2011 American Control Conference
on O'Farrell Street, San Francisco, CA, USA
June 29 - July 01, 2011

Discrete-time Local Dynamic Programming

Max Berniker and Konrad Kording

Abstract— **Optimal control theory is a powerful analytical tool useful for many diverse fields, including biological motor control, where the theory is used to predict characteristics of motor control problems under optimal conditions. However, finding solutions to these control problems can be very difficult when examining biological systems, where nonlinearity and stochasticity are typical. In an effort to overcome this dilemma and analyze more realistic problems, we present an algorithm that approximates the solution to the discrete-time Hamilton-Jacobi-Bellman equations. As with similar local dynamic programming algorithms, the algorithm approximates a local solution around a nominal trajectory and progressively improves the trajectory and the value function's local estimate. Using this algorithm, we obtain optimal solutions for a single joint musculo-skeletal system. In particular, we take advantage of this new algorithm to examine solutions with fast and discontinuous dynamics and non-Gaussian noise. These solutions are examined for some of the stereotypical responses of biological systems, such as the tri-phasic muscle activations and bell-shaped velocity profiles. The results are also compared with their deterministic counterparts, emphasizing the need for stochastic solutions.**

I. INTRODUCTION

Controller design, that is directing or influencing a dynamical system to achieve a desired goal, is a common theme across many scientific fields. From robotics [1] and aircraft dynamics [2] to economics and decision theory [3], whenever we can quantify a system's goals we can utilize optimal control theory to describe how best to command the system. Optimal control has been especially helpful in the examination of biological motor control. By analyzing controllers that optimize potential cost functions, optimal control is helpful in exploring how the nervous system is accomplishing motor behaviors; the analysis of movements in terms of a minimum jerk cost function [4], [5] is one such example of how an optimal control analysis can be instructive. However, linear, deterministic analyses such as these have their limitations. Many of the motor systems we examine cannot be accurately described by linear models and demand a nonlinear analysis. Moreover, deterministic analyses yield phenomena such as desired trajectories, encouraging paradigms that may be misleading when examining what are truly stochastic systems. As such, there is a genuine need for nonlinear stochastic analysis methods.

Finding optimal controllers for biological systems is notoriously difficult [6]. Many studies have bypassed these difficulties by focusing on what are more tractable problems, such as linear stochastic systems, e.g. [7], [8], or nonlinear but deterministic systems, e.g. [9], [10], [11]. Yet, neither approach can accurately capture the salient features of biological systems under many normal circumstances. Similarly, the assumption of Gaussian additive noise may be overly restrictive under many circumstances. Under restricted conditions the certainty equivalence property is met (e.g. [12], [13]) and stochastic solutions have the same form as their deterministic counterparts. However, in general, nonlinear deterministic solutions cannot correctly capture even the mean behavior of their stochastic counterparts.

Methods for solving nonlinear stochastic systems usually attempt to approximate the continuous-time Hamilton-Jacobi-Bellman (HJB) equations. The temporal dynamics of these equations along with the model dynamics are discretely integrated to approximate optimal solutions. Discretizing time is also a practical necessity to simulate random stochastic influences such as noise. There are many algorithms for approximating solutions to the HJB equations and then iteratively improving these approximations until the solutions are adequately accurate, e.g. Differential Dynamic Programming [14], Heuristic Dynamic Programming [15], [16], Local Dynamic Programming [17], Adaptive Dynamic Programming [2] and iterative linear quadratic gaussian (iLQG) control [18]. These methods usually approximate the value function along a current, best estimate of the optimal trajectory, and then successively improve both this estimate of the trajectory and the value function.

A new algorithm, iterative Local Dynamic Programming (iLDP) has been proposed for finding solutions to stochastic systems [19]. As with the algorithms above, this one aims to discretely integrate a local approximation to a system's value function. While these and similar algorithms are well suited to solve many problems, there are some drawbacks. For one, the continuous-time HJB equations prescribe the optimal conditions for small additive, Gaussian noise. This is likely a shortcoming when examining biological systems, where noise often violates these assumptions. Further, because these approaches attempt to approximate a continuously changing value function through discrete integration, a "fine-grained" time step is required when the HJB equation dictates a fast temporal derivative. These conditions, fast value function dynamics and non Gaussian noise, are easily met in biological systems. If the time step during integration is not fine enough, the algorithm may fail. Yet, using a small time step increases both the number of computations, and the memory required.

For the above reasons, solving the discrete-time (DT)HJB equations directly may be preferable. The DTHJB equations make no assumptions about the noise distributions. In addition, since the equations are discrete, solving them

M.B. and K.K. are with the Department of Physical Medicine and Rehabilitation, Northwestern Univerity, IL 60622, USA mbernike@northwestern.edu

bypasses the dangers and memory requirements of approximating the continuous-time solution with discrete integration. Moreover, as both the forward simulation of the model and the algorithm require the discretization of time, there is little practical difference in turning the overall problem into one of a discrete-time optimization.

In this work we shall present a discrete-time analogue of the iLDP algorithm for solving the DTHJB equations explicitly. The algorithm can deal with non-Gaussian noise sources and relatively large time steps. Furthermore, the algorithm introduces an asymmetrical component to the value function approximation for solving nonlinear problems where quadratic approximations are not sufficient (a known limitation for stochastic systems with even small additive noise [19]). Using this DT algorithm we examine the optimal solutions to a system whose optimal commands would be difficult to obtain otherwise. In particular, we will look at optimal solutions for a nonlinear musculo-skeletal system, and we will examine the effects of non-Gaussian noise, and discontinuous dynamics. These solutions will be examined for some of the stereotypical responses of biological systems, such as triphasic muscle activations and bell-shaped velocity profiles. Further, to emphasize the need for stochastic methods we will demonstrate how these stochastic solutions differ from their nonlinear but deterministic counterparts.

