A Regret Minimization Approach to Iterative Learning Control

Naman Agarwal [1] **Elad Hazan** [1 2] **Anirudha Majumdar** [1 2] **Karan Singh** [3]

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 learn 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

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

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 $\mathcal{A}$ may be judged via the aggregate cost it suffers on a cost function sequence $c_1, \dots c_T$ along its state-action trajectory $(x_1^{\mathcal{A}}, u_1^{\mathcal{A}}, \dots)$:

$$J(\mathcal{A}|w_{1:T}) = \frac{1}{T}\sum_{t=1}^{T} c_t\left(x_t^{\mathcal{A}}, u_t^{\mathcal{A}}\right).$$

For deterministic systems, an optimal open-loop control sequence $u_1 \dots u_T$ can be chosen to minimize the cost sequence. The presence of unanticipated disturbances often necessitates the superposition of a *closed-loop* correction policy π to obtain meaningful performance. Such closed-loop policies can modify the open-loop control sequence $u_1 \dots u_T$ to $u'_t = \pi(u_{1:t}, x_{1:t})$ which is a function of the observed history till time t, 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 Π). 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:

Planning Regret

$$\sum_{i=1}^{N} J(\mathcal{A}|w^i_{1:T}) - \min_{u^\star_{1:T}} \sum_{i=1}^{N} \min_{\pi_i^\star \in \Pi} J(u^\star_{1:T}, \pi_i^\star | w^t_{i,1:T})$$

*Equal contribution [1]Google AI Princeton, Princeton, NJ, USA [2]Princeton University, Princeton, NJ, USA [3]Microsoft Research, Redmond, WA, USA. Correspondence to: Naman Agarwal <namanagarwal@google.com>, Elad Hazan <ehazan@cs.princeton.edu>, Anirudha Majumdar <ani.majumdar@princeton.edu>, Karan Singh <karansingh@microsoft.com>.

Proceedings of the 38th International Conference on Machine Learning, PMLR 139, 2021.

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.

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.

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 (Pontryagin et al., 1962; Ross, 2015), and differs from dynamic programming approaches (Sutton & Barto, 2018).

3. We accomodate arbitrary disturbance processes, and choose regret as a performance metric. This is a significant deviation from the literature on optimal and robust control (Zhou et al., 1996; Stengel, 1994), and follows the lead of the recent paradigm of non-stochastic control (Agarwal et al., 2019a; Hazan et al., 2020; Simchowitz et al., 2020).

4. Our approach leverages multiple real-world rollouts. This access model is most similar to the iterative learning control (ILC) paradigm (Owens & Hätönen, 2005; Ahn et al., 2007). 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 (Bertsekas, 2005; Zhou et al., 1996; Tedrake, 2020) 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 (Zhou & Doyle, 1998) compute the best controller under worst-case noise.

Recent work in machine learning (Abbasi-Yadkori & Szepesvári, 2011; Dean et al., 2018; Mania et al., 2019; Cohen et al., 2018; Agarwal et al., 2019b) 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 (Agarwal et al., 2019a; Hazan et al., 2020; Simchowitz et al., 2020; Simchowitz, 2020).

Planning with inaccurate models. Model predictive control (MPC) (Mayne, 2014) provides a general scheme for planning with inaccurate models. MPC operates by applying model-based planning, (eg. iLQR (Li & Todorov, 2004; Todorov & Li, 2005)), in a receding-horizon manner. MPC can also be extended to robust versions (Bemporad & Morari, 1999; Mayne et al., 2005; Langson et al., 2004) 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 (Wagener et al., 2019). 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* (Hewing et al., 2020; Rosolia & Borrelli, 2017) 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., (Abbeel et al., 2006)). For further details regarding ILC, we refer the reader to the text (Moore, 2012). Robust versions of ILC have also been developed in the control theory literature (de Roover, 1996), 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 (Finn et al., 2017) 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 (Balcan et al., 2019). We further detail certain improvements/advantages our algorithm and analysis provides over the results in (Balcan et al., 2019). We believe this connection with Meta-Learning to be of independent interest.

2. Problem Setting

2.1. Notation

The norm $\|\cdot\|$ refers to the ℓ_2 norm for vectors and spectral norm for matrices. For any natural number n, the set $[n]$ refers to the set $\{1, 2 \dots n\}$. We use the notation $v_{a:b} \triangleq \{v_a \dots v_b\}$ 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. $\forall j \in [a, b], v_j \in S$; $\mathrm{Proj}_S(v_{a:b})$ represents the element-wise ℓ_2 projection onto to the set S. $v_{a:b,c:d}$ denotes a sequence of sequences, i.e. $v_{a:b,c:d} = \{v_{a,c:d} \dots v_{b,c:d}\}$ with $v_{a,c:d} = \{v_{a,c} \dots v_{a,d}\}$.

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 $f_{1:T} = \{f_t | f_t : \mathbb{R}^{d_x} \times \mathbb{R}^{d_u} \to \mathbb{R}^n\}$. The system produces a T-length sequence of states $(x_1, \dots x_{T+1})$ when subject to an T-length sequence of actions $(u_1 \dots u_T)$ and disturbances $\{w_1, \dots w_T\}$ according to the following dynamical equation[1]

$$x_{t+1} = f_t(x_t, u_t) + w_t.$$

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, ie. $u_t = \pi_t(x_1, \dots x_t)$ for some non-stationary policy $\pi_{1:T} = \{\pi_1, \dots \pi_T\}$. 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 $x_{t+1} = f_t(x_t, u_t) + w_t, u_t = \pi_t(x_1 \dots x_t)$, thus produced is a sequence determined by $w_{1:T}$, fixing the policy, and the system.

A ***rollout*** *of horizon* $\mathbf{T}$ *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 $\{c_t\} : \mathbb{R}^n \times \mathbb{R}^d \to \mathbb{R}$ the ***loss*** of executing a policy π on the dynamical system f with a particular disturbance sequence given by $w_{1:T}$ is defined as

$$J(\pi_{1:T} | f_{1:T}, w_{1:T}) \triangleq \frac{1}{T} \left[\sum_{\tau=1}^{T} c_t(x_t, u_t) \right].$$

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 $\|x\|, \|u\| \le D$ for some $D > 0$, then $\|\nabla_x c_t(x, u)\|, \|\nabla_u c_t(x, u)\| \le GD$.*
- ***Smoothness:*** *There exists a constant β 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 $AB_{1:T} = \{(A_t, B_t) \in \mathbb{R}^{d_x, d_x} \times \mathbb{R}^{d_x, d_u}\}_{t=1}^T$ 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 (κ, δ)- strongly stable for some $\kappa > 0$ and $\delta \in (0, 1]$, i.e. for if every t, we have that $\|A_t\| \le 1 - \delta, \|B_t\| \le \kappa$.*

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 (κ, δ)- strongly stable if there exists a sequence of matrices $K_{1:T}$, such that for every t, we have that $\|A_t - B_t K_t\| \le 1 - \delta, \|B_t\|, \|K_t\| \le \kappa$. A

[1] For the sake of simplicity, we do not consider a terminal cost, and consequently drop the last state from the description.

system satisfying such an assumption can be easily transformed to a system satisfying Assumption 2.2 by setting $A_t = A_t - B_tK_t$. 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 (Cohen et al., 2018; Agarwal et al., 2019a) and the above notion generalizes it to non-stationary systems.

2.3. Policy Classes

Open-Loop Policies. Given a convex set $\mathcal{U} \in \mathbb{R}^{d_u}$, 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 $\Pi_{\mathcal{U}} \triangleq \mathcal{U}^{\otimes T}$.

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 π_1 and π_2.

Linear Policies. Given a matrix $K \in \mathbb{R}^{d_u, d_x}$, a *linear policy* [2] denoted (via an overload of notation) by K is a policy that plays action $u_t = Kx_t$. Such linear state-feedback policies are known to be optimal for the LQR problem and for H_∞ control (Zhou et al., 1996).

