Proceedings of the 1999 IEEE/RSJ
International Conference on
Intelligent Robots and Systems

Newton-Type Algorithms for Robot Motion Optimization

Junggon Kim Jonghyun Baek F. C. Park

School of Mechanical and Aerospace Engineering
Seoul National University
Seoul 151-742, Korea
junggon@robotics.snu.ac.kr
fcp@robotics.snu.ac.kr

Abstract

This paper presents a class of Newton-type algorithms for the optimization of robot motions that take into account the dynamics. Using techniques from the theory of Lie groups and Lie algebras, the equations of motion of a rigid multibody system can be formulated in such a way that both the first and second derivatives of the dynamic equations with respect to arbitrary joint variables can be computed analytically. The result is that one can formulate the exact gradient and Hessian of an objective function involving the dynamics, and develop efficient second-order Newton-type optimization algorithms for generating optimal robot motions. The methodology is illustrated with a nontrivial example.

1 Introduction

Of all the remarkable physical abilities of humans, motor control is the skill that is most often taken for granted, as it seems to require the least conscious effort on our part. Our aim in this paper is to emulate, for robots, 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 motor control and learning. Indeed, in the literature one can find many optimal control-based studies of various human motions, e.g., maximum-height jumping [1], and voluntary arm movements [2]. In addition to the more obvious optimization criteria like minimum energy or control effort, strategies that involve minimizing the derivative of acceleration (or jerk) [3], as well as muscle or metabolic energy costs [4], have also been examined in the context of specific arm motions.

From an engineering perspective an optimization-based approach to motion generation usually comes to mind as the first reasonable thing to try. Past approaches have usually met failure, however, because the complexity of the governing equations of motion usually led to intractable optimization problems. In a series of previous papers [6], [7], it was shown that by appealing to techniques from the theory of Lie groups, one can formulate the equations of motion of even complicated multibody systems like the human body in such a way as to render the optimization problem tractable. In many cases the solutions can even be obtained quite efficiently and in a numerically robust way. The key, as we discuss below, lies in the ability to compute exact analytic gradients of the objective function, without resorting to expensive and inaccurate numerical approximations that are often the cause of instability and lack of convergence.

Even with such a capability, however, the algorithms still require significant amounts of computation, and can be quite slow to converge. Such slow convergence is characteristic of optimization algorithms that rely only on first-order gradient information—the classical example involves the steepest descent algorithm, which leads to the well-known zig-zag phenomena near poorly conditioned local minima (i.e., local minima at which the Hessian becomes poorly conditioned). Quasi-Newton methods such as the DFP and BFGS methods construct an approximation to the Hessian (or more accurately, its inverse) with the gradient information gathered during the descent process. It has been pointed out, however [10], that the performance of quasi-Newton methods is highly sensitive to the accuracy of the line search algorithm, and that in general exact Hessian information, if it can be obtained efficiently, will lead to more robust and efficient optimization algorithms.

In this paper we show that, by suitably reformulating the recursive geometric dynamics algorithms derived in [6] and [7], one can formulate the Hessian of the objective function in analytic form, thereby making the implementation of second-order Newton-type optimization algorithms possible. As is well-known, Newton-type algorithms have quadratic convergence properties, and in general offer much greater robustness and faster convergence compared to strictly first-order gradient-based methods.

We begin by first describing the dynamic modeling and gradient-based optimization algorithms developed using techniques from Lie group theory. We then show how to modify and extend this formulation such that second derivatives of the dynamic equations can be obtained recursively. We then investigate a class of natural motion problems that involve generating minimum torque motions: after parameterizing the trajectories in terms of B-splines, we formulate the trajectory generation problem as a parameter optimization problem involving the B-spline control points. Experimental results obtained using steepest descent, the BFGS quasi-Newton method, and a modified Newton method are obtained and compared. Natural motions obtained using these algorithms are illustrated for a two-link planar open chain, and the numerical efficiency and convergence properties of the various algorithms are discussed.

