A Survey of Convex Optimization for Control

Ademi Adeniji, Yuqing Du, Paula Gradu, Olivia Watkins, Brian Yu

December 5, 2021

Contents

1 Introduction

We give a survey of the role of convex relaxations and convex optimization for online control. We begin by looking at the canonical H_2 and H_∞ control problems which have analytical solutions under certain assumptions of the system dynamics. However, in the real world these disturbance assumptions are unlikely to hold, and thus and we seek an optimization approach that can actively adapt to the disturbance experienced online. We analyze convex relaxations of the resulting linear policy solutions to enable us to apply online convex optimization for such an adaptive policy. We then cover in detail recent work [1], which proposes a Disturbance Action Controller policy class, and analyze gradient descent over this convex policy class in order to obtain regret bounds on the proposed algorithm. To compare this method and canonical approaches, we carry out experiments in a simple linear dynamical system, under different noise distributions. We then cover extensions to this line of work and look at more challenging cases cases with partial observability, unknown systems which require system identification, better rates, bandit feedback, and time-varying systems. Finally, we summarize the main takeaways from these convex optimization results as a pipeline for the online control problem, and give an example of a new algorithm that can arise as a result.

2 An overview of classical policy parametrizations

As an introduction to the problem setting, in Section 2.1 we first look at the following three classical parametrizations: linear, H_2, and H_∞ controllers, and derive their analytical solutions. We then show that the resulting linear policy classes are non-convex, preventing their direct usage in an online setting. To address this, in Section 2.2 we present the System Level Synthesis (SLS) convex relaxation.

2.1 Linear, H_2, and H_∞ Controllers: The Classics

Optimal Control. The LQR (linear quadratic regulator) and LQG (linear quadratic Gaussian) controllers offer both a model and an analytical solution to control problems under certain strong constraints [12]. Primarily, the LQR/LQG formulations assume known linear dynamics and quadratic cost. The LQR formulation assumes determinisitc dynamics, while LQG extends LQR to stochastic dynamics in an analogous manner. Formally, we express our control problem over some horizon T as:

$$\min_{u_1 \dots u_T} \sum_{t=1}^{T} c(x_t, u_t) \text{ s.t. } x_t = f_t(x_{t-1}, u_{t-1})$$

where x_t is the state of the system at time t, u_t is the control input at time t, c is the cost function, and f is the dynamics function. In a linear dynamical system we have:

$$f(x_t, u_t) = \begin{bmatrix} A_t & B_t \end{bmatrix} \begin{bmatrix} x_t \\ u_t \end{bmatrix}$$

$$c(x_t, u_t) = \begin{bmatrix} x_t \\ u_t \end{bmatrix}^\top \begin{bmatrix} Q_t & R_t \end{bmatrix} \begin{bmatrix} x_t \\ u_t \end{bmatrix}$$

where $Q_t, R_t \succeq 0$. In order to obtain a closed-loop control law that is linear in the state, we leverage a backward recursion (concisely expressing $C_t = [Q_t; R_t]$ and $F_t = [A_t; B_t]$):

for $t = T$ to 1:

$$\begin{aligned}
Q_t &= C_t + F_t^T V_{t+1} F_t \\
q_t &= c_t + F_t^T v_{t+1} \\
Q(x_t, u_t) &= \frac{1}{2} \begin{bmatrix} x_t \\ u_t \end{bmatrix}^\top Q_t \begin{bmatrix} x_t \\ u_t \end{bmatrix} + \begin{bmatrix} x_t \\ u_t \end{bmatrix}^\top q_t \\
u_t &\leftarrow \arg\min_{u_t} Q(x_t, u_t) = K_t x_t \\
K_t &= -Q_{u_t,u_t}^{-1} Q_{u_t,x_t} \\
k_t &= -Q_{u_t,u_t}^{-1} q_{u_t} \\
V_t &= Q_{x_t,x_t} + Q_{x_t,u_t} K_t + K_t^T Q_{u_t,x_t} + K_t^T Q_{u_t,u_t} K_t \\
v_t &= q_{x_t} + Q_{x_t,u_t} k_t + K_t^T Q_{u_t} + K_t^T Q_{u_t,u_t} k_t \\
V(x_t) &= \frac{1}{2} x_t^T V_t x_t + x_t^T v_t
\end{aligned}$$

where K_t is a recursive expression in terms of the future costs and dynamics matrices and Q is a value function representing the cumulative future cost in terms of arbitrary control inputs and their resulting states according to known dynamics. Since our problem provides the initial state x_1, the control inputs $u_1, ..., u_T$ can be computed with forward recursion according to this simple linear model: which results in the following forward recursion for $t = 1$ to T :

$$\begin{aligned}
u_t &= K_t x_t \\
x_{t+1} &= f(x_t, u_t)
\end{aligned}$$

The LQG formulation assumes instead assumes Gaussian distributed dynamics where dynamics function f produces the mean of the resulting state with some unknown covariance. However, the optimal solution is to choose control inputs according to the same forward recursion.

