A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems

Emanuel Todorov and Weiwei Li

Abstract— **This paper presents an iterative Linear-Quadratic-Gaussian (ILQG) method for nonlinear stochastic systems subject to control constraint. Such local iterative methods have only been applied to deterministic unconstrained problems in the past. We derive a linear control law by minimizing a novel quadratic approximation to the optimal cost-to-go function. The performance of the algorithm is illustrated in a limited-torque pendulum problem, as well as a complex biomechanical control problem involving an arm model with 10 state dimensions and 6 muscle actuators.**

I. Introduction

Optimal control theory has received a lot of attention in the last 50 years, and has found numerous applications in both science and engineering. Despite many theoretical and algorithmic advances, however, solving complex optimal control problems in practice remains a challenge [14].

When the control problem of interest does not fall in one of the few classes of analytically tractable problems, one has to resort to general-purpose approximation methods. Most existing approximation methods attempt to design a control law that is global (i.e. applicable to all states of the controlled plant), and are based on the Hamilton-Jacobi-Bellman equations and the idea of dynamic programming. For continous systems, the only numerical methods guaranteed to converge to the globally-optimal solution [13] involve discretizations of the state and control spaces, and run into the curse of dimensionaly. Generalizations to continuous high-dimensional spaces typically involve function approximations whose properties are not yet well understood [15].

An alternative to global optimization is provided by *local* methods, that only find sub-optimal solutions but do so efficiently without running into the curse of dimensionality. What these methods have in common is the idea of constructing a non-parametric discrete-time representation of the open-loop control sequence, and improving it iteratively by using only local information. Such methods are typically related to Pontryagin's Maximum Principle – which provides a necessary condition that optimal state-control trajectories for deterministic systems must satisfy. Trajectories consistent with the Maximum Principle can be found by solving a set of ODEs under boundary-value conditions, or by using gradient descent in the space of open-loop control sequences [2].

An ideal blend of the advantages of local and global methods is provided by Differential Dynamic Programming (DDP) [4]. This method is local, in the sense that it maintains a representation of a single trajectory and improves it locally, but the improvement itself is based on dynamic programming – within a "tube" around the current trajectory. DDP was recently applied to the hard problem of robot locomotion, with remarkable success [9]. DDP is known to have second-order convergence [10][11], and numerically appears to be more efficient [12] than (efficient implementations of) Netwon's method [8]. We have also compared the performance of DDP to other local methods, and found it to be generally superior [6].

Recently, we have developed a new method (iterative linear-quadratic regulator design, or ILQR) which is closely related to DDP but turns out to be significantly more efficient: by a factor of 10 on reasonably complex control problems [7]. We use iterative linearizations of the nonlinear dynamics around the current trajectory, adapt the well-developed linear-quadratic methodology to derive Riccati-like equtations, and then improve the trajectory. The increased efficiency seems to be due to the fact that, unlike DDP, we do not use second-order approximations to the systems dynamics; building such approximations is computationally expensive and potentially inaccurate, and (on our numerical tests) appears unnecessary.

The goal of the present paper is to continue to develop iterative linear-quadratic methods: we preserve the computational efficiency of our recent ILQR method, while avoiding a number of important limitations of existing methods.

A. What is new here

Our newILQG has the following advantages over existing methods based on local quadratic approximations (such as DDP and ILQR):

- Local methods are presently restricted to deterministic systems. Indeed, quadratic approximations to the optimal cost-to-go function are "blind" to additive noise and its potential influence on the optimal control law. However, in many problems of interest the noise is control-dependent (i.e. multiplicative in the control signal), and such noise is easily captured by quadartic approximations as we show below. We are particularly interested in multiplicative noise, because it is a

This work is supported by NIH Grant 1-R01-NS045915-01.

Emanuel Todorov is with the faculty of the Cognitive Science Department, University of California San Diego.

Weiwei Li is with the Deparment of Mechanical and Aerospace Engineering, University of California San Diego.

fundamental characteristics of neural control systems [3][5][6].

- Local methods are presently restricted to unconstrained problems. Generally speaking, constraints make the optimal cost-to-go function non-quadratic; but since we are approximating that function anyway we might as well take into account the effects of (control) constraints to the extent possible. We are very interested in efficient methods that handle control constraints because such constraints are always present in biomechanics (muscle actviations, which are the control signals being sent by the brain, are always non-negative and are also limited from above). Our new ILQG method does that – by modifying the linear feedback gain matrix whenever an element of the open-loop control sequence lies on the constraint boundary.
- Quadartic approximation methods are based on Riccati equations: define a quadratic optimization problem that the optimal controls satisfy at time step t, solve it analytically, and obtain a formula for the optimal cost-to-go function at time step t-1. Optimizing a quadratic is only possible when the Hessian is positive-definite. This is of course true in the classic LQG setting, but when LQG methods are used to approximate general nonlinear dynamics with non-quadartic costs, the Hessian can (and in practice does) have zero and even negative eigenvalues. The traditional remedy is to "fix" the Hessian, using a Levenberg-Marquardt method, or an adaptive shift scheme [12], or simply replace it with the identity matrix (which yields the steepest descent method). The problem is that after fixing the Hessian, the optimization at time step t is not performed exactly – in contrast to what the Riccati equations assume. Instead of making an ivalid assumption, our method uses the actual "fixed" Hessian to approximate the optimal cost-to-go function.

II. Definitions

Consider the nonlinear dynamical system described by the stochastic differential equation

$$d\mathbf{x} = \mathbf{f}(\mathbf{x},\mathbf{u})\,dt + F(\mathbf{x},\mathbf{u})\,d\boldsymbol{\omega}$$

with state $\mathbf{x} \in \mathrm{R}^n$, control $\mathbf{u} \in \mathrm{R}^m$, and standard Brownian motion noise $\boldsymbol{\omega} \in \mathrm{R}^p$. Let $\ell(t,\mathbf{x},\mathbf{u}) \geq 0$ be an instantaneous cost rate, $h(\mathbf{x}(T)) \geq 0$ a final cost, T a specified final time, and $\mathbf{u} = \boldsymbol{\pi}(t,\mathbf{x})$ a deterministic control law. Define the cost-to-go function $v^{\boldsymbol{\pi}}(t,\mathbf{x})$ as the total cost expected to accumulate if the system is initialized in state $\mathbf{x}$ at time t, and controlled until time T according to the control law $\boldsymbol{\pi}$:

$$v^{\boldsymbol{\pi}}(t,\mathbf{x}) \triangleq \mathrm{E}\left[h(\mathbf{x}(T)) + \int_t^T \ell(\tau,\mathbf{x}(\tau),\boldsymbol{\pi}(\tau,\mathbf{x}(\tau)))\,d\tau\right]$$

The expectation is taken over the instantiations of the stochastic process $\boldsymbol{\omega}$. The admissible control signals may be constrained: $\mathbf{u}(t) \in \mathcal{U}$. While the present formulation assumes full observability, the method developed below should be extendable to situations where the system state is only observable through delayed and noisy sensors.

