Benchmarking Computer Systems for Robot Control

J. A. Tenreiro Machado, *Member, IEEE,* and Alexandra M. S. F. Galhano

Abstract— The high computational burden posed by modern control algorithms often precludes their industrial application using present day microcomputers. In this paper we evaluate the computational load of different logical and arithmetic operations and the capabilities of several computing systems (software and hardware). Real-time limitations are alleviated through the adoption of general techniques associated with the data representation. Such techniques achieve not only a more efficient management of the computational resources but also provide a deeper insight on developments toward future computer control architectures.

I. INTRODUCTION

ROBOT manipulators are mechanical systems that have several links connected through rotational or prismatic joints. These devices have complex kinematic and dynamic phenomena that require efficient controllers. Present day industrial robots use linear PID controllers; however, they are inappropriate for high performance applications because they lead to limited path tracking accuracy and often exhibit vibrations at high speeds. The low efficiency of these systems motivated the appearance of controllers based on different concepts [1]–[5]. However, the high computational burden posed by many of these algorithms precludes their industrial application using present day microcomputers. Moreover, powerful monoprocessor systems may be expensive while multi-microprocessor architectures [6]–[10] are still in a research stage. These limitations require the development of control strategies more adapted to microcomputer-based structures. In this line of thought, we may question the feasibility of the implementation of a given algorithm and which are the most adequate techniques to do the job. On the other hand, the technical literature is scarce on the evaluation of the computational load posed by each algorithm and its dependence on the software and hardware structure.

This paper compares the capabilities of several computer systems and introduces techniques that render more efficient practical implementations. Section II evaluates the computational load of different logical and arithmetic operations of several computer systems. Section III outlines general techniques amenable to practical implementations and Section IV presents the conclusions.

II. EVALUATION OF THE COMPUTER SYSTEMS

The evaluation of the computational load required by an algorithm and the capabilities of a given computational system, are essential steps in any preliminary study regarding its development. The literature suggests many control algorithms;
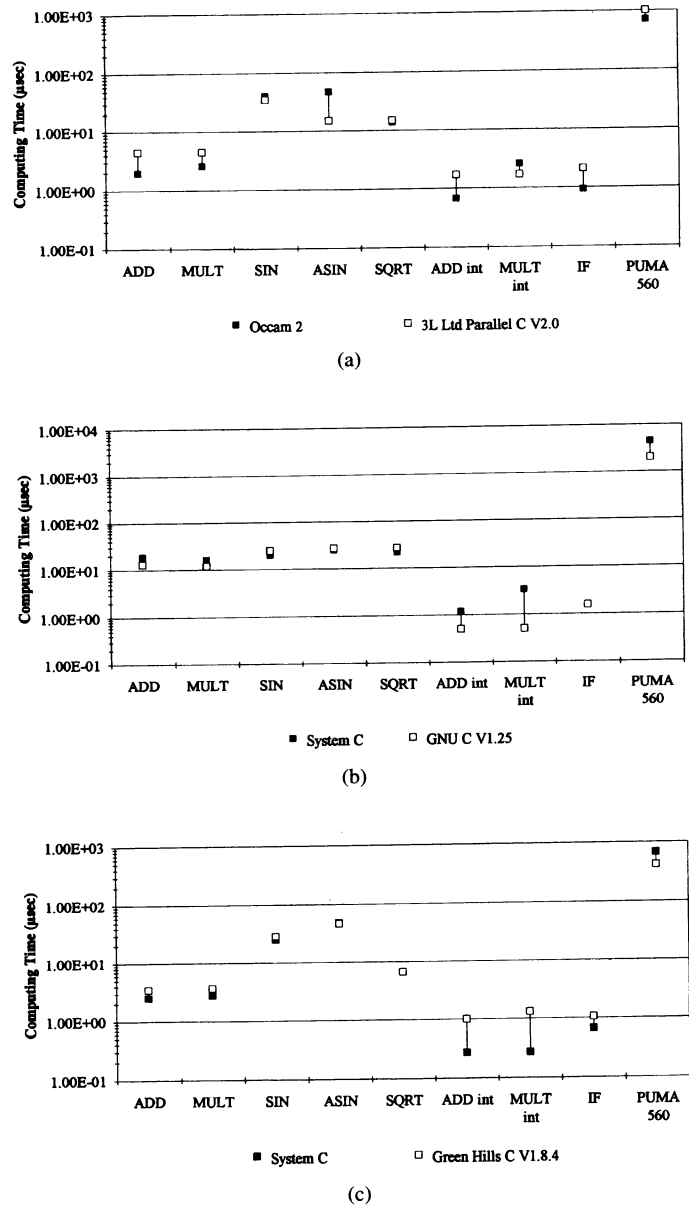
Fig. 1. Computing time τ of the PUMA 560 simplified dynamic equations using different compilers for (a) IMS T800–20; (b) SUN 3; (c) AViiON AVX 300.

