Internal Model-Based Online Optimization

Nicola Bastianello, Member, IEEE, Ruggero Carli, Member, IEEE, and Sandro Zampieri, Fellow, IEEE

Abstract—In this article, we propose a model-based approach to the design of online optimization algorithms, with the goal of improving the tracking of the solution trajectory (trajectories) w.r.t. state-of-the-art methods. We focus first on quadratic problems with a time-varying linear term, and use digital control tools (a robust internal model principle) to propose a novel online algorithm that can achieve zero tracking error by modeling the cost with a dynamical system. We prove the convergence of the algorithm for both strongly convex and convex problems. We further discuss the sensitivity of the proposed method to model uncertainties and quantify its performance. We discuss how the proposed algorithm can be applied to general (nonquadratic) problems using an approximate model of the cost, and analyze the convergence leveraging the small gain theorem. We present numerical results that showcase the superior performance of the proposed algorithms over previous methods for both quadratic and nonquadratic problems.

Index Terms—Digital control, online gradient descent, online optimization, robust control, structured algorithms.

I. INTRODUCTION

In applications ranging from control [1], [2] to signal processing [3], [4], [5], to machine learning [6], [7], [8], recent technological advances have made the class of online optimization problems of central importance. These problems are characterized by cost functions that vary over time, capturing the complexity of dynamic environments and changing optimization goals [9], [10]. Online problems require the design of solvers that can be applied in real time, with the objective of tracking the time-varying solution(s) within some precision. Formally, we are interested in solving the following:

$$\mathbf{x}_k^* = \arg \min_{\mathbf{x} \in \mathbb{R}^n} f_k(\mathbf{x})$$  \hspace{1cm} (1)

where consecutive problems arrive $T_e > 0$ s apart. A first approach to developing online algorithms is to repurpose methods from static optimization (say, gradient descent) by applying them to each problem in the sequence. The convergence properties of the resulting unstructured [10] online algorithms have been studied for different classes of problems, e.g., [3], [4], [6], [7]. In general, it is possible to prove the convergence of these algorithms’ output to a neighborhood of the dynamic solution trajectory (assuming, for simplicity, its uniqueness), and to bound its radius as a function of the problem properties [9], [10].

However, unstructured algorithms are passive with respect to the dynamic nature of online problems, since they treat each problem in the sequence as self-standing. To remedy this fact, a broad part of the literature has been devoted to the design and analysis of structured algorithms [10], which aim to leverage knowledge of the problem in order to improve the performance by reducing the tracking error. Usually, this knowledge comes in the form of a set of assumptions that bound the rate of change of the problem, say by bounding the difference between consecutive cost functions. A particular class of structured algorithms is that of prediction-correction methods, in which we build an approximation of a future problem from past observations—the prediction—and use it to warm-start the solution of the next problem [11], [12], [13], [14]. Structured methods have been shown to achieve smaller tracking error than their unstructured counterparts, highlighting their benefits. In this article, we are interested in carrying this structured approach to online optimization one step further, by exploiting control-theoretical tools to design novel online algorithms that can track the optimal trajectory with greater precision than unstructured methods.

Before we detail our proposed contribution, we review some works at the fruitful intersection of control theory and (online) optimization. An important class of online problems is that analyzed in feedback optimization, in which the goal is to drive the output of a system to the optimal trajectory of an online optimization problem [15], [16], [17]. This setup, which includes model predictive control applications [1], is characterized by the interconnection of an online algorithm and a system, and control theory has been leveraged to analyze the stability of this closed loop. While in feedback optimization, we employ optimization techniques to solve a control problem, using control theory has also proved useful in the design and analysis of optimization algorithms, see, e.g., [18], [19], [20], [21], [22]. Finally, we mention [23] and [24] in which online algorithms are developed under the assumption that the optimal trajectory evolves according to a linear dynamical system.

Before presenting the proposed contribution, we discuss two motivating examples.

Example 1 (Online learning): Consider an online learning problem in which we need to train a learner based on time-varying data [6]. We are then interested in algorithms that can exploit a (possibly rudimentary) model of the cost variation to improve accuracy of the learned model in the long run.

Example 2 (Signal processing over sensors network): Consider a signal processing problem in which the data are provided by measurements taken by a network of sensors. In this context, one can envision two sources of time-variability, one stemming from the dynamic nature of the process being monitored, and the other by the possible intermittent failure of the sensors to deliver a new measurement.
As discussed so far, the focus of this article is designing structured online algorithms that rely on a model of the dynamic cost, as derived, e.g., from first principles or historical data, and apply control theoretical tools in order to achieve better performance in the long run. Assuming that the cost is smooth, our approach is to recast the online optimization problem as the goal of driving the gradient of the time-varying cost (the plant) to zero. In turn, this implies perfect tracking of the optimal trajectory, an achievement which unstructured methods cannot reach in general. We first explore this idea as applied to quadratic problems with a time-varying linear term, which arise for example in signal processing [7] and machine learning [6]. In the context of quadratic problems, we leverage tools from digital control to develop a novel online algorithm, which, making use of a model of the cost as a dynamical system, can achieve zero tracking error. In particular, we employ the internal model principle in combination with robust control theory to design the algorithm. We prove the convergence of the proposed method both for strongly convex (to the unique optimal trajectory) and convex problems (to the time-varying set of solutions). The proposed algorithm, however, relies on the knowledge of the cost model, which in practice may be inaccurate. Therefore, we analyze its sensitivity to inexact models, by characterizing a bound for the tracking error as a function of the model uncertainty. After laying the groundwork by focusing on a specific class of quadratic problems, we then discuss the application of the proposed algorithms to general quadratic (with a time-varying Hessian) and nonquadratic problems as well. We show how the proposed methods can be applied to any online (smooth) optimization problem with the use of an approximate cost model, and analyze their convergence to a neighborhood of the optimal trajectory by leveraging the small gain theorem. Finally, we showcase the better performance of the algorithms as compared with state-of-the-art alternatives, both for quadratic and non-problems. The contribution is summarized in the following points.

1) We propose a control-based online algorithm that leverages a model of the cost to improve the performance over unstructured methods. In particular, we show that for quadratic problems with a time-varying linear term, the algorithm tracks with zero error the optimal trajectory (for strongly convex problems) or the optimal set (for convex problems).