Our objective is to control the system optimally, i.e. to find the control law $\boldsymbol{\pi}^*$ that minimizes $v^{\boldsymbol{\pi}}(0,\mathbf{x}_0)$. Note that the optimal control law $\boldsymbol{\pi}^*(t,\mathbf{x})$ is defined globally and does not depend on a specific initial state. The reason we emphasize the dependence on $\mathbf{x}_0$ is because here we will construct locally-optimal control laws: we will approximate $\boldsymbol{\pi}^*$ in the vicinity of the trajectory $\overline{\mathbf{x}}^*(t)$ that results from applying $\boldsymbol{\pi}^*$ to the deterministic system $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x},\mathbf{u})$. Since $\overline{\mathbf{x}}^*$ depends on $\mathbf{x}_0$, so does our approximation to $\boldsymbol{\pi}^*$.

The approximation will be constructed iteratively. Each iteration i will begin with an open-loop control sequence $\overline{\mathbf{u}}^{(i)}(t)$ and the corresponding "zero-noise" trajectory $\overline{\mathbf{x}}^{(i)}(t)$, obtained by applying $\overline{\mathbf{u}}^{(i)}(t)$ to the deterministic dynamics $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x},\mathbf{u})$ with $\overline{\mathbf{x}}^{(i)}(0) = \mathbf{x}_0$. By discretizing time, linearizing the system dynamics and quadratizing the cost functions around $\overline{\mathbf{x}}^{(i)}, \overline{\mathbf{u}}^{(i)}$, we will obtain a discrete-time linear dynamical system with quadratic cost. We will then adapt the well-developed methodology for solving linear-quadratic optimal control problems, and use it to design a control law $\boldsymbol{\pi}^{(i)}(t,\mathbf{x})$ which achieves better performance than $\overline{\mathbf{u}}^{(i)}(t)$ on the linearized system. Applying $\boldsymbol{\pi}^{(i)}$ while enforcing the control constraints will yield the pair $\overline{\mathbf{u}}^{(i+1)}$, $\overline{\mathbf{x}}^{(i+1)}$ for the next iteration. This results in a second-order method, which in practice converges very rapidly.

When we linearize the original system around $\overline{\mathbf{x}}, \overline{\mathbf{u}}$, the dynamics we obtain no longer describes the state and control variables. Instead it describes the state and control *deviations* $\delta\mathbf{x} \equiv \mathbf{x} - \overline{\mathbf{x}}$, $\delta\mathbf{u} \equiv \mathbf{u} - \overline{\mathbf{u}}$. Written in terms of these deviations, the modified Linear-Quadratic-Gaussian (LQG) approximation to our original optimal control problem becomes

$$\begin{aligned}
\delta\mathbf{x}_{k+1} &= A_k\delta\mathbf{x}_k + B_k\delta\mathbf{u}_k + \mathcal{C}_k(\delta\mathbf{u}_k)\,\boldsymbol{\xi}_k \qquad (1)\\
\mathcal{C}_k(\delta\mathbf{u}_k) &\triangleq [\mathbf{c}_{1,k} + C_{1,k}\delta\mathbf{u}_k \;\cdots\; \mathbf{c}_{p,k} + C_{p,k}\delta\mathbf{u}_k]\\
\mathrm{cost}_k &= q_k + \delta\mathbf{x}_k^\mathsf{T}\mathbf{q}_k + \frac{1}{2}\delta\mathbf{x}_k^\mathsf{T}Q_k\delta\mathbf{x}_k\\
&\quad + \delta\mathbf{u}_k^\mathsf{T}\mathbf{r}_k + \frac{1}{2}\delta\mathbf{u}_k^\mathsf{T}R_k\delta\mathbf{u}_k
\end{aligned}$$

where $\delta\mathbf{x}_1 = 0$, $\boldsymbol{\xi}_k \sim \mathrm{N}(0;\mathrm{I}_p)$, the last time step is K, and the final cost $q_K + \delta\mathbf{x}_K^\mathsf{T}\mathbf{q}_K + \frac{1}{2}\delta\mathbf{x}_K^\mathsf{T}Q_K\delta\mathbf{x}_K$ does not depend on the control signal (which is undefined at $k = K$). The quantities that define this modified LQG problem are $A_k, B_k, \mathbf{c}_{i,k}, C_{i,k}, q_k, \mathbf{q}_k, Q_k, \mathbf{r}_k, R_k, K$. We will specify later how these quantities are computed, given $\overline{\mathbf{x}}, \overline{\mathbf{u}}$ and the definition of the continuous-time problem.

The i^{th} column of the matrix $\mathcal{C}_k(\delta\mathbf{u}_k)$ is $\mathbf{c}_i + C_i\delta\mathbf{u}_k$. Thus the noise covariance is

$$\mathrm{Cov}\left[\mathcal{C}_k(\delta\mathbf{u}_k)\,\boldsymbol{\xi}_k\right] = \sum_{i=1}^{p}(\mathbf{c}_{i,k} + C_{i,k}\delta\mathbf{u}_k)(\mathbf{c}_{i,k} + C_{i,k}\delta\mathbf{u}_k)^\mathsf{T}$$

Note that we are using a simplified noise model, where $F(\mathbf{x},\mathbf{u})$ is only linearized with respect to $\delta\mathbf{u}$. This is sufficient to capture noise that is multiplicative in the control signal (which is what we are mainly interested in). Also, the approximation to the cost function does not include $\delta\mathbf{u}\delta\mathbf{x}$ cross terms – since most physically meaningful cost functions include separate state and control terms. It is straightforward (although tedious) to remove the above limitations and repeat the derivation that follows.

While the general methodology we will adapt to solve (1) is well-developed, there are several features of our problem that require special treatment: (i) the noise is control-dependent and the cost includes extra terms; (ii) since the LQG cost is only an approximation to the true cost, it does not have to be positive semi-definite; (iii) the unconstrained LQG problem is supposed to approximate a constrained nonlinear problem – and so the control constraints should be respected as much as possible even if that appears suboptimal from the LQG point of view. Because of (ii) and (iii), we will not always be able to find the optimal control law for (1) but only a control law which is better than the default $\delta\mathbf{u}_k = 0$. This involves two computations that run in parallel backwards in time: (i) computing the cost-to-go function given the control law in the future; (ii) computing the control law given the cost-to-go function now.

III. Computing the cost-to-go function

The control law we will design is linear, in the form $\delta\mathbf{u} = \boldsymbol{\pi}_k(\delta\mathbf{x}) = \mathbf{l}_k + L_k\delta\mathbf{x}$. This restriction is necessary in order to apply the LQG methodology. Suppose this control law is already defined for time steps $k \cdots K-1$. Then the cost-to-go function $v_k(\delta\mathbf{x})$ is also defined – as the cost expected to accumulate if system (1) is initialized in state $\delta\mathbf{x}$ at time step k, and controlled according to $\boldsymbol{\pi}$ for the remaining time steps.

