Iterative Local Dynamic Programming

Emanuel Todorov and Yuval Tassa

***Abstract*— We develop an iterative local dynamic programming method (iLDP) applicable to stochastic optimal control problems in continuous high-dimensional state and action spaces. Such problems are common in the control of biological movement, but cannot be handled by existing methods. iLDP can be considered a generalization of Differential Dynamic Programming, inasmuch as: (a) we use general basis functions rather than quadratics to approximate the optimal value function; (b) we introduce a collocation method that dispenses with explicit differentiation of the cost and dynamics and ties iLDP to the Unscented Kalman filter; (c) we adapt the local function approximator to the propagated state covariance, thus increasing accuracy at more likely states. Convergence is similar to quasi-Netwon methods. We illustrate iLDP on several problems including the "swimmer" dynamical system which has 14 state and 4 control variables.**

I. Introduction

DESPITE an intense interest in optimal control theory over the past 50 years, solving complex optimal control problems even approximately remains a challenge. Existing general-purpose methods can be classified as *global* and *local*. Global methods, which have been the focus in Reinforcement Learning, are typically based on Bellman's Optimality principle. Such methods attempt to find optimal controls for all states, and consequently tend to run into the curse of dimensionality. Function approximation alleviates the problem somewhat. However, the state spaces of complex dynamical systems (especially biological systems) can be so vast that only a small fraction of all states are actually visited in the system's lifetime – and so the data needed for global optimization may be unobtainable. Local methods, more commonly studied in Control Theory (e.g. [2]), are related to Pontryagin's Maximum principle. Such methods find controls that are only locally-optimal in the vicinity of an "extremal" trajectory, and furthermore assume deterministic dynamics and yield open-loop controls. But the curse of dimensionality is avoided – which makes local methods a more natural point of departure in tackling complex problems.

A good blend of the advantages of local and global optimization is provided by Differential Dynamic Programming [1], as well as iterative LQG [7]. These are still local methods in the sense that they maintain a representation of a single trajectory and improve it iteratively using local information; the improvement however is based on dynamic programming – within a narrow "tube" in trajectory space. This feature allows construction of feedback control laws and some extensions to stochastic dynamics.

E. Todorov is with the Department of Cognitive Science, University of California San Diego, todorov@cogsci.ucsd.edu

Y. Tassa is with the Center for Neural Computation, Hebrew University of Jerusalem, Israel, tassa@alice.nc.huji.ac.il

This work was supported by the US National Science Foundation.

But as we show here, the 1st- and 2nd-order value function approximations employed by existing local methods make stochastic generalizations difficult. Even the simplest form of additive noise turns out to cause complications that require higher-order approximations.

The goal of the present paper is to develop a new local method (Section 3), that is better suited for stochastic problems and is generally more accurate. At the same time we want to preserve computational efficiency – by avoiding discretization of state or action spaces, as well as extensive Monte Carlo sampling.

A. Overview of existing local methods

Since local methods are rarely used in the Reinforcement Learning community, and are closely related to our new algorithm, we provide a brief overview. Consider a deterministic dynamical system $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$ with state $\mathbf{x}(t) \in \mathbf{R}^D$ and control $\mathbf{u}(t) \in \mathbf{R}^M$. Throughout the paper we assume that the initial state $\mathbf{x}_0$ and the final time T are specified. Given a final cost $h(\mathbf{x}(T)) \geq 0$ and a cost rate $\ell(\mathbf{x}, \mathbf{u}) \geq 0$, we want to find the open-loop control law $\mathbf{u}(t)$ minimizing the total cost $J(\mathbf{u}) = h(\mathbf{x}(T)) + \int_0^T \ell(\mathbf{x}, \mathbf{u})\, dt$. A necessary condition for $\mathbf{u}(t)$ and its corresponding state trajectory $\mathbf{x}(t)$ to be optimal is the existence of a "costate" trajectory $\mathbf{p}(t) \in \mathbf{R}^D$ satisfying Pontryagin's Maximum principle:

$$\begin{aligned} \mathbf{u} &= \underset{\mathbf{v}}{\operatorname{argmin}} \left\{ \ell(\mathbf{x}, \mathbf{v}) + \mathbf{f}(\mathbf{x}, \mathbf{v})^{\mathsf{T}} \mathbf{p} \right\} \\ -\dot{\mathbf{p}} &= \ell_{\mathbf{x}}(\mathbf{x}, \mathbf{u}) + \mathbf{f}_{\mathbf{x}}(\mathbf{x}, \mathbf{u})^{\mathsf{T}} \mathbf{p} \\ \dot{\mathbf{x}} &= \mathbf{f}(\mathbf{x}, \mathbf{u}) \end{aligned} \tag{1}$$

subject to the boundary conditions:

$$\begin{aligned} \mathbf{p}(T) &= h_{\mathbf{x}}(\mathbf{x}(T)), \\ \mathbf{x}(0) &= \mathbf{x}_0 \end{aligned}$$

Defining the Hamiltonian function

$$H(\mathbf{x}, \mathbf{u}, \mathbf{p}) \triangleq \ell(\mathbf{x}, \mathbf{u}) + \mathbf{f}(\mathbf{x}, \mathbf{u})^{\mathsf{T}} \mathbf{p},$$

