# Prediction-Correction Methods for Time-Varying Convex Optimization

Andrea Simonetto[§], Alec Koppel[⋆], Aryan Mokhtari[⋆], Geert Leus[§], and Alejandro Ribeiro[⋆]

[§]Dept. of EEMCS, Delft University of Technology, 2826 CD Delft, The Netherlands
{a.simonetto, g.j.t.leus}@tudelft.nl
[⋆]Dept. of ESE, University of Pennsylvania, 200 South 33rd Street, Philadelphia, PA 19104, USA
{akoppel, aryanm, aribeiro}@seas.upenn.edu

***Abstract*—We consider unconstrained convex optimization problems with objective functions that vary continuously in time. 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 $1/h$. The prediction step is derived by analyzing the iso-residual dynamics of the optimality conditions, while the correction step consists either of one or multiple gradient steps or Newton's 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. Numerical simulations demonstrate the practical utility of the proposed 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

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} f(\boldsymbol{x};t), \quad \text{for } t \geqslant 0. \tag{1}$$

Our goal is to determine the solution $\boldsymbol{x}^*(t)$ of (1) for each time $t$ which corresponds to the solution *trajectory*. The time-varying optimization problems of the form (1) arise in optimal control problems [1], signal processing problems [2], and countless others [3], [4].

With large enough computational resources, one could sample the objective functions $f(\boldsymbol{x};t)$ at time instants $t_k$ with $k = 0, 1, 2, \ldots$, and sampling interval $h = t_k - t_{k-1}$, arbitrarily close to each other and then solve the resulting time-invariant problems

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} f(\boldsymbol{x};t_k). \tag{2}$$

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 [5]. In addition, the idea of solving each instance of the problem up to an arbitrary accuracy does not lead to quantifiable real-time bounds. That is, it is challenging to reasonably bound the time each problem instance will take to be solved [6]. 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 [7]–[10] [3, Chapter 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 [11] designs a gradient method for unconstrained optimization problems using an arbitrary starting point, which achieves a $O(h)$ asymptotic error bound. 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 [12]–[14] and references therein. Tracking algorithms for optimization problems with parameters that change in time are given in [4], [15] and are based on predictor-corrector schemes. Even though these algorithms are applicable to constrained problems, an important assumption is the availability of an initial solution $\boldsymbol{x}^*(t_0)$, which is often hard to compute in practice. Some of the theoretical advances in these works have been used to ease the computational burden of sequential convex programming while solving nonconvex optimization problems, or nonlinear model predictive control [1], [16], [17].

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. In particular, the methods proposed here yield a sequence of approximate time-varying optimizers $\{\boldsymbol{x}_k\}$, for which

$$\lim_{k\to\infty} \|\boldsymbol{x}_k - \boldsymbol{x}^*(t_k)\| \leqslant \delta \tag{3}$$

with $\delta$ dependent on the sampling interval $h$. 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 $O(h^2)$,

This research was 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.

which outperforms the $O(h)$ error bound achieved by the state-of-the-art method of [11].

## II. Algorithm Development

In this section we introduce a class of algorithms for solving optimization problem (2) 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 considering both how the solution changes in time and how far our current update is from optimality at time $t_k$.

### A. Gradient Trajectory Tracking

In this paper, we consider 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, much like a Kalman filter strategy in estimation theory. 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.

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

$$\nabla_{\boldsymbol{x}} f(\boldsymbol{x}^*(t); t) = \boldsymbol{0}, \tag{4}$$

for each $t$. These two problems are equivalent since the objective functions $f(\boldsymbol{x}^*(t); t)$ are strongly convex with respect to $\boldsymbol{x}$ and only their optimal solutions satisfy the condition in (4).

