Online Nonstochastic Control vs. Retrospective Cost Adaptive Control

Usman Syed, Yingying Li and Bin Hu

Abstract— **Recently, online optimization methods have been leveraged to develop the online nonstochastic control framework which is capable of learning online gradient perturbation controllers in the presence of nonstochastic adversarial disturbances. Interestingly, using online optimization for adapting controllers in the presence of unknown disturbances is not a completely new idea, and a similar algorithmic framework called Retrospective Cost Adaptive Control (RCAC) has already appeared in the controls literature in 2000s. In this paper, we present the connections between online nonstochastic control and RCAC, and discuss the different strengths of both approaches: i.e., RCAC is able to stabilize unknown unstable plants via the use of target model, while online nonstochastic control enjoys provably near optimal regret bounds given a stabilizing policy a priori. We further propose an integration of these two approaches. We hope that our insights will help the development of new algorithms that complement the two approaches.**

Online non-stochastic control, Retrospective Cost Adaptive Control (RCAC), Online learning

I. Introduction

Over the past decade, online optimization/learning techniques have achieved great success in numerous sequential decision making tasks including online portfolio selection, advertisement placement, and web ranking [1]–[3]. These methods take full advantage of the available streaming data, and use regret as a metric for balancing between exploration and exploitation in the face of uncertainty. Recently, online optimization methods have been leveraged to develop the online nonstochastic control framework [4]–[11] which is capable of learning online gradient perturbation controllers (GPC) in the presence of unknown nonstochastic disturbances. For many control applications, the disturbance information is not fully known at the control design stage, and such a regret-based online control framework can provide unique benefits in addressing the trade-offs between disturbance learning (exploration) and system control (exploitation). This is in contrast to H_2 optimal control [12], [13], which makes optimistic stochastic assumptions about the disturbances, or H_∞ optimal control [14], [15], which makes pessimistic worst-case assumptions. The connections between online optimization and nonstochastic control have led to promising developments in addressing nonstochastic disturbances that are not known a priori but are learnable in real time.

Interestingly, another online algorithmic framework, Retrospective Cost Adaptive Control (RCAC), which has been developed from the control community since 2000s, is based on a very similar idea [16]–[19]. Specifically, RCAC is based on the idea of "retrospectively optimized control" [16], [17], and recursively optimizes the policy parameters in a way that the control performance over the previous window of operation would have been improved if the re-optimized policy had been used over that window. Despite the conceptual similarities, the properties and strengths of RCAC and online nonstochastic control can be vastly different. In this paper, we investigate the promise of connecting and combining RCAC and non-stochastic control. Our paper aims at bridging this gap via examining the connections and differences between RCAC and online nonstochastic control.

Our discussions will be based on two key observations. First, we observe that online nonstochastic control typically requires either open-loop stable systems or previously known stabilizing controllers for output feedback systems [4], [5], [8]. Given stabilizing controllers, online nonstochastic control can achieve strong theoretical guarantees in terms of provably sublinear regret bounds. However, stabilizing unknown linear systems is not a simple task by itself.[1] In the setting where there lacks an output-feedback stabilizing controller in the first place, the online nonstochastic control framework may not be directly applicable. In contrast, via a novel use of target models, RCAC is able to stabilize unknown linear output-feedback systems in many practical settings without requiring a pre-stabilizing policy. This demonstrates that the stability issue is not an inherent weakness of online-optimization-based design philosophy. Second, we observe that it is quite difficult to establish sublinear regret bound for RCAC, and directly adapting the theoretical arguments from online nonstochasic control does not work. The practical benefits of RCAC come from the use of larger policy classes, which causes significant difficulties for theoretical analysis. In light of the above observations, our key contributions are summarized as follows:

1) We discuss the similarities and differences between online nonstochastic control (or equivalently GPC) and RCAC from an algorithmic perspective. We show that RCAC uses more general policy parameterizations, making stabilization plausible. We also explain the special trick used in RCAC to formulate an online convex optimization problem under such more general policy parameterization.
2) We discuss the strengths and weaknesses of the two methods in relation to each other: RCAC can stabilize unknown output-feedback linear systems in many practical settings, while online nonstochastic control enjoys strong regret guarantees given stabilizing policies known a priori.

[1] Many learning-based stabilization methods require the assumption of full state observability [7], [20], [21]. Even in such a setting, stabilization can be hard [22].

3) We present an integration of RCAC and online nonstochstic control to achieve the best of both worlds and compare its performance to H_2 and H_∞ controllers.

The key difficulty in correlating the two methodologies lies in the disparate architectures each utilizes. To fully understand the relationship between these approaches, it is essential to thoroughly examine the foundational objectives of their respective control policies, the structure of their cost functions, and how these control policies are interrelated with their corresponding cost functions. Here, we want to emphasize that the purpose of our paper is not to undermine either online nonstochastic control or RCAC. Rather, we hope that our insights can help clarify the connections and differences of online nonstochastic control and RCAC, which potentially will help the future developments of more powerful algorithms.