2) We discuss the sensitivity of the proposed method to model uncertainties, and provide an upper bound to the tracking error achieved when an exact model is used.

3) We show how the proposed structured algorithm can be applied to general (nonquadratic) online problems by choosing a suitable approximate model. Interpreting the inexact model as a source of disturbance, we then provide a convergence analysis of the resulting algorithms.

4) We present numerical results that showcase the superior performance of the proposed algorithms with respect to state-of-the-art methods, both for quadratic and nonquadratic problems.

**II. CONTROL-BASED ONLINE OPTIMIZATION**

We focus first on the class of following class of online quadratic problems:

\[ f_k(x) := \frac{1}{2} x^\top A x + x^\top b_k \quad (2) \]

in which we assume that the symmetric matrix \( A \) is fixed and that only \( b_k \) is time-varying. Restricting our attention to these problems will allow us to design the algorithm proposed in this section. In Section III, its convergence will be first proved in the context described in (2), and then extended to more general (not necessarily quadratic) problems.

In the following we assume that Assumptions 1 and 2 hold.

**Assumption 1 (Strongly convex and smooth):** The symmetric matrix \( A \) is such that \( \lambda \leq A \leq \lambda I \), with \( 0 < \lambda \leq \lambda < \infty \). This is equivalent to imposing that the cost functions \( \{f_k\}_{k \in \mathbb{N}} \) are \( \lambda \)-strongly convex and \( \lambda \)-smooth for any time \( k \in \mathbb{N} \).

**Assumption 2 (Model of \( b_k \):** The coefficients of the linear term \( \{b_k\}_{k \in \mathbb{N}} \) have a rational \( \mathcal{Z} \)-transform, namely

\[ Z[b_k] = B(z) = \frac{B_N(z)}{B_D(z)} \]

for some \( B_N(z) \in \mathbb{R}^n[z] \), where the poles of \( B_D(z) \) are all marginally or asymptotically stable.

Assumption 1 is widely used in online optimization [9], since the strong convexity of the cost implies that each problem in (1) has a unique minimizer, and we can define the optimal trajectory \( \{x^*_k\}_{k \in \mathbb{N}} \). Assumption 2 instead provides the model of the cost function used for the design of the proposed algorithm. We rule out unstable modes in \( b_k \), since they cause unbounded growth of \( ||x_k^* - x_{k-1}^*|| \), which is the distance between subsequent minimizers. Observe that the assumption that \( ||x_k^* - x_{k-1}^*|| \) is bounded for all \( k \in \mathbb{N} \) is standard in online optimization [9]. Nevertheless the proposed model is general enough to capture any kind of vector-valued signal \( b_k \) characterized by a finite number of frequencies. We remark moreover that only knowledge of the denominator \( B_D(z) \) is required, whereas the numerator does not play any role in the algorithm.

**Remark 1 (Gradient oracle):** In the following, we assume that only an oracle of the gradient can be accessed by the online algorithm, i.e., we only know the gradient \( \nabla f_k(x_k) \), and we adopt the closed-form solution of the optimization when (2) holds, i.e.,

\[ x_k^* = \arg\min_{x \in \mathbb{R}^n} \{ f_k(x) \} \]

\[ x_k^* = -A^{-1}b_k \]

This assumption is in accordance with frameworks adopted by all gradient methods that our technique aims to improve.

**Remark 2 (State of the art):** A widely used algorithm that can be applied to these time-varying problems is the online gradient method characterized by \( x_{k+1} = x_k - \alpha \nabla f_k(x_k) \). Under Assumption 1 and provided that \( ||x_k^* - x_{k-1}^*|| \) is bounded for any \( k \), then it can be proved that the online gradient satisfies \( \lim \sup_{k \to \infty} ||x_k - x_k^*|| \leq R < \infty \) when \( \alpha \in (0, 2/\lambda) \) [9]. The proposed control-based approach will allow us to design an algorithm that can instead guarantee \( \lim \sup_{k \to \infty} ||x_k - x_k^*|| = 0 \) for (2).
A. Control Scheme

In order to drive the gradient signal \( e_k := \nabla f_k(x_k) \) to zero asymptotically, we setup the control scheme depicted in Fig. 1. In the block diagram, we see that the gradient \( e_k \) is connected in feedback with a controller \( C(z) \)—to be designed—which produces the output \( x_k \) that is the estimate of the minimizer \( x_k^\ast \) provided by the algorithm.

Our goal then is to design \( C(z) \) so that \( e_k \) converges to zero. We can do this by leveraging the internal model principle [25], namely by choosing

\[
C(z) = \frac{C_N(z)}{B_D(z)}, \quad \text{with} \quad C_N(z) = \sum_{i=0}^{m-1} c_i z^i
\]

(4)

where we have to find a polynomial \( C_N(z) \) of degree less than \( m \) to guarantee that \( C(z) \) is strictly proper such that the closed-loop system is asymptotically stable.

From Fig. 1, we argue that \( X(z) = C(z)E(z) \) and \( E(z) = AX(z) + B(z) \), and combining them with (4) we get

\[
E(z) = (B_D(z)I - C_N(z)A)^{-1} B_N(z).
\]

(5)

Therefore, the poles of \( E(z) \), when using the controller structure (4), come from the term \( (B_D(z)I - C_N(z)A)^{-1} \). Hence, to achieve convergence to zero, we need to choose \( C_N(z) \) such that these poles are stable. In the next section, we will show how the design of \( C_N(z) \) can be translated into a robust control problem.

