1986

A matrix manipulation method using phase shifted sinewaves for real-time robot control

T. Vu-Dinh
University of Wollongong

ADDENDUM

An additional comment on the solution method presented under Section 4.5

1. Summary

This addendum is a brief note which discusses the effects of round-off and other errors which arise in the kinematic solution method discussed in Section 4.5.

The absolute and percentage errors are tabulated and it is shown that, unlike conventional solution methods, the errors which inevitably arise in any computer based solution have a self compensating effect with the new method rather than accumulating as with conventional methods. It is also shown that the new method will generally produce a very accurate result for orientation of a robot manipulator despite any intrinsic round-off errors for individual joint angles.

2. Discussion of Errors for RM501 Manipulator

Table 4.5.1 shows a numerical comparison between the solutions satisfying a prescribed target location obtained from the sinewave simulation presented in Section 4.5, and those, numerically exact, calculated from conventional arithmetic expressions for the same target location.

The table shows a discrepancy of less than 0.4 degree between the two sets of solutions with the sinewave simulation result of the global positional error percentages reaching 0.59% maximum. However, the

TABLE 4.5.1

	Position Solution in Degrees Presented in the Order Required by the Solving Sequence			Orientation Solution in Degrees	
	θ_3	θ_2	θ_1	θ_4	θ_5
Sinewave Simulation Result. (See figures: 4.5.1, 4.5.2, page 106 ff.)	79	-22	16	33	-46
Arithmetic Calculaton Result. (Formulas from Table 4.4.2, Page 102.)	79.11021013	-22.38216408	15.9453959	33.2719539	-45.94612362
Solution Discrepancy in Degrees	$\Delta\theta_3=$ -0.11021013	$\Delta\theta_2=$ 0.38216408	$\Delta\theta_1=$ 0.0546041	$\Delta\theta_4=$ -0.27195395	$\Delta\theta_5=$ -0.05387638
Global Result from Computer Printout. (Page 259)	Target Position: $\begin{bmatrix} -80 \\ 280 \\ 300 \end{bmatrix}$	Position Achieved: $\begin{bmatrix} -80.24 \\ 279.85 \\ 301.77 \end{bmatrix}$	Error Percentage: 0.300% 0.054% 0.590%	Error Matrix $^{5}ROT_{5I} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	
	See Matrices Marked [0]T[5].			See [5]T[5].	

orientation part of the error matrix is remarkably close to the ideal complexion of a (3x3) unity matrix to within at least six significant digits in the actual computer printout, indicating that an orientational exact solution has been closely approached with the sinewave simulation method.

Two distinct features of the new solution procedure are discernible from the table.

First, the step size, one degree in the sinewave program example, determines the error limit. The sinewave simulation would yield solutions deviating from the exact ones by a half, one or two degrees maximum, if the step size had been chosen to be a half, one or two degrees, respectively. This effect is known in quantizing systems and is consistent with the fact that the smaller the step size (or the higher the quantizing resolution) is, the more accurate the solutions will be. However, there is a trade-off to be made between solution speed and solution accuracy, as smaller step sizes require more iterations to be performed in arriving at the solutions.

For the example presented in Section 4.5, it has turned out that the step size of one degree is a good choice. This integer step size minimized the programming iterations while at the same time, keeping the solution errors well within the one degree limit, resulting in negligible global position errors as Table 4.5.1 shows. Also in this table, the accuracy of the error matrix using integer solutions

resulting from the sinewave simulation is remarkable. Thus, a step size smaller (or a variable resolution higher) than one degree would have been unnecessary.

3. Discussion of Errors in General

Table 4.5.1 also indicates an immunity of the new solution method against progressive accumulation of numerical positional errors. In particular, the error caused by rounding the solution for θ_3 does not contribute to the position error which is caused by the integer rounding of θ_2 for the following reason. Indeed, by referring to the comparison Table 4.4.1, page 98, it is realized that in arriving at the solution for θ_2, the only criterion is to make $|{}^{O}\mathbf{TRL}_{5I}|^2$ as close to $|{}^{O}\mathbf{TRL}_{5D}|^2$ as possible. A solution for θ_2 is found, when $|{}^{O}\mathbf{TRL}_{5I}|^2=|{}^{O}\mathbf{TRL}_{5D}|^2$; and, in general, this equality can be achieved even if the solution for θ_3 is inaccurate, for example due to the use of integers for this variable. The same applies for the solution of the remaining positional variable θ_1. The particular errors of θ_1, θ_2 and θ_3 will contribute to the global position and orientation errors. However, these errors are independent from each other as Section 4 shows, which in turn has the effect of compensating rather than accumulating the errors of the position vectors. The main reason for this is that this method

From (2), we can find the solution θ_{3E} where subscript "E" indicates that the solution is exact. As an error e_3 due to integer rounding is introduced, a constrained solution θ_{2C}, indicated by subscript "C", is found from:

$$^{o}\mathbf{f}_{5I}(\theta_{3E}+e_3,\theta_{2C}) = const_o \qquad (1a)$$

Introducing an error e_2 to θ_{2C}, the constrained solution θ_{1C} is obtained from:

$$^{5}f_{5I}(\theta_{3E}+e_3,\theta_{2C}+e_2,\theta_{1C}) = const_5 \qquad (3a)$$

In (1a) and (3a), the integer rounding algorithm is such that e_3 and e_2 have been found so as to minimize $|^{1}\mathbf{f}_{5I}(\theta_{3E}+e_3)-const_1|$ and $|^{o}\mathbf{f}_{5I}(\theta_{3E}+e_3,\theta_{2C}+e_2)-const_o|$, respectively, where $\theta_{3E}+e_3$ and $\theta_{2C}+e_2$ are integers.

Very much like the conventional method of matrix comparisons, an error e_3 of θ_3 does affect the solution for θ_{2C} which deviates from the exact one θ_{2E} as much as to satisfy (1a). However, it is also seen that the basic constraint (1) can still be satisfied despite the error e_3. Thus, the deviation

of θ_{2C} from θ_{2E} has the effect of compensating rather than accumulating, as far as the magnitude square of the relevant vector ${}^{O}\mathbf{TRL}_{5I}$ is concerned. Similarly, θ_{1C} can be found to satisfy constraint (3) and to minimize error of ${}^{5}\mathbf{TRL}_{5I}$.

The artificial errors e_2 and e_3 caused by integer rounding are independent from each other as they stand in no functional relationship to each other. The only mechanism affecting them is the integer rounding algorithm. Therefore, their error contributions on ${}^{O}\mathbf{TRL}_{5I}$ are independent.

In particular, if e_2 and e_3 were not independent from each other, e_2 could be found from e_3 and an integer solution θ_{2I} would be found from $\theta_{2I}=\theta_{2C}+e_2$, regardless of the condition that $|{}^{O}\mathbf{f}_{5I}(\theta_{3E}+e_3,\theta_{2I})-\text{const}_O|$ be minimized. Hence, e_2 and e_3 must be independent from each other.

To find the relationship between the error e_3 and the actual deviation $d_2[e_3]$ of θ_{2C} from θ_{2E}, (1a) is rewritten as:

$${}^{O}\mathbf{f}_{5I}(\theta_{3E}+e_3,\theta_{2E}+d_2[e_3]) = \text{const}_O \qquad (1b)$$

Similarly, from (3a):

$$^{5}\mathbf{f}_{5I}(\theta_{3E}+e_3,\theta_{2E}+d_2[e_3]+e_2,\theta_{1E}+d_1[e_2,e_3]) = const_5 \quad (3b)$$

Where $d_1[e_2,e_3]$ represents the solution deviation of θ_{1C} from the exact one due to the errors e_2 and e_3. $d_2[e_3]$ and $d_1[e_2,e_3]$ do depend on each other and are systematic errors due to the constraints (1) and (3).

Thus:

$$\theta_{2C} = \theta_{2E} + d_2[e_3] \quad (4)$$

$$\theta_{1C} = \theta_{1E} + d_1[e_2,e_3] \quad (5)$$

Using the above equations, it can be shown that the following applies in the proximity of the exact solutions θ_{2E} and θ_{3E}.

$$d_2[e_3] = \frac{d\theta_2}{d\theta_3} e_3 \quad (6)$$

$$d_1[e_2,e_3] = \frac{d\theta_1}{d\theta_3} e_3 + \frac{d\theta_1}{d\theta_2} e_2 \quad (7)$$

(6) expresses the fact that the deviation of θ_{2C} from

θ_{2E} is zero only if e_3 is zero. However, (7) indicates that an exact solution for θ_1 is still possible despite the particular errors e_2 and e_3.

Adopting (7), the deviations from the ideal "one-values" of the particular components of the trace vector $(n_{x\_error}, o_{y\_error}, a_{z\_error})^T$ of the error matrix are given as:

$$d_{x\_error}(e_1,e_2,\ldots,e_5) = \sum_{i=1}^{5} \frac{\delta}{\delta\theta_i} n_{x\_error}\, e_i \qquad (8)$$

$$d_{y\_error}(e_1,e_2,\ldots,e_5) = \sum_{i=1}^{5} \frac{\delta}{\delta\theta_i} o_{y\_error}\, e_i \qquad (9)$$

$$d_{z\_error}(e_1,e_2,\ldots,e_5) = \sum_{i=1}^{5} \frac{\delta}{\delta\theta_i} a_{z\_error}\, e_i \qquad (10)$$

In the program presented under Section 4.5, since the quantity:

diff_trace_square

$$= (1-n_{x\_error})^2+(1-o_{y\_error})^2+(1-a_{z\_error})^2$$

$$= (d_{x\_error})^2+(d_{y\_error})^2+(d_{z\_error})^2$$

has been minimized to near zero by adjusting θ_4 and θ_5, those trace deviations must have approached zero ideally. This explains the perfectness of the error matrix despite the presence of the rounding errors e_1, e_2, ..., e_5.

A MATRIX MANIPULATION METHOD USING PHASE SHIFTED SINEWAVES FOR REAL-TIME ROBOT CONTROL

A thesis submitted in (partial) fulfillment of the requirements for the award of the degree of

Doctor of Philosophy

from

THE UNIVERSITY OF WOLLONGONG

by

T. Vu-Dinh (Dipl.-Ing., RWTH Aachen)

DEPARTMENT OF ELECTRICAL
AND COMPUTER ENGINEERING

(December 1986)

ACKNOWLEDGEMENT

A.) PERSONS

Dr. Christopher D. Cook (Supervisor)
Senior Lecturer, Department of Electrical and Computer Engineering, The University of Wollongong, N.S.W. 2500.
Managing Director of the Automation and Engineering Application Centre Ltd., Wollongong, N.S.W. 2500.

Prof. Dr. Brian H. Smith
Chairman, Department of Electrical and Computer Engineering, The University of Wollongong, N.S.W. 2500.

Mr. M. Comensoli, Mr. D. McLean, Mr. A. Mourad, Mr. S. Forst, Mr. G. Samways.

Staff Members
of the Department of Electrical and Computer Engineering, The University of Wollongong, N.S.W. 2500.

Staff Members
of the Automation and Engineering Application Centre Ltd., Wollongong, N.S.W. 2500.

B.) DEPARTMENTS/INSTITUTIONS

Commonwealth Scientific and Industrial Research Organisation (CSIRO, Australia), Division of Manufacturing, Melbourne, Victoria 3000; Funder of the CSIRO/Wollongong University's collaborative research project "Development of a General Purpose Low Cost Robot Controller" over the period from June 1983 to June 1986.

Wollongong University's Computer Centre.

Illawarra Technology Centre Ltd., Wollongong, N.S.W. 2500.

Department of Education and Youth Affairs, Sydney, N.S.W. 2000.

ABSTRACT

This work involves the theoretical development of a new real-time matrix manipulation method and the practical proof of the new theory. There is a need for such a method because it has not yet proved possible to provide non-recursive solutions to robot dynamic problems in real-time. This is because the matrix products and matrix derivatives required are, as is well known, too time consuming to numerically compute in real-time, due to the large number of additions and multiplications involved in the matrix manipulation.

It is shown that the partial orthonormality of the standard Denavit/Hartenberg matrices used in the robot control equations allows them to be expressed in sinewaves such that real-time matrix manipulations are performed simply by phase shifting the sinewaves rather than by numerically calculating the matrix products. The development is introduced by initially considering two-dimensional kinematic descriptions and this is then extended to three-dimensional kinematics and later to the finding of partial derivatives used in robot dynamics.

The way in which the sinewave method allows a general systematic and non-intuitive approach to robot inverse kinematic problems is also described.

The practical proof of the new system is performed by Pascal simulation and also by hardware implementation. The hardware implementation involves dedicated digital hardware whose basic structure has cascade characteristics and possesses an address/data/control bus somewhat like that of a digital computer. In this hardware, sinewaves representing the various relevant matrices are stored in individual random access memories such that they can be read in a controlled manner to simulate the information about the matrix elements to be manipulated.

The time it takes for the hardware, which is named the Sinusoidal Matrix Processor (SMP), to stabilize whenever the joint variables are being changed is a linear function of n compared with n^4 for conventional techniques, where n is the number of degrees of freedom of the manipulator. In this time, the SMP provides all relevant first and second order partial derivatives which are involved in the inverse dynamic calculations and so a substantial potential time saving is provided and in fact, the computing time becomes comparable to that of the recursive formulations involving the use of (4X4) matrices. Unlike recursive formulations, however, the states remain available explicitely so greatly facilitating the application of advanced control schemes.

CONTENTS

TABLE OF CONTENTS

LIST OF SYMBOLS USED

${}^{i-1}\mathbf{A}_i$: Elementary homogeneous joint matrix describing orientation and position of the i-th joint coordinate system with respect to the (i-1)th joint coordinate system.

$$ {}^{i-1}\mathbf{A}_i = \begin{bmatrix} \mathbf{n} & \mathbf{o} & \mathbf{a} & \mathbf{p} \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

where

$\mathbf{n} = [n_x, n_y, n_z]^T$ $\quad$ $\mathbf{o} = [o_x, o_y, o_z]^T$

$\mathbf{a} = [a_x, a_y, a_z]^T$ $\quad$ $\mathbf{p} = [p_x, p_y, p_z]^T$

n, **o**, **a** and **p** are the three-dimensional Cartesian normal, orientation, approach and position vector, respectively, embedded in the A-matrix.

The same naming conventions apply for the vector partitioning of the T-matrices.

${}^{j}\mathbf{T}_i$: Homogeneous joint matrix describing orientation and position of the i-th joint coordinate system with respect to the j-th joint coordinate system.

${}^{j}\mathbf{T}_i = {}^{j}\mathbf{A}_{j+1} \cdot {}^{j+1}\mathbf{A}_{j+2} \ldots {}^{i-2}\mathbf{A}_{i-1} \cdot {}^{i-1}\mathbf{A}_i$ $\quad$ for i>j

${}^{j}\mathbf{T}_i = {}^{j}\mathbf{A}_{j-1} \cdot {}^{j-1}\mathbf{A}_{j-2} \ldots {}^{i+2}\mathbf{A}_{i+1} \cdot {}^{i+1}\mathbf{A}_i$ $\quad$ for i<j

For writing simplication, $\mathbf{T}_i$ means the same as ${}^{o}\mathbf{T}_i$ and $\mathbf{A}_i$ the same as ${}^{i-1}\mathbf{A}_i$

${}^{j}\mathbf{ROT}_{i\star}$	=	[**n**,**o**,**a**]	Orientation matrix subpartition, orthogonal
${}^{j}\mathbf{TRL}_{i\star}$	=	**p**	Positional vector subpartition.

(Substitute "D" for "*" in the case of direct, "I" in the case of inverse manipulation. Leave out "*" in the case of the A-matrices and their inverses)

$\mathbf{J}$, $\mathbf{J}^{-1}$, $\dot{\mathbf{J}}$: The Jacobian, its inverse and time derivative, respectively.

n: Number of degrees of freedom of the robot arm.

Denavit/Hartenberg parameters/variables:

θ_i : Rotation angle about the (i-1)th z-axis.
d_i : Translation units along the (i-1)th z-axis.
a_i : Translation units along the (i-1)th x-axis.
α_i : Rotation angle about the i-th x-axis.

q_i, $\dot{q}_i$, $\ddot{q}_i$: Generalized coordinate of the i-th coordinate system expressed with respect to the (i-1)th system, its velocity and acceleration, respectively.

d/dq_j : Operator of standard derivative with respect to q_j

$\delta/\delta q_j$: Operator of partial derivative with respect to q_j

d^2/dq_j^2 : Second order operator of standard derivative with respect to q_j

$\delta^2/\delta q_j q_k$: Second order operator of partial derivative with respect to q_j and q_k

$\mathbf{A}_i'$, $\mathbf{A}_i''$: First order and second order standard derivative of $\mathbf{A}_i$, respectively.

$\mathbf{A}_i' =$	$d/dq_i(\mathbf{A}_i)$	$\mathbf{A}_i'' =$	$d^2/dq_i^2(\mathbf{A}_i)$
$\mathbf{ROT}_i' =$	$d/dq_i(\mathbf{ROT}_i)$	$\mathbf{ROT}_i'' =$	$d^2/dq_i^2(\mathbf{ROT}_i)$
$\mathbf{TRL}_i' =$	$d/dq_i(\mathbf{TRL}_i)$	$\mathbf{TRL}_i'' =$	$d^2/dq_i^2(\mathbf{TRL}_i)$

${}^k f_{nI}$: Magnitude square of ${}^k\mathbf{TRL}_{nI}$

$^{k}f_{nD}$: Magnitude square of $^{k}\mathbf{TRL}_{nD}$

$^{n}\mathbf{E}_{n}$: "Error matrix"

Trace(**T**) : The trace vector of the orientation submatrix of the transform **T**, dimensioning (3X1).

diff_trace_square : The square of the magnitude of the difference vector between the trace vectors of $^{n}\mathbf{E}_{n}$ and $^{n}\mathbf{T}_{nD}$

$Q^{(n)}$: n-th order general derivative matrix operator

$QT^{(n)}$: n-th order translation derivative matrix operator

$QR^{(n)}$: n-th order rotation derivative matrix operator

$\mathbf{U}_{pj}$: First order partial derivative of $\mathbf{T}_{p}$ with respect to the generalized variable q_j

$\mathbf{U}_{pjk}$: Second order partial derivative of $\mathbf{T}_{p}$ with respect to the generalized variables q_j and q_k

$^{k}\dot{\mathbf{T}}_{p}$, $^{k}\ddot{\mathbf{T}}_{p}$: Velocity and acceleration of the p-th coordinate system seen from the k-th system, respectively.

F_i : Generalized force acting at joint i, representing scalar component of the force/torque vector that has only the z-component at joint i.

D_{ij} : Inertia "dynamics projection function".

D_{ijk} : "Dynamics projection function" of the centrifugal and Coriolis forces/torques.

D_i : Gravity related "dynamics projection function".

J_p : (4X4) symmetric pseudo inertia matrix.

m_p : Mass of link p.

$\mathbf{r}_p$: Mass center vector of link p with respect to the p-th coordinate system, dimensioning (4X1).

$\mathbf{g}$: Gravity vector dimensioning (4X1).

T_p : Period of sinewave carriers.

T_M : Machine cycle.

m : Bit number of angle resolution.

r : Bit number of data resolution.

T : General designation for time delays.

t : General designation for time variable.

Chapter 1

INTRODUCTORY REVIEW

1.1 Survey of Relevant Literature

Table 1.1 illustrates the diversity of backgrounds required in robotics research. For sophisticated robot systems, all the six aspects listed in the table are equally important.

In sections 3.) and 4.), of Table 1.1, the following five major areas of research and development have been identified:

- Kinematics and dynamics,
- Feedback control,
- Trajectory planning,
- Compliance
- Task planning.

These five major areas involve the most important problems of manipulator planning and control, and this thesis is particularly concerned with kinematics and dynamics of robot manipulators. Interest is concentrated on the real-time manipulation of the matrix equations involved in kinematic and dynamic calculations of robot manipulators.

Firstly, the inverse kinematic problem is defined.

Major Research/Development Aspects of Robotics

1.) Representation, Modeling.

- a.) Objects.
- b.) Laws of Nature.
- c.) Processes.

2.) Sensors.

- a.) Hardware.
- b.) Interpretation.
- c.) Interaction.

3.) Manipulation.

- a.) Robot Design.
- b.) Kinematics and Dynamics.
- c.) Feedback Control.
- d.) Trajectory Planning.

4.) Intelligent Superstructure.

- a.) Organization.
- b.) Robot Programming Languages.
- c.) Compliance.
- d.) Task Planning.
- e.) Artificial Intelligence.

5.) Locomotion.

6.) Integration and Applications.

Table 1.1.

The formula for the transformation of the n-th coordinate system of a kinematic open-chain robot arm with n degrees of freedom (d.o.f.) into the zero-th coordinate system is given by $\mathbf{T}_n = \mathbf{A}_o \mathbf{A}_1 \mathbf{A}_2 \ldots \mathbf{A}_n$ Eq.1.1

where in Eq. 1.1 we make use of the four-parameter/variable representation of Denavit/Hartenberg:

$$ {}^{i-1}\mathbf{A}_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Eq. 1.2

which expresses orientation and position of the i-th coordinate system with respect to the (i-1)th coordinate system. The meanings of the four parameter/variable θ, α, a and d can be found in a number of literature [1, 4, 5]. If the i-th joint is rotational, the generalized variable q_i of the matrix Eq. 1.2 will be θ_i. However, if the joint is translational, the variable will be $q_i = d_i$. q_i is called the generalized coordinate of the i-th joint. Thus, the set of n generalized coordinates q_i for $i = 1,2,\ldots,n$ denoted by $\mathbf{q} = (q_1, q_2, \ldots, q_n)^T$ is called the position vector of the generalized coordinates in the space R_q^n. Correspondingly, the vector

$\dot{\mathbf{q}} = (\dot{q}_1, \dot{q}_2, \ldots, \dot{q}_n)^T$ is called the velocity vector of the gneralized coordinates in the space $R_{\dot{q}}^n$, and $\ddot{\mathbf{q}} = (\ddot{q}_1, \ddot{q}_2, \ldots, \ddot{q}_n)^T$ the acceleration vector of the generalized coordinates in the space $R_{\ddot{q}}^n$.

Assuming all the geometric parameters of a robot arm with n degrees of freedom are known, the most general statement of the inverse kinematic problem of this robot is formulated as follows.

If the motion characteristics of the robot arm at a particular time are defined by:

a) the position and orientation of the n-th coordinate system expressed with respect to the zero-th coordinate system ${}^{o}\mathbf{T}_n$,

b) The Cartesian velocity $\mathbf{v}=(v_x, v_y, v_z)^T$ and the angular velocity $\omega=(\omega_x, \omega_y, \omega_z)^T$ of the n-th coordinate system seen from the zero-th coordinate system,

c) the Cartesian acceleration $\dot{\mathbf{v}}$ and the angular acceleration $\dot{\boldsymbol{\omega}}$ of the n-th coordinate system realized form the zero-th coordinate system.

Then the problem is to find the vectors $\mathbf{q}$, $\dot{\mathbf{q}}$ and $\ddot{\mathbf{q}}$ in the space R_q^n, $R_{\dot{q}}^n$ and $R_{\ddot{q}}^n$, respectively, such that the characteristics of the motion of the n-th coordinate system at time t coincide with the desired ones.

For simple manipulator types, the usual method for deriving the solution by a given ${}^{o}\mathbf{T}_{n}$ is to compare matrices or matrix compositions obtained in some intuitive ways from Eq. 1.1 on the basis of element by element where, as indicated, the solution derivation may require geometric intuitions because there is no algorithm for obtaining the solution. The solution $\mathbf{q}$ is sometimes called "inverse kinematic solution for position" or, preferably, "inverse geometric solution" because it only considers the geometric configuration of the robot manipulator. For real-time applications, this solution is of the "utmost importance" [5] and can be found sufficiently quickly as, typically, for practical manipulators, it involves only some ten additions and multiplications, and about ten/twelve transcendental function calls [5, 6, 43, 44].

In general, the inverse kinematic solution for velocity, namely $\dot{\mathbf{q}}$, can be obtained by the inversion of

$$\dot{\mathbf{x}} = \mathbf{J} . \dot{\mathbf{q}} \qquad \text{Eq. 1.3}$$

where $\dot{\mathbf{x}} = (v, \omega)^{T}$ such that

$$\dot{\mathbf{q}} = \mathbf{J}^{-1} . \dot{\mathbf{x}} \qquad \text{Eq. 1.4}$$

$\mathbf{J}$ is known as the Jacobian matrix of dimension 6xn relating the first order differential changes between the cartesian coordinates and the generalized coordinates of the robot arm (with n d.o.f.). Note that when n<>6, the matrix $\mathbf{J}$ will not

have square form and thus some mathematical measures ("pseudo-inverse") must be devised such that Eq. 1.4 is defined. However, even when **J** is a square matrix, the inversion of **J** in Eq. 1.4 represents an undesirable computating operation considering that, for example, **J** may become singular. In addition, it should also be said that very often only the solution is of interest but not the Jacobian itself. A practical derivation of the Jacobian and its inverse can be found in Ref 007 and Ref044.

The inverse acceleration solution $\ddot{\mathbf{q}}$ can be derived by first differentiating Eq. 1.3 to obtain $\ddot{\mathbf{x}} = \mathbf{J}.\ddot{\mathbf{q}} + \dot{\mathbf{J}}.\ \dot{\mathbf{q}}$ which then yields:

$$\ddot{\mathbf{q}} = \mathbf{J}^{-1}.\ (\ddot{\mathbf{x}} - \dot{\mathbf{J}}.\ \dot{\mathbf{q}}) \qquad \text{Eq. 1.4.b}$$

As can be seen, this solution involves not only the inverse of the Jacobian, but also its first order time derivative. In addition, it also requires the knowledge of the inverse velocity solution $\dot{\mathbf{q}}$.

If the inverse kinematic solution for velocity is seldom performed due to impractical computation procedures associated with the Jacobian matrix, the inverse kinematic solution for acceleration is performed less often.

Unlike robot kinematics that deals with the motion of the robot without regard to the forces/torques which cause the

motion, robot dynamics deals with the mathematical formulations of the equations of robot arm motion that take those forces/torque into account. Such motion equations are indispensable to the development of suitable control laws/strategies.

The Lagrange dynamics formula is [4,5,8].

$$F_i = \sum_{j=1}^{n} D_{ij}\,\ddot{q}_j + \sum_{j=1}^{n}\sum_{k=1}^{n} D_{ijk}\,\dot{q}_j\,\dot{q}_k + D_i \qquad ; i=1,\ldots,n$$

Eq. 1.5

where

$$D_{ij} = \sum_{p=\max(i,j)}^{n} \mathrm{Trace}\left(U_{pj}\, J_p\, U_{pi}^{T}\right) \qquad ; U_{pj} = \frac{\partial}{\partial q_j}{}^{o}T_p$$

$$D_{ijk} = \sum_{p=\max(i,j,k)}^{n} \mathrm{Trace}\left(U_{pjk}\, J_p\, U_{pi}^{T}\right) \qquad ; U_{pjk} = \frac{\partial^2}{\partial q_j \partial q_k}{}^{o}T_p$$

$$D_i = \sum_{p=i}^{n} -m_p \cdot g^{T} \cdot U_{pi} \cdot r_p$$

This formula represents a set of n second-order, nonlinear, highly coupled, ordinary differential equations which, in the "inverse dynamic calculations", involve a number of multiplications and additions of the order $O(n^4)$ too time consuming [23] to compute in real-time. Luh, Walker and Paul

state that a FORTRAN simulation takes 7.9 seconds on the PDP11/45 to compute a single set of the generalized forces F_i i=1, 2,.....6; for a robot with n=6 d.o.f. This is, in the most optimistic estimation, several hundred times too slow for some manipulation tasks which require a sampling frequency of, for example, no lower than 60Hz (or 16.67 milli second sampling time-interval).

By contrast, using an improved version for the Newton-Euler dynamic formulation (forward/backward recursions), the authors state that it only takes 33.5 milli seconds in the FORTRAN implementation. This is mainly because of the recursive nature of the Newton-Euler formulation. The computing time can be reduced to 4.5 milli-seconds if the procedure is written in assembly language.

Such computational efficiencies lead to methods of solving for the relative joint accelerations[24] given the input torques/forces and trajectory pre-specified by $\mathbf{q}$ and $\dot{\mathbf{q}}$. Such calculations are of interest to robot simulation as a computing cost of order n^3 and n^2 is still involved.

Let us stay with the "inverse dynamic problem". The re-formulation of the Eq. 1.5 using recursive formulas reveals equivalence between the Lagrange and the Newton-Euler formulation [9] reducing the number of multiplications and additions to the linear order O(n) so that in principle it

should be possible to compute the Lagrange dynamics in real-time [8,22].

However, the recursive formulations whether Lagrangian or Newton-Euler destroy the structure of the dynamic model. In particular, the state variables $\mathbf{q}$, $\dot{\mathbf{q}}$ and $\ddot{\mathbf{q}}$ disappear from those recurrent dynamic equations so making it more difficult to apply advanced control laws.

