THE INSTITUTION OF ELECTRICAL ENGINEERS

International Conference

CONTROL 91

Conference Publication
Number 332

Volume Two

TOWARDS THE REAL-TIME CONTROL OF ROBOTIC SYSTEMS

J.A. Tenreiro Machado, J.L. Martins de Carvalho and Alexandra M.S.F. Galhano

University of Porto, Portugal

INTRODUCTION

In the last decades robot control has been an area of active research and development. Mechanical manipulators have a complex dynamics which makes difficult the design efficient controllers. Linear PID algorithms are still used in industrial robotics however, they are inappropriate for high performances. The limited efficiency of these systems motivated the development of controllers based on different concepts Dubowsky and DesForges (1), Luh et al (2), Freund (3). Nevertheless, the high computational load posed by many of the modern algorithms precludes its industrial application using present day microcomputers. In fact, the use of powerful monoprocessor computers is expensive while, on the other hand, multi-microprocessor architectures Luh and Lin (4), Binder and Herzog (5), Watanabe et al (6) are still in a research stage. This situation can be overcome through the development of control strategies more adapted to the microcomputer. In this line of thought, this article compares the computational capabilities of several computers and introduces techniques capable of rendering more efficient practical implementations. It is organized as follows. In section two we evaluate the computational load posed by different operations to several computer systems. Based on the results, in section three, we present general techniques amenable to robot control implementation. Finally, in section four conclusions are drawn.

EVALUATION OF THE CALCULATION LOAD OF DIFFERENT OPERATIONS AND THE CAPABILITIES OF SEVERAL COMPUTER SYSTEMS

The evaluation of the computational load required by an algorithm and the capabilities of a given computer, are essential steps in any preliminary study regarding its implementation. The difficulty of this study lies in the large number of factors involved, such as the type of microprocessor, clock frequency, memory wait states, type and version of the compiler, type and accuracy of the operations, etc. Considering the factors more relevant in robot control, the numerical results displayed in Tables 1 and 2 clarify this issue Machado et al (7,8,9). This data corresponds to the range of variation of the computational time required by each operation for a given system (software and hardware). Inspecting the results we conclude that:

- Trigonometric operations are the most time consuming while, logical and integer arithmetic operations are the fastest.
- The arithmetic coprocessor is essential to speed-up floating point operations (f.p.o.'s), but logical and integer arithmetic operations are not affected by its presence.
- For a given hardware, large variations of computing time may occur depending on the compiler being used.
- Reduced Instruction Set Computer (RISC) architectures appear to be much more efficient than conventional Complex Instruction Set Computers (CISC).

In order to provide a better perspective of the influence of these properties in robot control algorithms, we decided to measure the frequency of calculation for an adequate benchmark. Because extrapolation from generic benchmarks Price (10), INMOS (11) are questionable, we selected a benchmark capable of reflecting the requirements associated with modern robot controllers. In Fig. 1 we show the frequencies of computation of the inverse dynamics for the PUMA 560 manipulator and for the computer systems mentioned in Tables 1 and 2. The dynamic equations follow the simplifications presented by Armstrong et al (12); furthermore, several computational improvements introduced by the authors have reduced the number of operations to 5 sines, 6 cosines, 89 sums and 151 multiplications. In Fig. 1 the vertical coordinate (f) represents the "absolute" computing frequencies while the horizontal coordinate (f/f_{CLOCK}) corresponds to "normalized" frequencies. By other words, the horizontal coordinate is the ratio between the actual computing frequency (f) and the clock frequency of the processor (f_{CLOCK}) in megahertz. This graph reveals that computation speed-up through the acceleration of f_{CLOCK} - which is , essentially, technological dependent - is less significant than the improvement attained by the optimization of the processor architecture ("measured" by the horizontal coordinate). Moreover, these results demonstrate that, by the time the whole control system is implemented, which contains several sub-algorithms such as kinematics, dynamics, control and trajectory planning, the computational load can be incompatible with the requirement for of high sampling frequencies. The development of techniques amenable for an efficient implementation of the control algorithms is presented in the next section.

TECHNIQUES TO IMPROVE THE PERFORMANCES OF COMPUTER CONTROL SYSTEMS FOR MANIPULATORS

Modern robot control algorithms pose very stringent computational requirements. Technological progress leads to systems with increasing performances, but their use in robot control may not be economically feasible. Consequently, in the sequel we present several techniques that improve the performances of the control system and which are independent of the computer system. We group such techniques into six categories:

- Programming in assembly language.
- Computation using a low precision.
- Use of memory.
- Multirate sampling.
- Preview techniques.
- Dedicated compilers.