Robust Control. A natural question which arises is: what happens when there is non-benign noise? This question has given rise to the study of robust control. Luckily, for full observation of the state, the previous analysis yields a solution to the following robust control problem:

$$\arg\min_{u_1 \ldots u_T} \max_{w_1 \ldots w_T} \left(\sum_{t=1}^{T} c(x_t, u_t) - \gamma \|w_t\| \right) \text{ s.t. } x_t = f_t(x_{t-1}, u_{t-1}, w_{t-1})$$

where

$$\begin{aligned}
f(x_t, u_t, w_t) &= \begin{bmatrix} A_t & B_t & I \end{bmatrix} \begin{bmatrix} x_t \\ u_t \\ w_t \end{bmatrix} \\
c(x_t, u_t) &= \begin{bmatrix} x_t \\ u_t \end{bmatrix}^\top \begin{bmatrix} Q & R \end{bmatrix} \begin{bmatrix} x_t \\ u_t \end{bmatrix}
\end{aligned}$$

Note that we can reformulate the adversary's optimization to look like what we saw above by denoting $(x_t, u_t) = v_t$, $\tilde{A}_t^w = [A_t, B_t]$, $\tilde{Q} = [Q, R]$. For any $u_1 ... u_T$, the adversary must solve

$$\min_{w_1 \dots w_T} \sum_{t=1}^{T} c_w(v_t, w_t) \text{ s.t. } x_t = f_t^w(v_{t-1}, w_{t-1})$$

$$f_t^w(v_t, w_t) = \begin{bmatrix} \tilde{A}_t & I \end{bmatrix} \begin{bmatrix} v_t \\ w_t \end{bmatrix}$$
$$c_w(v_t, w_t) = \begin{bmatrix} v_t \\ w_t \end{bmatrix}^\top \begin{bmatrix} -\tilde{Q} & \gamma I \end{bmatrix} \begin{bmatrix} v_t \\ w_t \end{bmatrix}$$

So we can use the Bellman Recursion from before to obtain that the optimal noise action sequence is a linear policy of the expanded state $w_t = K_t^w v_t$. Based on this, the robust controller can also reduce its task to the original LQR formulation. Split the blocks of $K_t^w = [K_t^{w,x} K_t^{w,u}]$. Then the controller needs to solve the LQR problem for the proxy transition matrices $\tilde{A}_t^u = [A_t K_t^{w,x}]$, $\tilde{B}_t = [B_t K_t^{w,u}]$, i.e.

$$\min_{u_1 \dots u_T} \sum_{t=1}^{T} c(x_t, u_t) \text{ s.t. } x_t = f_t^u(x_{t-1}, u_{t-1})$$

$$f_t^u(x_t, u_t) = \begin{bmatrix} \tilde{A}_t^u & \tilde{B}_t \end{bmatrix} \begin{bmatrix} x_t \\ u_t \end{bmatrix}$$
$$c(x_t, u_t) = \begin{bmatrix} x_t \\ u_t \end{bmatrix}^\top \begin{bmatrix} Q & R \end{bmatrix} \begin{bmatrix} x_t \\ u_t \end{bmatrix}$$

so again the optimal control is $u_t = K_t^u x_t$. Importantly note that K_t^u is not equal to K_t from the LQR solution! Also it depends on γ. A different gamma yields different solutions (or some none at all). In practice solvers binary search over gamma to find the smallest such that the induced system $\tilde{A}_t^u, \tilde{B}_t$ is stabilizable.

Non-convexity of Stabilizing Linear Policies. We see that optimal and robust controllers are all stabilizing linear. But the world is usually neither perfectly benign nor completely adverserial. Can we hope to do something akin to gradient descent on stabilizing linear policies and adapt online to what the action sequence is? A first barrier is that the set itself is non-convex, so a gradient step can take you out of it (and projection onto non-convex sets is ill specified). To see why it's not convex, look at the simple LTI system given by $A = B = I$. Observe that $K_1 = \begin{bmatrix} 1 & 0 & -10 \\ -1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ and $K_2 = \begin{bmatrix} 1 & -10 & 0 \\ 0 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix}$ both stabilize the system, but their interpolation $K = \frac{K_1+K_2}{2}$ does not.

2.2 SLS Convex Relaxation of LTI Linear Policies

To address the non-convexity issue raised above, we look at a relaxation by Wang et al. in both the frequency and time domains [18].

The state progression equation for an LTI system is

$$x_{t+1} = Ax_t + Bu_t + w_t$$

where x_t is the state vector, u_t is the control vector, and w_t is the disturbance vector.