Consider an arbitrary vector $\boldsymbol{x} \in \mathbb{R}^n$ (for example the approximate solution $\boldsymbol{x}_k$). The objective function gradient $\nabla_{\boldsymbol{x}} f(\boldsymbol{x}; t)$ computed at point $\boldsymbol{x}$ is

$$\nabla_{\boldsymbol{x}} f(\boldsymbol{x}; t) = \boldsymbol{r}(t) \tag{5}$$

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, which is tantamount to predicting $\boldsymbol{x}_k$ such that we stay close to the iso-residual manifold. Define $\nabla_{\boldsymbol{xx}} f(\boldsymbol{x}; t)$ as the partial Hessian of the objective function $f(\boldsymbol{x}; t)$ with respect to $\boldsymbol{x}$, and $\nabla_{t\boldsymbol{x}} f(\boldsymbol{x}; t)$ as the mixed partial derivative of the objective function $f(\boldsymbol{x}; t)$. We aim to maintain the evolution of the trajectory close to the residual vector $\boldsymbol{r}(t)$, i.e.

$$\nabla_{\boldsymbol{x}} f(\boldsymbol{x}; t) + \nabla_{\boldsymbol{xx}} f(\boldsymbol{x}; t)\delta\boldsymbol{x} + \nabla_{t\boldsymbol{x}} f(\boldsymbol{x}; t)\delta t = \boldsymbol{r}(t), \tag{6}$$

where $\delta\boldsymbol{x}$ and $\delta t$ are the variations of the decision variable $\boldsymbol{x}$ and the time variable $t$, respectively. By subtracting (5) from (6) and dividing the resulting equation by the time variation $\delta t$, we obtain the continuous dynamical system

$$\dot{\boldsymbol{x}} = -[\nabla_{\boldsymbol{xx}} f(\boldsymbol{x}; t)]^{-1} \nabla_{t\boldsymbol{x}} f(\boldsymbol{x}; t), \tag{7}$$

where $\dot{\boldsymbol{x}} = \delta\boldsymbol{x}/\delta t$. We consider the discrete time approximation of (7), 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 (7) by using an Euler scheme. Let

**Algorithm 1** Gradient trajectory tracking (GTT)

**Require:** Initial variable $\boldsymbol{x}_0$. Initial objective function $f(\boldsymbol{x}; t_0)$

1: **for** $k = 0, 1, 2, \ldots$ **do**
2: Predict the solution using the prior information [cf (8)]

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

3: Acquire the updated function $f(\boldsymbol{x}; t_{k+1})$
4: Initialize the sequence of corrected variables $\hat{\boldsymbol{x}}^0_{k+1} = \boldsymbol{x}_{k+1|k}$
5: **for** $s = 0 : \tau - 1$ **do**
6: Correct the variable by the projected gradient step [cf (9)]

$$\hat{\boldsymbol{x}}^{s+1}_{k+1} = P_X\left[\hat{\boldsymbol{x}}^s_{k+1} - \gamma \nabla_{\boldsymbol{x}} f(\hat{\boldsymbol{x}}^s_{k+1}; t_{k+1})\right]$$

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

$\boldsymbol{x}_{k+1|k}$ be the predicted decision variable based on the available information up to time $t$, then we may write the Euler integral approximation of (7) as

$$\boldsymbol{x}_{k+1|k} = \boldsymbol{x}_k - [\nabla_{\boldsymbol{xx}} f(\boldsymbol{x}_k; t_k)]^{-1} \nabla_{t\boldsymbol{x}} f(\boldsymbol{x}_k; t_k)\, h. \tag{8}$$

Observe that the prediction step in (8) is computed by only incorporating information available at time $t$; 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 $t_{k+1}$. We discuss how to enforce this property next.

The gradient trajectory tracking (GTT) algorithm uses the projected gradient descent method to correct the predicted decision variable $\boldsymbol{x}_{k+1|k}$ so that it approximately satisfies the iso-residual condition (6). To do so, we modify the predicted variable $\boldsymbol{x}_{k+1|k}$ towards the optimal argument of the objective function at time $t_{k+1}$. Therefore, the correction step of GTT requires the execution of the projected gradient descent method based on the updated objective $f(\boldsymbol{x}; t_{k+1})$. The number of projected gradient descent steps that can be afforded until sampling the next function depends on the sampling rate $h$.

