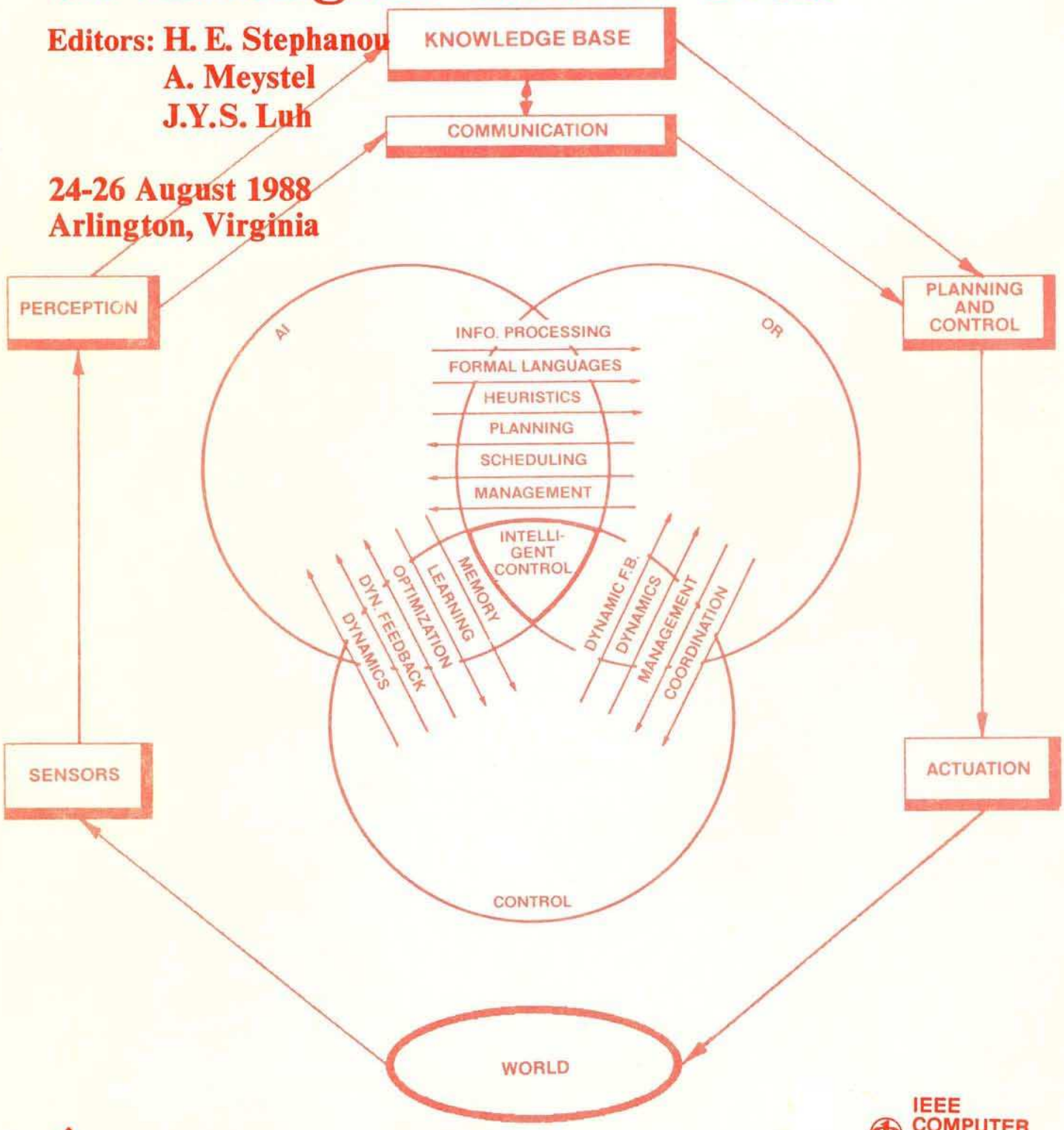
Proceedings

IEEE International Symposium on Intelligent Control 1988

Editors: H. E. Stephanou
A. Meystel
J.Y.S. Luh

24-26 August 1988
Arlington, Virginia



The Institute of Electrical and Electronics, Engineers, Inc.