We will show by induction (backwards in time) that if the control law is linear, the corresponding cost-to-go function remains in the quadratic form

$$v_k(\delta\mathbf{x}) = s_k + \delta\mathbf{x}^\mathsf{T}\mathbf{s}_k + \tfrac{1}{2}\delta\mathbf{x}^\mathsf{T}S_k\delta\mathbf{x}$$

for all k. At the last time step this holds, because the final cost is a quadratic that does not depend on the control signal. Now suppose that v is in the above form for time steps $k+1 \cdots K$. Using the shortcut $\boldsymbol{\pi}$ in place of the control signal $\mathbf{l}_k + L_k\delta\mathbf{x}$ that our control law generates, the Bellman equation for the cost-to-go function is

$$\begin{aligned} v_k(\delta\mathbf{x}) &= \text{immediate cost } + \mathrm{E}\left[v_{k+1}(\text{next state})\right] \\ &= q_k + \delta\mathbf{x}^\mathsf{T}\left(\mathbf{q}_k + \tfrac{1}{2}Q_k\delta\mathbf{x}\right) + \boldsymbol{\pi}^\mathsf{T}\left(\mathbf{r}_k + \tfrac{1}{2}R_k\boldsymbol{\pi}\right) \\ &\quad +\mathrm{E}\left[v_{k+1}\left(A_k\delta\mathbf{x} + B_k\boldsymbol{\pi} + \text{noise}_k\right)\right] \end{aligned}$$

Evaluating the expectation term $\mathrm{E}[\cdot]$ above yields

$$\begin{aligned} & s_{k+1} + \left(A_k\delta\mathbf{x} + B_k\boldsymbol{\pi}\right)^\mathsf{T}\mathbf{s}_{k+1} + \\ & \tfrac{1}{2}\left(A_k\delta\mathbf{x} + B_k\boldsymbol{\pi}\right)^\mathsf{T}S_{k+1}\left(A_k\delta\mathbf{x} + B_k\boldsymbol{\pi}\right) + \\ & \tfrac{1}{2}\ \mathrm{trace}\left(\textstyle\sum_{i=1}^{p}\left(\mathbf{c}_{i,k} + C_{i,k}\boldsymbol{\pi}\right)\left(\mathbf{c}_{i,k} + C_{i,k}\boldsymbol{\pi}\right)^\mathsf{T}S_{k+1}\right) \end{aligned}$$

Using the fact that $\mathrm{trace}(UV) = \mathrm{trace}(VU)$, the $\frac{1}{2}\ \mathrm{trace}(\cdot)$ term above becomes

$$\begin{aligned} & \tfrac{1}{2}\boldsymbol{\pi}^\mathsf{T}\left(\textstyle\sum_i C_{i,k}^\mathsf{T}S_{k+1}C_{i,k}\right)\boldsymbol{\pi} + \\ & \boldsymbol{\pi}^\mathsf{T}\left(\textstyle\sum_i C_{i,k}^\mathsf{T}S_{k+1}\mathbf{c}_{i,k}\right) + \tfrac{1}{2}\left(\textstyle\sum_i \mathbf{c}_{i,k}^\mathsf{T}S_{k+1}\mathbf{c}_{i,k}\right) \end{aligned}$$

Combining the results and grouping terms, the cost-to-go function is

$$\begin{aligned} v_k(\delta\mathbf{x}) &= q_k + s_{k+1} + \tfrac{1}{2}\textstyle\sum_i \mathbf{c}_i^\mathsf{T}S_{k+1}\mathbf{c}_i \\ &\quad +\delta\mathbf{x}^\mathsf{T}\left(\mathbf{q}_k + A_k^\mathsf{T}\mathbf{s}_{k+1}\right) \\ &\quad +\tfrac{1}{2}\delta\mathbf{x}^\mathsf{T}\left(Q_k + A_k^\mathsf{T}S_{k+1}A_k\right)\delta\mathbf{x} \\ &\quad +\boldsymbol{\pi}^\mathsf{T}\left(\mathbf{g} + G\delta\mathbf{x}\right) + \tfrac{1}{2}\boldsymbol{\pi}^\mathsf{T}H\boldsymbol{\pi} \end{aligned} \tag{2}$$

The shortcuts $\mathbf{g}, G, H$ appearing on the last line of (2) are defined at each time step as

$$\begin{aligned} \mathbf{g} &\triangleq \mathbf{r}_k + B_k^\mathsf{T}\mathbf{s}_{k+1} + \textstyle\sum_i C_{i,k}^\mathsf{T}S_{k+1}\mathbf{c}_{i,k} \\ G &\triangleq B_k^\mathsf{T}S_{k+1}A_k \\ H &\triangleq R_k + B_k^\mathsf{T}S_{k+1}B_k + \textstyle\sum_i C_{i,k}^\mathsf{T}S_{k+1}C_{i,k} \end{aligned} \tag{3}$$

At this point one may notice that the expression for $v_k(\delta\mathbf{x})$ is a quadratic function of $\boldsymbol{\pi}$, and set the control signal to the value of $\boldsymbol{\pi}$ which makes the gradient vanish: $\delta\mathbf{u}_k = -H^{-1}\mathbf{g}_k - H^{-1}G\delta\mathbf{x}$. But we will not assume this specific form of the control law here, because H may have negative eigenvalues (in which case the above $\delta\mathbf{u}$ is not a minimum), and also because some control constraints may be violated. Instead we will defer the computation of the control law to the next section. All we assume for now is that the control law computed later will be in the general form $\delta\mathbf{u} = \mathbf{l}_k + L_k\delta\mathbf{x}$. With this assumption we can complete the computation of the cost-to-go function. Replacing $\boldsymbol{\pi}$ with $\mathbf{l}_k + L_k\delta\mathbf{x}$, and noting that the square matrix S_k is symmetric, the $\boldsymbol{\pi}$-dependent expression in the second line of (2) becomes

$$\begin{aligned} & \mathbf{l}_k^\mathsf{T}\mathbf{g} + \tfrac{1}{2}\mathbf{l}_k^\mathsf{T}H\mathbf{l}_k + \delta\mathbf{x}^\mathsf{T}\left(G^\mathsf{T}\mathbf{l}_k + L_k^\mathsf{T}\mathbf{g} + L_k^\mathsf{T}H\mathbf{l}_k\right) \\ & +\tfrac{1}{2}\delta\mathbf{x}^\mathsf{T}\left(L_k^\mathsf{T}HL_k + L_k^\mathsf{T}G + G^\mathsf{T}L_k\right)\delta\mathbf{x} \end{aligned}$$

We now see that the cost-to-go function remains quadratic in $\delta\mathbf{x}$, which completes the induction proof. The parameters of the quadratic form $v_k(\delta\mathbf{x})$ are initialized with $S_K = Q_K,\ \mathbf{s}_K = \mathbf{q}_K,\ s_K = q_K$. Recalling definition (3), the backward recursion for $S, \mathbf{s}, s$ is

$$\begin{aligned} S_k &= Q_k + A_k^\mathsf{T}S_{k+1}A_k + L_k^\mathsf{T}HL_k + L_k^\mathsf{T}G + G^\mathsf{T}L_k \\ \mathbf{s}_k &= \mathbf{q}_k + A_k^\mathsf{T}\mathbf{s}_{k+1} + L_k^\mathsf{T}H\mathbf{l}_k + L_k^\mathsf{T}\mathbf{g} + G^\mathsf{T}\mathbf{l}_k \\ s_k &= q_k + s_{k+1} + \tfrac{1}{2}\textstyle\sum_i \mathbf{c}_{i,k}^\mathsf{T}S_{k+1}\mathbf{c}_{i,k} + \tfrac{1}{2}\mathbf{l}_k^\mathsf{T}H\mathbf{l}_k + \mathbf{l}_k^\mathsf{T}\mathbf{g} \end{aligned} \tag{4}$$

Since $\delta\mathbf{x}_1 = 0$, the total expected cost resulting from the control law $(\mathbf{l}, L)$ is just s_1.