Eq (1) becomes:

$$\begin{aligned} \mathbf{u} &= \underset{\mathbf{v}}{\operatorname{argmin}} H(\mathbf{x}, \mathbf{v}, \mathbf{p}) \\ -\dot{\mathbf{p}} &= H_{\mathbf{x}}(\mathbf{x}, \mathbf{u}, \mathbf{p}) \\ \dot{\mathbf{x}} &= H_{\mathbf{p}}(\mathbf{x}, \mathbf{u}, \mathbf{p}). \end{aligned}$$

If $\mathbf{u}(t)$ is a local minimizer of $J(\mathbf{u})$, it can be shown[1] that $\mathbf{p}(t)$ equals the gradient of the value function.

[1]Consider a closed-loop control law $\mathbf{u}(t, \mathbf{x})$ and its value function $V(t, \mathbf{x})$. Differentiating Bellman's equation $-V_t(t, \mathbf{x}) = \ell(\mathbf{x}, \mathbf{u}) + \mathbf{f}(\mathbf{x}, \mathbf{u})^{\mathsf{T}} V_{\mathbf{x}}(t, \mathbf{x})$ w.r.t. $\mathbf{x}$ and using the identity $\dot{V}_{\mathbf{x}} = V_{t\mathbf{x}} + V_{\mathbf{xx}}\dot{\mathbf{x}}$ yields $-\dot{V}_{\mathbf{x}} = \ell_{\mathbf{x}} + \mathbf{f}_{\mathbf{x}}^{\mathsf{T}} V_{\mathbf{x}} + H_{\mathbf{u}}\mathbf{u}_{\mathbf{x}}$. When $\mathbf{u}(t, \mathbf{x})$ is locally optimal the last term drops because $H_{\mathbf{u}} = 0$. By setting $\mathbf{p} = V_{\mathbf{x}}$ we recover Pontryagin's equation $-\dot{\mathbf{p}} = \ell_{\mathbf{x}} + \mathbf{f}_{\mathbf{x}}^{\mathsf{T}} \mathbf{p}$.

ODE: The minimization in Eq (1) implicitly defines a function $\mathbf{u}(\mathbf{x},\mathbf{p})$. If that function can be computed efficiently, one can solve the ODE by general-purpose numerical methods for boundary value problems (such as Matlab's BVP4C). Efficient minimization of the Hamiltonian is possible for problems in the general form

$$\begin{aligned}\mathbf{f}(\mathbf{x},\mathbf{u}) &= \mathbf{a}(\mathbf{x}) + B(\mathbf{x})\,\mathbf{u} \\ \ell(\mathbf{x},\mathbf{u}) &= q(\mathbf{x}) + \tfrac{1}{2}\mathbf{u}^{\mathsf{T}} R\mathbf{u}\end{aligned} \tag{2}$$

where the solution is $\mathbf{u}(\mathbf{x},\mathbf{p}) = -R^{-1}B(\mathbf{x})^{\mathsf{T}}\mathbf{p}$. Note that many problems of interest are in this form, because mechanical systems have multiplicative control-dependent dynamics, and cost functions are naturally decomposed into a state-dependent term and an energy-related control penalty.

Policy gradient: Another appealing property of Eq (1) is that it yields the gradient of the total cost $J(\mathbf{u})$ w.r.t. any open-loop control law $\mathbf{u}(t)$, not necessarily a minimizer of the Hamiltonian. Suppose $\mathbf{x}(t)$ and $\mathbf{p}(t)$ satisfy Eq (1) for a given $\mathbf{u}(t)$. Such a state-costate trajectory can be easily computed by first integrating $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x},\mathbf{u})$ forward in time, and then integrating $-\dot{\mathbf{p}} = \ell_{\mathbf{x}}(\mathbf{x},\mathbf{u}) + \mathbf{f}_{\mathbf{x}}(\mathbf{x},\mathbf{u})^{\mathsf{T}}\mathbf{p}$ backward in time. It can then be shown that $\delta J/\delta\mathbf{u} = H_{\mathbf{u}}(\mathbf{x},\mathbf{u},\mathbf{p})$. Thus the policy gradient can be computed analytically, and a discrete-time $\mathbf{u}(t)$ can be found with any general-purpose gradient-based optimization method.

DDP: Differential Dynamic Programming [1] also improves $\mathbf{u}(t)$ iteratively, but is a second-order method [5]. Instead of computing just the $\mathbf{x}(t)$ and $\mathbf{p}(t) = V_{\mathbf{x}}(t)$ that are consistent with the current $\mathbf{u}(t)$, DDP also computes the Hessian $V_{\mathbf{xx}}(t)$ along the trajectory $\mathbf{x}(t)$, by using quadratic approximations to ℓ and $\mathbf{f}$ centered at $\mathbf{x}(t)$. This locally quadratic approximation to the value function yields a locally linear feedback control law: For problems in the form of Eq(2) the optimal control is $\mathbf{u}(\mathbf{x}) = -R^{-1}B(\mathbf{x})^{\mathsf{T}}V_{\mathbf{x}}(\mathbf{x})$. Using the approximation $V_{\mathbf{x}}(\mathbf{x}+\Delta\mathbf{x}) = V_{\mathbf{x}}(\mathbf{x}) + V_{\mathbf{xx}}(\mathbf{x})\,\Delta\mathbf{x}$ yields the feedback component $-R^{-1}B(\mathbf{x})^{\mathsf{T}}V_{\mathbf{xx}}(\mathbf{x})\,\Delta\mathbf{x}$, which is linear in the displacement $\Delta\mathbf{x}$. This feedback component makes the policy improvement stage much more efficient, with a quadratic convergence rate comparable to Newton's method, which can offset the increased computational burden of calculating the quadratic approximations to ℓ and $\mathbf{f}$.

