A Class of Prediction-Correction Methods for Time-Varying Convex Optimization

Simonetto, Andrea; Mokhtari, Aryan; Koppel, Alec; Leus, Geert; Ribeiro, Alejandro

DOI
10.1109/TSP.2016.2568161

Publication date
2016

Published in
IEEE Transactions on Signal Processing

Citation (APA)
Simonetto, A., Mokhtari, A., Koppel, A., Leus, G., & Ribeiro, A. (2016). A Class of Prediction-Correction Methods for Time-Varying Convex Optimization. *IEEE Transactions on Signal Processing*, *64*(17), 4576-4591. Article 7469393. https://doi.org/10.1109/TSP.2016.2568161

A Class of Prediction-Correction Methods for Time-Varying Convex Optimization

Andrea Simonetto, *Member, IEEE*, Aryan Mokhtari, Alec Koppel, Geert Leus, *Fellow, IEEE*, and Alejandro Ribeiro, *Member, IEEE*

***Abstract*—This paper considers unconstrained convex optimization problems with time-varying objective functions. We propose algorithms with a discrete time-sampling scheme to find and track the solution trajectory based on prediction and correction steps, while sampling the problem data at a constant rate of $1/h$, where h is the sampling period. The prediction step is derived by analyzing the iso-residual dynamics of the optimality conditions. The correction step adjusts for the distance between the current prediction and the optimizer at each time step, and consists either of one or multiple gradient steps or Newton steps, which respectively correspond to the gradient trajectory tracking (GTT) or Newton trajectory tracking (NTT) algorithms. Under suitable conditions, we establish that the asymptotic error incurred by both proposed methods behaves as $O(h^2)$, and in some cases as $O(h^4)$, which outperforms the state-of-the-art error bound of $O(h)$ for correction-only methods in the gradient-correction step. Moreover, when the characteristics of the objective function variation are not available, we propose approximate gradient and Newton tracking algorithms (AGT and ANT, respectively) that still attain these asymptotical error bounds. Numerical simulations demonstrate the practical utility of the proposed methods and that they improve upon existing techniques by several orders of magnitude.**

***Index Terms*—Time-varying optimization, non-stationary optimization, parametric programming, prediction-correction methods.**

I. Introduction

In this paper, we consider unconstrained optimization problems whose objective functions vary continuously in time. In particular, consider a variable $\boldsymbol{x} \in \mathbb{R}^n$ and a non-negative continuous time variable $t \in \mathbb{R}_+$, which determine the choice of a *smooth strongly convex function* $f : \mathbb{R}^n \times \mathbb{R}_+ \to \mathbb{R}$. We study the problem

$$\boldsymbol{x}^*(t) := \underset{\boldsymbol{x}\in\mathbb{R}^n}{\operatorname{argmin}} f(\boldsymbol{x}; t), \quad \text{for } t \geqslant 0. \tag{1}$$

Our goal is to determine the solution $\boldsymbol{x}^*(t)$ of (1) for each time t which corresponds to the solution *trajectory*. Time-varying optimization problems of the form (1) arise in control [3]–[5], when, for instance, one is interested in generating a control action such that the system remains close to a dynamical reference trajectory, as well as in signal processing [6], where one seeks to estimate a dynamical process based on time-varying observations. Other examples arise in robotics [7]–[11] and economics [12].

The problem in (1) can be solved based on a continuous time platform [13]–[16] or can be interpreted as a sequence of time-invariant problems. In particular, one could sample the objective functions $f(\boldsymbol{x}; t)$ at time instants t_k with $k = 0, 1, 2, \ldots$, and sampling period $h = t_k - t_{k-1}$, arbitrarily close to each other and then solve the resulting time-invariant problems

$$\boldsymbol{x}^*(t) := \underset{\boldsymbol{x}\in\mathbb{R}^n}{\operatorname{argmin}} f(\boldsymbol{x}; t). \tag{2}$$

By decreasing h, an arbitrary accuracy may be achieved when approximating (1) by (2). However, solving (2) for each sampling time t_k is not a viable option in most application domains, even for moderate-size problems. The requisite computation time for solving each instance of the problem often does not meet the requirements for real-time applicability, as in the control domain [17]. It is also challenging to reasonably bound the time each problem instance will take to be solved [18]. In short, the majority of iterative methods for convex problems with static objectives may not be easily extended to handle time-varying objectives, with the exception of when the changes in the objective occur more slowly than the time necessary for computing the optimizer.

Instead, we consider using the tools of *non-stationary* optimization [19]–[22] [23, Ch. 6] to solve problems of the form (1). In these works the authors consider perturbations of the time-varying problem when an initial solution $\boldsymbol{x}^*(t_0)$ is known. More recently, the work presented in [24] designs a gradient method for unconstrained optimization problems using an arbitrary starting point, which achieves a $\| \boldsymbol{x}(t_k) - \boldsymbol{x}^*(t_k) \| = O(h)$ asymptotic error bound with respect to the optimal trajectory. Time-varying optimization has also been studied in the context of *parametric programming*, where the optimization problem is parametrized over a parameter vector $\boldsymbol{p} \in \mathbb{R}^p$ that may represent time, as studied in [25]–[27]. Tracking algorithms for optimization problems with parameters that change in time are given in [12], [28] and are based on predictor-corrector schemes. Even though these algorithms are applicable to constrained problems, they assume the access to an initial solution $\boldsymbol{x}^*(t_0)$, which may not be available in practice. Some of the theoretical advances in

Manuscript received September 17, 2015; revised April 04, 2016 and May 02, 2016; accepted May 03, 2016. Date of publication May 12, 2016; date of current version July 22, 2016. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Anthony So. The work in this paper is supported in part by STW under the D2S2 project from the ASSYS program (project 10561) and in part by NSF CAREER CCF-0952867, and ONR N00014-12-1-0997. This paper expands the results and presents convergence proofs that are referenced in *Proceedings of the Forty-Ninth Asilomar Conference on Signals, Systems, and Computers*, November 2015 [1] and *Proceedings of the IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, December 2015 [2].

A. Simonetto and G. Leus are with the Department of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, 2826 CD Delft, The Netherlands (e-mail: a.simonetto@tudelft.nl; g.j.t.leus@tudelft.nl).

A. Mokhtari, A. Koppel, and A. Ribeiro are with the Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA 19104 USA (e-mail: aryanm@seas.upenn.edu; akoppel@seas.upenn.edu; aribeiro@seas.upenn.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSP.2016.2568161

these works have been used to ease the computational burden of sequential convex programming while solving nonconvex optimization problems, or nonlinear model predictive control [3], [29], [30].

In this paper, we design iterative discrete-time sampling algorithms initialized at an arbitrary point $\boldsymbol{x}_0$ which converge asymptotically to the solution trajectory $\boldsymbol{x}^*(t)$ up to an error bound which may be specified as arbitrarily small and depends on the sampling period h. In particular, the methods proposed here yield a sequence of approximate time-varying optimizers $\{\boldsymbol{x}_k\}$, for which $\limsup_{k\to\infty}\|\boldsymbol{x}_k - \boldsymbol{x}^*(t_k)\| \leqslant \delta$ with δ dependent on the sampling period h. To do so, we predict where the optimal continuous-time trajectory will be at the next sampling time and track the associated prediction error based upon estimating the curvature of the solution trajectory. Under suitable assumptions, we establish that the proposed prediction-correction scheme attains an asymptotic error bound of $O(h^2)$ and in some cases $O(h^4)$, which outperforms the $O(h)$ error bound achieved by the state-of-the-art method of [24].

In Section II, we analyze unconstrained optimization problems and we propose algorithms to track their time-varying solution which fall into the family of tracking algorithms with *an arbitrary* starting point. The proposed methods are based on a predictor-corrector approach, where the predictor step is generated via a Taylor expansion of the optimality conditions, and the correction step may either be a single or multiple gradient descent or Newton steps. In Section III, we show that our tracking methods converge to the solution trajectory asymptotically, with an error bound $O(h^2)$ (and in some cases $O(h^4)$ locally) dependent on the sampling period h. This error bound improves upon the existing methods which attain an $O(h)$ bound. We further extend the tracking framework to account for the case where the dependence of the cost function on the time parameter is not known a priori but has to be estimated, and establish that the $O(h^2)$ and the (local) $O(h^4)$ asymptotical error bound are achieved despite the associated estimation uncertainty. In Section IV we numerically analyze the performance of the proposed methods as compared with existing approaches. In particular, in Section IV.A we consider a scalar example and show the convergence bounds hold in practice, and in Section IV.B we apply the proposed method to a reference path following problem and use the tools developed here to yield an effective control strategy for an intelligent system. Finally, in Section V we close the paper by concluding remarks.

Notation: Vectors are written as $\boldsymbol{x} \in \mathbb{R}^n$ and matrices as $\boldsymbol{A} \in \mathbb{R}^{n\times n}$. We use $\|\cdot\|$ to denote the Euclidean norm, both in the case of vectors, matrices, and tensors. The gradient of the function $f(\boldsymbol{x};t)$ with respect to $\boldsymbol{x}$ at the point $(\boldsymbol{x},t)$ is indicated as $\nabla_{\boldsymbol{x}} f(\boldsymbol{x};t) \in \mathbb{R}^n$, while the partial derivative of the same function w.r.t. t at $(\boldsymbol{x},t)$ is written as $\nabla_t f(\boldsymbol{x};t) \in \mathbb{R}$. Similarly, the notation $\nabla_{\boldsymbol{xx}} f(\boldsymbol{x};t) \in \mathbb{R}^{n\times n}$ denotes the Hessian of $f(\boldsymbol{x};t)$ w.r.t. $\boldsymbol{x}$ at $(\boldsymbol{x},t)$, whereas $\nabla_{t\boldsymbol{x}} f(\boldsymbol{x};t) \in \mathbb{R}^n$ denotes the partial derivative of the gradient of $f(\boldsymbol{x};t)$ w.r.t. the time t at $(\boldsymbol{x},t)$, i.e., the mixed first-order partial derivative vector of the objective. The tensor $\nabla_{\boldsymbol{xxx}} f(\boldsymbol{x};t) \in \mathbb{R}^{n\times n\times n}$ indicates the third derivative of $f(\boldsymbol{x};t)$ w.r.t. $\boldsymbol{x}$ at $(\boldsymbol{x},t)$, the matrix $\nabla_{\boldsymbol{x}t\boldsymbol{x}} f(\boldsymbol{x};t) = \nabla_{t\boldsymbol{xx}} f(\boldsymbol{x};t) \in \mathbb{R}^{n\times n}$ indicates the time derivative of the Hessian of $f(\boldsymbol{x};t)$ w.r.t. the time t at $(\boldsymbol{x},t)$, and the vector $\nabla_{tt\boldsymbol{x}} f(\boldsymbol{x};t) \in \mathbb{R}^n$ indicates the second derivative in time of the gradient of $f(\boldsymbol{x};t)$ w.r.t. the time t at $(\boldsymbol{x},t)$.

II. Algorithm Definition

In this section we introduce a class of algorithms for solving optimization problem (1) using prediction and correction steps. In order to converge to the solution trajectory $\boldsymbol{x}^*(t)$, we generate a sequence of near optimal decision variables $\{\boldsymbol{x}_k\}$ by taking into account both how the solution changes in time and how different our current update is from the optimizer at each time step.

A. Gradient Trajectory Tracking

In this paper we assume that the initial decision variable $\boldsymbol{x}_0$ is not necessarily the optimal solution of the initial objective function $f(\boldsymbol{x};t_0)$, i.e., $\boldsymbol{x}_0 \neq \boldsymbol{x}^*(t_0)$. We model this assumption by defining a residual error for the gradient of the initial variable $\nabla_{\boldsymbol{x}} f(\boldsymbol{x}_0;t_0) = \boldsymbol{r}(0)$. To improve the estimation for the decision variable $\boldsymbol{x}$, we set up a prediction-correction scheme motivated by the Kalman filter strategy in estimation theory [31] and by continuation methods in numerical analysis [32]. In the first step, we predict how the solution changes, and in the correction step we use descent methods to push the predicted variable towards the optimizer at that time instance[1].

To generate the prediction step, we reformulate the time-varying problem (1) in terms of its optimality conditions. Minimizing the objective in (1) is equivalent to computing the solution of the following nonlinear system of equations

$$\nabla_{\boldsymbol{x}} f\left(\boldsymbol{x}^*\left(t\right);t\right) = \boldsymbol{0}, \tag{3}$$

for each t. These two problems are equivalent since the objective functions $f(\boldsymbol{x};t)$ are strongly convex with respect to $\boldsymbol{x}$ and only their optimal solutions satisfy the condition in (3).

Consider an arbitrary vector $\boldsymbol{x} \in \mathbb{R}^n$ which may be interpreted as the state of a dynamical system. The objective function gradient $\nabla_{\boldsymbol{x}} f(\boldsymbol{x};t) \in \mathbb{R}^n$ computed at point $\boldsymbol{x}$ is

$$\nabla_{\boldsymbol{x}} f\left(\boldsymbol{x};t\right) = \boldsymbol{r}\left(t\right), \tag{4}$$

where $\boldsymbol{r}(t) \in \mathbb{R}^n$ is the residual error. The aim of the prediction step is to keep the residual error as constant as possible while the optimization problem is changing. To say it in another way, we want to predict how to update $\boldsymbol{x}_k$ such that we stay close to the iso-residual manifold. We try to keep the evolution of the trajectory close to the residual vector $\boldsymbol{r}(t)$ which is equivalent to

$$\begin{aligned}&\nabla_{\boldsymbol{x}} f\left(\boldsymbol{x}+\delta\boldsymbol{x};t+\delta t\right) \approx \nabla_{\boldsymbol{x}} f\left(\boldsymbol{x};t\right)\\ &+\nabla_{\boldsymbol{xx}} f\left(\boldsymbol{x};t\right)\delta\boldsymbol{x} + \nabla_{t\boldsymbol{x}} f\left(\boldsymbol{x};t\right)\delta t = \boldsymbol{r}\left(t\right),\end{aligned} \tag{5}$$

where $\delta\boldsymbol{x} \in \mathbb{R}^n$ and the positive scalar δt are the variations of the decision variable $\boldsymbol{x}$ and the time variable t, respectively. By subtracting (4) from (5) and dividing the resulting equation by the time variation δt, we obtain the continuous dynamical system

$$\dot{\boldsymbol{x}} = -\left[\nabla_{\boldsymbol{xx}} f\left(\boldsymbol{x};t\right)\right]^{-1}\nabla_{t\boldsymbol{x}} f\left(\boldsymbol{x};t\right), \tag{6}$$

where $\dot{\boldsymbol{x}} := \delta\boldsymbol{x}/\delta t$. We then consider the discrete time approximation of (6), which amounts to sampling the problem at times t_k, for $k = 0, 1, 2, \ldots$. The prediction step consists of a discrete-time approximation of integrating (6) by using an Euler scheme.

[1]This correction strategy has been called differently by different authors: an alternative term is *adaptation*, as reported in [33], [34].

Algorithm 1: Gradient trajectory tracking (GTT).

Require: Initial variable $\boldsymbol{x}_0$. Initial objective function $f(\boldsymbol{x}; t_0)$, no. of correction steps τ

1: **for** $k = 0, 1, 2, \ldots$ **do**
2: Predict the solution using the prior information [cf. (7)]

$$\boldsymbol{x}_{k+1|k} = \boldsymbol{x}_k - h\left[\nabla_{\boldsymbol{xx}} f\left(\boldsymbol{x}_k; t_k\right)\right]^{-1} \nabla_{t\boldsymbol{x}} f\left(\boldsymbol{x}_k; t_k\right)$$

3: Acquire the updated function $f(\boldsymbol{x}; t_{k+1})$
4: Initialize the sequence of corrected variables $\widehat{\boldsymbol{x}}_{k+1}^0 = \boldsymbol{x}_{k+1|k}$
5: **for** $s = 0 : \tau - 1$ **do**
6: Correct the variable by the gradient step [cf. (8)]

$$\widehat{\boldsymbol{x}}_{k+1}^{s+1} = \widehat{\boldsymbol{x}}_{k+1}^{s} - \gamma \nabla_{\boldsymbol{x}} f\left(\widehat{\boldsymbol{x}}_{k+1}^{s}; t_{k+1}\right)$$

7: **end for**
8: Set the corrected variable $\boldsymbol{x}_{k+1} = \widehat{\boldsymbol{x}}_{k+1}^{\tau}$
9: **end for**

Let $\boldsymbol{x}_{k+1|k} \in \mathbb{R}^n$ be the predicted decision variable based on the available information up to time t_k, then we may write the Euler integral approximation of (6) as

$$\boldsymbol{x}_{k+1|k} = \boldsymbol{x}_k - h\left[\nabla_{\boldsymbol{xx}} f\left(\boldsymbol{x}_k; t_k\right)\right]^{-1} \nabla_{t\boldsymbol{x}} f\left(\boldsymbol{x}_k; t_k\right). \quad (7)$$

Observe that the prediction step in (7) is computed by only incorporating information available at time t_k; however, the decision variable $\boldsymbol{x}_{k+1|k}$ is supposed to be close to the iso-residual manifold of the objective function at time t_{k+1}.