Note that if we are not concerned with negative eigenvalues of H or violations of control constraints, and set $\mathbf{l}_k = -H^{-1}\mathbf{g},\ L_k = -H^{-1}G$ as mentioned above, a number of terms cancel and (4) reduces to

$$\begin{aligned} S_k &= Q_k + A_k^{\mathsf{T}} S_{k+1} A_k - G^{\mathsf{T}} H^{-1} G \\ \mathbf{s}_k &= \mathbf{q}_k + A_k^{\mathsf{T}} \mathbf{s}_{k+1} - G^{\mathsf{T}} H^{-1} \mathbf{g} \\ s_k &= q_k + s_{k+1} + \tfrac{1}{2} \textstyle\sum_i \mathbf{c}_{i,k}^{\mathsf{T}} S_{k+1} \mathbf{c}_{i,k} - \tfrac{1}{2} \mathbf{g}^{\mathsf{T}} H^{-1} \mathbf{g} \end{aligned}$$

If we further remove the control-dependent noise (by setting $C_{i,k} = 0$) and the linear terms in the cost function (by setting $\mathbf{q}_k = \mathbf{r}_k = 0$), we see that $\mathbf{g} = \mathbf{l}_k = \mathbf{s}_k = 0$ and the first line of (4) reduces to the familiar LQR discrete-time Riccati equation

$$\begin{aligned} S_k = \; & Q_k + A_k^{\mathsf{T}} S_{k+1} A_k \\ & - A_k^{\mathsf{T}} S_{k+1} B_k \left(R_k + B_k^{\mathsf{T}} S_{k+1} B_k \right)^{-1} B_k^{\mathsf{T}} S_{k+1} A_k \end{aligned}$$

Thus our method provides a generalization: it can be reduced to the familiar methods, but has the added flexibility of keeping the quadratic cost-to-go calculation consistent regardless of how the linear feedback control law is computed.

IV. Computing the control law

As we saw in (2), the cost-to-go function $v_k(\delta\mathbf{x})$ depends on the control $\delta\mathbf{u}_k = \boldsymbol{\pi}_k(\delta\mathbf{x})$ through the term

$$a(\delta\mathbf{u},\delta\mathbf{x}) = \delta\mathbf{u}^{\mathsf{T}}(\mathbf{g} + G\delta\mathbf{x}) + \tfrac{1}{2}\delta\mathbf{u}^{\mathsf{T}} H \delta\mathbf{u} \qquad (5)$$

where we have suppressed the time index k. Ideally we would choose the $\delta\mathbf{u}$ that minimizes a for every $\delta\mathbf{x}$, subject to whatever control constraints are present. However, this is not always possible within the family of linear control laws $\delta\mathbf{u} = \mathbf{l} + L\delta\mathbf{x}$ that we are considering. Since the goal of the LQG stage is to approximate the optimal controller for the nonlinear system in the vicinity of $\overline{\mathbf{x}}$, we will give preference to linear control laws that are optimal/feasible for small $\delta\mathbf{x}$, even if that (unavoidably) makes them sub-optimal/infeasible for larger $\delta\mathbf{x}$. In particular we need to make sure that for $\delta\mathbf{x} = 0$, the new open-loop control $\delta\mathbf{u} = \mathbf{l}$ performs no worse than the current open-loop control $\delta\mathbf{u} = 0$. Since $a(0,\delta\mathbf{x}) = 0$, this holds if $a(\mathbf{l},0) = \mathbf{l}^{\mathsf{T}}\mathbf{g} + \frac{1}{2}\mathbf{l}^{\mathsf{T}} H \mathbf{l} \leq 0$. The latter is always achievable by setting $\mathbf{l} = -\epsilon\mathbf{g}$ for a small enough $\epsilon \geq 0$.

A. First-order methods

The gradient $\nabla_{\delta\mathbf{u}} a(\delta\mathbf{u},\delta\mathbf{x})$ evaluated at $\delta\mathbf{u} = 0$ is $\mathbf{g} + G\delta\mathbf{x}$. Therefore, we can make an improvement along the gradient of a by setting $\delta\mathbf{u} = -\epsilon(\mathbf{g} + G\delta\mathbf{x})$. If we are only interested in open-loop control, we can use $\delta\mathbf{u} = -\epsilon\mathbf{g}$. If the new control signal (for the nonlinear system) $\overline{\mathbf{u}} - \epsilon\mathbf{g}$ violates the constraints, we have to reduce ϵ until the constraints are satisfied. Note that $\overline{\mathbf{u}}$ is by definition a feasible control signal, so unless it lies on the constraint boundary (and $\mathbf{g}$ points inside the feasible set) we can find an $\epsilon > 0$ for which $\overline{\mathbf{u}} - \epsilon\mathbf{g}$ is also feasible.

This method is related to policy gradient, but is more efficient – because the improvement at time k takes into account improvements at future times. If we wish to obtain a pure policy gradient method with respect to the open-loop control sequence, we have to: (i) compute the cost-to-go for the control law $\delta\mathbf{u}_1 = \cdots = \delta\mathbf{u}_{K-1} = 0$; (ii) make the improvement $\delta\mathbf{u}_k = -\epsilon\mathbf{g}_k$ for all k in a separate pass.

B. Second-order methods

If the symmetric matrix H in (5) is positive semi-definite[1] we can compute the unconstrained optimal control law $\delta\mathbf{u} = -H^{-1}(\mathbf{g} + G\delta\mathbf{x})$, and deal with the control constraints as described below. But when H has negative eigenvalues, there exist $\delta\mathbf{u}$'s that make a (and therefore v) arbitrarily negative. Note that the cost-to-go function for the nonlinear problem is always non-negative, but since we are using an approximation to the true cost we may (and in practice do) encounter situations where a does not have a minimum. In that case the gradient $\nabla_{\delta\mathbf{u}} a = \mathbf{g} + G\delta\mathbf{x}$ is still correct, and so the true cost-to-go decreases in the direction $-\mathcal{H}^{-1}(\mathbf{g} + G\delta\mathbf{x})$ for any positive definite matrix $\mathcal{H}$. We want $\mathcal{H}$ to resemble H, because H still contains correct second-order information.

One possibility is to set $\mathcal{H} = (H + (\epsilon - \lambda_{\min}(H))\,\mathrm{I})$ where $\lambda_{\min}(H)$ is the minimum eigenvalue of H and $\epsilon > 0$. This is related to the Levenberg-Marquardt trick, and has the potentially undesirable effect of increasing all eigenvalues of H and not just those that are negative. Another possibility is to compute the eigenvalue decomposition $[V,D] = \mathrm{eig}(H)$, replace all elements of the diagonal matrix D that are smaller than ϵ with ϵ (obtaining a new diagonal matrix $\mathcal{D}$), and then set $\mathcal{H} = V\mathcal{D}V^{\mathsf{T}}$. The eigenvalue decomposition is not a significant slowdown, because we have to perform a matrix inversion anyway and we can do so by $\mathcal{H}^{-1} = V\mathcal{D}^{-1}V^{\mathsf{T}}$. It is not yet clear which of the two methods works better in practice. Note that we may also want to use $\mathcal{H}$ instead of H when the eigenvalues are positive but very small – because in that case H^{-1} can cause very large $\delta\mathbf{u}$'s that will push the original system outside the range of validity of our LQG approximation.