iLQG: The generalized iterative LQG method developed recently [7] also uses quadratic approximations to V and ℓ, but only a linear approximation to $\mathbf{f}$. Since the calculation of $\mathbf{f}$ is usually the computational bottleneck in such methods, iLQG is faster than DDP and equally accurate in practice.

II. The need for higher-order methods in stochastic problems

In deterministic problems the above methods converge to local minima of $J(\mathbf{u})$, and the only approximation errors arise from discretizing time. Intuitively, this is possible because Eq 1 propagates $V_{\mathbf{x}}$ along a single trajectory, without any knowledge of how V behaves along nearby trajectories. Noise destroys this property[2]: to compute $V_{\mathbf{x}}$ exactly, it seems necessary to have knowledge of V in a neighborhood of $\mathbf{x}(t)$ whose size is related to the noise magnitude. One might have hoped that 2nd-order methods like DDP and iLQG can approximate V in such a neighborhood, but as we show next such approximations are generally inadequate.

Consider the stochastic system $d\mathbf{x} = \mathbf{f}(\mathbf{x},\mathbf{u})\,dt + F(\mathbf{x},\mathbf{u})\,d\omega$, where $\omega(t) \in \mathbf{R}^W$ is a standard Brownian motion process, and let $\Sigma(\mathbf{x},\mathbf{u}) = F(\mathbf{x},\mathbf{u})\,F(\mathbf{x},\mathbf{u})^{\mathsf{T}}$. For a given control law $\mathbf{u}(t,\mathbf{x})$ the value function satisfies the Hamilton-Jacobi-Bellman (HJB) equation

$$\begin{aligned}-V_t(t,\mathbf{x}) = \ell(\mathbf{x},\mathbf{u}) + \mathbf{f}(\mathbf{x},\mathbf{u})^{\mathsf{T}} V_{\mathbf{x}}(t,\mathbf{x}) \\ + \tfrac{1}{2}\operatorname{tr}\left(\Sigma(\mathbf{x},\mathbf{u})\,V_{\mathbf{xx}}(t,\mathbf{x})\right)\end{aligned} \tag{3}$$

First- and second-order approximations treat $V_{\mathbf{xx}}$ as a constant independent of $\mathbf{x}$. Now suppose we have noise with constant Σ. Then the entire $\operatorname{tr}(\cdot)$ term in Eq 3 is treated as a constant, and therefore has no effect on the optimization process. Thus existing local methods are blind to the simplest form of additive noise[3].

We now discuss an important problem in biological control where additive noise makes all the difference. Suppose you want to keep your arm in a fixed posture. Normally you would do that with your muscles fairly relaxed. But if I tell you that I am about to shake your arm, you will probably stiffen it by co-activating opposing muscles (this works because muscle stiffness and damping scale with activation). Indeed, adjustment of limb impedance through muscle co-activation is an integral part motor behavior, and is extensively studied in the Motor Control literature. Optimal control provides a fruitful theoretical framework for biological movement [8]. Building more realistic models, however, will require efficient methods for nonlinear stochastic problems, and in particular methods that predict muscle co-activation.

Here is a concrete example that captures the essence of the above discussion. Let $\mathbf{x}(t) = [p(t)\,;\,k(t)]$ where p corresponds to position and k to stiffness. The control problem is

$$dp = -kp\;dt + \sigma\;d\omega \qquad h = 0 \tag{4a}$$

$$dk = (u-k)\;dt \qquad p(0) = 0 \tag{4b}$$

$$\ell(p,k,u) = p^2 + u^2 \qquad k(0) = 0 \tag{4c}$$

We want to keep $p = 0$ with minimal control energy. The position p decays to 0 at a rate proportional to k. The stiffness k is set by the control u through a low-pass filter. Since k cannot change instantaneously, it seems appropriate to

[2] Although a stochastic variant of the Maximum principle has been derived [3], it is a stochastic forward-backward differential equation that does not appear to yield local methods. Another way to see the problem is to repeat the derivation of Eq 1 starting with the stochastic HJB in Eq 3. This now yields terms containing $V_{\mathbf{xx}}$ and $V_{\mathbf{xxx}}$, so Eq 1 is no longer a closed system of equations.

