# Mathematical Research

# Systems Analysis and Simulation 1988 II

edited by
A. Sydow · S. G. Tzafestas
R. Vichnevetsky

Band 47

AKADEMIE-VERLAG BERLIN

# DEDICATED COMPUTER SYSTEM FOR ROBOT MANIPULATORS

J. A. Tenreiro Machado [1,2], J. L. Martins de Carvalho [1,2], Antonio M. C. Costa [2] and Jose S. Matos [1,2]

## 1. INTRODUCTION

In the last decade, robot dynamic computational algorithms have been steadily improved. Lagrangian [1] and Newton-Euler [2,3] recursive numerical algorithms, Horak's mixed calculation [4], Lisp based symbolic derivation of the robot closed form equations [5,6] and recently, costumized computing algorithms [7-9] have been important stages towards more efficient procedures. Nevertheless, the still existing high computational loads, have prevented the industrial application of such algorithms. A new robot computational system which overcomes those limitations is presented. This system is motivated by the following considerations:

-Symbolic formulae pose a high computational burden, due to the large number of arithmetic and transcendental computations.

-These computations, usually performed at a high level compiled computer language are complex, namely, they are compiled to a large number of the microprocessor's machine code instructions.

-These computations must be performed with very high precision; in order to reduce problems associated with finite precision arithmetics, a large word lengh is required.

-There are two situations, as far as precision is concerned: the computer internal representation having very high precision contrasting with the computer interface to the outside world, usually consisting of 8 to 16 bit precision.

A better "management" of the existing hardware resources, should bear in mind the following points:

-The computations should be performed with another algebra simple enough for any processor to perform.

-Computations in this algebra should'nt suffer from finite precision problems, consequently requiring a smaller number of bits.

We conclude on the necessity of a "special" algebra that provides a better management of the existing hardware/software resources. Or to state, alternatively, our position: we need to find a new compiler that generates a more efficient object code in the manipulator hardware/software environment. Such a compiler may be called a dedicated compiler, contrasting with general purpose compilers such as Fortran or C. The Boolean algebra satisfies such requirements. The robot inverse dynamics are converted to an input/output binary table, and afterwords translated to Boolean algebra, through the simplification of the corresponding Boolean truth table. Unfortunately, upon further examination it turned up that this ideal situation is not feasible in practice, as it poses critical obstacles to its computation due to the necessity of implementing a huge Boolean table. Nevertheless, it points to a methodology for a realizable implementation that, with modifications, can achieve a remarkable reduction of the on-line computing time. This realizable (suboptimal) implementation is developed in the next section.

## 2. A HYBRID COMPUTATIONAL ALGORITHM FOR ROBOT MANIPULATOR DYNAMICS

Noting that the dynamic equations, for a manipulator with n degrees of freddom, are of the form [1,10]

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

a natural way of achieving a realizable implementation is to split each link torque in its terms. Then each term can be calculated by Boolean algebra, and the final result, i.e. the link torque, by an ordinary arithmetic sum of these terms. Note that for the truth tables required on each Boolean-based term computation, we have the following maximum of input word variables (i.w.v.):

| | |
|---|---|
| [n(position)+1(acceleration)] for the inertial terms | [n(position)+1(velocity)] for the centripetal terms |
| [n(position)+2(velocities)] for the Coriolis terms | n(position) for the gravitational terms |

where i.w.v. means the appropriate binary-coded representation of each input variable. Therefore, the Boolean computation becomes alleviated as we have a maximum of n+2 i.w.v., contrasting with the requirement of 3n i.w.v. for the fully Boolean-based computation.

One restriction imposed by the coexistence of two different algebras, is that arithmetic summation degrades the overall precision, hence resulting in the need for m extra bits, given by

[1] Faculdade de Engenharia da Universidade do Porto, Dep. Eng. Electrotecnica Computadores, R. dos Bragas, 4099 Porto Codex, PORTUGAL
[2] INESC-Instituto de Engenharia e Sistemas de Computadores, R. Jose Falcao 15, 4º, 4000 Porto, PORTUGAL.

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

in each term in order to achieve the desired accuracy. Nevertheless, this effect is of small influence since m increases much more slowly than p.