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

$$\hat{\boldsymbol{x}}^{s+1}_{k+1} = P_X[\hat{\boldsymbol{x}}^s_{k+1} - \gamma \nabla_{\boldsymbol{x}} f(\hat{\boldsymbol{x}}^s_{k+1}; t_{k+1})], \tag{9}$$

where $P_X$ denotes the Euclidean projection operator onto the set $X$ and $\gamma > 0$ is the stepsize. After executing $\tau$ steps of (9) the GTT algorithm yields the decision variable $\boldsymbol{x}(t_{k+1}) := \boldsymbol{x}_{k+1} = \hat{\boldsymbol{x}}^\tau_{k+1}$ at time $t_{k+1}$.

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 optimal solution gradient $\nabla_{\boldsymbol{x}} f(\boldsymbol{x}^*(t_k); t)$ at time $t_k$ [cf. (8)]. Then we correct the predicted solution by executing $\tau$ projected gradient descent steps as stated in (9) 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 $\hat{\boldsymbol{x}}^0_{k+1} = \boldsymbol{x}_{k+1|k}$ in Step 4 and the output of the recursion is considered as the updated variable $\boldsymbol{x}_{k+1} = \hat{\boldsymbol{x}}^\tau_{k+1}$ in Step 8. The implementation

**Algorithm 2** Newton trajectory tracking (NTT)

**Require:** Initial variable $\boldsymbol{x}_0$. Initial objective function $f(\boldsymbol{x}; t_0)$

1: **for** $k = 0, 1, 2, \ldots$ **do**
2: Predict the solution using the prior information [cf (8)]

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

3: Acquire the updated function $f(\boldsymbol{x}; t_{k+1})$
4: Initialize the sequence of corrected variables $\hat{\boldsymbol{x}}^0_{k+1} = \boldsymbol{x}_{k+1|k}$
5: **for** $s = 0 : \tau - 1$ **do**
6: Correct the variable by the projected Newton step [cf (9)]

$$\hat{\boldsymbol{x}}^{s+1}_{k+1} = P_X \left[\hat{\boldsymbol{x}}^s_{k+1} - \nabla_{\boldsymbol{xx}} f(\hat{\boldsymbol{x}}^s_{k+1}; t_{k+1})^{-1} \times \nabla_{\boldsymbol{x}} f(\hat{\boldsymbol{x}}^s_{k+1}; t_{k+1})\right]$$

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

of projected 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 $\tau$ projected gradient descent steps which only uses first-order information. We accelerate this procedure using second-order information in the following subsection.

### *B. Newton trajectory tracking*

The GTT prediction step introduced in (8) requires computation of the partial Hessian inverse $[\nabla_{\boldsymbol{xx}} f(\boldsymbol{x}_k; t_k)]^{-1}$ which has a computational complexity of order $O(n^3)$. This observation implies that the partial Hessian inverse of the objective is available. Moreover, the computational complexity of the Hessian inverse is affordable. These two observations justify using Newton's method for the correction 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 the both prediction and correction steps.

The prediction step of the NTT algorithm is identical to the prediction step of the GTT method as introduced in (8); however, in the correction steps NTT updates the predicted solution trajectory by applying $\tau'$ steps of the Newton method. In particular, the predicted variable $\boldsymbol{x}_{k+1|k}$ in (8) is used for initializing the sequence of corrected variables $\hat{\boldsymbol{x}}^s_{k+1}$, i.e., $\hat{\boldsymbol{x}}^0_{k+1} := \boldsymbol{x}_{k+1|k}$. The sequence of corrected variables $\hat{\boldsymbol{x}}^s_{k+1}$ is updated using projected Newton steps as

$$\hat{\boldsymbol{x}}^{s+1}_{k+1} = P_X \left[\hat{\boldsymbol{x}}^s_{k+1} - \nabla_{\boldsymbol{xx}} f(\hat{\boldsymbol{x}}^s_{k+1}; t_{k+1})^{-1} \nabla_{\boldsymbol{x}} f(\hat{\boldsymbol{x}}^s_{k+1}; t_{k+1})\right]. \tag{10}$$

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 $\tau'$ iterations of (10) such that $\boldsymbol{x}_{k+1} = \hat{\boldsymbol{x}}^{\tau'}_{k+1}$.

