procedure, regret minimization is converted to wealth maximization. By choosing the betting fraction β_t properly, one can simultaneously ensure low cost and low movement in OLO.

Theorem 1. *For all $\lambda, \gamma \geq 0$, $G > 0$ and $0 < \varepsilon \leq G\bar{R}$, with any loss sequence such that $|g_t| \leq G$ for all t, applying Algorithm 1 yields the following guarantee.*

1. *For all $T \in \mathbb{N}_+$ and $u \in \mathcal{V}_{1d}$, with C defined in Line 1 of the algorithm,*

$$\sum_{t=1}^{T}\left(g_t x_t - g_t u + \lambda\left|x_t - x_{t+1}\right| + \frac{\gamma}{\sqrt{t}}\left|x_t\right|\right) \leq \varepsilon + uC\sqrt{2T}\left(\frac{3}{2} + \log\frac{\sqrt{2}uCT^{5/2}}{\varepsilon}\right).$$

2. *For all $a \leq b$, $\sum_{t=a}^{b}\left|x_t - x_{t+1}\right| \leq 48\bar{R}\sqrt{b-a+1}$.*

The highlights of Theorem 1 are the following.

1. Part 1 provides the first comparator-adaptive bound for OLO with movement cost: the sum of movement cost and regret with respect to the null comparator $u = 0$ is at most a user-specified constant, and the sum grows almost linearly in $|u|$ which is the optimal rate (Orabona, 2020, Chapter 5). This leads to an important *parameter-free* property: *without knowing the optimal comparator u^* in advance*, Algorithm 1 automatically adapts to it, and the performance bound almost matches the optimally-tuned *Online Gradient Descent* (OGD) whose learning rate depends on u^*. Note that the latter is a hypothetical (unimplementable) baseline, since the optimal comparator u^* in hindsight is unknown before all the losses are revealed. Nonetheless, our algorithm is still able to (nearly) match it using a perfectly implementable procedure.

   Furthermore, Part 1 does not need a bounded domain; the same bound holds even with $\bar{R} = \infty$, making Algorithm 1 an appealing approach for general unconstrained settings as well.

Algorithm 1 One-dimensional comparator-adaptive OLO with movement cost.

```
Require: Hyperparameters $(\lambda, \gamma, \varepsilon, G)$, with $\lambda, \gamma \geq 0$ and $\varepsilon, G > 0$; a 1-dimensional domain $\mathcal{V}_{1d} = [0, \bar{R}]$; loss gradients $g_1, g_2, \ldots \in \mathbb{R}$ with $|g_t| \leq G$, $\forall t$.
1: Initialize internal variables as $\mathrm{Wealth}_0 = \varepsilon$, and $\beta_1, x_1, \tilde{x}_1 = 0$. Define $C = G + \lambda + \gamma$.
2: for $t = 1, 2, \ldots$ do
3:     Make a prediction $x_t$, observe a loss gradient $g_t$. Define the surrogate loss $\tilde{g}_t$ as
           $\tilde{g}_t = \begin{cases} g_t, & \text{if } g_t\tilde{x}_t \geq g_t x_t, \\ 0, & \text{otherwise.} \end{cases}$
4:     Let $\hat{\beta}_{t+1} = -\sum_{i=1}^{t}\tilde{g}_i/(2C^2 t)$. Define $\mathcal{B}_{t+1} = [0, 1/(C\sqrt{2t})]$ and let $\beta_{t+1} = \Pi_{\mathcal{B}_{t+1}}(\hat{\beta}_{t+1})$.
5:     Assign $\mathrm{Wealth}_t$ as the solution to the following equation (uniqueness shown in Lemma B.2),
           $\mathrm{Wealth}_t = (1 - \tilde{g}_t\beta_t - \gamma\beta_t/\sqrt{t})\mathrm{Wealth}_{t-1} - \lambda|\beta_t\mathrm{Wealth}_{t-1} - \beta_{t+1}\mathrm{Wealth}_t|.$   (1)
6:     Let $\tilde{x}_{t+1} = \beta_{t+1}\mathrm{Wealth}_t$ and $x_{t+1} = \Pi_{\mathcal{V}_{1d}}(\tilde{x}_{t+1})$.
7: end for
```

2. As for Part 2, we bound the movement cost alone over *any* time interval, which is also technically nontrivial. Our surrogate loss $\tilde{g}_t$ (Line 3) is due to an existing black-box reduction from unconstrained OLO to constrained OLO (see Appendix B.2). However, the proof of Part 2 requires a *non-black-box* use of this procedure: we investigate how using the surrogate loss $\tilde{g}_t$ instead of the true loss g_t changes the growth rate of Wealth_t, an *internal* quantity of the unconstrained OLO algorithm. To the best of our knowledge, this is the first analysis that takes this perspective. The revealed insights could be of separate interest.

2.2 Extension to higher dimensions

After the one-dimensional analysis, we present Algorithm 2, which extends Algorithm 1 to a higher dimensional ball $\mathsf{B}^d(0, \bar{R})$ via a polar decomposition. Intuitively, Algorithm 2 learns the direction and magnitude separately: the former via standard OGD on the unit norm ball, and the latter via Algorithm 1. Such an idea was first proposed by Cutkosky and Orabona (2018); here we further incorporate movement cost into its analysis. The performance guarantee has a similar flavor as Theorem 1; for conciseness, we defer it to Appendix B.4.

Algorithm 2 Extension of Algorithm 1 to $\mathsf{B}^d(0, \bar{R})$.

```
Require: Hyperparameters $(\lambda, \varepsilon, G)$ with $\lambda \geq 0$ and $\varepsilon, G > 0$; $g_1, g_2, \ldots \in \mathbb{R}^d$ with $\|g_t\| \leq G$, $\forall t$.
1: Define $\mathcal{A}_r$ as Algorithm 1 on the domain $[0, \bar{R}]$, with hyperparameters $(\lambda, \lambda, \varepsilon, G)$.
2: Define $\mathcal{A}_B$ as Online Gradient Descent (OGD) on $\mathsf{B}^d(0,1)$ with learning rate $\eta_t = 1/(G\sqrt{t})$, initialized at the origin 0.
3: for $t = 1, 2, \ldots$ do
4:     Obtain $y_t \in \mathbb{R}$ from $\mathcal{A}_r$ and $z_t \in \mathbb{R}^d$ from $\mathcal{A}_B$. Predict $x_t = y_t z_t \in \mathbb{R}^d$, observe $g_t \in \mathbb{R}^d$.
5:     Return $\langle g_t, z_t \rangle$ and $g_t$ as the $t$-th loss gradient to $\mathcal{A}_r$ and $\mathcal{A}_B$, respectively.
6: end for
```

3 STRONGLY ADAPTIVE OCOM

Next, we introduce our second contribution - a novel strongly adaptive algorithm for *Online Convex Optimization with Memory* (OCOM) (Anava et al., 2015). After introducing the problem setting, we present our approach step-by-step which builds on Algorithms 1 and 2.

3.1 Problem setting of OCOM

Consider a convex and compact domain $\mathcal{V} \subset \mathsf{B}^d(0, R)$ with $R > 0$. Without loss of generality, assume $\mathcal{V}$ contains the origin 0.[2] In each round, a player makes a prediction $x_t \in \mathcal{V}$, observes a loss function $l_t : \mathcal{V}^{H+1} \to \mathbb{R}$ and suffers the loss $l_t(x_{t-H}, \ldots, x_t)$ that depends on the H-round prediction history. For all $t \leq 0$, $x_t = 0$.

We define an instantaneous loss function as $\tilde{l}_t(x) = l_t(x, \ldots, x)$. Two assumptions are imposed: (i) l_t is L-Lipschitz with respect to each argument separately; (ii) $\tilde{l}_t(x)$ is convex and $\tilde{G}$-Lipschitz, with $0 < \tilde{G} \leq L(H+1)$.

For this OCOM problem, our goal is a strongly adaptive regret bound on the policy regret: for *all* time intervals $\mathcal{I} = [a : b] \subset [1 : T]$,

$$\sum_{t=a}^{b} l_t(x_{t-H:t}) - \min_{x \in \mathcal{V}} \sum_{t=a}^{b} \tilde{l}_t(x) = O\left(\text{poly}(\log T) \cdot \sqrt{|\mathcal{I}|}\right), \quad (2)$$

where $O(\cdot)$ subsumes polynomial factors on the problem constants. In other words, on any time interval $\mathcal{I} \subset [1 : T]$, the regret compared to the best $\mathcal{I}$*-dependent* fixed prediction should be $\tilde{O}(\sqrt{|\mathcal{I}|})$.

[2]By shifting the coordinate system, this can be achieved for any nonempty set $\mathcal{V}$.

3.2 Preliminary: GC intervals

First of all, we review an important concept. Similar to achieving strong adaptivity without memory (Daniely et al., 2015; Cutkosky, 2020), our OCOM algorithm has a hierarchical structure. It maintains a subroutine on each *Geometric-Covering* (GC) interval, and the overall prediction combines the outputs from all the active subroutines. Such a structure benefits from a nice property (Daniely et al., 2015): an online learning algorithm is strongly adaptive if it has the desirable strongly adaptive guarantee *on all the GC intervals*. Consequently for our objective (2), we can only focus on achieving this bound on GC intervals instead of general intervals $\mathcal{I} \subset [1 : T]$.

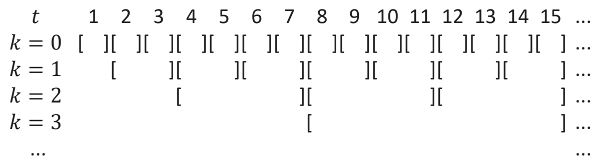


Figure 1: Geometric-Covering intervals.

The class of GC intervals is visualized in Figure 1. Concretely, for all $k \in \mathbb{N}$ and $i \in \mathbb{N}_+$, a GC interval is defined as $\mathcal{I}^{k,i} = [2^k i : 2^k(i+1) - 1]$. If it contains t, then we say it is active in the t-th round.

3.3 Subroutine on GC intervals

The next step is to construct the subroutine on each GC interval. It consists of two parts:

1. *Subroutine-1d*, an OLO algorithm operating on the one-dimensional domain $[0, 1]$.
2. *Subroutine-ball*, an OLO algorithm operating on the ball $\mathsf{B}^d(0, R)$ that contains $\mathcal{V}$.

Intuitively, each Subroutine-1d produces the "confidence" on its corresponding Subroutine-ball. Then, the Subroutine-ball with higher confidence contributes a larger portion in the prediction of the meta-algorithm. Algorithms 1 and 2 constitute the basis of these two parts respectively, but we need one extra step (Algorithm 3): Subroutine-1d is the version of Algorithm 3 with Line 1(a) and $g_t \in \mathbb{R}$, while Subroutine-ball is the version with Line 1(b) and $g_t \in \mathbb{R}^d$. Note that the time index t in the pseudo-code represents the local clock counting from the start of the considered GC interval. That is, if we consider $\mathcal{I}^{k,i}$ starting from the $2^k i$-th round, then the index t in Algorithm 3 represents the $(2^k i - 1 + t)$-th round globally.

Algorithm 3 serves two purposes: (i) improving the dependence on hyperparameters G and λ (ultimately,

Algorithm 3 Subroutine on GC intervals.

```
Require: Hyperparameters $(\lambda, \varepsilon, G)$ with $\lambda \geq 0$ and $\varepsilon, G > 0$; gradients $g_1, g_2, \ldots$, with $\|g_t\| \leq G$, $\forall t$.
1: (a) Subroutine-1d: Define $\mathcal{A}$ as Algorithm 1 with hyperparameters $(\lambda, 0, \varepsilon, \max\{\lambda, G\} + G)$, on the domain $[0,1] \subset \mathbb{R}$.
   (b) Subroutine-ball: Define $\mathcal{A}$ as Algorithm 2 with hyperparameters $(\lambda, \varepsilon, \max\{\lambda, G\} + G)$, on the domain $\mathsf{B}^d(0, R)$.
2: Initialize $i = 1$ and an accumulator $Z_i = 0$. Query the first output of $\mathcal{A}$ and assign it to $w_i$.
3: for $t = 1, 2, \ldots$ do
4:     Predict $x_t \leftarrow w_i$, observe $g_t$, let $Z_i \leftarrow Z_i + g_t$.
5:     if $\|Z_i\| > \max\{\lambda, G\}$ then
6:         Send $Z_i$ to $\mathcal{A}$ as the $i$-th loss. Let $i \leftarrow i + 1$.
7:         Set $Z_i = 0$. Query the $i$-th output of $\mathcal{A}$ and assign it to $w_i$.
8:     end if
9: end for
```

problem constants of OCOM); and (ii) achieving adaptivity to the observed gradients, which leads to better practical performance. Its key mechanism is to *adaptively "slow down"* the base algorithm $\mathcal{A}$. To this end, an accumulator Z_i tracks the sum of the received loss gradients. The base algorithm $\mathcal{A}$ is only queried when Z_i exceeds a threshold $\max\{\lambda, G\}$. Using this procedure, we essentially replace the time horizon T in the performance guarantee of $\mathcal{A}$ by an adaptive quantity $\sum_{t=1}^{T} \|g_t\| / \max\{\lambda, G\}$.

3.4 Meta-algorithm

Given the two-part subroutine, we now introduce our OCOM meta-algorithm. Compared to online learning without memory (Cutkosky, 2020), our technical improvement is the incorporation of movement cost which is a nontrivial task. The complete pseudo-code is deferred to Appendix C.2, and an abridged version (Algorithm 4) is provided here. Specifically, Algorithm 4 simplifies a complicated projection scheme by allowing improper predictions ($x_t \notin \mathcal{V}$).

In each round, Algorithm 4 combines the subroutines by recursively running Line 6. Such a procedure is different from the well-known *boosting* strategy (Freund and Schapire, 1997; Beygelzimer et al., 2015) applied in (Daniely and Mansour, 2019), as the updated temporary prediction $x_t^{(k)}$ is *not a convex combination* of the old temporary prediction $x_t^{(k+1)}$ and the output $w_t^{(k)}$ from Subroutine-ball. By plugging the comparator-adaptive property of the subroutines into Line 6, Algorithm 4 achieves an important property: for all k, $x_t^{(k)}$ matches the performance of $w_t^{(k)}$ on time

Algorithm 4 The OCOM meta-algorithm. (Abridged from Algorithm 7)

```
Require: $T \geq 1$; a hyperparameter $\varepsilon_0 > 0$.
1: for $t = 1, \ldots, T$ do
2:     Find the $(k, i)$ index pair for all the GC intervals that start in the $t$-th round. For each pair, initialize $\mathcal{A}_B^k$ as a copy of Subroutine-ball and $\mathcal{A}_{1d}^k$ as a copy of Subroutine-1d, with *some* hyperparameters that depend on $k$, $\varepsilon_0$ and problem constants. If $\mathcal{A}_B^k$ and $\mathcal{A}_{1d}^k$ already exist in the memory, overwrite them.
3:     Define $K_t = \lceil \log_2(t+1) \rceil - 1$; $x_t^{(K_t+1)} = 0 \in \mathbb{R}^d$.
4:     for $k = K_t, \ldots, 0$ do
5:         Query a prediction from $\mathcal{A}_B^k$ and assign it to $w_t^{(k)}$; query a prediction from $\mathcal{A}_{1d}^k$ and assign it to $z_t^{(k)}$.
6:         Let $x_t^{(k)} = (1 - z_t^{(k)}) x_t^{(k+1)} + w_t^{(k)}$.
7:     end for
8:     Predict $x_t = x_t^{(0)}$, suffer $l_t(x_{t-H:t})$, receive $l_t$, obtain a subgradient $g_t \in \partial \tilde{l}_t(x_t)$.
9:     for $k = 0, \ldots, K_t$ do
10:        Return $g_t$ to $\mathcal{A}_B^k$ and $-\langle g_t, x_t^{(k+1)} \rangle$ to $\mathcal{A}_{1d}^k$ as the loss gradients respectively.
11:    end for
12: end for
```

intervals of length 2^k while achieving the performance of $x_t^{(k+1)}$ on longer time intervals.[3] As the result, the final prediction $x_t^{(0)}$ matches the performance of *any* subroutine on its corresponding GC interval.

To recap, we demonstrate the structure of our OCOM algorithm in Figure 2. Collecting all the pieces, we state the performance guarantee in Theorem 2.

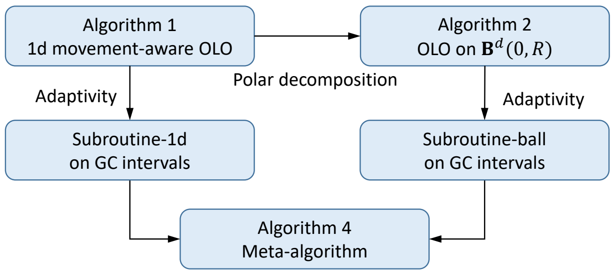


Figure 2: An overview of our OCOM strategy.

Theorem 2. *Consider running our OCOM algorithm (the complete version, Algorithm 7) for T rounds. If $\varepsilon_0 = \tilde{G}R/(T+1)$, then on any time interval $\mathcal{I} = [a :$*

[3]Compared to (Daniely and Mansour, 2019), this intuitively generalizes the "easy-to-combine" idea from expert problems to OLO.

$b] \subset [1:T]$,

$$\sum_{t=a}^{b} l_t(x_{t-H:t}) - \min_{x\in\mathcal{V}} \sum_{t=a}^{b} \tilde{l}_t(x)$$
$$= O(RLH^3 \log|\mathcal{I}|) + \tilde{O}\left(RLH^2 + RH\sqrt{L\sum_{t=a}^{b}\|g_t\|}\right),$$

where $g_t \in \partial\tilde{l}_t(x_t)$, $O(\cdot)$ subsumes absolute constants, and $\tilde{O}(\cdot)$ subsumes poly-logarithmic factors on problem constants and T.

Notice that the obtained bound is not only strongly adaptive according to Equation (2), but also adaptive to the observed gradients. In easy environments, it would be a lot better than $\tilde{O}(\sqrt{|\mathcal{I}|})$.

Remark 3.1. *Strongly adaptive regret is not the only performance metric that compares to dynamic comparators; alternatives include dynamic regret and competitive ratio (see Appendix A for an overview). If we have an algorithm $\mathcal{A}$ with such (alternative) guarantees on $[1:T]$ and a slow-moving property similar to Part 2 of Theorem 1, then we can assign the prediction of $\mathcal{A}$ to $x_t^{(K_t+1)}$. The resulting algorithm would not only remain strongly adaptive, but also essentially achieve the dynamic regret or competitive ratio guarantee of $\mathcal{A}$ on $[1:T]$.*

4 ADVERSARIAL TRACKING CONTROL

Finally we present our third contribution: a reduction from adversarial tracking control to strongly adaptive OCOM. Let us start with the problem setting.

4.1 Problem setting of adversarial tracking

We consider a time-varying linear system

$$x_{t+1} = A_t x_t + B_t u_t + w_t.$$

Matrices $A_t \in \mathbb{R}^{d_x\times d_x}$ and $B_t \in \mathbb{R}^{d_x\times d_u}$ are known. For all $t \le 0$, $x_t = 0$, $u_t = 0$; for all $t < 0$, $w_t = 0$.