C. Constrained second-order methods

The problem here is to find the linear control law $\delta\mathbf{u} = \mathbf{l} + L\delta\mathbf{x}$ minimizing (5) subject to constraints $\delta\mathbf{u} + \overline{\mathbf{u}} \in \mathcal{U}$, assuming that H has already been replaced with a positive definite $\mathcal{H}$ (see above). Given that $\delta\mathbf{x}$ is unconstrained, the only general way to enforce the constraints $\mathcal{U}$ is to set $L = 0$. In practice we do not want to be that conservative, since we are looking for an approximation to the nonlinear problem that is valid around $\delta\mathbf{x} = 0$. Either way we can ignore the $L\delta\mathbf{x}$ term in the constraint satisfaction phase, and come back to the computation of L after the open-loop component $\mathbf{l}$ has been determined.

The unconstrained minimum of $\delta\mathbf{u}^{\mathsf{T}}\mathbf{g} + \frac{1}{2}\delta\mathbf{u}^{\mathsf{T}}\mathcal{H}\delta\mathbf{u}$ is $\delta\mathbf{u}^* = -\mathcal{H}^{-1}\mathbf{g}$. If it satisfies $\delta\mathbf{u}^* + \overline{\mathbf{u}} \in \mathcal{U}$ we are done. Otherwise we have two options. The more efficient but less accurate method is to backtrack once, i.e. to find the maximal $\epsilon \in [0;1]$ such that $\epsilon\delta\mathbf{u}^* + \overline{\mathbf{u}} \in \mathcal{U}$. This is

[1]Whenever H is singular, the notation H^{-1} will denote the Moore-Penrose pseudoinverse.

appropriate in the early phase of the iterative algorithm when $\overline{\mathbf{x}}$ is still far away from $\overline{\mathbf{x}}^*$; in that phase it makes more sense to quickly improve the control law rather than refine the solution to an LQG problem that is an inaccurate approximation to the original problem. But in the final phase of the iterative algorithm we want to obtain the best control law possible for the given LQG problem. In that phase we use quadratic programming. When the constraint set is specified by a collection of linear inequalities, and given that $\mathcal{H}$ is positive definite, the active set algorithm (which is a greedy quadratic programming method) can be used to quickly find the global constrained minimum.

Once the open-loop component $\mathbf{l}$ is determined, we have to compute the feedback gain matrix L. If $\mathbf{l}+\overline{\mathbf{u}}$ is inside $\mathcal{U}$, small changes $L\delta\mathbf{x}$ will not cause constraint violations and so we can use the optimal $L = -\mathcal{H}^{-1}G$. But if $\mathbf{l}+\overline{\mathbf{u}}$ lies on the constraint boundary $\partial\mathcal{U}$, we have to modify L so that $L\delta\mathbf{x}$ can only cause changes along the boundary. This is not only because we want to avoid constraint violations. The fact that $\mathbf{l}+\overline{\mathbf{u}}$ is on $\partial\mathcal{U}$ means that the unconstrained minimum $\delta\mathbf{u}^*$ is actually outside $\mathcal{U}$, and so a change $L\delta\mathbf{x}$ orthogonal to the boundary $\partial\mathcal{U}$ cannot produce a better feasible control. Modifying L is straightforward in the typical case when the range of each element of $\mathbf{u}$ is specified independently. In that case we simply set to 0 the rows of $-\mathcal{H}^{-1}G$ corresponding to elements of $\mathbf{l}+\overline{\mathbf{u}}$ that have reached their limits.

V. Summary of the algorithm

Choose a time discretization $k = 1 \cdots K$, with step size $\Delta t = T/(K-1)$ and $t = (k-1)\Delta t$. Initialize the open-loop controls $\overline{\mathbf{u}}_1 \cdots \overline{\mathbf{u}}_{K-1}$ to 0, or use problem-dependent initialization when available. Apply the following iterative algorithm until convergence:

1) Apply $\overline{\mathbf{u}}$ to the deterministic nonlinear dynamics $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x},\mathbf{u})$ to obtain the corresponding zero-noise trajectory $\overline{\mathbf{x}}$. This could be done by Euler integration

$$\overline{\mathbf{x}}_{k+1} = \overline{\mathbf{x}}_k + \Delta t\ \mathbf{f}(\overline{\mathbf{x}}_k, \overline{\mathbf{u}}_k)$$

or by defining a continuous $\overline{\mathbf{u}}(t)$ via interpolation, applying a continuous-time integrator such as Runge-Kutta, and discretizing the resulting $\overline{\mathbf{x}}(t)$.

2) At each $(\overline{\mathbf{x}}_k, \overline{\mathbf{u}}_k)$ compute the linearized dynamics and quadratized costs

$$\begin{aligned} A_k &= \mathrm{I} + \Delta t\ \partial\mathbf{f}/\partial\mathbf{x}; \quad B_k = \Delta t\ \partial\mathbf{f}/\partial\mathbf{u}; \\ \mathbf{c}_{i,k} &= \sqrt{\Delta t}\ F^{[i]}; \quad C_{i,k} = \sqrt{\Delta t}\ \partial F^{[i]}/\partial\mathbf{u} \\ q_k &= \Delta t\ \ell; \quad \mathbf{q}_k = \Delta t\ \partial\ell/\partial\mathbf{x}; \quad \mathbf{r}_k = \Delta t\ \partial\ell/\partial\mathbf{u}; \\ Q_k &= \Delta t\ \partial^2\ell/\partial\mathbf{x}\partial\mathbf{x}; \quad R_k = \Delta t\ \partial^2\ell/\partial\mathbf{u}\partial\mathbf{u} \end{aligned}$$

where $F^{[i]}$ denotes the i^{th} column of the matrix F. At the final time step $k = K$, $\mathbf{r}_K$ and R_K are undefined and $q_K = h$; $\mathbf{q}_K = \partial h/\partial\mathbf{x}$; $Q_K = \partial^2 h/\partial\mathbf{x}\partial\mathbf{x}$. The $\sqrt{\Delta t}$ term appears because the covariance of Brownian motion grows linearly with time. Defining the deviations $\delta\mathbf{x}_k \equiv \mathbf{x}_k - \overline{\mathbf{x}}_k$ and $\delta\mathbf{u}_k \equiv \mathbf{u}_k - \overline{\mathbf{u}}_k$, the nonlinear system is approximated as given by (1).

