N89-20072

MODEL EVALUATION, RECOMMENDATION, AND PRIORITIZING OF FUTURE WORK FOR THE MANIPULATOR EMULATOR TESTBED

Final Report

NASA/ASEE Summer Faculty Fellowship Program--1988

Johnson Space Center

Prepared By:	Frederick A. Kelly, Ph.D.
Academic Rank:	Assistant Professor
University & Department:	Michigan Technological University Mechanical Engineering - Engineering Mechanics Department Houghton, Michigan 49931

NASA/JSC

Directorate:	Engineering
Division:	Systems Development and Simulation
Branch:	Teleoperator Systems
JSC Colleague:	Edith C. Taylor
Date Submitted:	August 12, 1988
Contract Number:	NGT 44-005-803

ABSTRACT

The Manipulator Emulator Testbed (MET) is to provide a facility capable of hosting the simulation of various manipulator configurations to support concept studies, evaluation, and other engineering development activities. Specifically, the testbed is intended to support development of the Space Station Remote Manipulator System (SSRMS) and related systems. The MET is required to permit that components simulated in software may be replaced by future hardware components.

The objective of this study is to evaluate the math models developed for the MET simulation of a manipulator's rigid body dynamics and the servo systems for each of the driven manipulator joints. Specifically, the math models are examined with regard to their amenability to pipeline and parallel processing processing. Based on this evaluation and the project objectives, a set of prioritized recommendations are offered for future work.

INTRODUCTION

The Manipulator Emulator Testbed (MET) is to provide a facility in which different manipulator configurations may be simulated. The MET will be used as a tool to support Space Station manipulator development. It will be used to develop and evaluate concepts, support design and development studies, and evaluate hardware components and flight software modules. The MET is required to be designed such that initial software simulated components may be replaced by future hardware elements [1] - [3].

The MET is currently built around a network of IBM PC-AT computers. Each computer operates at an 8-MHz clock rate and is equipped with an Intel 80287 math co-processor, 640K bytes of memory, an extended graphics adapter (EGA) card, a color monitor, a 30-MB hard disk drive, and a high-capacity floppy disk drive. Each PC is equipped with a National Instruments GPIB-PCAA IEEE-488 interface card. The network includes an operator control station, data recording and display capability, and hardcopy output. A Multibus II "Network-in-a-Box" is scheduled to be added to the MET in August 1988.

The software includes the operating system (iRMK), intercomputer communication software from National Instruments, external interface, executives, operator support, data recording, and math model modules. The math model application software includes (1) a multi-link manipulator rigid body dynamics model and (2) joint servo models for each of the driven manipulator joints. The MET software is intended to provide a simulation of proposed manipulator designs. The math model software is distributed across the PC's such that a hardware component may be substituted and consequently the math model removed.

The objective of this study is to evaluate the math models developed for the MET simulation of a manipulator's rigid body dynamics and the servo systems for each of the driven manipulator joints. Specifically, the math models were examined with regard to their amenability to pipeline and parallel processing. Based on this evaluation and the project objectives, a set of prioritized recommendations are offered for future work.

MODEL EVALUATION

A recursive rigid body dynamics formulation was developed for real-time simulation on the MET by Nasser [4]. Nasser considers it to be a generalization of a method due to Armstrong [5]. The formulation is general in the sense that translational or rotational joint types can be modeled. Nasser also claims that the model assumes a topological tree configuration, but doesn't describe how this would be done. Procedures described in [6] - [7] are particularly suitable to extending Nasser's method to general topological trees.

The procedure for solving for the reaction force and torques at each joint and the joint accelerations given the external forces and moments on each link and the actuator forces and moments can be described by referring to Figure 1 [4].

