Self-Identifying Internal Model-Based Online Optimization [*]

Wouter J. A. van Weerelt [*] **Lantian Zhang** [*] **Silun Zhang** [*]
Nicola Bastianello [**]

[*] *Department of Mathematics, KTH Royal Institute of Technology, Stockholm, Sweden*
[**] *School of Electrical Engineering and Computer Science, and Digital Futures, KTH Royal Institute of Technology, Stockholm, Sweden*

Abstract: In this paper, we propose a novel online optimization algorithm built by combining ideas from control theory and system identification. The foundation of our algorithm is a control-based design that makes use of the internal model of the online problem. Since such prior knowledge of this internal model might not be available in practice, we incorporate an identification routine that learns this model on the fly. The algorithm is designed starting from quadratic online problems but can be applied to general problems. For quadratic cases, we characterize the asymptotic convergence to the optimal solution trajectory. We compare the proposed algorithm with existing approaches, and demonstrate how the identification routine ensures its adaptability to changes in the underlying internal model. Numerical results also indicate strong performance beyond the quadratic setting.

Keywords: online optimization, online learning, system identification, online gradient descent, control-based optimization

1. INTRODUCTION

The technological advances of recent years have increased the available sources of high resolution, streaming data in several applications, including the power grid, transportation networks, connected consumer devices (Simonetto et al., 2020; Dall'Anese et al., 2020). Leveraging these data at the relevant time-scales, *e.g.* for control (Liao-McPherson et al., 2018; Paternain et al., 2019; Chachuat et al., 2009), signal processing, (Natali et al., 2021; Hall and Willett, 2015; Fosson, 2021), machine learning (Shalev-Shwartz, 2011; Dixit et al., 2019; Rakhlin and Sridharan, 2013), therefore requires the solution of *online optimization* problems.

Online optimization problems are characterized by time-varying cost functions, which capture the evolving nature of the dynamic environments from which they arise (Simonetto et al., 2020). Formally, the problem of interest is

$$\boldsymbol{x}_k^* = \arg\min_{\boldsymbol{x} \in \mathbb{R}^n} f_k(\boldsymbol{x}), \quad k \in \mathbb{N}, \qquad (1)$$

where consecutive optimization instances arrive at intervals of length $T_s > 0$.

In addressing such problems, two categories of algorithms are typically considered: *unstructured* and *structured* (Si-

monetto et al., 2020). By unstructured we refer to algorithms which solve the optimization problem revealed at each time step k, similar to traditional batch algorithms; online gradient descent is a typical example. However, this approach does not tailor the design of the algorithms to the dynamic nature of the problem, thus in general we can only guarantee convergence to a neighborhood of the solution trajectory $\{\boldsymbol{x}_k^*\}_{k \in \mathbb{N}}$ (assuming uniqueness to simplify this discussion) (Dall'Anese et al., 2020; Simonetto et al., 2020).

Structured algorithms, on the other hand, are designed specifically for dynamic problems by incorporating a (possibly simplified) model of their time-variability. The benefit of designing structured algorithms is that they generally achieve significantly lower, and in some scenarios zero, asymptotic error in tracking the solution trajectory (Simonetto et al., 2020). Different approaches to structured algorithm design have been proposed, with the use of control theoretical tools recently achieving great success for unconstrained (Bastianello et al., 2024; Bianchin and van Scoy, 2025), constrained (Casti et al., 2025), stochastic (Casti and Zampieri, 2025; Simonetto and Massioni, 2024), and distributed (van Weerelt and Bastianello, 2025) problems.

These algorithms typically leverage the internal model principle, meaning they rely on some prior knowledge of the dynamic modes that generate the time-varying signals involved in the online problem. However, having such prior knowledge is oftentimes impractical (Bianchin and van Scoy, 2025), or may be inaccurate in practice (Bastianello et al., 2024).

Therefore, this paper seeks to remove the need for prior knowledge of the internal model by using the tools of system identification. System identification provides a control-theoretic framework for extracting dynamic information directly from observed data (Ljung, 1998), and has been successfully applied in various fields including control systems, signal processes, machine learning. Common methods include the least mean squares algorithm, Kalman filtering, and the recursive least squares (RLS) algorithm (Guo, 1994), the latter of which we leverage in this paper. In an online optimization context then, the knowledge gained through the application of these techniques can be leveraged to construct an internal model and, consequently, develop a novel structured algorithm.

In this paper we address this objective, offering the following contributions:

- We design a novel online algorithm which integrates the internal model-based design proposed in (Bastianello et al., 2024) with a system identification procedure (in particular, recursive least squares) that serves to construct the internal model from observations of (1).
- We analyze the convergence of the resulting algorithm applied to online quadratic problems.
- We test the proposed algorithm for both quadratic and non-quadratic problems, showcasing its improved performance over the state of the art and, importantly, adaptability to changes in the internal model.

2. PROBLEM FORMULATION

In order to design the algorithm proposed in section 3, we now focus our attention to a particular case of (1), characterized by the following quadratic function:

$$f_k(\boldsymbol{x}) := \frac{1}{2}\boldsymbol{x}^\intercal \boldsymbol{A}\boldsymbol{x} + \boldsymbol{x}^\intercal \boldsymbol{b}_k, \qquad (2)$$

where $\boldsymbol{x} \in \mathbb{R}^n$, the symmetric matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ is positive definite, and sequence $\{\boldsymbol{b}_k \in \mathbb{R}^n\}_{k \in \mathbb{N}}$ represents the time-varying signal driving the online problem.