[3] When Σ depends on $\mathbf{u}$ and/or $\mathbf{x}$, one can use a 2nd-order expansion of $\operatorname{tr}(\Sigma V_{\mathbf{xx}})$ and pick up some noise-dependent terms even when $V_{\mathbf{xx}}$ is assumed constant. We have done that in iLQG [7], but it remains to be established how close to locally-optimal the resulting control law is.

maintain $k > 0$ in order to resist unexpected perturbations. To see this more formally, consider the stochastic Hamiltonian $H = p^2 + u^2 - kpV_p^* + (u-k)V_k^* + \sigma^2 V_{pp}^*/2$ where V^* is the optimal value function. Since this problem is in the form of Eq 2, the optimal control is $u^* = -V_k^*/2$. Focusing on the state $p = k = 0$ and substituting u^* we get the minimized Hamiltonian $H^* = \sigma^2 V_{pp}^*/2 - \left(V_k^*\right)^2/4$. Consider for simplicity the case $T \to \infty$ and a discounted cost with time constant $\tau \to \infty$. In that case the HJB equation simplifies to $H^* = 0$, yielding

$$u^* = \sigma\sqrt{V_{pp}^*/2}$$

Thus the optimal control at state $[0;0]$ is roughly proportional to the noise magnitude (but note that V_{pp}^* varies with σ). The numerical solution is shown in Fig 1. In contrast, existing local methods yield the solution $u^* = 0$. This is because such methods ignore additive noise with constant covariance, and for $\sigma = 0$ the optimal control is clearly $u^* = 0$.

III. Iterative Local Dynamic Programming (iLDP)

Our new iLDP algorithm seeks a feedback control law $\mathbf{u} = \pi(t, \mathbf{x})$ that is a local minimum of the performance criterion $J(\pi) = \mathrm{E}_\omega\left(h(\mathbf{x}(T)) + \int_0^T \ell(\mathbf{x}, \pi(t,\mathbf{x}))\ dt\right)$ for the stochastic system $d\mathbf{x} = \mathbf{f}(\mathbf{x},\mathbf{u})\,dt + F(\mathbf{x},\mathbf{u})\,d\omega$, with $\mathbf{x}(0) = \mathbf{x}_0$ and $t \in [0;T]$. Each iteration of the algorithm starts with a $\pi(t,\mathbf{x})$ and generates a candidate improvement $\pi'(t,\mathbf{x})$, using quasi-Newton optimization. Global convergence is achieved via backtracking linesearch, which seeks a $\pi^{\mathrm{new}} = \alpha\pi' + (1-\alpha)\pi$, $0 < \alpha \le 1$, such that $J(\pi^{\mathrm{new}}) < J(\pi)$.

As in DDP and iLQG, the improvement is based on approximate dynamic programming in the vicinity of the average trajectory $\overline{\mathbf{x}}(k)$ generated by $\pi(k,\mathbf{x})$[4]. Unlike DDP and iLQG, the new algorithm works with any smooth approximation $\widetilde{V}(k,\mathbf{x})$ to the value function $V(k,\mathbf{x})$ corresponding to the control law $\pi'(k,\mathbf{x}) \cdots \pi'(K-1,\mathbf{x})$. At the last time step we set $\widetilde{V}(K,\mathbf{x}) = h(\mathbf{x})$. Then for each time step k, starting at $k = K-1$ and going backwards to $k = 1$, we perform the following dynamic programming backup:

1. Select a set of states $\left\{\mathbf{x}^{(n)}\right\}_{n=1\cdots N}$ that are clustered around the mean state $\overline{\mathbf{x}}(k)$.
2. At each $\mathbf{x}^{(n)}$ compute the optimal control $\mathbf{u}^{(n)}$ by minimizing the Hamiltonian

$$H(k,\mathbf{x},\mathbf{u}) \triangleq \ell(\mathbf{x},\mathbf{u}) + \mathbf{f}(\mathbf{x},\mathbf{u})^{\mathsf{T}}\widetilde{V}_{\mathbf{x}}(k+1,\mathbf{x}) + \tfrac{1}{2}\operatorname{tr}\left(\Sigma(\mathbf{x},\mathbf{u})\widetilde{V}_{\mathbf{xx}}(k+1,\mathbf{x})\right)$$

   The minimization uses a quasi-Newton method (or sequential Quadratic Programming when $\mathbf{u}$ is constrained) with initialization $\pi\left(k,\mathbf{x}^{(n)}\right)$. For added efficiency we can compute the Hessian only at $\overline{\mathbf{x}}(k)$ and reuse it for all $\mathbf{x}^{(n)}$. For problems in the form of Eq 2, H is quadratic[5] in $\mathbf{u}$ and so $\mathbf{u}^{(n)}$ is found immediately.
3. At each $\mathbf{x}^{(n)}$ approximate the value $v^{(n)} \triangleq V\left(k,\mathbf{x}^{(n)}\right)$ using the HJB equation

$$V\left(k,\mathbf{x}^{(n)}\right) \approx \Delta\ H\left(k,\mathbf{x}^{(n)},\mathbf{u}^{(n)}\right) + \widetilde{V}\left(k+1,\mathbf{x}^{(n)}\right)$$

   Note that it is possible to increase the accuracy of $v^{(n)}$ by sampling (Section 3.3)
4. Fit the function approximation $\widetilde{V}(k,\mathbf{x})$ to the set of state-value pairs $\left\{\mathbf{x}^{(n)}, v^{(n)}\right\}$. If a feedback control law $\pi'(k,\mathbf{x})$ defined for all $\mathbf{x}$ is needed, fit another function approximation to the state-control pairs $\left\{\mathbf{x}^{(n)}, \mathbf{u}^{(n)}\right\}$.

