Abstract

We consider the setting of iterative learning control, or model-based policy learning in the presence of uncertain, time-varying dynamics. In this setting, we propose a new performance metric, planning regret, which replaces the standard stochastic uncertainty assumptions with worst case regret. Based on recent advances in non-stochastic control, we design a new iterative algorithm for minimizing planning regret that is more robust to model mismatch and uncertainty. We provide theoretical and empirical evidence that the proposed algorithm outperforms existing methods on several benchmarks.

1 Introduction

Consider a robotic system learning to perform a novel task, e.g., a quadrotor learning to fly to a specified goal, a manipulator learning to grasp a new object, or a fixed-wing airplane learning to perform a new maneuver. We are particularly interested in settings where (i) the task requires one to plan over a given time horizon, (ii) we have access to an inaccurate model of the world (e.g., due to unpredictable external disturbances such as wind gusts or misspecification of physical parameters such as masses, inertias, and friction coefficients), and (iii) the robot is allowed to iteratively refine its control policy via multiple executions (i.e., rollouts) on the real world. Motivated by applications where real-world rollouts are expensive and time-consuming, our goal in this paper is to perform the given task as rapidly as possible. More precisely, given a cost function that specifies the task, our goal is to learn a low-cost control policy using a small number of rollouts.

The problem described above is challenging due to a number of factors. The primary challenge we focus on in this paper is the existence of unmodeled deviations from nominal dynamics, and external disturbances acting on the system. Such disturbances may either be random or potentially even adversarial. In this paper we adopt a regret minimization approach coupled with a recent paradigm called non-stochastic control to tackle this problem in generality. Specifically, consider a time-varying dynamical system given by the equation

$$x_{t+1} = f_t(x_t, u_t) + w_t, \quad (1.1)$$

where $x_t$ is the state, $u_t$ is the control input, and $w_t$ is an arbitrary disturbance at time $t$. Given a horizon $T$, the performance of a control algorithm $A$ may be judged via the aggregate cost it suffers on a cost function sequence $c_1, \ldots, c_T$ along its state-action trajectory $(x^A_t, u^A_t, \ldots)$:

$$J(A|w_1:T) = \frac{1}{T} \sum_{t=1}^T c_t(x^A_t, u^A_t).$$

For deterministic systems, an optimal open-loop control sequence $u_1 \ldots u_T$ can be chosen to minimize the cost sequence. The presence of unanticipated disturbances often necessitates the superposition of a closed-loop correction policy $\pi$ to obtain meaningful performance. Such closed-loop policies can modify $u'_t = \pi(u_{1:t}, x_{1:t})$ as a function of the observed history thus far, and facilitate adaptation to realized disturbances. To capture this, we define a comparative performance metric, which we call Planning.
Regret. In an episodic setting, for every episode $i$, an algorithm $\mathcal{A}$ adaptively selects control inputs while the rollout is performed under the influence of an arbitrary disturbance sequence $w_i^1:T$. Planning regret is the difference between the total cost of the algorithm’s actions and that of the retrospectively optimal open-loop plan coupled with episode-specific optimal closed-loop policies (from a policy class $\Pi$). Regret, therefore, is the relative cost of not knowing the to-be realized disturbances in advance. Formally for a total of $N$ rollouts, each of horizon $T$, it is defined as:

$$\text{Planning Regret} \sum_{i=1}^{N} J(\mathcal{A} | w_i^1:T) - \min_{u_i^1:T} \min_{\pi_i^1 \in \Pi} J(u_i^1:T, \pi_i^1 | w_i^1:T)$$

The motivation for our performance metric arises from the setting of Iterative Learning Control (ILC), where one assumes access to an imperfect (differentiable) simulator of real-world dynamics as well as access to a limited number of rollouts in the real world. In such a setting the disturbances capture the model-mismatch between the simulator and the real-world. The main novelty in our formulation is the fact that, under vanishing regret, the closed-loop behavior of $\mathcal{A}$ is almost instance-wise optimal on the specific trajectory, and therefore adapts to the passive controls, dynamics and disturbance for each particular rollout. Indeed, worst-case regret is a stronger metric of performance than commonly considered in the planning/learning for control literature.

Our main result is an efficient algorithm that guarantees vanishing average planning regret for non-stationary linear systems and disturbance-action policies. We experimentally demonstrate that the algorithm yields substantial improvements over ILC in linear and non-linear control settings.

1.1 Related Work
The literature on planning and learning in partially known MDPs is vast, and we focus here on the setting with the following characteristics:

1. We consider model-aided learning, which is suitable for situations in which the learner has some information about the dynamics, i.e. the mapping $f_t$ in Equation (1.1), but not the disturbances $w_t$. We further assume that we can differentiate through the model. This enables efficient gradient-based algorithms.

2. We focus on the task of learning an episodic-agnostic control sequence, rather than a policy. This is aligned with the Pontryagin optimality principle [PBGM62, Ros15], and differs from dynamic programming approaches [SB18].

3. We accommodate arbitrary disturbance processes, and choose regret as a performance metric. This is a significant deviation from the literature on optimal and robust control [ZDG96, Ste94], and follows the lead of the recent paradigm of non-stochastic control [ABH19, HKS20, SSH20].

4. Our approach leverages multiple real-world rollouts. This access model is most similar to the iterative learning control (ILC) paradigm [OH05, ACM07]. For comparison, the model-predictive control (MPC) paradigm allows for only one real-world rollout on which performance is measured, and all other learning is permitted via access to a simulator.

Optimal, Robust and Online Control. Classic results [Ber05, ZDG96, Ted20] in optimal control characterize the optimal policy for linear systems subject to i.i.d. perturbations given explicit knowledge of the system in advance. Beyond stochastic perturbations, robust control approaches [ZD98] compute the best controller under worst-case noise.

Recent work in machine learning study regret bounds vs. the best linear controller in hindsight for online control with known and unknown linear dynamical systems. Online control was extended to adversarial perturbations, giving rise to the nonstochastic control model. In this general setting regret bounds were obtained for known/unknown systems as well as partial observation [ABH19, HKS20, SSH20, Sim20].

Paper structure. We present the relevant definitions including the setting in Section 2. The algorithm and the formal statement of the main result can be found in Section 3. In Section 4 we provide an overview of the algorithm and the proof via the proposal of a more general and abstract nested online convex optimization (OCO) game. This formulation can be of independent interest. Finally in Section 5 we provide the results and details of the experiments. Proofs and other details are deferred to the Appendix.
Planning with inaccurate models. Model predictive control (MPC) [May14] provides a general scheme for planning with inaccurate models. MPC operates by applying model-based planning, (e.g. iLQR [LT04] [TL05]), in a receding-horizon manner. MPC can also be extended to robust versions [BM99] [MSR05] [LCRM04] that explicitly reason about the parametric uncertainty or external disturbances in the model. Recently, MPC has also been viewed from the lens of online learning [WCSB19].

The setting we consider here is more general than MPC, allowing for iterative policy improvement across multiple rollouts on the real world.

An adjacent line of work on learning MPC [HWMZ20] [RB17] focuses on constraint satisfaction and safety considerations while learning models simultaneously with policy execution.

Iterative Learning Control (ILC). ILC is a popular approach for tackling the setting considered. ILC operates by iteratively constructing a policy using an inaccurate model, executing this policy on the real world, and refining the policy based on the real-world rollout. ILC can be extended to use real-world rollouts to update the model (see, e.g., [AQN06]). For further details regarding ILC, we refer the reader to the text [Moo12]. Robust versions of ILC have also been developed in the control theory literature [dR96], using H-infinity control to capture bounded disturbances or uncertainty in the model.

However, most of the work in robust control, typically account for worst-case deviations from the model and can lead to extremely conservative behavior. In contrast, here we leverage the recently-proposed framework of non-stochastic control to capture instance-specific disturbances. We demonstrate both empirically and theoretically that the resulting algorithm provides significant gains in terms of sample efficiency over the standard ILC approach.

Meta-Learning. Our setting, analysis and, in particular, the nested OCO setup bears similarity to formulations for gradient-based meta-learning (see [FAL17] and references therein). In particular, as we detail in the Appendix (Section A), the nested OCO setting we consider is a generalization of the setting considered in [BKT19]. We further detail certain improvements/advantages our algorithm and analysis provides over the results in [BKT19]. We believe this connection with Meta-Learning to be of independent interest.

2 Problem Setting

2.1 Notation

The norm \( \| \cdot \| \) refers to the \( \ell_2 \) norm for vectors and spectral norm for matrices. For any natural number \( n \), the set \([n]\) refers to the set \([1, 2 \ldots n]\).

We use the notation \( v_{a:b} \equiv \{v_a \ldots v_b\} \) to denote a sequence of vectors/matrices. Given a set \( S \), we use \( v_{a:b} \in S \) to represent element wise inclusion, i.e. \( \forall j \in [a, b], v_j \in S \). \( \text{Proj}_S(v_{a:b}) \) represents the element-wise \( \ell_2 \) projection onto the set \( S \). \( v_{a:b,c:d} \) denotes a sequence of sequences, i.e. \( v_{a:b,c:d} = \{v_{a:c} \ldots v_{b:d}\} \) with \( v_{a:c} = \{v_a \ldots v_c\} \).

2.2 Basic Definitions

A dynamical system is specified via a start state \( x_0 \in \mathbb{R}^{d_x} \), a time horizon \( T \) and a sequence of transition functions \( f_{1:T} = \{f_t|f_t : \mathbb{R}^{d_x} \times \mathbb{R}^{d_x} \rightarrow \mathbb{R}^{d_x}\} \). The system produces a \( T \)-length sequence of states \( (x_1, \ldots x_{T+1}) \) when subject to an \( T \)-length sequence of actions \( (u_1 \ldots u_T) \) and disturbances \( \{w_1 \ldots w_T\} \) according to the following dynamical equation\footnote{For the sake of simplicity, we do not consider a terminal cost, and consequently drop the last state from the description.}

\[
x_{t+1} = f_t(x_t, u_t) + w_t.
\]

Through the paper the only assumption we make about the disturbance \( w_t \) is that it is supported on a set of bounded diameter \( W \). We assume full observation of the system, i.e. the states \( x_t \) are visible to the controller. We also assume the passive transition function to be known beforehand. These assumptions imply that we fully observe the instantiation of the disturbances \( w_{1:T} \) during runs of the system.

The actions above may be adaptively chosen based on the observed state sequence, i.e. \( u_t = \pi_t(x_{1:t}, x_t) \) for some non-stationary policy \( \pi_t: \mathbb{R}^{d_x} \rightarrow \mathbb{R}^{d_x} \). We consider the policy to be deterministic (a restriction made for convenience). Therefore the state-action sequence \( \{x_t, u_t\}_{t=1}^T \), defined as \( x_{t+1} = f_t(x_t, u_t) + w_t, u_t = \pi_t(x_1, \ldots x_t) \), thus produced is a sequence determined by \( w_{1:T} \), fixing the policy, and the system.