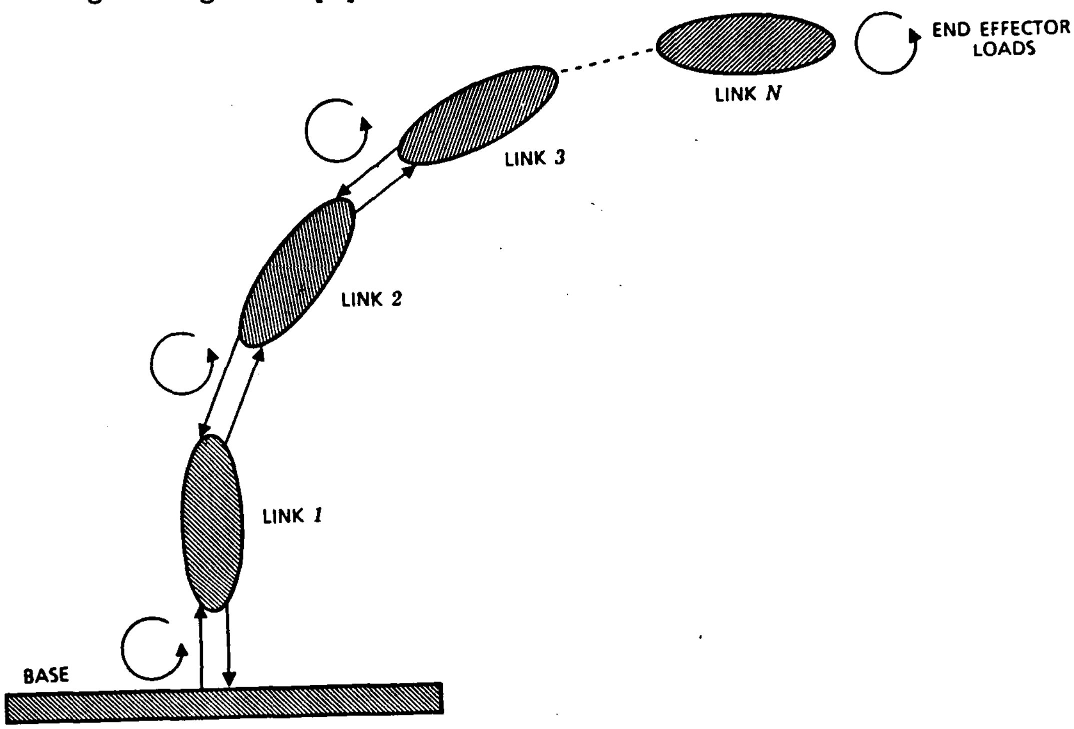

Figure 1. Open Kinematic Chain

First, set up the equations of dynamic equilibrium for link N and then solve for the reaction forces and moments and the joint acceleration in terms of the remaining variables. The remaining variables include the proximal links' joint displacements velocities and accelerations and the distal links' applied forces and moments, displacements, and velocities. Next, proceed to link N-1 and substitute the expressions for the reactions that involve link N-1 joint accelerations into the equations of dynamic equilibrium for link N-1. The reaction forces and moments exerted by link N-2 on link N-1 and the joint acceleration of link N-1 should then be solved for in terms of the remaining variables. This step is repeated until link 1 is reached. The joint acceleration for link 1 is then obtained in terms of the distal links' external forces and moments, actuator forces and moments and state variables, the state of link 1, and the base state. Then move distally to link 2 and substitute for the link 1 joint acceleration to find the joint 2 acceleration. This step is repeated until link N is reached and all the joint accelerations are found.

A free body diagram for any link i of the open kinematic chain is shown in Figure 2.