This general algorithm can be instantiated in many ways by making different choices in each step. The specific instantiations described next optimize computational efficiency, while allowing the user to set the balance between speed and accuracy.

A. Efficient function approximation through collocation

Fitting $\widetilde{V}$ and evaluating $\widetilde{V}, \widetilde{V}_{\mathbf{x}}, \widetilde{V}_{\mathbf{xx}}$ is done repeatedly, therefore these operations must be highly optimized. This calls for a linear function approximation scheme. Let $\phi(\mathbf{x}) \in \mathbf{R}^P$ be a vector of $P < N$ real-valued twice-differentiable basis functions (i.e. features). We will use the time-varying linear approximation

$$\widetilde{V}(k,\mathbf{x}) = \phi\left(\mathbf{x} - \overline{\mathbf{x}}(k)\right)^{\mathsf{T}}\mathbf{w}(k)$$

where $\mathbf{w}(k)$ is a parameter vector. Then $\widetilde{V}_{\mathbf{x}} = \phi_{\mathbf{x}}^{\mathsf{T}}\mathbf{w}$ and, with slight abuse of notation, $\widetilde{V}_{\mathbf{xx}} = \phi_{\mathbf{xx}}^{\mathsf{T}}\mathbf{w}$. Note that the average trajectory $\overline{\mathbf{x}}(k)$ is fixed throughout one iteration of the algorithm. The reason for subtracting $\overline{\mathbf{x}}(k)$ is to center the approximation at the average state, and also to speed-up the fitting process (see below).

The parameters $\mathbf{w}$ can be estimated at each time step k via linear regression. Defining the vector of target values $\mathbf{v} = \left[v^{(1)};\cdots;v^{(N)}\right]$, and the matrix of features $\Phi = \left[\phi\left(\mathbf{x}^{(1)} - \overline{\mathbf{x}}(k)\right)\cdots\phi\left(\mathbf{x}^{(N)} - \overline{\mathbf{x}}(k)\right)\right]$, the square error $\left\|\mathbf{v} - \Phi^{\mathsf{T}}\mathbf{w}\right\|^2$ is minimized by

$$\mathbf{w} = \left(\Phi\Phi^{\mathsf{T}}\right)^{-1}\Phi\mathbf{v}$$

The feature matrix Φ depends on the time-varying cloud of states $\left\{\mathbf{x}^{(n)}\right\}$, and so it might appear that we have to invert $\Phi\Phi^{\mathsf{T}}$ at each time step. Fortunately this can be avoided by using the following idea: we will always choose $\mathbf{x}^{(n)} = \overline{\mathbf{x}}(k) + \varepsilon^{(n)}$, where the set of "collocation" vectors $\left\{\varepsilon^{(n)}\right\}$ is the same for all times k. This creates a fixed cloud of states $\left\{\mathbf{x}^{(n)}\right\}$ sliding along the average trajectory $\overline{\mathbf{x}}(k)$. We then have $\widetilde{V}\left(k,\mathbf{x}^{(n)}\right) = \phi\left(\varepsilon^{(n)}\right)^{\mathsf{T}}\mathbf{w}(k)$, and so the feature matrix Φ is constant (not only over time, but also over iterations of the algorithm). Therefore $\left(\Phi\Phi^{\mathsf{T}}\right)^{-1}\Phi$ can be precomputed.

[4] In the rest of the paper we use discrete-time notation. The time step is Δ and the discrete-time index is k, thus $t = k\Delta$. The number of time steps is $K = \operatorname{floor}(T/\Delta)$.

[5] Control-dependent multiplicative noise is of particular interest in Motor Control. Suppose the i-th column of $F(\mathbf{u})$ is $C_i\mathbf{u}$. Then it can be verified that $\operatorname{tr}\left(\Sigma(\mathbf{u})V_{\mathbf{xx}}\right) = \mathbf{u}^{\mathsf{T}}\left(\sum C_i^{\mathsf{T}}V_{\mathbf{xx}}C_i\right)\mathbf{u}$.

Another benefit of propagating the collocation cloud $\mathbf{x}^{(n)}$ arises when the controller does not operate in the full state-space of the dynamics. This is most common in the transformation from an external to an internal ("egocentric") coordinate frame. While variables such as locations and angles are measured externally relative to some coordinate system, it makes no sense for the controller to have access to these values and would indeed prevent the learned policy from being invariant to translations and rotations. By basing our algorithm on the explicit propagation of the collocation cloud, any such "internalizing transformation" can be trivially applied to the $\mathbf{x}^{(n)}$ without any further modifications. It should however be noted that if such a transformation is state-dependent, the precomputation of $\left(\Phi\Phi^{\mathsf{T}}\right)^{-1}\Phi$ will no longer be possible.