however, in most cases, those proposing an algorithm do not analyze the computational requirements associated with its implementation. Some studies take into account the computational load based on the required number of additions and multiplications, while others only mention the maximum sampling frequencies achieved after installing the algorithm on a given system. Obviously such approaches are far from satisfactory, because they do not consider all the "factors"

TABLE I
COMPUTATION TIMES FOR SEVERAL SYSTEMS

Computer	IBM PS/2	IBM PS/2	IBM PS/2	IBM PS/2	IBM PS/2	IBM PS/2	IBM PS/2	IBM PS/2	IBM PS/2
OS	MsDOS	MsDOS	MsDOS	MsDOS	MsDOS	MsDOS	MsDOS	MsDOS	MsDOS
Clock (MHz)	8	10	16	25	8	10	16	25	25
Processor	8086	80286	80386 SX	80386	8086/8087	80286/80287	80386 SX/ /80387 SX	80386/80387	80486
Programming 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
Operation	Computation Time (µsec)								
Add fp	540 - 660	223 - 251	181 - 202	84 - 95	52 - 55	46 - 50	20 - 24	9 - 11	1.7 - 3.7
Mult fp	870 - 940	293 - 299	225 - 230	110 - 114	57 - 63	51 - 53	22 - 24	9 - 11	1.7 - 3.7
Sin	3500 - 4300	939 - 1175	697 - 884	335 - 418	300 - 360	236 269	54 - 66	27 - 39	11 - 22
Asin	6580 - 9560	1856 - 2802	1389 - 2158	664 - 1011	220 - 305	181 - 253	82 - 121	43 - 66	21 - 28
Sqrt	1220 - 1540	392 - 432	317 - 348	150 - 163	76 - 80	58 - 61	31 - 33	14 - 17	5.3 - 7.3
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

Computer	Apollo DN 3500	SUN 3	SUN 4	SUN 4 SparcSt. 1	AViiON AVX 300	Next Cube	DecStation 3100	IBM 6000 St. 530	Transputer T800-20
OS	BSD 4.2 Do/IX	UNIX 4.2	UNIX 4.3.2	SUN 4.0.3	DG/UX 4.2	Nextstep 2.1 Mach 2.5	Ultrix 3.1	AIX V3.1	
Clock (MHz)	25	20	14.3	20	16.7	25	16.7	25	20
Processor	68030/68882	68020/68881	SPARC	SPARC	88100	68040	MIPS 2000/2010	IBM Power System 6000	
Programming Language	System C	System C	System C	System C	System C	System C	System C	System C	Occam 2
Operation	Computation Time (µsec)								
Add fp	7.3 - 7.5	18 - 19	3.0 - 4.0	1.9 - 2.1	2.2 - 2.5	0.3 - 0.6	0.16 - 0.33	0.13 - 0.15	1.8 - 2.0
Mult fp	7.3 - 7.5	15 - 16	3.0 - 4.0	2.1 - 2.2	2.3 - 2.8	0.3 - 0.6	0.16 - 0.33	0.13 - 0.14	2.5 - 2.6
Sin	23 - 25	17 - 20	5.0 - 7.0	4.0 - 4.3	22 - 25	6 - 7	5.0 - 8.3	2.3 - 3.2	39 - 41
Asin	143 - 145	22 - 25	27 - 32	14.7 - 15.5	41 - 48	6 - 8	8.3 - 10.0	1.8 - 1.8	30 - 47
Sqrt	26.7 - 28.3	15 - 22	3.0 - 7.0	3.3 4.0	3.3 - 6.7	3 - 4	1.6 - 1.7	3.8 - 3.9	13 - 14
Add int	0.42	1.2	0.87	0.63	0.27	0.08	0.024	0.013	0.66
Mult int	0.58	3.3	2.8	1.7	0.27	0.08	0.024	0.013	2.54
If	1.3	1.6	1.2	6.2	0.67	0.36	0.056	0.034	0.92

TC - Borland Turbo C V2.0, fp: floating point, int: integer, Precision of the calculations- fp: 8 bytes, int: 4 bytes

involved. The difficulty of the problem lies precisely 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. Although not considering all possible combinations the data displayed in Table I attempts to clarify these issues [11], [33], [34]. This data corresponds to the range of variation of the computational time τ required by each arithmetic or logical operation for a given system (software and hardware). Among the large number of possibilities we have chosen those combinations more relevant in controller implementation. Inspecting the data we can draw several conclusions:

