A ROBOT ARITHMETIC PROCESSOR CONCEPT FOR CARTESIAN CLOSED-LOOP CONTROL WITH PRESCRIBED DYNAMICS

E. Ersü, K. Rathgeber, M. Schnell and W. Neddermeyer

Institute of Control Engineering, Section Control Theory, Technical University Darmstadt, Darmstadt, FRG

Abstract. The hardware implementation of a cartesian closed-loop control scheme will be presented which allows to define the dynamic behaviour of each degree of freedom of the cartesian coordinate system in a prescribed sense. The control system at joint level is designed by multivariable design methods with an additional feedforward component using the concept of inverse dynamics.
To achieve high accuracy for cartesian motions quasi-continuous control mode with cartesian sampling periods of not greater than 5 ms is aimed at. A special purpose processor for calculation of kinematic and dynamic terms is designed and integrated into a multiprocessor architecture. This implementation concept with Robot Arithmetic Processor provides the necessary computational power and allows real-time cartesian closed-loop control which is also essential for cartesian sensory control tasks.

Keywords. Robots; cartesian robot control; special purpose processors; force control; closed-loop systems; real-time control; multivariable control systems; control system synthesis.

INTRODUCTION

Control system design methods implemented in conventional robot control systems are focused on joint control level, where the control design procedure incorporates conventional PID controllers to optimize the dynamic response in each joint. In case of fast motions along a desired time-based cartesian trajectory two significant problems arise:

i. Due to the highly nonlinear coupled dynamic relationship between the joints resulting trajectory shows deformations even if the dynamic performance of each single joint is acceptable or optimal.
ii. Even if the control scheme at joint level can cope with the above mentioned problem of robots cross-coupled nonlinear dynamics the described dynamic performance at joint level is strongly modified at the cartesian level due to the nonlinear coordinate transformation between the joint coordinate and cartesian coordinate system.

The purpose of robot control should be to maintain the prescribed motion along the desired cartesian trajectory or to have a well defined response in each direction of task coordinate - as it is required for sensory loops-. Thus a dynamic performance has to be prescribed with respect to the task coordinate system, i.e. the cartesian coordinate system. The goal has to be to design a control system which achieves the same dynamic performance in a prescribed sense for the whole cartesian work space, so that the dynamic response for the end-effector is not influenced by the actual joint configuration. This goal requires a re-definition of the robot control design task and the control design at joint level plays a subordinate role.

Very few control algorithms have been proposed which focus on this goal /1-6/. The major part of them utilizes adaptive schemes which are confronted with problems of changing dynamics due to fast cartesian motions. One of the widely used design method is the resolved motion acceleration control /1/ which is also conceptually adopted by some of the adaptive algorithms /4/.

This paper discusses a nonadaptive approach. At cartesian level control system design strategies of state-space are used. The proposed decoupling scheme utilizes at joint level the inverse dynamics concept and allows arbitrary pole-placement at cartesian level in each cartesian coordinate, similar to the conventional joint servo techniques. Furthermore a task-dependent on-line modi-fiable pole-placement strategy is realized which is essential for various sensory feedback control tasks /7/.

At the joint level the ideas of inverse dynamic concept is adopted to a robust closed-loop velocity control concept with acceleration-feedforward-decoupling. This strategy overcomes the difficulties of pure non-linear feed-forward decoupling due to the modelling errors, parameter and load variations.

The main subject of this paper, however, is the hardware implementation of such a sophisticated robot control scheme. Here the problem is that present standard 16/32 bit microprocessors do not offer effective solutions to achieve real-time control.

For sampling periods of less than 5 msec in a quasi-continously sampled cartesian control loop the amount of computational effort requires special purpose processors and a multiprocessor architecture all designed specially for real-time control calculations. The special purpose processor we designed is called Robot Arithmetic Processor (RAP). The concept of RAP aims at very fast processing which utilizes all the symmetries and parallelism inherent in the kinematic and dynamic transformation and compensation equations.