In conclusion, we may say that the off-line "compilation" (i.e. truth table simplification) requirements, either in computing time or in computer memory space, are considerably alleviated, as we pass from one huge Boolean table with 3n i.v.v. to several truth tables, each having a much smaller number of i.v.v., and this without altering significantly the on-line computing time.

## 3. THE NEW COMPUTATIONAL ALGORITHM IMPLEMENTATION ON A 2R ROBOT MANIPULATOR

To illustrate the implementation of our algorithm, we consider a 2R robot manipulator with the following dynamic model:

$$T_1=[(m_1+m_2)r_1^2+m_2r_2^2+2r_1r_2m_2C_2+J_1]\ddot{q}_1+(m_2r_2^2+r_1r_2m_2C_2)\ddot{q}_2-$$
$$-r_1r_2m_2\dot{q}_2^2-2r_1r_2m_2S_2\dot{q}_1\dot{q}_2+m_1gr_1C_1+m_2g(r_1C_1+r_2C_{12})$$
$$T_2=(m_2r_2^2+r_1r_2m_2C_2)\ddot{q}_1+(m_2r_2^2+J_2)\ddot{q}_2+r_1r_2m_2S_2q_1^2+m_2gr_2C_{12} \qquad (3)$$

In our study we use the same data as Young [11], Russel et al. [12] and Tenreiro et al. [13], namely:

$$m_1=0.5\ \text{Kg};\quad m_2=6.25\ \text{Kg};\quad r_1=1\ \text{m};\quad r_2=0.8\ \text{m};\quad J_1=5\ \text{Kgm};\quad J_2=5\ \text{Kgm} \qquad (4)$$

and assume the amplitude of each link variable within the ranges:

$$-\pi\,\text{rad}\le q_i\le\pi\,\text{rad};\quad -1\,\text{rad/s}\le\dot{q}_i\le 1\,\text{rad/s};\quad -1\,\text{rad/s}^2\le\ddot{q}_i\le 1\,\text{rad/s}^2;\quad i=1,2 \qquad (5)$$

For these ranges and for the load $(m_2,J_2)$ referred in (4) we have: $-152.5\ \text{Nm} \le T_1 \le 152.5\ \text{Nm}$ and $-69.1\ \text{Nm} \le T_2 \le 69.1\ \text{Nm}$; nevertheless, motivated by the natural assumption that the torque computation shall be a block of a larger control structure, the feedback loop may demand higher torques, and therefore we assume

$$-200\ \text{Nm} \le T_1 \le 200\ \text{Nm};\ -100\ \text{Nm} \le T_2 \le 100\ \text{Nm} \qquad (6)$$

From these considerations it results that equations (5) and (6) give the quantization ranges for the input and output word variables respectively.

The truth table simplification may be alleviated by the use of some general rules. These rules stem from considerations like:

a) The quantization ranges (6) must be the same for all terms of each equation. As usual, amplitude range of each term only covers part of the quantization interval, and the remainder becomes "unused".

b) The accuracy compensating extra number of bits m, as represented in equation (2), varies in discrete steps, and much more slowly than p.

c) Bearing the physical (mathematical) robot manipulator requirements (specifications) in mind [14], it would appear that the Gray code is the more suitable and this, indeed, was confirmed by experimentation.

d) For even arithmetic functions of a single input variable, it was observed that the most significant bit of the corresponding input word variable could be dropped out if the Gray code was used.

e) On terms which have several input variables, the required final precision implies a similar accuracy on each input word variable.

The application of these rules to $T_2$ gives the results depicted in Fig. 1. The on-line computing time can be further optimized by the use of Binary Decision Diagrams (B.D.D.) [15,16]. Figure 2 shows an histogram of the required computational time for the term $T_{24}=r_1r_2m_2S_2\dot{q}_1^2$, comparing the conventional arithmetic method with the new algorithm. Both codes were written in Turbo Pascal V4.0, and running on a 8086 8 MHz machine under MSDOS V3.2. Notice the remarkable improvement achieved with the new algorithm.