We first take the frequency domain perspective and apply the z-transform to convert this discrete-time time-domain equation into the frequency domain. Then, we plug in our linear control law, $u = Kx$.

$$\begin{aligned} (zI - A)x &= Bu + w && \text{(z-transform)} \\ (zI - A - BK)x &= w && (u = Kx) \end{aligned}$$

Thus, the optimal control can be written directly as a function of the disturbance w.

$$\begin{aligned} x &= (zI - A - BK)^{-1}w \\ \implies u &= K(zI - A - BK)^{-1}w \end{aligned}$$

Now that we have a formal description of our control input, we reparameterize using the following transfer matrices. Crucially, this subspace parameterizes all system responses achievable by stabilizing K.

$$\begin{aligned} &R = (zI - A - BK)^{-1} \\ &M = KR = K(zI - A - BK)^{-1} \\ &\begin{bmatrix} zI - A & -B \end{bmatrix} \begin{bmatrix} R \\ M \end{bmatrix} = I \\ &zR, zM \text{ are real-rational proper transfer matrices} \end{aligned}$$

As a consequence, $u = Mw$, which is exactly the robust control strategy of affine recourse. R, M are constrained to be norm-bounded and in an affine subspace (while the set of K is nonconvex), a convex relaxation.

For the time-domain perspective, we plug in our control law $u_t = Kx_t$ and recursively unroll the state progression equation. For a stabilizing controller K, there is some t' such that any effect beyond t' is negligible, so we can apply a finite horizon on our unrolling.

$$\begin{aligned} x_{t+1} &= Ax_t + BKx_t + w_t && (u_t = Kx_t) \\ x_t &= \sum_{i=0}^{t-1} (A + BK)^{i-1} w_{t-i} && \text{(unrolling)} \\ \implies u_t &= K \sum_{i=0}^{t-1} (A + BK)^{i-1} w_{t-i} \\ u_t &= K \sum_{i=0}^{\min(t', t-1)} (A + BK)^{i-1} w_{t-i} && \text{(finite horizon)} \end{aligned}$$

We reparameterize our K using M, and relax M to be in a convex set. This is exactly the same reparameterization result as in the frequency-domain perspective.

$$u_t = \sum_{i=0}^{\min(t',t-1)} M_i w_{t-i}$$
$$M_i = K(A+BK)^{i-1}$$

In summary, we can achieve all stable system responses using a linear control policy that is a weighted function of the past observed disturbances.

3 Online Convex Optimization for Control

In this section we present more recent work [1] which has looked into applying online learning techniques to a generalized form of the control problem. To do so, they introduce policy classes that are effectually equivalent to the SLS one discussed above. In the sections that follow we discuss the main algorithms and results proposed in this line of work. The main comparator of interest are state feedback policies. We will optimize with respect to convex relaxations of this class: DAC/SLS. We express all three in terms of a length-m policy parameter $M = [M^{[i]}]_{i<m}$ in a bounded ball $\mathcal{M}(m, R_M)$:

$$\mathcal{M}(m, R_M) = \{(M^{[0]}, \ldots, M^{[m-1]}) : \sum_{i=0}^{m-1} \|M^{[i]}\|_{op} \le R_M\}$$

Definition 3.1 (Feedback policy class)**.** A feedback control policy π^M_{feed} of length m is given by $u^M_t = \sum_{i=0}^{m-1} M^{[i]} x_{t-i}$ where $M = [M^{[i]}]_{i<m}$ is the parameter of the policy. Define the bounded feedback policy class as $(m, R_M) = \{\pi^M_{\text{feed}} : M \in \mathcal{M}(m, R_M)\}$. In the special case of memory $m = 1$, denote the *state feedback* policy class as $= (m = 1)$.

3.1 Online convex optimization over DAC/SLS polcies

[1] is the first work to propose the setup of regret minimization for control for tackling two main challenges: 1) handling arbitrary disturbances in the dynamics in an adaptive manner, and 2) handling arbitrary convex costs (whereas prior work has mostly focused on quadratic costs).

The work is motivated by the fact that linear controllers can be arbitrarily well approximated by what is termed as Disturbance-Action Controller (DAC) policies. Because DAC policies are convex, the problem of finding the best controller becomes convex and is hence amenable to convex optimization methods. We first provide the motivation, formalize the problem setting, then give the main algorithm and the result, with a comparison the proposed class to the SLS class introduced earlier.

3.1.1 Problem Setup and Motivation

The setting under consideration is the standard Linear Dynamical System. At time t, $x_t \in \mathbb{R}^d$ is the system state and $u_t \in \mathbb{R}^k$ is the control action. We have the following system dynamics, which are time-invariant

$$x_{t+1} = Ax_t + Bu_t + w_t \tag{3.1}$$

where w_t is the disturbance to the system at time t. Typically, w_t is assumed to be i.i.d. noise vectors from a zero-mean Gaussian. In this work we relax this, and make no assumptions on the distribution of w_t, and the controller is not aware of w_t in advance. This allows the proposed method to handle adversarial disturbances without taking a worst-case approach, which may be overly pessimistic. The system also incurs a cost $c_t(x_t, u_t)$ at each timestep. Typically c_t is assumed to be quadratic, but this work only makes the assumption that c_t is convex.

Given c_t at each step, and an algorithm $\mathcal{A}$ that maps previous states and costs to an action u_t, we can define the cost of $\mathcal{A}$ across T timesteps as

$$J_T(\mathcal{A}) = \mathbb{E}\left[\sum_{t=1}^{T} c_t(x_t, u_t)\right] \tag{3.2}$$

where the expectation is over the randomness in the state transition dynamics and/or the policy.