The gradient trajectory tracking (GTT) algorithm uses the gradient descent method to correct the predicted decision variable $\boldsymbol{x}_{k+1|k}$. This procedure modifies the predicted variable $\boldsymbol{x}_{k+1|k}$ towards the optimal argument of the objective function at time t_{k+1}. Therefore, the correction (or adaptation) step of GTT requires execution of the gradient descent method based on the updated objective function $f(\boldsymbol{x}; t_{k+1})$. Depending on the sampling period h, we can afford a specific number of gradient descent steps until sampling the next function.

Define τ as the number of gradient descent steps used for correcting the predicted decision variable $\boldsymbol{x}_{k+1|k}$. Further, define $\widehat{\boldsymbol{x}}_{k+1}^s \in \mathbb{R}^n$ as the corrected decision variable after executing s steps of the gradient descent method. Therefore, the sequence of variables $\widehat{\boldsymbol{x}}_{k+1}^s$ is initialized by $\widehat{\boldsymbol{x}}_{k+1}^0 = \boldsymbol{x}_{k+1|k}$ and updated by the recursion

$$\widehat{\boldsymbol{x}}_{k+1}^{s+1} = \widehat{\boldsymbol{x}}_{k+1}^{s} - \gamma \nabla_{\boldsymbol{x}} f\left(\widehat{\boldsymbol{x}}_{k+1}^{s}; t_{k+1}\right), \quad (8)$$

where $\gamma > 0$ is the stepsize. The output of the recursive update (8) after τ steps is the decision variable of the GTT algorithm at time t_{k+1}, i.e., $\boldsymbol{x}(t_{k+1}) := \boldsymbol{x}_{k+1} = \widehat{\boldsymbol{x}}_{k+1}^{\tau}$.

We summarize the GTT scheme in Algorithm 1. Observe that Step 2 and Step 6 implement the prediction-correction scheme. In Step 2, we compute a first-order approximation of the gradient $\nabla_{\boldsymbol{x}} f(\boldsymbol{x}; t)$ at time t_k [cf. (7)]. Then we correct the predicted solution by executing τ gradient descent steps as stated in (8) for the updated objective function $f(\boldsymbol{x}; t_{k+1})$ in Steps 5–7. The sequence of corrected variables is initialized by the predicted solution $\widehat{\boldsymbol{x}}_{k+1}^0 = \boldsymbol{x}_{k+1|k}$ in Step 4 and the output of the recursion is considered as the updated variable $\boldsymbol{x}_{k+1} = \widehat{\boldsymbol{x}}_{k+1}^{\tau}$ in Step 8. The implementation of gradient descent for the correction process requires access to the updated function $f(\boldsymbol{x}; t_{k+1})$ which is sampled in Step 3.

Note that the GTT correction step is done by executing τ gradient descent steps which only uses first-order information of the objective function f. We accelerate this procedure using second-order information in the following subsection.

B. Newton Trajectory Tracking

The GTT prediction step introduced in (7) requires computation of the partial Hessian inverse $[\nabla_{\boldsymbol{xx}} f(\boldsymbol{x}_k; t_k)]^{-1}$. Note that the computational complexity of the Hessian inverse is of order $O(n^3)$, which is affordable when n is of moderate size or a certain level of latency associated with this inverse computation will not degrade performance. These two observations justify using the Newton method for the correction (or adaptation) step as well, which requires computation of the partial Hessian inverse of the objective function. Therefore, we introduce the Newton trajectory tracking (NTT) method as an algorithm that uses second-order information for both the prediction and correction steps.

The prediction step of the NTT algorithm is identical to the prediction step of the GTT method as introduced in (7); however, in the correction steps NTT updates the predicted solution trajectory by applying τ steps of the Newton method. In particular, the predicted variable $\boldsymbol{x}_{k+1|k}$ in (7) is used for initializing the sequence of corrected variables $\widehat{\boldsymbol{x}}_{k+1}^s$, i.e., $\widehat{\boldsymbol{x}}_{k+1}^0 := \boldsymbol{x}_{k+1|k}$. The sequence of corrected variables $\widehat{\boldsymbol{x}}_{k+1}^s$ is updated using Newton steps as

$$\widehat{\boldsymbol{x}}_{k+1}^{s+1} = \widehat{\boldsymbol{x}}_{k+1}^{s} - \nabla_{\boldsymbol{xx}} f\left(\widehat{\boldsymbol{x}}_{k+1}^{s}; t_{k+1}\right)^{-1} \nabla_{\boldsymbol{x}} f\left(\widehat{\boldsymbol{x}}_{k+1}^{s}; t_{k+1}\right). \quad (9)$$

The decision variable (solution) at step t_{k+1} for the NTT algorithm $\boldsymbol{x}(t_{k+1}) := \boldsymbol{x}_{k+1}$ is the outcome of τ iterations of (9) such that $\boldsymbol{x}_{k+1} = \widehat{\boldsymbol{x}}_{k+1}^{\tau}$.

Observe that the computational time of the Newton step and the gradient descent step are different. The complexity of the Newton step is in the order of $O(n^3)$, while the gradient descent step requires a computational complexity of order $O(n)$. Since the sampling period is a fixed value, the number of Newton iterations in one iteration of the NTT algorithm is smaller than the number of gradient descent steps that we can afford in the correction step of GTT. On the other hand, the Newton method requires less iterations relative to the gradient descent method to achieve a comparable accuracy. In particular, for an optimization problem with a large condition number the difference between the convergence speeds of these algorithms is substantial, in which case NTT is preferable to GTT.

In developing the prediction steps of the GTT and NTT algorithms we assumed that the mixed partial derivative $\nabla_{t\boldsymbol{x}} f(\boldsymbol{x}; t)$ is available; however, frequently in applications the variation of the objective function over time is not known. This motivates the idea of approximating the objective function variation which we study in the following subsection.

C. Time Derivative Approximation

Consider the mixed partial derivative at time t_k using the gradient of the objective with respect to $\boldsymbol{x}$ at times t_k and t_{k-1},

Algorithm 2: Newton trajectory tracking (NTT).

Require: Initial variable $\boldsymbol{x}_0$. Initial objective function $f(\boldsymbol{x};t_0)$, no. of correction steps τ

1: **for** $k=0,1,2,\ldots$ **do**
2: Predict the solution using the prior information [cf. (7)]

$$\boldsymbol{x}_{k+1|k}=\boldsymbol{x}_k-h\left[\nabla_{\boldsymbol{xx}}f\left(\boldsymbol{x}_k;t_k\right)\right]^{-1}\nabla_{t\boldsymbol{x}}f\left(\boldsymbol{x}_k;t_k\right)$$

3: Acquire the updated function $f(\boldsymbol{x};t_{k+1})$
4: Initialize the sequence of corrected variables $\widehat{\boldsymbol{x}}^0_{k+1}=\boldsymbol{x}_{k+1|k}$
5: **for** $s=0:\tau-1$ **do**
6: Correct the variable by the gradient step [cf. (8)]

$$\widehat{\boldsymbol{x}}^{s+1}_{k+1}=\widehat{\boldsymbol{x}}^{s}_{k+1}-\nabla_{\boldsymbol{xx}}f\big(\widehat{\boldsymbol{x}}^{s}_{k+1};t_{k+1}\big)^{-1}\nabla_{\boldsymbol{x}}f\left(\widehat{\boldsymbol{x}}^{s}_{k+1};t_{k+1}\right)$$

7: **end for**
8: Set the corrected variable $\boldsymbol{x}_{k+1}=\widehat{\boldsymbol{x}}^{\tau}_{k+1}$
9: **end for**

Algorithm 3: Approximate gradient tracking (AGT).

Require: Initial variable $\boldsymbol{x}_0$. Initial objective function $f(\boldsymbol{x};t_0)$, no. of correction steps τ

1: **for** $k=0,1,2,\ldots$ **do**
2: Predict the solution using the prior information [cf. (7)–(10)]

$$\boldsymbol{x}_{k+1|k}=\boldsymbol{x}_k-\left[\nabla_{\boldsymbol{xx}}f\left(\boldsymbol{x}_k;t_k\right)\right]^{-1}\tilde{\nabla}_{t\boldsymbol{x}}f\left(\boldsymbol{x}_k;t_k\right)\,h$$

3: Acquire the updated function $f(\boldsymbol{x};t_{k+1})$
4: Initialize the sequence of corrected variables $\widehat{\boldsymbol{x}}^0_{k+1}=\boldsymbol{x}_{k+1|k}$
5: **for** $s=0:\tau-1$ **do**
6: Correct the variable by the gradient step [cf. (8)]

$$\widehat{\boldsymbol{x}}^{s+1}_{k+1}=\widehat{\boldsymbol{x}}^{s}_{k+1}-\gamma\nabla_{\boldsymbol{x}}f\left(\widehat{\boldsymbol{x}}^{s}_{k+1};t_{k+1}\right)$$

7: **end for**
8: Set the corrected variable $\boldsymbol{x}_{k+1}=\widehat{\boldsymbol{x}}^{\tau}_{k+1}$
9: **end for**

that is, the approximate partial mixed gradient $\tilde{\nabla}_{t\boldsymbol{x}}f_k$ as

$$\tilde{\nabla}_{t\boldsymbol{x}}f\left(\boldsymbol{x}_k;t_k\right)=\frac{1}{h}\left(\nabla_{\boldsymbol{x}}f\left(\boldsymbol{x}_k;t_k\right)-\nabla_{\boldsymbol{x}}f\left(\boldsymbol{x}_k;t_{k-1}\right)\right). \quad (10)$$

which is called a *first-order backward finite difference* since it requires information of the first previous step for approximating the current mixed partial derivative. The error of this approximation is bounded on the order of $O(h)$ [35], which may be improved by using the gradients and mixed partial derivative $\tilde{\nabla}_{t\boldsymbol{x}}f(\boldsymbol{x}_k;t_k)$ of more than one previous step, if needed[2].

Substituting the partial mixed gradient $\nabla_{t\boldsymbol{x}}f(\boldsymbol{x}_k;t_k)$ in (7) by its approximation $\tilde{\nabla}_{t\boldsymbol{x}}f(\boldsymbol{x}_k;t_k)$ in (10) leads to the *approximate prediction step*

$$\boldsymbol{x}_{k+1|k}=\boldsymbol{x}_k-h\left[\nabla_{\boldsymbol{xx}}f\left(\boldsymbol{x}_k;t_k\right)\right]^{-1}\tilde{\nabla}_{t\boldsymbol{x}}f\left(\boldsymbol{x}_k;t_k\right). \quad (11)$$

The predicted variable $\boldsymbol{x}_{k+1|k}$ is an initial estimate for the optimal solution of the objective function $f(\boldsymbol{x};t_{k+1})$. This estimation can be corrected by descending through the optimal argument of the objective function $f(\boldsymbol{x};t_{k+1})$. To do so, one may either use a gradient algorithm as in (8) or Newton steps as in (9). Based on this idea, we introduce the approximate gradient tracking (AGT) algorithm which is different from GTT in using the approximate prediction step in (11) instead of the exact update in (7). Likewise, we introduce the approximate Newton tracking (ANT) method as a variation of the NTT algorithm. We summarize the AGT and ANT methods which make use of this approximation scheme in Algorithms 3and 4, respectively. As we can observe, the main difference with Algorithms 1and 2 is in Step 2, where we use the approximate time derivative. In Section III we establish that this time derivative approximation does not degrade significantly the performance of the algorithms presented here.

III. Convergence Analysis

We turn to establishing that the prediction-correction schemes derived in Section II solve the continuous-time problem stated in (1) up to an error term which is dependent on the discrete-time sampling period. In order to do so, some technical conditions are required which we state below.

Assumption 1: The function $f(\boldsymbol{x};t)$ is twice differentiable and m-strongly convex in $\boldsymbol{x}\in\mathbb{R}^n$ and uniformly in t, that is, the Hessian of $f(\boldsymbol{x};t)$ with respect to $\boldsymbol{x}$ is bounded below by m for each $\boldsymbol{x}\in\mathbb{R}^n$ and uniformly in t,

$$\nabla_{\boldsymbol{xx}}f\left(\boldsymbol{x};t\right)\succeq m\mathbf{I},\quad \forall\boldsymbol{x}\in\mathbb{R}^n,t.$$

Assumption 2: The function $f(\boldsymbol{x};t)$ is sufficiently smooth both in $\boldsymbol{x}\in\mathbb{R}^n$ and in t, and in particular, $f(\boldsymbol{x};t)$ has bounded second and third order derivatives with respect to $\boldsymbol{x}\in\mathbb{R}^n$ and t as

$$\|\nabla_{\boldsymbol{xx}}f\left(\boldsymbol{x};t\right)\|\leqslant L,\ \|\nabla_{t\boldsymbol{x}}f\left(\boldsymbol{x};t\right)\|\leqslant C_0,\ \|\nabla_{\boldsymbol{xxx}}f(x;t)\|\leqslant C_1,$$
$$\|\nabla_{\boldsymbol{x}t\boldsymbol{x}}f\left(\boldsymbol{x};t\right)\|\leqslant C_2,\quad \|\nabla_{tt\boldsymbol{x}}f\left(\boldsymbol{x};t\right)\|\leqslant C_3.$$

Assumption 1, besides guaranteeing that problem (1) is strongly convex and has a *unique* solution for each time instance, is needed to ensure that the Hessian of the objective function $f(\boldsymbol{x};t)$ is invertible. The fact that the solution is unique for each time instance, implies that the solution trajectory is unique. This mathematical setting frequently appears in the analysis of optimization tools in time-varying settings, and is essential to establishing trajectory tracking results—see, for instance [6], [12], [24], [37]. Assumption 2 ensures that the Hessian is bounded from above, a property which is equivalent to the Lipschitz continuity of the gradient, and that the third derivative tensor $\nabla_{\boldsymbol{xxx}}f(x;t)$ is also bounded above (typically required for the analysis of Newton-type algorithms), as well as boundedness of the time variations of gradient and Hessian. These last properties ensure the possibility to build a prediction scheme based on the (estimated) knowledge of how the function and its derivatives change in time. A similar assumption was required (albeit only locally) for the local convergence analysis in [12, (3.2)].

Assumptions 1 and 2 are sufficient to show that the solution *mapping* $t\mapsto\boldsymbol{x}^*(t)$ is single-valued and locally Lipschitz

[2]Approximation errors of the order of $O(h^2)$, $O(h^3)$, and $O(h^4)$ can be achieved, e.g., by the recursive method presented in [36].

Algorithm 4: Approximate Newton tracking (ANT).

