On the Reliable Computation of Optimal Motions for Underactuated Manipulators

JAMES E. BOBROW & GARETT A. SOHL
Department of Mechanical and Aerospace Engineering
University of California, Irvine
Irvine, CA 92697

Abstract

In this paper we develop a numerical solution to optimal control problems for underactuated manipulators. Our approach finds a local solution to the nonlinear optimal control problem using an SQP approach with exact gradients of the cost function. We show that a dramatic improvement in computation times can be obtained if one computes the required gradient using the adjoint equations and a backward integration standard in optimal control theory. We demonstrate the the algorithm on a 19 dof model high-diver for various initial conditions and performance criteria.

1 Introduction

The goal of this work is to create an efficient and numerically stable algorithm for solving optimal control problems for underactuated serial kinematic chains. Examples of such chains include free-floating bodies in space and some robot manipulators that use passive joints. Many of these dynamic systems could benefit from a reliable algorithm for producing motions that are optimal for performance measures like effort, power, or time.

This work builds upon approaches developed for fully actuated systems by Chen [1], Deluca [2], Field and Stepanenko, [3], Gilbert and Johnson [4], Martin and Bobrow [5], and others. However, the solution is more difficult because the dynamics algorithm must handle active and passive joints, and the gradients needed require integration of sensitivity equations in the nonlinear differential equations of motion.

2 Problem Statement

In this paper we distinguish between joints whose motion can be controlled directly with actuators, called *active joints*, and joints without actuators whose motion depends on coupling in the dynamics equations called *passive joints.* Let the active joints be denoted as $q^a(P,t) \in \Re^{n_a}$, where $P \in \Re^k$ is a set of k spline parameters that define the geometric path, and t is time. We assume that $q^a(P,t)$ is at least twice differentiable. Let the passive joints be denoted as $q^p(t) \in \Re^{n_p}$ and let $q = (q^a, q^p) \in \Re^n$, where $n = n_a + n_p$ is the total number of joints. Similarly, let the torques/forces on the active joints be denoted $\tau^a(t)$, the torques/forces on the passive joints be denoted as $\tau^p(t)$, and $\tau = (\tau^a, \tau^p)$. Note that typically $\tau^p(t) = 0$ for freely moving bodies, and that $\tau^a(t)$ can be found from the dynamics of the system given $q^a(P,t)$ and its time-derivatives.

At any instant, assume that the current state is $(q, \dot{q})$. Given the active joint accelerations $\ddot{q}^a$ and the passive joint torques τ^p, we assume that a dynamics algorithm is available that can compute as output the active joint torques, τ^a, and passive joint accelerations $\ddot{q}^p$. Let this algorithm be referred to as g with

$$[\tau^a, \ddot{q}^p] = g(q, \dot{q}, \ddot{q}^a, \tau^p), \tag{1}$$

where we have used Matlab's style of input/output format.

Given an input trajectory for the active joints $\ddot{q}^a(t)$, the forces/torques on the passive joints $\tau^p(t)$, and initial conditions $q(0), \dot{q}(0)$, we can use (1) to solve the differential equation

$$\frac{d}{dt} \left\{ \begin{array}{c} q^p \\ \dot{q}^p \end{array} \right\} = \left\{ \begin{array}{c} \dot{q}^p \\ \ddot{q}^p(q, \dot{q}, \ddot{q^a}, \tau^p) \end{array} \right\} \tag{2}$$

for the motion of the passive joints $q^p(t)$.

The form of the optimal control problem of interest is

$$\underset{\tau^a(\cdot)}{\text{Minimize}} \; J(\tau^a) = \Phi(q(t_f), \dot{q}, (t_f)) + \int_0^{t_f} L(q, \dot{q}, \tau, t) \; dt, \tag{3}$$

subject to (1) with a given initial state $(q(0), \dot{q}(0))$. For our example problems, Φ penalizes deviations from the desired final condition and $L = \int_0^t \frac{1}{2} ||\tau^a||^2 dt$, captures the desire to minimize the active joint torques. Because $\tau^a(t)$ is uniquely determined by the specified $\ddot{q}^a(P, t)$, the optimal control problem can be transformed to a static parameter optimization. The minimization over the function $\tau^a(t)$ becomes a minimization over the path parameters P.