IEEE COMPUTER SOCIETY PRESS

AN EFFICIENT COMPUTATIONAL SCHEME FOR ROBOT MANIPULATORS

J.A. Tenreiro Machado, J.L. Martins de Carvalho and J.A. Silva Matos
Faculty of Engineering of University of Porto
Dept. of Electrical and Computer Engineering
4099 Porto Codex, Portugal

Antonio M.C. Costa
INESC-Norte, Largo de Monpilher 22
4000 Porto, Portugal

Abstract-A new robot manipulator computational scheme which is a blend of ordinary and Boolean algebra is presented. This method may also be interpreted as a dedicated compiler that optimizes the on-line computing time at expenses of the off-line stage. The off-line requirements are alleviated, by the implementation of some general rules that stem from the structure of the robot manipulator equations, and the on-line computing time is optimized through the use of Binary Decision Diagrams. The results show a considerable computational improvement on conventional sequential machines but, furthermore, they clearly point out new computational parallel architectures. Finally, it is observed that the proposed algorithm is not restricted to robot dynamic computations, but is also applicable to many other computing structures.

I. Introduction

The robot manipulator inverse dynamics problem, has been an active field of research for the last two decades. Solutions for speeding up the on-line calculation of the required joint torques have first focused on numerical recursive methods and later on the use of symbolic computation techniques. Numerical recursive algorithms, based on the Lagrangian and on the Newton-Euler methods were presented in [1,2] and [3] respectively. A mixed algorithm using both numerical and symbolic computations was proposed by Horak [4], and the automatic generation of the dynamic symbolic formulae, using LISP-based computer algebra systems, was presented in [5,6]. In addition to a considerable improvement in computational efficiency and performance, symbolic solutions have been shown to allow a better insight into manipulator dynamics, providing strategies for the design of the robot itself [7]. More recently Neuman and Murray [8-10] developed the concept of customized computing. Yet, these methods share some fundamental limitations in so far as:

-Both involve conversion to some form of low level code, by means of a general purpose high level language compiler;

-They require floating point arithmetic with precision (and word length) much larger than the one associated with the typical accuracy of manipulator hardware (sensors, A/D, D/A, etc);

-They require a large number of floating point operations and transcendental function evaluations.

These considerations suggest that if computations could be performed with similar precision to the manipulator hardware, then calculation time would decrease. Therefore, the use of a lower accuracy computation without finite precision problems implies that the arithmetic operations, as well as the transcendental functions, can not be executed in the ordinary way; moreover, an alternative method must be simple enough for any microprocessor to perform, that is it should be well adapted to the microprocessor's machine code instruction set. Then we are led to the conclusion that there is the need for a "special" algebra that provides a better management of the existing hardware/software resources. That is to say, we need a new (dedicated) compiler that generates a more efficient object code in the manipulator hardware/software environment.

Clearly, Boolean algebra satisfies all the above requirements; nevertheless, we have to find a way of translating the ordinary arithmetic and transcendental formulae to Boolean algebra. The robot manipulator dynamic algorithms correspond to multivariable functions, and, as such, can be numerically tabulated. If both input and output variables are quantified and converted to a suitable binary code, then the resulting table may be interpreted, alternatively, as a truth table where one can use standard Boolean function simplifying techniques, having as input variables the bits of the quantified binary coded position, velocity and acceleration vectors, and output Boolean functions the bits of the quantified binary coded required actuator torques (Fig. 1). Unfortunately, a further examination reveals that this ideal situation is not feasible due to the dimension of the corresponding Boolean table. Nevertheless, this philosophy points to a strategy for a feasible procedure that, with modifications, can achieve a remarkable reduction of the on-line computing time. This method is developed in the following sections.

II. Algorithm description

It was found that the full Boolean-based computation is impractical, therefore we have to carry out a compromise between feasibility and on-line computational optimization. Such compromise may be obtained through the use of a hybrid computation, having both ordinary arithmetic sums and Boolean algebra [11-13].