Disturbance Action Controllers (DAC). A generalization of the class of linear policies can be obtained via the notion of disturbance-action policies (see (Agarwal et al., 2019a)) defined as follows. A disturbance action policy $\pi_{M_{1:L}}$ of memory length L is defined by a sequence of matrices $M_{1:L} \triangleq \{M_1 \dots M_L\}$ where each $M_i \in \mathcal{M} \subseteq \{\mathbb{R}^{d_u \times d_x}\}$, with the action at time step t given by

$$[\pi_{M_{1:L}}]_t \triangleq \sum_{j=1}^{L} M_j w_{t-j}. \tag{2.1}$$

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 γ, i.e. $\mathcal{M}_\gamma \triangleq \{M | M \in \mathbb{R}^{d_u \times d_x}, \|M\| \le \gamma\}$. We further overload the notation for a disturbance action policy to incorporate an open-loop control sequence $u_{1:T}$, defined as $\pi_{M_{1:L}}(u_{1:T}) \triangleq \{u_t + \sum_{j=1}^{L} M_j w_{t-j}\}_{t=1}^{T}$.

[2]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.

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 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 $f^i_{1:T} = \{f^i_1 \dots f^i_T\}$, proposes a policy $\pi^i_{1:T} = \{\pi^i_1 \dots \pi^i_T\}$.*
- *The adversary then proposes a noise sequence $w^i_{1:T}$ and a cost sequence $c^i_{1:T}$.*
- *A rollout of policy $\pi^i_{1:T}$ is performed on the system $f^i_{1:T}$ with disturbances $w^i_{1:T}$ and the cost suffered by the player $J_i(\pi^i_{1:T}) \triangleq J(\pi^i_{1:T}|f^i_{1:T}, w^i_{1:T})$.*

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 B), 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:

Planning Regret

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

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. [3] 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 $\{w^i_{1:T}\}_{i=1}^N$ with each $\|w^i_t\| \leq W$ and any $\gamma \geq 0$, produces a sequence of actions with planning regret bounded as*

$$\frac{1}{N}\left(\sum_{i=1}^{N} J_i(\pi^i_{1:T}) - \min_{u_{1:T}\in\mathcal{U}}\left(\sum_{i=1}^{N}\min_{M_{1:L}\in\mathcal{M}_\gamma} J_i\left(\pi_{M_{1:L}}(u_{1:T})\right)\right)\right) \leq \tilde{O}\left(\frac{1}{\sqrt{T}}+\frac{1}{\sqrt{N}}\right).$$

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

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

Require: [Online] $f^{1:N}_{1:T}$: Dynamical Systems, $w^{1:N}_{1:T}$: Disturbances , $c^{1:N}_{1:T}$
Parameters: Set : $\mathcal{U}$, η_{out} : Learning Rate

Initialize $u^1_{1:T} \in \mathcal{U}$ arbitrarily.
for $i = 1 \ldots N$ **do**
  Receive a dynamical system $f^i_{1:T}$.
  Rollout the policy $u^i_{1:T}$ with GPC ▷ (Alg. 2),

$$\{x^i_{1:T}, a^i_{1:T}, w^i_{1:T}, o^i_{1:T}\} = \text{GPCRollout}(f^i_{1:T}, u^i_{1:T}).$$

  Update: Compute the update to the policy,

$$\nabla_i = \nabla_{u_{1:T}} J(u^i_{1:T} + o^i_{1:T} | f^i_{1:T}, w^i_{1:T})$$
$$u^{i+1}_{1:T} = \text{Proj}_{\mathcal{U}}\left(u^i_{1:T} - \eta_{\text{out}}\nabla_i\right).$$

end for

[3] In Appendix Section D, we provide a more general version of the algorithm defined for any base policy class.

Algorithm 2 GPCRollout

Require: $f_{1:T}$: dynamical system, $u_{1:T}$: input policy, [Online] $w_{1:T}$: disturbances, $c_{1:T}$: costs.
Parameters: L : Window, η_{in} : Learning rate, γ : Feedback bound, S : Lookback

Initialize $M_{1,1:L} = \{M_{1,j}\}_{j=1}^L \in \mathcal{M}_\gamma$.
Set $w_i = 0$ for all $i \leq 0$.
for $t = 1 \ldots T$ **do**
  Compute Offset: $o_t = \sum_{r=1}^L M_{t,r} \cdot w_{t-r}$.
  Play action: $a_t = u_t + o_t$.
  Suffer Cost: $c_t(x_t, a_t)$
  Observe state: x_{t+1}.
  Compute perturbation:

$$w_t = x_{t+1} - f_t(x_t, a_t).$$

  Do a gradient step on the GPCLoss (4.1)

$$M_{t+1,1:L} = \text{Proj}_{\mathcal{M}_\gamma}\left(M_{t,1:L} - \eta_{\text{in}}\nabla\text{GPCLoss}(arg)\right),$$

  where *arg* captures policy $M_{t,1:L}$, open-loop plan $u_{t-S+1:t}$, disturbances $w_{t-S-L+1:t-1}$, transition $f_{t-S+1,t-1}$, cost c_t in Equation 4.1 and gradient is taken with respect to the M parameter.
end for
return $x_{1:T}, a_{1:T}, w_{1:T}, o_{1:T}$.

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 main building block of our algorithm. 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 (Hazan, 2016), where the iterations have a nested structure, divided into inner and outer iterations. Fix two convex sets $\mathcal{K}_1$ and $\mathcal{K}_2$. After every one out of N outer iterations, the player chooses a point $x_i \in \mathcal{K}_1$. After that there is a sequence of T inner iterations, where the player chooses $y^i_t \in \mathcal{K}_2$ at every iteration. After this choice, the adversary chooses a convex cost function $f^i_t \in \mathcal{F} \subseteq \mathcal{K}_1 \times \mathcal{K}_2 \to \mathbb{R}$, and the player suffers a cost of $f^i_t(x_i, y^i_t)$. The goal of the player is to minimize Planning Regret:

Planning Regret

$$\sum_{i=1}^{N}\sum_{t=1}^{T}\frac{f^i_t(x_i, y^i_t)}{T} - \min_{x^\star\in\mathcal{K}_1}\sum_{i=1}^{N}\min_{y^\star\in\mathcal{K}_2}\sum_{t=1}^{T}\frac{f^i_t(x^\star, y^\star)}{T}$$

To state a general result, we assume access to two online learners denoted by $\mathcal{A}_1, \mathcal{A}_2$, that are guaranteed to provide sub-linear regret bounds over *linear* cost functions on the sets $\mathcal{K}_1, \mathcal{K}_2$ respectively in the standard OCO model. We denote the corresponding regrets achieved by $R_N(\mathcal{A}_1), R_T(\mathcal{A}_2)$. A canonical algorithm for online linear optimization (OLO) is online gradient descent (Zinkevich, 2003), which is what we use in the sequel. The theory presented here applies more generally. [4] Algorithm 3 lays out a general algorithm for the Nested-OCO setup.

Algorithm 3 Nested-OCO Algorithm

Require: Algorithms $\mathcal{A}_1, \mathcal{A}_2$.

Initialize $x_1 \in \mathcal{K}_1$ arbitrarily.
for $i = 1 \dots N$ **do**
  Initialize $y_0^i \in \mathcal{K}_2$ arbitrarily.
  for $t = 1 \dots T$ **do**
    Define loss function over $\mathcal{K}_2$ as

$$h_t^i(y) \triangleq \nabla_y f_t^i(x_i, y_t^i) \cdot y.$$

    Update $y_{t+1} \leftarrow \mathcal{A}_2(h_0^i \dots h_t^i)$.
  end for
  Define loss function over $\mathcal{K}_1$ as

$$g_i(x) \triangleq \sum_{t=1}^{T} \nabla_x f_t^i(x_i, y_t^i) \cdot x.$$

  Update $x_{s+1} \leftarrow \mathcal{A}_1(g_1, ..., g_i)$.
end for

Theorem 4.1. *Algorithm 3 with sub-algorithms $\mathcal{A}_1, \mathcal{A}_2$ with regrets $R_N(\mathcal{A}_1), R_T(\mathcal{A}_2)$ ensures the following regret guarantee on the average planning regret,*

$$\frac{\text{PlanningRegret}}{N} \leq \frac{R_N(\mathcal{A}_1)}{N} + \frac{R_T(\mathcal{A}_2)}{T}.$$

When using Online Gradient Descent as the base algorithm, the average regret scales as $O\left(\frac{1}{\sqrt{N}} + \frac{1}{\sqrt{T}}\right)$.

