2017 IEEE 56th Annual Conference on Decision and Control (CDC)
December 12-15, 2017, Melbourne, Australia

Thinking Fast and Slow: Optimization Decomposition Across Timescales

Gautam Goel Niangjun Chen Adam Wierman

Abstract— **Many real-world control systems, such as the smart grid and human sensorimotor control systems, have decentralized components that react quickly using local information and centralized components that react slowly using a more global view. This paper seeks to provide a theoretical framework for how to design controllers that are decomposed across timescales in this way. The framework is analogous to how the network utility maximization framework uses optimization decomposition to distribute a global control problem across independent controllers, each of which solves a local problem; except our goal is to decompose a global problem temporally, extracting a timescale separation. Our results highlight that decomposition of a multi-timescale controller into a fast timescale, reactive controller and a slow timescale, predictive controller can be near-optimal in a strong sense. In particular, we exhibit such a design, named Multi-timescale Reflexive Predictive Control (MRPC), which maintains a per-timestep cost within a constant factor of the offline optimal in an adversarial setting.**

I. INTRODUCTION

Modern control systems nearly always operate at multiple timescales. In the power grid, slow timescale economic dispatch is used to determine which baseload generators will supply power, while fast timescale frequency regulation is used to correct any imbalance between demand and supply that may arise [8]. In networking, software defined networks use a slow timescale "control plane" controller to decide where to send data packets, whereas fast timescale "data plane" controllers are responsible for routing the actual data [22]. Even human sensorimotor control exhibits the same phenomenon, with slow timescale behaviors such as trajectory planning and fast timescale behaviors such as involuntary reflexes [19], [31], [42], [46]. In fact, such timescale separation has consequently been proposed for the control of robotic systems [16], [46].

Thus, the design and analysis of multi-timescale control systems has received considerable attention. However, the design of control policies for multi-timescale control systems typically does not address the joint problem of designing control policies across timescales. Instead, controllers for each timescale are designed independently. For example, in the power grid, the slow timescale problem of economic dispatch is usually studied separately from the fast timescale problem of frequency regulation. Similarly, in software defined networking, the design of the control plane and data plane controllers are usually considered separately.

Gautam Goel, Niangjun Chen, and Adam Wierman are with the Department of Computing and Mathematical Sciences, California Institute of Technology. Email:{ggoel, ncchen, adamw}@caltech.edu
This work was supported by the NSF through grants AitF-1637598, CNS-1518941, CPS-154471, CNS-1319820, EPAS-1307794. We also thank Desmond Cai and Nik Matni for helpful discussions.

Across these and other applications timescale separation is *assumed* rather than derived, and the resulting subproblems are then studied independently, without guarantees about how they operate jointly. As a result, there are significant inefficiencies that are inherent to the resulting designs, even if each timescale problem is solved optimally. For example, recent work jointly designing economic dispatch and frequency regulation in the the power grid highlights significant inefficiency in designs that treated the two timescales independently [8].

In this paper, our goal is to develop a framework for *deriving* rather than *assuming* a timescale separation in global optimization problems. In particular, we adapt the idea of optimization decomposition from the domain of distributed control into the domain of multi-timescale control.

There is a vast literature on optimization decomposition, in fields as diverse as Internet congestion control [28], [44], smart grid control [8], [14], robotics [7], [16] and beyond [11]. The idea of this approach is to decompose a global optimization problem into smaller localized subproblems, each of which is solved by independent controllers. See [11] for a survey. In a similar way, our goal in this paper is to look for decompositions of a global optimization problem in *time*, as opposed to in *space*.

However, this goal is made challenging by the tight coupling between the timescales due to the underlying dynamics of the system under consideration. Typically, spatial optimization decomposition is performed for static optimizations, but in multi-timescale control the dynamics of the system cannot be ignored. Any slow timescale action will impact the future state via the dynamics and hence must be taken into account when designing the fast controller; conversely, any fast timescale action impacts the state seen by a slow controller and thus impacts its design as well. This makes it unclear whether it is possible to achieve a clean separation between controllers at different timescales.

A. Contributions of this paper

We make three main contributions in this paper.

Firstly, we introduce a simple but general model for studying multi-timescale optimal control. We consider a system subject to linear dynamics which is perturbed by noise; we make absolutely no assumptions about the nature of the noise, i.e., it may be random or even adversarial. This system can be controlled by two controllers, one of which is a traditional, "fast timescale" controller that can react immediately to the noise, and another which is a novel, "slow timescale" controller that is only able to react slowly,

978-1-5090-2873-3/17/$31.00 ©2017 IEEE

but which is empowered with access to more information than the fast controller and is potentially cheaper to use.

Secondly, we prove that one cannot expect to be able to design near-optimal controllers for multi-timescale control problems without the use of predictions. Our proof technique is based on a blackbox reduction to online convex optimization, a problem that has been intensively studied within the online algorithms community over the past decade. We use this reduction to describe a novel algorithm for the classic, fast timescale problem, a result which is of interest in its own right.

Thirdly, we introduce a new multi-timescale control policy, MRPC, and derive strong guarantees on its performance. In particular, we prove that the per-step cost incurred by our algorithm is at most a constant more than that incurred by the offline optimal. The design of our policy is motivated by a structural result about the offline optimal control action, which highlights a strong decomposition between fast and slow timescale controllers. Applying this idea to the design of the online algorithm, we are able to achieve a clean separation between timescales. Remarkably, our decomposition results in a purely reflexive, "dumb" fast controller, which performs no optimization or lookahead. Thus, all of the computational burden is shifted onto the slow, "smart" controller. This property of MRPC is desirable in many applications since the slow controller is often centralized and able to take a global view of the system, but the fast controllers are decentralized and myopic, e.g., the power systems, networking, and robotics examples mentioned above.