The dynamic equations for a n d.o.f. manipulator are of the form [2,14]

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

where $J(q)$ is the n x n inertial matrix
$C(q,\dot{q})$ is the n dimensional vector of Coriolis and centripetal torques
$G(q)$ is the n dimensional vector of gravitational torques
q is the n dimensional vector of link positions
$\dot{q}$ is the n dimensional vector of link velocities
$\ddot{q}$ is the n dimensional vector of link accelerations

A natural way of achieving our objective to consider each link torque terms separately. Then each term can be calculated by Boolean

0-8186-2012-9/89/0000/0435$01.00 © 1989 IEEE

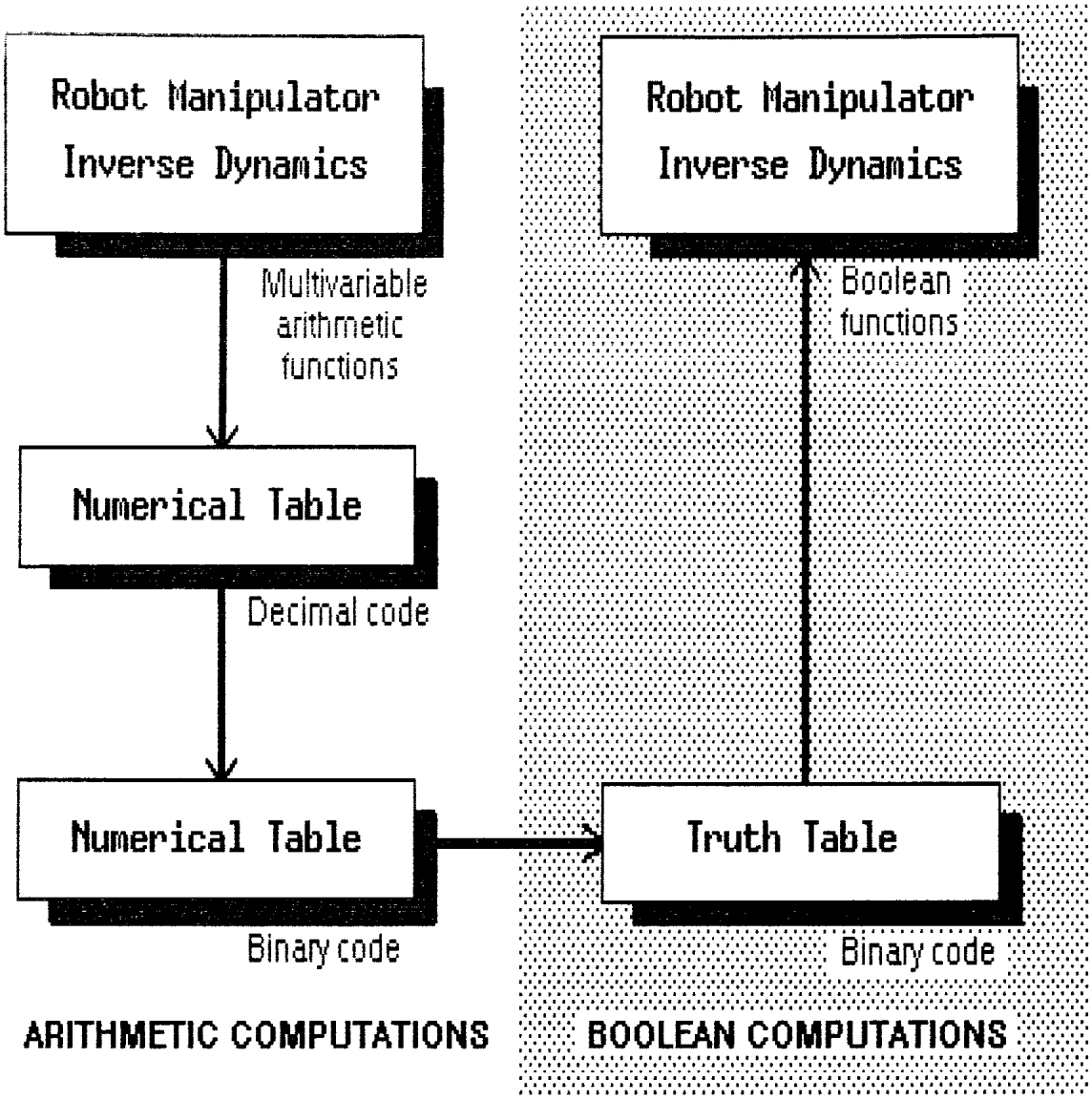


Fig. 1. Scheme for the conversion of the robot inverse dynamics, expressed in ordinary arithmetic, up to Boolean algebra.

algebra, and the final result, that is each actuator torque, by an ordinary (binary) arithmetic sum of the aforementioned terms. Now, for each joint torque there are several truth tables, requiring up to a maximum of

n+1 i.w.v. for the inertial terms
n+2 i.w.v. for the Coriolis terms
n+1 i.w.v. for the centripetal terms
n i.w.v. for the gravitational terms

where input word variable (i.w.v.) means the appropriate binary-coded representation of each input variable. The resulting Boolean computation becomes strongly alleviated, as we have to deal with much smaller truth tables, in contrast with the requirement of 3n input word variables for the fully Boolean-based computation.