II. Background

In this section, we briefly review online nonstochastic control and RCAC. Both methods recursively update policy parameters in an online manner. For ease of exposition, we consider a common setting, which addresses the following linear time-invariant (LTI) system[2]:

$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t + B_w w_t \\ y_t &= Cx_t + v_t \end{aligned} \tag{1}$$

where $x_t \in \mathbb{R}^{n_x}$, $u_t \in \mathbb{R}^{n_u}$, $w_t \in \mathbb{R}^{n_w}$, $y_t \in \mathbb{R}^{n_y}$, $v_t \in \mathbb{R}^{n_y}$ represent the system's state, input, disturbance, output, and measurement noise, respectively.

A. Online Nonstochastic Control with GPC

In [4]–[11], online nonstochastic control was developed to achieve "small" (sublinear) regret, which is defined as

$$\text{Regret}_T(\mathcal{A}) = \sum_{t=1}^{T} l_t(y_t^{\mathcal{A}}, u_t^{\mathcal{A}}) - \min_{\pi \in \mathcal{K}} \sum_{t=1}^{T} l_t(y_t^{\pi}, u_t^{\pi}) \tag{2}$$

where $\{u_t^{\mathcal{A}}, y_t^{\mathcal{A}}\}$ are the inputs and the corresponding outputs generated by an online control algorithm $\mathcal{A}$, l_t is a convex performance metric that quantifies the policy quality, $\mathcal{K}$ is the policy class where the controller in hindsight belongs to, and $\{u_t^{\pi}, y_t^{\pi}\}$ are generated under any policy $\pi \in \mathcal{K}$. In this work we will focus on the GPC method [8], [23], which is the main algorithm for online non-stochastic control.

GPC will adopt a special policy parameterization to ensure online convex optimization can be applied. The policy parameterization in GPC requires defining the so-called Nature's $y's$ as follows.

Definition 1: Given a sequence of disturbances $\{w_t\}_{t\geq 1}$, we have $y_t^{nat} := \sum_{i=1}^{t-1} CA^{i-1} B_w w_{t-i}$.

Since (1) is linear, we have $y_t^{nat} = y_t - \sum_{i=1}^{t-1} G^{[t-i]} u_i$, where $G^{[i]} := CA^{i-1}B$ denotes the i-th Markov parameter of the underlying system. In practice, only first h Markov parameters are used to compute $\hat{y}_t^{nat} = y_t - \sum_{i=t-h}^{t-1} G^{[t-i]} u_i$. In this work, we call h the simulation window for the GPC algorithm. With the above definition, we can introduce the GPC policy parameterization. Specifically, GPC generates control actions via the following policy parameterization:

$$u_t(\mathbf{M}) = \sum_{i=0}^{m-1} M^{[i]} \hat{y}_{t-i}^{nat} \tag{3}$$

where the matrices $\mathbf{M} = \left(M^{[0]}, \ldots, M^{[m-1]}\right)$ are the policy parameters to be updated in an online manner. Here, the integer m denotes the control window for GPC. The policy parameters are updated using the online gradient descent (OGD) method $\mathbf{M}_{t+1} = \Pi_{\mathcal{M}}\left(\mathbf{M}_t - \alpha_t \nabla f_t(\mathbf{M}_t)\right)$, where α_t is the learning rate, $\Pi_{\mathcal{M}}$ represents the projection of the iterates on a norm bounded constraint set $\mathcal{M}$, and $f_t(\mathbf{M}_t)$ is the online surrogate (or "ideal") cost function which is calculated over a certain cost window W as given below:

$$f_t(\mathbf{M}_t) := \sum_{i=t-W}^{t} l_i(y_i(\mathbf{M}_t), u_i(\mathbf{M}_t)). \tag{4}$$

Here we have $y_i(\mathbf{M}_t) = \sum\limits_{j=1}^{i} \sum\limits_{k=0}^{m-1} G^{[j]} M_t^{[k]} \hat{y}_{t-j-k}^{nat} + \hat{y}_i^{nat}$, and $u_i(\mathbf{M}_t)$ is given by (3). On the conceptual level, f_t is quantified on the "ideal" trajectories as if the policy $\mathbf{M}_t$ had been used over the control window. It is important to note that GPC is not a model free approach, since it requires the information of the first h Markov parameters, i.e. $G^{[1:h]} = \begin{bmatrix} G^{[1]} & G^{[2]} & \ldots & G^{[h]} \end{bmatrix}$. The above formulation actually requires (1) to be stable. If (1) is unstable, then a stabilizing controller is needed to generate a closed-loop stable system such that GPC can be applied next. In this case, the total control input is given by $u_t = \mathring{u}_t + u_t^{ex}$ where $\mathring{u}_t$ is the stabilizing policy input and u_t^{ex} is the exogenous policy given by (3). See [8] for a detailed discussion. Recent work by Chen et al. [7], have sought to overcome the need of a stabilizing controller in the online non-stochastic control approach. However, their research is confined to the full state feedback scenario and does not encompass the output feedback setting that is the focus of the current study.

