Optimal Control Methods Suitable for Biomechanical Systems

E. Todorov[1], W. Li[2]
[1]Department of Cognitive Science, University of California San Diego, CA, USA
[2]Department of Mechanical and Aerospace Engineering, University of California San Diego, CA, USA

Abstract—**Optimal control theory plays a key role in the study of biological movement. Further progress requires optimal control methods for realistic biomechanical systems, that have a number of distinguishing characteristics. Here we consider available control methods in light of those characteristics, compare them empirically on a 2-link arm model, and propose appropriate extensions.**

Acknowledgements—**Supported by NIH 1R01-NS045915-01**

I. INTRODUCTION

Many theories in the physical sciences are expressed in terms of optimality principles, which often provide the most parsimonious description of the laws governing a system's behavior. Optimality has also played a key role in the field of motor control. This is partly motivated by the parsimony and empirical success of optimal control models of biological movement. Perhaps more importantly, such models are appealing because all the processes that give rise to a specific motor system under investigation (evolution, development, learning, adaptation, recovery) are in a sense optimization processes, that over time cause the system to perform better and better. It is therefore natural to use the limit of optimal performance as the starting point for theoretical investigations of motor control.

Such investigations can only be productive, however, if we have efficient optimal control methods that are suitable for biomechanical systems. Biomechanical systems have a number of characteristics that distinguish them from the synthetic systems commonly studied in control theory.
Thus the purpose of this paper is to:

1. Consider these distinguishing characteristics and their consequences.
2. Summarize the existing optimal control methods that appear most suitable for such systems.
3. Illustrate the performance of such methods on a standard 2-link arm model.
4. Propose appropriate extensions.

A. Distinguishing features of biomechanical systems

1. The state spaces of biomechanical systems have unusually high dimensionality. Consider for example the simple 2-link, 6-muscle arm model often studied in motor control. The state space includes 2 joint angles and 2 joint velocities – since we are dealing with a second-order system. A realistic state description should also include 6 muscle activations – because muscles act as low-pass filters of neural activity, with non-negligible time constant. A similar count for a complete model of the arm (excluding the hand) yields ~20 dynamic states, and ~50 muscle states. Such state spaces cannot be discretized, which rules out all methods relying on discretization. Consequently, the focus of this paper is on continuous trajectory-based methods.

2. The substantial variability of biological movements indicates that the sensory-motor system operates in the presence of large (mostly internal) disturbances. Specifically, motor noise is well-known to be control-dependent, with standard deviation increasing linearly with the magnitude of the control signal. Optimal control of such systems should obviously take this phenomenon into account, because an appropriately chosen control signal can actually decrease the noise. In this paper we develop an extension of trajectory-based optimal control methods, which makes them applicable to control-dependent noise.

II. APPROACHES TO OPTIMAL CONTROL

We begin with the deterministic case, and include stochasticity later. Consider a continuous dynamical system $\mathbf{f}(t,\mathbf{x},\mathbf{u}) \in \mathbb{R}^m$ with state $\mathbf{x}(t) \in \mathbb{R}^m$, control signal $\mathbf{u}(t) \in \mathbb{R}^n$, specified final time T, initial state $\mathbf{x}_0 \in \mathbb{R}^m$, final cost $h(\mathbf{x}) \in \mathbb{R}$, and instantaneous cost rate $l(t,\mathbf{x},\mathbf{u}) \in \mathbb{R}$. The *system dynamics* is

$$\dot{\mathbf{x}}(t) = \mathbf{f}(t,\mathbf{x}(t),\mathbf{u}(t)); \quad \mathbf{x}(0) = \mathbf{x}_0$$

and the *performance criterion* to be minimized is

$$J = h(\mathbf{x}(T)) + \int_0^T l(t,\mathbf{x}(t),\mathbf{u}(t))\,dt\,.$$

The development of optimal control theory has followed two different approaches, which, interestingly, correspond to the distinction between the Lagrangian and Hamiltonian formulations of physics. Both approaches are based on the value function