- Trigonometric operations are the most time consuming.
- Logical and integer (int) arithmetic operations are the fastest.
- The arithmetic coprocessor is essential to speed-up floating point (fp) operations.
- The presence or absence of coprocessors does not affect the computing time of logical or integer arithmetic operations.
- For a given hardware, large variations of computing time may occur depending on the compiler being used (Fig. 1).
- Theoretically, the speed of calculations increases linearly with the clock frequency. However, for large frequencies, wait states may occur when accessing memory [12].
- Reduced Instruction Set Computer (RISC) architectures appear to be far more efficient than conventional Complex Instruction Set Computers (CISC).

In order to provide a better perspective of these properties, we have decided to measure the frequency of calculation for an adequate benchmark. Because extrapolation from generic benchmarks is questionable [12]–[14], we have decided to select a benchmark capable of reflecting the requirements normally associated with robot control. In Fig. 2 we show the frequencies f of calculation of the inverse dynamics for the six DOF PUMA 560 (Table II) manipulator *versus* the index $\mu = f/f_{clock}$ for the computer systems mentioned in Table I. The inverse dynamics equations include not only the simplifications presented in [15] but also some improvements introduced by the authors, which reduced the number of required operations to 11 sines/cosines, 89 additions and 150 multiplications. In the chart of Fig. 2, f represents an "absolute" computation frequency while μ corresponds to a "normalized frequency" because it is independent of the clock frequency f_{clock}. The results reveal that the 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 computer architecture. Furthermore, the order of magnitude of the results demonstrates that the implementation of a controller containing algorithms such as the kinematics, the dynamics, and the trajectory planning can easily reach a computational load incompatible with the use of high sampling frequencies. The development of techniques for the real-time implementation of these algorithms is the matter of the next section.

III. TECHNIQUES TO IMPROVE THE PERFORMANCES OF COMPUTER CONTROL SYSTEMS

Modern robot control algorithms pose very stringent computational requirements. Although technological progress makes available systems with ever increasing performances, the truth
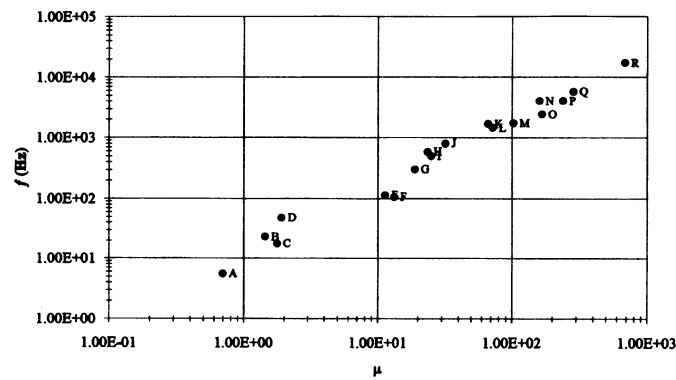
A	IBM PS/2	8086	8 MHz	MSDOS 3.3	TC V2.0
B	IBM PS/2	80386 SX	16 MHz	MSDOS 3.3	TC V2.0
C	IBM PS/2	80286	10 MHz	MSDOS 3.3	TC V2.0
D	IBM PS/2	80386	25 MHz	MSDOS 3.3	TC V2.0
E	IBM PS/2	80286/80287	10 MHz	MSDOS 3.3	TC V2.0
F	IBM PS/2	8086/8087	8 MHz	MSDOS 3.3	TC V2.0
G	IBM PS/2	80386 SX/80387 SX	16 MHz	MSDOS 3.3	TC V2.0
H	IBM PS/2	80386/80387	25 MHz	MSDOS 3.3	TC V2.0
I	SUN 3/60	68020/68881	20 MHz	UNIX 4.2	GNU C V1.25
J	Apollo DN3500	68030/68882	25 MHz	BSD 4.2 Do/IX	System C
K	IBM PS/2	80486	25 MHz	MSDOS 3.3	TC V2.0
L	T800-20	IMS T800-20	20 MHz		Occam 2
M	AViiON AVX 300	88100	16.7 MHz	DG/UX 4.2	System C
N	NEXT CUBE	68040	25 MHz	Nextstep 2.1 Mach 2.5	GNU C V1.36
O	SUN 4/110	SPARC	14.3 MHz	UNIX 4.3.2	System C
P	DECSTATION 3100	MIPS 2000/2010	16.7 MHz	Ultrix 3.1	System C
Q	SUN 4/60 SPARCSt.1	SPARC	20 MHz	SUN 4.0.3	System C
R	IBM 6000 st.530	IBM Power Syst/6000	25 MHz	AIX V3.1	System C