B. Retrospective Cost Adaptive Control (RCAC)

RCAC is based on the concept of retrospectively optimized control, where past policy parameters are re-optimized in a way that the control performance over the previous window of operation would have been improved if the re-optimized policy had been used over that window. The intuition here is that a policy that works well over the last window of operation should also work reasonably well for the next step if the disturbance pattern has not changed drastically. A large body of research results on RCAC have been developed in the past 25 years [16]–[19]. Interestingly, the concept of regret is not involved in the developments of RCAC, although the original version of RCAC in [16] is also just based on OGD.

Similar to GPC, RCAC will use some control parameterization and then do online optimization over the policy

[2] RCAC can address more general forms of LTI systems via using the performance variable z_t. We keep $z_t \equiv y_t \ \forall t$ for the ease of exposition.

parameters. For an LTI system (1), RCAC generates control actions based on the following policy parameterization [18]:

$$u_t = \sum_{i=1}^{m} P_t^{(i)} u_{t-i} + \sum_{i=1}^{m} S_t^{(i)} y_{t-i}. \tag{5}$$

where m is the length of the control window. We can clearly see that the RCAC policy parameterization (5) is quite different from the GPC parameterization (3). As discussed in the tutorial paper [18], one can rewrite (5) compactly as $u_t = \phi(t)^T \theta_t$ where θ_t is a vector augmented from all the policy parameters $\{P_t^{(i)}, S_t^{(i)}\}_{i=1}^{m}$ (see Equations (6) and (7) in [18] for formal definitions of $\{\phi_t, \theta_t\}$). Conceptually, $\phi(t)$ holds the past input-output data, and θ_t is the time-varying policy parameter. RCAC updates θ_t via optimizing the so-called retrospective cost $J_t(\hat{\theta})$ defined as follows:

$$J_t(\hat{\theta}) := \sum_{i=t-W}^{t} c_i(\hat{z}_i, \hat{\theta}) + (\hat{\theta} - \theta_t)^T R_\theta (\hat{\theta} - \theta_t) \tag{6}$$

where $c_i(\hat{z}_i, \hat{\theta}) := \hat{z}(i, \hat{\theta})^T R_z \hat{z}(i, \hat{\theta}) + (G_f u_i)^T R_u (G_f u_i)$ with $\{R_z, R_u, R_\theta\}$ being user-specified weight matrices and $\hat{z}(i, \hat{\theta})$ being the retrospective performance variable given as

$$\hat{z}(t, \hat{\theta}) = y_t - G_f(\boldsymbol{q})\,(u_t - \hat{u}_t) \quad \text{with} \quad \hat{u}_t = \phi(t)^T \hat{\theta}. \tag{7}$$

Here we just follow the commonly-used notation in the RCAC literature, and restate Equation (11) in [18] as (7). We emphasize that $G_f(\boldsymbol{q})$ is a dynamical system by itself. We acknowledge that (7) may mix frequency-domain and time-domain notations in a non-standard way, but such notations have been widely adopted in the RCAC literature. In the above formulation, $\hat{u}_t = \phi(t)^T \hat{\theta}$ is the retrospectively computed input, and G_f is the so-called target model which can facilitate the removal of the exact contribution of u_t towards y_t and replace it with a new one in the form of $\hat{u}_t$. Intuitively, (7) can simulate what happens if the re-optimized policy had been used over the simulation window.

Early work in [16] used the OGD method to update the RCAC policy as $\theta_{t+1} = \theta_t - \alpha_t \nabla J_t(\theta_t)$. This version of RCAC is very similar to GPC, despite the differences in the policy parameterizations and the time-varying cost functions. Over the years, two other optimization algorithms have been used for RCAC. The work in [17] uses the online proximal point method, while the most recent version of RCAC relies on the recursive least squares (RLS) update law [18], [19].

The target model is a key component that distinguishes RCAC from GPC. One can choose various target models to achieve goals such as disturbance rejection and stabilization. Via choosing the target model smartly, RCAC is capable of stabilizing unknown LTI systems in many practical settings [18], [24], given limited amount of information such as the relative degree[3], the first non-zero Markov parameter of the system , and the Non-minimum phase (NMP) zeros in case they exist (this is not completely model-free, but the information required is quite minimal as claimed in [18]).

III. Main Results and Discussions

In this section, we discuss the algorithmic similarities and differences between RCAC and GPC, clarify the strengths for each method, and propose an integration to achieve the best of both worlds.

A. Algorithmic Similarities and Differences

To make the connections between RCAC and GPC transparent, we will examine the first version of RCAC [16], which uses OGD for updating control policies. As mentioned in Section II-B, the OGD-based RCAC update in [16] takes the form of $\theta_{t+1} = \theta_t - \alpha_t \nabla J_t(\theta_t)$, which is similar to the OGD-based GPC update $\mathbf{M}_{t+1} = \Pi_{\mathcal{M}}\left(\mathbf{M}_t - \alpha_t \nabla f_t(\mathbf{M}_t)\right)$. Despite such similarities, the policy parameterizations θ_t and $\mathbf{M}_t$ are quite different. Due to the difference in policy parameterizations, the cost functions J_t and f_t are also convexified differently. Next, we elaborate on these points.

