Feedback Controller Parameterizations for Reinforcement Learning

John W. Roberts
CSAIL, MIT
Cambridge, MA 02139
Email: jwr@mit.edu

Ian R. Manchester
CSAIL, MIT
Cambridge, MA 02139
Email: irm@mit.edu

Russ Tedrake
CSAIL, MIT
Cambridge, MA 02139
Email: russt@mit.edu

***Abstract*—Reinforcement Learning offers a very general framework for learning controllers, but its effectiveness is closely tied to the controller parameterization used. Especially when learning feedback controllers for weakly stable systems, ineffective parameterizations can result in unstable controllers and poor performance both in terms of learning convergence and in the cost of the resulting policy. In this paper we explore four linear controller parameterizations in the context of REINFORCE, applying them to the control of a reaching task with a linearized flexible manipulator. We find that some natural but naive parameterizations perform very poorly, while the Youla Parameterization (a popular parameterization from the controls literature) offers a number of robustness and performance advantages.**

I. Introduction

Reinforcement Learning (RL) offers a very general set of methods for optimization which has been used successfully in numerous controls applications, particularly in robotics. Most uses of learning which have been successful on hardware involved learning an open-loop trajectory or control tape (see, for example [2], [13], [20], [23]). These trajectories can then be wrapped in stabilizing controllers designed using more traditional methods, such as Model Predictive Control [7] or the Linear Quadratic Regulator (LQR). More rarely, feedback policies themselves have been learned directly on hardware (e.g., [14], [24]), but not in situations in which instability greatly disrupts the learning process[1].

There is less work focusing on RL for feedback design when the system and task are such that instability can be an issue even as the policy converges. This has prevented the use of RL in high-performance tasks near the edge of stability, such as learning with hardware in-the-loop when the hardware could be damaged through the use of an unstable controller. This lack of work is not because RL is ill-suited to this domain of high-performance, low-stability-margin control, but because what seem to be the most natural parameterizations actually behave poorly in this domain.

In this paper we will study four parameterizations of linear feedback controllers and examine their performance on an example learning task in which instability can be a significant issue. The task is to quickly reach with a flexible manipulator to catch a ball, with learning performed using REINFORCE. While the manipulator is modeled as an open-loop stable linear system, it is underactuated, lacks full-state information and can easily be driven unstable through the choice of an inappropriate controller.

Two of the most natural-seeming parameterizations (state feedback gains with a fixed observer and a feedback controller transfer function) do not guarantee stability, and suffer both from non-convexity of the set of stabilizing controllers and from small changes in parameters causing a policy to go from high-performance to unstable. The other two parameterizations studied (LQR cost matrices with a fixed observer and the Youla Parameterization (YP)) *do* guarantee stability. The LQR cost matrix parameterization can offer high-performance, high-bandwidth controllers in situations where there is no delay, the cost function is of a form similar to the quadratic cost assumed by LQR, and the noise is not excessively structured. However, the Youla Parameterization offers a richer set of controllers, allowing for better performance when the cost function is significantly non-quadratic or when the noise is structured but non-Gaussian.

The remainder of this paper is organized as follows:

- **Section II** An introduction to the four parameterizations for linear feedback control studied in this paper: Feedback Controller Transfer Function, State Feedback Gains with a Fixed Observer, LQR Cost Matrices with a Fixed Observer and the Youla Parameterization
- **Section III** A description of the flexible arm ball-catching dynamics and task
- **Section IV** A presentation of the REINFORCE method used for learning in this paper
- **Section V** An analysis of the various parameterizations' performance in the context of the ball-catching task and REINFORCE learning
- **Section VI** A discussion of extensions of the YP to broader classes of systems
- **Section VII** A brief conclusion highlighting the critical points of this paper

[1]Feedback policies have been developed for stability critical systems through the use of Iterative LQR (see, for example, [1]), but the only policies which can result are those produced as a byproduct of iLQR , i.e., an LQR policy about the converged-to trajectory.

II. Linear Feedback Controller Parameterizations

The general task of designing a linear feedback controller may be thought of in the context of Fig. 1, where $K(s)$ and $P(s)$ are linear transfer functions in the laplace variable s. The designer must choose a $K(s)$ that, from measurements of the output y produces an input u that results in good performance by some designer-specified metric. In this section we consider four representations of $K(s)$ in the context of learning: Directly learning a linear transfer function for $K(s)$ in § II-A, representing $K(s)$ as a Kalman filter combined with learned state feedback gains (§ II-B), representing $K(s)$ as a Kalman filter then finding LQR feedback gains by learning LQR cost matrices Q and R (§ II-C) and finally, representing the controller in the Youla Parameterization (§ II-D). In § V we present their performance using REINFORCE learning on the example ball-catching task described in § III.

A. Feedback Transfer Function

Perhaps the simplest parameterization is to learn the transfer function of a feedback controller $K(s)$ (see Fig. 1) directly. As in theory $K(s)$ can be of arbitrary order, one must choose a set of $K(s)$ over which to actually search. In this paper, the $K(s)$ considered were of the form:

$$K(s) = \frac{\prod_{i=0}^{n} \beta_i s^i}{\prod_{i=1}^{n} (s - \alpha_i)}, \tag{1}$$

where the β_i are learned and the α_i are fixed. In effect, the poles of $K(s)$ are set by the designer while the numerator coefficients (and thus the zeros) are learned.

