N90-23011

COMPUTATIONAL DYNAMICS FOR ROBOTICS SYSTEMS USING A NON-STRICT COMPUTATIONAL APPROACH

David E. Orin and Ho-Cheung Wong
Dept. of Electrical Engineering
The Ohio State University
Columbus, OH 43210

P. Sadayappan
Dept. of Computer & Information Science
The Ohio State University
Columbus, OH 43210

Abstract

This paper proposes a Non-Strict computational approach for real-time robotics control computations. In contrast to the traditional approach to scheduling such computations, based strictly on task dependence relations, the proposed approach relaxes precedence constraints and scheduling is guided instead by the relative sensitivity of the outputs with respect to the various paths in the task graph. An example of the computation of the Inverse Dynamics of a simple inverted pendulum is used to demonstrate the reduction in effective computational latency through use of the Non-Strict approach. A speedup of 5 has been obtained when the processes of the task graph are scheduled to reduce the latency along the crucial path of the computation. While error is introduced by the relaxation of precedence constraints, the Non-Strict approach has a smaller error than the conventional Strict approach for a wide range of input conditions.

I. Introduction

Complex robot dynamics computations have typically been represented using directed task precedence graphs, in order to facilitate the exploitation of parallelism in their execution [1,2,3]. The nodes of such a task graph represent computational modules, with the directed edges imposing a strict partial order on their execution sequence. While such a computational approach is faithfully accurate to the underlying physical model of the robot system if executed instantaneously, in practice the computational latency can be significant even with state-of-the-art computers. The use of pipelining also does not reduce this latency, even though it increases computational throughput.

The Non-Strict computational approach is motivated by a need to reduce the "effective latency" from input to output for complex robotics computations. The terms Strict and Non-Strict are derived from the manner in which precedence is treated in scheduling the tasks of a task graph such as the one shown in Fig. 1. The approach proposed here has the same motivation as the "Imprecise Results" approach recently proposed in [4], but uses a different model. The key concept behind our Non-Strict approach is that of relaxing the strict precedence constraints imposed by a conventional task-graph model, to facilitate trading off accuracy for timeliness in real-time computation over sampled, continuously varying inputs. Rather than sample all inputs simultaneously and then schedule tasks in strict adherence to the precedence constraints dictated by the edges of the task graph, tasks are scheduled so as to minimize delay between input and output along paths in the task graph that most strongly affect the output, perhaps using some previously computed values along less crucial paths.

Consider the task graph shown in Fig. 1. The circles represent computational tasks to be performed and the arcs represent data flow and thus precedence. According to the conventional Strict model of computation, the input, $\ddot{q}$, should first be sampled. Then it should proceed through the sequence of 1-2-3-4-5, completing the computation by furnishing the output, τ. The latency between input and output is just the total computation time; it represents a lag in the real-time control implementation and tends to destabilize the system.

If the output of process 5 is more strongly affected by the output of process 4 and further, if process 4 is strongly affected by the output of process 1 and only weakly dependent on process 3, then a better schedule, which does not strictly adhere to the natural precedence relationships, may be developed. In particular, it may be best to sample the input, compute process 1, resample the input, compute process 4 using the last computed value of process 3, and compute process 5 to generate the output. After process 5 is complete, then processes 2 and 3 may be scheduled which gives the following ordering on the computations 1-4-5-2-3 which does not follow strict precedence. However, if the 1-4-5 path from input to output is the most crucial to the computation, then the effective latency has been reduced. This is especially true if the computation times of processes 2 and 3 are long as compared to the others.

From the above example, several important characteristics of the Non-Strict computational approach may be noted:

1. It is appropriate to the case of real-time control in which the same computation is to be repeated over successive time samples of the inputs.
2. The main objective is to relax the precedence so as to reduce the effective latency from input to output. Rather than sampling the inputs once for a computation so that the results are strictly correct, albeit delayed, before each process which requires an input, a fresh sample of the input is taken so that the effects of delay are minimized. Also, computation generally proceeds along the crucial paths using the freshest (but past) values generated by the other less crucial processes. While this is often not according to precedence, again it can have the effect of reducing the latency.
3. While the Non-Strict approach does introduce error into the computation by violating task-graph precedence constraints, and can therefore only be an approximation of the