In [16], the system (1) was transformed into an equivalent $\mu-$ARMARKOV model, where μ is the number of Markov parameters used to specify the dynamics. Similar to the GPC setting where h Markov parameters are used to compute $\hat{y}_t^{nat}$, setting $\mu = h$ results in the following target model[4]:

$$G_f(\boldsymbol{q}) = \frac{H_0 \boldsymbol{q}^{n-h} + H_1 \boldsymbol{q}^{n-h} + \ldots + H_{h-1}}{\boldsymbol{q}^n + \mathcal{B}_1 \boldsymbol{q}^{n-1} + \ldots + \mathcal{B}_n}$$

where $\boldsymbol{q}$ is the forward-shift operator. The parameters H_i (Markov parameter) and $\mathcal{B}_i$ are determined from the LTI system using system identification tools [25]. Using the above G_f, this version of RCAC defines $\hat{z}(t, \hat{\theta})$ as:

$$\hat{z}(t, \hat{\theta}) = y_t - B_{zu}\left(U_t - \Phi(t)\hat{\theta}\right)$$

where $B_{zu} = \begin{bmatrix} H_0 & \ldots & H_{h-1} & \mathcal{B}_1 & \ldots & \mathcal{B}_n \end{bmatrix}$. Moreover, U_t and $\Phi(t)$ hold the history of u_t and $\phi(t)$, respectively. Similar to the target model, the policy parameterization of this version of RCAC uses μ_c-ARMARKOV architecture. Setting $\mu_c = 0$ in [16], the policy parameterization becomes $u_t = \sum_{i=1}^{m} \alpha_{c,i} u_{t-i} + \sum_{i=1}^{m} \mathcal{B}_{c,i} y_{t-i}$, where $\alpha_{c,i}$ and $\mathcal{B}_{c,i}$ are similar to $P_t^{(i)}$ and $S_t^{(i)}$ defined in Section II-B. Now we can see that the policy parameterization for RCAC is more general than the policy parameterization for GPC. This algorithmic difference can be formalized as follows.

Lemma 1: The GPC policy parameterization is a special case of the RCAC policy parameterization.

Proof: The output of (1) can be decomposed as $\{y_t^{nat}\}$ and the forced response $\{y_t^{fr}\}$. The RCAC policy becomes $u_t = \sum_{i=1}^{m} \alpha_{c,i} u_{t-i} + \sum_{i=1}^{m} \mathcal{B}_{c,i} y_{t-i}^{nat} + \sum_{i=1}^{m} \mathcal{B}_{c,i} y_{t-i}^{fr}$, reducing to a disturbance response policy by setting:

$$\alpha_{c,i} = \frac{-\mathcal{B}_{c,i}}{u_{t-i}} y_{t-i}^{fr} = \frac{-\mathcal{B}_{c,i}}{u_{t-i}} \sum_{i=1}^{h} \hat{G}^{[i]} u_{t-i}, \; i = 1, 2, \ldots, m.$$

This completes the proof. ■

By choosing more advanced target models, the policy parameterization of RCAC can become even more general,

[3] Relative degree is the difference between the number of poles and number of zeros in the system.

[4] Note that setting $\mu = h$ for RCAC does not enforce the simulation window length to be h. Simulation window length still equals to one

allowing the adaptation of the closed-loop poles and making the stabilization plausible. In contrast, online nonstochastic control with GPC does not change the closed-loop system poles during the online optimization process, and hence cannot stabilize open-loop unstable systems.

The control parameterization for GPC naturally allows for convex formulations of $\{f_t\}$ [26]. However, even with the more complicated control parameterizations for RCAC, the resultant online optimization problem is still convex if we choose $h = 1$. Next we explain this point. For the above version of RCAC, the loss is defined as $c_t(\hat{z}_t, \hat{\theta}) = \frac{1}{2}\hat{z}^T(t,\hat{\theta})\hat{z}(t,\hat{\theta})$, and the cost function $J_t(\hat{\theta})$ is given by (6)) using a cost window W. We can obtain the following result.

Lemma 2: The cost function $J_t(\hat{\theta})$ for RCAC is convex with respect to $\hat{\theta}$ for $h = 1$ and non-convex for $h > 1$.

Proof: Recall from (6), the second term in $J_t(\hat{\theta})$ is a quadratic function of $\hat{\theta}$ and thus it is convex in $\hat{\theta}$. $J_t(\hat{\theta})$ will be a convex function if the c_t is convex in $\hat{\theta}$. In case of RCAC, c_t is a quadratic function of $\hat{z}(i,\hat{\theta})$ and $\hat{u}_i(\hat{\theta})$. Thus c_t will be convex in $\hat{\theta}$ only if it is linear in $\hat{\theta}$. Next, we show that this requirement holds only for $h = 1$.