*Remark 1:* 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 gradient descent step requires a computational complexity of order $O(n)$. Since the sampling increment is a fixed value, the number of Newton iterations $\tau'$ in one iteration of the NTT algorithm is smaller than the number of gradient descent steps $\tau$ that we can use in the correction step of GTT. On the other hand, Newton's 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.

## 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 interval. All proofs are given in [18]. In order to do so, some technical conditions are required which we state below.

*Assumption 1:* The solution trajectory $\boldsymbol{x}^*(t)$ of (1) is contained in the interior of a convex set $X \subseteq \mathbb{R}^n$ for each $t$.

*Assumption 2:* The function $f(\boldsymbol{x}; t)$ is twice differentiable and $m$-strongly convex in $\boldsymbol{x} \in X$ and uniformly in $t$, which allows the Hessian of $f(\boldsymbol{x}; t)$ with respect to $\boldsymbol{x}$ to be bounded below as,

$$\nabla_{\boldsymbol{xx}} f(\boldsymbol{x}; t) \succeq m\mathbf{I}, \quad \forall\ \boldsymbol{x} \in X, t. \tag{11}$$

*Assumption 3:* The function $f(\boldsymbol{x}; t)$ for all $\boldsymbol{x} \in X$ and $t$ has bounded second and third derivatives with respect to $\boldsymbol{x} \in X$ and $t$ with constants

$$\|\nabla_{\boldsymbol{xx}} f(\boldsymbol{x}; t)\| \leqslant L,\ \|\nabla_{t\boldsymbol{x}} f(\boldsymbol{x}; t)\| \leqslant C_0,\ \|\nabla_{\boldsymbol{xxx}} f(x; t)\| \leqslant C_1$$
$$\|\nabla_{\boldsymbol{xtx}} f(\boldsymbol{x}; t)\| \leqslant C_2, \quad \|\nabla_{tt\boldsymbol{x}} f(\boldsymbol{x}; t)\| \leqslant C_3. \tag{12}$$

Assumption 1 is easily satisfied: take $X$ to be $\mathbb{R}^n$, then we require the existence of a solution for (1) at each time $t$. However, it is very useful in practice, when we know a priori that the solution trajectory has to be, for instance, positive. Then, we may use more tailored algorithms and may relax Assumptions 2 and 3 and take $X = \mathbb{R}^n_+$. Assumption 2, besides guaranteeing that Problem (1) is convex and has a *unique* solution for each time instance, ensures that the Hessian of the objective function $f(\boldsymbol{x}; t)$ is invertible. The solution uniqueness at each time instance implies that the solution trajectory is unique, and is often required in time-varying settings [2], [4], [11], [19]. Assumption 3 gives to the time-varying problem the boundedness required to ensure solution tracking (a similar assumption is required in [4]). With unbounded derivatives, little may be said about how the approximate solution computed at $t_k$ would differ from the one computed at $t_{k+1}$.

Under Assumptions 1, 2, and a relaxed version of Assumption 3, one may establish that the solution *mapping* $t \mapsto \boldsymbol{x}^*(t)$ is one-to-one and does not vary arbitrarily in time, stated as

$$\|\boldsymbol{x}^*(t_{k+1}) - \boldsymbol{x}^*(t_k)\| \leqslant \frac{1}{m} \|\nabla_{t\boldsymbol{x}} f(\boldsymbol{x}; t)\| (t_{k+1} - t_k) \leqslant \frac{C_0 h}{m}, \tag{13}$$

as presented in [13, Theorem 2F.10]. This property allows GTT and NTT to converge to a neighborhood of the optimal solution.

