Newton-Type Algorithms for Dynamics-Based Robot Movement Optimization

Sung-Hee Lee, Junggon Kim, F. C. Park, Munsang Kim, and James E. Bobrow

***Abstract*—This paper describes Newton and quasi-Newton optimization algorithms for dynamics-based robot movement generation. The robots that we consider are modeled as rigid multibody systems containing multiple closed loops, active and passive joints, and redundant actuators and sensors. While one can, in principle, always derive in analytic form the equations of motion for such systems, the ensuing complexity, both numeric and symbolic, of the equations makes classical optimization-based movement-generation schemes impractical for all but the simplest of systems. In particular, numerically approximating the gradient and Hessian often leads to ill-conditioning and poor convergence behavior. We show in this paper that, by extending (to the general class of systems described above) a Lie theoretic formulation of the equations of motion originally developed for serial chains, it is possible to recursively evaluate the dynamic equations, the analytic gradient, and even the Hessian for a number of physically plausible objective functions. We show through several case studies that, with exact gradient and Hessian information, descent-based optimization methods can be forged into an effective and reliable tool for generating physically natural robot movements.**

***Index Terms*—Closed chain, movement optimization, multibody system dynamics, Newton's method, redundant actuation, robot dynamics.**

I. Introduction

AMONG the many innate physical abilities of humans, motor control is the skill that is most often taken for granted, as it seems to require very little conscious effort on our part. Only when a particular motor skill is impaired or lost does one then begin to fully appreciate the difficulty of motor control. It comes as no surprise that these exact same difficulties are encountered, indeed, even magnified, when attempting to endow robots with a movement-generation capability like that of humans.

The broad aim of this paper is to emulate the low-level capabilities of human motor coordination and learning within the framework of optimal control theory. Our approach is based on the simple observation that, in nearly all of the motor learning scenarios that we have observed, some form of optimization with respect to a physical criterion is taking place.

There is ample biological evidence to justify an optimization-based approach to movement generation. In the literature, one can find many optimal control-based studies of various human motions, e.g., maximum-height jumping [1], voluntary arm movements [2], maintaining postural balance [3], minimum-time running and swimming [4], and even wheelchair propelling [5]. Besides some of the more obvious optimization criteria like minimum energy or control effort, strategies that involve minimizing the derivative of acceleration (or jerk) [6], as well as muscle or metabolic energy costs [7], have also been examined in the context of specific arm motions. Models for human motor learning and control that take into account both the muscle dynamics and features of the neural system have been proposed, e.g., in [8]–[10].

Some researchers have also presented biological evidence suggesting that the nervous system implicitly performs inverse dynamics to generate feedforward motor commands [11], particularly for fast motions. Previous research also shows that it is possible to identify accurate internal models from movement data, and that such strategies can be successfully implemented in robots (see [12] and the references therein). Approaches to motor coordination and learning based on equilibrium and hierarchical approaches inspired by biological systems [13], [14] and dynamical systems theory [15] have also been presented.

From an engineering perspective, an optimization-based approach to movement generation usually strikes one as the first reasonable thing to try. The reason that such approaches have been largely unsuccessful, it seems, is that the complexity of the dynamic equations inevitably lead to intractable optimization problems. Indeed, the intractability of the optimization seems at least partly, if not largely, responsible for the recent flurry of attention given to, e.g., neural networks, genetic algorithms, and other evolutionary optimization approaches to motor learning (see, e.g., [16]–[18]).

One of the arguments put forth in this paper is that movement generation based on dynamic models and classical descent-type optimization methods is indeed a computationally feasible paradigm. In addition to our work, [19] has also demonstrated the feasibility of this approach to generate motions for tree-structure-like animated characters, by a suitable choice of physics-based constraints and objective functions. Stable open-loop motions for performing forward somersaults have also been obtained via numerical solution of an optimal control problem in [20].

Aside from the complexity of the nonlinear dynamics, another reason classical descent methods, despite their reliability

Manuscript received September 28, 2003; revised May 4, 2004 and August 30, 2004. This paper was recommended for publication by Associate Editor K. Lynch and Editor I. Walker upon evaluation of the reviewers' comments.

S.-H. Lee is with the Department of Computer Science, New York University, New York, NY 10003 USA (e-mail: shl283@nyu.edu).

J. Kim and F. C. Park are with the School of Mechanical and Aerospace Engineering, Seoul National University, Seoul 151-742, Korea (e-mail: junggon@robotics.snu.ac.kr; fcp@plaza.snu.ac.kr).

M. Kim is with the Center for Intelligent Robotics, Korea Institute of Science and Technology, Seoul, Korea (e-mail: munsang@kist.re.kr).

J. E. Bobrow is with the Department of Mechanical and Aerospace Engineering, University of California, Irvine, CA 92697 USA (e-mail: jebobrow@uci.edu).

Digital Object Identifier 10.1109/TRO.2004.842336

(indeed, in many cases, these algorithms are the only ones that can guarantee local optimality and convergence), are bypassed in many of today's motion-learning schemes is their reliance on gradient and Hessian information. Although, in principle, one can numerically approximate these quantities, for problems involving even moderately complex multibody systems, approximated gradients and Hessians more often than not lead to ill-conditioning, instability, and poor convergence behavior, not to mention a significant increase in computation.

One of the primary contributions of this paper is that, by appealing to techniques from the theory of Lie groups, it is possible to formulate the equations of motion of even complicated antagonistic multibody systems, like the human body, in such a way as to render the optimization problem tractable. In many cases, the optimized motions can even be obtained quite efficiently and in a numerically robust way. The key lies in the ability to recursively evaluate the nonlinear dynamics, and also to recursively compute exact analytic gradients and Hessians without resorting to numerical approximations. The resulting algorithms are still computation-intensive by today's standards, but are $O(n)$ with respect to the number of rigid bodies comprising the system, and perhaps most important of all, robust.

We begin by describing the dynamic modeling and optimization algorithms developed using techniques from Lie group theory. Some of the preliminary results specific to serial chains have been reported in [21], including the recursive computation of analytic gradients for serial chains [22], [23]. In this paper, we considerably enlarge the class of candidate mechanisms, to chains containing multiple closed loops and an arbitrary number of actuators; this includes antagonistic, redundantly actuated systems like the human body. We also develop general recursive algorithms for obtaining higher order derivatives of the dynamics for this class of chains.

Based on these new algorithms, recursive Newton-type optimization algorithms are then developed for generating optimal movements. As is well known, Newton methods have quadratic convergence properties, and offer superior performance over purely gradient-based optimization methods like steepest descent. Examples of minimum-effort motions for various multibody systems are provided to demonstrate that these algorithms can serve as a basis for generating efficient, physically natural movements in a robust, computationally effective manner.

II. Problem Statement

The equations of motion for our systems, which are modeled as a set of coupled rigid bodies, are of the form

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + N(q,\dot{q}) = \tau \tag{1}$$

where $M(q) \in \Re^{n\times n}$ is the mass matrix, $C(q,\dot{q}) \in \Re^{n\times n}$ is the Coriolis matrix, and $N(q,\dot{q}) \in \Re^{n}$ includes gravity and other forces (for the moment, we consider only holonomic systems in which the equations of motion are expressed in independent coordinates). One should not be deceived by the apparent simplicity of (1); for even kinematically straightforward structures like standard six-axis industrial robots, analytic expressions for $M(q), C(q,\dot{q})$, and $N(q)$ are extremely complicated.

We will be interested in minimizing cost functionals of the form

$$J(\tau) = \Phi(q,\dot{q},t_f) + \int_0^{t_f} L(q,\dot{q},\tau,t)\,dt \tag{2}$$

subject to (1) and the boundary conditions

$$q(0) = q_0 \quad \dot{q}(0) = 0 \tag{3}$$

$$q(t_f) = q_f \quad \dot{q}(t_f) = 0 \tag{4}$$

where, for some of our examples, Φ penalizes deviations from the desired final condition. For most of our examples, the effort $L = \|\tau\|^2$ captures the desire to minimize the exerted joint torques. The final time t_f may be either free or fixed in our formulation; for the examples presented in this paper, t_f is assumed fixed. For the case of free final time, it is possible to modify the equations developed below to include a time-scale parameter, as shown in [24]. We also note that the Newton method developed in this work applies to systems with no hard constraints on actuator torques or workspace motion. Soft constraints can be straightforwardly added to the problem by adding penalty functions to handle motor torque limitations; again, see [24].

Numerical methods for solving optimal control problems of the above form can be classified into direct and indirect methods. Indirect methods attempt to solve the optimality conditions, which are expressed in terms of the maximum principle [25], the adjoint equations, and the tranversality (boundary) conditions. Various relaxation and shooting methods have been developed to solve the associated two-point boundary value problem.

Despite the well-documented successes of the indirect approach, they have been displaced in recent years by direct methods. The primary reasons are that the region of convergence for indirect methods is relatively small, and it is difficult to incorporate path inequality constraints. For direct methods, the state and control variables are adjusted directly to optimize the objective function. In this paper, we focus exclusively on the direct approach; the reader is referred to, e.g., [26] and [27] for a survey of developments in numerical optimal control since the 1960s.