These techniques are the matter of the next sub-sections.

Programming in Assembly Language

This is a well known technique and constitutes a natural choice to improve the on-line performances of any controller. In fact, direct programming in assembly language allows a considerable optimization of the object code. However, modern control algorithms are very complex which makes the use of this technique a laborious task. This is the main reason why only a few researchers have adopted this strategy Spong et al (13), Murphy and Saridis (14) as an alternative to high level languages.

Computation Using a Low Precision

This method has been one of the standard alternatives to the programming in assembly. The most common technique consists in giving up f.p.o.'s in favour of fixed point arithmetic An et al (15), Suehiro and Takase (16). At a first sight it would seem more practical to reduce only the accuracy of the f.p.o.'s. However, this may be deceptive because many of the compilers implement low accuracy f.p.o.'s on the basis of operations with standard accuracy. Therefore, and contrary to our expectations, there is no improvement of the computation time. Clearly, if the compiler has a distinct library for each different accuracy then the computation can be speeded up. This is shown in Fig. 2 for several f.p.o.'s running in the IMS T800-20 transputer, programmed in Occam 2.

Use of Memory

This method transfers, a part or the total calculation of an algorithm to memory-based evaluations. A common practice consists on replacing the calculation of trigonometric functions by memory look-up tables. The use of memory can be generalized to a greater portion of the algorithm; nevertheless, the adoption of this technique has not received much attention with the exception of controllers based on learning strategies Albus (17), Kawamura et al (18), Miller III (19). Therefore, a large field of research remains open.

Multirate Sampling

The effects of the sampling frequency upon the performances of a controller are very important. In a control system, made of several feedback and feedforward paths, it is natural to expect different bandwidths along them. Therefore, it is reasonable to allocate different sampling rates to such paths (i.e. multirate sampling) Berg et al (20), Machado and de Carvalho (21), Kircanski et al (22). In this way the computing power is assigned to each path in accordance to its needs, allowing a more efficient management of the system resources.

Preview Techniques

The finite sampling frequency of a computer control system causes a time delay between the instants sensors are read and control actions are taken. Preview techniques Vinante et al (23), Yoshimoto and Sugiuchi (24) minimize this detrimental effect leading to better performances. Unfortunately, this technique has a limited range of application because the "preview" get worst the higher the computational load of the preview and control algorithms (i.e. the lower the sampling frequency).

Dedicated Compilers

The implementation of a control algorithm assumes, implicitly, the representation of process variables through real numbers. To these real numbers corresponds a computer internal representation by means of f.p.o.'s which require several bytes, in sharp contrast with the 8 to 16 bit accuracy of the external hardware (i.e. A/D and D/A converters, instrumentation, etc.). The authors developed a method for implementing control algorithms Machado et al (25) (26) that eliminates this discrepancy, by optimizing the chain:

Data Collection-> Processing-> Control Action

With this method we adopt an "uniform" representation for all the variables that is, we use a computational method where the variables have identical accuracy both in the computer and in the external hardware. Once the range of variation of each variable is known, we can define the quantization levels as a function of the accuracy of the A/D and D/A converters. Moreover, to each level we can assign an integer binary code. Then the control algorithm can be processed using Boolean operations upon the bits representing each variable (Fig. 3). On the other hand, the Boolean computation is optimized by means of Binary Decision Diagrams (BDD's), that are very efficient once converted to IF_THEN_ELSE structures as shown in Tables 1 and 2. This method is still in a research stage however, it has already suggested interesting extensions to RISC computational structures and parallel architectures. In the first case, the RISC processor structure is suggested by the fact that processing by means of BDD's only requires microprocessors with a (very) reduced number of instructions. In the second case, the advantage stems from the simplicity of task scheduling to parallel processors.

CONCLUSIONS

A large number of algorithms for robot control has been proposed so far however, its validation through practical implementations is confined to a few examples. Moreover, the high computational burden posed by many of the modern control algorithms precludes its industrial application using cost-effective microcomputers. This situation can be overcome through the development of control strategies more adapted to the microcomputer control structure. In fact, the analysis of both the computational requirements and microcomputer capabilities reveals that limitations can be alleviated through the adoption of several techniques associated with the data and code representation. In conclusion, these techniques allow not only a more efficient management of the computational resources, but also provide a better perspective on the developments towards future computer control architectures.