$$V(\tau,\mathbf{x}(\tau)) = h(\mathbf{x}(T)) + \int_\tau^T l(t,\mathbf{x}(t),\mathbf{u}(t))\,dt$$

which tells us how much cost will accumulate if the system is initialized in state $\mathbf{x}(\tau)$ at time τ, and controlled according to a certain control law until the end of the movement. We distinguish the value function $V^{\boldsymbol{\pi}}$ for a specific control law $\mathbf{u} = \boldsymbol{\pi}(t,\mathbf{x})$, and the optimal value function V (often labeled as V^*) which corresponds to an optimal control law. The presentation of both approaches is simplified by introducing the (extended) Hamiltonian

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

which differs from its physics counterpart by the control $\mathbf{u}$. With these definitions[1], the two approaches to optimal control can be stated as:

Bellman's Optimality Principle **(Eq 1)**

$$V_t(t,\mathbf{x}) = -H\left(t,\mathbf{x},\mathbf{u}^*(t,\mathbf{x}),V_{\mathbf{x}}(t,\mathbf{x})\right)$$
$$V(T,\mathbf{x}) = h(\mathbf{x})$$
$$\mathbf{u}^*(t,\mathbf{x}) = \arg\min_{\mathbf{v}} H\left(t,\mathbf{x},\mathbf{v},V_{\mathbf{x}}(t,\mathbf{x})\right)$$

Pontryagin's Maximum Principle **(Eq 2)**

$$\dot{V}_{\mathbf{x}}(t) = -H_{\mathbf{x}}\left(t,\mathbf{x}(t),\mathbf{u}^*(t),V_{\mathbf{x}}(t)\right)$$
$$V_{\mathbf{x}}(T) = h_{\mathbf{x}}\left(\mathbf{x}(T)\right)$$
$$\mathbf{u}^*(t) = \arg\min_{\mathbf{v}} H\left(t,\mathbf{x}(t),\mathbf{v},V_{\mathbf{x}}(t)\right)$$

Both principles involve optimization of the Hamiltonian w.r.t. $\mathbf{u}$, initialization of a V-related quantity using the final cost h, and a differential equation that can be used in a backward pass through time. Despite the apparent similarities, however, the two are fundamentally different.

Bellman's Optimality Principle refers to $V(t,\mathbf{x})$ and $\mathbf{u}^*(t,\mathbf{x})$ at all possible states $\mathbf{x}$, and therefore leads to global methods that compute an optimal control law everywhere. Such methods typically represent the value function on a discrete grid, and use difference equations that relate the spatial and temporal derivatives of V to compute $V(t-\Delta t,\mathbf{x})$ from $V(t,\mathbf{x})$ in a backward pass. This is known as dynamic programming or value iteration. One can also guess a control law $\mathbf{u} = \boldsymbol{\pi}(t,\mathbf{x})$, compute its value function $V^{\boldsymbol{\pi}}$, and use it in the minimization in Eq 1 (instead of the optimal value function) in order to obtain an improved control law. This latter method is know as policy iteration. On a finite grid, both methods are guaranteed to converge to the globally optimal solution in finite time. But the problem is that one cannot discretize a high dimensional state space due to the exponential growth of the number of grid points with dimensionality (i.e. "the curse of dimensionality"). One way to avoid the curse of dimensionality is to use function approximation, but such methods are not guaranteed to converge, and even when they do it is not yet clear what they converge to.

Pontryagin's Maximum Principle, on the other hand, refers only to quantities along a specific trajectory. Suppose the controls $\mathbf{u}^*(t)$ are chosen optimally. The Maximum principle states that if we were to compute the state trajectory $\mathbf{x}(t)$ resulting from $\mathbf{u}^*(t)$, then compute the corresponding "costate" trajectory $V_{\mathbf{x}}(t)$ from Eq 2, and perform the minimization in Eq 2, $\mathbf{u}^*(t)$ will not change. In other words, Eq 2 is a necessary condition that an optimal sequence of controls (and the corresponding state and costate trajectories) have to satisfy. It becomes a sufficient condition when the Hamiltonian is convex in both $\mathbf{x}$ and $\mathbf{u}$ for all values of t and $\mathbf{p}$. When convexity does not hold, the solutions to Eq 2 (called "extremal" trajectories) are canditate trajectories one of which is the optimal.