A rollout of horizon \( T \) on \( f_{1:T} \) refers to an evaluation of the above sequence for \( T \) time steps. When the dynamical system will be clear from the context, for the rest of the paper, we drop it from our notation. Given a cost function sequence \( \{c_t\} : \mathbb{R}^n \times \mathbb{R}^d \rightarrow \mathbb{R} \) the loss of executing a policy \( \pi \) on the dynamical system \( f \) with a particular disturbance sequence given
by \(w_{1:T}\) is defined as
\[
J(\pi_{1:T}|f_{1:T},w_{1:T}) \triangleq \frac{1}{T} \left[ \sum_{t=1}^{T} c_t(x_t,u_t) \right].
\]

**Assumption 2.1.** We will assume that the cost \(c_t\) is a twice differentiable convex function and that the value, gradient and hessian of the cost function \(c_t\) is available. Further we assume,

- **Lipschitzness:** There exists a constant \(G\) such that if \(\|x\|,\|u\| \leq D\) for some \(D > 0\), then \(\|\nabla_x c_t(x,u)\|,\|\nabla_u c_t(x,u)\| \leq GD\).

- **Smoothness:** There exists a constant \(\beta\) such that for all \(x,u\), \(\nabla^2 c_t(x,u) \preceq \beta I\).

When the dynamical system and the noise sequence are clear from the context we suppress them from the notation for the cost denoting it by \(J(\pi_{1:T})\).

A particular sub-case which will be of special interest to us is the case of linear dynamical systems (LDS). Formally, a (non-stationary) linear dynamical system is described by a sequence of matrices \(AB_1:T = \{(A_t,B_t) \in \mathbb{R}^{d_x \times d_x} \times \mathbb{R}^{d_u \times d_u}\}^{T}_{t=1}\) and the transition function is defined as \(x_{t+1} = A_t x_t + B_t u_t\).

**Assumption 2.2.** We will assume that the linear dynamical system \(AB_1:T\) is \((\kappa,\delta)\)- strongly stable for some \(\kappa > 0\) and \(\delta \in (0,1]\), i.e. for every \(t\), we have that \(\|A_t\| \leq 1 - \delta,\|B_t\| \leq \kappa\).

We note that all the results in the paper can be easily generalized to a weaker notion of strong stability where the linear dynamical system is \((\kappa,\delta)\)- strongly stable if there exists a sequence of matrices \(K_1:T\), such that for every \(t\), we have that \(\|A_t - B_t K_t\| \leq 1 - \delta,\|B_t\|,\|K_t\| \leq \kappa\). A system satisfying such an assumption can be easily transformed to a system satisfying Assumption 2.2 by setting \(A_t = A_t - B_t K_t\). This redefinition is equivalent to appending the linear policy \(K_t\) on top of the policy being executed. While we present the results for the case when \(K_T = 0\), the only difference the non-zero case makes to our analysis is potentially increasing the norm of the played actions which can still be shown to be bounded. Overall this nuance leads to a difference to our main result only in terms of factors polynomial in the system parameters. Hence for convenience, we state our results under Assumption 2.2.

The assumption of strong-stability (in a weaker form as allowed by stationary systems) has been popular in recent works on online control \cite{chowdhary2018online} \cite{abboud2019online} and the above notion generalizes it to non-stationary systems.

### 2.3 Policy Classes

#### Open-Loop Policies

Given a convex set \(\mathcal{U} \subseteq \mathbb{R}^{d_u}\), consider a sequence of control actions, \(u_{1:T} \in \mathcal{U}\). We define (by an overload of notation), the open-loop policy \(u_{1:T}\) as a policy which plays at time \(t\), the action \(u_t\). The set of all such policies is defined as \(\mathcal{U}_\mathcal{T} \triangleq \mathcal{U}^{\otimes T}\).

Given two policies we define the sum of the two (denoted by \(\pi_1 + \pi_2\)) as the policy for which the action at time \(t\) is the sum of the action recommended by policy \(\pi_1\) and \(\pi_2\).

#### Linear Policies

Given a matrix \(K \in \mathbb{R}^{d_u,d_x}\), a linear policy \(\pi\) denoted (via an overload of notation) by \(K\) is a policy that plays action \(u_t = K x_t\). Such linear state-feedback policies are known to be optimal for the LQR problem and for \(H_\infty\) control \cite{zhou1996 robust}.

#### Disturbance Action Policies

A generalization of the class of linear policies can be obtained via the notion of disturbance-action policies (see \cite{abboud2019online}) defined as follows. A disturbance action policy \(\pi_{M_{1:L}}\) of memory length \(L\) is defined by a sequence of matrices \(M_{1:L} \triangleq \{M_1 \ldots M_L\}\) where each \(M_i \in \mathcal{M} \subseteq \{\mathbb{R}^{d_u \times d_x}\}\). with the action at time step \(t\) given by
\[
\pi_{M_{1:L}}(u_{1:T}) \triangleq u_t + \sum_{j=1}^{L} M_j w_{t-j}.
\]

A natural class of matrices from which the above feedback matrices can be picked is given by fixing a number \(\gamma > 0\) and picking matrices spectrally bounded by \(\gamma\), i.e. \(\mathcal{M}_\gamma \triangleq \{M | M \in \mathbb{R}^{d_u \times d_x}, \|M\| \leq \gamma\}\). We further overload the notation for a disturbance action policy to incorporate an open-loop control sequence \(u_{1:T}\), defined as \(\pi_{M_{1:L}}(u_{1:T}) \triangleq u_t + \sum_{j=1}^{L} M_j w_{t-j}\).

### 2.4 Planning Regret With Disturbance-Action Policies

As discussed, a natural idea to deal with adversarial process disturbance is to plan (potentially oblivious to it), producing a sequence of open loop \((u_{1:T})\) actions and appending an adaptive controller to correct for the disturbance online. However the disturbance in practice could have structure across rollouts, which can be leveraged to improve the plan \((u_{1:T})\), with

\footnote{For notational simplicity, we do not include an affine offset \(c_t\) in the definition of our linear policy; this can be included with no change in results across the paper.}
the knowledge that we have access to an adaptive controller. To capture this, we define the notion of an online planning game and the associated notion of planning regret below.

**Definition 2.3 (Online Planning).** It is defined as an $N$ round/rollout game between a player and an adversary, with each round defined as follows:

- At every round $i$ the player given the knowledge of a new dynamical system $f_{i:T} = \{ f^i \ldots f^T \}$, proposes a policy $\pi_{i:T} = \{ \pi^i \ldots \pi^T \}$.
- The adversary then proposes a noise sequence $w_{i:T}$ and a cost sequence $c_{i:T}$.
- A rollout of policy $\pi_{i:T}$ is performed on the system $f_{i:T}$ with disturbances $w_{i:T}$ and the cost suffered by the player $J_i(\pi_{i:T}) \triangleq J(\pi_{i:T}|f_{i:T}, w_{i:T})$.

The task of the controller is to minimize the cost suffered. We measure the performance of the controller via the following objective, defined as Planning-Regret, which measures the performance against the metric of producing the best-in-hindsight open-loop plan, having been guaranteed the optimal adaptive control policy for every single rollout. The notion of adaptive control policy we use is the disturbance-action policy class defined in (2.1). In the Appendix (Section D), we discuss the expressiveness of the disturbance-actions policies. In particular, they generalize linear policies for stationary systems and lend convexity. Formally, planning regret is defined as follows:

\[
\sum_{i=1}^{N} J_i(\pi_{i:T}) - \min_{u_{i:T}} \sum_{i=1}^{N} \left( \min_{M_{i:L}} J_i(\pi_{M_{i:L}}(u_{1:T})) \right)
\]

### 3 Main Algorithm and Result

In this section we propose the algorithm iGPC (Iterative Gradient Perturbation Controller; Algorithm 1) to minimize Planning Regret. The algorithm at every iteration given an open-loop policy $u_{1:T}$ performs a rollout overlaying an online DAC adaptive controller GPC (Algorithm 2). Further the base policy $u_{1:T}$ is updated by performing gradient descent (or any other local policy improvement) on $u$ fixing the offsets suggested by GPC. We show the following guarantee on average planning regret for Algorithm 1 for linear dynamical systems.

**Theorem 3.1.** Let $\mathcal{U} \subseteq \mathbb{R}^{d_u}$ be a bounded convex set with diameter $U$. Consider the online planning game (Definition 2.3) with linear dynamical systems $\{ AB^i_{1:T} \}_{i=1}^{N}$ satisfying Assumption 2.2 and cost functions $\{ c_{1:T} \}_{i=1}^{N}$ satisfying Assumption 2.1. Then we have that Algorithm 1 (when executed with appropriate parameters), for any sequence of disturbances $\{ w_{i:T}\}_{i=1}^{N}$ with each $\| w^i \| \leq W$ and any $\gamma \geq 0$, produces a sequence of actions with planning regret bounded as

\[
\frac{1}{N} \sum_{i=1}^{N} J_i(\pi_{i:T}) - \min_{u_{i:T}} \left( \sum_{i=1}^{N} \min_{M_{i:L}} J_i(\pi_{M_{i:L}}(u_{1:T})) \right) \leq \tilde{O}\left( \frac{1}{\sqrt{T}} + \frac{1}{\sqrt{N}} \right),
\]

where $M_{\gamma} = \{ M | M \in \mathbb{R}^{d_u,d_x} , \| M \| \leq \gamma \}$.

The $\tilde{O}$ notation above subsumes factors polynomial in system parameters $\kappa, \gamma, \delta^{-1}, U, W, G$ and $\log(T)$. A restatement of the theorem with all the details is present in the Appendix (Section C).

**Algorithm 1 iGPC Algorithm**

**Parameters:** Set $\mathcal{U}$, $\eta_{out}$ : Learning Rate

1. Initialize $u_{1:T}^0 \in \mathcal{U}$ arbitrarily.
2. for $i = 1 \ldots N$ do
3. Receive a dynamical system $f_{i:T}$.
4. **Rollout** $u_{1:T}^i$ with GPC $\triangleright$ (Alg. 2),
   \[
   \{ x_{i:T}^i, a_{i:T}^i, w_{i:T}^i, o_{i:T}^i \} = \text{GPRollout}(f_{i:T}, u_{1:T}^i).
   \]
5. **Update:** Compute the update to the policy,
   \[
   \nabla_i = \nabla_{u_{1:T}} J(u_{1:T} + o_{1:T}^i|f_{i:T}, w_{1:T}^i) \\
   u_{1:T}^{i+1} = \text{Proj}_\mathcal{U}(u_{1:T}^i - \eta_{out} \nabla_i).
   \]
6. end for

### 4 Algorithm and Analysis

In this section we provide an overview of the derivation of the algorithm and the proof for Theorem 3.1. The formal proof is deferred to Appendix (Section C).
We introduce an online learning setting that is the basis of algorithms for planning in linear dynamical systems. The setting applies more generally to control/planning and our formulation of planning regret in linear dynamical systems is a specification of this setting.