The α_i used for $K(s)$ are given below:

- α_1 through α_{15} are distributed logarithmically between $-10^{-0.5}$ and $-10^{0.5}$
- α_{16} and α_{17} are $-0.5 \pm 0.1i$
- α_{18} and α_{19} are $-1.1 \pm 0.05i$
- α_{20} and α_{21} are $-1.1 \pm 0.5i$
- α_{22} and α_{23} are $-2.3 \pm 1i$

These α_i were chosen because they covered a range of time-scales around the time-scale of the system, as well as several oscillatory modes with frequencies near the open-loop oscillation frequency of the system.

Learning $K(s)$ directly effectively allows the possibility of an arbitrary observer structure and the behavior of additional controller states (e.g., for integral control) to be learned. This parameterization is quite rich, with any linear controller possessing the poles specified with the α_i capable of being represented. However, less knowledge of the system is encoded in the structure of the parameterization than in the other parameterizations explored in this paper and thus achievable performance can depend greatly on the set of transfer functions $K(s)$ that are actually considered. Also, stability is not guaranteed in this case, with the relationship between the structure of $K(s)$ and the stability of the resulting system quite complicated.

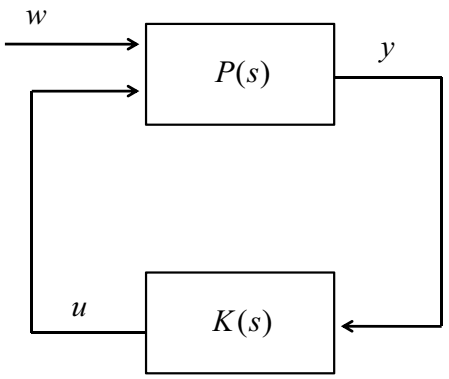

Fig. 1: Block diagram of the system used for Direct $K(s)$ Learning. Controllers are parameterized by selecting a linear system $K(s)$, with the specific form shown in Eq. (1).

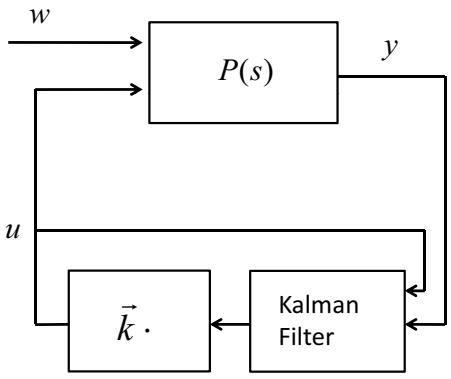

Fig. 2: Block diagram of the State Feedback Gain parameterization with a fixed observer. Controllers are parameterized by selecting n scalar gains k_1 through k_n where n is the number of states of the system.

B. State Feedback Gains with Fixed Observer

Fixing a high-performance observer (e.g., a Kalman Filter) and learning state-feedback gains seems like a very natural parameterization for learning a feedback controller (see Fig. 2), with LQR optimal controllers being within the set of available controllers. The dimension of the learned policy is the same dimension as the state, and it is easy to initialize the policy to LQR gains found for reasonably chosen Q and R matrices. However, as §V-B shows, the performance of this parameterization during learning is actually very poor.

C. LQR Cost Matrices

Directly parameterizing controllers in the space of LQR cost matrices and using a fixed observer (see Fig. 3) has a number of advantages. Perhaps foremost among these is that by restricting one's attention to the cone of positive-definite matrices for Q and R (a convex set) only stabilizing controllers are considered. However, the set of possible controllers that

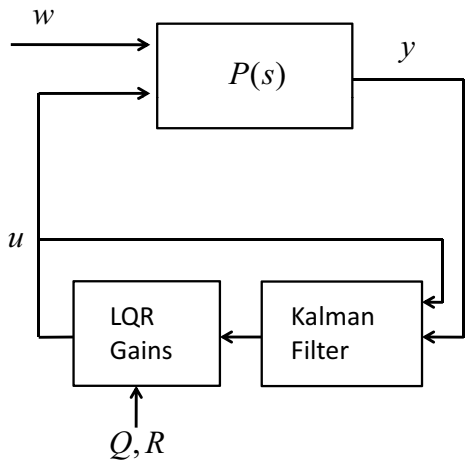

Fig. 3: Block diagram of the LQR Cost Matrix parameterization. Controllers are parameterized by two positive definite cost matrices Q and R, as is done in the design LQR controllers.

can be represented in this parameterization is actually quite limited due to being restricted to scalar gains on the state variables. Furthermore, the set of controllers is heavily over-parameterized. While symmetric positive definite Q and R matrices have $n(n+1)/2$ and $m(m+1)/2$ free parameters respectively for an n state system with m inputs, the space of controllers is of dimension nm. Thus, the set of controllers will always be over-parameterized, making for a possibly inefficient representation as learning must operate in an unnecessarily high-dimensional space. As learning performance has been shown to degrade with the number of learned parameters (see [21]), this is best avoided. Furthermore, dealing with delays in continuous time is non-trivial, and as the stability margins of LQR controllers with Kalman filters can be very small [10], even a small delay can be destabilizing.