Let $\underline{\lambda}$ and $\bar{\lambda}$ denote the minimal and maximal eigenvalues of matrix $\boldsymbol{A}$, respectively. Since $\boldsymbol{A}$ is positive definite, it follows that $\underline{\lambda}\boldsymbol{I} \preceq \boldsymbol{A} = \boldsymbol{A}^\intercal \preceq \bar{\lambda}\boldsymbol{I}$, and $\underline{\lambda} > 0$. Consequently, for any $k \in \mathbb{N}$, each cost function f_k is $\underline{\lambda}$-strongly convex and $\bar{\lambda}$-smooth. Moreover, it guarantees that, for any given sequence $\{\boldsymbol{b}_k\}$, each optimization problem in (1) admits a unique minimizer, thereby defining a unique optimal trajectory $\{\boldsymbol{x}_k^* = \boldsymbol{A}^{-1}\boldsymbol{b}_k\}_{k \in \mathbb{N}}$. We remark that, despite the problem having a closed-form solution, we aim to design an algorithm that only uses gradient information. The goal indeed is for the algorithm to be applicable to a broader range of online problems, not only quadratic ones.

We introduce now the following model for $\boldsymbol{b}_k$ to enable our control-based algorithm design.

Assumption 1. (Model of $\boldsymbol{b}_k$). We assume that the time series $\{\boldsymbol{b}_k\}_{k \in \mathbb{N}}$ admits a rational $\mathcal{Z}$-transform of the form

$$\mathcal{Z}[\boldsymbol{b}_k] = \boldsymbol{B}(z) = \frac{\boldsymbol{B}_N(z)}{B_D(z)}, \qquad (3)$$

where real polynomials $B_D(z) = z^m + \sum_{i=0}^{m-1} d_i z^i$, and $\boldsymbol{B}_N(z) = \sum_{i=0}^{p} \boldsymbol{u}_i z^i$ with $p \leq m$. Furthermore, we assume

that all roots of the denominator polynomial $B_D(z)$ have non-positive real parts.

Assumption 1 is standard in control-based design for online optimization problems, as it rules out unstable modes in $\boldsymbol{b}_k$, which would otherwise cause unbounded growth in the sequence of minimizers, *i.e.* $\|\boldsymbol{x}_k^* - \boldsymbol{x}_{k-1}^*\| \to \infty$ as $k \to \infty$. The difference with previous works such as (Bastianello et al., 2024; Casti et al., 2025), is that *we do not assume* $B_D(z)$ *(and* $\boldsymbol{B}_N(z)$*) to be known*, only that $\boldsymbol{b}_k$ admits the $\mathcal{Z}$-transform.

Finally, as discussed above we assume the algorithm has access to an *oracle of the gradient*, as well as the bounds on $\boldsymbol{A}$'s eigenvalues, $\underline{\lambda}$ and $\bar{\lambda}$. This assumption is typical in structured online optimization algorithms (see, *e.g.*, (Bastianello et al., 2024)).

In the setting discussed above, the goal then is to design a control-based algorithm which integrates online optimization with a system identification routine, removing the need of prior knowledge on the evolution of $\boldsymbol{b}_k$. Applying identification indeed allows to asymptotically reconstruct the internal model (specifically, $B_D(z)$), and as a consequence achieving perfect tracking of the optimal trajectory.

3. PROPOSED ALGORITHM

In this section we design the proposed algorithm, which we call *SIMBO*, for *Self-Identifying Internal Model-Based Online Optimizer*. As discussed above, the foundation of our algorithm design is the algorithm proposed in (Bastianello et al., 2024), which is characterized by the updates:

$$\boldsymbol{w}_{k+1} = (\boldsymbol{F} \otimes \boldsymbol{I})\boldsymbol{w}_k + (\boldsymbol{G} \otimes \boldsymbol{I})\nabla f_k(\boldsymbol{x}_k) \qquad (4a)$$
$$\boldsymbol{x}_{k+1} = (\boldsymbol{K} \otimes \boldsymbol{I})\boldsymbol{w}_{k+1} \qquad (4b)$$

where $\boldsymbol{w}$ is the internal state of the algorithm, and

$$\boldsymbol{F} = \begin{bmatrix} 0 & 1 & 0 & \cdots \\ & & \ddots & \\ 0 & \cdots & 0 & 1 \\ -d_0 & \cdots & \cdots & -d_{m-1} \end{bmatrix}, \quad \boldsymbol{G} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} \qquad (5)$$

$$\boldsymbol{K} = [c_0 \ c_1 \ \cdots \ \cdots \ c_{m-1}],$$

with $\boldsymbol{K}$ that could be computed according to (Bastianello et al., 2024) if $B_D(z)$ were known; but in the setting of this paper it is not.