B. Designing the Robust Stabilizing Controller

Observe first that the problem can be simplified. Consider the eigendecomposition of \( A = V \Lambda V^\top \), with \( V^\top V = I \) and \( \Lambda = \text{diag} \{ \lambda_i \}_{i=1}^m, \lambda_i \in [\hat{\lambda}, \bar{\lambda}] \). Using it in (5) yields

\[
E(z) = V (B_D(z)I - C_N(z)A)^{-1} V^\top B_N(z).
\]

(6)

We remark that, even though the eigendecomposition of \( A \) is used in this theoretical discussion, we do not need to know it to implement algorithm (7). By (6), we see that the poles of \( E(z) \) in the scheme of Fig. 1 only come from the inverse of the diagonal matrix \( B_D(z)I - C_N(z)A \). Namely, the poles of \( E(z) \) are the roots of the \( n \) polynomials \( \{ B_D(z) - C_N(z)\lambda_i \}_{i=1}^m, \lambda_i \in [\hat{\lambda}, \bar{\lambda}] \). Therefore, in order to ensure asymptotic stability, it is sufficient to choose a \( C_N(z) \) that stabilizes the polynomials

\[
p(z; \lambda) := B_D(z) - C_N(z)\lambda = z^m + \sum_{i=0}^{m-1} (b_i - \lambda c_i) z^i
\]

for all \( \lambda \in [\hat{\lambda}, \bar{\lambda}] \). This ensures that, as long as the eigenvalues of \( A \) lie in the range \([\hat{\lambda}, \bar{\lambda}]\), the controller is stabilizing. This can be seen as a robust control problem, which we solve by solving an linear matrix inequality (LMI), as shown in the following. Indeed, observe that the stability of \( p(z; \lambda) \) is equivalent to the stability of the associated companion matrix

\[
F_c(\lambda) := F + \lambda G K,
\]

with

\[
F = \begin{bmatrix} 0 & 1 & 0 & \cdots \\ & \ddots & & \\ -b_0 & 0 & \cdots & 1 \end{bmatrix}, \quad G = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}, \quad K = \begin{bmatrix} c_0 & c_1 & \cdots & c_{m-1} \end{bmatrix}.
\]

Then, the goal is to find \( c_0, c_1, \ldots, c_{m-1} \) so that \( F_c(\lambda) \) is stable for all \( \lambda \in [\hat{\lambda}, \bar{\lambda}] \). We can write \( \lambda \) as the convex combination of the two extreme values by

\[
\lambda = \alpha \bar{\lambda} + (1 - \alpha) \hat{\lambda}, \quad \alpha = \frac{\bar{\lambda} - \lambda}{\bar{\lambda} - \hat{\lambda}}
\]

which allows us to use the following result [26, Th. 3].

**Lemma 1 (Stabilizing controller design):** The matrix \( F_c(\lambda) \) is asymptotically stable for all \( \lambda \in [\hat{\lambda}, \bar{\lambda}] \) if there exist symmetric matrices \( P, P > 0, Q \in \mathbb{R}^{m \times m}, \) and \( R \in \mathbb{R}^{1 \times m} \) such that the following two linear matrix inequalities are verified:

\[
\begin{bmatrix} P & \frac{F - \lambda G}{\lambda - \hat{\lambda}} \\ \frac{\hat{\lambda} F^\top + R^\top \hat{\lambda} G}{\lambda - \hat{\lambda}} & Q + Q^\top - P \end{bmatrix} > 0,
\]

\[
\begin{bmatrix} P & \frac{\bar{\lambda} F^\top + R^\top \bar{\lambda} G}{\bar{\lambda} - \lambda} \\ \frac{\bar{\lambda} F^\top + R^\top \bar{\lambda} G}{\lambda - \hat{\lambda}} & Q + Q^\top - P \end{bmatrix} > 0.
\]

A stabilizing controller is then \( K = RQ^{-1} \).

**Remark 3 (Computing the controller):** We remark that the LMIs’ dimension depends on the degree \( m \) of \( B_D(z) \), and not on the size of the problem \( n \). We further remark that, depending on the model of \( b_k \) and the bounds \( \hat{\lambda} \) and \( \bar{\lambda} \), there are cases in which the controller in Lemma 1 may not exist, that is, in which the LMIs are not feasible. While running the numerical experiments, we observed that, in some cases, the LMIs tend to be infeasible when the roots of \( B_D(z) \) are close to one and the condition number of \( A (\hat{\lambda} / \bar{\lambda}) \) is large. Finally, while in the following, we assume that a stabilizing controller does indeed exist and can be computed, if this is not the case we can fall back to the online gradient descent (cf., Remark 2).

C. Online Algorithm

We are finally ready to propose the online algorithm derived from the scheme of Fig. 1. Since \( X(z) = C(z)E(z) \), and with the choice (4), it is straightforward to see that the system defines the following online algorithm:

\[
w_{k+1} = (F \otimes I) w_k + (G \otimes I) \nabla f_k(x_k)
\]

(7a)

\[
x_{k+1} = (K \otimes I) w_{k+1}
\]

(7b)

where as defined above, \( \nabla f_k(x_k) = e_k = Ax_k + b_k, \) and \( w \in \mathbb{R}^{nm} \) is a set of \( m \) auxiliary vectors, representing the state of the closed-loop canonical control form realization. Notice that the updates of algorithm (7) are fully defined with the knowledge of 1) the internal model \( B_D(z) \), and 2) the bounds \( \hat{\lambda} \) and \( \bar{\lambda} \), which, as aforementioned, is a common assumption in (online) optimization. Indeed, of the model \( Z[b_k] \), we need only to know the poles, while no information on the numerator \( B_N(z) \) is required.

**1) Nonquadratic Problems:** We remark that algorithm (7) can be applied in a straightforward manner also to the more general (not necessarily quadratic) problem (1), since its updates receive as input the gradient \( \nabla f_k(x_k) \). But, while for more general costs, we can still...
access the bounds \( \hat{\lambda} \) and \( \bar{\lambda} \); 1 we no longer have access to the internal model \( B_D(z) \). Instead, \textit{which internal model is used becomes a design parameter of the algorithm}. Tuning the internal model can be done by leveraging some information on the variability of the cost function (as discussed in Example 3), or it may be estimated from historical data. Regardless of how the internal model is derived, when deriving it we need to keep in mind that smaller models (having lower degree \( m \)) are generally better. Indeed, larger models may lead to infeasibility of the controller LMIs (cf., Remark 3), or much longer transients (cf., Sections IV-B and IV-C).

**Example 3 (Periodic \( f_k \)):** Consider a problem of the form (1) in which the cost function is periodic with period \( P \). In this case, a reasonable choice of internal model is the transfer function

\[
B_D(z) = (z - 1) \prod_{\ell=1}^L (z^2 - 2 \cos(\ell \theta) z + 1)
\]

whose poles are multiples of the frequency \( \theta = (2\pi/P)T_k \).\(^2\)

**III. CONVERGENCE ANALYSIS**

As aforementioned, the proposed algorithm (7) can be applied both to the quadratic problem, for which (2) holds, as well as to the more general problem (1). The following sections then analyze the convergence of the algorithm in both scenarios, providing bounds to the tracking error \( \|x_k - x_k^*\| \) and discussing their implications.