The purpose of robot arm control is to maintain a desired dynamic response of the robot arm. In general, the control problem consists of obtaining a suitable dynamic model of the robot arm necessary for the design of the controller and specifying control laws/strategies in order to achieve the desired system response and performance.

The formula of Eq. 1.5 represents the general formulation of the explicit state equations of robot arm dynamics. It is motivated by the fundamental idea of feedback control which states that the control inputs will be computed from the state of the system because, by definition of a dynamic system, the state incorporates all information necessary to determine the control action to be taken for a desired system behaviour.

Various control schemes exist. One of these, related to the fundamental idea of feedback control is the "inverse problem" control scheme (or"computed torque" control scheme).

Consider the Eq. 1.5. we may write it in the form

$$\mathbf{F} = \mathbf{D}(\mathbf{q}) \cdot \ddot{\mathbf{q}} + \mathbf{H}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) \qquad \text{Eq. 1.6}$$

Where

$\mathbf{F}$: (nx1) vector of generalized forces.

$\mathbf{D}(\mathbf{q})$: (nxn) acceleration-related inertia matrix.

$\mathbf{H}(\mathbf{q},\dot{\mathbf{q}})$: (nx1) velocity-related vector involving all the nonlinear and coupled terms of the equation.

$\mathbf{G}(\mathbf{q})$: (nx1) gravity-related vector.

Basically, the "inverse problem" control technique is a feedforward controller and assumes that we can accurately compute the computer parts of $\mathbf{D}(\mathbf{q})$, $\mathbf{H}(\mathbf{q},\dot{\mathbf{q}})$ and $\mathbf{G}(\mathbf{q})$.

The desired input torque/force is computed from[16]

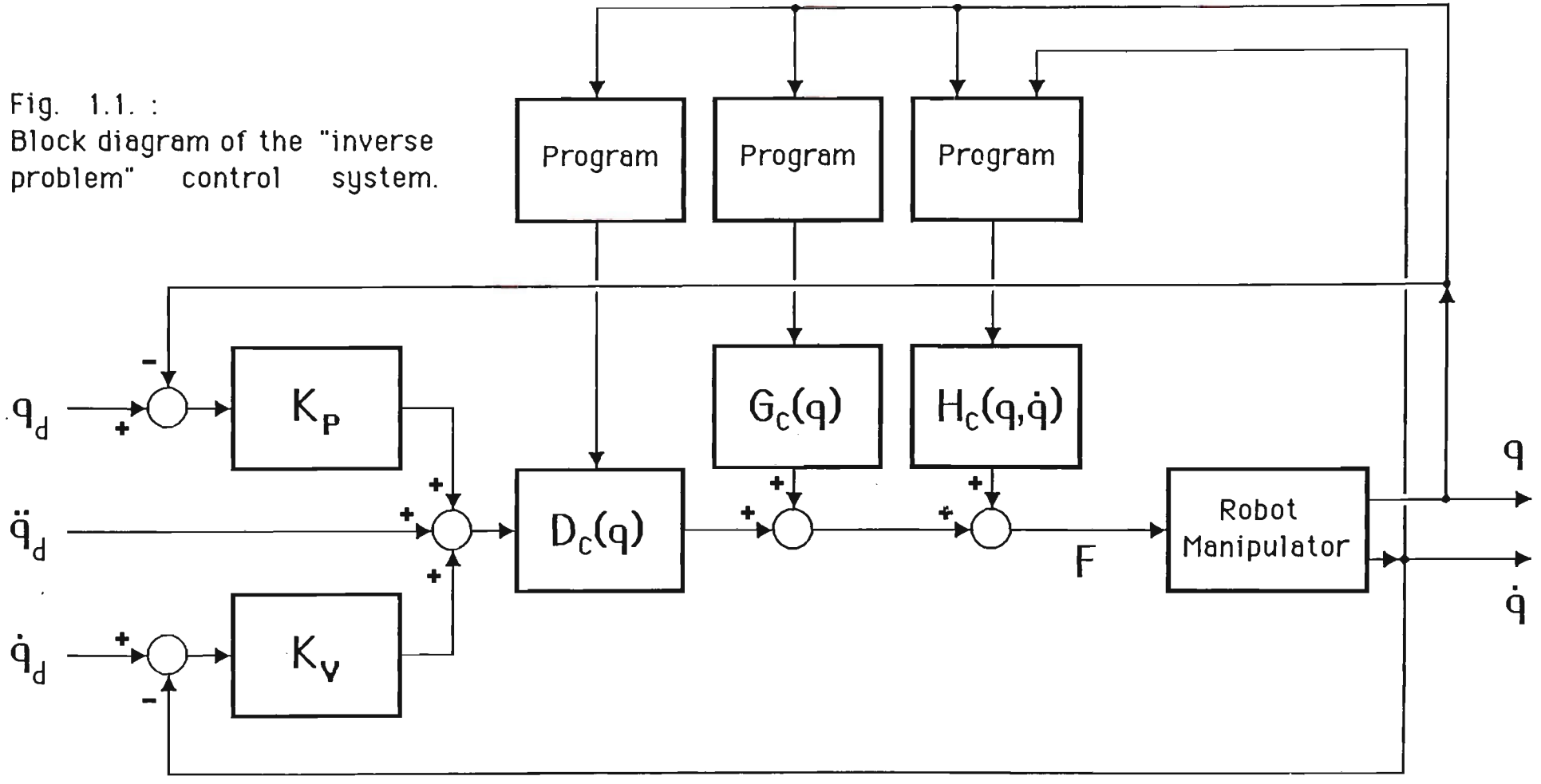
$$\mathbf{F} = \mathbf{D}_c(\mathbf{q}) \cdot \{\ddot{\mathbf{q}}_d + K_v(\dot{\mathbf{q}}_d - \dot{\mathbf{q}}) + K_p(\mathbf{q}_d - \mathbf{q})\} + \mathbf{H}_c(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{G}_c(\mathbf{q}) \qquad \text{Eq. 1.7}$$

$_c$ is used to indicate computed quantities.

$_d$ is used to indicate desired quantities.

where $\mathbf{K}_v$ and $\mathbf{K}_p$ are (nxn) velocity and position feedback gain matrices, respectively. Fig. 1.1 (slightly revamped from Ref016) shows the block diagram of the "inverse problem" control system.

Fig. 1.1. :
Block diagram of the "inverse problem" control system.



$$F = D_c(q)\cdot\{\ddot{q}_d + K_v(\dot{q}_d - \dot{q}) + K_p(q_d - q)\} + H_c(q,\dot{q}) + G_c(q)$$

This technique is an effective method of manipulator control provided $\mathbf{q}$ converges to $\mathbf{q}_d$. Ideally, we wish to have the computed values $\mathbf{D}_c(\mathbf{q})$, $\mathbf{H}_c(\mathbf{q},\dot{\mathbf{q}})$ and $\mathbf{G}_c(\mathbf{q})$ exactly equal to their actual counterparts

$$\mathbf{D}_c(\mathbf{q}) = \mathbf{D}(\mathbf{q}) \qquad \text{Eq. 1.8}$$

$$\mathbf{H}_c(\mathbf{q},\dot{\mathbf{q}}) = \mathbf{H}(\mathbf{q},\dot{\mathbf{q}}) \qquad \text{Eq. 1.9}$$

$$\mathbf{G}_c(\mathbf{q}) = \mathbf{G}(\mathbf{q}) \qquad \text{Eq. 1.10}$$

Assuming, Eq. 1.8, Eq. 1.9 and Eq. 1.10 are satisfied, then their substitution into Eq. 1.7 followed by a comparison with Eq. 1.6 yields:

$$\mathbf{D}(\mathbf{q}).\{\ddot{\mathbf{q}}_d-\ddot{\mathbf{q}} + \mathbf{K}_v(\dot{\mathbf{q}}_d-\dot{\mathbf{q}}) + \mathbf{K}_p(\mathbf{q}_d-\mathbf{q})\} = 0 \qquad \text{Eq. 1.11}$$

Let $\mathbf{e}_q = \mathbf{q}-\mathbf{q}_d$ and consider that the inertia matrix $\mathbf{D}(\mathbf{q})$ is nonsingular, then Eq. 1.11 reduces to

$$\ddot{\mathbf{e}}_q + \mathbf{K}_v\dot{\mathbf{e}}_q + \mathbf{K}_p\mathbf{e}_q = 0 \qquad \text{Eq. 1.12}$$

If $\mathbf{K}_v$ and $\mathbf{K}_p$ are so chosen that the characteristic roots of Eq. 1.12 have negative real part, then $\mathbf{e}_q$ approaches zero asymptotically. Note that the convergence relies on the validity of Eq. 1.8, Eq. 1.9 and Eq. 1.10, which, however, for real-time applications are very difficult to use because they are nothing else than the particular parts of the

complete dynamic equations of Eq. 1.5 which take 7.9 seconds to be computed.

Adopting the idea of the "inverse problem" control, Luh, Walker and Paul (1980) developed the Resolved Motion Acceleration Control technque (RMAC). Basically, the RMAC is a technique for position control of the manipulator in terms of the position and orientation of the end effector where the computation of $\mathbf{D}_c(\mathbf{q})$, $\mathbf{H}_c(\mathbf{q},\dot{\mathbf{q}})$ and $\mathbf{G}_c(\mathbf{q})$ is avoided and thus the satisfaction of Eq. 1.8, Eq. 1.9 and Eq. 1.10 for the purpose of convergence $\mathbf{q} \dashrightarrow \mathbf{q}_d$ is "immaterial". The basis for the RMAC is the following equation [16]:

$$\ddot{\mathbf{q}} = \mathbf{J}^{-1}.\ [\ddot{\mathbf{x}} - \dot{\mathbf{J}}.\dot{\mathbf{q}}] \qquad \text{Eq. 1.13}$$

where $\mathbf{J}^{-1}$ and $\dot{\mathbf{J}}$ are the inverse and the time derivative of the Jacobian matrix, respectively.

In this control scheme, $\ddot{\mathbf{q}}$ is specified while $\dot{\mathbf{q}}$ and $\mathbf{q}$ are measured; recalling that these state variables are needed in the "inverse problem" calculation resulting in the required forces/torques whereby, however, for real-time reasons the Newton-Euler dynamic formulation is used.

In a particular simulation on the PDP11/45 the authors stated that this control scheme made a sampling frequency of 87 Hz possible.

One major drawback with the RMAC is the involvement of the inverse of the Jacobian matrix as well as its time-derivative. It is therefore not surprising that this method has not been applied for n<>6. Johnson states in a review that the practicality of RMAC-implementation must be questioned [25].

The RMAC has been an improvement of the RMRC: the Resolved Motion Rate Control technique, which was developed by Whitney [13]. The basic equation is:

$$\dot{\mathbf{q}} = \mathbf{J}^{-1} . \dot{\mathbf{x}} \qquad \text{Eq. 1.14}$$

In this method, the velocity $\dot{\mathbf{q}}$ is specified, the position $\mathbf{q}$ is measured and the acceleration $\ddot{\mathbf{q}}$ is obtained numerically. The involvement of the inverse Jacobian in this control scheme represents a setback as in RMAC because additional computations and all the problems associated with matrix singularities arise.

Another system, the Resolved Motion Position Control (RMPC) technique relies on the solution of Eq. 1.1. In this method, the Eq. 1.1 provides the position vector $\mathbf{q}$. However, the velocity $\dot{\mathbf{q}}$ and the acceleration $\ddot{\mathbf{q}}$ are obtained numerically; based on this information, the RMPC closes n position servos around n joint actuators.

Fig. 1.2 shows a "popular" joint servo used in today's industrial robots using this technique. This simple form of "linear de-centralized independent joint control scheme" works well and is more reliable than present multivariable methods [25]. However, adequate disturbance rejection requires very high power actuators which tend to be over-powered at low manipulation speeds giving undesirable vibrations.

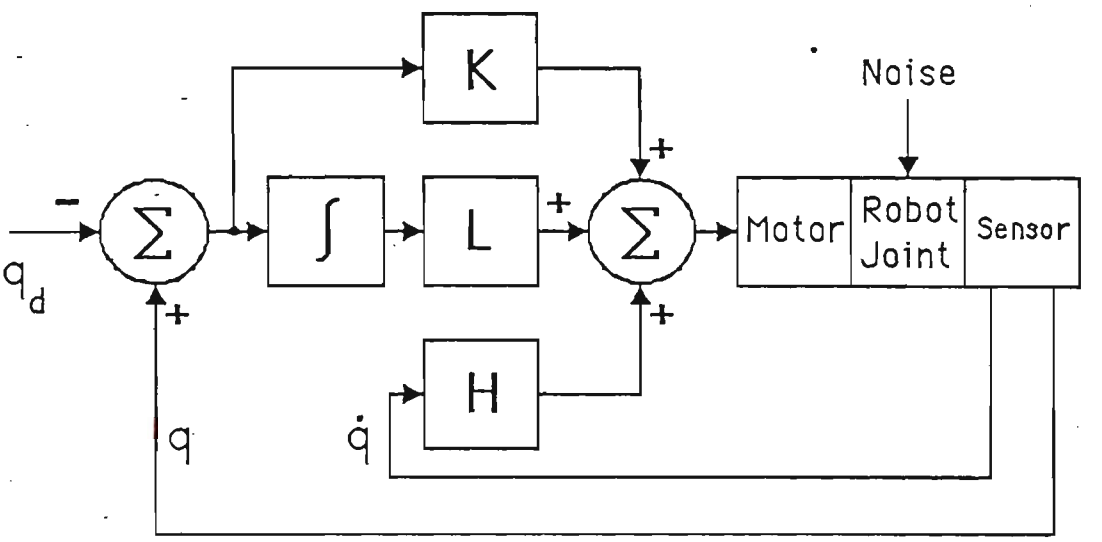


Fig. 1.2.

Let us return to Eq. 1.5. Due to excessive computational requirements, attempts [10] have been made to simplify the dynamic model. However, such simplified models apply only for a particular robot's geometry. In the same sense of reducing computational requirements, where the structure of the dynamic model is retained, the Cerebellar Model Articulated Control Scheme has been developed.

The Cerebellar Model Articulated Control CMAC is a table lookup method* . It computes control functions by referring to a table stored in the computer memory rather than computing analytic expressions. However, before useful applications are possible, several problems such as memory size management and accuracy need to be solved. The table is entered for a particular robot and becomes useless as soon as the effect of unknown loads comes into account [8], especially when the load changes suddenly.

Concurrently, Adaptive Control Schemes for robotic applications have also been developed. In this relation, we would like to state the prominent names of Dubowsky (1979) and Lee (1982), for example, from Ref015 and Ref021. These control schemes fundamentally face the problems of parameter/variable estimation and identification which, as is known, may take considerable efforts. It is not surprising that we are provided only with bare simulation results as the methods have virtually not been applied in real-time environments yet.

Ref017 is a relatively recent development. One feature of this Resolved Motion Force Control Scheme (RMFC) we should mention is that the transpose of the Jacobian is involved rather than its inverse.

---

* The relevant names encountered in the literature are Raibert and Albus.

1.2 Emphasis of this Research Project

None of the above-mentioned multivariable control algorithms is sufficiently mature to be applied in the practice [25]. A common method employed in today's robot control is the linear independent joint control scheme [27]. Improvements can be achieved by independently applying nonlinear control laws for each joint drive [26]; however, the strategy of decoupling the robot's joints still remains the same. And the computation burdens (see Fig 1.1) associated with the "complete" explicit-state Lagrange dynamic equation have been the main reason for the development of the various control algorithms. Even though it is not necessary to insist on the complete and exact dynamic model it is desirable to keep the explicit-state structure of the model so that the system states are available for use in implementing control strategies.

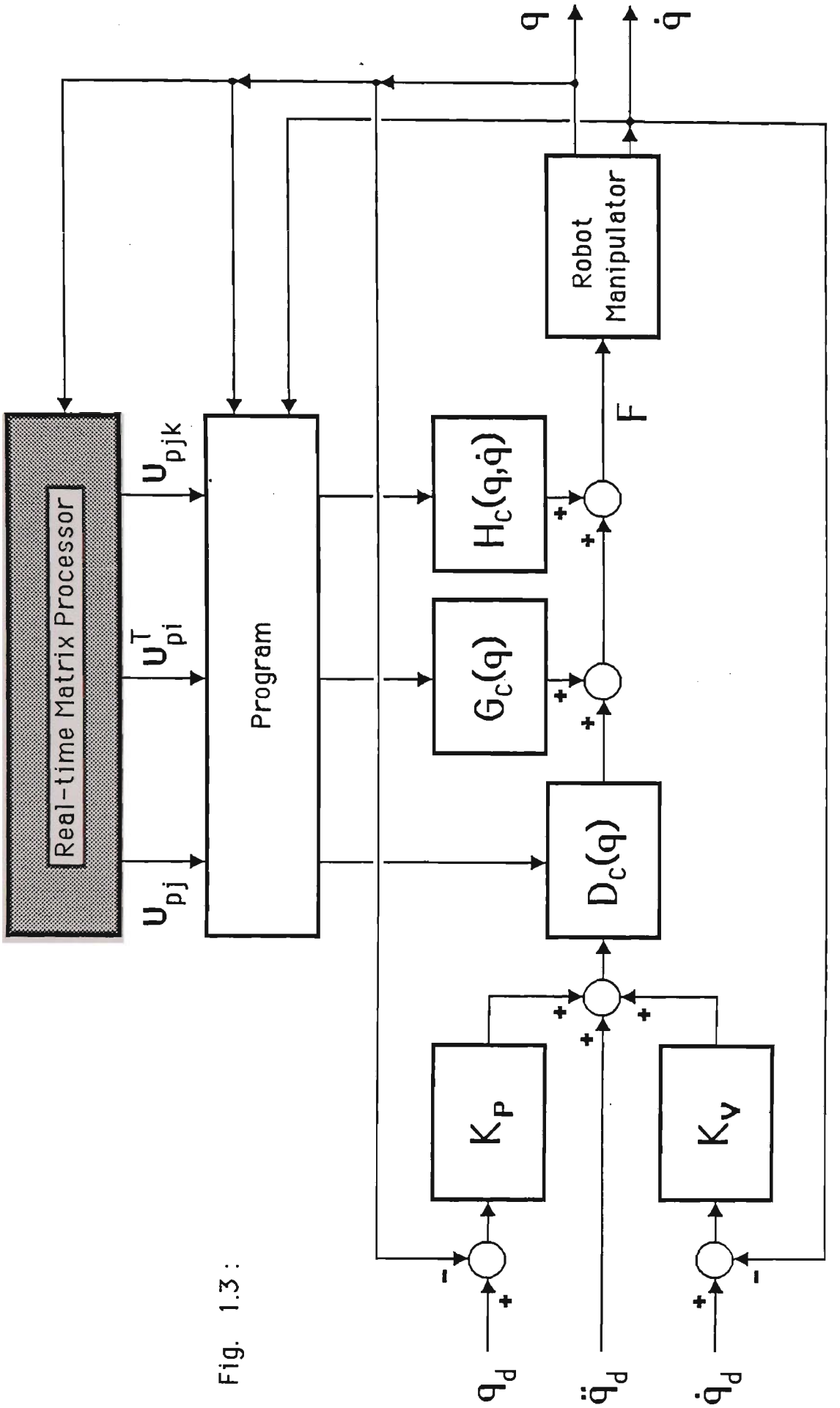

Real-time Matrix Processor
U_{pjk}
U_{pi}^{T}
U_{pj}
Program
$H_c(q,\dot{q})$
$G_c(q)$
$D_c(q)$
K_P
K_V
Robot Manipulator
F
q
$\dot{q}$
q_d
$\ddot{q}_d$
$\dot{q}_d$

Fig. 1.3 :

In order to provide a solution to the Lagrange inverse dynamic Problem in real time, the overall strategy is to locate the computation burden "outside" the computer as much as possible. The recent literature also shows this tendency [36,37,38,40]. However, one common characteristic of all these approaches is that they still employ conventional arithmetic operations which are executed on the external hardware. In the research work to be presented here this is done by developing a real-time matrix processor to perform the function as indicated by the shaded box in Fig. 1.3.

In this thesis a strategy is developed to avoid arithmetic operations such as additions and multiplications as far as possible.

In particular, let us consider the following:

Examining, Eq. 1.5, we realize that the coefficients D_i, D_{ij} and D_{ijk} of the robot's dynamic differential equation system are not constant quantities, but they vary with every robot's position. This is, indeed, because they involve the first and second order partial derivatives $\mathbf{U}_{pj}$ and $\mathbf{U}_{pjk}$ which are given as:

$$\mathbf{U}_{pj} = {}^{0}\mathbf{A}_1 . \; {}^{1}\mathbf{A}_2 \ldots \; {}^{j-1}\mathbf{A}'_j \ldots {}^{p-1}\mathbf{A}_p$$

$$\mathbf{U}_{pjk} = {}^{0}\mathbf{A}_1 . \; {}^{1}\mathbf{A}_2 \ldots \; {}^{j-1}\mathbf{A}'_j \ldots \; {}^{k-1}\mathbf{A}'_k \ldots {}^{p-1}\mathbf{A}_p$$

$j, k =< p$ and $p = 1,2,\ldots, n$

where ${}^{j-1}\mathbf{A}'_j$ and ${}^{k-1}\mathbf{A}'_k$ are first order derivative of the matrix ${}^{j-1}\mathbf{A}_j$ and ${}^{k-1}\mathbf{A}_k$ with respect to the generalized coordinate q_j and q_k, respectively.

As is known, calculating those derivatives involves a number of additions and multiplications of the order $O(n^3)$ and $O(n^4)$ representing the substantial computing costs of the dynamic equations. Thus, it would be a substantial and potential saving on the total computing cost if we could reduce those orders $O(n^3)$ and $O(n^4)$ to a lower order, say $O(n^2)$ or even linear $O(n)$, without unnecessarily neglecting any terms of those derivatives.

It would be best if those number of additions and multiplications could be eliminated entirely. Saying this, we inevitably arrive at the idea of implementing those derivatives using some simulation technique on external hardware rather than employing conventional arithmetic expressions on the digital computer. The computer would then be released from the task of calculating $\mathbf{U}_{pj}$ and $\mathbf{U}_{pjk}$. This is necessary towards the real-time computation of Eq. 1.5.

Thus, in this thesis, we focus on this problem with the ultimate aim of enabling the application of advanced control

laws for robot control in the fundamental sense of feedback control.

If the A-matrix of Eq. 1.2 is examined. It can be seen that it is made up of four successive elementary transformations as shown below:

$$
{}^{i-1}A_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & 0 \\ \sin\theta_i & \cos\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
\cdot \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha_i & -\sin\alpha_i & 0 \\ 0 & \sin\alpha_i & \cos\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

Eq. 1.15

Examining this matrix multiplication and bearing in mind that the two (orthonormal) elementary rotations primarily involve trigonometric functions such as sine and cosine, an approach which expresses particular matrix elements in the form of some orthogonal time signals, for example sinusoidal signals is developed. The manipulation of the relevant matrices is then performed by directly manipulating the signals rather than by numerically computing conventional arithmetic expressions.

This approach is introduced in the following chapters by considering a new method of describing robot kinematics, initially in 2-dimensions followed by a generalization to three dimensions.

It is shown that this technique has several advantageous novel characteristics when applied to robot kinematics. The method is then applied to the development of robot dynamics to overcome some of the problems with existing techniques described above.

Chapter 2

THE TWO-DIMENSIONAL ROBOT JOINT DESCRIPTOR (2D-RJD)

2.1 INTRODUCTION

Let us consider the following general homogeneous transformation matrix

$$\mathbf{T} = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ o & o & o & 1 \end{bmatrix}. \qquad \text{Eq. 2.1.1}$$

In order not to define new phrases unnecessarily, we will use the same term "equation" (abbr.: Eq.) for some definitions such as Eq.2.1.1 as well as for "true" equations.

The subscripts "x" "y" and "z" in Eq. 2.2.1 indicate the general cartesian components of the four particular column vectors of the transform. The (4x4) transform is homogeneous in the sense that a particular perspective projection will recreate the three-dimensional space. Indeed, the three-dimensional cartesian components of the vector $[p_x, p_y, p_z, 1]^T$ are found by taking division of its first three elements by the fourth element. In the same sense, a vector of the form $[n_x, n_y, n_z, 0]^T$ is a vector at infinity and represents a direction or orientation since the addition of any other finite vector does not change its meaning in anyway. We will only consider those transforms whose first three orientation

vectors constitute an orthonormal right handed coordinate system. It has become customary to call the vectors $[n_x, n_y, n_z]^T$, $[o_x, o_y, o_z]^T$, $[a_x, a_y, a_z]^T$ and $[p_x, p_y, p_z]^T$ the normal vector, orientation vector, approach vector and position vector of the transform **T**, respectively.

The homogeneous transformation matrices are used to describe one robot joint coordinate system with respect to another. In the interest of clarity, the matrix indices are indispensable. Thus, let us introduce the notation ${}^j\mathbf{T}_i$.

The leading superscript j and the subscript i of ${}^j\mathbf{T}_i$ will denote the transform **T** describing the i-th coordinate system with respect to the j-th coordinate systems of the robot arm.

Immediately from this, one sees that any transform ${}^j\mathbf{T}_i$ describing location (location = position and orientation) of the i-th system with respect to itself is a unit matrix. "System" means here the same as "coordinate system".

$$ {}^i\mathbf{T}_i = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ i = 0, 1, 2 .... \qquad \text{Eq. 2.1.2} $$

There are special cases of the difference between i and j being +1 or -1, in which we use to use the traditional notation **A** for the transforms rather than **T**. These are known as the elementary "A-matrices", whereby each of them describes position and orientation of one robot link with respect to one of its adjacent links. "Elementary" in the sense that the transform cannot be divided further into more primitive functions, concerning its function as the complete orientational and positional description of the geometrical relationships between adjacent links [4].

The notation $\mathbf{A}_i$ will have the same meaning as ${}^{i-1}\mathbf{A}_i$ if we sometimes omit the leading superscript for writing simplification.

Assuming, a vector quantity $\boldsymbol{r}$ is given in the (i+1)th coordinate system

$$\boldsymbol{r} = [r_x, r_y, r_z, r_w]^T \qquad \text{Eq.2.1.3}$$

where two cases are possible for $\mathbf{r}_w$:

$$\mathbf{r}_w = \begin{cases} 0 & \text{implying } \mathbf{r} \text{ is direction vector with } 0</r/<1 \\ 1 & \text{implying } \mathbf{r} \text{ describes space point} \end{cases} \qquad \text{Eq.2.1.4}$$

In both cases, the following applies:

The transformation of **r** into the i-th system is performed as follows:

$$\mathbf{s} = {}^{i}\mathbf{A}_{i+1} . \mathbf{r} \qquad \text{Eq.2.1.5}$$

The next transformation to the (i-1)th system will be:

$$\mathbf{t} = {}^{i-1}\mathbf{A}_{i} . \mathbf{s} \qquad \text{Eq.2.1.6}$$

Substitution of Eq.2.1.5 into Eq.2.1.6 yields:

$$\mathbf{t} = {}^{i-1}\mathbf{A}_{i} . {}^{i}\mathbf{A}_{i+1} . \mathbf{r} \qquad \text{Eq.2.1.7}$$

It is obvious that $\mathbf{t} = [t_x, t_y, t_z, t_w]^T$

where $t_w = \begin{cases} 0 & 0<[\mathbf{t}]< 1 \text{ if } \mathbf{r} \text{ is directional} \\ 1 & \text{if } \mathbf{r} \text{ is positional} \end{cases}$

we may write:

$$\mathbf{t} = {}^{i-1}\mathbf{T}_{i+1} . \mathbf{r} \qquad \text{Eq.2.1.8}$$

which implies:

$${}^{i-1}\mathbf{T}_{i+1} = {}^{i-1}\mathbf{A}_{i} . \; . {}^{i}\mathbf{A}_{i+1} \qquad \text{Eq.2.1.9}$$

and is interpreted as the representation of the (i+1)th system with respect to the (i-1)th system, as the matrix $\mathbf{A}_{i+1}$ is nothing else than a composition of four column vectors in the sense of Eq. 2.1.3 and Eq. 2.1.4.

If **t** had been given in the (i-1)th system, we would have found its representation in the (i+1)th system to be

$$\mathbf{r} = [{}^{i}\mathbf{A}_{i+1}]^{-1} . [{}^{i-1}\mathbf{A}_{i}]^{-1} \mathbf{t}$$

where we might write

$$\mathbf{r} = {}^{i+1}\mathbf{T}_{i-1}\,\mathbf{t}$$

implying

$${}^{i+1}\mathbf{T}_{i-1} = [{}^{i}\mathbf{A}_{i+1}]^{-1}\,[{}^{i-1}\mathbf{A}_{i}]^{-1} \qquad \text{Eq.2.1.10}$$

and is interpreted as the representation of the (i-1)th system with respect to the (i+1)th system.