Thus computational methods based on Pontryagin's Maximum Principle avoid the curse of dimensionality. Given the very large dimensionality of biomechanical state spaces, we believe that such methods are more promising than the global methods based on dynamic programming.

III. TRAJECTORY-BASED ALGORITHMS

Below we consider three classes of algorithms that find extremal trajectories satisfying Eq 2.

A. Solving a boundary-value problem (ODE)

In this approach the trajectory is represented as functions $\mathbf{x}(t)$ and $V_{\mathbf{x}}(t)$, and it is assumed that $\mathbf{u}(t)$ can be computed explicitly via the minimization in Eq 2, so that we have $\mathbf{u}(t) = \boldsymbol{\pi}\left(t,\mathbf{x}(t),V_{\mathbf{x}}(t)\right)$. This reduces the problem to the following system of ordinary differential equations:

$$\dot{\mathbf{x}}(t) = \mathbf{f}\left(t,\mathbf{x},\boldsymbol{\pi}(t,\mathbf{x},V_{\mathbf{x}})\right), \qquad t \in [0;T]$$
$$\dot{V}_{\mathbf{x}}(t) = -H_{\mathbf{x}}\left(t,\mathbf{x}(t),\boldsymbol{\pi}(t,\mathbf{x},V_{\mathbf{x}}),V_{\mathbf{x}}(t)\right)$$

with boundary conditions from Eq 2 and $\mathbf{x}(0) = \mathbf{x}_0$. The explicit minimization is possible, for example, when the system dynamics is linear in $\mathbf{u}$ and the cost rate is quadratic in $\mathbf{u}$. Then we have $\boldsymbol{\pi}(t,\mathbf{x},V_{\mathbf{x}}) = -l_{\mathbf{uu}}{}^{-1}l_{\mathbf{u}}{}^{T}V_{\mathbf{x}}/2$.

The initial conditions are specified at different times, so this is a boundary value problem. MATLAB 6 provides the function BVP4C, which implements a state-of-the-art relaxation method for solving such problems, using an adaptive grid (in time) and polynomial co-location. The ODE method we refer to below uses BVP4C.

B. Minimizing the performance criterion (CG)

The alternative is to represent the trajectory with the function $\mathbf{u}(t)$, and use the system dynamics and Eq 2 to compute $\mathbf{x}(t)$ and $V_{\mathbf{x}}(t)$ given $\mathbf{u}(t)$. The performance

[1] The notation V_t stands for $\partial V/\partial t$.

criterion J is a function of $\mathbf{u}(\bullet)$, and so it can be optimized directly. What does that have to do with the Maximum Principle? It turns out that the gradient of J w.r.t $\mathbf{u}$ is exactly $\partial H(t,\mathbf{x},\mathbf{u},V_{\mathbf{x}})/\partial\mathbf{u}$, with the functions $\mathbf{x}(t)$ and $V_{\mathbf{x}}(t)$ as given by Eq 2. Given the gradient we can apply steepest descent: find the gradient and peform a line search in that directon. It is better however to use a conjugate gradient method (CG). CG is known to exhibit quadratic convergence near a local minimum, which is also a property of second-order methods that compute the Hessian. Efficient methods for computing the Hessian of J have been derived. But they are very similar to the next method we consider, and so we do not address them here.

C. Differential dynamic programming (DDP)

This method uses dynamic programming locally, around the current estimate of the extremal trajectory, to generate a locally optimal feedback control law. Then the estimate of the exremal trajectory is improved. As in (B), the current estimate of the trajectrory is represented as $\mathbf{u}(t)$, and the system dynamics is integrated forward to obtain $\mathbf{x}(t)$. The following backwards recursion yields a quadratic approximation of V in the vicinity if $\mathbf{x}(t)$, by computing the gradient $V_{\mathbf{x}}(t)$ and Hessian $V_{\mathbf{xx}}(t)$. Discretizing time by Δt, the update equations are the following.