We must note that the coexistence of two different algebras imposes some restrictions, namely due to the fact that arithmetic summation degrades the overall precision. This must be compensated for by the addition of m extra bits given by the expression

$$m=\text{int}[\log_2(p+1)], \quad p=\text{total number of terms} \qquad (2)$$

in each term, in order to achieve the desired accuracy. Nevertheless, this problem is of minor influence since m increases much more slowly than p. Consequently, for a desired output resolution of r bits, each term must have W bits, where

$$W=r+m \qquad (3)$$

Concluding, we may say that the "compilation" (i.e. truth table simplification) requirements, both in computing time and computer memory space, are considerably alleviated, as we pass from one huge Boolean table with 3n input word variables, to several truth tables, each with a much smaller number of input word variables, and this without significant modification on the on-line computing time.

III. Implementation Example

In this section the hybrid algorithm is implemented on a 2R (n=2) robot manipulator

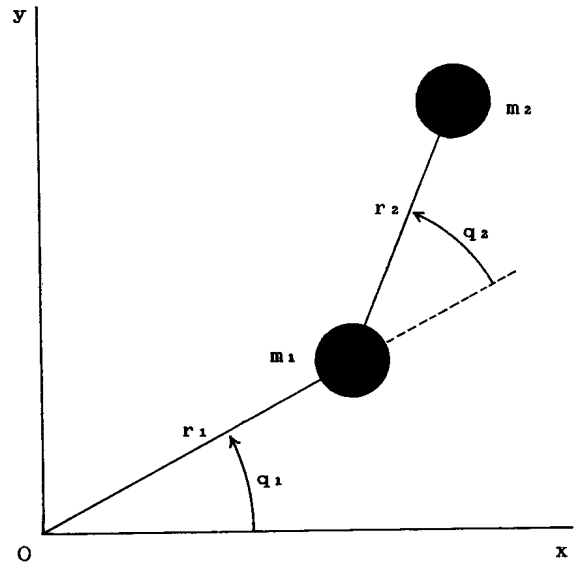


Fig. 2. 2R robot manipulator

(Fig. 2). The corresponding dynamic equations are of the form

$$J(q)=\begin{bmatrix} (m_1+m_2)r_1^2+m_2r_2^2+2m_2r_1r_2C_2+J_1 & m_2r_2^2+m_2r_1r_2C_2 \\ m_2r_2^2+m_2r_1r_2C_2 & m_2r_2^2+J_2 \end{bmatrix} \quad (4a)$$

$$C(q,\dot{q})=\begin{bmatrix} -m_2r_1r_2S_2\dot{q}_2^2-2m_2r_1r_2S_2\dot{q}_1\dot{q}_2 \\ m_2r_1r_2S_2\dot{q}_1^2 \end{bmatrix} \quad (4b)$$

$$G(q)=\begin{bmatrix} g(m_1+m_2)r_1C_1+gm_2r_2C_{12} \\ m_2gr_2C_{12} \end{bmatrix} \quad (4c)$$

with $C_1=\cos(q_1)$, $C_{12}=\cos(q_1+q_2)$, $C_2=\cos(q_2)$ and $S_2=\sin(q_2)$. Similarly to other studies [12-14], the manipulator parameters were set to

$$m_1=0.5\ \text{Kg},\ m_2=6.25\ \text{Kg} \quad (5a)$$
$$r_1=1\ \text{m},\ r_2=0.8\ \text{m} \quad (5b)$$
$$J_1=5\ \text{Kgm},\ J_2=5\ \text{Kgm} \quad (5c)$$

and we assume that the amplitude of each link variable obeys (i=1,2)

$$-\pi\ \text{rad} \le q_i \le \pi\ \text{rad} \quad (6a)$$
$$-1\ \text{rad/s} \le \dot{q}_i \le 1\ \text{rad/s} \quad (6b)$$
$$-1\ \text{rad/s}^2 \le \ddot{q}_i \le 1\ \text{rad/s}^2 \quad (6c)$$