REFERENCES

1. Dubowsky, S., and DesForges, D.T., 1979, "The Application of Model-Referenced Adaptive Control to Robotic Manipulators," J. Dynamic Syst. Measurement, Contr., Trans. ASME, 101, 193-200.

2. Luh, J.Y.S., Walker, M.W., and Paul, R.P.C., 1980, "Resolved-Acceleration Control of Mechanical Manipulators," IEEE Trans. Automat. Contr., 25, 468-474.

3. Freund, E., 1982, "Fast Nonlinear Control with Arbitrary Pole-Placement for Industrial Robots and Manipulators," The Int. J. Robotics Research, 1, 65-78.

4. Luh, J.Y.S., and Lin, C.S., 1982, "Scheduling of Parallel Computation for a Computer-Controlled Mechanical Manipulator," IEEE Trans. Syst., Man, Cybern., 12, 214-234.

5. Binder, E.E., and Herzog, J.H., 1986, "Distributed Computer Architecture and Fast Parallel Algorithms in Real-Time Robot Control," IEEE Trans. Syst., Man, Cybern., 16, 543-549.

6. Watanabe, T., Kametani, M., Kawata, K., and Tetsuya, K., 1986, "Improvement in the Computing Time of Robot Manipulators Using a Multimicroprocessor," J. Dynamic Syst. Measurement, Contr., Trans. ASME, 108, 190-197.

7. Machado, J.A.T., and de Carvalho, J.L.M., "Microprocessor-Based Controllers for Robotic Manipulators," in "Microprocessors in Robotic and Manufacturing Systems" (to be published) Kluwer Academic Publishers (Spyro Tzafestas, editor).

8. Machado, J.A.T., de Carvalho, J.L.M., and Galhano, A.M.S.F., 1990, "Computer System Evaluation in Robot Control," Proc. IEEE Int. Workshop on Intelligent Motion Control, Istanbul, Turkey.

9. Machado, J.A.T., de Carvalho, J.L.M., and Galhano, A.M.S.F., 1990, "Microcomputer Evaluation in Robot Control," 33rd Midwest Symposium on Circuits and Systems, Calgary, Alberta, Canada.

10. Price, W.J., 1989, "A Benchmark Tutorial," IEEE Micro, 9, 28-43.

11. "Lies, damned lies and benchmarks" in "The Transputer Applications Notebook: Systems and Performance", INMOS, 258-279, 1989.

12. Armstrong, B., Khatib, O., and Burdick, J., 1986, "The Explicit Dynamic Model and Inertial Parameters of the PUMA 560 Arm," Proc. 1986 IEEE Int. Conf. on Robotics and Automation, San Francisco, CA.

13. Spong, M.W., Thorp, J.S., and Kleinwaks, J.M., 1987, "Robust Microprocessor Control of Robot Manipulators," Automatica, 23, 373-379.

14. Murphy, S.H., and Saridis, G.N., 1987, "Experimental Evaluation of Two Forms of Manipulator Adaptive Control," Proc. 26th IEEE Conf. on Decision and Control, Los Angeles, CA.

15. An, C.H., Atkeson, C.G., Griffiths, J.D., and Hollerbach, J.M., 1987, "Experimental Evaluation of Feedforward and Computed Torque Control," Proc. 1987 IEEE Int. Conf. on Robotics and Automation, Raleigh, NC.

16. Suehiro, T., and Takase, K., 1985, "A Manipulation System Based on Direct-Computational Task-Coordinate Servoing," Robotics Research: The Second International Symposium, MIT Press.

17. Albus, J.S., 1975, "A New Approach to Manipulator Control: The Cerebellar Model Articulation Controller (CMAC)," J. Dynamic Syst. Measurement, Contr., Trans. ASME, 97, 220-227.

18. Kawamura, S., Miyazaki, F., and Arimoto, S., 1988, "Realization of Robot Motion Based on a Learning Method," IEEE Trans. Syst., Man, CYbern., 18, 126-134.

19. Miller III, W.T., 1989, "Real-Time Application of Neural Networks for Sensor-Based Control of Robots with Vision," IEEE Trans. Syst., Man, Cybern., 19, 825-831.

20. Berg, M.C., Amit, N., and Powell, J.D., 1988, "Multirate Digital Control System Design," IEEE Trans. Automat. Contr., 33, 1139-1150.