Obviously $${}^{i+1}\mathbf{T}_{i-1} = [{}^{i-1}\mathbf{T}_{i+1}]^{-1} \qquad \text{Eq.2.1.11}$$

We may generalize Eq.2.1.9 and Eq.2.1.10 to:

$${}^{k}\mathbf{T}_{i} = {}^{k}\mathbf{T}_{j}\;{}^{j}\mathbf{T}_{i} \qquad \text{Eq.2.1.12}$$

for any i,j,k of interest.

In this, if we substitute k = i, we find ${}^{i}\mathbf{T}_{i} = {}^{i}\mathbf{T}_{j}\;{}^{j}\mathbf{T}_{i}$

or $${}^{i}\mathbf{T}_{j} = [{}^{j}\mathbf{T}_{i}]^{-1} \qquad \text{Eq.2.1.13}$$

which is a generalization of Eq.2.1.11.

We may write ${}^{j}\mathbf{T}_{i}$ in the following partitioning representation:

$$ {}^{j}\mathbf{T}_i = \begin{bmatrix} {}^{j}\mathbf{ROT}_i & {}^{j}\mathbf{TRL}_i \\ 0 & 1 \end{bmatrix} \qquad \text{Eq.2.1.14} $$

which has the general form:

$$ {}^{j}\mathbf{T}_i = \begin{bmatrix} \text{Orthonormal} & \\ \text{Rotation} & \text{Translation} \\ \text{Sub-matrix} & \text{Vector} \\ (3\text{x}3) & (3\text{x}1) \\ & \\ \text{Perspective} & \text{Scaling} \\ \text{Transformation} & \text{Factor} \\ (1\text{x}3) & (1\text{x}1) \end{bmatrix} \qquad \text{Eq.2.1.15} $$

In this presentation, the homogeneous transformation matrix can be considered as comprising four sub-matrices: The upper left (3x3) orthonormal sub-matrix represents the orientation and the upper right (3x1) sub-matrix or column vector the position of a co-ordinate system with respect to some reference system; the lower left (1x3) sub-matrix or row vector is the perspective transformation and the lower (1x1) "sub-matrix" consisting of one scalar element is the global scaling factor for the position or translation vector.

We are interested in the partitioning representation of Eq.2.1.12. In this equation, however, let us apply k=m and i=n>m to have:

$$ {}^{m}\mathbf{T}_{n} = {}^{m}\mathbf{A}_{m+1} \cdot {}^{m+1}\mathbf{A}_{m+2} \cdots {}^{n-2}\mathbf{A}_{n-1} \cdot {}^{n-1}\mathbf{A}_{n} \qquad \text{Eq.2.1.16} $$

Let us also write for ${}^{m}\mathbf{T}_{n}$

$$ {}^{m}\mathbf{T}_{n} = \begin{bmatrix} {}^{m}\mathbf{ROT}_{n} & {}^{m}\mathbf{TRL}_{n} \\ 0 & 1 \end{bmatrix} \qquad \text{Eq.2.1.17} $$

The manipulation of Eq.16 then yields in the partitioning representation:

$$ {}^{m}\mathbf{ROT}_{n} = {}^{m}\mathbf{ROT}_{m+1} \cdot {}^{m+1}\mathbf{ROT}_{m+2} \cdots {}^{n-2}\mathbf{ROT}_{n-1} \cdot {}^{n-1}\mathbf{ROT}_{n} \qquad \text{Eq.2.1.18} $$

$$ {}^{m}\mathbf{TRL}_{n} = {}^{m}\mathbf{ROT}_{m+1} \cdot {}^{m+1}\mathbf{TRL}_{n} + {}^{m}\mathbf{TRL}_{m+1} \qquad \text{Eq.2.1.19} $$

The proof can easily be performed by direct substitution and induction (see Appendix I).

The conventions of Eq.2.1.2, Eq.2.1.12 and Eq.2.1.13 also apply for the ROT partitions without any limitation, as far as the use of the matrix indices is concerned.

2.2 DIRECT, INVERSE EQUATIONS AND ERROR MATRIX

Eq 2.1.18 and Eq.2.1.19 of the previous section represent the "direct Equations" indicated by the subscript "D"

$$^{m}\mathbf{ROT}_{nD} = {}^{m}\mathbf{ROT}_{m+1} . {}^{m+1}\mathbf{ROT}_{nD} \qquad \text{Eq.2.2.1}$$

$$^{m}\mathbf{TRL}_{nD} = {}^{m}\mathbf{ROT}_{m+1} . {}^{m+1}\mathbf{TRL}_{nD} + {}^{m}\mathbf{TRL}_{m+1} \qquad \text{Eq.2.2.2}$$

for n > m and m= n-1, n-2,...2,1,0

from which, the "inverse equations" indicated by the subscript "I", follow:

$$^{m+1}\mathbf{ROT}_{nI} = ({}^{m}\mathbf{ROT}_{m+1})^{-1} . {}^{m}\mathbf{ROT}_{nI}$$

$$^{m+1}\mathbf{TRL}_{nI} = ({}^{m}\mathbf{ROT}_{m+1})^{-1} . ({}^{m}\mathbf{TRL}_{nI} - {}^{m}\mathbf{TRL}_{m+1})$$

for n > m+1 and m = 0,1,...n-2,n-1

Replacing the index m+1 by m, we have

$$^{m}\mathbf{ROT}_{nI} = ({}^{m-1}\mathbf{ROT}_{m})^{-1} . {}^{m-1}\mathbf{ROT}_{nI} \qquad \text{Eq.2.2.3}$$

$$^{m}\mathbf{TRL}_{nI} = ({}^{m-1}\mathbf{ROT}_{m})^{-1} . ({}^{m-1}\mathbf{TRL}_{nI} - {}^{m-1}\mathbf{TRL}_{m}) \qquad \text{Eq.2.2.4}$$

for n >m and m= 1,2,...,n-2, n-1,n

By varying the index m, the "direct equations" represent the orientational and positional intermediate results of the manipulation of ${}^{0}\mathbf{A}_1.{}^{1}\mathbf{A}_2\ldots{}^{n-2}\mathbf{A}_{n-1}.{}^{n-1}\mathbf{A}_n$.

In particular

$$\left.\begin{aligned}
{}^{n-1}\mathbf{T}_{nD} &= {}^{n-1}\mathbf{A}_n \\
{}^{n-2}\mathbf{T}_{nD} &= {}^{n-2}\mathbf{A}_{n-1}.\ {}^{n-1}\mathbf{T}_{nD} \\
&\vdots \\
{}^{1}\mathbf{T}_{nD} &= {}^{1}\mathbf{A}_2.\ {}^{2}\mathbf{T}_{nD} \\
{}^{0}\mathbf{T}_{nD} &= {}^{0}\mathbf{A}_1.\ {}^{1}\mathbf{T}_{nD}
\end{aligned}\right\} \qquad \text{Eq.2.2.5}$$

Similarly, the "inverse equations" represent the orientational and positional intermediate results of the manipulation of
${}^{n}\mathbf{A}_{n-1}.{}^{n-1}\mathbf{A}_{n-2}\ldots{}^{3}\mathbf{A}_2.{}^{2}\mathbf{A}_1.{}^{1}\mathbf{A}_0.{}^{0}\mathbf{T}_{nI}$.

In particular

$$\left.\begin{aligned}
{}^{1}\mathbf{T}_{nI} &= {}^{1}\mathbf{A}_0.{}^{0}\mathbf{T}_{nI} \\
{}^{2}\mathbf{T}_{nI} &= {}^{2}\mathbf{A}_1.{}^{1}\mathbf{T}_{nI} \\
&\vdots \\
{}^{n-1}\mathbf{T}_{nI} &= {}^{n-1}\mathbf{A}_{n-2}.{}^{n-2}\mathbf{T}_{nI} \\
{}^{n}\mathbf{T}_{nI} &= {}^{n}\mathbf{A}_{n-1}.{}^{n-1}\mathbf{T}_{nI}
\end{aligned}\right\} \qquad \text{Eq.2.2.6}$$

We recall that ${}^{i}\mathbf{A}_{i-1}.$ is the inverse of ${}^{i-1}\mathbf{A}_i$ for i=1,2,...n. Further, it is noted the ${}^{0}\mathbf{T}_{nI}$ may have any prescribed value. The indices of the ${}^{n}\mathbf{T}_{nI}$ in Eq.2.2.6 suggest that this transform should be unit matrix. However, it will be unit

matrix, only if by a given ${}^{0}\mathbf{T}_{nI}$ the particular A-matrices in the inverse manipulation have some certain, appropriate values. Thus, in general this transform will differ from the unit matrix and, hence, is called the error matrix.

Even this indicates that although derived from the mutually inverse equations, the terms ${}^{m}\mathbf{ROT}_{nD}$ and ${}^{m}\mathbf{ROT}_{nI}$ will, in general, not necessarily be equal. The same applies for the position parts ${}^{m}\mathbf{TRL}_{nD}$ and ${}^{m}\mathbf{TRL}_{nI}$.

The problem of how to get orientation and position of the direct manipulation equal to their counterparts in the inverse manipulation by a given ${}^{0}\mathbf{T}_{nI}$ is, in the terminology of robot control, known as the "inverse kinematic problem for position" of a robot arm with n degrees of freedom.

We introduce the notation ${}^{n}\mathbf{E}_{n}$ for the error matrix ${}^{n}\mathbf{T}_{nI}$ in order to distinguish this particular matrix from the remaining T-matrices.

Let us rewrite the Eq. 2.2.1 and Eq.2.2.2 as

$${}^{i-1}\mathbf{ROT}_{nD} = {}^{i-1}\mathbf{ROT}_{i} \cdot {}^{i}\mathbf{ROT}_{nD} \qquad \text{Eq.2.2.7}$$

$${}^{i-1}\mathbf{TRL}_{nD} = {}^{i-1}\mathbf{ROT}_{i} \cdot {}^{i}\mathbf{TRL}_{nD} + {}^{i-1}\mathbf{TRL}_{i} \qquad \text{Eq.2.2.8}$$

for i = n, n-1, ..., 2,1

Eq.2.2.3. and Eq. 2.2.4 are now rewritten as

$$^{i}\mathbf{ROT}_{nI} = (^{i-1}\mathbf{ROT}_{i})^{-1} . \quad ^{i-1}\mathbf{ROT}_{nI} \qquad \text{Eq.2.2.9}$$

$$^{i}\mathbf{TRL}_{nI} = (^{i-1}\mathbf{ROT}_{i})^{-1} . \quad (^{i-1}\mathbf{TRL}_{nI} - ^{i-1}\mathbf{TRL}_{i}) \qquad \text{Eq.2.2.10}$$

$i = 1,2,\ldots,n-1,n$

Fig.2.2.1 shows a graphical representation of Eq.2.2.7 - Eq.2.2.10 which defines the general Robot Joint Descriptor (RJD) with the 180° symbol being used to represent the inversion of $^{i-1}\mathbf{TRL}_{i}$ to form $- ^{i-1}\mathbf{TRL}_{i}$ as required by Eq. 2.2.10.

By inspection, these equations are readable from the figure. Thus, as far as the graphical representation is concerned, the structure of the RJD is justified. It is obvious that the RJD implements the cascade characteristics of the equations. That is, staying on the direct manipulation path (subscripts D), one sees that the outputs of the i-th RJD will serve as inputs for the next adjacent (i-1)th RJD in arriving at the end result $^{o}\mathbf{T}_{nD}$ for the direct manipulation of Eq.2.2.5, when a number of n appropriate RJDs is cascaded. As is seen, the inverse manipulation paths (subscripts I) also possesses the same characteristics. However, the direction of manipulation is reversed.

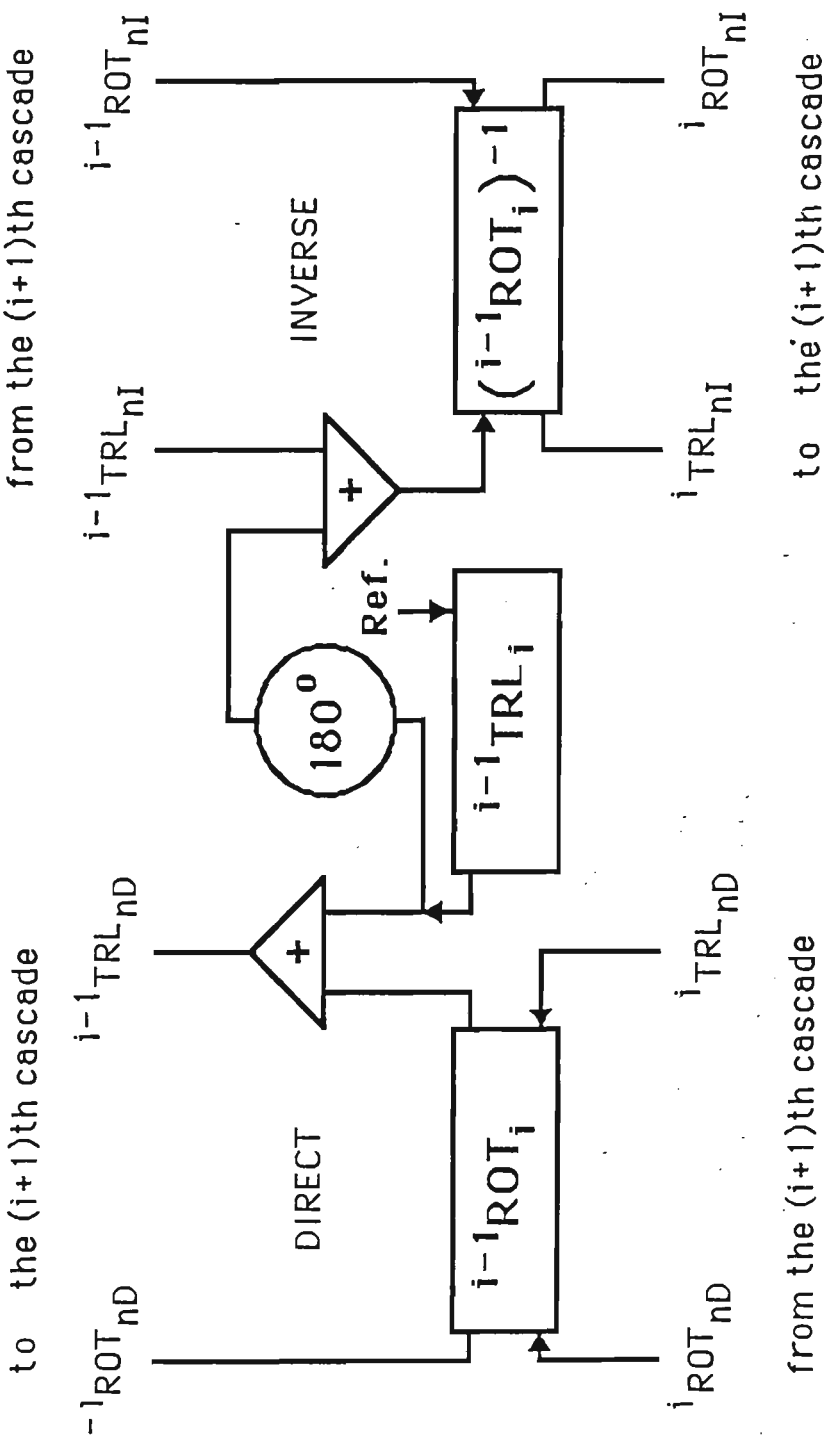


Fig. 2.2.1. : The Generalized Robot Joint Descriptor.

The case for n=3 is demonstrated in Fig 2.2.2. In this figure, the direct kinematic problem of Eq.2.2.5. is represented in the lefthand half. A unit matrix is manipulated in accordance with the direct equations to give the position ${}^{0}\mathbf{TRL}_{3D}$ and orientation ${}^{0}\mathbf{ROT}_{3D}$ of the "end-effector" with respect to the reference.

${}^{3}\mathbf{TRL}_{3D}$ describing the position of system 3 with respect to system 3 is, of course, zero, but is shown for completness to emphasize that the algorithm consists of cascaded joint descriptors, one for each manipulator link, so that manipulators with any number of links can be accommodated by the addition of extra RJDs.

The inverse kinematic problem of Eq.2.2.6 is represented on the righthand half of Fig.2.2.2, where ${}^{0}\mathbf{TRL}_{3I}$ and ${}^{0}\mathbf{ROT}_{3I}$ have the meaning of the desired position and orientation, respectively. If the robot arm is in the desired configuration, the outputs from the $({}^{2}\mathbf{ROT}_{3})^{-1}$ block produce a unit matrix. If not, an error matrix is produced.

We note that at this stage, Fig 2.2.1. and Fig 2.2.2 still have the meaning of a graphical representation of the direct and inverse equations derived above. Their physical implementation will be the subject of the next section.

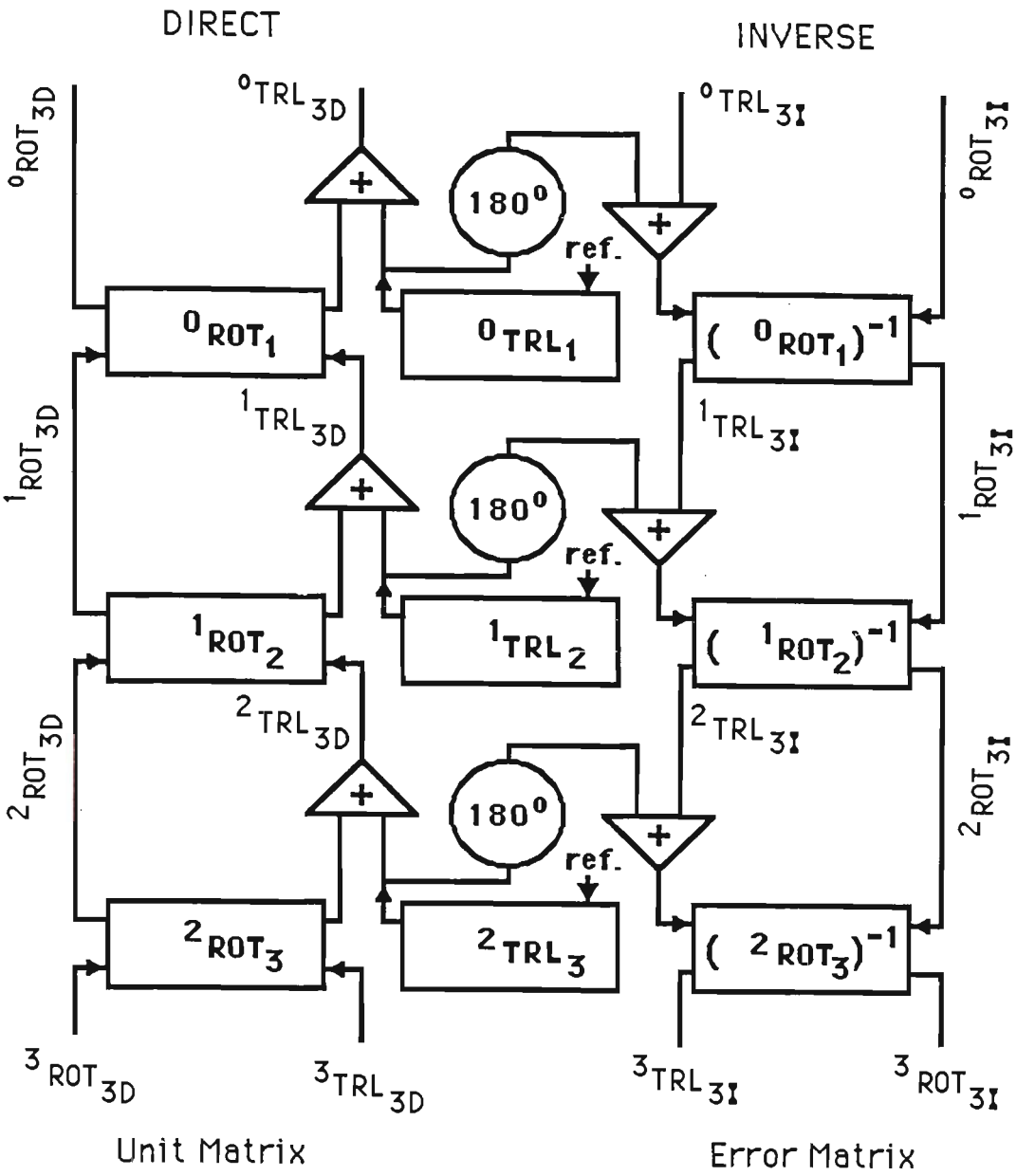


Fig. 2.2.2. : Cascaded Robot Joint Descriptors.

2.3 TWO DIMENSIONAL APPROACH USING SINEWAVE SIMULATION

Let us consider the real and imaginary part of the complex quantity $s = re^{-j(2\Pi ft-\theta)}$

$$re\{s\} = r.\cos(2.\Pi.f.t-\theta)$$
$$= r.\cos(\theta).\cos(2.\Pi.f.t) + r.\sin(\theta).\sin(2.\Pi.f.t)$$
$$= x.\cos(2.\Pi.f.t) + y.\sin(2.\Pi.f.t)$$

and

$$Im\{s\} = -r.\sin(2.\Pi.f.t-\theta)$$
$$= -r.\cos(\theta).\sin(2.\Pi.f.t) + r.\sin(\theta).\cos(2.\Pi.f.t)$$
$$= -[x.\sin(2.\Pi.f.t) - y.\cos(2.\Pi.f.t)]$$

(where $x=r.\cos\theta$ and $y = r.\sin\theta$)
which lead to the following interpretations.

(I) If we interpret a physical sinusoidal signal as the real part of $r.e.^{-j.(2.\Pi.f.t-\theta)}$, which is $r.\cos(2.\Pi.f.t-\theta)$, it will then be constructed by summing the two modulated carriers together, where the two dimensional cartesian components $x=r.\cos\theta$ and $y=r\sin\theta$, represent the modulating information. In this case, the x-carrier is $\cos(2.\Pi.f.t)$ while the y-carrier is $\sin(2.\Pi.f.t)$.

(II) However, if we interpret the same physical signal as the negative of the imaginary part of $r.e^{-j.(2.\Pi.f.t-\theta)}$, which is $r.\sin(2.\Pi.f.t-\theta)$, it will also be constructed by adding the

two modulated carriers together. But this time, by contrast, the x-carrier is $\sin(2.\Pi.f.t)$ while the y-carrier is $-\cos(2.\Pi.f.t)$.

Thus once having chosen the x-carrier, the y-carrier is determined by delaying the x-carrier by 90° as can be verified. We may prefer the x-carrier to be $\cos 2.\Pi.f.t$.

In a direct relation to the information $x=\cos\theta$ and $y = \sin\theta$ mentioned above, let us also consider the following homogeneous transformation matrix:

$$\mathbf{A} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 & 0 \\ \sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{Eq.2.3.2}$$

The orientation submatrix **ROT** embedded in this matrix describes in three dimensions a rotation of an orthonormal coordinate system about the z-axis by an angle θ. Therefore, given a three-dimensional vector **v**, the transformation **ROT.v** will not affect the z-component of that vector in any way. Thus, we consider this as a two-dimensional problem which may conveniently be made related to the complex quantity s mentioned above as far as the definition of the carriers is concerned. These carriers are required in the time signal representation of Eq.2.3.2.

Let us briefly examine this "convenience".

If we consider a cosinewave cos((2.Π.f.t) with unit magnitude (r=1) and delay it by an angle θ to obtain cos(2.Π.f.t-θ), the first two column elements n_x and n_y of the n-vector embedded inthe A-matrix of Eq.2.3.2 will be observable at 2.Π.f.t=0° and 2.Π.f.t=90°, respectively.

In the reverse order, assume that the two elements n_x=cos(θ) and n_y=sin(θ) are given. Now, let n_x modulate the x-carrier cos(2.Π.f.t) and n_y the y-carrier sin(2.Π.f.t) the phase-shifted cosinewave c(t) is, indeed, formed by adding both modulated carriers together; let us repeat this for later reference:

$$c(t) = \cos(\theta).\cos(2.\Pi.f.t) + \sin(\theta).\sin(2.\Pi.f.t) \qquad \text{Eq.2.3.3}$$

The orientation vector of Eq.2.3.2 is then implemented by delaying c(t) by 90°, as it is perpendicular to the normal vector as can be verified.

A matrix of the form

$$A = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 & a.\cos(\theta) \\ \sin(\theta) & \cos(\theta) & 0 & a.\sin(\theta) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{Eq.2.3.4}$$

is then implemented by the extra phase - shifted signal $a.\cos(2.\Pi.f.t.-\theta)$

Given the modulating components n_x and n_y, the production of the signal c(t) in accordance with Eq.2.3.3 is the function of the vector combiner.

c(t) may then be phase-shifted to produce $c^1(t)$ to simulate the rotation function of the A-matrix. The reverse process of recovering the new x/y components from the phase-shifted signal $c^1(t)$, is the function of the vector splitter.

Both the vector combiner and the vector splitter are presented in Appendix 2.

The multiplication of matrices that have the form of Eq. 2.3.4 can then be implemented by successively delaying the relevant sinewave signals. A description on this multiplication technique, presented down to the matrix elements level, is given in Appendix 3.

The inversion of Eq. 2.3.4 is, for the rotation part **ROT**, performed simply by introducing the phase shifts in the reverse order. In particular, the signal $\cos(2.\Pi.f.t +\theta)$ would represent the normal vector embedded in the inverse matrix $\mathbf{A}^{-1}$ of Eq.2.3.4 as can be verified.

The effect of the inversion on the **TRL** part of the matrix representation may be explained by first considering the function blocks $^{i-1}\mathbf{ROT}_i$, $(^{i-1}\mathbf{ROT}_i)^{-1}$ and $^{i-1}\mathbf{TRL}_i$, of Fig.2.2.1 as phase shifters that introduce the required phase angle to the incoming sinewaves represented by the relevant "direct" and "inverse" terms, $\mathbf{TRL}_{nD}$ and $\mathbf{TRL}_{nI}$, respectively, in the figure.

Let us follow the reference signal Ref = a.cos2.Π.f.t. at the input of $^{i-1}\mathbf{TRL}_i$. The output of $^{i-1}\mathbf{TRL}_i$ would provide a.cos(2.Π.f.t. -θ) which is passed through the 180° block to yield - a.cos(2.Π.f.t -θ). Let us assume that the inverse path is being fed with signals representing the unit matrix such that the same signal would appear at the relevant input of $(^{i-1}\mathbf{ROT}_i)^{-1}$. This then introduces a phase angle in the reverse direction to - a.cos(2Π.f.t -θ) so that the result appearing at the $^{i}\mathbf{TRL}_{nI}$ output would be simply -a.cos2.Π.f.t.

This signal -a.cos2.Π.f.t. represents, indeed, the position part of the position vector embedded in the inverse matrix of Eq. 2.3.4.

To verify, let us write the inverse of Eq. 2.3.4 as follows:

$$A^{-1} = \begin{bmatrix} 1 & 0 & 0 & -a \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 & 0 \\ -\sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{Eq.2.3.5}$$

Its position vector $[-a,o,o1]^T$ is, in the sinewave representation, expressed as $-a.\cos 2.\Pi.f.t.$

Fig 2.2.1 and Fig.2.2.2 of section 2.2 may now be physically realized by essentially employing cascaded phase shifters. The direct and inverse equations Eq.2.2.7 to Eq.2.2.10 are then accommodated by the manipulation of sinewave signals representing the input conditions ($^{n}\mathbf{TRL}_{nD}$ and $^{n}\mathbf{ROT}_{nD}$ for the direct path, $^{o}\mathbf{TRL}_{nI}$ and $^{o}\mathbf{ROT}_{nI}$ for the inverse path) fed to the relevant inputs of the phase shift system.

The two-dimensional Robot Joint Descriptor (RJD) as presented in Fig.2.2.1 implements both the A-matrix and its inverse. For some applications where the matrix inverse is not required, the complete RJD will be reduced to having only the direct path.

It should be said that in this presentation, only functions such as **ROT** and **TRL** representing the A and T-matrices are expressed, but not those parameters, such as the angle θ, that control these functions. We emphasize that using the RJDs of Fig. 2.2.1 and Fig.2.2.2, with the relevant, phase-shifted sinewave implemented no conventional arithmetic expressions are involved in arriving at the results of either direct or inverse calculation.