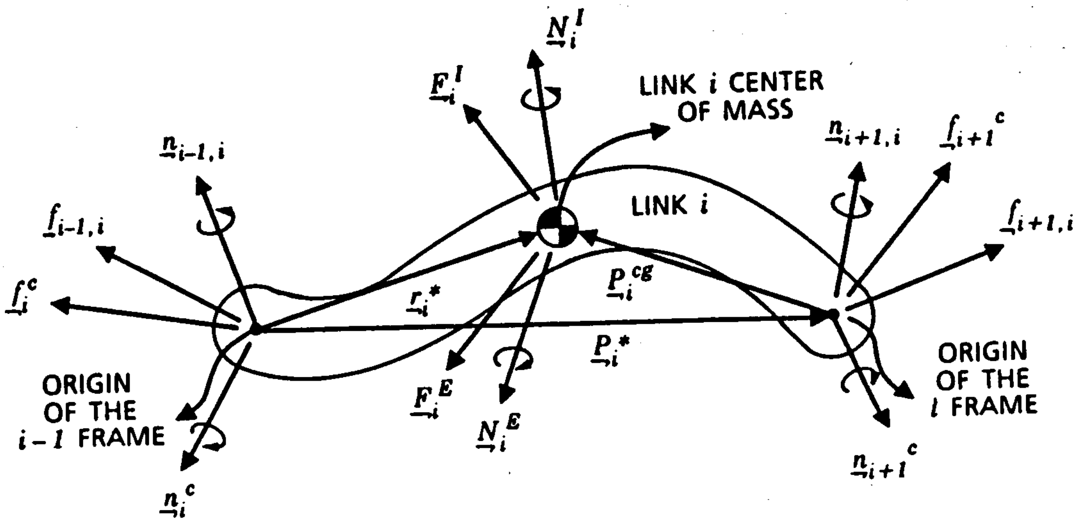

Figure 2. Link Free Body Diagram

where

$\underline{F}_i^I$ is the link i inertia force,

$\underline{N}_i^I$ is the link i inertia torque,

$\underline{f}_i^C$ is the control force exerted on link i at joint i-1,

$\underline{n}_i^C$ is the control torque exerted on link i at joint i-1,

$\underline{f}_{i+1}^C$ is the control force exerted on link i+1 at joint i,

$\underline{n}_{i+1}^C$ is the control torque exerted on link i+1 at joint i,

$\underline{f}_{i+1,i}$ is the reaction force that link i+1 exerts on link i,

$\underline{n}_{i+1,i}$ is the reaction moment that link i-1 exerts on link i

$\underline{f}_{i-1,i}$ is the reaction force that link i-1 exerts on link i,

$\underline{n}_{i-1,i}$ is the reaction torque that link i-1 exerts on link i,

$\underline{F}_i^E$ is the sum of the external forces acting on link i referred to the mass center of link i,

$\underline{N}_i^E$ is the sum of the external moments acting on link i when the external forces are referred to the mass center of link i,

$\underline{r}_i^*$ is the position vector of the mass center of link i relative to the origin of the i-1 frame,

$\underline{p}_i^{cg}$ is the position vector of the mass center of link i relative to the origin of the i frame,

$\underline{p}_i^*$ is the position vector of the origin of frame i relative to the origin of frame i-1

A number of researchers have investigated pipeline and parallel implementations of the equations of motion for various purposes. Luh and Lin [8] described a procedure for scheduling the subtasks of a group of microprocessors computing the inverse dynamics using the Newton-Euler equations of motion. One microprocessor is assigned to each manipulator link. A variable branch-and-bound search finds an optional subtask-ordered schedule for each of the microprocessors. However, the total processing time of solving the minimum-time scheduling problem could not be easily reduced to a manageable level.

Lathrop [9] proposed two parallel algorithms for solving the inverse dynamics problem using a group of special-purpose processors. One is a linear Newton-Euler algorithm, which is most closely related to the method proposed by Luh and Lin. The other is a logarithmic parallel Newton-Euler algorithm based on the "partial sum" technique, which achieves on $O(\lceil \log_2 N \rceil)$ total processing time. However, both methods have two main effects that deteriorate the performance of parallel computations. They both require potentially massive internal buffering to achieve pipelined computation between forward and backward recursions. They both involve complex intercommunication and bussing, which frequently cause data to be fetched, and as a result data for operand pairs are not properly aligned for parallel computations.