The system has the following interaction protocol. At the beginning of the t-th round, after observing x_t, the controller commits to an action u_t. Then, an adversary selects the disturbance w_t, a reference state-action pair (x_t^*, u_t^*) and a loss function l_t, possibly depending on past controller actions $u_1, \dots, u_t$. (x_t^*, u_t^*) and l_t together induce a tracking loss function $l_t^*(x, u|x_t^*, u_t^*) := l_t(x - x_t^*, u - u_t^*)$ for all $(x, u) \in \mathbb{R}^{d_x} \times \mathbb{R}^{d_u}$, which is revealed to the controller and incurs a loss $l_t^*(x_t, u_t)$. After that, the state evolves to x_{t+1} following the system equation. Intuitively, l_t represents the shape of the loss function and (x_t^*, u_t^*) is the location parameter; an example is the quadratic control problem with $l_t(x, u) = \|x\|^2 + \|u\|^2$.

Our goal is a strongly adaptive tracking guarantee with the following shape: on any time interval $\mathcal{I}$ contained in the time horizon $[1:T]$, for all action sequences $u_{1:T}^C$ that are fixed on $\mathcal{I}$,

$$\sum_{t\in\mathcal{I}} l_t^*(x_t, u_t)\Big|_{\text{our algorithm}} - \sum_{t\in\mathcal{I}} l_t^*(x_t^C, u_t^C)\Big|_{\text{induced by } u_{1:T}^C} = \tilde{O}\left(\sqrt{|\mathcal{I}|}\right).$$

Such a guarantee subsumes the conventional static regret bound as one can choose $\mathcal{I} = [1:T]$. Moreover, the key benefit is that on any time interval $\mathcal{I}$, the optimal comparator is optimized for $\mathcal{I}$ instead of the entire time horizon $[1:T]$. From this perspective, we aim at a considerably stronger goal than existing works (Abbasi-Yadkori et al., 2014; Foster and Simchowitz, 2020).

For our setting, we impose the following assumptions. κ, γ, U and L^* are assumed to be known.

Assumption 1 (On the system)**.** *There exist $\kappa \ge 1$ and $U, W, \gamma > 0$ such that for all t, $\|B_t\| \le \kappa$, $\|u_t\| \le U$, $\|w_t\| \le W$ and $\|A_t\| \le 1-\gamma$.*

Assumption 2 (On the losses)**.** *For all t, l_t^* is convex. In addition, $l_t^*(x, u)$ is L^*-Lipschitz with respect to each argument separately, on the set $\{(x, u); \|x\| \le \gamma^{-1}(\kappa U + W), \|u\| \le U\}$.*

Remark 4.1. *The assumption $\|A_t\| \le 1-\gamma$ may seem restrictive as many real world systems are not open-loop stable. However, such an assumption allows a simplified exposition without excessively altering the essence of the problem. For general (open-loop unstable) systems, we can assume oracle stabilizing controllers (matrices) $K_{1:\infty}$ such that $\|\prod_{t=s}^{s+k}(A_t + B_t K_t)\| \le const \cdot (1-\gamma)^k$ for all s and k, and the multiplying constant can be larger than 1. Such an extension is somewhat standard in the analysis of linear time-varying systems (Minasyan et al., 2021, Appendix A.2). Given $K_{1:\infty}$, we can replace our action u_t with $K_t x_t + u_t$ so that a similar analysis follows.*

4.2 Difference with nonstochastic regulation

Before proceeding, we (re)-emphasize the difference between our work and a series of nonstochastic regulation controllers (most notably, Agarwal et al. 2019a). For a clear comparison, consider time-invariant dynamics ($A_t = A$, $B_t = B$) and state-tracking (l_t^* only depends on x_t). The procedure of (Agarwal et al., 2019a) can be summarized as follows.

1. Before observing any data, the controller computes a stabilizing feedback matrix K based on (A, B).
2. The actions are determined by a specific parameterization called *Disturbance-Action Controller* (DAC):
$$u_t = -Kx_t + \sum_{i=1}^{H} M_t^{[i]} w_{t-i},$$
where H is a constant, w_{t-i} is a past disturbance, and $M_t^{[1]}, \dots, M_t^{[H]}$ are parameter matrices updated via online gradient descent. The idea is to stabilize the system by $-Kx_t$, and adapt to the disturbances by applying their linear combinations.
3. It can be shown that the DAC class approximates a class of stabilizing linear controllers, therefore the regret guarantee can be stated with respect to the latter (as the comparator class).

Such an approach works well for the regulation problem, but in tracking it has a substantial limitation. Consider a simple example: what if the system is disturbance-free? In that case, the controller reduces to a static linear feedback, and the gain matrix is determined without seeing any data. In other words, *nothing is learned*. The state sequence would converge to the origin, therefore the tracking loss can be always high as long as the target state x_t^* is far away from the origin.

If x_t^* is known a priori, there is a standard remedy (Yu et al., 2020, Section 2): define a shifted state $\tilde{x}_t$ as the tracking error $x_t - x_t^*$ and apply the DAC on the shifted system to determine u_t. However, this is not applicable in our *adversarial* tracking problem, as x_t^* is not revealed before u_t is committed. (Even worse, x_t^* can *adapt* to u_t and sabotage any controller that selects u_t based on an *assumed or predicted* x_t^*.) In this paper, instead of fixing this framework, we propose an approach with a different principle.

Finally, our approach can be complementary to (Agarwal et al., 2019a) in two ways: (i) Our strongly adaptive OCOM algorithm (Algorithm 7) can be combined with DAC to improve a recent regulation controller for time-varying systems (Gradu et al., 2020). On all $\mathcal{I} \subset [1:T]$, the regret of regulation is improved from $\tilde{O}(\sqrt{T})$ to $\tilde{O}(\sqrt{|\mathcal{I}|})$. (ii) Our adversarial tracking controller could be *added* to a regulation controller to achieve both goals simultaneously.

4.3 Reduction to strongly adaptive OCOM

Now we sketch the key idea of our reduction, which is to truncate history and directly optimize on the action space. To the best of our knowledge, our approach is the first that uses the "tracking" property of online learning algorithms in tracking control.

Algorithm 5 A reduction from adversarial tracking control to strongly adaptive OCOM.

```
Require: Time horizon $T > 1$ and a strongly adaptive OCOM algorithm.
1: Initialize the strongly adaptive OCOM algorithm as $\mathcal{A}$, with time horizon $T$. Problem constants for OCOM are defined using those for adversarial tracking: $\mathcal{V} \leftarrow \mathsf{B}^{d_u}(0, U)$, $R \leftarrow U$, $H \leftarrow \max\{\lceil -\log T/\log(1-\gamma)\rceil, 2\gamma^{-1}\}$, $L \leftarrow \kappa L^*$ and $\tilde{G} \leftarrow 2\kappa\gamma^{-1}L^*$.
2: for $t = 1, \dots, T$ do
3:     Observe $x_t$ and compute $w_{t-1} = x_t - A_{t-1}x_{t-1} - B_{t-1}u_{t-1}$.
4:     Obtain $u_t$ from $\mathcal{A}$, apply it, observe the loss function $l_t^*$ and suffer $l_t^*(x_t, u_t)$.
5:     Compute the ideal loss function $f_t$ from (3), and return it to $\mathcal{A}$.
6: end for
```

To begin with, note that old actions have diminishing effect on future states due to the stability of the system. Therefore, given a large enough memory constant H, the actual state x_t can be approximated by an ideal state
$$y_t(u_{t-H:t-1}) = \sum_{i=t-H}^{t-1} \left(\prod_{j=i+1}^{t-1} A_j \right) (B_i u_i + w_i),$$
which is the value x_t would take if $x_{t-H} = 0$. Using y_t to replace x_t, the actual loss $l_t^*(x_t, u_t)$ can also be approximated by an ideal loss
$$f_t(u_{t-H:t}) = l_t^*(y_t(u_{t-H:t-1}), u_t). \qquad (3)$$

Compared to $l_t^*(x_t, u_t)$, the ideal loss $f_t(u_{t-H:t})$ only depends on a finite length action history $u_{t-H:t}$ instead of all the past actions. Therefore, one may use a strongly adaptive OCOM algorithm to *dynamically track* the optimal input that minimizes f_t, which should be close to the optimal action that minimizes l_t^*. Formally, we present the pseudo-code in Algorithm 5.

Technically, the main benefit of our approach is that *on any time interval* it guarantees a regret bound against an *interval-dependent* comparator class.

Definition 4.1 (Interval-dependent comparator class). *Given any time interval $\mathcal{I} = [a : b] \subset [H+1 : T]$, the comparator class $\mathcal{C}_\mathcal{I}$ is defined as the set of action sequences $u_{1:T}^C$ such that for all $t \in [a-H : b]$, $u_t^C = u_b^C$.*

In other words, the comparator class $\mathcal{C}_\mathcal{I}$ contains all action sequences that are essentially fixed on the investigated time-interval $\mathcal{I}$, but arbitrarily varying elsewhere.

The performance guarantee of Algorithm 5 is stated in Theorem 3. We write $x_t(u_{1:t-1}^A)$ and u_t^A as the state-action pair induced by Algorithm 5. Similarly, $x_t(u^C_{1:t-1})$ is the state induced by a comparator. (Superscripts A and C represent "Adversarial tracking" and "Comparator".)

Theorem 3. *Given any strongly adaptive OCOM algorithm satisfying Equation (2), for all $\mathcal{I} = [a:b] \subset [H+1:T]$, Algorithm 5 guarantees*

$$\sum_{t=a}^{b} l^*_t\left(x_t(u^A_{1:t-1}), u^A_t\right) - \min_{u^C_{1:T}\in\mathcal{C}_{\mathcal{I}}} \sum_{t=a}^{b} l^*_t\left(x_t(u^C_{1:t-1}), u^C_t\right) = \tilde{O}\left(\sqrt{|\mathcal{I}|}\right),$$

where $\tilde{O}(\cdot)$ subsumes problem constants and poly$(\log T)$.

Theorem 3 can be interpreted as: on *any* time interval, the cumulative tracking loss approaches that of the best *interval-dependent* action. If Algorithm 5 uses our strongly adaptive OCOM algorithm, then the obtained bound further adapts to the observed gradients.

Notably, Theorem 3 improves existing results on adversarial tracking (e.g., Abbasi-Yadkori et al. 2014), especially when the reference trajectory has a large range of movement. To make it clear, consider tracking a piecewise constant reference trajectory. In that case, existing regret bounds are only established *on the entire time horizon* $[1:T]$, and the comparator class only contains static linear controllers which are weak baselines for tracking this moving target. In comparison, Theorem 3 induces a regret bound *on any time interval*, including $[1:T]$ and its much shorter sub-intervals. The regret bound on $[1:T]$ suffers from the same problem (the comparator class is weak). However, on all time intervals where the target is fixed, the interval-dependent comparator class is strong, and the regret bound makes much more sense.

To make the above discussion even more concrete, we construct the following example. Here we can further derive a *non-comparative* tracking error bound.

Example 1. *Consider a time interval $\mathcal{I} = [a:b] \subset [H+1:T]$. For all $t \in \mathcal{I}$, we assume*

1. *$(I - A_t)^{-1}B_t = B_{\mathcal{I}}$ for some time-invariant matrix $B_{\mathcal{I}}$, which includes static A_t and B_t as a special case. Note that $I - A_t$ is invertible since $\|A_t\| < 1$.*
2. *The target $x^*_t = x^*_{\mathcal{I}}$ for some time-invariant $x^*_{\mathcal{I}} \in \{B_{\mathcal{I}}u; \|u\| \leq U\}$, and $l^*_t(x,u) = \|x - x^*_t\|$.*

Corollary 4. *Consider running Algorithm 5 on an adversarial tracking problem that satisfies Example 1 on a time interval $\mathcal{I}$. For all $t \in \mathcal{I} = [a:b]$,*

$$\frac{1}{t-a+1}\sum_{i=a}^{t}\left\|x_i(u^A_{1:i-1}) - x^*_{\mathcal{I}}\right\| \leq \gamma^{-1}W + \tilde{O}\left((t-a+1)^{-1/2}\right).$$

Corollary 4 directly characterize the tracking error *without any comparator* which is the performance metric of interest in most classical control-theoretic literature. Further applying Jensen's inequality, the time-average of the states *on any time interval satisfying Example 1* converges to a norm ball around the target. Notably, Algorithm 5 does not need to know any favorable problem structure a priori: when running on a long time horizon $[1:T]$, it can *automatically* exploit the inactivity of the target (if any) on shorter sub-intervals. This is fundamentally different from the classical idea in tracking control where a generative model of the target is hard-coded into the controller.

Experiments For conciseness, we defer experimental results to Appendix E. All three components of our contribution (cf. Section 1.1) are tested numerically there.

5 CONCLUSION

We consider tracking adversarial targets in a general linear system. Three techniques are developed in a hierarchical manner, and their combination is a strongly adaptive tracking controller that significantly improves existing results. Our approach could facilitate the application of online learning ideas to a wider range of linear control problems.

Acknowledgements

We thank the anonymous reviewers for their constructive feedback. This research was partially supported by the NSF under grants IIS-1914792, DMS-1664644, and CNS-1645681, by the ONR under grants N00014-19-1-2571 and N00014-21-1-2844, by the DOE under grants DE-AR-0001282 and DE-EE0009696, by the NIH under grants R01 GM135930 and UL54 TR004130, and by Boston University.

References

Y. Abbasi-Yadkori and C. Szepesvári. Regret bounds for the adaptive control of linear quadratic systems. In *Proceedings of the 24th Annual Conference on Learning Theory*, pages 1–26. JMLR Workshop and Conference Proceedings, 2011.

Y. Abbasi-Yadkori, P. Bartlett, and V. Kanade. Tracking adversarial targets. In *International Conference on Machine Learning*, pages 369–377. PMLR, 2014.

D. Adamskiy, W. M. Koolen, A. Chernov, and V. Vovk. A closer look at adaptive regret. *The Journal of Machine Learning Research*, 17(1):706–726, 2016.

N. Agarwal, B. Bullins, E. Hazan, S. Kakade, and K. Singh. Online control with adversarial disturbances. In *International Conference on Machine Learning*, pages 111–119. PMLR, 2019a.

N. Agarwal, E. Hazan, and K. Singh. Logarithmic regret for online control. In *Advances in Neural Information Processing Systems*, pages 10175–10184, 2019b.

O. Anava, E. Hazan, and S. Mannor. Online learning for adversaries with memory: price of past mistakes. In *Advances in Neural Information Processing Systems*, pages 784–792, 2015.

A. Astolfi. *Tracking and Regulation in Linear Systems*, pages 1469–1475. Springer London, London, 2015. ISBN 978-1-4471-5058-9. doi: 10.1007/978-1-4471-5058-9_198. URL `https://doi.org/10.1007/978-1-4471-5058-9_198`.

A. Beygelzimer, E. Hazan, S. Kale, and H. Luo. Online gradient boosting. *arXiv preprint arXiv:1506.04820*, 2015.

A. Bhaskara, A. Cutkosky, R. Kumar, and M. Purohit. Power of hints for online learning with movement costs. In *International Conference on Artificial Intelligence and Statistics*, pages 2818–2826. PMLR, 2021.

K. Bhatia and K. Sridharan. Online learning with dynamics: A minimax perspective. *arXiv preprint arXiv:2012.01705*, 2020.

O. Bousquet and M. K. Warmuth. Tracking a small set of experts by mixing past posteriors. *Journal of Machine Learning Research*, 3(Nov):363–396, 2002.

N. Cesa-Bianchi, O. Dekel, and O. Shamir. Online learning with switching costs and other adaptive adversaries. In *Advances in Neural Information Processing Systems*, pages 1160–1168, 2013.

N. Chen, A. Agarwal, A. Wierman, S. Barman, and L. L. Andrew. Online convex optimization using predictions. In *Proceedings of the 2015 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, pages 191–204, 2015.

N. Chen, G. Goel, and A. Wierman. Smoothed online convex optimization in high dimensions via online balanced descent. In *Conference On Learning Theory*, pages 1574–1594. PMLR, 2018.

A. Cohen, A. Hasidim, T. Koren, N. Lazic, Y. Mansour, and K. Talwar. Online linear quadratic control. In *International Conference on Machine Learning*, pages 1029–1038. PMLR, 2018.

A. Cutkosky. *Algorithms and Lower Bounds for Parameter-free Online Learning*. Stanford University, 2018.

A. Cutkosky. Parameter-free, dynamic, and strongly-adaptive online learning. In *International Conference on Machine Learning*, pages 2250–2259, 2020.

A. Cutkosky and F. Orabona. Black-box reductions for parameter-free online learning in Banach spaces. In *Conference On Learning Theory*, pages 1493–1529, 2018.

A. Daniely and Y. Mansour. Competitive ratio vs regret minimization: achieving the best of both worlds. In *Algorithmic Learning Theory*, pages 333–368. PMLR, 2019.

A. Daniely, A. Gonen, and S. Shalev-Shwartz. Strongly adaptive online learning. In *International Conference on Machine Learning*, pages 1405–1411, 2015.

S. Dean, H. Mania, N. Matni, B. Recht, and S. Tu. On the sample complexity of the linear quadratic regulator. *Foundations of Computational Mathematics*, pages 1–47, 2019.

D. Foster and M. Simchowitz. Logarithmic regret for adversarial online control. In *International Conference on Machine Learning*, pages 3211–3221. PMLR, 2020.

D. J. Foster, A. Rakhlin, and K. Sridharan. Online learning: Sufficient statistics and the burkholder method. In *Conference On Learning Theory*, pages 3028–3064. PMLR, 2018.

Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

G. Goel, Y. Lin, H. Sun, and A. Wierman. Beyond online balanced descent: An optimal algorithm for smoothed online optimization. *Advances in Neural Information Processing Systems*, 32:1875–1885, 2019.

E. Gofer. Higher-order regret bounds with switching costs. In *Conference on Learning Theory*, pages 210–243. PMLR, 2014.

P. Gradu, E. Hazan, and E. Minasyan. Adaptive regret for control of time-varying dynamics. *arXiv preprint arXiv:2007.04393*, 2020.

E. Hazan and C. Seshadhri. Efficient learning algorithms for changing environments. In *Proceedings of the 26th International Conference on Machine Learning*, pages 393–400, 2009.

M. Herbster and M. K. Warmuth. Tracking the best expert. *Machine learning*, 32(2):151–178, 1998.

A. Jadbabaie, A. Rakhlin, S. Shahrampour, and K. Sridharan. Online optimization: Competing with dynamic comparators. In *Artificial Intelligence and Statistics*, pages 398–406. PMLR, 2015.

K.-S. Jun, F. Orabona, S. Wright, and R. Willett. Improved strongly adaptive online learning using coin betting. In *Artificial Intelligence and Statistics*, pages 943–951, 2017.

A. Kalai and S. Vempala. Efficient algorithms for online decision problems. *Journal of Computer and System Sciences*, 71(3):291–307, 2005.

M. Kapralov and R. Panigrahy. Prediction strategies without loss. *arXiv preprint arXiv:1008.3672*, 2010.

Y. Li, X. Chen, and N. Li. Online optimal control with linear dynamics and predictions: Algorithms and regret analysis. In *NeurIPS*, pages 14858–14870, 2019.