II. DISCRETE-TIME LOCAL DYNAMIC PROGRAMMING ALGORITHM

A. Overview

The optimal control problem we are trying to solve is defined through discrete-time state space equations,

$$x_{k+1} = f(x_k, u_k) + L(x_k, u_k)w_k \tag{1}$$

where x_k and u_k are the state and command at the k^{th} step, and w_k is a stochastic random variable described by a given probability distribution, $p(w)$. We shall assume that at each time step the state can be measured without error. The cost function to be minimized is of the form,

$$J = \phi(x, N) + \sum_{k=1}^{N-1} \mathcal{L}(x, u, k) \tag{2}$$

where the kernel, $\mathcal{L}$, is a penalty for being in state x with command u at the k^{th} time step and ϕ is the terminal penalty. We seek a control policy, $\pi(x, k)$ such that when starting from some initial state, x_1 and continuing until the N^{th} time step this cost function is minimized. We note that the analog continuous-time equations can easily be converted to this discrete-time form.

Since this control problem is stochastic, finding a single optimal, or desired, trajectory is not sufficient to minimize the cost function. Instead, we must find a family of extremal trajectories the state will follow as the noise term perturbs it from one time step to the next. The solution to this problem is defined by the discrete-time Hamilton-Jacobi-Bellman (DTHJB) equations. The DTHJB equations describe the conditions for optimality in terms of the problem's value function, $V(x, k)$, also known as the cost-to-go. $V(x, k)$ is the value, or total expected cost associated with starting in state x at time k, and using a control policy, π, until the terminal time step. The optimal value function as defined by the recursive DTHJB equations is,

$$\begin{aligned} V^o(x, N) &= \phi(x) \\ V^o(x, k) &= \min_u E\{\mathcal{L}(x, u) + V^o(x^o_{k+1}, k+1)\} \end{aligned} \tag{3}$$

and the corresponding optimal command is,

$$\begin{aligned} u^o &= \pi^o(x, k) \\ &= \arg\min_u E\{\mathcal{L}(x, u) + V^o(x^o_{k+1}, k+1)\} \end{aligned} \tag{4}$$

These equations express how the optimal value at the k^{th} step is the value that minimizes the expected sum of the instantaneous cost, $\mathcal{L}$, plus the optimal value associated with the proceeding optimal state at the $k+1$ time step. Note that the optimal state at the next time step is not known *a priori* and depends on the current state and the optimal command.

Starting with the terminal state and progressing back to the initial step, $k = 1$, the above equations provide a description for how the value function must be updated backwards through time. This is the basis for all dynamic programming algorithms. The difficulty is that in general the value function is a highly nonlinear function of time and space. We could try to approximate the value function with an n^{th} order polynomial in x at each time step. However, as the dimension of x (say, m) increases we must fit an ever increasing number of parameters (i.e. the curse of dimensionality). Provided we could solve for the expected value of this polynomial, we would then be faced with the difficulty of minimizing this n^{th} order polynomial in an m-dimensional space to find optimal commands (equation (4)). However, for systems where equation (3) is smooth, we expect a small region around the extremal trajectories to be well approximated by a Taylor expansion, alleviating the need for higher-order dynamics.

The algorithm we propose is based on the iterative local dynamic programming algorithm [19]. In essence, to overcome the difficulties of solving for equations (3)-(4) analytically, we locally approximate the value function about a nominal trajectory. Denoting $\bar{x}_k$ as our nominal state trajectory, and $\tilde{x}_k = x - \bar{x}_k$ as a deviation from it, we can express the value function as,

$$V(\tilde{x}_k, k) = a_k + b_k^T \tilde{x}_k + \tilde{x}_k^T C_k \tilde{x}_k + h.o.t. \tag{5}$$

We note that by defining the value function locally, rather than globally, we may avoid the need for higher order terms all together. In fact, including up to 2^{nd} order terms is sufficient for approximating well behaved systems with small additive noise. Next, we define a local approximation to the value function as follows,

$$\tilde{V}(\tilde{x}_k, k) = \hat{a}_k + \hat{b}_k^T \tilde{x}_k + \tilde{x}_k^T \hat{C}_k \tilde{x}_k + \xi(\tilde{x}_k) \tag{6}$$

This is a matrix representation of a 2^{nd} order polynomial in the state, with higher order terms in $\xi()$. With the exception

of the higher order terms in $\xi()$, the function is linear in its parameters. This greatly simplifies the update of equation (3), and solving for their estimated values merely requires a matrix inversion.

To incorporate higher order terms beyond $\tilde{x}_i\tilde{x}_j$, should they become important, we note that ideal candidate functions $\xi()$ should be nonnegative (as is the true value function) and have analytic expressions for its expectation, $E[\xi]$, and its gradient. As such, we propose the following exponential term,

$$\xi(\tilde{x}_k) = d_k exp(e_k^T \tilde{x}_k) \tag{7}$$

where e_k specifies the direction in state space in which the cost function's gradient increases fastest and d_k denotes the relative magnitude of this change. As suggested, analytic expressions for this function's expectation and its derivatives are readily available (see Appendix). Furthermore, the exponential is nonnegative, a characteristic shared with the true value function. To be clear, this does not ensure approximations to the value function cannot achieve negative values (the 2^{nd} order approximation alone may do this). However, in practice we have found that the approximations rarely, if ever, achieve such values. Rather, this is a practical attempt to approximate higher order terms not accounted for with the second order fit. In addition, by including an exponential term, the local approximation is still convex, simplifying the search for optimal commands. Finally, by including this function we can approximate asymmetric value functions. This is a characteristic that may be of functional importance for many problems, and cannot be achieved with only 2^{nd} order terms. While including cubic terms (or any odd powers) can also help to approximate asymmetric value functions, they have the distinct disadvantage of rendering the value function approximation non-positive definite and the search for optimal commands non-convex. Employing the exponential term means the value function is no longer linear in its parameters, but we can solve for d_k and e_k by boot-strapping; we first solve for the 2^{nd} order parameters, fit the exponential terms to the residual error and then repeat when necessary. This also assures that the exponential term only contributes when 2^{nd} order terms are an insufficient approximation to the value function.