The type and number of features $\phi(\mathbf{x})$ can be chosen in many ways, e.g. Gaussians or polynomials, and adapted to the specific problem and desired accuracy level. A particularly interesting feature set is: $1, x_i,\ x_i x_j,\ x_i^2 x_j$. It contains some 3rd order terms (and so can avoid the problems illustrated in Section II but has order D^2 elements. This feature set also has the property $\phi_1(\mathbf{x}) = 1$ and $\phi_{p>1}(0) = 0$, which means that $w_1(k) \approx V(k, \overline{\mathbf{x}}(k))$ and so the remaining parameters $w_{2\ldots P}$ are devoted to fitting derivative-related information.

A similar function approximator can be used to fit $\pi'(k, \mathbf{x})$ given $\left\{\mathbf{x}^{(n)}, \mathbf{u}^{(n)}\right\}$. Note however that the basic iteration (Steps 1-4) does not require access to a $\pi'(k, \mathbf{x})$ that is defined for all $\mathbf{x}$.

B. Adapting the collocation cloud to the state distribution

How should the collocation vectors be chosen? A basic implementation can simply use a spherical zero-mean cloud $\left\{\varepsilon^{(n)}\right\}$. However, a more accurate value function approximation is obtained if the cloud resembles the distribution of states resulting from π'. This is hard to accomplish because π' is constructed after $\left\{\varepsilon^{(n)}\right\}$. Instead we will make $\left\{\varepsilon^{(n)}\right\}$ consistent with the current control law π, and hope that π and π' are similar. At the same time we will preserve the efficiency of the collocation method.

Before the backups (Step 1-4) can begin, we have to apply π to the stochastic system[6]

$$\mathbf{x}(k+1) = \mathbf{x}(k) + \Delta\ \mathbf{f}(\mathbf{x}(k), \pi(k, \mathbf{x}(k))) + \sqrt{\Delta}\ F(\mathbf{x}(k), \pi(k, \mathbf{x}(k)))\,\xi(k) \quad (5)$$

and compute the average trajectory $\overline{\mathbf{x}}(k)$. A basic implementation can approximate $\overline{\mathbf{x}}(k)$ using several sample trajectories, or perhaps even a single noiseless trajectory with $F = 0$. But a better way to find $\overline{\mathbf{x}}(k)$ is to propagate a Gaussian approximation $\mathrm{N}(\overline{\mathbf{x}}(k); S(k))$ to the distribution of $\mathbf{x}(k)$ – which we do using the Unscented Transform [4]. This involves evaluating $\mathbf{f}$ at $\overline{\mathbf{x}}(k)$ and $\overline{\mathbf{x}}(k) \pm \sqrt{3}\mathbf{q}_i$, where $\mathbf{q}_i$ are the columns of the matrix square root $QQ^{\mathsf{T}} =$ $S(k)$. Then $\overline{\mathbf{x}}(k+1); S(k+1)$ are estimated as certain weighted averages [4]. Finally, we add to $S(k+1)$ the noise contribution $\Delta F F^{\mathsf{T}}$ evaluated at $\overline{\mathbf{x}}(k)$. As an aside, note that once we have the $(\overline{\mathbf{x}}; S)$ resulting from some control law π we can approximate $J(\pi)$. This is useful in the linesearch phase, where we need to compute $J(\pi^{\mathrm{new}})$ without running the dynamic programming algorithm.

The function approximator can now be adapted to the covariance $S(k) = Q(k)\,Q(k)^{\mathsf{T}}$. Suppose we have a spherical set $\left\{\varepsilon^{(n)}\right\}$ such that $\langle\varepsilon\rangle = 0$ and $\left\langle\varepsilon\varepsilon^{\mathsf{T}}\right\rangle = r\mathrm{I}$; the scale r sets the "locality" of the approximation. For the backup at stage k we will use the transformed collocation vectors $\underline{\varepsilon}^{(n)} = Q(k)\,\varepsilon^{(n)}$, which have the desired covariance $rS(k)$.

In section 3.1 it was possible to precompute $\left(\Phi\Phi^{\mathsf{T}}\right)^{-1}\Phi$ because $\left\{\varepsilon^{(n)}\right\}$ was fixed. How can this appealing property be preserved here, when we now have a new $\left\{\underline{\varepsilon}^{(n)}\right\}$ at each stage? The idea is to transform not only $\varepsilon^{(n)}$ but also the features ϕ. We will now use[7]

$$\underline{\phi}(\mathbf{x}) \triangleq \phi\left(Q(k)^{-1}\mathbf{x}\right)$$

so that $\underline{\phi}\left(\underline{\varepsilon}^{(n)}\right) = \phi\left(Q(k)^{-1}Q(k)\,\varepsilon^{(n)}\right) = \phi\left(\varepsilon^{(n)}\right)$. Thus Φ remains unchanged, while the backups now use a new set of collocation vectors $\left\{\underline{\varepsilon}^{(n)}\right\}$ at each time step k. The new function approximator evaluated at the new collocation vectors is $\widetilde{V}\left(k, \overline{\mathbf{x}}(k) + \underline{\varepsilon}^{(n)}\right) = \phi\left(\varepsilon^{(n)}\right)^{\mathsf{T}}\mathbf{w}(k)$.