Therefore, we propose to identify it, and using the output of the identification routine to characterize (4). However, at initialization ($k = 0$) we do not have any historical data on the problem (1) that could be used to identify the internal model beforehand. The idea then is to design a two-phase algorithm. In the first phase, we apply online gradient descent (OGD) (Dall'Anese et al., 2020) to the problem:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - h\nabla f_k(\boldsymbol{x}_k), \qquad (6)$$

where $h < 2/\bar{\lambda}$ is the step-size. Since OGD is unstructured, we can directly apply it to the problem without any model information. The output of OGD then can be used as data to feed the system identification routine, while also providing a (sub-optimal) sequence of decisions $\boldsymbol{x}_k$. Once enough data on the problem has been collected to

construct a first approximation of the internal model, we trigger the second phase. During the second phase we use the approximate internal model to construct $\boldsymbol{F}$ in (4), and to compute the controller $\boldsymbol{K}$. We can then switch to using the control-based algorithm (4) to compute the decisions $\boldsymbol{x}_k$. But we do not stop using system identification, since during the second phase we can refine the identified internal model or, more importantly, adapt to changes in the problem.

In section 3.1 we discuss the system identification routine that we use during the two phases, and in section 3.2 we lay out the overall proposed algorithm.

3.1 System identification routine

The system identification tool that we select to identify the internal model $B_D(z)$ is recursive least squares (RLS) (Guo, 1994). Letting $\boldsymbol{d} = [d_0, \ldots, d_{m-1}]^{\mathsf{T}} \in \mathbb{R}^m$ be the vector of coefficients of the polynomial $B_D(z)$, then RLS identifies it using the recursion:

$$\hat{\boldsymbol{d}}_{k+1} = \hat{\boldsymbol{d}}_k + \boldsymbol{L}_k(y_k - \boldsymbol{\phi}_k^{\mathsf{T}}\hat{\boldsymbol{d}}_k), \tag{7}$$

where y_k denotes the observation data, $\boldsymbol{\phi}_k$ denotes the regressor, and $\boldsymbol{L}_k$ denotes the gain (Guo, 1994):

$$\boldsymbol{L}_k = \boldsymbol{P}_k\boldsymbol{\phi}_k(\alpha\boldsymbol{I} + \boldsymbol{\phi}_k^{\top}\boldsymbol{P}_k\boldsymbol{\phi}_k)^{-1}, \tag{8}$$

with

$$\boldsymbol{P}_{k+1} = \frac{1}{\alpha}\left(\boldsymbol{P}_k - \boldsymbol{P}_k\boldsymbol{\phi}_k(\alpha\boldsymbol{I} + \boldsymbol{\phi}_k^{\top}\boldsymbol{P}_k\boldsymbol{\phi}_k)^{-1}\boldsymbol{\phi}_k^{\top}\boldsymbol{P}_k\right), \tag{9}$$

where $\boldsymbol{P}_0 > 0$, and $\alpha \in (0,1)$ is a forgetting factor which weights recent entries more heavily (Guo, 1994). The following paragraphs delineate the specific expressions that y_k and $\boldsymbol{\phi}_k$ take during the two phases of the algorithm when either (6) or (4) are applied.

Phase 1: initializing the identification During the first phase starting at $k = 0$ we apply the online gradient descent characterized by (6). Therefore, we need to extract information about $B_D(z)$ from OGD as follows. Taking the $\mathcal{Z}$-transform of the output of OGD, $\mathcal{Z}\{\boldsymbol{x}_{k+1}\} = \mathcal{Z}\{\boldsymbol{x}_k - h\nabla f_k(\boldsymbol{x}_k)\}$ and using (2), yields

$$z\boldsymbol{X}(z) = (\boldsymbol{I} - h\boldsymbol{A})\boldsymbol{X}(z) - h\boldsymbol{B}(z). \tag{10}$$

Rewriting the $\mathcal{Z}$-transforms as infinite sums (Graf, 2004), and recalling Assumption 1, we get

$$\sum_{k=0}^{\infty}\left(\boldsymbol{x}_{k+m} + \sum_{i=0}^{m-1} d_i\boldsymbol{x}_{k+i}\right)z^{-k} = $$
$$-h\sum_{k=0}^{\infty}(\boldsymbol{I} - h\boldsymbol{A})^k z^{-k-1}\sum_{j=0}^{p}\boldsymbol{u}_j z^j$$

which then gives

$$\boldsymbol{x}_{k+m} + \sum_{i=0}^{m-1} d_i\boldsymbol{x}_{k+i} = -h\sum_{j=0}^{p}(\boldsymbol{I} - h\boldsymbol{A})^{k-1+j}\boldsymbol{u}_j z^j. \tag{11}$$

Since we have selected a step-size $h < 2/\bar{\lambda}$ then we know that:

$$\lim_{k\to\infty}(\boldsymbol{I} - h\boldsymbol{A})^{k-1+j} = \boldsymbol{0}, \tag{12}$$

and thus the recurrence (11) becomes:

$$\boldsymbol{x}_{k+m} + \sum_{i=0}^{m-1} d_i\boldsymbol{x}_{k+i} = \boldsymbol{0}, \quad \forall k \geq m+1. \tag{13}$$

With this recurrence in place, we can then apply RLS with $y_k = \boldsymbol{\phi}_k^{\top}\boldsymbol{d}$ and

$$\boldsymbol{\phi}_k = [-\boldsymbol{x}_{k-1}, -\boldsymbol{x}_{k-2}, \cdots, -\boldsymbol{x}_{k-m}]^{\top}, \tag{14}$$

and $\boldsymbol{L}_k$ as introduced in (8).