### 4.1 Nested OCO and Planning Regret

**Setting:** Consider an online convex optimization (OCO) problem, where the iterations have a nested structure, divided into inner and outer iterations. Fix two convex sets and inner sets. After every one out of outer iterations, the player chooses a point . After that there is a sequence of inner iterations, where the player chooses a point . The goal of the player is to minimize Planning Regret:

$$\text{Planning Regret} = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{T} \sum_{t=1}^{T} f_t^i(x_t, y_t^i) - \min_{x^* \in \mathcal{K}_1} \sum_{i=1}^{N} \min_{y^i \in \mathcal{K}_2} \frac{1}{T} \sum_{t=1}^{T} f_t^i(x^*, y_t^i)$$

To state a general result, we assume access to two online learners denoted by , that are guaranteed to provide sub-linear regret bounds over linear cost functions on the sets respectively in the standard OCO model. We denote the corresponding regrets achieved by . A canonical algorithm for online linear optimization (OLO) is online gradient descent [Zin03], which is what we use in the sequel. The theory presented here applies more generally. Algorithm 3 lays out a general algorithm for the Nested-OCO setup.

**Algorithm 3 Nested-OCO Algorithm**

**Require:** Algorithms \( A_1, A_2 \).

1. Initialize \( x_1 \in \mathcal{K}_1 \) arbitrarily.
2. for \( i = 1 \ldots N \)
3. Initialize \( y_0^i \in \mathcal{K}_2 \) arbitrarily.
4. for \( t = 1 \ldots T \)
5. Define loss function over \( \mathcal{K}_2 \) as
6. Update \( y_{t+1} \leftarrow A_2(h_0^i \ldots h_t^i) \).
7. end for
8. Define loss function over \( \mathcal{K}_1 \) as
9. Update \( x_{t+1} \leftarrow A_1(g_1, \ldots, g_i) \).
10. end for
Algorithm 1). with memory framework proposed by [AHM15] and This nuance in the reduction from Theorem 4.1 is only the number of previous rounds, say $S$. We then define a proxy loss denoted by GPCLoss, corresponding to the cost incurred by a stationary closed-loop policy executing for the previous $S$ time steps. Formally, given a dynamical system $f_{1:S}$, perturbations $w_{1:S}$, a cost function $c$, a non-stationary open-loop policy $u_{1:S}$, GPCLoss is a function of closed-loop transfer $M_{1:L}$ defined as follows. Consider the following iterations with $y_1 = 0$,

$$a_j \triangleq u_j + \sum_{r=1}^{L} M_r w_{j-r},$$

$$y_j \triangleq f_{j-1}(y_{j-1}, a_{j-1}) + w_{j-1} \quad \forall j \in [1,S],$$

$$\text{GPCLoss}(M_{1:L}, u_{1:S}, w_{L+1:S-1}, f_{1:S-1}, c) \triangleq c(y_S, a_S). \quad (4.1)$$

The algorithm updates by performing a gradient descent step on this loss, i.e. $M_{t+1,1:L} = M_{t,1:L} - \eta \nabla M \text{GPCLoss}()$. The proof proceeds by showing that the actual cost and its gradient is closely tracked by their proxy GPC Loss counterparts with the difference proportional to the learning rate (Appendix Lemma C.4). Choosing the learning rate appropriately then completes the proof.

5 Experiments

We demonstrate the efficacy of the proposed approach on two sets of experiments: the theory-aligned one performs basic checks on linear dynamical systems; the subsequent set demonstrates the benefit on highly non-linear systems distilled from practical applications. In the following we provide a detailed description of the setup and the results are presented in Figure 1.

5.1 Experimental Setup

We briefly review the methods that we compare to: The ILQG agent obtains a closed loop policy via the Iterative Linear Quadratic Gaussian algorithm [TL05], proposed originally to handle Gaussian noise while planning on non-linear systems, on the simulator dynamics, and then executes the policy thus obtained. This approach does not learn from multiple rollouts and, if the dynamics are fixed, provides a constant (across rollouts) baseline.

The Iterative Learning Control (ILC) agent [AQN06] learns from past trajectories to refine its actions on the next real-world rollout. We provide precise details in the Appendix (Section E). Finally, the IGPC agent adapts Algorithm 1 by replacing the policy update step (Line 5) with a LQR step on locally linearized dynamics. In all our experiments, the metric we compare is the number of real-world rollouts required to achieve...
a certain loss value on the real dynamics. For further details on the setups and hyperparameter tuning please see Appendix (Section E).

5.2 Linear Control

This section considers a discrete-time Double Integrator (detailed below), a basic kinematics model well studied in control theory. This linear system (described below) is subject to a variety of perturbations that vary either within or across episodes,

\[ A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \]

We pick three instructive perturbation models: First, as a sanity check, we consider constant offsets. While both ILC and IGPC adapt to this change, IGPC is quicker in doing so as evident by the cost on the first rollout itself. In the second, we treat constant offsets that gradually increase in magnitude from zero with rollouts/episodes. While gradual inter-episodic changes are well suited to ILC, IGPC still offers consistently better performance. The final scenario considers time-varying sinusoidal perturbations subject to rollout-varying phase shifts. In contrast to the former setups, such conditions make intra-episodic learning crucial for good performance. Indeed, IGPC outperforms alternatives here by a margin, reflecting the benefit of rollout-adaptive feedback policy in the regret bound.

5.3 Non-linear Control with Approximate Models

Here, we consider the task of controlling non-linear systems whose real-world characteristics are only partially known. In the cases presented below, the proposed algorithm IGPC either converges to the optimal cost with fewer rollouts (for Quadrotor), or, even disregarding speed of convergence, offers a better terminal solution quality (for Reacher). These effects are generally more pronounced in situations where the model mismatch is severe.

Concretely, consider the following setup: the agent is scored on the cost incurred on a handful of sequentially executed real-world rollouts on a dynamical system \( g(x, u) \); all the while, the agent has access to an inaccurate simulator \( f(x, u) \neq g(x, u) \). In particular, while limited to simply observing its trajectories in the real world \( g \), the agent is permitted to compute the function value and Jacobian of the simulator \( f(x, y) \) along arbitrary state-action pairs. The disturbances here are thus the difference between \( g \) and \( f \) along the state-action pairs visited along any given real world rollout. Here, we also consider a statistically-omnipotent infeasible agent ILQR (oracle) that executes the Iterative Linear Quadratic Regulator algorithm \cite{LT04} directly via Jacobians of the real world dynamics \( g \) (a cheat), indicating a lower bound on the best possible cost.

Quadrotor with Wind The simulator models an underactuated planar quadrotor (6 dimensional state, 2 dimensional control) attempting to fly to \((1, 1)\) from the origin. The real-world dynamics differ from the simulator in the presence of a dispersive force field \((x\hat{i} + y\hat{j})\), to accommodate wind. The cost is measured as the distance squared from the origin along with a quadratic penalty on the actions.

Reacher with Impulse The simulator dynamics model a 2-DOF arm (6 dimensional state, 2 dimensional control) attempting to place its end-effector at a pre-specified goal. The true dynamics \( g \) differs from the simulator in the application of periodic impulses to the center of mass of the arm links. The cost involves a quadratic penalty on the controls and the distance of the end effector from the goal.

In both scenarios, JAX-based \cite{BFH18} differentiable implementations of the underlying dynamics were adapted from \cite{GHS21}.

6 Conclusion

In this work, we cast the task of disturbance-resilient planning into a regret minimization framework. We outline a gradient-based algorithm that refines an open loop plan in conjunction with a near instance-optimal closed loop policy. We provide a theoretical justification for the approach by proving a vanishing average regret bound. We also demonstrate our approach on simulated examples and observe empirical gains compared to the popular iterative learning control (ILC) approach.

There are several exciting directions for future work. On the theoretical front, one interesting direction is to establish lower bounds on regret for the setting we consider here. This may provide an indication of how sub-optimal (in terms of regret) the approach we present here is and potentially guide the way towards improved algorithms. On the algorithmic front, extending our approach to handle partially observable settings would be of significant practical interest (e.g.,
settings where there is a mismatch in the sensor model in addition to the dynamics model). Finally, a particular exciting direction is to experimentally explore the benefits in terms of sim-to-real transfer conferred by our approach.
References

[ABH+19] Naman Agarwal, Brian Bullins, Elad Hazan, Sham Kakade, and Karan Singh. Online control with adversarial disturbances. In Proceedings of the 36th International Conference on Machine Learning, pages 111–119, 2019.

[ACM07] Hyo-Sung Ahn, YangQuan Chen, and Kevin L Moore. Iterative learning control: Brief survey and categorization. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 37(6):1099–1121, 2007.

[AHM15] Oren Anava, Elad Hazan, and Shie Mannor. Online learning for adversaries with memory: price of past mistakes. In Advances in Neural Information Processing Systems, pages 784–792, 2015.

[AHS19] Naman Agarwal, Elad Hazan, and Karan Singh. Logarithmic regret for online control. arXiv preprint arXiv:1909.05062, 2019.

[AQN06] Pieter Abbeel, Morgan Quigley, and Andrew Y Ng. Using inaccurate models in reinforcement learning. In Proceedings of the 23rd international conference on Machine learning, pages 1–8. ACM, 2006.

[AYS11] Yasin Abbasi-Yadkori and Csaba Szepesvári. Regret bounds for the adaptive control of linear quadratic systems. In Proceedings of the 24th Annual Conference on Learning Theory, pages 1–26, 2011.

[Ber05] Dimitri Bertsekas. Dynamic programming and optimal control, volume 1. Athena scientific Belmont, MA, 2005.

[BFH+18] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

[BKT19] Maria-Florina Balcan, Mikhail Khodak, and Ameet Talwalkar. Provable guarantees for gradient-based meta-learning. In International Conference on Machine Learning, pages 424–433. PMLR, 2019.

[BM99] Alberto Bemporad and Manfred Morari. Robust model predictive control: A survey. In Robustness in identification and control, pages 207–226. Springer, 1999.

[CHK+18] Alon Cohen, Avinatan Hasidim, Tomer Koren, Nevena Lazic, Yishay Mansour, and Kunal Talwar. Online linear quadratic control. In International Conference on Machine Learning, pages 1028–1037, 2018.