BASIC CARTESIAN CLOSED-LOOP FORMULATION

One of the main problems of cartesian closed-loop robot control is that sensory information must be provided which defines the end-effector's position and orientation in 3-D space with respect to the base coordinate system. There are only a few experimental systems which are rather expensive and complex and do not offer cost-effective practical solutions for on-line applications/11/. An indirect method is the calculation of cartesian position from measurements in joint coordinates via kinematic coordinate transformations. In case of link deformations due to end-effector contact with environment or due to large pay-loads and ,also in case of flexible links, this type of indirect measurement does not reflect the real cartesian position. These errors can be compensated by mathematical modelling and calculation of the link deformations.

One of the most useful mathematical tools to handle kinematic structures is the homogeneous transformation matrices/8/. The effective position and orientation of the robot's endeffector can be described in terms of a homogeneous transformation indicating the coordinate frame attached to robot's last link:

$$T_n = \begin{pmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad (1)$$

where $\underline{p}^T = (p_x, p_y, p_z)$ specifies the position of the coordinate frame with respect to the reference frame , and

$\underline{n}^T = (n_x, n_y, n_z)$, $\underline{o}^T = (o_x, o_y, o_z)$

and $\underline{a}^T = (a_x, a_y, a_z)$

are the unit vectors in the reference coordinate frame and describe the orientation. For each frame a corresponding position vector can be calculated

$$\underline{X}^T = (p_x, p_y, p_z, \vartheta, \phi, \theta) \qquad (2)$$

giving the end-effectors position (p_x, p_y, p_z) and its angles of orientation $(\vartheta, \phi, \theta$, e.g. RPY-angles)/8/

For serial link robots eq.s (1) and (2) are utilized to formulate the kinematic coordinate transformation equations. Attaching a coordinate frame to each link a transformation matrix A_i for link i can be defined by the corresponding geometric link parameters /8/ including the joint coordinate q (rotation θ_i for revolute joints or translation d for prismatic joints). The position of the terminal link of a series of one degree-of-freedom joints is then given by the transformation T_n :

$$T_n = A_1 \cdot \ldots \cdot A_n \qquad (3)$$

where T_n is then a function of

$\underline{q}^T = (q_1, \ldots, q_n)$

vector of joint coordinates.

Calculating the corresponding cartesian position vector (eq. 2) to T_n one obtains the solution of the general kinematic problem:

$$\underline{X} = \underline{f}(\underline{q}) \qquad (4)$$

which can be used to evaluate robots cartesian position from measurements in joint coordinates.

The cartesian velocity can be computed from the joint velocities via the 6xn Jacobian matrix J. Based on A_i matrices the Jacobian can be calculated for any series of one-degree-of-freedom joints/9/. The cartesian velocity of the end-effector can be then obtained by the following equation:

$$\dot{\underline{X}} = J \cdot \dot{\underline{q}} \qquad (5)$$

where

$$\dot{\underline{X}} = \begin{pmatrix} \underline{v} \\ \underline{\omega} \end{pmatrix} \quad \text{with} \quad \begin{array}{l} \text{cartesian velocities in coordinate frame T} \\ \text{angular velocities about the frame axes} \end{array}$$

The cartesian accelaration can be obtained from (5) by simple differentiation:

$$\ddot{\underline{X}} = J \cdot \ddot{\underline{q}} + \dot{\underline{J}} \cdot \dot{\underline{q}} \qquad (6)$$

where $\dot{J}$ is the time derivative of the Jacobian which can be calculated analytically by the following equation:

$$\dot{J} = \frac{\partial J}{\partial \underline{q}} \cdot \dot{\underline{q}} = J' \cdot \dot{\underline{q}} \qquad (7)$$

For on-line calculations a numerical differentiation of J may be more suitable.