D. Limon and T. Alamo. *Tracking Model Predictive Control*, pages 1475–1484. Springer London, London, 2015. ISBN 978-1-4471-5058-9. doi: 10.1007/978-1-4471-5058-9_3. URL `https://doi.org/10.1007/978-1-4471-5058-9_3`.

H. B. McMahan and F. Orabona. Unconstrained online linear learning in hilbert spaces: Minimax algorithms and normal approximations. In *Conference on Learning Theory*, pages 1020–1039, 2014.

Z. Mhammedi and W. M. Koolen. Lipschitz and comparator-norm adaptivity in online learning. In *Conference on Learning Theory*, pages 2858–2887. PMLR, 2020.

E. Minasyan, P. Gradu, M. Simchowitz, and E. Hazan. Online control of unknown time-varying dynamical systems. *Advances in Neural Information Processing Systems*, 34, 2021.

F. Orabona. A modern introduction to online learning. *arXiv preprint arXiv:1912.13213v3*, 2020.

F. Orabona and D. Pál. Coin betting and parameter-free online learning. In *Advances in Neural Information Processing Systems*, pages 577–585, 2016.

U. Sherman and T. Koren. Lazy oco: Online convex optimization on a switching budget. *arXiv preprint arXiv:2102.03803*, 2021.

G. Shi, Y. Lin, S.-J. Chung, Y. Yue, and A. Wierman. Online optimization with memory and competitive control. In *Thirty-fourth Conference on Neural Information Processing Systems*, 2020.

M. Simchowitz. Making non-stochastic control (almost) as easy as stochastic. *arXiv preprint arXiv:2006.05910*, 2020.

M. Simchowitz, K. Singh, and E. Hazan. Improper learning for non-stochastic control. *arXiv preprint arXiv:2001.09254*, 2020.

D. van der Hoeven. User-specified local differential privacy in unconstrained adaptive online learning. In *NeurIPS*, pages 14080–14089, 2019.

C. Yu, G. Shi, S.-J. Chung, Y. Yue, and A. Wierman. The power of predictions in online control. *arXiv preprint arXiv:2006.07569*, 2020.

L. Zhang, T. Yang, J. Yi, R. Jin, and Z.-H. Zhou. Improved dynamic regret for non-degenerate functions. *arXiv preprint arXiv:1608.03933*, 2016.

L. Zhang, T. Yang, Z.-H. Zhou, et al. Dynamic regret of strongly adaptive methods. In *International conference on machine learning*, pages 5882–5891. PMLR, 2018.

L. Zhang, T.-Y. Liu, and Z.-H. Zhou. Adaptive regret of convex and smooth functions. In *International Conference on Machine Learning*, pages 7414–7423, 2019a.

L. Zhang, G. Wang, W.-W. Tu, and Z.-H. Zhou. Dual adaptivity: A universal algorithm for minimizing the adaptive regret of convex functions. *arXiv preprint arXiv:1906.10851*, 2019b.

L. Zhang, S. Lu, and T. Yang. Minimizing dynamic regret and adaptive regret simultaneously. In *International Conference on Artificial Intelligence and Statistics*, pages 309–319. PMLR, 2020.

P. Zhao, Y.-X. Wang, and Z.-H. Zhou. Non-stationary online learning with memory and non-stochastic control. *arXiv preprint arXiv:2102.03758*, 2021.

M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *International Conference on Machine Learning*, pages 928–936, 2003.

Supplementary Material: Adversarial Tracking Control via Strongly Adaptive Online Learning with Memory

Organization Appendix A contains additional discussion of existing works omitted in the main paper. Appendix B, C and D contain details of our three technical contributions. Finally, empirical results are provided in Appendix E.

A ADDITIONAL DISCUSSION OF EXISTING WORKS

As reviewed in Section 1.2, our approach to adversarial tracking control relies on its connection to tracking nonstationary comparators in online learning. There are multiple performance metrics to quantify the latter goal. In this paper we choose strong adaptivity. Other than this, one may use dynamic regret or competitive ratio. We briefly review them as follows.

Dynamic regret In the context of OLO, dynamic regret (Jadbabaie et al., 2015; Zhang et al., 2016, 2018; Zinkevich, 2003) is the regret that directly compares to a nonstationary prediction sequence $u_{1:T}$ on the entire time horizon $[1:T]$. Such bounds in general depend on the cumulative variation of the comparator over $[1:T]$ (the *path length*), and sometimes also the variation of the loss function. The path length can be defined in multiple ways; when defined as $P=\sum_{t=1}^{T}\|u_t-u_{t+1}\|$, the optimal dynamic regret bound is $O(\sqrt{PT})$. The idea is that if the comparator is static ($P=0$), then the dynamic regret reduces to the static regret; with a large path length ($P=O(T)$), the dynamic regret becomes vacuous.

Existing works (Cutkosky, 2020; Zhang et al., 2018, 2020) have investigated the relation between dynamic regret and strongly adaptive regret in OLO. It has been suggested that the former could be a slightly weaker notion than the latter, as dynamic regret is derived from strongly adaptive regret in (Cutkosky, 2020; Zhang et al., 2018) while no result in the opposite direction has been given (to the best of our knowledge). Generalizing it from OLO to online learning with memory, (Zhao et al., 2021) provided a dynamic regret analysis of OCOM and nonstochastic control. It is possible that such a result could be achieved via strongly adaptive approaches (such as our Algorithm 7 or (Daniely and Mansour, 2019)), although detailed analysis is beyond the scope of this paper.

Competitive ratio Competitive ratio is a largely different performance metric in online learning compared to the regret framework. For online control, the relevant setting for competitive ratio analysis is *Smoothed Online Convex Optimization* (SOCO) (Chen et al., 2015, 2018; Goel et al., 2019). It has two key differences with OCO:

1. The loss function l_t is revealed to the player before his prediction x_t is made.
2. In addition to the loss $l_t(x_t)$, the player further suffers a movement cost $c(x_{t-1},x_t)$ in each round, where $c(\cdot,\cdot)$ is some penalty function for large movements.

From this setting, SOCO and the accompanying competitive ratio analysis are particularly suitable for *online control with predictions*, as we review later. The general form of a competitive ratio guarantee is

$$\text{Cost on }[1:T]\le\alpha\cdot\text{Comparator cost on }[1:T]+\beta,$$

where α and β are constants, and α is defined as the competitive ratio. The comparator class contains all the prediction sequences, therefore intrinsically the benchmarks are nonstationary. Compared to a dynamic regret bound, the competitive ratio analysis (i) does not depend on the path length; and (ii) characterizes the cost of the algorithm in a multiplicative manner with respect to the comparator cost, instead of an additive one.

Advantages of strong adaptivity in adversarial tracking Following the above discussion, we next discuss the advantages of strong adaptivity over the other two performance metrics in adversarial tracking. First, compared to the other two, a strongly adaptive regret guarantee is *local*: on *all* sub-intervals in $[1 : T]$ we have a regret bound that compares to the *interval-dependent optimal* comparator, and the bound depends on the sub-interval length instead of T. In contrast, the other two performance metrics are stated for the entire time horizon. Second, both (Daniely and Mansour, 2019) and our Algorithm 7 can incorporate algorithms with dynamic regret or competitive ratio guarantees (for our approach, see Remark 3.1). The resulting algorithm guarantees the best of both worlds. Third, as we discussed above, dynamic regret could be conceptually weaker than strongly adaptive regret, and competitive ratio analysis requires a different setting (predictions).

Linear control with predictions Inspired by the classical idea of model-predictive control, a series of recent works (Li et al., 2019; Shi et al., 2020; Yu et al., 2020) considered learning-based approaches for linear control with predictions. Specifically for tracking, predictions of the reference trajectory are typically required, which is less general than our fully adversarial setting. Furthermore, the loss functions are strongly convex, resulting in less modeling power (e.g., for modeling target regions, where the minimizer of the loss function is not unique).

Notably, for online learning, (Shi et al., 2020) presented an interesting generalization of SOCO called *OCO with structured memory*:

1. The one-step memory in SOCO is generalized to longer memory similar to OCOM.
2. Accurate prediction of the loss function is not required. In each round, the adversary first reveals a function h_t. After the player picks $x_t \in \mathcal{V}$, the adversary further reveals $v_t \in \mathcal{V}$ and induces a loss $h_t(x_t - v_t)$. In other words, the player only needs to accurately predict the *shape* of the loss function; the actual incurred loss is further shifted by an adversarial component.

Based on this new setting, (Shi et al., 2020) provided a competitive ratio analysis of the regulation control problem. It is possible that such analysis could be extended to track fully adversarial targets, but still, (i) accurate predictions of strongly convex loss functions are required; (ii) the resulting algorithm could be combined with our approach, as discussed in Remark 3.1.

B DETAILS ON COMPARATOR-ADAPTIVE OLO WITH MOVEMENT COST

This section presents details on our first contribution, movement-aware OLO. We rely heavily on a duality between unconstrained OLO and the coin-betting game, which is summarized in Appendix B.1. After that, Appendix B.2 introduces an existing reduction from constrained OLO to unconstrained OLO, adopted in our Algorithm 1 and Algorithm 7. The last two subsections provide detailed analysis of Algorithm 1 and 2, respectively.

B.1 An overview of coin-betting and unconstrained OLO

We start from the definition of the coin-betting game: A player has initial wealth $\mathrm{Wealth}_0 = \varepsilon$. In each round, he picks a betting fraction $\beta_t \in [-1, 1]$ and bets an amount $x_t = \beta_t \mathrm{Wealth}_{t-1}$. Then, an adversarial coin tossing $c_t \in [-1, 1]$ is revealed, and the wealth of the player is changed by $c_t x_t$. In other words, the player wins money if $c_t x_t > 0$, and loses money if $c_t x_t < 0$. The goal of the player is to design betting fractions $\beta_1, \beta_2, \ldots$ such that his wealth in the T-th round is maximized. We are particularly interested in *parameter-free* betting strategies, for example the Krichevsky-Trofimov (KT) bettor: $\beta_t = \sum_{i=1}^{t-1} c_i/t$. Notice that it does not rely on any hyperparameters.

We can associate the coin-betting game to one-dimensional unconstrained OLO (Orabona 2020, Theorem 9.6). For an OLO problem with loss gradient $g_t \in \mathbb{R}$, one can maintain a coin-betting algorithm with $c_t = -g_t$, and predict *exactly* its betting amount x_t in OLO. The wealth lower bound for coin-betting is equivalent to a regret upper bound for OLO. Induced by a parameter-free bettor (such as KT), the resulting OLO algorithm can enjoy the following benefits: (i) There are no hyperparameters to tune. (ii) The regret bound has optimal dependence on the comparator norm. (iii) When the comparator is the null comparator 0, the regret upper bound reduces to a constant. In other words, the *cumulative cost* is at most a constant. Properties (ii) and (iii) are often called *comparator-adaptivity.*

To further appreciate the power of such approach, let us compare the resulting 1d unconstrained OLO algorithm to standard Online Gradient Descent (OGD).

1. Analytically, with an unconstrained domain, L-Lipschitz losses and learning rate η, OGD has the regret bound
$$\sum_{t=1}^{T}\langle g_t, x_t-u\rangle \leq \frac{|u-x_1|^2}{2\eta}+\frac{\eta L^2 T}{2},\ \forall u\in\mathbb{R}.$$
Since the optimal comparator u is unknown beforehand, one has to choose $\eta = O(1/(L\sqrt{T}))$, leading to the sub-optimal regret bound $O(|u|^2 L\sqrt{T})$. In comparison, KT-based OLO algorithm guarantees a regret bound $\tilde{O}(|u|L\sqrt{T})$, matching the lower bound up to logarithmic factors.

2. Intuitively, assume the loss gradients are
$$g_t=\begin{cases}-1, & \text{if } x_t\leq x^*,\\ 1, & \text{otherwise,}\end{cases}$$
where x^* is a fixed "target". With a pre-determined learning rate η, OGD approaches the target linearly. However, since x^* is unknown, there are always cases where x^* is far enough from the starting point of OGD, making OGD very slow to find x^*. In comparison, KT-based OLO algorithm approaches x^* with *exponentially increasing speed* (Orabona, 2020, Figure 9.1), finding x^* a lot faster.

As a final note, in this paper we aim to bound the sum of regret and movement in coin-betting-based OLO algorithms. Although the exponentially increasing per-step movement is good for regret minimization, it poses a significant challenge for the control of movement cost. Using a movement-restricted bettor (Algorithm 1), we achieve this in Theorem 1.

B.2 Adding constraints in OLO

Our approach requires a reduction from constrained OLO to unconstrained OLO, proposed in (Cutkosky, 2020). The pseudo-code is Algorithm 6. We use this reduction in both the movement-aware OLO algorithm (Algorithm 1) and the OCOM meta-algorithm (Algorithm 7).

Algorithm 6 Adding constraints in OLO.

```
Require: An OLO algorithm $\mathcal{A}$ and an arbitrary nonempty, closed and convex domain $\mathcal{V}$.
1: for $t = 1, \ldots, T$ do
2:     Obtain the prediction $\tilde{x}_t$ from $\mathcal{A}$.
3:     Predict $x_t = \Pi_{\mathcal{V}}(\tilde{x}_t)$ and receive the loss subgradient $g_t$.
4:     Define a surrogate loss function $h_t$ as
           $h_t(x)=\begin{cases}\langle g_t,x\rangle, & \text{if } \langle g_t,\tilde{x}_t\rangle\geq\langle g_t,x_t\rangle,\\ \langle g_t,x\rangle+\langle g_t,x_t-\tilde{x}_t\rangle\frac{\|x-\Pi_{\mathcal{V}}(x)\|}{\|x_t-\tilde{x}_t\|}, & \text{otherwise.}\end{cases}$
5:     Obtain a subgradient $\tilde{g}_t\in\partial h_t(\tilde{x}_t)$ and return it to $\mathcal{A}$ as the $t$-th loss subgradient.
6: end for
```

Lemma B.1 (Cutkosky 2020, Theorem 2). *Algorithm 6 has the following properties for all t: (1) h_t is a convex function on $\mathcal{V}$. (2) $\|\tilde{g}_t\|\leq\|g_t\|$. (3) For all $u\in\mathcal{V}$, $\langle g_t, x_t-u\rangle\leq\langle\tilde{g}_t,\tilde{x}_t-u\rangle$.*

B.3 Analysis of Algorithm 1

This subsection provides analysis of Algorithm 1, which is organized as follows. We first show the well-posedness of Line 5 (the existence and uniqueness of solution). After that, we present a few useful lemmas before proving the performance guarantee of Algorithm 1 (Theorem 1).

Lemma B.2. *For all $t\geq 1$, Equation (1) has a unique solution and the solution is positive.*

Proof of Lemma B.2. For clarity, Equation (1) is copied here.
$$\text{Wealth}_t=(1-\tilde{g}_t\beta_t-\gamma\beta_t/\sqrt{t})\text{Wealth}_{t-1}-\lambda|\beta_t\text{Wealth}_{t-1}-\beta_{t+1}\text{Wealth}_t|.$$

By definition, $|\lambda\beta_{t+1}|\leq 1/2$. The RHS of (1) is 1/2-Lipschitz with respect to Wealth_t, and the LHS is Wealth_t itself. Therefore, a solution exists and is unique.

To prove $\mathrm{Wealth}_t > 0$, we use induction. $\mathrm{Wealth}_0 = \varepsilon > 0$. Suppose $\mathrm{Wealth}_{t-1} > 0$, then

$$\mathrm{Wealth}_t \geq (1 - \tilde{g}_t\beta_t - \gamma\beta_t/\sqrt{t})\mathrm{Wealth}_{t-1} - \lambda\beta_t\mathrm{Wealth}_{t-1} - \lambda\beta_{t+1}\left|\mathrm{Wealth}_t\right|.$$

Let $z = \lambda\beta_{t+1}\mathrm{sign}(\mathrm{Wealth}_t)$. Note that $|z|\leq 1/2$ and $|\tilde{g}_t + \gamma/\sqrt{t} + \lambda|\beta_t \leq 1/2$. Therefore,

$$\mathrm{Wealth}_t \geq \frac{1 - \tilde{g}_t\beta_t - \gamma\beta_t/\sqrt{t} - \lambda\beta_t}{1+z}\mathrm{Wealth}_{t-1} > 0. \qquad \square$$

B.3.1 Auxiliary lemmas for Algorithm 1

The first auxiliary lemma states that the betting fraction β_t changes slowly.

Lemma B.3. *For all $t \geq 1$, $|\beta_{t+1} - \beta_t|\leq 2/(Ct)$.*

Proof of Lemma B.3. The result for $t = 1$ trivially holds. We only consider $t \geq 2$.

Since the Euclidean projection to a closed convex set is contractive, we have

$$\left|\Pi_{\mathcal{B}_t}(\hat{\beta}_t) - \Pi_{\mathcal{B}_t}(\hat{\beta}_{t+1})\right| \leq \left|\hat{\beta}_t - \hat{\beta}_{t+1}\right| = \left|\frac{\tilde{g}_t + 2C^2\hat{\beta}_t}{2C^2t}\right| \leq \frac{G}{C^2t}.$$

Moreover,

$$\left|\Pi_{\mathcal{B}_t}(\hat{\beta}_{t+1}) - \Pi_{\mathcal{B}_{t+1}}(\hat{\beta}_{t+1})\right| \leq \left|\frac{1}{\sqrt{2}C\sqrt{t-1}} - \frac{1}{\sqrt{2}C\sqrt{t}}\right| \leq \frac{1}{2\sqrt{2}C\sqrt{t}(t-1)} \leq \frac{1}{Ct}.$$

Applying the triangle inequality yields the result. $\square$

The next lemma quantifies the movement of Algorithm 1 using Wealth_t. By doing this, bounding the movement cost (Part 2 of Theorem 1) reduces to bounding the growth of Wealth_t.