Lee and Chang [10] proposed an algorithm for solving the inverse dynamics problem of an N-link manipulator using p processors in parallel on the Newton-Euler equations of motion. It was the "recursive doubling" technique to achieve a total processing time of $O(k_1 \lceil N/p \rceil + k_2 \lceil \log_2 p \rceil)$, where k_1 and k_2 are constants. When p = N, the algorithm is $O(\lceil \log_2 N \rceil)$, the same as Lathrop. The p-fold parallel algorithm consists of p-parallel blocks with pipelined elements within each parallel block. The results from the p-parallel blocks form a homogeneous linear recurrence of size p. The parallel algorithm can also be implemented in a systolic pipelined architecture, requiring three floating-point operations for each complete set of joint torques.

Binder and Herzog [11] described an algorithm based on the recursive Newton-Euler equations of motion where the parallel computations are distributed over multiple computing elements, one for each joint. Concurrency is achieved by substituting "predicted" values for the actual values of variables involved in the recursive equations. The authors simulated this method and compared it to other approaches such as simplification of the dynamic equations.

Amin-Javaheri and Orin [12] proposed systolic architectures consisting of 1, N, and N(N+1)/2 processors for computing the inertia matrix of an N degree of freedom manipulator. A VLSI-based Robotics Processor is being developed as the fundamental component of the architecture. The algorithm used is based on recursive computation of the inertial parameters of sets of composite rigid bodies and is programmed to exploit any inherent parallelism.

Lee and Chang [13] proposed two parallel algorithms for computing the forward dynamics for real-time simulation with N processors for an N degree of freedom manipulator. The first algorithm is based on Walker and Orin's [14] composite rigid-body method. It generates the inertia matrix using the parallel Newton-Euler algorithm, the parallel linear recurrence algorithm [10], and the modified row-sweep algorithm, and then inverts the inertia matrix to obtain the joint acceleration vector. The time complexity of this parallel algorithm is of the order $O(N^2)$ with $O(N)$ processors. Further reduction of the order of time complexity can be achieved by implementing the Cholesky factorization procedure on VLSI array processors to invert the symmetric, positive-definite, inertia matrix. The second parallel algorithm is based on Walker and Orin's [14] conjugate gradient method.

Of the methods just described, only Nasser's method [4] was formulated to allow for hardware substitution in a simulation. Currently, it is implemented on the MET in a sequential algorithm with one IBM PC-AT. The Nasser method appears to be a variation of a method by Featherstone [15]. The A_i^* matrix in Nasser's method seems to correspond to the $\hat{I}_i^A$ matrix in Featherstone's method. This would imply that Nasser's method has the same nonlinear recurrence that makes Featherstone's method impossible to solve with known methods of parallel solution of recurrence problems [13]. Kung [16] has shown that any parallel algorithm using any number of processors cannot be essentially faster than the obvious sequential algorithm for any nonlinear polynomial recurrence problem. If not for this, the recursive doubling technique of Kogge and Stone [17] could have been used. The method of Nasser does not seem to be particularly adaptable to pipelining either. The calculation of the A_i^* matrix alone requires about 42% of the total computations. The other processors in the pipeline would have to be idle, waiting for the A_i^* to complete.

The hardware that makes up the MET limits the use of pipelining. Instruction pipelining is possible on the Intel 80286 only by prefetching instructions. As an alternative, the MET would seem to be an ideal application for the new Intel 80960 architecture which allows parallel and out-of-order instruction execution of 32-bit operations at 20 MHz [18].

The Multibus II "Network-in-a-Box" promises to greatly speed up interprocessor communications with initially three Intel 80386 processors on board. The National Instruments IEEE-488 interface card and software are much too slow.

RECOMMENDATION AND PRIORITIZING OF FUTURE WORK

My recommendation of future work for the MET will be listed in order of decreasing priority.

1. The Nasser algorithm does not appear to be amenable to pipeline or parallel processing. One option would be to run it as a very fast sequential algorithm on a fast processor. Another option would be to modify the method of Binder and Herzog [11] and use prediction to estimate values, instead of waiting for a complete set of computed values to be ready.

2. The simulation should include the rigid body dynamics of the base link. A mobile base, where motion relative to either a fixed base or base with rigid body dynamics should be included in the model.

3. The servo systems are decoupled in their current design. This makes these computations naturally parallel. The rigid body model of Nasser could run on a fast processor and broadcast results to N processors, one for each of the N servo systems, operating in parallel.