The prediction steps of both GTT and NTT in (8) are the approximations of the first-order forward Euler integral in (8). The error of this approximation, $\boldsymbol{\Delta}$, is upper bounded as in the following proposition.

*Proposition 1:* Under Assumptions 1-3, the error norm $\|\boldsymbol{\Delta}\|$ of the Euler approximation (8) is upper bounded by

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

Proposition 1 states that the Euler error norm $\|\boldsymbol{\Delta}\|$ is bounded above by a constant which is in the order of $O(h^2)$. We use this upper bound in proving convergence of the proposed methods. We study the convergence properties of the sequence of variables $\boldsymbol{x}_k$ generated by GTT for different choices of stepsizes in the following theorem.

*Theorem 1:* Denote the gradient trajectory tracking algorithm generated by (8)-(9) as $\{\boldsymbol{x}_k\}$. Let Assumptions 1-3 hold and define the constants $\rho$ and $\sigma$ as $\rho = (1 + \gamma^2 L^2 - \gamma m)^{1/2}$ and $\sigma = 1 + h(C_0 C_1/m^2 + C_2/m)$.

i) For any sampling increment $h$, if the stepsize satisfies $\gamma < m/L^2$ which implies $\rho < 1$, the sequence $\{\boldsymbol{x}_k\}$ converges to $\boldsymbol{x}^*(t_k)$ Q-linearly up to a bounded error as

$$\begin{aligned}\|\boldsymbol{x}_k - \boldsymbol{x}^*(t_k)\| &\leqslant \rho^{\tau k}\|\boldsymbol{x}_0 - \boldsymbol{x}^*(t_0)\| \\ &+ \rho^{\tau}\left[h\left[\frac{2C_0 L}{m^2} + \frac{2C_0}{m}\right] + h^2\left[\frac{C_0 C_2}{2m^2} + \frac{C_3}{2m}\right]\right]\left[\frac{1-\rho^{\tau k}}{1-\rho^{\tau}}\right].\end{aligned} \tag{15}$$

ii) If the sampling increment $h$ and the stepsize $\gamma > 0$ are chosen such that the condition $\rho^{\tau}\sigma < 1$ is satisfied, then the sequence $\{\boldsymbol{x}_k\}$ converges to $\boldsymbol{x}^*(t_k)$ Q-linearly up to a bounded error as,

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

Theorem 1 states the convergence properties of the GTT algorithm for different choices of the parameters. In both cases the linear convergence to a neighborhood is shown, however, the accuracy of convergence depends on the choice of the sampling increment $h$, the stepsize parameter $\gamma$, and the number of projected gradient descent steps $\tau$. To guarantee that the constant $\rho$ is strictly smaller than 1, the stepsize must satisfy $\gamma < m/L^2$. Then, for any choice of the sampling increment $h$ the result in (15) holds, which implies linear 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 projected gradient descent iterations $\tau$ improves the speed of linear convergence by decreasing the factor $\rho^{\tau}$. Moreover, a larger choice of $\tau$ leads to a better accuracy since the asymptotic error bound is proportional to $\rho^{\tau}/(1-\sigma\rho^{\tau})$.

The result in (16) shows that the accuracy of convergence is in the order of $O(h^2)$ when the stepsize $\gamma$ is smaller than a threshold. Observe that the error bound in (16) is strictly smaller than the error bound in (15), since it does not contain the term which is proportional to $h$. This more accurate convergence result is achieved at the cost of choosing a smaller stepsize. To be more precise, the constant $\sigma$ in Theorem 1 is strictly greater than 1. To satisfy the condition in Theorem 1, which is equivalent to $\rho^{\tau} < 1/\sigma < 1$, the stepsize must be chosen as

$$\gamma < \frac{m + \left(m^2 - \left(\frac{\sigma^{2/\tau}-1}{\sigma^{2/\tau}}\right)L^2\right)^{1/2}}{2L^2} < m/L^2. \tag{17}$$

As it is shown in (17), to satisfy the requirement of the result in (16), the stepsize should be smaller relative to the required stepsize for the result in (15).