For our approach, a local solution to the optimal control problem is found by assuming that the joint coordinates $q(t)$ in (1) are parameterized by B-splines and varying these parameters in the following manner. The B-spline curve depends on the blending or basis functions $B_i(t)$ and the control points $P = \{p_1,\ldots,p_m\}$, with $p_i \in \Re^n$. The joint trajectories then have the form $q = q(t,P)$ with

$$q(t,P) = \sum_{i=1}^{m} B_i(t)p_i. \tag{5}$$

The control points p_i of the spline have only a local effect on the curve geometry, so given any t, there will a maximum of four nonzero $B_i(t)$ in (5) for a cubic spline. In addition, the convex hull property of B-splines makes them useful for smoothing or approximating data. The fact that $\sum_{i=1}^{m} B_i(t) = 1$ also gives the desirable property that limits on joint displacements can be imposed through limits on the spline parameters p_i. That is, if one constrains $p_i \le \bar{q}$, then it follows that $q(t) \le \bar{q}\ \forall t \in [0,t_f]$.

It should also be noted that the use of B-spline polynomials as the basic primitives in terms of which our motions are expressed is consistent with recent results in neuroscience. In [28], it was clinically observed that when human subjects move their hand in a circular motion, the trajectory obtained can be best described as a summation of "bell-shaped" basis functions. These functions are then translated and scaled to find the best match to the human movement. In essence, we achieve the same basic effect through (5).

By parametrizing the trajectory in terms of B-splines, the original optimal control reduces to a parameter optimization problem of the form

$$\underset{P}{\text{Minimize}}\ J(P) = \Phi(P, t_f) + \int_0^{t_f} L(P, t)\, dt \tag{6}$$

$$\text{subject to } \underline{q} \le p_i \le \bar{q}, \quad i = 1, \ldots, m \tag{7}$$

where $\tau = \tau(P, t); q, \dot{q}$, and $\ddot{q}$ are all given functions of t and P from (5) and its time derivatives, and τ becomes an explicit function of the spline parameters through (1). By a proper choice of spline basis functions at both ends of the joint trajectory, the path end conditions (3) and (4) can be satisfied.

By converting the original problem into a parameter optimization problem with no nonlinear constraints, efficient descent methods can then be used to minimize the cost function. However, to assure convergence of these algorithms, two conditions must be met: the second derivatives of $J(P)$ must be bounded, and every approximate Hessian (found, for example, from a BFGS update [29]) used in a quasi-Newton algorithm must remain positive definite with bounded condition number [30]. When one uses finite difference approximations for the gradient of $J(P)$, it is usually not possible to ensure a bounded condition number for the approximated Hessian; most often, the result is that the algorithms terminate prematurely [31].

We note that the gradient and Hessian of the cost functional are given by

$$\nabla J(P) = \int_0^{t_f} \tau \cdot (\nabla_P \tau)\, dt \tag{8}$$

$$\nabla^2 J(P) = \int_0^{t_f} \nabla_P^2 \tau \cdot \tau + \nabla_P \tau \cdot \nabla_P \tau\, dt. \tag{9}$$

The most computationally difficult step in the evaluation of the gradient and Hessian is determining the derivatives of the joint torques with respect to the path parameters P. These evaluations require that the dynamic equations be differentiated twice with respect to the joint variables. In a later section, we show how to compute these derivatives recursively for the general class of multibody systems addressed in this paper.

III. Recursive Dynamics

We now present the recursive dynamics formulation for multibody systems based on Lie group techniques. We begin with a brief review of serial chain dynamics, as presented in [21], followed by the dynamics of exactly actuated closed chains, as presented in [32]. We then present a recursive formulation of the dynamics of redundantly actuated closed chains.

A. Geometric Preliminaries

We begin with some geometric preliminaries. The Special Euclidean group of rigid body motions, denoted $\mathrm{SE}(3)$, is represented by matrices of the form

$$G = \begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix} \in \mathrm{SE}(3) \tag{10}$$

where $R \in \mathrm{SO}(3)$ is a 3×3 rotation matrix, and $p \in \Re^3$. Elements of $\mathrm{SE}(3)$ will also be denoted by the pair (R, p). The corresponding Lie algebra $\mathrm{se}(3)$ then admits the matrix representation

$$g = \begin{bmatrix} [\omega] & v \\ 0 & 0 \end{bmatrix} \quad [\omega] = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \tag{11}$$

where $v \in \Re^3$. Elements of $\mathrm{se}(3)$ will alternatively be denoted by (ω, v) for notational convenience. Later, ω and v will be interpreted physically as an angular velocity and linear velocity, respectively. In this case, we refer to (ω, v) as a generalized velocity or twist.

The exponential map $\exp : \mathrm{se}(3) \to \mathrm{SE}(3)$ can be computed by the following closed-form formula. If $\omega = \hat{\omega}\phi$, where $\|\hat{\omega}\| = 1$ and $\phi \in \mathbb{R}$, then

$$\exp\left(\begin{bmatrix} [\hat{\omega}] & v \\ 0 & 0 \end{bmatrix} \phi \right) = \begin{bmatrix} e^{[\hat{\omega}]\phi} & p \\ 0 & 1 \end{bmatrix} \tag{12}$$

where

$$p = (I\phi + (1 - \cos\phi)[\hat{\omega}] + (\phi - \sin\phi)[\hat{\omega}]^2)v \tag{13}$$

$$e^{[\hat{\omega}]\phi} = I + \sin\phi[\hat{\omega}] + (1 - \cos\phi)[\hat{\omega}]^2. \tag{14}$$

Given an element $G \in \mathrm{SE}(3)$, we also recall the adjoint mapping $\mathrm{Ad}_G : \mathrm{se}(3) \to \mathrm{se}(3)$, defined as $\mathrm{Ad}_G(g) = GgG^{-1}$. Given an element $g_1 \in \mathrm{se}(3)$, the Lie bracket $\mathrm{ad}_{g_1} : \mathrm{se}(3) \to \mathrm{se}(3)$ is given by $\mathrm{ad}_{g_1}(g_2) = g_1 g_2 - g_2 g_1$. The two adjoint mappings can also be represented in matrix form by

$$\mathrm{Ad}_G(g) = \begin{bmatrix} R & 0 \\ [p]R & R \end{bmatrix} \begin{pmatrix} \omega \\ v \end{pmatrix} \tag{15}$$

$$\mathrm{ad}_{g_1}(g_2) = \begin{bmatrix} [\omega_1] & 0 \\ [v_1] & [\omega_1] \end{bmatrix} \begin{pmatrix} \omega_2 \\ v_2 \end{pmatrix} \tag{16}$$

where $G = (R, p) \in \mathrm{SE}(3)$ and $g_i = (\omega_i, v_i), i = 1, 2$. The corresponding dual adjoint mappings $\mathrm{Ad}_G^* : \mathrm{se}^*(3) \to \mathrm{se}^*(3)$ and $\mathrm{ad}_g^* : \mathrm{se}^*(3) \to \mathrm{se}^*(3)$ can be represented in matrix form as

$$\mathrm{Ad}_G^*(\eta) = \begin{bmatrix} R & 0 \\ [p]R & R \end{bmatrix}^T \begin{pmatrix} m \\ f \end{pmatrix} \tag{17}$$

$$\mathrm{ad}_g^*(\eta) = \begin{bmatrix} [\omega] & 0 \\ [v] & [\omega] \end{bmatrix}^T \begin{pmatrix} m \\ f \end{pmatrix} \tag{18}$$

where $\eta \in \mathrm{se}^*(3)$ has the form

$$\eta = \begin{bmatrix} [m] & f \\ 0 & 0 \end{bmatrix} \tag{19}$$

with $m \in \mathbb{R}$, $f \in \mathbb{R}$. In what follows, m and f will be interpreted physically as a moment and force, respectively. In this case, we refer to (m, f) as a generalized force or wrench.

B. Serial Chains

Given an n-link serial chain, let $f_{k-1,k} \in \mathrm{SE}(3)$ be the transformation from link frame $k-1$ to link frame k, $S_k \in \mathrm{se}(3)$ the joint twist associated with joint k, $M_k \in \mathrm{SE}(3)$ the value of $f_{k-1,k}$ when joint $q_k = 0$, $V_k \in \mathrm{se}(3)$ the six-dimensional (6-D) generalized velocity of the link $\{\mathrm{k}\}$ frame expressed in frame $\{\mathrm{k}\}$, $\mathrm{F_k}$ the 6-D generalized force exerted by link $\{\mathrm{k}-1\}$ on link $\{\mathrm{k}\}$ expressed in frame $\{\mathrm{k}\}$, $\mathrm{J_k}$ the 6×6 generalized inertia matrix of link $\{\mathrm{k}\}$, and τ_k the torque exerted at joint $\{\mathrm{k}\}$.[1] J_k has the following structure:

$$J_k = \begin{bmatrix} I_k - m_k[r_k]^2 & m_k[r_k] \\ -m_k[r_k] & m_k \mathrm{I} \end{bmatrix} \tag{20}$$

where $I_k \in \Re^{3\times 3}$ is the rotational inertia of link $\{\mathrm{k}\}$ about the center of mass,[2] r_k is the 3-D vector from the frame $\{\mathrm{k}\}$ origin to the center of mass, m_k is the mass of link $\{\mathrm{k}\}$, and I is the 3×3 identity matrix. With these definitions, the inverse dynamics can be computed recursively as follows.