B. Related literature

This paper broadly falls into the category of optimal control [4], [47]. Typical methods for solving optimal control problem involve Pontryagin's principle [39], [40] and Hamilton-Jacobi-Bellman equation [4], [5]. With the rare exception of Linear Quadratic (LQ) systems, optimal control problems are generally nonlinear and do not admit analytical solutions. It is therefore necessary to solve optimal control problem via numerical methods. However, most existing numerical methods for optimal control (see [38] for a survey) do not scale well, and decomposing large scale problems into smaller subproblems is often required. There is large of literature on decomposition in the field of convex optimization and distributed computing. Common approaches include primal-dual decomposition [27], [33], alternating direction method of multipliers [6], [13] etc. These approaches have been crucial in developing distributed algorithms in various applications, e.g., communication networks [11], [28], [44], power systems [15], [34], robotics [37], [45]. However, these approaches are focused on spatial decomposition, and our focus in this paper is on temporal decomposition into independent controllers at different timescales.

The most related prior work is [30], which proposes an architectural decomposition of the optimal control problem into two layers: a top level trajectory planning problem that generates reference signals and a low level tracking problem that simply follows the reference points. However [30] does not provide optimality guarantees for the decomposition. Another related recent paper is [8], which focuses on temporal decomposition in the context of power systems. The work provides an optimality condition for timescale decomposition of optimal control in power systems. But, note that [8] considers a problem without dynamics. In this paper, we propose timescale decomposition for a general optimal control problem with linear dynamics and we provide provable performance guarantees.

II. MODEL

Our goal in this paper is to study the design of controllers for systems that operate at multiple timescales. To this end, we focus on a simple but general optimal control problem.

The multi-timescale problem we consider builds on the following optimal control problem, which operates at a single timescale:

$$\begin{aligned} \min_{x,f} \quad & \sum_{t=1}^{T} c_x(x_t) + c_f(f_t) \\ \text{s.t.} \quad & x_t = Ax_{t-1} + B^f f_t + w_t \\ & x_0 = 0 \end{aligned} \tag{1}$$

Here $x_t \in \mathbb{R}^n$ is the state variable, $f_t \in \mathbb{R}^n$ is the control action and $w_t \in \mathbb{R}^n$ is the disturbance. In our technical results, we assume that the control matrix B^f is invertible; considering the non-invertible case is an interesting direction for future work. The cost functions $c_x(\cdot), c_f(\cdot)$ are usually assumed to be non-negative and convex. The special case when each noise increment w_t is an i.i.d. Gaussian random variable and $c_x(\cdot), c_f(\cdot)$ are positive definite quadratic forms represents the Linear Quadratic Regulator (LQR) framework [12], [23], [43].

To extend (1) to a multi-timescale control problems, we introduce a "slow" controller. The slow controller reacts much less quickly to noise than the fast controller; however there are two potential benefits afforded by the existence of the slow controller.

First, in many situations the slow controller is centralized, and hence can use global information to make make better decisions than the decentralized, localized fast controllers. In our context, we model this by allowing the slow controller access to predictions of future noise increments. An example where the slow controller has this benefit is software defined networking, where the centralized controller has access to much more information than the local distributed controllers that provide congestion control via simple reactive policies [22]. Similarly, this type of interaction between a "smart" slow controller and a "reflexive" fast controller is common in robotics [16].

Second, in many cases the slow controller is much cheaper to operate than the fast controller. Thus, making use of the slow controller is crucial for minimizing cost. For example, in the smart grid cheap "baseload" generators are used to supply the bulk of demand, whereas fast and relatively expensive "peaker" generators are used to quickly correct any imbalances between supply and demand that may arise

[29], [35], [41], [8]. A similar distinction happens between economic dispatch and frequency regulation. In fact, a motivation for this paper comes from recent work in [8] that highlights a timescale separation between these controllers.

Adding a slow controller to the optimal control problem in (1) gives:

$$\begin{aligned} \min_{x,f,s} \quad & \sum_{t=1}^{T} c_x(x_t) + c_f(f_t) + c_s(s_t) \qquad (2) \\ \text{s.t.} \quad & x_t = Ax_{t-1} + B^f f_t + B^s s_t + w_t \\ & x_0 = 0 \\ & s_t = s_{t-1} \qquad \forall t \notin 0, k, 2k, \dots \end{aligned}$$

Here s_t denotes the control action of a slow controller. The constraint on s_t means that the slow controller cannot react quickly, i.e., it can only change its action every k timesteps.

This formulation leads to a intrinsic notion of timescales: there is a *fast timescale* consisting of the timesteps $\{1, 2, \dots\}$ in which the fast controller reacts, and a *slow timescale* consisting of the timesteps $\{1, k+1, 2k+1, \dots\}$ at which the slow controller reacts. Clearly one could continue to add other timescales to this formulation as well, but we focus on the two timescale case for clarity.

The focus of this paper is the design of a set of fast timescale and slow timescale controllers that operate independently but, together, approximate the optimal value of (2) without fully knowing the w_t's in advance. Motivated by the applications mentioned above, our goal is to develop designs where the slow controller is sophisticated and predictive, but the fast controller is simple and reactive.

One of the key differences of our approach compared to classical control theory lies in how we measure performance. Typical results in the control theory literature focus on settings with distributional assumptions about the noise vector w (for example, i.i.d. Gaussian), and seek to minimize the expected cost with respect to this distribution. In contrast, we use the approach of the online algorithms community and analyze the worst-case performance without distributional assumptions via the *competitive ratio* [17], [21].