Phase 2: continual identification Once phase 1 has successfully constructed an approximation of the internal model, characterized by $\hat{\boldsymbol{d}}_k$, we can use it to define $\boldsymbol{F}$ according to (5), and to compute the controller $\boldsymbol{K}$. We can then deploy the control-based algorithm (4) to replace OGD.

However, the first phase only approximates the internal model, and in addition the model might change over time. Therefore, during the second phase we continue running RLS to improve the identified model, and to do so, we need to extract information from the output of (4) as follows. Letting the inexact model computed at the end of phase 1 (iteration k_1) be

$$\hat{B}_D(z) = z^m + \sum_{i=0}^{m-1} \hat{d}_{k_1,i}z^i,$$

and using the controller $C(z) = C_N(z)/\hat{B}_D(z)$, algorithm (4) is represented by the $\mathcal{Z}$-transform:

$$\boldsymbol{X}(z) = \left(\hat{B}_D(z)\boldsymbol{I} - C_N(z)\boldsymbol{A}\right)^{-1}\frac{\boldsymbol{B}_N(z)C_N(z)}{B_D(z)}$$

and rearranging:

$$B_D(z)\boldsymbol{X}(z) = \left(\hat{B}_D(z)\boldsymbol{I} - C_N(z)\boldsymbol{A}\right)^{-1}\boldsymbol{B}_N(z)C_N(z). \tag{15}$$

Again using the properties of the $\mathcal{Z}$-transform we rewrite (15) as

$$\sum_{k=0}^{\infty}\left(\boldsymbol{x}_{k+m} + \sum_{i=0}^{m-1} d_i\boldsymbol{x}_{k+i}\right)z^{-k} = $$
$$\left(\hat{B}_D(z)\boldsymbol{I} - C_N(z)\boldsymbol{A}\right)^{-1}\boldsymbol{B}_N(z)\,C_N(z). \tag{16}$$

Finally, since the right-hand side is made up of two anticausal signals, and $C_N(z)$ is chosen in such a way that $\left(\hat{B}_D(z)\boldsymbol{I} - C_N(z)\boldsymbol{A}\right)$ is stable, the recurrence $\boldsymbol{x}_{k+m} + \sum_{i=0}^{m-1} d_i\boldsymbol{x}_{k+i} = \boldsymbol{0}$, $k \geq m+1$, holds (same as (13)). We can then apply the RLS (7) with $y_k = \boldsymbol{\phi}_k^{\top}\boldsymbol{d}_k$ and $\boldsymbol{\phi}_k = [-\boldsymbol{x}_{k-1}, -\boldsymbol{x}_{k-2}, \cdots, -\boldsymbol{x}_{k-m}]^{\top}$.

3.2 Algorithm overview

We are now ready to formalize the overall algorithm that we propose, SIMBO. Figure 1 represents the flowchart of SIMBO, highlighting the two phases, 1. identification initialization with online gradient descent, and 2. continual identification with the control-based algorithm (4). We now need to define the triggering condition that switches from phase 1 to 2, and the condition that triggers a recomputation of the controller in phase 2.

Given that RLS is applied to the recurrence (13), we can evaluate the system identification performance by the error $e_k := \left\|y_k - \boldsymbol{\phi}_k^{\mathsf{T}}\hat{\boldsymbol{d}}_k\right\|_1$. If the estimated coefficients $\hat{\boldsymbol{d}}_k$ coincide with the coefficients of the actual model this error is zero. Therefore, we can select a threshold $\theta > 0$ such
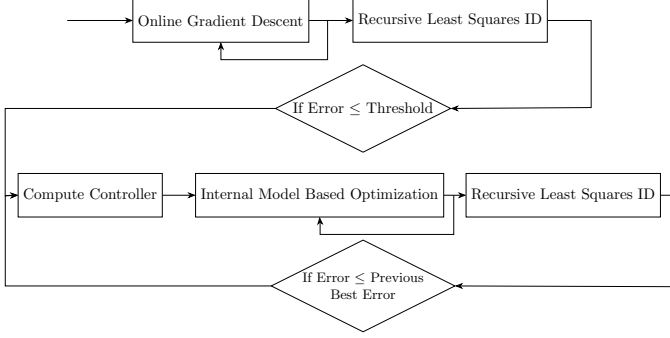
Fig. 1. Flowchart of the SIMBO algorithm

that, if $e_k \leq \theta$ then we know that phase 1 has constructed a good approximation of the internal model, and we can reliably use this model to compute the controller characterizing (4) and switch to phase 2. The trigger in phase 2, instead, is to ensure that we continuously improve the approximation of the internal model, especially in the case of changes of the actual model. This trigger then determines a recomputation of the controller when $e_k \leq e_{\bar{k}}$ where $e_{\bar{k}}$ was the previous best error. In other words, this condition is satisfied when the approximation of the model has improved.

Practical heuristics To complete the algorithm, we now discuss two heuristics that improve the performance in practice. First of all, in principle the triggering condition during phase 2 might be verified at each iteration k, thus requiring a large number of controller recomputations, which require the solution of two LMIs (Bastianello et al., 2024). This, however, would increase significantly the computational complexity and the risk of incurring in infeasible controller design problems. The idea then is to allow a recomputation of the controller only if the identification error e_k has not improved for a number of iterations; that is $e_k \leq e_{\bar{k}}$ and $k \geq \bar{k} + t$. Additionally, if the controller design problem happens to be infeasible we easily fall back on the previous controller.