- **Initialization**

$$V_0, \dot{V}_0, F_{n+1}.$$

- **Forward recursion: for $\mathrm{k} = 1$ to n**

$$f_{k-1,k} = M_k e^{S_k q_k} \tag{21}$$

$$V_k = \mathrm{Ad}_{f_{k-1,k}^{-1}} V_{k-1} + S_k \dot{q}_k \tag{22}$$

$$\dot{V}_k = \mathrm{Ad}_{f_{k-1,k}^{-1}} \dot{V}_{k-1} + S_k \ddot{q}_k + \mathrm{ad}_{V_k} S_k \dot{q}_k. \tag{23}$$

- **Backward recursion: for $\mathrm{k} = \mathrm{n}$ to 1**

$$F_k = \mathrm{Ad}^*_{f_{k,k+1}^{-1}} F_{k+1} + J_k \dot{V}_k - \mathrm{ad}^*_{V_k} J_k V_k \tag{24}$$

$$\tau_k = S_k^T F_k. \tag{25}$$

A recursive formulation of serial chain forward dynamics based on the same geometric concept is given in [33], the details of which we omit for space reasons.

C. Exactly Actuated Closed Chains

The traditional approach to solving the inverse dynamics of closed chains is to first solve the inverse dynamics of the reduced system, followed by an application of D'Alembert's Principle to solve for the actuated joint torques of the closed-chain mechanism. By adopting the Lie theoretic framework, we can exploit the recursive algorithms for the serial-chain case, and derive a similar set of recursive algorithms for exactly evaluating the closed-chain dynamics [32].

To illustrate the approach, let us consider the mechanism of Fig. 1. The motion of this mechanism is generated by the actuator at joint 1 and the external applied force F_{at}. We can think of another identical mechanism that generates the same motion,

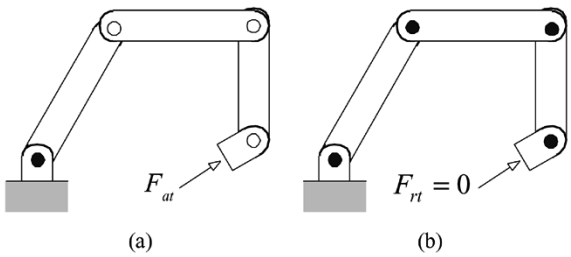


Fig. 1. Reduced system. (a) Actual system. (b) Reduced system.

but $F_{rt} = 0$ and every joint is assumed actuated;[3] this is essentially the concept of the reduced system. Note that the internal forces in the reduced system are different from those of the actual system.

Since the actual system and the reduced system produce the same motion, the work performed by each system is the same. Let $\dot{W}_a$ and $\dot{W}_r$ be the work done by the actual and reduced systems, respectively. Likewise, let τ_a and τ_r be the torque vectors, and q_a and q_r be the actuated joint angle vectors of the actual and reduced systems, respectively. V_t is the generalized velocity of the tool frame. Then

$$\dot{W}_a = \tau_a^T \dot{q}_a - F_{at}^T V_t \tag{26}$$

$$\dot{W}_r = \tau_r^T \dot{q}_r = \tau_{ra}^T \dot{q}_a + \tau_{rp}^T \dot{q}_p \tag{27}$$

where q_r is partitioned into $q_r = (q_a, q_p)$, with q_p corresponding to the passive (unactuated) joint of the actual system. τ_r is partitioned into $\tau_r = (\tau_{ra}, \tau_{rp})$, with τ_{ra} the torque at joint q_a and τ_{rp} the torque at joint q_p. Using the fact that $V_t = J\dot{q} = J_a\dot{q}_a + J_p\dot{q}_p$[4] and (26), we get

$$\dot{W}_a = \left(\tau_a^T - F_{at}^T J_a\right) \dot{q}_a - F_{at}^T J_p \dot{q}_p. \tag{28}$$

Since $\dot{q}_a$ and $\dot{q}_p$ are independent, the following relations hold:

$$F_{at} = -J_p^{-T} \tau_{rp} \tag{29}$$

$$\tau_a = J_a^T F_{at} + \tau_{ra} \tag{30}$$

$$= \tau_{ra} - \left(J_p^{-1} J_a\right)^T \tau_{rp}. \tag{31}$$

Note that J_p must be nonsingular; if it is singular, the mechanism is in a configuration corresponding to an actuator singularity [34].

With the above, we are now ready to describe the dynamics algorithm for exactly actuated closed chains. For a closed-chain mechanism, $V_t = 0$ and F_{at} is the constraint force applied to the mechanism so as to satisfy the constraint equations. The inverse dynamics can be computed using the recursive inverse dynamics algorithm of the reduced system as follows.

- **Initialization**

$$V_0, \dot{V}_0, F_{rt} = 0.$$

- **Forward recursion: for $\mathrm{k} = 1$ to n**

$$f_{k-1,k} = M_k e^{S_k q_k} \tag{32}$$

$$V_k = \mathrm{Ad}_{f_{k-1,k}^{-1}} V_{k-1} + S_k \dot{q}_k \tag{33}$$

$$\dot{V}_k = \mathrm{Ad}_{f_{k-1,k}^{-1}} \dot{V}_{k-1} + S_k \ddot{q}_k + \mathrm{ad}_{V_k} S_k \dot{q}_k. \tag{34}$$

[1] F_{n+1} is the external force, e.g., tool force, applied to link $\{\mathrm{n}\}$ expressed in frame $\{\mathrm{n}+1\}$.

[2] I_k is calculated with respect to a frame attached at the center of mass, with the same orientation as frame $\{\mathrm{k}\}$.

[3] F_{at} is the external applied force of the actual system, while F_{rt} is that of the reduced system.

[4] $J = [J_a\, J_p]$ is the Jacobian matrix, where J_a is the Jacobian matrix of the actuated joint, and J_p that of the passive joint of the actual system.

- **Backward recursion: for $\mathbf{k = n}$ to 1**

$$F_{r,k} = \mathrm{Ad}^*_{f_{k,k+1}^{-1}} F_{r,k+1} + J_k \dot{V}_k - \mathrm{ad}^*_{V_k} J_k V_k \tag{35}$$

$$\tau_k = S_k^T F_{r,k}. \tag{36}$$

- **Solve constraint force**

$$F_{a,t} = -J_p^T \tau_{rp}. \tag{37}$$

- **Backward recursion: for $\mathbf{k = n}$ to 1**

$$F_{a,k} = F_{r,k} + \mathrm{Ad}^*_{f_{k,n+1}^{-1}} F_{a,t} \tag{38}$$

$$\tau_{a,k} = S_k^T F_{a,k}. \tag{39}$$

If one needs to solve only τ_a, one can use (31) without performing the last backward recursion. See [32] for a discussion of the recursive forward dynamics formulation for exactly actuated closed chains.

D. Redundantly Actuated Closed Chains

A system is redundantly actuated when the number of actuated joints is greater than the degrees of freedom of the mechanism. In this case, the inverse dynamics will, in general, not have a unique solution. Equations (26) and (27) are still valid for the redundantly actuated case, however. Using $V_t = J\dot{q} = J_a \dot{q}_a + J_p \dot{q}_p$, we have

$$\tau_{rp} = -J_p^T F_c \tag{40}$$

$$\tau_{ra} = -J_a^T F_c + \tau_a \tag{41}$$

where F_{at} in (26) is replaced with F_c. Therefore

$$F_c = -\left(J_p^T\right)^{\dagger} \tau_{rp} + \mathrm{Null}\left(J_p^T\right) \lambda^T \tag{42}$$

$$\tau_a = \tau_{ra} - J_a^T \left(J_p^T\right)^{\dagger} \tau_{rp} + J_a^T \mathrm{Null}\left(J_p^T\right) \lambda^T \tag{43}$$

where λ is the Lagrange multiplier, $\mathrm{Null}(\cdot)$ is a null-space basis, and $(J_p^T)^{\dagger} = J_p (J_p^T J_p)^{-1}$.

We now determine the relationship between the torques of a redundantly actuated system and its corresponding exactly actuated system. Let the actuated joints and corresponding torques of the redundantly actuated system be q_{oa} and τ_{oa}, respectively, and the actuated joints and corresponding torques of the exactly actuated system be q_{ea} and τ_{ea}, respectively. Then, from D'Alembert's Principle, we have

$$\tau_{oa}^T \dot{q}_{oa} = \tau_{ea}^T \dot{q}_{ea}. \tag{44}$$

Now let us partition q_{oa} and τ_{oa} into $q_{oa} = (q_{ea}, q_v)$ and $\tau_{oa} = (\tau_u, \tau_v)$, where q_v is not actuated in the corresponding exactly actuated system, and its torque is τ_v. From $\dot{q}_{ep} = -J_p^{-1} J_a \dot{q}_{ea}$, let $\dot{q}_v = \phi \dot{q}_{ea}$ for some ϕ. Then

$$\tau_{oa}^T \dot{q}_{oa} = \tau_u^T \dot{q}_{ea} + \tau_v^T \phi \dot{q}_{ea} = \tau_{ea}^T \dot{q}_{ea} \tag{45}$$

from which it follows that

$$\tau_u + \phi^T \tau_v = \tau_{ea}. \tag{46}$$

Note that J_p is required to be nonsingular. In the event that it is singular, one can simply construct the corresponding exactly actuated system in such a way that J_p is nonsingular, by choosing the appropriate set of actuated joints. We will use the above relation in Section IV-B.