Notice that the GTT algorithm does not incorporate the second-order information of the update 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 the following theorem we show that when the initial estimate $\boldsymbol{x}_0$ is close enough to the initial solution $\boldsymbol{x}^*(t_0)$, NTT yields a more accurate convergence than that of GTT.

*Theorem 2:* Consider the Newton trajectory tracking algorithm generated by (8) and (10). Recall the definition of $\sigma$ in Theorem 1 and assume that all the conditions in Assumptions 1-3 hold. Assume that the initial optimality gap $\|\boldsymbol{x}_0 - \boldsymbol{x}^*(t_0)\|$ can be written as $\|\boldsymbol{x}_0 - \boldsymbol{x}^*(t_0)\| = \beta\|\boldsymbol{\Delta}\|$ where $\beta > 0$ is a constant. Further, assume that the sampling increment $h$ is small enough that the upper bound for the discretization error norm $\|\boldsymbol{\Delta}\|$ in (1) is bounded above by 1, i.e., $\|\boldsymbol{\Delta}\| \leqslant 1$.

i) If the condition $C_1\|\boldsymbol{\Delta}\|/2m \leqslant \beta/(\sigma\beta+1)^2$ is satisfied, then the sequence $\boldsymbol{x}_k$ generated by NTT converges as

$$\|\boldsymbol{x}_k - \boldsymbol{x}^*(t_k)\| \leqslant \beta\|\boldsymbol{\Delta}\| = O(h^2); \tag{18}$$

ii) Further, if $C_1/2m \leqslant \beta/(\sigma\beta+1)^2$ holds, then the sequence $\boldsymbol{x}_k$ satisfies

$$\|\boldsymbol{x}_k - \boldsymbol{x}^*(t_k)\| \leqslant \beta\|\boldsymbol{\Delta}\|^2 = O(h^4). \tag{19}$$

Theorem 1 establishes that the NTT tracks the optimal trajectory $\boldsymbol{x}^*(t_k)$ up to an error bound not larger than $O(h^2)$, where $h$ is the sampling rate, which is comparable to the result of the gradient-based tracking algorithm. Moreover, when $C_1/2m \leqslant \frac{\beta}{(\sigma\beta+1)^2}$, then NTT achieves an error bound at worst $O(h^4)$, as a result of the quadratic phase of Newton's method. One case where this condition is satisfied is for quadratic cost functions, for which $C_1 = 0$, yielding perfect convergence.

## IV. Numerical Analysis

We empirically verify the properties established in Section III by running Algorithms 1 and 2 on a problem where $x \in X \subset \mathbb{R}$ is scalar-valued and the objective is given by

$$f(x; t) = \frac{1}{2}(x - \cos(\omega t))^2 + \frac{\kappa}{2}\sin^2(\omega t)\exp(\mu x^2). \tag{20}$$

Here $\omega$, $\kappa$, and $\mu$ are nonnegative scalars which we set to $\omega = 0.02\,\pi$, $\kappa = 0.1$, and $\mu = 0.5$ in the subsequent experiments. The function $f(x; t)$ in (20) satisfies the Assumptions 1-3. By setting the convex set of feasible points to $X = [-1.1, 1.1]$, the required conditions are satisfied for constants $m = 1$, $L = 1.2024$, $C_0 = 0.0755$, $C_1 = 0.4240$, $C_2 = 0.0254$, $C_3 = 0.0047$. We select a constant stepsize $\gamma = 0.1$ in the projected gradient method and initialize $x$ as $x_0 = 0$.