Having proposed an approximate local value function, we require the corresponding optimal commands. To do this we must solve equation (4). Fortunately, under some broad assumptions of noise we can solve for the expectation of our approximate value function (see appendix). Optimal commands are then found by setting the partial derivative of $\mathcal{L} + E[V]$ to zero. This can be achieved through many nonlinear optimization techniques. However, we can greatly simplify the computation of u^o through the following procedure. First, under many circumstances the state dynamics are linear in their commands (or can be assumed to be) and are expressed as,

$$x_{k+1} = f(x_k) + B(x_k)u + L(x_k)w_k \tag{8}$$

Then, we approximate the optimal command through a Taylor expansion of the value function's derivative. By neglecting $\xi()$ we can find a closed form solution for a 2^{nd} order approximate optimal command, u_2^o. We can then Taylor expand $E[\tilde{V}]$ about $\{x(u_2^o)_{k+1}, u_2^o\}$, again retaining up to second order terms, and solve for an improved approximate u^o (if even further accuracy is required, this procedure can be repeated, see Appendix).

With a number of such pairs of states and their corresponding optimal commands, we can approximate the value function for this time step. By successively refining our approximation to the local value function, $\tilde{V}$ we can march the nominal state trajectory, $\bar{x}$ towards a trajectory that minimizes the value function, and is on average optimal. Below we outline the discrete-time local dynamic programming (DLDP) algorithm.

B. DLDP Algorithm

Our objective is to find a locally optimal controller, $\pi^o(x,k)$, and value function approximation, $\tilde{V}^o$, centered around a nominal state trajectory, $\bar{x}$ that is on average optimal. The value function approximation is parameterized by a set of values, $\Theta = \{\theta_k\}_{k=1,,N}$ where $\theta_k = \{\hat{a}_k, \hat{b}_k, \hat{C}_k, \hat{d}_k, \hat{e}_k\}$ are the parameters corresponding to the k^{th} step. The steps below describe the algorithm's iterative attempt to find a set of parameters, Θ^o and nominal trajectory, optimizing equation (2). These steps are performed repeatedly, using the current parameters to advance Θ^i towards Θ^o until appropriate convergence criteria are met. Note that the value function specifies the optimal command (equation (4)) and an explicit approximation for $\pi^o(x,k)$ is not necessary.

Step 0. Initialization. To start, an initial controller, $\pi(x,k)$, and $\Theta^{i=0}$ are defined. For simplicity, this initial controller can be a time sequence of values $\bar{u}_k$ rather than a function of x. $\Theta^{i=0}$ can be initialized with zeros.

Step 1. Determine nominal trajectory. Using the current controller, compute the nominal state trajectory around which the value function will be approximated, $\bar{x}_k$. This can be obtained by averaging over many forward integrations. Then find θ_N^{next} by approximating, $\tilde{V}(\tilde{x}, N, \theta_N) = \phi(\tilde{x}_N)$.

Step 2. Value function backup. Select a cloud of states centered around the mean trajectory, $\{x^n\}_{n=1,...M}$, then, starting at $k = N-1$, and incrementing backwards, perform the following steps.

- **Step 2a.** For each state, x^n, compute the approximate optimal command, u^n, given the current approximate value function. This is done by solving equation (4) (see Appendix and equation (26)).
- **Step 2b.** Use the optimal command from above to compute the expected state for the next stage, $x_{k+1}^n = f(x^n, u^n) + E[L(x^n, u^n)w_k]$, and compute the discrete-time Hamiltonian, $H(x^n, u^n) = \mathcal{L}(x^n, u^n) + E[\tilde{V}(x_{k+1}^n, k+1, \theta_{k+1}^i)]$.

- **Step 2c.** Use the states and Hamiltonians found above, $\{x^n, H\}_{n=1,\ldots M}$ to find θ_k^{next} by approximating, $\tilde{V}(x^n, k, \theta_k) = H(x^n, u^n)$. As described above, this can be simply done through a matrix inversion to compute the parameters, $\hat{a}_k, \hat{b}_k, \hat{C}_k$, and then fitting the residual errors to the d_k, e_k terms. If desired, an approximate policy can also be computed from the pairs $\{x^n, u^n\}_{n=1,\ldots M}$.

Step 3. After reaching $k = 1$, advance the new value function and policy forward, $\Theta^{i+1} \longrightarrow \Theta^{next}$ and compute a convergence criteria. To ensure the cost function (equation (2)) decreases after each iteration, a line search can be performed, where $\Theta^{i+1} = \epsilon\Theta^{next} + (1-\epsilon)\Theta^i$ and ϵ is a number between zero and one. If the policy and value function have not converged, return to step 1 and repeat.

III. RESULTS

A. Overview

In the examples presented below, the cloud of states was randomly drawn from a multivariate Gaussian distribution. The covariance of this distribution was updated repeatedly, such that at each time step k it coincided with the distribution of states obtained using the current estimate of the value function and policy. Approximately 500 sample states ($M \approx 500$) were used for the clouds in all solutions shown. Large values of M allow for consistent value function approximations from one backup to the next, but linearly scales the number of computations required.

Initially, when the value function approximation is relatively uninformed and inaccurate, the backups result in large helpful changes to the parameters, Θ. However, in the late stages of the algorithm, when the value function is close to converged, small fluctuations in the cloud of states from one backup to the next, result in slightly different approximations to the value function. Therefore, rather than performing a line search to update the value function, ϵ was initialized close to 1.0, and was slowly lowered to a value typically around 0.01. This allowed the algorithm to capitalize on novel information early on, and to effectively average the value function updates late in the algorithm as the value function converged. Convergence was determined when changes from one backup to the next were acceptably small in Θ and the nominal trajectory, $\bar{x}_k$.

B. Example System

To demonstrate the new algorithm, we examine optimal controllers and the resulting trajectories for a nonlinear, 4-dimensional musculo-skeletal system. The model we use is chosen to be simple enough to interpret the results, while still exhibiting some of the basic nonlinear features of muscular systems, namely stiffness, damping and equilibria, that vary with command.

