IFAC

INTERNATIONAL FEDERATION OF AUTOMATIC CONTROL

PREPRINTS

SYMPOSIUM ON ROBOT CONTROL '88

KARLSRUHE
OCTOBER 5-7, 1988

VDI/VDE-GESELLSCHAFT
MESS- UND AUTOMATISIERUNGSTECHNIK (GMA)

ROBOT MANIPULATOR DYNAMICS - TOWARDS BETTER COMPUTATIONAL ALGORITHMS

J.A. Tenreiro Machado, J.L. Martins de Carvalho and J.A. Silva Matos

Faculdade de Engenharia da Universidade do Porto, Dep. Eng. Electrotecnica e de Computadores, 4099 Porto Codex, Portugal

Antonio M.C. Costa

INESC-Instituto de Engenharia e Sistemas de Computadores, Largo de Monpilher 22, 4000 Porto, Portugal

Abstract. A new robot manipulator inverse dynamics computational algorithm is announced. The novel feature resides in the computations which are a blend of ordinary and Boolean algebra. As such, this method may also be interpreted as a dedicated compiler that optimizes the on-line computing time at expenses of the off-line stage. Nevertheless, the high off-line requirements are alleviated, through the derivation of some general rules that stem from the structure of the robot manipulator equations. In a practical implementation, a computer program based on a Quine-McCluskey truth table simplification method was used and experimented on a 2R robot manipulator. The results show a considerable computational improvement on a conventional sequential machine. Furthermore, they clearly point out new computational parallel architectures, without scheduling problems, and where performance improvement is proportional to the number of processors. Finally, it is observed that the proposed algorithm is not restricted to robot inverse dynamic computations, but is also applicable to kinematic and control computations.

Keywords. Robots; computational methods; Boolean algebra; computer architecture; parallel processing; compilers.

INTRODUCTION

The so called manipulator inverse dynamics problem, i.e. the task of computing the required joint torques for a desired set of positions, velocities and accelerations, has been a major topic of research and development in the last two decades (Albus, 1975; Craig, 1986; Hirzinger, 1986; Machado and Carvalho, 1988; Miller, 1987; Raibert and Horn, 1978; Stone and Neuman, 1984; Whitney, 1972). In the period since then, effectiveness in the area of computational aspects of robot manipulator dynamic modelling has achieved high results, namely in the numerical recursive methods, i.e. Newton-Euler (Luh, Walker and Paul, 1980) and Lagrangian (Hollerbach, 1980) algorithms in the optimum stages, which were thereafter proved by Silver (1982) to be equivalent. More recently, Horak's (1984) development of a more efficient mixed algorithm may be considered as a significant step forward towards present robot manipulator inverse dynamic computing methods. More recently still, a further step towards computational improvement was attained with the the automatic generation of the dynamic symbolic formulae, either using a general computer language (Faessler, 1986) or with LISP-based computer algebra systems (Koplic and Leu, 1986; Leu and Hemati, 1986).

Computation through these schemes achieves better performances than the previous ones; moreover, it allows physical insight into the manipulator dynamics and provides strategies for the manipulator design, that, as pointed out by Yang and Tzeng (1986), can reduce the calculation burden. One may think of today's method as an "investment" in off-line computational time so that the on-line calculation time is abbreviated. The present tradeoff between off-line vs. on-line computing time can be pushed further if we bear in mind the facts pointed out in the sequel.

About the symbolic formulae computational implementation we know:

-The symbolic formulae are converted to the computer internal code through a high level language compiler like Fortran or C;

-The resulting object code imposes a complex burden, as far as a microprocessor is concerned, due to the high number of arithmetic and transcendental floating point calculations;

-The floating point calculations correspond to a large number of microprocessor's machine code instructions;

-The floating point calculations must be performed with high precision in order to reduce problems associated with finite precision arithmetic;

-Floating point calculations with high precision require a large word length.