21. Machado, J.A.T., and de Carvalho, J.L.M., 1989, "Engineering Design of a Multirate Nonlinear Controller for Robot Manipulators," J. Robotic Systems, 6, 1-17.

22. Kircanski, M., Vukobratovic, M., Kircanski, N., and Timcenko, T., 1988, "A New Program Package for the Generation of Efficient Manipulator Kinematic and Dynamic Equations in Symbolic Form," Robotica, 6, 311-318.

23. Vinante, C.D., Bermudez, C., and Tarre, F., 1986, "Predictive Compensation Compensation Implemented with a Microprocessor," IEEE Control Systems Magazine, 6, 40-43.

24. Yoshimoto, K., and Sugiuchi, H., 1985, "Trajectory Control of Robot Manipulator Based on the Preview Tracking Control Algorithm," Robotics Research: The Second International Symposium, MIT Press.

25. Machado, J.A.T., de Carvalho, J.L.M., Matos, J.A.S., and Costa, A.M.C., 1988, "An Efficient Computational Scheme for Robot Manipulators," Proc. IEEE Int. Symp. on Intelligent Control, Arlington, Virginia.

26. Machado, J.A.T., de Carvalho, J.L.M., Matos, J.A.S., and Costa, A.M.C., 1988, "Robot Manipulator Dynamics - Towards Better Computational Algorithms," Proc. IFAC Symp. on Robot Control, Karlsruhe, FRG.

TABLE 1 - Calculation time for different operations running on several computers.

Computer	IBM PS/2 8530-021	IBM PS/2 8560-061	IBM PS/2 8555-061	IBM PS/2 8570-A21	IBM PS/2 8530-021	IBM PS/2 8560-061	IBM PS/2 8555-061	IBM PS/2 8570-A21	IBM PS/2 486/25 P.
OS	DOS 3.3	DOS 3.3	DOS 3.3	DOS 3.3	DOS 3.3	DOS 3.3	DOS 3.3	DOS 3.3	DOS 3.3
Clock	8 MHz	10 MHz	16 MHz	25 MHz	8 MHz	10 MHz	16 MHz	25 MHz	25 MHz
Proc./ /Coproc.	8086	80286	80386SX	80386	8086/ /8087	80286/ /80287	80386SX/ /80387SX	80386/ /80387	80486
Language	TC V2.0	TC V2.0	TC V2.0	TC V2.0	TC V2.0	TC V2.0	TC V2.0	TC V2.0	TC V2.0
Minimum-Maximum Calculation Time for Different Operations (µsec)									
ADD fp	540-660	223-251	181-202	83-93	52-55	46-50	20-24	9-11	1.7-3.7
MULT fp	870-940	293-299	225-231	110-114	57-63	51-53	21-24	9-11	1.7-3.7
SIN fp	3500-4300	939-1175	697-884	335-418	300-360	236-269	54-66	27-39	11-22
ADD int	8.3	4.3	2.9	1.2	8.3	4.3	2.9	1.2	0.5
MULT int	22.5	5.2	3.2	1.5	22.5	5.2	3.2	1.5	1.1
IF	9.1	4.5	2.8	1.1	9.1	4.5	2.8	1.1	0.47

TABLE 2 - Calculation time for different operations running on several computers.

Computer	DECSTATION 3100	Apollo 3500	SUN 3-60	SUN 4-110	SUN 4-60 SPARCSt.1	AViiON AVX 300	IBM 6000 St.530	INMOS T800-20
OS	Ultrix 3.1	BSD4.2 IX	Unix 4.2	Unix 4.3.2	SUN 4.0.3	DG/UX 4.2	AIX V3.1	
Clock	16.7 MHz	25 MHz	20 MHz	14.3 MHz	20 MHz	16.7 MHz	25 MHz	20 MHz
Proc./ /Coproc	MIPS 2000/ /2010	68030/ /68882	68020/ /68881	SPARC	SPARC	88100	IBM Power Syst/6000	IMS T800-20
Language	System C	System C	System C	System C	System C	System C	System C	Occam 2
Minimum-Maximum Calculation Time for Different Operations (µsec)								
ADD fp	0.16-0.33	7.3-7.5	18-19	3-4	1.9-2.1	2.2-2.5	0.13-0.15	1.8-2.0
MULT fp	0.16-0.33	7.3-7.5	15-16	3-4	2.1-2.2	2.3-2.8	0.13-0.14	2.5-2.6
SIN fp	5.0-8.3	23-25	17-20	5-7	4.0-4.3	22-25	2.3-3.2	38.8-40.7
ADD int	0.024	0.4	1.2	0.9	0.63	2.7	0.013	0.66
MULT int	0.024	0.6	3.3	2.8	1.7	2.7	0.013	2.54
IF	0.055	1.3	1.6	1.2	6.2	0.67	0.034	0.922