2.4 SUMMARY

The sub-partitioning representation of the general homogeneous transformation matrix

$$\mathbf{T} = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_Y & o_Y & a_Y & p_Y \\ n_Z & o_Z & a_Z & p_Z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{ROT} & \mathbf{TRL} \\ 0 & 1 \end{bmatrix}$$

with **ROT** being of dimension (3x3) and orthonormal, has had the consequence of the intermediate results of the matrix product ${}^{o}\mathbf{T}_{nD} = \mathbf{A}_1 \mathbf{A}_2 \ldots \mathbf{A}_{n-1} \mathbf{A}_n$

where the A-matrices are known as D/H matrices and subscript "D" denotes "direct" manipulation being expressed in the partitioning representation as

$${}^{i-1}\mathbf{ROT}_{nD} = {}^{i-1}\mathbf{ROT}_{i} \,.\, {}^{i}\mathbf{ROT}_{nD} \qquad \text{Eq.2.2.7}$$

$${}^{i-1}\mathbf{TRL}_{nD} = {}^{i-1}\mathbf{ROT}_{i} \,.\, {}^{i}\mathbf{TRL}_{nD} + {}^{i-1}\mathbf{TRL}_{i} \qquad \text{Eq.2.2.8}$$

for i = n,n-1,...,2,1

The intermediate results of the "inverse" manipulation

$${}^{n}\mathbf{T}_{nI} = \mathbf{A}^{-1}{}_{n} \; \mathbf{A}^{-1}{}_{n-1} \ldots \mathbf{A}^{-1}{}_{2} \; \mathbf{A}^{-1}{}_{1} \,.\, {}^{o}\mathbf{T}_{nI}$$

(where ${}^{o}\mathbf{T}_{nI}$ is pre-specified and may have any value) are:

$${}^{i}\mathbf{ROT}_{nI} = ({}^{i-1}\mathbf{ROT}_{i})^{-1} \; {}^{i-1}\mathbf{ROT}_{nI} \qquad \text{Eq.2.2.9}$$

$$^{i}\mathbf{TRL}_{nI} = (^{i-1}\mathbf{ROT}_{i})^{-1}.[^{i-1}\mathbf{TRL}_{nI} - {}^{i-1}\mathbf{TRL}_{nI}] \qquad \text{Eq.2.2.10}$$

for i = 1, 2, ..., n-1, n

The index convention of Eq. 2.1.2 ($^{i}\mathbf{T}_{i}$ is unit matrix) has given rise to the resultant matrix $^{n}\mathbf{T}_{nI}$ being called the error matrix $^{n}\mathbf{E}_{n}$ because, in general, it differs from the unit matrix.

Eq.2.2.7 to Eq.2.2.10 constitute the Robot Joint Descriptor (RJD) symbolically shown in Fig.2.2.1 which has a fundamental importance in the new matrix manipulation method being developed.

Indeed, in the next step we have been interested in finding the result of either direct and inverse calculation using some innovative simulation technique rather than employing conventional arithmetic expressions.

In this sense, we have realized that the projection on the xy-plane of the normal vector

$$\mathbf{n} = n_x.\mathbf{e}_x + n_y.\ \mathbf{e}_y + n_z.\ \mathbf{e}_z$$

embedded in the A-matrices, where $\mathbf{e}_x$, $\mathbf{e}_y$ and $\mathbf{e}_z$ are the three unit vectors of the principal axes of the relevant right-

handed orthonormal reference coordinate system, may be expressed using time-signal as $n_x.e_x(t) + n_y.e_y(t)$

with $e_x(t)$ and $e_y(t)$ being a set of two orthonormal signals.

However, we have been tempted to use sinusoidal signals of the same frequency for $e_x(t)$ and $e_y(t)$ because the information about n_x and n_y carried in the resultant sinusoidal signal can be manipulated simply by phase shifting the resultant signal.

The projections on the xy-plane of **n** and **o** have then been implemented using phase shifted sinewaves of unit magnitude withthe relative phase difference of 90°. Due to the Denavit/Hartenberg conventions (D/H), the projection on the x/y-plane of the position vector **p** has the same direction with that of the normal vector. Therefore, it has been implemented using another sinewavce signal which is in phase with that representing the projection of the normal vector, however, of magnitude corresponding to the D/H parameter "a".

As can be seen, one limitation of the sinewaves is that they carry only two pieces informations. That is, either the quantities (x,y) of the vector under manipulation in the representation of Cartesian coordinates or, equivalently, the quantities (r,θ) in the polar coordinate representation. Because of this, we have been restricted to two dimensions and haven't been able yet to consider the z-components.

However, we have arrived at the knowledge that

1) The use of time signals to implement the matrices is essential for the new concept, as then their manipulation will not involve any conventional arithmetic calculations.

2) Using the cosine-signal and the sine-signal of the same frequency as a trivial set of orthornormal references (or carriers) for the implementation, the relationship between the sinewaves representing the A-matrices and those representing their inverses is that the sign of the phase angle accommodated in them is reversed.

3) The basic equations for the manipulation of the T-matrices in both direct and inverse direction are implemented simply by cascading an appropriate number of phase shifter stages and so matrix multiplications are performed without using any conventional arithmetic expressions.

In this light, the two-dimensional RJD has been presented which, however, will have only a limited importance because of the two-dimensional restriction.

Its expansion into three dimensions will be the subject of the next chapter.

Chapter 3

SINUSOIDAL IMPLEMENTATION OF THE THREE-DIMENSIONAL ROBOT JOINT DESCRIPTOR (3D-RJD)

3.1 INTRODUCTION

Consider the general joint matrix:[4,5]

$$\mathbf{A} = \begin{bmatrix} C(\theta) & -S(\theta).C(\alpha) & S(\theta).S(\alpha) & a.C(\theta) \\ S(\theta) & C(\theta).C(\alpha) & -C(\theta).S(\alpha) & a.S(\theta) \\ 0 & S(\alpha) & C(\alpha) & d \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{Eq.3.1.1}$$

where $C(\theta)=\cos(\theta)$ and $S(\theta)=\sin(\theta)$

$C(\alpha)=\cos(\alpha)$ and $S(\alpha)=\sin(\alpha)$

If the joint is revolute, the independent variable of this matrix will be $q=\theta$; however, if the joint is prismatic, the generalized independent variable q will be q=d.

This matrix can be considered as the result of the following matrix multiplication:

$$A=ROT_z(\theta).Trans_z(d)..Trans_x(a).ROT_x(\alpha) \qquad \text{Eq.3.1.1b}$$

where

$$ROT_z(\theta) = \begin{bmatrix} C(\theta) & -S(\theta) & 0 & 0 \\ S(\theta) & C(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{Eq.3.1.2}$$

$$ROT_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) & 0 \\ 0 & \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{Eq.3.1.3}$$

The basic homogeneous translation matrices are:

$$Trans_x(a) = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{Eq.3.1.4}$$

$$Trans_z(d) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{Eq.3.1.5}$$

As is seen, Eq.3.1.2 - Eq.3.1.5 represents elementary transformations and so can be realized by elementary one or two-dimensional operations.

The breaking of the A-matrix into these four basic transformations represents a crucial step towards the development of the three-dimensional RJD. That is, judging from Eq.3.1.1b, we realize that the expansion of the sinewave concept into three dimensions is possible merely because the

three-dimensional RJD will be, in principle, nothing else than a combination of two-dimensional RJDs.

3.2 PRELIMINARY

For ease of explanation, we will, without loss of generality, refer to the 5-axis Mitsubishi robot arm RM-501 (see Fig.3.2.1.1) which has all features of a typical industrial robot. For reference purposes, we will establish the A and T-matrices for this robot arm. However, let us first recall the Denavit/Hartenberg rules.

3.2.1 The Denavit/Hartenberg Rules

The Denavit/Hartenberg rules for establishing robot joint's coordinate frames are well known and can be found in a number of literature [1,4,5].

For ease of reference, they are now stated in the following.

Every coordinate frame is determined and established on the basis of three rules:

a) The z_{i-1} axis denoting the z-axis of the (i-1)th coordinate frame lies along the axis of motion of the i-th joint.

b) The x_i axis denoting the x-axis of the i-th coordinate frame is normal to the z_{i-1} axis, pointing away from it.

c) The y_i axis denoting the y-axis of the i-th coordinate frame completes the right-handed coordinate system.

The four geometric quantities associated with each link are defined as follows:

θ_i : The joint angle from the x_{i-1} axis to the x_i axis about the z_{i-1} axis (right-handed rule).

d_i : The distance along the z_{i-1} axis from the origin of the (i-1)th frame to the intersection of the z_{i-1} axis with the x_i axis.

a_i : The shortest distance between the z_{i-1} and z_i axes.

α_i : The offset angle from the z_{i-1} axis to the z_i axes about the x_i axis (right-handed rule).

The RM-501 robot arm has only revolute joints, hence d_i and a_i remain constant.

The "twist angles" α_i are constant. This applies not only for this particular robot arm but also for most commercially available robots.

For the RM-501, the remaining geometric quantity θ_i is the joint variable of the i-th coordinate frame that changes when link i rotates about the z_{i-1} axis.

Fig. 3.2.1.1 illustrates conventions of joint, link numbers and all the robot's geometry parameters.

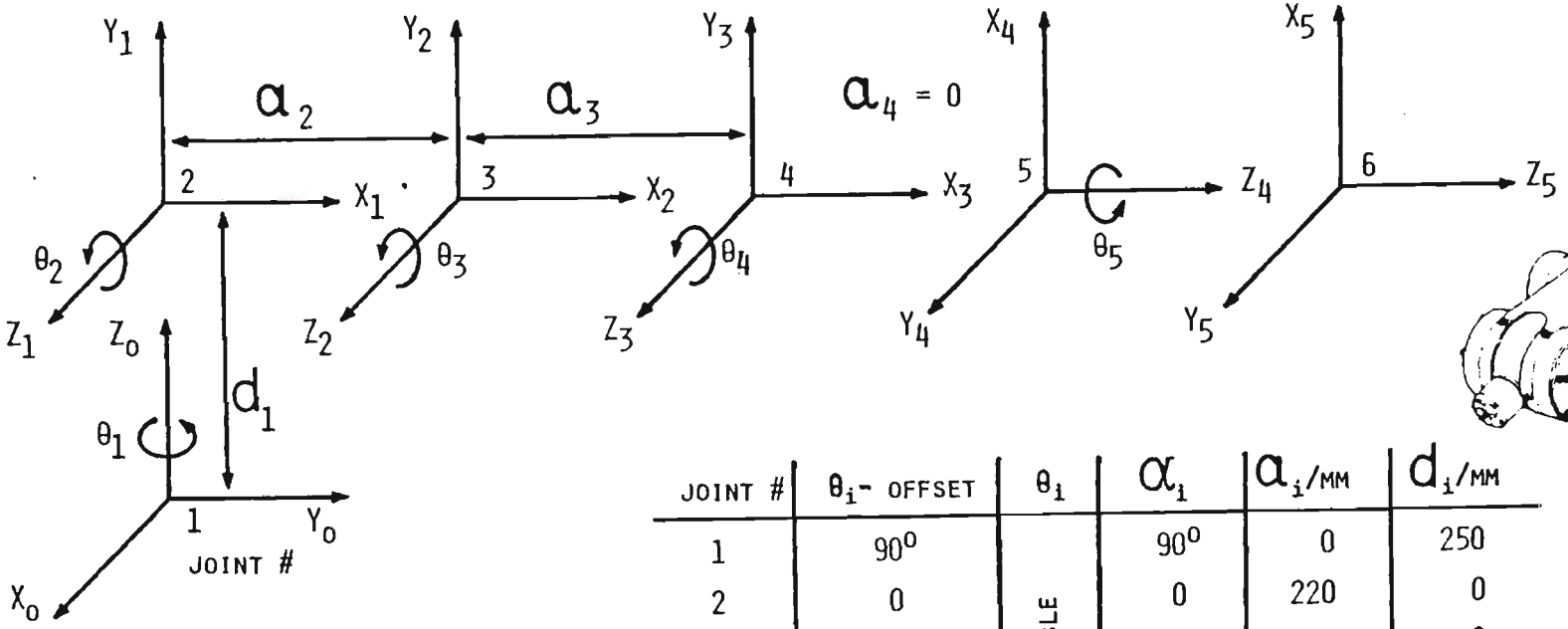
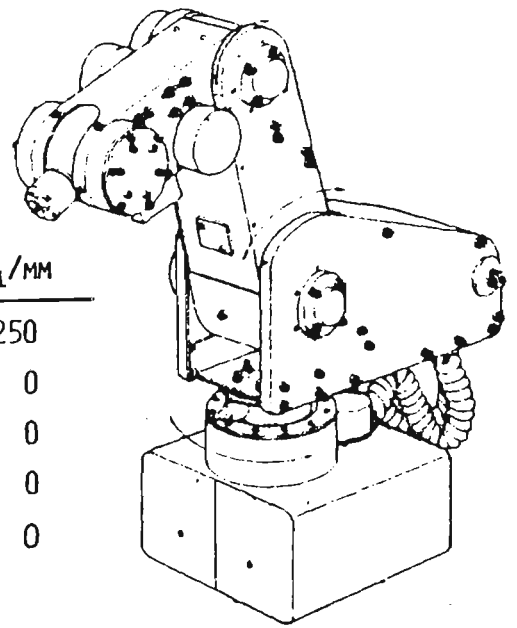


Fig. 3.2.1.1

JOINT #	θ_i - OFFSET	θ_i	α_i	$a_{i/MM}$	$d_{i/MM}$
1	90^0	VARIABLE	90^0	0	250
2	0		0	220	0
3	0		0	160	0
4	90^0		90^0	0	0
5	0		0	0	0



3.2.2 The A and T-Matrices of the RM-501.

Once having established the coordinate frames, the A-matrices are written straightforwardly as follows:

$$ {}^{0}\mathbf{A}_1 = \begin{bmatrix} -S_1 & 0 & C_1 & 0 \\ C_1 & 0 & S_1 & 0 \\ 0 & 1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{Eq.3.2.2.1.a} $$

$$ {}^{1}\mathbf{A}_2 = \begin{bmatrix} C_2 & -S_2 & 0 & a_2.C_2 \\ S_2 & C_2 & 0 & a_2.S_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{Eq.3.2.2.1.b} $$

$$ {}^{2}\mathbf{A}_3 = \begin{bmatrix} C_3 & -S_3 & 0 & a_3.C_3 \\ S_3 & C_3 & 0 & a_3.S_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{Eq.3.2.2.1.c} $$

$$ {}^{3}\mathbf{A}_4 = \begin{bmatrix} -S_4 & 0 & C_4 & 0 \\ C_4 & 0 & S_4 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \qquad \text{Eq.3.2.2.1.d} $$

$$ {}^{4}\mathbf{A}_5 = \begin{bmatrix} C_5 & -S_5 & 0 & 0 \\ S_5 & C_5 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{Eq.3.2.2.1.e} $$

$$ {}^{5}\mathbf{E}_g = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{Eq.3.2.2.1.f} $$

where $C_i = \cos(\theta_i)$; $s_i = \sin(\theta_i)$ and the transform ${}^{5}\mathbf{E}_g$ represents the "tool transform".

The Denavit/Hartenberg rules do not specify the A-matrices uniquely. For example, we have been free to arbitrarily choose the direction of the X_0-axis and so with a different choice, we could have arrived at a somewhat different matrix ${}^{0}\mathbf{A}_1$. In this same sense, the 5-th coordinate system could have been defined differently.

An important attribute of these matrices is that there is only a "one-element" in the third components of the orientation sub-matrices, and therefore, every one A-matrix as stated above describes a two-dimensional rotating/ translating transformation on the local z-plane. In some

cases where the geometric quantity "d" is the joint variable rather than the quantity "θ", the A-matrix describing a particular coordinate frame will merely be considered as a one-dimensional translation of that particular coordinate frame along the adjacent local z-axis. We shall make use of this, considering that three-dimensional problems can be treated as a combination of one- and/or two- dimensional problems.

In the following, the T-matrices are established for future reference:

$$ {}^{5}\mathbf{T}_{5} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{Eq.3.2.2.2.a} $$

$$ {}^{4}\mathbf{T}_{5} = \begin{bmatrix} C_5 & -S_5 & 0 & 0 \\ S_5 & C_5 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \qquad \text{Eq.3.2.2.2.b} $$

$$
{}^{3}\mathbf{T}_5 = \begin{bmatrix} -S_4.C_5 & S_4.S_5 & C_4 & 0 \\ C_4.C_5 & -C_4.S_5 & S_4 & 0 \\ S_5 & C_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{Eq.3.2.2.2.c}
$$

$$
{}^{2}\mathbf{T}_5 = \begin{bmatrix} -S_{34}.C_5 & S_{34}.S_5 & C_{34} & a_3.C_3 \\ C_{34}.C_5 & -C_{34}.S_5 & S_{34} & a_3.S_3 \\ S_5 & C_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{Eq.3.2.2.2.d}
$$

$$
{}^{1}\mathbf{T}_5 = \begin{bmatrix} -S_{234}.C_5 & S_{234}.S_5 & C_{234} & a_3.C_{23} + a_.{}_2.C_2 \\ C_{234}.C_5 & -C_{234}.S_5 & S_{324} & a_3.S_{23} + a_2.S_2 \\ S_5 & C_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

Eq.3.2.2.2.e

$$
{}^{0}\mathbf{T}_5 = \begin{bmatrix} S_1.S_{234}.C_5+C_1.S_5 & -S_1.S_{234}.S_5+C_1.C_5 & -S_1.C_{234} & -S_1.(a_3.C_{23} + a_2.C_2) \\ -C_1.S_{234}.C_5+S_1.S_5 & C_1.S_{234}.S_5+S_1.C_5 & C_1.C_{234} & C_1.(a_3.C_{23} + a_2.C_2) \\ C_{234}.C_5 & -C_{234}.S_5 & S_{234} & a_3.S_{23} + a_2.S_2 + d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{Eq.3.2.2.2.f}
$$

where

$C_{ij} = \cos(\theta_i + \theta_j)$;

$S_{ij} = \sin(\theta_i + \theta_j)$;

$C_{ijk} = \cos(\theta_i + \theta_j + \theta_k)$;

$S_{ijk} = \sin(\theta_i + \theta_j + \theta_k)$;

3.3 INTERPRETATION OF THE T-MATRICES USING SINGLE-FREQUENCY SINEWAVES IN THREE DIMENSIONS

We refer to the T-matrices established under the previous section without loss of generality.

Fig. 3.3.1 illustrates the geometric relationships between the two coordinate frames 4 and 5 of the RM-501 robot arm, in which we denote the unitvectors on the X, Y and Z axis of the i-th frame with X_i, Y_i and Z_i, respectively.

The Fig. 3.3.1 shows that a rotation of the 5th frame by θ_5 about Z_4 does not affect Z_5, but creates four new component-vectors expressed in the 4th frame namely:

$+C_5.X_4$ and $+S_5.Y_4$ as results of the rotation of the vector X_5

$-S_5.X_4$ and $+C_5.Y_4$ as results of the rotation of the vector Y_5.

In the next step we leave θ_5 unchanged and rotate the 4th coordinate frame by θ_4 about Z_3.

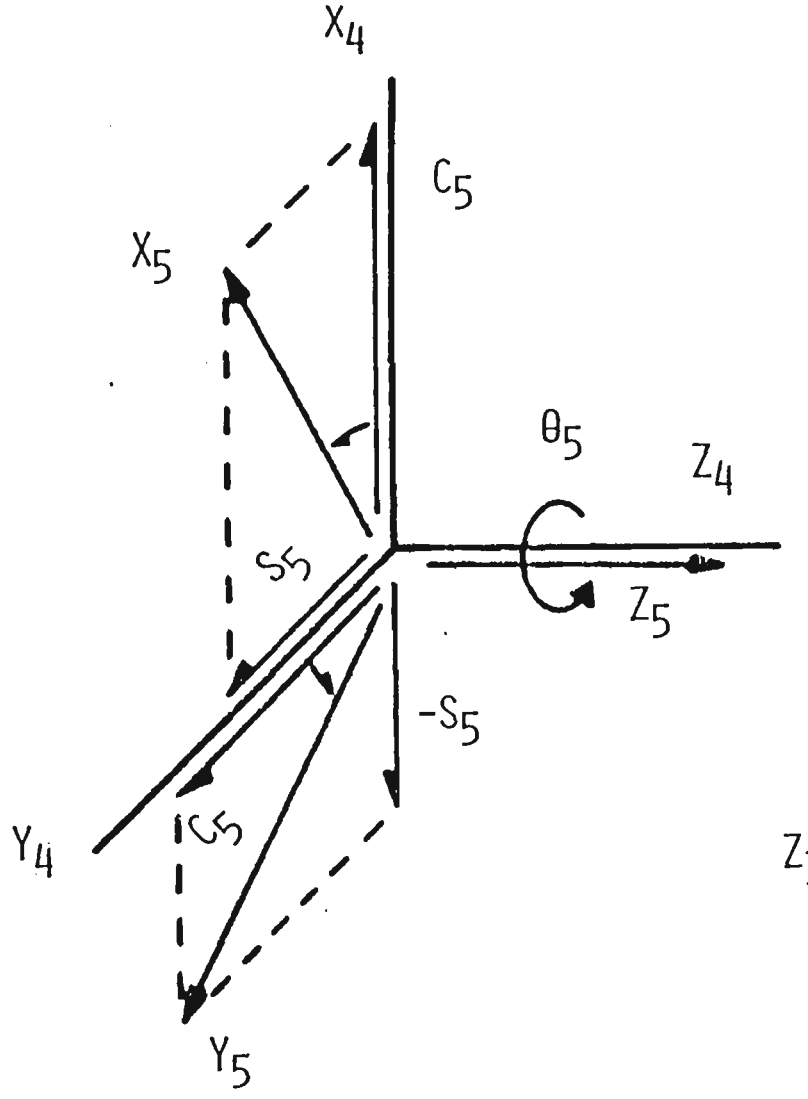


Fig. 3.3.1

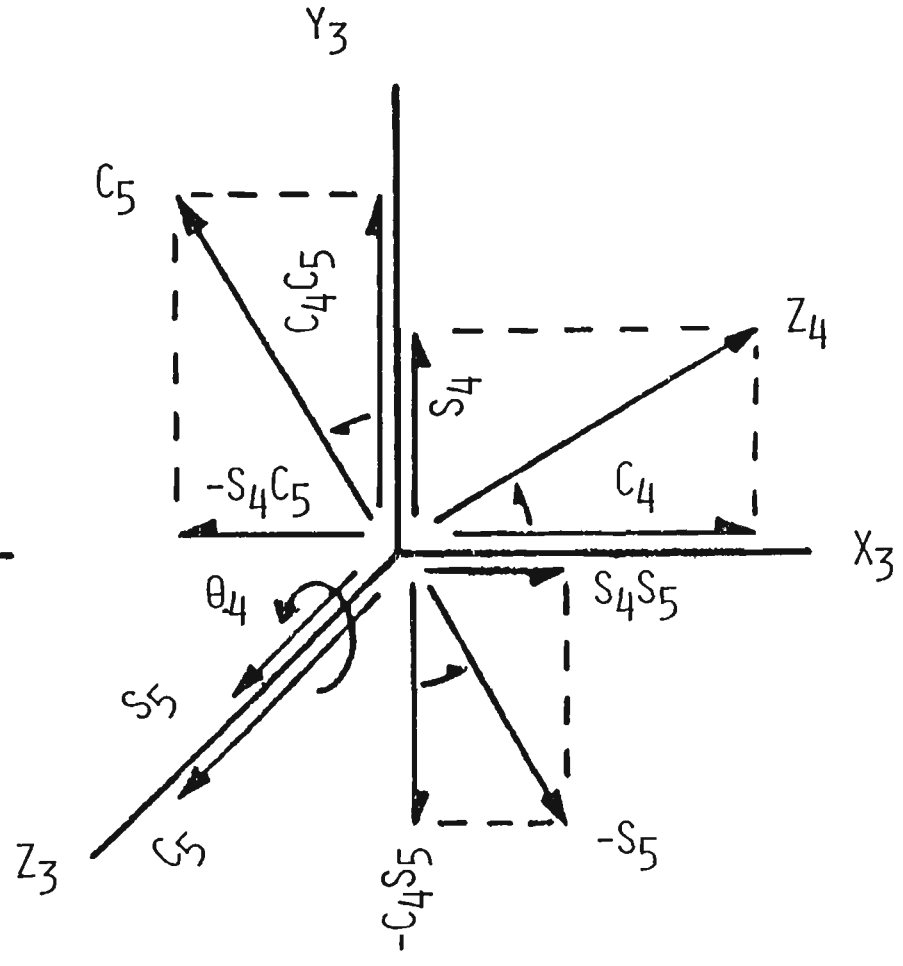


Fig. 3.3.2

Because Y_4 is parallel to Z_3, all the component-vectors of the Y_4 axis remain unchanged upon this rotation. However, the three vectors Z_4, $C_5.X_4$ and $-S_5.X_4$ create in the 3rd coordinate system two new component-vectors each. See Fig.3.3.2.

The first mechanisms behind these rotating operations become now obvious.

3.3.1. The Basic Statements.

The following statements apply only for matrix configuration of robot manipulators whose link parameters α_i are either 0^0, $\pm 90^0$ or $\pm 180^0$. Fortunately, almost all commercial robot manipulators do have this geometric configuration.

This condition will be removed later as shown in Sec 3.5.

Statement 1.

The rotation of a link about its rotation or Z axis causes change in only X and Y components, and so these changes can be presented by a single appropriately phased sinewave as has been shown under the previous chapter.

Statement 2.

The four vectors embedded in the A-matrix describing a coordinate system in general have three components each with respect to a new reference system.

Statement 3.

Because the angle α_i, which indicates a pre-rotation of the i-th frame about its X_i axis by that angle α_i, is a constant, we can take it into account simply by first renaming the component vectors before rotating them. For example:

in System 4	re-named to	in System 3
$n_x.X_4$		$n_y.Y_3$
$n_y.Y_4$		$n_z.Z_3$
$n_z.Z_4$		$n_x.X_3$

Statement 4.

The renamed Z components remain unchanged upon the following rotation.

Statement 5.

A rotation affects the renamed X and Y components by creating two new components each in the X/Y plane, with magnitude determined according to the sine and cosine law of the rotation angle.

Statement 6.

The X components of the same vector must be added together if they have been created by different rotations. This also

applies for the Y components of that same vector. These then modulate the X and Y references to create a suitable sinewave, which is then phase shifted by the rotation angle θ to give the new components. By translating these verbal statements into graphical form using single frequency sinewave signals, we obtain Fig. 3.3.1.1 and Fig. 3.3.1.1.b, where we choose the x-carrier to be $\cos(2.\Pi.f.t)$ and the y-carrier $\sin(2.\Pi.f.t)$.

An alternative graphical representation is presented in Appendix 4.