To close the cartesian control loop it is necessary to compute the cartesian position and velocity errors. The actual cartesian position $\underline{X}_a$ and velocity $\dot{\underline{X}}_a$ can be obtained by (3) or (4) and (5), respectively. The desired cartesian position $\underline{X}_d$ can be expressed according to (1) and (2) as T_{nd}; thus the cartesian position error

$$\underline{X}_e = \underline{X}_d - \underline{X}_a \qquad (8)$$

can be calculated as a differential rotation and translation by the corresponding homogeneous transformation matrices of the actual T_{na} and desired position T_{nd} /9/ in coordinate frame of T_n:

$$\Delta T_e = T_{na}^{-1} \cdot T_{nd} \qquad (9)$$

where

$$\Delta T_e = \begin{pmatrix} 1 & -\delta_z & \delta_y & x_e \\ \delta_z & 1 & -\delta_x & y_e \\ -\delta_y & \delta_x & 1 & z_e \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad (10)$$

with $\underline{X}_e^T = (x_e, y_e, z_e, \delta_x, \delta_y, \delta_z)$ (11)

Based on equation (5) for the actual cartesian velocity

$$\dot{\underline{X}}_a = J_a \cdot \dot{\underline{q}}_a \qquad (12)$$

the cartesian velocity error can be obtained from

$$\dot{\underline{X}}_e = \dot{\underline{X}}_d - \dot{\underline{X}}_a \qquad (13)$$

when $\dot{\underline{X}}_d$ is given in an appropriate form in the corresponding coordinate frame. If $\underline{X}_d(t)$ is given as $T_{nd}(t)$ then $\dot{\underline{X}}_d$ can be calculated similar to eq. (9) /2/ by

$$\Delta t \cdot \begin{pmatrix} 1 & -\omega_{zd} & \omega_{yd} & v_{xd} \\ \omega_{zd} & 1 & -\omega_{xd} & v_{yd} \\ -\omega_{yd} & \omega_{xd} & 1 & v_{zd} \\ 0 & 0 & 0 & 1 \end{pmatrix} = T_{nd}^{-1}(t) T_{nd}(t+\Delta t) \qquad (14)$$

Having determined the formulas for the cartesian position and velocity error, now, the control problem is to find a feedback law

$$\underline{m}_c = \underline{f}(\underline{X}_e, \dot{\underline{X}}_e) \qquad (15)$$

where

$\underline{m}_c^T = (m_1, \ldots, m_n)$ torque vector commanded to joint actuators

such that the closed-loop robot control system is stable and shows a prescribed dynamic behavior.

PRESCRIBED DYNAMICS

The developed cartesian closed-loop control design is based on state-space plant description. Defining the state vector for the cartesian position and velocity as

$$\underline{Z} = \begin{pmatrix} \underline{x} \\ \dot{\underline{x}} \end{pmatrix} \tag{16}$$

eq. (6) can be expressed in state-space representation as

$$\dot{\underline{Z}} = \begin{pmatrix} 0 & I \\ 0 & 0 \end{pmatrix} \cdot \underline{Z} + \begin{pmatrix} 0 \\ J \end{pmatrix} \cdot \ddot{\underline{q}} + \begin{pmatrix} 0 \\ \dot{J} \end{pmatrix} \cdot \dot{\underline{q}} \tag{17}$$

where I is the (6x6)-identity matrix.

The control problem is to design a feedback law which minimizies the performance index

$$I = \lim_{t \to \infty} \int_0^t \| \underline{Z}_d(t) - \underline{Z}_a(t) \| \cdot dt \tag{18}$$

where

$$\dot{\underline{Z}}_d = G_d \cdot \underline{Z}_d \tag{19}$$

represents the desired i.e. prescribed closed-loop dynamics. G_d is a (12x12) stable real matrix, i.e. all eigenvalues of G_d have negative real parts. A very convenient choice for G_d is

$$G_d = \begin{pmatrix} 0 & I \\ A & B \end{pmatrix} \tag{20}$$

where A and B are (6x6) real matrices.