We model the limb as a point mass acted upon by an agonist-antagonist pair of muscles. This is equivalent to a rotational joint with one degree of freedom. We shall designate the limb's mass as m, its displacement with y, and its velocity with v. The equations of motion are,

$$m\dot{v} = -F_r(y, v, a_r) + F_l(y, v, a_l) \tag{9}$$

where F_r and F_l are the muscle forces that pull the mass to the right, and left, respectively. We model the nonlinear muscle force as a spring-damper with variable stiffness and damping. For instance the right muscle's force is,

$$F_r = (k_{or} + \alpha_r a_r)(y_{or} - y) - (b_{or} + \beta_r a_r)v \tag{10}$$

where a_r is the activation level for this muscle, k_{or}, b_{or}, α_r and β_r are model parameters that control the muscle's stiffness and viscosity and y_{or} is this muscle's rest, or equilibrium length. The excitation-activation dynamics of muscle are modeled as a low-pass filtered version of the command, or excitation to the muscle,

$$\dot{a}_i = \frac{1}{\tau}(u_i - a_i) \tag{11}$$

where u_l and u_r are non-negative commands to the two muscles. The excitation-activation dynamics ensure that the model cannot instantaneously change the muscle's activation state, a feature that is often necessary for co-activation to be optimal. Note too that the muscle force as defined is a nonlinear function of state, dependent on the products of activation, limb displacement and velocity. The state of this nonlinear system is defined with the four variables, $x = [y, v, a_r, a_l]^T$, and the command as $u = [u_l, u_r]^T$. After discretizing the system, and allowing for noise in the excitation command, the dynamics are rewritten in the form of equation (8).

C. Optimal Solutions

In an effort to examine the merits of the new algorithm, all the results we present will be the result of minimizing the same basic cost function. Only the magnitude of the penalties and the noise sources will change. This will allow us to demonstrate how the new algorithm can perform well when similar algorithms may require larger memory and computational resources, or simply fail. The cost function is of the following form:

$$J = \frac{1}{2}(x_N - x_d)^T \Phi (x_N - x_d) + \sum_{k=1}^{N} u_k^T R u_k \tag{12}$$

where R is a penalty term for large commands and x_d is the desired terminal state. For all solutions $R = \Delta t I$, so the cost approximates the integrated norm of the command. x_d corresponds to a final state where the limb has been displaced one unit and is at rest with zero velocity.

Step Size

The terminal penalty's weight, Φ, plays a key role in our first comparisons. When this term is small relative to R, the value function's temporal gradient is small; that is, the value function changes slowly as it approaches it's terminal value, $V^o(x, N) = \frac{1}{2}(x - x_d)^T \Phi (x - x_d)$. If, on the other hand, Φ is relatively large, then the value function changes

quickly as it approaches the, relatively large, terminal value, $V^o(x, N)$. The implications for algorithms that attempt to discretely integrate the continuous-time HJB equations are clear; when the terminal penalty is relatively large, a fine discretization of time is necessary to accurately integrate the value function. This in turn requires large memory resources and more computations. Avoiding this problem is in part why we attempt to solve the discrete-time HJB equations with our new algorithm.

To illustrate these complications, and demonstrate the new algorithm's advantages, we first present a comparison between the DT algorithm and its CT counterpart. Consider the case when the value function is integrated as, $V(x,k) \approx V(x,k+1) + \Delta t \dot{V}$. Under this scenario, Δt must be chosen small enough so that the integration does not fail. It can fail when small errors integrate out to large variations in the value function, or when the quantity $\Delta t \dot{V}$ is large and negative, and drives the value function approximation towards negative values. Under a best-case scenario, let us assume the value function is approximately correct and the current nominal state trajectory is close to optimal. Under such conditions, we can linearize the system about the nominal trajectory and solve for a locally accurate value function. We can then rewrite the integration step in terms of this linearized solution as,

$$V = \frac{1}{2}\Delta\tilde{x}_k^T S_k \Delta\tilde{x}_k \approx \frac{1}{2}\Delta\tilde{x}_k^T (S_{k+1} + \Delta t \dot{S})\Delta\tilde{x}_k \qquad (13)$$

where $\dot{S}$ is defined by its matrix Ricatti equations. In this form, we see that the value function will achieve negative values, and the integration will fail, when the sum $S_{k+1} + \Delta t \dot{S}$ is negative-definite. Under appropriate conditions, we can find this maximum upper bound on Δt. For instance, when we set $\Phi = 5I$ and consider a 1 second movement, we find $\Delta t \leq 0.02$. However, in practice we find that this upper bound is very liberal. In order to solve this problem with the iLDP algorithm, we were forced to reduce Δt to 0.005 seconds. The results obtained with the iLDP and DLDP algorithms are nearly identical (see figure 1A). However, using the DLDP algorithm we could also solve this problem by doubling the step size to 0.01 seconds, or quadrupling it to 0.02 seconds (figure 1A), halving and quartering the memory and computations, respectively.

We note some intriguing characteristics of these optimal movements that are common across all the solutions presented here. First, the movements have a smooth bell-shaped velocity profile. This is not uncommon and can be found when optimizing many cost functions, e.g.[4], [9], [11]. We also note that this model, simple though it may be, does capture the phenomena of triphasic burst patterns. This is a feature commonly observed in human reaching movements [20] and typical among the solutions obtained with this model.

To further demonstrate the merits of the new algorithm, we found the optimal controller after increasing the terminal penalty Φ from $5I$ to $40I$. Under these conditions, the value function changes quickly and we were no longer able to find a solution using the iLDP method. Our discrete-time algorithm was able to find a solution under a wide range of time steps (see figure 1B for $\Delta t = .01$). As can be seen, by increasing the terminal penalty, the mean terminal position is now closer to the desired target location. In addition, the variance of the trajectories under this controller is necessarily reduced, to decrease terminal errors. We also note what can be interpreted as an increase in co-activation under these conditions, as there is a larger amount of overlap between the agonist and antagonist musucle's activations, which acts to increase the muscle's combined stiffness without increasing the muscle force. This heightened stiffness acts to further decrease the influences of noise on the trajectory.