IV. Differentiation of the Equations of Motion

To obtain the gradient and Hessian of the objective function, it is necessary to differentiate the equations of motion with respect to the joint variables. In this section, we derive recursive algorithms to compute the gradient and Hessian for both open-loop and closed-loop systems.

A. Serial Chains

Let $q = q(p,t)$, where $t \in \mathbb{R}$ and $p \in \mathbb{R}^k$. Also let $S, V \in \mathrm{se}(3)$, $M \in \mathrm{SE}(3)$. Then the following identities can be established by a straightforward but involved calculation:

$$\frac{\partial}{\partial p} \mathrm{Ad}_{e^{Sq}M} = \frac{\partial q}{\partial p} \mathrm{ad}_S \, \mathrm{Ad}_{e^{Sq}M} \tag{47}$$

$$\frac{\partial}{\partial p} \mathrm{Ad}_{Me^{Sq}} = \frac{\partial q}{\partial p} \mathrm{Ad}_{Me^{Sq}} \mathrm{ad}_S \tag{48}$$

$$\frac{\partial}{\partial p} \mathrm{Ad}^*_{e^{Sq}M} = \frac{\partial q}{\partial p} \mathrm{Ad}^*_{e^{Sq}M} \mathrm{ad}^*_S \tag{49}$$

$$\frac{\partial}{\partial p} \mathrm{Ad}^*_{Me^{Sq}} = \frac{\partial q}{\partial p} \mathrm{ad}^*_S \, \mathrm{Ad}^*_{Me^{Sq}} \tag{50}$$

$$\frac{\partial}{\partial p} \mathrm{ad}_V = \mathrm{ad}_{\frac{\partial V}{\partial p}} \tag{51}$$

$$\frac{\partial}{\partial p} \mathrm{ad}^*_V = \mathrm{ad}^*_{\frac{\partial V}{\partial p}}. \tag{52}$$

An earlier, more complicated version of the above identities was also obtained in [22].

Using these results, we can differentiate the inverse dynamics of a serial chain with respect to the joint parameter vector p via the following algorithm.

- **Initialization**

$$\frac{\partial V_0}{\partial p_i}, \frac{\partial \dot{V}_0}{\partial p_i}, \frac{\partial F_{n+1}}{\partial p_i}.$$

- **Forward recursion: for $\mathbf{k = 1}$ to $\mathbf{n}$**

$$\frac{\partial V_k}{\partial p_i} = \mathrm{Ad}_{f_{k-1,k}^{-1}} \frac{\partial V_{k-1}}{\partial p_i} + S_k \frac{\partial \dot{q}_k}{\partial p_i} - \mathrm{ad}_{S_k} V_k \frac{\partial q_k}{\partial p_i} \tag{53}$$

$$\frac{\partial \dot{V}_k}{\partial p_i} = \mathrm{Ad}_{f_{k-1,k}^{-1}} \frac{\partial \dot{V}_{k-1}}{\partial p_i} + S_k \frac{\partial \ddot{q}_k}{\partial p_i} - \mathrm{ad}_{S_k} \left\{ \frac{\partial q_k}{\partial p_i} \mathrm{Ad}_{f_{k-1,k}^{-1}} \dot{V}_{k-1} + \frac{\partial V_k}{\partial p_i} \dot{q}_k + V_k \frac{\partial \dot{q}_k}{\partial p_i} \right\}. \tag{54}$$

- **Backward recursion: for $\mathbf{k = n}$ to 1**

$$\frac{\partial F_k}{\partial p_i} = \mathrm{Ad}^*_{f_{k,k+1}^{-1}} \left(\mathrm{ad}^*_{-S_{k+1}} F_{k+1} \frac{\partial q_{k+1}}{\partial p_i} + \frac{\partial F_{k+1}}{\partial p_i} \right) + J_k \frac{\partial \dot{V}_k}{\partial p_i} - \mathrm{ad}^*_{\frac{\partial V_k}{\partial p_i}} J_k V_k - \mathrm{ad}^*_{V_k} J_k \frac{\partial V_k}{\partial p_i} \tag{55}$$

$$\frac{\partial \tau_k}{\partial p_i} = S_k^T \frac{\partial F_k}{\partial p_i}. \tag{56}$$

By differentiating the above recursive gradient algorithm, the second derivatives of the dynamics can, in turn, be analytically computed via the following recursive algorithm.

- **Initialization**

$$\frac{\partial^2 V_0}{\partial p_i \partial p_j}, \frac{\partial^2 \dot{V}_0}{\partial p_i \partial p_j}, \frac{\partial^2 F_{n+1}}{\partial p_i \partial p_j}.$$

- **Forward recursion: for $\mathbf{k = 1}$ to $\mathbf{n}$**

$$\begin{aligned}\frac{\partial^2 V_k}{\partial p_i \partial p_j} &= \mathrm{Ad}_{f_{k-1,k}^{-1}} \frac{\partial^2 V_{k-1}}{\partial p_i \partial p_j} + S_k \frac{\partial^2 \dot{q}_k}{\partial p_i \partial p_j} \\ &\quad - \mathrm{ad}_{S_k} \left\{ \frac{\partial q_k}{\partial p_i} \mathrm{Ad}_{f_{k-1,k}^{-1}} \frac{\partial V_{k-1}}{\partial p_j} \right. \\ &\quad \left. + \frac{\partial V_k}{\partial p_i} \frac{\partial q_k}{\partial p_j} + V_k \frac{\partial^2 q_k}{\partial p_i \partial p_j} \right\} \qquad (57) \\ \frac{\partial^2 \dot{V}_k}{\partial p_i \partial p_j} &= \mathrm{Ad}_{f_{k-1,k}^{-1}} \frac{\partial^2 \dot{V}_{k-1}}{\partial p_i \partial p_j} + S_k \frac{\partial^2 \ddot{q}_k}{\partial p_i \partial p_j} \\ &\quad - \mathrm{ad}_{S_k} \left\{ \mathrm{Ad}_{f_{k-1,k}^{-1}} \left(\dot{V}_{k-1} \frac{\partial^2 q_k}{\partial p_i \partial p_j} \right. \right. \\ &\quad \left. + \frac{\partial \dot{V}_{k-1}}{\partial p_j} \frac{\partial q_k}{\partial p_i} + \frac{\partial \dot{V}_{k-1}}{\partial p_i} \frac{\partial q_k}{\partial p_j} \right) + V_k \frac{\partial^2 \dot{q}_k}{\partial p_i \partial p_j} \\ &\quad + \frac{\partial^2 V_k}{\partial p_i \partial p_j} \dot{q}_k + \frac{\partial V_k}{\partial p_i} \frac{\partial \dot{q}_k}{\partial p_j} + \frac{\partial V_k}{\partial p_j} \frac{\partial \dot{q}_k}{\partial p_i} \\ &\quad \left. - \mathrm{ad}_{S_k} \mathrm{Ad}_{f_{k-1,k}^{-1}} \dot{V}_{k-1} \frac{\partial q_k}{\partial p_i} \frac{\partial q_k}{\partial p_j} \right\}. \qquad (58)\end{aligned}$$

- **Backward recursion: for $\mathbf{k = n}$ to $\mathbf{1}$**

$$\begin{aligned}\frac{\partial^2 F_k}{\partial p_i \partial p_j} &= \mathrm{Ad}^*_{f_{k,k+1}^{-1}} \left\{ \mathrm{ad}^*_{-S_{k+1}} \left(F_{k+1} \frac{\partial^2 q_{k+1}}{\partial p_i \partial p_j} \right. \right. \\ &\quad + \mathrm{ad}^*_{-S_{k+1}} F_{k+1} \frac{\partial q_{k+1}}{\partial p_i} \frac{\partial q_{k+1}}{\partial p_j} + \frac{\partial F_{k+1}}{\partial p_j} \frac{\partial q_{k+1}}{\partial p_i} \\ &\quad \left. \left. + \frac{\partial F_{k+1}}{\partial p_i} \frac{\partial q_{k+1}}{\partial p_j} \right) + \frac{\partial^2 F_{k+1}}{\partial p_i \partial p_j} \right\} + J_k \frac{\partial^2 \dot{V}_k}{\partial p_i \partial p_j} \\ &\quad - \mathrm{ad}^*_{\frac{\partial^2 V_k}{\partial p_i \partial p_j}} J_k V_k - \mathrm{ad}^*_{\frac{\partial V_k}{\partial p_i}} J_k \frac{\partial V_k}{\partial p_j} \\ &\quad - \mathrm{ad}^*_{\frac{\partial V_k}{\partial p_j}} J_k \frac{\partial V_k}{\partial p_i} - \mathrm{ad}^*_{V_k} J_k \frac{\partial^2 V_k}{\partial p_i \partial p_j} \qquad (59) \\ \frac{\partial^2 \tau_k}{\partial p_i \partial p_j} &= S_k^T \frac{\partial^2 F_k}{\partial p_i \partial p_j}. \qquad (60)\end{aligned}$$

It should be noted that many of the computations embedded in the forward and backward recursions above need only be evaluated once, thereby reducing the computational burden.