Formally, the competitive ratio is defined as follows. Let OPT denote the optimal value of (2) and ALG the cost incurred by a specific algorithm. Then, the *competitive ratio* is defined as

$$CR(ALG) = \sup_{w} \frac{ALG}{OPT}.$$

This quantity measures the worst-case performance of an algorithm relative to the offline optimal and, in particular, makes no distributional assumptions on w. An algorithm is said to be *constant competitive* if its competitive ratio is bounded by a finite constant, independent of T.

We show in Section III that it is impossible to design constant competitive algorithms for (2) without using predictions of the future w_t. For this reason, our results focus on settings where algorithms have access to a limited number of noisy predictions of future w_t. In particular, we assume that, at the start of each slow timescale interval, we have estimates $\hat{w}_t$ of the true noise increments over that slow timescale interval. Importantly, we do not make distributional assumptions about the predictions or prediction errors.

Finally, one note on notation: throughout this paper, we follow the standard convention that vector valued variables are lowercase and matrix valued values are uppercase. When we write $\|A\|_a$, we mean the matrix norm of A induced by the vector norm $\|\cdot\|_a$. Also, we follow the convention of the algorithms community and often abuse notation to let an algorithm's name denote the cost it incurs.

III. Hardness of Multi-Timescale Control

Before turning to the design and analysis of an online algorithm for multi-timescale control, it is natural to ask what performance we should expect to be able to attain. In particular, should we expect to be able to find a constant competitive algorithm?

We show in this section that the answer is "no" in general, but that it becomes "yes" when the algorithm has access to a limited number of noisy predictions. This observation is crucial to the design and analysis of the algorithm we present in Section IV.

Interestingly, we cannot even expect to be able to design a constant competitive algorithm for the fast control subproblem of (2) given by (1). To show this, we prove below that (1) can be reformulated as an online convex optimization problem, which is a classical online algorithms problem that has received considerable attention in the last decade [1], [9], [10], [25], [26]. Importantly, competitive algorithms for online convex optimization algorithms do not exist in general, unless the algorithms are given access to noisy predictions about the future.

Formally, the equivalence to online convex optimization is stated as follows.

Proposition 1. *Suppose both $c_s(\cdot)$ and $c_f(\cdot)$ are a norm, $\|\cdot\|$, and suppose $B^f = B^s$. Then (1) is equivalent to (2) and, further, (1) can be reformulated as*

$$\min_{y} \sum_{t=1}^{T} c_t(y_t) + \|(B^f)^{-1}(y_t - Ay_{t-1})\|, \qquad (3)$$

where $y_0 = 0$ and $c_t(y_t) = c_x(y_t + v_t)$ for some v_t.

Proof. First, we need to argue that the multi-timescale problem (2) is equivalent to the fast timescale problem in (1) under the assumptions of the proposition. Suppose f^* is an optimal control action for (1). Then the pair $(f^*, 0)$ is an optimal pair of fast and slow control actions for (2), since it is feasible and achieves the same cost. Conversely, suppose (f^*, s^*) is an optimal pair of control actions for (2). Then the action $f^* + s^*$ is an optimal action for (1), by the same reasoning.

The second part of the proposition is to prove the reformulation of (1) as (3). To do this, we can iterate the dynamics and apply a change of variables. Specifically, iterating the

dynamics in (1) backwards in time, we see that

$$x_t = \sum_{i=1}^{t} A^{t-i} B^f f_i + \sum_{i=1}^{t} A^{t-i} w_i.$$

Next, introduce the change of variables

$$y_t = \sum_{i=1}^{t} A^{t-i} B^f f_i, v_t = \sum_{i=1}^{t} A^{t-i} w_i.$$

This yields (3). Notice that, given the solution to (3), we can construct the corresponding solution to (1) by setting $f_t = (B^f)^{-1}(y_t - Ay_{t-1})$. □

Problem (3) is a specific kind of online convex optimization problem known as a "Smoothed" Online Convex Optimization (SOCO). Specifically, convex cost functions $c_t(y_t) = c_x(y_t + v_t)$ arrive online and the goal of the online algorithm is to minimize the cost paid by choosing the sequence of actions $\{y_t\}$. The term $\|(B^f)^{-1}(y_t - Ay_{t-1})\|_f$ acts as a regularizer, penalizing choices that differ from the previous choice y_{t-1} under the dynamics of A. Proposition 1 provides a reduction from SOCO to (2): if we had a competitive online algorithm for (2), we would have one for SOCO as well.

SOCO problems have been intensely studied in the past decade due to their widespread applications in fields as diverse as motion tracking [24], power management for large data centers [26], geographical load-balancing for Internet scale applications [25] [36], and video streaming [32], [20]. In general, while there exist constant competitive algorithms for one dimensional [3], [25] and two dimensional [1] SOCO problems, it is unknown whether there exist constant competitive algorithms for higher dimensions. Further, it has been shown that SOCO problems are equivalent to Convex Body Chasing [18] in the sense that a competitive algorithm for one implies the existence of a competitive algorithm for the other [1]. This highlights the difficulty of obtaining constant competitive algorithms since Convex Body Chasing has been open for several decades.

Due to the difficulty of SOCO-style problems, much of the work on these problems has focused on settings where the online algorithms have access to (possibly noisy) predictions about future cost functions. For example, given perfect lookahead in a prediction window of length w, there exist algorithms whose competitive ratio is $1 + O(1/w)$, independent of dimension [25]. Similar positive results are possible in cases with noisy predictions, e.g., [9], [10].

Given Proposition 1, the positive results described above can yield effective algorithms for the single timescale, optimal control with linear dynamics in (1). To highlight this, we focus on a particularly promising algorithm from the SOCO literature called *Averaging Fixed Horizon Control (AFHC)*.