## 4. PARALLEL COMPUTATION

We have shown that the proposed hybrid computational algorithm offers considerable performance improvement over purely arithmetic alternatives, even in a monoprocessor sequential computing environement. 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 restrictions. In fact, the operations involved in the on-line computations required by our algorithm, allow simple distribution of the computational load amongst several processors. Indeed, this can be achieved without any of the complex scheduling problems, common to arithmetic manipulator inverse dynamics parallel computing structures [17-19]. Parallel architectures like the one depicted in Fig. 3, are a natural consequence of the calculation decoupling allowed by our algorithm. The improvement in the on-line computing time is proportional to the number of processors, and it is not subject to any theoretical limit. The importance of this fact must be emphasized since the other manipulator inverse dynamics parallel computing structures achieved a limited improvement, in the sense that the resulting speed-up is not proportional to the number of processors, and, is in fact, restricted to a maximum of n (a condition that is achieved only if some errors are allowed [20]). Finally, it should be noted that the R.D.D. algorithms are bit oriented instead of word oriented. Consequently, general purpose microprocessors with 8, 16 or 32 data buses, and large instructions sets (unused in this algorithm), that require several clock cycles for each of the machine code instructions, are of little use in improving the overall computing performance. Much more promising is the use of single bit, reduced instruction set and special purpose microprocessors. The design of a dedicated processor based on R.D.D., is being investigated.

## 5. CONCLUSIONS

A new computational algorithm for robot manipulator inverse dynamics was presented. This algorithm is very efficient because it takes full advantage of both hardware and software capabilities of the robot manipulator system. Another important consequence of the proposed calculation method is the natural appearance of simple, yet powerful, parallel computing structures.

## REFERENCES


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

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

[3] John 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, Nov. 1980, pp. 730-736.

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

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

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

[7] 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, July/Aug. 1987, pp. 635-644.

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

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

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

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

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

[13] 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., April, 1988.

[14] J. A. Tenreiro Machado, Antonio M. C. Costa and J. L. Martins de Carvalho, Robot Manipulator Dynamics - Towards Better Computational Algorithms, INESC Internal Report, 1987.

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

[16] 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.

[17] Chang-Huan Liu, Yen-Ming Chen, Multi-Microprocessor-Based Cartesian-Space Control Techniques for a Mechanical Manipulator, IEEE J. Robotics and Automation, vol RA-2, June 1986, pp. 110-115.

[18] J. Y. S. Luh, C. S. Lin, Scheduling of Parallel Computation for a Computer-Controlled Mechanical Manipulator, IEEE Trans. Syst., Man, Cybern., vol. SMC-12, March/April 1982, pp. 214-234.

[19] T. Watanabe, H. Yamatani, K. Kawata, I. Tetsuya, Improvement in the Computing Time of Robot Manipulators Using a Multimicroprocessor, ASME J. Dynamic Syst. Meas. Contr., vol. 108, Sept. 1986, pp. 190-197.

[20] Eli E. Binder, James M. Herzog, Distributed Computer Architecture and Fast Parallel Algorithms in Real-Time Robot Control, IEEE Trans. Syst., Man, Cybern., vol. SMC-16, July/Aug. 1986, pp. 543-549.

| r:6 | r:8 , m:2 :> $W_{INITIAL}$:10 bits | | | | | | PRECISION |
|---|---|---|---|---|---|---|---|
| $T_2$ | $T_{21}^I$ | $T_{22}^I$ | $T_{23}$ | $T_{24}$ | $T_{25}^I$ | $T_{26}^I$ | TERM |
| 8 | 6 | 6 | 7 | 6 | 8 | 8 | $W_{FINAL}$ |
| 40 | 11 | 11 | 7 | 11 | 16 | 16 | TOTAL NUMBER OF INPUT BITS |
| $0.76\ 10^{11}$ | 1474 | 1474 | 92 | 1638 | 64226 | 64226 | USED STATES |
| $0.34\ 10^{11}$ | 574 | 574 | 36 | 410 | 1310 | 1310 | UNUSED STATES |
| 100 | 6.25 | 6.25 | 12.5 | 6.25 | 25 | 25 | QUANTIZATION AMPLITUDE RANGE |
| 69.1 | 4.5 | 4.5 | 9 | 9 | 24.5 | 24.5 | AMPLITUDE RANGE OF EACH TERM |