Proof of Theorem 4.1. Let $x^\star \in \mathcal{K}_1$ be any point and let $y^\star_{1:T} \in \mathcal{K}_2$ be any sequence. We have

$$\begin{aligned}
&\frac{\sum_{i=1}^{N}\sum_{t=1}^{T} f_t^i(x_i, y_t^i) - f_t^i(x^\star, y_i^\star)}{TN} \\
\leq &\frac{\sum_{i=1}^{N}\sum_{t=1}^{T} \nabla_x f_t^i(x_i - x^\star)}{TN} \\
&\qquad + \frac{\sum_{i=1}^{N}\sum_{t=1}^{T} \nabla_y f_t^i(y_t^i - y^\star)}{TN} \\
= &\frac{\sum_{i=1}^{N}[g_i(x_i) - g_i(x^\star)]}{TN} \\
&\qquad + \frac{\sum_{i=1}^{N}\sum_{t=1}^{T}[h_t^i(y_t) - h_t^i(y_i^\star)]}{TN} \\
\leq &\frac{R_N(\mathcal{A}_1)}{N} + \frac{\cdot R_T(\mathcal{A}_2)}{T},
\end{aligned}$$

where the first inequality follows by convexity and the last inequality follows by the regret guarantees and noting that the functions g_i are naturally scaled up by a factor of T. $\square$

4.2. Proof Sketch for Theorem 3.1

The main idea behind the proof is to reduce to the setting of Theorem 4.1. In the reduction the x variable corresponds to the open loop controls $u_{1:T} \in \mathcal{U}$ and the variables y_t^i correspond to the closed-loop disturbance-action policy $M_{t,1:L}^i \in \mathcal{M}_\gamma$. The algorithms $\mathcal{A}_1$ and $\mathcal{A}_2$ are instantiated as Online Gradient Descent with appropriately chosen learning rates.

We begin the reduction by using the observation in (Agarwal et al., 2019a) that costs are convex with respect to the variables u, M, for *linear dynamical systems* with convex costs. With convexity, prima-facie the reduction seems immediate, however this is impeded by the counterfactual notion of policy regret which implies that cost at any time is dependent on previous actions. This nuance in the reduction from Theorem 4.1 is only applicable to the closed loop policies M, the open loop part $u_{1:T}$) on the other hand, follows according to the reduction and hence direct OGD is applied (Line 6, Algorithm 1).

To resolve the issue of the counterfactual dependence, we use the techniques introduced in the OCO with memory framework proposed by (Anava et al., 2015) and recently employed in the work of (Agarwal et al., 2019a). We leverage the underlying stability of the dynamical system to ensure that cost at time t depends only on a bounded 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

[4]Regret for OLO depends on function bounds, which correspond to gradient bounds here. For clarity we omit this dependence from the notation for regret.

with $y_1 = 0$,

$$a_j \triangleq u_j + \sum_{r=1}^{L} M_r w_{j-r},$$
$$y_j \triangleq f_{j-1}(y_{j-1}, a_{j-1}) + w_{j-1} \quad \forall j \in [1, S],$$
$$\text{GPCLoss}(M_{1:L}, u_{1:S}, w_{-L+1:S-1}, f_{1:S-1}, c) \triangleq c(y_S, a_S). \tag{4.1}$$

The algorithm updates by performing a gradient descent step on this loss, i.e. $M^i_{t+1,1:L} = M^i_{t,1:L} - \eta \nabla_M \text{GPCLoss}(\cdot)$. 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 (Todorov & Li, 2005), 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 (Abbeel et al., 2006) *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 (Li & Todorov, 2004) 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 $(x\hat{\mathbf{i}} + y\hat{\mathbf{j}})$, to accomodate wind. The cost is measured as the distance sqaured from the origin along with a quadratic penalty on the actions.

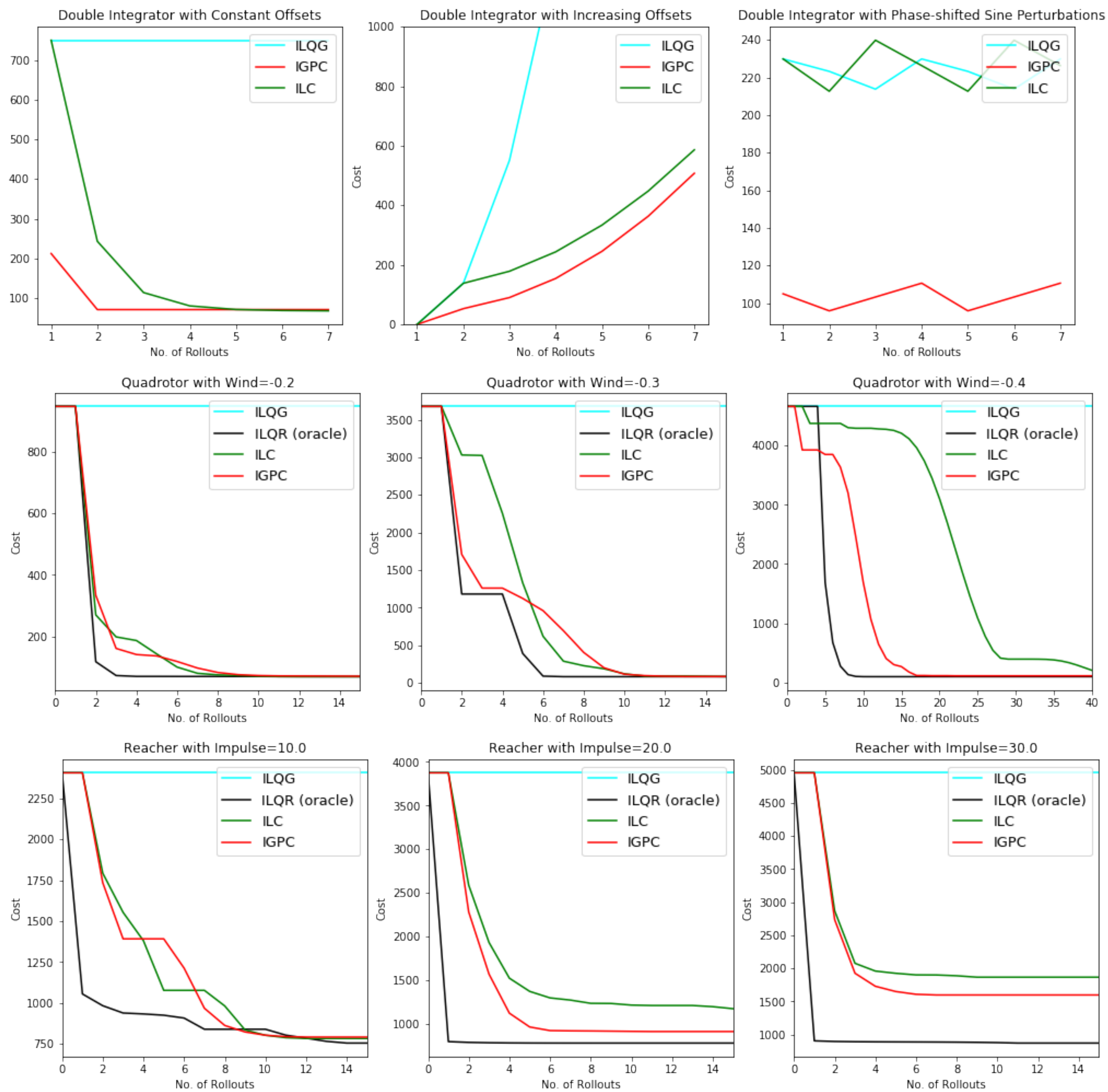

Figure 1. On top is a linear system, Double Integrator, setup subject to: (L) constant offset, (M) offset that increments with rollout count, (R) phase-shifted sinusoidal perturbations. The middle section displays results on the quadrotor environment for varying magnitudes of wind. Bottom figure captures performance on the reacher environment with varying magnitudes of periodic impulses. **ILQR (oracle)** is an infeasible agent with access to Jacobians on the real world.