AFHC was introduced in [25] and has since been studied in [2], [9], [10]. AFHC is parameterized by the size of the prediction window it uses, which we denote by w. It works by averaging together the control actions of $w + 1$ independent Fixed Horizon Control (FHC) algorithms. The k-th FHC algorithm ($k = 1 \dots w+1$) starts at timestep k by greedily choosing the set of control actions that minimize the cost over time $[k, k+w]$, and then repeatedly chooses control actions to minimize cost over each consecutive length $w+1$ window. The control action output by AFHC is the average of the control actions of all $w + 1$ FHC algorithms.

In general, [25] proves that AFHC is $1 + O(1/w)$ competitive for SOCO problems with costs bounded below by a positive constant c_0. In the setting of this paper, we can prove a more precise result that highlights the impact of the structure of the dynamics.

Theorem 1. *Suppose each c_t is m-strongly convex and bounded below by a positive constant c_0. Then the competitive ratio of AFHC for (3) is at most*

$$1 + \frac{\|(B^f)^{-1}A\|^2}{2m(w+1)c_0}$$

and in particular is $1 + O(1/w)$.

This result highlights the ability of AFHC to perform well in a classic control problem, even in the *adversarial* setting. Further, the bound in the theorem highlights the impact of the structure of the dynamics on the performance of the algorithm. In general, as w tends to infinity the competitive ratio of AFHC will tend to one. This is unsurprising, since the algorithm will have access to more and more information about the future and hence will be able to make better decisions. However, Theorem 1 shows that even when w is small, AFHC can attain near optimal performance provided $\|(B^f)^{-1}A\|$ is sufficiently small. The matrix A can be interpreted as the gain of the system dynamics, and the matrix B^f as the gain of the fast controller; hence the expression $\|(B^f)^{-1}A\|$ is intuitively a measure of the fast controllers ability to counteract the gain of the system.

Proof. Let $\Omega_k = \{k, k + w, k + 2w, \dots\}$ be the set of times when the k-th FHC algorithm recomputes its control trajectory. Let y^* denote the optimal trajectory for (3) and y^k denote the choice of the k-th FHC algorithm. We define a function which measures the total cost incurred by a trajectory over $[s, s + w]$, starting from the point y^k_{s-1}:

$$g_{s,s+w}(y) = \sum_{t=s}^{s+w} c_t(y_t) + \sum_{t=s+1}^{s+w} \|(B^f)^{-1}(y_t - Ay_{t-1})\| + \|(B^f)^{-1}(y_s - Ay^k_{s-1})\|$$

Notice that $g_{s,s+w}$ is itself m-strongly convex; it is the sum of the m-strongly convex cost functions and the convex switching costs. Hence for all s we have

$$g_{s,s+w}(y^k) - g_{s,s+w}(y^*) \leq \nabla g_{s,s+w}(y^k)^T (y^k - y^*) - \frac{m}{2} \sum_{t=s}^{s+w} \|y^k_t - y^*_t\|^2$$

Notice that for all $s \in \Omega_k$, the gradient term vanishes, since by definition the k-th FHC algorithm chooses a trajectory

which minimizes $g_{s,s+w}$ at each $s \in \Omega_k$. Letting $d_t = \|y_t^k - y_t^*\|$ and summing up over all $s \in \Omega_k$ gives:

$$\sum_{s\in\Omega_k} g_{s,s+w}(y^k) - \sum_{s\in\Omega_k} g_{s,s+w}(y^*) \leq -\frac{m}{2}\sum_{s\in\Omega_k}\sum_{t=s}^{s+w} d_t^2$$

The first sum on the left hand side is the total cost incurred by the k-th FHC algorithm, which we denote by FHC^k. Using the definition of $g_{s,s+w}$ and the reverse triangle inequality, we can bound the second term:

$$\sum_{s\in\Omega_k} g_{s,s+w}(y^*) \leq OPT + \|(B^f)^{-1}A\| \sum_{s\in\Omega_k} d_{s-1}$$

from which we obtain

$$FHC^k - OPT \leq \sum_{s\in\Omega_k} \|(B^f)^{-1}A\| d_{s-1} - \frac{m}{2} d_{s-1}^2$$

Here we used the fact that $-\frac{m}{2}d_t^2$ is always nonpositive, so throwing away some of these terms can only increase the righthand side. Maximizing the summands in d_{s-1}, we obtain

$$FHC^k - OPT \leq \sum_{s\in\Omega_k} \frac{\|(B^f)^{-1}A\|^2}{2m}$$

We average all $w+1$ FHC algorithms and apply Jensen's Inequality to obtain

$$AFHC - OPT \leq \frac{1}{w+1}\sum_{t=1}^{T} \frac{\|(B^f)^{-1}A\|^2}{2m}$$

Finally we divide by OPT and use the bound $OPT \geq c_0 T$ to get a bound on the competitive ratio:

$$1 + \frac{\|(B^f)^{-1}A\|^2}{2m(w+1)c_0}$$

which establishes the $1 + O(1/w)$ claim. □

IV. Architectural Decomposition for Multi-Timescale Control

We now turn our attention to the joint multi-timescale control problem in (2), and focus on the co-design of fast and slow controllers. Recall that, while the slow controller cannot act as frequently, there are two benefits it usually provides: (i) it may have more information and computational power than the fast controller, e.g., in software defined networking and robotics, and (ii) it may be cheaper to operate than the fast controller, e.g., when scheduling generation in the smart grid. To capture these benefits of a slow controller, we consider a setting where the slow controller has access to noisy predictions but the fast controller does not. We also specifically highlight the case where the slow controller is cheaper to operate, though our results apply more generally.

Our main result in this section provides a performance bound for a new, near-optimal algorithm – *Multi-timescale Reflexive Predictive Control* (MRPC) – that consists of a simple, reflexive fast timescale controller and a predictive slow timescale controller. For concreteness and ease of presentation we focus on the case where the cost functions c_x, c_s, c_f in (2) are norms $\|\cdot\|_x, \|\cdot\|_s, \|\cdot\|_f$.