Clearly, this type of implementation is far from satisfactory. But further reasons can be put forward:

-The computer internal numerical repre-

sentation has a much higher accuracy than the manipulator hardware (A/D, D/A, etc.), usually with only 8 to 16 bit precision;
-If computations could be performed with a precision of the same magnitude as the one used in the manipulator hardware, calculation time would decrease;
-The use of 8 to 16 bit computing accuracy without finite precision problems implies that the arithmetic operations, as well as the transcendental functions, can not be executed in the ordinary way.

An alternative method must be simple enough for any microprocessor to perform, i.e. it should be well adapted to the microprocessor's machine code instruction set.

To conclude, we observe that there is a need for 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. A possible candidate for the "special" algebra is the Boolean algebra, as it satisfies all the above requirements. Nevertheless, we are now faced with the problem of translating the ordinary arithmetic and transcendental formulae to Boolean algebra. Although formally independent, there is a way of doing it. The symbolic formulae are multivariable functions, and, as such, can be tabulated. If both input and output variables are quantified and converted to a suitable binary code, then the resulting table may be viewed 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 + acceleration› vector, and output Boolean functions the bits of the quantified binary coded required torques.

Unfortunately, upon further examination 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.

AN HYBRID COMPUTATIONAL ALGORITHM FOR ROBOT MANIPULATOR DYNAMICS

Since it has been found that the full Boolean-based computation is impractical, the necessity of a compromise between realizability and computational improvement is implied. If a compromise between the two (existing) extreme methods for torque computation, i.e.

a) entirely Boolean-based methods
b) entirely arithmetic-based methods

can be found, then it may benefit from the virtues of each (Machado, Costa and Carvalho, 1987)

Proposed Solution: Hybrid computation by ordinary arithmetic sums and Boolean algebra

Noting that the manipulator dynamic equations are of the form

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

where $J(q)$ is the n x n inertial matrix, $C(q,\dot{q})$ is the n dimentional Coriolis/centripetal vector, $G(q)$ is the n dimentional gravitational vector and q, $\dot{q}$ and $\ddot{q}$ are the n dimentional vectors of link positions, velocities and accelerations, respectively. A natural way of achieving our purpose 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. In this case, for each joint torque there are several truth tables, requiring 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 appropriated binary-coded representation of each input variable. The Boolean computation becomes alleviated as we have a maximum of (n+2) input word variables, contrasting with the requirement of 3n input word variables for the fully Boolean-based computation.

One of the more serious restrictions 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

$$m=\mathrm{int}[\log_2(p+1)] \qquad (2)$$

p=total number of terms

in each term in order to achieve the desired accuracy. Nevertheless, due to (2), m increases much more slowly than p. Moreover, as will be explained in section 3, the work involved with the truth table grows exponentially with the number of input bits when compared to the relatively "inexpensive" linear dependence with the output number of bits.

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 to several, much smaller, truth tables; furthermore, the on-line computing time does not alter significantly. In favour of this method we have smaller Boolean expressions, corresponding to simpler truth tables; against it, we have the need of m extra bits and the use of ordinary arithmetic.

THE NEW ALGORITHM IMPLEMENTATION ON A 2R ROBOT MANIPULATOR

To illustrate the implementation of our algorithm, we consider a 2R robot manipulator described by the following dynamic equations (Brady and others, 1982; Paul, 1981):

$$J(q)=\begin{bmatrix} (m_1+m_2)r_1^2+m_2r_2^2 & m_2r_2^2+r_1r_2m_2C_2 \\ \quad +2r_1r_2m_2C_2+J_1 & \\ m_2r_2^2+r_1r_2m_2C_2 & m_2r_2^2+J_2 \end{bmatrix}$$

$$C(q,\dot{q})=\begin{bmatrix} -r_1 r_2 m_2 S_2 \dot{q}_2^2 - 2 r_1 r_2 m_2 S_2 \dot{q}_1 \dot{q}_2 \\ m_2 r_1 r_2 S_2 \dot{q}_1^2 \end{bmatrix}$$

$$G(q)=\begin{bmatrix} g[m_1 r_1 C_1 + m_2 (r_1 C_1 + r_2 C_{12})] \\ g m_2 r_2 C_{12} \end{bmatrix} \qquad (3)$$

where

$$C_1=\cos(q_1),\ C_2=\cos(q_2) \qquad (4)$$
$$C_{12}=\cos(q_1+q_2),\ S_2=\sin(q_2)$$