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 (Bradbury et al., 2018) differentiable implementations of the underlying dynamics were adapted from (Gradu et al., 2021). The implementations along with some further experiments are present at `https://github.com/MinRegret/deluca-igpc`.

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.

A particularly exciting direction for future work is to theoretically and empirically explore the benefits in terms of sim-to-real transfer conferred by our approach. Note that while we consider state-independent perturbations, our regret analysis also extends to affine state-dependent perturbations. Nevertheless, we experimentally demonstrate the potential of our algorithm in the non-linear case. Establishing regret-like theoretical guarantees for non-linear state-dependent perturbations is a challenging avenue for future work.

Acknowledgements

Elad Hazan was supported in part by National Science Foundation Award 1704860. Anirudha Majumdar was partially supported by the Office of Naval Research [Award Number: N00014-18- 1-2873].

References

Abbasi-Yadkori, Y. and Szepesvári, C. Regret bounds for the adaptive control of linear quadratic systems. In *Proceedings of the 24th Annual Conference on Learning Theory*, pp. 1–26, 2011.

Abbeel, P., Quigley, M., and Ng, A. Y. Using inaccurate models in reinforcement learning. In *Proceedings of the 23rd international conference on Machine learning*, pp. 1–8. ACM, 2006.

Agarwal, N., Bullins, B., Hazan, E., Kakade, S., and Singh, K. Online control with adversarial disturbances. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 111–119, 2019a.

Agarwal, N., Hazan, E., and Singh, K. Logarithmic regret for online control. *arXiv preprint arXiv:1909.05062*, 2019b.

Ahn, H.-S., Chen, Y., and Moore, K. L. Iterative learning control: Brief survey and categorization. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 37(6):1099–1121, 2007.

Anava, O., Hazan, E., and Mannor, S. Online learning for adversaries with memory: price of past mistakes. In *Advances in Neural Information Processing Systems*, pp. 784–792, 2015.

Balcan, M.-F., Khodak, M., and Talwalkar, A. Provable guarantees for gradient-based meta-learning. In *International Conference on Machine Learning*, pp. 424–433. PMLR, 2019.

Bemporad, A. and Morari, M. Robust model predictive control: A survey. In *Robustness in identification and control*, pp. 207–226. Springer, 1999.

Bertsekas, D. *Dynamic programming and optimal control*, volume 1. Athena scientific Belmont, MA, 2005.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

Cohen, A., Hasidim, A., Koren, T., Lazic, N., Mansour, Y., and Talwar, K. Online linear quadratic control. In *International Conference on Machine Learning*, pp. 1028–1037, 2018.

de Roover, D. Synthesis of a robust iterative learning controller using an h/sub/spl infin//approach. In *Proceedings of 35th IEEE Conference on Decision and Control*, volume 3, pp. 3044–3049. IEEE, 1996.

Dean, S., Mania, H., Matni, N., Recht, B., and Tu, S. Regret bounds for robust adaptive control of the linear quadratic regulator. In *Advances in Neural Information Processing Systems*, pp. 4188–4197, 2018.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*, 2017.

Gradu, P., Hallman, J., Suo, D., Yu, A., Agarwal, N., Ghai, U., Singh, K., Zhang, C., Majumdar, A., and Hazan, E. Deluca – a differentiable control library: Environments, methods, and benchmarking, 2021.

Hazan, E. Introduction to online convex optimization. *Foundations and Trends in Optimization*, 2(3-4):157–325, 2016. ISSN 2167-3888. doi: 10.1561/2400000013. URL http://dx.doi.org/10.1561/2400000013.

Hazan, E., Kakade, S., and Singh, K. The nonstochastic control problem. In Kontorovich, A. and Neu, G. (eds.), *Proceedings of the 31st International Conference on Algorithmic Learning Theory*, volume 117 of *Proceedings of Machine Learning Research*, pp. 408–421, San Diego, California, USA, 08 Feb–11 Feb 2020. PMLR. URL http://proceedings.mlr.press/v117/hazan20a.html.

Hewing, L., Wabersich, K. P., Menner, M., and Zeilinger, M. N. Learning-based model predictive control: Toward safe learning in control. *Annual Review of Control, Robotics, and Autonomous Systems*, 3:269–296, 2020.

Langson, W., Chryssochoos, I., Raković, S., and Mayne, D. Q. Robust model predictive control using tubes. *Automatica*, 40(1):125–133, 2004.

Li, W. and Todorov, E. Iterative linear quadratic regulator design for nonlinear biological movement systems. In *ICINCO (1)*, pp. 222–229, 2004.

Mania, H., Tu, S., and Recht, B. Certainty equivalent control of lqr is efficient. *arXiv preprint arXiv:1902.07826*, 2019.

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

Mayne, D. Q., Seron, M. M., and Raković, S. Robust model predictive control of constrained linear systems with bounded disturbances. *Automatica*, 41(2):219–224, 2005.

Moore, K. L. *Iterative learning control for deterministic systems*. Springer Science & Business Media, 2012.

Owens, D. H. and Hätönen, J. Iterative learning control—an optimization paradigm. *Annual reviews in control*, 29(1): 57–70, 2005.

Pontryagin, L. S., Boltyanskii, V., Gamkrelidze, R., and Mishchenko, E. The mathematical theory of optimal processes, translated by kn trirogoff. *New York*, 1962.

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

Ross, I. M. *A primer on Pontryagin's principle in optimal control*. Collegiate publishers, 2015.

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

Simchowitz, M., Singh, K., and Hazan, E. Improper learning for non-stochastic control, 2020.

Stengel, R. F. *Optimal control and estimation*. Courier Corporation, 1994.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

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

Todorov, E. and Li, W. A generalized iterative lqg method for locally-optimal feedback control of constrained nonlinear stochastic systems. In *Proceedings of the 2005, American Control Conference, 2005.*, pp. 300–306. IEEE, 2005.

Wagener, N., Cheng, C.-A., Sacks, J., and Boots, B. An online learning approach to model predictive control. *arXiv preprint arXiv:1902.08967*, 2019.

Zhou, K. and Doyle, J. C. *Essentials of robust control*, volume 104. Prentice hall Upper Saddle River, NJ, 1998.

Zhou, K., Doyle, J. C., and Glover, K. *Robust and Optimal Control*. Prentice-Hall, Inc., USA, 1996. ISBN 0134565673.

Zinkevich, M. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pp. 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 upto a small constant factor, the gradient based meta-learning results presented in (Balcan et al., 2019) by reducing their setting to the nested-OCO setting and applying Algorithm 3. The reduction requires setting the x, y space, i.e. $\mathcal{K}_1, \mathcal{K}_2$ to be $\Theta \subseteq \mathcal{R}^d$ and a ball in $\mathcal{R}^d$ of diameter D^* (according to the notation in (Balcan et al., 2019)). Further, we set the function $f_t^i(x, y) \triangleq l_{t,i}(x + y)$. The reduction recovers the same guarantee as the result in (Balcan et al., 2019) upto a factor of 2.