By this choice the control law for (18) can be given as

$$\ddot{\underline{q}}_d = J^{-1} \cdot (A\underline{X}_e + B\dot{\underline{X}}_e) - J^{-1} \cdot \dot{J} \cdot \dot{\underline{q}} \tag{21}$$

In general it is a nontrivial problem to determine appropriate non-diagonal matrices for A and B which reflect a prescribed dynamic behaviour. In practice they will be chosen as positive diagonal matrices, thus a linear second-order time invariant dynamic behaviour will result for each cartesian degree of freedom. The advantages of this choice are the desired dynamic behaviour characteristics, such as overshoot, rise time or natural frequency and damping ratio respectively, can be specified for each degree of freedom independently by the correspondin matrix elements a_{ii} and b_{ii} .

On the other hand this choice allows an on-line tas dependent programMming of the prescribed dynamic behaviour by simply specifying a new set of a_{ii} b_{ii} where constraints on actuator dynamics can to be taken into consideration.

The control law (21) requires that the Jacobian matrix J has an inverse. If J is not square because the arm has redundant degrees of freedom a pseudo-inverse J^* can be found /12/. Otherwise linear programming techniques can be applied by rewriting eq. (21) as a set of linear algebraic equations

$$J \cdot \ddot{\underline{q}}_d = d\ddot{\underline{X}} \tag{22}$$

where

$$d\ddot{\underline{X}} = A\underline{X}_e + B\dot{\underline{X}}_e - \dot{J} \cdot \dot{\underline{q}}$$

to obtain a numerical "best approximate" solution /10/. As the whole control strategy is implemented in a quasi-continous mode the solution procedure

CONTROL AT JOINT LEVEL

Given the reference input by equation (21) the control system at joint level has two components:

i. For the feedforward component the computed torque technique based on inverse dynamic model is used. A modified Lagrange-Euler-formulation /13/ is implemented which greatly reduces the computation time. The resulting equations are well structured explicitly expressed matrix form, thus insignificant terms or matrix elements can be easily identified and eliminated during computations.

ii. One of the main disadvantages of the computed torque technique is that the resulting control loop is very sensitive to modelling inaccurancies, payload and parameter changes. To eliminate these effects an additional velocity feedback loop is integrated at joint level. For the velocityloop the reference input is generated by simple integration of the desired acceleration in eq. (21). Robust control design methods are used to specify the controller /14/.
A detailed discussion of the application of this design method to manipulators can be found in /15/. Due to the additional feedforward via inverse model the multivariable design problem can be simplified and constraints on actuator dynamics can be handled easily.

Figure 1 illustrates the overall control system. The whole design is simulated for a r3-arm of MANTEC, a six-link robot, on VAX 11/780. The simulations has shown good results. The implementation for r3-arm is now in work and will be discussed in following sections .

PRESCRIBED CARTESIAN DYNAMICS IN SENSORY LOOPS

The proposed concept for robot-contol with prescribed cartesian dynamics allows the integration of force/torque-sensory loops. Thereby the approach permits the simultaneous position and force control which is a prerequisite to application of robot manipulations to complex handling and assembly tasks.

The architecture for the control of the position and the velocity of the manipulator and the contact forces generated at the hand is presented in Figure 2. (For the control of cartesian and joint level refer to Figure 1.) The basic idea within this concept is the formulation of 6-dimensional stiffness E in a frame C located arbitrarily in cartesian space /16/.

$$\underline{F} = E \cdot \Delta \underline{X}_F \tag{23}$$

where $\Delta \underline{X}_F$ is the displacement from the commanded position of the hand.

The frame C is a task-orientated coordinate system within which trajectories position/force constraints are determined /17/.
The desired values of position ${}^c\underline{X}_d$, velocity ${}^c\dot{\underline{X}}_d$ and contact forces ${}^c\underline{F}_d$ are specified due to the frame C.
Therefore the actual values $\underline{X}_a$, $\dot{\underline{X}}_a$ and $\underline{F}_a$ must be transformed into the same coordinate system C before the compliance selection is applied and errors are found.