The second heuristic we integrate in the algorithm is how to deal with changes in the actual internal model. The idea is to trigger *phase 1* again once the identification error has worsened significantly. In particular, we trigger phase 1 when

$$\left\| \boldsymbol{y}_k - \boldsymbol{\phi}_k^{\mathsf{T}} \boldsymbol{d}_{\bar{k}} \right\|_1 > C \left\| \boldsymbol{y}_{k-1} - \boldsymbol{\phi}_{k-1}^{\mathsf{T}} \boldsymbol{d}_{\bar{k}} \right\|_1,$$

with $C \gg 1$ and where $\boldsymbol{d}_{\bar{k}}$ is the identified model with the best error $\bar{k}$ up to time $k-1$ (in section 5 we use $C = 100$).

4. CONVERGENCE ANALYSIS

In this section we discuss the convergence of the proposed algorithm SIMBO. To this end, we assume that the actual internal model does not change (*i.e.* Assumption 1 holds). This means that phase 1 is executed once, and then we switch to phase 2 (no heuristics are applied).

Proposition 1. Let Assumption 1 hold, and assume that $\{b_k\}_{k\in\mathbb{N}}$ is persistently exciting of order m [1]. Then the output of SIMBO, $\{\boldsymbol{x}_k\}_{k\in\mathbb{N}}$ verifies

---

[1] The real-valued sequence $\{b_k\}_{k\in\mathbb{N}}$ is said to be persistently exciting (PE) of order m, if the Hankel matrix $H_m(b_{[0,h-1]})$ has full row rank for some integer $h \geq m$. Here $H_m(b_{[0,h-1]}) =$

$$\lim_{k\to\infty} \|\boldsymbol{x}_k - \boldsymbol{x}_k^*\| = 0.$$

Proof We start by analyzing the convergence during phase 1. The data used in the RLS comes from the online gradient descent (6), and we need to show that $\hat{\boldsymbol{d}}_k$ of (7) is indeed converging towards the internal model coefficients $\boldsymbol{d}$. First of all, we remark that OGD is converging to a neighborhood of the optimal trajectory, as proved in the following. By (Simonetto et al., 2020, Theorem 1) we have

$$\left\| \boldsymbol{x}_{k+1} - \boldsymbol{x}_{k+1}^* \right\| \leq \varrho \left\| \boldsymbol{x}_k - \boldsymbol{x}_k^* \right\| + \left\| \boldsymbol{x}_k^* - \boldsymbol{x}_{k+1}^* \right\| \quad (17)$$

where $\varrho := \max\{|1 - h\underline{\lambda}|, |1 - h\bar{\lambda}|\} \in (0, 1)$ for $h < 2/\bar{\lambda}$, and where

$$\left\| \boldsymbol{x}_k^* - \boldsymbol{x}_{k+1}^* \right\| \leq \left\| \boldsymbol{A}^{-1} \right\| \left\| \boldsymbol{b}_k - \boldsymbol{b}_{k+1} \right\| \leq \frac{1}{\underline{\lambda}} \left\| \boldsymbol{b}_k - \boldsymbol{b}_{k+1} \right\|$$

since $\boldsymbol{x}_k^* = -\boldsymbol{A}^{-1} \boldsymbol{b}_k$. Let $k_1 \in \mathbb{N}$ be the time when the algorithm switches to phase 2; then by (17) we have

$$\left\| \boldsymbol{x}_{k_1} - \boldsymbol{x}_{k_1}^* \right\| \leq \varrho^{k_1} \left\| x_0 - \boldsymbol{x}_0^* \right\|$$
$$+ \frac{1 - \varrho^{k_1+1}}{1 - \varrho} \max_{k \in [0, k_1]} \frac{1}{\underline{\lambda}} \left\| \boldsymbol{b}_k - \boldsymbol{b}_{k+1} \right\|$$

and OGD indeed converges to a neighborhood of the optimal trajectory during phase 1.

Now, by (6) and the fact that the cost is quadratic, the output of (6) can be written as

$$\boldsymbol{x}_{k+1} = (\boldsymbol{I} - h\boldsymbol{A})^{k+1} \boldsymbol{x}_0 - h \sum_{j=0}^{k} (\boldsymbol{I} - h\boldsymbol{A})^{k-j} \boldsymbol{b}_j, \quad (18)$$

which generates the regressors $\boldsymbol{\phi}_k$ according to (14). The RLS then is converging provided that, defining $\boldsymbol{\Phi}_k = [\boldsymbol{\phi}_{k+m} \cdots \boldsymbol{\phi}_{k+h-m}]$, the matrix $\boldsymbol{\Phi}_k \boldsymbol{\Phi}_k^{\mathsf{T}}$ is invertible for each $k \geq 0$. But this is a consequence of the fact that there is no noise in (18), that $\{b_k\}_{k\in\mathbb{N}}$ is persistently exciting of order m, and that $\boldsymbol{I} - h\boldsymbol{A}$ is full rank for any $h < 2/\bar{\lambda}$ (Haykin, 1991). In this scenario, we can characterize that the regressor sequence $\{x_k\}_{k\in\mathbb{N}}$ is persistently exciting of order m, which yields the following bound on the identification error (Guo, 1994; Haykin, 1991): $\left\| \hat{\boldsymbol{d}}_k - \boldsymbol{d} \right\| \leq M\zeta^k \left\| \hat{\boldsymbol{d}}_0 - \boldsymbol{d} \right\|$ for some $\zeta \in (0, 1)$ and $M > 0$. This implies that indeed, at the end of phase 1, RLS has approximately identified the internal model. Thus, there exists $k_1 \in \mathbb{N}$ large enough so that phase 2 is triggered.