**A. Convergence for Quadratic Problem (2)**

**Proposition 1 (Convergence for (2)):** Assume that \( f_k \) is quadratic as in (2) and that Assumptions 1 and 2 hold. Assume that the controller \( C(z) = C_D(z) \) is such that

1. **internal model:** \( C_D(z) \) includes all poles of \( B_D(z) \);
2. **stability:** \( B_D(z) - C_N(z)A \) is stable for all \( \lambda \in [\bar{\lambda}, \hat{\lambda}] \).

Then, the output \( \{x_k\}_{k \in \mathbb{N}} \) of the online algorithm (7) is such that

\[
\lim_{k \to \infty} \|x_k - x_k^*\| = 0.
\]

**Proof:** By choosing a controller that stabilizes the matrix \( F_\lambda(k), \lambda \in [\bar{\lambda}, \hat{\lambda}] \), the poles of \( E(z) \) [cf., (6)] are asymptotically stable, and the gradient \( e_k \) converges to zero, implying the thesis. \( \blacksquare \)

**1) Convex Problems:** So far we have dealt, according to Assumption 1, with strongly convex problems. In the following, we show that the results derived in this section can be leveraged to prove convergence of (7) for convex problems. In particular, we are able to show that the output of (7) converges linearly to the subspace of solutions.

**Proposition 2 (Convergence for convex (2)):** In the setup of Proposition 1, assume that \( A = A^+ \geq 0 \) with \( \text{rank}(A) = r < n \), and assume that the nonzero eigenvalues lie in \([\bar{\lambda}, \hat{\lambda}]\). Assume that \( b_k \in \text{range}(A) \) for all \( k \) and let \( X_k^\dagger \) be the (affine) set of solutions to (2). Assume, moreover, that the controller \( C(z) = C_D(z) \) verifies (c1) and (c2) of Proposition 1. Then, the output \( \{x_k\}_{k \in \mathbb{N}} \) of the online algorithm (7) is such that

\[
\lim_{k \to \infty} \text{sup} \ \text{dist}(x_k, X_k^\dagger) = 0.
\]

**Proof:** Consider the eigendecomposition of the (now positive semidefinite) matrix \( A, A = VAV^\top \), where we let \( V = \begin{bmatrix} U & W \end{bmatrix} \).

\( ^1 \) They represent the strong convexity and smoothness moduli of \( f_k \).

\( ^2 \) Recall that \( T_k \) is the ‘sampling time’, i.e., the time that elapses between the arrival of two consecutive cost functions.

\[ A = \text{diag}\{\lambda_1, \ldots, \lambda_r, 0, \ldots, 0\}, \quad U \in \mathbb{R}^{n \times r} \quad \text{having as columns the eigenvectors of the nonzero eigenvalues}. \]

With this notation, we know that the affine set of solution is defined by \( X_k^\dagger = \{-A^+b_k + (I - A^+A)\mathbf{w} \mid \mathbf{w} \in \mathbb{R}^n\} \), where \( A^+ \) is the pseudoinverse

\[
A^+ = V \text{diag}\{\lambda_1^{-1}, \ldots, \lambda_r^{-1}, 0, \ldots, 0\} V^\top.
\]

The projection of \( x_k \) onto \( X_k^\dagger \) is defined as \( \text{proj}_{X_k^\dagger}(x_k) := (I - A^+A)x_k - A^+b_k \), \( 27, \text{Sec. 6.2.2} \), and our goal is to prove that, asymptotically, \( x_k \to \text{proj}_{X_k^\dagger}(x_k) \to 0 \). Indeed, since \( \text{proj}_{X_k^\dagger}(x_k) \) is the element of \( X_k^\dagger \) closest (in Euclidean norm) to \( x_k \), this implies that the distance of \( x_k \) from \( X_k^\dagger \) converges to zero asymptotically.

By the aforementioned definitions, it is straightforward to see that \( x_k \to \text{proj}_{X_k^\dagger}(x_k) = A^+e_k \) and if we prove that \( A^+e_k \to 0 \) as \( k \to \infty \), then the thesis is proved.

**B. Convergence for the General Problem (1)**

So far, we have leveraged the particular class of quadratic problems (2) to inspire the design of the proposed online algorithm, and proved in the previous section that for these problems the algorithm achieves perfect tracking of \( x_k^\dagger \). Now turn our attention to the more general problem (1) where we assume the costs \( f_k \) satisfy the following assumption.

**Assumption 3 (General problem):** The cost functions \( \{f_k\}_{k \in \mathbb{N}} \) can be decomposed as

\[
f_k(x) = \frac{1}{2} x^\top Ax + b_k^\top x + \varphi_k(x)
\]

where:

i) \( A \) satisfies Assumption 1;
ii) \( \{b_k\}_{k \in \mathbb{N}} \) satisfies Assumption 2 and additionally \( \|b_k\| \leq \beta \) for all \( k \in \mathbb{N} \);
iii) \( \varphi_k(x) = \varphi_k^\dagger(x) + \varphi_k^\ast(x) \) with

\[
\|\nabla \varphi_k^\dagger(x)\| \leq \delta \quad \text{and} \quad \|\nabla \varphi_k^\ast(x)\| \leq \gamma \|x\| \quad \forall x \in \mathbb{R}^n.
\]
Before proving the main convergence result, let us comment on the choice of the cost (10). We can write the cost at hand as \( f_k(x) = f_k(x) + \varphi_k(x) \), where \( f_k(x) = \frac{1}{2} x^T A x + b_k^T x \). We can then interpret (10) as a perturbation of the quadratic \( f_k \), where the perturbation has both a term whose gradient is bound and another term whose gradient has bound gain. Consider now the scheme of Fig. 1 in which we replace \( \nabla f_k(\cdot) \) with the perturbed gradient \( \nabla f_k(\cdot) + \nabla \varphi_k(\cdot) \). Clearly, if \( \nabla \varphi_k(x) = 0 \), Proposition 1 applies\(^3\) and we can control the tracking error to zero. We can then think of \( d_k := \nabla \varphi_k(x_k) \) as a disturbance, as depicted in Fig. 2.