[DMM+18] Sarah Dean, Horia Mania, Nikolai Matni, Benjamin Recht, and Stephen Tu. Regret bounds for robust adaptive control of the linear quadratic regulator. In Advances in Neural Information Processing Systems, pages 4188–4197, 2018.

[dR96] Dick de Roover. Synthesis of a robust iterative learning controller using an h/sub/spl inf-in approach. In Proceedings of 35th IEEE Conference on Decision and Control, volume 3, pages 3044–3049. IEEE, 1996.

[FAL17] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. arXiv preprint arXiv:1703.03400, 2017.

[GHS+21] Paula Gradu, John Hallman, Daniel Suo, Alex Yu, Naman Agarwal, Udaya Ghai, Karan Singh, Cyril Zhang, Anirudha Majumdar, and Elad Hazan. Deluca—a differentiable control library: Environments, methods, and benchmarking. arXiv preprint arXiv:2102.09968, 2021.
[Haz16] Elad Hazan. Introduction to online convex optimization. *Foundations and Trends in Optimization*, 2(3-4):157–325, 2016.

[HKS20] Elad Hazan, Sham Kakade, and Karan Singh. The nonstochastic control problem. In Aryeh Kontorovich and Gergely Neu, editors, *Proceedings of the 31st International Conference on Algorithmic Learning Theory*, volume 117 of *Proceedings of Machine Learning Research*, pages 408–421, San Diego, California, USA, 08 Feb–11 Feb 2020. PMLR.

[HWMZ20] Lukas Hewing, Kim P Wabersich, Marcel Menner, and Melanie N Zeilinger. Learning-based model predictive control: Toward safe learning in control. *Annual Review of Control, Robotics, and Autonomous Systems*, 3:269–296, 2020.

[LCRM04] Wilbur Langson, Ioannis Chryssochoos, SV Raković, and David Q Mayne. Robust model predictive control using tubes. *Automatica*, 40(1):125–133, 2004.

[LT04] Weiwei Li and Emanuel Todorov. Iterative linear quadratic regulator design for nonlinear biological movement systems. In *ICINCO (1)*, pages 222–229, 2004.

[May14] David Q Mayne. Model predictive control: Recent developments and future promise. *Automatica*, 50(12):2967–2986, 2014.

[Moo12] Kevin L Moore. *Iterative learning control for deterministic systems*. Springer Science & Business Media, 2012.

[MSR05] David Q Mayne, Maria M Seron, and SV Raković. Robust model predictive control of constrained linear systems with bounded disturbances. *Automatica*, 41(2):219–224, 2005.

[MTR19] Horia Mania, Stephen Tu, and Benjamin Recht. Certainty equivalent control of lqr is efficient. *arXiv preprint arXiv:1902.07826*, 2019.

[OH05] David H Owens and Jari Hätönen. Iterative learning control—an optimization paradigm. *Annual reviews in control*, 29(1):57–70, 2005.

[PBMG62] Lev S Pontryagin, VG Boltyanskii, RV Gamkrelidze, and EF Mishchenko. The mathematical theory of optimal processes, translated by kn trirogoff. *New York*, 1962.

[RB17] Ugo Rosolia and Francesco Borrelli. Learning model predictive control for iterative tasks, a data-driven control framework. *IEEE Transactions on Automatic Control*, 63(7):1883–1896, 2017.

[Ros15] I Michael Ross. *A primer on Pontryagin’s principle in optimal control*. Collegiate publishers, 2015.

[SB18] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[Sim20] Max Simchowitz. Making non-stochastic control (almost) as easy as stochastic. *arXiv preprint arXiv:2006.05910*, 2020.

[SSH20] Max Simchowitz, Karan Singh, and Elad Hazan. Improper learning for non-stochastic control, 2020.

[Ste94] Robert F Stengel. *Optimal control and estimation*. Courier Corporation, 1994.

[Ted20] Russ Tedrake. *Underactuated Robotics: Algorithms for Walking, Running, Swimming, Flying, and Manipulation (Course Notes for MIT 6.832)*, 2020.

[TL05] Emanuel Todorov and Weiwei Li. A generalized iterative lqg method for locally-optimal feedback control of constrained nonlinear stochastic systems. In *Proceedings of the 2005, American Control Conference, 2005.*, pages 300–306. IEEE, 2005.
[WCSB19] Nolan Wagener, Ching-An Cheng, Jacob Sacks, and Byron Boots. An online learning approach to model predictive control. *arXiv preprint arXiv:1902.08967*, 2019.

[ZD98] Kemin Zhou and John Comstock Doyle. *Essentials of robust control*, volume 104. Prentice hall Upper Saddle River, NJ, 1998.

[ZDG96] Kemin Zhou, John C. Doyle, and Keith Glover. *Robust and Optimal Control*. Prentice-Hall, Inc., USA, 1996.

[Zin03] Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 928–936, 2003.
A Relationship with Meta-Learning

In this section, we detail how the nested-OCO formulation proposed in the paper can be used to derive up to a small constant factor, the gradient-based meta-learning results presented in [BKT19] by reducing their setting to the nested-OCO setting and applying Algorithm 3. The reduction requires setting the \( x, y \) space, i.e. \( K_1, K_2 \) to be \( \Theta \subseteq \mathbb{R}^d \) and a ball in \( \mathbb{R}^d \) of diameter \( D^* \) (according to the notation in [BKT19]). Further, we set the function \( f_t(x, y) \triangleq l_t(x + y) \). The reduction recovers the same guarantee as the result in [BKT19] up to a factor of 2.

We note that [BKT19] provide an algorithm that works without the knowledge of \( D^* \), but such an extension is standard in OCO literature and can be handled similarly to [BKT19]. Further we acknowledge that for the particular problem considered in [BKT19], the constant factor is important as a straightforward algorithm also achieves the same rate if constant factors are ignored, a fact highlighted in the original paper. On the other hand, our formulation allows for a stronger comparator even in the [BKT19] setup.

We would like to highlight that our nested-OCO setup allowing for different \( x, y \) spaces is more general than the setup typically considered in initialization-based meta-learning. Owing to this generality, the algorithm we provide naturally performs a gradient step on the true function value for the outer loop as opposed to a distance-based function as in [BKT19]. Further exploring the effectiveness of our algorithm for meta-learning is left as interesting future work.

B Comparison of Policy Classes

In this section we make a comparison of various policy classes introduced in the paper.

**Linear state-action policies.** In classical optimal control with full observation, the cost function is typically assumed to be quadratic in the state and control, i.e.

\[
c_t(x, u) = x^\top Q x + u^\top R u.
\]

Under this assumption and infinite horizon time-invariant \((A_i, B_i = A_j, B_j)\) linear dynamical system (LDS), and assuming independent Gaussian disturbances at every time step, the optimal solution can be computed using the Bellman optimality equations (see e.g. [Tea20]). This gives rise to the Discrete time Algebraic Riccati Equation (DARE), whose solution is a linear policy commonly denoted by

\[
u_t = K x_t.
\]

The finite-horizon solution is also computable and results in a non-stationary linear policy, where the linear policies converge exponentially fast to the first solution of the Riccati equation. It is thus reasonable to consider the class of all linear policies as a reasonable comparator class. Denote the class of all linear policies as

\[
\Pi_L = \{ K \in \mathbb{R}^{d_x \times d_x} \}.
\]

**State of the art: linear dynamical control policies.** A generalization of static state-action control policies is that of linear dynamical controllers (LDC). LDC are particularly useful for partially observed LDS and maintain their own internal dynamical system according to the observations in order to recover the hidden state of the system. A formal definition is below.

**Definition B.1** (Linear Dynamic Controllers). A linear dynamic controller \( \pi \) is a linear dynamical system \((A_\pi, B_\pi, C_\pi, D_\pi)\) with internal state \( s_t \in \mathbb{R}^{d_\pi} \), input \( x_t \in \mathbb{R}^{d_x} \) and output \( u_t \in \mathbb{R}^{d_u} \) that satisfies

\[
s_{t+1} = A_\pi s_t + B_\pi x_t, \quad u_t = C_\pi s_t + D_\pi x_t.
\]
LDC are state-of-the-art in terms of performance and prevalence in control applications involving LDS, both in the full and partial observation settings. They are known to be theoretically optimal for partially observed LDS with quadratic cost functions and normally distributed noise, but are more widely used. Denote the class of all LDC as

\[ \Pi_{\text{LDC}} = \{ A \in \mathbb{R}^{d_x \times d_x}, B \in \mathbb{R}^{d_x \times d_x}, C \in \mathbb{R}^{d_x \times d_x}, D \in \mathbb{R}^{d_x \times d_x} \} . \]

**Disturbance-Action Controllers (DAC)** As we have defined earlier, we consider an even more general class of policies, i.e. that of disturbance-action control. For linear time invariant systems, this policy class is more general than that of LDC and linear controllers, in the sense that for every LDS there exists a DAC which outputs exactly the same controls on the same system and sequence of noises. With a finite and fixed \( H \), an approximate version of this statement is true. The precise approximation statement and formal proof can be found in \( ABH^{+19} \). A similar statement can be made for LDC as well.

However we note that all of the above statements hold only in linear time invariant case. In the time varying case, these generalizations are not necessarily true, however note that we are using disturbance action feedback control only as an adaptive control policy to correct against noise, and it is added upon an open-loop plan.

**C Main Theorem and Proof**

We provide the following restatement of Theorem 3.1 with details regarding the parameters and the dependence on the system parameters. To state the results concisely, we assume that all the appropriate assumed constants, i.e. \( \kappa, \gamma, G, \beta, U, W \) are greater than 1. This is done to upper bound the sum of two constants by twice their product. All the results hold by replacing any of these constants by the max of the constant and 1.