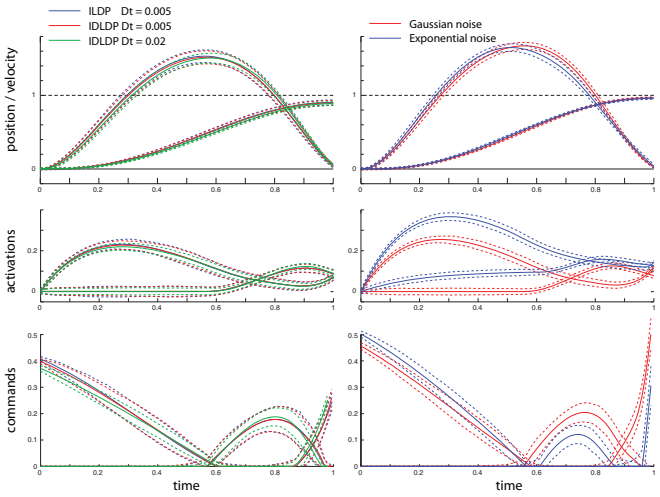

Fig. 1. Comparison of optimal solutions. Solid lines represent the mean trajectories and dashed lines depict one standard deviation. A) Optimal solutions when $\Phi = 5I$. Solutions were obtained using the iLDP algorithm with a time step of 0.005 seconds (blue traces), and the DLDP algorithm with time steps of 0.005 (red traces) and 0.02 (green traces) seconds. B) Optimal solutions when the terminal penalty is increased to $\Phi = 40I$. Solutions were obtained for two noise sources, Gaussian (red traces) and Exponential (blue traces).

Non-Gaussian Noise

For musculo-skeletal systems, noise is a salient feature, and in general it is not characterized by Gaussian distributions [21]. However, most algorithms for solving optimal problems require this assumption. As a further demonstration of the new algorithm, we solved for the optimal controller, again using $\Phi = 40I, dt = 0.01$, but with an exponential distribution over the motor noise. An exponential distribution is a further refinement to our model, ensuring muscle commands are never negative, or inhibitory, as with true muscle physiology. Furthermore, experimental studies suggest that the firing rates of neurons are characterized by a family of exponential distributions (e.g. [22]).

The resulting optimal controller (figure 1B) is qualitatively distinct from the controllers found under the Gaussian noise assumption. Just as before, the movements made still retain the familiar bell-shaped velocity profile, however, now the

commands to the muscles are distinct from the previous solutions (compare figure 1A, B). The commands are initially larger, but decrease to values below those obtained with Gaussian noise. Furthermore, there is little to no commanded co-activation under these conditions. However, due to the non-zero mean of the noise distribution, the two muscles invariably experience some level of co-activation. These results not only demonstrate how different classes of noise can impact the optimal solutions, but also the need for algorithms such as the one presented here, to analyze these systems.

Deterministic vs. Stochastic Solutions

Our last solutions serve not only to demonstrate the usefulness of the DLDP algorithm, but also the need for stochastic methods in general. Under the assumption of small additive noise, optimal controllers for stochastic systems are found by first solving for the deterministic trajectory obtained under zero noise conditions, and obtaining a linearized controller (i.e. LQG controller) about this trajectory. However, in general the mean commands and trajectories of a stochastic system will differ from those of it's deterministic counterpart. This fact limits the usefulness of deterministic analysis for stochastic systems such as the human motor system and emphasizes the need for a stochastic analysis.

Consider the task of walking along the edge of a cliff. Assume that you want to walk as close to the edge as is possible, without tumbling over (a large penalty). Under a deterministic scenario, the best possible solution has you walking along the edge of the cliff; this minimizes your distance from it, while keeping you from tumbling over it. However, common sense would suggest that this is an overly parsimonious (and dangerous) solution. Under more realistic assumptions of stochasticity, (i.e. motor noise, uncertain feedback and knowledge of the cliff's edge, etc.) the optimal solution should demand you walk at a distance from the edge that balances your fear of tumbling over with your desire to remain close to it. Furthermore, as your fear or uncertainty of the edge of the cliff increases, the optimal distance between you and it ought to increase as well.

Using our musculo-skeletal model, we solve the analog of this cliff-walking problem. We model a force, F_{cliff} that tends to drive the limb in the negative direction, but only when $y \leq y_{cliff}$. Then we propose a cost function (equation (12)) whose initial and desired terminal position are just slightly larger than y_{cliff} ($y_d = 1.0$). Now, the problem is to remain close to the edge of the cliff, while avoiding falling over it. However, to remain at values $y > y_{cliff}$ requires non-zero motor commands, so the optimal controller is non-trivial. We solved this problem under the assumption of Gaussian noise using our DLDP algorithm. For each value of F_{cliff} ten solutions were found, and their results were averaged. In addition, we also solved for the optimal deterministic trajectory. Under the deterministic assumption, the limb's trajectory moves slightly toward the cliff (to minimize motor costs), but does not reach it (see figure 2A). Importantly, since the limb remains above the cliff's

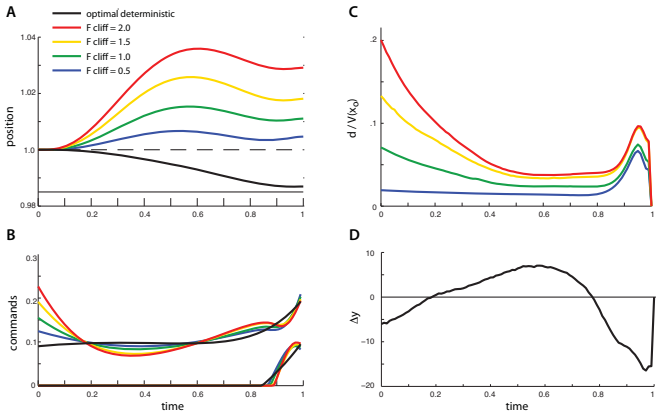

Fig. 2. Optimal solutions for cliff problem. A) Deterministic and mean paths for optimal displacements as the strength of the cliff is varied. y_{cliff} is depicted with thin black line. B) Deterministic and mean commands for optimal commands. C) The increasing exponential weight term, d_k as the cliff strength increases. D) The y-component of the exponential direction term, e_k.

edge, the magnitude of F_{cliff} has no influence on the limb's movement and the optimal deterministic solution remains invariant regardless of how large F_{cliff} is. Examining the stochastic solutions, we find a qualitatively different result. Since random errors in the muscle command may drive the limb across the cliff's edge, the optimal trajectories move the limb away from it (see figure 2A). Furthermore, as we increase F_{cliff}, crossing the edge becomes a greater risk, justifying larger motor commands to avoid it (figure 2B). As a result, the limb's trajectories move farther from the edge.