B. Exactly Actuated Closed Chains

Recall that for an exactly actuated closed chain

$$\tau_a = \tau_{ra} - \left(J_p^{-1} J_a \right)^T \tau_{rp}. \qquad (61)$$

Differentiating (61) with respect to p using our earlier basic identities, we obtain[5]

$$\frac{\partial \tau_a}{\partial p_i} = \frac{\partial \tau_{ra}}{\partial p_i} - \frac{\partial J_a^T}{\partial p_i} J_p^{-T} \tau_{rp} - J_a^T \frac{\partial \left(J_p^{-T} \right)}{\partial p_i} \tau_{rp} - J_a^T J_p^{-T} \frac{\partial \tau_{rp}}{\partial p_i}. \qquad (62)$$

[5]Here we use the fact that $(\partial (J_p^{-T}))/(\partial p_i) = -J_p (\partial J_p^T)/(\partial p_i) J_p$.

Differentiating again yields

$$\begin{aligned}\frac{\partial^2 \tau_a}{\partial p_i \partial p_j} &= \frac{\partial^2 \tau_{ra}}{\partial p_i \partial p_j} - \frac{\partial^2 J_a^T}{\partial p_i \partial p_j} J_p^{-T} \tau_{rp} - \frac{\partial J_a^T}{\partial p_i} \frac{\partial \left(J_p^{-T} \right)}{\partial p_j} \tau_{rp} \\ &\quad - \frac{\partial J_a^T}{\partial p_i} J_p^{-T} \frac{\partial \tau_{rp}}{\partial p_j} - \frac{\partial J_a^T}{\partial p_j} \frac{\partial \left(J_p^{-T} \right)}{\partial p_i} \tau_{rp} \\ &\quad - J_a^T \frac{\partial^2 \left(J_p^{-T} \right)}{\partial p_i \partial p_j} \tau_{rp} - J_a^T \frac{\partial \left(J_p^{-T} \right)}{\partial p_i} \frac{\partial \tau_{rp}}{\partial p_j} \\ &\quad - \frac{\partial J_a^T}{\partial p_j} J_p^{-T} \frac{\partial \tau_{rp}}{\partial p_i} - J_a^T \frac{\partial \left(J_p^{-T} \right)}{\partial p_j} \frac{\partial \tau_{rp}}{\partial p_i} \\ &\quad - J_a^T J_p^{-T} \frac{\partial^2 \tau_{rp}}{\partial p_i \partial p_j}. \qquad (63)\end{aligned}$$

C. Redundantly Actuated Closed Chains

To differentiate the equations of motion for a redundantly actuated closed chain, we can differentiate either (43) or (44); here, we choose the latter, since this form is more convenient for torque optimization. Suppose we want to minimize a suitably weighted torque, e.g., suppose

$$\tau = W \tau_{oa} = \begin{bmatrix} W_u \tau_u \\ W_v \tau_v \end{bmatrix} \qquad (64)$$

where W is a suitable weighting matrix. We wish to minimize

$$\begin{aligned}\tau^T \tau &= \tau_{oa}^T W^T W \tau_{oa} \qquad (65) \\ &= \tau_u^T \hat{W}_u \tau_u + \tau_v^T \hat{W}_v \tau_v \qquad (66) \\ &= \tau_v^T (\phi \hat{W}_u \phi^T + \hat{W}_v) \tau_v - 2 \tau_v^T \phi \hat{W}_u \tau_{ea} \\ &\quad + \tau_{ea}^T \hat{W}_u \tau_{ea} \qquad (67)\end{aligned}$$

where $\hat{W}_u = W_u^T W_u$, and $\hat{W}_v = W_v^T W_v$; specifically, we seek the τ_u and τ_v that minimize $\tau^T \tau$. This leads to an unconstrained calculus of variations problem of the form

$$\min_{q,\mu} \int_{t_0}^{t_f} L(q, \dot{q}, \ddot{q}, \mu)\, dt \qquad (68)$$

where $\mu = \tau_v$ and $L = \tau^T \tau$ in our formulation. Noting that q and μ are independent, the first-order necessary conditions for optimality are

$$\begin{aligned}\frac{\partial L}{\partial q} - \frac{d}{dt} \frac{\partial L}{\partial \dot{q}} + \frac{d^2}{dt^2} \frac{\partial L}{\partial \ddot{q}} &= 0 \qquad (69) \\ \frac{\partial L}{\partial \mu} &= 0. \qquad (70)\end{aligned}$$

In particular, the last condition leads to

$$\frac{\partial \tau^T \tau}{\partial \tau_v} = 2(\phi \hat{W}_u \phi^T + \hat{W}_v) \tau_v - 2 \phi \hat{W}_u \tau_{ea} = 0 \qquad (71)$$

from which we obtain

$$\begin{aligned}\tau_v &= (\phi \hat{W}_u \phi^T + \hat{W}_v)^{-1} \phi \hat{W}_u \tau_{ea} \qquad (72) \\ \tau_u &= (I - \phi^T (\phi \hat{W}_u \phi^T + \hat{W}_v)^{-1} \phi \hat{W}_u) \tau_{ea}. \qquad (73)\end{aligned}$$

The derivative of the torque with respect to p_i is

$$\frac{\partial \tau}{\partial p_i} = W \frac{\partial \tau_{oa}}{\partial p_i} = W_u \frac{\partial \tau_u}{\partial p_i} + W_v \frac{\partial \tau_v}{\partial p_i}. \qquad (74)$$

Let $\varphi = (\phi \hat{W}_u \phi^T + \hat{W}_v)$. Then we obtain

$$\frac{\partial \tau_v}{\partial p_i} = \varphi^{-1}\left(-2\frac{\partial \phi}{\partial p_i}\hat{W}_u\phi^T\varphi^{-1}\phi\hat{W}_u\tau_{ea} + \frac{\partial \phi}{\partial p_i}\hat{W}_u\tau_{ea} + \phi\hat{W}_u\frac{\partial \tau_{ea}}{\partial p_i}\right) \tag{75}$$

$$\frac{\partial \tau_u}{\partial p_i} = \frac{\partial \tau_{ea}}{\partial p_i} - \frac{\partial \phi^T}{\partial p_i}\tau_v - \phi^T\frac{\partial \tau_v}{\partial p_i}. \tag{76}$$

Differentiating the above equations once again, we get

$$\begin{aligned}\frac{\partial^2 \tau_v}{\partial p_i \partial p_j} = \varphi^{-1}\Bigg(&-\frac{\partial \varphi}{\partial p_i}\frac{\partial \tau_v}{\partial p_j} - 2\frac{\partial^2 \phi}{\partial p_i \partial p_j}\hat{W}_u\phi^T\varphi^{-1}\phi\hat{W}_u\tau_{ea} \\ &- 4\frac{\partial \phi}{\partial p_j}\hat{W}_u\frac{\partial \phi}{\partial p_i}^T\varphi^{-1}\phi\hat{W}_u\tau_{ea} \\ &+ 2\frac{\partial \phi}{\partial p_j}\hat{W}_u\phi^T\varphi^{-1}\frac{\partial \phi}{\partial p_i}\varphi^{-1}\phi\hat{W}_u\tau_{ea} \\ &- 2\frac{\partial \phi}{\partial p_j}\hat{W}_u\frac{\partial \phi}{\partial p_i}^T\varphi^{-1}\phi\hat{W}_u\frac{\partial \tau_{ea}}{\partial p_i} \\ &+ \frac{\partial^2 \phi}{\partial p_i \partial p_j}\hat{W}_u\tau_{ea} + \frac{\partial \phi}{\partial p_j}\hat{W}_u\frac{\partial \tau_{ea}}{\partial p_i} \\ &+ \frac{\partial \phi}{\partial p_i}\hat{W}_u\frac{\partial \tau_{ea}}{\partial p_j} + \phi\hat{W}_u\frac{\partial^2 \tau_{ea}}{\partial p_i \partial p_j}\Bigg)\end{aligned} \tag{77}$$

$$\begin{aligned}\frac{\partial^2 \tau_u}{\partial p_i \partial p_j} = &\frac{\partial^2 \tau_{ea}}{\partial p_i \partial p_j} - \frac{\partial^2 \phi^T}{\partial p_i \partial p_j}\tau_v - \frac{\partial \phi^T}{\partial p_j}\frac{\partial \tau_v}{\partial p_i} \\ &- \frac{\partial \phi}{\partial p_i}^T\frac{\partial \tau_v}{\partial p_j} - \phi^T\frac{\partial^2 \tau_v}{\partial p_i \partial p_j}.\end{aligned} \tag{78}$$

The above formulas can be used to analytically compute the gradient and Hessian for the class of objective functions considered in this paper.