For these ranges and for the payload referred in (5) it comes

$$-151.5\ \text{Nm} \le T_1 \le 152.5\ \text{Nm} \quad (7a)$$
$$-69.1\ \text{Nm} \le T_2 \le 69.1\ \text{Nm} \quad (7b)$$

Considering the torque computation as a subsystem of a larger control architecture, then we may have higher torques, and therefore we assume

$$-200\ \text{Nm} \le T_1 \le 200\ \text{Nm} \quad (8a)$$
$$-100\ \text{Nm} \le T_2 \le 100\ \text{Nm} \quad (8b)$$

Summing up, we have equations (6) and (7) as the quantization ranges for the input and output word variables, respectively. Now, for the the robot dynamics (4), we may find the appropriate binary coded numerical tables, and from that the corresponding Boolean formulae. Both stages can be further optimized: the table requirements are alleviated through the implementation of some rules presented in [13], and the Boolean code on-line calculation, speeded-up using Binary Decision Diagrams (BDD's) [18,19].

With the BDD method, Boolean functions are computed using IF_THEN_ELSE structures, which for v (single bit) input variables need only a maximum of v evaluations, resulting in a much more efficient code for complex Boolean formulae.

Figure 3 depicts a hi-lo chart comparing the histograms of the required computing time using, either the conventional arithmetic method or the proposed algorithm. By a and b we represent the terms $-m_2r_1r_2S_2\dot{q}_2^2$ and $g(m_1+m_2)r_1C_1$ of T_1, and by c and d the terms $(m_2r_2^2+J_2)\ddot{q}_2$ and $m_2r_1r_2S_2\dot{q}_1^2$ of T_2, respectively. The codes were written in Turbo Pascal V4.0, and running on a 8086, 8 MHz machine, under MSDOS V3.2. The simple term c imposes a similar load to both methods, nevertheless the more complex terms a, b and d show a remarkable improvement when using the new algorithm. This means that for complex terms like those appearing in manipulators with more d.o.f., the speed-up factor resulting on the use of the proposed method will be even higher.

The "compilation" of terms depending on several input variables may be alleviated if we use a cascading procedure. In fact, tacking for our guideline example the centripetal torque $-2m_2r_1r_2S_2\dot{q}_1\dot{q}_2$ appearing on the first link, we have two options. In the first case we use one truth table; therefore, for a required output precision of W bits we need 3W input bits (Fig. 4a). Alternatively, we may cascade two blocks as depicted in Fig. 4b. The corresponding truth tables only require 2W input bits and, consequently, will impose much lower compiling requirements. Moreover, the on-line computing time will be of the same order of magnitude, as now we have two computing stages, but each one much simpler then the initial case.

On the other hand, the table generation may be obtained from experimental data instead of being model based. This is of utmost importance, since situations appear where mathematical models are inaccurate or difficult to derive. Therefore, our algorithm is well suited for the computation of a large set of physical phenomenae, the only restriction being the size of the binary tables, which as shown, depend solely on the required precision and the number of input variables.

IV. On Parallel Computing Structures

It was shown that the proposed algorithm offers considerable improvement over purely arithmetic alternatives, even in a monoprocessor sequential machine. In this section we show that the algorithm also leads naturally to simple parallel architectures allowing unlimited speed-up of the on-line computations, without accuracy degradation.

The operations involved in the on-line computations required by our algorithm allow a simple distribution of the computat[illegible]