C. Sampling-based refinements

The approximation errors in $\widetilde{V}(k, \mathbf{x})$ can in principle accumulate, because each backup uses value function estimates from the previous backup. While we have not found that to be a problem in practice, it is desirable to have ways of improving approximation accuracy – especially towards the end of the main iteration when we may want to refine the final result. An obvious way to do this is to increase the spread and density of the collocation set $\left\{\varepsilon^{(n)}\right\}$, as well as the number of parameters $\mathbf{w}$. Alternatively, one can use sampling.

The value function estimates $v^{(n)}$ obtained at stage k in Step 3 can be improved as follows: initialize a number of trajectories at $\mathbf{x}^{(n)}$, sample Eq 5 using $\pi'(k, \mathbf{x}) \ldots \pi'(K-1, \mathbf{x})$, and average the empirical costs. Note that this procedure avoids the potential accumulation of bias, but introduces variance. It is also much slower than adding extra collocation vectors. This is because trajectories starting at each time k have to be sampled until the final time K, resulting in order K^2 samples.

D. Related Work

The use of high-order local dynamic programming methods for the solution of biological control problems was pioneered in [10]. There, Morimoto and Atkeson use the

[6] $\xi(k) = (\omega(k\Delta+\Delta) - \omega(k\Delta))/\sqrt{\Delta} \sim \mathrm{N}(0; I)$. The square root term $\sqrt{\Delta}$ appears because the covariance of a Brownian motion process grows linearly with time.

[7] To better understand this transform, consider the feature $\phi(\mathbf{x}) = (\mathbf{x}-\mathbf{c})^{\mathsf{T}}(\mathbf{x}-\mathbf{c})$. Then $\underline{\phi}(\mathbf{x}) = \phi\left(Q^{-1}\mathbf{x}\right) = \left(Q^{-1}\mathbf{x}-\mathbf{c}\right)^{\mathsf{T}}\left(Q^{-1}\mathbf{x}-\mathbf{c}\right) = (\mathbf{x}-Q\mathbf{c})^{\mathsf{T}}S^{-1}(\mathbf{x}-Q\mathbf{c})$. So the center $\mathbf{c}$ is scaled like $\varepsilon^{(n)}$, and the Cartesian metric is replaced by S^{-1}. The same applies to Gaussians.

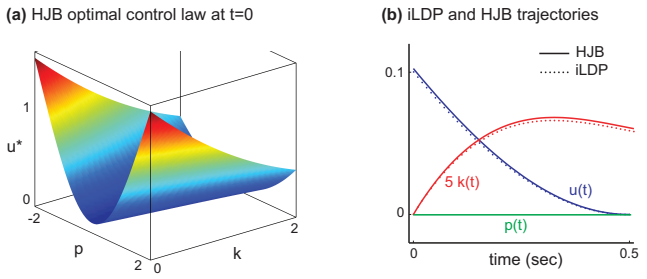


Fig. 1. Solution to the simple problem defined in Eqs (4) with $T = 0.5_{sec}$ and $\sigma = 1$. (a) The optimal value function at time $t = 0$, computed via numerical solution of the HJB equation on a dense grid. (b) Average state-control trajectory obtained by the discretized HJB solution and by the iLDP method. The variable k is scaled by 5 for visibility.

game-theoretic *minimax quadratic game* to extend the DDP algorithm. While the regular controller seeks to minimize the cost, a second adversarial controller (which models noise under a risk-sensitive integral) is assumed to simultaneously attempt to maximize it, leading to more prudent or robust policies. Minimax DDP is very appealing theoretically, especially due to the explicit modeling of noise, but not easy to implement numerically since the solution is not longer a local extremum but rather a saddle point, which is more difficult to find (e.g. monotonicity during convergence cannot be assured).

Finally, we note that iLDP has commonalities with the recently developed PSDP algorithm [9] although the latter was developed for discrete domains and does not exploit any differential properties of continuous problems. Specifically, PSDP is not an intrinsically local algorithm and is not naturally immune to dimensionality issues. While PSDP can be adapted to continuous domains it still requires as input, a sequence of distributions μ_k which provide a prior as to where the solution lies, but in the high-dimensional spaces in which we seek to solve control problems, that knowledge amounts to a rather large part of the solution.

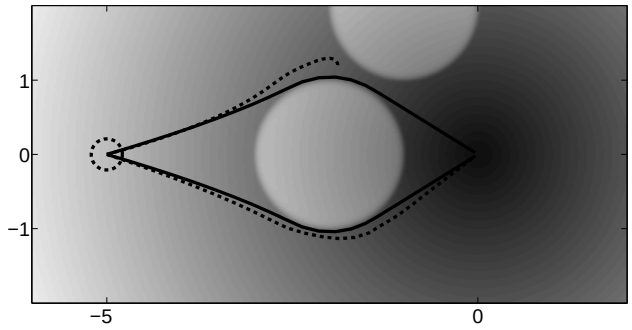


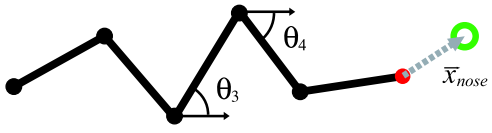
Fig. 2. Solution to a simple obstacle avoidance problem. For deterministic dynamics (solid lines) the solution trajectories pass very close to the obstacle. When the dynamics are stochastic and the estimated state covariance is propagated (dotted circle indicates 1 standard deviation), the solution trajectories (dotted lines) veer away from the obstacle (bottom solution) and are unable to pass in the narrow channel between the obstacles.