2 Recursive Differentiation of the Dynamics

In order to determine optimal, natural motions for the robot systems of interest, a complete dynamic model is needed. Within the fixed kinematic topology, our motion optimizer can vary the parameters of the model to find the optimal motion, or motor program, for whatever performance measure selected. The dynamic equations for an open kinematic chain can be written in the form

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

where q denotes the joint position vector, $M(q)$ is the inertia matrix, and C and G represent the Coriolis/centrifugal and gravity terms, respectively. The above dynamics equation can be computed recursively; first we begin with some definitions and notation. Recall that the group of rigid body transformations on $\Re^3$, denoted SE(3), is given by 4×4 matrices of the form

$$g = \begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix} \qquad (2)$$

where $p \in \Re^3$ and R is a 3×3 rotation matrix. The Lie algebra of $SE(3)$ is called $se(3)$, and can be represented by the following 4×4 matrix form

$$\xi = \begin{bmatrix} [w] & v \\ 0 & 0 \end{bmatrix} \qquad (3)$$

where $w, v \in \Re^3$, and $[w] = \begin{pmatrix} 0 & -w_3 & w_2 \\ w_3 & 0 & -w_1 \\ -w_3 & w_1 & 0 \end{pmatrix}$. Note that $se(3)$ can be regarded as 6-dimensional vector space, $\xi = (w, v) \in \Re^6$. Now, the exponential map $\exp : se(3) \to SE(3)$ is defined as the following relation

$$\exp \xi = \begin{bmatrix} e^{[w]} & Av \\ 0 & 1 \end{bmatrix} \qquad (4)$$

where

$$\begin{aligned} e^{[w]} &= I + \frac{[w]}{||w||}\sin||w|| + \frac{[w]^2}{||w||^2}(1 - \cos||w||) \\ A &= I + \frac{[w]}{||w||^2}(1 - \cos||w||) \\ &+ \frac{[w]^2}{||w||^3}(||w|| - \sin||w||). \end{aligned}$$

We will now give several mappings, which help to reduce the complexity of dynamics. With $g \in SE(3)$ and $\xi \in se(3)$, the adjoint mapping $\mathrm{Ad} : SE(3) \times se(3) \to se(3)$ is defined as follows.

$$\mathrm{Ad}_g\, \xi = g\, \xi\, g^{-1} \qquad (5)$$

Lie bracket can be represented by the mapping ad : $se(3) \times se(3) \to se(3)$, which is defined as

$$\mathrm{ad}_{\xi_1}\, \xi_2 = \xi_1\xi_2 - \xi_2\xi_1 \qquad (6)$$

It could be more convenient to treat the above mappings as the following 6×6 transformation matrices acting on $\Re^6$

$$\begin{aligned} \mathrm{Ad}_g\, \xi &= \begin{bmatrix} R & 0 \\ [p]R & R \end{bmatrix} \begin{pmatrix} w \\ v \end{pmatrix} \qquad (7) \\ \mathrm{ad}_{\xi_1}\, \xi_2 &= \begin{bmatrix} [w_1] & 0 \\ [v_1] & [w_1] \end{bmatrix} \begin{pmatrix} w_2 \\ v_2 \end{pmatrix} \qquad (8) \end{aligned}$$

where $g = (R, p)$, $\xi_1 = (w_1, v_1)$ and $\xi_2 = (w_2, v_2)$. We can also define the dual adjoint mapping and dual Lie bracket with the following matrix transformation.

$$\mathrm{Ad}_g^* \, \xi^* = \begin{bmatrix} R & 0 \\ [p]R & R \end{bmatrix} \begin{pmatrix} w^* \\ v^* \end{pmatrix} \tag{9}$$

$$\mathrm{ad}_{\xi_1}^* \, \xi_2^* = \begin{bmatrix} [w_1] & 0 \\ [v_1] & [w_1] \end{bmatrix} \begin{pmatrix} w_2^* \\ v_2^* \end{pmatrix} \tag{10}$$

Now, the Newton-Euler formulation of an n-dof serial robot dynamics can be written with the above notations:

- **Forward recursion: for k=1 to n**

$$\begin{aligned} f_{k-1,k} &= M_k e^{S_k q_k} \\ V_k &= \mathrm{Ad}_{f_{k-1,k}^{-1}} V_{k-1} + S_k \dot{q}_k \\ \dot{V}_k &= S_k \ddot{q}_k + \mathrm{Ad}_{f_{k-1,k}^{-1}} \dot{V}_{k-1} + \mathrm{ad}_{V_k} S_k \dot{q}_k \end{aligned}$$

- **Backward recursion: for k=n to 1**

$$\begin{aligned} F_k &= \mathrm{Ad}^*_{f_{k,k+1}^{-1}} F_{k+1} + J_k \dot{V}_k - \mathrm{ad}^*_{V_k} J_k V_k \\ \tau_k &= {S_k}^T F_k \end{aligned}$$

where $f_{k-1,k} = M_k e^{S_k q_k}$ denotes the rigid body transformation from link $k-1$ to k with $M_k \in SE(3)$ and $S_k \in se(3)$, $V_i \in \Re^6$ is the generalized velocity of link k, $F_i \in \Re^6$ is the total generalized force transmitted from link $k-1$ to k, and J_k is the 6×6 generalized inertia matrix of link k. J_k is composed of inertia matrix I_k, mass m_k and the vector r_k from the link k frame to the center of mass of link k with the following form:

$$J_k = \begin{bmatrix} I_k - m_k [r_k]^2 & m_k [r_k] \\ -m_k [r_k] & m_k \mathbf{1} \end{bmatrix}$$

For more details on the geometric dynamics and its notations, one is referred to [6] and [8].