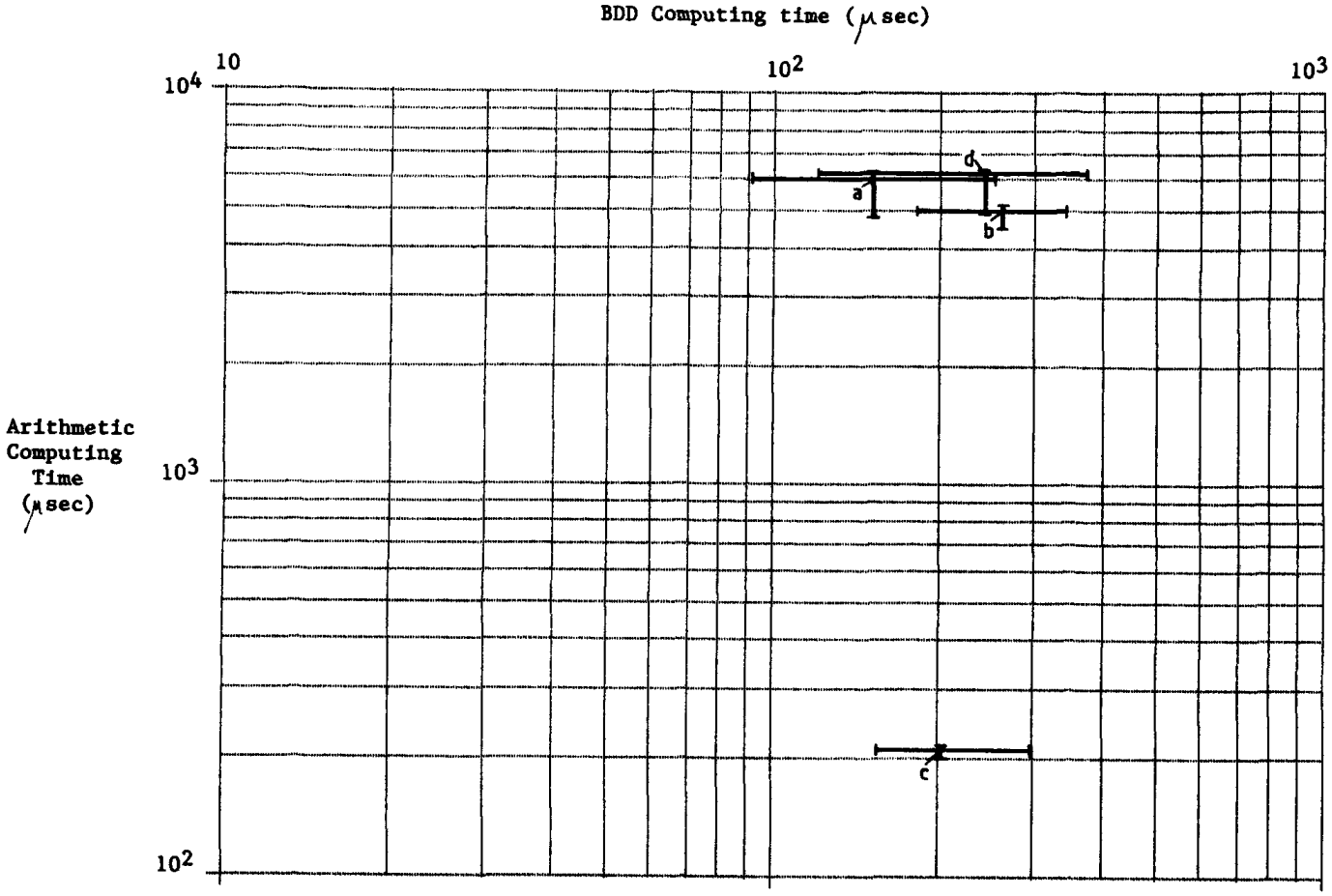
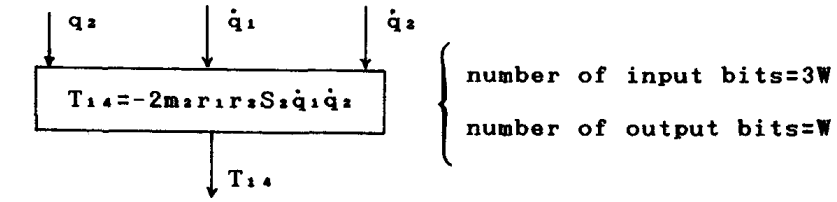
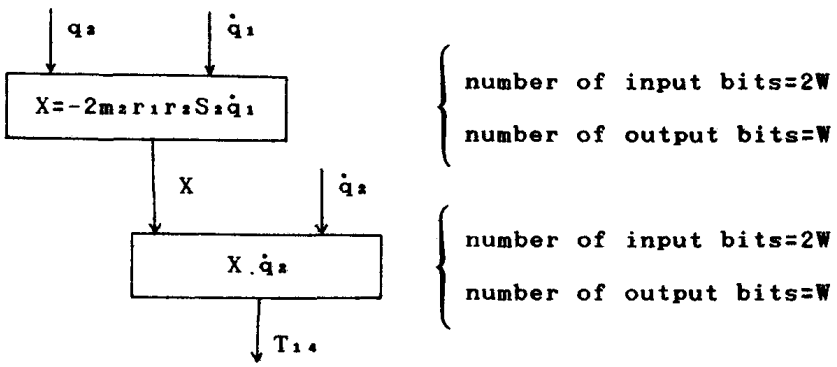