3) In a backward pass through time, compute a linear control law $\delta\mathbf{u}_k = \mathbf{l}_k + L_k\delta\mathbf{x}_k$ and the corresponding cost-to-go function $v_k(\delta\mathbf{x}) = s_k + \delta\mathbf{x}^{\mathsf{T}}\mathbf{s}_k + \frac{1}{2}\delta\mathbf{x}^{\mathsf{T}}S_k\delta\mathbf{x}$. The cost-to-go parameters are initialized as $s_K = q_K$, $S_K = Q_K$, $\mathbf{s}_K = \mathbf{q}_K$, and computed recursively according to (4) and (3). At each step, replace H with a positive-definite $\mathcal{H}$ using one of the methods described above, and set

$$\mathbf{l}_k = -\mathcal{H}^{-1}\mathbf{g}; \quad L_k = -\mathcal{H}^{-1}G$$

If $\mathbf{l}_k + \overline{\mathbf{u}}_k$ violates the control constraints $\mathcal{U}$, modify $\mathbf{l}_k$ and L_k as described above.

4) Apply the control law $\delta\mathbf{u}_k = \mathbf{l}_k + L_k\delta\mathbf{x}_k$ to the deterministic system $\delta\mathbf{x}_{k+1} = A_k\delta\mathbf{x}_k + B_k\delta\mathbf{u}_k$ in a forward pass, starting from $\delta\mathbf{x}_1 = 0$ and computing along the way the new open-loop controls $\widetilde{\mathbf{u}}_k = \overline{\mathbf{u}}_k + \delta\mathbf{u}_k$. If necessary, enforce $\widetilde{\mathbf{u}}_k \in \mathcal{U}$. If the sequences $\overline{\mathbf{u}}$ and $\widetilde{\mathbf{u}}$ are sufficiently close, end the iteration. Otherwise set $\overline{\mathbf{u}} = \widetilde{\mathbf{u}}$ and go to step 1. To implement line search, use $\delta\mathbf{u}_k = \alpha\mathbf{l}_k + L_k\delta\mathbf{x}_k$ in the forward pass, where α is the line search parameter. Convergence is guaranteed if we use backtracking linesearch: start with $\alpha = 1$, and decrease it by a factor of $n_\alpha < 1$ until the expected cost of the new open-loop control law becomes smaller than the old one.

VI. Optimal control problems to be studied

We have thus far tested the method on two problems, both of which have nonlinear dynamics, non-quadratic costs, control constraints, and (for one of the problems only) multiplicative noise.

A. An inverted pendulum

We use the popular inverted pendulum problem, with a limit on the torque that the motor can generate, and also a quadratic cost on the torque. Thus the optimal solutions are not always in the form of bang-bang control (which is the case when the control cost is absent) but exhibit both torque saturation and continuous transitions between torque limits. The dynamics are

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= u - 4\sin x_1 \end{aligned}$$

where the state variables are $x_1 = \theta$, $x_2 = \dot{\theta}$. The goal is to make the pendulum swing up (corresponding to a 180 deg angle) and also make it stop – at the final time T. The control objective is to find the control $u(t)$ that minimizes the performance index

$$J_0 = (1 + \cos x_1(T))^2 + 0.1\ x_2(T)^2 + 0.01\int_0^T u(t)^2\, dt$$

We use a time step of 10 msec, $T = 4$sec, and the maximum control / torque that can be generated is $|u| \leq 2$.

B. A model of the human arm

The second model we study is rather complex – see [7] for details. We model the nonlinear dynamics of a 2-link arm moving in the horizontal plane, with shoulder and elbow joints (Fig 1).The inverse dynamics is

$$\mathcal{M}(\theta)\ddot{\theta} + \mathcal{C}(\theta, \dot{\theta}) + \mathcal{B}\dot{\theta} = \tau,$$

where $\theta \in R^2$ is the joint angle vector (shoulder: θ_1, elbow: θ_2), $\mathcal{M}(\theta) \in R^{2\times2}$ is a positive definite symmetric inertia matrix, $\mathcal{C}(\theta, \dot{\theta}) \in R^2$ is a vector centripetal and Coriolis forces, $\mathcal{B} \in R^{2\times2}$ is the joint friction matrix, and $\tau \in R^2$ is the joint torque that the muscles generate. We have

$$\begin{aligned}
\mathcal{M} &= \begin{pmatrix} a_1 + 2a_2 cos\theta_2 & a_3 + a_2 cos\theta_2 \\ a_3 + a_2 cos\theta_2 & a_3 \end{pmatrix} \\
\mathcal{C} &= \begin{pmatrix} -\dot{\theta_2}(2\dot{\theta_1} + \dot{\theta_2}) \\ \dot{\theta_1}^2 \end{pmatrix} a_2 sin\theta_2 \\
\mathcal{B} &= \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} \\
a_1 &= I_1 + I_2 + m_2 l_1^2 \\
a_2 &= m_2 l_1 s_2 \\
a_3 &= I_2
\end{aligned}$$

where $b_{11} = b_{22} = 0.05, b_{12} = b_{21} = 0.025$, m_i is the mass (1.4kg, 1kg), l_i is the length of link i (30cm, 33cm), s_i is the distance from the joint center to the center of the mass for link i (11cm, 16cm), and I_i is the moment of inertia $(0.025kgm^2, 0.045kgm^2)$.

We can compute the forward dynamics as

$$\ddot{\theta} = \mathcal{M}(\theta)^{-1}(\tau - \mathcal{C}(\theta, \dot{\theta}) - \mathcal{B}\dot{\theta})$$

and write the system in state space form

$$\dot{x} = F(x) + G(x)u$$

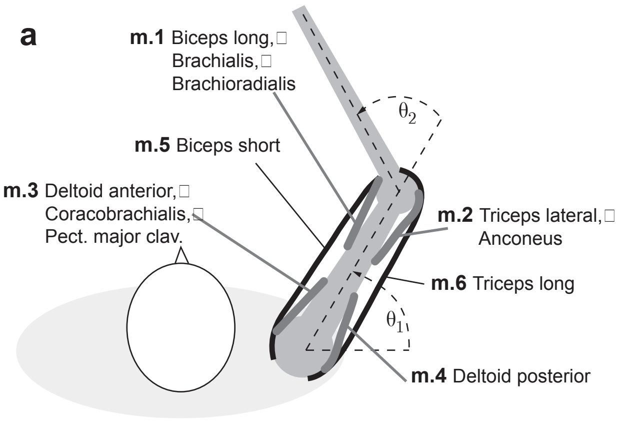
where the state variables and joint torque are given by

$$x = (\theta_1\ \theta_2\ \dot{\theta}_1\ \dot{\theta}_2)^T, \qquad \tau = (\tau_1\ \tau_2)^T.$$

There are a large number of muscles that act on the arm in the horizontal plane (see Fig 1A). But since we have only 2 degrees of freedom, these muscles can be organized into 6 actuator groups: elbow flexors (1), elbow extensors (2), shoulder flexors (3), shoulder extensors (4), biarticulate flexors (5), and biarticulate extensors (6). The joint torques produced by a muscle are a function of its moment arms, length-velocity-tension curve (illustrated in Fig 1B), and activation dynamics.

Moment arms are roughly constant for extensor muscles, but vary considerably with joint angle for flexor muscles. For each flexor group, this variation is modelled with a function of the form $a+b\ cos(c\ \theta)$, where the constants have been adjusted to match experimental data.We will denote the 2 by 6 matrix of muscle moment arms with $M(\theta)$.