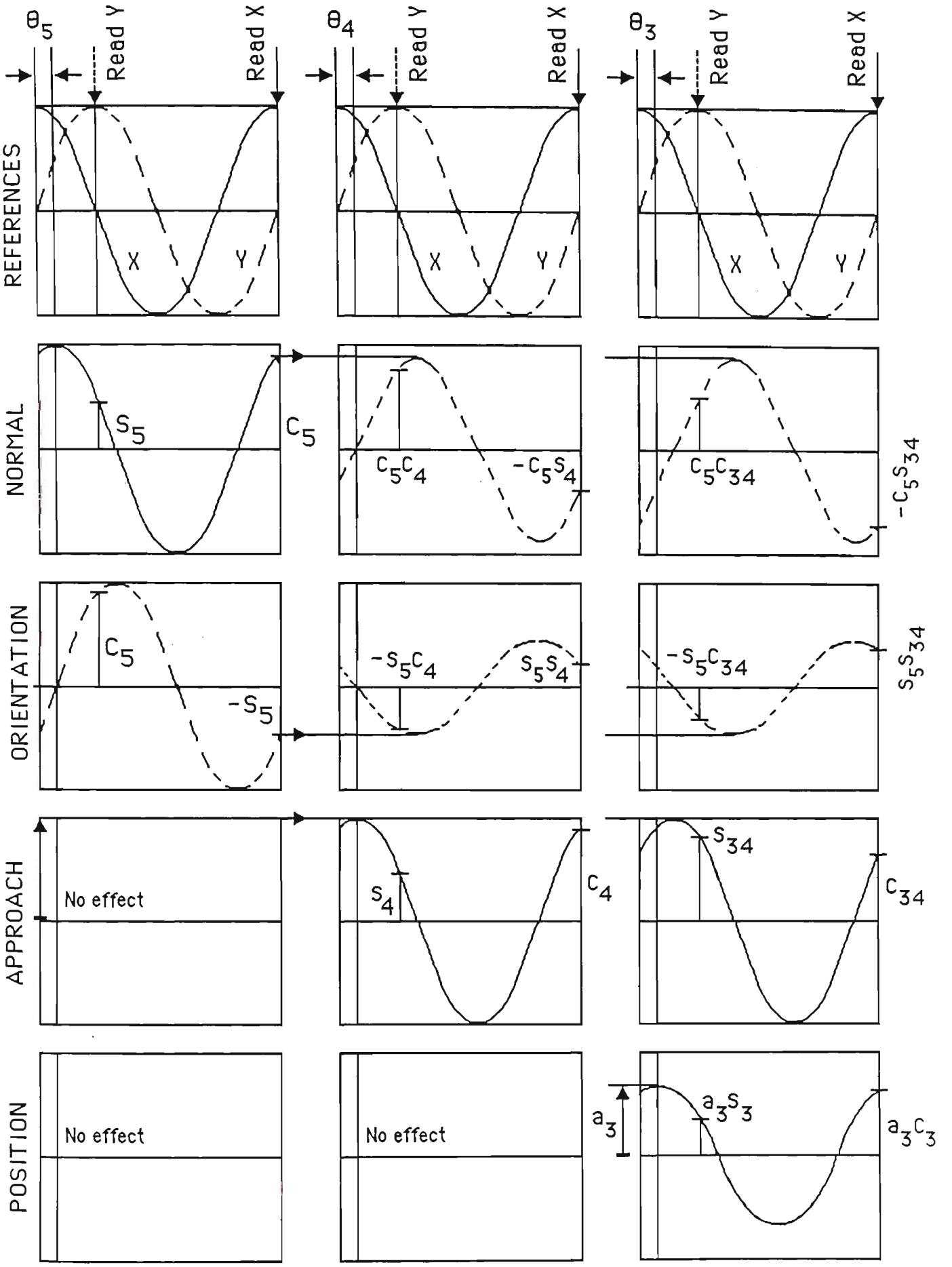

θ5
Read Y
Read X
θ4
Read Y
Read X
θ3
Read Y
Read X
REFERENCES
X
Y
X
Y
X
Y
NORMAL
S5
C5
C5C4
-C5S4
C5C34
-C5S34
ORIENTATION
C5
-S5
-S5C4
S5S4
-S5C34
S5S34
APPROACH
No effect
S4
C4
S34
C34
POSITION
No effect
No effect
a3
a3S3
a3C3

Fig. 3.3.1.1

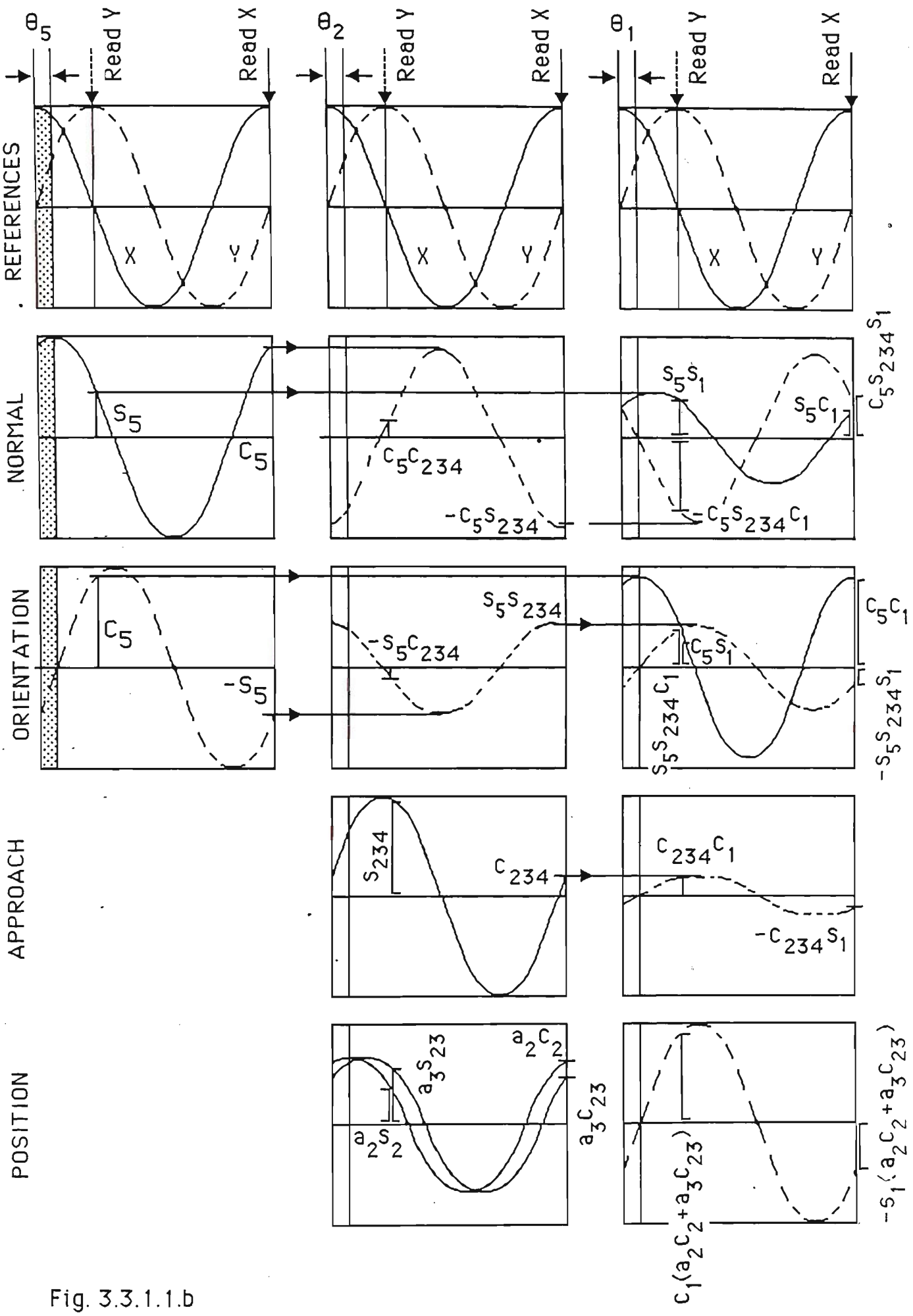

θ5
θ2
θ1
Read Y
Read X
REFERENCES
X
Y
NORMAL
ORIENTATION
APPROACH
POSITION
S5
C5
-S5
C5C234
-C5S234
S5S1
S5C1
C5S234S1
-C5S234C1
S5S234
-S5C234
C5C1
C5S1
S5S234C1
-S5S234S1
S234
C234
C234C1
-C234S1
a3S23
a2C2
a2S2
a3C23
-S1(a2C2+a3C23)
C1(a2C2+a3C23)

Fig. 3.3.1.1.b

3.3.2 Sinewave Implementation and Interpretation

The meanings of the phase-shifted sinewaves can be interpreted in some brief words in the following.

Let us follow the signals representing (X_5, Y_5, Z_5, P_5) where, however, only X_5 andy Y_5 are passed through the first stage of the two dimensional phase shift system, for Z_5 and P_5 are not affected by the first rotation and therefore must be kept unchanged.

Phase-shifting X_5, the so-obtained signal will provide the two components $(+C_5, +S_5)$.

Phase-shifting Y_5, the so-obtained signal yields the two components $(-S_5, +C_5)$.

In matrix form, we have:

$$\begin{bmatrix} C_5 & -S_5 & 0 & 0 \\ S_5 & C_5 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

This is the matrix ${}^4\mathbf{T}_5$, where for completeness we have added the fourth row vector $(0,0,0,1)$.

Before processing the outputs from the first phase shift stage, we have to consider statement 3 that states for this particular case:

X component	re-named to	Y component
Y		Z
Z		X

The so re-named X and Y components now modulate the appropriate carriers. Recalling, that in the figures 3.3.1.3 and 3.3.1.3.b presented under the previous subsection, we have chosen $\cos(2.\Pi.f.t)$. as the x-carrier and $\sin(2.\Pi.f.t)$. as the y-carrier.

Now, phase-shifting the new modulated sinewaves by θ_4, the corresponding T-matrices are read from the Figs. 3.3.1.3 and 3.3.1.3.b.

$$ {}^{3}\mathbf{T}_5 = \begin{bmatrix} -S_4.C_5 & S_4.S_5 & C_4 & 0 \\ C_4.C_5 & -C_4.S_5 & S_4 & 0 \\ S_5 & C_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

In the next step, we rename the components as follows:

X component	"re-named" to	X component
Y		Y
Z		Z

Using statement 6, the T-matrix is read to

$$
{}^{2}\mathbf{T}_{5} = \begin{bmatrix} -S_{34}.C_5 & S_{34}.S_5 & C_{34} & a_3.C_3 \\ C_{34}.C_5 & -C_{34}.S_5 & S_{34} & a_3.S_3 \\ S_5 & C_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

Similarly, we read from the outputs of the next phase-shifter stage

$$
{}^{1}\mathbf{T}_{5} = \begin{bmatrix} -S_{234}.C_5 & S_{234}.S_5 & C_{234} & a_3.C_{23} + a_2.C_2 \\ C_{234}.C_5 & -C_{234}.S_5 & S_{234} & a_3.S_{23} + a_2.S_2 \\ S_5 & C_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

In the last step, we rename the components to

X component	re-named to	Y component
Y		Z
Z		X

The resultant T-matrix is read to:

$$
{}^{0}\mathbf{T}_5 = \begin{bmatrix}
S_1 \cdot S_{234} \cdot C_5 + C_1 \cdot S_5 & -S_1 \cdot S_{234} \cdot S_5 + C_1 \cdot C_5 & -S_1 \cdot C_{234} & -S_1 \cdot (a_3 \cdot C_{23} + a_2 \cdot C_2) \\
-C_1 \cdot S_{234} \cdot C_5 + S_1 \cdot S_5 & C_1 \cdot S_{234} \cdot S_5 + S_1 \cdot C_5 & C_1 \cdot C_{234} & C_1 \cdot (a_3 \cdot C_{23} + a_2 \cdot C_2) \\
C_{234} \cdot C_5 & -C_{234} \cdot S_5 & S_{234} & a_3 \cdot S_{23} + a_2 \cdot S_2 + d_1 \\
0 & 0 & 0 & 1
\end{bmatrix}
$$

where

$C_{ij} = \cos(\theta_i + \theta_j)$;

$S_{ij} = \sin(\theta_i + \theta_j)$;

$C_{ijk} = \cos(\theta_i + \theta_j + \theta_k)$;

$S_{ijk} = \sin(\theta_i + \theta_j + \theta_k)$;

3.4 DERIVATION OF THE 3D-RJD FOR THE CASE OF COS α=0 OR COS α= + 1

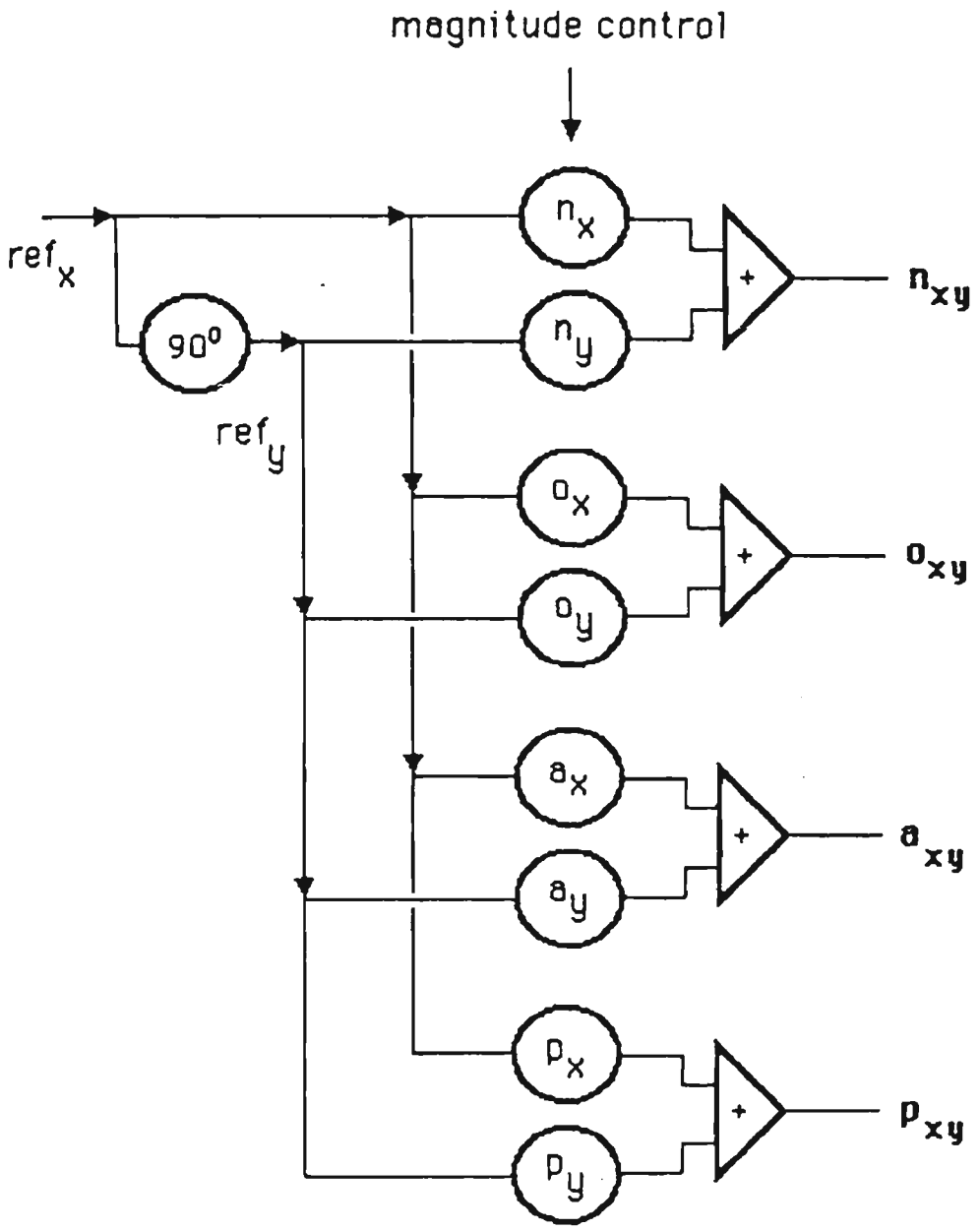
To implement the 3D-RJD, the previous section provides the following three results:

1) We need vector combiners that select and prepare input signals for the phase shifters (Statements 3,4 and 6).

2) Two-dimensional phase shifters are needed (Statement 1).

3) We also need vector splitters that process the output signals from the phase shifters such that ready-to-use signals are available for the next cascade (Statements 2 and 5).

The realisation of the vector combiner (**VC**) and the vector splitter (**VS**) can be found in Appendix 2. The next pages show them symbolically as the particular parts of the vector combiner unit (**VCU**) and the vector splitter unit (**VSU**) required in the implementation of the 3D-RJD.



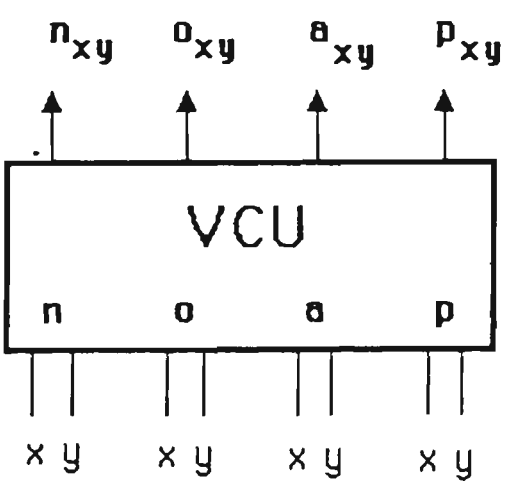


Fig. 3.4.1: The Vector Combiner Unit and its symbol.

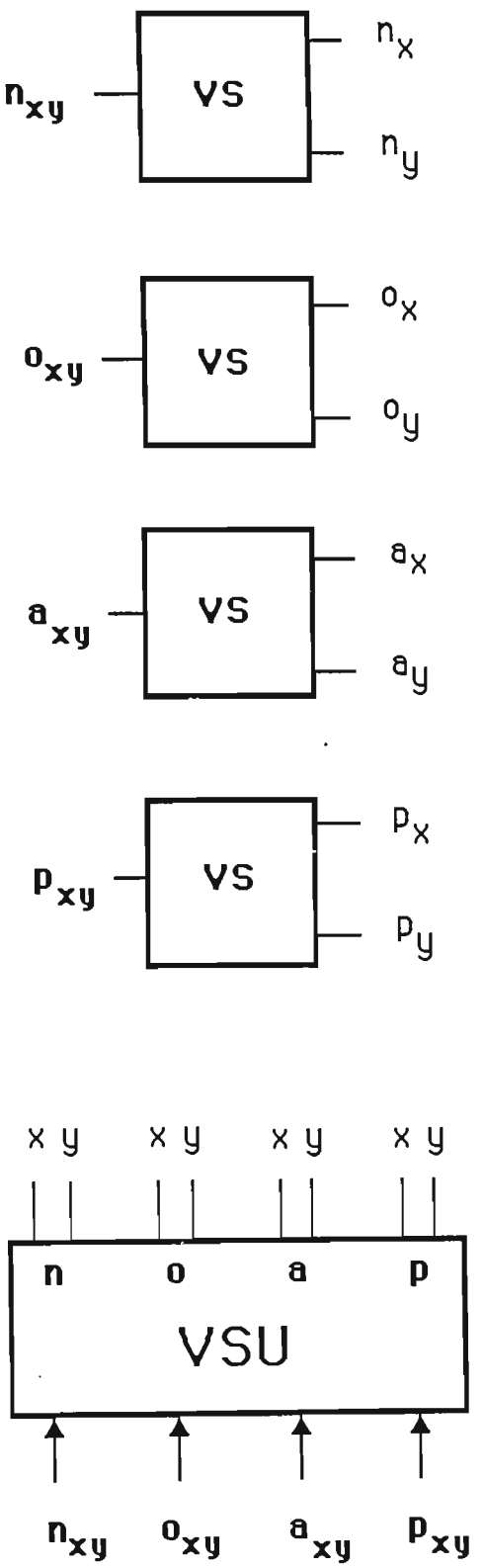


Fig. 3.4.2: The Vector Splitter Unit and its symbol.

Let us consider Fig. 3.4.3. In this diagram, the box representing the three-dimensional translate block indicated by $Cyl(a_i, O_i, d_i)$ looks in essence like that of the two-dimensional case. However, the difference can be seen in the Z output of the box which is led to one new adder on the lefthand side of the figure that takes the z-component of the incoming position vector of the direct manipulation path into account. Recall that all the z-components remain unchanged upon a rotation about the z-axis. The z component of the three-dimensional inverse position vector can then be implemented simply by inverting the Z output of the 3D-translate block which is led to the second new adder on the right hand side of the figure that takes the z-component of the incoming position vector in the reverse manipulation path into account.

For ease of reference, the 3D-translate block $Cyl(a_i, \Theta_i, d_i)$ is shown as a separate block. The remaining is collected in the 180° function block.

The "s" and "t" variables in Fig.3.4.3 have the following meaning:

s_D: The projection of $^{i}\mathbf{TRL}_{nD}$ on the xy-plane.

t_D: The projection of $^{i-1}\mathbf{TRL}_{nD}$ on the xy-plane.

s_I: The projection of $^{i-1}\mathbf{TRL}_{nI}$ on the xy-plane.

t_I: The projection of $^{i}\mathbf{TRL}_{nI}$ on the xy-plane.

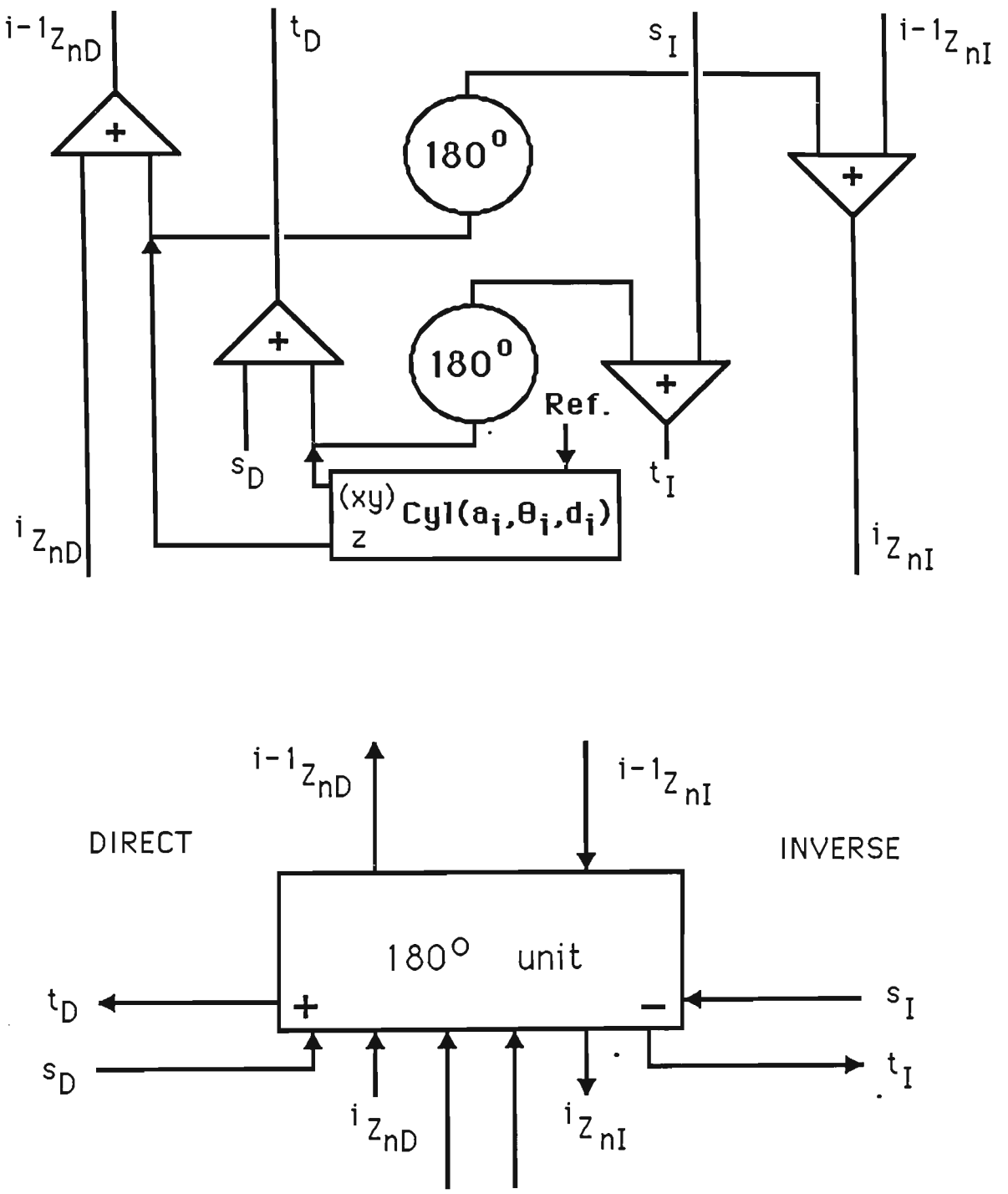


Fig. 3.4.3: The 180° conversion unit in connection with $\mathbf{Cyl(a_i,\theta_i,d_i)}$ and its symbol.

Using the units derived above, we are now in the position to present the three-dimensional Robot Joint Descriptor shown in Fig. 3.4.4.

This diagram represents the general case of the Three-Dimensional Robot Joint Descriptor. However, for the case of a slide joint, the hardware implementation will reduce drastically as shown in Fig. 3.4.5.

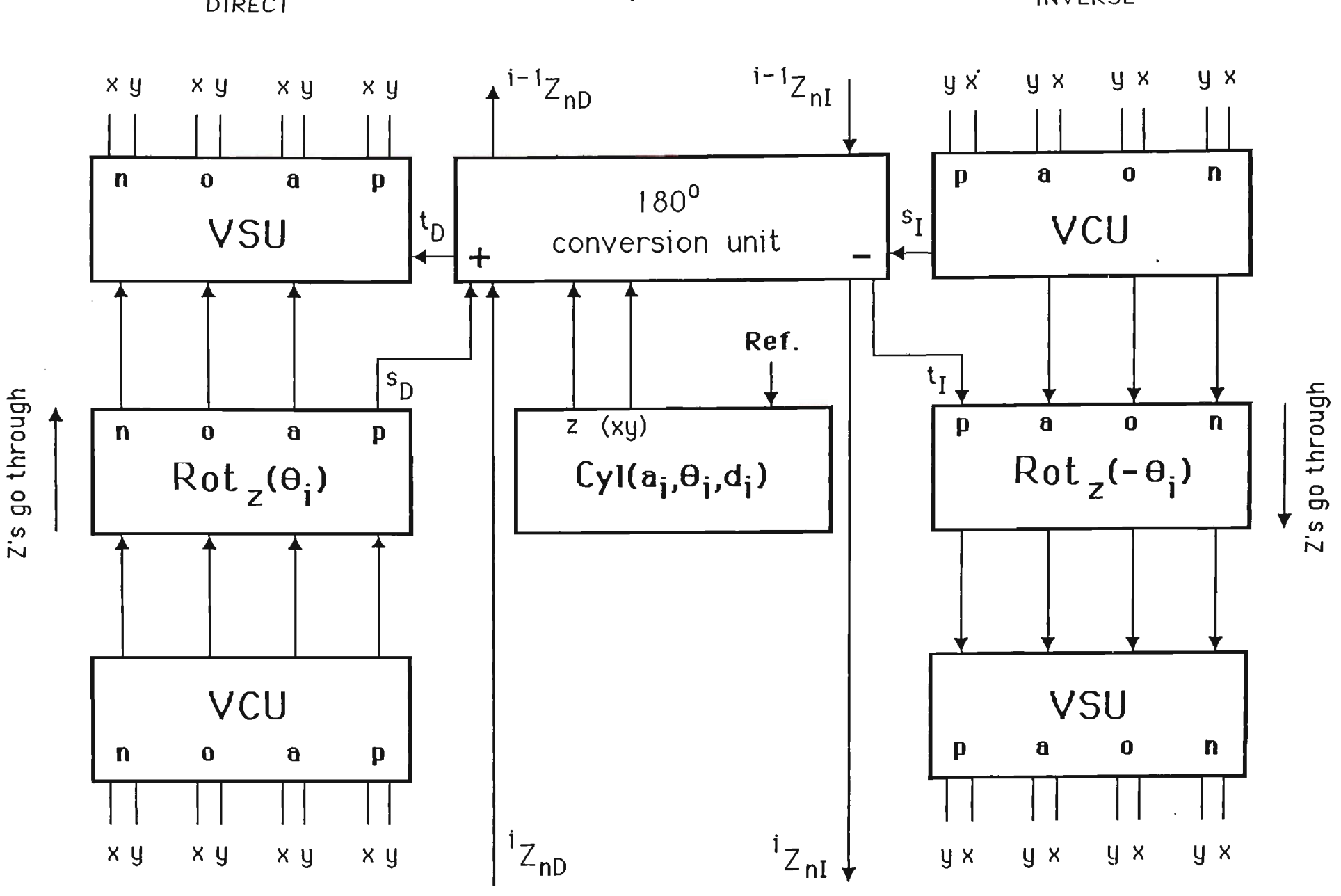


Fig. 3.4.4: The Three-dimensional Robot Joint Descriptor: The i-th Joint.

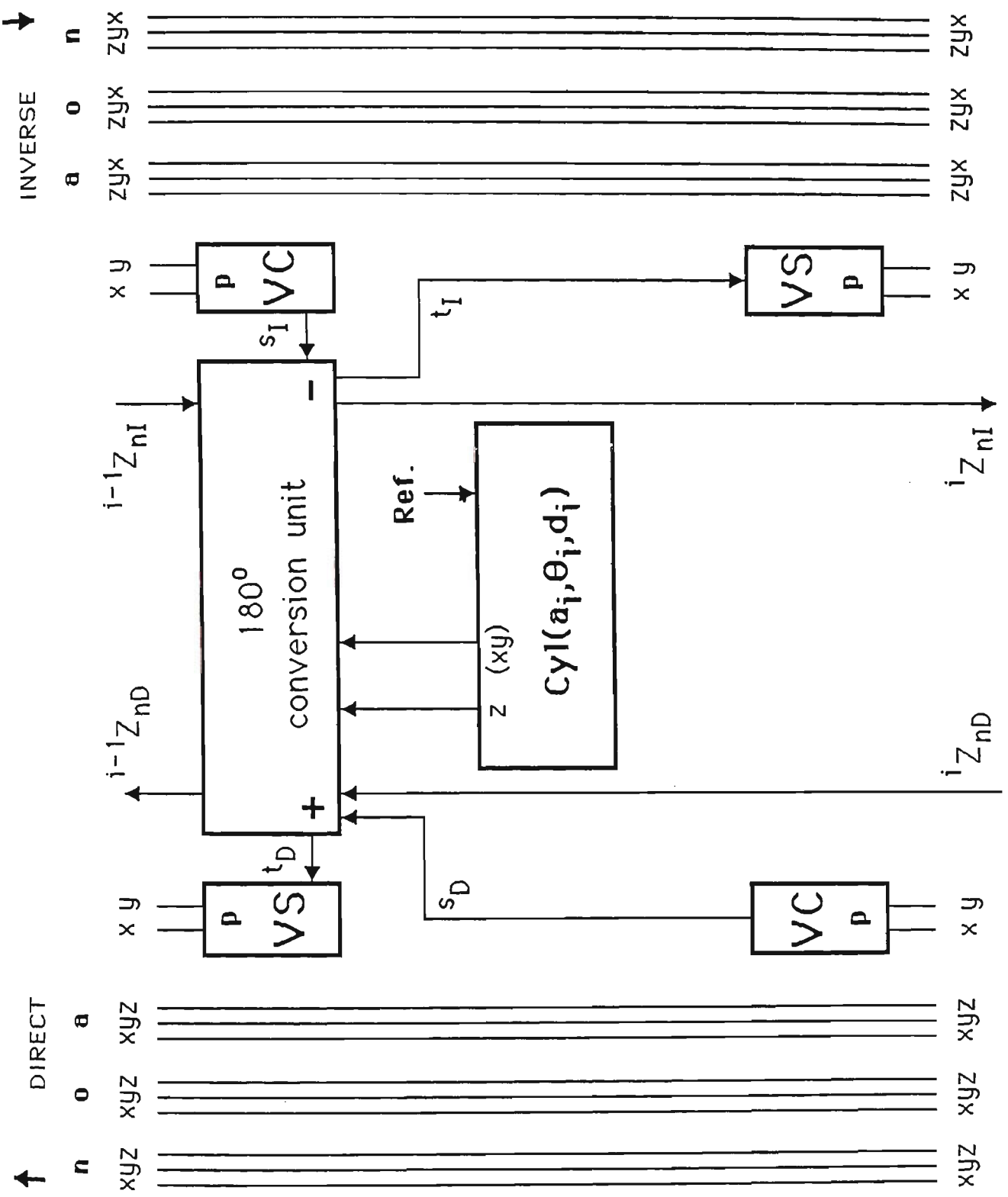


Fig. 3.4.5: Reduction of the 3D-RJD in the case of slide joint.