**Theorem C.1.** Let \( U \subseteq \mathbb{R}^{d_a} \) be a bounded convex set with diameter \( U \). Consider the online planning game (Definition 2.3) with linear dynamical systems \( \{ AB_1^T \}_{i=1}^N \) satisfying Assumption 2.2 and cost functions \( \{ c_1 \} \) satisfying Assumption 2.7. Then we have that Algorithm 7 when executed with appropriate parameters, for any sequence of disturbances \( \{ w_1^T \}_{i=1}^N \) with \( \| w_i^T \| \leq W \) and any \( \gamma \geq 0 \), produces a sequence of actions with planning regret bounded as

\[
\frac{1}{N} \left( \sum_{i=1}^N J_i^\pi - \min_{u_1:T \in U} \left( \sum_{i=1}^N \min_{M_1:L \in M} J_i^\pi (M_1:L(u_1:T)) \right) \right) \leq \left( \frac{c_{\text{in}} \log^2 (T)}{\sqrt{T}} + \frac{c_{\text{out}}}{\sqrt{N}} \right),
\]

where \( M_\gamma = \{ M \mid M \in \mathbb{R}^{d_u \times d_x}, \| M \| \leq \gamma \} \) and \( c_{\text{in}}, c_{\text{out}} \) are constants depending on system parameters as follows

\[
c_{\text{out}} = \tilde{O} \left( GU (U + \gamma LW) \kappa^2 \delta^{-2} \right),
\]

\[
c_{\text{in}} = \tilde{O} \left( \sqrt{\gamma^3 \kappa^4 \delta^{-3} \beta G^2 L^5 W^3 (U + \gamma LW)^2} \right).
\]

Here \( \tilde{O} \) subsumes constant factors and factors poly-logarithmic in the the arguments of \( \tilde{O} \). To achieve the above bound, Algorithm 7 is to be executed with parameters, learning rate \( \eta_{\text{out}} = \frac{GU \delta^{-2} (U + \kappa LW + W) \sqrt{N}}{\gamma \kappa^2 L^2} \) and with the inner execution of Algorithm 4 is performed with parameters \( \eta_{\text{in}} = \frac{G \delta^{-2} (U + \kappa LW + W) \sqrt{N}}{\gamma \kappa^2 L^2} \) and \( S = \delta^{-1} \log (\eta_{\text{in}}) \).

**C.1 Requisite Definitions**

Before proving the theorem we set up some useful definitions. Fix a linear dynamical system \( AB_1^T \) and a disturbance sequence \( w_1:T \). For any sequence \( u_1:T \in U \) and \( M_1:T;L \in M_\gamma \), we define \( T \) functions \( x_1:T(\cdot | AB_1^T, w_1:T), a_1:T(\cdot | AB_1^T, w_1:T) \), denoting the action played and the state visited at time \( t \) upon
execution of the policies together. Herein we drop \(AB_{1:T}, w_{1:T}\) from the notation when clear from the context. Formally, consider the following definitions for all \(t\),

\[
a_t(u_{1:T}, M_{1:T,1:L}) \triangleq u_t + \sum_{r=1}^{L} M_{t,r} w_{t-r}
\]

\[
x_1(u_{1:T}, M_{1:T,1:L}) \triangleq 0 \quad x_{t+1}(u_{1:T}, M_{1:T,1:L}) \triangleq A_t x_t(u_{1:T}, M_{1:T,1:L}) + B_t a_t + w_t
\]

Given a sequence of cost functions \(c_{1:T}(x,u) : \mathbb{R}^{d_x \times d_u} \rightarrow \mathbb{R}\), satisfying Assumption 2.1, define via an overload of notation, the cost functions \(c_t\) as a function of \(u_{1:T}, M_{1:T,1:L}\) as follows

\[
\forall t \in [1 : T], \quad c_t(u_{1:t}, M_{1:T,1:L}) = c_t(x_t(u_{1:t}, M_{1:T,1:L}), a_t(u_{1:T}, M_{1:T,1:L}))
\]

Naturally, according to our definition of the total cost \(J\) of the rollout we get that

\[
J(u_{1:T}, M_{1:T,1:L}) = \frac{1}{T} \sum_{t=1}^{T} c_t(u_{1:t}, M_{1:T,1:L})
\]

Next, we expand upon the recursive definition of \(x_t(\cdot, \cdot)\) via the following operators,

**Definition C.2.** Given a linear dynamical system \(AB_{1:T}\), define the following transfer matrices

\[
\forall j \in [T], \forall k \in [j + 1, T] \quad T_{j\rightarrow k} \in \mathbb{R}^{d_x \times d_u} \quad T_{j\rightarrow k} \triangleq \begin{cases} I & \text{if } k = j + 1 \\ \Pi_{t=j+2}^{k} A_t & \text{otherwise} \end{cases}
\]

Additionally given a disturbance sequence \(w_{1:T}\), define the following linear operator over matrix sequences \(M_{1:T,1:L}\)

\[
\forall j \in [T], \forall k \in [j + 1, T] \quad \psi^M_{j\rightarrow k}(M_{1:T,1:L}) : [\mathbb{R}^{d_x \times d_u}]^{T \times L} \rightarrow \mathbb{R}^{d_x}
\]

\[
\psi^M_{j\rightarrow k}(M_{1:T,1:L}) = \sum_{t=j}^{k-1} \left( T_{t\rightarrow k} B_t \left( \sum_{r=1}^{L} M_{t,r} w_{k-r} \right) \right)
\]

It can be observed via unrolling the recursion and the definitions above that

\[
x_t(u_{1:T}, M_{1:T,1:L}) = \sum_{j=1}^{t-1} T_{j\rightarrow t}(B_j u_j + w_j) + \psi^M_{1\rightarrow t}(M_{1:T,1:L}).
\]

Since \(x_t, a_t\) are linear functions of \(u_{1:T}, M_{1:T,1:L}\), therefore we have that \(c_t(u_{1:T}, M_{1:T,1:L})\) is a convex function of its arguments. The next lemma further shows that the gradient of the total cost with respect to the argument \(u_{1:T}\) is bounded, as stated in the following lemma.

**Lemma C.3.** Given a linear system \(AB_{1:T}\) satisfying Assumption 2.2, a bounded disturbance sequence \(w_{1:T}\) and a cost sequence \(c_t\) satisfying Assumption 2.1, then for any \(\gamma \geq 0, U, M_{1:T,1:L} \in \mathcal{M}_\gamma\) be two sequences, then we have that

\[
\left\| \nabla_{u_j} \left( \sum_{t=1}^{T} c_t(u_{1:T}, M_{1:T,1:L}) \right) \right\| \leq 2GK\delta^{-2}(\kappa U + \kappa \gamma LW + W)
\]

We provide the proof of the lemma further in the section. Using the lemma we are now ready to prove Theorem 3.1.
Proof of Theorem 3.1

Let fix a particular rollout \( i \). Let \( AB^{i}_{1:T} \) be the dynamical system and \( w^{i}_{1:T} \) be the disturbance supplied. Further \( u^{i}_{1:T} \) be the open loop control sequence played at round \( i \) and \( M^{i}_{1:T;1:L} \) be the disturbance feedback sequence played by the GPC subroutine. By definition we have that the state achieved 
\[
x^{i}_{t} = x_{t}(u^{i}_{1:T}, M^{i}_{1:T;1:L}) \quad a^{i}_{t} = a_{t}(u^{i}_{1:T}, M^{i}_{1:T;1:L})
\]
We have for convenience dropped the system and disturbance from our notation. The total cost at round \( i \) incurred by the algorithm by definition is
\[
J = \sum_{i=1}^{N} \frac{1}{T} \left( \sum_{t=1}^{T} c_{t}^{i}(u^{i}_{1:T}, M^{i}_{1:T;1:L}) \right)
\]
Fix the sequence of comparators to be \( \hat{u}_{1:T}, \{ \hat{M}^{i}_{1:L} \}_{i=1}^{N} \). The comparator cost by definition then is
\[
\hat{J} = \sum_{i=1}^{N} \frac{1}{T} \left( \sum_{t=1}^{T} c_{t}^{i}(\hat{u}_{1:T}, T M^{i}_{1:L}) \right),
\]
where given a sequence \( v_{a:b} \), we define the tiling operator \( T_{k} \), which creates a nested sequence of outer length \( k \) by tiling with copies of the sequence \( v_{a:b} \) i.e. \( T_{k}v_{a:b} = [v_{a:b}, v_{a:b}, \ldots, v_{a:b}] \). We therefore have the following calculation for the regret which follows from the convexity of the cost function \( c_{t} \) with respect to \( u, M \) as established before,
\[
\sum_{i=1}^{N} \sum_{t=1}^{T} \left( c_{t}^{i}(u^{i}_{1:T}, M^{i}_{1:T;1:L}) - c_{t}^{i}(\hat{u}_{1:T}, T M^{i}_{1:L}) \right)
\]
\[
\leq \sum_{i=1}^{N} \sum_{t=1}^{T} \left( \nabla u c_{t}^{i}(u^{i}_{1:T}, M^{i}_{1:T;1:L})(u^{i}_{t} - \hat{u}_{1:T}) + \nabla M c_{t}^{i}(u^{i}_{1:T}, M^{i}_{1:T;1:L})(M^{i}_{1:T;1:L} - \hat{M}^{i}_{1:L}) \right)
\]
\[
= \sum_{i=1}^{N} \sum_{t=1}^{T} \left( \nabla u c_{t}^{i}(u^{i}_{1:T}, M^{i}_{1:T;1:L})(u^{i}_{t} - \hat{u}_{1:T}) \right) + \sum_{i=1}^{N} \sum_{t=1}^{T} \left( \nabla M c_{t}^{i}(u^{i}_{1:T}, M^{i}_{1:T;1:L})(M^{i}_{1:T;1:L} - \hat{M}^{i}_{1:L}) \right)
\]
We analyze the both the terms above separately. We begin by analyzing the first term.

**Outer Regret:**

Consider the following calculation
\[
\sum_{i=1}^{N} \sum_{t=1}^{T} \left( \nabla u c_{t}^{i}(u^{i}_{1:T}, M^{i}_{1:T;1:L})(u^{i}_{t} - \hat{u}_{1:T}) \right) = \sum_{j=1}^{T} \sum_{i=1}^{N} \nabla u_{j} \left( \sum_{t=1}^{T} c_{t}^{i}(u^{i}_{1:T}, M^{i}_{1:T;1:L}) \right) (u^{i}_{j} - \hat{u}_{j}).
\]
Note that by definition of the algorithm, we have that for all \( i, j \)
\[
u^{i+1}_{j} = \text{Proj}_{M}(u^{i}_{j} - \eta_{out} g^{u}_{ij}),
\]
which via the pythagorean inequality implies that
\[
||u^{i+1}_{j} - \hat{u}_{j}||^{2} \leq ||u^{i}_{j} - \eta_{out} g^{u}_{ij} - \hat{u}_{j}||^{2}
\]
Combining the above equations we immediately get that
\[
\sum_{i=1}^{N} \sum_{t=1}^{T} \left( \nabla u c_{t}^{i}(u^{i}_{1:T}, M^{i}_{1:T;1:L})(u^{i}_{t} - \hat{u}_{1:T}) \right) \leq \sum_{j=1}^{T} \sum_{i=1}^{N} \frac{1}{2} \left( \eta_{out} ||g^{u}_{ij}||^{2} + \frac{||(u^{i}_{j} - \hat{u}_{j})^{2} - (u^{i+1}_{j} - \hat{u}_{j})^{2}||}{\eta_{out}} \right) \leq \sum_{j=1}^{T} \frac{1}{2} \left( \eta_{out} \left( \sum_{i=1}^{N} ||g^{u}_{ij}||^{2} \right) + \frac{||(u^{i}_{j} - \hat{u}_{j})^{2}||}{\eta_{out}} \right) \leq 2 U G \kappa \delta^{-2} (\kappa U + \kappa \gamma LW + W) T \sqrt{N}
\]
where the last inequality follows using Lemma C.3 and choice of $\eta_{\text{out}}$.