V. Algorithm for Generating Optimal Motions

In this section, we present the general iterative procedure for computing optimal motions of the various classes of kinematic chains. The algorithm is expressed in sufficiently general form so as to allow for different (unconstrained) optimization algorithms, although the primary ones we will be interested in are steepest descent and Newton-type methods.

The algorithm below describes the motion-optimization procedure for a serial-chain mechanism.

- **Given:** Initial and final values of the joint angle displacement, velocity, and acceleration.
- **Step 1:** Choose the number of control points and the order of the B-spline curve.
- **Step 2:** Make an initial assumption of the joint angle profiles.
- **Step 3:** For time t $= t_0$ to t_f:

— Compute the spline function to get the status of joint angle [(5)].
— Solve the inverse dynamics to determine the torque [(25)].
— Solve for the gradient and Hessian [(56) and (60)].

- **Step 4:** Compute the objective function, gradient, and Hessian [(6), (8), and (9)].
- **Step 5:** Check convergence. If yes, terminate. If no, line search for the next point and go to Step 3.

We next present the algorithm for generating optimal motions of a redundantly actuated closed chain; exactly actuated closed chains can be considered as a special case of this general case.

- **Given:** Initial and final values of the joint angle displacement, velocity, and acceleration for an independent set of joints.
- **Step 1:** Choose the number of control points and the order of the B-spline curve.
- **Step 2:** Make an initial assumption of the kinematically independent joint angle profiles.
- **Step 3:** For time t $= t_0$ to t_f:

— Compute the spline function to get the status of the joint angles [(5)].
— Solve for the remaining joint angle displacements, velocities, and accelerations.
— Solve for the inverse dynamics, gradient, and Hessian of the reduced system. [(25), (56), and (60)].
— Solve the inverse dynamics to determine the torque [(72) and (73)].
— Solve for the gradient and Hessian of the torque [(75), (76), (77), and (78)].

- **Step 4:** Compute the objective function, gradient and Hessian [(6), (8), and (9)].
- **Step 5:** Check convergence. If yes, terminate. If no, line search for the next point and go to Step 3.

Kinematic chains containing closed loops must satisfy a set of loop-closure constraint equations, and it would seem natural to attempt to apply a constrained optimization procedure rather than solving the constraint equations at each iteration. However, for the class of mechanisms of interest to us, the passive joint values can, in general, be determined from the values of the actuated joint values without too much difficulty; in this regard; there exist a number of procedures (i.e., the Paden–Kahan subproblems—see [35]) for solving in closed form the inverse kinematics of a large class of serial chains, as well as efficient and reliable numerical methods. In this case, it is often simpler and more efficient to directly solve the constraint equations and apply the unconstrained optimization methods above.

VI. Case Studies

In this section, we present case studies of optimal motions generated for a number of representative kinematic chains. All of the simulations were performed on a Pentium II 392-MHz computer, and any performance statistics given are with respect to this computer specification.

A. Two-Link Open Chain

We first consider the minimum torque lifting motion for a two-link planar open chain in the presence of gravity. The motion was obtained in 13 iterations, with a stopping criterion of

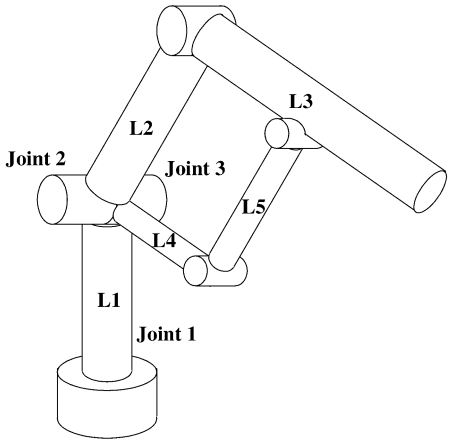


Fig. 2. Exactly actuated closed-loop manipulator.

$\|g_k\| < 10^{-2}$, and took 1.1 s.[6] The final value of the objective function was 314.069. Nine control points were used for each joint, together with a third-order curve.

To evaluate the performance of the modified Newton's algorithm, we optimize the motion using both steepest descent and the BFGS quasi-Newton method. In the case of steepest descent, the number of iterations was forcefully terminated after 229 after the algorithm failed to meet the stopping criterion. The final value of the objective function was 314.069. For the BFGS quasi-Newton method, the algorithm was forcefully terminated after 45 iterations, and convergence was extremely slow near the solution. The total elapsed time was 1.27 s, and the final value of the objective function 314.069. The BFGS method, on the other hand, approached the vicinity of the solution in the shortest time among the three methods.

B. Exactly Actuated Closed Chain

We now consider the exactly actuated closed chain of Fig. 2. Joints 1–3 are actuated, where joint 2 rotates both L_2 and L_4 together, and joint 3, lying coaxially with joint 2, rotates only L_4. The actuated joint angles in the initial pose are given by $(-30°, -30°, 120°)$, while in the final pose, the angles are $(30°, 10°, 70°)$. We seek the minimum torque motion such that the manipulator moves between two poses symmetrically situated about the workspace in exactly 1 s. For our test case, the Modified Newton method converged after seven iterations, with a total computation time of 6.94 s.

The convergence speed and number of iterations are compared for the steepest descent, Newton method, and the BFGS quasi-Newton method. Fig. 3 shows the number of iterations for each method, while Table I shows the computation time of each optimization method. As is evident from the figure, the steepest descent method failed to converge due to ill-conditioning, while both the BFGS quasi-Newton method and the modified Newton method showed good convergence. Although the number of iterations for the modified Newton method is smaller than that for the BFGS quasi-Newton method, for this example, the overall computation time is slightly greater due to the computation of the analytic Hessian at each step. In general, for complex examples, we have found that the total computation time increases approximately linearly with the number of parameters. The reason for this is that for Newton's method, the asymptotic convergence rate is quadratic and also independent of the number of parameters [29]. Hence, the computation time is dominated by the time needed to compute the objective function, gradient, and Hessian, all of which increase linearly with the number of parameters.

[6]Using the norm of the gradient normalized with respect to the objective function value does not significantly alter the convergence behavior in this or any of the other examples.

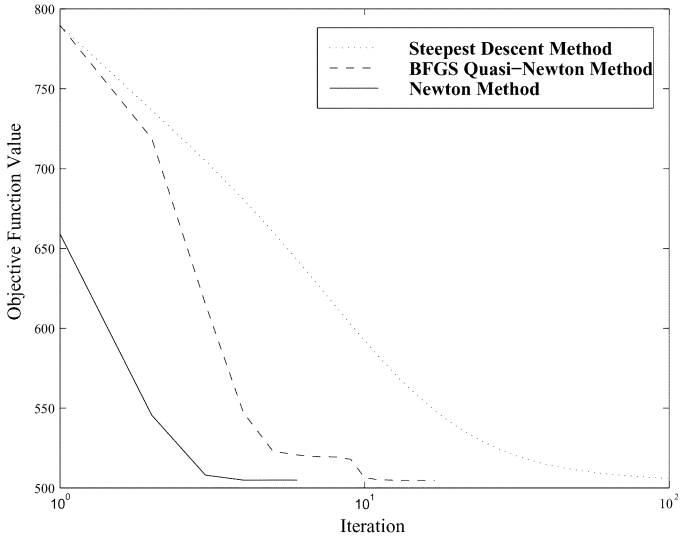


Fig. 3. Comparison of optimization methods.

TABLE I
NUMBER OF ITERATIONS AND TIME CONSUMED FOR EACH METHOD

Algorithm	Number of iterations	Time(sec)
Steepest Descent	200+	58.184
Modified Newton's	7	6.94
BFGS Quasi-Newton	18	4.086

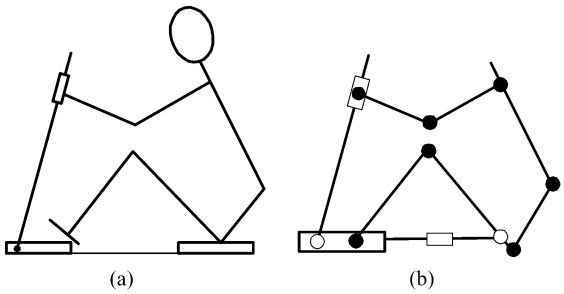


Fig. 4. Schematic picture of rowing. (a) Human on rowing machine. (b) Corresponding mechanism.

C. Redundantly Actuated Closed Chain: The Rower

As our final example, we consider the minimum-torque rowing motion of a human consisting of two closed loops. Fig. 4(b) shows the planar kinematic chain used to model the human. The mechanism has five kinematic degrees of freedom and seven actuated joints. In the figure, the circles represent rotational joints and rectangles represent prismatic joints; filled-in circles imply that the joint is actuated. The prismatic

TABLE II
MASSES AND MOMENTS OF INERTIA FOR THE HUMAN MODEL

Link	Mass(kg)	Moment of inertia(kgm^2)
Pelvis	16.61	(0.23, 0.18, 0.16)
Trunk	29.27	(0.73, 0.63, 0.32)
Arm	2.97	(0.025, 0.025, 0.005)
Forearm	1.21	(0.005, 0.0054, 0.0012)
Thigh	8.35	(0.15, 0.16, 0.025)
Shank	4.16	(0.055, 0.056, 0.007)

joints and the base link are not shown explicitly in the figure for visualization purposes.