We are now ready to analyze the convergence of phase two in (k_1, ∞). During this phase, we concurrently run RLS, which outputs $\hat{\boldsymbol{d}}_k$, and the control-based algorithm (4), whose matrices $\boldsymbol{F}_k$ and $\boldsymbol{K}_k$ are computed using $\hat{\boldsymbol{d}}_k$. By (Bastianello et al., 2024, Proposition 4) we know that (4) converges to a bounded neighborhood of the optimal trajectory when it is using an inexact internal model $\hat{\boldsymbol{d}}_k$, and that the neighborhood shrinks as $\hat{\boldsymbol{d}}_k \to \boldsymbol{d}$. To ensure this happens, we need then to guarantee that the RLS is converging to $\boldsymbol{d}$. This, similarly to the proof for phase 1, is a consequence of $\boldsymbol{b}_k$ being persistently exciting. This is because when $\hat{\boldsymbol{d}}_k$ closely matches $\boldsymbol{d}$, the regressor is determined by the the output of the internal model-based

---

$[\psi_0, \psi_1 \cdots, \psi_{h-m}]$, where $\psi_k = [b_k, \cdots, b_{k+m-1}]^{\top} (0 \leq k \leq h - m)$. (Jan C et al., 2005).

algorithm of (Bastianello et al., 2024). Since we know that the asymptotic tracking error of this algorithm is zero from (Bastianello et al., 2024, Proposition 1), the output of the algorithm is approximately $\boldsymbol{x}_{k+1} \approx \boldsymbol{A}^{-1}\boldsymbol{b}_k$. Hence the matrix $\boldsymbol{\Phi}$ is defined by $\boldsymbol{b}_k$ like in phase 1 as the matrix $\boldsymbol{A}$ is not time varying. The convergence then depends on the persistent excitation of $\boldsymbol{b}_k$. $\qquad\square$

5. NUMERICAL RESULTS

In this section we analyze the performance of SIMBO, and compare it to OGD (6) and the control-based algorithm (4) (Bastianello et al., 2024). The latter algorithm is designed using the exact internal model; as such, it serves as a baseline for SIMBO, but as discussed before the exact model would not be available in practice. All simulations were implemented using the `tvopt` Python package (Bastianello, 2021).

5.1 Quadratic problems

We start comparing the three algorithms for quadratic problems characterized by (2), with $n = 15$, and $\boldsymbol{A}$ is randomly generated so that $\underline{\lambda} = 1$, $\bar{\lambda} = 5$. For the linear term $\boldsymbol{b}_k$ we use the following four internal models:

(1) $\boldsymbol{b}_k = \sin(\omega_0 k T_s)\mathbf{1}$ (Sine function)
(2) $\boldsymbol{b}_k = k T_s \bar{\boldsymbol{b}}$ (Ramp function)
(3) $\boldsymbol{b}_k = \sin(\omega_0 k T_s)\mathbf{1} + k T_s \bar{\boldsymbol{b}}$ (Sine plus ramp function)
(4) $\boldsymbol{b}_k = \sin^2(\omega_1 k T_s)\mathbf{1}$ (Sine squared function)

where $T_s = 0.1$, $\omega_0 = 1$, $\omega_1 = 10$, and $\bar{\boldsymbol{b}} \in \mathbb{R}^n$ is randomly generated.

In Figure 2 we report the evolution of the tracking error $\{\|\boldsymbol{x}_k - \boldsymbol{x}_k^*\|\}_{k \in \mathbb{N}}$ for the three algorithms. As expected,
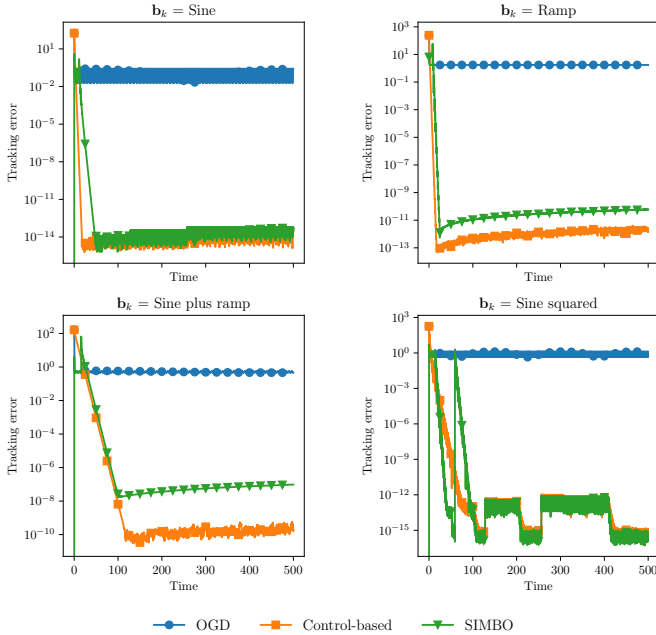