D. The Youla Parameterization

As the Youla Parameterization (YP) is not widely known outside of the controls literature, we will first briefly describe the YP for single-input single-output linear systems, for which the main arguments are very simple, although similar parameterizations exist for much more complex systems (see Section VI).

Given a stable plant $P(s)$ and a feedback controller $K(s)$, both finite-dimensional SISO systems, the closed-loop system from u to y is:

$$H(s) := \frac{Y(s)}{U(s)} = \frac{P(s)}{1 - P(s)K(s)}. \tag{2}$$

Note that the relationship between the parameters of the feedback controller $K(s)$ and the closed loop system are highly nonlinear. The Youla parameter associated with this controller is defined as

$$Q(s) := \frac{K(s)}{1 - K(s)P(s)}. \tag{3}$$

Clearly for any $K(s)$ this $Q(s)$ exists.

Alternatively, given a plant $P(s)$ and a Youla parameter $Q(s)$, it is simple to construct the controller $K(s)$:

$$K(s) = \frac{Q(s)}{1 + Q(s)P(s)} \tag{4}$$

and the closed-loop system is

$$H(s) = P(s)[1 + Q(s)P(s)] \tag{5}$$

which is affine in $Q(s)$. Furthermore, since $P(s)$ is stable it is clear that $H(s)$ is stable if and only if $Q(s)$ is stable.

In summary, every controller $K(s)$ can be represented equivalently by a $Q(s)$, and the closed loop system is affine in $Q(s)$ and stable if and only if $Q(s)$ is, so the set of stable $Q(s)$ is complete affine parameterization of *all* stabilizing linear controllers for $P(s)$. Given a Youla parameter $Q(s)$, one can easily construct the feedback controller from (4). Note that it is not necessarily the case that $K(s)$ itself is stable considered as an isolated system.

Furthermore, since the closed-loop system $H(s)$ is affine in $Q(s)$, many important cost functions are convex in the $Q(s)$, including LQG, H^∞, and time-domain bounds on overshoot, rise-time, and settling time. For this reason it has long been a central tool for designing feedback controllers via optimization methods (see, e.g, [29], [30], [6], [3], [11] and many others).

There are many cases in which, even if the system model $P(s)$ is known well, online learning can be advantageous over model-based design. For example, if reference or disturbance commands have characteristics which are changing over time, or do not fit a mathematical framework which is easy to optimize over, such as Gaussian noise with a known spectral density, or bounded energy signals. Another case is when the cost/reward for the learning process is not easily represented mathematically, e.g. qualitative ranking by human teachers. Furthermore, the results proposed in this paper can be easily extended to cases in which $P(s)$ is only approximately known, and clear advantage could be gained by optimizing on hardware-in-the-loop experiments (see Section VI).

The most intuitive view of the YP is to consider it as a stable system $P(s)$ (possibly with a disturbance input w) in feedback with controller consisting of a copy of the system dynamics $P(s)$ (without the disturbance input) and a second arbitrary stable system $Q(s)$ acting on the difference (see Fig. 4a). As $Q(s)$ varies over all stable linear systems, the combination of $P(s)$ and $Q(s)$ will produce only stabilizing controllers.

While theoretically $Q(s)$ can be any stable linear system, in practice one must choose some limited subset of systems to study (much as was done in the case of representing $K(s)$). While the literature has proposed a number of affine representations of $Q(s)$ (see [6]), this paper will focus on choosing a $Q(s)$ of the form shown in Eq. (1). This representation of

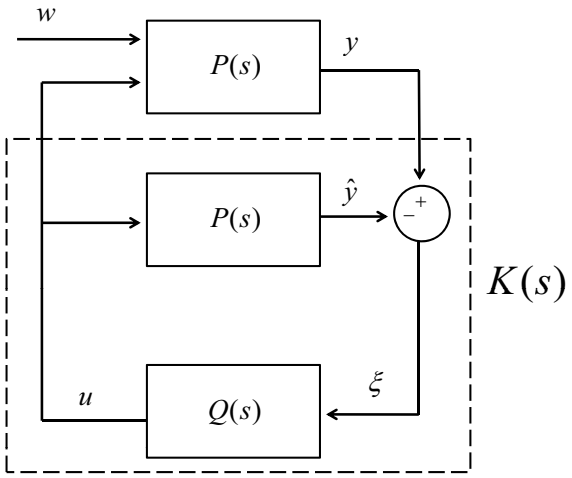

Fig. 4: Block diagram of the Youla Parameterization (YP). Controllers are parameterized by selecting a stable linear system $Q(s)$ of the form Eq. (1).