Require: Initial variable $\boldsymbol{x}_0$. Initial objective function $f(\boldsymbol{x}; t_0)$, no. of correction steps τ

1: **for** $k = 0, 1, 2, \ldots$ **do**
2: Predict the solution using the prior information [cf. (7)–(10)]

$$\boldsymbol{x}_{k+1|k} = \boldsymbol{x}_k - \left[\nabla_{\boldsymbol{xx}} f\left(\boldsymbol{x}_k; t_k\right)\right]^{-1} \tilde{\nabla}_{t\boldsymbol{x}} f\left(\boldsymbol{x}_k; t_k\right) h$$

3: Acquire the updated function $f(\boldsymbol{x}; t_{k+1})$
4: Initialize the sequence of corrected variables $\widehat{\boldsymbol{x}}^0_{k+1} = \boldsymbol{x}_{k+1|k}$
5: **for** $s = 0 : \tau - 1$ **do**
6: Correct the variable by the gradient step [cf. (8)]

$$\widehat{\boldsymbol{x}}^{s+1}_{k+1} = \widehat{\boldsymbol{x}}^{s}_{k+1} - \nabla_{\boldsymbol{xx}} f\left(\widehat{\boldsymbol{x}}^{s}_{k+1}; t_{k+1}\right)^{-1} \nabla_{\boldsymbol{x}} f\left(\widehat{\boldsymbol{x}}^{s}_{k+1}; t_{k+1}\right)$$

7: **end for**
8: Set the corrected variable $\boldsymbol{x}_{k+1} = \widehat{\boldsymbol{x}}^{\tau}_{k+1}$
9: **end for**

continuous in t, and in particular,

$$\|\boldsymbol{x}^*\left(t_{k+1}\right) - \boldsymbol{x}^*\left(t_k\right)\| \leqslant \frac{1}{m} \|\nabla_{t\boldsymbol{x}} f\left(\boldsymbol{x}; t\right)\| \left(t_{k+1} - t_k\right) \leqslant \frac{C_0 h}{m}, \tag{12}$$

see for example [26, Theorem 2F.10]. This gives us a link between the sampling period h and the allowed variations in the optimizers. This property also allows our algorithms to converge to a neighborhood of the optimal solution. We remark that, in most of the current literature the condition in (12) is taken as an assumption (that is, one assumes that the optimizer does not change more than a certain upper bound in time), while here is a consequence of our smoothness and boundedness assumptions.

Remark 1: Assumptions 1 and 2 can be weakened if a priori knowledge of the domain of the optimizers and the sequence generated by the algorithms is given by the structure of the problem, i.e., the optimal trajectory is contained within a subset X of $\mathbb{R}^n$. In this case, we can concentrate on functions that verify Assumptions 1 and 2 only for $\boldsymbol{x} \in X \subset \mathbb{R}^n$. We explore this scenario in the second numerical example. An alternative setting in which Assumptions 1 and 2 need not hold is if we restrict (project) the algorithms to a neighborhood of the optimal trajectory. In this latter case, the convergence analysis becomes local only.

We start the convergence analysis by deriving an upper bound on the norm of the approximation error $\boldsymbol{\Delta}_k \in \mathbb{R}^n$ of the first-order forward Euler integral in (7) (w.r.t. the continuous dynamics (6)). This error is sometimes referred to as the local truncation error [35]. The error is defined as the difference between the predicted $\boldsymbol{x}_{k+1|k}$ in (7) and the exact prediction $\boldsymbol{x}(t_{k+1})$ obtained by integrating the continuous dynamics (6) from the same initial condition $\boldsymbol{x}_k$, i.e.,

$$\boldsymbol{\Delta}_k := \boldsymbol{x}_{k+1|k} - \boldsymbol{x}\left(t_{k+1}\right). \tag{13}$$

The upper bound for the norm $\| \boldsymbol{\Delta}_k \|$ is central in all our algorithms, since it encodes the error coming from the prediction step. We study this upper bound in the following proposition.

Proposition 1: Under Assumptions 1–2, the error norm $\| \boldsymbol{\Delta}_k \|$ of the Euler approximation (7) defined in (13) is upper bounded by

$$\| \boldsymbol{\Delta}_k \| \leqslant \frac{h^2}{2} \left[\frac{C_0^2 C_1}{m^3} + \frac{2 C_0 C_2}{m^2} + \frac{C_3}{m}\right] = O\left(h^2\right). \tag{14}$$

Proof: See Appendix A. ■

Proposition 1 states that the norm of the discretization error $\| \boldsymbol{\Delta}_k \|$ is bounded above by a constant which is in the order of $O(h^2)$. We use this upper bound in proving convergence of all the proposed methods.

A. Gradient Trajectory Tracking Convergence

We study the convergence properties of the sequence of variables $\boldsymbol{x}_k$ generated by GTT for different choices of the stepsize. In the following theorem we show that the optimality gap $\|\boldsymbol{x}_k - \boldsymbol{x}^*(t_k)\|$ converges exponentially to an error bound.

Theorem 1: Consider the gradient trajectory tracking algorithm as defined in (3)–(8). Let Assumptions 1–2 hold true and define the constants ρ and σ as

$$\rho := \max\{|1 - \gamma m|, |1 - \gamma L|\},\ \sigma := 1 + h(C_0 C_1/m^2 + C_2/m). \tag{15}$$

Let the stepsize γ be chosen as $0 < \gamma < 2/L$, which implies $\rho < 1$.

i) For any sampling period h, the sequence $\{\boldsymbol{x}_k\}$ converges to $\boldsymbol{x}^*(t_k)$ exponentially up to a bounded error as

$$\|\boldsymbol{x}_k - \boldsymbol{x}^*\left(t_k\right)\| \leqslant \rho^{\tau k} \|\boldsymbol{x}_0 - \boldsymbol{x}^*\left(t_0\right)\| + \rho^{\tau} \left[h\left[\frac{2C_0}{m}\right] + \frac{h^2}{2}\left[\frac{C_0^2 C_1}{m^3} + \frac{2C_0 C_2}{m^2} + \frac{C_3}{m}\right]\right]\left[\frac{1 - \rho^{\tau k}}{1 - \rho^{\tau}}\right]. \tag{16}$$

ii) If the sampling period h is chosen such that $\rho^{\tau}\sigma < 1$, i.e.,

$$h < \left[\frac{C_0 C_1}{m^2} + \frac{C_2}{m}\right]^{-1} \left(\rho^{-\tau} - 1\right), \tag{17}$$

then the sequence $\{\boldsymbol{x}_k\}$ converges to $\boldsymbol{x}^*(t_k)$ exponentially up to a bounded error as

$$\|\boldsymbol{x}_k - \boldsymbol{x}^*\left(t_k\right)\| \leqslant \left(\rho^{\tau}\sigma\right)^k \|\boldsymbol{x}_0 - \boldsymbol{x}^*\left(t_0\right)\| + \rho^{\tau} \frac{h^2}{2}\left[\frac{C_0^2 C_1}{m^3} + \frac{2C_0 C_2}{m^2} + \frac{C_3}{m}\right]\left[\frac{1 - \left(\rho^{\tau}\sigma\right)^k}{1 - \rho^{\tau}\sigma}\right]. \tag{18}$$

Proof: See Appendix B. ■

Theorem 1 states the convergence properties of the GTT algorithm for different choices of the parameters. In both cases the exponential convergence to a neighborhood is shown, however, the accuracy of convergence depends on the choice of the sampling period h, the stepsize parameter γ, and the number of gradient descent steps τ. To guarantee that the constant ρ is strictly smaller than 1, the stepsize must satisfy $\gamma < 2/L$: this can be seen by the definition of ρ and the fact that $m \leqslant L$ by Assumptions 1 and 2. Then, for any choice of the sampling period h the result in (16) holds, which implies exponential convergence to a neighborhood of the optimal solution. In this case the error bound contains two terms that are proportional to h and h^2. Therefore, we can say that the accuracy of convergence is in the order of $O(h)$. Notice that increasing the number of gradient

descent iterations τ improves the speed of exponential convergence by decreasing the factor ρ^τ. Moreover, a larger choice of τ leads to a better accuracy since the asymptotic error bound is proportional to $\rho^\tau/(1-\rho^\tau)$.

The result in (18) shows that the accuracy of convergence is proportional to the square of the sampling period h, if the sampling period is chosen to satisfy the condition $\rho^\tau\sigma<1$. In the following corollary we formalize this observation by studying the asymptotic convergence results of GTT for different choices of stepsize.

Corollary 1: Under the same conditions of Theorem 1, the sequence of variables $\{\boldsymbol{x}_k\}$ generated by GTT converges to a neighborhood of $\boldsymbol{x}^*(t_k)$ asymptotically. The error bound when the parameters ρ and σ in (15) are chosen as $\rho^\tau\sigma\geqslant 1, \rho^\tau<1$ is

$$\limsup_{k\to\infty}\|\boldsymbol{x}(t_k)-\boldsymbol{x}^*(t_k)\|\leqslant\frac{2C_0\rho^\tau h}{m(1-\rho^\tau)}=O(h), \quad (19)$$

and if they satisfy $\rho^\tau\sigma<1$ the error bound is

$$\limsup_{k\to\infty}\|\boldsymbol{x}(t_k)-\boldsymbol{x}^*(t_k)\| \leqslant\frac{\rho^\tau h^2}{2(1-\rho^\tau\sigma)}\left(\frac{C_0^2C_1}{m^3}+\frac{2C_0C_2}{m^2}+\frac{C_3}{m}\right)=O\left(h^2\right). \quad (20)$$

The asymptotic results in Corollary 1 are implied by considering the results in Theorem 1 when $k\to\infty$. Notice that when the stepsize satisfies conditions $\rho^\tau\sigma\geqslant 1, \rho^\tau<1$ the convergence accuracy of GTT is in the order of $O(h)$. Moreover, if the sampling period h is chosen such that $\rho^\tau\sigma<1$ then the error bound is in the order of $O(h^2)$.

B. Newton Trajectory Tracking Convergence

Notice that the GTT algorithm does not incorporate the second-order information of the objective function $f(\boldsymbol{x};t_{k+1})$ to correct the predicted variable $\boldsymbol{x}_{k+1|k}$, while the NTT algorithm uses Newton's method in the correction step. Similar to the advantages of Newton's method relative to the gradient descent algorithm, we expect to observe faster convergence and more accurate estimation for NTT relative to GTT. In particular, one would expect that if Newton's method is in its quadratic phase, the error should be at least in the order of $O(h^4)$. In the following theorem we show that when both the initial estimate $\boldsymbol{x}_0$ is close enough to the initial solution $\boldsymbol{x}^*(t_0)$ and the sampling period h is chosen properly, then NTT yields a more accurate convergence relative to GTT.

Theorem 2: Consider the NTT algorithm generated by (7) and (9). Assume that all the conditions in Assumptions 1–2 hold. Define constants δ_1,δ_2 and Q as

$$\delta_1:=\frac{C_0C_1}{m^2}+\frac{C_2}{m},\ \delta_2:=\frac{C_0^2C_1}{2m^3}+\frac{C_0C_2}{m^2}+\frac{C_3}{2m}, Q:=\frac{2m}{C_1}. \quad (21)$$

Further, recall τ as the number of Newton steps in the correction step. For any constant $c>0$, if the sampling period h satisfies

$$h\leqslant\min\left\{1,\left[\frac{Q^{2\tau-1}c}{((1+\delta_1)c+\delta_2)^{2\tau}}\right]^{\frac{1}{4\tau-2}}\right\}, \quad (22)$$

and the initial error $\|\boldsymbol{x}_0-\boldsymbol{x}^*(t_0)\|$ satisfies the condition

$$\|\boldsymbol{x}_0-\boldsymbol{x}^*(t_0)\|\leqslant ch^2, \quad (23)$$

then the sequence $\|\boldsymbol{x}_k-\boldsymbol{x}^*(t_k)\|$ generated by NTT for $k\geqslant 1$ is bounded above as

$$\|\boldsymbol{x}_k-\boldsymbol{x}^*(t_k)\|\leqslant Q^{-(2\tau-1)}(\sigma c+\delta_2)^{2\tau}h^{4\tau}. \quad (24)$$

Proof: See Appendix C. ■

Theorem 2 establishes that, under additional conditions, the NTT tracks the optimal trajectory $\boldsymbol{x}^*(t_k)$ up to an error bound not larger than

$$Q^{-(2\tau-1)}(\sigma c+\delta_2)^{2\tau}h^{4\tau}=O\left(h^{4\tau}\right), \quad (25)$$

where h is the sampling period. This is a result of the quadratic convergence of Newton's method.

The conditions can be intuitively explained as follows. Condition (23) formalizes the local nature of the convergence analysis of Theorem 2: due to the dependence of (22) on c, the right-hand side of (23) is in fact upper bounded. For example, when $c\to\infty$, then $h\to 0$ and $ch^2\to Q/(1+\delta_1)$. We notice that the initial gap is proportional to h^2, since the integration error $\|\boldsymbol{\Delta}\|$ has the same dependence on h. Finally, (22) derives an upper bound on the allowable sampling period. It comprises of two terms, the first coming from the need for a local analysis, the second from convergence arguments. Despite the fact that Theorem 2 is a local convergence result, in the numerical simulations we will display how NTT behaves very well even in a global sense, and for $\tau=1$ achieves the proven $O(h^4)$ error bound.

Remark 2 (Quadratic Functions and Backtracking): Conditions (23) is a locality requirement, which is rather typical in for the analysis of Newton methods. The closer the function $f(\boldsymbol{x};t)$ is to be quadratic, the smaller the parameter C_1 is. When the function is quadratic, then $C_1=0$, which in turns means $Q,ch^2\to\infty$, i.e., global convergence is achieved (as expected). When C_1 becomes important, then one can think of initializing the Newton method with a backtracking strategy (as done often in practice), see [18].

Remark 3 (Hybrid Strategy): Theorem 2 suggests also a warm start procedure to implement the NTT algorithm. In particular, consider the condition $\|\boldsymbol{x}_0-\boldsymbol{x}^*(t_0)\|\leqslant ch^2$. Given the strong convexity assumption and the fact that the gradient vanishes at optimality, this condition is implied by the following sufficient condition

$$\|\nabla_{\boldsymbol{x}}f(\boldsymbol{x}_0;t_0)\|\leqslant m\,ch^2, \quad (26)$$

which is easier to check in practice than condition (23) (since normally one does not have access to the optimizer $\boldsymbol{x}^*(t_0)$). In fact, one might implement a hybrid strategy, where at the beginning we run the GTT algorithm and then we switch to NTT when the condition in (26) is satisfied. In order to make sure that the GTT algorithm eventually arrives at an error $\|\boldsymbol{x}_k-\boldsymbol{x}^*(t_k)\|\leqslant ch^2$, we need to pick c in a way that ch^2 is strictly bigger than the asymptotical error of GTT in (20). Therefore, we must choose c as

$$c>\frac{\rho^\tau\delta_2}{1-\rho^\tau\sigma}. \quad (27)$$

Hence, start with GTT and choose a sampling period h that verifies (22) and switch to NTT when condition (26) is satisfied. We will see how this strategy performs in the simulation results.

C. Convergence of Methods with Approximated Time Derivative

We focus now on the approximated version of GTT and NTT (i.e., the AGT and ANT algorithms), where we approximate the time derivative of the gradient. In the following theorems, we formalize the fact that this approximation does not affect the order of the asymptotic error w.r.t. h.

Theorem 3: Consider the AGT algorithm as defined in Algorithm 3, recall the definitions of the constants ρ and σ in (15), and let Assumptions 1–2 hold true. Let the stepsize γ be chosen as $0 < \gamma < 2/L$, which implies $\rho < 1$.

i) For any sampling period h, the sequence $\{\boldsymbol{x}_k\}$ converges to $\boldsymbol{x}^*(t_k)$ exponentially up to a bounded error as

$$\|\boldsymbol{x}_k - \boldsymbol{x}^*(t_k)\| \leqslant \rho^{\tau k}\|\boldsymbol{x}_0 - \boldsymbol{x}^*(t_0)\| + \rho^{\tau}\left[h\left[\frac{2C_0}{m}\right] + \frac{h^2}{2}\left[\frac{C_0^2 C_1}{m^3} + \frac{2C_0C_2}{m^2} + \frac{2C_3}{m}\right]\right]\left[\frac{1-\rho^{\tau k}}{1-\rho^{\tau}}\right]. \quad (28)$$

ii) If the sampling period h is chosen such that $\rho^{\tau}\sigma < 1$, i.e.,

$$h < \left[\frac{C_0C_1}{m^2} + \frac{C_2}{m}\right]^{-1}\left(\rho^{-\tau} - 1\right), \quad (29)$$

then the sequence $\{\boldsymbol{x}_k\}$ converges to $\boldsymbol{x}^*(t_k)$ exponentially up to a bounded error as,

$$\|\boldsymbol{x}_k - \boldsymbol{x}^*(t_k)\| \leqslant (\rho^{\tau}\sigma)^k\|\boldsymbol{x}_0 - \boldsymbol{x}^*(t_0)\| + \rho^{\tau}\frac{h^2}{2} \times \left[\frac{C_0^2C_1}{m^3} + \frac{2C_0C_2}{m^2} + \frac{2C_3}{m}\right]\left[\frac{1-(\rho^{\tau}\sigma)^k}{1-\rho^{\tau}\sigma}\right]. \quad (30)$$

Proof: See Appendix D. ■

Theorem 3 states the convergence properties of the AGT algorithm for different choices of the parameters. In both cases the exponential convergence to a neighborhood is shown with convergence accuracy depending on the sampling period h, the stepsize γ, and the number of gradient descent steps τ. Moreover, for particular sampling period selections depending on smoothness properties of the objective, the asymptotic error bound either converges up to an $O(h)$ or $O(h^2)$ term. Notice that the convergence properties of AGT in (28) and (30) are identical to the convergence results of GTT in (16) and (18), respectively, except for the coefficients of h^2. To be more precise, the coefficient of h^2 in (28) and (30) is $C_0^2C_1/2m^3 + C_0C_2/m^2 + C_3/m$, while the coefficient of h^2 in (16) and (18) is $C_0^2C_1/2m^3 + C_0C_2/m^2 + C_3/2m$. This observation implies that the error bound of AGT is slightly larger than the error of GTT which is implied by the error of the derivative approximation. However, the orders of the error bounds for these two algorithms are identical.