Lemma B.4. *For all $t \geq 1$,*

$$|\tilde{x}_t - \tilde{x}_{t+1}| \leq \frac{6}{Ct}\mathit{Wealth}_{t-1}.$$

Proof of Lemma B.4. Assume $t > 1$ for the rest of this proof; the case of $t = 1$ can be verified similarly. Starting from (1), some simple algebra yields

$$\begin{aligned}\tilde{x}_{t+1} - \tilde{x}_t &= \beta_{t+1}\mathrm{Wealth}_t - \beta_t\mathrm{Wealth}_{t-1}\\ &= \left(\beta_{t+1} - \beta_t - \beta_{t+1}\tilde{g}_t\beta_t - \beta_{t+1}\beta_t\frac{\gamma}{\sqrt{t}}\right)\mathrm{Wealth}_{t-1} - \lambda\beta_{t+1}\left|\beta_{t+1}\mathrm{Wealth}_t - \beta_t\mathrm{Wealth}_{t-1}\right|.\end{aligned}$$

From Lemma B.2, $\mathrm{Wealth}_{t-1} > 0$, therefore,

$$(1 - \lambda\beta_{t+1})\left|\beta_{t+1}\mathrm{Wealth}_t - \beta_t\mathrm{Wealth}_{t-1}\right| \leq \left|\beta_{t+1} - \beta_t - \beta_{t+1}\tilde{g}_t\beta_t - \beta_{t+1}\beta_t\frac{\gamma}{\sqrt{t}}\right|\mathrm{Wealth}_{t-1}.$$

Note that $1 - \lambda\beta_{t+1} \geq 1/2$.

$$\begin{aligned}\left|\beta_{t+1}\mathrm{Wealth}_t - \beta_t\mathrm{Wealth}_{t-1}\right| &\leq 2\left|\beta_{t+1} - \beta_t - \beta_{t+1}\tilde{g}_t\beta_t - \beta_{t+1}\beta_t\frac{\gamma}{\sqrt{t}}\right|\mathrm{Wealth}_{t-1}\\ &\leq 2\left|\beta_{t+1} - \beta_t\right|\mathrm{Wealth}_{t-1} + 2\beta_t\beta_{t+1}\left|\tilde{g}_t + \frac{\gamma}{\sqrt{t}}\right|\mathrm{Wealth}_{t-1}.\end{aligned}$$

Applying Lemma B.3 and the definition of β_t and β_{t+1},

$$\|\tilde{x}_t - \tilde{x}_{t+1}\| \leq \left(\frac{4}{Ct} + \frac{2C}{2C^2\sqrt{t(t-1)}}\right)\mathrm{Wealth}_{t-1} \leq \frac{6}{Ct}\mathrm{Wealth}_{t-1}. \qquad \square$$

Following the reasoning from the previous lemma, we next bound the growth rate of Wealth_t in Lemma B.5 which could be of special interest. The key idea is that, the surrogate loss (Line 3 of Algorithm 1) incentivizes the *unconstrained prediction* $\tilde{x}_t$ to be bounded. Equivalently, the betting amount in the coin-betting algorithm is bounded, and hence the wealth cannot grow too fast. (For some background knowledge on this argument, Appendix B.1 provides an overview of the interplay between coin-betting and OLO.)

As discussed in Section 2, our proof makes a novel use of the black-box reduction from unconstrained OLO to constrained OLO (Algorithm 6): actually, we *do not use it as a black-box*, but rather analyze its impact on the unconstrained algorithm. To our knowledge, this is the first analysis that takes this perspective.

Lemma B.5. *For all $t \geq 1$, $\text{Wealth}_t \leq 4\bar{R}C\sqrt{t}$.*

Proof of Lemma B.5. Note that from Lemma B.2, $\text{Wealth}_t \geq 0$. Additionally from our definition of β_t, we have $\beta_t, x_t, \tilde{x}_t \geq 0$.

We prove this lemma in three steps. First, we show a weaker result, $\text{Wealth}_t \leq G\bar{R}(t+1)$. Using this result, we then prove that $\tilde{x}_t \leq 2\sqrt{2}\bar{R}$. In other words, even though $\tilde{x}_t$ is the output of a coin-betting-based OLO algorithm that works in the unbounded domain, *it is actually bounded* due to the effect of the surrogate losses. Finally, we revisit wealth and show that $\text{Wealth}_t \leq 4\bar{R}C\sqrt{t}$.

Step 1 Prove that for all $t \geq 0$, $\text{Wealth}_t \leq G\bar{R}(t+1)$.

Consider the two cases in the definition of $\tilde{g}_t$. If $g_t\tilde{x}_t \geq g_t x_t$, then $\tilde{g}_t = g_t$, and

$$\begin{aligned}\text{Wealth}_t &= \text{Wealth}_{t-1} - \tilde{g}_t\tilde{x}_t - \lambda|\tilde{x}_t - \tilde{x}_{t+1}| - \frac{\gamma}{\sqrt{t}}|\tilde{x}_t| \\ &\leq \text{Wealth}_{t-1} - g_t x_t \leq \text{Wealth}_{t-1} + |g_t|\bar{R}.\end{aligned}$$

If $g_t\tilde{x}_t < g_t x_t$, then $\tilde{g}_t = 0$ and $\text{Wealth}_t \leq \text{Wealth}_{t-1}$. An induction and $\varepsilon \leq G\bar{R}$ yield the result.

Step 2 Prove that for all $t \geq 1$, $\tilde{x}_t \leq 2\sqrt{2}\bar{R}$.

This holds trivially for $t = 1$. We use induction: suppose this result holds for t, and we need to show $\tilde{x}_{t+1} \leq 2\sqrt{2}\bar{R}$. There are two cases: (1) $\tilde{x}_t \notin \mathcal{V}_{1d}$; (2) $\tilde{x}_t \in \mathcal{V}_{1d}$. Note that the first case is only possible when $t > 1$.

Case (1.1) $\tilde{x}_t \notin \mathcal{V}_{1d}$, $g_t\tilde{x}_t \geq g_t x_t$.

In this case, $\tilde{g}_t = g_t \geq 0$ and $g_t x_t \geq 0$. It follows,

$$\text{Wealth}_t \leq \text{Wealth}_{t-1} - g_t x_t \leq \text{Wealth}_{t-1}.$$

Next we consider the three cases of β_t.

(i) First, note that $\beta_t \neq 0$; otherwise $\tilde{x}_t = \beta_t \text{Wealth}_{t-1} = 0 \in \mathcal{V}_{1d}$.

(ii) If $\beta_t = \hat{\beta}_t = -\sum_{i=1}^{t-1}\tilde{g}_i/[2C^2(t-1)]$, then

$$\beta_{t+1} \leq \left|\hat{\beta}_{t+1}\right| = \frac{1}{2C^2t}\left|-\sum_{i=1}^{t}\tilde{g}_i\right| = \frac{\left|2C^2(t-1)\beta_t - g_t\right|}{2C^2t} \leq \max\left\{\frac{t-1}{t}\beta_t, \frac{g_t}{2C^2t}\right\}.$$

The last inequality is due to $\beta_t, g_t \geq 0$. Therefore,

$$\tilde{x}_{t+1} = \beta_{t+1}\text{Wealth}_t \leq \max\left\{\beta_t\text{Wealth}_{t-1}, G\text{Wealth}_{t-1}/(2C^2t)\right\} \leq \max\{2\sqrt{2}\bar{R}, G^2\bar{R}/(2C^2)\} \leq 2\sqrt{2}\bar{R},$$

where we use the result from Step 1.

(iii) If $\beta_t = 1/(C\sqrt{2(t-1)})$, then

$$\tilde{x}_{t+1} = \beta_{t+1}\text{Wealth}_t \leq \frac{1}{C\sqrt{2t}}\text{Wealth}_{t-1} \leq \frac{1}{C\sqrt{2(t-1)}}\text{Wealth}_{t-1} = \beta_t\text{Wealth}_{t-1} \leq 2\sqrt{2}\bar{R}.$$

Case (1.2) $\tilde{x}_t \notin \mathcal{V}_{1d}$, $g_t\tilde{x}_t < g_t x_t$.

In this case, $\tilde{g}_t = 0$ and $\text{Wealth}_t \leq \text{Wealth}_{t-1}$. Same as Case (1.1), $\beta_t \neq 0$, leading to $\hat{\beta}_t \geq 0$ and $\beta_t = \min\{\hat{\beta}_t, 1/(C\sqrt{2(t-1)})\}$. Also note that

$$\left|\hat{\beta}_{t+1}\right| = \frac{1}{2C^2 t}\left|-\sum_{i=1}^{t}\tilde{g}_i\right| = \frac{1}{2C^2 t}\left|-\sum_{i=1}^{t-1}\tilde{g}_i\right| \leq \frac{1}{2C^2(t-1)}\left|-\sum_{i=1}^{t-1}\tilde{g}_i\right| = \left|\hat{\beta}_t\right|.$$

Therefore,

$$\beta_{t+1} \leq \min\left\{\left|\hat{\beta}_{t+1}\right|, \frac{1}{C\sqrt{2t}}\right\} \leq \min\left\{\left|\hat{\beta}_t\right|, \frac{1}{C\sqrt{2(t-1)}}\right\} = \beta_t,$$

and $\tilde{x}_{t+1} = \beta_{t+1}\text{Wealth}_t \leq \beta_t \text{Wealth}_{t-1} \leq \tilde{x}_t \leq 2\sqrt{2}\bar{R}$.

Case (2) $\tilde{x}_t \in \mathcal{V}_{1d}$.

In this case, $\tilde{x}_t = x_t$ and $\tilde{g}_t = g_t$. $\tilde{x}_{t+1} = \beta_{t+1}\text{Wealth}_t \leq (1 - g_t\beta_t)\beta_{t+1}\text{Wealth}_{t-1}$.

If $t = 1$, then $\tilde{x}_{t+1} = \beta_{t+1}\text{Wealth}_t \leq \sqrt{2}G\bar{R}/C \leq \sqrt{2}\bar{R}$, where we use $\text{Wealth}_1 \leq 2G\bar{R}$ from Step 1 and $\beta_2 \leq 1/(\sqrt{2}C)$.

If $t > 1$, we consider the three cases of β_t as follows. (For the rest of the discussion assume $t > 1$.)

(i) If $\beta_t = 0$, then from Lemma B.3 we have $\beta_{t+1} \leq 2/(Ct)$, and $\tilde{x}_{t+1} \leq (1 - g_t\beta_t)\beta_{t+1}\text{Wealth}_{t-1} = \beta_{t+1}\text{Wealth}_{t-1} \leq 2G\bar{R}/C \leq 2\bar{R}$.

(ii) If $\beta_t = \hat{\beta}_t = -\sum_{i=1}^{t-1}\tilde{g}_i/[2C^2(t-1)]$, then

$$\beta_{t+1} \leq \left|\hat{\beta}_{t+1}\right| = \frac{1}{2C^2 t}\left|-\sum_{i=1}^{t}\tilde{g}_i\right| = \frac{\left|2C^2(t-1)\beta_t - g_t\right|}{2C^2 t} \leq \frac{t-1}{t}\beta_t + \frac{G}{2C^2 t}.$$

Note that since $\tilde{x}_t \in \mathcal{V}_{1d}$, we have $\beta_t\text{Wealth}_{t-1} \leq \bar{R}$. Using $\tilde{x}_{t+1} \leq (1 - g_t\beta_t)\beta_{t+1}\text{Wealth}_{t-1}$ and $|g_t\beta_t| \leq 1/2$ we have

$$\tilde{x}_{t+1} \leq \frac{3}{2}\left(\frac{t-1}{t}\beta_t\text{Wealth}_{t-1} + \frac{G}{2C^2 t}\text{Wealth}_{t-1}\right) \leq \frac{3}{2}\left(1 + \frac{G^2}{2C^2}\right)\bar{R} \leq 2\sqrt{2}\bar{R}.$$

(iii) If $\beta_t = 1/(C\sqrt{2(t-1)})$, then

$$\beta_{t+1} \leq 1/(C\sqrt{2t}) \leq 1/(C\sqrt{2(t-1)}) = \beta_t,$$

$$\tilde{x}_{t+1} \leq (1 - g_t\beta_t)\beta_{t+1}\text{Wealth}_{t-1} \leq 2\beta_t\text{Wealth}_{t-1} \leq 2\bar{R}.$$

Step 3 Prove that for all $t \geq 1$, $\text{Wealth}_t \leq 4\bar{R}C\sqrt{t}$.

Considering β_{t+1}, there are three cases: (1) $\beta_{t+1} = 1/(C\sqrt{2t})$; (2) $\beta_{t+1} = \hat{\beta}_{t+1}$; and (3) $\beta_{t+1} = 0$. For the first case, this result follows from $\tilde{x}_{t+1} = \beta_{t+1}\text{Wealth}_t \leq 2\sqrt{2}\bar{R}$. Now consider the second case.

$$\begin{aligned}\log\text{Wealth}_t &\leq \log\varepsilon + \sum_{i=1}^{t}\log(1 - \tilde{g}_i\beta_i) \\ &\leq \log\varepsilon - \sum_{i=1}^{t}\tilde{g}_i\beta_i \\ &= \log\varepsilon - \sum_{i=1}^{t}\left(\tilde{g}_i\beta_i + C^2\beta_i^2\right) + C^2\sum_{i=1}^{t}\beta_i^2.\end{aligned}$$

β_t is the output of Follow the Leader (FTL) on the strongly convex losses $\psi_t(\beta) = \tilde{g}_t\beta + C^2\beta^2 + I\{0 \leq \beta \leq 1/(C\sqrt{2t})\}(\beta)$, where $I\{0 \leq \beta \leq 1/(C\sqrt{2t})\}(\beta)$ is a convex function of β that equals 0 when $0 \leq \beta \leq 1/(C\sqrt{2t})$ and infinity otherwise. Therefore we can use standard FTL results to show that the regret is non-negative.

Let $F_t(\beta) = \sum_{i=1}^{t-1} \psi_i(\beta)$, then $\beta_t \in \arg\min F_t(\beta)$. From Lemma 7.1 of (Orabona, 2020), for any $u \in \mathbb{R}$,

$$\sum_{i=1}^{t} [\psi_i(\beta_i) - \psi_i(u)] = \sum_{i=1}^{t} [F_i(\beta_i) - F_{i+1}(\beta_{i+1}) + \psi_i(\beta_i)] + F_{t+1}(\beta_{t+1}) - F_{t+1}(u).$$

Note that if $u = \beta_{t+1}$, we have RHS ≥ 0. Therefore,

$$\begin{aligned}
\log \mathrm{Wealth}_t &\leq \log \varepsilon - \min_{0 \leq \beta \leq 1/(C\sqrt{2t})} \sum_{i=1}^{t} \left(\tilde{g}_i \beta + C^2 \beta^2\right) + C^2 \sum_{i=1}^{t} \beta_i^2 \\
&\leq \log \varepsilon - \min_{\beta \in \mathbb{R}} \sum_{i=1}^{t} \left(\tilde{g}_i \beta + C^2 \beta^2\right) + C^2 \sum_{i=1}^{t} \beta_i^2 \\
&\leq \log \varepsilon + \frac{\left(\sum_{i=1}^{t} \tilde{g}_i\right)^2}{4C^2 t} + \frac{1}{2} \sum_{\tau=1}^{t-1} \tau^{-1}.
\end{aligned}$$

The last term is bounded by $(1 + \log t)/2$. From the assumption of the second case, $|\sum_{i=1}^{t} \tilde{g}_i| < C\sqrt{2t}$. Combining everything we have $\log \mathrm{Wealth}_t \leq 1 + \log \varepsilon + (\log t)/2$ and $\mathrm{Wealth}_t \leq e\varepsilon\sqrt{t} \leq e\bar{R}C\sqrt{t}$.

Finally consider the third case. Same as the above, we have

$$\log \mathrm{Wealth}_t \leq \log \varepsilon - \min_{0 \leq \beta \leq 1/(C\sqrt{2t})} \sum_{i=1}^{t} \left(\tilde{g}_i \beta + C^2 \beta^2\right) + C^2 \sum_{i=1}^{t} \beta_i^2.$$

Since $\beta_{t+1} = 0$, we have $\sum_{i=1}^{t} \tilde{g}_i \geq 0$. Therefore,

$$\log \mathrm{Wealth}_t \leq \log \varepsilon + C^2 \sum_{i=1}^{t} \beta_i^2 \leq \log \varepsilon + \frac{1}{2}(1 + \log t),$$

and $\mathrm{Wealth}_t \leq \sqrt{e}\bar{R}C\sqrt{t}$. □

B.3.2 Proof of Theorem 1

Now we are ready to prove Theorem 1, the performance guarantee of Algorithm 1. This is our first main theoretical result.

Theorem 1. *For all $\lambda, \gamma \geq 0$, $G > 0$ and $0 < \varepsilon \leq G\bar{R}$, with any loss sequence such that $|g_t| \leq G$ for all t, applying Algorithm 1 yields the following guarantee.*

1. *For all $T \in \mathbb{N}_+$ and $u \in \mathcal{V}_{1d}$, with C defined in Line 1 of the algorithm,*

$$\sum_{t=1}^{T} \left(g_t x_t - g_t u + \lambda |x_t - x_{t+1}| + \frac{\gamma}{\sqrt{t}} |x_t|\right) \leq \varepsilon + uC\sqrt{2T} \left(\frac{3}{2} + \log \frac{\sqrt{2}uCT^{5/2}}{\varepsilon}\right).$$

2. *For all $a \leq b$, $\sum_{t=a}^{b} |x_t - x_{t+1}| \leq 48\bar{R}\sqrt{b - a + 1}$.*

Proof of Theorem 1. We prove the two parts of Theorem 1 separately, starting from the second part.

Combining Lemma B.4 and Lemma B.5, for all $t \geq 2$,

$$|\tilde{x}_t - \tilde{x}_{t+1}| \leq \frac{6}{Ct} \cdot 4\bar{R}C\sqrt{t-1} \leq 24\bar{R}\frac{1}{\sqrt{t}}.$$

For $t = 1$, the same result can be verified. Therefore, for all $[a : b] \subset [1 : T]$,

$$\sum_{t=a}^{b} |x_t - x_{t+1}| \leq 24\bar{R} \sum_{t=a}^{b} \frac{1}{\sqrt{t}} \leq 24\bar{R} \int_{a-1}^{b} \frac{1}{\sqrt{x}} dx \leq 24\bar{R}\left(2\sqrt{b} - 2\sqrt{a-1}\right) \leq 48\bar{R}\sqrt{b - a + 1}.$$

The fourth inequality is due to $\sqrt{b} - \sqrt{a-1} \leq \sqrt{b - a + 1}$.

Now consider the proof of the first part of the theorem. Due to the complexity, we proceed in steps.

Step 1 The overall strategy

The considered bound does not rely on the bounded domain, therefore the first step is to apply the reduction from constrained OLO to unconstrained OLO (Lemma B.1) and the contraction property of Euclidean projection to show that

$$\sum_{t=1}^{T}\left(g_t x_t - g_t u + \lambda\left|x_t - x_{t+1}\right| + \frac{\gamma}{\sqrt{t}}\left|x_t\right|\right) \leq \sum_{t=1}^{T}\left(\tilde{g}_t\tilde{x}_t - \tilde{g}_t u + \lambda\left|\tilde{x}_t - \tilde{x}_{t+1}\right| + \frac{\gamma}{\sqrt{t}}\left|\tilde{x}_t\right|\right). \tag{4}$$

Note that Wealth_{t-1} is positive due to Lemma B.2, and $\beta_t \geq 0$ from our construction. Therefore, $\tilde{x}_t \geq 0$. From here, we can focus on bounding the RHS of (4) with $|\tilde{x}_t|$ replaced by $\tilde{x}_t$. Also note that $|\tilde{g}_t| \leq |g_t| \leq G$ from Lemma B.1.

From (1), we can rewrite wealth as

$$\text{Wealth}_T = \varepsilon - \sum_{t=1}^{T}\left(\tilde{g}_t\tilde{x}_t + \lambda\left|\tilde{x}_t - \tilde{x}_{t+1}\right| + \frac{\gamma}{\sqrt{t}}\tilde{x}_t\right).$$

If we guarantee $\text{Wealth}_T \geq F(-\sum_{t=1}^{T}\tilde{g}_t)$ for an arbitrary function F, then

$$\begin{aligned}\sum_{t=1}^{T}\left(\tilde{g}_t\tilde{x}_t - \tilde{g}_t u + \lambda\left|\tilde{x}_t - \tilde{x}_{t+1}\right| + \frac{\gamma}{\sqrt{t}}\tilde{x}_t\right) &= \varepsilon + \left\langle -\sum_{t=1}^{T}\tilde{g}_t, u\right\rangle - \text{Wealth}_T\\ &\leq \varepsilon + \left\langle -\sum_{t=1}^{T}\tilde{g}_t, u\right\rangle - F\left(-\sum_{t=1}^{T}\tilde{g}_t\right)\\ &\leq \varepsilon + \sup_{X\in\mathbb{R}}\left(\langle X, u\rangle - F\left(X\right)\right) = \varepsilon + F^*(u),\end{aligned}$$

where F^* is the Fenchel conjugate of F. Therefore, our goal is to find such an lower bound for Wealth_T, and then take its Fenchel conjugate.