To derive the derivatives of the dynamic equations, we need the following basic derivatives:

$$\begin{aligned} \frac{\partial}{\partial p} \mathrm{Ad}_{e^{Sq}M} &= \frac{\partial q}{\partial p} \mathrm{ad}_S \mathrm{Ad}_{e^{Sq}M} \\ \frac{\partial}{\partial p} \mathrm{Ad}_{Me^{Sq}} &= \frac{\partial q}{\partial p} \mathrm{Ad}_{Me^{Sq}} \mathrm{ad}_S \\ \frac{\partial}{\partial p} \mathrm{Ad}^*_{e^{Sq}M} &= \frac{\partial q}{\partial p} \mathrm{Ad}^*_{e^{Sq}M} \mathrm{ad}^*_S \\ \frac{\partial}{\partial p} \mathrm{Ad}^*_{Me^{Sq}} &= \frac{\partial q}{\partial p} \mathrm{ad}^*_S \mathrm{Ad}^*_{Me^{Sq}} \\ \frac{\partial}{\partial p} \mathrm{ad}_V &= \mathrm{ad}_{\frac{\partial V}{\partial p}} \\ \frac{\partial}{\partial p} \mathrm{ad}^*_V &= \mathrm{ad}^*_{\frac{\partial V}{\partial p}} \end{aligned}$$

where $S \in se(3)$ and $M \in SE(3)$ are all constants, and $q \in \Re$ and $V \in \Re^6$ are variables.

Now one can easily differentiate the recursive dynamic equations with respect to a variable $p = (p_1, \cdots, p_m) \in \Re^m$.

- **Forward recursion: for k=1 to n**

$$\begin{aligned} \frac{\partial V_k}{\partial p_i} &= \frac{\partial q_k}{\partial p_i} \mathrm{ad}_{-S_k} \mathrm{Ad}_{f_{k-1,k}^{-1}} V_{k-1} \\ &+ \mathrm{Ad}_{f_{k-1,k}^{-1}} \frac{\partial V_{k-1}}{\partial p_i} + S_k \frac{\partial \dot{q}_k}{\partial p_i} \\ \frac{\partial \dot{V}_k}{\partial p_i} &= S_k \frac{\partial \ddot{q}_k}{\partial p_i} + \frac{\partial q_k}{\partial p_i} \mathrm{ad}_{-S_k} \mathrm{Ad}_{f_{k-1,k}^{-1}} \dot{V}_{k-1} \\ &+ \mathrm{Ad}_{f_{k-1,k}^{-1}} \frac{\partial \dot{V}_{k-1}}{\partial p_i} + \mathrm{ad}_{\frac{\partial V_k}{\partial p_i}} S_k \dot{q}_k \\ &+ \mathrm{ad}_{V_k} S_k \frac{\partial \dot{q}_k}{\partial p_i} \end{aligned}$$

- **Backward recursion: for k=n to 1**

$$\begin{aligned} \frac{\partial F_k}{\partial p_i} &= \frac{\partial q_{k+1}}{\partial p_i} \mathrm{Ad}^*_{f_{k,k+1}^{-1}} \mathrm{ad}^*_{-S_{k+1}} F_{k+1} \\ &+ \mathrm{Ad}^*_{f_{k,k+1}^{-1}} \frac{\partial F_{k+1}}{\partial p_i} + J_k \frac{\partial \dot{V}_k}{\partial p_i} \\ &- \mathrm{ad}^*_{\frac{\partial V_k}{\partial p_i}} J_k V_k - \mathrm{ad}^*_{V_k} J_k \frac{\partial V_k}{\partial p_i} \\ \frac{\partial \tau_k}{\partial p_i} &= {S_k}^T \frac{\partial F_k}{\partial p_i} \end{aligned}$$

One can also recognize the above first derivative algorithm as being similar to the form as presented in [7] with some subtle yet important differences. Differentiating this modified form of the recursive gradient algorithm, we can obtain a recursive algorithm for analytically evaluating the second derivatives of the dynamics as follows:

- **Forward recursion: for k=1 to n**

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

- **Backward recursion: for k=n to 1**

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

Despite the seeming complexity of the recursive computations, it should be noted that many of the computations embedded in the forward and backward recursions need only be evaluated once.