Fig. 1 Source code chart for the truth table implementation. $T_2=(T_{21}^I+T_{22}^I)+T_{23}+T_{24}+(T_{25}^I+T_{26}^I)$
$T_{21}^I=T_{22}^I=(4+5C_2\bar{q}_1)/2$, $T_{23}=9\bar{q}_2$, $T_{24}=5B_2\bar{q}_1^2$, $T_{25}^I=T_{26}^I=49C_{12}/2$

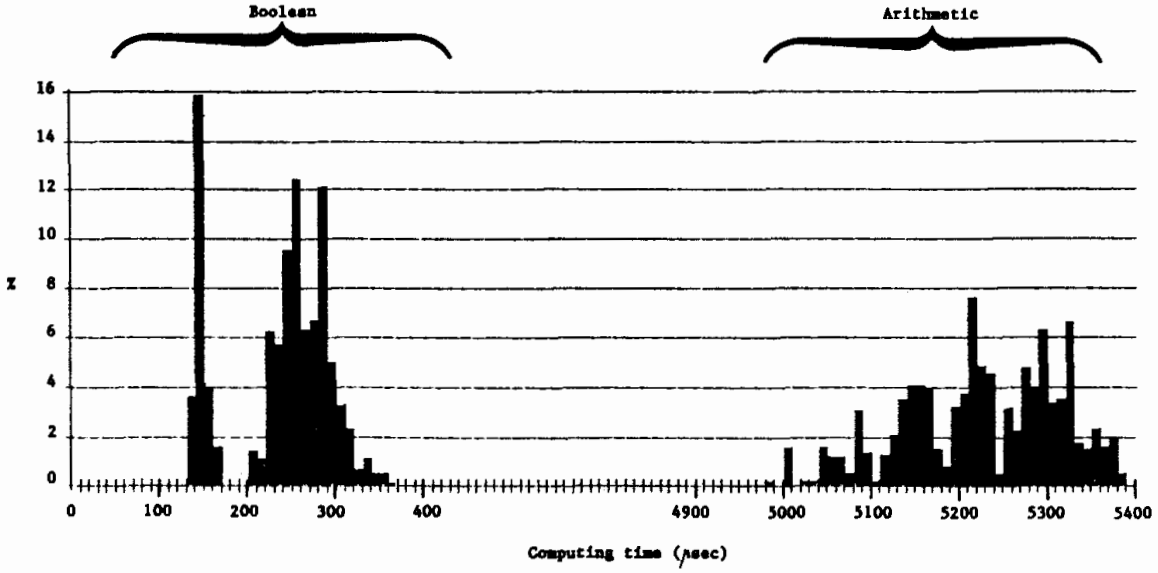


Fig. 2 Histogram of the computational time required by the term $T_{24}=P_1P_2B_2\bar{q}_1^2$.

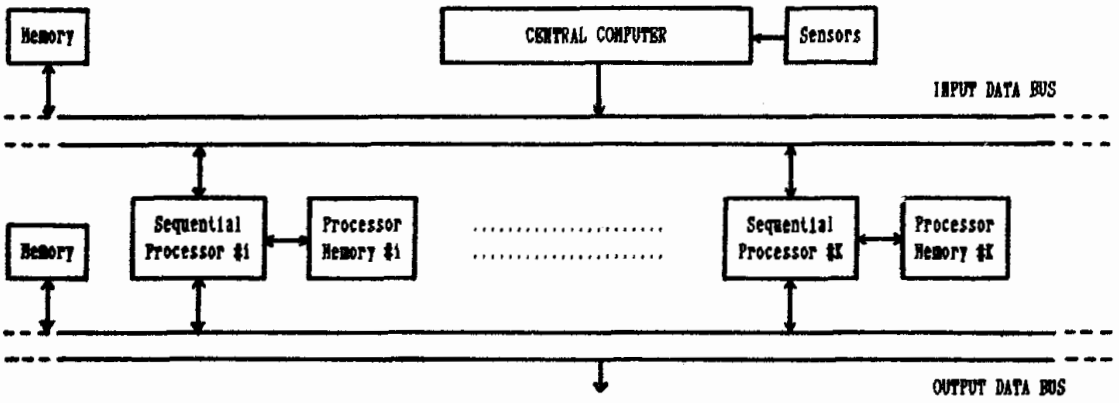


Fig. 3 A possible (decoupled) sequential/parallel computer structure, suggested by the new algorithm.