A. An overview of MRPC

Informally, MRPC works as follows. Over each slow timescale slot, the slow controller greedily plays the slow control action which minimizes the expected cost using the predictions $\hat{w}_t$, under the assumption that the fast controller will keep the state at zero. As the true noise increments w_t are revealed one by one, the fast controller myopically corrects any noise so as to keep the state at zero.

Formally, let $\hat{f}$ and $\hat{s}$ denote the fast and slow control actions of MRPC. Then, the operation of each is as follows:

$$\hat{s}_r = \min_s \left[k\|s_r\|_s + \sum_{t=r}^{r+k-1} \|(B^f)^{-1}(B^s s_r + \hat{w}_t)\|_f \right] \quad (4)$$

$$\hat{f}_t = -(B^f)^{-1}(B^s \hat{s}_r + w_t) \qquad t = r, \ldots r+k-1 \quad (5)$$

Notice that the fast controller is very simple; it uses no predictions and performs no optimization. All of the prediction and optimization is shifted onto the slow controller. This is consistent with how the two controllers are used in many applications, where the slow controller is often centralized, with access to global information, but the fast controllers are usually decentralized, localized, and computationally limited. For example, in the smart grid a slow timescale global optimization problem is solved (economic dispatch) and then localized fast timescale controllers myopically correct any deviations that may arise (frequency regulation).

B. Performance of MRPC

Our main technical result is a performance bound for MRPC. In particular, the following result shows that, despite the difficulty of the multi-timescale control problem, MRPC maintains a per-stage cost within a constant factor of the offline optimal, even when adversarial inputs are considered.

Theorem 2. *Assume the cost functions c_x, c_s, c_f in (2) are norms $\|\cdot\|_x, \|\cdot\|_s, \|\cdot\|_f$. Then MRPC has an average per-stage cost within a constant factor of optimal. Specifically,*

$$\frac{MRPC}{T} \leq \max\left(\frac{(1+\|A\|_x)\,\|(B^f)^{-1}\|}{c}, 1\right)\frac{OPT}{T} + 2\|(B^f)^{-1}\|E(\hat{w}, w)$$

where c is a constant such that $\|v\|_x \geq c\|v\|_f$ for all v and $E(\hat{w}, w)$ is the sample path average prediction error:

$$E(\hat{w}, w) = \frac{1}{T}\sum_{t=1}^{T} \|\hat{w}_t - w_t\|_f$$

Before moving to the proof, let us make a few remarks about Theorem 2. To begin, recall that even the single timescale problem could not be solved optimally by a fast timescale controller, and so the performance bound in Theorem 2 is surprisingly strong, especially given that the fast time scale controller in MRPC does not use any predictions – it is simply reflexive.

To get intuition for the bound itself, let us first look at the second term. The second term in the bound corresponds to the inefficiency due to noisy predictions. In particular, if we assume perfect lookahead (i.e $\hat{w}_t = w_t$ for all t), then the

second term disappears. Thus, we see that prediction error has only an additive effect. It is important to realize that the analysis makes no modeling assumptions on the form of the prediction error. The error can be adversarial or stochastic and the result still holds.

The first term bounds the per-step cost incurred by our algorithm relative to the per-step cost incurred by the offline optimal. To get intuition for it, consider the case where control costs dominate the state costs. Specifically, consider the case where $c \geq 2\|(B^f)^{-1}\|$, and there are no errors in predictions. In this case, we have $MRPC = OPT$. It is worth highlighting this result in words: *when state costs dominate control costs and prediction errors are small, our distributed algorithm achieves the optimal value of* (2). This is remarkable, since the offline optimal has a formidable advantage compared to our online algorithm - it knows the full noise vector w in advance, whereas during each slow timescale interval our online algorithm only has access to predictions about the noise in that interval.

Finally, it is important to note that Theorem 2 is incomparable to Theorem 1 since Theorem 2 compares to the offline optimal of the multi-timescale problem while Theorem 1 compares to the offline optimal of a single stage problem. In settings where the slow timescale controller is much cheaper than the fast timescale controller the cost difference between these problems can be arbitrarily large.

We now move to the proof of Theorem 2. The proof is technical, but also provide crucial intuition into the form of MRPC. In particular, the proof includes a lower bound on the offline optimal which motivates the decomposition between the fast and slow timescales used in the design of MRPC. It is this bound that highlights the ability to obtain timescale separation via the optimization decomposition in MRPC.

C. Proof of Theorem 2

Recall, that we are considering the case where the convex cost functions in (2) are norms. Further, it is convenient to absorb the constraint on the slow controller directly into the objective and rewrite (2) as

$$\begin{aligned} \min_{x,f,s} \quad & \sum_{r\in\mathcal{S}}\left[k\|s_r\|_s + \sum_{t=r}^{r+k-1}\|x_t\|_x + \|f_t\|_f\right] \\ \text{s.t.} \quad & x_t = Ax_{t-1} + B^f f_t + B^s s_t + w_t \\ & x_0 = 0 \end{aligned} \tag{6}$$

Here $\mathcal{S} = \{1, k+1, 2k+1, \ldots\}$ is the set of slow timescale steps, i.e., when the slow controller can change its control action.

The first step in the analysis is to establish a lower bound on the cost incurred by the offline optimal. As mentioned above, this lower bound highlights the decomposition between fast and slow used in the design of MRPC.

To prove the lower bound we make use of the following technical lemma.