4. Flexible links should be added to the model. For open topological trees, this should be straight forward since the complete force system acting on each link is known. A beam element model of a flexible manipulator is described in Kelly and Huston [19].

5. Constrained motion should be modeled so that closed loops can be simulated. Rigid links should be simulated first and then flexible links.

REFERENCES

1. Van Valkenburg, J. A.: Manipulator Emulator Testbed Development Project Plan. LEMSCO-23536, January 1987.

2. Schindeler, R. E.: Manipulator Emulator Testbed Verification Plan. LEMSCO-24095, October 1987.

3. Rob, K. K.: Manipulator Emulator Testbed Verification Report. LEMSCO-24113, May 1988.

4. Nasser, M. G.: Recursive Newton-Euler Formulation of Manipulator Dynamics. LEMSCO-24673, February 1988.

5. Armstrong, W. W.: Recursive Solution to the Equations of Motion of an N-Link Manipulator. Proceedings of the 5th World Congress on the Theory of Machines and Mechanisms, Volume 2, Montreal, July 1979, pp. 1343-1346.

6. Huston, R. L., Passerello, C. E., and Harlow, M. W.: Dynamics of Multirigid-Body Systems. Trans. SME, J. Applied Mechanics, Volume 45, December 1978, pp. 889-894.

7. Kelly, F. A.: On the Dynamics of Flexible Multibody Systems. Ph.D. Dissertation, University of Cincinnati, June 1982.

8. Luh, J. Y. S., and Lin, C. S.: Scheduling of Parallel Computation for a Computer-Controlled Mechanical Manipulator. IEEE Trans. on Systems, Man, and Cybernetics, Volume SMC-12, No. 2, March/April 1982.

9. Lathrop, R. H.: Parallelism in Manipulator Dynamics. The International Journal of Robotics Research, Volume 4, No. 2, Summer 1985, pp. 80-102.

10. Lee, C. S. G., and Chang, P. R.: Efficient Parallel Algorithm for Robot Inverse Dynamics Computation. IEEE Trans. on Systems, Man, and Cybernetics, Vol. MC-16, No. 4, July/August 1986, pp. 532-542.

11. Binder, E. E., and Herzog, J. H.: Distributed Computer Architecture and Fast Parallel Algorithms in Real-Time Robot Control. Ibid., pp. 543-549.

12. Amin-Javaheri, M., and Orin, D. E.: A Systolic Architecture for Computation of the Manipulator Inertia Matrix. Proceedings of the 1987 IEEE International Conference on Robotics and Automation, Vol. 2, pp. 647-653.

13. Lee, C. S. G., and Chang, P. R.: Efficient Parallel Algorithms for Robot Forward Dynamics Computation. IEEE Trans. on Systems, Man, and Cybernetics, Volume 18, No. 2, March/April 1988.

14. Walker, M. W., and Orin, D. E.: Efficient Dynamic Computer Simulation of Robot Mechanisms. Trans. ASME J. Dynamic Systems, Measurement and Control, Volume 104, September 1982, pp. 205-211.

15. Featherstone, R.: The Calculation of Robot Dynamics Using Articulated-Body Inertia. The International Journal of Robotics Research, Volume 2, No. 1, Spring 1983, pp. 13-30.

16. Kung, H. T.: New Algorithms and Lower Bounds for the Parallel Evaluation of Certain Rational Expressions and Recurrences. Journal of the Association for Computing Machinery, Volume 23, No. 2, April 1976, pp. 252-261.

17. Kogge, P. M., and Stone, H. S.: A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations. IEEE Trans. on Computers, Volume C-22, No. 8, August 1973, pp. 786-793.

18. Ryan, D. P.: Intel's 80960: An Architecture Optimized for Embedded Control. IEEE Micro, June 1988, pp. 63-76.

19. Kelly, F. A., and Huston, R. L.: Statics and Dynamics of a Flexible Manipulator. Proceedings of the 5th ASME International Computers in Engineering Conference, Boston, MA, 1985.