AGT uses only first-order information of the objective $f(\boldsymbol{x}; t_{k+1})$ to correct the predicted variable $\boldsymbol{x}_{k+1|k}$, while ANT uses the Newton method in the correction step. Similar to the advantages of NTT relative to GTT, we show more accurate estimation for ANT relative to AGT in the following theorem.

Theorem 4: Consider the ANT algorithm as defined in Algorithm 4, recall the definitions of the constants δ_1, δ_2 and Q as in (21), and let Assumptions 1–2 hold true. Further, recall τ as the number of Newton steps in the correction step and define δ_2' as

$$\delta_2' := \delta_2 + \frac{C_3}{2m}. \quad (31)$$

For any constant $c > 0$, if the sampling period h satisfies

$$h \leqslant \min\left\{1, \left[\frac{Q^{2\tau-1}c}{\left((1+\delta_1)c + \delta_2'\right)^{2\tau}}\right]^{\frac{1}{4\tau-2}}\right\}, \quad (32)$$

and the initial error $\|\boldsymbol{x}_0 - \boldsymbol{x}^*(t_0)\|$ satisfies the condition

$$\|\boldsymbol{x}_0 - \boldsymbol{x}^*(t_0)\| \leqslant ch^2, \quad (33)$$

then the sequence $\|\boldsymbol{x}_k - \boldsymbol{x}^*(t_k)\|$ generated by ANT for $k \geqslant 1$ is bounded above as

$$\|\boldsymbol{x}_k - \boldsymbol{x}^*(t_k)\| \leqslant Q^{-(2\tau-1)}\left(\sigma c + \delta_2'\right)^{2\tau}h^{4\tau}. \quad (34)$$

Proof: See Appendix E. ■

Theorem 4 states that the ANT algorithm reaches an estimation error of order $O(h^{4\tau})$. Observe that the error bound in (34) for ANT is slightly worse than the bound in (24) for NTT, since $\delta_2' > \delta_2$. On the other hand, the bound for both algorithms is in the order of $O(h^{4\tau})$. According to the results in Theorems 3 and 4, we can approximate the time derivative simply by a first-order scheme without changing the functional dependence of the error in h, but increasing its magnitude. In the simulation results, we show that this increase in error is in fact extremely limited. These analytical results therefore suggest the advantage of the proposed prediction-correction algorithms even in cases in which the knowledge of the time variability of the objective function is only estimated, which is important in many practical scenarios, e.g., in robotics or in statistical signal processing.

IV. Numerical Experiments

In this section, we implement the algorithms derived in Section II for a couple practical examples in order to asses their performance in practice. Specifically, in Section IV.A, we consider a simple time-varying function and apply the GTT, NTT, AGT, and the hybrid method of Remark 3. Additionally, in Section IV.B, we consider the task of designing a derivative control law for an autonomous system to follow a reference path. In this practical setting, we only consider the case where the time-derivative of the objective is not available, and hence must be approximated. Here this approximation corresponds to not having perfect information regarding the reference path the system aims to track.

A. Scalar Example

As a simple example, consider the case where the decision variable $x \in \mathbb{R}$ is a scalar and the time-varying optimization problem is

$$\min_{x\in\mathbb{R}} f(x;t) := \frac{1}{2}(x - \cos(\omega t))^2 + \kappa\log\left[1 + \exp(\mu x)\right]. \quad (35)$$

The function in (35) represents, for instance, the goal of staying close to a periodically varying trajectory plus a logistic term that penalizes large values of x. The terms ω, κ, and μ are arbitrary nonnegative scalar parameters. In our experiments these parameters are set to $\omega = 0.02\,\pi$, $\kappa = 7.5$, and $\mu = 1.75$. The function

$f(x;t)$ satisfies all the conditions in Assumptions 1 and 2. In particular, one can compute in close-form the quantities

$$\nabla_{xx} f(x;t) = 1 + \kappa\mu^2 \frac{\exp(\mu x)}{[1+\exp(\mu x)]^2}, \tag{36a}$$

$$\nabla_{tx} f(x;t) = \omega \sin(\omega t), \tag{36b}$$

$$\nabla_{xxx} f(x;t) = \kappa\mu^3 \frac{\exp(\mu x)[1-\exp(\mu x)]}{[1+\exp(\mu x)]^3}, \tag{36c}$$

$$\nabla_{xtx} f(x;t) = 0. \tag{36d}$$

$$\nabla_{ttx} f(x;t) = \omega^2 \cos(\omega t), \tag{36e}$$

and the bounds

$$m = \min_{x\in\mathbb{R},t} \nabla_{xx} f(x;t) = 1, \tag{37a}$$

$$L = \max_{x\in\mathbb{R},t} \nabla_{xx} f(x;t) = 1 + \frac{\kappa\mu^2}{4} = 6.7422, \tag{37b}$$

$$C_0 = \max_{x\in\mathbb{R},t} \nabla_{tx} f(x;t) = \omega = 0.0628, \tag{37c}$$

$$C_1 = \max_{x\in\mathbb{R},t} \nabla_{xxx} f(x;t) = \kappa\mu^3 \frac{(2-\sqrt{3})(\sqrt{3}-1)}{[3+\sqrt{3}]^3} = 3.8678, \tag{37d}$$

$$C_2 = \max_{x\in\mathbb{R},t} \nabla_{xtx} f(x;t) = 0, \tag{37e}$$

$$C_3 = \max_{x\in\mathbb{R},t} \nabla_{ttx} f(x;t) = \omega^2 = 0.0039. \tag{37f}$$

We choose the constant stepsize as $\gamma = 0.2 < 2/L$ in the gradient method stated in (8) and initialize $x_0 = 0$ for all the algorithms. According to (17) the sampling period that guarantees an $O(h^2)$ error bound needs to be chosen as $h < 1.028$. for all $\tau \geqslant 1$.

In Fig. 1, we plot the error $\|x_k - x^*(t_k)\|$ versus the discrete time t_k for a sampling period of $h = 0.1$, for different schemes, along with the asymptotical bounds computed via Theorems 1 and 3. Observe that the running gradient (RG) method [24] which uses only a gradient correction step (and no prediction) performs the worst, achieving an error of 10^{-2}, while GTT for $\tau = 1, \tau = 3$, and $\tau = 5$ achieves an error of approximately 10^{-5}. Numerically we may conclude that tracking with gradient-based prediction (GTT) for different values of τ has a better error performance than running, even in the case we use an approximate time derivative (AGT); in addition, tracking with Newton-based prediction (NTT) with $\tau = 1$ achieves a superior performance compared to the others, i.e., an error stabilizing near 10^{-10} is achieved.

In Fig. 1, we also display the behavior of the hybrid strategy advocated in Remark 3. We can see how after we switch to NTT (when the condition $\|\nabla_{\boldsymbol{x}} f(\boldsymbol{x}_k; t_k)\| \leqslant 0.0034$, derived from (27), is met), then in only one step we regain the same performance as NTT.

The differences in performance can be also appreciated by varying h and observing the worst case error floor size which is defined as $\max_{k>\overline{k}}\{\|x_k - x^*(t_k)\|\}$, where $\overline{k} = 10^4$ in the simulations. Fig. 2 illustrates the error as a function of h. The performance differences between the proposed methods that may be observed here corroborate the differences evident in Fig. 1. In particular, the running method achieves the largest worst case error bound, followed in descending order by AGT, GTT with increasing τ, and lastly NTT (or equivalently the hybrid strategy), which achieves the minimal worst-case error bound. Notice also the dashed lines displaying the theoretical performance of $O(h), O(h^2)$, and $O(h^4)$, which are attained in this simulation.

We continue the simulation example by changing the parameters κ and μ in (35) to the values $\kappa = .1$ and $\mu = 0.5$. This brings $L = 1.0063$, and a condition number L/m close to 1. In this settings a first-order method, such as the gradient, is expected to perform better than in the case of high condition numbers (as in the previous example). We pick the stepsize $\gamma = 1 < 2/L$. In Figs. 3 and 4, we appreciate how the relative performances of GTT and NTT change with the new parameters[3].

B. Target Tracking Experiments

The second numerical example consists of a more realistic application scenario. We consider an autonomous system (i.e., a mobile robot) which is charged with the task of following an object whose position is varying continuously in time. Denote the reference trajectory of this object as a curve $\boldsymbol{y}(t)$, i.e., a function $\boldsymbol{y} : \mathbb{R}_+ \to \mathbb{R}^n$ and $\boldsymbol{x} \in \mathbb{R}^n$ be the decision variable of the robot, in terms of the waypoint it aims to reach next. We aim to solve tracking problems of the form

$$\min_{\boldsymbol{x}\in\mathbb{R}^2} f(\boldsymbol{x};t) := \frac{1}{2}\left(\|\boldsymbol{x} - \boldsymbol{y}(t)\|^2 + \mu_1 \exp\left(\mu_2 \|\boldsymbol{x} - \boldsymbol{b}\|^2\right)\right), \tag{38}$$

which corresponds to tracking the reference path $\boldsymbol{y}(t)$ while remaining close enough to a base station located in $\boldsymbol{b}$, which may correspond to a recharging station or a domain constraint associated with maintaining viable communications. Using the methods developed in Section II for problems of this type correspond to deriving derivative-based control laws for fully actuated systems with simple integrator dynamics.

For the example considered here, we consider a planar example ($n = 2$) and fix $\mu_1 = 1000$ m^2, $\mu_2 = .005$ m^{-2} with the base located at $\boldsymbol{b} = [100; 100]$ m. In addition, we suppose the target trajectory $\boldsymbol{y}(t)$ follows the specified path

$$\boldsymbol{y}(t) = 100\,[\cos(\omega t), \sin(3\omega t)]\ \text{m}$$

where $\omega = 0.01$ Hz. Moreover, the position domain is given as $X = [-150, 150] \times [-150, 150]$ m^2 and we know that $\boldsymbol{x}^*(t) \in X$. We can compute the constants of Assumptions 1 and 2 over $X \subset \mathbb{R}^n$ [Cfr. Remark 1] $m = 1.01, L = 3.45, C_0 = 3.16$ [m/s], $C_1 = 0.06$ [m^{-1}], $C_2 = 0, C_3 = 0.10$ [m/s^2]. We select stepsize $\gamma = 0.05 < 2/L$. With these parameters and $h = 1$s, the target moves with maximum speed of 3.16 m/s. This is comparable with the speed of current quad-rotors (max speed ~ 10 m/s).

In any practical setting, the actuation capability of an autonomous system is limited either in terms of velocity or degrees of freedom. We consider the case where the autonomous system may move with the same number of degrees as its decision variable dimension, i.e., it may move in any direction, yet its maximum velocity is limited to some value $v_{\max}$. A typical velocity maximum for ground vehicles is $v_{\max} = 4$ m/s, which is the choice made in the numerical experiments here. Thus, we

[3]The code of the simulation example will be made available for the readers, to appreciate how different stepsizes may influence the asymptotical bounds.

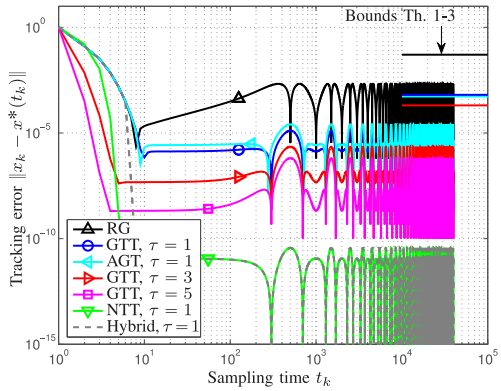

Fig. 1. Error with respect to the sampling time t_k for different algorithms applied to the scalar problem (35), with $h = 0.1, \kappa = 7.5, \mu = 1.5$.

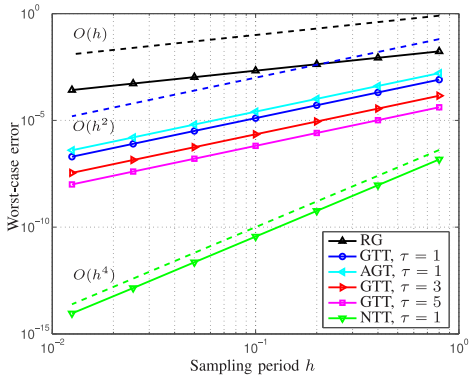

Fig. 2. Worst case error floor with respect to the sampling time interval h for different algorithms applied to the scalar problem (35), $\kappa = 7.5, \mu = 1.5$.

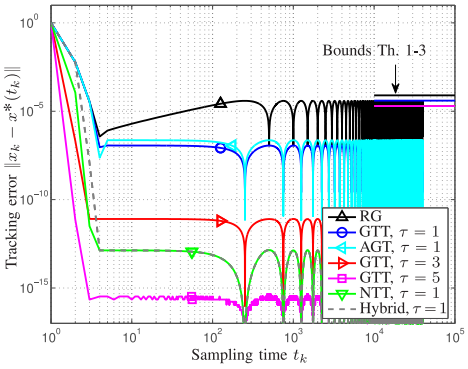

Fig. 3. Error with respect to the sampling time t_k for different algorithms applied to the scalar problem (35), with $h = 0.1, \kappa = .1, \mu = .5$.

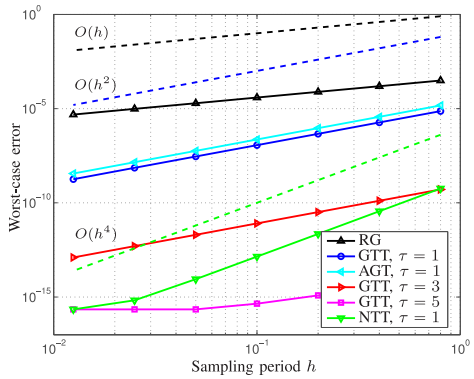

Fig. 4. Worst case error floor with respect to the sampling time interval h for different algorithms applied to the scalar problem (35), $\kappa = .1, \mu = .5$.

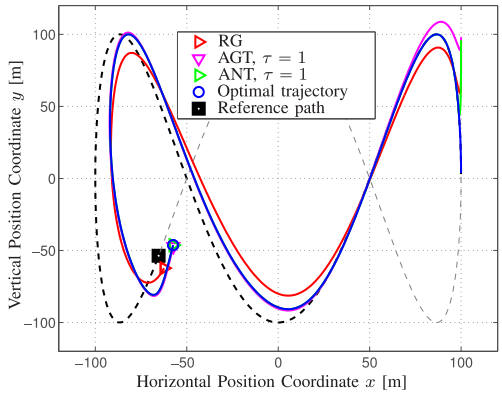

Fig. 5. Sample trajectories of the object to be tracked (dashed) and trajectories generated by the different algorithms (continuous). All algorithms track the optimum effectively, yet AGT and ANT track $\boldsymbol{x}^*(t)$ closer than RG.

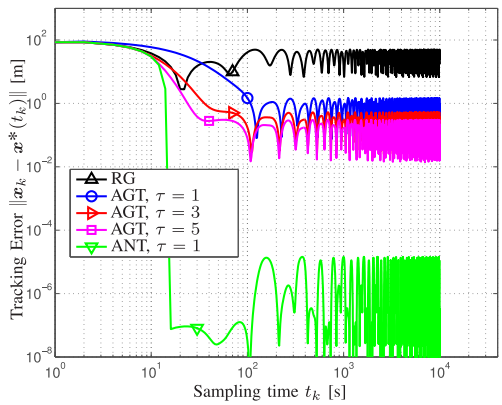

Fig. 6. Error [m] with respect to the sampling time t_k for $h = 1$ [s] for different algorithms applied to the tracking problem (38).

modify our algorithms to account for this constraint by rescaling the prediction-correction step to the allowable velocity limit. Of course more complicated actuation models may be considered, but these are beyond the scope of this work.