Fig. 2. Computing frequencies f for PUMA 560 simplified dynamic equations vrsus the index $\mu = f/f_{clock}$. Floating point operations are evaluated using the standard precision of 8 bytes (refer to legend above.)

is that their use as robot controllers may not be economically feasible. In the next four subsections we present a set of techniques to improve the real-time performances of the control system, which are, to a large extent "hardware independent."

A. Assembly Language Programming

This is a well-known technique and constitutes a natural starting point to improve the real-time performances of any controller. Direct programming in assembly language allows considerable optimization of the generated code. However, modern control algorithms are very complex and that makes their programming in assembly very time consuming. This is why only a few researchers have adopted this strategy [16]–[18] as an alternative to high level languages such as FORTRAN or C.

B. Low Precision Arithmetic Calculations

This option has been one of the principal alternatives to assembly language. The most common technique consists in giving up floating point calculations in favor of fixed point arithmetic [19]–[20]. At first sight it would seem more

practical to reduce only the accuracy of the fp calculations. However, this is somewhat deceptive because many of the high level languages implement low accuracy fp operations based on operations with standard accuracy. Therefore, and contrary to our expectations, there is no improvement of the computation times. Nevertheless, computation can be faster as long as the high level language has a distinct mathematical library for each accuracy. Fig. 3 shows the computational speed-up for several fp arithmetic operations on the IMS T800-20 transputer (source code: Occam 2). Using, again, the PUMA 560 dynamics as our benchmark, we measure $f = 1.4$ KHz ($\mu = 72$) and $f = 2.3$ KHz ($\mu = 114$) for 8 and 4 byte fp precision, respectively, that corresponds to a speed-up factor of 1.6.

In the same philosophy we can also mention the evaluation of transcendental functions through polynomial approximations. H. Henrichfreise [31] reports the approximation of the sine/cosine functions by a 9th/10th least square polynomials fit and the adoption of 32-bit (24-bit mantissa) fp format for robot control, giving a speed-up of the computing time from $\tau = 4.5$ μsec to $\tau = 1.7$ μsec. Using these techniques [32], the PUMA 560 dynamic equations were computed on a TMS320C30-33

TABLE II
SIMPLIFIED DYNAMIC EQUATION OF THE ROBOT PUMA 560

$S_2 = Sin(q_2)$
$C_2 = Cos(q_2)$
$S_3 = Sin(q_3)$
$C_3 = Cos(q_3)$
$S_4 = Sin(q_4)$
$C_4 = Cos(q_4)$
$S_5 = Sin(q_5)$
$C_5 = Cos(q_5)$
$q_{23} = q_2 + q_3$
$S_{23} = Sin(q_{23})$
$C_{23} = Cos(q_{23})$
$C_{223} = Cos(q_2 + q_{23})$

$\alpha_1 = S_{23} \cdot S_{23}$
$\alpha_2 = S_{23} \cdot C_{23}$
$\alpha_3 = C_2 \cdot S_{23}$
$\alpha_4 = C_2 \cdot C_{23}$
$\alpha_5 = S_4 \cdot S_5$
$\alpha_6 = C_4 \cdot S_5$
$\alpha_7 = S_4 \cdot C_5$
$\alpha_8 = C_4 \cdot C_5$
$\alpha_9 = C_{23} \cdot \alpha_8$

$\beta_1 = \dot{q}_1 \cdot \dot{q}_1$	$\beta_6 = \dot{q}_2 \cdot \dot{q}_2$	$\beta_{11} = \dot{q}_3 \cdot \dot{q}_4$
$\beta_2 = \dot{q}_1 \cdot \dot{q}_2$	$\beta_7 = \dot{q}_2 \cdot \dot{q}_3$	$\beta_{12} = \dot{q}_3 \cdot \dot{q}_5$
$\beta_3 = \dot{q}_1 \cdot \dot{q}_3$	$\beta_8 = \dot{q}_2 \cdot \dot{q}_4$	$\beta_{13} = \dot{q}_4 \cdot \dot{q}_4$
$\beta_4 = \dot{q}_1 \cdot \dot{q}_4$	$\beta_9 = \dot{q}_2 \cdot \dot{q}_5$	$\beta_{14} = \dot{q}_4 \cdot \dot{q}_5$
$\beta_5 = \dot{q}_1 \cdot \dot{q}_5$	$\beta_{10} = \dot{q}_3 \cdot \dot{q}_3$	$\beta_{15} = \dot{q}_5 \cdot \dot{q}_5$