The following Section III-B1 provides a convergence result for all nonquadratic costs that can be written as the “perturbed” quadratic (10), of which quadratic costs with time-varying Hessian are an example (see Section III-B2). In Section III-B3, we further analyze the convergence for quadratic problems when only an inexact knowledge of the internal model is available.

1) Main Convergence Result: The following result is a consequence of the small gain theorem, which allows us to prove that the disturbance \( d_k \) leads to a bounded (but, in general, nonzero) tracking error for the proposed algorithm (7). The proof is given in the Appendix.

Proposition 3 (Convergence for (1)): Consider problem (1) for which Assumption 3 holds. Assume that the controller \( C(z) = \frac{C_D(z)}{C_N(z)} \) such that the following holds:

\( C_D(z) \) includes all poles of \( B_D(z) \);
\( C(z) - C_N(z) \) is stable for all \( \lambda \in [\bar{\lambda}, \lambda] \);
\( C(z) - C_N(z) \) achieves small gain: \( \|C(z)(I - C(z)A)^{-1}\|_{\infty} < \frac{1}{2} \).

Then, the output \( \{x_k\}_{k \in \mathbb{N}} \) of the online algorithm (7) is such that

\[
\limsup_{k \to \infty} \|x_k - x_k^*\| \leq \|\left(I - C(z)A\right)^{-1}\|_{\infty} \delta + \beta \gamma \left\|\frac{C(z)}{C(z)} - C_N(z)\right\|_{\infty} \left\|\left(I - C(z)A\right)^{-1}\right\|_{\infty},
\]

where \( \delta \) is the disturbance and \( \beta \) is the gain margin.

The result we derived depends on the \( \| \cdot \|_\infty \)-norm of the two transfer matrices \( (I - C(z)A)^{-1} \) and \( C(z)(I - C(z)A)^{-1} \). The following lemma provides bounds to these norms that are useful in practice when numerically designing the controller.

Lemma 2: We have the following bounds:

\[
\left\|\left(I - C(z)A\right)^{-1}\right\|_{\infty} \leq \max_{\lambda \in [\bar{\lambda}, \lambda]} \left\|\left(1 - \bar{\lambda} C(z)\right)^{-1}\right\|_{\infty},
\]

\[
\left\|C(z)(I - C(z)A)^{-1}\right\|_{\infty} \leq \frac{\max_{\lambda \in [\bar{\lambda}, \lambda]} \left\|C(z)\right\|_{\infty}}{\left(1 - \bar{\lambda} C(z)\right)^{-1}}.
\]

\(^3\)Indeed, by Assumption 3(i), the cost \( f_k \) is strongly convex.

Proof: Consider the eigendecomposition \( A = V \Lambda V^T \), then by definition we have

\[
\left\|\left(I - C(z)A\right)^{-1}\right\|_{\infty} = \left\|V \left(I - C(z)\Lambda\right)^{-1} V^T\right\|_{\infty} \leq \left\|\left(I - C(z)\Lambda\right)^{-1}\right\|_{\infty} \leq \max_{\theta \in [0, \pi]} \left(\left(I - C(e^{i\theta})\Lambda\right)^{-1}\right)^{-1} \leq \max_{\theta \in [0, \pi]} \left(1 - C(e^{i\theta})\Lambda\right)^{-1}
\]

where we used Assumption 3(i). The thesis follows by swapping the maxima (which we can do since they are defined on compacts). The second bound follows using the same arguments.

2) Time-Varying \( A_k \): We consider now the following quadratic, online optimization problem:

\[
x_k^* = \arg \min_{x \in \mathbb{R}^n} f_k(x) := \frac{1}{2} x^T A_k x + b_k^T x
\]

which, differently from (2), also the quadratic term is time-varying. We prove the following convergence result in which the Hessian \( A_k \) is modeled as the sum of a static matrix and a time-varying perturbation.

Corollary 1 (Time-varying \( A_k \)): Consider problem (12) for which Assumption 1, Assumption 3(i), and Assumption 3(ii) hold. Furthermore, assume that we can write \( A_k = A + \hat{A}_k \) with \( I \preceq A \preceq \lambda I \) and \( \|\hat{A}_k\| \leq \gamma \) for all \( k \). Assume that the controller \( C(z) = \frac{C_D(z)}{C_N(z)} \) satisfies (c1) and (c2) of Proposition 3. Then, the output \( \{x_k\}_{k \in \mathbb{N}} \) of the online algorithm (7) satisfies the bound (11) with \( \delta = 0 \).

Proof: We can rewrite the cost (12) as \( f_k(x) = \hat{f}_k(x) + \varphi_k(x) \) with \( \hat{f}_k(x) = \frac{1}{2} x^T A x + b_k x \) and \( \varphi_k(x) = \hat{A}_k x \). The cost thus conforms to (10) with \( \delta = 0 \) since \( \|\nabla \varphi_k(x)\| \leq \|\hat{A}_k\|\|x\| \leq \gamma \|x\| \), and we can apply Proposition 3.

Corollary 1 shows how a time-varying perturbation on the Hessian of the quadratic cost leads to a bounded asymptotic tracking error. Clearly, if \( \gamma = 0 \) (the Hessian is static), then we recover the result of Proposition 1.

3) Inexact Internal Model: The convergence results of Section III-A were derived under the assumption that precise knowledge of the model \( \{b_k\}_{k \in \mathbb{N}} \) is available. In this section, we are interested in evaluating the performance of the online algorithm (7) for the quadratic problem (2) when it relies on an inexact knowledge of \( B_D(z) \), and specifically on the inexact model

\[
\hat{B}_D(z) = z^{n-1} + \sum_{i=0}^{m-1} \hat{b}_i z^i.
\]

In this setup, we can derive the following result, whose proof follows a similar argument to Proposition 3 based on the small gain theorem.