Lemma 1. *Let $v \in \mathbb{R}^n$, and let $M \in \mathbb{R}^{n\times n}$ be an invertible matrix. Let $\|\cdot\|_a, \|\cdot\|_b$ be any two norms on $\mathbb{R}^n$, and let c be a constant such that $\|v\|_a \geq c\|v\|_b$ for all v. For all $\alpha, \beta > 0$ we have*

$$\min_x \alpha\|v + Mx\|_a + \beta\|x\|_b \geq \min\left(\frac{\alpha c}{\|M^{-1}\|_b}, \beta\right)\|M^{-1}v\|_b.$$

Proof. We have:

$$\begin{aligned} & \min_x \alpha\|v + Mx\|_a + \beta\|x\|_b \\ \geq & \min_x \alpha c\|M(x + M^{-1}v)\|_b + \beta\|x\|_b \\ \geq & \min_x \frac{\alpha c}{\|M^{-1}\|_b}\|x + M^{-1}v\|_b + \beta\|x\|_b \\ \geq & \min_x \frac{\alpha c}{\|M^{-1}\|_b}\Big|\|x\|_b - \|M^{-1}v\|_b\Big| + \beta\|x\|_b \end{aligned}$$

The first inequality follows from the equivalence of norms in finite dimensional linear spaces. The second inequality is because $\|y\| = \|M^{-1}My\| \leq \|M^{-1}\|\|My\|$, hence $\|My\| \geq \frac{1}{\|M^{-1}\|}\|y\|$ for all y. The last inequality is just the reverse triangle inequality.

The last optimization is an optimization over the scalar variable $\|x\|_b$ and it is easy to see that it is lower bounded by

$$\min\left(\frac{\alpha c}{\|M^{-1}\|}, \beta\right)\|M^{-1}v\|_b.$$

□

Now we are ready to prove the lower bound.

Lemma 2. *Letting OPT denote the optimal solution for* (6)*, we have:*

$$OPT \geq \min_s \sum_{r\in\mathcal{S}}\left[k\|s_r\|_s + C\sum_{t=r}^{r+k-1}\|(B^f)^{-1}(B^s s_t + w_t)\|_f\right]$$

where

$$C = \min\left(\frac{c}{(1+\|A\|_x)\,\|(B^f)^{-1}\|}, 1\right)$$

and c is a constant such that $\|v\|_x \geq c\|v\|_f$ for all v.

Proof. Suppose $\hat{x}, \hat{f}, \hat{s}$ are some arbitrary feasible choices of the decision variables in (6), which incur the associated cost $COST$. We have

$$\begin{aligned} COST = & \sum_{r\in\mathcal{S}}\sum_{t=r}^{r+k-1}\|\hat{x}_t\|_x + \|\hat{f}_t\|_f + \|\hat{s}_r\|_s \\ = & \sum_{r\in\mathcal{S}}\sum_{t=r}^{r+k-1}\|A\hat{x}_{t-1} + B^f\hat{f}_t + B^s\hat{s}_r + w_t\|_x \\ & \qquad + \|\hat{f}_t\|_f + \|\hat{s}_t\|_s \\ \geq & \sum_{r\in\mathcal{S}}\sum_{t=r}^{r+k-1}\Big[\|B^f\hat{f}_t + B^s\hat{s}_r + w_t\|_x - \|A\|_x\|\hat{x}_{t-1}\|_x \\ & \qquad + \|\hat{f}_t\|_f + \|\hat{s}_r\|_s\Big] \\ \geq & -\|A\|_x COST + \sum_{r\in\mathcal{S}}\sum_{t=r}^{r+k-1}\|B^f\hat{f}_t + B^s\hat{s}_r + w_t\|_x \\ & \qquad + (1+\|A\|_x)\left(\|\hat{f}_t\|_f + \|\hat{s}_r\|_s\right) \end{aligned}$$

from which we obtain the lower bound on $COST$ given by

$$\sum_{r\in\mathcal{S}}\left[k\|\hat{s}_r\|_s+\sum_{t=r}^{r+k-1}\beta\|B^f\hat{f}_t+B^s\hat{s}_r+w_t\|_x+\|\hat{f}_t\|_f\right]$$

where $\beta=\frac{1}{1+\|A\|_x}$. Since $\hat{x},\hat{f},\hat{s}$ were arbitrary feasible values, and in particular, could be taken to be the optimal values for (6), we obtain a lower bound on OPT given by

$$\min_{f,s}\sum_{r\in\mathcal{S}}\left[k\|\hat{s}_r\|_s+\sum_{t=r}^{r+k-1}\beta\|B^f f_t+B^s s_r+w_t\|_x+\|f_t\|_f\right].$$

Examining the structure of this expression, we observe that once each s_r is fixed, the resulting optimization in f resembles that in Lemma 1, which leads directly to the theorem. □

The lower bound has the following interpretation. Suppose the state is set at zero. After the slow controller has set its action to be s_r, the fast control action which corrects the remaining deviation from zero is $(B^f)^{-1}(B^s s_r+w_t)$, and our lower bound is the sum of the resulting costs (up to the constant C). Notice that the fast controller is extremely simple - all it does is continually correct any residual noise so that the state is always kept at zero. This is a crucial observation: *the form of the lower bound highlights a clear separation between a "smart", slow controller that does the planning and a "dumb" reactive fast controller*. This separation is then what we mimic in the design of MRPC, and also guides our analysis of the algorithm, as is evident in the following lemma, which provides an upper bound on the cost of MRPC.