**Inner Regret:** Next we analyze the second Inner Regret term. Before doing so we recommend the reader to re-familiarize with the notations defined in Definition C.2 and Equations C.1, C.2, C.3. We will also need the following further definitions again for a fixed rollout. Therefore given a dynamical system $AB_{1:T}$, a disturbance sequence $w_{1:T}$, and an open loop sequence $u_{1:T}$ define the notion of surrogate state at time $t$ which is parameterized by a lookback window $S$ and is a function of an input sequence $M_{1:L} \in \mathbb{R}^{d_x \times d_x}$. Intuitively it corresponds to the state achieved by executing the stationary policy $M_{1:L}$ along with $u_{1:T}$ for $S$ time steps, starting at time $t - S$ with a resetted state. This is exactly the computation performed in the GPCLoss definition in Equation 4.1. We can use the linear operator $\psi$ defined in Definition C.2 for an alternative and succinct definition as follows.

$$\hat{x}_t(u_{1:T}, M_{1:L}) = \sum_{j=t-S}^{t-1} T_j \to t(B_j u_j + w_j) + \psi_{t-S}^M(T_t M_{1:L}).$$  \hfill (C.6)

Further given a cost function $c_t$, we can use the above definition to also define a surrogate cost

$$\hat{c}_t(u_{1:T}, M_{1:L}) = c_t \left( \hat{x}_t(u_{1:T}, M_{1:L}), u_t + \sum_{j=1}^{L} M_j w_{t-j} \right).$$  \hfill (C.7)

It can be observed now by the definition of Algorithm 2 the sequence $M_{t+1,1:L}$ played by the algorithm is chosen iteratively as follows

$$M_{t+1,1:L} = \text{Proj}_{M_{\gamma}}(M_{t,1:L} - \eta_{\text{in}} \nabla_M \hat{c}_t(u_{1:T}, M_{t,1:L})).$$  \hfill (C.8)

To proceed with the proof we will need the following lemma

**Lemma C.4.** Consider a linear system $AB_{1:T}$ satisfying Assumption 2.2 a bounded disturbance sequence $w_{1:T}$ and a sequence of cost functions $c_{1:T}$ satisfying Assumption 2.1. Given any open loop sequence $u_{1:T} \in U$ and a closed-loop matrix sequence $M_{1:T,1:L} \in \mathcal{M}_\gamma$ generated through the iteration specified in Equation C.8 we have that the following properties hold for all $t \in [T]$

- For all $j > t$, $\nabla_{M_{j,1:L}} c_t(u_{1:T}, M_{1:T,1:L}) = 0$.
- For all $j < t$, $\|\nabla_{M_{j,1:L}} c_t(u_{1:T}, M_{1:T,1:L})\| \leq \kappa^2 G(U + \gamma LW)LW^{-1}(1 - \delta)^{t-j}$.
- For all $t$, $\|\nabla_{M_{1:L}} \hat{c}_t(u_{1:T}, M_{1:L})\| \leq GLW(U + \gamma LW) \left(1 + \frac{3}{2\gamma}\right)$.
- Furthermore, for any $\hat{M}_{1:L} \in \mathcal{M}_\gamma$ and for any $t$, we have that

$$\sum_{j=t-S}^{t} \nabla_{M_{j,1:L}} c_t(u_{1:T}, M_{1:T,1:L})(M_{j,1:L} - \hat{M}_{1:L}) \leq \nabla_{M_{1:L}} \hat{c}_t(u_{1:T}, M_{1:L})(M_{t,1:L} - \hat{M}_{1:L})$$

$$+ 20 \eta_{\text{in}} \log^2(\eta_{\text{in}}) \gamma \kappa^4 \delta^{-3} \beta G^2 L^3 W^3 (U + \gamma LW)^2$$

We are now ready to analyze the inner regret term. We analyze this term for one particular rollout say $i$ (thereby dropping $i$ from our notation). We get the following series of calculations,
\[
\sum_{t=1}^{T} \left( \nabla_{MC_t(u_{1:T}, M_{1:T,1:L})}(M_{1:T,1:L} - T_T \hat{M}_{1:L}) \right) \\
= \sum_{t=1}^{T} \sum_{j=1}^{T} \left( \nabla_{M_{j,1:L}}c_t(u_{1:T}, M_{1:T,1:L})(M_{j,1:L} - \hat{M}_{1:L}) \right) \\
= \sum_{t=1}^{T} \sum_{j=1}^{t} \left( \nabla_{M_{j,1:L}}c_t(u_{1:T}, M_{1:T,1:L})(M_{j,1:L} - \hat{M}_{1:L}) \right) \\
\leq \sum_{t=1}^{T} \sum_{j=t-S}^{t} \left( \nabla_{M_{j,1:L}}c_t(u_{1:T}, M_{1:T,1:L})(M_{j,1:L} - \hat{M}_{1:L}) \right) + 2\kappa^2 \gamma GLW(U + \gamma LW) \delta^{-2}(1 - \delta)^S \\
\leq \sum_{t=1}^{T} \left( \nabla_{M_{t,1:L}}c_t(u_{1:T}, M_{t,1:L})(M_{t,1:L} - \hat{M}_{1:L}) \right) + 22T \eta_n \log^2(\eta_n) \gamma \kappa^4 \delta^{-3} \beta G^2 L^3 W^3 (U + \gamma LW)^2,
\]

where the statements follow via repeated application of Lemma \[\text{C.4}\] and the choice of \(S = \delta^{-1} \log(\eta_n)\). To analyse further once again via a similar argument as in the case of the outer regret regarding projected gradient descent with learning rate \(\eta_n\), we get that,

\[
\sum_{t=1}^{T} \left( \nabla_{M_{t,1:L}}c_t(u_{1:T}, M_{t,1:L})(M_{t,1:L} - \hat{M}_{1:L}) \right) \\
\leq \sum_{t=1}^{T} \left( \frac{\eta_n}{2} \|g_t\|^2 + \frac{\|M_{t,1:L} - \hat{M}_{1:L}\|^2 - \|M_{t+1,1:L} - \hat{M}_{1:L}\|^2}{2\eta_n} \right) \\
\leq \frac{\eta_n T}{2} \|g_t\|^2 + \frac{\|M_{t,1:L} - \hat{M}_{1:L}\|^2}{2\eta_n}
\]

Combining the above equations, Equation \[\text{C.9}\] and the choice of \(\eta_n\), we get that the inner regret is bounded as,

\[
\sum_{t=1}^{T} \left( \nabla_{M_{t,1:L}}c_t(u_{1:T}, M_{t,1:L})(M_{t,1:L} - T_T \hat{M}_{1:L}) \right) \leq \hat{O} \left( \sqrt{T \gamma^3 \kappa^4 \delta^{-3} \beta G^2 L^3 W^3 (U + \gamma LW)^2} \right)
\]

Combining the outer and inner regret terms we finish the proof. \(\square\)

In the remaining subsections we prove Lemmas \[\text{C.3, C.4}\] thereby finishing the proof of Theorem 3.1.

### C.2 Proof of Lemma \[\text{C.3}\]

In this section we prove Lemma \[\text{C.3}\] Before the proof we establish some other lemmas which will be useful to us.

**Lemma C.5.** Given a linear system \(AB_{1:T}\) satisfying Assumption \[2.2\] then the transfer matrices defined in Definition \[\text{C.2}\] are bounded as follows

\[
\forall j, k \in [T], [j + 1, T] \quad \|T_{j \to k}\| \leq (1 - \delta)^{k-j-1}
\]
Proof of Lemma C.5. If \( k = j + 1 \) then by definition and Assumption 2.2,
\[
\|T_{j \rightarrow k}\| = \|I\| \leq 1.
\]
Otherwise, again by definition and Assumption 2.2,
\[
\|T_{j \rightarrow k}\| \leq (\Pi_{t=j+2}^{k}\|A_{t}\|) \leq (1-\delta)^{k-j-1}.
\]

Lemma C.6. Given a linear system \( AB_{1:T} \) satisfying Assumption 2.2, a bounded disturbance sequence \( w_{1:T} \) and a cost sequence \( c_{t} \) satisfying Assumption 2.1, then for any \( \gamma \geq 0, U, M_{1:T;1:L} \in M_{\gamma} \) be two sequences, the following bounds hold for \( x_{t}, a_{t} \) for all \( t, \)
\[
\|x_{t}(u_{1:T}, M_{1:T;1:L})\| \leq \delta^{-1}(\kappa U + \kappa\gamma LW + W),
\]
\[
\|a_{t}(u_{1:T}, M_{1:T;1:L})\| \leq U + \gamma LW.
\]
Furthermore we have that for all \( j, t \in [T] \) we have that
\[
\left\| \frac{\partial x_{t}(u_{1:T}, M_{1:T;1:L})}{\partial u_{j}} \right\| \leq \begin{cases} \kappa(1-\delta)^{t-j-1} & \text{if } j < t \\ 0 & \text{otherwise} \end{cases}
\]
\[
\left\| \frac{\partial a_{t}(u_{1:T}, M_{1:T;1:L})}{\partial u_{j}} \right\| = \begin{cases} 1 & \text{if } j = t \\ 0 & \text{otherwise} \end{cases}
\]
Furthermore we have that for \( j, t \in [T] \) and \( r \in [L] \), we have that
\[
\left\| \frac{\partial x_{t}(u_{1:T}, M_{1:T;1:L})}{\partial M_{jr}} \right\| \leq \begin{cases} \kappa W(1-\delta)^{t-j-1} & \text{if } j < t \\ 0 & \text{otherwise} \end{cases}
\]
\[
\left\| \frac{\partial a_{t}(u_{1:T}, M_{1:T;1:L})}{\partial M_{jr}} \right\| \leq \begin{cases} W & \text{if } j = t \\ 0 & \text{otherwise} \end{cases}
\]