Let us first compute $\hat{z}(t,\hat{\theta})$ using $h = 1$ (re-simulating the system by removing input applied at iteration $t-1$ and using a new input). Without loss of generality, we assume the system has a relative degree $r = 1$ and use a $G_f = \frac{H_1}{\boldsymbol{q}}$ (the result holds for any $r > 1$ and the general target models in [24] as well). Using (7), $\hat{z}(t,\hat{\theta})$ is given as follows:

$$\hat{z}(t,\hat{\theta}) = CAx_{t-1} + CB_w w_{t-1} + G^{[1]}\sum_{i=1}^{m}(\hat{P}^i u_{t-i-1} + \hat{S}^i y_{t-i-1})$$

This shows that $\hat{z}(t,\hat{\theta})$ is linear in the policy parameters $\hat{P}$ and $\hat{S}$. Similarly, $\hat{u}_t(\hat{\theta})$ given by (5) is linear in $\hat{P}$ and $\hat{S}$. Notice that even though RCAC specializes c_t to a quadratic cost, the linearity of $\hat{z}(t,\hat{\theta})$ in $\hat{\theta}$ allows the use of any cost function c_t which is convex in $\hat{z}(t,\hat{\theta})$ and $\hat{u}_t(\hat{\theta})$. Thus any convex cost $c_t(\hat{z}_t, \hat{u}_t)$ will be convex in $\hat{\theta}$. Extending $\hat{z}(t,\hat{\theta})$ to $h = 2$ by substituting $u_{t-i-1} = \phi(t-i-1)\hat{\theta}$ gives us:

$$\begin{aligned}\hat{z}(t,\hat{\theta}) &= CAx_{t-1} + CB_w w_{t-1} + G^{[1]}\sum_{i=1}^{m}\hat{S}^i y_{t-i-1} \\ &\quad + G^{[1]}\sum_{i,j=1}^{m}\left(\hat{P}^i\hat{P}^j u_{t-i-j-1} + \hat{P}^i\hat{S}^j y_{t-i-j-1}\right)\end{aligned}$$

Clearly, this shows a nonlinear relation between $\hat{\theta}$ and $\hat{z}(t,\hat{\theta})$. Thus any cost function $J_t(\hat{\theta})$ convex in $\hat{z}_t$ will be non-convex in the policy parameters $\hat{\theta}$ for any $h > 1$. ■

Due to the use of the more general policy parameterizations in RCAC, we have to choose $h = 1$ to convexify the retrospective cost J_t, making it very difficult to connect a regret bound on J_t to the original loss $\sum_{t=1}^{T} l_t$. This is the reason why the regret arguments for online nonstochastic control cannot be applied to RCAC. This is even the case when the system state dimension is 1. In terms of the required modelling information, both GPC and the above version of RCAC require the same set of Markov parameters. However, the more recent version of RCAC has reduced this dependence to only the first non-zero Markov parameter [18].

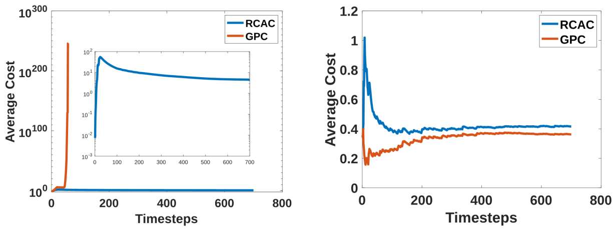
B. Clarifications on Strengths of RCAC and GPC

In light of the above discussions, now we clarify the strengths of RCAC and GPC. The policy parameterization for GPC allows simple theoretical analysis. However, such policy parameterization does not change the closed-loop poles, and hence lose the ability to stabilize an unstable system. In contrast, the RCAC policy parameterization is more general and can potentially adapt the closed-loop poles to enable stabilizing unknown LTI systems. There is a trade-off between regret guarantees versus stabilizing abilities.

The stabilization property in RCAC comes from the target model G_f. When applying RCAC to unstable systems, one can choose $G_f := \frac{H}{\boldsymbol{q}^{r+n_p}D(\boldsymbol{q})}$ where r is the relative degree of the system, and H is the first non-zero Markov parameter. Here, $D(\boldsymbol{q})$ is a polynomial of degree $n_p = m + n_1$ specifying the desired closed loop poles of the system where n_1 is the number of unstable poles of the system. RCAC achieves pole placement under the assumption of persistent excitation or alternatively use of a forgetting factor $\lambda \in (0,1]$[5] with $c_i(\hat{z}_i, \hat{\theta})$ in $J_t(\hat{\theta})$ [19]. RCAC updates $J_t(\hat{\theta})$ in such a way that the closed-loop poles of the system converges to the poles of $D(\boldsymbol{q})$, thereby stabilizing an unstable system. To see this, define $\tilde{u}_t := u_t - \hat{u}_t$ to be the input perturbation then asymptotically as $\hat{z}(t,\hat{\theta}) \to 0$, $y_t \approx G_f(\boldsymbol{q})\tilde{u}_t$ from (7). Using u_t from (5) with LTI system (1) leads to a closed loop system given by $y_t = \tilde{G}_{y\tilde{u}}(\boldsymbol{q})\tilde{u}_t + \tilde{G}_{yw}(\boldsymbol{q})w_t$ where $\tilde{G}_{y\tilde{u}}(\boldsymbol{q})$ and $\tilde{G}_{yw}(\boldsymbol{q})$ share the same set of poles [24]. Thus asymptotically RCAC makes $\tilde{G}_{y\tilde{u}}(\boldsymbol{q}) \to G_f(\boldsymbol{q})$ and thus achieve system stabilization. GPC lacks this capability (in addition, so far there does not exist frequency-domain interpretations for GPC).