$\chi_1 = \beta_7 + 0.5 \cdot \beta_{10}$
$\chi_2 = \beta_8 + \beta_{11}$
$\chi_3 = \beta_9 + \beta_{12}$

$\zeta_1 = -8.4 \cdot S_{23}$
$\zeta_2 = -0.134 \cdot C_{23}$
$\zeta_3 = -0.0025 \cdot S_5$
$\zeta_4 = [0.00164 + 0.0003 \cdot (1 - 2 \cdot S_4 \cdot S_4)] \cdot S_{23} - 0.0025 \cdot C_{23} \cdot \alpha_6$
$\zeta_5 = -(0.0025 \cdot C_5 + 0.000642) \cdot C_{23} \cdot S_4$
$\zeta_6 = 0.6 \cdot \alpha_2 - 0.0213 \cdot (1 - 2 \cdot \alpha_1)$
$\zeta_7 = 0.022 \cdot S_3 + 0.744 \cdot C_3$
$\zeta_8 = 0.267 \cdot S_{23} - 0.00758 \cdot C_{23}$
$\zeta_9 = \zeta_3 + 0.00248 \cdot (C_3 \cdot \alpha_8 - S_3 \cdot S_5)$

Inertials Coefficients

$D[1,1] = 2.57 + 1.38 \cdot C_2 \cdot C_2 + 0.3 \cdot \alpha_1 + 0.744 \cdot \alpha_3$
$D[1,2] = 0.69 \cdot S_2 + \zeta_2 + 0.0238 \cdot C_2$

MHz, running on a dSPACE DS1002 hardware system (source code: Texas Instruments C Vs. 4.40) at a rate of $f = 23$ KHz ($\mu = 705$).

C. Use of Memory

This method transfers, in part or entirely, the load of the on-line calculations to memory-based evaluations. A common procedure consists of replacing the fp calculations of trigonometric functions by memory tables [17], [20]. This method can be generalized to a greater portion of the algorithm; however, the application of this technique to control algorithms has not received much attention to date with the exception of controllers based on learning strategies [21]–[26]. Therefore, a large field of research remains open.

D. Multirate Schemes

The effects of the sampling frequency upon the performances of a digital controller are very important. In a robot system having feedback and feedforward paths, it is natural to expect different speeds of response along them. Therefore, it is reasonable to allocate different sampling (and computing) rates to such paths. The multirate schemes [27]–[29] assign the computing power to each loop in accordance to its needs leading to a more rational management of the

TABLE II (CONTINUED)
SIMPLIFIED DYNAMIC EQUATION OF THE ROBOT PUMA 560

$D[1,3] = \zeta_2 - 0.00397 \cdot S_{23}$
$D[2,2] = 6.79 + 0.744 \cdot S_3$
$D[2,3] = 0.333 + 0.372 \cdot S_3 - 0.011 \cdot C_3$
$D[3,3] = 1.16$
$D[3,4] = -0.00125 \cdot \alpha_5$
$D[3,5] = 0.00125 \cdot \alpha_8$
$D[4,4] = 0.2$
$D[5,5] = 0.18$
$D[6,6] = 0.19$