In our study we use the same data as Young (1978) and Morgan and Ozguner (1985), namely:

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

and assume the amplitude of each link variable within the following ranges (i=1,2)

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

For these ranges and for the load (m_2, J_2) referred in (5) we have

$$-151.5\ Nm \le T_1 \le 152.5\ Nm \qquad (7a)$$
$$-69.1\ Nm \le T_2 \le 69.1\ Nm \qquad (7b)$$

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

$$-200\ Nm \le T_1 \le 200\ Nm \qquad (8a)$$
$$-100\ Nm \le T_2 \le 100\ Nm \qquad (8b)$$

From these considerations it results that equations (6) and (8) give the quantization ranges for the input and output word variables respectively.

Concerning the truth table simplification, a Quine-McCluskey method (McCluskey, 1956; Quine, 1952; Rhyne, 1973) including some heuristics, was implemented on a microVAX with a program source code written in VAX-11 Pascal and running under the DEC VAX-VMS 4.5 operating system. This multiple output minimization program, simplifies each truth table considering simultaneously all the outputs, so that there is an optimization of the global Boolean expressions, i.e. through the sharing of some Boolean subexpressions between several output functions, and has the following requirements:

input data memory space: $O(w*\log_2 3*3^w)$
output data memory space: $O(v*3^w)$ (10)
simplification computing time: $O(v*k^w)$

where $k \approx 6$, w is the number of the truth table input bits and v is the number of the truth table output bits. As previously referred to in section 2, the exponential nature of the Boolean simplification problem when relating to w, contrasting with its linear relationship with v, should be noted. Nevertheless, the implementation of some general rules may somewhat alleviate this problem. These rules stem from considerations like:

a) The quantization ranges (8) 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 (Machado, Costa and Carvalho, 1987), 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 the 2R robot manipulator inverse dynamic hybrid algorithm gives the results depicted in Fig. 1.

PARALLEL COMPUTATION

We have shown that the proposed hybrid computational algorithm for robot manipulator inverse dynamics, offers considerable performance improvement over purely arithmetic alternatives, even in a mono-processor sequential computing environment. In this section we will show that the algorithm also leads naturally to simple parallel architectures allowing unlimited speed-up in the on-line computations, without accuracy restrictions. In fact, the operations involved in the on-line computations required by our algorithm (AND, OR and arithmetic sums) are both comutative and associative, allowing 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 (Liu and Chen, 1986; Luh and Lin, 1982; Nigam and Lee, 1985; Wanatabe and others, 1986).

Parallel architectures like the ones shown in Fig. 2, are a natural consequence of the calculation decoupling allowed by our algorithm. Part a) shows a parallel structure in which a number (k) of sequential processors can be used. the result of the off-line computation, i.e. the object code, is stored in the processor memories in a way that assures optimum computational load distribution. Part b) shows an alternative parallel pipelined solution. 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 <u>not</u> proportional to the number of processors, and, is in fact, <u>restricted</u> to a maximum of n (a condition that is achieved only if some errors are allowed (Binder and Herzog, 1986)).

Finally, it should be noted that the

Boolean formulae 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 AND, OR and arithmetic sum calculations, 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. The design of a dedicated processor based on Binary Decision Diagrams (Matos and Oldfield, 1983) is being investigated.

SUMMARY AND 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. Finally, it is observed that the dedicated compiler philosophy is not restricted to robot manipulator inverse dynamic computations, but can be successfully generalized to other computing structures, like kinematics and control, 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

Albus, J.S. (1975). A new approach to manipulator control: the cerebellar model articulation controller (CMAC), ASME J. Dyn. Syst. M. C., 97, 220-227.

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.

Brady, M., Hollerbach, J.M., Johnson, T.L., Lozano-Perez, T. and Mason, M.T. (1982). Robot Motion: Planning and Control, MIT Press, Cambridge.

Craig, J.J. (1986). Introduction to robotics: mechanics & control, Addison-Wesley Publishing Company.