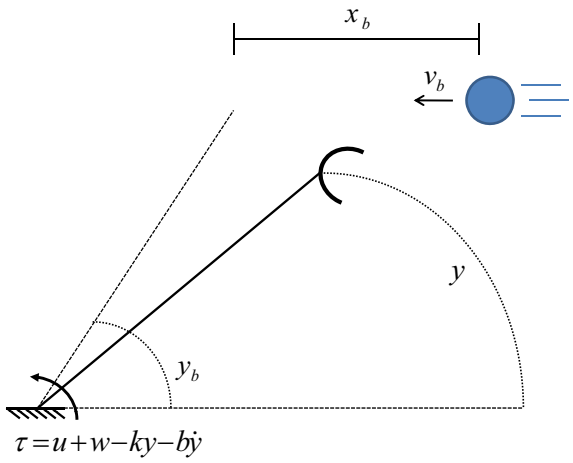

Fig. 5: Diagram of the simulated system. A ball enters at a distance x_b traveling at a speed v_b and at a height such that it will be caught if the manipulator's end effector reaches an angle y_b. The speed v_b and target angle y_b are drawn from uniform random distributions, with the distance x_b fixed. The input is a torque u, the disturbance a torque w and the output a measured angle y. The arm attempts to catch the ball by driving y to y_b in time to catch the ball, while minimizing arm velocity at the time the ball arrives. Costs are also imposed for torque, as well as additional costs for exceeding specified torque and angle values. Catching performance is judged by Eq. (10).

$Q(s)$ is clearly affine, is relatively rich and has shown good performance in practice.

III. Catching with a Flexible Arm: An Example System

The simulations in this paper are based on a simulated repeated ball-catching task for a flexible arm (see Fig. 5). The task is as follows: A flexible manipulator starts at rest at the zero position (i.e., $y = \dot{y} = 0$). A ball enters at a distance x_b traveling at a speed v_b and at a height such that it will be caught if the manipulator's end effector reaches an angle y_b. The speed v_b and target angle y_b are drawn from uniform random distributions, with the distance x_b fixed. The arm reaches for the ball, inducing flexural modes in the beam that can not be directly sensed or controlled. The frequency and magnitude of the flexural modes are of the same order as those of the reaching task, causing them to have non-negligible effects on end-effector position, and thus catching performance.

The arm is modeled as a homogeneous rectangular beam of length L, Young's Modulus E, linear density λ and cross-sectional moment of inertia I_y. The total rotational inertia of the arm and actuator about the revolute joint is I. The actuator acts with a torque u on the base of the arm while the end of the arm is measured, giving an angular measurement y. A spring and very weak damper with constants k and b respectively work to hold the arm at zero, making the system weakly open-loop stable. A structured disturbance acts on the base of the beam with a torque w. This disturbance is sinusoidal with fixed frequency ω_d and amplitude f_d, as shown below:

$$w(t) = f_d \sin(\omega_d t) \tag{6}$$

The system is modeled in the linear regime (i.e., small deformations and neglecting the centrifugal component of the dynamics) using the first three resonant modes of the system. The parameters may be seen in Table I.

L	1.5 m	I_y	2.77×10^{-7} m^4
E	3 GPa	b	0.01 Ns/m
k	0.5 kN/rad	v_b	60 ± 3 m/s
y_b	1 ± 0.05 rad	x_b	100 m
f_d	10 Nm	ω_d	π rad/s
λ	0.1 kg/m	I	1 kg m^2

TABLE I: Parameters of flexible-arm ball-catching task.

This system is underactuated and does not have full-state information, making it a non-trivial control task. While it is open-loop stable, it is easily capable of being driven unstable with a poorly chosen control law.

A catching trajectory was considered "good" if the end of the arm was at the desired position and a low velocity when the ball arrives. A trajectory was penalized for using torque and exceeding specified torque and angle thresholds. The specific function used to quantitatively evaluate a trajectory is shown in Eq. (10). An example of a successful catching trajectory can be seen in Fig. 7.

IV. The Learning Algorithm: Episodic REINFORCE

The algorithm used for learning is Episodic REINFORCE, in which the policy $\pi(y;\phi)$ is a function of the output of the system y and a set of parameters ϕ. The learning is performed episodically (i.e., the system states are reset between policy evaluations), and the stochasticity of the policy is on the policy parameters, rather than the outputs. The update appears in [26] for learning the mean of a Gaussian element, but we will briefly derive here the specific update used in this paper, in which a vector of parameters is learned with identical noise and learning rate.

Consider the REINFORCE update (formulated for cost instead of reward) in which a vector of parameters share the same learning rate eta:

$$\phi_{i+1} = \phi_i - \eta \left(J(\phi_i') - b\right) \frac{\partial}{\partial \phi_i'} \ln(g(\phi_i')), \tag{7}$$

with ϕ_i' the actual parameters used on trial i, b a cost baseline, $J(\phi_i')$ the cost associated with the policy parameters ϕ_i' and $g(\phi_i')$ the probability of using parameters ϕ_i' on trial i. If $g(\phi_i')$ is a multivariate Gaussian distribution with mean ϕ_i and independent noise on each element with covariance σ^2, the eligibility $\frac{\partial}{\partial \phi_i} \ln(g(\phi_{p_i}))$ can be computed as:

$$\frac{\partial}{\partial \phi_i} \ln(g(\phi_{p_i})) \propto (\phi_i' - \phi_i). \tag{8}$$