Lemma 3.

$$MRPC\leq\min_s\sum_{r\in\mathcal{S}}\left[k\|s_r\|_s+\sum_{t=r}^{r+k-1}\|(B^f)^{-1}(B^s s_r+w_t)\|_f\right]+2\|(B^f)^{-1}\|\sum_{t=1}^{T}\|\hat{w}_t-w_t\|_f$$

Proof. Plugging our control actions into the cost function and applying the Triangle Inequality, we have

$$\begin{aligned}MRPC&=\sum_{r\in\mathcal{S}}\left[k\|\hat{s}_r\|_s+\sum_{t=r}^{r+k-1}\|(B^f)^{-1}(B^s\hat{s}_r+w_t)\|_f\right]\\&\leq\sum_{r\in\mathcal{S}}\left[k\|\hat{s}_r\|_s+\sum_{t=r}^{r+k-1}\|(B^f)^{-1}(B^s\hat{s}_r+\hat{w}_t)\|_f\right]+\sum_{t=1}^{T}\|(B^f)^{-1}(\hat{w}_t-w_t)\|_f\end{aligned}$$

Now we use the definition of $\hat{s}_r$ to obtain the upper bound

$$\min_s\sum_{r\in\mathcal{S}}\left[k\|s_r\|_s+\sum_{t=r}^{r+k-1}\|(B^f)^{-1}(B^s s_r+\hat{w}_t)\|_f\right]+\|(B^f)^{-1}\|\sum_{t=1}^{T}\|\hat{w}_t-w_t\|_f$$

Notice that the minimization in s depends on the estimates $\hat{w}_t$, not the true values w_t. Applying the Triangle Inequality once again allows us to produce an upper bound where the minimization is over the true values:

$$\min_s\sum_{r\in\mathcal{S}}\left[k\|s_r\|_s+\sum_{t=r}^{r+k-1}\|(B^f)^{-1}(B^s s_r+w_t)\|_f\right]+2\|(B^f)^{-1}\|\sum_{t=1}^{T}\|\hat{w}_t-w_t\|_f$$

This proves the claim. □

The combination of Lemmas (2) and (3) immediately yield Theorem 2.

V. Concluding Remarks

In this paper we present a simple and general model of multi-timescale control problems. We prove a hardness result using a blackbox reduction to online convex optimization, and show that predictions are necessary to construct a constant competitive algorithm. Further, we propose a simple control policy with a clean separation between timescales that uses only a small number of noisy predictions.

Our decomposition results in a sophisticated, predictive slow controller and a simple, reactive fast controller. This framework mirrors the architecture of many real-world control systems, where a slow, centralized controller guides the system towards global optimality while fast, decentralized controllers help to quickly counteract any perturbations that may arise. Remarkably, despite the simplicity of our fast controller and the fact that our policy has access to only limited information about the future, we derive strong guarantees on the performance of our policy. In particular, we prove that the per-step cost incurred by our algorithm is at most a constant more than that incurred by the offline optimal, and in some cases our policy even matches the offline optimal costs.

There are several natural and important problems left open by our work. Firstly, we do not consider delay in our model, though many real-world systems feature information-sharing constraints arising from delay. It would be natural to add such constraints to the slow controller. Secondly, it would be interesting to consider a hybrid model that is distributed across both time and space, i.e one that features both decentralization and multiple timescales. Most systems that operate across multiple timescales, such as the smart grid, also feature both centralized and localized controllers.

References

[1] A. Antoniadis, N. Barcelo, M. Nugent, K. Pruhs, K. Schewior, and M. Scquizzato. Chasing convex bodies and functions. In *Latin American Symposium on Theoretical Informatics*, pages 68–81. Springer, 2016.

[2] M. Badiei, N. Li, and A. Wierman. Online convex optimization with ramp constraints. In *Decision and Control (CDC), 2015 IEEE 54th Annual Conference on*, pages 6730–6736. IEEE, 2015.

[3] N. Bansal, A. Gupta, R. Krishnaswamy, K. Pruhs, K. Schewior, and C. Stein. A 2-competitive algorithm for online convex optimization with switching costs. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 40. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2015.

[4] D. P. Bertsekas, D. P. Bertsekas, D. P. Bertsekas, and D. P. Bertsekas. *Dynamic programming and optimal control*, volume 1. Athena Scientific Belmont, MA, 1995.

[5] F. Borrelli, M. Baotić, A. Bemporad, and M. Morari. Dynamic programming for constrained optimal control of discrete-time linear hybrid systems. *Automatica*, 41(10):1709–1721, 2005.

[6] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011.

[7] R. Brooks. A robust layered control system for a mobile robot. *IEEE journal on robotics and automation*, 2(1):14–23, 1986.

[8] D. Cai, E. Mallada, and A. Wierman. Distributed optimization decomposition for joint economic dispatch and frequency regulation. In *Decision and Control (CDC), 2015 IEEE 54th Annual Conference on*, pages 15–22. IEEE, 2015.

[9] N. Chen, A. Agarwal, A. Wierman, S. Barman, and L. L. H. Andrew. Online convex optimization using predictions. In *Proc. ACM SIGMETRICS*, pages 191–204. ACM, 2015.

[10] N. Chen, J. Comden, Z. Liu, A. Gandhi, and A. Wierman. Using predictions in online optimization: Looking forward with an eye on the past. In *Proceedings of the 2016 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Science*, pages 193–206. ACM, 2016.

[11] M. Chiang, S. H. Low, A. R. Calderbank, and J. C. Doyle. Layering as optimization decomposition: A mathematical theory of network architectures. *Proceedings of the IEEE*, 95(1):255–312, 2007.

[12] G. C. Chow et al. *Analysis and control of dynamic economic systems*. Wiley, 1975.

[13] P. L. Combettes and V. R. Wajs. Signal recovery by proximal forward-backward splitting. *Multiscale Modeling & Simulation*, 4(4):1168–1200, 2005.

[14] N. Deeb and S. Shahidehpour. Linear reactive power optimization in a large power network using the decomposition approach. *IEEE Transactions on power systems*, 5(2):428–438, 1990.