We show the result of this experiment in terms of the actual reference path and trajectories generated by the approximate algorithms AGT and ANT in Fig. 5 over a truncated time interval $0 < t < 300$ s. The reference trajectory $\boldsymbol{y}(t)$ is the dotted line, and the optimal continuous-time trajectory $\boldsymbol{x}^*(t)$ associated with solving (38) is in blue. By running gradient we mean a method which has no prediction step, and operates only by correction. Observe that the trajectories generated running gradient (RG), AGT, and ANT successfully track the optimal trajectory $\boldsymbol{x}^*(t)$, and consequently the reference path $\boldsymbol{y}(t)$ up to a small error.

This trend may be more easily observed in Fig. 6 which shows the magnitude of the difference between the generated path and the optimal path $\|\boldsymbol{x}^*(t_k) - \boldsymbol{x}_k\|$, or the tracking error, as compared with the sampling time t_k. Note that the asymptotical bounds computed via Theorems 1 and 3 are less meaningful here since the velocity of the robot is scaled. The approximate steady state errors achieved by RG, AGT, and ANT are respectively 10, 10^{-1}, and 10^{-5}. AGT experiences comparable

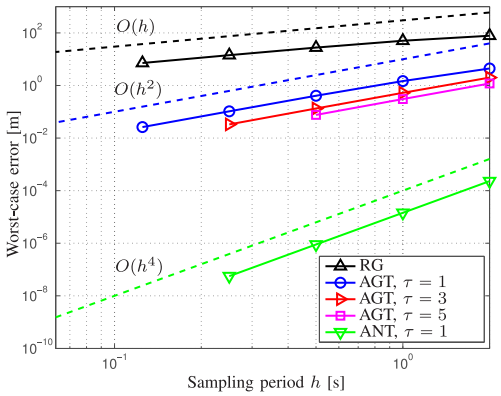

Fig. 7. Worst case error floor with respect to the sampling time interval h for different algorithms applied to the tracking problem (38).

levels of error across different values of τ, the number of correction steps, and ANT far outperforms the other methods. This pattern is corroborated in Fig. 7, which plots the worst-case error $\max_{k \geqslant \overline{k}} \|\boldsymbol{x}^*(t_k) - \boldsymbol{x}_k\|$ versus the sampling interval size h for $\overline{k} = 8 \times 10^3$. In particular, we observe that RG experiences an error comparable to $O(h)$, as it theoretically guarantees, whereas our proposed methods AGT and ANT achieve a worst-case error of approximately $O(h^2)$ and $O(h^4)$, respectively. Observe that as the problem (38) is sampled less often, i.e., when h increases, the optimality gap increases.

Computational Considerations: We empirically observe ANT to far outperform the other methods; however, this performance gap ignores the increased computational cost associated with Newton steps. To obtain a more fair comparison, we consider how the different algorithms perform when the computational time per correction and prediction steps are fixed. Theoretically, each prediction step and Newton step require $O(n^3)$ computations (because of the matrix inversion), while the gradient step only $O(n)$. Practically, in this simulation setting, the most demanding task is however the *evaluation* of the gradient and the Hessian, while the actual prediction or correction step is less critical (less than 1/10 time). In particular, evaluating the Hessian requires *twice* the computational effort of evaluating the gradient, so a Newton step is three times slower than a gradient step.

The workflow for each optimization iteration is the following:

t_k) A new function is acquired;
1) A new way point $\boldsymbol{x}_k$ is generated via a correction step;
2) The way point is implemented and the robot moves;
3) Either a new prediction $\boldsymbol{x}_{k+1|k}$ is made, based on past information, or the correction is refined by more correction steps.

We see that at step 3 the robot can either implement the prediction part of our prediction-correction algorithms, or refine the correction to have, perhaps, a better starting point when the next function is acquired. We consider here the running gradient RG (which we remark is nothing less than AGT without prediction), the AGT, the ANT, and a running version of the Newton method, which uses only correction steps (later indicated as RN).

We now outline how Table I is generated. We set $\Delta t_{\mathrm{c}} = h/10$ as the allowable computational time for the correction step (step 1), and we set the gradient evaluation to require $1/120$ s. As a consequence, for this setting the robot can perform only $\tau = 1$ gradient correction step for a sampling time of $h = 0.1$ s. With this as our basic unit of measurement, we fill in Table I with how many gradient evaluations τ may be afforded with increasing the sampling interval h. As previously noted, ANT requires three times the computation time of AGT, and consequently experiences too much latency to be used when $h = .1$ s.

TABLE I
NUMBER OF CORRECTION STEPS TO KEEP THE SAME COMPUTATIONAL TIME

Sampling period h [s]	1/10	1/4	1/3	1/2	2/3	3/4	1
RG	1	3	4	6	8	9	12
RN	—	1	1	2	2	3	4
AGT	1	3	4	6	8	9	12
ANT	—	1	1	2	2	3	4

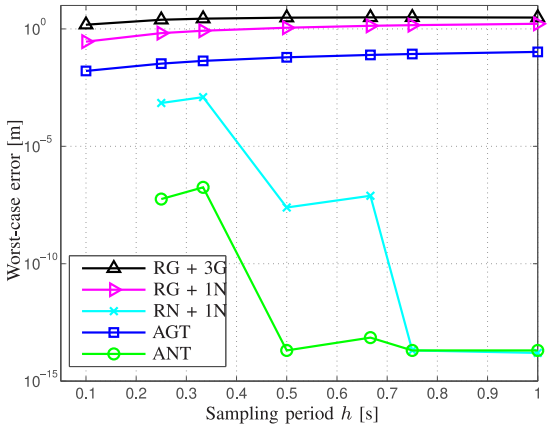

Fig. 8. Worst case Error [m] w.r.t. h [s] with fixed computational complexity.

We set as $\Delta t_{\mathrm{p}} = 1/40$ s as the allowable computational time for step 3, so that we can either run one prediction step, 3 gradient correction refinement steps, or 1 Newton correction refinement step.

We run the different algorithms when the computation time is fixed (i.e., for $h = .1$ s, in step 1. $\tau = 1$ steps of RG and AGT may be afforded, but *zero* of ANT) and record the worst-case error achieved versus h in Fig. 8. We run RG both with 3 additional gradient refinement steps (3G) and with 1 Newton refinement step (1N), while RN is run with 1 Newton refinement (1N). Broadly, one may observe that if ANT may be afforded (i.e., for large h), it is much preferable to AGT regardless of the number of correction steps τ. However, for small sampling periods h, i.e., when one requires very low latencies in the control loop, ANT is infeasible. We also observe that prediction is to be preferred to additional refinement steps, especially when the sampling period is small (i.e., when the time derivative approximation makes a significant difference because one does not have enough time to perform many correction steps).

V. CONCLUSION

We have designed algorithms to track the solution of time-varying unconstrained and strongly convex optimization problems. These algorithms leverage the knowledge of how the cost function changes in time and are based on a predictor-

corrector scheme. We have also developed approximation schemes for when the rate at which the objective varies in time is not known. We established that these methods yield convergence to a neighborhood of the optimal trajectory, with a neighborhood of convergence dependent on the sampling period. Moreover, the size of this neighborhood is an order of magnitude smaller than state-of-the art running algorithms which only perform correction steps. In some cases when the problem parameters are appropriately chosen and second-order information is incorporated, the neighborhood of the optimal trajectory to which the algorithm converges is several orders of magnitude smaller than existing approaches.

Moreover, we conducted a numerical analysis of the proposed methods in a simple setting which empirically supported the established error bounds. We also considered the task of developing a control strategy for an autonomous system to follow an object whose position varies continuously in time, showing that the developed tools yield an effective strategy. In some cases, the algorithms which achieve higher accuracy require too much computational latency to be used in a closed loop control setting; however, when this latency may be afforded, the second-order methods yield highly accurate tools.

Future research directions encompass the generalization of this work to constrained problems, general convex cost functions, as well as approximate second-order methods to weaken the computational requirements of computing the Hessian inverse in the prediction step.

Appendix A
Proof of Proposition 1

Let us analyze the forward Euler method applied to the vector-valued nonlinear dynamical system

$$\dot{\boldsymbol{x}} = \boldsymbol{F}\left(\boldsymbol{x}\left(t\right), t\right). \tag{39}$$

If we apply the forward Euler method to the relation in (39), starting at a certain point $\boldsymbol{x}(t_k)$, we obtain

$$\boldsymbol{x}_{k+1|k} = \boldsymbol{x}\left(t_k\right) + h\,\boldsymbol{F}\left(\boldsymbol{x}\left(t_k\right), t_k\right). \tag{40}$$

On the other hand, we can write $\boldsymbol{x}(t_{k+1})$ by using a Taylor expansion as

$$\boldsymbol{x}\left(t_{k+1}\right) = \boldsymbol{x}\left(t_k\right) + h\,\boldsymbol{F}\left(\boldsymbol{x}\left(t_k\right), t_k\right) + \frac{h^2}{2}\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{F}\left(\boldsymbol{x}\left(s\right), s\right), \tag{41}$$

for a certain time $s \in [t_k, t_{k+1}]$. Subtracting $\boldsymbol{x}(t_{k+1})$ from the both sides of the equality in (40) and computing the norm of the resulting relation implies that

$$\left\|\boldsymbol{x}_{k+1|k} - \boldsymbol{x}\left(t_{k+1}\right)\right\| = \left\|\frac{h^2}{2}\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{F}\left(\boldsymbol{x}\left(s\right), s\right)\right\|. \tag{42}$$

By considering the definition of the discretization error vector $\boldsymbol{\Delta}_k := \boldsymbol{x}_{k+1|k} - \boldsymbol{x}(t_{k+1})$, we can write (42) as

$$\left\|\,\boldsymbol{\Delta}_k\,\right\| = \frac{h^2}{2}\left\|\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{F}\left(\boldsymbol{x}\left(s\right), s\right)\right\|. \tag{43}$$

We proceed to find an upper bound for the right-hand side of (43). Observing the continuous dynamical system in (6) we know that $\boldsymbol{F}(\boldsymbol{x}(t), t)$ is given by

$$\boldsymbol{F}\left(\boldsymbol{x}\left(t\right), t\right) = -\left[\nabla_{\boldsymbol{xx}} f\left(\boldsymbol{x}; t\right)\right]^{-1}\nabla_{t\boldsymbol{x}} f\left(\boldsymbol{x}; t\right). \tag{44}$$

Then, by the chain rule we can write

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{F}(\boldsymbol{x}(t), t) &= \nabla_t \boldsymbol{F}(\boldsymbol{x}, t) + \left[\nabla_{\boldsymbol{x}}\boldsymbol{F}(\boldsymbol{x}, t)\right]\dot{\boldsymbol{x}} \\ &= \nabla_t \boldsymbol{F}(\boldsymbol{x}, t) + \left[\nabla_{\boldsymbol{x}}\boldsymbol{F}(\boldsymbol{x}, t)\right]\boldsymbol{F}(\boldsymbol{x}(t), t),\end{aligned} \tag{45}$$

where we have used the relation (39). By using the triangle inequality, we can upper bound the norm of the right-hand side of (45) as

$$\left\|\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{F}(\boldsymbol{x}(t), t)\right\| \leqslant \left\|\nabla_t \boldsymbol{F}(\boldsymbol{x}, t)\right\| + \left\|\left[\nabla_{\boldsymbol{x}}\boldsymbol{F}(\boldsymbol{x}, t)\right]\boldsymbol{F}(\boldsymbol{x}(t), t)\right\|. \tag{46}$$

We now upper bound the right-hand side of (46) by analyzing its two components. First, based on the definition in (44), the partial derivative w.r.t. time can be written as, $\nabla_t \boldsymbol{F}(\boldsymbol{x}, t) = -\nabla_t\left[\left[\nabla_{\boldsymbol{xx}} f(\boldsymbol{x}; t)\right]^{-1}\nabla_{t\boldsymbol{x}} f(\boldsymbol{x}; t)\right]$. By applying the chain rule,

$$\begin{aligned}\nabla_t\left[\left[\nabla_{\boldsymbol{xx}} f\left(\boldsymbol{x}; t\right)\right]^{-1}\nabla_{t\boldsymbol{x}} f\left(\boldsymbol{x}; t\right)\right] &= \left[\nabla_{\boldsymbol{xx}} f\left(\boldsymbol{x}; t\right)\right]^{-1}\nabla_{tt\boldsymbol{x}} f(\boldsymbol{x}; t) \\ &\quad - \left[\nabla_{\boldsymbol{xx}} f\left(\boldsymbol{x}; t\right)\right]^{-2}\nabla_{t\boldsymbol{xx}} f\left(\boldsymbol{x}; t\right)\nabla_{t\boldsymbol{x}} f\left(\boldsymbol{x}; t\right).\end{aligned} \tag{47}$$

Compute the norm of both sides of (47). Substitute the norm $\|\nabla_t[[\nabla_{\boldsymbol{xx}} f(\boldsymbol{x}; t)]^{-1}\nabla_{t\boldsymbol{x}} f(\boldsymbol{x}; t)]\|$ by $\|\nabla_t \boldsymbol{F}(\boldsymbol{x}, t)\|$. Further, apply the triangle inequality to the right-hand side of the resulting expression to obtain

$$\begin{aligned}\left\|\nabla_t \boldsymbol{F}\left(\boldsymbol{x}, t\right)\right\| &\leqslant \left\|\left[\nabla_{\boldsymbol{xx}} f\left(\boldsymbol{x}; t\right)\right]^{-2}\nabla_{t\boldsymbol{xx}} f\left(\boldsymbol{x}; t\right)\nabla_{t\boldsymbol{x}} f\left(\boldsymbol{x}; t\right)\right\| \\ &\quad + \left\|\left[\nabla_{\boldsymbol{xx}} f\left(\boldsymbol{x}; t\right)\right]^{-1}\nabla_{tt\boldsymbol{x}} f\left(\boldsymbol{x}; t\right)\right\|.\end{aligned} \tag{48}$$

Observe the fact that $\nabla_{t\boldsymbol{xx}} f(\boldsymbol{x}; t) = \nabla_{\boldsymbol{x}t\boldsymbol{x}} f(\boldsymbol{x}; t)$. We use the Cauchy-Schwartz inequality and the bounds in Assumptions 1 and 2 to update the upper bound in (48) as

$$\left\|\nabla_t \boldsymbol{F}\left(\boldsymbol{x}, t\right)\right\| \leqslant \frac{C_0 C_2}{m^2} + \frac{C_3}{m}. \tag{49}$$

We can now do the same for the second component of the right-hand side of (46), and in particular

$$\begin{aligned}&\left\|\nabla_{\boldsymbol{x}}\boldsymbol{F}(\boldsymbol{x}, t)\,\boldsymbol{F}(\boldsymbol{x}(t), t)\right\| = \left\|\left(\left[\nabla_{\boldsymbol{xx}} f(\boldsymbol{x}; t)\right]^{-1}\nabla_{\boldsymbol{x}t\boldsymbol{x}} f(\boldsymbol{x}; t)\right.\right. \\ &\left.\left. - \left[\nabla_{\boldsymbol{xx}} f(\boldsymbol{x}; t)\right]^{-2}\nabla_{\boldsymbol{xxx}} f(\boldsymbol{x}; t)\nabla_{t\boldsymbol{x}} f(\boldsymbol{x}; t)\right)\boldsymbol{F}(\boldsymbol{x}(t), t)\right\| \\ &\leqslant \left(\frac{C_2}{m} + \frac{C_1 C_0}{m^2}\right)\frac{C_0}{m}.\end{aligned} \tag{50}$$

By combining the relation in (43) and (46) with the upper bounds in (49) and (50), the claim in (14) follows. ■

Appendix B
Proof of Theorem 1

In order to prove Theorem 1, we start by bounding the error in the prediction step by the terms that depend on the functional smoothness and the discretization error using Taylor expansions. Then we bound the tracking error of the gradient step using convergence properties of the gradient on strongly convex functions. By substituting the error of the correction step into the prediction step, we establish the main result.