Fig. 2. Tracking error comparison with different internal models in (2)

OGD can only converge to a bounded neighborhood of the optimal trajectory, while the control-based algorithm

converges asymptotically to it (up to numerical precision). This is of course dependent on knowing the exact model, which in practice might not be available. Therefore, seeing that SIMBO performs very closely to the control-based algorithm guarantees that indeed incorporating system identification allows to reduce the prior knowledge requirements without sacrificing accuracy. In other words, the system identification routine successfully identifies the internal model. Inspecting Figure 2, some additional observations are in order. First of all, the cases where a ramp signal is involved, numerical precision is slightly worse since the ramp grows unbounded. Incorporating an integrator in the algorithm design might serve to reduce this numerical issue. The second observation is that switching from phase 1 to 2, or triggering the recomputation of the controller (bottom right plot at around $k = 50$), might give rise to a transient. Further changes to the design might serve to reduce the size of these transients.

We conclude this section by providing the asymptotic value of the tracking errors in Table 1. The asymptotic error is estimated in practice by taking the maximum error over the final 4/5 of the simulation.

Table 1. Asymptotic tracking errors

Algorithm	Ramp	Sine	Sine²	Sine + ramp
OGD	1.73e+00	2.70e−01	1.29e+00	5.97e−01
Control-Based	4.02e−12	5.51e−14	8.03e−13	5.65e−10
SIMBO	6.09e−11	6.40e−14	7.28e−13	1.90e−07

5.2 Adapting to a changing internal model

The use of SIMBO, as opposed to the control-based algorithms (4), is especially necessary when the internal model might change over time. In this section we test the algorithms for two quadratic problems whose linear term $\boldsymbol{b}_k$ changes internal model half-way through, (1) ramp then sine, (2) sine then squared sine (using the models of section 5.1). The results are depicted in Figure 3. We
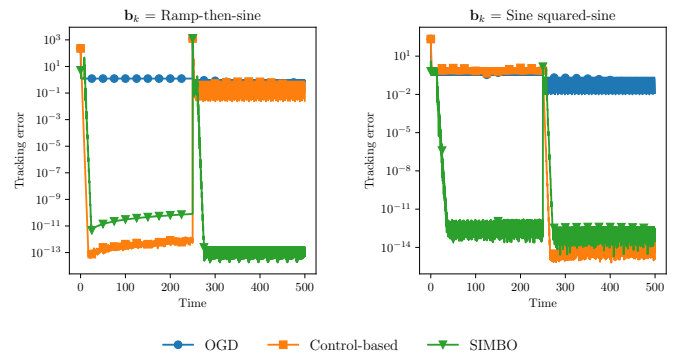

Fig. 3. Tracking error with changing internal models

can see that, as expected, SIMBO is able to adapt to changes in the internal model. Indeed, after the change, SIMBO re-identifies the model (by triggering phase 1 again, see section 3.2), recomputes the controller, and then successfully tracks the optimal trajectory. On the other hand, the control-based algorithm is fixed, based on the internal model used to compute the controller at the beginning. Thus, when the internal model coincides with

the model of $\boldsymbol{b}_k$ (first half in the left plot, second half in the left plot) the algorithm tracks the optimal trajectory. However, as soon as the model changes half-way through the simulation, its performance decays, in some cases worse than the unstructured OGD.

5.3 Time-varying Hessians

In this paper we have designed SIMBO for problems with quadratic cost (2), where only the linear term varies. However, as proved in (Bastianello et al., 2024), this control-theoretical approach can equally be applied to more general problems, and the same applies to SIMBO [2]. In this section then we test the algorithms on the following costs where also the Hessian changes over time: $f_k(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^\intercal \boldsymbol{A}_k \boldsymbol{x} + \boldsymbol{x}^\intercal \boldsymbol{b}_k$, where $\boldsymbol{A}_k = \boldsymbol{A} + \tilde{\boldsymbol{A}}_k$, $\boldsymbol{A} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^\intercal$ and $\tilde{\boldsymbol{A}}_k = \boldsymbol{V}\mathrm{diag}\{\sin(\omega_0 k t_s)\boldsymbol{v}\}\boldsymbol{V}^\intercal$. Similarly to section 5.1, we guarantee that $\boldsymbol{A}_k$ has eigenvalues in $[1,5]$ for all $k \in \mathbb{N}$. For the sake of simplicity, we also assume that $\boldsymbol{b}_k = \bar{\boldsymbol{b}} \in \mathbb{R}^n$.

Figure 4 reports the evolution of the tracking errors in this setting. As expected, OGD reaches a large neighbor-
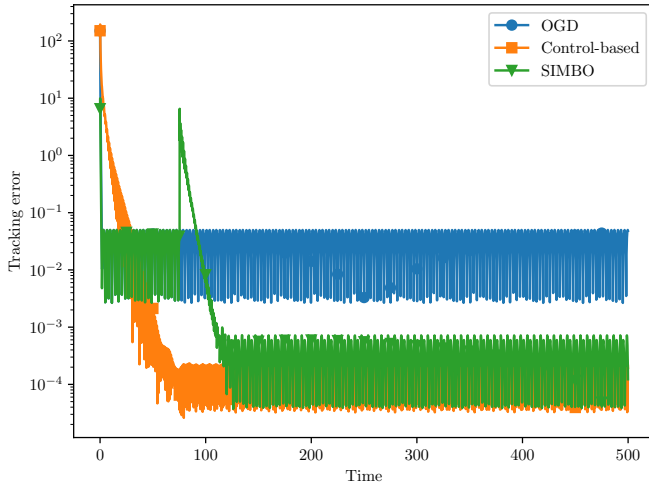
Fig. 4. Tracking error with time-varying Hessian

hood of the optimal trajectory, while the control-based algorithm (4) and SIMBO reach a tighter neighborhood (although not zero, as the assumptions of section 2 are not verified). Importantly, the control-based algorithm (4) is defined on an internal model that is hand-tuned to improve performance (which requires prior information), while SIMBO reaches similar performance automatically adapting the internal model.