3 Approaches to nonlinear parameter optimization

As mentioned above, through the introduction of a parameterized trajectory for the active joints $q^a(P, t)$, the optimal control problem (3), becomes a nonlinear parameter optimization for $J(P)$. The optimization is unconstrained if, given P, the integrand $L(q, \dot{q}, \tau, t)$ can computed from the solution to (2) and there are no other constraints on the motion such as joint limits, torque limits or obstacles in the workspace.

In order to minimize $J(P)$, gradient-based algorithms have been used with success for fully actuated systems by several researchers [5, 3, 2]. Finding optimal motions for underactuated systems is more difficult than fully actuated systems since for any choice of P, the equations (2) must be integrated in order to determine the position of the passive joints and the final cost $J(P)$. Fully actuated systems have no passive joints and therefore no differential equations must be integrated in order to evaluate the cost function.

During the optimization, at certain points P, the gradient of $J(P)$ must be computed. In the following, we compare three methods to compute this gradient $\frac{\partial J(P)}{\partial p}$ for $p \in P$.

3.1 Use finite-difference derivatives.

The most direct approach to computing the required gradient is to fix P and perturb all $p \in P$ by some step size h. For instance the forward difference is $\frac{\partial J(P)}{\partial p} \cong \frac{J(p+h) - J(p)}{h}$. To the best of the author's knowledge this is the only approach that has been used for finding motions for underactuated systems. Note that each numerical derivative requires the solution of (2) which is of order $2n_p$. There are k (the dimension of P) such numerical derivatives that must be computed for each gradient evaluation in the nonlinear optimization.

In our attempts to minimize $J(P)$ using this technique, much ad-hoc tweaking of h and the numerical integration tolerance were needed. In fact, often little progress was made in the SQP optimization due to the numerical innacuracy of the gradient. The ill-conditioned nature of similar types of problems was reported in Martin and Bobrow [5]. In order to obtain useful solutions, we were forced to compute the exact gradient of our cost function.

3.2 Integrate the sensitivity equations.

In order to eliminate the errors involved with finite difference gradients, one can analytically differentiate the cost function, which, for fixed final-time has the form

$$\frac{\partial J(P)}{\partial p} = \frac{\partial \Phi(q(t_f), \dot{q}, (t_f))}{\partial p} + \int_0^{t_f} \frac{\partial L(q, \dot{q}, \tau, t)}{\partial p}\, dt. \tag{4}$$

Since the trajectory of the active joints $q^a(P,t)$, is a specified function of P, it is easy to compute the derivatives of q^a, and $\dot{q}^a$ in (4). However, the derivatives of the passive joints q^p, and $\dot{q}^p$ with respect to $p \in P$ are not easy to compute. In order to obtain these derivatives one can integrate the following set of differential equations

$$\frac{d}{dt}\begin{Bmatrix} q^p \\ \dot{q}^p \\ \frac{\partial q^p}{\partial p} \\ \frac{\partial \dot{q}^p}{\partial p} \end{Bmatrix} = \begin{Bmatrix} \dot{q}^p \\ \ddot{q}^p(q, \dot{q}, \ddot{q^a}, \tau^p) \\ \frac{\partial \dot{q}^p}{\partial p} \\ \frac{\partial \ddot{q}^p}{\partial p}(q, \dot{q}, \ddot{q}^a, \tau^p, \frac{\partial q^p}{\partial p}, \frac{\partial \dot{q}^p}{\partial p}) \end{Bmatrix} \tag{5}$$

starting from a given $q^p(0), \dot{q}^p(0)$, and $\frac{\partial q^p(0)}{\partial p} = 0, \frac{\partial \dot{q}^p(0)}{\partial p} = 0$.

Note that the lower right-hand term in this equation requires that we differentiate the dynamics algorithm (1) with respect to p. This in itself is a nontrivial task and has been reported in [7]. In addition, the order of (5) has grown from $2n^p$ in (2) to $2n^p(1+k)$, which could be very large depending on how the motion of the active joints is parameterized. Hence the $k+1$ numerical integrations of (2) needed to obtain the *approximate* finite difference gradient of J is replaced be one numerical integration of (5) in order to obtain the *exact* gradient of J.