Step 2 Recursion on the wealth update

Now consider (1). There are two cases: (i) $\beta_t\text{Wealth}_{t-1} \geq \beta_{t+1}\text{Wealth}_t$; (ii) $\beta_t\text{Wealth}_{t-1} < \beta_{t+1}\text{Wealth}_t$. If $\beta_t\text{Wealth}_{t-1} \geq \beta_{t+1}\text{Wealth}_t$, then

$$(1-\lambda\beta_{t+1})\text{Wealth}_t = (1 - \tilde{g}_t\beta_t - \lambda\beta_t - \gamma\beta_t/\sqrt{t})\text{Wealth}_{t-1},$$

$$\log\text{Wealth}_t = \log\text{Wealth}_{t-1} + \log[1 - \beta_t(\tilde{g}_t + \lambda + \gamma/\sqrt{t})] - \log(1-\lambda\beta_{t+1}).$$

Note that $\beta_t|\tilde{g}_t + \lambda + \gamma/\sqrt{t}| \leq 1/2$ and $\lambda\beta_{t+1} < 1$. Applying $\log(1-x) \geq -x - x^2$ for all $x \leq 1/2$ and $\log(1+x) \leq x$ for all $x > 1$, we have

$$\begin{aligned}\log\text{Wealth}_t &\geq \log\text{Wealth}_{t-1} - \beta_t(\tilde{g}_t + \lambda + \gamma/\sqrt{t}) - \beta_t^2(\tilde{g}_t + \lambda + \gamma/\sqrt{t})^2 + \lambda\beta_{t+1}\\ &\geq \log\text{Wealth}_{t-1} - \tilde{g}_t\beta_t - \gamma\beta_t/\sqrt{t} - C^2\beta_t^2 + \lambda(\beta_{t+1} - \beta_t).\end{aligned}$$

Similarly, if $\beta_t\text{Wealth}_{t-1} < \beta_{t+1}\text{Wealth}_t$, then

$$\log\text{Wealth}_t \geq \log\text{Wealth}_{t-1} - \tilde{g}_t\beta_t - \gamma\beta_t/\sqrt{t} - C^2\beta_t^2 + \lambda(\beta_t - \beta_{t+1}).$$

Therefore, combining both cases, we have

$$\log\text{Wealth}_t \geq \log\text{Wealth}_{t-1} - \tilde{g}_t\beta_t - \gamma\beta_t/\sqrt{t} - C^2\beta_t^2 + \lambda|\beta_t - \beta_{t+1}|,$$

and summed over $[1:T]$,

$$\log\text{Wealth}_T \geq \log\varepsilon - \sum_{t=1}^{T}\tilde{g}_t\beta_t - C^2\sum_{t=1}^{T}\beta_t^2 - \gamma\sum_{t=1}^{T}\frac{\beta_t}{\sqrt{t}} - \lambda\sum_{t=1}^{T}|\beta_t - \beta_{t+1}|. \tag{5}$$

Step 3 Bounding the sums on the RHS of (5)

We start from the first two sums on the RHS of (5). β_t is the output of Follow the Leader (FTL) on the strongly convex losses $\psi_t(\beta) = \tilde{g}_t\beta + C^2\beta^2 + I\{0 \leq \beta \leq 1/(C\sqrt{2t})\}(\beta)$, where $I\{0 \leq \beta \leq 1/(C\sqrt{2t})\}(\beta)$ is a convex function of β that equals 0 when $0 \leq \beta \leq 1/(C\sqrt{2t})$ and infinity otherwise. Note that ψ_t is $2C^2$-strongly convex, therefore a standard result shows that the regret of this FTL problem is logarithmic in T. Concretely, from Corollary 7.17 of (Orabona, 2020),

$$\sum_{t=1}^{T}\left(\tilde{g}_t\beta_t + C^2\beta_t^2\right) - \min_{0\leq u\leq 1/(C\sqrt{2T})}\sum_{t=1}^{T}\left(\tilde{g}_t u + C^2u^2\right) \leq \frac{G^2}{4C^2}\left(1+\log T\right).$$

Moreover, taking $u = 1/(C\sqrt{2T})$,

$$\min_{0\leq u\leq 1/(C\sqrt{2T})}\sum_{t=1}^{T}\left(\tilde{g}_t u + C^2u^2\right) \leq \frac{\sum_{t=1}^{T}\tilde{g}_t}{C\sqrt{2T}} + \frac{1}{2}.$$

As for the other sums in (5),

$$\sum_{t=1}^{T}\frac{\beta_t}{\sqrt{t}} = \frac{1}{\sqrt{2}C}\sum_{t=1}^{T}\frac{1}{t} \leq \frac{1}{\sqrt{2}C}(1+\log T).$$

Applying Lemma B.3,

$$\sum_{t=1}^{T}|\beta_t - \beta_{t+1}| \leq \frac{2}{C}\sum_{t=1}^{T}\frac{1}{t} \leq \frac{2}{C}(1+\log T).$$

Plugging the above into (5),

$$\log \text{Wealth}_T \geq \log\varepsilon - \frac{\sum_{t=1}^{T}\tilde{g}_t}{C\sqrt{2T}} - 2(1+\log T) - \frac{1}{2},$$

$$\text{Wealth}_T \geq \frac{\varepsilon}{\exp(5/2)\cdot T^2}\exp\left(-\frac{\sum_{t=1}^{T}\tilde{g}_t}{C\sqrt{2T}}\right).$$

Step 4 Taking Fenchel conjugate

From the Fechel conjugate table, if $f(x) = a\exp(bx)$ with $a, b > 0$, then for all $\theta \geq 0$,

$$f^*(\theta) = \frac{\theta}{b}\left(\log\frac{\theta}{ab} - 1\right).$$

Applying this result on

$$F(x) = \frac{\varepsilon}{\exp(5/2)\cdot T^2}\exp\left(\frac{x}{C\sqrt{2T}}\right),$$

for all $u \geq 0$ we have

$$F^*(u) = uC\sqrt{2T}\left(\frac{3}{2} + \log\frac{\sqrt{2}uCT^{5/2}}{\varepsilon}\right).$$

Combining the above with Step 1 completes the proof. □

B.4 Analysis of Algorithm 2

Algorithm 2 extends the one-dimensional coin-betting-based OLO algorithm to higher dimensions via a polar decomposition. Here we incorporate movement cost into the analysis of (Cutkosky and Orabona, 2018).

Theorem 5. *For all $\lambda \geq 0$, $G > 0$ and $0 < \varepsilon \leq GR$, applying Algorithm 2 yields the following performance guarantee:*

1. For all $T \in \mathbb{N}_+$ and $u \in \mathsf{B}^d(0, R)$,

$$\sum_{t=1}^{T} \langle g_t, x_t - u\rangle + \lambda \sum_{t=1}^{T-1} \|x_t - x_{t+1}\| \le \varepsilon + \|u\| \tilde{O}\left[(G+\lambda)\sqrt{T}\right],$$

where $\tilde{O}(\cdot)$ subsumes logarithmic factors on u, G, λ, T and ε^{-1}.

2. For all $b \ge a \ge 1$,

$$\sum_{t=a}^{b-1} \|x_t - x_{t+1}\| \le 50R\sqrt{b-a}.$$

Proof of Theorem 5. We only consider the case of $u \neq 0$. If $u = 0$, the result can be easily verified. Notice that $|\langle g_t, z_t\rangle| \le G$, therefore we can apply Theorem 1 on $\mathcal{A}_r$.

$$\begin{aligned}
&\sum_{t=1}^{T} \langle g_t, y_t z_t - u\rangle + \lambda \sum_{t=1}^{T-1} \|y_t z_t - y_{t+1} z_{t+1}\| \\
\le\ &\sum_{t=1}^{T} \left(\langle g_t, z_t\rangle y_t - \langle g_t, z_t\rangle \|u\|\right) + \|u\| \sum_{t=1}^{T} \left\langle g_t, z_t - \frac{u}{\|u\|}\right\rangle + \lambda \sum_{t=1}^{T-1} |y_t - y_{t+1}| \|z_{t+1}\| + \lambda \sum_{t=1}^{T-1} \|z_t - z_{t+1}\| |y_t| \\
\le\ &\sum_{t=1}^{T} \left(\langle g_t, z_t\rangle y_t - \langle g_t, z_t\rangle \|u\|\right) + \lambda \sum_{t=1}^{T-1} |y_t - y_{t+1}| + \sum_{t=1}^{T-1} \frac{\lambda}{\sqrt{t}} y_t + \|u\| \sum_{t=1}^{T} \left\langle g_t, z_t - \frac{u}{\|u\|}\right\rangle. \qquad (6)
\end{aligned}$$

The last inequality is due to $\|z_{t+1}\| \le 1$ and $\|z_t - z_{t+1}\| \le \eta_t G = 1/\sqrt{t}$.

The first three terms of (6) are bounded by Theorem 1,

$$\begin{aligned}
&\sum_{t=1}^{T} \left(\langle g_t, z_t\rangle y_t - \langle g_t, z_t\rangle \|u\|\right) + \lambda \sum_{t=1}^{T-1} |y_t - y_{t+1}| + \sum_{t=1}^{T-1} \frac{\lambda}{\sqrt{t}} y_t \\
&\qquad\qquad \le \varepsilon + \|u\| (G + 2\lambda)\sqrt{2T}\left(\frac{3}{2} + \log \frac{\sqrt{2}\|u\| (G+2\lambda) T^{5/2}}{\varepsilon}\right).
\end{aligned}$$

As for the last term of (6), we can use the standard OGD regret bound. From Section 4.2.1 of (Orabona, 2020),

$$\sum_{t=1}^{T} \left\langle g_t, z_t - \frac{u}{\|u\|}\right\rangle \le \frac{3}{2} G\sqrt{T}.$$

Combining everything so far yields the first part of the theorem.

As for the second part of the theorem, for all $b \ge a \ge 1$,

$$\sum_{t=a}^{b-1} \|x_t - x_{t+1}\| \le \sum_{t=a}^{b-1} \left(|y_t - y_{t+1}| + \frac{R}{\sqrt{t}}\right) \le 50R\sqrt{b-a}.$$

The last inequality is due to Theorem 1 and $\sum_{t=a}^{b-1} 1/\sqrt{t} \le 2\sqrt{b-a}$. □

C DETAILS ON STRONGLY ADAPTIVE OCOM

This section provides detailed analysis of our strongly adaptive OCOM algorithm. We first present the performance guarantees of our subroutines based on Algorithm 3. Then, we introduce the complete version of our meta-algorithm (Algorithm 7) and present its analysis.

C.1 Analysis of Algorithm 3

Algorithm 3 is used to define our two-part subroutine (on GC intervals). The idea of adaptively slowing down the base algorithm is inspired by Algorithm 7 of (Cutkosky, 2018) for memoryless OLO. Here we make two improvements: (i) incorporating movement costs; (ii) using this framework to achieve better dependence on problem constants.

Theorem 6. *For all $\lambda, G > 0$ and $0 < \varepsilon \leq G$, Subroutine-1d defined from Algorithm 3 yields the following performance guarantee:*

1. *For all $T \in \mathbb{N}_+$ and $u \in [0,1]$,*

$$\sum_{t=1}^{T} g_t(x_t - u) + \lambda \sum_{t=1}^{T-1} |x_t - x_{t+1}| \leq \varepsilon + |u| \, \tilde{O}\left(\max\{\lambda, G\} + \sqrt{\max\{\lambda, G\} \sum_{t=1}^{T} |g_t|}\right),$$

where $\tilde{O}(\cdot)$ subsumes logarithmic factors on u, G, λ, T and ε^{-1}.

2. *For all $b \geq a \geq 1$,*

$$\sum_{t=a}^{b-1} \|x_t - x_{t+1}\| \leq 48 \left(1 + \sqrt{\frac{\sum_{t=a}^{b-1} |g_t|}{\max\{\lambda, G\}}}\right).$$

Theorem 7. *For all $\lambda, G > 0$ and $0 < \varepsilon \leq GR$, Subroutine-ball defined from Algorithm 3 yields the following performance guarantee:*

1. *For all $T \in \mathbb{N}_+$ and $u \in \mathsf{B}^d(0, R)$,*

$$\sum_{t=1}^{T} \langle g_t, x_t - u \rangle + \lambda \sum_{t=1}^{T-1} \|x_t - x_{t+1}\| \leq \varepsilon + \|u\| \, \tilde{O}\left(\max\{\lambda, G\} + \sqrt{\max\{\lambda, G\} \sum_{t=1}^{T} \|g_t\|}\right),$$

where $\tilde{O}(\cdot)$ subsumes logarithmic factors on u, G, λ, T and ε^{-1}.

2. *For all $b \geq a \geq 1$,*

$$\sum_{t=a}^{b-1} \|x_t - x_{t+1}\| \leq 50R \left(1 + \sqrt{\frac{\sum_{t=a}^{b-1} \|g_t\|}{\max\{\lambda, G\}}}\right).$$

We only prove the guarantee on Subroutine-ball (Theorem 7). The guarantee on Subroutine-1d (Theorem 6) is similar, therefore the proof is omitted.

Proof of Theorem 7. Consider the first part of the theorem. Let i_T be the index i at the beginning of the T-th round, and let $Z_1, \ldots, Z_{i_T}$ be their final value at the end of the algorithm. Notice that

$$\sum_{t=1}^{T} \langle g_t, x_t - u \rangle + \lambda \sum_{t=1}^{T-1} \|x_t - x_{t+1}\| = \sum_{i=1}^{i_T} \langle Z_i, w_i - u \rangle + \lambda \sum_{i=1}^{i_T - 1} \|w_i - w_{i+1}\|.$$

For the RHS we can use Theorem 5, since for all i, $\|Z_i\| \leq \max\{\lambda, G\} + G$. The remaining task is to bound i_T. Note that $\sum_{i=1}^{i_T} \|Z_i\| \leq \sum_{i=1}^{T} \|g_t\|$ and $\|Z_i\| > \max\{\lambda, G\}$ for all $i < i_T$, therefore $i_T \leq 1 + (\sum_{t=1}^{T} \|g_t\|)/\max\{\lambda, G\}$. Plugging this into Theorem 5 completes the proof of the first part.

As for the second part of the theorem, let i_a, i_b be the index i at the beginning of the a-th and the b-th round.

$$\sum_{t=a}^{b-1} \|x_t - x_{t+1}\| = \sum_{i=i_a}^{i_b - 1} \|w_i - w_{i+1}\|.$$

Next consider $i_b - i_a$. Let $Z^*_{i_a}$ and $Z^*_{i_b}$ be the value of accumulators Z_{i_a} and Z_{i_b} at the beginning of the a-th round and the b-th round, respectively. Note that

$$\left\|Z_{i_a} - Z^*_{i_a}\right\| + \left\|Z^*_{i_b}\right\| + \sum_{i=i_a+1}^{i_b - 1} \|Z_i\| \leq \sum_{t=a}^{b-1} \|g_t\|,$$

and $\|Z_i\| > \max\{\lambda, G\}$ for all $i \in [i_a+1, i_b-1]$. Therefore, $i_b - i_a \le 1 + (\sum_{t=a}^{b-1}\|g_t\|)/\max\{\lambda, G\}$. Applying the second part of Theorem 5 completes the proof. □

C.2 Analysis of the meta-algorithm

Now we proceed to our meta-algorithm for strongly adaptive OCOM. The pseudo-code is Algorithm 7. Before providing its performance guarantee, we present a lemma that explains the adopted projection scheme. (Line 6 and 15)

Algorithm 7 The meta-algorithm for strongly adaptive OCOM. (The complete version of Algorithm 4)

```
Require: Time horizon $T \ge 1$ and a hyperparameter $\varepsilon_0 > 0$.
 1: Define a constant $\lambda = LH(H+1)$.
 2: for $t = 1, \dots, T$ do
 3:     Find the $(k, i)$ index pair for all the GC intervals that start in the $t$-th round. For each, (i) initialize a copy
        of Subroutine-ball as $\mathcal{A}_B^k$, with hyperparameters $(\lambda, 2^k\varepsilon_0, \tilde{G})$; and (ii) initialize a copy of Subroutine-1d as
        $\mathcal{A}_{1d}^k$, with hyperparameters $(\lambda R, 2^k\varepsilon_0, \tilde{G}R)$. If $\mathcal{A}_B^k$ and $\mathcal{A}_{1d}^k$ already exist in the memory, overwrite them.
 4:     Define $K_t = \lceil \log_2(t+1) \rceil - 1$. Let $\tilde{x}_t^{(K_t+1)} = 0 \in \mathbb{R}^d$.
 5:     for $k = K_t, \dots, 0$ do
 6:         Let $x_t^{(k+1)} = \Pi_{\mathsf{B}^d(0,R)}(\tilde{x}_t^{(k+1)})$.
 7:         Query a prediction from $\mathcal{A}_B^k$ and assign it to $w_t^{(k)}$; query a prediction from $\mathcal{A}_{1d}^k$ and assign it to $z_t^{(k)}$.
 8:         Let $\tilde{x}_t^{(k)} = (1 - z_t^{(k)})x_t^{(k+1)} + w_t^{(k)}$.
 9:     end for
10:     Let $\tilde{x}_t = \tilde{x}_t^{(0)}$, predict $x_t = \Pi_{\mathcal{V}}(\tilde{x}_t)$, suffer $l_t(x_{t-H:t})$, receive $l_t$.
11:     Obtain a subgradient $g_t \in \partial \tilde{l}_t(x_t)$. Define a surrogate loss function $h_t$ as
            $$h_t(x) = \begin{cases} \langle g_t, x\rangle, & \text{if } \langle g_t, \tilde{x}_t\rangle \ge \langle g_t, x_t\rangle, \\ \langle g_t, x\rangle + \langle g_t, x_t - \tilde{x}_t\rangle \frac{\|x - \Pi_{\mathcal{V}}(x)\|}{\|x_t - \tilde{x}_t\|}, & \text{otherwise.} \end{cases}$$
12:     Obtain a subgradient $\tilde{g}_t \in \partial h_t(\tilde{x}_t)$. Let $g_t^{(0)} = \tilde{g}_t$.
13:     for $k = 0, \dots, K_t$ do
14:         Return $g_t^{(k)}$ to $\mathcal{A}_B^k$, and $-\langle g_t^{(k)}, x_t^{(k+1)}\rangle$ to $\mathcal{A}_{1d}^k$ as the loss gradients.
15:         Let $e_t^{(k+1)} = \tilde{x}_t^{k+1}/\|\tilde{x}_t^{k+1}\|$, and
                $$g_t^{(k+1)} = \begin{cases} g_t^{(k)}, & \text{if } \langle g_t^{(k)}, \tilde{x}_t^{(k+1)}\rangle \ge \langle g_t^{(k)}, x_t^{(k+1)}\rangle, \\ g_t^{(k)} - \left\langle g_t^{(k)}, e_t^{(k+1)}\right\rangle e_t^{(k+1)}, & \text{otherwise.} \end{cases}$$
16:     end for
17: end for
```

Lemma C.1. *For all t,*

1. $\|g_t^{(K_t+1)}\| \le \|g_t^{(K_t)}\| \le \dots \le \|g_t^{(0)}\| \le \|g_t\| \le \tilde{G}$.