Proof. From the definition in Equation C.1 it follows that
\[
\|a_{t}(u_{1:T}, M_{1:T;1:L})\| \leq \|u_{t}\| + \sum_{r=1}^{L} \|M_{t,r}\||w_{t-r}|| \leq U + \gamma LW.
\]
Also from the definition it follows that
\[
\left\| \frac{\partial a_{t}(u_{1:T}, M_{1:T;1:L})}{\partial u_{j}} \right\| = \|\delta_{j}I\| = \begin{cases} 1 & \text{if } j = t \\ 0 & \text{otherwise} \end{cases}
\]
From the expansion in Equation C.4 we have that
\[
\|x_{t}(u_{1:T}, M_{1:T;1:L})\| \leq \sum_{j=1}^{t-1}(\|T_{j \rightarrow t}(B_{j}u_{j} + w_{j})\| + \|\psi_{t \rightarrow t}(M_{1:T;1:L})\|)
\]
\[
\leq \sum_{j=1}^{t-1}(\|T_{j \rightarrow t}\||(B_{j}u_{j} + w_{j})\|) + \sum_{j=1}^{t-1}(\|T_{j \rightarrow t}\||(\sum_{r=1}^{L} \|M_{j,r}\||w_{j-r}||)
\]
\[
\leq (\kappa U + \kappa\gamma LW + W) \sum_{j=1}^{t-1}(1-\delta)^{t-j-1} \quad \text{(Definition C.2 & \Delta-inequality)}
\]
\[
\leq \frac{1}{\delta}(\kappa U + \kappa\gamma LW + W) \quad \text{(Lemma C.5 and definitions)}
\]
Also from the definition it follows that for \( j \geq t \),
\[
\frac{\partial x_t(u_1:T, M_1:T, L_1)}{\partial u_j} = 0,
\]
and if \( j < t \), we have that
\[
\left\| \frac{\partial x_t(u_1:T, M_1:T, L_1)}{\partial u_j} \right\| \leq \| T_{j \rightarrow t} B_j \| \leq \kappa (1 - \delta)^{t-j-1} \quad \text{(Lemma C.3)}
\]
From the definition in Equation C.1 it follows that for any \( r \in \mathbb{R}^L \)
\[
\left\| \frac{\partial a_t(u_1:T, M_1:T, L_1)}{\partial M_{j,r}} \right\| = \| \delta_j I \otimes w_{t-r}^\top \| \leq \begin{cases} W & \text{if } j = t \\ 0 & \text{otherwise} \end{cases}
\]
From the expansion in Equation C.4 it follows that for any \( r \) and \( j \geq t \),
\[
\frac{\partial x_t(u_1:T, M_1:T, L_1)}{\partial M_{j,r}} = 0,
\]
and if \( j < t \), we have that
\[
\left\| \frac{\partial x_t(u_1:T, M_1:T, L_1)}{\partial M_{j,r}} \right\| \leq \| T_{j \rightarrow t} B_j (I \otimes w_{j-r}^\top) \| \leq \kappa W (1 - \delta)^{t-j-1} \quad \text{(Lemma C.5)}
\]

We are now ready to prove Lemma C.3.

**Proof of Lemma C.3.** Consider the following calculations for all \( j, t \), following from Lemma C.6
\[
\left\| \nabla u_j \left( c_t(u_1:T, M_1:T, L_1) \right) \right\| 
\leq G \max(\| x_t(u_1:T, M_1:T, L_1) \|, \| a_t(u_1:T, M_1:T, L_1) \|) \left( \left\| \frac{\partial x_t(u_1:T, M_1:T, L_1)}{\partial u_j} \right\| + \left\| \frac{\partial a_t(u_1:T, M_1:T, L_1)}{\partial u_j} \right\| \right)
\leq \begin{cases} G \kappa \delta^{-1}(\kappa U + \kappa \gamma LW + W)(1 - \delta)^{t-j-1} & \text{if } j < t \\ G \kappa \delta^{-1}(\kappa U + \kappa \gamma LW + W) & \text{if } j = t \\ 0 & \text{otherwise} \end{cases}
\]
Therefore we have that,
\[
\left\| \nabla u_j \left( \sum_{i=1}^{T} c_i(u_1:T, M_1:T, L_1) \right) \right\| \leq 2G \kappa \delta^{-2}(\kappa U + \kappa \gamma LW + W)
\]

**C.3 Proof of Lemma C.4**

In this section we prove Lemma C.4. To this end we will need the following lemma that is the extension of Lemma C.6 to surrogate states.

**Lemma C.7.** Given a linear system \( AB_1:T \) satisfying Assumption 2.2, a bounded disturbance sequence \( w_1:T \) and a cost sequence \( c_t \) satisfying Assumption 2.7, then for any \( \gamma \geq 0, U, \) let \( u_1:T \in U, M_1:L \in M_\gamma \) be two sequences, then we have that for all \( j, t \in [T] \),

\[
\]
Furthermore we have that for $t \in [T]$ and $r \in [L]$, we have that
\[
\left\| \frac{\partial \hat{x}_t(u_{1:T}, M_{1:L})}{\partial M_r} \right\| \leq \delta^{-1} \kappa W
\]

**Proof.** From the expansion in Equation [C.6], we have that
\[
\left\| \hat{x}_t(u_{1:T}, M_{1:L}) \right\| \leq \sum_{j=t-S}^{t-1} \left( \|T_{j\rightarrow t}(B_j u_j + w_j)\| + \|\psi_{t-S\rightarrow t}(T_T M_{1:L})\| \right)
\]
\[
\leq \sum_{j=t-S}^{t-1} \left( \|T_{j\rightarrow t}\| \|B_j u_j + w_j\| + \|T_{j\rightarrow t}\| \left( \sum_{r=1}^{L} \|M_r\| \|w_{j-r}\| \right) \right) \quad \text{(Definition [C.2] & \Delta-inequality)}
\]
\[
\leq (\kappa U + \kappa \gamma LW + W) \sum_{j=t-S}^{t-1} (1 - \delta)^{t-j-1} \quad \text{(Lemma [C.5] and Definitions)}
\]
\[
\leq \frac{1}{\delta} (\kappa U + \kappa \gamma LW + W)
\]
From the expansion in Equation [C.6] it follows that
\[
\left\| \frac{\partial x_t(u_{1:T}, M_{1:T;1:L})}{\partial M_r} \right\| \leq \left\| \sum_{j=t-S}^{t-1} T_{j\rightarrow t} B_j I \otimes \tilde{u}_{j-r}^T \right\| \leq \delta^{-1} \kappa W \quad \text{(Lemma [C.6])}
\]

We are now ready to prove Lemma [C.4].

**Proof of Lemma [C.4]** Since for any $j > t$, by Lemma [C.6] we have that
\[
\frac{\partial x_t(u_{1:T}, M_{1:T;1:L})}{\partial M_{j;1:L}} = 0, \quad \frac{\partial a_t(u_{1:T}, M_{1:T;1:L})}{\partial M_{j;1:L}} = 0,
\]
it immediately follows that for all $j > t$,
\[
\nabla_{M_{j;1:L}} c_t(u_{1:T}, M_{1:T;1:L}) = 0.
\]
Furthermore again from Lemma [C.6] we have that for all $j < t$ and for all $r \in [L]$,
\[
\left\| \frac{\partial x_t(u_{1:T}, M_{1:T;1:L})}{\partial M_{j;r}} \right\| \leq \kappa W (1 - \delta)^{t-j-1}
\]
and further if $j < t$ and for all $r \in [L]$,
\[
\frac{\partial a_t(u_{1:T}, M_{1:T;1:L})}{\partial M_{j;r}} = 0
\]
Therefore, since the cost function $c_t$ satisfies the Assumption [2.1] using Lemma [C.6] we have that for all $j < t$ and for any $r \in [L]$
\[
\left\| \nabla_{M_{j;r}} c_t(u_{1:T}, M_{1:T;1:L}) \right\| \leq G \|x_t(u_{1:T}, M_{1:T;1:L})\| \left\| \frac{\partial x_t(u_{1:T}, M_{1:T;1:L})}{\partial M_{j;r}} \right\| \leq G \kappa \delta^{-1} W (\kappa U + \kappa \gamma LW + W) (1 - \delta)^{t-j}
\] (C.9)
Using Lemma C.7 for the surrogate states and using Assumption 2.1, we have that for all \( t \) and for all \( r \in [L] \),

\[
\| \nabla_{M_t} c_t(u_{1:T}, M_{t+1:L}) \| \leq 2G\kappa\delta^{-2}W(\kappa U + \kappa\gamma LW + W)
\]

Since the gradient is bounded according to the above calculation and the \( M_{t+1:L} \) are generated via gradient descent with a learning rate \( \eta_n \), it is immediate that for any \( j, k \in [T] \) and for any \( r \in [L] \),

\[
\| M_{j:r} - M_{k:r} \| \leq \eta_n |j - k| \cdot 2G\kappa\delta^{-2}W(\kappa U + \kappa\gamma LW + W) \tag{C.10}
\]

Given the above we show that for any execution the surrogate states and the real states are close to each other. To this end consider the following calculations.

\[
\| x_t(u_{1:T}, M_{t+1:L}) - \hat{x}_t(u_{1:T}, M_{t+1:L}) \| \\
\leq \left\| \sum_{j=t-S}^{t-1} T_{j \to t} (B_j u_j + w_j) + \psi_t^M(M_{1:T,1:L}) - \sum_{j=t-S}^{t-1} T_{j \to t} (B_j u_j + w_j) - \psi_{t-S}^M(T_t M_{t+1:L}) \right\| \\
= \left\| \sum_{j=1}^{t-1} \left( T_{j \to t} (B_j u_j + w_j + \sum_{r=1}^{L} M_{j:r} w_{j-r}) \right) + \sum_{j=t-S}^{t-1} T_{j \to t} \left( \sum_{r=1}^{L} (M_{j:r} - M_{1:r}) w_{j-r} \right) \right\| \\
\leq (\kappa U + \kappa\gamma LW + W) (\delta^{-1}(1 - \delta)^S + 2\eta_n\kappa\delta^{-2}S^2GLW^2) \tag{C.11}
\]

Furthermore, note by definitions that

\[
\sum_{j=t-S}^{t-1} \frac{\partial x_t(u_{1:T}, M_{t+1:L})}{\partial M_{j+1:L}} = \frac{\partial \hat{x}_t(u_{1:T}, M_{t+1:L})}{\partial M_{t+1:L}} \tag{C.12}
\]