TC: Turbo C, fp: floating point operation, int: integer operation
Precision of the Operations- Real numbers: 8 bytes, Integer numbers: 4 bytes

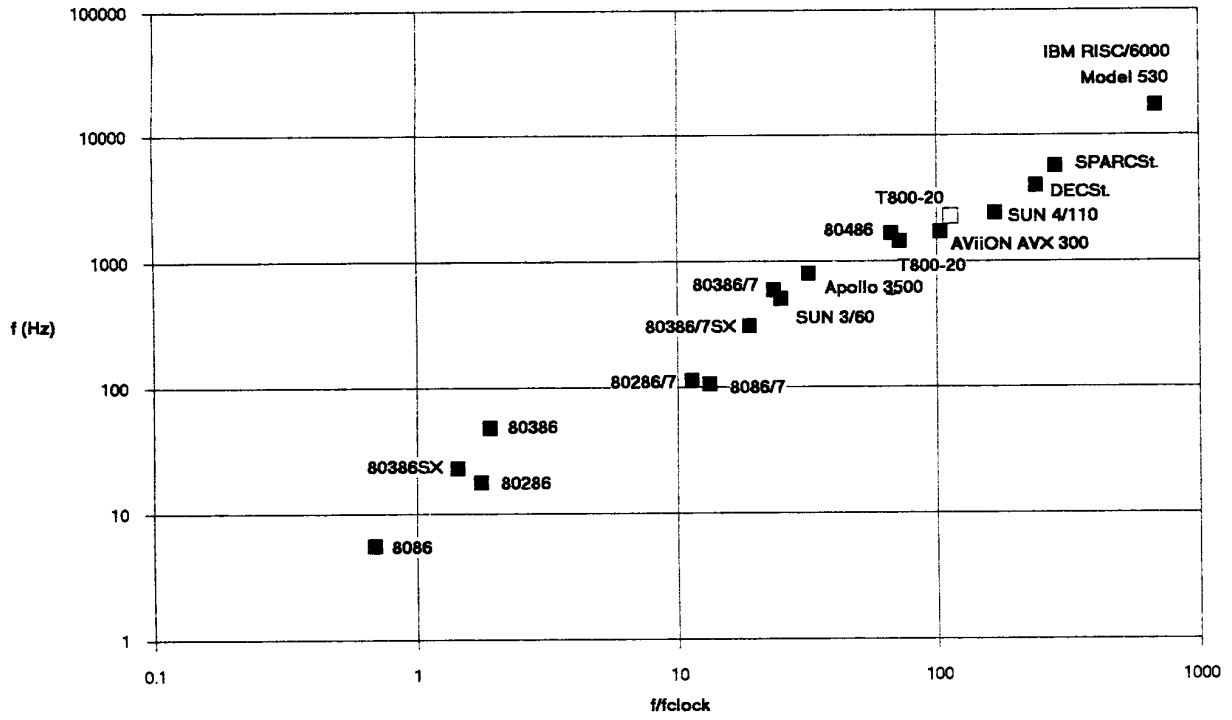

Figure 1 Chart of the calculation frequencies (f) for PUMA 560 dynamic equations versus the index f/f_{CLOCK} using the computers considered in Tables 1 and 2. Floating point operations are evaluated using a precision of: ■ 8 bytes, □ 4 bytes.