2. *For all $k \in [0 : K_t]$ and $x \in \mathcal{V}$,* $\left\langle g_t^{(k)}, x_t^{(k+1)} - x\right\rangle \le \left\langle g_t^{(k+1)}, \tilde{x}_t^{(k+1)} - x\right\rangle$.

Observe that Line 6 and 15 of Algorithm 7 are essentially applying Algorithm 6 on the unprojected prediction $\tilde{x}_t^{(k+1)}$. Therefore, the proof of Lemma C.1 follows from recursively applying Lemma B.1. Line 11 follows a similar principle.

Now we are ready to prove the performance guarantee.

Theorem 2. *Consider running our OCOM algorithm (the complete version, Algorithm 7) for T rounds. If*

$\varepsilon_0 = \tilde{G}R/(T+1)$, then on any time interval $\mathcal{I} = [a:b] \subset [1:T]$,

$$\sum_{t=a}^{b} l_t(x_{t-H:t}) - \min_{x\in\mathcal{V}} \sum_{t=a}^{b} \tilde{l}_t(x) = O(RLH^3 \log|\mathcal{I}|) + \tilde{O}\left(RLH^2 + RH\sqrt{L\sum_{t=a}^{b}\|g_t\|}\right),$$

where $g_t \in \partial\tilde{l}_t(x_t)$, $O(\cdot)$ subsumes absolute constants, and $\tilde{O}(\cdot)$ subsumes poly-logarithmic factors on problem constants and T.

Proof of Theorem 2. Our strategy is to associate the regret of the meta-algorithm on any GC interval with the regret of the corresponding subroutines (Theorem 6 and Theorem 7). Then, applying these performance guarantees yields $\tilde{O}(\sqrt{|\mathcal{I}^{k,i}|})$ regret on all GC interval $\mathcal{I}^{k,i} \subset [1:T]$. This can be further extended to all general intervals $\mathcal{I} \subset [1:T]$ using an argument similar to (Daniely et al., 2015).

To this end, we proceed in steps. Let $\mathcal{I}^{k^*,i^*} = [q:s] \subset [1:T]$ be a GC interval with indices k^* and i^*. Since the amount of active GC intervals cannot increase in the duration of any GC interval, we can replace K_t for all $t \in \mathcal{I}^{k^*,i^*}$ by a constant K^*. In other words, for all $t \in \mathcal{I}^{k^*,i^*}$, $K_t = K^* \le \lfloor \log_2(T+1)\rfloor - 1$.

Step 1 Reducing to one-step movement.

We start from the Lipschitzness of l_t. For all t,

$$l_t(x_{t-H:t}) \le \tilde{l}_t(x_t) + L\sum_{h=1}^{H}\|x_{t-h} - x_t\| \le \tilde{l}_t(x_t) + L\sum_{h=1}^{H}\sum_{j=1}^{h}\|x_{t-j} - x_{t-j+1}\|.$$

Using the convexity of $\tilde{l}_t$, for all $x \in \mathcal{V}$,

$$\sum_{t=q}^{s}\left[l_t(x_{t-H:t}) - \tilde{l}_t(x)\right] \le \sum_{t=q}^{s}\langle g_t, x_t - x\rangle + L\sum_{t=q}^{s}\sum_{h=1}^{H}\sum_{j=1}^{h}\|x_{t-j} - x_{t-j+1}\|.$$

Observe that

$$\begin{aligned}
\sum_{t=q}^{s}\sum_{h=1}^{H}\sum_{j=1}^{h}\|x_{t-j} - x_{t-j+1}\| &\le \frac{1}{2}H(H+1)\sum_{t=q}^{s-1}\|x_t - x_{t+1}\| + \sum_{h=1}^{H}\frac{1}{2}(H+1-h)(H+2-h)\|x_{q-h} - x_{q-h+1}\| \\
&\le \frac{1}{2}H(H+1)\sum_{t=q}^{s-1}\|x_t - x_{t+1}\| + R\sum_{h=1}^{H}h(h+1) \\
&= \frac{1}{2}H(H+1)\sum_{t=q}^{s-1}\|x_t - x_{t+1}\| + \frac{1}{3}RH(H+1)(H+2).
\end{aligned}$$

Therefore, combining the above and plugging in λ for conciseness,

$$\begin{aligned}
\sum_{t=q}^{s}\left[l_t(x_{t-H:t}) - \tilde{l}_t(x)\right] &\le \sum_{t=q}^{s}\langle g_t, x_t - x\rangle + \frac{1}{2}LH(H+1)\sum_{t=q}^{s-1}\|x_t - x_{t+1}\| + O(RLH^3) \\
&\le \sum_{t=q}^{s}\langle \tilde{g}_t, \tilde{x}_t - x\rangle + \frac{\lambda}{2}\sum_{t=q}^{s-1}\|\tilde{x}_t - \tilde{x}_{t+1}\| + O(RLH^3),
\end{aligned}$$

where the last line is due to Lemma B.1 and the contraction property of Euclidean projection.

Step 2 Showing that the "temporary" prediction $x_t^{(k^*)}$ after combining $\mathcal{A}_B^{(k^*)}$ and $\mathcal{A}_{1d}^{(k^*)}$ is good enough for the considered GC interval, although improper.

Starting from the definition of $\tilde{x}_t^{(k^*)}$, for all $x \in \mathcal{V}$,

$$\left\langle g_t^{(k^*)}, \tilde{x}_t^{(k^*)} - x\right\rangle = \left\langle g_t^{(k^*)}, w_t^{(k^*)} - x\right\rangle + \left(-\left\langle g_t^{(k^*)}, x_t^{(k^*+1)}\right\rangle\right)\left(z_t^{(k^*)} - 1\right),$$

$$\begin{aligned}
\left\|\tilde{x}_t^{(k^*)}-\tilde{x}_{t+1}^{(k^*)}\right\| &= \left\|\left(1-z_t^{(k^*)}\right)x_t^{(k^*+1)}+w_t^{(k^*)}-\left(1-z_{t+1}^{(k^*)}\right)x_{t+1}^{(k^*+1)}-w_{t+1}^{(k^*)}\right\| \\
&\le \left\|\left(1-z_t^{(k^*)}\right)\left(x_t^{(k^*+1)}-x_{t+1}^{(k^*+1)}\right)\right\|+\left\|\left(z_t^{(k^*)}-z_{t+1}^{(k^*)}\right)x_{t+1}^{(k^*+1)}\right\|+\left\|w_t^{(k^*)}-w_{t+1}^{(k^*)}\right\| \\
&\le \left\|x_t^{(k^*+1)}-x_{t+1}^{(k^*+1)}\right\|+R\left|z_t^{(k^*)}-z_{t+1}^{(k^*)}\right|+\left\|w_t^{(k^*)}-w_{t+1}^{(k^*)}\right\| \\
&\le \left\|\tilde{x}_t^{(k^*+1)}-\tilde{x}_{t+1}^{(k^*+1)}\right\|+R\left|z_t^{(k^*)}-z_{t+1}^{(k^*)}\right|+\left\|w_t^{(k^*)}-w_{t+1}^{(k^*)}\right\| \qquad (7)\\
&\le R\sum_{k=k^*}^{K^*}\left|z_t^{(k)}-z_{t+1}^{(k)}\right|+\sum_{k=k^*}^{K^*}\left\|w_t^{(k)}-w_{t+1}^{(k)}\right\|.
\end{aligned}$$

The second line is due to triangle inequality. The third line is due to $z_t^{(k^*)}\in[0,1]$ and $\|x_{t+1}^{(k^*+1)}\|\le R$. The fourth line is due to the contraction of Euclidean projection, and the last line follows from a recursion. Combining the above,

$$\begin{aligned}
&\sum_{t=q}^{s}\left\langle g_t^{(k^*)},\tilde{x}_t^{(k^*)}-x\right\rangle+\frac{\lambda}{2}\sum_{t=q}^{s-1}\left\|\tilde{x}_t^{(k^*)}-\tilde{x}_{t+1}^{(k^*)}\right\| \\
&\quad\le \sum_{t=q}^{s}\left\langle g_t^{(k^*)},w_t^{(k^*)}-x\right\rangle+\frac{\lambda}{2}\sum_{t=q}^{s-1}\left\|w_t^{(k^*)}-w_{t+1}^{(k^*)}\right\|+\sum_{t=q}^{s}\left(-\left\langle g_t^{(k^*)},x_t^{(k^*+1)}\right\rangle\right)\left(z_t^{(k^*)}-1\right) \\
&\qquad +\frac{\lambda R}{2}\sum_{t=q}^{s-1}\left|z_t^{(k^*)}-z_{t+1}^{(k^*)}\right|+\frac{\lambda R}{2}\sum_{k=k^*+1}^{K^*}\sum_{t=q}^{s-1}\left|z_t^{(k)}-z_{t+1}^{(k)}\right|+\frac{\lambda}{2}\sum_{k=k^*+1}^{K^*}\sum_{t=q}^{s-1}\left\|w_t^{(k)}-w_{t+1}^{(k)}\right\|.
\end{aligned}$$

Note that from Lemma C.1, $\|g_t^{(k^*)}\|\le\|g_t^{(k^*-1)}\|\le\ldots\le\|g_t\|\le\tilde{G}$. Moreover, $\varepsilon_0\le\tilde{G}R/(T+1)$ leads to $2^{k^*}\varepsilon_0\le\tilde{G}R$. Therefore, we can use the performance guarantees of the subroutine for the sums on the RHS. Applying Part 1 of Theorem 6 and Theorem 7,

$$\sum_{t=q}^{s}\left\langle g_t^{(k^*)},w_t^{(k^*)}-x\right\rangle+\frac{\lambda}{2}\sum_{t=q}^{s-1}\left\|w_t^{(k^*)}-w_{t+1}^{(k^*)}\right\|\le 2^{k^*}\varepsilon_0+\tilde{O}\left(R\max\{\lambda,\tilde{G}\}+R\sqrt{\max\{\lambda,\tilde{G}\}\sum_{t=q}^{s}\|g_t\|}\right),$$

$$\sum_{t=q}^{s}\left(-\left\langle g_t^{(k^*)},x_t^{(k^*+1)}\right\rangle\right)\left(z_t^{(k^*)}-1\right)+\frac{\lambda R}{2}\sum_{t=q}^{s-1}\left|z_t^{(k^*)}-z_{t+1}^{(k^*)}\right|\le 2^{k^*}\varepsilon_0+\tilde{O}\left(R\max\{\lambda,\tilde{G}\}+R\sqrt{\max\{\lambda,\tilde{G}\}\sum_{t=q}^{s}\|g_t\|}\right).$$

Also note that GC intervals longer than $\mathcal{I}^{k^*,i^*}$ cannot be initialized in the duration of $\mathcal{I}^{k^*,i^*}$. Therefore applying Part 2 of Theorem 6 and Theorem 7, for all $k\in[k^*+1:K^*]$,

$$\sum_{t=q}^{s-1}\left\|w_t^{(k)}-w_{t+1}^{(k)}\right\|\le 50R\left(1+\sqrt{\frac{\sum_{t=q}^{s-1}\left\|g_t^{(k)}\right\|}{\max\{\lambda,\tilde{G}\}}}\right)\le 50R\left(1+\sqrt{\frac{\sum_{t=q}^{s}\|g_t\|}{\max\{\lambda,\tilde{G}\}}}\right).$$

$$\sum_{t=q}^{s-1}\left|z_t^{(k)}-z_{t+1}^{(k)}\right|\le 48\left(1+\sqrt{\frac{\sum_{t=q}^{s-1}\left\|g_t^{(k)}\right\|}{\max\{\lambda,\tilde{G}\}}}\right)\le 48\left(1+\sqrt{\frac{\sum_{t=q}^{s}\|g_t\|}{\max\{\lambda,\tilde{G}\}}}\right).$$

Notice that $K^*=O(\log T)$ and $\lambda\le\tilde{G}$ from our definition. Combining everything so far, we have

$$\sum_{t=q}^{s}\left\langle g_t^{(k^*)},\tilde{x}_t^{(k^*)}-x\right\rangle+\frac{\lambda}{2}\sum_{t=q}^{s-1}\left\|\tilde{x}_t^{(k^*)}-\tilde{x}_{t+1}^{(k^*)}\right\|\le 2^{k^*+1}\varepsilon_0+\tilde{O}\left(\lambda R+R\sqrt{\lambda\sum_{t=q}^{s}\|g_t\|}\right).$$

Intuitively, suppose we are allowed to predict the improper prediction $\tilde{x}_t^{(k^*)}$ on $\mathcal{I}^{k^*,i^*}$ that may not comply with the constraint $\mathcal{V}$, and suppose $g_t^{(k^*)}=g_t$. Then, the above result shows that on $\mathcal{I}^{k^*,i^*}$ we have the desirable $\tilde{O}(\sqrt{|\mathcal{I}^{k^*,i^*}|})$ regret bound. The rest of the proof aims to show that adding predictions from shorter subroutines does not ruin the performance on $\mathcal{I}^{k^*,i^*}$.

Step 3 Analyzing the effect of adding shorter subroutines.

The goal of this step is to quantify the difference between

$$\sum_{t=q}^{s}\left\langle g_t^{(k^*)}, \tilde{x}_t^{(k^*)}-x\right\rangle+\frac{\lambda}{2}\sum_{t=q}^{s-1}\left\|\tilde{x}_t^{(k^*)}-\tilde{x}_{t+1}^{(k^*)}\right\|,$$

and

$$\sum_{t=q}^{s}\left\langle g_t^{(0)}, \tilde{x}_t^{(0)}-x\right\rangle+\frac{\lambda}{2}\sum_{t=q}^{s-1}\left\|\tilde{x}_t^{(0)}-\tilde{x}_{t+1}^{(0)}\right\|.$$

For all $k\in[0:k^*-1]$, applying the definition of $\tilde{x}^{(k)}$ and Part 2 of Lemma C.1,

$$\begin{aligned}\left\langle g_t^{(k)}, \tilde{x}_t^{(k)}-x\right\rangle&=\left\langle g_t^{(k)}, x_t^{(k+1)}-x\right\rangle+\left\langle g_t^{(k)}, w_t^{(k)}-0\right\rangle+\left(-\left\langle g_t^{(k)}, x_t^{(k+1)}\right\rangle\right)\left(z_t^{(k)}-0\right)\\&\leq\left\langle g_t^{(k+1)}, \tilde{x}_t^{(k+1)}-x\right\rangle+\left\langle g_t^{(k)}, w_t^{(k)}-0\right\rangle+\left(-\left\langle g_t^{(k)}, x_t^{(k+1)}\right\rangle\right)\left(z_t^{(k)}-0\right).\end{aligned}$$

Similar to Equation (7),

$$\left\|\tilde{x}_t^{(k)}-\tilde{x}_{t+1}^{(k)}\right\|\leq\left\|\tilde{x}_t^{(k+1)}-\tilde{x}_{t+1}^{(k+1)}\right\|+R\left|z_t^{(k)}-z_{t+1}^{(k)}\right|+\left\|w_t^{(k)}-w_{t+1}^{(k)}\right\|.$$

Therefore,

$$\begin{aligned}\sum_{t=q}^{s}\left\langle g_t^{(k)}, \tilde{x}_t^{(k)}-x\right\rangle+\frac{\lambda}{2}\sum_{t=q}^{s-1}\left\|\tilde{x}_t^{(k)}-\tilde{x}_{t+1}^{(k)}\right\|&\leq\sum_{t=q}^{s}\left\langle g_t^{(k+1)}, \tilde{x}_t^{(k+1)}-x\right\rangle+\frac{\lambda}{2}\sum_{t=q}^{s-1}\left\|\tilde{x}_t^{(k+1)}-\tilde{x}_{t+1}^{(k+1)}\right\|\\&+\sum_{t=q}^{s}\left\langle g_t^{(k)}, w_t^{(k)}-0\right\rangle+\frac{\lambda}{2}\sum_{t=q}^{s-1}\left\|w_t^{(k)}-w_{t+1}^{(k)}\right\|\\&+\sum_{t=q}^{s}\left(-\left\langle g_t^{(k)}, x_t^{(k+1)}\right\rangle\right)\left(z_t^{(k)}-0\right)+\frac{\lambda R}{2}\sum_{t=q}^{s-1}\left|z_t^{(k)}-z_{t+1}^{(k)}\right|.\end{aligned}$$

We next bound the last four sums on the RHS using Theorem 6 and Theorem 7. Let $[a,b]$ be any GC interval of length 2^k contained in $[q:s]$. Note that by our definition, the subroutines $\mathcal{A}_B^k$ and $\mathcal{A}_{1d}^k$ are initialized at 0. That is, $w_a^{(k)}=w_{b+1}^{(k)}=0\in\mathbb{R}^d$, $z_a^{(k)}=z_{b+1}^{(k)}=0\in\mathbb{R}$. From Theorem 7,

$$\begin{aligned}\sum_{t=a}^{b}\left\langle g_t^{(k)}, w_t^{(k)}-0\right\rangle+\frac{\lambda}{2}\sum_{t=a}^{b}\left\|w_t^{(k)}-w_{t+1}^{(k)}\right\|&=\sum_{t=a}^{b}\left\langle g_t^{(k)}, w_t^{(k)}-0\right\rangle+\frac{\lambda}{2}\sum_{t=a}^{b-1}\left\|w_t^{(k)}-w_{t+1}^{(k)}\right\|+\frac{\lambda}{2}\left\|w_b^{(k)}\right\|\\&\leq\sum_{t=a}^{b}\left\langle g_t^{(k)}, w_t^{(k)}-0\right\rangle+\lambda\sum_{t=a}^{b-1}\left\|w_t^{(k)}-w_{t+1}^{(k)}\right\|\leq 2^k\varepsilon_0.\end{aligned}$$

Summed over all GC intervals of length 2^k contained in $[q:s]$,

$$\begin{aligned}\sum_{t=q}^{s}\left\langle g_t^{(k)}, w_t^{(k)}-0\right\rangle+\frac{\lambda}{2}\sum_{t=q}^{s-1}\left\|w_t^{(k)}-w_{t+1}^{(k)}\right\|&\leq\sum_{t=q}^{s}\left\langle g_t^{(k)}, w_t^{(k)}-0\right\rangle+\frac{\lambda}{2}\sum_{t=q}^{s}\left\|w_t^{(k)}-w_{t+1}^{(k)}\right\|\\&\leq 2^{k^*-k}\cdot 2^k\varepsilon_0=2^{k^*}\varepsilon_0.\end{aligned}$$

Similarly,

$$\sum_{t=q}^{s}\left(-\left\langle g_t^{(k)}, x_t^{(k+1)}\right\rangle\right)\left(z_t^{(k)}-0\right)+\frac{\lambda R}{2}\sum_{t=q}^{s-1}\left|z_t^{(k)}-z_{t+1}^{(k)}\right|\leq 2^{k^*}\varepsilon_0.$$

Therefore,

$$\sum_{t=q}^{s}\left\langle g_t^{(k)}, \tilde{x}_t^{(k)}-x\right\rangle+\frac{\lambda}{2}\sum_{t=q}^{s-1}\left\|\tilde{x}_t^{(k)}-\tilde{x}_{t+1}^{(k)}\right\|\leq\sum_{t=q}^{s}\left\langle g_t^{(k+1)}, \tilde{x}_t^{(k+1)}-x\right\rangle+\frac{\lambda}{2}\sum_{t=q}^{s-1}\left\|\tilde{x}_t^{(k+1)}-\tilde{x}_{t+1}^{(k+1)}\right\|+2^{k^*+1}\varepsilon_0.$$

Completing the recursion, we have

$$\begin{aligned}\sum_{t=q}^{s}\left\langle g_t^{(0)},\tilde{x}_t^{(0)}-x\right\rangle+\frac{\lambda}{2}\sum_{t=q}^{s-1}\left\|\tilde{x}_t^{(0)}-\tilde{x}_{t+1}^{(0)}\right\| &\leq \sum_{t=q}^{s}\left\langle g_t^{(k^*)},\tilde{x}_t^{(k^*)}-x\right\rangle+\frac{\lambda}{2}\sum_{t=q}^{s-1}\left\|\tilde{x}_t^{(k^*)}-\tilde{x}_{t+1}^{(k^*)}\right\|+k^*\cdot 2^{k^*+1}\varepsilon_0\\ &\leq (k^*+1)\cdot 2^{k^*+1}\varepsilon_0+\tilde{O}\left(\lambda R+R\sqrt{\lambda\sum_{t=q}^{s}\|g_t\|}\right)\\ &\leq \tilde{O}\left(\lambda R+R\sqrt{\lambda\sum_{t=q}^{s}\|g_t\|}\right),\end{aligned}$$

where the last line follows from $(k^*+1)\cdot 2^{k^*+1}\varepsilon_0\leq 2\tilde{G}R\lceil\log_2(T+1)\rceil$. Plugging this into the result from Step 1,

$$\sum_{t=q}^{s}\left[l_t(x_{t-H:t})-\tilde{l}_t(x)\right]\leq O(RLH^3)+\tilde{O}\left(\lambda R+R\sqrt{\lambda\sum_{t=q}^{s}\|g_t\|}\right).$$

This bound holds for all GC intervals contained in $[1:T]$. The final step is to extend this property to general intervals, following the classical idea from (Daniely et al., 2015).