In the following, some obligatory comments about the 3D-RJD should be made.

In the view of the circuitry designer and analyser, the 3D-RJD is much more complex than the 2D-RJD. However, in the view of the system thinker, the design philosophy still remains the same. In fact, the basic manipulation equations derived under chapter 2

"Direct" equations:

$$^{i-1}\mathbf{ROT}_n = {}^{i-1}\mathbf{ROT}_i \cdot {}^{i}\mathbf{ROT}_n$$

$$^{i-1}\mathbf{TRL}_n = {}^{i-1}\mathbf{ROT}_i \cdot {}^{i}\mathbf{TRL}_n + {}^{i-1}\mathbf{TRL}_i$$

"Inverse" equations:

$$^{i}\mathbf{ROT}_n = (^{i-1}\mathbf{ROT}_i)^{-1} \cdot {}^{i-1}\mathbf{ROT}_n$$

$$^{i}\mathbf{TRL}_n = (^{i-1}\mathbf{ROT}_i)^{-1} \cdot (^{i-1}\mathbf{TRL}_n - {}^{i-1}\mathbf{TRL}_i)$$

hold not only in two dimensions, but also hold in three dimensions.

Coincidentally, in two dimensions we can implement a two-dimensional vector using just a single frequency sinewave that need not be processed further before being passed through a phase shifter. In three dimensions, if we had the possibility to implement three-dimensional vectors using a single signal, the 3D-RJD would look like the same as the 2D-RJD. However, if we temporarily ignore the Vector Combiner Unit and the Vector Splitter Unit, the 2D-features are then obvious from Fig. 3.4.4.

Because of the general validity of the basic manipulation equations, we prefer to use the symbol Robot Joint Descriptor of Fig. 2.2.1 for both cases two-dimensional as well as three-dimensional manipulation.

3.5. IMPLEMENTATION OF THE 3D-RJD FOR THE MORE GENERAL CASE OF $\cos(\alpha) \neq 0$ AND $\cos(\alpha) \neq \mp 1$

In a further step of generalization, the rotation about the x-axis by the angle α can be treated the ordinary way of phase shifting the sinewave signals representing the y and z components (right hand rule) of the vectors comprised in the incoming T-matrix. This is done by employing one more phase shift stage located before the actual delay of the angle θ as required by Eq.3.1.1.b in the case of direct manipulation. Reversely, in the inverse path, this one more phase shift stage is located after the actual delay stage of $-\theta$. This is shown in Fig. 3.5.1 and Fig. 3.5.2 where we should indicate that 1) the new phase shifters are accommodated with constant phase shifts corresponding to α and $-\alpha$ as α itself is a constant, and that 2) the x-components will remain unchanged upon this α- manipulation.

As is obvious, in the case of $\cos \alpha = 0$ or $\cos \alpha = \pm 1$ as applied for most commercially available robots, this one extra phase shifter stage can be by-passed by the process

which we have previously called "re-naming" the components of the relevant vectors.

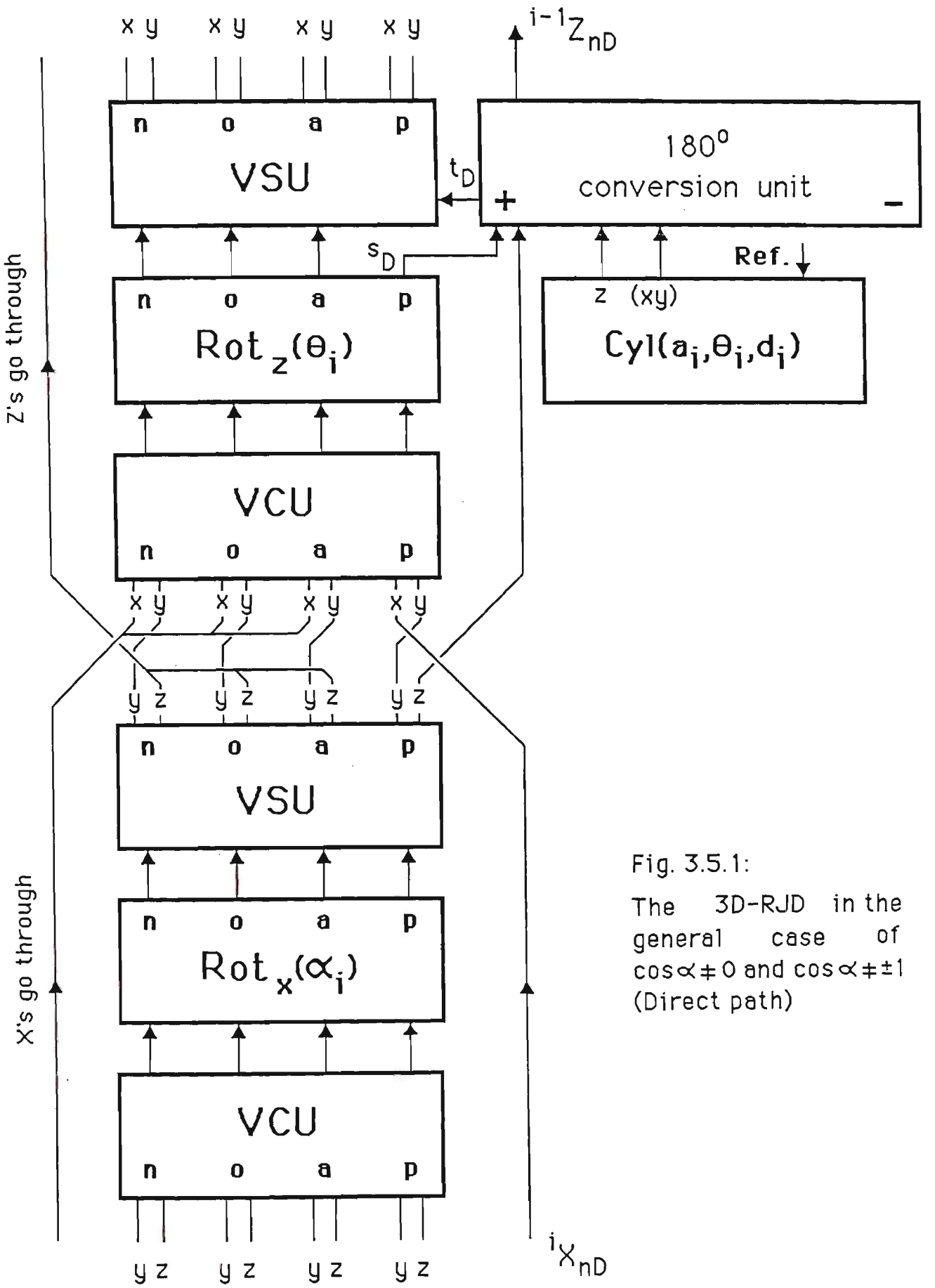


Fig. 3.5.1:

The 3D-RJD in the general case of $\cos\alpha \neq 0$ and $\cos\alpha \neq \pm 1$ (Direct path)

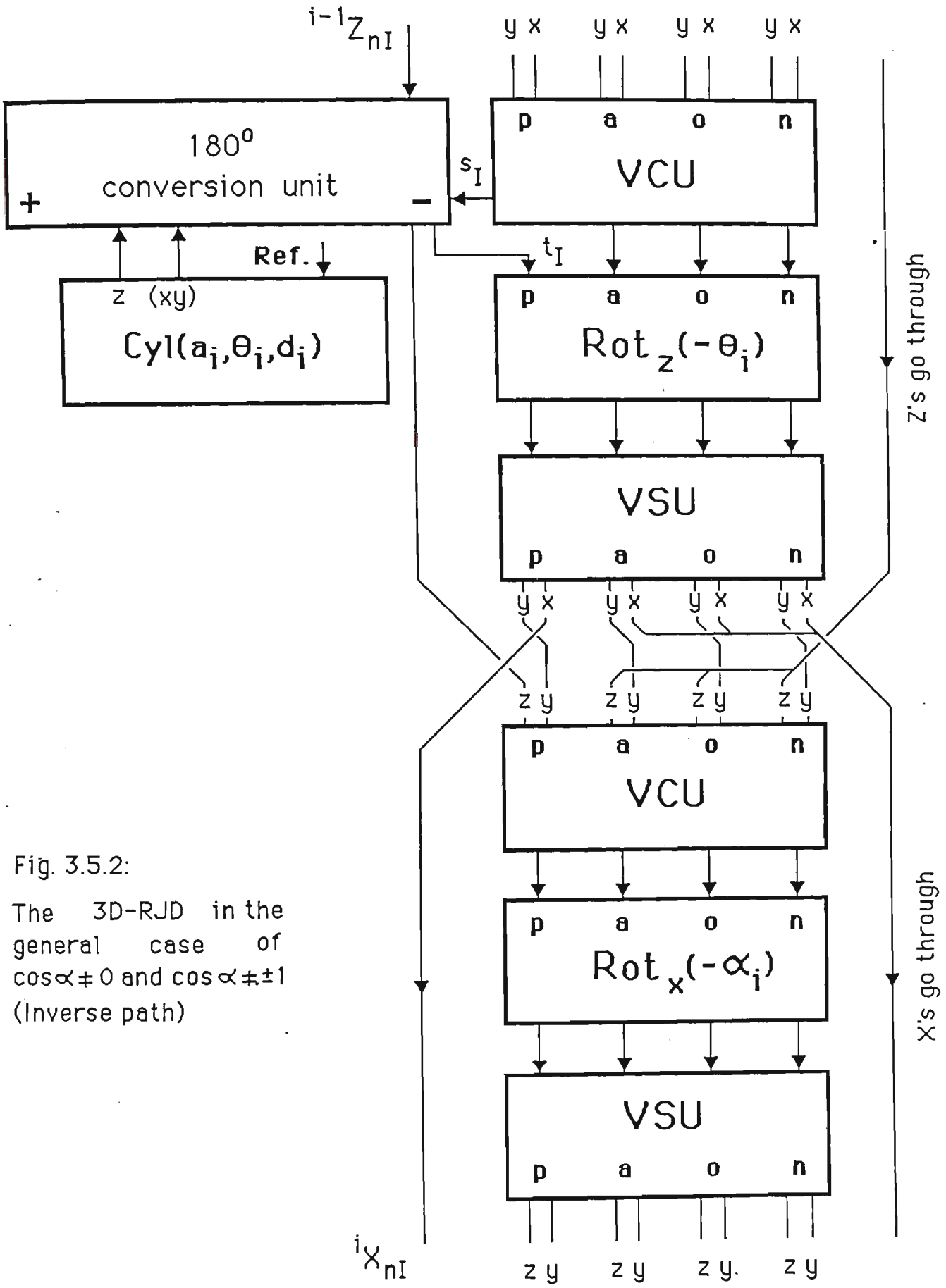


Fig. 3.5.2:

The 3D-RJD in the general case of $\cos\alpha \neq 0$ and $\cos\alpha \neq \pm 1$ (Inverse path)

3.6 SUMMARY

The three-dimensional problem has been described as a combination of two-dimensional problems. In this sense, the 3D-RJD has been developed which, from the point of view of the design philosophy, essentially consists of the 2D-RJD. All the other units developed in this chapter, in particular, the vector combiner unit and the vector splitter unit, have the function of preparing and processing signals to and from the 2D-RJD, respectively.

A hardware implementation has been developed which processes sinewaves to perform the matrix manipulations required. It has also been shown that this implementation of the 3-D RJD can be used even for cases when $\cos \alpha \neq 0$ and $\cos \alpha \neq \pm 0$.

Chapter 4

SOLVING THE INVERSE KINEMATIC PROBLEM USING THE ROBOT JOINT DESCRIPTOR TECHNIQUE

4.1 INTRODUCTION

The consideration presented under section 2.2 has indicated that although derived from the mutually inverse equations, the terms ${}^{m}\mathbf{ROT}_{nd}$ and ${}^{m}\mathbf{ROT}_{ni}$, as well as the terms ${}^{m}\mathbf{TRL}_{nd}$ and ${}^{m}\mathbf{TRL}_{ni}$, will in general not necessarily be equal, for m=o,1,...,n. where n is the number of degrees of freedom (d.o.f.) of the robot arm under consideration. Thus, a comparison of the ROT and TRL functions available on the direct RJD-path with their counterparts on the inverse path would provide a meaningful measure for the position and orientation errors of the robot arm, useful in the search for inverse kinematic solutions for this robot arm. This chapter then inevitably arises from the desire of making use of those errors in finding inverse kinematic solutions. If this can be done it will be possible to find a new alternative solution technique using sinewave simulation.

This alternative solution technique would then provide a general systematic means of obtaining solutions applicable to any manipulator rather than intuitive methods which tend to apply to specific classes of manipulator. It will also be shown that this general method applies to manipulators irrespective of the "twist angle" α.

4.2 THE ROBOT JOINT DESCRIPTOR (RJD) AND ITS FUNDAMENTAL CHARACTERISTICS.

Let us recall the manipulation of the chain products

$$ {}^{o}\mathbf{T}_{nD} = {}^{o}\mathbf{A}_1 . {}^{1}\mathbf{A}_2 \ldots {}^{i-1}\mathbf{A}_1 \ldots {}^{n-2}\mathbf{A}_{n-1} . {}^{n-1}\mathbf{A}_n \quad \text{for } n>0 \qquad \text{Eq.4.2.1.a} $$

$$ {}^{n}\mathbf{E}_n = \{ {}^{n}\mathbf{A}_{n-1} . {}^{n-1}\mathbf{A}_{n-2} \ldots {}^{i}\mathbf{A}_{i-1} \ldots {}^{2}\mathbf{A}_1 . {}^{1}\mathbf{A}_o \} . \; {}^{o}\mathbf{T}_{nI} \qquad \text{Eq.4.2.1.b} $$

where ${}^{i-1}\mathbf{A}_i$ is the matrix representation of Denavit/Hartenberg in the form of a (4x4) homogeneous transformation matrix describing orientation and position of the i-th coordinate system with respect to the (i-1)th coordinate system assigned at the i-th and (i-1)th robot joint, respectively.

Before proceeding further we remark that, by definition of the inverse geometric problem, the matrix manipulation of the right-hand side of Eq.4.2.1.b will result in a (4x4) unit matrix for a given ${}^{o}\mathbf{T}_{nI}$ if we have found a solution. Until then, however, the matrix ${}^{n}\mathbf{E}_n$ on the left-hand side of this equation will differ from the unit matrix in general and is here called the error matrix.

Let us also recall the partitioning representation:

$$ {}^{i-1}\mathbf{A}_i = \begin{bmatrix} {}^{i-1}\mathbf{ROT}_i & {}^{i-1}\mathbf{TRL}_i \\ 0 & 1 \end{bmatrix} \qquad \text{Eq.4.2.2} $$

where ${}^{i-1}\mathbf{ROT}_i$ is a (3x3) matrix and ${}^{i-1}\mathbf{TRL}_i$ is (3x1) vector containing orientation and position information of that A-matrix, respectively.

It has been shown by induction (See Appendix 1) that the following applies:

$${}^{i-1}\mathbf{ROT}_{nD} = {}^{i-1}\mathbf{ROT}_i \,.\, {}^{i}\mathbf{ROT}_{nD} \qquad \text{Eq.4.2.3}$$

$${}^{i-1}\mathbf{TRL}_{nD} = {}^{i-1}\mathbf{ROT}_i \,.\, {}^{i}\mathbf{TRL}_{nD} + {}^{i-1}\mathbf{TRL}_i \qquad \text{Eq.4.2.4}$$

for i=n,n-1,...,1. The subscript "D" is used to show that we are dealing with the intermediate results of the direct kinematic problem of Eq.4.2.1.a. The "inverse" equations are:

$${}^{i}\mathbf{ROT}_{nI} = ({}^{i-1}\mathbf{ROT}_i)^{-1} \,.\, {}^{i-1}\mathbf{ROT}_{nI} \qquad \text{Eq.4.2.5}$$

$${}^{i}\mathbf{TRL}_{nI} = ({}^{i-1}\mathbf{ROT}_i)^{-1} \,.\, \{{}^{i-1}\mathbf{TRL}_{nI} - {}^{i-1}\mathbf{TRL}_i\} \qquad \text{Eq.4.2.6}$$

for i=1,2,...,n, where the subscript "I" is used to indicate that we are considering the intermediate results of the inverse kinematic problem of Eq. 4.2.1.b.

Fig. 4.2.1 shows a graphical representation of Eq. 4.2.3. - Eq.4.2.6 which will be used as a general symbol for the Robot

Joint Descriptor (RJD), with the 180^0 symbol being used to represent the inversion of ${}^{i-1}\mathbf{TRL}_i$ to form $-{}^{i-1}\mathbf{TRL}_i$.

The RJD has the following two fundamental characteristics:

- Cascade characteristics. This is obvious from the equations. That is, staying on the direct manipulation path (subscripts D), one sees that the outputs of the i-th RJD will serve as inputs for the next adjacent (i-1)th RJD in arriving at the result for the direct kinematic manipulation of Eq.4.1.2.a. As is seen, the inverse manipulation path (subscripts I) also possesses the same characteristics. However, the direction of manipulation is reversed.

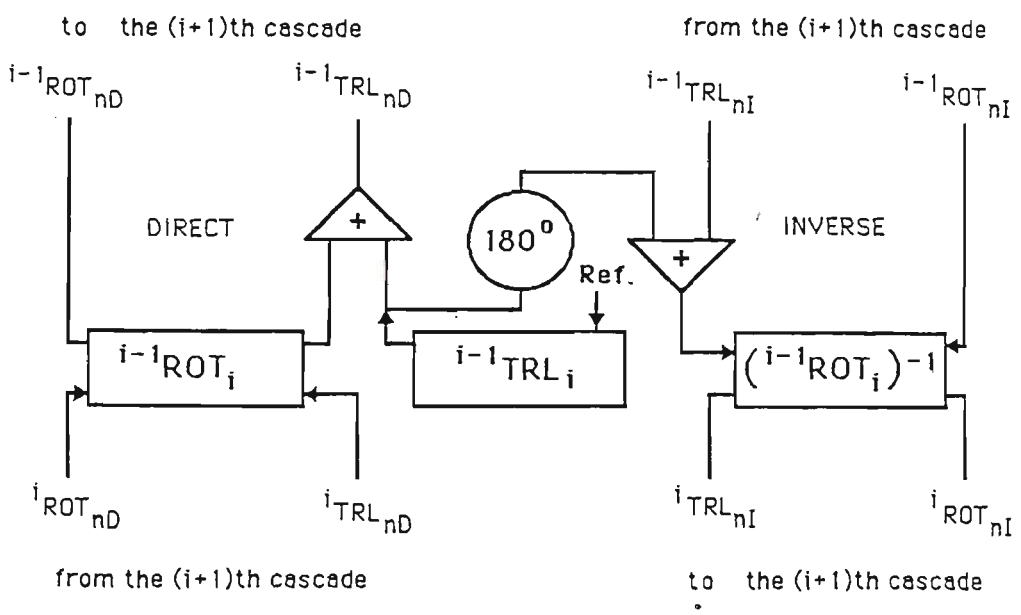


Fig. 4.2.1. : The i-th Generalized Robot Joint Descriptor.

- Balance characteristics. That is, if we have found the orientation pairs ${}^{i}\mathbf{ROT}_{nI} = {}^{i}\mathbf{ROT}_{nD}$, we will also find ${}^{i-1}\mathbf{ROT}_{nI} = {}^{i-1}\mathbf{ROT}_{nD}$ and vice versa. The same applies for the position pairs ${}^{i}\mathbf{TRL}_{nI}$, ${}^{i}\mathbf{TRL}_{nD}$, ${}^{i-1}\mathbf{ROT}_{nI}$, ${}^{i-1}\mathbf{ROT}_{nD}$.

Let us verify the balance characteristics of the RJDs.

Substitution in Eq.4.2.3 and Eq.4.2.5, shows that:

if ${}^{i}\mathbf{ROT}_{nI}={}^{i}\mathbf{ROT}_{nD}$ then ${}^{i-1}\mathbf{ROT}_{nI}={}^{i-1}\mathbf{ROT}_{nD}$ and vice versa, for i=1,2,...,n.

Similarly, by substitution in Eq.4.2.4 and Eq.4.2.6 we have:

if ${}^{i}\mathbf{TRL}_{nI}={}^{i}\mathbf{TRL}_{nD}$ then${}^{i-1}\mathbf{TRL}_{nI}={}^{i-1}\mathbf{TRL}_{nD}$ and vice versa, for i=1,2,...n.

In the following, before we make use of this, the necessity for decoupling the tool from the robot arm is illustrated

4.3 TRANSFORM EQUATIONS

For any robot arm with n d.o.f., the following relationship is true:

$${}^{w}\mathbf{Z}_{o}.{}^{o}\mathbf{T}_{n}.{}^{n}\mathbf{E}_{g}={}^{w}\mathbf{E}_{g} \qquad \text{Eq.4.3.1}$$

where:

${}^{w}\mathbf{Z}_{o}$: Transform describing location of the robot arm with respect to some reference world coordinate system.

${}^{o}\mathbf{T}_{n}$: Robot arm matrix as matrix product of the n elementary A-matrices.

${}^{n}\mathbf{E}_{g}$: Tool transform describing location of the robot's gripper with respect to the robot's end coordinate system (fixed).

${}^{w}\mathbf{E}_{g}$: Transform describing location of the gripper with respect to the world's coordinate system.

(The notations ${}^{n}\mathbf{E}_{g}$ and ${}^{w}\mathbf{E}_{g}$ are not to be confused with the error matrix ${}^{n}\mathbf{E}_{n}$ of Eq. 4.2.1.b).

To conform with common usage Eq.4.3.1 is here rewritten as:

${}^{w}\mathbf{Z}_{o}+{}^{o}\mathbf{T}_{n}+{}^{n}\mathbf{E}_{g}={}^{w}\mathbf{E}_{g}$

where the sign "+" is used to denote matrix transform multiplication [11]. Now, given a desired location of the gripper with respect to the world's coordinate system namely ${}^{w}\mathbf{E}_{g}$, the location of the gripper ${}^{o}\mathbf{GRIPPER}_{g}$ with respect to the zero-th coordinate system (usually at the robot's base) is then:

$${}^{o}\mathbf{T}_{n}+{}^{n}\mathbf{E}_{g}=-{}^{w}\mathbf{Z}_{o}+{}^{w}\mathbf{E}_{g}={}^{o}\mathbf{GRIPPER}_{g} \qquad \text{Eq.4.3.2}$$

where $-{}^{w}\mathbf{Z}_{o}$ denotes the inverse of ${}^{w}\mathbf{Z}_{o}$ which is, by definition fixed. Let us "decouple" the tool ${}^{n}\mathbf{E}_{g}$ from the robot arm to obtain the arm matrix ${}^{o}\mathbf{T}_{n}$ as follows:

$$ {}^{o}\mathbf{T}_{n}=-{}^{w}\mathbf{Z}_{o}+{}^{w}\mathbf{E}_{g}-{}^{n}\mathbf{E}_{g}={}^{o}\mathbf{WRIST}_{g} \qquad \text{Eq.4.3.3} $$

As can be seen, given the transform of the right hand side ${}^{o}\mathbf{GRIPPER}_{g}$ or ${}^{o}\mathbf{WRIST}_{g}$, it is easier to use Eq.4.3.3 to solve for the joint variables embedded in the arm matrix ${}^{o}\mathbf{T}_{n}$. The transform ${}^{o}\mathbf{WRIST}_{g}$ is here referred to as ${}^{o}\mathbf{T}_{5I}$. Thus, in the following, we will keep this same notation for ${}^{o}\mathbf{WRIST}_{g}$.

4.4 ARITHMETIC EXPRESSIONS FOR THE SOLUTIONS

In order to give insight into the RJD procedure we first of all show how it can be used as a systematic method of finding the conventional arithmetic expressions.

We make use of the balance characteristics of the RJD phase shift system to derive the solutions by comparing the variables of the direct manipulation path with those of the inverse manipulation path. If some unambiguous constraints arise from this equalizing process, we will be able to establish a solving procedure for the q_i for i=1,2,...,n at least for the type of robot manipulator under consideration.

The arm matrix ${}^{o}\mathbf{T}_n$ of Eq. 4.3.3 typically involves only a number of q_k for k=1,2,...,j with j<n affecting the position vectors in the phase shift system. Thus, those q_k for k=1,2,...j are called position-sensitive and the remaining q_l for l=j+1,j+2,...,n orientation-sensitive. Therefore, the inverse geometric solutions are derived in two stages: a) Deriving positional solutions and b) Deriving orientational solutions.

a) Deriving positional Solutions

The constraints

$${}^{k}\mathbf{TRL}_{nI}={}^{k}\mathbf{TRL}_{nD} \qquad \text{for } k=o,1,2,...,n$$

indicate element by element equalities denoting that these vectors should have the same magnitude as well as the same direction. However, let us have a look at the particular case of k=n. ${}^{n}\mathbf{TRL}_{nD}$ is a null-vector having undefined orientation, because it is the positional part of the (4x4) unit matrix ${}^{n}\mathbf{T}_{nD}$ that serves as "initial condition" for the direct manipulation. This means that when equalizing the pair ${}^{n}\mathbf{TRL}_{nI}$ and ${}^{n}\mathbf{TRL}_{nD}$, we shall not be concerned about their orientations, but indeed, only their magnitudes or, preferably, their magnitudes squares. This represents a tremendous constraint reduction, because we then only have to equalize the magnitude squares of the ${}^{k}\mathbf{TRL}_{nI}$ and ${}^{k}\mathbf{TRL}_{nD}$ for

k=0,1,2...,n in the attempt to equalize the particular pair ${}^{n}\mathbf{TRL}_{nI}$ and ${}^{n}\mathbf{TRL}_{nD}$. If this is achieved, i.e. ${}^{n}\mathbf{TRL}_{nI}={}^{n}\mathbf{TRL}_{nD}$, all the position pairs of the RJD-system will be equal in magnitude as well as in orientation due to the balance characteristics of the RJD. Therefore,

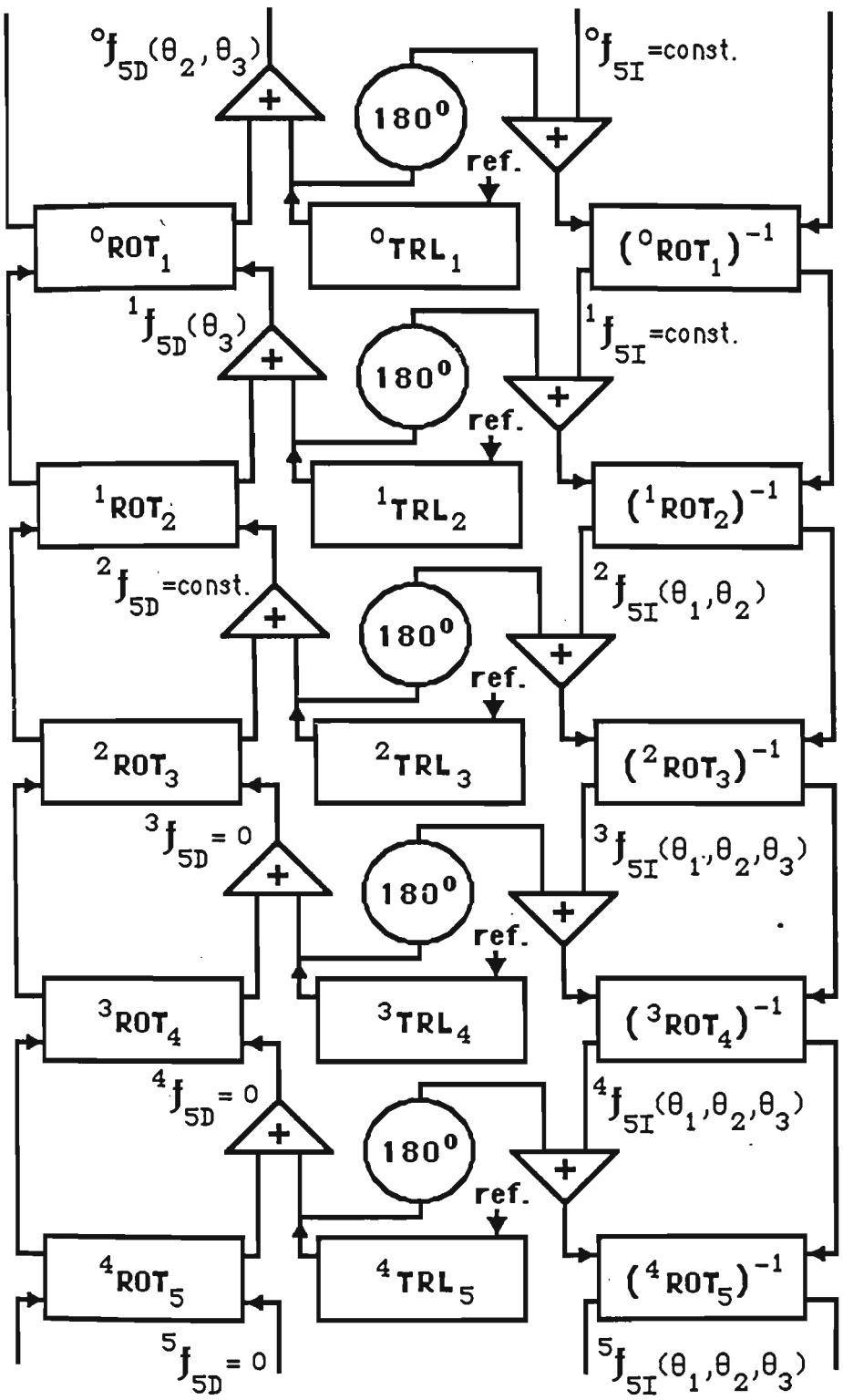
if $[{}^{n}\mathbf{TRL}_{nI}]^2=[{}^{n}\mathbf{TRL}_{nD}]^2=0$, then ${}^{k}\mathbf{TRL}_{nI}={}^{k}\mathbf{TRL}_{nD}$ for k=0,1,2...,n

Indeed, the solutions to the inverse kinematic problem of Eq.4.3.3 are, for the position-sensitive variables, found by equalizing the magnitude squares of the relevant position vectors in the direct manipulation with their counterparts in the inverse manipulation.