Before moving further, consider the following calculations

\[
\sum_{j=t-S}^{t-1} \left( \nabla_{M_{j+1:L}} c_t(u_{1:T}, M_{j+1:L}) \right) \\
= \sum_{j=t-S}^{t-1} \left( \frac{\partial x_t(u_{1:T}, M_{1:t+1:L})}{\partial M_{j+1:L}} \nabla_x c_t(x_t(u_{1:T}, M_{1:t+1:L}), a_t(u_{1:T}, M_{1:t+1:L})) \right) \\
= \sum_{j=t-S}^{t-1} \frac{\partial x_t(u_{1:T}, M_{1:t+1:L})}{\partial M_{j+1:L}} \left( (\nabla_x c_t(\hat{x}_t(u_{1:T}, M_{1:t+1:L}), a_t(u_{1:T}, M_{1:t+1:L}))) + v) \right)
\]

where

\[
\| v \| \triangleq \| \nabla_x c_t(x_t(u_{1:T}, M_{1:t+1:L}), a_t(u_{1:T}, M_{1:t+1:L})) - \nabla_x c_t(\hat{x}_t(u_{1:T}, M_{1:t+1:L}), a_t(u_{1:T}, M_{1:t+1:L})) \| \\
\leq \beta(\kappa U + \kappa\gamma LW + W) (\delta^{-1}(1 - \delta)^S + 2\eta_n\kappa\delta^{-2}S^2GLW^2) \tag{C.13}
\]

using Equation C.11 and the \( \beta \)-smoothness of \( c_t \) via Assumption 2.1. Using Equation C.12 and Lemma C.6, we now get that

\[
\sum_{j=t-S}^{t-1} \left( \nabla_{M_{j+1:L}} c_t(u_{1:T}, M_{1:t+1:L}) \right) \\
= \left( \sum_{j=t-S}^{t-1} \frac{\partial x_t(u_{1:T}, M_{1:t+1:L})}{\partial M_{j+1:L}} \right) \left( \nabla_x c_t(\hat{x}_t(u_{1:T}, M_{1:t+1:L}), a_t(u_{1:T}, M_{1:t+1:L}))) + v) \right) \\
= \frac{\partial \hat{x}_t(u_{1:T}, M_{1:t+1:L})}{\partial M_{t+1:L}} \left( \nabla_x c_t(\hat{x}_t(u_{1:T}, M_{1:t+1:L}), a_t(u_{1:T}, M_{1:t+1:L}))) + v' \right) \tag{C.14}
\]
where \( v' \) is a vector whose norm using Equation C.13 and Lemma C.7 can be bounded as follows
\[
\beta \delta^{-1} LW (\kappa U + \kappa \gamma LW + W) \left( \delta^{-1} (1 - \delta)^S + 2\eta_{\text{in}} \kappa \delta^{-2} S^2 GLW^2 \right).
\]
Now, consider the following computation which follows from Equation C.14 and using the definitions for the \( j = t \) case,
\[
\sum_{j=t-S}^{t} (\nabla M_{j,1:L} c_t(u_{1:T}, M_{1:T,1:L})) = \nabla M_{t,1:L} \hat{c}_t(u_{1:T}, M_{1:L}) + v'.
\]
We can now perform the calculation to relate the gradient inner products for surrogate cost to those of real cost.
\[
\sum_{j=t-S}^{t} (\nabla M_{j,1:L} c_t(u_{1:T}, M_{1:T,1:L})(M_{j,1:L} - \hat{M}_{1:L}))
\]
\[
= \sum_{j=t-S}^{t} (\nabla M_{j,1:L} c_t(u_{1:T}, M_{1:T,1:L})(M_{j,1:L} - \hat{M}_{1:L}) + \nabla M_{j,1:L} c_t(u_{1:T}, M_{1:T,1:L})(M_{j,1:L} - M_{t,1:L}))
\]
\[
\leq \sum_{j=t-S}^{t} (\nabla M_{j,1:L} c_t(u_{1:T}, M_{1:T,1:L})(M_{t,1:L} - \hat{M}_{1:L}) + \eta_{\text{in}} 2G^2 L S^2 \kappa^2 \delta^{-3} W^2 (\kappa U + \kappa \gamma LW + W)^2
\]
\[
\leq \nabla M_{t,1:L} \hat{c}_t(u_{1:T}, M_{1:L})(M_{t,1:L} - \hat{M}_{1:L}) + \beta \delta^{-1} \gamma L^2 W (\kappa U + \kappa \gamma LW + W)^2 \left( \delta^{-1} (1 - \delta)^S + 4\eta_{\text{in}} \kappa^2 \delta^{-2} S^2 G^2 L W^2 \right)
\]
\[
\leq \nabla M_{t,1:L} \hat{c}_t(u_{1:T}, M_{1:L})(M_{t,1:L} - \hat{M}_{1:L}) + 5\eta_{\text{in}} \log^2(\eta_{\text{in}}) \gamma \kappa^2 \delta^{-3} \beta G^2 L^3 W^3 (\kappa U + \kappa \gamma LW + W)^2
\]
where the first inequality follows from applying Equations C.9 C.10 and Lemma C.7, the second last inequality follows from Equations C.15 and C.16 and the last inequality follows from the choice of the parameter \( S = \delta^{-1} \log(\eta_{\text{in}}). \) This finishes the proof. \( \square \)

D Adaptation of Algorithm to General Policies

In this section we provide a more general version of our algorithms 1 and 2, defined for any base outer policy class II. Note that our formal results dont cover this generalization and it is provided with practical use in mind.
Algorithm 4 iGPC Algorithm

Require: [Online] $f_{1:T}$: Dynamical Systems, $w_{1:T}$: Disturbances
Parameters: Policy class: $\Pi$, $\eta_{\text{out}}$: Learning Rate

1: Initialize $\pi_{1:T}^{1} \in \Pi$
2: for $i = 1 \ldots N$ do
3:   Receive the dynamical system $f_{1:T}^i$ for the next rollout.
4:   Rollout: Collect trajectory data by rolling out policy $\pi_{1:T}^i$ with GPC
5:   Update: Compute update to the policy $
6:   \pi_{1:T}^{i+1} = \text{Proj}_{\Pi} (\pi_{1:T}^i - \eta_{\text{out}} \nabla \pi_{1:T} J(\pi_{1:T}^i + \pi(o_{1:T})|f_{1:T}^i, w_{1:T}^i)))$
6: end for

Algorithm 5 GPCRollout

Require: $f_{1:T}$: dynamical system, $\pi_{1:T}$: input policy, [Online] $w_{1:T}$: disturbances.
Parameters: $L$:Window, $\eta_{\text{in}}$: Learning rate, $\gamma$: Feedback bound, $S$: Lookback

1: Initialize $M_{1:1:L} = \{M_{1,j}^L\}_{j=1}^L \in \mathcal{M}_\gamma.$
2: Set $w_i = 0$ for any $i \leq 0$
3: for $t = 1 \ldots T$ do
4:   Compute GPC Offset
5:   Play action
6:   Observe state $x_{t+1}$.
7:   Compute perturbation
8:   Update $M_{t+1:1:L}$ for the next round as:
9: end for
10: return $x_{1:T}, a_{1:T}, w_{1:T}, o_{1:T}$.

E Details of ILQR/ILC/IGPC Algorithms

To succinctly state the algorithms define the following policy which takes as arguments a nominal trajectory $\hat{x}_{1:T} \in \mathbb{R}^{d_x}, \hat{u}_{1:T} \in \mathbb{R}^{d_u}$, open-loop gain sequence $k_{1:T}$ and closed-loop gain sequence $K_{1:T}$ and a parameter $\alpha$. The policy defined as $\pi(\alpha, x_{1:T}, k_{1:T}, K_{1:T})$, in the sequel executes the following standard rollout on a dynamical system $f_{1:T}$.

$$a_t = \hat{u}_t + \alpha k_t + K(x_{t-1} - \hat{x}_{t-1})$$
$$x_{t+1} = f_t(x_t, a_t)$$

24
Before stating the algorithm we also need the following quadratic approximation of the cost function $c$ around pivots $x_0, u_0$

$$Q(c, x_0, u_0)(x, u) \triangleq \nabla c_x(x_0, u_0)(x - x_0) + \nabla c_u(x_0, u_0)(u - u_0)$$

$$+ \frac{1}{2}([x, u] - [x_0, u_0])^T \nabla^2 c(x, u)([x, u] - [x_0, u_0]) \quad (E.1)$$

Algorithm 6 now presents a combined layout for ILQG, ILC and IGPC.

Algorithm 6 Iterative Planning Algorithm

**Require:** $g_{1:T}$ Real Dynamical Systems, $g_{1:T}$ Simulator.

1. Initialize starting sequence of actions $u_{1:T}^0$.
2. Initialize sequence of open loop $k_{1:T}^0$ and closed loop gains $K_{1:T}^0 = 0$.
3. for $i = 1 \ldots N$ do
4. Rollout the Policy:
   - **ILQG:** Standard Rollout on $f_{1:T}$.
     $$x_{1:T}^i, u_{1:T}^i = \text{Rollout}(f_{1:T}, \pi(\alpha, x_{1:T}^{i-1}, u_{1:T}^{i-1}, k_{1:T}^{i-1}))$$
   - **ILC:** Standard Rollout on $g_{1:T}$.
     $$x_{1:T}^i, u_{1:T}^i = \text{Rollout}(g_{1:T}, \pi(\alpha, x_{1:T}^{i-1}, u_{1:T}^{i-1}, k_{1:T}^{i-1}))$$
   - **IGPC:** GPCRollout on $g_{1:T}$,
     $$x_{1:T}^i, u_{1:T}^i = \text{GPCRollout}(g_{1:T}, \pi(\alpha, x_{1:T}^{i-1}, u_{1:T}^{i-1}, k_{1:T}^{i-1}))$$
5. Update: Obtain $k_{1:T}^i \in \mathbb{R}^{d_u}, K_{1:T}^i \in \mathbb{R}^{d_u \times d_x}$ as the optimal non-stationary affine policy to the following LQG problem.

$$\min \mathbb{E}_z \left[ \sum_{t=1}^{T} Q(c_t, x_t^i, u_t^i)(x_t, u_t) \right]$$

subject to

$$x_{t+1} - x_t^i = \frac{\partial f_t(x_t^i, u_t^i)}{\partial x_t^i}(x_t - x_t^i) + \frac{\partial f_t(x_t^i, u_t^i)}{\partial u_t^i}(u_t - u_t^i) + z_t$$

where $z_t$ are independent Gaussians of any non-zero variance.

6. end for

### E.1 Hyperparameter Selection for Experiments

ILQG, ILC, IGPC in particular share one hyperparameter $\alpha$ which corresponds essentially to a step size towards the updated policy. As is common in implementations, this hyperparameter is adjusted online during the run of the algorithm using a simple retracting line search from a certain upper bound $\alpha^+$. We optimize over choices for $\alpha^+$ for ILC and report the best performance obtained as baseline. For IGPC, we use the same $\alpha^+$ as obtained for ILC and the same line search for strategy for selecting $\alpha$. We include the rollouts needed for line search in the rollout cost of the algorithm. Further, IGPC introduces certain other hyperparameters, $L$ the window, $S$ the lookback, and $\eta_m$, the inner learning rate. We chose $L, S = 3$ arbitrarily for our experiments and tuned $\eta_m$ per experiment. Overall we observed that for every experiment, the selection of $\eta$ was robust in terms of performance.