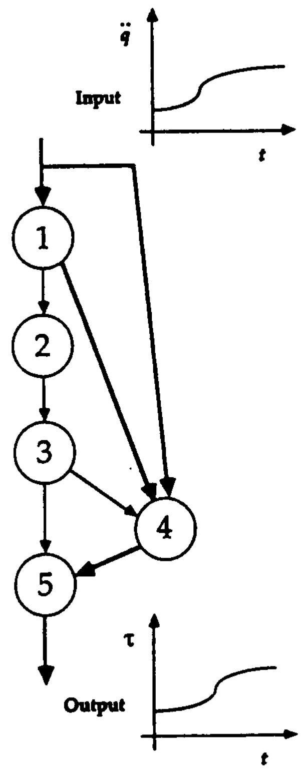


Figure 1: Example Task Graph to Illustrate Non-Strict Computation.

ideal output, the maximum instantaneous error and average error are often much less when compared to that resulting from the use of a Strict approach (which produces an output with the same shape as the ideal, but delayed in time).

4. By relaxing precedence constraints, potential parallelism may be increased. That is, the task precedence relationships constrain the flow of computation to certain sequences/orderings. With these relaxed, a greater amount of parallel scheduling of processes may proceed.

In order to further develop the basic concepts, an example of the dynamics of a simple inverted pendulum will be given. A description of the system, including its dynamic equation of motion and task graph for its evaluation, will be given in the next section. Following that, results using the Non-Strict computational approach will be given and will be compared with the Strict approach. Finally, the paper ends with a summary of the results, and conclusions on which to base further investigations.

II. Simple Inverted Pendulum Example

The computation of the dynamics of the simple inverted pendulum system shown in Fig. 2 is to be considered. The system consists of a single link which is connected through a one-degree-of-freedom revolute joint to a fixed base. Gravity acts in the vertical direction, and an actuator is mounted at the joint to power the pendulum movement. While the model is quite simple, it has been used in the past to study the dynamics of biped walking in the single support phase of locomotion [5].

The basic computational task is to solve the Inverse Dynamics problem [6] in real time. The dynamic equation of motion for the pendulum may be written as follows:

$$\tau = I\ddot{q} + B\dot{q} - mgl\ \sin(q) \tag{1}$$

where

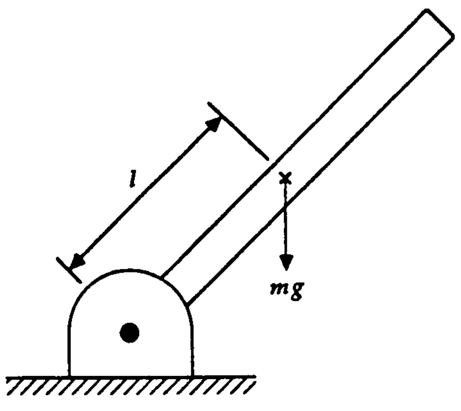
$$\begin{aligned}
\tau &= \text{Actuator torque,} \\
q, \dot{q}, \ddot{q} &= \text{Joint position (as referenced to the vertical),} \\
&\quad\ \text{velocity, and acceleration} \\
I &= \text{Moment of inertia of the link about the joint axis,} \\
B &= \text{Joint actuator damping coefficient,} \\
m &= \text{Mass of the link, and} \\
l &= \text{Position of the center of gravity of the link} \\
&\quad\ \text{from the joint axis.}
\end{aligned}$$

Note that values for the system parameters are also given in Fig. 2.

For Inverse Dynamics, the joint position, velocity, and acceleration are given and the joint torque is to be computed. Inverse Dynamics is used in advanced control schemes for robotic systems to determine the torque required for a desired motion trajectory and must be computed at real-time rates of up to a few hundred hertz [7].

A task graph for the Inverse Dynamics computation is shown in Fig. 3. There are 6 processes with the top number in the circle giving the process number. The operation performed is given in the middle section of the circle while the computation time is given in the bottom part and is normalized to units of a basic computation time, Δ. Note that it is assumed that adds and multiplies take a single unit of time and that the sine computation takes ten times as long. For the purposes of this example, it is assumed that the input position, q, is a simple sinusoid:

$$q(t) = \sin(\omega t)\,. \tag{2}$$



m	=	20 lbm
l	=	1.5 ft
I	=	1.86 ft-lb-s^2
B	=	30 ft-lb-s

Figure 2: Simple Inverted Pendulum System.