The examples presented later in this paper used integration of (5) to obtain the gradient of J during the nonlinear SQP parameter optimization. Although our solutions reliably converged to locally optimal solutions, they were relatively time-consuming to compute for the high degree of freedom system in this paper. This was mainly because the order of (5) was high. The following alternative approach dramatically decreases the computational cost.

3.3 Integrate the adjoint equations.

The following approach to obtaining the gradient J uses a linear systems theory perspective similar to that given in [8]. Define $x \equiv \begin{pmatrix} q \\ \dot{q} \end{pmatrix}$, and recall that $q \equiv \begin{pmatrix} q^a \\ q^p \end{pmatrix}$ contains the active and the passive joints. Define the controls as $u(t) \equiv \ddot{q}^a(t)$. Then using (1), the dynamics can be written as

$$\dot{x} = f(x(t), u(t)). \tag{6}$$

That is, the entire motion for the underactuated system is determined by specification of the motion of the active joints $q^a(t)$. Now linearize (6) about the input joint trajectory $u_o(t) = \ddot{q}^a(t)$, and the

solution $x_o(t)$, with $\dot{x}_o = f(x_o(t), u_o(t))$. Let $\delta x = x - x_o$ and $\delta u = u - u_o$. For small changes from the nominal trajectory, the equations of motion have the linear form

$$\delta\dot{x} = A(t)\delta x + B(t)\delta u(t), \tag{7}$$

where $A(t) = \frac{\partial f}{\partial x}(x_o(t), u_o(t))$, and $B(t) = \frac{\partial f}{\partial u}(x_o(t), u_o(t))$.

The optimal control problem to be solved has the general form

$$J(u(t)) = \phi(x(t_f)) + \int_0^{t_f} L(x, u, t)dt. \tag{8}$$

In order to construct the gradient of $J(u)$, first rewrite (8) as follows. Introduce an additional state $z(t) \equiv \int_0^t L(x, u, \tau)d\tau$, so that $\dot{z}(t) = L(x, u, t)$. Then (8) becomes

$$J(u(t)) = \tilde{\phi}(\tilde{x}(t_f)), \tag{9}$$

where $\tilde{x} \equiv \begin{pmatrix} x \\ z \end{pmatrix}$. For small perturbations from any nominal trajectory $x_o(t)$, (9) can be written as

$$J(u(t)) = J(u_o) + \delta J \cong \tilde{\phi}(\tilde{x_o}(t_f)) + \frac{\partial\tilde{\phi}}{\partial\tilde{x}}(\tilde{x_o}(t_f))\ \delta\tilde{x}(t_f) \tag{10}$$

Our control variables are parameterized by a finite set of basis functions $B(t)$. Each basis function $B(t)$ has an associated amplitude $p \in P$. Noting that only $\delta\tilde{x}(t_f)$ will be effected by changing p from its nominal value, the required derivative $\frac{\partial J}{\partial p}$ is

$$\frac{\partial J(u(t))}{\partial p} = \frac{\partial\tilde{\phi}}{\partial\tilde{x}}(\tilde{x_o}(t_f))\ \frac{\partial\delta\tilde{x}(t_f)}{\partial p} \tag{11}$$

In order to solve for $\frac{\partial\delta\tilde{x}(t_f)}{\partial p}$ in (11), let $\Phi(t, \tau)$ be the state-transition matrix for (7) so that $\dot{\Phi}(t, \tau) = A(t)\Phi(t, \tau)$, with $\Phi(\tau, \tau) = I$. Then

$$\delta\tilde{x}(t_f) = \Phi(t_f, 0)\delta\tilde{x}(0) + \int_0^{t_f} \Phi(t_f, \tau)B(\tau)\delta u(\tau)d\tau, \tag{12}$$

and differentiating (12) with respect to p and premultiplying by $\frac{\partial\tilde{\phi}}{\partial\tilde{x}}$, we have

$$\frac{\partial J(u(t))}{\partial p} = \int_0^{t_f} \frac{\partial\tilde{\phi}(t_f)}{\partial\tilde{x}}\Phi(t_f, \tau)B(\tau)\frac{\partial\delta u(\tau)}{\partial p}d\tau. \tag{13}$$