- Initialization:

$$\mathbf{x}^{new}(0)=\mathbf{x}(0) \qquad V_{\mathbf{x}}(T)=h_{\mathbf{x}}(\mathbf{x}(T))$$
$$\mathbf{u}^{new}(0)=\mathbf{u}(0) \qquad V_{\mathbf{xx}}(T)=h_{\mathbf{xx}}(\mathbf{x}(T))$$

- Backward recursion[2] at time t:

$$H_{\mathbf{x}}=l_{\mathbf{x}}+\mathbf{f}_{\mathbf{x}}{}^{T}V_{\mathbf{x}}^{+} \qquad L=C^{-1}B$$
$$H_{\mathbf{u}}=l_{\mathbf{u}}+\mathbf{f}_{\mathbf{u}}{}^{T}V_{\mathbf{x}}^{+} \qquad V_{\mathbf{x}}=H_{\mathbf{x}}-L^{T}H_{\mathbf{u}}$$
$$A=l_{\mathbf{xx}}+\mathbf{f}_{\mathbf{x}}^{T}V_{\mathbf{xx}}^{+}\mathbf{f}_{\mathbf{x}}+\mathbf{f}_{\mathbf{xx}}^{T}V_{\mathbf{x}}^{+} \qquad V_{\mathbf{xx}}=A-L^{T}B$$
$$B=l_{\mathbf{ux}}+\mathbf{f}_{\mathbf{u}}^{T}V_{\mathbf{xx}}^{+}\mathbf{f}_{\mathbf{x}}+\mathbf{f}_{\mathbf{ux}}^{T}V_{\mathbf{x}}^{+}$$
$$C=l_{\mathbf{uu}}+\mathbf{f}_{\mathbf{u}}^{T}V_{\mathbf{xx}}^{+}\mathbf{f}_{\mathbf{u}}+\mathbf{f}_{\mathbf{uu}}^{T}V_{\mathbf{x}}^{+}$$

- Forward recursion:

$$\mathbf{u}^{new}=\mathbf{u}-C^{-1}H_{\mathbf{u}}-L(\mathbf{x}^{new}-\mathbf{x})$$
$$\mathbf{x}^{new}(t+\Delta t)=\mathbf{f}(x^{new}(t),\mathbf{u}^{new}(t))$$

[2] The $^{+}$ superscipt denotes quantities at time $t+\Delta t$. The tensor product $\mathbf{f}_{\mathbf{xx}}^{T}V_{\mathbf{x}}$ stands for $\sum_i (\mathbf{f}^i)_{\mathbf{xx}} V_{\mathbf{x}}^i$.

Note that in this section the dynamics has been redefined, and encodes the new state after time Δt, as opposed to the change-in-state notation used above.

IV. APPLICATIONS TO BIOMECHANICAL CONTROL

A. Simplified arm model

We implelented a 2-link arm model moving in the horizontal plane, with shoulder angle θ_1 and elbow angle θ_2. For now the control signals are the joint torques τ_1,τ_2:

$$\ddot{\boldsymbol{\theta}}=I(\boldsymbol{\theta})^{-1}\left(\boldsymbol{\tau}-\mathbf{c}(\boldsymbol{\theta},\dot{\boldsymbol{\theta}})\right)$$
$$I=\begin{bmatrix} i_1+i_2+2a\cos\theta_2+m_2l_1^2 & i_2+a\cos\theta_2 \\ i_2+a\cos\theta_2 & i_2 \end{bmatrix}$$
$$\mathbf{c}=\begin{bmatrix} -a\dot{\theta}_2\sin\theta_2\left(2\dot{\theta}_1+\dot{\theta}_2\right) \\ a\dot{\theta}_1^2\sin\theta_2 \end{bmatrix}+B\dot{\boldsymbol{\theta}}$$
$$a=m_2l_1s_2$$