The compliance selection S is a 6x6 - matrix that specifies which cartesian degrees of freedom are under position control ($S_i = 0$), and which are under force control ($S_i = 1$):

$$S = \begin{pmatrix} s_1 & & & & & 0 \\ & s_2 & & & & \\ & & \cdot & & & \\ & & & \cdot & & \\ & & & & \cdot & \\ 0 & & & & & s_6 \end{pmatrix} \qquad (24)$$

The error signals in position and velocity are found by the following equations:

$$^{C}\underline{X}_{es} = {}^{C}\underline{\Delta X}_{Fs} + (I-S)\cdot({}^{C}\underline{X}_{d} - {}^{C}\underline{X}_{a}) \qquad (25)$$

$$^{C}\dot{\underline{X}}_{es} = (I-S)\cdot({}^{C}\dot{\underline{X}}_{d} - {}^{C}\dot{\underline{X}}_{a}) \qquad (26)$$

The commanded position displacement ${}^{c}\underline{\Delta X}_{Fs}$ is given by the following equation, when equation (23) is applied:

$$^{C}\underline{\Delta X}_{Fs} = E^{-1}\cdot S\cdot({}^{C}\underline{F}_{d} - {}^{C}\underline{F}_{a}) \qquad (27)$$

The desired cartesian accelaration $\ddot{\underline{X}}_d$ can be obtained , after a transformation of the error signals from the frame C to the base-coordinate system by:

$$\ddot{\underline{X}}_d = A\,\underline{X}_{es} + B\,\dot{\underline{X}}_{es} \qquad (28)$$

where

$\underline{X}_{es}$ = position error in base coordinates

$\dot{\underline{X}}_{es}$ = velocity error in base coordinates

A,B : see equation (20).

The compliance selection tune is a mechanism which allows the modification of the selection matrix S during the performance of a special task. The main reason for tuning this matrix is the possible lack of sensor-signals. In this case it is necessary to continue the processing with all degrees of freedom under position control. Other necessarities for tuning are discrepancies in the preplanned motion or the occurance of unexpected obstacles. For these reasons the compliance selection tuning allows the adaption of the coordinate system C to the variation of the processing task.

REAL-TIME IMPLEMENTATION PROBLEMS

The number of mathematic operations - in particular multiplications, additions and transcendental functions - could be used to estimate the necessary computational performance the hardware should have. The number of these operations for the calculation of the kinematic equations, the inverse dynamics and the cartesian controller of a robot with six degrees-of-freedom is tabulated in TABLE 1.
The computation time of the cartesian controller includes the calculation of $\underline{X}_a, \dot{\underline{X}}_a, \ddot{\underline{X}}_d, \ddot{\underline{q}}_d$ .The inverse dynamic implemented with an algorithm based on Lagrange-Euler formulation is optimized for minimum computation effort /13/. It is obvious, that the calculation of these terms requires the greatest computation time (s. TABLE 1). Therefore the calculations must be executed in two different cycles to get a short sampling period. Thus it is sufficient to refresh the terms of inverse model in a multiple k of the sampling period t_p.

TABLE 1 lists also the computation times for a six-link arm with a 8086 microprocessor with 8087 arithmetic coprocessor. Hence it follows the minimum clock cycle using one processor for the control and one for the calculation of the terms of the inverse dynamics:

$$t_p = 24{,}8 \text{ msec} \quad 25 \text{ msec}, \; k=2$$

The large sampling time t_p (25 msec) leads to an unsatisfactory control behaviour for high-speed cartesian motion. Furthermore the computation time rapidly increases with rising number of degrees of freedom .

To improve the performance it is necessary to develop a fast advanced floating-point processor which handles floating-point numbers with a single precision through put greater than 1 MEGAFLOP (8087, 5MHz, ca. 0,05 MEGAFLOP). TABLE 1 contains the sum of the calculation times of a bit-slice-processor with a computation performance for multiplications and additions which is about the factor 20 higher than the performance of the 8087-coprocessor. Herewith it is possible to reduce the sampling time t_p to 2,2 msec and $k\cdot t_p$ = 4,4msec Only one processor is required. The sampling times increases to t_p = 2,5 msec, $k\cdot t_p$ = 5 msec if the control architecture is enlarged by sensory loops.