First, we establish that discrete-time sampling error bound stated in (18) is achieved by the updates (7)–(8). For simplicity,

we modify the notation to omit the arguments $\boldsymbol{x}_k$ and t_k of the function f. In particular, define

$$\nabla_{\boldsymbol{xx}} f := \nabla_{\boldsymbol{xx}} f\left(\boldsymbol{x}_k; t_k\right),\ \nabla_{t\boldsymbol{x}} f := \nabla_{t\boldsymbol{x}} f\left(\boldsymbol{x}_k; t_k\right),$$
$$\nabla_{\boldsymbol{xx}} f^* := \nabla_{\boldsymbol{xx}} f\left(\boldsymbol{x}^*\left(t_k\right); t_k\right), \nabla_{t\boldsymbol{x}} f^* := \nabla_{t\boldsymbol{x}} f\left(\boldsymbol{x}^*\left(t_k\right); t_k\right). \quad (51)$$

Begin by considering the update in (7), the prediction step, evaluated at a generic point $\boldsymbol{x}_k$ sampled at the current sample time t_k and with associated optimizer $\boldsymbol{x}^*(t)$, which due to optimality will have null residual vector $\boldsymbol{r}(t) = \mathbf{0}$. Thus we may write

$$\begin{cases} \boldsymbol{x}_{k+1|k} = \boldsymbol{x}_k - h\left[\nabla_{\boldsymbol{xx}} f\right]^{-1} \nabla_{t\boldsymbol{x}} f \\ \boldsymbol{x}^*\left(t_{k+1}\right) = \boldsymbol{x}^*\left(t_k\right) - h\left[\nabla_{\boldsymbol{xx}} f^*\right]^{-1} \nabla_{t\boldsymbol{x}} f^* + \boldsymbol{\Delta}_k. \end{cases} \quad (52)$$

By subtracting the equalities in (52), considering the norm of the resulting expression, and applying the triangle inequality we obtain

$$\|\boldsymbol{x}_{k+1|k} - \boldsymbol{x}^*\left(t_{k+1}\right)\| \leqslant \|\boldsymbol{x}_k - \boldsymbol{x}^*\left(t_k\right)\| + h\|[\nabla_{\boldsymbol{xx}} f]^{-1}\nabla_{t\boldsymbol{x}} f - [\nabla_{\boldsymbol{xx}} f^*]^{-1}\nabla_{t\boldsymbol{x}} f^*\| + \|\boldsymbol{\Delta}_k\|. \quad (53)$$

Substituting the discretization error norm $\|\boldsymbol{\Delta}_k\|$ by its upper bound in (14) follows

$$\|\boldsymbol{x}_{k+1|k} - \boldsymbol{x}^*\left(t_{k+1}\right)\| \leqslant \|\boldsymbol{x}_k - \boldsymbol{x}^*\left(t_k\right)\| + \frac{h^2}{2}\left[\frac{C_0^2 C_1}{m^3} + \frac{2C_0 C_2}{m^2} + \frac{C_3}{m}\right] + h\|[\nabla_{\boldsymbol{xx}} f]^{-1}\nabla_{t\boldsymbol{x}} f - [\nabla_{\boldsymbol{xx}} f^*]^{-1}\nabla_{t\boldsymbol{x}} f^*\|. \quad (54)$$

We proceed to find an upper bound for the norm $\|[\nabla_{\boldsymbol{xx}} f]^{-1}\nabla_{t\boldsymbol{x}} f - [\nabla_{\boldsymbol{xx}} f^*]^{-1}\nabla_{t\boldsymbol{x}} f^*\|$ in the right-hand side of (54). By adding and subtracting the term $[\nabla_{\boldsymbol{xx}} f^*]^{-1}\nabla_{t\boldsymbol{x}} f$ and using triangle inequality we can write

$$\|[\nabla_{\boldsymbol{xx}} f]^{-1}\nabla_{t\boldsymbol{x}} f - [\nabla_{\boldsymbol{xx}} f^*]^{-1}\nabla_{t\boldsymbol{x}} f^*\| \leqslant \|[\nabla_{\boldsymbol{xx}} f]^{-1}\nabla_{t\boldsymbol{x}} f - [\nabla_{\boldsymbol{xx}} f^*]^{-1}\nabla_{t\boldsymbol{x}} f\| + \|[\nabla_{\boldsymbol{xx}} f^*]^{-1}\nabla_{t\boldsymbol{x}} f - [\nabla_{\boldsymbol{xx}} f^*]^{-1}\nabla_{t\boldsymbol{x}} f^*\|. \quad (55)$$

We may bound the first and second-order derivative terms in (55) by using Assumption 2 regarding the functional smoothness as well as the strong convexity constant m of the Hessian in Assumption 1 to write

$$\|[\nabla_{\boldsymbol{xx}} f]^{-1}\nabla_{t\boldsymbol{x}} f - [\nabla_{\boldsymbol{xx}} f^*]^{-1}\nabla_{t\boldsymbol{x}} f^*\| \leqslant C_0\|[\nabla_{\boldsymbol{xx}} f]^{-1} - [\nabla_{\boldsymbol{xx}} f^*]^{-1}\| + \frac{1}{m}\|\nabla_{t\boldsymbol{x}} f - \nabla_{t\boldsymbol{x}} f^*\|. \quad (56)$$

We now further bound the first term of the right-hand side. To do that, we use the non-singularity of the Hessian to write

$$\|[\nabla_{\boldsymbol{xx}} f]^{-1} - [\nabla_{\boldsymbol{xx}} f^*]^{-1}\| = \|[\nabla_{\boldsymbol{xx}} f^*]^{-1}\left(\nabla_{\boldsymbol{xx}} f - \nabla_{\boldsymbol{xx}} f^*\right)[\nabla_{\boldsymbol{xx}} f]^{-1}\|, \quad (57)$$

which by employing, once again, the strong convexity constant m of the Hessian in Assumption 1 we can bound as

$$\|[\nabla_{\boldsymbol{xx}} f]^{-1} - [\nabla_{\boldsymbol{xx}} f^*]^{-1}\| \leqslant \frac{1}{m^2}\|\nabla_{\boldsymbol{xx}} f - \nabla_{\boldsymbol{xx}} f^*\|. \quad (58)$$

Substituting the upper bound in (58) for the norm $\|[\nabla_{\boldsymbol{xx}} f]^{-1} - [\nabla_{\boldsymbol{xx}} f^*]^{-1}\|$ into (56) yields

$$\|[\nabla_{\boldsymbol{xx}} f]^{-1}\nabla_{t\boldsymbol{x}} f - [\nabla_{\boldsymbol{xx}} f^*]^{-1}\nabla_{t\boldsymbol{x}} f^*\| \leqslant \frac{C_0}{m^2}\|\nabla_{\boldsymbol{xx}} f - \nabla_{\boldsymbol{xx}} f^*\| + \frac{1}{m}\|\nabla_{t\boldsymbol{x}} f - \nabla_{t\boldsymbol{x}} f^*\|. \quad (59)$$

We consider the Taylor expansion of the second-order term in (59), and apply the Mean Value Theorem with $\widetilde{\boldsymbol{x}}$ as a point on the line between $\boldsymbol{x}_k$ and $\boldsymbol{x}^*(t_k)$ to obtain

$$\|\nabla_{\boldsymbol{xx}} f - \nabla_{\boldsymbol{xx}} f^*\| \leqslant \|\nabla_{\boldsymbol{xxx}} f\left(\widetilde{\boldsymbol{x}}; t_k\right)\|\|\boldsymbol{x}_k - \boldsymbol{x}^*\left(t_k\right)\| \leqslant C_1\|\boldsymbol{x}_k - \boldsymbol{x}^*\left(t_k\right)\|. \quad (60)$$

Applying the same argument for the mixed second-order term implies

$$\|\nabla_{t\boldsymbol{x}} f - \nabla_{t\boldsymbol{x}} f^*\| \leqslant \|\nabla_{\boldsymbol{x}t\boldsymbol{x}} f\left(\widetilde{\boldsymbol{x}}; t_k\right)\|\|\boldsymbol{x}_k - \boldsymbol{x}^*\left(t_k\right)\| \leqslant C_2\|\boldsymbol{x}_k - \boldsymbol{x}^*\left(t_k\right)\| \quad (61)$$

The expressions in (60) and (61) may be substituted together into (59) to yield

$$\|[\nabla_{\boldsymbol{xx}} f]^{-1}\nabla_{t\boldsymbol{x}} f - [\nabla_{\boldsymbol{xx}} f^*]^{-1}\nabla_{t\boldsymbol{x}} f^*\| \leqslant \left(\frac{C_0 C_1}{m^2} + \frac{C_2}{m}\right)\|\boldsymbol{x}_k - \boldsymbol{x}^*\left(t_k\right)\|. \quad (62)$$

By substituting the upper bound in (62) into (54) and considering the definition of σ in (15), we obtain that

$$\|\boldsymbol{x}_{k+1|k} - \boldsymbol{x}^*\left(t_{k+1}\right)\| \leqslant \sigma\|\boldsymbol{x}_k - \boldsymbol{x}^*\left(t_k\right)\| + \frac{h^2}{2}\left[\frac{C_0^2 C_1}{m^3} + \frac{2C_0 C_2}{m^2} + \frac{C_3}{m}\right]. \quad (63)$$

For the correction step [cf. (8)], we may use the standard property of gradient descent for strongly convex functions with Lipschitz gradients. In particular, the Euclidean error norm of the gradient descent method converges as

$$\|\widehat{\boldsymbol{x}}_{k+1}^{s+1} - \boldsymbol{x}^*\left(t_{k+1}\right)\| \leqslant \rho\|\widehat{\boldsymbol{x}}_{k+1}^{s} - \boldsymbol{x}^*\left(t_{k+1}\right)\|. \quad (64)$$

where $\rho = \max\{|1-\gamma m|, |1-\gamma L|\}$. To see this, it is sufficient to write the gradient step as

$$\|\widehat{\boldsymbol{x}}_{k+1}^{s+1} - \boldsymbol{x}^*\left(t_{k+1}\right)\| = \|\widehat{\boldsymbol{x}}_{k+1}^{s} - \gamma\nabla_{\boldsymbol{x}} f\left(\widehat{\boldsymbol{x}}_{k+1}^{s}; t_{k+1}\right) - \boldsymbol{x}^*\left(t_{k+1}\right)\|. \quad (65)$$

According to the optimality condition we can write $\nabla_{\boldsymbol{x}} f(\boldsymbol{x}^*(t_{k+1}); t_{k+1}) = \mathbf{0}$. Considering this observation and the equality in (65) we obtain

$$\|\widehat{\boldsymbol{x}}_{k+1}^{s+1} - \boldsymbol{x}^*\left(t_{k+1}\right)\| = \|\widehat{\boldsymbol{x}}_{k+1}^{s} - \boldsymbol{x}^*\left(t_{k+1}\right) - \gamma\left[\nabla_{\boldsymbol{x}} f\left(\widehat{\boldsymbol{x}}_{k+1}^{s}; t_{k+1}\right) - \nabla_{\boldsymbol{x}} f\left(\boldsymbol{x}^*\left(t_{k+1}\right); t_{k+1}\right)\right]\|. \quad (66)$$

Consider now the continuous function $g : \mathbb{R}^n \times \mathbb{R}_+ \to \mathbb{R}$ defined as $g(\boldsymbol{x}; t) := \boldsymbol{x} - \gamma\nabla_{\boldsymbol{x}} f(\boldsymbol{x}; t)$. Given the boundedness of the Hessian and the strong convexity of $f(\boldsymbol{x}; t)$, the gradient of $g(\boldsymbol{x}; t)$ is bounded as [38, p. 13]

$$\|\nabla_{\boldsymbol{x}} g\left(\boldsymbol{x}; t\right)\| \leqslant \max\left\{|1-\gamma m|, |1-\gamma L|\right\} = \rho, \quad (67)$$

for all $\boldsymbol{x} \in \mathbb{R}^n$. The bound (67) implies that $g(\boldsymbol{x};t)$ is Lipschitz, therefore we can upper bound (66) as

$$\begin{aligned}\|\widehat{\boldsymbol{x}}_{k+1}^{s+1} - \boldsymbol{x}^*(t_{k+1})\| &= \|g\left(\widehat{\boldsymbol{x}}_{k+1}^{s};t_{k+1}\right) \\ &- g\left(\boldsymbol{x}^*(t_{k+1});t_{k+1}\right)\| \leqslant \rho\|\widehat{\boldsymbol{x}}_{k+1}^{s} - \boldsymbol{x}^*(t_{k+1})\|. \end{aligned} \tag{68}$$

Notice that the relation (68) is equivalent to the claim in (64).

Observe that the sequence $\widehat{\boldsymbol{x}}_{k+1}^{s}$ is initialized by the predicted variable $\boldsymbol{x}_{k+1|k}$ and the corrected variable $\boldsymbol{x}_{k+1}$ is equal to $\widehat{\boldsymbol{x}}_{k+1}^{\tau}$. Considering these observations and the relation in (64) between two consecutive iterates of the sequence $\widehat{\boldsymbol{x}}_{k+1}^{s}$ we can write

$$\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*(t_{k+1})\| \leqslant \rho^{\tau}\|\boldsymbol{x}_{k+1|k} - \boldsymbol{x}^*(t_{k+1})\|. \tag{69}$$

We are ready to consider the combined error bound achieved by the prediction-correction scheme. By plugging the correction error of (69) into the prediction error of (63) we obtain

$$\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*(t_{k+1})\| \leqslant \rho^{\tau}\sigma\|\boldsymbol{x}_{k} - \boldsymbol{x}^*(t_{k})\| + \rho^{\tau}\Gamma, \tag{70}$$

where $\Gamma := (h^2/2)[C_0^2C_1/m^3 + 2C_0C_2/m^2 + C_3/m]$ is defined to simplify the notation. Notice that the relation between $\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*(t_{k+1})\|$ and $\|\boldsymbol{x}_{k} - \boldsymbol{x}^*(t_{k})\|$ in (70) also holds true for $\|\boldsymbol{x}_{k} - \boldsymbol{x}^*(t_{k})\|$ and $\|\boldsymbol{x}_{k-1} - \boldsymbol{x}^*(t_{k-1})\|$, i.e.,

$$\|\boldsymbol{x}_{k} - \boldsymbol{x}^*(t_{k})\| \leqslant \rho^{\tau}\sigma\|\boldsymbol{x}_{k-1} - \boldsymbol{x}^*(t_{k-1})\| + \rho^{\tau}\Gamma. \tag{71}$$

Substituting the upper bound in (71) for $\|\boldsymbol{x}_{k} - \boldsymbol{x}^*(t_{k})\|$ into (70) implies an upper bound for $\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*(t_{k+1})\|$ in terms of the norm difference for time $k-1$ as

$$\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*(t_{k+1})\| \leqslant (\rho^{\tau}\sigma)^2\|\boldsymbol{x}_{k-1} - \boldsymbol{x}^*(t_{k-1})\| + \rho^{\tau}\Gamma(\rho^{\tau}\sigma + 1). \tag{72}$$

Now recursively apply the relationship (70) backwards in time to the initial time sample and use the same argument form (70) to (72) to write

$$\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*(t_{k+1})\| \leqslant (\rho^{\tau}\sigma)^{k+1}\|\boldsymbol{x}_{0} - \boldsymbol{x}^*(t_{0})\| + \rho^{\tau}\Gamma\sum_{i=0}^{k}(\rho^{\tau}\sigma)^{i}. \tag{73}$$

Substituting $k+1$ by k and simplifying the sum in (73) (remembering that $\rho^{\tau}\sigma < 1$) leads to

$$\|\boldsymbol{x}_{k} - \boldsymbol{x}^*(t_{k})\| \leqslant (\rho^{\tau}\sigma)^{k}\|\boldsymbol{x}_{0} - \boldsymbol{x}^*(t_{0})\| + \rho^{\tau}\Gamma\left[\frac{1-(\rho^{\tau}\sigma)^{k}}{1-\rho^{\tau}\sigma}\right]. \tag{74}$$

Considering the result in (74) and the definition for the constant Γ, the result in (18) follows.