E. Simulations

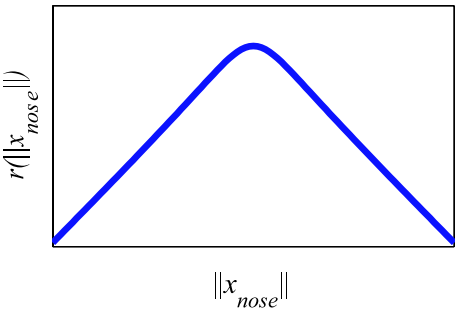
We present several instantiations of iLDP, of increasing complexity.

In Fig 1 we plot the solution to the simple problem given in Eqs (4). The iLDP method finds a solution very close to the global minimum obtained by discretizing the HJB equation. Note that we are using a substantial amount of noise, $\sigma = 1$.

In Fig 2 we show the solution to simple obstacle avoidance task, under deterministic and stochastic conditions. The position of a point mass in a frictionless plane evolves under linear Newtonian dynamics. The cost function (drawn in grayscale) is the distance to the origin with two circular obstacles of higher cost. When the dynamics are deterministic, two symmetric solutions are found for the initial condition $x(0) = (-5, 0)$, circumventing the obstacle on either side. When Gaussian noise is added to the system and the distribution of the state is propagated (dotted circle indicating 1 standard deviation), one trajectory veers away from the obstacle to account for the possibility of being forced onto it by the noise, while the other trajectory is unable to pass between the two obstacles which are effectively fused together by the noise.



(a) swimmer.



(b) reward.

Fig. 3. (a) A 5-link swimmer with a red "nose" and a green ring-shaped target. (b) The functional form of the state reward component. This form translates into a steady swimming gait at large distances with a smooth braking and stopping at the target.

In Fig 3 we depict a far more complex problem involving the control of a multilink "swimmer", a simulated chain of sticks in a viscous medium. The 14-dimensional state is composed of 5 angles, 5 angular velocities, 2 center-of-mass coordinates and their corresponding velocities. The controller applies 4 torques at the joints in order to maximize the reward

function[8]

$$r(x,u) = -c_x \frac{||x_{nose}||^2}{\sqrt{||x_{nose}||^2 + 1}} - c_u ||u||^2$$

While the dynamics involve the angles $\{\theta_k\}_{k=1..5}$ of the links WRT some external axis, the controller has access only to the joint angles $\{\theta_{k+1} - \theta_k\}_{k=1..4}$. Besides being biologically sensible, this transformation makes the policy rotationally invariant. By using the collocation method described in section III-A, this internalizing transformation can be trivially applied to the collocation cloud $\mathbf{x}^{(n)}$.

Further details of the solution are reported in [6] and a package allowing live user interaction with the controlled swimmers is available online[9].

IV. Discussion

Though local or trajectory-based dynamic programming techniques have been available to the control community for many years, they have only recently made inroads in the fields of biomechanics, reinforcement-learning, and other biologically-related motor control fields. While local methods are intrinsically robust to dimensionality limitations, other characteristics of biological systems such as extreme nonlinearity and the omnipresence of noise in the system (much of it due to noisy control signals), require that local methods be modified to enhance their applicability. This has been the goal of this paper. We've presented iLDP, a generalized local dynamic programming algorithm of which many specific instantiations are possible. The classic DDP and its sibling iLQG can be considered two such instantiations. iLDP allows for stochastic dynamics, general basis functions, and arbitrary transformations of the state into an internal coordinate set.

[8]The problem is phrased in (equivalent) terms of reward-maximization rather than cost-minimization for biological similitude.

[9]`http://alice.nc.huji.ac.il/~tassa`

References

[1] Jacobson, D & Mayne, D (1970) *Differential Dynamic Programming.* Elsevier: New York

[2] Kirk, D (1970) *Optimal Control Theory: An Introduction.* Prentice Hall: New Jersey

[3] Yong, J & Zhou, X (1999) *Stochastic Controls: Hamiltonian Systems and HJB Equations*. Springer: New York

[4] Julier, S; Uhlmann, J & Durrant-Whyte, H (2000) A new method for the nonlinear transformation of means and covariances in filters and estimators. *IEEE Trans Automatic Control* **45**: 447.

[5] Pantoja, J (1988) Differential dynamic programming and Newton's method. *Int J Control* **47**:1539

[6] Tassa, Y et al (2007) Receding horizon differential dynamic programming. In *NIPS* 2007.

[7] Todorov, E & Li, W (2004) A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems. In proceedings of the *American Control Conference*.

[8] Todorov, E & Jordan, M (2002) Optimal feedback control as a theory of motor coordination. *Nature Neuroscience* **5**: 1226

[9] Bagnell, J et al (2003) Policy search by dynamic programming. In *NIPS* 2003.

[10] Morimoto, J et al (2002) Minimax differential dynamic programming: An application to robust biped walking. In *NIPS* 2002.