A 40-N·m torque is applied clockwise at the joint to which the oar is attached. Table II shows the masses and rotational inertias of the various links, obtained from [36] to closely approximate the actual values for a typical human. The top row of Fig. 5 depicts the initial motion, obtained by linear interpolation of the joint values between the initial and final poses, while the optimized motion is shown in the bottom row.[7] It is interesting to note the similarity between the optimized motion and the actual rowing motion exerted by a human.

For this example, the optimized motion was obtained after 38 iterations using the BFGS quasi-Newton method, with a total computation time of 54.43 s. Ten control points were used for each joint trajectory, with a B-spline of order three for the interpolating curves. The same optimal motion was obtained with the modified Newton algorithm, but with a longer computation time. Our experience simulating a wide array of multibody systems indicates that, in general, the BFGS quasi-Newton method requires between 10%–50% less computation time than the modified Newton method.

VII. CONCLUSION

In this paper, we have presented an optimization-based methodology for motor learning that emulates the low-level capabilities of human motor coordination and learning. The systems we address include chains containing multiple closed loops and an arbitrary number of actuators; this includes antagonistic, redundantly actuated systems like the human body.

Previous classical optimization-based approaches to motor learning were limited in their effectiveness to kinematically simple, low degree-of-freedom systems; for even moderately complex systems, these algorithms typically led to ill-conditioning, instability, and poor convergence behavior because of their inability to deal with the complexity of the nonlinear dynamics, and their reliance on approximated gradient and Hessian information.

In this paper, we have shown that by appealing to techniques from the theory of Lie groups, both the equations of motion and gradient and Hessian information can be exactly and recursively computed for even complicated antagonistic multibody systems. The resulting algorithms are still computation-intensive, but are $O(n)$ with respect to the number of rigid bodies comprising the system. Examples of minimum-effort motions for various multibody systems demonstrate that these algorithms can serve as a basis for a robust, computationally effective, model-based motor learning capability.

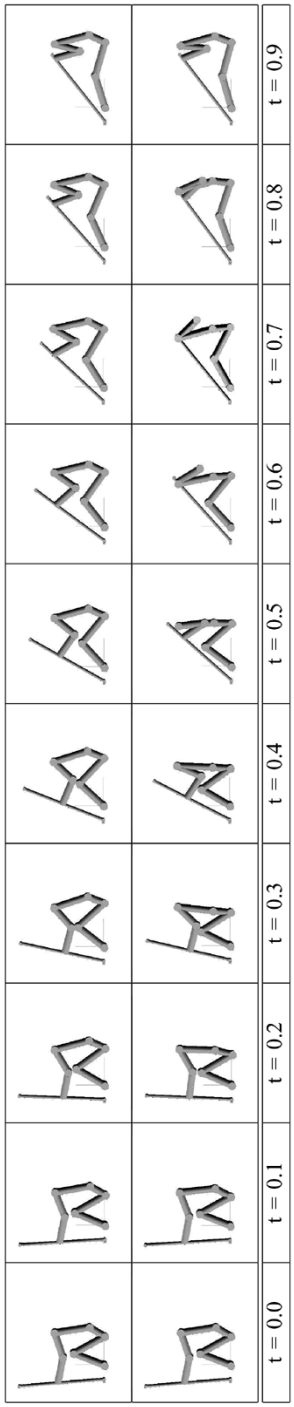


Fig. 5. Initial and optimized motions for rowing.

Our initial results suggest a number of topics for further study. First, as shown from our case studies, the number of control points and the order of the curve play an important role in both the computational efficiency and final shape of the optimized motions. From this point of view, B-spline wavelets are also worth considering in that the trajectory is represented hierarchically instead of in terms of a B-spline basis [37]. Other objective functions, e.g., minimum-time motions, also deserve further attention.

[7]The movie file can be downloaded from http://ieeexplore.ieee.org.

APPENDIX

We provide formulas for obtaining the derivatives of the constraint Jacobian (note that the Jacobian in space frame coordinates is used throughout). Let

$$J = [J_1 \dots J_n] \tag{79}$$

$$J_i = \mathrm{Ad}_{f_{0,i}} S_i. \tag{80}$$

Then

$$\frac{\partial J_l}{\partial q_i} = \begin{cases} J_i J_l - J_l J_i = [J_i, J_l], & \text{if } i < l \\ J_l J_i - J_l J_i = 0, & \text{if } i \geq l \end{cases} \tag{81}$$

$$\frac{\partial}{\partial q_i}\left(\frac{\partial J_l}{\partial q_j}\right) = \begin{cases} [\mathrm{ad}_{J_i} J_j, J_l] + [J_j, \mathrm{ad}_{J_i} J_l] \\ \quad = \mathrm{ad}_{\mathrm{ad}_{J_i} J_j} J_l + \mathrm{ad}_{J_j} \mathrm{ad}_{J_i} J_l, \\ \qquad \text{if} i < j < l \\ [J_j, \mathrm{ad}_{J_i} J_l] = \mathrm{ad}_{J_j} \mathrm{ad}_{J_i} J_l, \\ \qquad \text{if } j \leq i < l \\ 0, \qquad \text{elsewhere} \end{cases} \tag{82}$$

$$\dot{J}_i = \sum_{j=1}^{i-1} \mathrm{ad}_{J_j} J_i \dot{q}_j \tag{83}$$

$$\frac{\partial J_l}{\partial p_j} = \sum_{k=1}^{l-1} \frac{\partial q_k}{\partial p_j} \mathrm{ad}_{J_k} J_l \tag{84}$$

$$\begin{aligned} \frac{\partial^2 J_l}{\partial p_i \partial p_j} = \sum_{k=1}^{l-1} \Bigg\{ & \frac{\partial^2 q_k}{\partial p_i \partial p_j} \mathrm{ad}_{J_k} J_l \\ & + \frac{\partial q_k}{\partial p_j} \sum_{m=1}^{l-1} \frac{\partial q_m}{\partial p_i} (\mathrm{ad}_{J_k} \mathrm{ad}_{J_m} J_l) \\ & + \frac{\partial q_k}{\partial p_j} \sum_{m=1}^{k-1} \frac{\partial q_m}{\partial p_i} \left(\mathrm{ad}_{\mathrm{ad}_{J_m} J_k} J_l\right) \Bigg\} \end{aligned} \tag{85}$$

$$\begin{aligned} \frac{\partial^2 \dot{J}_l}{\partial p_i \partial p_j} = \sum_{k=1}^{l-1} \Bigg\{ & \left(\mathrm{ad}_{\frac{\partial^2 J_k}{\partial p_i \partial p_j}} J_l + \mathrm{ad}_{\frac{\partial J_k}{\partial p_j}} \frac{\partial J_l}{\partial p_i} + \mathrm{ad}_{\frac{\partial J_k}{\partial p_i}} \frac{\partial J_l}{\partial p_j} \right. \\ & \left. + \mathrm{ad}_{J_k} \frac{\partial^2 J_l}{\partial p_i \partial p_j} \right) \dot{q}_k + \left(\mathrm{ad}_{\frac{\partial J_k}{\partial p_i}} J_l \right. \\ & \left. + \mathrm{ad}_{J_k} \frac{\partial J_l}{\partial p_j} \right) \frac{\partial \dot{q}_k}{\partial p_i} + \left(\mathrm{ad}_{\frac{\partial J_k}{\partial p_i}} J_l + \mathrm{ad}_{J_k} \frac{\partial J_l}{\partial p_i} \right) \\ & \times \frac{\partial \dot{q}_k}{\partial p_j} + \mathrm{ad}_{J_k} J_l \frac{\partial^2 \dot{q}_k}{\partial p_i \partial p_j} \Bigg\}. \end{aligned} \tag{86}$$

REFERENCES

[1] M. G. Pandy, F. E. Zajac, E. Sim, and W. S. Levine, "An optimal control model for maximum-height human jumping," *J. Biomech.*, vol. 23, no. 12, pp. 1185–1198, 1990.

[2] N. Lan and P. E. Patrick, "Optimal control of antagonistic muscle stiffness during voluntary movements," *Biol. Cybern.*, vol. 71, no. 2, pp. 123–135, 1994.

[3] A. Kuo, "Optimal control model for analyzing human postural balance," *IEEE Trans. Biomed. Eng.*, vol. 42, no. 1, pp. 87–101, Jan. 1995.

[4] R. Maronski, "Minimum-time running and swimming: An optimal control approach," *J. Biomech.*, vol. 29, no. 2, pp. 245–249, 1996.

[5] Y. Ting, P. N. Sheth, and C. E. Brubaker, "Application of energy optimal control to muscular force distribution for wheelchair propulsion," *ASME Bioeng. Div. Publ. Bed.*, vol. 20, pp. 517–520, 1991.

[6] G. J. Garvin, M. Zefran, E. A. Henis, and V. Kumar, "Two-arm trajectory planning in a manipulation task," *Biol. Cybern.*, vol. 76, pp. 53–62, 1997.

[7] R. M. Alexander, "A minimum energy cost hypothesis for human arm trajectories," *Biol. Cybern.*, vol. 76, pp. 97–105, 1997.