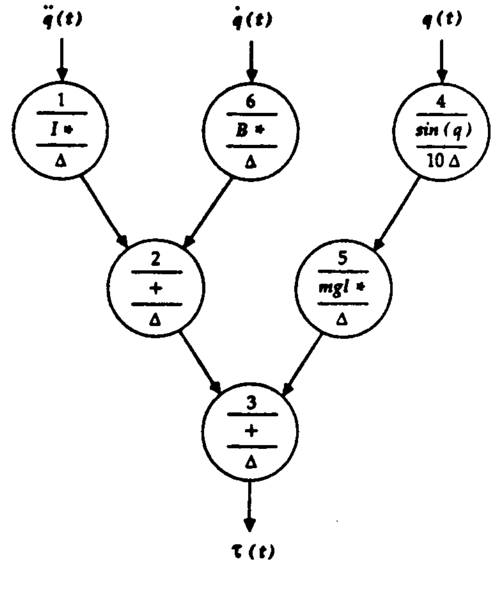


Figure 3: Task Graph for Inverse Dynamics Computation.

where ω is the frequency.

III. Computational Results

In this section, using the Inverse Dynamics of an inverted pendulum as an example, a number of results will be given which illustrate the Non-Strict computational approach and compare it with the Strict approach. The output of the computation under the Strict and Non-Strict approach are contrasted for a simple sinusoidal input to the system. If the computational delays in evaluating the component operations in the task graph were zero, then both the Strict and Non-Strict approaches would yield identical results. With the Strict approach, the generated output is identical in shape to the ideal output, but shifted in time by the sum of the computational delays of the tasks in the task graph. With the Non-Strict approach, the output (in general) approximates the shape of the ideal output but is not identical in form. However, the overall error (including the effects of time delay) can often be significantly less than the error with the Strict computation.

In this section, the two approaches are compared using two measures – 1) *mean square error*, and 2) *effective latency*. The *mean square error* over a period is defined as:

$$mean\ square\ error\ =\ \frac{1}{T}\int_0^T [\tau_i(t) - \tau(t)]^2\, dt \tag{3}$$

where T is the period of the input and $\tau_i(t)$ is the ideal output. The *effective latency* Δt_{eff} is defined in the following way:

$$\Delta t_{eff}\ =\ \Delta t \quad : \quad Min\ \frac{1}{T}\int_0^T [\tau_i(t-\Delta t) - \tau(t)]^2\, dt. \tag{4}$$

Thus a best fit is attempted between a delayed version of the ideal output and the Non-Strict output, and the shift in the delayed version of the ideal output is interpreted as an effective latency of the computation. As long as the mean square error between the output and the shifted ideal output is sufficiently small, the error in output may be related to that arising from a time delay of the ideal output. Thus, the outputs of the Strict and Non-Strict approaches may be compared in terms of effective computational latencies, where the scheme with the lower effective latency is preferable.

In the following, speedup through reduction in the effective latency of the computation will be shown. The relationship of the error in both the Strict and Non-Strict cases will be given as a function of the input frequency. The objective is to determine a range of frequencies for which the Non-Strict approach will give speedup while maintaining reasonable limits on the error. Finally, the relationship between the effective latency and the input frequency will be investigated and will provide motivation for adaptive scheduling strategies.

ORIGINAL PAGE IS
OF POOR QUALITY

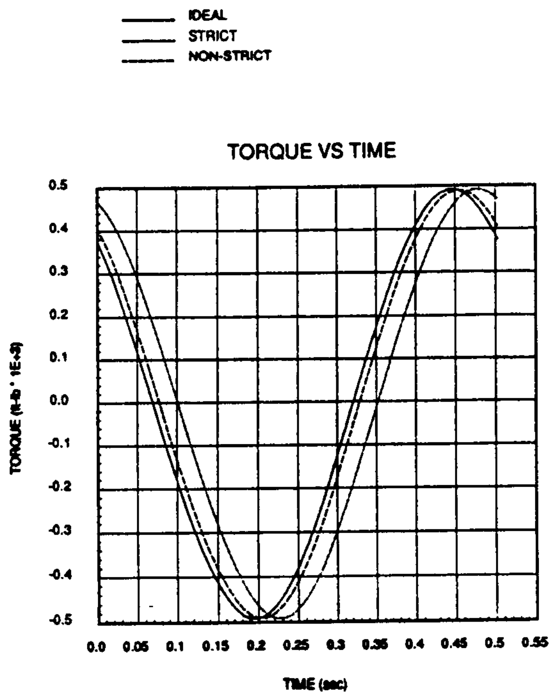


Figure 4: Plot of the Torque for the Ideal, Strict, and Non-Strict Cases Over a Period ($\omega = 12.5\ rad/s$ and $\Delta = \pi/1500\ s$).