ROBOT ARITHMETIC PROCESSOR (RAP)

Figure 3 is the block diagram of the high-speed floating-point processor. The main modules of the processor are two special floating-point chips, which handle full 32-bit and 64-bit floating-point formats and operations, confirming to the proposed IEEE standard 742, version 10.0. A fast mode of operation is included which removes the time penalty of underflow exception handling by substitution of zero for denormalized numbers. The array flow-through time for floating-point multiplication and addition is under 180ns for single precision operations.

Furthermore, a control unit is required to build an independent processor-subsystem. The microprogram control architecture is instruction-address-data based, three stages of pipeline are involved. A 16 bit ALU is used as a microprogrammed address generator because directly desired memory addresses from the microcode are inefficient for large program A register-file connects the floating-point units with the data-bus to support efficient computation of vector and matrix operations. The supervisory machine has direct access to the memory of RAP for communication and data transfer.

TABLE 2 lists the total computation time for memory-to-memory and full pipelined operations (clock time = 90 nsec). This corresponds to the maximum and minimum computation performance. The computation time in TABLE 1 for RAP refers to the worst case memory-to-memory operations. Therefore the total sampling time could be reduced if the microprogram uses the pipeline facilities and the parallel data paths.

MULTIPROCESSOR CONTROL ARCHITECTURE

The control system consists of 4 identical microcomputers based on iAPX-80186 processors with two bus interfaces. The global bus connects all microprocessors applying a dual-port memory concept for the inter-processor communication. The local private bus is used for I/O - interfaces, intelligent controllers, local memory expansion and interface to RAP. This increases the throughput of the system and allows parallel computations.

Figure 4 is the block diagram of the control system. The task of the cartesian motion controller with attached RAP and multisensory data processing unit has been described above. The third microcomputer connects the robot arm with the con-

trol system via power adapter and data acquisition subsystems and performs supervision. The task of interprocessor coordination and communication and trajectory planning are carried out by the forth computer including the interface to the supervisory machine and the human user. This microprocessor control system is expandable with additional processor units and I/O-moduls to increase the performance for complex and advanced applications.

SUMMARY

The hardware implementation of a cartesian closed-loop control scheme is presented which allows to define the dynamic behaviour of each degree of freedom of the cartesian coordinate system in a prescribed sense. The control system at joint level is designed by multivariable design methods with an additional feedforward component using the concept of inverse dynamics.

To achieve high accuracy for cartesian motions quasi-continuous control mode with cartesian sampling periods of not greater than 5 ms is aimed at. A special purpose processor for calculation of kinematic and dynamic terms is designed and integrated into a multiprocessor architecture. This implementation concept with Robot Arithmetic Processor provides the necessary computational power and allows real-time cartesian closed-loop control which is also essential for cartesian sensory control tasks.

References :

/1/ J.Y.S. Luh, M.W.Walker, and R.P. Paul (1980). "Resolved-Acceleration Control of Mechanical Manipulators," IEEE Transactions on Automatic Control, vol. AC-25, no. 3, pp. 468-474.

/2/ C.H. Wu and R.P. Paul (1982). "Resolved Motion Force Control of Robot Manipulators," IEEE Trans. on Systems, Man and Cybernetics vol. SMC-12, no. 3, pp. 266-275.

/3/ C.H. Chung and G.G. Leininger (1984), "Adaptive Self-Tuning Control of Manipulators in Task Coordinate System," 1st IEEE Conf. on Robotics, March 1984, Atlanta, GA, USA.

/4/ C.S.G. Lee, M.J. Chung and B.H. Lee (1984) "Adaptive Control for Robot Manipulators in Joint and Cartesian Coordinates"1st IEEE Conf. on Robotics, March 1984, Atlanta, GA, USA.

/5/ A. Balestina, G.De Maria and L.Sciavicco (1984) "Robust Control of Robotic Manipulators," 9th IFAC World Congress, Budapest, Hungary, July 2-6 1984.