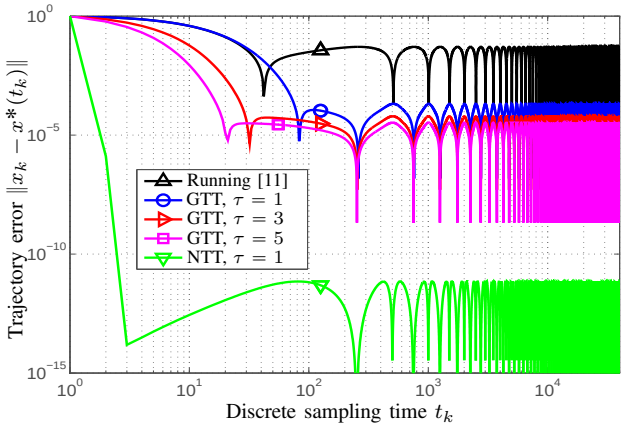


Fig. 1. Error $\|x_k - x^*(t_k)\|$ with respect to the sampled time $t_k$ for different algorithms applied to (20). GTT outperforms the running method by several orders of magnitude, with its advantage increasing with increasing $\tau$. Moreover, NTT with $\tau = 1$ outperforms the gradient tracking and GTT algorithms in terms of error to the optimal trajectory by near 5 orders of magnitude.

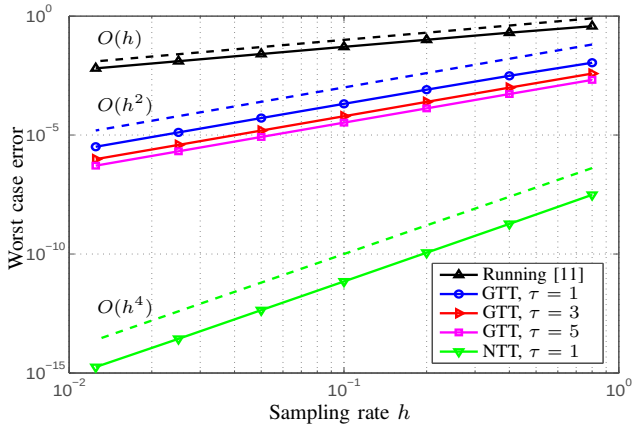


Fig. 2. The worst case error $\max_{k>\bar{k}}\{\|x_k - x^*(t_k)\|\}$ with respect to the sampling increment $h$ after a fixed sampling iteration $\bar{k} = 10^4$. The trends observed in Figure 1 are corroborated here: GTT outperforms the running method, while NTT achieves the smallest worst case error by 10 orders of magnitude.

In Figure 1, we plot the error $\|x_k - x^*(t_k)\|$ versus the discrete time $t_k$ for a sampling interval of $h = 0.1$, for different schemes. Observe that the running method [11] which uses only a gradient correction step 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 $\tau$ has a better error performance than running; however, tracking with Newton-based prediction (NTT) with $\tau = 1$ achieves a superior performance compared to the others, i.e., an error stabilizing near $10^{-12}$ is achieved.

The differences in performance can be also appreciated by varying $h$ and observing the worst case error floor size which is $\max_{k>\bar{k}}\{\|x_k - x^*(t_k)\|\}$, where $\bar{k} = 10^4$ in the simulations. Figure 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 Figure 1. In particular, the running method achieves the largest worst case error bound, followed in descending order by GTT with increasing $\tau$, and lastly NTT, which achieves the minimal worst-case error bound.

## V. Conclusion

We designed algorithms to track the solution of time-varying unconstrained and strongly convex optimization problems. They leverage on the knowledge of how the cost function changes in time and are based on a predictor-corrector scheme. Convergence analysis of the GTT algorithm shows a convergence accuracy of the order of $O(h^2)$ for a properly chosen stepsize. Further, the NTT algorithm improves this accuracy to $O(h^4)$ by incorporating second-order information in the correction step. We have proved the proposed methods gain in performance w.r.t. a state-of-the-art running algorithm, that only performs correction steps.