To establish the result stated in (16), observe that in the worst case, we may upper bound the term $\|[\nabla_{\boldsymbol{xx}}f]^{-1}\nabla_{t\boldsymbol{x}}f - [\nabla_{\boldsymbol{xx}}f^*]^{-1}\nabla_{t\boldsymbol{x}}f^*\|$ in (53) by using the bounds in Assumption 2 to obtain the right-hand side of the following expression

$$\|[\nabla_{\boldsymbol{xx}}f]^{-1}\nabla_{t\boldsymbol{x}}f - [\nabla_{\boldsymbol{xx}}f^*]^{-1}\nabla_{t\boldsymbol{x}}f^*\| \leqslant \frac{2C_0}{m}. \tag{75}$$

Substituting the bound in (75) into (54) yields

$$\begin{aligned}\|\boldsymbol{x}_{k+1|k} - \boldsymbol{x}^*(t_{k+1})\| &\leqslant \|\boldsymbol{x}_{k} - \boldsymbol{x}^*(t_{k})\| + h\,\frac{2C_0}{m} \\ &+ \frac{h^2}{2}\left[\frac{C_0^2C_1}{m^3} + \frac{2C_0C_2}{m^2} + \frac{C_3}{m}\right]. \end{aligned} \tag{76}$$

To simplify the notation we define a new constant $\Gamma_2 := 2hC_0/m$ and we use again the definition $\Gamma := (h^2/2)[C_0^2C_1/m^3 + 2C_0C_2/m^2 + C_3/m]$. Considering this definition and observing the relation in (69) we can write

$$\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*(t_{k+1})\| \leqslant \rho^{\tau}\|\boldsymbol{x}_{k} - \boldsymbol{x}^*(t_{k})\| + \rho^{\tau}\left(\Gamma_2 + \Gamma\right). \tag{77}$$

Now recursively apply the relationship (77) backwards in time to the initial time sample and use the same argument from (70) to (74) to write

$$\begin{aligned}\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*(t_{k+1})\| &\leqslant \rho^{\tau(k+1)}\|\boldsymbol{x}_{0} - \boldsymbol{x}^*(t_{0})\| \\ &+ \rho^{\tau}\left(\Gamma_2 + \Gamma\right)\left[\frac{1-\rho^{\tau(k+1)}}{1-\rho^{\tau}}\right]. \end{aligned} \tag{78}$$

Note that relation (78) shows an upper bound for $\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*(t_{k+1})\|$ in terms of the initial error $\|\boldsymbol{x}_{0} - \boldsymbol{x}^*(t_{0})\|$ and an extra error term for the bound of convergence. If we substitute $k+1$ by k in (78) and recall the definition of Γ_2 and Γ, then the result in (16) follows.

For completeness, we show that $\rho < 1$ requires the stepsize to be selected as $\gamma < 2/L$, which therefore enforce a finite right-hand side in (78). Starting by the definition of ρ, we require

$$\rho := \max\{|1-\gamma m|, |1-\gamma L|\} < 1. \tag{79}$$

Solving this equation for γ and recalling that $m \leqslant L$ by Assumptions 1 and 2, the condition $\gamma < 2/L$ follows. ■

Appendix C
Proof of Theorem 2

We consider once again the proof of Theorem 1, in particular (63) for $k = 0$, due to the prediction step. For the correction step, if we applied one time the Newton method, we would have

$$\|\boldsymbol{x}_{1} - \boldsymbol{x}^*(t_{1})\| \leqslant \frac{C_1}{2m}\|\boldsymbol{x}_{1|0} - \boldsymbol{x}^*(t_{1})\|^2. \tag{80}$$

We proceed to check the validity of (80). To do so, we first simplify the notations as

$$\begin{aligned}&\nabla_{\boldsymbol{xx}}f_1 = \nabla_{\boldsymbol{xx}}f\left(\boldsymbol{x}_{1|0};t_1\right),\ \nabla_{\boldsymbol{x}}f_1 = \nabla_{\boldsymbol{x}}f\left(\boldsymbol{x}_{1|0};t_1\right), \\ &\nabla_{\boldsymbol{xx}}f_1^* = \nabla_{\boldsymbol{xx}}f\left(\boldsymbol{x}^*(t_1);t_1\right),\ \nabla_{\boldsymbol{x}}f_1^* = \nabla_{\boldsymbol{x}}f\left(\boldsymbol{x}^*(t_1);t_1\right). \end{aligned} \tag{81}$$

Considering the update of the Newton method which is used in the correction step of NTT we can write

$$\|\boldsymbol{x}_{1} - \boldsymbol{x}^*(t_{1})\| = \|\boldsymbol{x}_{1|0} - \nabla_{\boldsymbol{xx}}f_1^{-1}\nabla_{\boldsymbol{x}}f_1 - \boldsymbol{x}^*(t_{1})\|,. \tag{82}$$

By factoring the Hessian inverse $\nabla_{\boldsymbol{xx}}f_1^{-1}$ and using the fact that the norm of a product is smaller than the product of the norms, we can show that the right-hand side of (82) is bounded

above as

$$\begin{aligned}&\|\boldsymbol{x}_{1|0}-\nabla_{\boldsymbol{xx}}f_1^{-1}\nabla_{\boldsymbol{x}}f_1-\boldsymbol{x}^*(t_1)\|\\&\leqslant\|\nabla_{\boldsymbol{xx}}f_1^{-1}\|\|\nabla_{\boldsymbol{xx}}f_1\left(\boldsymbol{x}_{1|0}-\boldsymbol{x}^*(t_1)\right)-\nabla_{\boldsymbol{x}}f_1\|.\end{aligned}\tag{83}$$

Notice that the norm $\|\nabla_{\boldsymbol{xx}}f_1^{-1}\|$ is bounded above by $1/m$ according to the strong convexity assumption. Further, the optimality conditions imply $\nabla_{\boldsymbol{x}}f_1^*=\mathbf{0}$. These observations imply that we can rewrite (83) as

$$\begin{aligned}&\|\boldsymbol{x}_{1|0}-\nabla_{\boldsymbol{xx}}f_1^{-1}\nabla_{\boldsymbol{x}}f_1-\boldsymbol{x}^*(t_1)\|\\&\leqslant\frac{1}{m}\|\nabla_{\boldsymbol{xx}}f_1\left(\boldsymbol{x}_{1|0}-\boldsymbol{x}^*(t_1)\right)-(\nabla_{\boldsymbol{x}}f_1-\nabla_{\boldsymbol{x}}f_1^*)\|.\end{aligned}\tag{84}$$

Define $\boldsymbol{r}_1=\boldsymbol{x}_{1|0}-\boldsymbol{x}^*(t_1)$ and $\boldsymbol{x}i(\tau)=\boldsymbol{x}^*(t_1)+\tau(\boldsymbol{x}_{1|0}-\boldsymbol{x}^*(t_1))$. We now use the fundamental theorem of calculus and the Lipschitz continuity of the Hessian (Assumption 2) to upper bound the rightmost term of (84) as

$$\begin{aligned}&\|\nabla_{\boldsymbol{xx}}f_1\boldsymbol{r}_1-(\nabla_{\boldsymbol{x}}f_1-\nabla_{\boldsymbol{x}}f_1^*)\|\\&=\|\nabla_{\boldsymbol{xx}}f_1\boldsymbol{r}_1-\int_0^1\nabla_{\boldsymbol{xx}}f\left(\boldsymbol{\xi}(\tau);t_1\right)\boldsymbol{r}_1\mathrm{d}\tau\|\\&=\|\boldsymbol{r}_1\int_0^1\nabla_{\boldsymbol{xx}}f_1-\nabla_{\boldsymbol{xx}}f\left(\boldsymbol{\xi}(\tau);t_1\right)\mathrm{d}\tau\|\\&\leqslant\|\,\boldsymbol{r}_1\,\|\int_0^1\|\nabla_{\boldsymbol{xx}}f_1-\nabla_{\boldsymbol{xx}}f\left(\boldsymbol{\xi}(\tau);t_1\right)\|\mathrm{d}\tau\\&\leqslant C_1\,\|\,\boldsymbol{r}_1\|^2\int_0^1(1-\tau)\,\mathrm{d}\tau=\frac{C_1}{2}\|\,\boldsymbol{r}_1\,\|^2.\end{aligned}\tag{85}$$

Notice that the first inequality in (85) is implied by the Cauchy-Schwarz inequality and the second inequality is true because of the Lipschitz continuity of the gradients with constant C_1. By plugging the bound (85) into (84) and recalling the definition $\boldsymbol{r}_1=\boldsymbol{x}_{1|0}-\boldsymbol{x}^*(t_1)$ we obtain that

$$\|\boldsymbol{x}_{1|0}-\nabla_{\boldsymbol{xx}}f_1^{-1}\nabla_{\boldsymbol{x}}f_1-\boldsymbol{x}^*(t_1)\|\leqslant\frac{C_1}{2m}\|\boldsymbol{x}_{1|0}-\boldsymbol{x}^*(t_1)\|^2.\tag{86}$$

Combining the inequalities in (82) and (86) follows the claim in (80).

Now consider the case that τ steps of the Newton method are applied in the correction step of the NTT algorithm. Then, the error $\|\boldsymbol{x}_1-\boldsymbol{x}^*(t_1)\|$ at step t_1 is bounded above as

$$\|\,\boldsymbol{x}_1-\boldsymbol{x}^*(t_1)\,\|\leqslant\left(\frac{C_1}{2m}\right)^{2\tau-1}\|\boldsymbol{x}_{1|0}-\boldsymbol{x}^*(t_1)\,\|^{2\tau}.\tag{87}$$

Notice that the upper bound for the prediction error in (63) implies that the norm $\|\boldsymbol{x}_{1|0}-\boldsymbol{x}^*(t_1)\|$ is bounded above as

$$\|\boldsymbol{x}_{1|0}-\boldsymbol{x}^*(t_1)\|\leqslant\sigma\|\boldsymbol{x}_0-\boldsymbol{x}^*(t_0)\|+\frac{h^2}{2}\left[\frac{C_0^2C_1}{m^3}+\frac{2C_0C_2}{m^2}+\frac{C_3}{m}\right].\tag{88}$$

where $\sigma:=1+h\delta_1$, and δ_1 is defined in (21). Combining the inequalities in (87) and (88) and considering the definitions $Q:=2m/C_1$ and $\delta_2:=C_0^2C_1/2m^3+C_0C_2/m^2+C_3/2m$ yield

$$\|\boldsymbol{x}_1-\boldsymbol{x}^*(t_1)\,\|\leqslant Q^{-(2\tau-1)}\left(\sigma\|\boldsymbol{x}_0-\boldsymbol{x}^*(t_0)\,\|+h^2\delta_2\right)^{2\tau}.\tag{89}$$

Based on the assumption in (23) the initial error is bounded above by ch^2 (with c an arbitrary positive constant). Substituting this upper bound into the right-hand side of (89) follows

$$\|\boldsymbol{x}_1-\boldsymbol{x}^*(t_1)\,\|\leqslant Q^{-(2\tau-1)}\left((\sigma c+\delta_2)\,h^2\right)^{2\tau}.\tag{90}$$

Notice that the inequality in (90) shows that the error $\|\boldsymbol{x}_t-\boldsymbol{x}^*(t_t)\|$ for the step $t=1$ is in the order of $O(h^{4\tau})$ which is a better error bound with respect to the initial error $\|\boldsymbol{x}_0-\boldsymbol{x}^*(t_0)\|=O(h^2)$. We now proceed to find under which conditions the error in inequality (90) is valid for all $\|\boldsymbol{x}_k-\boldsymbol{x}^*(t_k)\|$ with $k\geqslant1$. To do so, we use induction. We first establish the sufficient conditions for which $\|\boldsymbol{x}_1-\boldsymbol{x}^*(t_1)\|\leqslant ch^2$; then we substitute $\|\boldsymbol{x}_2-\boldsymbol{x}^*(t_2)\|$ with $\|\boldsymbol{x}_1-\boldsymbol{x}^*(t_1)\|$ and $\|\boldsymbol{x}_1-\boldsymbol{x}^*(t_1)\|$ with $\|\boldsymbol{x}_0-\boldsymbol{x}^*(t_0)\|$ in (89) and by induction on the error term $\|\boldsymbol{x}_k-\boldsymbol{x}^*(t_k)\|$ we will prove the claim that $\|\boldsymbol{x}_k-\boldsymbol{x}^*(t_k)\|=O(h^4)$ with $k\geqslant1$. In particular, we need to make sure that the sampling period h is chosen such that the upper bound in (90) is smaller than ch^2, i.e.,

$$Q^{-(2\tau-1)}\left(\sigma ch^2+\delta_2h^2\right)^{2\tau}\leqslant ch^2.\tag{91}$$

Observe that according to the required condition for the sampling period h in (22) we can write $h\leqslant1$. Therefore, the constant $\sigma:=1+h\delta_1$ is bounded above by $1+\delta_1$. Substituting $1+\delta_1$ for σ in (91) implies a sufficient condition for (91) as

$$Q^{-(2\tau-1)}\left((1+\delta_1)\,ch^2+\delta_2h^2\right)^{2\tau}\leqslant ch^2.\tag{92}$$

We emphasize that if the inequality in (92) holds true then the statement in (91) is satisfied. Regrouping the terms in (92) leads to the following condition for the sampling interval h as

$$h\leqslant\left[\frac{Q^{(2\tau-1)}c}{\left((1+\delta_1)\,c+\delta_2\right)^{2\tau}}\right]^{\frac{1}{4\tau-2}}.\tag{93}$$

Therefore, if (93) is satisfied then (92) and subsequently (91) are satisfied. Based on the assumption in (22), we know that (93) is valid and the condition in (91) is satisfied. This observation in conjunction with the inequality in (90) implies that

$$\|\boldsymbol{x}_1-\boldsymbol{x}^*(t_1)\,\|\leqslant ch^2.\tag{94}$$

By starting again from (89), and by substituting $\|\boldsymbol{x}_2-\boldsymbol{x}^*(t_2)\|$ with $\|\boldsymbol{x}_1-\boldsymbol{x}^*(t_1)\|$ and $\|\boldsymbol{x}_1-\boldsymbol{x}^*(t_1)\|$ with $\|\boldsymbol{x}_0-\boldsymbol{x}^*(t_0)\|$, we arrive at the inequality

$$\|\boldsymbol{x}_2-\boldsymbol{x}^*(t_2)\,\|\leqslant Q^{-(2\tau-1)}\left((\sigma c+\delta_2)\,h^2\right)^{2\tau}.\tag{95}$$

Since the condition in (93) does not depend on the optimality gap, they yield $\|\boldsymbol{x}_2-\boldsymbol{x}^*(t_2)\|\leqslant ch^2$. By applying the induction argument, we can now show that

$$\|\boldsymbol{x}_k-\boldsymbol{x}^*(t_k)\,\|\leqslant Q^{-(2\tau-1)}\left((\sigma c+\delta_2)\,h^2\right)^{2\tau},\tag{96}$$

for all $k\geqslant1$, which is (24). ■

APPENDIX D
PROOF OF THEOREM 3

We prove Theorem 3 by evaluating the extra error term coming from the approximate time derivative in (10). In particular,

consider the Taylor's expansion of the gradient $\nabla_{\boldsymbol{x}} f(\boldsymbol{x}_k; t_{k-1})$ near the point $(\boldsymbol{x}_k, t_k)$ which is given by

$$\nabla_{\boldsymbol{x}} f(\boldsymbol{x}_k; t_{k-1}) = \nabla_{\boldsymbol{x}} f(\boldsymbol{x}_k; t_k) - h \nabla_{t\boldsymbol{x}} f(\boldsymbol{x}_k; t_k) + h^2/2 \nabla_{tt\boldsymbol{x}} f(\boldsymbol{x}_k; s). \quad (97)$$

for a particular $s \in [t_{k-1}, t_k]$. Regrouping the terms in (97) it follows that the partial mixed gradient $\nabla_{\boldsymbol{x}} f(\boldsymbol{x}_k; t_k)$ can be written as

$$\nabla_{t\boldsymbol{x}} f(\boldsymbol{x}_k; t_k) = \frac{\nabla_{\boldsymbol{x}} f(\boldsymbol{x}_k; t_k) - \nabla_{\boldsymbol{x}} f(\boldsymbol{x}_k; t_{k-1})}{h} + h/2 \nabla_{tt\boldsymbol{x}} f(\boldsymbol{x}_k; s). \quad (98)$$

Considering the definition of the approximate partial mixed gradient $\tilde{\nabla}_{t\boldsymbol{x}} f(\boldsymbol{x}_k; t_k)$ in (10) and the expression for the exact mixed gradient $\nabla_{\boldsymbol{x}} f(\boldsymbol{x}_k; t_k)$ in (98), we obtain that

$$\nabla_{t\boldsymbol{x}} f(\boldsymbol{x}_k; t_k) - \tilde{\nabla}_{t\boldsymbol{x}} f(\boldsymbol{x}_k; t_k) = \frac{h}{2} \nabla_{tt\boldsymbol{x}} f(\boldsymbol{x}_k; s). \quad (99)$$