[8] S. Stroeve, "Learning combined feedback and feedforward control of a musculoskeletal system," *Biol. Cybern.*, vol. 75, pp. 73–83, 1996.

[9] A. Karniel and G. F. Inbar, "A model for learning human reaching movements," *Biol. Cybern.*, vol. 77, pp. 173–185, 1997.

[10] G. L. Gottlieb, D. M. Corcos, and G. C. Agarwal, "Strategies for the control of voluntary movements with one mechanical degree of freedom," *Behav. Brain Sci.*, vol. 12, pp. 189–250, 1989.

[11] N. Schweighofer, M. A. Arbib, and M. Kawato, "Role of the cerebellum in reaching movements in humans. I. Distributed inverse dynamics control," *Eur. J. Neurosci.*, vol. 10, pp. 86–94, 1998.

[12] C. G. Atkeson, "Learning arm kinematics and dynamics," *Ann. Rev. Neurosci.*, vol. 12, pp. 157–183, 1989.

[13] F. A. Mussa-Ivaldi, "Nonlinear force fields: A distributed system of control primitives for representing and learning movements," in *Proc. IEEE Int. Symp. Computat. Intell. Robot. Autom.*, 1997, pp. 84–90.

[14] L. S. Crawford and S. S. Sastry, "Biological motor control approaches for a planar diver," in *Proc. 34th IEEE Conf. Decision Control*, vol. 4, New Orleans, LA, 1995, pp. 3881–3886.

[15] A. Ijspeert, J. Nakanishi, and S. Schaal, "Movement imitation with nonlinear dynamical systems in humanoid robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2002, pp. 1398–1403.

[16] J. C. W. Sullivan and A. G. Pipe, "An evolutionary optimization approach to motor learning with first results of an application to robot manipulators," in *Proc. IEEE Int. Conf. Computat. Cybern. Simul.*, vol. 5, 1997, pp. 4406–4411.

[17] N. Ogihara and N. Yamazaki, "Generation of human reaching movement using a recurrent neural network model," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, vol. 2, 1999, pp. 692–697.

[18] D. H. Rao and H. V. Kamat, "Artificial neural network for the emulation of human locomotion patterns," *Eng. Med. Biol. Soc.*, vol. 2, pp. 80–81, 1995.

[19] A. C. Fang and N. Pollard, "Efficient computation of optimal, physically valid motion," *J. Robot. Soc. Japan*, vol. 22, no. 2, pp. 23–27, 2004.

[20] K. D. Mombaur, H. G. Bock, J. P. Schloder, and R. W. Longman, "Self-stabilizing somersaults," *IEEE Trans. Robot. Autom.*, to be published.

[21] F. C. Park, J. E. Bobrow, and S. R. Ploen, "A Lie group formulation of robot dynamics," *Int. J. Robot. Res.*, vol. 14, no. 6, pp. 609–618, Dec. 1995.

[22] B. J. Martin and J. E. Bobrow, "Minimum effort motions for open chain manipulators with task-dependent end-effector constraints," in *Proc. IEEE Int. Conf. Robot. Autom.*, 1997, pp. 2044–2049.

[23] J. Kim, J. Baek, and F. C. Park, "Newton-type algorithms for robot motion optimization," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, Kyongju, Korea, 1999, pp. 1842–1847.

[24] C.-Y. E. Wang, W. K. Timoszyk, and J. E. Bobrow, "Payload maximization for open chained manipulators: Finding weightlifting motions for a Puma 762 robot," *IEEE Trans. Robot. Autom.*, vol. 17, no. 2, pp. 218–224, Apr. 2001.

[25] A. E. Bryson and Y. C. Ho, *Applied Optimal Control*. New York: Wiley, 1995.

[26] J. T. Betts, "Survey of numerical methods for trajectory optimization," *J. Guid., Control, Dyn.*, vol. 21, no. 2, pp. 193–207, 1999.

[27] H. J. Pesch, "A practical guide to the solution of real-life optimal control problems," *Control Cybern.*, vol. 23, no. 1, pp. 7–60, 1994.

[28] H. I. Krebs, N. Hogan, M. L. Aisen, and B. T. Volpe, "Robot-aided neurorehabilitation," *IEEE Trans. Rehab. Eng.*, vol. 6, no. 1, pp. 75–87, Mar. 1998.

[29] D. Luenberger, *Linear and Nonlinear Programming*. Reading, MA: Addison-Wesley, 1989.

[30] P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization*. London, U.K.: Academic, 1981.

[31] B. Martin and J. E. Bobrow, "Minimum effort motions for open chained manipulators with task-dependent end-effector constraints," *Int. J. Robot. Res.*, vol. 18, no. 2, pp. 213–224, Feb. 1999.

[32] F. C. Park, J. Choi, and S. R. Ploen, "Symbolic formulation of closed chain dynamics in independent coordinates," *J. Mech. Mach. Theory*, vol. 34, no. 5, pp. 731–751, Jul. 1999.

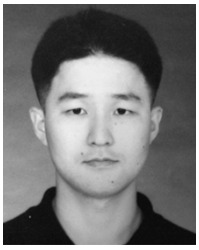
[33] S. R. Ploen and F. C. Park, "Coordinate-invariant algorithms for robot dynamics," *IEEE Trans. Robot. Autom.*, vol. 15, no. 6, pp. 1130–1135, Dec. 1999.

[34] F. C. Park and J. Kim, "Singularity analysis of closed kinematic chains," *ASME J. Mech. Des.*, vol. 121, no. 1, pp. 32–38, 1999.

[35] R. M. Murray, Z. X. Li, and S. S. Sastry, *A Mathematical Introduction to Robotic Manipulation*. Boca Raton, FL: CRC Press, 1994.

[36] Y. Nakamura and A. Dasgupta, "Generation of physically consistent interpolant motion from key frames for human-like multibody systems in flight," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, vol. 2, 1999, pp. 1102–1107.

[37] A. Ude, C. G. Atkeson, and M. Riley, "Planning of joint trajectories for humanoid robots using B-spline wavelets," in *Proc. IEEE Int. Conf. Robot. Autom.*, San Francisco, CA, Apr. 2000, pp. 2223–2228.



Sung-Hee Lee received the B.S. and M.S. degrees in mechanical engineering from Seoul National University, Seoul, Korea, in 1996 and 2000, respectively. He is currently working toward the Ph.D. degree at New York University, New York, NY.

From 2000 to 2002, he was a Research Engineer with Samsung Advanced Institute of Technology. His research interests are in multibody system dynamics and physics-based computer graphics.

Junggon Kim received the B.S. and M.S. degrees in mechanical engineering from Seoul National University, Seoul, Korea, in 1996 and 1998, respectively, where he is currently working toward the Ph.D. degree.

He has also been with the Mechatronics Research Institute of Hyundai Heavy Industries, Mabook-Ri, Korea, since 2000. His current research interests are in the area of dynamics-based robot motion generation.

F. C. Park received the B.S. degree in electrical engineering and computer science from the Massachusetts Institute of Technology, Cambridge, in 1985, and the S.M. and Ph.D. degrees in applied mathematics from Harvard University, Cambridge, MA, in 1991.

In 1988, he was a Researcher with the Manufacturing Research Group, IBM T.J. Watson Research Center, Yorktown Heights, NY. From 1991 to 1995, he was an Assistant Professor of Mechanical and Aerospace Engineering with the University of California, Irvine, and in 2001, he was a Visiting Professor with the Courant Institute of Mathematical Sciences, New York University, New York. Since 1995, he has been with the School of Mechanical and Aerospace Engineering, Seoul National University, Seoul, Korea, where he is currently a full Professor. His research interests are in robotic manipulation, planning and control, multibody system dynamics, robot design, and mathematical systems theory.

Munsang Kim received the B.S. and M.S. degrees in mechanical engineering from Seoul National University, Seoul, Korea, in 1980 and 1982, respectively, and the Dr.-Ing. degree in robotics from the Technical University of Berlin, Berlin, Germany, in 1987.

Since 1987, he has been a Research Scientist with the Korea Institute of Science and Technology, Seoul, where he has led the Advanced Robotics Research Center since 2000, and became the Director of the "Intelligent Robot—The Frontier 21 Program" in October 2003. His current research interests are design and control of novel mobile manipulation systems, haptic device design and control, and sensor application to intelligent robots.

James E. Bobrow received the M.S. and Ph.D. degrees in engineering from the University of California, Los Angeles, in 1983. His dissertation was on the optimal control of robotic manipulators.

He is currently a Professor of Mechanical and Aerospace Engineering with the University of California, Irvine (UCI). After graduate school, he was a Senior Programmer Analyst with McDonnell Douglas Automation Company, where he developed CAM software for the Unigraphics system. In July 1984, he joined UCI as an Assistant Professor, where he conducted research in robotics and applied control systems. In the 1991–1992 academic year, he was a Visiting Associate Professor with the Computer Science Department, Stanford University, Stanford, CA, where he investigated applications of numerical optimization algorithms to learning systems. He has created robots and automation devices for several start-up companies, including Robomedica, Inc., and Cobra Technologies. He serves on the Board of Directors of Robomedica, Inc., and he has served on the program committees or organizing committees of the leading conferences in control systems and robotics.