A. Reduction in the Effective Latency

Fig. 4 compares the outputs of Strict and Non-Strict evaluations of the task graph of Fig. 3 against the ideal output obtained with zero computational delay. Results for $\omega = 12.5\ rad/s$ and $\Delta = \pi/1500\ s$ are given. If the tasks are scheduled in the order 1-2-3-4-5-6 instead of an order dictated by strict adherence to task precedence constraints, then speedup should result by a reduction of the effective latency of the computation. In particular, at high frequencies when the inertial term dominates the torque computation, it is anticipated that the computational delay may be as little as 3Δ since this is the most crucial path in that case. Of course, the schedule is not according to strict precedence and some amount of error will be introduced.

Note that the Strict curve is an exact form of the ideal but delayed in time by 15Δ. While the Non-Strict curve is not an exact form of the ideal, it gives the least amount of error for almost the entire period. For the present results, little difference can be seen between the shape of the ideal and Non-Strict curves. In fact, the Non-Strict curve does not quite reach the desired peak and the slopes are slightly different along the trajectory. However, the effective latency has been reduced considerably to as little as 20-30% of the Strict case.

ORIGINAL PAGE IS
OF POOR QUALITY

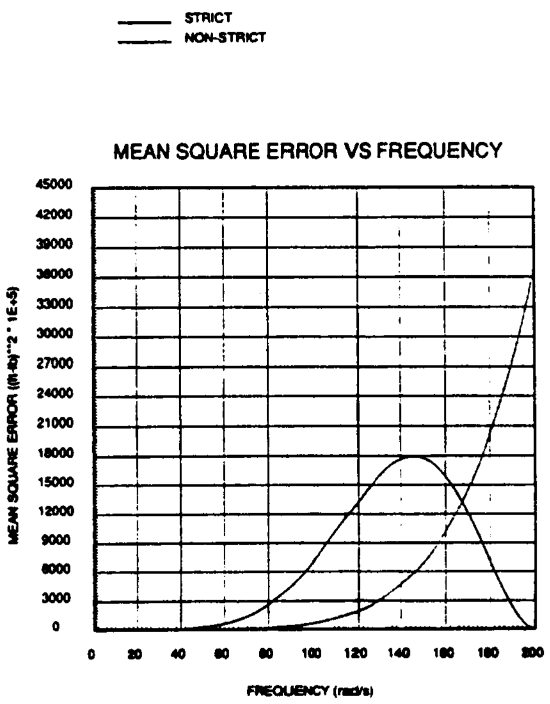


Figure 5: Mean Square Error for the Strict and Non-Strict Cases as a Function of the Frequency ($\Delta = \pi/1500\ s$).

B. Error Vs. Frequency

If the frequency is increased such that the period is exactly 15Δ, then the Strict approach will give perfect results. The question becomes: is there a range of frequencies over which the Non-Strict approach will give the best results (least error)? Toward this end, the error of the Strict and Non-Strict approaches with respect to the ideal have been evaluated as a function of the frequency. Typical results are given in Fig. 5.

The results indicate that the Non-Strict approach has a smaller error up to a frequency of approximately $\omega = 168\ rad/s$ when $\Delta = \pi/1500$. As might be expected, this crossover frequency varies with the value of Δ chosen. As the value of Δ increases, the crossover value of frequency decreases. In fact, as long as the value for Δ is less than a certain fraction of the period, then the Non-Strict approach will be better. For the example system, this ratio has been determined as:

$$\frac{\Delta}{T} \leq 0.056. \tag{5}$$

For this example, if the Strict computation delay (15Δ) is less than approximately 80% of

ORIGINAL PAGE IS
OF POOR QUALITY

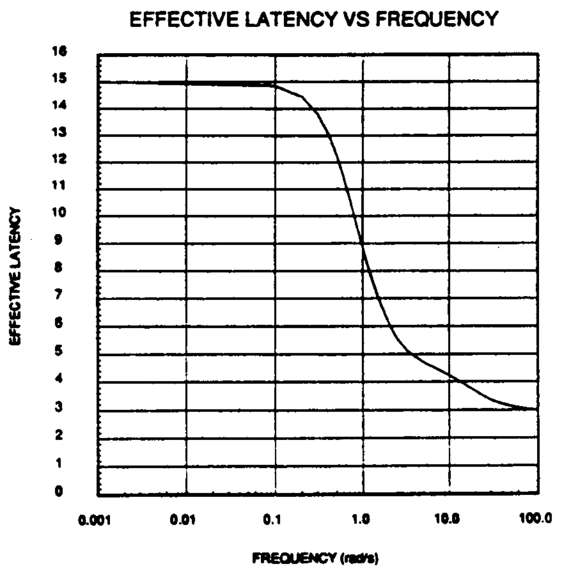


Figure 6: Effective Latency as a Function of Frequency for the Schedule (1-2-3-4-5-6).

the period, then the Non-Strict approach will give better results.