/6/ N. Hogan (1984) "Impedance Control of Industrial Robots," Robotics & Computer Integrated Manufacturing, Vol. 1, no. 1.

/7/ K. Arbter, J. Heindl, G. Hirzinger, K. Landzettel and F. Lange (1984). "New Techniques for Teach-In Acceleration and Learning in Sensor-Controlled Robots," 9th IFAC World Congress, Budapest, Hungary, July 2-6 1984.

/8/ J. Denavit and R.S. Hartenberg (1955). "A Kinematic Notation for Lower Pairs Mechanisms Based on Matrices," ASME Journal of Applied Mechanics, June 1955.

/9/ R.P. Paul (1981). "Robot Manipulators," MIT Press, Cambridge, MA,USA.

/10/ E. Ersü and D. Nungesser (1984). "A Numerical Solution of the General Kinematic Problem," 1st IEEE Conf. on Robotics, March 1984, Atlanta, GA, USA.

/11/ J. Mac Farlane and M. Donath (1984). "Tracking robot motion in three-dimensional space," NATO, ASI-Series, vol.F11 Robotics and Artificial Intelligence, Springer 1984.

/12/ C.A. Klein and C.H. Huang (1983). " Review of Pseudoinverse Control for Use with Kinematically Redundant Manipulators," IEEE, Trans- SMC, vol. SMC-13, no. 3, March/April 1983.

/13/ S. Megahed and M. Renaud (1982). "Minimization of the Computation Time Necessary for the Dynamic Control of Robot Manipulators," Proc. 12th Int. Symp. on Industrial Robots, Paris, 1982.

/14/ I.Horowitz (1982). "Improved Design Technique for Uncertain MIMO-Feedback Systems," Int. J. of Control, Vol. 36, no. 6.

/15/ E. Ersü, W. Neddermeyer and K. Amborski (1985). "Control of a Class of Nonlinear Systems by Multivariable Robust Design Methods" "Control 85," Julv, 9-11, 1985, Cambridge, U.K.

/16/ J.K.Salisbury (19b`). "Active Stiffness Control of a Manipulator in Cartesian Coordinates, "Proc. 19th IEEE Conf. on Decision and Control.

/17/ M.T. Mason (1979) "Compliance and Force Control for Computer Controlled Manipulators," MIT Artificial Intelligence Laboratory Memo 515.

ACKNOWLEDGEMENTS

The authors wish to acknowledge the valuable comments and suggestions of Prof. Dr. H. Tolle on this work. In addition, the authors appreciate the assistance of Dipl.-Ing. H. Reichenwallner for preparation of the figures and editing the text.

TABLE 1

Computational effort and comparison of the calculation times (six degree-of-freedom robot)

Calculation of	Number of Multiplications	Number of Additions	Number of Transcendental functions	Computation time 8086/87 (5MHz)	Computation time RAP
kinematic equations	387	272	23	15,6 ms	0,9 ms
terms of the inverse dynamics	1312	1055	0	50 ms	2,4 ms
inverse dynamics (accumulation of the terms)	180	183	0	7,2 ms	0,36 ms
cartesian controller	48	60	0	2 ms	0,11 ms

TABLE 2

RAP : Times for execution

operation	computation time for memory-to-memory transfer	computation time for total pipelined operations
multiply	720 ns	450 ns
add/substract convert/compose	720 ns	450 ns
divide	2220 ns	1950 ns
transcendental functions	3330 ns	3150 ns