Thus, in the first stage, the RJD computing diagram offers a means of comparing the position vectors ${}^{k}\mathbf{TRL}_{nD}$ of the direct manipulation path with those ${}^{k}\mathbf{TRL}_{nI}$ of the inverse manipulation path for k=0 to n. The first step of this stage is to identify the functional relationships between the particular position vectors on both paths and their independent variables q_k, for example, by establishing arithmetic expressions for the $[{}^{k}\mathbf{TRL}_{nI}]^2$ and $[{}^{k}\mathbf{TRL}_{nD}]^2$ for k=0,1,2,...,n. Then, the comparison is in principle performed by equalizing those magnitude squares in some systematic way as presented in the following.

Let us demonstrate this on the example of the 5-axis Mitsubishi robot arm RM-501 (see Fig 4.4.1) where we have

n=5. For this particular robot arm the table 4.4.1 developed by inspection of Fig.4.4.1 offers an overall insight into the problem.



$$^{0}A_1=\begin{bmatrix} -S_1 & 0 & C_1 & 0 \\ C_1 & 0 & S_1 & 0 \\ 0 & 1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^{1}A_2=\begin{bmatrix} C_2 & -S_2 & 0 & a_2C_2 \\ S_2 & C_2 & 0 & a_2S_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^{2}A_3=\begin{bmatrix} C_3 & -S_3 & 0 & a_3C_3 \\ S_3 & C_3 & 0 & a_3S_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^{3}A_4=\begin{bmatrix} -S_4 & 0 & C_4 & 0 \\ C_4 & 0 & S_4 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^{4}A_5=\begin{bmatrix} C_5 & -S_5 & 0 & 0 \\ S_5 & C_5 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Solving sequence	Position			Orientation
Variable(s)	θ_3	θ_2	θ_1	$[\theta_4,\theta_5]$ arbitrary order
Error reference	$^{1}f_{5I}$	$^{0}f_{5I}$	$^{5}f_{5D}$	diff_trace_square

Fig. 4.4.1

Examining the table, we realize that $[{}^{1}\mathbf{TRL}_{5D}]^2$ is a function of only one variable θ_3 while $[{}^{1}\mathbf{TRL}_{5I}]^2$ is a constant. Thus, the solution for θ_3 is determined by equalizing this particular pair. This represents the first unambiguous constraint obtainable from the table. In the next step, we realize that the pair $[{}^{\circ}\mathbf{TRL}_{5I}]^2$ and $[{}^{\circ}\mathbf{TRL}_{5D}]^2$ has now become a function of only one variable, θ_2 as the quantity $[{}^{\circ}\mathbf{TRL}_{5I}]^2$ is a constant and $[{}^{\circ}\mathbf{TRL}_{5D}]^2$ has now become a function of only one variable, θ_2, since θ_3 has been found. In the last step, keeping the solutions θ_2 and θ_3 unchanged, we can equalize any remaining pair $[{}^{k}\mathbf{TRL}_{5I}]^2$ and $[{}^{k}\mathbf{TRL}_{5D}]^2$ for k=2,3,4 or 5 and solve for θ_1.

Direct manipulation	Inverse manipulation
$\lvert{}^{\circ}\mathbf{TRL}_{5D}\rvert^2 = {}^{\circ}f_{5D}(\theta_2,\theta_3)$ Two variables	$\lvert{}^{\circ}\mathbf{TRL}_{5I}\rvert^2 = \text{const.}$ No variables
$\lvert{}^{1}\mathbf{TRL}_{5D}\rvert^2 = {}^{1}f_{5D}(\theta_3)$ One variables	$\lvert{}^{1}\mathbf{TRL}_{5I}\rvert^2 = \text{const.}$ No variables
$\lvert{}^{2}\mathbf{TRL}_{5D}\rvert^2 = \text{const.}$ No variables	$\lvert{}^{2}\mathbf{TRL}_{5I}\rvert^2 = {}^{\circ}f_{5I}(\theta_1,\theta_2)$ Two variables
$\lvert{}^{3}\mathbf{TRL}_{5D}\rvert^2 = 0$ No variables	$\lvert{}^{3}\mathbf{TRL}_{5I}\rvert^2 = {}^{3}f_{5I}(\theta_1,\theta_2,\theta_3)$ Three variables
$\lvert{}^{4}\mathbf{TRL}_{5D}\rvert^2 = 0$ No variables	$\lvert{}^{4}\mathbf{TRL}_{5I}\rvert^2 = {}^{4}f_{5I}(\theta_1,\theta_2,\theta_3)$ Three variables
$\lvert{}^{5}\mathbf{TRL}_{5D}\rvert^2 = 0$ No variables	$\lvert{}^{5}\mathbf{TRL}_{5I}\rvert^2 = {}^{5}f_{5I}(\theta_1,\theta_2,\theta_3)$ Three variables

Table 4.4.1: Comparison table for RM-501

b) Deriving orientational solutions

The next step is to achieve the correct orientation. We make use of the trace vector of the error matrix ${}^5\mathbf{E}_5$ available at the end output of the inverse manipulation path of the RJD computing diagram to derive the solution for the remaining variables θ_4 and θ_5. Mathematically, this matrix is determined by:

$$ {}^5\mathbf{E}_5 = ({}^\circ\mathbf{T}_5)^{-1} . {}^\circ\mathbf{T}_{5I} \qquad \text{Eq.4.4.1} $$

Where ${}^\circ\mathbf{T}_{5I}$ represents the desired location of the robot manipulator and $({}^\circ\mathbf{T}_5)^{-1}$ is the result of the inverse manipulation path if its input matrix has the value of the unity matrix. This matrix equation implies

$$ {}^5\mathbf{ROT}_{5I} = ({}^\circ\mathbf{ROT}_5)^{-1} . {}^\circ\mathbf{ROT}_{5I} \qquad \text{Eq.4.4.2} $$

which becomes a (3x3) unity matrix when we have found solutions. That is, in particular, if ${}^5\mathbf{ROT}_{5I} = {}^5\mathbf{ROT}_{5D}$ then ${}^i\mathbf{ROT}_{5I} = {}^I\mathbf{ROT}_{5D}$ for I=0,1,2,...,5. (We recall that ${}^5\mathbf{ROT}_{5D}$ is a (3x3) unity matrix because it is the orientation part of ${}^5\mathbf{T}_{5D}$). However, we are only interested in the trace vector of ${}^5\mathbf{ROT}_{5I}$.

We define the quantity:

$$ \mathbf{diff\_trace\_square} = (1-n_{x\_error})^2 + (1-o_{y\_error})^2 + (1-a_{z\_error})^2 >= 0 \qquad \text{Eq.4.4.3} $$

where n_{x_error} and o_{y_error} and a_{z_error} are the elements of the trace vector of $^5\mathbf{ROT}_{5I}$. As is evident, **diff_trace_square** is a function of all the orientational variables. Thus, the nearest orientations are found in its minimal extremes if they exist. Solutions are found when **diff_trace_square**=0.

In practice, the arithmetic expression for **diff_trace square** is quite complicated and so not very convenient to use. However, it turned out that for the particular robot arm RM-501, this quantity **diff_trace_square** is not necessary. Instead, we take partial derivatives of any two components of the trace vector of $^5\mathbf{ROT}_{5I}$ with respect to the variables of interest, require them to be zero and find the maximal extremes. The maximum of those components will give the closest orientation and the solutions for the remaining variables θ_4 and θ_5 are found from these maximal extremes. A detailed solution derivation is given in Appendix 5. A solution summary is given on the next page, Table 4.4.2. As is seen, the solution expressions involve 9 transcendental function calls, 3 arithmetic function calls, 22 multiplications, 17 additions and 2 divisions.

c) Degeneracy

The robot arm degenerates when a particular position error, defined as the difference between the magnitude squares of the corresponding position vectors (see table 4.4.1), does

not change at all upon the variation of the relevant variable or variables. The same term applies whenever the quantity **diff_trace_square** remains unchanged upon the variation of the relevant orientational variable(s).

Table 4.4.2 **RM-501 SOLUTION SUMMARY**

$$ {}^{0}T_5 = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Given desired location ${}^{0}T_5$ (DH-Matrix)

a_i , d_i , θ_i DH-parameters/variables $i=1,2,\ldots,5$

Positional solutions :

$$ \theta_3 = \text{ATAN2}\left[\pm\sqrt{1-(\cos\theta_3)^2} \, , \, \cos\theta_3 \right] $$

$$ \text{where} \quad \cos\theta_3 = \frac{p_x^2 + p_y^2 + (p_z - d_1)^2 - (a_2^2 + a_3^2)}{2 \cdot a_2 \cdot a_3} $$

$$ \theta_2 = \text{ATAN2}\left[\frac{p_z - d_1}{r} \, , \, \pm\sqrt{1-\left(\frac{p_z - d_1}{r}\right)^2} \right] - \varphi $$

$$ \text{where} \quad \varphi = \text{ATAN2}\left[a_3 \cdot S_3 \, , \, a_3 \cdot C_3 + a_2 \right] $$

$$ r = +\sqrt{(a_3 \cdot S_3)^2 + (a_3 \cdot C_3 + a_2)^2} $$

$$ \theta_1 = \text{ATAN2}\left[\mp p_x \, , \, \pm p_y \right] $$

ATAN2 is arctan function with two real arguments

Orientational solutions :

$$ \theta_4 = \text{ATAN2}\left[\pm a_z \, , \, \pm(-a_x \cdot S_1 + a_y \cdot C_1) \right] - \theta_2 - \theta_3 $$

where $C_1 = \cos\theta_1$, $S_1 = \sin\theta_1$

$$ \theta_{5a} = \text{ATAN2}\left[n_x \cdot C_1 + n_y \cdot S_1 \, , \, n_x \cdot S_1 \cdot S_{234} - n_y \cdot C_1 \cdot S_{234} + C_{234} \cdot n_z \right] $$

$$ \theta_{5b} = \theta_{5a} \pm \pi $$

where $C_{234} = \cos(\theta_2 + \theta_3 + \theta_4)$,

$S_{234} = \sin(\theta_2 + \theta_3 + \theta_4)$

Degenerate when $p_x = 0$ and $p_y = 0$. Any θ_1 is then solution.

No orientational solution possible when $\frac{p_x}{p_y} \neq \frac{a_x}{a_y}$

Solutions involve 9 transcendental function calls, 3 arithmetic function calls, 17 additions, 22 multiplications and 2 divisions.

Given a desired location ${}^{0}T_5$, $(\theta_{3+}, \theta_{2+}, \theta_1)$ is a set of positional solutions. $(\theta_{3+}, \theta_{2-}, \theta_1 + \pi)$ is another set of solutions. Another two sets with θ_{3-} are also solutions. Orientational solutions virtually depend on the choice of θ_3 only.

The same solution method has been applied to several common geometric configurations (see Appendix 6). The existence of the solution sequences demonstrates the wide applicability of the new solution procedure.

d) Solving for orientation only, or for position only.

We remark that, applying the same solution technique for Eq.4.3.2., solutions for the gripper's position embedded in ${}^{o}\mathbf{GRIPPER}_{g}$ will be found, in general, at incorrect orientation. Similarly, correct orientation can be achieved using the method with the **diff_trace_square** quantity; however, the positional constraints will not necessarily be satisfied.

4.5 ITERATIVE NON-INTUITIVE SOLUTIONS USING SINEWAVE SIMULATION

Based on the theoretical development of the RJD, the three-dimensional RJD has been software-simulated for the inverse geometric problem of the RM-501 (Using Mainframe UNIVAC 1100 of the Wollongong University's Computer Centre with University Wisconsin Pascal, written in 1984. See Appendix 7 for printout) where both the direct and the inverse manipulation have been implemented as required by the solving procedure. In particular, three hundred and sixty points of one cycle of a sinewave have been generated once and stored

in a global look-up table. All matrix manipulations have then been performed with phase shifting procedures using the look-up table and apart from calculations of the **TRL** vector magnitudes of the **diff_trace_square** quantity, no significant arithmetic expressions have been used in the program. Indeed, all the **TRL** vectors and the trace elements n_{x_error}, o_{y_error} and a_{z_error} needed in Eq. 4.4.3 are directly available from the simulated RJD phase shift system as sinewave amplitudes.

In particular, for a given desired location expressed in a target matrix ${}^{o}\mathbf{T}_{5I}$, the computer first performs the matrix manipulations using the phase shifting procedures mentioned above to determine the actual **ROT** and **TRL** of the RJD phase shift system for the actual values of the θ_i i=1,2,...,5. This is done by, for the direct manipulation path, consecutively introducing the required phase shift angles θ_i to the relevant sinewaves, starting by the n-th RJD cascade back to the first RJD cascade. For the inverse manipulation path, the program introduces the angles $-\theta_i$ to the relevant sinewaves, in the reverse order, starting by the first RJD cascade consecutively to the n-th RJD cascade. Then, the relevant variables are adjusted iteratively in the order required by the solving sequence. That is, for the RM-501, the program has to adjust θ_3 first, then θ_2 and then θ_1. As the program makes use of the **diff_trace_square** quantity, the adjust order for the remaining orientational variables θ_4 and θ_5 is arbitrary. Varying only one variable at a time,

the determination of the relevant actual **ROT** and **TRL** of the RJD phase shift system is performed repeatedly until the relevant error has become minimal or zero. The last iterative value of the one variable having just undergone the variation is then one solution. Now, the computer proceeds to the next relevant variable and repeats the iteration until all variables have been considered.

It was interesting to note that manipulator degeneracy, for example at $p_x=0$, $p_y=0$, $p_z<>0$ caused no problem at all with finding a solution for θ_1 as there was no change in the relevant error. In this case the current value of θ_1 is assumed as a solution and the next relevant variable proceeded to.

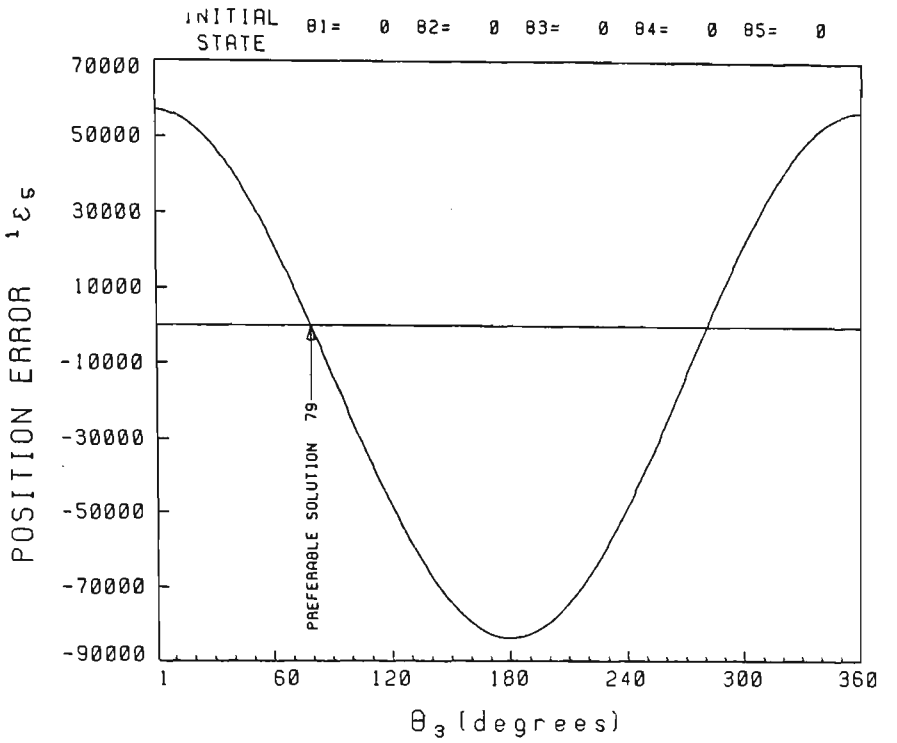
Fig. 4.5.1 and Fig. 4.5.2 show graphical representations of all possible position and orientation errors (for a given $^{o}\mathbf{T}_{5I}$, see target matrix on Fig. 4.5.1) as they arise when the joint variables are being changed (FORTRAN77 plot routines called fromthe Pascal program). We recall that the position errors are defined as the difference between the magnitude squares of the corresponding position vectors. As is indicated in Fig.4.5.1 the position-sensitive variables are adjusted in the order required from the comparison table 4.4.1. However, the orientation errors of Fig. 4.5.2 are the various representations of the quantity **diff_trace_square** which is the square of the magnitude of the difference vector of the error matrix and that of the unit matrix. The so-

obtained numerical solutions agreed with those obtained using conventional arithmetic expressions. As is obvious, the number of iteration steps depends on how close to the target the actual gripper is at the first iteration. With this iterative method, real-time solutions are possible, because the Cartesian motion of the robot arm is usually represented in terms of differential joint-interpolated motions.

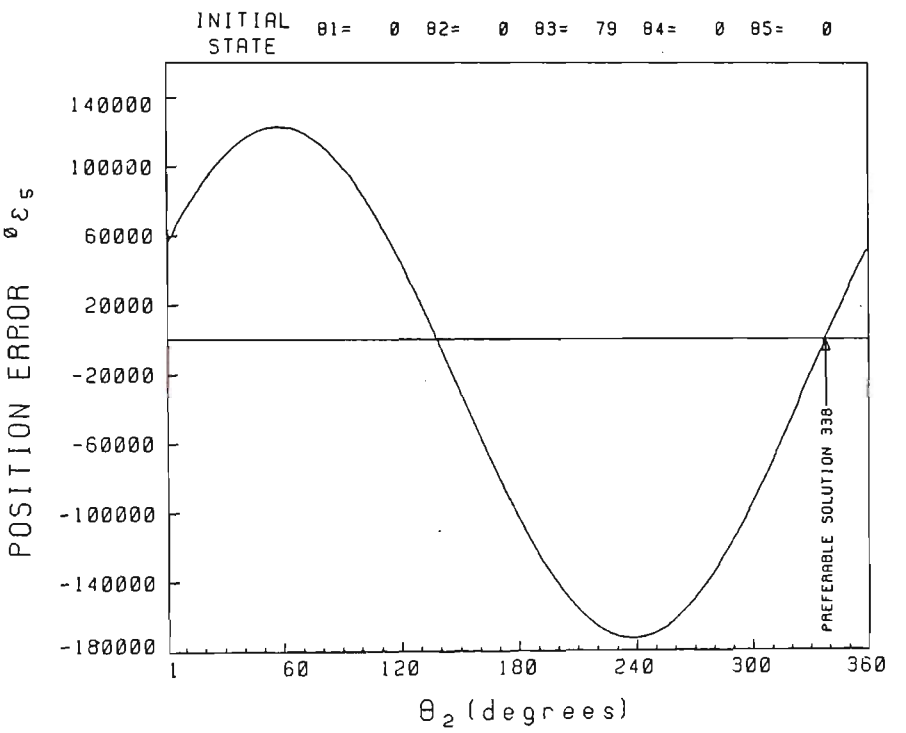
Target Matrix:

$$ {}^{5}T_{5I} = \begin{bmatrix} -0.5 & 0.866 & 0 & -80 \\ -0.866 & -0.5 & 0 & 280 \\ 0 & 0 & 1 & 300 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

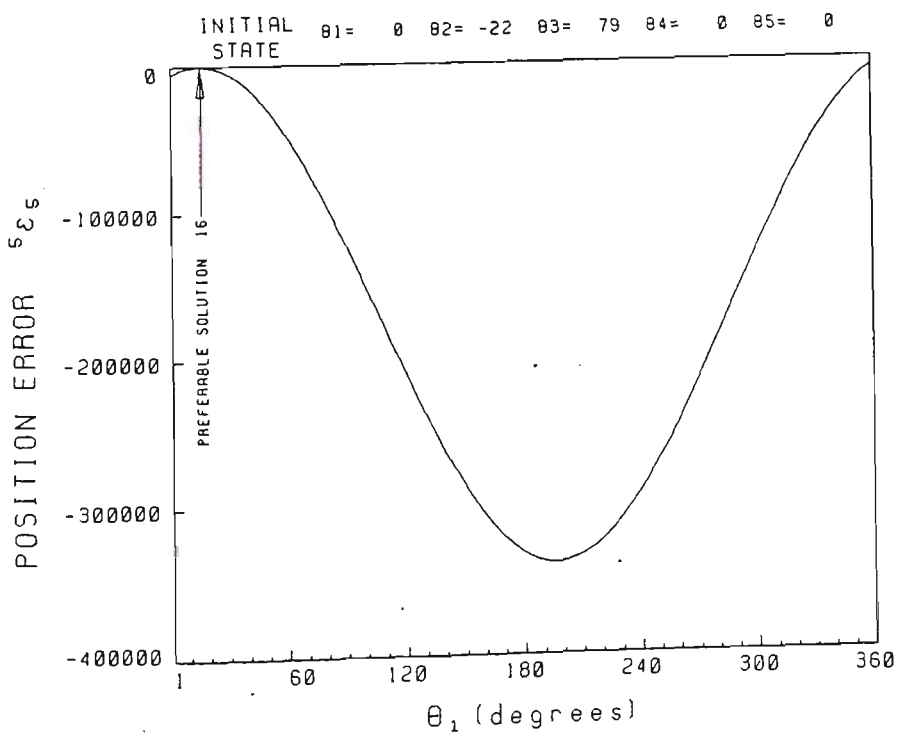
Fig. 4.5.1 a.)



b.)



c.)



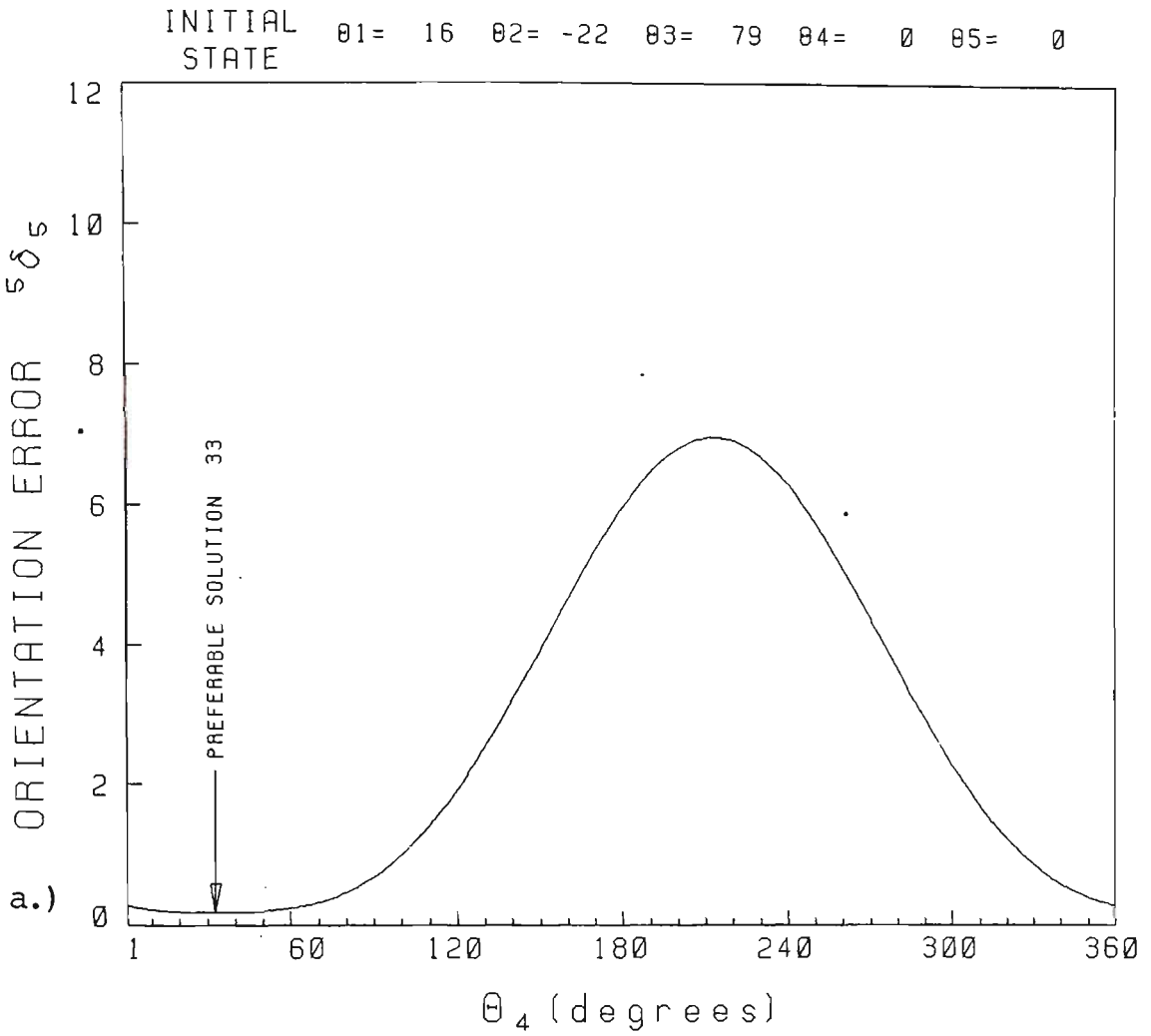
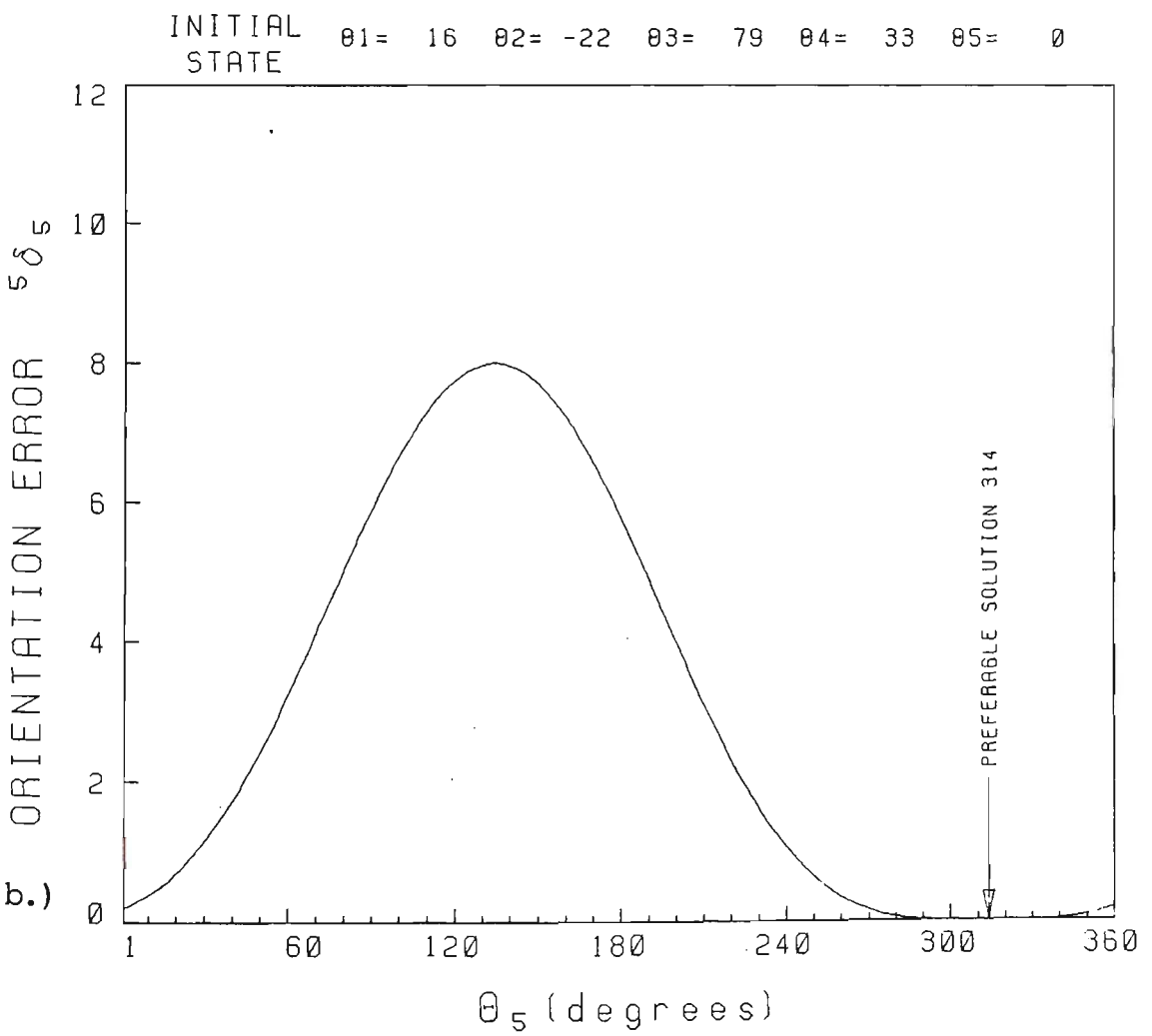

INITIAL STATE θ1= 16 θ2= -22 θ3= 79 θ4= 0 θ5= 0
ORIENTATION ERROR $_5\delta_5$
PREFERABLE SOLUTION 33
0
2
4
6
8
10
12
1
60
120
180
240
300
360
Θ_4 (degrees)

Fig. 4.5.2 a.)