## References


[1] J.-H. Hours and C. N. Jones, "A Parametric Non-Convex Decomposition Algorithm for Real-Time and Distributed NMPC," 2014. arXiv:1408.5120.

[2] F. Y. Jakubiec and A. Ribeiro, "D-MAP: Distributed Maximum a Posteriori Probability Estimation of Dynamic Systems," *IEEE Transactions on Signal Processing*, vol. 61, no. 2, pp. 450 – 466, 2013.

[3] B. T. Polyak, *Introduction to Optimization*. Optimization Software, Inc., 1987.

[4] 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 Journal of Control and Optimization*, vol. 51, no. 51, pp. 1823 – 1840, 2013.

[5] S. Bittanti and F. Cuzzola, "A Mixed g$H_2/H_\infty$ Approach for Stabilization and Accurate Trajectory Tracking of Unicycle-like Vehicles," *Internation Journal of Control*, vol. 74, no. 9, pp. 880 – 888, 2001.

[6] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[7] A. Gupal, "Optimization Method Under Nonstationary Conditions," *Cybernetics*, vol. 10, no. 3, pp. 529 – 532, 1974.

[8] V. E. Kheisin, "Iterative Minimization Procedures with Drift of the Extremum," *Avtomatika i Telemekhanika*, vol. 37, no. 11, pp. 1704 – 1713, 1976. (in Russian).

[9] E. Nurminskii, "A Problem of Nonstationary Optimization," *Cybernetics*, vol. 13, no. 2, pp. 223 – 225, 1977.

[10] Y. M. Ermoliev and A. A. Gaivoronski, "Simultaneous Nonstationary Optimization, Estimation and Approximation Procedures," tech. rep., IIASA Collaborative Paper CP-82-016, 1982.

[11] A. Y. Popkov, "Gradient Methods for Nonstationary Unconstrained Optimization Problems," *Automation and Remote Control*, vol. 66, no. 6, pp. 883 – 891, 2005. Translated from Avtomatika i Telemekhanika, No. 6, 2005, pp. 38 – 46.

[12] S. M. Robinson, "Strongly Regular Generalized Equations," *Mathematics of Operations Research*, vol. 5, no. 1, pp. 43 – 62, 1980.

[13] A. L. Dontchev and R. T. Rockafellar, *Implicit Functions and Solution Mappings*. Springer, 2009.

[14] J. Guddat and F. Guerra Vazquez and H. T. Jongen, *Parametric Optimization: Singularities, Pathfollowing and Jumps*. John Wiley & Sons, Chichester, UK, 1990.

[15] V. M. Zavala and M. Anitescu, "Real-Time Nonlinear Optimization as a Generalized Equation," *SIAM Journal of Control and Optimization*, vol. 48, no. 8, pp. 5444 – 5467, 2010.

[16] Q. T. Dinh, C. Savorgnan, and M. Diehl, "Adjoint-Based Predictor-Corrector Sequential Convex Programming for Parametric Nonlinear Optimization," *SIAM Journal on Optimization*, vol. 22, no. 4, pp. 1258 – 1284, 2012.

[17] M. Diehl, H. G. Bock, and J. P. Schlöder, "A Real-Time Iteration Scheme for Nonlinear Optimization in Optimal Feedback Control," *SIAM Journal on Control and Optimization*, vol. 43, no. 5, pp. 1714 – 1736, 2005.

[18] A. Simonetto, A. Koppel, A. Mokhtari, G. Leus, and A. Ribeiro, "A Class of Prediction-Correction Methods for Time-Varying Convex Optimization," *In preparation*, 2015. Available at http://ens.ewi.tudelft.nl/~asimonetto/TimeVarying_part1.pdf.

[19] Q. Ling and A. Ribeiro, "Decentralized Dynamic Optimization Through the Alternating Direction Method of Multipliers," *IEEE Transactions on Signal Processing*, vol. 62, no. 5, pp. 1185 – 1197, 2014.