We note that (Balcan et al., 2019) 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 (Balcan et al., 2019). Further we acknowledge that for the particular problem considered in (Balcan et al., 2019), 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 (Balcan et al., 2019) 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 (Balcan et al., 2019). 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. (Tedrake, 2020)). 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

$$\Pi_L = \{K \in \mathbb{R}^{d_x \times d_u}\}.$$

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 π 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*

$$s_{t+1} = A_\pi s_t + B_\pi x_t, \;\; u_t = C_\pi s_t + D_\pi x_t.$$

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

$$\Pi_{LDC} = \{A \in \mathbb{R}^{d_s \times d_s}, B \in \mathbb{R}^{d_s \times d_x}, C \in \mathbb{R}^{d_u \times d_s}, D \in \mathbb{R}^{d_u \times d_x}\}.$$

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 (Agarwal et al., 2019a). 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 $\mathcal{U} \subseteq \mathbb{R}^{d_u}$ 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 $\{w^i_{1:T}\}_{i=1}^N$ with each $\|w^i_t\| \leq W$ and any $\gamma \geq 0$, produces a sequence of actions with planning regret bounded as*

$$\frac{1}{N}\left(\sum_{i=1}^{N} J_i(\pi^i_{1:T}) - \min_{u_{1:T}\in\mathcal{U}}\left(\sum_{i=1}^{N}\min_{M_{1:L}\in\mathcal{M}_\gamma} J_i\left(\pi_{M_{1:L}}(u_{1:T})\right)\right)\right) \leq \left(\frac{c_{\text{in}}\log^2(T)}{\sqrt{T}} + \frac{c_{\text{out}}}{\sqrt{N}}\right).$$

where $\mathcal{M}_\gamma = \{M | M \in \mathbb{R}^{d_u, d_x}, \|M\| \leq \gamma\}$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-logarithimic in the the arguments of $\tilde{O}$. To achieve the above bound, Algorithm 1 is to be executed with parameters, learning rate $\eta_{\text{out}} = \frac{U}{G\kappa\delta^{-2}(\kappa U+\kappa\gamma LW+W)\sqrt{N}}$, with the inner execution of Algorithm 2 is performed with parameters $\eta_{\text{in}} = \frac{\gamma^2 L^2}{\sqrt{12\gamma\kappa^4\delta^{-5}\beta G^2 L^3 W^3 (U+\gamma LW)^2}}$ 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 $u_{1:T} \in \mathcal{U}$ and $M_{1:T,1:L} \in \mathcal{M}_\gamma$, 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,

$$a_t(u_{1:T}, M_{1:T,1:L}) \triangleq u_t + \sum_{r=1}^{L} M_{t,r} w_{t-r} \tag{C.1}$$

$$x_1(u_{1:T}, M_{1:T,1:L}) \triangleq 0 \qquad x_{t+1}(u_{1:T}, M_{1:T,1:L}) \triangleq A_t x_t(u_{1:T}, M_{1:T,1:L}) + B_t a_t + w_t \tag{C.2}$$

Given a sequence of cost functions $c_{1:T}(x,u) : \mathbb{R}^{d_x \times d_u} \to \mathbb{R}$, 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

$$\forall t \in [1:T], \qquad c_t(u_{1:t}, M_{1:T,1:L}) = c_t(x_t(u_{1:t}, M_{1:T,1:L}), a_t(u_{1:T}, M_{1:T,1:L})) \tag{C.3}$$

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*

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

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

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

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

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

$$x_t(u_{1:T}, M_{1:T,1:L}) = \sum_{j=1}^{t-1} T_{j\to t}(B_j u_j + w_j) + \psi^M_{1\to t}(M_{1:T,1:L}). \tag{C.4}$$

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 $\gamma \geq 0, \mathcal{U}$, let $u_{1:T} \in \mathcal{U}, M_{1:T,1:L} \in \mathcal{M}_\gamma$ be two sequences, then we have that*

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

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. Lets 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

$$x^i_t = x_t(u^i_{1:T}, M^i_{1:T,1:L}) \qquad a^i_t = a_t(u^i_{1:T}, M^i_{1:T,1:L})$$

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

$$J = \sum_{i=1}^{N} \frac{1}{T} \left(\sum_{t=1}^{T} c^i_t(u^i_{1:T}, M^i_{1:T,1:L}) \right)$$

Fix the sequence of comparators to be $\mathring{u}_{1:T}, \{\mathring{M}^i_{1:L}\}_{i=1}^N$. The comparator cost by definition then is

$$\mathring{J} = \sum_{i=1}^{N} \frac{1}{T} \left(\sum_{t=1}^{T} c^i_t(\mathring{u}_{1:T}, \mathcal{T}_T \mathring{M}^i_{1:L}) \right),$$

where given a sequence $v_{a:b}$, we define the tiling operator $\mathcal{T}_k$, which creates a nested sequence of outer length k by tiling with copies of the sequence $v_{a:b}$, i.e. $\mathcal{T}_k v_{a:b} = [v_{a:b}, v_{a:b} \ldots v_{a:b}]$. 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,

$$\begin{aligned}
&\sum_{i=1}^{N}\sum_{t=1}^{T}\left(c_t^i(u^i_{1:T}, M^i_{1:T,1:L}) - c_t^i(\overset{*}{u}_{1:T}, \mathcal{T}_T\overset{*}{M}{}^i_{1:L})\right)\\
&\leq \sum_{i=1}^{N}\sum_{t=1}^{T}\left(\nabla_u c_t^i(u^i_{1:T}, M^i_{1:T,1:L})(u^i_{1:T} - \overset{*}{u}_{1:T}) + \nabla_M c_t^i(u^i_{1:T}, M^i_{1:T,1:L})(M^i_{1:T,1:L} - \overset{*}{M}{}^i_{1:L})\right)\\
&= \underbrace{\sum_{i=1}^{N}\sum_{t=1}^{T}\left(\nabla_u c_t^i(u^i_{1:T}, M^i_{1:T,1:L})(u^i_{1:T} - \overset{*}{u}_{1:T})\right)}_{\text{Outer Regret}} + \underbrace{\sum_{i=1}^{N}\sum_{t=1}^{T}\left(\nabla_M c_t^i(u^i_{1:T}, M^i_{1:T,1:L})(M^i_{1:T,1:L} - \overset{*}{M}{}^i_{1:L})\right)}_{\text{Inner Regret}}
\end{aligned}$$

We analyze the both the terms above separately. We begin by analyzing the first term.

Outer Regret: Consider the following calculation

$$\sum_{i=1}^{N}\sum_{t=1}^{T}\left(\nabla_u c_t^i(u^i_{1:T}, M^i_{1:T,1:L})(u^i_{1:T} - \overset{*}{u}_{1:T})\right) = \sum_{j=1}^{T}\sum_{i=1}^{N}\underbrace{\nabla_{u_j}\left(\sum_{t=1}^{T} c_t^i(u^i_{1:T}, M^i_{1:T,1:L})\right)}_{\triangleq g^u_{ij}}(u^i_j - \overset{*}{u}_j).$$

Note that by definition of the algorithm, we have that for all i, j

$$u_j^{i+1} = \mathrm{Proj}_{\mathcal{U}}(u^i_j - \eta_{\mathrm{out}} g^u_{ij}),$$

which via the pythagorean inequality implies that

$$\|u_j^{i+1} - \overset{*}{u}_j\|^2 \leq \|u^i_j - \eta_{\mathrm{out}} g^u_{ij} - \overset{*}{u}_j\|^2$$

Combining the above equations we immediately get that

$$\begin{aligned}
\sum_{i=1}^{N}\sum_{t=1}^{T}\left(\nabla_u c_t^i(u^i_{1:T}, M^i_{1:T,1:L})(u^i_{1:T} - \overset{*}{u}_{1:T})\right) &\leq \sum_{j=1}^{T}\sum_{i=1}^{N}\frac{1}{2}\left(\eta_{\mathrm{out}}\|g^u_{ij}\|^2 + \frac{(u^i_j - \overset{*}{u}_j)^2 - (u^{i+1}_j - \overset{*}{u}_j)^2}{\eta_{\mathrm{out}}}\right)\\
&\leq \sum_{j=1}^{T}\frac{1}{2}\left(\eta_{\mathrm{out}}\left(\sum_{i=1}^{N}\|g^u_{ji}\|^2\right) + \frac{(u^1_j - \overset{*}{u}_j)^2}{\eta_{\mathrm{out}}}\right)\\
&\leq 2UG\kappa\delta^{-2}(\kappa U + \kappa\gamma LW + W)T\sqrt{N} \qquad \text{(C.5)}
\end{aligned}$$

where the last inequality follows using Lemma C.3 and choice of η_{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 $M_{1:L} \in \mathbb{R}^{d_u \times d_x}$. 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 ψ defined in Definition C.2 for an alternative and succinct definition as follows.