3.1.2 Online Convex Optimization

We consider the online setting, where the cost c_t and disturbance w_t are chosen by the environment and are unknown to the learner ahead of time. At each step t, the learner observes the state x_t, chooses an action u_t based on the current policy, and incurs the cost c_t, at which point the system transitions to the next state. In using an online approach rather than assuming worst case noise beforehand and solving for the best linear controller, the method is able to adjust to the noise encountered at each timestep iteratively. As a result, we can avoid having an overly conservative controller if we only assume worst case noise, or a controller that is not robust enough if we assume i.i.d. Gaussian noise, for example.

3.1.3 Regret Formulation

It is well known that in the infinite-horizon setting for a controllable system, with zero-mean Gaussian noise and quadratic cost, that the optimal policy is given by a linear feedback controller $u_t = Kx_t$. Thus in this work we use a linear controller as a comparator for computing the regret. We can then define the regret as

$$J_T(\mathcal{A}) - \min_{K \in \mathcal{K}} J_T(K) \tag{3.3}$$

where $\mathcal{K}$ is a certain linear controller class. The goal of the learner is to minimize the regret. In this work the authors consider the class $\mathcal{K}$ of (κ, γ) strongly stable policies.

Definition 3.2. A policy K is (κ, γ)-strongly stable if $\|K\| \leq \kappa$ and there exists matrices L, H such that $A + BK = HLH^{-1}$ with $\|L\| \leq 1 - \gamma$ and $\|H\|\|H^{-1}\| \leq \kappa$ [4].

3.1.4 Disturbance-Action Policy Class

This work proposes the Disturbance-Action Controller policy (DAC). The DAC policy parametrization defines the policy as a linear function of the disturbances encountered in the past, which is the same policy setup as in SLS [18]. However, SLS does not consider the online case, and thus is unable to actively adapt to disturbances and costs iteratively, and must therefore solve for the worst case response beforehand.

Definition 3.3 (DAC policy class)**.** A DAC control policy $\pi_{\text{dac}}(M, K)$ of length H is given by

$$u_t = -Kx_t + \sum_{i=1}^{H} M^{[i]} w_{t-i} \tag{3.4}$$

where $M = [M^{[i]}]_{i \leq H}$ are parameters of the policy. We define the policy at time t as $M_t = \{M_t^{[i]}\}$, with i being the response of M_t on the past disturbance $w_{t-i} = x_{t-i+1} - Ax_{t-i} - Bu_{t-i}$. We assume K is fixed, and acts as an initial linear controller.

The DAC policy class allows for efficient optimization over the policy parameters. Firstly, the truncated horizon allows us to consider only a fixed number of policy parameters rather than having a potentially intractable number of M for an infinite horizon setting. Secondly, the DAC class has the nice property that the policy is linear in all the past disturbances. While a linear state controller is also linear in past disturbances *through* being linear in the current state, the resultant state sequence from carrying out a linear policy is not a linear function of the policy parameters since it is not directly linear in the past disturbances. On the other hand, a DAC policy is linear in all the past disturbances, and this ensures that the resultant state sequence is also a linear function of the policy parameters.

3.1.5 Convex Optimization with Memory

To minimize the cost, we use Online Gradient Descent [3]. For memory efficiency, we only consider a truncated horizon of cost, only looking at the past H steps. The DAC policy parametrization with a memory horizon H means that the method falls under the regime of Online Convex optimization with a truncated memory. The authors show that memory-based online gradient descent allows them to bound the algorithm regret to be in $O(\sqrt{T})$, where T is the length of the full horizon.

3.1.6 Algorithm

Algorithm 1 summarizes the algorithm.

Inputs: Algorithm hyperparameters include step size α, policy time horizon H, and cost horizon N. See [1] for recommended choices for these parameters which allow us to define regret bounds as shown in section 3.2.

Initial Computations During this initial phase, we also compute a (κ, γ)-stable linear control matrix K. As discussed above, we cannot directly compute the optimal linear controller since cost is not convex in K, so instead we can optimize with an SDP relaxation as proposed in [5].

Action Selection At each time step, we select an action according to our DAC policy, as defined in Equation 4.1. We execute this action and observe the next state x_{t+1}. From this, we can compute the disturbance at the past time step: $w_t = x_{t+1} - Ax_t - Bu_t$.

Proxy Costs Computing the SGD update using the cost at all prior timesteps is memory- and time-inefficient, so instead we use this an approximate proxy cost $\tilde{c}_t(M_t)$ which only considers the prior N timesteps. Since our policy aims to keep the state near 0, we approximate $x_{t-N} = 0$. Using the known environment dynamics and observed disturbances, we compute the cost the current policy would have received for the prior N timesteps.

Gradient Update We update our policy parameters M with gradient descent on the proxy cost. Finally, we project the parameters onto a bounded norm ball.