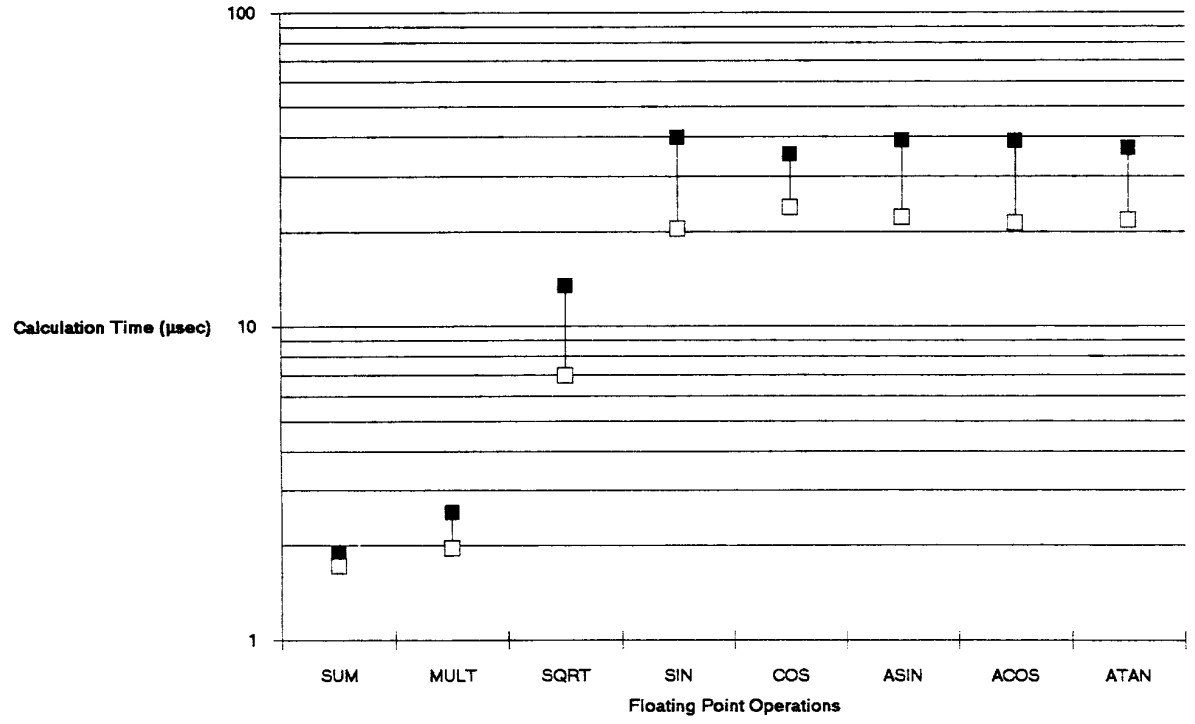

Figure 2 Chart of the calculation time for different floating point operations in transputer T800-20. The source code is programmed in Occam 2 and the real numbers are represented with a precision of: ■ 8 bytes, □ 4 bytes.

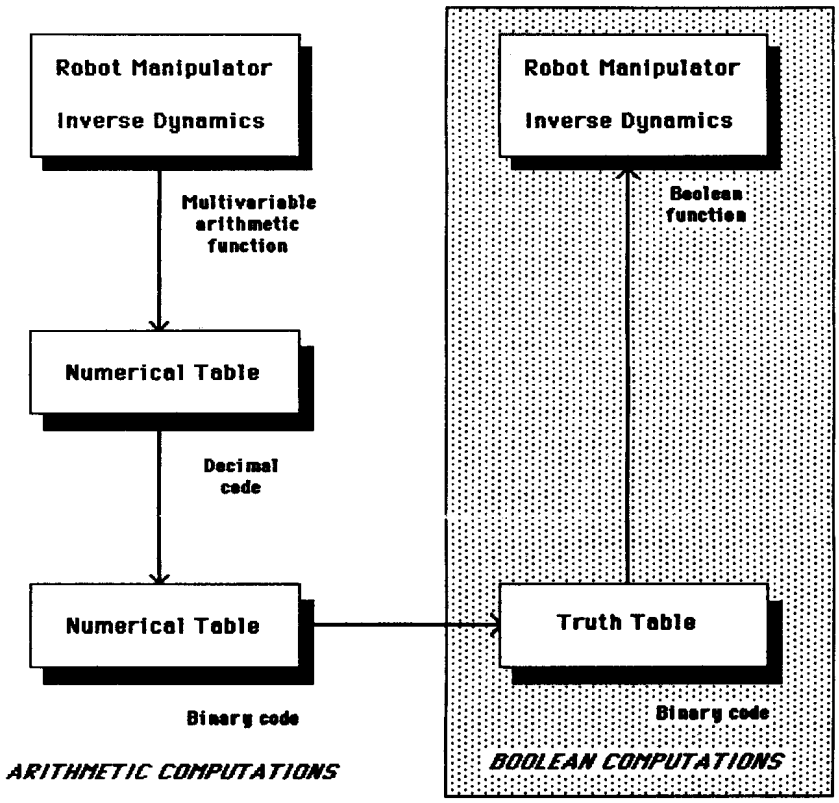

Figure 3 Scheme for the conversion of the robot inverse dynamics from the ordinary arithmetic representation to Boolean algebra.