Clearly, $\phi_i' = \phi_i + \phi_{p_i}$, thus, after including all scalars in the learning rate, the update may be written:

$$\phi_{i+1} = \phi_i - \eta \left(J(\phi_i + \phi_{p_i}) - J(\phi_i)\right) \phi_{p_i}. \tag{9}$$

Note that while an averaged baseline is common in the literature, we have here used a second policy evaluation (i.e., a "nominal" policy evaluation). This was found to learn using fewer total policy evaluations on the system, even though two evaluations were needed per update. Thus, the resulting learning algorithm was performed as follows: at iteration i the policy ϕ_i is executed and used to find the baseline cost b. The policy is then perturbed by a small amount ϕ_{p_i}. This perturbed policy is executed, and the difference in performance is used to compute the policy update, which is shown above in Eq. (9):

The cost function $J(\phi)$ used for this experiment is shown below:

$$\begin{aligned} J(\phi) \quad = \quad & \int_0^{x_b/v_b} r u^2 dt + c_1 \left((y - y_b)|_{x_b/v_b}\right)^2 + c_2 \left(\dot{y}|_{x_b/v_b}\right)^2 \\ & + c_{sat} \max\left(0, \max((u/u_{sat})^2 - 1)\right) \\ & + c_{crash} \max\left(0, \max((y/y_{crash})^2 - 1)\right). \end{aligned} \tag{10}$$

It rewards achieving the desired angle and velocity $y = y_b, \dot{y} = 0$ when the ball arrives at $t = x_b/v_b$, while minimizing applied torque u and avoiding angles greater than y_{crash} and torques greater than u_{sat}. The constants r, c_1, c_2, c_{sat} and c_{crash} are used to weight the relative importance of these costs.

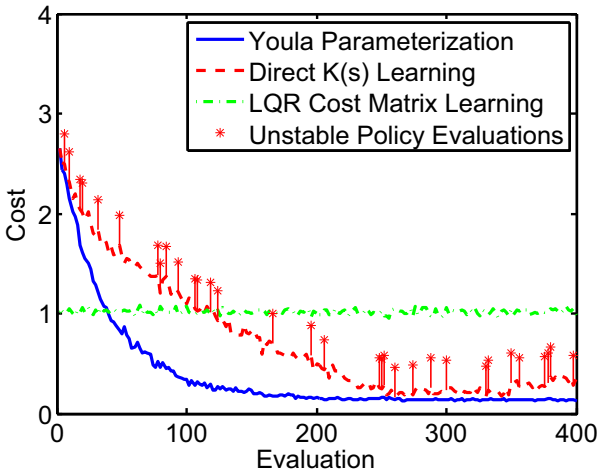

Fig. 6: Performance of YP and learning $K(s)$ directly for zero initial policy parameters and using tuned policy gradient. Ten trials of 200 iterations (400 evaluations) were performed for each curve, then averaged. The Y-axis is normalized by the cost of the initial LQR controller performing this task. Note how the weakly stable nature of the task and system result in many unstable policy evaluations even as the Direct $K(s)$ learning converges.

V. Results

The four parameterizations previously introduced in § II were used for REINFORCE learning of controllers for the ball-catching example task. The specific results for each parameterization are discussed below, but the primary result is that the Youla Parameterization provides the best learning performance as well as the best performance from the resulting controller. While clearly the performance for both the YP and directly learning $K(s)$ depend on the selection of the the roots of $K(s)$ (or $Q(s)$), several reasonable selections of the α_i produced effectively the same behavior.

A. Direct $K(s)$ Learning Results

Experiments on the flexible arm ball-catching task have shown that although learning converges reasonably well (see Fig. 6) for direct $K(s)$ learning, numerous unstable controllers are applied to the system throughout the learning process, even as the controller converges. This is due to the task effectively rewarding behavior on the margin of stability. Moreover, the convergence was significantly slower than that of the YP. Finally, while the resulting cost was similar to that of the YP (due to being capable of avoiding saturations and actually reaching the target point reasonably well), the actual trajectory followed was somewhat erratic. Thus, while this parameterization is reasonably well-suited to learning when instability is not a significant issue, and in theory a sufficiently rich $K(s)$ can produce the same controllers as any of the

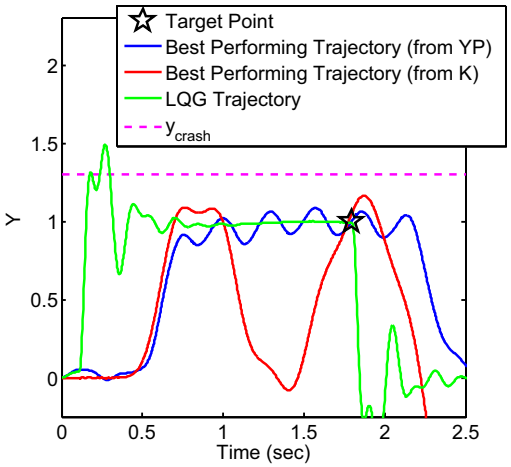

Fig. 7: Tracking performance of resulting controllers from the YP, LQR Cost Matrices Learning and Direct $K(s)$ Learning. Note that while all three parameterizations shown here hit the target point well for the nominal values, randomization of v_b (as well as greater use of actuation) degraded the performance of the controller from Direct $K(s)$ Learning and overshoot that could not be eliminated by learning caused the LQR Cost Matrix controller to incur unacceptable cost. As a result, the Youla Parameterization controller achieved the best performance.