Faessler, H. (1986). Computer-assisted generation of dynamic equations for multibody systems, The Int. J. Robotics Research, 5, 129-141.

Hirzinger, G. (1986). Robot systems completely based on sensory feedback, IEEE Trans., Ind. Elect., 33, 105-109.

Hollerbach, J. M. (1980). A recursive Lagrangian formulation of manipulator dynamics and a comparative study of dynamics formulation complexity, IEEE Tra. Syst., Man, Cybern., 10, 730-736.

Horak, D.T. (1984), A simplified modeling and computational scheme for manipulator dynamics, ASME J. Dynamic Syst. Meas. Contr., 106, 350-353.

Koplic, J. and Leu, M.C. (1986). Computer generation of robot dynamic equations and the related issues, J. Robotic Systems, 3, 301-319.

Leu, M.C. and Hemati, N. (1986). Automated symbolic derivation of dynamic equations of motion for robotic manipulators, ASME J. Dyn. Syst. Meas. Contr., 108, 172-179.

Liu, C.-H and Chen, Y.-M (1986). Multi-microprocessor-based cartesian-space control techniques for a mechanical manipulator, IEEE J. Robotics and Automation, 2, 110-115.

Luh, J.Y.S., Walker, M.W. and Paul, R.P. (1980). On-line computational scheme for mechanical manipulators, ASME J. Dyn. Syst., Meas., Contr., 102, 69-76.

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.

Machado, J.A.T., Costa, A.M.C., Carvalho, J.L.M, (1987). Robot manipulator dynamics-towards better computational algorithms, INESC Internal Report.

Machado, J.A.T. and Carvalho, J.L.M, (1988). Robot manipulator systems: analysis and control, 3rd Int. Symposium on Systems Analysis and Simulation, Berlin, GDR.

Matos, J.S. and Oldfield, J.V. (1983). Binary decision diagrams: from abstract representations to physical implementations, 20th IEEE/ACM Design Automation Conference, Florida, USA.

McCluskey, E.J. (1956). Minimization of Boolean functions, Bell Syst. Tech. Jornal, 35, 1413-1444.

Miller, W.T. (1987). Sensor-based control of robotic manipulators using general learning algorithms, IEEE J.Robot., Automation, 3, 157-165.

Morgan, R.G. and Ozguner, U., (1985). A decentralized variable structure control algorithm for robotic manipulators, IEEE J. Robotics and Automation, 1, 57-65.

Nigam, R. and Lee, C.S.G. (1985). A multiprocessor-based controller for the control of mechanical manipulators, IEEE J. Robot. Aut., 1, 173-182.

Paul, R. P. (1981). Robot manipulators: mathematics, programming and control, MIT Press, Cambridge.

Quine, W.V. (1952). The problem of simplifying truth functions, Am. Math. Monthly, 59, 521-531.

Raibert, M.H., Horn, B.K.P. (1978). Manipulator control using the configuration space method, The Industrial Robot, 5, 69-73.

Rhyne, V.T. (1973). Fundamentals of digital design, Prentice-Hall, N.J.

Silver, W.M. (1982). On the equivalence of lagrangian and Newton-Euler dynamics for manipulators, The Int. J. Robotics Research, 1, 60-70.

Stone, H.W., Neuman, I.P. (1984). Dynamic modelling of a three-degrees-of-freedom robotic manipulator, IEEE Trans., Syst., Man, Cyber., 14, 643-654.

Yang, D.C. and Tzeng S.W. (1986). Simplification and linearization of manipulator dynamics by the design of inertia distribution, J. Robotics Research, 5, 120-128.

Young, K.-K (1978). Controller design for a manipulator using theory of variable structure systems, IEEE Trans. Syst., Man, Cybern., 8, 101-109.

Watanabe, T., Kametani, M., Kawata, K. and Tetsuya, K. (1986). Improvement in the computing time of robot manipulators using a multimicroprocessor, ASME J. Dyn. Syst. M. Contr., 108, 190-197.

Whitney, D.E. (1972). The mathematics of coordinated control of prosthetic arms and manipulators, ASME J. Dynamic Syst., Meas., Contr., 94, 303-305.

