Proceedings of BIO2006
2006 Summer Bioengineering Conference
June 21-25, Amelia Island Plantation, Amelia Island, Florida, USA

BIO2006-XXXXX

BENIFITS OF AUTOMATIC DIFFERENTIATION FOR BIOMECHANICAL OPTIMIZATIONS

Jeffrey A. Reinbolt (1) and Benjamin J. Fregly (1,2)

(1) Department of Mechanical & Aerospace Engineering
University of Florida
Gainesville, FL

(2) Department of Biomedical Engineering
University of Florida
Gainesville, FL

INTRODUCTION

Optimization algorithms are often used to solve biomechanical system identification or movement prediction problems employing complicated three-dimensional (3D) musculoskeletal models [1]. When gradient-based methods are used to solve large-scale problems involving hundreds of design variables, the computational cost of performing repeated simulations to calculate finite difference gradients can be extremely high. In addition, as 3D movement model complexity increases, there is a considerable increase in the computational expense of repeated simulations. Frequently, in spite of advances in processor performance, optimizations remain limited by computation time.

Both speed and robustness of gradient-based optimizations are dramatically improved by using an analytical Jacobian matrix (all first-order derivatives of dependent objective function variables with respect to independent design variables) rather than relying on finite difference approximations. Granted the objective function may involve thousands or perhaps millions of lines of computer code, the task of computing analytical derivatives by hand or even symbolically may prove impractical. For more than eight years, the Network Enabled Optimization Server (NEOS) at Argonne National Laboratory has been using Automatic Differentiation (AD), also called Algorithmic Differentiation, to compute Jacobians of remotely supplied user code [2]. AD is a technique for computing derivatives of arbitrarily complex computer programs by mechanical application of the chain rule of differential calculus. AD exploits the fact every computer program, no matter how complicated, executes a sequence of elementary arithmetic operations. By applying the chain rule repeatedly to these operations, derivatives can be computed automatically and accurately to working precision.

In this paper, we evaluate the benefits of using AD methods to calculate analytical Jacobians for biomechanical optimization problems. We performed the evaluation by applying a freely-available AD package, Automatic Differentiation by OverLoading in C++ (ADOL-C) [3], to two biomechanical optimization problems. The first is a system identification problem for a 3D kinematic ankle joint model involving 252 design variables and 1800 objective function elements. The second is a movement prediction problem for a 3D full-body gait model involving 660 design variables and 4100 objective function elements. Both problems are solved using a nonlinear least squares optimization algorithm.

METHODS

Experimental kinematic and kinetic data were collected from a single subject using a video-based motion analysis system (Motion Analysis Corporation, Santa Rosa, CA) and two force plates (AMTI, Watertown, MA). Institutional review board approval and informed consent were obtained prior to the experiments.

The ankle joint problem was first solved without AD using the Matlab (The Mathworks, Inc., Natick, MA) nonlinear least squares optimizer [4]. The optimization approach simultaneously adjusted joint parameter values and model motion to minimize errors between model and experimental marker locations. The ankle joint model possessed 12 joint parameters and 12 degrees-of-freedom. Each of the 12 generalized coordinate curves was parameterized using 20 B-spline nodal points (240 total). Altogether, there were 252 design variables. The problem contained 18 error quantities for each of the 100 time frames of data. The Jacobian matrix consisted of 1800 rows and 252 columns estimated by finite difference approximations.

The movement prediction problem was first solved without AD using the same Matlab nonlinear least squares optimizer [5]. The

optimization approach simultaneously adjusted model motion and ground reactions to minimize knee adduction torque and 5 categories of tracking errors (foot path, center of pressure, trunk orientation, joint torque, and fictitious ground-to-pelvis residual reactions) between model and experiment. The movement prediction model possessed 27 degrees-of-freedom (21 adjusted by B-spline curves and 6 prescribed for arm motion) and 12 ground reactions. Each adjustable generalized coordinate and reaction curve was parameterized using 20 B-spline nodal points (660 total design variables). The problem contained 41 error quantities for each of the 100 time frames of data. The Jacobian matrix consisted of 4100 rows and 660 columns estimated by finite difference approximations.