Step 4 Extension to general intervals.

From Lemma 5 of (Daniely et al., 2015), we have the following result: any interval $\mathcal{I}\subset[1:T]$ can be partitioned into two finite sequences of disjoint and consecutive GC intervals, denoted as $(\mathcal{I}_{-k},\ldots,\mathcal{I}_0)$ and $(\mathcal{I}_1,\ldots,\mathcal{I}_p)$. Moreover, for all $i\geq 1$, $|\mathcal{I}_{-i}|/|\mathcal{I}_{-i+1}|\leq 1/2$; for all $i\geq 2$, $|\mathcal{I}_i|/|\mathcal{I}_{i-1}|\leq 1/2$.

The strongly adaptive regret of our meta-algorithm (Equation 2) over $\mathcal{I}$ can be bounded by the sum of regret over $(\mathcal{I}_{-k},\ldots,\mathcal{I}_0)$ and $(\mathcal{I}_1,\ldots,\mathcal{I}_p)$. For an index i, denote the regret over $\mathcal{I}_i$ as Regret_i. Then,

$$\sum_{t\in\mathcal{I}}l_t(x_{t-H:t})-\min_{x\in\mathcal{V}}\sum_{t\in\mathcal{I}}\tilde{l}_t(x)\leq\sum_{i=0}^{k}\text{Regret}_{-i}+\sum_{i=1}^{p}\text{Regret}_i,$$

where $k\leq\log_2|\mathcal{I}|$ and $p\leq 1+\log_2|\mathcal{I}|$. Consider the first sum on the RHS,

$$\begin{aligned}\sum_{i=0}^{k}\text{Regret}_{-i}&\leq(k+1)O(RLH^3)+\sum_{i=0}^{k}\tilde{O}\left(\lambda R+R\sqrt{\lambda\sum_{t\in\mathcal{I}_{-i}}\|g_t\|}\right)\\ &\leq O(RLH^3\log|\mathcal{I}|)+\tilde{O}\left(\lambda R+R\sqrt{\lambda\sum_{t\in\mathcal{I}}\|g_t\|}\right).\end{aligned}$$

The second sum can be bounded similarly. Combining everything completes the proof. □

D DETAILS ON ADVERSARIAL TRACKING CONTROL

This section presents our results on adversarial tracking. We first prove its reduction to strongly adaptive OCOM. Then, we consider a special case that induces a non-comparative tracking error bound.

D.1 Details on the reduction

We present a few lemmas before proving Theorem 3. First we bound the norm of state and action. Similar to Section 4 we expand the dependence of state on past actions; that is, let $x_t(u_{1:t-1})$ be the state induced by the action sequence $u_{1:t-1}$. Note that $u_{1:t-1}$ is a dummy variable, not necessarily a comparator or the action sequence generated by Algorithm 5.

Lemma D.1. *For all $t \geq 1$, with any $u_{1:t-1}$,*

$$\|x_t(u_{1:t-1})\|, \|y_t(u_{t-H:t-1})\| \leq \gamma^{-1}(\kappa U + W),$$
$$\|x_t(u_{1:t-1}) - y_t(u_{t-H:t-1})\| \leq \gamma^{-1}(\kappa U + W)(1-\gamma)^H.$$

Proof of Lemma D.1. From the evolution of states we have

$$\|x_t(u_{1:t-1})\| = \left\| \sum_{i=0}^{t-1} \left(\prod_{j=i+1}^{t-1} A_j \right) (B_i u_i + w_i) \right\| \leq \left\| (\kappa U + W) \sum_{i=0}^{t-1} (1-\gamma)^{t-i-1} \right\| \leq \gamma^{-1}(\kappa U + W).$$

Similarly,

$$\|y_t(u_{t-H:t-1})\| = \left\| \sum_{i=t-H}^{t-1} \left(\prod_{j=i+1}^{t-1} A_j \right) (B_i u_i + w_i) \right\| \leq \gamma^{-1}(\kappa U + W).$$

If $t \leq H$, then $x_t(u_{1:t-1}) = y_t(u_{t-H:t-1})$. Otherwise,

$$\begin{aligned}\|x_t(u_{1:t-1}) - y_t(u_{t-H:t-1})\| &= \left\| \sum_{i=0}^{t-H-1} \left(\prod_{j=i+1}^{t-1} A_j \right) (B_i u_i + w_i) \right\| \\ &\leq (\kappa U + W)(1-\gamma)^H \sum_{i=0}^{t-H-1} (1-\gamma)^i \\ &\leq \gamma^{-1}(\kappa U + W)(1-\gamma)^H. \end{aligned}$$ □

Next, we characterize the approximation error between f_t and l_t^*. This directly follows from the previous lemma and the Lipschitzness of l_t^*.

Lemma D.2. *For all $t \geq 1$, with any $u_{1:t}$,*

$$\|l_t^*\left(x_t(u_{1:t-1}), u_t\right) - f_t(u_{t-H:t})\| \leq \gamma^{-1} L^* (\kappa U + W)(1-\gamma)^H.$$

Finally, we characterize the Lipschitzness of the ideal loss function f_t.

Lemma D.3. *For all $t \geq 1$ and $h \in [0:H]$, with any $\tilde{u}_{t-h}$ and $u_{t-H:t}$,*

$$|f_t(u_{t-H:t}) - f_t(u_{t-H:t-h-1}, \tilde{u}_{t-h}, u_{t-h+1:t})| \leq \kappa L^* \|u_{t-h} - \tilde{u}_{t-h}\|.$$

Proof of Lemma D.3. If $h \neq 0$, we consider the difference in the ideal state.

$$\begin{aligned}&\|y_t(u_{t-H:t-1}) - y_t(u_{t-H:t-h-1}, \tilde{u}_{t-h}, u_{t-h+1:t-1})\| \\ =& \left\| \left(\prod_{j=t-h+1}^{t-1} A_j \right) B_{t-h} \left(u_{t-h} - \tilde{u}_{t-h}\right) \right\| \leq \kappa (1-\gamma)^{h-1} \|u_{t-h} - \tilde{u}_{t-h}\|.\end{aligned}$$

Applying the Lipschitzness of l_t^*,

$$|f_t(u_{t-H:t}) - f_t(u_{t-H:t-h-1}, \tilde{u}_{t-h}, u_{t-h+1:t})| \leq \kappa L^* (1-\gamma)^{h-1} \|u_{t-h} - \tilde{u}_{t-h}\|.$$

If $h = 0$, then directly from the Lipschitzness of l_t^*,

$$|f_t(u_{t-H:t}) - f_t(u_{t-H:t-h-1}, \tilde{u}_{t-h}, u_{t-h+1:t})| \leq L^* \|u_{t-h} - \tilde{u}_{t-h}\|.$$

Combining the above completes the proof. □

Lemma D.4. *For all t, let $\tilde{f}_t(u) = f_t(u, \dots, u)$. Then, for all $u, \tilde{u} \in \mathsf{B}^{d_u}(0, U)$,*

$$\left|\tilde{f}_t(u) - \tilde{f}_t(\tilde{u})\right| \leq 2\kappa\gamma^{-1} L^* \left\| u - \tilde{u} \right\|.$$

Proof of Lemma D.4. For conciseness, let $\tilde{y}_t(u) = y_t(u, \dots, u)$. Then,

$$\left\| \tilde{y}_t(u) - \tilde{y}_t(\tilde{u}) \right\| = \left\| \sum_{i=t-H}^{t-1} \left(\prod_{j=i+1}^{t-1} A_j \right) B_i \left(u - \tilde{u} \right) \right\| \leq \kappa \left\| u - \tilde{u} \right\| \sum_{i=t-H}^{t-1} (1-\gamma)^{t-i-1} \leq \kappa\gamma^{-1} \left\| u - \tilde{u} \right\|.$$

The result follows from the Lipschitzness of l^*_t. □

Now we are ready to prove Theorem 3.

Theorem 3. *Given any strongly adaptive OCOM algorithm satisfying Equation (2), for all $\mathcal{I} = [a:b] \subset [H+1:T]$, Algorithm 5 guarantees*

$$\sum_{t=a}^{b} l^*_t \left(x_t(u^A_{1:t-1}), u^A_t \right) - \min_{u^C_{1:T} \in \mathcal{C}_{\mathcal{I}}} \sum_{t=a}^{b} l^*_t \left(x_t(u^C_{1:t-1}), u^C_t \right) = \tilde{O}\left(\sqrt{|\mathcal{I}|} \right),$$

where $\tilde{O}(\cdot)$ subsumes problem constants and poly$(\log T)$.

Proof of Theorem 3. For all $u^C_{1:T} \in \mathcal{C}_{\mathcal{I}}$,

$$\begin{aligned}
&\sum_{t=a}^{b} l^*_t \left(x_t(u^A_{1:t-1}), u^A_t \right) - \sum_{t=a}^{b} l^*_t \left(x_t(u^C_{1:t-1}), u^C_t \right) \\
= \;& \sum_{t=a}^{b} \left[l^*_t \left(x_t(u^A_{1:t-1}), u^A_t \right) - f_t(u^A_{t-H:t}) \right] + \sum_{t=a}^{b} \left[f_t(u^A_{t-H:t}) - f_t(u^C_{t-H:t}) \right] + \sum_{t=a}^{b} \left[f_t(u^C_{t-H:t}) - l^*_t \left(x_t(u^C_{1:t-1}), u^C_t \right) \right] \\
\leq \;& 2\gamma^{-1} L^* (\kappa U + W)(1-\gamma)^H |\mathcal{I}| + \sum_{t=a}^{b} \left[f_t(u^A_{t-H:t}) - f_t(u^C_b, \dots, u^C_b) \right],
\end{aligned}$$

where the last inequality is due to Lemma D.2. From our choice of H, we have $(1-\gamma)^H |\mathcal{I}| \leq 1$. Therefore, the first term on the RHS is a constant that can be neglected in our result. The second term on the RHS is upper-bounded by the strongly adaptive regret on f_t. □

D.2 Non-comparative tracking error bound

In the following we consider the example from Section 4. The comparative regret guarantee from Theorem 3 translates to a non-comparative tracking error bound.

Corollary 4. *Consider running Algorithm 5 on an adversarial tracking problem that satisfies Example 1 on a time interval $\mathcal{I}$. For all $t \in \mathcal{I} = [a:b]$,*

$$\frac{1}{t-a+1} \sum_{i=a}^{t} \left\| x_i(u^A_{1:i-1}) - x^*_{\mathcal{I}} \right\| \leq \gamma^{-1} W + \tilde{O}\left((t-a+1)^{-1/2} \right).$$

Proof of Corollary 4. We start by characterizing the power of the comparator class (Definition 4.1). From Example 1, there exists some u^* such that $x^*_{\mathcal{I}} = B_{\mathcal{I}} u^*$. Consider the comparator sequence $u^C_{1:T} \in \mathcal{C}_{\mathcal{I}}$ such that $u^C_t = u^*$ for all $t \in [1:T]$. From the system equation, for all $t \in [a:b]$,

$$\begin{aligned}
x_{t+1}(u^C_{1:t}) &= A_t x_t(u^C_{1:t-1}) + B_t u^* + w_t \\
&= A_t \left[x_t(u^C_{1:t-1}) - (I - A_t)^{-1} B_t u^* \right] + (I - A_t)^{-1} B_t u^* + w_t \\
&= A_t \left[x_t(u^C_{1:t-1}) - x^*_{\mathcal{I}} \right] + x^*_{\mathcal{I}} + w_t.
\end{aligned}$$

Rearranging the terms and applying norms on both sides,

$$\left\|x_{t+1}(u^C_{1:t}) - x^*_{\mathcal{I}}\right\| \leq (1-\gamma)\left\|x_t(u^C_{1:t-1}) - x^*_{\mathcal{I}}\right\| + W.$$

From Lemma D.1,

$$\left\|x_a(u^C_{1:a-1}) - x^*_{\mathcal{I}}\right\| \leq \left\|x_a(u^C_{1:a-1})\right\| + \left\|x^*_{\mathcal{I}}\right\| \leq \gamma^{-1}(\kappa U + W) + \kappa U\left\|(I - A_t)^{-1}\right\| \leq \gamma^{-1}(2\kappa U + W).$$

Following a recursion, for all $t \in [a:b]$,

$$\begin{aligned}\left\|x_t(u^C_{1:t-1}) - x^*_{\mathcal{I}}\right\| &\leq \gamma^{-1}(2\kappa U + W)(1-\gamma)^{t-a} + W\sum_{i=0}^{t-a}(1-\gamma)^i \\ &\leq \gamma^{-1}W + \gamma^{-1}(2\kappa U + W)(1-\gamma)^{t-a}.\end{aligned}$$

$$\begin{aligned}\frac{1}{t-a+1}\sum_{i=a}^{t}\left\|x_i(u^C_{1:i-1}) - x^*_{\mathcal{I}}\right\| &\leq \gamma^{-1}W + \gamma^{-1}(2\kappa U + W)\cdot\frac{1}{t-a+1}\sum_{i=0}^{t-a}(1-\gamma)^i \\ &\leq \gamma^{-1}W + \gamma^{-2}(2\kappa U + W)(t-a+1)^{-1}. \qquad (8)\end{aligned}$$

Next, consider the regret of Algorithm 5. Applying Theorem 3 on all time intervals $[a:t]$ with $t \in [a:b]$, we have

$$\sum_{i=a}^{t}\left\|x_i(u^A_{1:i-1}) - x^*_{\mathcal{I}}\right\| - \sum_{i=a}^{t}\left\|x_i(u^C_{1:i-1}) - x^*_{\mathcal{I}}\right\| = \tilde{O}\left(\sqrt{t-a+1}\right),$$

where $\tilde{O}(\cdot)$ subsumes polynomial factors on problem constants and poly-logarithmic factors on T. Normalizing on both sides,

$$\frac{1}{t-a+1}\left(\sum_{i=a}^{t}\left\|x_i(u^A_{1:i-1}) - x^*_{\mathcal{I}}\right\| - \sum_{i=a}^{t}\left\|x_i(u^C_{1:i-1}) - x^*_{\mathcal{I}}\right\|\right) = \tilde{O}\left((t-a+1)^{-1/2}\right). \qquad (9)$$

Combining (8) and (9) completes the proof. □

E EXPERIMENTS

In this section we test the proposed approach on three separate levels: (i) One-dimensional movement-aware OLO (Algorithm 1); (ii) Strongly adaptive OCOM (Algorithm 7); (iii) Adversarial tracking control (Algorithm 5).

E.1 One-dimensional movement-aware OLO

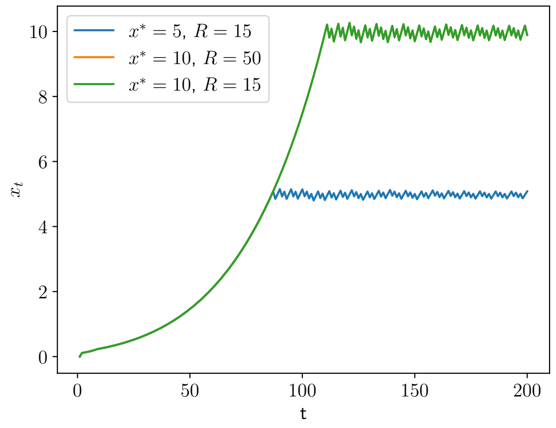
First, we test our one-dimensional movement-aware OLO algorithm (Algorithm 1). The domain is the interval $[0, R] \subset \mathbb{R}$, and the loss functions are defined as $l_t(x) = |x - x^*|$, where x^* is a fixed "target". Throughout this experiment, we set hyperparameters $\gamma = 0$, $\varepsilon = 1$ and $G = 1$ since the loss functions are 1-Lipschitz.

In Figure 3a we vary (i) the target x^*; and (ii) the size of the domain R.

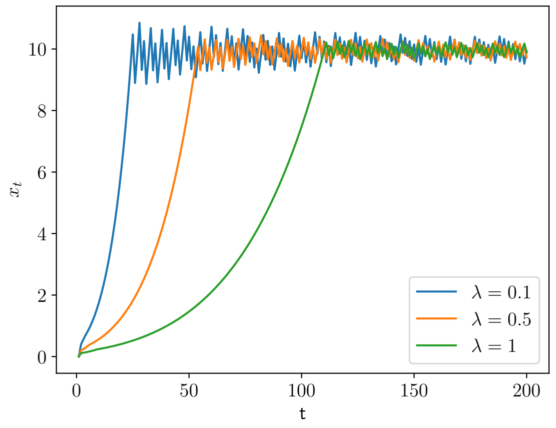
1. Consider the green line and the orange line (which is completely covered by the former). In this case, increasing the size of the domain leaves the performance of the algorithm unchanged. This is different from standard Online Gradient Descent (OGD) where the correct learning rate depends on the size of the domain.
2. Consider the blue line and the green line. Starting from the origin, the predictions of the algorithm approach the target with *exponentially increasing speed*, without knowing the target in advance. This is also different from OGD, where the speed of approaching the target is constant (with constant learning rate) or decreasing (with time-varying learning rate).