RCAC is capable of performing system stabilization, disturbance rejection and reference tracking in many practical simulations, but it lacks formal guarantees. Even in case of simple scalar system with full state observation, stability guarantee is not easily available. In order to highlight this difficulty, we consider the most basic setting of constant disturbance rejection: $x_{t+1} = ax_t + u_t + w$; stable system with $|a| < 1$ and $w \in \mathbb{R}$. $J_t(\hat{\theta}) = \hat{z}(t,\hat{\theta})^T\hat{z}(t,\hat{\theta})$ and RCAC policy $u_t = P_t u_{t-1} + S_t x_{t-1}$ is updated by the OGD update rule. The equilibrium point for this system is given by $(x^*, u^*, P^*, S^*) = (0, -w, 1, \mathbb{R})$. It is easy to verify that S^* is dependent on the initial condition of the system and it will be different for each trial of this system, thus highlighting the difficulty of establishing even local stability guarantees for RCAC. In contrast, GPC enjoys strong regret guarantees as shown in [26], i.e. GPC can achieve the sublinear regret bound $\mathcal{O}(\sqrt{T})$ under the assumption that $\rho(A) < 1$. In order to bridge the strengths of both approaches, we propose an integration in the next subsection.

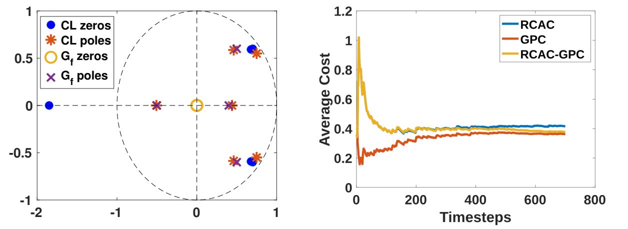
[5] Here, we assume $\lambda = 1$ for simplicity.



(a) Unstable system

(b) Stable system

Fig. 1: Cost comparison of RCAC and GPC algorithms for LTI systems under time varying disturbance sequences. In (a) we show a magnified view for RCAC.



(a) Closed loop poles of the unstable system (8)

(b) Cost comparison for the system (9)

Fig. 2: Performance of the RCAC-GPC integrated policy.

C. Integration of GPC and RCAC

Based on the underlying similarities between RCAC and GPC, it seems natural to consider integration. Let us first consider two LTI systems as examples to further motivate the need to integrate the two algorithms. In both examples, we used the control window $m = n$, simulation window $h = 6$, and cost window $W = 1$ with a quadratic loss function. Consider the following system:

$$x_{t+1} = \begin{bmatrix} 2.85 & -2.41 & 0.85 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} x_t + \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} (u_t + w_t) \qquad (8)$$
$$y_t = \begin{bmatrix} -1 & 1.4 & -0.85 \end{bmatrix} x_t$$

This system has an unstable pole at $z = 1.75$ and it is subject to the disturbance sequence given by $w_t \sim \mathcal{N}(0, 0.1)$ for $t \in [0, 100]$, $w_t = 0.5(\cos 0.25t + \sin(0.5t))$ for $t \in (100, 200]$, and $w_t = 0.1 \cos 2t + \sin(0.2t) + w_0$ for $t \in (200, 700]$ where $w_0 \sim \mathcal{N}(0, 0.1)$. Fig. 1(a) performs the cost comparison for RCAC and GPC algorithms on log scale. Clearly, RCAC can simultaneously stabilize a system while rejecting disturbance unlike GPC which is unable to handle such a system. Consider another example given by:

$$x_{t+1} = \begin{bmatrix} 0.94 & -0.44 \\ 1.00 & 0 \end{bmatrix} x_t + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u_t + \begin{bmatrix} 0 \\ 1 \end{bmatrix} w_t$$
$$y_t = \begin{bmatrix} 1 & -0.8 \end{bmatrix} x_t \qquad (9)$$

We consider a mixed disturbance given by $w_t = \cos 2t + w_0$ for $t \in [0, 100]$, $w_t = \sin 0.2t + w_0$ for $t \in (100, 200]$ and $w_t = \cos 2t + \sin(0.2t) + w_0$ for $t \in (200, 700]$, where $w_0 \sim \mathcal{N}(0, 0.5)$ is used for this system. As shown in Fig. 1(b), GPC achieves a lower cost than RCAC, which reflects its strength in rejecting adversarial disturbances while minimizing the tracking cost for stable systems.

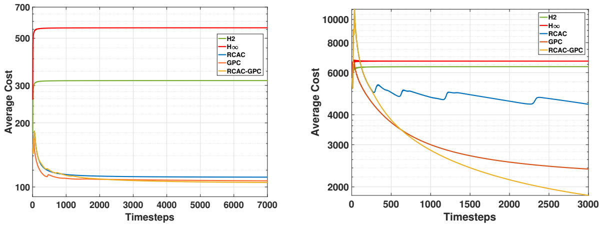
The two examples discussed above motivate the need of a new control policy that can harness the best attributes of the two algorithms. A natural idea is to synthesize a control policy that uses GPC for disturbance rejection while using RCAC for system stabilization. Specifically, we propose to use a fixed RCAC parameterization as a stabilizing policy while actively using the OGD in the GPC framework to adapt to unknown and changing disturbance signals. This way, the two algorithms not only are seemingly integrated to synthesize a new control policy, but also aid in estimating the Markov parameters required in the GPC algorithm. This proposed framework is summarized in Algorithm 1.