ADOL-C was incorporated into each objective function to compute an analytical Jacobian matrix. This package was chosen because the objective functions were comprised of Matlab mex-files, which are dynamic link libraries of compiled C or C++ code. ADOL-C was implemented into the C++ source code by the following steps:

1. Mark the beginning and end of active section (portion computing dependent variables from independent variables) using built-in functions `trace_on` and `trace_off`, respectively.
2. Select a set of active variables (those considered differentiable at some point in the program execution) and change type from `double` to built-in type `adouble`.
3. Define a set of independent variables using the output stream operator (<<=).
4. Define a set of dependent variables using the input stream operator (>>=).
5. Call the built-in driver function `jacobian` to compute first-order derivatives using reverse mode AD.
6. Compile the code including built-in header file `adolc.h` and linking with built-in library file `adolc.lib`.

All optimizations with and without AD were performed on a 1.73 GHz Pentium M laptop with 2.00 GB of RAM. The computation time performance was compared.

RESULTS

For each problem, performance comparisons of optimizing with and without AD are summarized in Table 1 and Table 2. The use of AD increased the computation time per objective function evaluation. However, the number of function evaluations necessary per optimization iteration was less with AD. For the system identification problem, the computation time required per optimization iteration with AD was approximately 20.5% (or reduced by a factor of 4.88) of the time required without AD. For the movement prediction problem, the computation time required per optimization iteration with AD was approximately 51.1% (or reduced by a factor of 1.96) of the time required without AD.

DISCUSSION

The main motivation for investigating the use of AD for biomechanical optimizations was to improve both speed and robustness of solutions. Speed improvement for the movement prediction optimization in particular was not as significant as anticipated. Further investigation is necessary to determine the effect of AD characteristics such as forward mode vs. reverse mode and source code transformation vs. operator overloading on computational speed. Whichever AD method is used, having analytical derivatives eliminates inaccurate search directions and sensitivity to design variable scaling which can plague optimizations that use finite difference gradients. If central (more accurate) instead of forward differencing was used in the movement prediction optimization without AD, the performance improvement would have been a factor of four instead of two.

Special dynamics formulations can also be utilized to compute analytical derivatives concurrently while evaluating the equations of motion, and the trade-offs between those approaches and AD require further investigation. While AD-based analytical derivatives may be less efficient computationally than those derived using special dynamics formulations, AD provides effortless updating of the derivative calculations should the biomechanical model used in the optimization be changed.

For large-scale problems, AD provides a relatively simple means for computing analytical derivatives to improve the speed and robustness of biomechanical optimizations.

ACKNOWLEDGMENTS

This study was funded by Whitaker Foundation and NIH National Library of Medicine (R03 LM07332-01) grants to B.J.F.

REFERENCES

1. Pandy, M.G., 2001, "Computer Modeling and Simulation of Human Movement," Annual Reviews in Biomedical Engineering, Vol. 3, pp. 245-273.
2. Griewank, A., 2000, *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*, Society for Industrial and Applied Mathematics, Philadelphia, PA.
3. Griewank, A., Juedes, D., and Utke, J., 1996, "ADOL-C: a Package for the Automatic Differentiation of Algorithms Written in C/C++," Association for Computing Machinery Transactions on Mathematical Software, Vol. 22, pp. 131-167.
4. Reinbolt, J.A. and Fregly, B.J., 2005, "Creation of Patient-Specific Dynamic Models from Three-Dimensional Movement Data Using Optimization," Proceedings, 10th International Symposium on Computer Simulation in Biomechanics, Cleveland, OH.
5. Fregly, B.J., Rooney, K.L., and Reinbolt, J.A., 2005, "Predicted Gait Modifications to Reduce the Peak Knee Adduction Torque," Proceedings, 20th Congress of the International Society of Biomechanics, Cleveland, OH, pp. 283.

Table 1. Performance results for system identification problem for a 3D kinematic ankle joint model involving 252 design variables and 1800 objective function elements.

Performance Criteria	Without AD	With AD
Time per Function Evaluation (s)	0.0189	0.978
Number of Function Evaluations per Optimization Iteration	252	1
Time per Optimization Iteration (s)	4.77	0.978

Table 2. Performance results for movement prediction problem for a 3D full-body gait model involving 660 design variables and 4100 objective function elements.

Performance Criteria	Without AD	With AD
Time per Function Evaluation (s)	0.217	73.1
Number of Function Evaluations per Optimization Iteration	660	1
Time per Optimization Iteration (s)	143	73.1