We used estimated parameters for the human arm, with joint viscosity matrix B = [0.1 0.05; 0.05 0.1] kg m^2/ s.

	length l	center s	mass m	inertia i
upper arm	0.3 m	0.11 m	1.4 kg	0.025 kg m^2
forearm	0.33 m	0.16 m	1.1 kg	0.045 kg m^2

The arm was initialized in one of 3 possible initial postures, and the task was to move in 0.5 sec to one of 3 possible final postures (see **Fig 1**). The final cost was $h=\left\|\boldsymbol{\theta}(T)-\boldsymbol{\theta}^*\right\|^2$ and the cost rate was $l=0.0001\left\|\mathbf{u}(t)\right\|^2$.

B. Optimal trajectories

Fig 1 shows the optimal paths of the hand (the inset shows the same paths in joint space). Note that the cartesian paths are less curved than the joint paths, although the minimization problem and dynamics are defined in joint space. **Fig 2** shows how the cost of the current trajectory decreases with the number of gradient descent iterations; costs for 50 randmly initialized individual runs are also shown (thin lines). The conjugate gradient descent and steepest descent used exactly the same line search method, the only difference was that in conjugate gradient descent the direction of the gradient was corrected according to the Polack-Ribiere formula. **Fig 3** shows that, although the trajectory-based methods are not guaranteed to find the global minimum, the conjugate gradient algorithm is surprisingly robust and finds the same solution for a very wide range of (bad) initial guesses. The four plots show how the cloud of 50 randomly initialized trajectories gradually converges to a single trajectory.

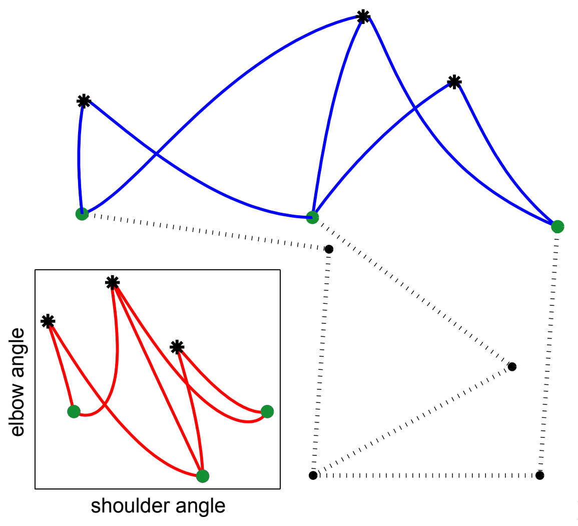

Fig 1

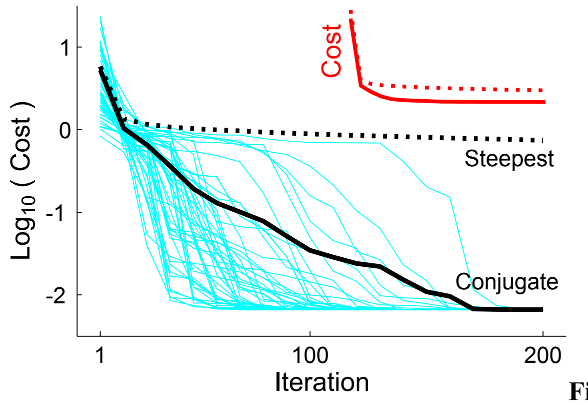

Fig 2

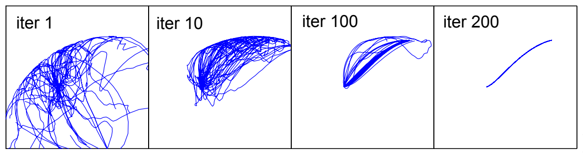

Fig 3

The ODE and DDP methods were not as robust with respect to initialization, but exhibited much faster convergence on reasonable initial conditions. Initializing the control trajectory to $\mathbf{u}(t) = 0$ and averaging over all combinations of start and end postures, we obtained the following times:

Method	CG	ODE	DDP
CPU time	5.9 sec	1.25 sec	0.61 sec

Thus all 3 methods are very efficient, although the code is currently in MATLAB. We also found that when ODE and DDP are initialized with a trajectory produced by a few iterations of CG, they always converged rapidly (much faster than CG). This suggests a mixed method, starting with a few iterations of CG and continuing with DDP.