Define the costate or adjoint vector λ as

$$\lambda^T(t) \equiv \frac{\partial\tilde{\phi}(t_f)}{\partial\tilde{x}}\Phi(t_f, t). \tag{14}$$

Differentiating (14) with respect to time and using the properties of the state-transition matrix, it is seen that $\lambda(t)$ is the solution to the differential equation

$$\dot{\lambda} = -A(t)^T\lambda \ \text{ with } \lambda^T(t_f) = \frac{\partial\tilde{\phi}(t_f)}{\partial\tilde{x}}, \tag{15}$$

which can be solved backwards in time. The solution for $\lambda(t)$ needs to be stored numerically. Substituting (14) into (13) gives

$$\frac{\partial J(u(t))}{\partial p_i} = \int_0^{t_f} \lambda^T(\tau) B(\tau) \frac{\partial \delta u(\tau)}{\partial p} d\tau. \tag{16}$$

The importance of (16) is that it reduces the problem of computing the gradient of the cost function from that of integrating an ode of order $2n^p(k+1)$ in (2) into the problem of integrating an ode of order $2n^p$ for $\lambda(t)$ and then a quadrature in (16). Note that this quadrature can be done numerically efficiently since, for splines of limited support, $\frac{\partial \delta u(\tau)}{\partial p_i}$ is zero during most of the time interval.

4 Example: Branched High Diver: Twisting One-and-one Half Dive

We have developed a Matlab package [7] to evaluate (1) and its derivatives for general dynamic systems. The algorithm is an efficient, recursive $O(n)$, solution. Matlab's standard toolboxes can then be used in conjunction with our software to solve optimal control problems for general serial mechanisms. For a non-trivial example, we solved a three dimensional high dive. The diver model has 13 active degrees of freedom and is connected to the ground with 6 passive degrees of freedom. The active degrees of freedom include knees, hips, pelvis (2 dof), shoulders (2 dof), elbows and neck. The 6 passive degrees of freedom determine the location of the center of mass of the the diver's torso and allow for a full range of three dimensional motions. A total of 104 parameters are used to define the motion – 8 parameters for each of the 13 active joints. A cost function of the form

$$J = c_1 \left(q_4(t_f) - 3\pi\right)^2 + c_2 \left(q_5\right)^2 + c_3 \left(q_6 - \pi\right)^2 + c_4 \int_0^{t_f} ||\tau||^2 \, dt \tag{17}$$

was used in an attempt to make the diver perform a forward one-and-one-half dive with a half twist. The terms inside the integral reflect our desire to minimize the effort used during the dive.

For the initial guess of the motion shown on the left in Figure 1, we chose the spline parameters to bend the knees a small amount. Other than that, we made no other guesses about the joint motions required to perform the dive. The resulting dive had a cost function value of $J = 470.5$ and does not look at all close to the desired dive. The optimization produced the final dive shown on the right of Figure 1 with a cost of $J = 4.6$. It is interesting that, for the optimized dive, the legs bend backwards during the motion.

We obtained different dives from different initial guesses. Figure 2 shows one such dive in which the diver kept one leg fairly straight during most of the rotation motion. Figure 3 shows a dive starting from the same initial guess shown in Figure 2 but with a constrained leg motion. We constrained the legs to remain in a tucked position throughout the dive. This forced the diver to use arm and pelvis motion to produce the desired one-half twist as shown in Figure 3.

5 Conclusion

In this paper we have presented an approach to computing optimal motions for underactuated manipulators. We have demonstrated that our algorithm can reliably handle complex systems such as humanoids, and that interesting bio-mimetic like motions can be found with it. Our example