C. Effective Latency Vs. Frequency

The schedule assumed for the previous results tends to work well for high input frequencies, since in this region the crucial path is 1-2-3 and is scheduled first. Then for very high frequencies, it is anticipated that the overall effective latency will be approximately 3Δ which is the computational delay along this particular path. At low frequencies, the gravitational term will dominate the inertial term. With the same schedule, then, it is anticipated that the effective latency will be longer. In fact, the delay from the input q to the output τ is seen to be 15Δ. Therefore, the effective latency of the computation should be in the range of $3 - 15\Delta$ and will vary with the frequency.

Results have been obtained for the effective latency as a function of the frequency and are shown in Fig. 6. As expected, the effective delay is 15Δ at low frequencies and decreases to 3Δ at high frequencies. The crossover takes place in the region between $\omega = 0.1$ to $\omega = 10.0$ during which the significance of the inertial, damping, and gravitational terms changes.

If operation of the inverted pendulum is at lower frequencies, then an alternate schedule may be more appropriate. In particular, the schedule 4-5-3-1-6-2 should have an effective latency of 12Δ at low frequencies – better than the previous schedule. Results for this case are given in Fig. 7. The results are as expected. Note that the delay at high frequencies is 15Δ because of the long delay from sampling $\ddot{q}$ to the output. Also note that the damping

ORIGINAL PAGE IS
OF POOR QUALITY

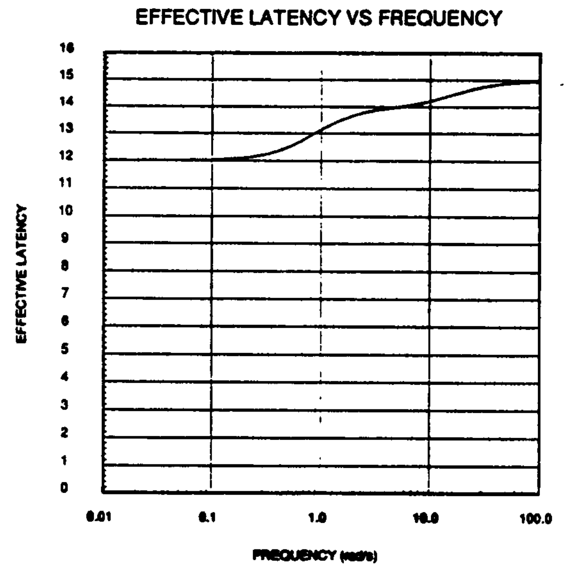


Figure 7: Effective Latency as a Function of Frequency for the Schedule (4-5-3-1-6-2).

term with $\dot{q}$ appears to dominate at approximate $\omega = 4$ so that the effective delay is 14Δ.

In fact, in general, the schedule should vary depending upon the rate of change of the inputs, and their effect on the output. For low frequencies, the path 4-5-3 should be favored while for high frequencies, the path 1-2-3 should be favored. This gives rise to a need for an adaptive scheduling scheme and will be considered in future investigations.

IV. Summary and Conclusions

This paper has proposed a Non-Strict computational approach attractive for real-time applications such as robotics control. In this approach, the precedence set in a task graph model of the computation is relaxed so as to reduce the effective latency from input to output. This is achieved by appropriately ordering the execution of the processes so that the most crucial path from input to output is given priority. While some amount of error is introduced since the precedence relationships are not strictly adhered to, there are many cases in which the overall error is less than that introduced by the sheer delay of the Strict case.

The Non-Strict approach has been applied to compute Inverse Dynamics [6] for a simple inverted pendulum. At high frequencies of motion, the effective latency was reduced from 15Δ to 3Δ, giving a speedup of 5. For a simple sinusoidal input on the joint angle, the error for the Non-Strict case was better than the Strict case as long as the Strict computational delay was less than approximately 80% of the period. This is the case in practical applications. The effective latency for the Non-Strict case was shown to be a function of the frequency of the sinusoidal input. This results from a change in the crucial path through the task graph as the frequency varies. Two different schedules showed the range of possibilities for reducing the effective latency and motivated the need for an adaptive scheduling scheme.