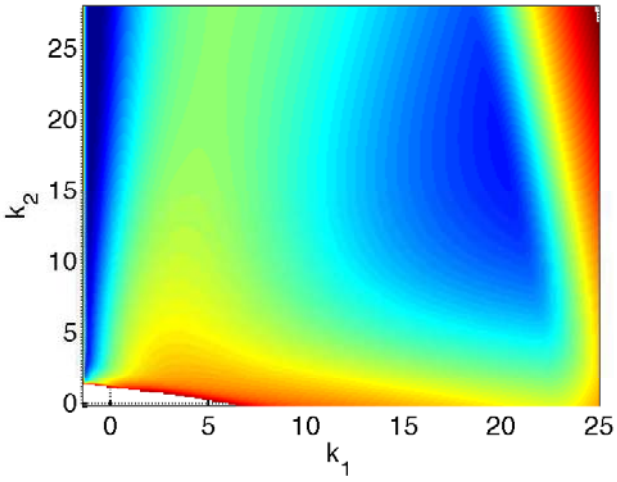

Fig. 8: Cost landscape for state feedback gains with a fixed observer for k_1 and k_2 with k_3 through k_8 set to their LQR values. Colors correspond to the log of the cost normalized to the LQR cost, and policies with with a cost greater than five times that of LQR are omitted. Note the very non-uniform gradients and non-convexity.

parameterizations examined here, in practice performance is worse for $K(s)$ with natural selections of the α_i than for similarly reasonable selections of $Q(s)$.

B. State Feedback Gain Learning Results

When applied to the example system, it was found that the sensitivity of cost to the state feedback gains is very non-uniform, resulting in either unacceptably slow learning (by learning slowly enough to identify the steep gradients warning of instability) or frequent unstable policy evaluations (by using perturbations and updates too large to identify oncoming instability). Both of these will result in unacceptably poor learning performance. Figure 8 shows the cost landscape for two of the eight feedback gains on the flexible arm catching task. The non-convexity and non-uniformity evident in the figure makes learning in this parameterization completely impractical. Because of this the other limitations of this approach (e.g., a limited set of representable controllers) are of minor importance compared with the fundamental difficulty of excessively non-uniform cost sensitivity to parameters.

C. LQR Cost Matrix Learning Results

When the LQR Cost Matrix parameterization was used for learning, it was found to improve policy performance very little, if at all, from its initial value. This is likely due to the quite limited set of controllers available in this parameterization, and the very large parameter changes required to effect reasonable change in the output. No unstable controllers were applied to the system, and reasonable trajectory tracking was achieved, but performance with respect to the given cost function did not improve. As the cost function was not quadratic on state and action, is is quite likely that the true optimal controller was not within the set of controllers parameterized by LQR cost matrices, and indeed it could be that no controller of performance comparable to that attained by other parameterizations was capable of being represented. Thus, the ultimately achievable performance is inferior to that obtainable by representations that parameterize a more general controller set.

D. Youla Parameterization Learning Results

As Figures 6 and 7 show, the YP provided the best overall performance, with good learning performance (i.e., quick, predictable convergence) and good ultimate controller performance. The fact that no intermediate trials were unstable is an added benefit, as it would allow for the application of learning using the YP even to sensitive hardware-in-the-loop learning systems that could be too delicate to experience excessively violent unstable controllers. The ultimate ball-catching performance is clearly quite good (though with imperfect noise rejection), and the control and output saturations that are problematic in the context of the LQR Cost Matrix learning are easily dealt with through learning in the YP. While rise-time and noise rejection were not as strong as for the LQR cost matrix control, these performance measures depend upon the selection of the representation for $Q(s)$, as with appropriate $Q(s)$ *any* stabilizing linear controller can be represented, including the LQR controller. While this paper has not addressed in detail how $Q(s)$ should be represented,

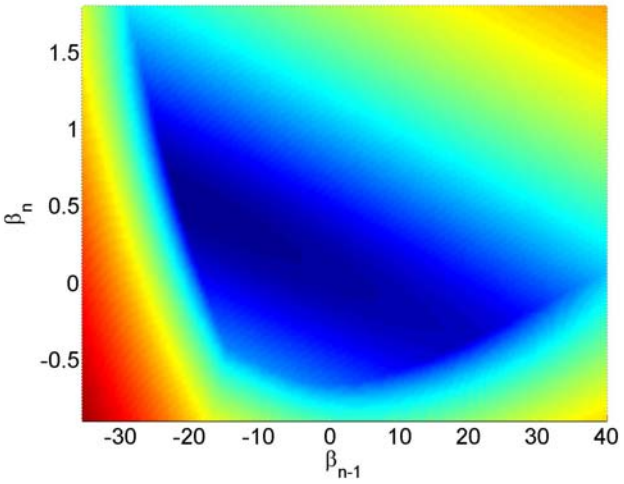

Fig. 9: Cost landscape for the Youla Parameterization for β_{n-1} and β_n with the rest of the β_i set to their converged to values. Colors correspond to the log of the cost normalized to cost of the converged to policy. Note the uniform gradients and convexity.

a wealth of work in the controls literature deals with this very problem, and exploring these different representations in the context of learning is an interesting future direction.