Fig. 3. Arithmetic vs. BDD computing time for the terms a, b, c and d. The hi-lo bars are centred on the average calculation time. $a=-m_2r_1r_2S_2\dot{q}_2^2$, $b=g(m_1+m_2)r_1C_1$, $c=(m_2r_2^2+J_2)\ddot{q}_2$, $d=m_2r_1r_2S_2\dot{q}_1^2$



a)



b)

Fig. 4. Alternative strategies for the BDD computation of $-2m_2r_1r_2S_2\dot{q}_1\dot{q}_2$
a) Single block procedure
b) Cascading structure (two blocks).

amongst several processors. In fact, this can be achieved without any of the complex scheduling problems, associated with arithmetic manipulator inverse dynamics parallel computing structures [20-23]. Moreover, the improvement of the on-line computing time is proportional to the number of processors, and it is not subjected to any theoretical limit. The importance of this fact must be emphasized since the other manipulator inverse dynamics parallel computing structures, only achieve a limited improvement, as the resulting speed-up is not proportional to the number of processors, but is restricted to a maximum of n (a condition that is achieved only if some errors are allowed [24]).

Finally, it should be noted that the BDD formulae are bit oriented instead of word oriented. Consequently, general purpose microprocessors with one, two or four byte data buses, and large instructions sets (unused in this algorithm), that require several clock cycles, are of little use in improving the overall computing performance. Much more promising seems to be the use of single bit, reduced instruction set and special purpose microprocessors.

Also, one should observe that the new method is not restricted to the present case study, but is also applicable to many other system descriptions.

V. Conclusions

In this paper a new hybrid computational algorithm was developed and illustrated with a 2R robot manipulator.

The proposed method is very efficient because it takes full advantage of both hardware and software capabilities of the robot manipulator system. Another important consequence of this method is the natural appearance of simple, yet powerful, parallel computing structures. Finally, it is observed that the dedicated compiler philosophy is not restricted to robot manipulator computations, but can be successfully generalized to many other computing structures, as long as we can redefine the management of the corresponding "environmental resources". This may lead to optimization procedures, having implications on either sequential or parallel computing system structures.

References

[1] J. M. Hollerbach, "A Recursive Lagrangian Formulation of Manipulator Dynamics and a Comparative Study of Dynamics Formulation Complexity," IEEE Trans. Syst., Man, Cybern., vol. SMC-10, pp. 730-736, Nov. 1980.

[2] R. P. Paul, Robot Manipulators: Mathematics, Programming and Control. Cambridge, MA: Mass. Inst. Tech., 1981.

[3] J. Y. S. Luh, M. W. Walker and R. P. C. Paul, "On-Line Computational Scheme for Mechanical Manipulators," ASME J. Dynamic Syst. Meas. Contr., vol. 102, pp. 69-76, June 1980.

[4] D. T. Horak, "A Simplified Modeling and Computational Scheme for Manipulator Dynamics," ASME J. Dynamic Syst. Meas. Contr., vol. 106, pp. 350-353, Dec. 1984.

[5] M. C. Leu and N. Hemati, "Automated Symbolic Derivation of Dynamic Equations of Motion for Robotic Manipulators," ASME J. Dynamic Syst. Meas. Contr., vol. 108, pp. 172-179, Sept. 1986.

[6] J. Koplic and M. C. Leu, "Computer Generation of Robot Dynamic Equations and the Related Issues," J. of Robotic Systems, vol. 3, n. 3, pp. 301-319, Fall 1986.