The tension $T\,(l, v, a)$ produced by a muscle obviously depends on the muscle activation a, but also varies substantially with the length l and velocity v of that muscle.



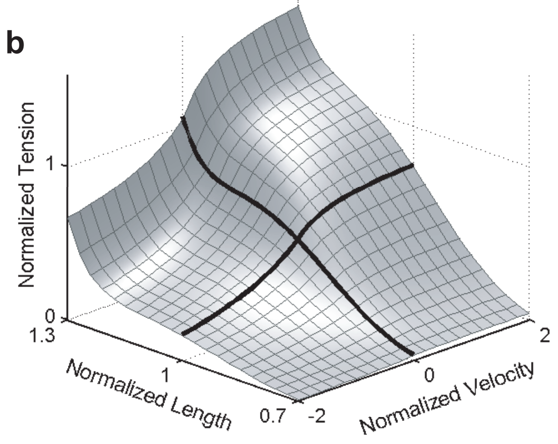


Fig. 1. A) A schematic illustration of the joint coordinate system and the muscle groups acting on the human arm in the horizontal plane. The felxors have variable moment arms, while extensor moment arms are roughly constant. B) The length-velocity-tension function for mammalian muscles (at maximal activation).

Our model (see Fig 1B) is based on the publicly available Virtual Muscle model [1]. The model parameters can be found in [7].

Muscle activation a is not equal to instantaneous neural input u, but is generated by passing u through a filter that describes calcium dynamics. This is reasonably well modelled with a first order nonlinear filter of the form $\dot{a} = (u - a)/t(u, a)$, where $t = t_{deact} + u(t_{act} - t_{deact})$ when $u > a$, and $t = t_{deact}$ otherwise. The input-dependent activation dynamics $t_{act} = 30msec$ is faster than the constant deactivation dynamics $t_{deact} = 60msec$.

To summarize, adding muscles to the dynamical system results in 6 new state variables, with dynamics

$$\dot{a} = (u - a)/t(u, a)$$

The joint torque vector generated by the muscles is given by

$$\tau = M(\theta)\ T(a, l(\theta), v(\theta, \dot{\theta}))$$

The task we study is a reaching task, where the arm has to start at some initial position and move to a target in a

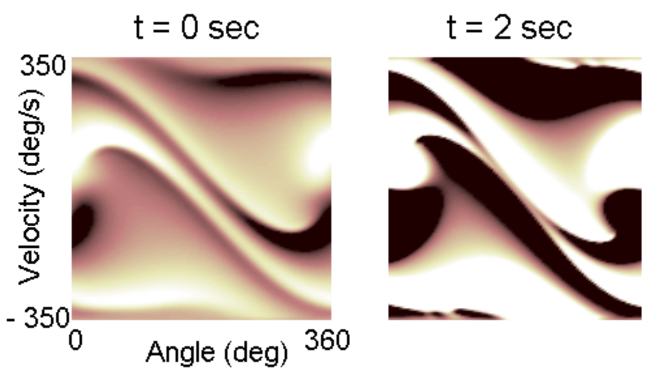


Fig. 2. Pixel intensity corresponds to the value of the optimal control signal at each state (shown for two points in time). White is $u = 2$, black is $u = -2$. The solution is obtained by discertizing the HJB equation.

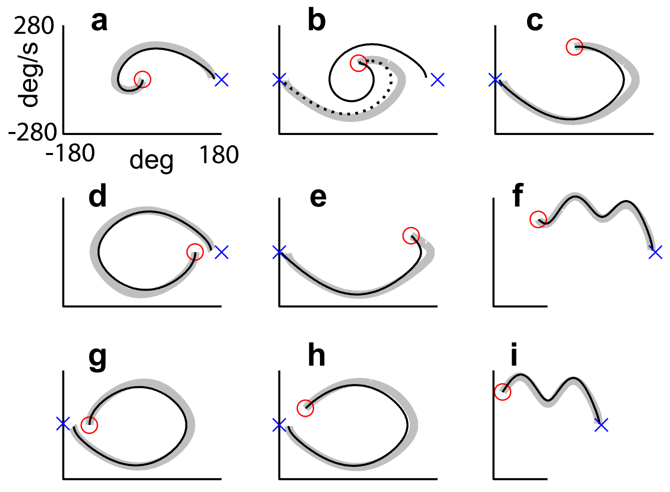


Fig. 3. Gray: optimal HJB trajectories. Black: optimal ILQG trajectories. The cross marks the target (note that the horizontal axis is circular). The different plots correspond to different initial conditions. The dotted line in **b** is a better local minimum found by ILQG with additional restarts. In **f** and **i** the pendulum passes through the upright target position multiple times. The slight undershoot is due to the control penalty.

specified time interval. It also has to stop at the target, and do all that with minimal energy consumption. There are good reasons to believe that such costs are indeed relevant to the neural control of movement [5]. The cost function is defined as

$$J_0 = \left\|e\left(\theta(T)\right) - e^*\right\|^2 + 0.001 \left\|\dot{e}\left(\theta(T), \dot{\theta}(T)\right)\right\|^2 + \frac{1}{2}\int_0^T r u^T u \, dt$$

where $r = 0.0001$, and $e(\theta)$ is the transformation from joint coordinates to end-point coordinates (where the target e^* is defined).

VII. Numerical results

Since the pendulum has a two-dimensional state space, we can discretize it with a dense grid and solve the time-varying Hamilton-Jacobi-Bellman PDE numerically. We used a 100x100 grid, and 400 time step (4 sec interval, 10 msec time step). The PDE solution for the optimal control is shown in Fig 2, at two points in time. The backup is based on the discrete-time version of the HJB equation:

$$V(t - \Delta, \mathbf{x}) = V(t, \mathbf{x}) + \Delta \min_{\mathbf{u}} \left\{ \ell(\mathbf{x}, \mathbf{u}) + \mathbf{f}(\mathbf{x}, \mathbf{u})^T V_{\mathbf{x}}(t, \mathbf{x}) \right\}$$

where the minimization w.r.t. u can be performed analytically since the control cost is quadratic and the dynamics is linear in u. Constrained optimization of u is easily performed since u is scalar.

Figure 3 shows the optimal trajectories of the pendulum, according to the HJB solution, in gray. The best solutions found by our method with 3 restarts (with constant intial control sequences $u(t) = 0, 1, -1$) are shown in black. Note the close correspondence. In the only case where we saw a mistatch, running the algorithm with additional initial coniditions found a better local minimum (dotted line) that agrees with the HJB solution.