Coriolis/Centripetal Coefficients

$\delta_1 = -2.76 \cdot S_2 \cdot C_2 + 0.744 \cdot C_{223} + \zeta_6$
$\delta_2 = 0.5 \cdot \delta_1$
$\delta_3 = 0.744 \cdot \alpha_4 + 0.022 \cdot \alpha_3 + \zeta_6$
$\delta_4 = 0.5 \cdot \delta_3$
$\delta_5 = (-0.0025 \cdot \alpha_2 + 0.00086 \cdot \alpha_6 - 0.00248 \cdot \alpha_4) \cdot \alpha_3$
$\delta_6 = -0.0025 \cdot (\alpha_1 \cdot S_5 - \alpha_2 \cdot \alpha_8) - 0.00248 \cdot C_2 \cdot (S_{23} \cdot S_5 - \alpha_9) + 0.00086 \cdot \alpha_7$
$\delta_7 = 0.69 \cdot C_2 + 0.134 \cdot S_{23} - 0.0238 \cdot S_2$
$\delta_8 = \zeta_6 \cdot \chi_1$
$\delta_9 = \zeta_4 + 0.00248 \cdot S_2 \cdot \alpha_6$
$\delta_{10} = \zeta_5 + 0.00248 \cdot S_2 \cdot \alpha_7$
$\delta_{11} = 0.5 \cdot \zeta_7$
$\delta_{12} = \zeta_7 \cdot \chi_1$
$\delta_{13} = -0.00248 \cdot C_3 \cdot \alpha_5$
$\delta_{14} = \delta_{13} \cdot \chi_2$
$\delta_{15} = 0.5 \cdot (\delta_3 \cdot \beta_1 + \delta_{13} \cdot \beta_6)$
$\delta_{16} = \zeta_9 \cdot \chi_3$
$\delta_{17} = 0.5 \cdot (\delta_6 \cdot \beta_1 + \zeta_9 \cdot \beta_6)$
$\delta_{18} = \zeta_4$
$\delta_{19} = \zeta_5$
$\delta_{20} = -0.0025 \cdot \alpha_7$
$\delta_{21} = -0.00125 \cdot \alpha_6 \cdot (\beta_{13} + \beta_{15})$
$\delta_{22} = -0.000642 \cdot S_{23} \cdot C_4$
$\delta_{23} = \zeta_3 \cdot \chi_3$
$\delta_{24} = \zeta_3 \cdot \chi_1$
$\delta_{25} = 0.000642 \cdot S_4$
$\delta_{26} = \delta_{25} \cdot \chi_3$
$\delta_{27} = -\delta_{25} \cdot \chi_2$

Gravitational Forces/Torques

$G[2] = -37.2 \cdot C_2 + \zeta_1 + 1.02 \cdot S_2$
$G[3] = \zeta_1 + 0.25 \cdot C_{23}$
$G[4] = 0.028 \cdot S_{23} \cdot \alpha_5$
$G[5] = -0.028 \cdot (C_{23} \cdot S_5 + S_{23} \cdot \alpha_8)$

Joint Forces/Torques

$T[1] = D[1,1] \cdot \ddot{q}_1 + D[1,2] \cdot \ddot{q}_2 + D[1,3] \cdot \ddot{q}_3 + \delta_1 \cdot \beta_2 + \delta_3 \cdot \beta_3 + \delta_5 \cdot \beta_4 + \delta_6 \cdot \beta_5 + \delta_8 \cdot \beta_7 \cdot \beta_6$

$T[2] = D[1,2] \cdot \ddot{q}_1 + D[2,2] \cdot \ddot{q}_2 + D[2,3] \cdot \ddot{q}_3 + \delta_9 \cdot \beta_4 + \delta_{10} \cdot \beta_5 + \delta_{12} \cdot \delta_{14} + \delta_{16} - \delta_2 \cdot \beta_1 + G[2]$

$T[3] = D[1,3] \cdot \ddot{q}_1 + D[2,3] \cdot \ddot{q}_2 + D[3,3] \cdot \ddot{q}_3 + D[3,4] \cdot \ddot{q}_4 + D[3,5] \cdot \ddot{q}_5 + \delta_{18} \cdot \beta_4 + \delta_{19} \cdot \beta_5 + \delta_{23} + \delta_{20} \cdot \beta_{14} - \delta_4 \cdot \beta_1 - \delta_{11} \cdot \beta_6 + \delta_{21} + G[3]$

$T[4] = D[3,4] \cdot \ddot{q}_3 + D[4,4] \cdot \ddot{q}_4 - \delta_9 \cdot \beta_2 - \delta_{18} \cdot \beta_3 + \delta_{22} \cdot \beta_5 + \delta_{26} - \delta_{15} + G[4]$

$T[5] = D[3,5] \cdot \ddot{q}_3 + D[5,5] \cdot \ddot{q}_5 - \delta_{10} \cdot \beta_2 - \delta_{19} \cdot \beta_3 - \delta_{22} \cdot \beta_4 - \delta_{24} - \delta_{27} - \delta_{17} + G[5]$

$T[6] = D[6,6] \cdot \ddot{q}_6$

Computational load: 11 sines/cosines, 89 additions and 150 multiplications
Simplification criteria of the symbolic equations: 1% of the significative value