Proposition 4 (Inexact internal model): Consider problem (2) for which Assumptions 1 and 3(ii) hold. Assume that the controller \( C(z) = \frac{C_D(z)}{C_N(z)} \) with \( C_D(z) = \hat{B}_D(z) \) is such that \( \hat{B}_D(z) - C_N(z) \) is stable for all \( \lambda \in [\bar{\lambda}, \lambda] \). Then, the output \( \{x_k\}_{k \in \mathbb{N}} \) of the online algorithm (7) is such that

\[
\limsup_{k \to \infty} \|x_k - x_k^*\| \leq \beta \left\|\left(B_D(z)I - C_N(z)A\right)^{-1}\right\|_{\infty} \|\delta\|_1
\]

where \( \delta = \left[b_0 - \hat{b}_0 \quad \cdots \quad b_{m-1} - \hat{b}_{m-1}\right]^T \).

Proof: Let \( B(z) = \frac{B_D(z)}{B_D(z)} \) be the Z-transform of \( b_k \). Then, starting from (6), it is straightforward to see that \( E(z) = E'(z) + E''(z) \) where

\[
E'(z) = (I + C(z)A)^{-1} \frac{B_N(z)}{B_D(z)}
\]
\[ E'(z) = (I + C(z)A)^{-1} \frac{B_D(z) - \hat{B}_D(z)}{B_D(z)} B(z). \]

From Proposition 1, we know that the inverse Z-transform \( e' \) of \( E'(z) \) is converging to zero. As far as \( E'(z) \), by applying (1) in the Appendix, we can argue that its inverse Z-transform \( e' \) is such that
\[
\| e' \| \leq \left\| (I + C(z)A)^{-1} \frac{B_D(z) - \hat{B}_D(z)}{B_D(z)} \right\|_\infty \beta.
\]

Observe finally that
\[
\left\| (I + C(z)A)^{-1} \frac{B_D(z) - \hat{B}_D(z)}{B_D(z)} \right\|_\infty \leq \left\| (\hat{B}_D(z)I - C_N(z)A)^{-1} \right\|_\infty \| B_D(z) - \hat{B}_D(z) \|_\infty
\]
where we have
\[
\| B_D(z) - \hat{B}_D(z) \|_\infty = \max_{|z| \leq 1, |z| \leq 1} \sum_{j=0}^{m-1} |b_j - \hat{b}_j| = |\delta|_1
\]
and where a bound for \( \| (\hat{B}_D(z)I - C_N(z)A)^{-1} \|_\infty \) can be derived along the lines of Lemma 2.

We remark that if the internal model is exact, then \( \hat{B}_D(z) = B_D(z) \) and we recover the result of Proposition 1.

IV. SIMULATIONS

A. Time-Varying Linear Term

In this section, we compare the performance of the different proposed algorithms when applied to problem (2), where only the linear term is time-varying.\(^4\)

We consider problem (2) with \( x \in \mathbb{R}^n, n = 500, \) and \( A = VA^\top, \) where \( V \) is a randomly generated orthogonal matrix and \( A \) is diagonal with elements in \([1, 10]\). We employ four different models for the linear term \( \{b_k\}_{k \in \mathbb{N}}: \)
1. ramp \( b_k = kT_\omega b, b \in \mathbb{R}^n; \)
2. sine \( b_k = \sin(\omega kT_\omega)1, \omega = 1; \)
3. sine plus ramp \( b_k = \sin(\omega kT_\omega)1 + kT_\omega b; \)
4. squared sine \( b_k = \sin^2(\omega kT_\omega)1. \)

In Fig. 3, we report the tracking error for the online gradient [9], the predicted online gradient\(^5\) [29], and the control-based method (7). In accordance with the theoretical results of Section II, the control-based method (with adequately chosen internal model) can achieve practically zero tracking error. On the other hand, both the gradient and predicted gradient only achieve a nonzero tracking error, with the error of the latter being smaller. This is further highlighted in Table I, which reports the asymptotic tracking error\(^6\) for the different algorithms and problem models.

The previous numerical results were derived by applying the control-based (7) equipped with an exact model of \( b_k \) dynamics. We turn now to evaluating the performance of the proposed method when an inexact model is available, as discussed in Section III-A. Consider the case (ii) of a sinusoidal linear term, with \( \omega = 1 \) being the unique parameter that defines the model of \( b_k \), indeed \( B_D(z) = z^2 - 2 \cos(\omega kT_\omega)z - 1 \).

We run (7) equipped with a (possibly inexact) model characterized by \( \hat{b}_1 = 2 \cos(\pm 1) \). However, (7) still achieves better results than the online gradient, even when \( \hat{\omega} \) is wrong by 50% of the actual value of \( \omega \).

B. Time-Varying Quadratic Term

We consider now problem (12) where \( A_k = A + \Delta_k, \) with \( A = VA^\top \) and \( \Delta_k = V \text{diag} \{ \sin(\omega kT_\omega)d \} V^\top (d \in \mathbb{R}^n \) being a vector

\(^4\)All the simulations were implemented using CVXOPT [28].
\(^5\)This algorithm is characterized by the update \( x_{k+1} = x_k - \alpha(2V_f(x_k) - j \bar{f}_{k-1}(x_k)), k \in \mathbb{N}. \)
\(^6\)Computed as the maximum error over the last 4/5 of the simulation.
with decreasing components), and $A$ chosen such that $A_k$ have eigenvalues in $[1, 10]$. For simplicity, the linear term is an assumed constant, $b_k = \bar{b} \in \mathbb{R}^n$, $n = 500$.

As discussed in Section II-C, the proposed algorithm can be applied in this scenario, and in Fig. 5, we report its tracking error as compared with predicted and online gradient. Since to define algorithm (7), we need to specify a model, we compared the three different options (cf., Example 3)

$$B_D(z) = (z - 1) \prod_{\ell=1}^{L} (z^2 - 2 \cos(\ell \omega T) z + 1), \quad L = 1, 2, 3. \tag{14}$$

As we can see, the control-based method outperforms the other methods for all three choices of $L$, and we notice that larger models ($L$ larger) yield better performance, since they consider a higher number of multiples of the base frequency. Table II reports the exact asymptotic errors achieved by the different algorithms of the solution trajectory w.r.t. state-of-the-art methods. Future research directions that we will explore are the application of this paradigm to constrained problems (as stemming, e.g., in model predictive control), by designing primal-dual online algorithms that can handle time-varying equality and inequality constraints.