Based on Assumption 2 the norm $\nabla_{tt\boldsymbol{x}} f(\boldsymbol{x}_k; s)$ is bounded above by C_3. Therefore, the error of the partial mixed gradient approximation is upper bounded by

$$\| \nabla_{t\boldsymbol{x}} f(\boldsymbol{x}_k; t_k) - \tilde{\nabla}_{t\boldsymbol{x}} f(\boldsymbol{x}_k; t_k) \| \leqslant \frac{hC_3}{2}. \quad (100)$$

Consider the approximate prediction step of the AGT algorithm in (11). By adding and subtracting the exact prediction direction $h[\nabla_{\boldsymbol{xx}} f(\boldsymbol{x}_k; t_k)]^{-1} \nabla_{t\boldsymbol{x}} f_k$ to the right-hand side of the update in (11) we obtain

$$\boldsymbol{x}_{k+1|k} = \boldsymbol{x}_k - h [\nabla_{\boldsymbol{xx}} f(\boldsymbol{x}_k; t_k)]^{-1} \nabla_{t\boldsymbol{x}} f(\boldsymbol{x}_k; t_k) + h [\nabla_{\boldsymbol{xx}} f(\boldsymbol{x}_k; t_k)]^{-1} \left(\nabla_{t\boldsymbol{x}} f(\boldsymbol{x}_k; t_k) - \tilde{\nabla}_{t\boldsymbol{x}} f(\boldsymbol{x}_k; t_k) \right). \quad (101)$$

Subtracting $\boldsymbol{x}^*(t_{k+1}) = \boldsymbol{x}^*(t_k) - h [\nabla_{\boldsymbol{xx}} f^*]^{-1} \nabla_{t\boldsymbol{x}} f^* + \boldsymbol{\Delta}_k$ in (52) from (101), and applying the triangle inequality lead to

$$\begin{aligned} \| \boldsymbol{x}_{k+1|k} - \boldsymbol{x}^*(t_{k+1}) \| &\leqslant \| \boldsymbol{x}_k - \boldsymbol{x}^*(t_k) \| \\ &+ h \| [\nabla_{\boldsymbol{xx}} f]^{-1} \nabla_{t\boldsymbol{x}} f - [\nabla_{\boldsymbol{xx}} f^*]^{-1} \nabla_{t\boldsymbol{x}} f^* \| + \| \boldsymbol{\Delta}_k \| \\ &+ h \| [\nabla_{\boldsymbol{xx}} f(\boldsymbol{x}_k; t_k)]^{-1} \left(\nabla_{t\boldsymbol{x}} f(\boldsymbol{x}_k; t_k) - \tilde{\nabla}_{t\boldsymbol{x}} f(\boldsymbol{x}_k; t_k) \right) \|. \end{aligned} \quad (102)$$

Observe the upper bound for the norm $\| \boldsymbol{\Delta}_k \|$ in (14). Further, observe that $\| [\nabla_{\boldsymbol{xx}} f(\boldsymbol{x}_k; t_k)]^{-1} (\nabla_{t\boldsymbol{x}} f(\boldsymbol{x}_k; t_k) - \tilde{\nabla}_{t\boldsymbol{x}} f(\boldsymbol{x}_k; t_k)) \|$ is bounded above by $C_3 h/2m$ according to (100) and Assumption 2. Substituting these upper bounds into (102) yields

$$\begin{aligned} \| \boldsymbol{x}_{k+1|k} - \boldsymbol{x}^*(t_{k+1}) \| &\leqslant \| \boldsymbol{x}_k - \boldsymbol{x}^*(t_k) \| \\ &+ \frac{h^2}{2} \left[\frac{C_0^2 C_1}{m^3} + \frac{2C_0 C_2}{m^2} + \frac{2C_3}{m} \right] \\ &+ h \| [\nabla_{\boldsymbol{xx}} f]^{-1} \nabla_{t\boldsymbol{x}} f - [\nabla_{\boldsymbol{xx}} f^*]^{-1} \nabla_{t\boldsymbol{x}} f^* \|. \end{aligned} \quad (103)$$

Observe that the inequality for the AGT algorithm in (103) is identical to the result for the GTT method in (54) except for the multiplier of h^2. This observation implies that by following the same steps from (55) to (74) we can prove the claim in (28). Likewise, if we redo the steps from (75) to (78), the claim in (30) can be followed from the result in (103). ■

APPENDIX E
PROOF OF THEOREM 4

The proof of Theorem 4 is based on the proof of Theorems 2 and 3. Since the correction step of NTT and ANT are identical, we can redo the steps from (80) to (87) to show that

$$\| \boldsymbol{x}_1 - \boldsymbol{x}^*(t_1) \| \leqslant Q^{-(2\tau-1)} \| \boldsymbol{x}_{1|0} - \boldsymbol{x}^*(t_1) \|^{2\tau}, \quad (104)$$

where $Q = 2m/C_1$. The prediction step of AGT and ANT are identical, therefore the result in (103) also holds true for ANT. Consider the result in (103) for $k = 0$. Using the inequality in (62) we can simplify the right-hand side of (103) as

$$\| \boldsymbol{x}_{1|0} - \boldsymbol{x}^*(t_1) \| \leqslant \sigma \| \boldsymbol{x}_0 - \boldsymbol{x}^*(t_0) \| + \frac{h^2}{2} \left[\frac{C_0^2 C_1}{m^3} + \frac{2C_0 C_2}{m^2} + \frac{2C_3}{m} \right], \quad (105)$$

where $\sigma = 1 + h(C_0 C_1/m^2 + C_2/m)$. Combining the inequalities in (104) and (105) and considering the definition of δ_2' in (31) lead to

$$\| \boldsymbol{x}_1 - \boldsymbol{x}^*(t_1) \| \leqslant Q^{-(2\tau-1)} \left(\sigma \| \boldsymbol{x}_0 - \boldsymbol{x}^*(t_0) \| + h^2 \delta_2' \right)^{2\tau}. \quad (106)$$

The result for ANT in (106) is similar to the result for NTT in (89). By following the steps from (90) to (96) the result in (34) follows. ■

REFERENCES

[1] A. Simonetto, A. Koppel, A. Mokhtari, G. Leus, and A. Ribeiro, "Prediction-correction methods for time-varying convex optimization," in *Proc. 49th Asilomar Conf. Signals, Syst., Comput.*, Nov. 2015, pp. 666–670.

[2] A. Koppel, A. Simonetto, A. Mokhtari, G. Leus, and A. Ribeiro, "Target tracking with dynamic convex optimization," in *Proc. IEEE Global Conf. Signal Inf. Process. (GlobalSIP)*, Dec. 2015, pp. 1210–1214.

[3] J.-H. Hours and C. N. Jones, "A parametric non-convex decomposition algorithm for real-time and distributed NMPC," IEEE *Trans. Autom. Control*, vol. 61, no. 2, pp. 287–302, 2016.

[4] C. Castillo, W. Moreno, and K. Valavanis, "Unmanned helicopter waypoint trajectory tracking using model predictive control," in *Proc. IEEE Mediterranean Conf. Control Autom. (MED)*, 2007, pp. 1–8.

[5] A. Vahidi and A. Eskandarian, "Research advances in intelligent collision avoidance and adaptive cruise control," *IEEE Trans. Intell. Transport. Syst.*, vol. 4, no. 3, pp. 143–153, 2003.

[6] F. Y. Jakubiec and A. Ribeiro, "D-MAP: Distributed maximum a posteriori probability estimation of dynamic systems," *IEEE Trans. Signal Process.*, vol. 61, no. 2, pp. 450–466, 2013.

[7] T. Ardeshiri, M. Norrlöf, J. Löfberg, and A. Hansson, "Convex optimization approach for time-optimal path tracking of robots with speed dependent constraints," in *Proc. 18th IFAC World Congr.*, Milano, Italy, Aug. 28–Sep. 2, 2011, pp. 14648–14653.

[8] D. Verscheure, B. Demeulenaere, J. Swevers, J. De Schutter, and M. Diehl, "Time-optimal path tracking for robots: A convex optimization approach," *IEEE Trans. Autom. Control*, vol. 54, no. 10, pp. 2318–2327, 2009.

[9] A. Koppel, G. Warnell, E. Stumpe, and A. Ribeiro, "D4l: Decentralized dynamic discriminative dictionary learning," *IEEE Trans. Signal Inf. Process.*, Jun. 2016, submitted for publication.

[10] A. Koppel, G. Warnell, E. Stump, and A. Ribeiro, "D4l: Decentralized dynamic discriminative dictionary learning," presented at the Int. Conf. Intell. Robots, Syst., Hamburg, Germany, Sept. 28–Oct. 2, 2015.

[11] A. Koppel, G. Warnell, and E. Stump, "Task-driven dictionary learning in distributed online settings," presented at the Asilomar Conf. Signals, Sys., Comput., Pacific Grove, CA, Nov. 8–11, 2015.
[12] A. L. Dontchev, M. I. Krastanov, R. T. Rockafellar, and V. M. Veliov, "An Euler-Newton continuation method for tracking solution trajectories of parametric variational inequalities," *SIAM J. Control Optim.*, vol. 51, no. 51, pp. 1823–1840, 2013.
[13] Y. Zhao and W. Lu, "Training neural networks with time-varying optimization," in *Proc. IEEE Int. Joint Conf. Neural Netw. (IJCNN'93)*, Nagoya, Japan, 1993, vol. 2, pp. 1693–1696.
[14] H. Myung and J.-H. Kim, "Time-varying two-phase optimization and its application to neural-network learning," *IEEE Trans. Neural Netw.*, vol. 8, no. 6, pp. 1293–1300, 1997.
[15] M. Baumann, C. Lageman, and U. Helmke, "Newton-type algorithms for time-varying pose estimation," in *Proc. IEEE Intell. Sensors, Sensor Netw., Inf. Process. Conf.*, 2004, pp. 155–160.
[16] M. Fazlyab, S. Paternain, V. M. Preciado, and A. Ribeiro, "Interior point method for dynamic constrained optimization in continuous time," 2015 DOI: arXiv preprint arXiv:1510.01396.
[17] S. Bittanti and F. Cuzzola, "A mixed gH_2 / H_∞ approach for stabilization and accurate trajectory tracking of unicycle-like vehicles," *Int. J. Control*, vol. 74, no. 9, pp. 880–888, 2001.
[18] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[19] A. Gupal, "Optimization method under nonstationary conditions," *Cybernetics*, vol. 10, no. 3, pp. 529–532, 1974.
[20] V. E. Kheisin, "Iterative minimization procedures with drift of the extremum" in Russian, *Avtomatika i Telemekhanika*, vol. 37, no. 11, pp. 1704–1713, 1976.
[21] E. Nurminskii, "A problem of nonstationary optimization," *Cybernetics*, vol. 13, no. 2, pp. 223–225, 1977.
[22] Y. M. Ermoliev and A. A. Gaivoronski, "Simultaneous Nonstationary Optimization, Estimation and Approximation Procedures," Tech. Rep., IIASA Collaborative Paper CP-82-016, 1982.
[23] B. T. Polyak, *Introduction to Optimization*. New York, NY, USA: Optimization Software, Inc., 1987.
[24] A. Y. Popkov, "Gradient methods for nonstationary unconstrained optimization problems" from Avtomatika i Telemekhanika, No. 6, 2005, pp. 38–46, *Autom. Remote Control*, vol. 66, no. 6, pp. 883–891, 2005.
[25] S. M. Robinson, "Strongly regular generalized equations," *Math. Oper. Res.*, vol. 5, no. 1, pp. 43–62, 1980.
[26] A. L. Dontchev and R. T. Rockafellar, *Implicit Functions and Solution Mappings*. New York, NY, USA: Springer, 2009.
[27] J. Guddat, F. G. Vazquez, and H. T. Jongen, *Parametric Optimization: Singularities, Pathfollowing and Jumps*. Chichester, U.K.: Wiley, 1990.
[28] V. M. Zavala and M. Anitescu, "Real-time nonlinear optimization as a generalized equation," *SIAM J. Control Optim.*, vol. 48, no. 8, pp. 5444–5467, 2010.
[29] Q. T. Dinh, C. Savorgnan, and M. Diehl, "Adjoint-based predictor-corrector sequential convex programming for parametric nonlinear optimization," *SIAM J. Optim.*, vol. 22, no. 4, pp. 1258–1284, 2012.
[30] M. Diehl, H. G. Bock, and J. P. Schlöder, "A real-time iteration scheme for nonlinear optimization in optimal feedback control," *SIAM J. Control Optim.*, vol. 43, no. 5, pp. 1714–1736, 2005.
[31] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation With Applications to Tracking and Navigation*. New York, NY, USA: Wiley Inter-Science, 2001.
[32] E. L. Allgower and K. Georg, *Numerical Continuation Methods: An Introduction*. New York, NY, USA: Springer-Verlag, 1990.
[33] A. H. Sayed, "Adaptation, learning, and optimization over networks," *Found. Trends Mach. Learn.*, vol. 7, no. 4–5, pp. 311–801, 2014.
[34] X. Zhao and A. H. Sayed, "Asynchronous adaptation and learning over networks," *IEEE Trans. Signal Process.*, vol. 63, no. 4 1–3, pp. 811–858, 2015.
[35] A. Quarteroni, R. Sacco, and F. Saleri, *Numerical Mathematics*. New York, NY, USA: Springer, 2000.
[36] S. K. Lee, "Compact finite difference schemes with spectral-like resolution," *J. Comput. Phys.*, vol. 103, no. 1, pp. 16–42, 1992.
[37] Q. Ling and A. Ribeiro, "Decentralized dynamic optimization through the alternating direction method of multipliers," *IEEE Trans. Signal Process.*, vol. 62, no. 5, pp. 1185–1197, 2014.
[38] E. K. Ryu and S. Boyd, "Primer on monotone operator methods," *Appl. Comput. Math.*, no. 15, pp. 3–43, 2016.

Andrea Simonetto (M'12) received the Ph.D. degree in systems and control from Delft University of Technology, Delft, The Netherlands, in 2012. He is currently a Post-Doctoral Researcher with the ICTEAM institute, at Université Catholique de Louvain, Belgium. He was a Post-Doctoral Researcher with the Electrical Engineering Department, at Delft University of Technology. His current research interests include distributed estimation, control, and optimization.

Aryan Mokhtari received the B.Eng. degree in electrical engineering from Sharif University of Technology, Tehran, Iran, in 2011, and the M.S. degree in electrical engineering from the University of Pennsylvania, Philadelphia, PA, in 2014. Since 2012, he has been working towards the Ph.D. degree in the Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA. His research interests include stochastic optimization, machine learning and distributed optimization.

Alec Koppel completed his B.A. in mathematics and M.S. in systems science at Washington University in St. Louis, MO. He is currently a Ph.D. student in the Department of Electrical and Systems Engineering at the University of Pennsylvania and a participant in the SMART Scholarship program sponsored by the U.S. Army Research Laboratory in Adelphi, MD. His research focuses on signal processing, machine learning, optimization, and robotics.

Geert Leus (F'12) received the M.Sc. and Ph.D. degree in applied sciences from the Katholieke Universiteit Leuven, Belgium, in June 1996 and May 2000, respectively. Currently, he is a Full Professor at the Faculty of Electrical Engineering, Mathematics and Computer Science of the Delft University of Technology, The Netherlands. He received a 2002 IEEE Signal Processing Society Young Author Best Paper Award and a 2005 IEEE Signal Processing Society Best Paper Award. He is a Fellow of EURASIP. He was the Chair of the IEEE Signal Processing for Communications and Networking Technical Committee. Currently, he is a Member-at-Large to the Board of Governors of the IEEE Signal Processing Society and he serves as the Editor in Chief of the *EURASIP Journal on Advances in Signal Processing*.

Alejandro Ribeiro (M'07) received the B.Sc. degree in electrical engineering from the Universidad de la Republica Oriental del Uruguay, Montevideo, in 1998 and the M.Sc. and Ph.D. degree in electrical engineering from the Department of Electrical and Computer Engineering, the University of Minnesota, Minneapolis in 2005 and 2007. From 1998 to 2003, he was a member of the technical staff at Bellsouth Montevideo. After his M.Sc. and Ph.D. studies, in 2008 he joined the University of Pennsylvania (Penn), Philadelphia, where he is currently the Rosenbluth Associate Professor at the Department of Electrical and Systems Engineering.