Algorithm 1 RCAC-GPC algorithm

```
1:  Input: Times steps: $t_1$, $N$, $T$, $(\alpha_t)_{t\geq 1}$, $h$, $m$, $W$, $G_f$,
    $\theta_0 = \mathbf{0}$, $M_0 = \mathbf{0}$, excitation signals: $(\eta_i, \nu_i)_{i=1}^{N}$
    /* System stabilization by RCAC */
2:  for $t = 1, \ldots, t_1$ do
3:      Simulate RCAC policy in (5) and update $\theta_t$ using RLS
    /* Data Collection for System ID */
4:  for $t = t_1 + 1, \ldots, t_1 + N$ do
5:      Simulate RCAC policy in (5) for a fixed $\theta_{t_1}$ with
        additional input-output excitation signals $(\eta_t, \nu_t)$
    /* System Identification */
6:  $\hat{G}^{[1:h]} \leftarrow$ Closed loop system ID with $\{(u_i, y_i)\}_{i=t_1}^{t_1+N}$
    and $(\eta_i, \nu_i)_{i=1}^{N}$ using least square estimation in [27].
    /* Online optimal control using RCAC-GPC */
7:  for $t = t_1 + N + 1, \ldots, T$ do
8:      Compute RCAC policy $\mathring{u}_t$ using (5) with fixed $\theta_{t_1}$
9:      Compute GPC policy $\bar{u}_t$ using (3)
10:     Simulate $u_t = \mathring{u}_t + \bar{u}_t$ and observe loss $l_t(y_t, u_t)$
11:     Update cost $f_t(\mathbf{M}_t)$ in (4)
12:     Update $\mathbf{M}_t$ using OGD for $f_t(\mathbf{M}_t)$
```

Algorithm 1 follows three main steps. In the first step, it simulates the system for t_1 iterations with RCAC to stabilize the system and simultaneously update policy parameters. In the second step, a fixed RCAC policy parameterization from Step 1 is used to identify Markov parameters which are needed for the next GPC step. In line 6, $\hat{G}^{[1:h]} := \begin{bmatrix} \hat{G}^{[1]} & \hat{G}^{[2]} & \ldots & \hat{G}^{[h]} \end{bmatrix}$ are estimated for the implementation of GPC and RCAC, where $\hat{G}^{[i]}$ is an estimate of $G^{[i]}$. In the last step, it uses a fixed RCAC policy to maintain system's stability while using GPC to mitigate unknown and changing disturbance signals.

Here, it is important to note that estimating Markov parameters is particularly a challenging task if the underlying system is unstable. In case of stable as well as unstable systems, RCAC can be seemingly deployed for the Markov parameter estimation process. Once RCAC successfully stabilizes a system, one can use the framework proposed by [27] to estimate the required Markov parameters of the system.



(a) Boeing 747 system (b) System with a NMP zero

Fig. 3: Cost comparison for two different control problems

Now, let us revisit the two examples considered above and use the proposed policy in Algorithm 1 under identical settings. Fig. 2 (a) shows the location of closed-loop poles for the system in (8) after 100 iterations of RCAC in comparison to the desired poles supplied by the target model G_f. Fig. 2 (b) shows the cost comparison the three algorithms under identical settings. Clearly, the two algorithms complement each another as the resulting closed loop exhibits superior performance.

IV. Empirical Comparisons

The efficacy of the proposed algorithm is demonstrated by empirical comparison to H_2 and H_∞ controllers in addition to the RCAC and GPC controllers. We consider the example of a Boeing 747 at an altitude of 40000ft with the speed of 774ft/s subject to non-stochastic disturbance w_t, a setting similar to the one in [28]. Conventionally, H_2 and H_∞ control paradigms are used to address such systems. Therefore, we provide a comparison to H_2 and H_∞ controllers in addition to RCAC and GPC control policies in Fig. 3 (a). The unified algorithm uses RCAC for the first 200 iterations followed by a fixed RCAC controller with the GPC for disturbance rejection.

We consider another interesting case where an LTI system with a NMP zero is subject to a non-stochastic disturbance signal. The significance of this system stems from the fact that RCAC requires a priori knowledge of the NMP zero where as GPC doesn't need such information.

$$x_{t+1} = \begin{bmatrix} 0.7 & -0.72 & 0.25 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} x_t + \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} (u_t + w_t)$$

$$y_t = \begin{bmatrix} 0 & 1 & -9.75 \end{bmatrix} x_t \tag{10}$$

This system has a relative degree $r = 2$. The target model used in this case was $G_f(\boldsymbol{q}) = \frac{-(\boldsymbol{q}-9)}{\boldsymbol{q}^2}$, an estimated of the NMP zero is used. As shown in Fig. 3 (b) the integrated RCAC-GPC policy outperform the other four control policies. Empirical studies has shown that often times explicit reliance of RCAC on the information of NMP zeros puts it at a disadvantage in comparison to GPC and GPC outperform RCAC in such cases. On the other hand, RCAC-GPC policy in such cases leads to superior performance in comparison to the individual policies. These results therefore, demonstrate the effectiveness of the integrated RCAC-GPC policy in system stabilization and disturbance rejection.