### C. Nonquadratic

In this section, we discuss the application of the proposed algorithms to the nonquadratic problem (1). In particular, we consider the following cost function, adapted from [11]:

$$f_k(x) = \frac{1}{2} x^T A x + \langle b, x \rangle + \sin(\omega T_k \cdot z) \log (1 + \exp(\langle c, x \rangle)) \tag{15}$$

where $\omega = 1, b \in \mathbb{R}^n$, and $n = 500$ are randomly generated, and $c \in \mathbb{R}^n$ is such that $\|c\| = 1$. The matrix $A$ is generated as in Section IV-A, and $f_k$, thus, satisfies Assumption 3 with $\delta = \|c\|^2/4$ and $\gamma = 0$. In Fig. 6, we report the tracking error of the proposed algorithm, with the control-based one using the three models (14). As we can see, similarly to the case of Section IV-B, the control-based method outperforms the other algorithms, to the cost of a slower convergence rate. Table II reports the exact asymptotic error achieved by the different algorithms.

### V. Conclusion

In this article, we proposed a model-based approach to the design of online optimization algorithms, with the goal of improving the tracking of the solution trajectory w.r.t. state-of-the-art methods. Focusing first on quadratic problems, we have proposed a novel online algorithm that achieves zero tracking error. Second, we have discussed the use of this algorithm for more general costs, and analyzed its convergence. The numerical results that we present validate our theoretical results and show the promise of our approach to online optimization algorithms design, which outperforms state-of-the-art methods. Future research directions that we will explore are the application of this paradigm to constrained problems (as stemming, e.g., in model predictive control), by designing primal-dual online algorithms that can handle time-varying equality and inequality constraints.

### APPENDIX A

#### PROOF OF PROPOSITION 3

Before proceeding with the proof of Proposition 3, we briefly recall some relevant notions, see [30] for a comprehensive introduction.

In the following, for any signal $a_k \in \mathbb{R}^n$, $k \geq 0$, let $\|a_k\|_\infty = \sup_{0 \leq k < \infty} \|a_k\|$.

1) Consider the system with input $u_k \in \mathbb{R}^n$, output $y_k \in \mathbb{R}^p$, and stable transfer matrix $T(z) \in \mathbb{R}^{p \times n}$ so that the $Z$-transforms $U(z) = Z[u_k]$ and $Y(z) = Z[y_k]$ are in the relation $Y(z) = T(z)U(z)$. Then, we know that if $\|u_k\|_\infty \leq \beta$, then $\|y_k\|_\infty \leq \|T(z)\|_\infty \beta$, where

$$\|T(z)\|_\infty := \max_{\theta \in [0, 2\pi]} \sigma_{\max}(T(e^{i\theta})) .$$

2) Moreover, assume now that $T(z)$ is in feedback with a (possibly nonlinear) operator $\varphi: \mathbb{R}^n \to \mathbb{R}^n$, such that $u_k = \varphi(y_k)$. If $\varphi$ is such that $\|\varphi(x)\| \leq \gamma \|x\|$ for $\gamma > 0$, then, according to the small gain theorem, if $\|T(z)\|_\infty \gamma < 1$, then the interconnection is asymptotically stable.
Proof of Proposition 3:

Let $d_k = \nabla \psi_k(x_k)$ and let $D(z) = Z[d_k]$. Following the same steps leading to (5), we can write

$$E(z) = (I - C(z)A)^{-1}B(z) + D(z) = E^b(z) + E^d(z)$$

(16)

where $E^b(z)$ is given in (5) and $E^d(z) = (I - C(z)A)^{-1}D(z)$. The results of Section III-A imply that, by choosing a controller that satisfies (c1) and (c2), the first component $e^b_k = Z^{-1}[E^b(z)]$ converges to zero. Hence, we can focus on the second term $e^d_k = Z^{-1}[E^d(z)]$.

From (r1), we have $\|e^d_k\|_{\infty} \leq \|(I - C(z)A)^{-1}\|_{\infty}\|d_k\|_{\infty}$, and, using Assumption 3(ii),

$$\|d_k\|_{\infty} \leq \|\nabla \psi_k(x_k)\| + \|\nabla \psi_k(x_k)\| \leq \delta + \gamma \|x_k\|_{\infty}. \quad (17)$$

We need, therefore, to ensure that $\|x_k\|_{\infty}$ is bounded for all $k \in \mathbb{N}$ in order to guarantee that the disturbance is bounded as well.

By the fact that $X(z) = C(z)E(z)$, and using (16) and (r1), we can write $\|x_k\|_{\infty} \leq \|C(z)(I - C(z)A)^{-1}\|_{\infty}\|d_k\|_{\infty}$. Using now (17) and the bound $\|d_k\|_{\infty} \leq \beta$ [cf., Assumption 3(i)], we get

$$\|x_k\|_{\infty} \leq \|C(z)(I - C(z)A)^{-1}\|_{\infty}(\beta + \delta + \gamma \|x_k\|_{\infty}). \quad (18)$$

Clearly, the first two terms are bounded, and we need to guarantee that $\|C(z)(I - C(z)A)^{-1}\|_{\infty}\|\gamma\|x_k\|_{\infty}$ is as well. But this can be done by applying the small gain theorem recalled in (23), therefore $\|x_k\|_{\infty}$ is bounded provided that $C(z)$ satisfies (c3). Finally, using (18) into (17) and rearranging we get

$$\|d_k\|_{\infty} \leq \delta + \beta \gamma \|C(z)(I - C(z)A)^{-1}\|_{\infty} \quad (19)$$

By $\|e^d_k\|_{\infty} \leq \|(I - C(z)A)^{-1}\|_{\infty}\|d_k\|_{\infty}$ the thesis follows. \hfill \blacksquare

Acknowledgment

The main part of the work was carried out when N. Bastianello was at the University of Padova, Italy.

References

[1] D. Liao-McPherson, M. Nicotra, and I. Kolmanovsky, “A semismooth predictor corrector method for real-time constrained parametric optimization with applications in model predictive control,” in Proc. IEEE Conf. Decis. Control, 2018, pp. 3600–3607.