Algorithm 1 Online Control Algorithm

```
**Environment Parameters:** environment time horizon T
**Input Hyperarameters:** step size $\alpha$, policy horizon $H$, cost horizon $N$, stability params $\kappa$, $\gamma$
Using an SDP relaxation as done in [5], compute a ($\kappa$, $\gamma$) strongly stable linear controller $K$.
Choose a DAC policy class $\mathcal{M}$ defined according to Eqn 4.1 and initialize $M_0 \in \mathcal{M}$ arbitrarily.
**for** t=0, . . . , T-1 **do**
    Choose an action according to the current policy: $u_t = -Kx_t + \sum_{i=1}^{H} M^{[i]} w_{t-i}$
    Take the action, incur cost $c_t(x_t, u_t)$, observe $c_t$ and new state $x_{t+1}$
    Compute disturbance $w_t = x_{t+1} - Ax_t - Bu_t$.
    Compute proxy cost $\tilde{c}_t(M_t)$ over the prior $N$ timesteps.
    Compute the gradient of the cost wrt each policy parameter in $M$: $\nabla\tilde{c}_t(M_t)$
    Do one SGD update: $M_{t+1} = (M_t - \alpha\nabla\tilde{c}_t(M))$ and clip parameter norms, if necessary.
**end for**
```

3.2 Bounds

To prove regret bounds on this algorithm, we require the assumptions that the cost function $c_t(x_t, u_t)$ is convex, transition matrices A and B have bounded norm, and disturbance $||w_t|| \leq W$ for some finite W. have bounded norms and that the cost is convex. If hyperparameters are chosen as described in [1], regret of the algorithm $\mathcal{A}$ is be bounded by:

$$J_T(\mathcal{A}) - \min_{K \in \mathcal{K}} J_T(K) \leq \mathcal{O}(GW^2\sqrt{T}\log(T)) \tag{3.5}$$

where T is the environment time horizon and G is a bound on the Euclidean norm bound of the proxy cost which scales with $\log(T)$.

We will provide an intuitive sketch of this result. For a formal proof, see [1].

1. Establish bounds on $\sum_t c(K) - \sum_t \tilde{c}(M)$ - i.e. the difference between the true cost of a linear policy and the proxy cost which only considers the past N timesteps. Intuitively, since the policy uses a stabilizing controller K, this controller keeps the state close to 0, limiting the cost which was occurred during the timesteps truncated by the proxy cost.

2. Establish bounds on $\sum_t \tilde{c}(M^*) - \sum_t c(K^*)$ - i.e. the difference between the hindsight-optimal proxy cost and cost of a hindsight-optimal linear controller. This is established by showing that the DAC policy class is expressive enough to capture any fixed linear policy.

3. Establish bounds on $\sum_t \tilde{c}(M) - \sum_t \tilde{c}(M^*)$ - i.e. the difference between the proxy cost obtained through gradient descent (with an appropriate step size) and the optimal proxy cost. We are able to obtain this bound by using the fact that the cost function is convex and has bounded gradient.

4. Sum bounds 1-3 to obtain Regret $= \sum_t c(K) - \sum_t \tilde{c}(K^*)$.

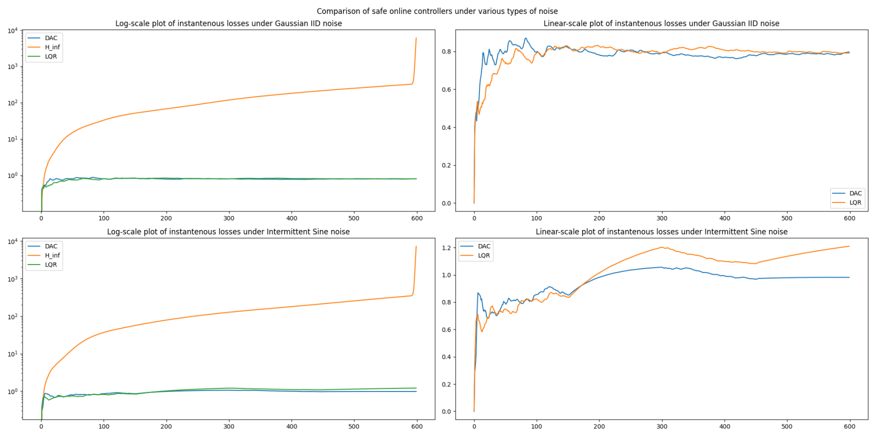

Figure 1: Recreated Experiments

3.3 Experiments

We investigate the applicability of the algorithms discussed experimentally. The H_∞ policy converges to a much higher loss than either the DAC or LQR policies under both i.i.d. Gaussian and intermittent sine noise. This is because H_∞ is only optimal under the noise being very specifically designed. Further, under Gaussian i.i.d. noise, the DAC and LQR policies converge to the same solution. Under intermittent sine noise, the LQR solution oscillates, while the DAC policy converges in $\sim$400 steps.