Since the dynamics of this task are asymmetric, we expect the value function to be as well. To this end, we display the exponential weights of the value function. As the cliff force increases, we find that the scaling term for the exponential, d_k increases (figure 2C), confirming the asymmetry of the value function. The direction of the exponential term, e_k indicates the direction in which the value function increases fastest. For illustrative purposes, we show the y-component of this four element vector (there was no trend with increasing F_{cliff}, so the average across all conditions is shown). We see that initially and finally, moving away from the nominal trajectory in the negative direction, towards the cliff, incurs costs fastest; moving towards the cliff at these times increases the chances of falling over it. However, at intermediate times, moving away from the nominal trajectory away from the cliff, incurs costs fastest; this unnecessarily incurs penalties by being too far from the cliff.

For comparison, we repeated the above procedure with the DLDP algorithm, but suppressed the exponential term. Though the differences were small, for each value of F_{cliff}, the cost-to-go with this 2^{nd} order solution was worse than that obtained above using the exponential term. The differences were statistically significant ($p < 0.05$) for all but the smallest value of F_{cliff}. These results demonstrate the need for an asymmetric value function, and the importance of a stochastic analysis for nonlinear systems, whose mean be-

havior can be distinct from that predicted by a deterministic analysis.

IV. CONCLUSIONS AND FUTURE WORKS

Here we have presented an algorithm that approximates the solution to the discrete-time HJB equations. The discrete-time problem requires a unique algorithm to solve its own version of the HJB equations. To this end we have extended the iDLP algorithm to an explicitly discrete-time domain and added a higher order, asymmetrical term to the value function approximation. What's more, by solving the discrete-time HJB equations, we work with summations over time, not an integral. This allows for a more liberal choice in step size when numerically integrating a value function with large (fast) temporal derivatives. Furthermore, the discrete-time formalization allows for more relaxed assumptions on noise sources. Our results on a musculo-skeletal system demonstrate the algorithm's use in solving optimal control problems with fast dynamics, non-Gaussian noise, and asymmetrical value functions that would be difficult to solve with similar algorithms.

It should be noted that discrete-time systems are not limited to continuous-time problems whose dynamics have been discretized. Many dynamical systems are best described by discrete-time, or multi-stage processes. Indeed, there are aspects of biological motor control that appear discretized as well. There is a great deal of evidence to suggest that the human motor behaviors are constructed with commands issued, not continuously, but at discrete times. The resulting movements are comprised of discrete components. These so-called submovements appear to be a fundamental character of the human motor system, evident in both normal and impaired subjects [23], [24]. The algorithm presented here may be an especially useful tool in analyzing a biological control architecture that is similarly restricted to discretely issued commands.

With our algorithm we have assumed that in the region of optimal trajectories a second order approximation to the value function would be adequate and approximated higher order influences with an exponential term. Under more disruptive assumptions of noise, or abruptly changing dynamics (as in the cliff problem) the distribution of optimal states grows in size and a second order approximation may do poorly. Under these circumstances the exponential term allows for a better approximation to the value function. Furthermore, because the exponential term is asymmetric, it can depict the directions in which the state achieves the least desirable values. If this choice of representing higher order terms with an exponential were found to be inadequate, other forms for the value function could be used with minor changes to the basic algorithm.

Several assumptions were made in this work to simplify implementation of the algorithm. For instance, we have modeled the state dynamics as linear in their commands. Although this was a natural choice for the systems we were modeling, it need not be true for the algorithm in general. When the state dynamics are linear in u, solving for the minimizing command is greatly simplified. However, many standard and proficient minimizing algorithms could be used for this step. Additionally, regarding noise sources we have assumed a fixed distribution (either Gaussian or exponential), as this makes the analysis relatively easy. However, it is known that for biological systems the noise in commands signal-dependent, making the control problem more difficult. In future implementations of this work we hope to address many of these assumptions while examining other, more sophisticated biological motor systems.

V. APPENDIX

In order to compute the value function updates (equation (3)) as well as the optimal commands (equation (4)), we need to compute the expected value function. First we shall consider the case of Gaussian noise. For notational convenience, we refer to the state and command at the $k-1$ time step as x and u respectively and $\tilde{V}(x_k, k, \theta_k)$ as $\tilde{V}(x_k)$. The expected value of x_k, conditioned on x and u is,

$$E[x_k] = \hat{x}_k = f(x) + B(x)u \tag{14}$$

and the expected value function, $\hat{\tilde{V}}(\tilde{x}_k, k)$ can be expanded as,

$$E[\tilde{V}(\tilde{x}_k)] = a_k + b_k^T E[\tilde{x}_k] + E[\tilde{x}_k^T C_k \tilde{x}_k] + E[\xi(\tilde{x}_k)] \tag{15}$$

where $\tilde{x}_k = x_k - \bar{x}_k$. The expectation of the first three terms is,

$$\begin{aligned} E[a_k + b_k^T \tilde{x}_k + \tilde{x}_k^T C_k \tilde{x}_k] = a_k + b_k^T(\hat{x}_k - \bar{x}_k) \\ +(\hat{x}_k - \bar{x}_k)^T C_k (\hat{x}_k - \bar{x}_k) + Tr(WL(x)^T C_k L(x)) \end{aligned} \tag{16}$$

The expectation of $\xi()$ is,

$$E[\xi(\tilde{x}_k)] = d_k \exp(e_k^T(\hat{x}_k - \bar{x}_k)) \exp(e_k^T L(x) W L(x)^T e_k/2) \tag{17}$$