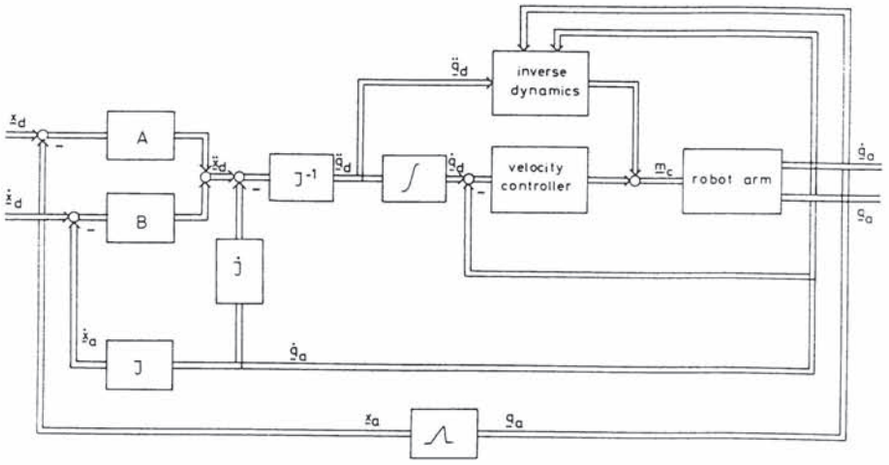

Figure 1. Block diagram of the cartesian closed-loop robot control system.

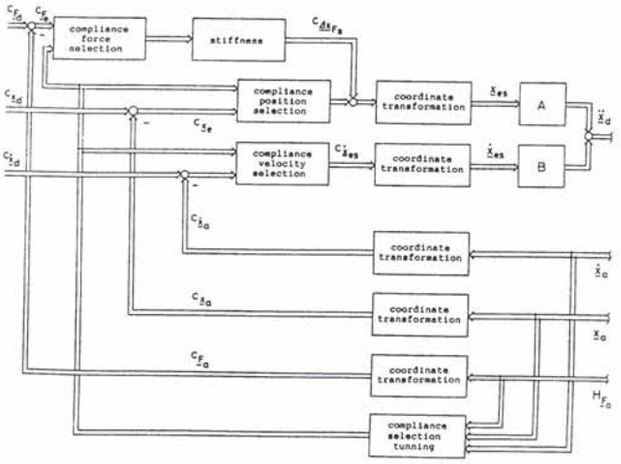

Figure 2. Block diagram of cartesian sensory contol loop.

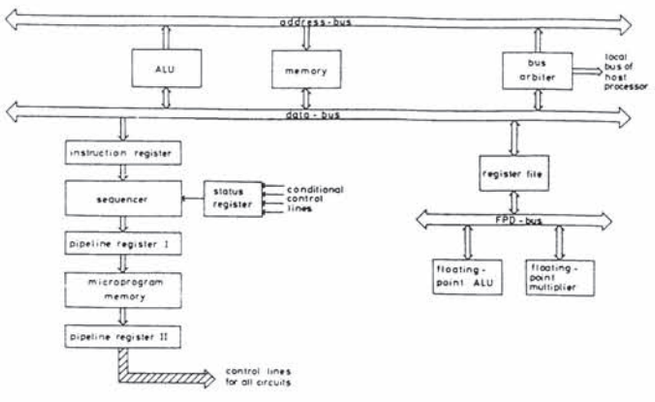

Figure 3. Principal structure of R A P.

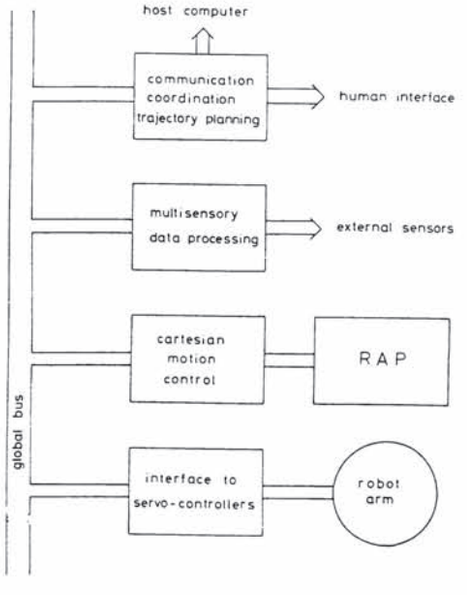

Figure 4. Multiprocessor control system architecture.