4 Extensions

In this section we look at follow-up work in the non-stochastic online control framework inrtoduced in the previous section: partial observability (4.1), unknown dynamics (4.2), bandit feedback (4.3), better rates (4.4), time-varying dynamics (4.5).

4.1 Partial Observability

An essential issue in control is that one does not have access to the full state of the system, but rather just a proxy (most likely noisy) observation. For example, you may observe the position of a cart you are controlling but not directly its velocity. Formally, you only have access to:

$$y_t = Cx_t + e_t$$

This poses a clear challenge to the DAC policies discussed previously which rely on extracting the perturbations w_t. Even if one knows the system, C not being full rank (which is almost always the case) makes it impossible to retrieve w_t. However [17] extends the previous work to partially observed systems by introducing a new controller parametrization that relies on an "accumulated noise" quantity. Formally, let the "nature y's" be the component of the observation that cannot be explained by the input controls:

$$y_t^{\text{nat}} = y_t - \sum_{j=1}^{t-1} CA^{j-1}Bu_{t-j}$$

The proposed new policy class, disturbance response controllers, simply act in terms of this quantity rather than w_t. By going through derivations akin to those in Section 2.2, one formalize the connection to the classic Youla parametrization [19] from the control literature.

Definition 4.1 (DRC policy class)**.** A DRC control policy $\pi_{\text{drc}}(M, K)$ of length H is given by

$$u_t = \sum_{i=1}^{H} M^{[i]} y_{t-i}^{\text{nat}} \tag{4.1}$$

where $M = [M^{[i]}]_{i \leq H}$ are parameters of the policy.

Note that in this setting we assume the system is either inherently stable or that we are given some exogenous stabilizing controls and act according to $u_t + u_t^{\text{ex}}$. Beyond this, the main algorithm and analysis are identical: apply online gradient descent on the parametrization and analyze the regret rate by reducing the problem to one of Online Convex Optimization with Memory.

4.2 Unknown System

All the work presented so far assumes perfect model knowledge. A natural question is "what can be achieved when starting from no system knowledge?". Another simpler relevant question is how sensitive the above algorithms are to model misspecification. [10] addresses the big question of dealing with an unknown initial system (in the non-stochastic setup), but as a subproblem they actually solve the second problem as well. The algorithm follows the classic "explore-then-commit" strategy. It first runs a system identification phase in which random controls are injected to obtain ϵ-approximations $\hat{A}$ and $\hat{B}$ via Least Squares (Algorithm 2), and then Applies Algorithm 1 (or the partially observed variant from 4.1) using these estimates. The analysis then can be entirely decomposed into: 1) analyze the approximation error of the sys-id phase, 2) analyze the error sensitivity of the existing algorithm.

Algorithm 2 System identification via random inputs

```
1: Input: $T_0, \mathbb{K}$
2: for $t = 0, \ldots, T_0$ do
3:     sample $\xi_t \in_{\text{R}} \{\pm 1\}^m$
4:     choose action $u_t = -\mathbb{K}x_t + \xi_t$
5:     Incur loss $c_t(x_t, u_t)$, record $x_t$
6: end for
7: Set $N_j = \frac{1}{T_0 - k} \sum_{\tau=0}^{T_0-H-1} x_{\tau+j+1}\xi_\tau^T$ for all $j$ in $[H]$
8: Let $C_0 = (N_0, \ldots N_{H-1})$, $C_1 = (N_1, \ldots N_H)$
9: Return $\hat{A} = C_1 C_0^T (C_0 C_0^T)^{-1} + N_0 K$ and $\hat{B} = N_0$
```

This procedure attains an $\tilde{O}(T^{2/3})$ regret for when exploring for the initial $O(T^{2/3})$ timesteps, which is optimized to balance the exploration cost and the error accumulation. [17] applies this general idea to the partial observed feedback as well, but estimating $[CB, CAB, \ldots, CA^H B]$ instead.

4.3 Bandit Feedback

In many applications of interest, it is unreasonable to assume access to the true underlying cost function. Instead, you may only have access to a scalar observation of the cost incurred $c_t(x_t, u_t)$. This is known in the literature as bandit feedback. [7] attains $\tilde{\mathcal{O}}(T^{3/4})$ regret for bandit feedback in the non-stochastic control model. The algorithm relies on a generalization of the FKM procedure [6] to the online learning with memory model. By integrating the analysis in 4.2, the same rate holds when starting with an unknown system. As per the discussion in 4.1, the same procedure can be applied to the DRC parametrization to extend the same results to the partially observed model. It is an interesting open question to attain a better rate, potentially using techniques in [11].