$$\hat{x}_t(u_{1:T}, M_{1:L}) = \sum_{j=t-S}^{t-1} T_{j\to t}(B_j u_j + w_j) + \psi^M_{t-S\to t}(\mathcal{T}_T M_{1:L}). \qquad \text{(C.6)}$$

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

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

It can be observed now by the definition of Algorithm 2, the sequence $M^i_{1:T,1:L}$ played by the algorithm is chosen iteratively as follows

$$M^i_{t+1,1:L} = \mathrm{Proj}_{\mathcal{M}_\gamma}\left(M^i_{t,1:L} - \eta_{\mathrm{in}} \nabla_M \hat{c}_t(u^i_{1:T}, M^i_{t,1:L})\right). \tag{C.8}$$

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

$$\sum_{j=t-S}^{t} \nabla_{M_{j,1:L}} c_t(u_{1:T}, M_{1:T,1:L})(M_{j,1:L} - \overset{*}{M}_{1:L}) \le \nabla_{M_{t,1:L}} \hat{c}_t(u_{1:T}, M_{t,1:L})(M_{t,1:L} - \overset{*}{M}_{1:L}) + 20\eta_{\mathrm{in}} \log^2(\eta_{\mathrm{in}})\gamma\kappa^4\delta^{-3}\beta G^2 L^3 W^3 (U + \gamma LW)^2$$

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,

$$\begin{aligned}
&\sum_{t=1}^{T} \left(\nabla_M c_t(u_{1:T}, M_{1:T,1:L})(M_{1:T,1:L} - \mathcal{T}_T \overset{*}{M}_{1:L})\right) \\
&= \sum_{t=1}^{T}\sum_{j=1}^{T} \left(\nabla_{M_{j,1:L}} c_t(u_{1:T}, M_{1:T,1:L})(M_{j,1:L} - \overset{*}{M}_{1:L})\right) \\
&= \sum_{t=1}^{T}\sum_{j=1}^{t} \left(\nabla_{M_{j,1:L}} c_t(u_{1:T}, M_{1:T,1:L})(M_{j,1:L} - \overset{*}{M}_{1:L})\right) \\
&\le \sum_{t=1}^{T}\sum_{j=t-S}^{t} \left(\nabla_{M_{j,1:L}} c_t(u_{1:T}, M_{1:T,1:L})(M_{j,1:L} - \overset{*}{M}_{1:L})\right) + 2\kappa^2\gamma GLW(U + \gamma LW)\delta^{-2}(1-\delta)^S \\
&\le \sum_{t=1}^{T} \left(\underbrace{\nabla_{M_{t,1:L}} \hat{c}_t(u_{1:T}, M_{t,1:L})}_{g_t}(M_{t,1:L} - \overset{*}{M}_{1:L})\right) + 22T\eta_{\mathrm{in}} \log^2(\eta_{\mathrm{in}})\gamma\kappa^4\delta^{-3}\beta G^2 L^3 W^3 (U + \gamma LW)^2,
\end{aligned}$$

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

$$\begin{aligned}
&\sum_{t=1}^{T} \left(\underbrace{\nabla_{M_{t,1:L}} \hat{c}_t(u_{1:T}, M_{1:T,1:L})}_{g_t}(M_{t,1:L} - \overset{*}{M}_{1:L})\right) \\
&\le \sum_{t=1}^{T} \left(\frac{\eta_{\mathrm{in}}}{2}\|g_t\|^2 + \frac{\|M_{t,1:L} - \overset{*}{M}_{1:L}\|^2 - \|M_{t+1,1:L} - \overset{*}{M}_{1:L}\|^2}{2\eta_{\mathrm{in}}}\right) \\
&\le \frac{\eta_{\mathrm{in}} T}{2}\|g_t\|^2 + \frac{\|M_{1,1:L} - \overset{*}{M}_{1:L}\|^2}{2\eta_{\mathrm{in}}}
\end{aligned}$$

Combining the above equations, Equation C.9 and the choice of η_{in}, we get that the inner regret is bounded as,

$$\sum_{t=1}^{T} \left(\nabla_M c_t(u_{1:T}, M_{1:T,1:L})(M_{1:T,1:L} - \mathcal{T}_T \overset{*}{M}_{1:L}) \right) \leq \tilde{O} \left(\sqrt{T\gamma^3\kappa^4\delta^{-3}\beta G^2 L^5 W^3 (U + \gamma LW)^2} \right)$$

Combining the outer and inner regret terms we finish the proof. □

In the remaining subsections we prove Lemmas C.3 and C.4, thereby finishing the proof of Theorem 3.1.

C.2. Proof of Lemma C.3

In this section we prove Lemma 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 C.2 are bounded as follows*

$$\forall j, k \in [T], [j+1, T] \qquad \|T_{j \to k}\| \leq (1-\delta)^{k-j-1}$$

Proof of Lemma C.5. If $k = j+1$ then by definition and Assumption 2.2,

$$\|T_{j \to k}\| = \|I\| \leq 1.$$

Otherwise, again by definition and Assumption 2.2,

$$\|T_{j \to k}\| \leq \left(\Pi_{t=j+2}^{k} \|A_t\| \right) \leq (1-\delta)^{k-j-1}.$$

□

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 $\gamma \geq 0, \mathcal{U}$, let $u_{1:T} \in \mathcal{U}, M_{1:T,1:L} \in \mathcal{M}_\gamma$ 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

$$\left\| \frac{\partial x_t(u_{1:T}, M_{1:T,1:L})}{\partial M_{j,r}} \right\| \leq \begin{cases} \kappa W(1-\delta)^{t-j-1} & \text{if } j < t \\ 0 & \text{otherwise} \end{cases}$$

$$\left\| \frac{\partial a_t(u_{1:T}, M_{1:T,1:L})}{\partial M_{j,r}} \right\| \leq \begin{cases} W & \text{if } j = t \\ 0 & \text{otherwise} \end{cases}$$

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

$$\left\| \frac{\partial a_t(u_{1:T}, M_{1:T,1:L})}{\partial u_j} \right\| = \|\delta_{jt} I\| = \begin{cases} 1 & \text{if } j = t \\ 0 & \text{otherwise} \end{cases}$$

From the expansion in Equation C.4, we have that

$$\begin{aligned}
\|x_t(u_{1:T}, M_{1:T,1:L})\| &\leq \sum_{j=1}^{t-1} (\|T_{j\to t}(B_j u_j + w_j)\|) + \|\psi^M_{1\to t}(M_{1:T,1:L})\| \\
&\leq \sum_{j=1}^{t-1} (\|T_{j\to t}\| \|(B_j u_j + w_j)\|) + \sum_{j=1}^{t-1} \left(\|T_{j\to t}\| \left(\sum_{r=1}^{L} \|M_{j,r}\| \|w_{j-r}\| \right) \right) && \text{(Definition C.2 \& } \Delta\text{-inequality)} \\
&\leq (\kappa U + \kappa\gamma L W + W)) \sum_{j=1}^{t-1} (1-\delta)^{t-j-1} && \text{(Lemma C.5 and definitions)} \\
&\leq \frac{1}{\delta} (\kappa U + \kappa \gamma L W + W)
\end{aligned}$$

Also from the definition it follows that for $j \geq t$,

$$\frac{\partial x_t(u_{1:T}, M_{1:T,1:L})}{\partial u_j} = 0,$$

and if $j < t$, we have that

$$\left\| \frac{\partial x_t(u_{1:T}, M_{1:T,1:L})}{\partial u_j} \right\| \leq \|T_{j\to t} B_j\| \leq \kappa (1-\delta)^{t-j-1} \qquad \text{(Lemma C.5)}$$