[15] T. Erseghe. Distributed optimal power flow using admm. *IEEE transactions on power systems*, 29(5):2370–2380, 2014.

[16] K. S. Espenschied, R. D. Quinn, R. D. Beer, and H. J. Chiel. Biologically based distributed control and local reflexes improve rough terrain locomotion in a hexapod robot. *Robotics and autonomous systems*, 18(1-2):59–64, 1996.

[17] A. Fiat. *Online Algorithms: The State of the Art (Lecture Notes in Computer Science)*. Springer, 1998.

[18] J. Friedman and N. Linial. On convex body chasing. *Discrete & Computational Geometry*, 9(3):293–321, 1993.

[19] S. Fusi, W. F. Asaad, E. K. Miller, and X.-J. Wang. A neural circuit model of flexible sensorimotor mapping: learning and forgetting on multiple timescales. *Neuron*, 54(2):319–333, 2007.

[20] V. Joseph and G. de Veciana. Jointly optimizing multi-user rate adaptation for video transport over wireless systems: Mean-fairness-variability tradeoffs. In *Proc. IEEE INFOCOM*, pages 567–575, 2012.

[21] R. M. Karp. On-line algorithms versus off-line algorithms: How much is it worth to know the future? In *IFIP Congress (1)*, volume 12, pages 416–429, 1992.

[22] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76, 2015.

[23] H. Kwakernaak and R. Sivan. *Linear optimal control systems*, volume 1. Wiley-interscience New York, 1972.

[24] H. M. Le, P. Carr, Y. Yue, and P. Lucey. Data-driven ghosting using deep imitation learning.

[25] M. Lin, Z. Liu, A. Wierman, and L. L. Andrew. Online algorithms for geographical load balancing. In *Int. Green Computing Conference (IGCC)*, pages 1–10. IEEE, 2012.

[26] M. Lin, A. Wierman, L. Andrew, and E. Thereska. Dynamic right-sizing for power-proportional data centers. *IEEE/ACM Trans. Networking*, 21(5):1378–1391, Oct 2013.

[27] S. H. Low and D. E. Lapsley. Optimization flow control. I. Basic algorithm and convergence. *IEEE/ACM Transactions on Networking*, 7(6):861–874, Dec 1999.

[28] S. H. Low, F. Paganini, and J. C. Doyle. Internet congestion control. *IEEE control systems*, 22(1):28–43, 2002.

[29] G. M. Masters. *Renewable and efficient electric power systems*. John Wiley & Sons, 2013.

[30] N. Matni and J. C. Doyle. A theory of dynamics, control and optimization in layered architectures. In *2016 American Control Conference (ACC)*, pages 2886–2893, July 2016.

[31] J. G. Milton. The delayed and noisy nervous system: implications for neural control. *Journal of neural engineering*, 8(6):065005, 2011.

[32] D. Niu, H. Xu, B. Li, and S. Zhao. Quality-assured cloud bandwidth auto-scaling for video-on-demand applications. In *INFOCOM, 2012 Proceedings IEEE*, pages 460–468, March 2012.

[33] D. P. Palomar and M. Chiang. Alternative distributed algorithms for network utility maximization: Framework and applications. *IEEE Transactions on Automatic Control*, 52(12):2254–2269, 2007.

[34] Q. Peng and S. H. Low. Distributed optimal power flow algorithm for radial networks, i: Balanced single phase case. *IEEE Transactions on Smart Grid*, 2016.

[35] R. Peters and C. Burda. *The Basics on Base Load: Meeting Ontario's Base Load Electricity Demand with Renewable Power Source*. Pembina Institute, 2007.

[36] A. Qureshi, R. Weber, H. Balakrishnan, J. Guttag, and B. Maggs. Cutting the electric bill for internet-scale systems. *ACM SIGCOMM Computer Communication Review*, 39(4):123–134, 2009.

[37] R. L. Raffard, C. J. Tomlin, and S. P. Boyd. Distributed optimization for cooperative agents: Application to formation flight. In *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, volume 3, pages 2453–2459. IEEE, 2004.

[38] A. V. Rao. A survey of numerical methods for optimal control. *Advances in the Astronautical Sciences*, 135(1):497–528, 2009.

[39] J. Raymond and H. Zidani. Pontryagin's principle for time-optimal problems. *Journal of Optimization Theory and Applications*, 101(2):375–402, 1999.

[40] R. W. Rishel. An extended pontryagin principle for control systems whose control laws contain measures. *Journal of the Society for Industrial and Applied Mathematics, Series A: Control*, 3(2):191–205, 1965.

[41] T. Senjyu, K. Shimabukuro, K. Uezato, and T. Funabashi. A fast technique for unit commitment problem by extended priority list. *IEEE Transactions on Power Systems*, 18(2):882–888, May 2003.

[42] M. A. Smith, A. Ghazizadeh, and R. Shadmehr. Interacting adaptive processes with different timescales underlie short-term motor learning. *PLoS Biol*, 4(6):e179, 2006.

[43] E. D. Sontag. *Mathematical control theory: deterministic finite dimensional systems*, volume 6. Springer Science & Business Media, 2013.

[44] R. Srikant. *The mathematics of Internet congestion control*. Springer Science & Business Media, 2012.

[45] I. Suzuki and M. Yamashita. Distributed anonymous mobile robots: Formation of geometric patterns. *SIAM Journal on Computing*, 28(4):1347–1363, 1999.

[46] Y. Yamashita and J. Tani. Emergence of functional hierarchy in a multiple timescale neural network model: a humanoid robot experiment. *PLoS Comput Biol*, 4(11):e1000220, 2008.

[47] K. Zhou, J. C. Doyle, K. Glover, et al. *Robust and optimal control*, volume 40. Prentice hall New Jersey, 1996.