References

[1] N. Cesa-Bianchi and G. Lugosi, *Prediction, learning, and games*. Cambridge university press, 2006.

[2] S. Shalev-Shwartz, "Online learning and online convex optimization," *Foundations and trends in Machine Learning*, vol. 4, no. 2, pp. 107–194, 2011.

[3] E. Hazan, "Introduction to online convex optimization," *Foundations and Trends® in Optimization*, vol. 2, no. 3-4, pp. 157–325, 2016.

[4] N. Agarwal, B. Bullins, E. Hazan, S. Kakade, and K. Singh, "Online control with adversarial disturbances," in *International Conference on Machine Learning*, 2019, pp. 111–119.

[5] D. Foster and M. Simchowitz, "Logarithmic regret for adversarial online control," in *International Conference on Machine Learning*, 2020, pp. 3211–3221.

[6] E. Hazan, S. Kakade, and K. Singh, "The nonstochastic control problem," in *Algorithmic Learning Theory*, 2020, pp. 408–421.

[7] X. Chen and E. Hazan, "Black-box control for linear dynamical systems," in *Conference on Learning Theory*, 2021, pp. 1114–1143.

[8] M. Simchowitz, K. Singh, and E. Hazan, "Improper learning for nonstochastic control," in *Conference on Learning Theory*, 2020, pp. 3320–3436.

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

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

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

[12] B. Anderson and J. Moore, *Optimal Control: Linear Quadratic Methods*. Prentice-Hall, 1989.

[13] A. Bryson and Y. Ho, *Applied Optimal Control: Optimization, Estimation and Control*. CRC Press, 1975.

[14] J. Doyle, K. Glover, P. Khargonekar, and B. Francis, "State-space solutions to standard $\mathcal{H}_2$ and $\mathcal{H}_\infty$ control problems," *IEEE Transactions on Automatic Control*, vol. 34, pp. 831–847, 1989.

[15] G. Dullerud and F. Paganini, *A Course in Robust Control Theory: A Convex Approach*. Springer, 1999.

[16] R. Venugopal and D. S. Bernstein, "Adaptive disturbance rejection using armarkov/toeplitz models," *IEEE Transactions on Control Systems Technology*, vol. 8, no. 2, pp. 257–269, 2000.

[17] M. A. Santillo and D. S. Bernstein, "Adaptive control based on retrospective cost optimization," *Journal of guidance, control, and dynamics*, vol. 33, no. 2, pp. 289–304, 2010.

[18] Y. Rahman, A. Xie, J. B. Hoagg, and D. S. Bernstein, "A tutorial and overview of retrospective cost adaptive control," in *2016 American Control Conference (ACC)*, 2016, pp. 3386–3409.

[19] Y. Rahman, A. Xie, and D. S. Bernstein, "Retrospective cost adaptive control: Pole placement, frequency response, and connections with LQG control," *IEEE Control Systems Magazine*, vol. 37, no. 5, pp. 28–69, 2017.

[20] S. Lale, K. Azizzadenesheli, B. Hassibi, and A. Anandkumar, "Reinforcement learning with fast stabilization in linear dynamical systems," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2022, pp. 5354–5390.

[21] A. Mete, R. Singh, and P. R. Kumar, "Augmented rbmle-ucb approach for adaptive control of linear quadratic systems," 2023.

[22] X. Zeng, Z. Liu, Z. Du, N. Ozay, and M. Sznaier, "On the hardness of learning to stabilize linear systems," in *2023 62nd IEEE Conference on Decision and Control (CDC)*. IEEE, 2023, pp. 6622–6628.

[23] E. Hazan and K. Singh, "Introduction to online nonstochastic control," *arXiv preprint arXiv:2211.09619*, 2022.

[24] A. Xie, "Retrospective cost adaptive control for feedback and feedforward noise control," Ph.D. dissertation, University of Michigan, 2017.

[25] J. C. Akers and D. S. Bernstein, "Armarkov least-squares identification," in *Proceedings of the 1997 American Control Conference (Cat. No. 97CH36041)*, vol. 1. IEEE, 1997, pp. 186–190.

[26] E. Hazan and K. Singh, "Introduction to online nonstochastic control," 2023.

[27] M. Phan, J.-N. Juang, L. G. Horta, and R. W. Longman, "System identification from closed-loop data with known output feedback dynamics," *Journal of guidance, control, and dynamics*, vol. 17, no. 4, pp. 661–669, 1994.

[28] O. Sabag, G. Goel, S. Lale, and B. Hassibi, "Regret-optimal controller for the full-information problem," in *2021 American Control Conference (ACC)*, 2021, pp. 4777–4782.