From the definition in Equation C.1 it follows that for any $r \in [L]$

$$\left\| \frac{\partial a_t(u_{1:T}, M_{1:T,1:L})}{\partial M_{j,r}} \right\| = \|\delta_{jt} I \otimes w_{t-r}^\top\| \leq \begin{cases} W & \text{if } j = t \\ 0 & \text{otherwise} \end{cases}$$

From the expansion in Equation C.4, it follows that for any r and $j \geq t$,

$$\frac{\partial x_t(u_{1:T}, M_{1:T,1:L})}{\partial M_{j,r}} = 0,$$

and if $j < t$, we have that

$$\left\| \frac{\partial x_t(u_{1:T}, M_{1:T,1:L})}{\partial M_{j,r}} \right\| \leq \|T_{j\to t} B_j (I \otimes w_{j-r}^\top)\| \leq \kappa W (1-\delta)^{t-j-1} \qquad \text{(Lemma C.5)}$$

□

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,

$$\begin{aligned}
&\|\nabla_{u_j} (c_t(u_{1:T}, M_{1:T,1:L}))\| \\
&\leq G \max(\|x_t(u_{1:T}, M_{1:T,1:L}\| \|a_t(u_{1:T}, M_{1:T,1:L}\|) \left(\left\| \frac{\partial x_t(u_{1:T}, M_{1:T,1:L})}{\partial u_j} \right\| + \left\| \frac{\partial a_t(u_{1:T}, M_{1:T,1:L})}{\partial u_j} \right\| \right) \\
&\leq \begin{cases} G\kappa\delta^{-1}(\kappa U + \kappa\gamma L W + W)(1-\delta)^{t-j-1} & \text{if } j < t \\ G\kappa\delta^{-1}(\kappa U + \kappa\gamma L W + W) & j = t \\ 0 & \text{otherwise} \end{cases}
\end{aligned}$$

Therefore we have that,

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

□

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 2.1, then for any $\gamma \geq 0, \mathcal{U}$, let $u_{1:T} \in \mathcal{U}, M_{1:L} \in \mathcal{M}_\gamma$ be two sequences, then we have that for all $j, t \in [T]$,*

$$\|\hat{x}_t(u_{1:T}, M_{1:L})\| \leq \delta^{-1}(\kappa U + \kappa\gamma LW + W)$$

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

$$\begin{aligned}
&\|x_t(u_{1:T}, M_{1:L})\| \leq \sum_{j=t-S}^{t-1} (\|T_{j\to t}(B_j u_j + w_j)\|) + \|\psi^M_{t-S\to t}(\mathcal{T}_T M_{1:L})\| \\
&\leq \sum_{j=t-S}^{t-1} (\|T_{j\to t}\|\|B_j u_j + w_j\|) + \sum_{j=t-S}^{t-1} \left(\|T_{j\to t}\| \left(\sum_{r=1}^{L} \|M_r\|\|w_{j-r}\| \right) \right) && \text{(Definition C.2 \& Δ-inequality)} \\
&\leq (\kappa U + \kappa\gamma LW + W) \sum_{j=t-S}^{t-1} (1-\delta)^{t-j-1} && \text{(Lemma C.5 and Definitions)} \\
&\leq \frac{1}{\delta}(\kappa U + \kappa\gamma LW + W)
\end{aligned}$$

From the expansion in Equation C.6, it follows that

$$\left\| \frac{\partial x_t(u_{1:T}, M_{1:T,1:L})}{\partial M_r} \right\| \leq \left\| \sum_{j=t-S}^{t-1} T_{j\to t} B_j I \otimes w_{j-r}^\top \right\| \leq \delta^{-1}\kappa W \qquad \text{(Lemma C.5)}$$

□

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

$$\frac{\partial x_t(u_{1:T}, M_{1:T,1:L})}{\partial M_{j,1:L}} = 0, \frac{\partial a_t(u_{1:T}, M_{1:T,1:L})}{\partial M_{j,1:L}} = 0,$$

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 $r \in [L]$

$$\begin{aligned} \left\| \nabla_{M_{j,r}} c_t(u_{1:T}, M_{1:T,1:L}) \right\| &\leq G \|x_t(u_{1:T}, M_{1:T,1:L})\| \left\| \frac{\partial x_t(u_{1:T}, M_{1:T,1:L})}{\partial M_{j,r}} \right\| \\ &\leq G\kappa\delta^{-1} W(\kappa U + \kappa\gamma LW + W)(1-\delta)^{t-j} \end{aligned} \tag{C.9}$$

Using Lemma C.7 for the surrogate states and using Assumption 2.1, we have that for all t and for all $r \in [L]$,

$$\|\nabla_{M_r} \hat{c}_t(u_{1:T}, M_{1:L})\| \leq 2G\kappa\delta^{-2} W(\kappa U + \kappa\gamma LW + W)$$

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 η_{in}, it is immediate that for any $j, k \in [T]$ and for any $r \in [L]$,

$$\|M_{j,r} - M_{k,r}\| \leq \eta_{\text{in}} |j-k| \cdot 2G\kappa\delta^{-2} W(\kappa U + \kappa\gamma LW + W) \tag{C.10}$$

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.

$$\begin{aligned} &\|x_t(u_{1:T}, M_{1:T,1:L}) - \hat{x}_t(u_{1:T}, M_{t,1:L})\| \\ &\leq \left\| \sum_{j=1}^{t-1} T_{j\to t}(B_j u_j + w_j) + \psi^M_{1\to t}(M_{1:T,1:L}) - \sum_{j=t-S}^{t-1} T_{j\to t}(B_j u_j + w_j) - \psi^M_{t-S\to t}(\mathcal{T}_T M_{t,1:L}) \right\| \\ &= \left\| \sum_{j=1}^{t-S-1} \left(T_{j\to t} \left(B_j u_j + w_j + \sum_{r=1}^{L} M_{j,r} w_{j-r} \right) \right) + \sum_{j=t-S}^{t-1} \left(T_{j\to t} \left(\sum_{r=1}^{L} (M_{j,r} - M_{t,r}) w_{j-r} \right) \right) \right\| \\ &\leq (\kappa U + \kappa\gamma LW + W) \left(\delta^{-1}(1-\delta)^S + 2\eta_{\text{in}}\kappa\delta^{-2} S^2 GLW^2 \right) \end{aligned} \tag{C.11}$$

Furthermore, note by definitions that

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

Before moving further, consider the following calculations

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

where

$$\begin{aligned} \|v\| &\triangleq \|\nabla_x c_t(x_t(u_{1:T}, M_{1:T,1:L}), a_t(u_{1:T}, M_{1:T,1:L})) - \nabla_x c_t(\hat{x}_t(u_{1:T}, M_{t,1:L}), a_t(u_{1:T}, M_{1:T,1:L}))\| \\ &\leq \beta(\kappa U + \kappa\gamma LW + W) \left(\delta^{-1}(1-\delta)^S + 2\eta_{\text{in}}\kappa\delta^{-2} S^2 GLW^2 \right) \end{aligned} \tag{C.13}$$

using Equation C.11 and the β-smoothness of c_t via Assumption 2.1. Using Equation C.12 and Lemma C.6 we now get that

$$\begin{aligned}
&\sum_{j=t-S}^{t-1} \left(\nabla_{M_{j,1:L}} c_t(u_{1:T}, M_{1:T,1:L})\right) \\
&= \left(\sum_{j=t-S}^{t-1} \frac{\partial x_t(u_{1:T}, M_{1:T,1:L})}{\partial M_{j,1:L}}\right) \left(\nabla_x c_t(\hat{x}_t(u_{1:T}, M_{t,1:L}), a_t(u_{1:T}, M_{1:T,1:L})) + v\right) \\
&= \frac{\partial \hat{x}_t(u_{1:T}, M_{1:T,1:L})}{\partial M_{t,1:L}} \left(\nabla_x c_t(\hat{x}_t(u_{1:T}, M_{t,1:L}), a_t(u_{1:T}, M_{1:T,1:L}))\right) + v' \qquad \text{(C.14)}
\end{aligned}$$

where v' is a vector whose norm using Equation C.13 and Lemma C.7 can be bounded as follows

$$\beta\delta^{-1} LW(\kappa U + \kappa\gamma LW + W)\left(\delta^{-1}(1-\delta)^S + 2\eta_{\text{in}}\kappa\delta^{-2}S^2GLW^2\right). \qquad \text{(C.15)}$$