$S_i = Sin(q_i)$, $C_i = Cos(q_i)$, $S_{ij} = Sin(q_i + q_j)$, $C_{ij} = Cos(q_i + q_j)$

system resources. M. Kircanski *et al.* [30] propose the computation of the inverse dynamics using a 4 : 2 : 1 multirate method for the acceleration, velocity and position dependent terms, respectively. In this sense, we can get lower computing rates the lower the bandwidth of each signal, without having a significant loss of precision. For the PUMA 560 simplified dynamic equations we have the distribution of operations shown in Table III. Therefore, measuring the average computational load L_{av} by the operations (sin/cos, $\pm$, *) and weighting the real-time evaluation through the 4 : 2 : 1 multirate method we get:

$$L_{min} < L_{av} < L_{max} \tag{1a}$$

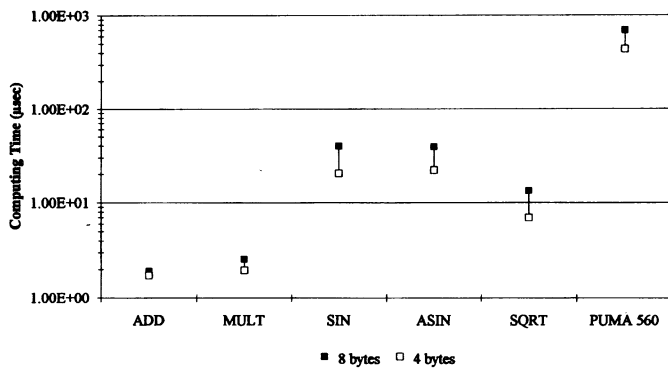$$\{0, 15, 16\} < \{2.75, 42.25, 62.25\} < \{11, 89, 150\} \tag{1b}$$

Fig. 3.   Computing time τ for several floating point operations in transputer T800–20 using 8 and 4 bytes precision (source code: Occam 2).

TABLE III
DISTRIBUTION OF OPERATIONS FOR THE 4:2:1 MULTIRATE METHOD

Operation	Acceleration Terms	Velocity Terms	Position Terms	Total Number of Terms
SIN/COS	0	0	11	11
ADD	15	35	39	89
MULT	16	51	83	150

where the lower $L_{\min}$ and upper bound $L_{\max}$ correspond to limit situations having the calculation of acceleration-dependent terms only and the total torques, respectively.

IV. CONCLUSIONS

A large number of controllers for robot manipulators have been proposed to date. However, the validation of these algorithms through practical implementations is still confined to a few examples. Moreover, many of these algorithms pose a high computational burden that precludes their industrial application using present day microcomputers. In order to overcome this situation, it is necessary to develop control strategies more adapted to microcomputer-based structures. The analysis of both the computational requirements and microcomputer capabilities reveals that some limitations are alleviated through the adoption of general techniques associated with data representation. Furthermore, the use of these techniques achieves not only a more efficient management of the computational resources but also provides a deeper insight on developments toward future control architectures.