Surprisingly, the cost obtained by the ILQG method was actually better than that of the HJB method (Fig 4). This

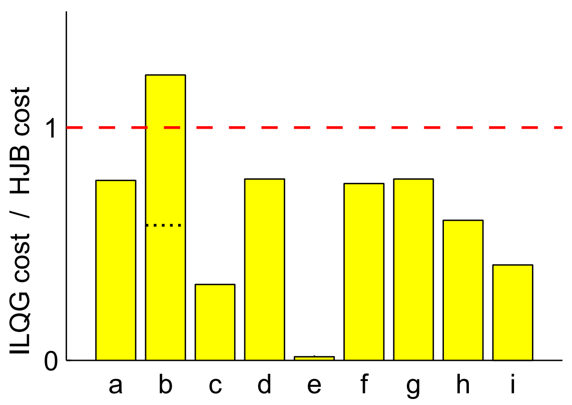


Fig. 4. Cost ratio of the ILQG and HJB methods, for each initial condition. The dotted line corresponds to the better local minimum we found using additional restarts of ILQG.

is theoretically impossible, but occurs in practice because PDE solutions can be very unstable numerically. In fact, we had to use smoothing at each time step, with a Gaussian kernel of 1 grid point, to obtain a solution at all. We also attempted to solve numerically the HJB equation for a stochastic pendulum – which involves an extra term dependent on the Hessian of the optimal cost-to-go. In that case, however, the amount of smoothing needed to obtain solutions was so much that the resulting control laws were meaningless. It is possible that Markov Chain approximation methods [13] would yield a more accurate solution than direct discretization of the HJB PDE.

To test for the presence of multiple local minima, we initialized the ILQG algorithm with 20 random control sequences – which generated the initial trajectories shown in Fig 5. The circle marks the initial state. Note that the

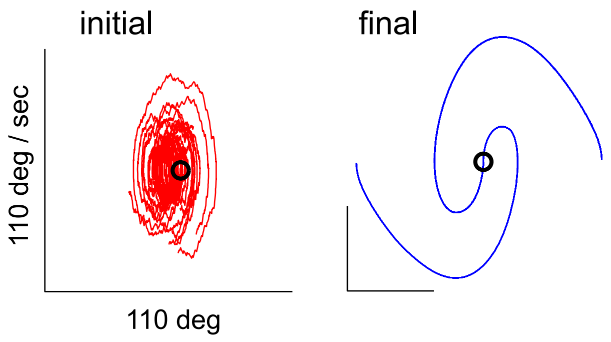


Fig. 5. Left: initial trajectories correspoding to random control sequences. Right: same trajectories after ILQG optimization. All solutions converge to one of two near-symmetric trajectories.

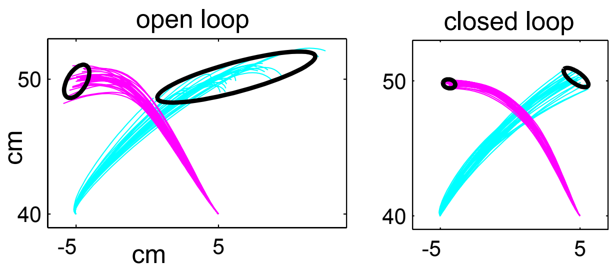


Fig. 6. End-point trajectories for the stochastic arm model, under open-loop and closed-loop ILQG control.

ILQG method always converges to one of two minima, which are essentially symmetric (although one is a slightly better solution than the other). It is encouraging that we do not find a very large family of local minima.

Finally, we applied the ILQG method to the human arm model described above. Note that this model is stochastic: we include multiplicative noise in the muscle activations, with standard deviation equal to 20% of the control signal. Fig 6 shows the hand trajectories (in Cartesian space) resulting from open-loop and closed-loop control. Closed-loop control is based on the time-varying feedback gain matrix L generated by the ILQG method. As the endpoint distribution ellipses show, this feedback control scheme substantially reduces the effects of the noise. A detailed comparison of this model to the behavior of human subjects will be presented elsewhere.

Another encouraging result is that in terms of CPU time, the complex arm model does not require much more computation than the pendulum. On average, ILQG converged in 10 sec on the pendulum task (with 400 time steps); convergence on the arm reaching task in the absence of noise took 12 sec (with 50 time steps), and 16 seconds when multiplicative noise was included. It should also be noted that the pendulum dynamics was linearized analytically, while in the case of the arm model we had to use a centered finite difference approximation (which requires 20 evaluations of the arm dynamics). Thus, unlike global methods, ILQG does not appear to suffer from the curse of dimensionality.

VIII. Discussion

Here we developed a new local method for optimal control of stochastic nonlinear systems subject to control constraints, and applied it to two test problems – a simple pendulum and a high-dimensional biomechancial model of the human arm. In the inverted pendulum porblem we demonstrated numerically that the ILQG solutions are close to the global minimum. We plan to implement a more stable global method, using Markov Chain approximations [13], which will hopefully allow us to extend the comparisons to the stochastic case. Additional work is needed to ascertain the properties of the algorithm in more complex problems, where we cannot use global methods for validation. One possibility that we are considering is to take the solution of the ILQG method, apply a stochastic policy gradient algorithm to it (which can be very inefficient, but in the limit of large sample size avoids approximation errors), and see if the ILQG solution is a minimum of gradient descent in the space of feedback control laws. We expect to be able to include such results in the final version of the paper.

References

[1] I. Brown, E. Cheng and G. Loeb (1999). Measured and modeled properties of mammalian skeletal muscle. II. The effects of stimulus frequency on force-length and force-velocity relationships. *J. Muscle Res Cell Motil* 20: 627-643

[2] A. Bryson and Y.-C. Ho (1969). *Applied Optimal Control*. Blaisdell Publishing Company, Massachusetts, USA

[3] C. Harris and D. Wolpert (1998). Signal-dependent Noise Determines Motor Planning. *Nature*, 394: 780-784

[4] D. Jacobson and D. Mayne (1970). *Differential Dynamic Programming*. Elsevier Publishing Company, New York, USA

[5] E. Todorov and M. Jordan (2002). Optimal Feedback Control as a Theory of Motor Coordination. *Nature Neuroscience*, 5(11): 1226-1235,

[6] E. Todorov and W. Li (2003). Optimal Control Methods Suitable for Biomechanical Systems. In Proceedings of the *25th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*

[7] W. Li and E. Todorov (2004). Iterative Linear Quadratic Regulator Design for Nonlinear Biological Movement Systems. Submitted to the *1st International Conference on Informatics in Control, Automation and Robotics*

[8] J. Pantoja (1988) Differential Dynamic Programming and Newton's method. *Intl J Control* 47: 1539.

[9] C. Atkeson and J. Murimoto (2002). Nonparametric representations of policies and value functions: A trajectory-based approach. In *Neural Information Processing Systems 15*

[10] L.-Z. Liao and C. Shoemaker (1991). Convergence in unconstrained discrete-time differential dynamic programming. *IEEE Transactions on Automatic Control* 36(6): 692-706

[11] C. Nh, L.-Z. Liao and D. Li (2002). A globally convergent and efficient method for unconstrained discrete-time optimal control. *Journal of Global Optimization* 23: 401-421

[12] L.-Z. Liao and C. Shoemaker (1993). Advantages of differential dynamic programming over Newton's method for discrete-time optimal control problems. Cornell University Technical Report.

[13] H. Kushner and P. Dupuis (2001). *Numerical methods for stochastic control problems in continuous time (2nd edition)*. Spinger, New York.

[14] R. Vinter (2000). *Optimal Control*. Birkhauser, Boston.

[15] D. Bertsekas and J. Tsitsiklis (1996). *Neuro-dynamic programming*. Athena Scientific, Belmont, MA