Now, consider the following computation which follows from Equation C.14 and using the defintiions for the $j = t$ case,

$$\sum_{j=t-S}^{t} \left(\nabla_{M_{j,1:L}} c_t(u_{1:T}, M_{1:T,1:L})\right) = \nabla_{M_{t,1:L}} \hat{c}_t(u_{1:T}, M_{t,1:L}) + v'. \qquad \text{(C.16)}$$

We can now perform the calculation to relate the gradient inner products for surrogate cost to those of real cost.

$$\begin{aligned}
&\sum_{j=t-S}^{t} \left(\nabla_{M_{j,1:L}} c_t(u_{1:T}, M_{1:T,1:L})(M_{j,1:L} - \overset{*}{M}_{1:L})\right) \\
&= \sum_{j=t-S}^{t} \left(\nabla_{M_{j,1:L}} c_t(u_{1:T}, M_{1:T,1:L})(M_{t,1:L} - \overset{*}{M}_{1:L}) + \nabla_{M_{j,1:L}} c_t(u_{1:T}, M_{1:T,1:L})(M_{j,1:L} - M_{t,1:L})\right) \\
&\leq \sum_{j=t-S}^{t} \left(\nabla_{M_{j,1:L}} c_t(u_{1:T}, M_{1:T,1:L})(M_{t,1:L} - \overset{*}{M}_{1:L})\right) + \eta_{\text{in}} 2G^2LS^2\kappa^2\delta^{-3}W^2(\kappa U + \kappa\gamma LW + W)^2 \\
&\leq \nabla_{M_{t,1:L}} \hat{c}_t(u_{1:T}, M_{t,1:L})(M_{t,1:L} - \overset{*}{M}_{1:L}) + \\
&\qquad \beta\delta^{-1}\gamma L^2 W(\kappa U + \kappa\gamma LW + W)^2\left(\delta^{-1}(1-\delta)^S + 4\eta_{\text{in}}\kappa^2\delta^{-2}S^2G^2LW^2\right) \\
&\leq \nabla_{M_{t,1:L}} \hat{c}_t(u_{1:T}, M_{t,1:L})(M_{t,1:L} - \overset{*}{M}_{1:L}) + 5\eta_{\text{in}} \log^2(\eta_{\text{in}})\gamma\kappa^2\delta^{-3}\beta G^2L^3W^3(\kappa U + \kappa\gamma LW + W)^2
\end{aligned}$$

where the first inequality follows from applying Equations C.9, C.10 and Lemma C.6, 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. □

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 Π. Note that our formal results dont cover this generalization and it is provided with practical use in mind.

Algorithm 4 iGPC Algorithm

Require: [Online] $f_{1:T}^{1:N}$: Dynamical Systems, $w_{1:T}^{1:N}$: Disturbances
Parameters: Policy class: Π, η_{out} : Learning Rate

Initialize $\pi_{1:T}^1 \in \Pi$.
for $i = 1 \ldots N$ **do**
  Receive the dynamical system $f_{1:T}^i$ for the next rollout.
  Rollout: Collect trajectory data by rolling out policy $\pi_{1:T}^i$ with GPC ▷ (Algorithm 2)

$$\text{TrajData}^i = \{x_{1:T}^i, a_{1:T}^i, w_{1:T}^i, o_{1:T}^i\} \leftarrow \text{GPCRollout}(f_{1:T}^i, \pi_{1:T}^i)$$

  Update: Compute update to the policy

$$\pi_{1:T}^{i+1} = \text{Proj}_{\Pi}\left(\pi_{1:T}^i - \eta_{\text{out}} \nabla_{\pi_{1:T}} J(\pi_{1:T}^i + \pi(o_{1:T}^i) | f_{1:T}^i, w_{1:T}^i)\right)$$

end for

Algorithm 5 GPCRollout

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

Initialize $M_{1,1:L} = \{M_{1,j}\}_{j=1}^L \in \mathcal{M}_\gamma$.
Set $w_i = 0$ for any $i \leq 0$.
for $t = 1 \ldots T$ **do**
  Compute GPC Offset

$$o_t = M_{t,1:L} \cdot w_{t-1:t-L}.$$

  Play action

$$a_t = \pi_t(\cdot) + o_t$$

  Observe state x_{t+1}.
  Compute perturbation

$$w_t = x_{t+1} - f_t(x_t, a_t).$$

  Update $M_{t+1,1:T}$ for the next round as:

$$M_{t+1,1:L} = \text{Proj}_{\mathcal{M}_\kappa}\left(M_{t,1:L} - \eta_{\text{inner}} \nabla_{M_{1:L}} \text{GPCLoss}(M_{t,1:L}, \pi_{t-S+1:t}, w_{t-S-L+1:t-1})\right)$$

▷ GPCLoss defined in Equation 4.1

end for
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 $\mathring{x}_{1:T} \in \mathbb{R}^{d_x}, \mathring{u}_{1:T} \in \mathbb{R}^{d_u}$, open-loop gain seqeunce $k_{1:T}$ and closed-loop gain sequence $K_{1:T}$ and a parameter α. 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 $f_{1:T}$.

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

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

$$Q(c, x_0, u_0)(x, u) \triangleq \nabla c_x(x_0, u_0)(x - x_0) + \nabla c_u(x_0, u_0)(u - u_0) + \frac{1}{2}([x, u] - [x_0, u_0])^\top \nabla^2 c(x, u)([x, u] - [x_0, u_0]) \quad \text{(E.1)}$$

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

Algorithm 6 Iterative Planning Algorithm

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

1: Initialize starting sequence of actions $u^0_{1:T}$
2: Initialize sequence of open loop $k^0_{1:T} = 0$ and closed loop gains $K^0_{1:T} = 0$.
3: **for** $i = 1 \dots N$ **do**
4: **Rollout the Policy**:

- **ILQG:** Standard Rollout on $f_{1:T}$.

$$x^i_{1:T}, u^i_{1:T} = \text{Rollout}(f_{1:T}, \pi(\alpha, x^{i-1}_{1:T}, u^{i-1}_{1:T}, k^{i-1}_{1:T}, K^{i-1}_{1:T}))$$

- **ILC:** Standard Rollout on $g_{1:T}$.

$$x^i_{1:T}, u^i_{1:T} = \text{Rollout}(g_{1:T}, \pi(\alpha, x^{i-1}_{1:T}, u^{i-1}_{1:T}, k^{i-1}_{1:T}, K^{i-1}_{1:T}))$$

.

- **IGPC:** GPCRollout on $g_{1:T}$,

$$x^i_{1:T}, u^i_{1:T} = \text{GPCRollout}(g_{1:T}, \pi(\alpha, x^{i-1}_{1:T}, u^{i-1}_{1:T}, k^{i-1}_{1:T}, K^{i-1}_{1:T}))$$

5: **Update**: Obtain $k^i_{1:T} \in \mathbb{R}^{d_u}, K^i_{1:T} \in \mathbb{R}^{d_u \times d_x}$ as the optimal non-stationary affine policy to the following LQG problem.

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

$$\text{subject to} \qquad x_{t+1} - x^i_{t+1} = \frac{\partial f_t(x^i_t, u^i_t)}{\partial x^i_t}(x_t - x^i_t) + \frac{\partial f_t(x^i_t, u^i_t)}{\partial u^i_t}(u_t - u^i_t) + z_t$$

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 α 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 α^+. We optimize over choices for α^+ for ILC and report the best performance obtained as baseline. For IGPC, we use the same α^+ as obtained for ILC and the same line search for strategy for selecting α. 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 η_{in}, the inner learning rate. We chose $L, S = 3$ arbitrarily for our experiments and tuned η_{in} per experiment. Overall we observed that for every experiment, the selection of η was robust in terms of performance.