In general, Algorithm 1 exhibits the advantage of *parameter-free* online learning algorithms: the algorithm works well without depending on the optimal comparator norm ($\|x^*\|$) or its (possibly very loose) upper bound R, hence requiring less tuning than OGD.

In Figure 3b we vary λ, the weight of movement costs. Practically, it yields another "degree of freedom" (beside ε) for tuning the algorithm's transient response. Larger λ means larger weight on movement costs: the algorithm moves slower initially, but has less fluctuation around the target.



(a) Varying x^* and R.



(b) Varying λ.

Figure 3: Experiments on Algorithm 1.

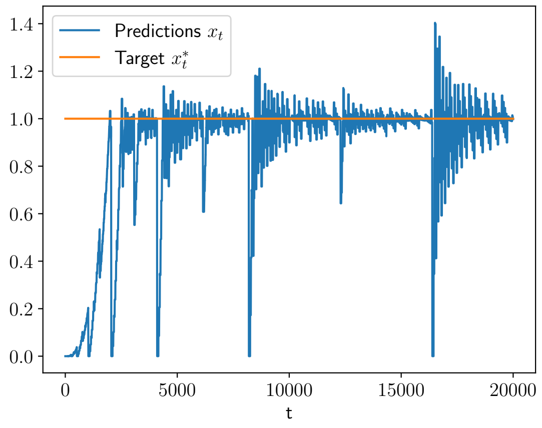
E.2 Strongly adaptive OCOM

Next, we test our strongly adaptive OCOM algorithm (Algorithm 7). For easier visualization, we set the domain as $\mathcal{V} = [-5, 5] \subset \mathbb{R}$. Let the memory constant $H = 5$. With a time-varying target x_t^*, we define the loss functions as

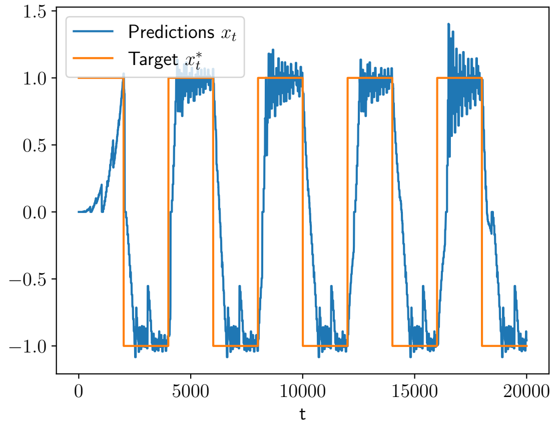
$$l_t(x_{t-H}, \ldots, x_t) = \sum_{h=0}^{H} \|x_{t-h} - x_t^*\| .$$

Note that the Lipschitz constants can be chosen as $L = 1$ and $\tilde{G} = H + 1$.

Our theoretical result requires $\varepsilon_0 = O(T^{-1})$. Although asymptotically this is correct, in practice such a small ε_0 makes the algorithm too conservative at the beginning. In other words, it can take a long time for the algorithm to *warm up*. Therefore, we set $\varepsilon_0 = 1$ in our experiments.



(a) x_t^* is a step signal.



(b) x_t^* is a square wave.

Figure 4: Experiments on Algorithm 7.

We plot the result of the experiments in Figure 4. On the left, the target is a step signal, $x_t^* = 1$. On the right, the target x_t^* is a square wave with period 4000. Several observations can be made:

1. In the warm-up phase of the algorithm (the first 2000 rounds), similar to the previous subsection, Algorithm 7 approaches the fixed target with increasing speed (if the "dips" are ignored).

2. Once the predictions reach the vicinity of the fixed target, they do not monotonically converge to the target as in standard online learning algorithms. Instead, the predictions fluctuate around the target, in a pattern determined by GC intervals. The rationale of this behavior is that, Algorithm 7 does not *know* or *assume* the target is fixed; to quickly adapt to possible sudden changes of the target, the algorithm regularly forgets the past and re-explores.
3. There is a practical issue not captured in our analysis. Every time a GC interval of a new length becomes active ($t = 2^n$ for some $n \in \mathbb{N}$), all the subroutines are reinitialized. Consequently, Algorithm 7 completely forgets all the received information and restarts from the origin (since the first output of the subroutine is always at the origin). This can cause large "dips" in the prediction sequence (e.g, $t \approx 8000$ and $t \approx 16000$ in Figure 4a). Such a behavior is undesirable if smooth predictions are preferred, but when the target regularly moves around the origin (Figure 4b) this can be acceptable.

E.3 A shifted version of our OCOM algorithm

To make the prediction sequence smoother, we also test a modified version of our strongly adaptive OCOM algorithm. The idea is simple: we incorporate a shifting procedure in the subroutines. Whenever a Subroutine-ball is reinitialized, its first prediction is set as the last prediction of the previous subroutine (before re-initialization). In this way, the meta-algorithm experiences less fluctuation due to the activation and deactivation of GC intervals.

Algorithm 8 A shifted version of Algorithm 2.

```
Require: Hyperparameters $(\lambda, \varepsilon, G)$ with $\lambda \geq 0$ and $\varepsilon, G > 0$; $g_1, g_2, \ldots \in \mathbb{R}^d$ with $\|g_t\| \leq G$, $\forall t$; a shift vector
    $v \in \mathbb{R}^d$.
1: Define $\mathcal{A}_r$ as Algorithm 1 on the domain $[0, R + \|v\|]$, with hyperparameters $(\lambda, \lambda, \varepsilon, G)$.
2: Define $\mathcal{A}_B$ as *Online Gradient Descent* (OGD) on $\mathsf{B}^d(0,1)$ with learning rate $\eta_t = 1/(G\sqrt{t})$, initialized at 0.
3: for $t = 1, 2, \ldots$ do
4:     Obtain $y_t \in \mathbb{R}$ from $\mathcal{A}_r$ and $z_t \in \mathbb{R}^d$ from $\mathcal{A}_B$. Predict $x_t = v + y_t z_t \in \mathbb{R}^d$, observe $g_t \in \mathbb{R}^d$.
5:     Return $\langle g_t, z_t \rangle$ and $g_t$ as the $t$-th loss subgradient to $\mathcal{A}_r$ and $\mathcal{A}_B$, respectively.
6: end for
```

Concretely, we first present a shifted version of Algorithm 2 (high dimensional movement-aware OLO) as Algorithm 8. Given a shift vector $v \in \mathbb{R}^d$, Algorithm 8 starts from predicting v, and all the predictions are within a larger norm ball $\mathsf{B}^d(v, R + \|v\|)$ centered at v. Using Algorithm 8 as the base algorithm of Subroutine-ball, we obtain a shifted version of the latter. When using this shifted Subroutine-ball in the meta-algorithm (Algorithm 7),

1. All the Subroutine-ball on GC intervals with indices $(k, 1)$ are initialized with shift vector $v = 0$. For example, at the beginning of the 2nd round, the meta-algorithm initializes A_B^1 with shift vector $v = 0$.
2. At the beginning of the $2^k i$-th round (with $i > 1$), when reinitializing $\mathcal{A}_B^k$, the shift vector v is set as the last prediction of the previous $\mathcal{A}_B^k$. For example, on the GC interval $[2:3]$ the meta-algorithm employs a Subroutine-ball $\mathcal{A}_B^1$. At the beginning of the 4th round, the meta-algorithm queries A_B^1 and assigns its prediction to a vector v. Then, A_B^1 is reinitialized with shift vector v.

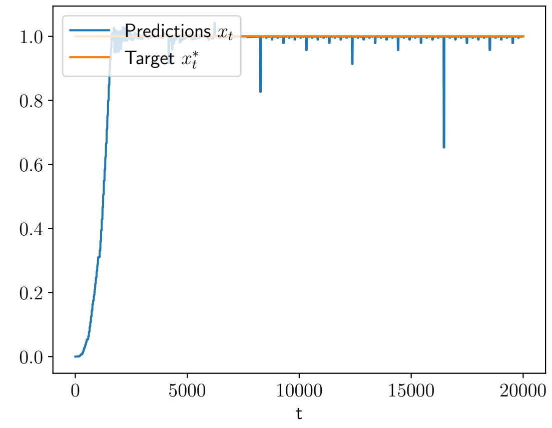
Empirical results for this *shifted OCOM algorithm* are presented in Figure 5. Specifically, the targets x_t^* in Figure 5a and 5b are the same as in Figure 4a and 4b. In Figure 5c, x_t^* is a sinusoidal wave with period 4000. In Figure 5d, x_t^* is the concatenation of a sinusoidal wave and a square wave:

$$x_t^* = \begin{cases} \sin(\pi t/2000), & \text{if } t < T/2, \\ 1, & \text{if } T/2 \leq t < 3T/4, \\ -1, & \text{otherwise.} \end{cases}$$

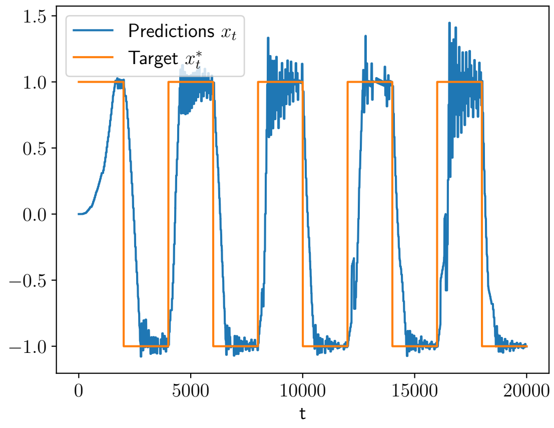
In general, the shifted version of Algorithm 7 tracks the target quite well, even when the target exhibits large, sudden changes. (The "tracking" here refers to the concept in online learning, not linear control.) Especially, the prediction sequence exhibits less fluctuation due to the reset of GC intervals.

E.4 Adversarial tracking

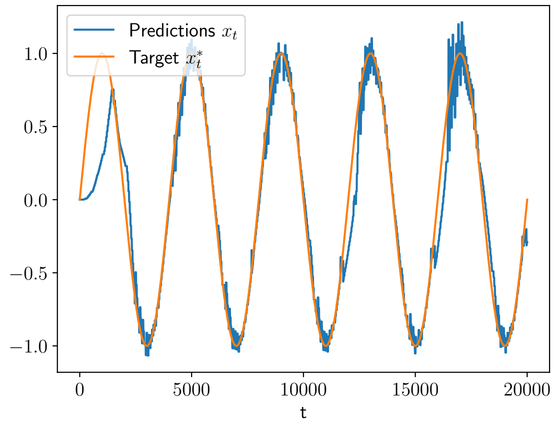
Finally, we test our adversarial tracking controller (Algorithm 5). We consider two cases: (i) $d_x = 1$; (ii) $d_x = 2$.



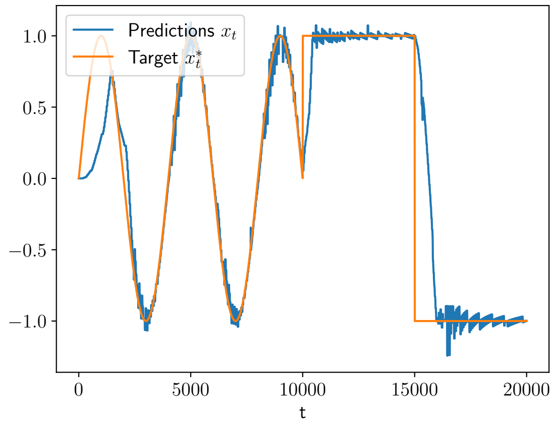
(a) x_t^* is a step signal.



(b) x_t^* is a square wave.



(c) x_t^* is a sinusoidal wave.



(d) x_t^* is a composite signal.

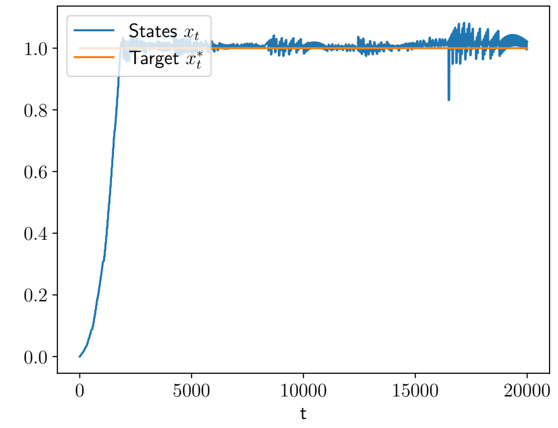
Figure 5: Experiments on the shifted OCOM algorithm.

One-dimensional control Starting from one-dimensional control, let $d_x = d_u = 1$, $U = 5$. The dynamics are time-varying: for all t, $A_t = 0.55 + 0.05 \cdot \sin(\pi t/10000)$; $B_t = 0.95 + 0.05 \cdot \sin(\pi t/5000)$. Therefore, $\kappa = 1$ and $\gamma = 0.4$. Further, we define the disturbances as $w_t = 0.05 \cdot \sin(\pi t/4000)$, $\forall t \in \mathbb{N}$.

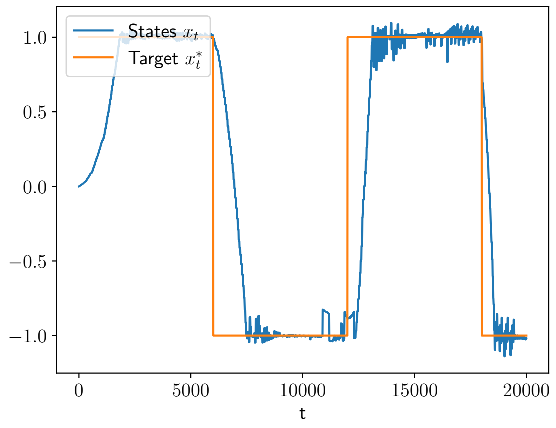
The loss functions are $l_t^*(x, u) = \|x - x_t^*\|$, where x_t^* is the adversarial reference trajectory. It is globally 1-Lipschitz, therefore $L^* = 1$.

We use the shifted version of Algorithm 7 as the base algorithm of our controller. Similar to the previous subsection, we set the hyperparameter as $\varepsilon_0 = 0.5$. Following the procedure in Algorithm 5, we set the problem constants in OCOM as: $\mathcal{V} \leftarrow \mathsf{B}^1(0, 5)$, $R \leftarrow 5$, $L \leftarrow 1$ and $\tilde{G} \leftarrow 5$. There is one exception: the memory H defined in Algorithm 5 is conservative. In our experiment, we treat H as a hyperparameter; specifically for the one-dimensional control experiment, we set $H = 8$. Intuitively, the choice of H trades off the responsiveness of the controller and its steady-state error.

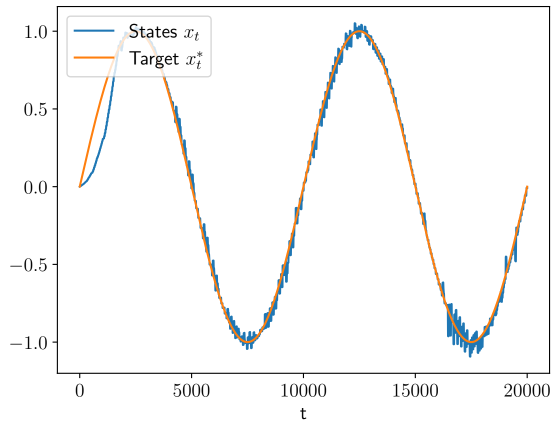
Empirical results for our controller are presented in Figure 6. Compared to the tracking results in online learning (Figure 5), we shoot for lower bandwidth since the dynamics introduce additional fluctuations. Figure 6a considers a fixed target $x_t^* = 1$. In Figure 6b, x_t^* is a square wave with period 12000. In Figure 6c, x_t^* is a sinusoidal wave with period 10000. Finally, we consider a composite target in Figure 6d:



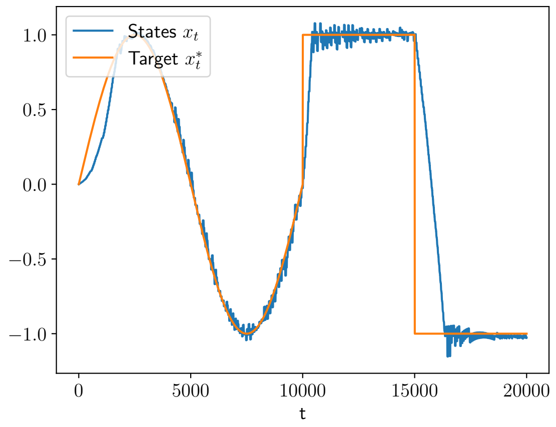
(a) x_t^* is a step signal.



(b) x_t^* is a square wave.



(c) x_t^* is a sinusoidal wave.



(d) x_t^* is a composite signal.

Figure 6: Testing the controller (Algorithm 5) in $\mathbb{R}$.

$$x_t^* = \begin{cases} \sin(\pi t/5000), & \text{if } t < T/2, \\ 1, & \text{if } T/2 \le t < 3T/4, \\ -1, & \text{otherwise.} \end{cases}$$

Two-dimensional control We also test the controller in a two-dimensional state space. Here, $d_x = d_u = 2$,

$$A_t = \begin{bmatrix} 0.55 & 0.3 \\ 0 & 0.55 \end{bmatrix} + I_2 \cdot 0.05\cos(\pi t/10000),$$

$$B_t = I_2 \cdot [0.95 + 0.05\cos(\pi t/5000)]\,,$$

where I_2 is the two-dimensional identity matrix. Same as before, $U = 5$, $\kappa = 1$ and $\gamma = 0.4$.

The loss functions are still $l_t^*(x,u) = \|x - x_t^*\|$, therefore $L^* = 1$. For all $t \in \mathbb{N}$, the disturbances are

$$w_t = 0.05\sin(\pi t/4000) \cdot [1, -1]^\top.$$

Same as before, $H = 8$, and we choose $\varepsilon_0 = 0.2$. The task is to track a circular reference trajectory (in an adversarial manner):

$$x_t^* = \begin{cases} [t/4000, 0]^\top, & \text{if } t \le 4000, \\ [\cos(\pi(t-4000)/8000), \sin(\pi(t-4000)/8000)]^\top, & \text{if } 4000 < t \le 20000. \end{cases}$$

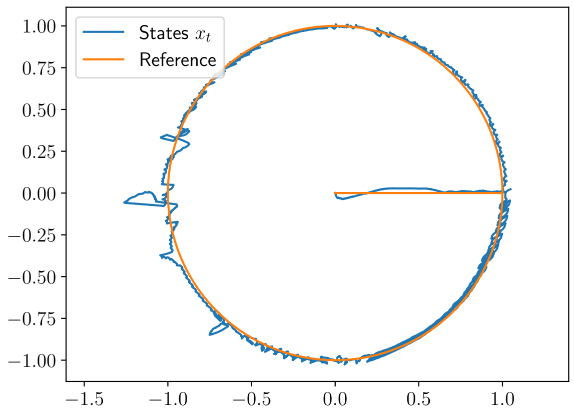


Figure 7: Testing the controller (Algorithm 5) in $\mathbb{R}^2$.

The result is shown in Figure 7. Both experiments show that the proposed controller tracks the adversarial reference trajectory quite well.