6. CONCLUSIONS AND FUTURE WORK

In conclusion, this paper proposes a novel class of structured online algorithms that merge the control theoretical design of (Bastianello et al., 2024) with a system identification routine. The use of system identification allows to construct an internal model of the problem's time-variability, without having to resort to prior knowledge/data which

---

[2] Indeed, the algorithm only requires oracle evaluations of the gradient to be applied in practice.

in practice are rarely available. Additionally, using identification allows to adapt in real time to changes in the behavior (*i.e.* in the internal model) of the online problem. The performance of the proposed algorithm is evaluated theoretically and validated numerically. Future research will focus for example on extending this identification approach to more general problems (using the non-linear internal model principle).

REFERENCES

Bastianello, N. (2021). tvopt: A python framework for time-varying optimization. In *2021 60th IEEE Conference on Decision and Control (CDC)*, 227–232.

Bastianello, N., Carli, R., and Zampieri, S. (2024). Internal model-based online optimization. *IEEE Transactions on Automatic Control*, 69, 689–696.

Bianchin, G. and van Scoy, B. (2025). The internal model principle of time-varying optimization. arXiv:2407.08037.

Casti, U., Bastianello, N., Carli, R., and Zampieri, S. (2025). A control theoretical approach to online constrained optimization. *Automatica*, 176, 112107.

Casti, U. and Zampieri, S. (2025). Stochastic models for online optimization. In *2025 European Control Conference (ECC)*, 1880–1885.

Chachuat, B., Srinivasan, B., and Bonvin, D. (2009). Adaptation strategies for real-time optimization. *Computers & Chemical Engineering*, 33(10), 1557–1567. Selected Papers from the 18th European Symposium on Computer Aided Process Engineering (ESCAPE-18).

Dall'Anese, E., Simonetto, A., Becker, S., and Madden, L. (2020). Optimization and learning with information streams: Time-varying algorithms and applications. *IEEE Signal Processing Magazine*, 37, 71–83.

Dixit, R., Bedi, A.S., Tripathi, R., and Rajawat, K. (2019). Online learning with inexact proximal online gradient descent algorithms. *IEEE Transactions on Signal Processing*, 67, 1338–1352.

Fosson, S.M. (2021). Centralized and distributed online learning for sparse time-varying optimization. *IEEE Transactions on Automatic Control*, 66(6), 2542–2557.

Graf, U. (2004). *z-Transformation*, 77–113. Birkhäuser Basel, Basel.

Guo, L. (1994). Stability of recursive stochastic tracking algorithms. *SIAM Journal on Control and Optimization*, 32, 1195–1225.

Hall, E.C. and Willett, R.M. (2015). Online convex optimization in dynamic environments. *IEEE Journal of Selected Topics in Signal Processing*, 9(4), 647–662.

Haykin, S.S. (1991). *Adaptive filter theory*. Prentice-Hall information and system sciences series. Prentice Hall, Englewood Cliffs, N.J, 2. ed. edition.

Jan C, W., Paolo, R., Ivan, M., and Bart L.M., D.M. (2005). A note on persistency of excitation. *Systems & Control Letters*, 54(4), 325–329.

Liao-McPherson, D., Nicotra, M.M., and Kolmanovsky, I.V. (2018). A semismooth predictor corrector method for real-time constrained parametric optimization with applications in model predictive control. In *2018 IEEE Conference on Decision and Control (CDC)*, 3600–3607.

Ljung, L. (1998). *System Identification*, 163–173. Birkhäuser Boston, Boston, MA.

Natali, A., Coutino, M., Isufi, E., and Leus, G. (2021). Online time-varying topology identification via prediction-correction algorithms. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 5400–5404.

Paternain, S., Morari, M., and Ribeiro, A. (2019). Real-time model predictive control based on prediction-correction algorithms. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, 5285–5291.

Rakhlin, A. and Sridharan, K. (2013). Online learning with predictable sequences. In *Conference on Learning Theory*, 993–1019. PMLR.

Shalev-Shwartz, S. (2011). Online learning and online convex optimization.

Simonetto, A., Dall'Anese, E., Paternain, S., Leus, G., and Giannakis, G.B. (2020). Time-varying convex optimization: Time-structured algorithms and applications. *Proceedings of the IEEE*, 108, 2032–2048.

Simonetto, A. and Massioni, P. (2024). Nonlinear optimization filters for stochastic time-varying convex optimization. *International Journal of Robust and Nonlinear Control*, 34(12), 8065–8089.

van Weerelt, W.J.A. and Bastianello, N. (2025). Control-based online distributed optimization. arXiv:2508.15498.