Collecting terms, the expected value function is,

$$\begin{aligned} \hat{\tilde{V}}(\tilde{x}_k) = a_k + b_k^T(\hat{x}_k - \bar{x}_k) + (\hat{x}_k - \bar{x}_k)^T C_k (\hat{x}_k - \bar{x}_k) \\ +Tr(WL(x)^T C_k L(x)) + \\ d_k \exp(e_k^T(\hat{x}_k - \bar{x}_k)) \exp(e_k^T L(x) W L(x)^T e_k/2) \end{aligned} \tag{18}$$

Now, to compute the optimal command (equation (4)) we must minimize the resulting Hamiltonian. This is done by taking the partial derivative of the Hamiltonian and solving for the extremum command, u^o such that,

$$\frac{\partial \mathcal{L}(x_k, u^o)}{\partial u} + \frac{\partial \hat{\tilde{V}}(\tilde{x}_k)}{\partial \hat{x}_k} \frac{\partial \hat{x}_k(u^o)}{\partial u} = 0 \tag{19}$$

To do this, we first compute the command based on the second order terms, this will be an accurate approximation when the x is close to the nominal state, $\bar{x}$. We then, compute a Taylor expansion of $\xi()$ about this second order solution to compute an updated, more accurate command. Define this second order command as,

$$u_2^o = \arg\min_u \{\mathcal{L}(x, u) + \hat{\tilde{V}}_2^o(\tilde{x}_k^o)\} \tag{20}$$

where,

$$\begin{aligned} \hat{\tilde{V}}_2 = a_k + b_k^T(\hat{x}_k - \bar{x}_k) + (\hat{x}_k - \bar{x}_k)^T C_k (\hat{x}_k - \bar{x}_k) \\ +Tr(WL(x)^T C_k L(x)) \end{aligned} \tag{21}$$

assuming a quadratic motor cost, $\mathcal{L} = 1/2 u^T R u$, we arrive at the following,

$$u_2^o = -(R + 2B(x)^T C_k B(x))^{-1} B(x)^T (b_k + 2C_k(f(x) - \bar{x}_k)) \tag{22}$$

We then compute the forward state based on this second order command, $\hat{x}_2^o = f(x) + B(x)u_2^o$. These second order approximations

are used to compute an updated second order approximation to the expected value function.

Expanding the value function we find,

$$\begin{aligned}\hat{\tilde{V}} \approx a_k + b_k^T(\hat{x}_k - \bar{x}_k) + (\hat{x}_k - \bar{x}_k)^T C_k(\hat{x}_k - \bar{x}_k) \\ +Tr(WL(x)^T C_k L(x)) + \hat{\xi}(\tilde{x}_2^o) \\ +\hat{\xi}'(\tilde{x}_2^o)(\hat{x}_k - \hat{x}_2^o) + \frac{1}{2}(\hat{x}_k - \hat{x}_2^o)^T \hat{\xi}''(\tilde{x}_2^o)(\hat{x}_k - \hat{x}_2^o) \end{aligned} \quad (23)$$

for ease of notation, we'll denote the following,

$$\eta(\hat{x}_2^o) = d_k e^{(e_k^T(\hat{x}_2^o - \bar{x}_k))} e^{e_k^T L(x) W L(x)^T e_k/2} \quad (24)$$

then we can rewrite the value function as,

$$\begin{aligned}\hat{\tilde{V}} \approx a_k + b_k^T(\hat{x}_k - \bar{x}_k) + (\hat{x}_k - \bar{x}_k)^T C_k(\hat{x}_k - \bar{x}_k) \\ +Tr(WL(x)^T C_k L(x)) + \hat{\xi}(\tilde{x}_2^o) + \eta(\hat{x}_2^o) e_k^T(\hat{x}_k - \hat{x}_2^o) \\ +\frac{1}{2}\eta(\hat{x}_2^o)(\hat{x}_k - \hat{x}_2^o)^T e_k e_k^T(\hat{x}_2^o)(\hat{x}_k - \hat{x}_2^o) \end{aligned} \quad (25)$$

Just as before, we minimize the Hamiltonian, by computing the appropriate derivatives and isolating the command. Dropping the x dependencies for notational convenience we finally we arrive at the optimal command,

$$\begin{aligned}u^o = -\left(R + 2B^T C_k B + \eta B^T e_k e_k^T B\right)^{-1} \\ B^T\left(b_k + 2C_k(f(x) - \bar{x}_k) + \eta e_k + \eta e_k e_k^T(f(x) - \hat{x}_2^o)\right) \end{aligned} \quad (26)$$

For the case of exponential noise we must first define the distribution over w.

$$p(w) = \Pi \lambda_i \exp(-\lambda^T w) \quad (27)$$

where the expected value is, $E[w] = \hat{w} = [1/\lambda_1, 1/\lambda_2, ..., 1/\lambda_s]$ and the covariance matrix is $E[ww^T] = W = diag(1/\lambda_i^2)$. We redefine $\hat{x}_k$ as $f(x) + B(x)u$. We can then express the expected value function as,

$$\begin{aligned}\hat{\tilde{V}}(\bar{x}_k) = a_k + b_k^T(\hat{x}_k + L(x)\bar{w} - \bar{x}_k) \\ +(\hat{x}_k - \bar{x}_k)^T C_k(\hat{x}_k - \bar{x}_k) + 2(\hat{x}_k - \bar{x}_k)^T C_k L(x)\bar{w} \\ +Tr(WL(x)^T C_k L(x)) \\ +d_k \exp(e_k^T(\hat{x}_k - \bar{x}_k))\frac{\Pi\lambda_i}{\Pi(\lambda^T - e_k^T L(x))_i} \end{aligned} \quad (28)$$

now, the second order approximate command and optimal state are,

$$\begin{aligned} u_2^o &= -(R + 2B(x)^T C_k B(x))^{-1}(B(x)^T b_k + 2B^T C_k L(x)\bar{w} \\ &\quad +2B(x)^T C_k(f(x) - \bar{x}_k)) \quad (29) \\ \hat{x}_2^o &= f(x) + B(x)u_2^o \quad (30) \end{aligned}$$