REFERENCES

[1]   A. K. Bejczy, "Robot arm dynamics and control," *Jet Propulsion Lab. Tech. Memo 33-669*, 1974.
[2]   S. Dubowsky and D. T. DesForges, "The application of model-referenced adaptive control to robotic manipulators," *J. Dynamic Syst. Measure., Cont., Trans. ASME*, vol. 101, pp. 193–200, 1979.
[3]   J. Y. S. Luh, M. W. Walker, and R. P. C. Paul, "Resolved-acceleration control of mechanical manipulators," *IEEE Trans. Automat. Cont.*, vol. AC-25, pp. 468–474, 1980.
[4]   E. Freund, "Fast nonlinear control with arbitrary pole-placement for industrial robots and manipulators," *Int. J. Robot. Res.*, vol. 1, pp. 65–78, 1982.
[5]   R. Horowitz and M. Tomizuka, "An adaptive control scheme for mechanical manipulators—Compensation of nonlinearity and decoupling control," *J. Dynamic Syst. Measure., Cont., Trans. ASME*, vol. 108, pp. 127–135, 1986.
[6]   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, 1982.
[7]   R. Nigam and C. S. G. Lee "A multiprocessor-based controller for the control of mechanical manipulators," *IEEE J. Robot. and Automat.*, vol. RA-1, pp. 173–182, 1985.
[8]   E. E. Binder and J. 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, 1986.
[9]   T. Watanabe, M. Kametani, K. Kawata, and K. Tetsuya, "Improvement in the computing time of robot manipulators using a multimicroprocessor," *J. Dynamic Syst. Measure., Cont., Trans. ASME*, vol. 108, pp. 190–197, 1986.
[10]   C.-H. Liu and Y.-M. Chen "Multi-microprocessor-based Cartesian-space control techniques for a mechanical manipulator," *IEEE J. Robot. and Automat.*, vol. RA-2, pp. 110–115, 1986.
[11]   J. A. Tenreiro Machado and J. L. Martins de Carvalho, "Microprocessor-based controllers for robotic manipulators," in *Microprocessors in Robotic and Manufacturing Systems* , S. G. Tzafestas, Ed.   Norwell, MA: Kluwer Academic Publishers, 1991, chap. 5.
[12]   K. T. Lua, "Relative performance measurement of 80386, 80286, and 8088 personal computer systems," *Microprocessing and Microprogramming*, vol. 26, pp. 85–95, 1989.
[13]   W. J. Price, "A benchmark tutorial," *IEEE Micro*, vol. 9, pp. 28–43, 1989.
[14]   "Lies, damned lies, and benchmarks," in *The Transputer Applications Notebook: Systems and Performance* INMOS, 1989, pp. 258–279.
[15]   B. Armstrong, O. Khatib, and J. Burdick, "The explicit dynamic model and inertial parameters of the PUMA 560 arm," in *Proc. 1986 IEEE Int. Conf. on Robot. and Automat.* 1986.
[16]   E. Freund and H. J. Klein, "Practical results with nonlinear control strategy for computer-controlled industrial robots," in *Proc. 14th ISIR*, 1984.
[17]   M. W. Spong, J. S. Thorp, and J. M. Kleinwaks, "Robust microprocessor control of robot manipulators," *Automatica* , vol. 23, pp. 373–379, 1987.
[18]   S. H. Murphy and G. N. Saridis, "Experimental evaluation of two forms of manipulator adaptive control," in *Proc. 26th IEEE Conf. on Decision and Cont.*, 1987.
[19]   C. H. An, C. G. Atkeson, J. D. Griffiths, and J. M. Hollerbach, "Experimental evaluation of feedforward and computed torque control," in *Proc. 1987 IEEE Int. Conf. on Robot. and Automat.*, 1987.
[20]   T. Suehiro and K. Takase, "A manipulation system based on direct-computational task-coordinate servoing," in *Robotics Research: The Second International Symposium*Cambridge, MA: MIT Press, 1985.
[21]   J. S. Albus, "A new approach to manipulator control: The cerebellar model articulation controller (CMAC)," *J. Dynamic Syst. Measure., Cont., Trans. ASME*, vol. 97, pp. 220–227, 1975.
[22]   _____, "Data storage in the cerebellar model articulation controller (CMAC)," *J. Dynamic Syst. Measure., Cont., Trans. ASME*, vol. 97, pp. 228–233, 1975.
[23]   M. H. Raibert, "A model for sensorimotor control and learning," *Biological Cybern.*, vol. 29, pp. 29–36, 1978.
[24]   G. Hirzinger, "Robot systems completely based on sensory feedback," *IEEE Trans. Indust. Electron.*, vol. IE-33, pp. 105–109, 1986.
[25]   S. Kawamura, F. Miyazaki, and S. Arimoto, "Realization of robot motion based on a learning method," *IEEE Trans. Syst., Man, Cybern.*, vol. 18, pp. 126–134, 1988.
[26]   W. T. Miller III, "Real-time application of neural networks for sensor-based control of robots with vision," *IEEE Trans. Syst., Man, Cybern.*, vol. 19, pp. 825–831, 1989.
[27]   D. P. Glasson, "Development and applications of multirate digital control," *IEEE Cont. Syst. Mag.* , vol. 3, pp. 2–8, 1983.
[28]   M. C. Berg, N. Amit, and J. D. Powell, "Multirate digital control system design," *IEEE Trans. Automat. Cont.* , vol. 33, pp. 1139–1150, 1988.
[29]   J. A. Tenreiro Machado and J. L. Martins de Carvalho, "Engineering design of a multirate nonlinear controller for robot manipulators," *J. Robot. Syst.*, vol. 6, pp. 1–17, 1989.
[30]   M. Kircanski, M. Vukobratovic, N. Kircanski, and T. Timcenko, "A new program package for the generation of efficient manipulator kinematic and dynamic equations in symbolic form," *Robotica*, vol. 6, pp. 311–318, 1988.
[31]   H. Henrichfreise, "Tremendous performance for robotic applications," *dSPACE NEWS*, vol. 1, no. 1, pp. 6, Apr. 1992.
[32]   _____, Personal correspondence, June 1992.
[33]   J. A. Tenreiro Machado, J. L. Martins de Carvalho, and A. M. S. F. Galhano, "Computer system evaluation in robot control," *IEEE Int. Workshop on Intell. Motion Cont.*, Istanbul, Turkey, 1990.
[34]   _____, "Toward the real-time control of robotic systems," in *IEEE Cont. '91*, 1991.