Figure 1: Initial guess and final dive

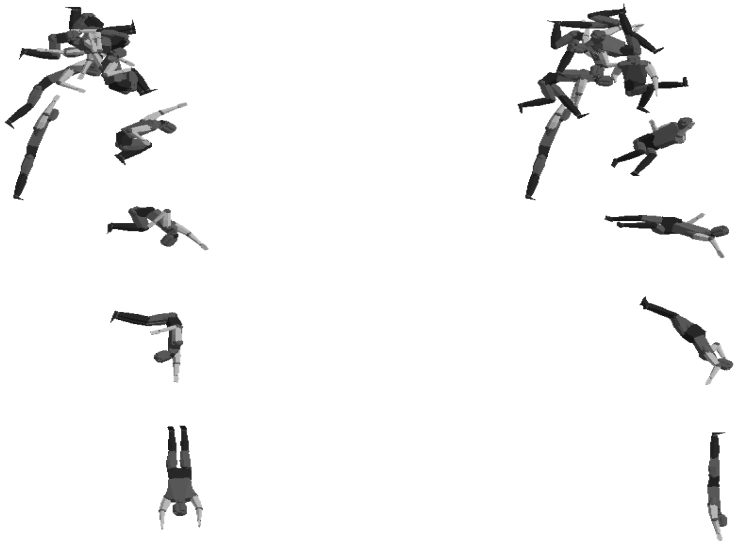

Figure 2: Initial guess and final dive

high-diver also demonstrates that humans perform tasks they could be minimizing cost functions similar to those discussed in this paper. We have shown that the computation of the gradient using the adjoint equation from optimal control provides a significant decrease in the required computation

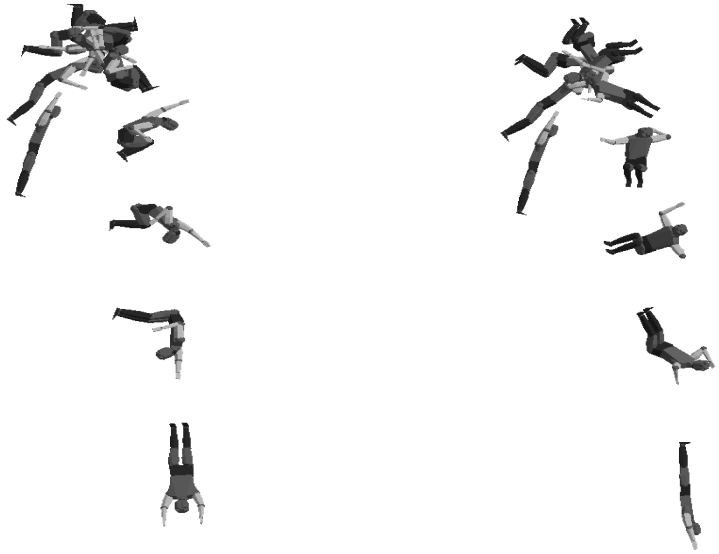

Figure 3: Final dive with constrained leg motion

time.

6 Acknowledgments

The authors thank Ken Mease for his help with Section 3.3. This research was supported by the National Science Foundation under grant IRI-9711782.

References

[1] Y-C. Chen,"Solving robot trajectory Planning problems with uniform cubic B-splines," *Optimal Control Applications and Methods*, Vol. 12, No. 4, pp. 247-262, 1991.

[2] A. De Luca, L. Lanari and G. Oriolo,"A Sensitivity Approach to Optimal Spline Robot Trajectories," *Automatica*, Vol. 27, No. 3, pp. 535-539, 1991.

[3] G. Field and Y. Stepanenko, "Iterative dynamic programming: An approach to minimum energy trajector planning for robotic manipulators," in *IEEE International Conference on Robotics and Automation*, (Minneapolis, MN), pp. 2755–2760, April 1996.

[4] E. G. Gilbert and D. W. Johnson, "Distance functions and their application to robot path planning in the presence of obstacles," *IEEE Journal of Robotics and Automation*, vol. RA-1, pp. 21–30, 1985.

[5] B. J. Martin and J. E. Bobrow, "Minimum effort motions for open chain manipulators with task-dependent end-effector constraints," *International Journal of Robotics Research*, Vol. 18, No. 2, pp. 213-224, 1999.

[6] F. C. Park, J. E. Bobrow and S. R. Ploen, "A Lie group formulation of robot dynamics," *International Journal of Robotics Research*, Vol. 14, No. 6, pp. 609-618, 1995.

[7] G. A. Sohl and J. E. Bobrow, "A Recursive Multibody Dynamics and Sensitivity Algorithm for Branched Kinematic Chains," to appear in the *ASME Journal of Dynamic Systems, Measurement, and Control,* 2001.

[8] T. Kailath, *Linear Systems,* Englewood Cliffs, N.J., Prentice-Hall 1980.