There are a number of avenues for further work which are presently being explored. First of all, speedup has been obtained here for the case of a serial machine. The use of a Non-Strict computational approach can clearly be expected to have an impact on exploitation of parallelism. The strict precedence constraints imposed by a conventional task-graph model tend to limit parallelism. The relaxation of precedence constraints with the Non-Strict approach should thus be conducive to the better exploitation of parallelism. However, in the parallel context, the determination of an optimal scheduling order becomes more difficult, with the two inter-related criteria of 1) task ordering for minimization of "effective latency" through use of a measure of "task crucialness", and 2) minimization of computational latency through maximization of processor utilization.

A second issue is that of adaptive scheduling. Based on the rate of change of a process output and its significance to the computation, a constantly adapting schedule should give better results than one which is fixed. Also, processes need not be scheduled at the same rate and such multi-rate schemes may be especially important in effectively utilizing the resources of a parallel processor system. Further work in scheduling strategies is needed.

Another area in which the work may be extended involves the use of prediction to reduce the effective latency and computational error. In particular, if prediction is made on the inputs before use by a process, then it may be possible to reduce the effective process latency and therefore the overall latency. Investigations into appropriate schemes for using the prediction are needed.

Finally, the applicability of the Non-Strict computational approach to a wider class of real-time control problems should be investigated. The preliminary results presented in this paper show much promise for speedup and should be applied to other difficult real-time computational problems.

The Non-Strict computational approach seems promising as a first step towards a framework for specifying real-time robot dynamics computations in a machine-independent manner. It is current practice to choose the model/algorithm with a view to what is implementable to provide real-time response when executed on a given target machine. The flexible and adaptive scheduling of tasks inherent with the Non-Strict approach suggests that a single algorithmic representation can be adequate for slow and fast processors alike, for single as well as multi-processors. This seems quite feasible especially if computationally inexpensive predictors/extrapolators are used as alternatives in conjunction with more accurate but computationally demanding task blocks. A powerful target machine might schedule the computationally demanding blocks every time needed and thus obtain great accuracy, whereas a weaker target machine might schedule those computationally intensive blocks relatively infrequently and use the predictors in between, thereby trading off accuracy for timeliness.

Work is currently in progress in evaluating the Non-Strict approach with more complex robot dynamics computations, and in investigating the use of predictors and multi-rate task scheduling strategies. A testbed is also under development on a BBN Butterfly multiprocessor to investigate Non-Strict scheduling in a parallel setting. It is hoped that with further work the Non-Strict approach can be established as an effective approach for complex real-time domains such as robotics control.

Acknowledgements

This work was supported by the National Science Foundation under Computational Engineering Grant No. EET-8718434.

References

[1] J. Barhen, "Robot Inverse Dynamics on a Concurrent Computation Ensemble," in *Proc. of 1985 ASME International Conference on Computers in Engineering*, pp. 415–429, Boston, MA, August 1985.

[2] J. Y. S. Luh and C. S. Lin, "Scheduling of Parallel Computation for a Computer-Controlled Mechanical Manipulator," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-12, no. 2, pp. 214–234, March 1982.

[3] D. E. Orin, K. W. Olson, and H. H. Chao, "Systolic Architectures for Computation of the Jacobian for Robot Manipulators," in *Computer Architectures for Robotics and Automation*, J. H. Graham, Ed., pp. 39–67, New York: Gordon and Breach Science Publishers, 1987.

[4] K. J. Lin, S. Natarajan, and J. W. S. Liu, "Imprecise Results: Utilizing Partial Computations in Real-Time Systems," in *Proc. of 8th IEEE Real-Time Systems Symposium*, pp. 210–217, December 1987.

[5] H. Hemami, I. C. Wall, F. O. Black, and G. L. Golliday, "Single Inverted Pendulum Biped Experiments," *Journal of Interdis. Model. & Simulation*, vol. 2, no. 3, pp. 211–227, 1985.

[6] J. Y. S. Luh, M. W. Walker, and R. P. C. Paul, "On-line Computational Scheme for Mechanical Manipulator," *ASME Journal of Dynamic Systems, Measurement, and Control*, vol. 102, pp. 69–76, June 1980.

[7] C. H. An, C. G. Atkeson, and J. M. Hollerbach, *Model-Based Control of a Robot Manipulator.* Cambridge, MA: The MIT Press, 1988.