4.4 Better Rates

Beyond extensions to new settings, a few works have looked at improving upon existing bounds in settings already considered. For known systems, [2] first attain a logarithmic rate under gaussian noise and strongly convex costs. A key challenge is that despite the strong convexity of the original true cost function, the proxy policy cost used for gradient descent may not be strongly convex under adverserial noise. This first paper shows that strong convexity can be maintained under gaussian noise however and hence logarithmic rates from classic online convex optimization can be transferred to the setup. [17] extend this result to partially observed systems, and to a more general nosie setup, called 'semi-adverserial noise', which allows for only limited adverserial noise inpusts. In some sense [15] finalizes this line of inquiry by considering instead a modified version of Online Newton Step (Chapter 5 in [9]) and attaining poly-logarithmic regret for known systems, and $\tilde{O}(\sqrt{T})$ for unknown systems. In some sense, this shows that non-stochastic control is almost (up to logarithmic factors) as easy as stochastic control, since in the same year [16] prove a $\sqrt{T}$ lowerbound on the LQR problem with unknown initial system and i.i.d. gaussian noise.

4.5 Time-Varying LDS

[8] looks into extending existing results to the setting of time-varying linear dynamical systems, i.e. $x_{t+1} = A_t x_t + B_t u_t + w_t$, in the known system setting. The biggest challenge in this setting is that regret bounds yield convergence guarantees with respect to a static controller over the entire trajectory, which is unsuitable for the time-varying scenario. Hence, they motivate looking at minimizing the metric of adaptive regret:

$$R_I(\mathcal{A}) \doteq J_I(\mathcal{A}) - \min_{\pi \in \Pi} J_I(\pi) \tag{4.2}$$

For either switching or slowly varying systems, a local fixed policy can perform much better, or even optimall (for switching). As such, this is a much more suitable metric to study. They attain a $\sqrt{OPT}$ rate for this general setting, against either the DAC or DRC policy class. [14] builds on this work and extends it to the unknown case, as well as providing new results for the general LTV setting. Most importantly, they show that the equivalence of linear, DAC and DRC classes breaks down in LTV systems. They then propose an algorithm which attains $\tilde{\mathcal{O}}(T^{2/3} + |I| \cdot \text{sys-var}_I)$ adaptive regret algorithm against the DAC and DRC classes, where sys-var_I captures the variability of the system over interval I. For switching systems, $\text{sys-var}_I) = 1$ for every fixed subinterval, reducing to the best existing rate for unknown, general convex costs (due to [10, 17]). Finally, they show a matching lower bound $\Omega\left(T \cdot \text{sys-var}_T\right)$, showing that, unlike for known systems, linearity in system variabbility is unavoidable. A big open question is to tie this to attaining results for nonlinear dynamics via iterative linearization.

5 A Pipeline for New Algorithms with Provable Guarantees

In light of the discussion in Section 4 and Subsection 3.2, a quite clear pattern emerges: results from the online learning community have a clear protocol for conversion into results for online non-stochastic control:

1. Identify the theoretical goal within the framework (e.g. better rate, bandit feedback, adaptive regret, safety constraints, multiple agents, etc.),
2. Scour the online learning literature for such a result/ prove such a result in the online learning model,
3. Extend the result to the online learning with memory framework,
4. Ensure the proxy cost remains a good approximator under proposed algorithmic novelties.
5. Wrap everything up!

We highlight that step (3) can be deceivingly challenging and is actually the main technical contribution of some of the papers discussed above. In fact, after some substantial effort on (3), one may actually come to the conclusion that the result may not hold in this setting and instead be motivated to show a lower bound. In fact, many of the papers in the previous section contain lower bounds as well [15, 8, 14], which we believe arised as a consequence of this.

5.1 Application: Online Control with Policy Constraints

For illustration, we follow our own 'recipe' above to create an online control algorithm that has additional constraints on its policy. That is, we want to minimize regret while ensuring that $g_t(M_t) \leq 0$ for all M_t chosen by an online learning algorithm run on the DAC parametrization. We follow the existing approach in [13] for which we define the associated Lagrangian:

$$\mathcal{L}_t(M, \lambda) \doteq \tilde{c}_t(M) + \lambda g_t(M) - \frac{\delta}{2}\lambda^2 \tag{5.1}$$

where $\tilde{c}_t(M)$ is the proxy loss function associated with the original cost c_t, as in Algorithm 1. As in previous literature, the algorithm will run two no-regret procedures: (1) OGD for M_t on the sequence $\mathcal{L}_t(\cdot, \lambda_t)$, (2) OGA for λ_t on the sequence $\mathcal{L}_t(M_t, \cdot)$.

Algorithm 3 Primal-Dual Online Gradient Descent/Ascent for DAC

```
1: **Input:** stepsize $\eta$, memory $m$, set $\mathcal{M}$
2: Initialize $M_1 = 0$, set $M_{\leq 0} = M_1$ by convention
3: **for** $t = 1 \dots T$ **do**
4:     Play $M_t$, incur $c_t(x_t, u_t)$, and observe $c_t, x_{t+1}$
5:     Incur $g_t(M_t)$ constraint violation, and $g_t$
6:     Construct Lagrangian loss $\mathcal{L}_t(M, \lambda) \doteq \tilde{c}_t(M_t) + \lambda_t g_t(M_t) - \frac{\delta}{2}\lambda_t^2$
7:     Update $M_{t+1} = \Pi_{\mathcal{M}}\left(M_t - \eta \nabla_M \mathcal{L}_t(M_t, \lambda_t)\right)$, $\lambda_{t+1} = \lambda_t + \mu \nabla_\lambda \mathcal{L}_t(M_t, \lambda_t)$
8: **end for**
```