3 Generating Minimum Control Effort Motions

We now show how the recursive algorithms of the previous section can be effectively put to use to generate minimum control effort motions for robots. The particular form of the objective function we consider is as follows:

$$\min_{q(t)} J = \frac{1}{2} \int_0^{t_f} \tau^T \tau \, dt \qquad (11)$$

subject to the dynamic equations of the system; here $q(t)$ and τ represent the joint and torque profiles, respectively. A local solution to the above 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 $P = \{p_1, ..., p_m\}$, with $p_i \in \Re^n$. The joint trajectories then have the form $q = q(t, P)$ with

$$q(t, P) = \sum_{i=1}^{m} B_i(t) p_i \qquad (12)$$

We should note that the use of B-Spline polynomials as the basic primitives upon which all of our motions are developed is consistent with recent results in neuroscience. In [5] it was observed clinically 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. We are achieving the same basic effect through (12).

With this approach $\tau = \tau(P, t)$; $q, \dot{q}$, and $\ddot{q}$ all are given functions of t and P from (12) and its time-derivatives, and hence τ is an explicit function of the spline parameters through (1).

We have converted the original problem into a parameter optimization problem, and efficient quasi-Newton algorithms can then be used to solve the problem. However, for assured 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 [10]) used in the quasi-Newton algorithm must remain positive definite with bounded condition number [9].

Due to the complexity of the dynamic equations of motion, most previous solutions to nontrivial optimal control problems for robotic systems use finite difference gradient approximations of $J(P)$. In these cases, it is usually not possible to ensure a bounded condition number of the approximate Hessian, and the algorithms usually terminate prematurely. In order to compute the gradient and Hessian of the objective function, we note that

$$\frac{\partial J}{\partial P} = \int_0^{t_f} \tau^T \frac{\partial \tau}{\partial P}\, dt \qquad (13)$$

$$\frac{\partial^2 J}{\partial P^2} = \int_0^{t_f} \frac{\partial \tau}{\partial P}^T \frac{\partial \tau}{\partial P} + \tau^T \frac{\partial^2 \tau}{\partial P^2}\, dt. \qquad (14)$$

The most significant step for evaluating the gradient and Hessian is computing the derivatives of the joint torques with respect to the path parameters P. We compute these derivatives analytically by the recursive algorithms derived previously.

We now consider the optimal lifting motion of a 2 d.o.f. planar open chain under the influence of gravity. The joint trajectory for the open chain is represented by a B-spline with 9 control points, and the following three optimization algorithms are evaluated for comparison purposes: steepest descent, BFGS quasi-Newton method, modified Newton's method with line search. A linearly interpolated trajectory is used as the initial trajectory in each of the optimization algorithms, for which the initial value of the objective function is 703.447, and the stopping criterion is set to $||g|| < 10^{-2}$. We briefly describe each algorithm and the corresponding optimization results.

3.1 Steepest Descent

The steepest descent method is described by the following recursive algorithm:

$$\begin{aligned} x_{k+1} &= x_k - \alpha_k g_k^T \\ \alpha_k &= \arg\min_{\alpha>0} J(x_k - \alpha g_k^T) \end{aligned}$$

where g_k denotes the gradient at x_k. The final value of the objective function is 314.069. The number of iterations was forcefully terminated at 229 after the algorithm failed to meet the stopping criterion—the zigzag phenomena was encountered. The elapsed time on a Pentium 300MHz PC was 472 seconds. It is precisely because of the poor convergence properties of the steepest descent method that more sophisticated second-order algorithms have been developed; we next consider the performance of these Newton-type algorithms.

3.2 BFGS Quasi-Newton Method

In the BFGS quasi-Newton Method, the Hessian H of the objective function is approximated with the gradient information gathered during the descent process:

$$\begin{aligned} x_{k+1} &= x_k - \alpha_k H_k^{-1} g_k^T \\ \alpha_k &= \arg\min_{\alpha>0} J(x_k - \alpha H_k^{-1} g_k^T) \\ p_k &= \alpha_k H_k^{-1} g_k^T\ ,\ \ q_k = g_{k+1}^T - g_k^T \\ H_{k+1}^{-1} &= H_k^{-1} + \frac{p_k p_k^T}{p_k^T q_k} - \frac{H_k q_k q_k^T H_k}{q_k^T H_k q_k} \end{aligned}$$

In spite of using an approximated Hessian, the convergence properties are not as good as expected. We also found that the algorithm was extremely slow to converge near the solution, and that the algorithm had to be forcefully terminated after 45 iterations. The final value of the objective function was 314.069, and the elapsed computation time was 88 seconds.