8	10 (n=2)						PRECISION (BITS)
T_1	T_{11}	T_{12}	T_{13}	T_{14}	T_{15}^*	T_{16}^*	TERM
200	50	12.5	6.25	12.5	100	50	QUANTIZATION RANGE
152.5	25.75	9	5	10	66.15	49	TERM AMPLITUDE RANGE
8	8	6	5	6	9	8	I.N.V. FINAL NUMBER OF BITS
48	15	11	9	18	8	16	TOTAL NUMBER OF INPUT BITS
$2.14\ 10^{14}$	16876	1475	410	209715	170	64226	USED STATES
$0.67\ 10^{14}$	15892	573	102	52429	86	1310	UNUSED STATES

a)

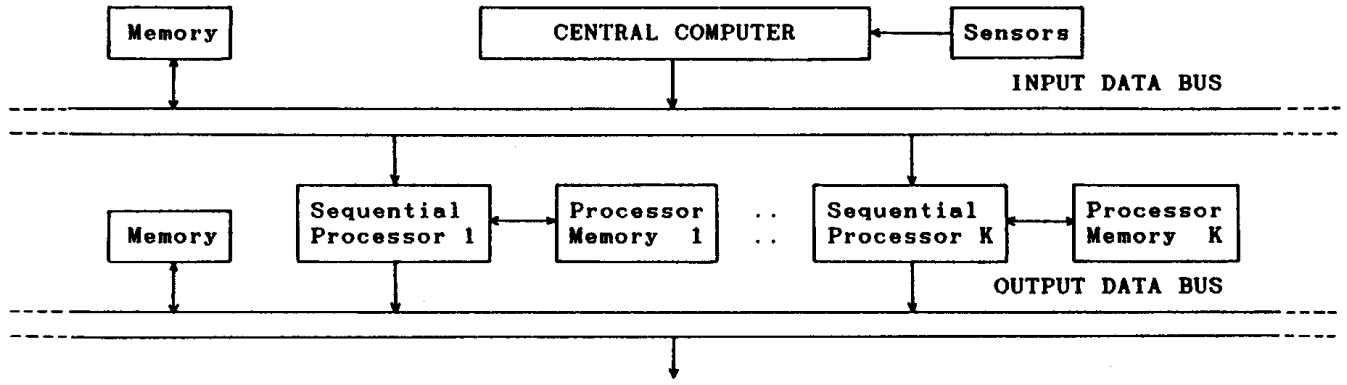
8	10 (n=2)						PRECISION (BITS)
T_2	T_{21}^*	T_{22}^*	T_{23}	T_{24}	T_{25}^*	T_{26}^*	TERM
100	6.25	6.25	12.5	6.25	25	25	QUANTIZATION RANGE
69.1	4.5	4.5	9	5	24.5	24.5	TERM AMPLITUDE RANGE
8	6	6	7	6	8	8	I.N.V. FINAL NUMBER OF BITS
40	11	11	7	11	16	16	TOTAL NUMBER OF INPUT BITS
$0.76\ 10^{11}$	1474	1474	92	1638	64226	64226	USED STATES
$0.34\ 10^{11}$	574	574	36	410	1310	1310	UNUSED STATES

b)