then, as before, we'll define a scalar variable,

$$\eta = d_k \exp(e_k^T(\hat{x}_k - \bar{x}_k)\frac{\Pi\lambda_i}{\Pi(\lambda^T - e_k^T L(x))_i} \quad (31)$$

and expand the expected value function up to second order terms,

$$\begin{aligned}\hat{\tilde{V}} \approx a_k + b_k^T(\hat{x}_k + L(x)\bar{w} - \bar{x}_k) + (\hat{x}_k - \bar{x}_k)^T C_k(\hat{x}_k - \bar{x}_k) \\ +2(\hat{x}_k - \bar{x}_k)^T C_k L(x)\bar{w} + Tr(WL(x)^T C_k L(x)) \\ +\eta(\hat{x}_2^o) e_k^T(\hat{x}_k - \hat{x}_2^o) + \frac{1}{2}\eta(\hat{x}_2^o)(\hat{x}_k - \hat{x}_2^o)^T e_k e_k^T(\hat{x}_2^o)(\hat{x}_k - \hat{x}_2^o) \end{aligned} \quad (32)$$

Taking the derivative with respect to u, and isolating, we find the optimal command,

$$\begin{aligned}u^o = -\left(R + 2B^T C_k B + \eta B^T e_k e_k^T B\right)^{-1} B^T \\ \left(b_k + 2C_k L\bar{w} + \eta e_k + 2C_k(f - \bar{x}_k)) + \eta e_k e_k^T(f - \hat{x}_2^o)\right) \end{aligned} \quad (33)$$

VI. ACKNOWLEDGMENTS

The authors would like to thank Emanuel Todorov for valuable discussions on this work.

REFERENCES

[1] A.R. Willms and S.X. Yang. An efficient dynamic system for real-time robot-path planning. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 36(4):755–766, Aug. 2006.

[2] J.J. Murray, C.J. Cox, G.G. Lendaris, and R. Saeks. Adaptive dynamic programming. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 32(2):140–153, May 2002.

[3] Jae Won Lee, Jonghun Park, Jangmin O, Jongwoo Lee, and Euyseok Hong. A multiagent approach to q-learning for daily stock trading. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 37(6):864–877, Nov. 2007.

[4] N Hogan. An organizing principle for a class of voluntary movements. *The Journal of Neuroscience*, 4(11):2745–2754, Nov 1984.

[5] T Flash and N Hogan. The coordination of arm movements: An experimentally confirmed mathematical model. *The Journal of Neuroscience*, 5(7):1688–1703, July 1985.

[6] A. Karniel and G.F. Inbar. Human motor control: learning to control a time-varying, nonlinear, many-to-one system. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 30(1):1–11, Feb 2000.

[7] E Todorov and M I Jordan. Optimal feedback controls as a theory of motor coordination. *Nature Neuroscience*, 5(11), Nov 2002.

[8] J Izawa, T Rane, O Donchin, and R Shadmehr. Motor adaptation as a process of reoptimization. *J Neurosci*, 28(11):2883–2891, Mar 2008.

[9] Y Uno, M Kawato, and R Suzuki. Formation and controlf of optimal trajectory in human multijoint arm movement. *Biological Cybernetics*, 61:89–101, 1989.

[10] M Kawato. Trajectory formation in arm movements: Minimization principles and procedures. In H.N. Zelaznik, editor, *Advances in Motor Learning and Control*, pages 225–259. Human Kinetics Publishers, Champaign IL, 1996.

[11] E Nakano, H Imamizu, R Osu, Y Uno, H Gomi, T Yoshioka, and M Kawato. Quantitative examinations of internal representation for arm trajectory planning: Minimum commanded torque change model. *Journal of Nuerophysiology*, 81:2140–2155, 1999.

[12] Y. Bar-Shalom and E. Tse. Dual effect, certainty equivalence, and separation in stochastic control. *Automatic Control, IEEE Transactions on*, 19(5):494–500, Oct 1974.

[13] Robert F. Stengel. *Optimal Control and Estimation*. Dover Publicaitons, 1994.

[14] D. Jacobson and D. Mayne. *Differential Dynamic Programming*. American Elsevier Publishing Company, Inc., New York, 1970.

[15] P J Werbos. Consistency of hdp applied to a simple reinforcement learning problem. *Neural Networks*, 3(2):179–189, 1990.

[16] A. Al-Tamimi, F.L. Lewis, and M. Abu-Khalaf. Discrete-time nonlinear hjb solution using approximate dynamic programming: Convergence proof. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 38(4):943–949, Aug. 2008.

[17] N A Borghese and M A Arbib. Generation of temporal sequences using local dynamic programming. *Neural Networks*, 8(1):39–54, 1995.

[18] Weiwei Li and E. Todorov. An iterative optimal control and estimation design for nonlinear stochastic system. In *Decision and Control, 2006 45th IEEE Conference on*, pages 3242–3247, Dec. 2006.

[19] E Todorov and Y Tassa. Iterative local dynamic programming. *To appear in IEEE ADPRL*, 2008.

[20] S H Brown and J D Cooke. Movement-related phasic muscle activation. i. relations with temporal profile of movement. *J Neurophysiol*, 63(3):455–464, Mar 1990.

[21] A A Faisal and D M Wolpert. Near optimal combination of sensory and motor uncertainty in time during a naturalistic perception-action task. *J Neurophysiol*, 101(4):1901–1912, Apr 2009.

[22] W Bair, C Koch, W Newsome, and K Britten. Power spectrum analysis of bursting cells in area mt in the behaving monkey. *J Neurosci*, 14(5 Pt 1):2870–2892, May 1994.

[23] R C Miall, D J Weir, and J F Stein. Intermittency in human manual tracking tasks. *J Mot Behav*, 25(1):53–63, Mar 1993.

[24] H I Krebs, M L Aisen, B T Volpe, and N Hogan. Quantization of continuous arm movements in humans with brain injury. *Proc Natl Acad Sci U S A*, 96(8):4645–4649, Apr 1999.