Furthermore, the convexity of many common cost functions in the YP (including Linear Quadratic costs and saturation costs in the time domain as were used here) can result in the convexity of the value function learning must descend (see Fig. 9). While this convexity is not guaranteed for arbitrary cost functions, if regions of the value function have this property it can still be advantageous to both the rate at which learning converges and to the avoidance of local minima.

VI. Extensions of the YP to More Complex Systems

As the Youla Parameterization clearly demonstrated the best performance, we will focus here on the possible extensions to the simple YP described in this paper. Due to their utility in feedback optimization problems, finding convex parameterizations of stabilizing controllers has been a major focus of control theory research for many decades. In this section we briefly overview a few major results related to the Youla parameterization which may also be applied in a policy learning architecture.

A. Unstable and Multivariable Systems

In this paper we have considered a stable single-input single-output system, mainly for simplicity's sake. The Youla parameterization for unstable and multivariable systems is classical and is widely used in control design, however the construction is somewhat more involved (see, e.g., [29], [6], [9]).

B. Rapid Switching Between Controllers

In this paper we have assumed that the learning is performed over repeated performances of a task, all starting from the same initial conditions. If, on the other hand, learning is performed by switching between candidate controllers during continuous operation (as in supervisory adaptive control) then stability under the switching process becomes an issue [16].

It is well known that even if two or more feedback controllers are stabilizing for a given system when considered in isolation, it is possible destabilize the system by rapidly switching between them. Giving tight conditions on stability of switched systems is intrinsically difficult – NP-Hard or undecideable, depending on the exact formulation [5], [25].

When the switching is chosen by the controller, stability can be ensured by enforcing a dwell-time between switching [16]. Particularly relevant to the work in this paper are designs in which switching can be performed arbitrarily rapidly. One such design is to represent all controllers in the Youla parameterization, with a reset on the states of $Q(s)$ at the moment of controller switch [12].

C. Uncertain Models

A natural application of learning or adaptive control is where the system model is not exactly known in advance. In this case, common methods include online system identification and control design, or direct policy search.

An alternative, made feasible in the framework proposed here, is to enforce that all control policies searched over are robustly stabilizing, but use the learning procedure to evaluate performance. In the control literature it is common to enforce robust stability via a small-gain condition by limiting the H^∞ norm of the closed-loop transfer function [30], [9], [6]. This constraint is convex in the Youla Parameterization. Characterizating uncertainty via a small-gain bound may be crude if more is known about the modelling errors. Convex parameterizations are also available for some tighter definitions of uncertainty [19].

D. Nonlinear Systems

For certain classes of nonlinear systems there exists parameterizations of all stabilizing controllers, similar to the Youla parameterization, for a variety of definitions of stability (see, e.g., [8], [18], [17], [4] and many others). In these cases, the Youla parameterization still gives a complete parameterization of all stabilizing controllers in terms of a stable nonlinear operator Q. However the closed-loop dynamics of the feedback system is no-longer affine in Q and the optimization process is much more difficult.

E. Decentralized Systems

For a long time, the problem of decentralized control - in which each controller/agent has access to data from a limited subset of sensors - has been a very difficult problem. Witsenhausen's classic counterexample showed that even in the linear-quadratic-gaussian case, the optimal decentralized controller may not be linear, and searching for the optimal

linear controller can be a nonconvex problem with many local minima [27]. Recently, very interesting results have emerged describing conditions under which the search for an optimal linear decentralized control is convex in the Youla parameter [22], [15]. An extension to the nonlinear case has also been considered [28].

Note that for these results to be applied directly to our method, it would be necessary that all controller locations have access to the reward signal. A substantially more challenging problem would be one where each individual controller can only see a subcomponent of the total reward.

VII. Conclusion

This paper has demonstrated the significance of appropriately choosing a parameterization for learning, with seemingly natural parameterizations making learning seem impossible, while well-chosen parameterizations provide high-performance both in the context of learning convergence and the resulting controller. For the flexible arm system studied here, the Youla Parameterization convincingly performed the best. Its flexibility, convexity in the case of many standard cost functions, guaranteed stability and richness make it very well-suited to learning in situations when the system model is known.

The extensions to the YP present in the controls literature offer a wealth of opportunities for study. From the appropriate representation of $Q(s)$ to applications in nonlinear and uncertain systems, many possibilities exist for improving learning through further study of the YP. With the stability guarantees offered by the Youla Parameterization, the authors hope more researchers will feel confident applying learning to actual hardware, offering improved performance and further insights into how best to use Reinforcement Learning to solve problems of real-world importance.

Acknowledgment

The authors would like to acknowledge the support of NSF Award IIS-0746194.