3.3 Modified Newton Method

The recursive steps for the modified Newton method are given as follows:

$$\begin{aligned} x_{k+1} &= x_k - \alpha_k H_k^{-1} g_k^T \\ \alpha_k &= \arg\min_{\alpha} J(x_k - \alpha H_k^{-1} g_k^T) \end{aligned}$$

The final value of the objective function was 314.069, which is same with the previous ones. The algorithm satisfied the stopping criteria after 13 iterations, and the elapsed computation time was 76 seconds.

3.4 Discussion of Results

Even though the steepest descent method shows the simplest formula, its poor convergence property makes one hesitate to adopt it. Also the BFGS method shows good performance in searching the local minima with only gradient information, but has the disadvantage of poor performance near the solution. In spite of its fastest convergence property, the speed of the modified Newton algorithm was not so fast because of its long calculation time in obtaining the Hessian. In fact, even though the fastest algorithm in obtaining the lifting motion of the 2 d.o.f. planar open chain was the modified Newton method, the BFGS method is the

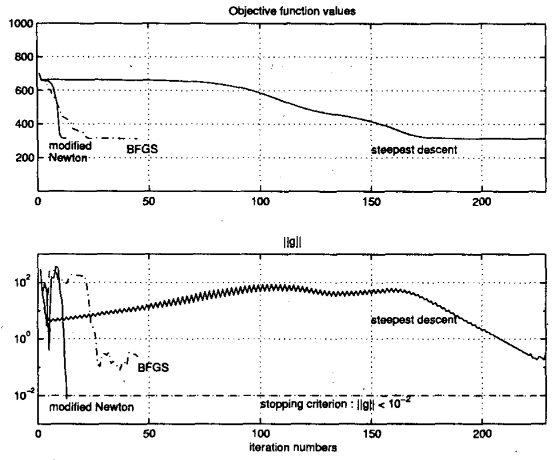

Figure 1: The objective function value and the norm of the gradient profiles

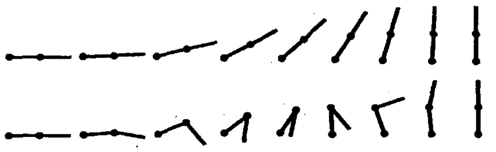

Figure 2: Lifting motion of a 2 d.o.f. planar open chain : initial and final trajectories

fastest one in finding near local minima for higher d.o.f. systems. So, it seems promising to combine the BFGS method with the algorithms, which use the exact Hessian, such as the modified Newton method for fast convergence property near a solution.

4 Conclusions

In this paper we have presented a methodology for generating optimal robot motions, by developing optimization algorithms that involve second-order derivative information of the dynamic equations. By formulating the equations of motion using techniques from the theory of Lie groups and Lie algebras, one is able to derive efficient recursive algorithms for evaluating both the gradient and Hessian of the objective function. The performance of several numerical optimization algorithms involving gradient and Hessian information are compared.

Some potential uses for our optimization algorithms include using the optimized motions as training data for, e.g., neural network-based learning schemes. While our algorithm produces reliable and physically plausible solutions, the computational requirements are still significant enough to prevent real-time trajectory generation. Using the optimized motions as training data for more advanced learning schemes appears to be a promising approach to generating motions in a more computationally efficient way.

Acknowledgements

This research was supported in part by a Korea-USA joint research grant provided by KOSEF and the NSF, and the Korea Institute of Science and Technology.

References

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

[2] N. Lan, "Analysis of an optimal control model of multi-joint arm movements," *Biol. Cybern.*, vol. 76, pp. 107-117, 1997.

[3] 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.

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

[5] H.I. Krebs, N. Hogan, M.L. Aisen, and B.T. Volpe, "Robot-Aided Neurorehabilitation," *IEEE Trans. Rehab. Eng.*, vol. 6, no. 1, March 1998, pp 75-87.

[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, December 1995, pp. 609-618.

[7] B. J. Martin, and J. E. Bobrow, "Determination of Minimum-Effort Motions for General Open Chains," *Proc. IEEE Int. Conf. Robotics and Automation*, Nagoya, Aichi, Japan, pp. 1160-1165.

[8] R. W. Murray, Z. Li, and S. S. Sastry, *A Mathematical Introduction to Robotic Manipulation*, CRC Press, Boca Raton, Florida, 1993.

[9] Gill, P. E., Murray, W. & Wright, M. H. (1981), *Practical optimization*, Academic Press.

[10] Luenberger, D. G. (1989), *Linear and Nonlinear Programming, Second Edition*, Addison-Wesley, Reading, Massachusetts.