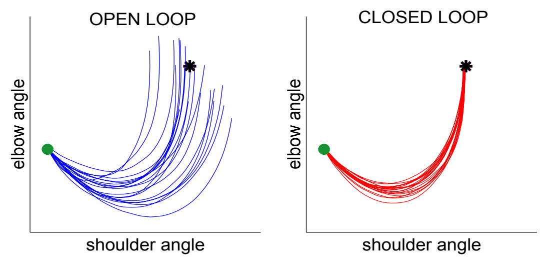

Fig 4

One complication with DDP is that the Newton step it uses $\left(C^{-1}H_{\mathbf{u}}\right)$ only makes sense when the matrix C is positive definite – which is not generally true. We therefore introduced a Levenberg-Marquardt modification: replace C^{-1} with $\left(C+\lambda I\right)^{-1}$, start with $\lambda = 0.01$, update $\lambda \leftarrow 10\lambda$ when the change prescribed by the algorithm increases the cost J, and update $\lambda \leftarrow \lambda/10$ otherwise. The update $\lambda \leftarrow 10\lambda$ was also used when $C+\lambda I$ was near singular (RCOND<0.01). This modification made the DDP algorithm much more stable, without obviously slowing it down.

C. Closed loop control under control-dependent noise

We now consider stochastic dynamics of the form:

$$d\mathbf{x}(t) = \mathbf{f}(t,\mathbf{x},\mathbf{u})\,dt + F(t,\mathbf{x},\mathbf{u})\,d\boldsymbol{\omega}; \quad \mathbf{x}(0) = \mathbf{x}_0$$

where $\boldsymbol{\omega}$ is standard Brownian motion. For such systems the optimal value function satisfies the modified Hamilton-Jacobi-Bellman (HJB) equation:

$$-V_t = l + \mathbf{f}^T V_{\mathbf{x}} + tr\left(FF^T V_{\mathbf{xx}}\right)/2\,.$$

Of special interest here is the case of multiplicative control noise, which is known to be a general property of biological systems. It can be modeled as $F(\mathbf{u})\,d\boldsymbol{\omega} = \sum_i S_i \mathbf{u}\, d\omega_i$.

For this noise model, the extra term in the HJB equation is

$$tr\left(FF^T V_{\mathbf{xx}}\right)/2 = \mathbf{u}^T\left(\sum_i S_i^{\;T} V_{\mathbf{xx}} S_i\right)\mathbf{u}/2$$

Because this is quadratic in $\mathbf{u}$, it is possible to re-derive the DDP algorithm. The only change is in the update for C:

$$C = l_{\mathbf{uu}} + \mathbf{f}_{\mathbf{u}}^T V_{\mathbf{xx}}^+ \mathbf{f}_{\mathbf{u}} + \mathbf{f}_{\mathbf{uu}}^T V_{\mathbf{x}}^+ + \sum_i S_i^{\;T} V_{\mathbf{xx}}^+ S_i$$

We used $S_1 = \mathbf{f}_{\mathbf{u}}$ [0.5 0; 0 0.5], $S_2 = \mathbf{f}_{\mathbf{u}}$ [0 -0.25; 0.25 0], and found that the run time is almost the same (0.73 sec vs. 0.61 sec). **Fig 4** shows the behavior of the stochastic system using the open loop control trajectory found by DDP, as well as the behavior under closed loop control (given by L). It is interesting to note that the locally optimal feedback controller only forces the trajectories to converge near the target, but allows them to deviate earlier in the movement – in agreement with the minimal intervention principle we have derived recently.