References

[1] Pieter Abbeel, Adam Coates, Morgan Quigley, and Andrew Y. Ng. An application of reinforcement learning to aerobatic helicopter flight. In *Proceedings of the Neural Information Processing Systems (NIPS '07)*, volume 19, December 2006.

[2] Pieter Abbeel, Varun Ganapathi, and Andrew Y. Ng. Learning vehicular dynamics, with application to modeling helicopters. *NIPS*, 2006.

[3] Anderson, J. M. Streitlien, K. Barrett, D.S. Triantafyllou, and M.S. Oscillating foils of high propulsive efficiency. *Journal of Fluid Mechanics*, 360:41–72, 1998.

[4] Brian D.O. Anderson. From youla-kucera to identification, adaptive and nonlinear control. *Automatica*, 34(12):1485 – 1506, 1998.

[5] Vincent D. Blondel and John N. Tsitsiklis. The boundedness of all products of a pair of matrices is undecidable. *Systems & Control Letters*, 41(2):135 – 140, 2000.

[6] Stephen P. Boyd and Craig H. Barratt. *Linear Controller Design: Limits of Performance*. Prentice Hall, NJ, 1991.

[7] E. F. Camacho and Carlos Bordons. *Model Predictive Control*. Springer-Verlag, 2nd edition, 2004.

[8] C. A. Desoer and R. W. Liu. Global parametrization of feedback systems with nonlinear plants. *Systems & Control Letters*, 1(4):249 – 251, 1982.

[9] John C. Doyle, Bruce A. Francis, and Allen R. Tannenbaum. *Feedback Control Theory*. Macmillan Publishing Company, New York, 1992.

[10] M. Green and D.J.N. Limebeer. *Linear Robust Control*. Prentice Hall, 1995.

[11] Joao P. Hespanha. *Linear Systems Theory*. Princeton Press, 2009.

[12] Joao P. Hespanha and A. Stephen Morse. Switching between stabilizing controllers. *Automatica*, 38(11):1905 – 1917, 2002.

[13] J. Kober, B. Mohler, and J. Peters. Imitation and reinforcement learning for motor primitives with perceptual coupling. In *From Motor to Interaction Learning in Robots*. Springer, 2009.

[14] J. Zico Kolter and Andrew Y. Ng. Policy search via the signed derivative. *Proceedings of Robotics: Science and Systems (RSS)*, 2009.

[15] L. Lessard and S. Lall. Quadratic invariance is necessary and sufficient for convexity. In *submitted to 2011 American Control Conference*, 2011.

[16] Daniel Liberzon. *Switching in Systems and Control*. Birkhauser, Boston, 2003.

[17] Wei-Min Lu. A state-space approach to parameterization of stabilizing controllers for nonlinear systems. *IEEE Transactions on Automatic Control*, 40(9):1576 –1588, sep. 1995.

[18] A. D. B. Paice and J. B. Moore. On the youla-kucera parametrization for nonlinear systems. *Systems & Control Letters*, 14(2):121 – 129, 1990.

[19] Rantzer, A., Megretski, and A. A convex parameterization of robustly stabilizing controllers. *Automatic Control, IEEE Transactions on*, 39(9):1802 –1808, sep. 1994.

[20] John W. Roberts, Lionel Moret, Jun Zhang, and Russ Tedrake. Motor learning at intermediate reynolds number: Experiments with policy gradient on the flapping flight of a rigid wing. In *From Motor to Interaction Learning in Robots*. Springer, 2009.

[21] John W. Roberts and Russ Tedrake. Signal-to-noise ratio analysis of policy gradient algorithms. In *Advances of Neural Information Processing Systems (NIPS) 21*, page 8, 2009.

[22] Rotkowitz, M., Lall, and S. A characterization of convex problems in decentralized control. *IEEE Transactions on Automatic Control*, 51(2):274 – 286, feb. 2006.

[23] Stefan Schaal, Auke Ijspeert, and Aude Billard. Computational approaches to motor learning by imitation. *Philosophical Transactions: Biological Sciences*, 358(1431):537–547, March 2003.

[24] Russ Tedrake, Teresa Weirui Zhang, and H. Sebastian Seung. Learning to walk in 20 minutes. In *Proceedings of the Fourteenth Yale Workshop on Adaptive and Learning Systems*, Yale University, New Haven, CT, 2005.

[25] Tsitsiklis, John N., Blondel, and Vincent D. The lyapunov exponent and joint spectral radius of pairs of matrices are hardwhen not impossibleto compute and to approximate. *Mathematics of Control, Signals, and Systems (MCSS)*, 10:31–40, 1997. 10.1007/BF01219774.

[26] R.J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.

[27] H. S. Witsenhausen. A counterexample in stochastic optimum control. *SIAM Journal on Control*, 6(1):131–147, 1968.

[28] J. Wu and S. Lall. Nonlinear youla parametrization and information constraints for decentralized control. In *American Control Conference (ACC), 2010*, pages 5614 –5619, Baltimore, MD, Jun 2010.

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

[30] Zames and G. Feedback and optimal sensitivity: Model reference transformations, multiplicative seminorms, and approximate inverses. *Automatic Control, IEEE Transactions on*, 26(2):301 – 320, apr. 1981.