[2] S. Paternain, M. Morari, and A. Ribeiro, “Real-time model predictive control based on prediction-correction algorithms,” in Proc. IEEE 58th Conf. Decis. Control, 2019, pp. 5285–5291.

[3] E. C. Hall and R. M. Willert, “Online convex optimization in dynamic environments,” IEEE J. Sel. Topics Signal Process., vol. 9, no. 4, pp. 647–662, Jun. 2015.

[4] S. M. Fosson, “Centralized and distributed online learning for sparse time-varying optimization,” IEEE Trans. Autom. Control, vol. 66, no. 6, pp. 2542–2557, Jun. 2021.

[5] A. Natali, M. Coutino, E. Iufi, and G. Leus, “Online time-varying topology identification via prediction-correction algorithms,” in Proc. IEEE Int. Conf. Acoust., Speech Signal Process., 2021, pp. 5400–5404.

[6] S. Shalev-Shwartz, “Online learning and online convex optimization,” Foundations Trends Mach. Learn., vol. 4, no. 2, pp. 107–194, 2011.

[7] R. Dixit, A. S. Bedi, R. Tripathi, and R. Rajawat, “Online learning with inexact proximal online gradient descent algorithms,” IEEE Trans. Signal Process., vol. 67, no. 5, pp. 1338–1352, Mar. 2019.

[8] T.-H. Chang, M. Hong, H.-T. Wai, X. Zhang, and S. Lu, “Distributed learning in the nonconvex world: From batch data to streaming and beyond,” IEEE Signal Process. Mag., vol. 37, no. 3, pp. 26–38, May 2020.

[9] E. Dall’Anese, A. Simonetto, S. Becker, and L. Madden, “Optimization and learning with information streams: Time-varying algorithms and applications,” IEEE Signal Process. Mag., vol. 37, no. 3, pp. 71–83, May 2020.

[10] A. Simonetto, E. Dall’Anese, S. Paternain, G. Leus, and G. B. Giannakis, “Time-varying convex optimization: Time-structured algorithms and applications,” Proc. IEEE, vol. 108, no. 11, pp. 2032–2048, Nov. 2020.

[11] A. Simonetto, A. Mokhtari, A. Koppel, G. Leus, and A. Ribeiro, “A class of prediction-correction methods for time-varying convex optimization,” IEEE Trans. Signal Process., vol. 64, no. 17, pp. 4576–4591, Sep. 2016.

[12] A. Simonetto and E. Dall’Anese, “Prediction-correction algorithms for time-varying constrained optimization,” IEEE Trans. Signal Process., vol. 65, no. 20, pp. 5481–5494, Oct. 2017.

[13] M. Fazlyab, S. Paternain, V. M. Preciado, and A. Ribeiro, “Prediction-correction interior-point method for time-varying convex optimization,” IEEE Trans. Automat. Control, vol. 63, no. 7, pp. 1973–1986, Jul. 2018.

[14] Y. Li, G. Qu, and N. Li, “Online optimization with predictions and switching costs: Fast algorithms and the fundamental limit,” IEEE Trans. Automat. Control, vol. 66, no. 10, pp. 4761–4768, Oct. 2021.

[15] A. Bernstein, E. Dall’Anese, and A. Simonetto, “Online primal-dual methods with measurement feedback for time-varying convex optimization,” IEEE Trans. Signal Process., vol. 67, no. 8, pp. 1978–1991, Apr. 2019.

[16] M. Colombino, E. Dall’Anese, and A. Bernstein, “Online optimization as a feedback controller: Stability and tracking,” IEEE Trans. Control Netw. Syst., vol. 7, no. 1, pp. 422–432, Mar. 2020.

[17] A. Hauswirth, S. Bolognani, G. Hug, and F. Dorfler, “Timestep separation in autonomous optimization,” IEEE Trans. Automat. Control, vol. 66, no. 2, pp. 611–624, Feb. 2021.

[18] L. Lessard, B. Recht, and A. Packard, “Analysis and design of optimization algorithms via integral quadratic constraints,” SIAM J. Optim., vol. 26, no. 1, pp. 57–95, Jan. 2016.

[19] A. Sundararajan, B. V. Scy, and L. Lessard, “A canonical form for first-order distributed optimization algorithms,” in Proc. Amer. Control Conf., 2019, pp. 4075–4080.

[20] G. Zhang, X. Bao, L. Lessard, and R. Grosse, “A unified analysis of first-order methods for smooth games via integral quadratic constraints,” J. Mach. Learn. Res., vol. 22, no. 103, pp. 1–39, 2021.

[21] C. Scherer and C. Ebenbauer, “Convex synthesis of accelerated gradient algorithms,” SIAM J. Control Optim., vol. 59, no. 6, pp. 4615–4645, Jan. 2021.

[22] G. França, D. P. Robinson, and R. Vidal, “Gradient flows and proximal splitting methods: A unified view on accelerated and stochastic optimization,” Phys. Rev. E, vol. 103, no. 5, May 2021, Art. no. 053304.

[23] S. Shahrampour and A. Jadbabaie, “An online optimization approach for multi-agent tracking of dynamic parameters in the presence of adversarial noise,” in Proc. Amer. Control Conf., 2017, pp. 3306–3311.

[24] S. Shahrampour and A. Jadbabaie, “Distributed online optimization in dynamic environments using mirror descent,” IEEE Trans. Automat. Control, vol. 63, no. 3, pp. 714–725, Mar. 2018.

[25] M. S. Fadali and A. Visioli, Digital Control Engineering: Analysis and Design, 3rd ed. New York, NY, USA: Acad. Press, 2019.

[26] M. de Oliveira, J. Bernussou, and J. Geromel, “A new discrete-time robust stability condition,” Syst. Control Lett., vol. 21, pp. 143–149, 1993.

[27] N. Parikh and S. Boyd, “Proximal algorithms,” Found. Trends Optim., vol. 1, no. 3, pp. 127–239, 2014.

[28] N. Bastianello, “tvopt: A python framework for time-varying optimization,” in Proc. IEEE 66th Conf. Decis. Control, 2021, pp. 227–232.

[29] M. Vidyasagar, “Control system synthesis: A factorization approach, Part II,” Synth. Lectures Control, Sys. Sci. Eng., vol. 2, no. 1, pp. 1–227, 2011.