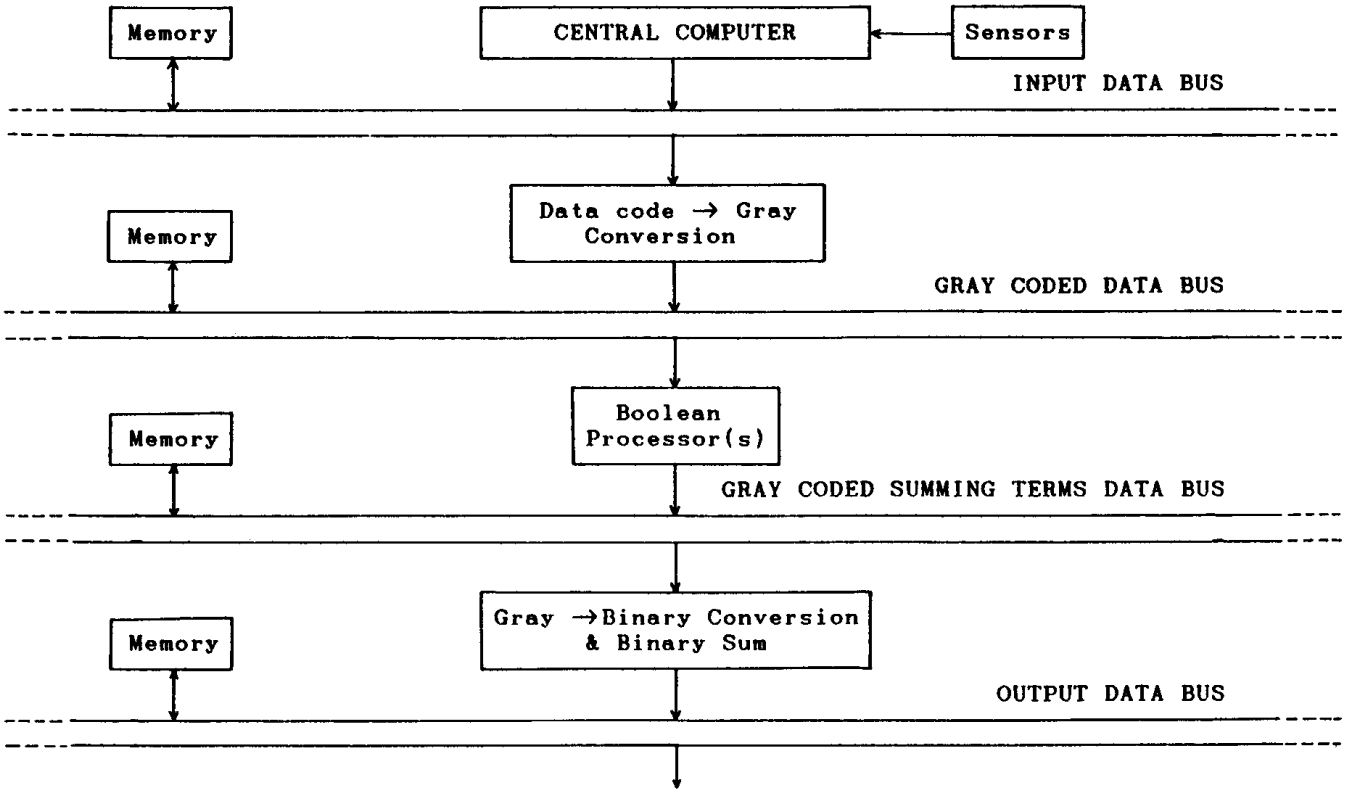
Fig. 1. 2R robot manipulator hybrid method source code chart, after using the relevant rules applicable for each summing term.

a) $T_1=T_{11}+T_{12}+T_{13}+T_{14}+(T_{15}^*+T_{16}^*)$
$T_{11}=(15.75+10C_2)\ddot{q}_1$, $T_{12}=(4+5C_2)\ddot{q}_2$, $T_{13}=-5S_2\dot{q}_2^2$
$T_{14}=-10S_2\dot{q}_1\dot{q}_2$, $T_{15}^*=66.15C_1$, $T_{16}^*=49C_{12}$

b) $T_2=(T_{21}^*+T_{22}^*)+T_{23}+T_{24}+(T_{25}^*+T_{26}^*)$
$T_{21}^*=T_{22}^*=0.5(4+5C_2)\ddot{q}_1$, $T_{23}=9\ddot{q}_2$
$T_{24}=5S_2\dot{q}_1^2$, $T_{25}^*=T_{26}^*=24.5C_{12}$



a)



b)

Fig. 2. Possible parallel computer structures, suggested by the new hybrid algorithm. The central computer sends information to the parallel processor so that, for each robot manipulator load mass a corresponding object code is selected; the code selection is made by each individual processor in its own code library memory.

a) Sequential/Parallel processor.

b) Pipeline/Parallel processor.