[7] D. C. Yang and S. W. Tzeng, "Simplification and Linearization of Manipulator Dynamics by the Design of Inertia Distribution, " The Int. J. Robotics Research, vol. 5, n. 3, pp. 120-127, Fall 1986.

[8] Charles P. Neuman and John J. Murray, "The Complete Dynamic Model and Costumized Algorithms of the Puma Robot," IEEE Trans. Syst., Man, Cybern., vol. SMC-17, pp. 635-644, July/Aug. 1987.

[9] Charles P. Neuman and John J. Murray, "Customized Computational Robot Dynamics" J. of Robotic Systems, vol. 4, n. 4, pp. 503-526., Aug. 1987.

[10] Charles P. Neuman and John J. Murray, "Symbolically Efficient Formulations for Computational Robot Dynamics," J. of Robotic Systems, vol. 4, n. 6, pp. 743-769, Dec. 1987.

[11] J. A. Tenreiro Machado, J. L. Martins de Carvalho, J. A. Silva Matos and Antonio M. C. Costa, "A Real-Time System for Robot Manipulator Inverse Dynamics Computation," 15th IFAC/IFIP Workshop on Real-Time Programming, Valencia, Spain, 1988.

[12] J. A. Tenreiro Machado, J. L. Martins de Carvalho, Antonio M. C. Costa and Jose S. Matos, "Dedicated Computer System for Robot Manipulators," 3rd Int. Symp. on Systems Analysis and Simulation, Berlin, GDR, 1988.

[13] J. A. Tenreiro Machado, J. L. Martins de Carvalho, Antonio M. C. Costa and Jose S. Matos, "Robot Manipulator Dynamics - Towards Better Computational Algorithms," IFAC Symp. on Robot Control, Karlsruhe, FRG, 1988.

[14] Michael Brady, John M. Hollerbach, Timothy L. Johnson, Tomas Lozano-Perez and Mattew T. Mason, Robot Motion: Planning and Control. Cambridge, MA: Mass. Inst. Tech., 1982.

[15] Kar-Keung Young, "Controller Design for a Manipulator Using Theory of Variable Structure Systems," IEEE Trans. Syst., Man, Cybern., vol. SMC-8, pp. 101-109, Fev. 1978.

[16] Russel G. Morgan and Umit Ozguner, "A Decentralized Variable Structure Control Algorithm for Robotic Manipulators," IEEE J. Robotics and Automation, vol. RA-1, pp. 57-65, March 1985.

[17] J. A. Tenreiro Machado and J. L. Martins de Carvalho, "A Smooth Variable Structure Control Algorithm for Robot Manipulators" IEE Control'88 Conference, Oxford, U.K., 1988.

[18] Jose S. Matos, "The Binary Decision Diagram: A Tool for Logic Design and Implementation," PhD Dissertation, Syracuse University, Syracuse, N. Y., 1983.

[19] Jose S. Matos and John V. Oldfield, "Binary Decision Diagrams: From Abstract Representations to Physical Implementations," 20th IEEE/ACM Design Automation Conference, Miami Beach, Florida, 1983.

[20] Chang Huan Liu and Yen-Ming Chen, "Multi-Microprocessor-Based Cartesian-Space Control Techniques for a Mechanical Manipulator," IEEE J. Robotics and Automation, vol RA-2, pp. 110-115, June 1986.
[21] J. Y. S. Luh and C. S. Lin, "Scheduling of Parallel Computation for a Computer-Controlled Mechanical Manipulator," IEEE Trans. Syst., Man, Cybern., vol. SMC-12, pp. 214-234, March/April 1982.
[22] Ravi Nigam and C. S. George Lee, "A Multiprocessor-Based Controller for the Control of Mechanical Manipulators," IEEE J. Robotics and Automation, vol. RA-1, pp. 173-182, Dec. 1985.
[23] T. Watanabe, M. Kametani, K. Kawata and K. Tetsuya, "Improvement in the Computing Time of Robot Manipulator Using a Multi-microprocessor," ASME J. Dynamic Syst. Meas. Contr., vol. 108, pp. 190-197, Sept. 1986.
[24] Eli E. Binder and James H. Herzog, "Distributed Computer Architecture and Fast Parallel Algorithms in Real-Time Robot Control," IEEE Trans. Syst., Man, Cybern., vol. SMC-16, pp. 543-549, July/Aug. 1986.