By Theorem 4 in [13], the above algorithm attains $\mathcal{O}(T^{1/2})$ regret on $\tilde{c}_t$ and $\mathcal{O}(T^{3/4})$ regret on g_t. By the exact same arguments as in [1], $\tilde{c}_t$ is a good approximation of the true cost. As such, on average, in the limit, the algorithm converges to the optimal controller with almost no policy constraints. Of course, one may find the regret rate on g_t to be unsatisfying. This luckily can also be fixed within the framework under an additional assumption. To do so, one can add a γ barrier to the original constraint, i.e. $g_t^\gamma(M) \doteq g_t(M) + \gamma$. Setting $\gamma = \Theta(T^{-1/4})$ yields a controller with 0 g_t regret. Under the additional assumption that the optimal DAC controller satisfying g_t^γ $M_\star^\gamma$ performs close to the optimal DAC controller satisfying g_t $M_\star$, i.e. $\sum_{t=1}^T \tilde{c}_t(M_\star^\gamma) - \sum_{t=1}^T \tilde{c}_t(M_\star) \leq O(\gamma T)$, we get $O(T^{3/4})$ c_t-regret with 0 g_t-regret.

This is an example in which steps (3-4) of the pipeline are actually not required, and instead one can directly port over online learning results to yield new algorithms for control with the desired guarantee.

6 Conclusion

In this literature review we focus on recent work in online convex optimization for control. We begin with background on classical policy parametrizations given different noise assumptions, and present the SLS convex relaxation for the linear policy class. We then give an in-depth overview of recent work on online convex optimization for control. This work in particular handles the case of adversarial perturbations and proposes the Disturbance-Action Policy Class which is amenable to online gradient descent and provides bounds on policy regret. We also test the DAC-OGD algorithm experimentally. We then examine extensions to the above work to more challenging scenarios, including: partial obervability, unknown systems, bandit feedback, better rates, and time-varying systems. Finally, we explore a new application of this online control pipeline to design an online controller which satisfies arbitrary convex constraints on its policy.

References

[1] Naman Agarwal, Brian Bullins, Elad Hazan, Sham M. Kakade, and Karan Singh. Online control with adversarial disturbances, 2019.

[2] Naman Agarwal, Elad Hazan, and Karan Singh. Logarithmic regret for online control, 2019.

[3] Oren Anava, Elad Hazan, and Shie Mannor. Online convex optimization against adversaries with memory and application to statistical arbitrage, 2014.

[4] Alon Cohen, Avinatan Hasidim, Tomer Koren, Nevena Lazic, Yishay Mansour, and Kunal Talwar. Online linear quadratic control. In *International Conference on Machine Learning*, pages 1029–1038. PMLR, 2018.

[5] Alon Cohen, Avinatan Hassidim, Tomer Koren, Nevena Lazic, Y. Mansour, and Kunal Talwar. Online linear quadratic control. In *ICML*, 2018.

[6] Abraham D. Flaxman, Adam Tauman Kalai, and H. Brendan McMahan. Online convex optimization in the bandit setting: gradient descent without a gradient, 2004.

[7] Paula Gradu, John Hallman, and Elad Hazan. Non-stochastic control with bandit feedback. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 10764–10774. Curran Associates, Inc., 2020.

[8] Paula Gradu, Elad Hazan, and Edgar Minasyan. Adaptive regret for control of time-varying dynamics, 2020.

[9] Elad Hazan. Introduction to online convex optimization. *Foundations and Trends® in Optimization*, 2(3-4):157–325, 2016.

[10] Elad Hazan, Sham M. Kakade, and Karan Singh. The nonstochastic control problem, 2020.

[11] Elad Hazan and Yuanzhi Li. An optimal algorithm for bandit convex optimization, 2016.

[12] Rudolf Kalman. Contribution to the theory of optimal control. *Bol. Soc. Mat. Mexicana*, 5, 02 2001.

[13] Mehrdad Mahdavi, Rong Jin, and Tianbao Yang. Trading regret for efficiency: Online convex optimization with long term constraints, 2012.

[14] Edgar Minasyan, Paula Gradu, Max Simchowitz, and Elad Hazan. Online control of unknown time-varying dynamical systems. In *Advances in Neural Information Processing Systems*, volume 34. Curran Associates, Inc., 2021.

[15] Max Simchowitz. Making non-stochastic control (almost) as easy as stochastic, 2020.

[16] Max Simchowitz and Dylan J. Foster. Naive exploration is optimal for online lqr, 2021.

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

[18] Yuh-Shyang Wang, Nikolai Matni, and John C. Doyle. A system level approach to controller synthesis, 2019.

[19] D. Youla, H. Jabr, and J. Bongiorno. Modern wiener-hopf design of optimal controllers–part ii: The multivariable case. *IEEE Transactions on Automatic Control*, 21(3):319–338, 1976.