INITIAL STATE θ1= 16 θ2= -22 θ3= 79 θ4= 33 θ5= 0
ORIENTATION ERROR $_5\delta_5$
PREFERABLE SOLUTION 314
0
2
4
6
8
10
12
1
60
120
180
240
300
360
Θ_5 (degrees)

b.)

In this iterative method, the calculation of the position errors $[{}^{k}\mathbf{TRL}_{nD}]^2 - [{}^{k}\mathbf{TRL}_{nI}]^2$ requries 6 multiplications and 5 additions for every index k. As we have to consider j such indexes, the total numbers of multiplications and additions in arriving at the solutions for j position-sensitive variables are found as

$$Z_{p}.j. \text{ (6 multiplications and 5 additions)}$$

where Z_p is the average number of iterations required in each optimization step.

The computation of the **diff_trace_square** quantity of Eq.4.4.3 requires 3 multiplications and 5 additions so that in arriving at the results for the (n-j) orientation-sensitive variables, the numbers of multiplications and additions calimed by this method are:

$$Z_{o}.(n-j).(3 \text{ multiplications and 5 additions})$$

where Z_o is the average number of iterations required in each step for the orientation - sensitive variables.

If the changes between two consecutive value of the variables are 2 degrees in average, we will apply $Z_p = 2$ and $Z_o = 2$, so that for the RM-501, the solutions require 48 multiplications and 50 additions. However, in this method, no transendental

function calls are involved. As is seen from this example, real-time solutions are possible.

Fig. 4.5.3 summarizes the implementation of the new solving technique. The flow chart of Fig. 4.5.3.a represents a computer program that only requires the knowledge of the geometric configuration of a particular robot arm, usually expressed in the well-known Denavit/Hartenberg matrices with relevant parameters/variables, to identify the particular solving sequence for that robot arm without geometric intuition. This identification need be performed only once for any one robot; therefore, the program requires only one pass. Once having identified the solving sequence, the solving procedure requires the routine shown in Fig. 4.5.3.b. As is seen, the routine makes use of the RJD-system that provides the relevant errors for comparison without conventional arithmetic expressions. There is no requirement for any inspection of equations or matrix elements and so no need for intuition based solutions.

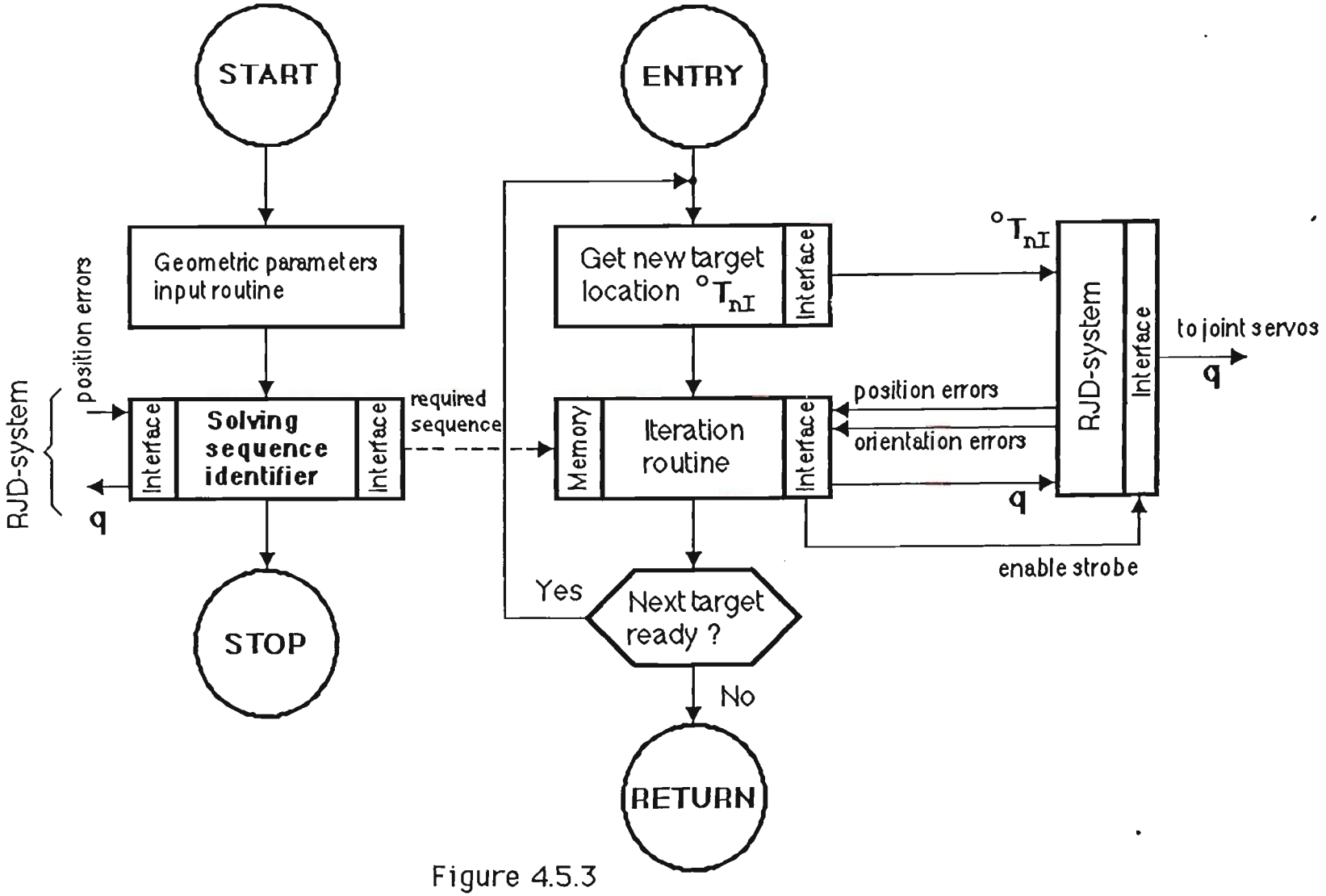


Figure 4.5.3

a) One pass program to identify solving sequence for a particular robot's geometry

b) Solving routine for a required $^{o}T_{nI}$

4.6 NUMERICAL SOLUTIONS FOR THE GENERAL CASE OF COS(α)$\neq$0 AND COS (α) $\neq \mp 1$

Obviously, for simple manipulator types such as the RM-501 and most commercially available robots, kinematic solutions of analytically closed form can be derived as presented under section 4.4. However, analytically closed form solutions are very difficult if not impossible to obtain for robot manipulators whose twist angles α do not satisfy $\cos(\alpha)=0$ or $\cos(\alpha)=\pm 1$.

The solution method using sinewave simulation presented under section 4.5) then offers a means of obtaining numerical solutions for the general case of $\cos(\alpha)\neq 0$ and $\cos(\alpha)\neq \pm 1$.

Indeed, as far as the positional solutions are concerned, if we take a look at table 4.4.1, we see that in general, the new solution procedure only requires the knowledge of which particular joint variables affect which particular position errors, but never assumes any specific values for the twist angles.

Similarly, orientational solutions are found by minimizing the quantity **diff_trace_square** of Eq.4.4.3 which is derived from the error matrix ${}^{n}\mathbf{E}_{n}$ that, in turn, doesn't require the twist angles to have any particular values.

We note that the RJD implementation of this general case has been discussed under section 3.5.

4.7 SUMMARY

This chapter has presented one unambiguous way for obtaining the inverse geometric solution. That is, the procedure enables the determination of a set of n simultaneous equations solvable for the robot's n independent joint variables without element by element comparisons in a matrix equation being performed on an intuitive basis.

Two solution techniques based on the same procedure have been demonstrated: 1) Arithmetic expressions for the solutions have been derived to give insight into how the method works. 2) Numerical solutions using sinewave simulation without using conventional arithmetic expressions for the solutions have also been obtained. Solutions can also be derived for the general case of cos $\alpha \neq \pm 1$ or 0. The technique has also been demonstrated with four different commonly available robot geometries. With all these configurations, the computer can use the error signals available from each RJD to derive for itself a suitable solving sequence. Fast iteration times have been demonstrated in all cases and degeneracy introduces no problem at all to the algorithm.

Chapter 5

THE SINUSOIDAL MATRIX PROCESSOR (SMP)

5.1 INTRODUCTION

A number of papers published by prominent authors [4,8,15,19,23,25] show that several problems in manipulator control remain to be solved. In particular two matters of importance are:

- Real time computing of the "inverse dynamics" of robot manipulators.
- Retaining the explicit-state dynamic model for the purpose of applying multivariable control theory in the fundamental sense of feedback control.

In general, these two problems can be considered as being made up of two major tasks:

a) Specifying nominal forces/torques acting at the robot joints for every given trajectory point.

b) Tracking a pre-specified motion trajectory. That is, assuming that the nominal forces/torques are known, the task is to find compensating forces/torques such that the robot tracks the desired trajectory.

In the first problem, the first task is emphasized where, for real-time reasons, the efficient computer algorithms for computing the nominal forces/torques precede the explicit-state structure of the dynamic model. However, in the second problem, the emphasis is on the second task, because the compensation of forces/torques is, in the fundamental sense

of feedback control, efficient only with advanced control laws/strategies which require the knowledge of the explicit-state dynamic model. The problem of this is that the explicit-state dynamic model is so complex that excessive computational time is required.

Various approaches are available for formulating manipulator dynamics [14,19]. However, the choice of the formulation depends on the task. Emphasizing the first task, the recursive formulation based on Newton-Euler dynamics (NE) is often used. By contrast, the "classic" Lagrange-Euler (LE) formulation is chosen if there is a desire for retaining the non-recursive dynamic structure of the explicit-state equations. If the two major tasks have equivalent importance for some control problems, compromises between the two dynamic approaches are inevitable. That is, one may employ the NE-formulation for real-time reasons to compute the nominal forces/torques and, for the purpose of gaining an overall insight into the dynamic problem and of applying advanced control laws/strategies, one may use the LE-formulation to determine the explicit-state dynamic functions necessary for the design of the controller/ compensator. However, because of excessive computational requirements, those dynamic functions are often either estimated in some way or obtained from look-up tables rather than directly computed from the standard LE-formulas.

Apart from computational considerations, it would also be desirable to have a systematic method for deriving the dynamic model. For some applications involving the use of (4x4) homogeneous transformation matrices, the LE-formation may be the most systematic and convenient formulation [4]. The derivation of such LE dynamic equations for open-chain manipulator systems can be found, for example, in [5]. For a robot arm with n degrees of freedom (d.o.f.) this dynamic equation system consists of n second-order, non-linear, highly coupled, ordinary differential equations which involve a number of additions and multiplications of the order n^4, too time consuming to compute in real-time [23].

Re-formulation of the LE-equations using recursive formulas for the matrix derivatives involved in the equations enables "real-time" calculations [8] and reveals equivalence between the LE- and the NE-formulations [9]. However, the recursive formulations whether LE or NE destroy the explicit-state of the dynamic model. In particular, the state variables disappear from those recurrent dynamic formulations making it more difficult to apply advanced control laws.

As discussed in Chapter 1, in an effort to decrease calculation times much work has been performed to locate some of the computer burden outside the main control computer [36,37,38,40], (off-CPU control). One common feature of these new approaches is that they still employ conventional arithmetic expressions that are executed on external

hardware. In addition they emphasize the use of the NE-formulations which for reasons mentioned above we would like to avoid. In this chapter a procedure is developed which applies the same strategy of Off-CPU control, and has an innovative matrix manipulation method not using conventional arithmetic expressions. This method produces the partial derivatives as indicated by the shaded area in Fig. 5.1.1 (repeated here for convenience from Fig 1.3). In particular, based on the development of the Robot Joint Descriptor, all the relevant first order and second order partial matrix derivatives involved in the LE-equations are found by phase shifting sinewaves. Bearing in mind that their computation using conventional techniques involves the order n^4 and n^3 of additions and multiplications (as Appendix 8 shows) it will be seen that a substantial potential saving of total computing time is achievable.

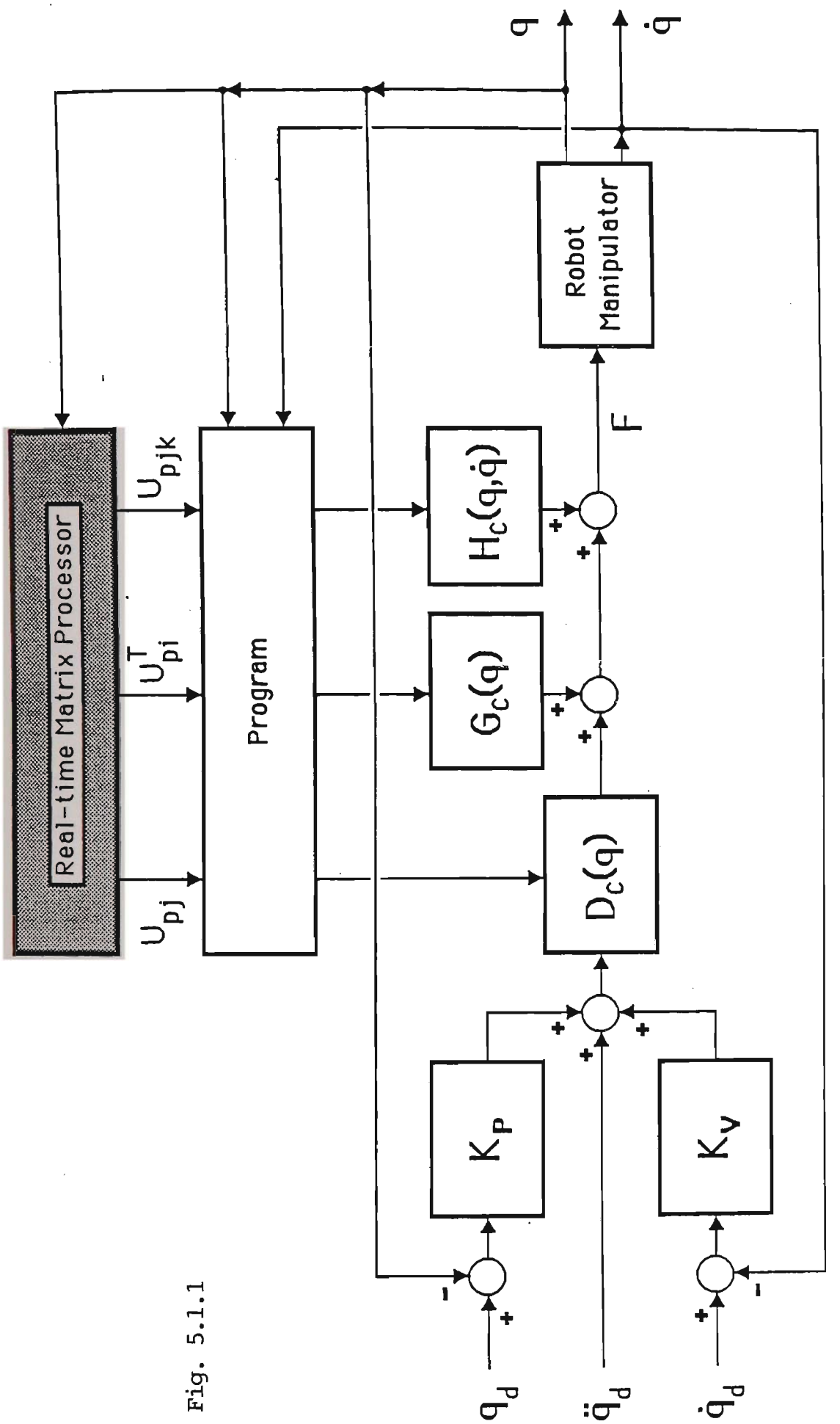

Real-time Matrix Processor
U_{pjk}
U_{pi}^T
U_{pj}
Program
$H_c(q,\dot{q})$
$G_c(q)$
$D_c(q)$
K_P
K_v
Robot Manipulator
F
q
$\dot{q}$
q_d
$\ddot{q}_d$
$\dot{q}_d$

Fig. 5.1.1

5.2 MATRIX DERIVATIVES IN THE SINEWAVE REPRESENTATION

The general A-matrix is known as:

$$ {}^{i-1}\mathbf{A}_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{Eq.5.2.1} $$

If the i-th joint is prismatic, we have for the first order and the second order derivative of this matrix (with respect to its variable d_i) as follows:

$$ \frac{d}{dd_i} {}^{i-1}\mathbf{A}_i = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \text{Eq. 5.2.2} $$

$$ \frac{d^2}{dd_i^2} {}^{i-1}\mathbf{A}_i = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \text{Eq.5.2.3} $$

If the joint is revolute, we have:

$$ \frac{d}{d\theta_i} {}^{i-1}\mathbf{A}_i = \begin{bmatrix} -\sin\theta_i & -\cos\theta_i\cos\alpha_i & \cos\theta_i\sin\alpha_i & -a_i\sin\theta_i \\ \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \text{Eq.5.2.4} $$

$$ \frac{d^2}{d\theta_i^2} {}^{i-1}\mathbf{A}_i = \begin{bmatrix} -\cos\theta_i & \sin\theta_i\cos\alpha_i & -\sin\theta_i\sin\alpha_i & -a_i\cos\theta_i \\ -\sin\theta_i & -\cos\theta_i\cos\alpha_i & \cos\theta_i\sin\alpha_i & -a_i\sin\theta_i \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \text{Eq.5.2.5} $$

We introduce the first order and the second order differential translational matrix operator:

$$\mathbf{QT'} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad \mathbf{QT''} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Eq.5.2.6 Eq.5.2.7

We also introduce the first order and the second order differential rotational matrix operator

$$\mathbf{QR'} = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad \mathbf{QR''} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Eq.5.2.8 Eq.5.2.9

As is obvious $\mathbf{QR''} = \mathbf{QR'}.\mathbf{QR'}$

For brevity, we may write **Q'** or **Q"** for both translational and rotational operators. By inspection, the following applies for both cases:

$$\mathbf{A'}_i = \mathbf{Q'}.\mathbf{A}_i \text{ and } \mathbf{A''}_i = \mathbf{Q''}.\mathbf{A}_i$$

Where $\mathbf{A'}_i$ and $\mathbf{A''}_i$ are respectively first order and second order matrix derivatives of ${}^{i-1}\mathbf{A}_i$ with respect to its corresponding generalized cordinate, of course, as shown from Eq. 5.2.2 to Eq. 5.2.5.

From the above it can be seen that:

Statement 5.2.1:

The differential matrix operator $\mathbf{QT}^{(n)}$, where n denotes the order of the operator, when applied on the prismatic joint matrix $\mathbf{A}_i$ to yield the n-th order derivative $\mathbf{A}_i{}^{(n)}=\mathbf{QT}^{(n)}.\mathbf{A}_i$ requires in the sinewave representation the following actions:

for n=1:

1) Set all the 9 elements of the orientation submatrix to zero.

2) Set the x and y component of the position vector to zero; however, scale its z component to the unit length of 1, represented by a sinusoid of unit magnitude and of zero phase.

3) Set the fourth (homogeneous) scaling factor of the position vector to zero meaning that this vector now becomes a direction vector.

for n>1:

Completely set all the 16 elements of the matrix to zero as $\mathbf{A}_i{}^{(n)}$ is null-matrix identically.

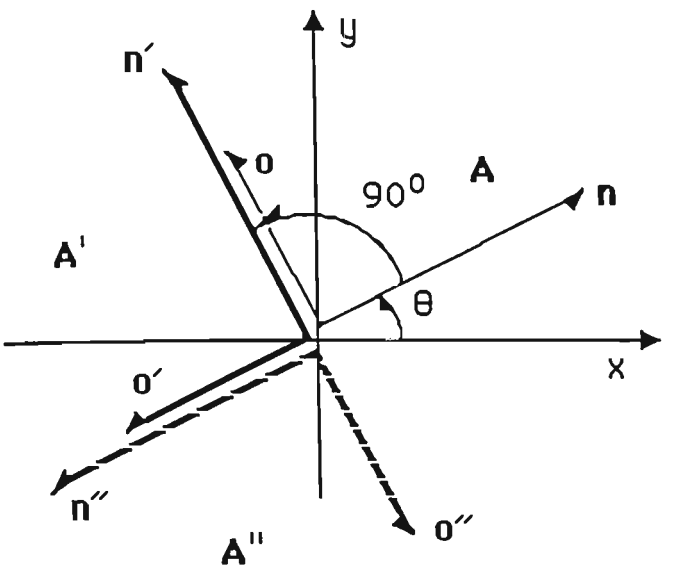


Fig. 5.2.1.a:
Relationships between
A , A' and A".

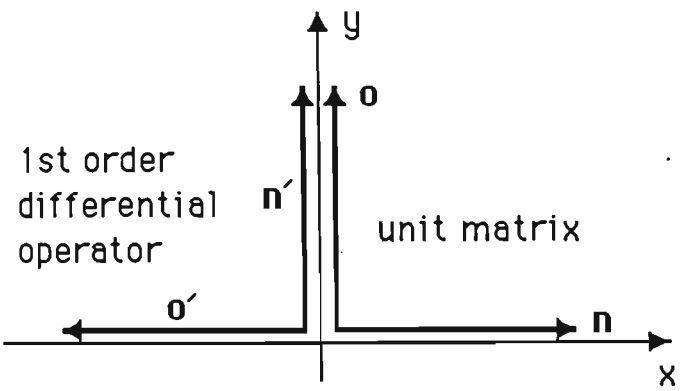


Fig. 5.2.1.b:
1st order diff. operator
as "1st order derivative"
of the unit matrix.

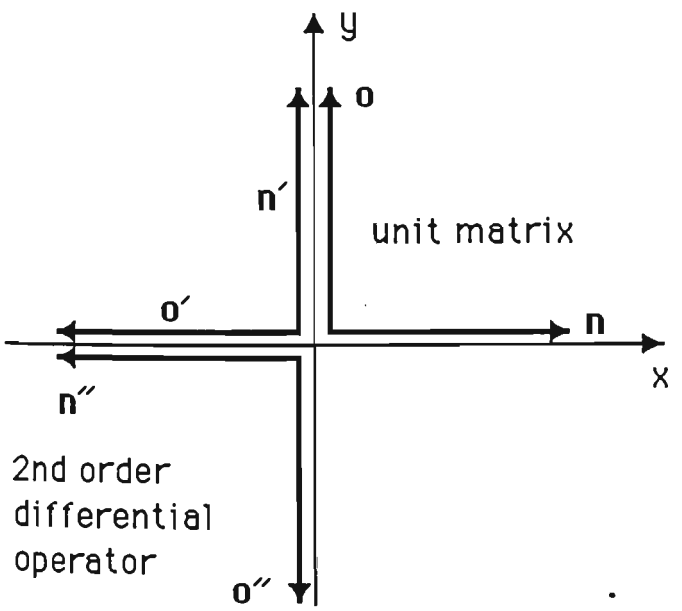


Fig. 5.2.1.c:
2nd order diff. operator
illustration

It is also seen that:

Statement 5.2.2:

The differential matrix $\mathbf{QR}^{(n)}$, where n denotes the order of the operator, when applied on the revolute joint matrix $\mathbf{A}_i$ to yield the n-th order derivative $\mathbf{A}_i^{(n)}=\mathbf{QR}^{(n)}.\mathbf{A}_i$ requires in the sinewave representation the following actions:

1) Introduce an additional phase angle of $+n.90^0$ to the sinusoidal signals representing the xy-components of the column vectors **n**, **o**, **a** and **p** embedded in that matrix.

2) Set all the z-components of the vectors to zero.

3) Set the fourth (homogeneous) scaling factor of the position vector to zero meaning that this vector now becomes a direction vector.

The first action is the only non-trivial action whose necessity is demonstrated in Fig. 5.2.1.a. This figure is self explanatory. The second action is obvious, because there are no changes at all in the z-direction, considering that we are dealing with an infinitesimal rotation about the z-axis. The third action is a consequence of the mathematical formulation of the homogeneous transformation matrices.

Fig 5.2.1.b and Fig.5.2.1.c demonstrate the obtaining of Eq.5.2.8 and Eq.5.2.9 using the concept of the phase-shifted sinewaves.

As is obvious, we will be only interested in the first and second order derivatives.

We recall that for the matrix composition

$$^{o}\mathbf{T}_p = {}^{o}\mathbf{A}_1 . {}^{1}\mathbf{A}_2 \ldots {}^{j-1}\mathbf{A}_j \ldots {}^{p-2}\mathbf{A}_{p-1} . {}^{p-1}\mathbf{A}_p$$

we have the following relationships for the first order partial derivative of $^{o}\mathbf{T}_p$ with respect to the j-th generalized coordinate, denoted by $\mathbf{U}_{pj}$:

$$\mathbf{U}_{pj} = {}^{o}\mathbf{A}_1 . {}^{1}\mathbf{A}_2 \ldots {}^{j-1}\mathbf{A'}_j \ldots {}^{p-2}\mathbf{A}_{p-1} . {}^{p-1}\mathbf{A}_p \quad \text{for } j=<p$$

$$\mathbf{U}_{pj} = 0 \quad \text{for } j>p \qquad \text{Eq.5.2.10}$$

The second order partial derivative of $^{o}\mathbf{T}_p$ with respect to the j-th and the k-th generalized coordinate, denoted by $\mathbf{U}_{pjk}$, is written as:

$$\mathbf{U}_{pjk} = {}^{o}\mathbf{A}_1 . {}^{1}\mathbf{A}_2 \ldots {}^{j-1}\mathbf{A'}_j \ldots {}^{k-1}\mathbf{A'}_k \ldots {}^{p-1}\mathbf{A}_p \quad \text{for } j,k=<p$$

$$\mathbf{U}_{pjk} = {}^{o}\mathbf{A}_1 . {}^{1}\mathbf{A}_2 \ldots {}^{j-1}\mathbf{A''}_j \ldots {}^{p-1}\mathbf{A}_p \quad \text{for } j=<p \ ; \ j=k$$

$$\mathbf{U}_{pjk} = 0 \quad \text{for } \max(j,k)>p \qquad \text{Eq.5.2.11}$$

Where it is worth mentioning that $\mathbf{U}_{pjk}=\mathbf{U}_{pkj}$

It is obvious that a matrix composition involving a matrix derivative will result in a partial derivative and that the

position vector of the T-matrix which is used to pre-multiply a partial matrix derivative will not contribute to the resultant matrix derivative. Therefore, partial matrix derivatives can be implemented using sinewave simulation as shown in Fig 5.2.2 where we are only interested in the "direct" manipulation path of the Robot Joint Descriptors. In particular, for the example of this figure, the revolute RJD ${}^{2}\mathbf{A}_{3}$ is accommodated by an additional phase shift angle of 90^{0}, as stated by Statement 5.2.2 to provide derivatives in the sense of Eq.5.2.4. The result of Eq.5.2.10 is obtained at the end outputs of the RJD-path.

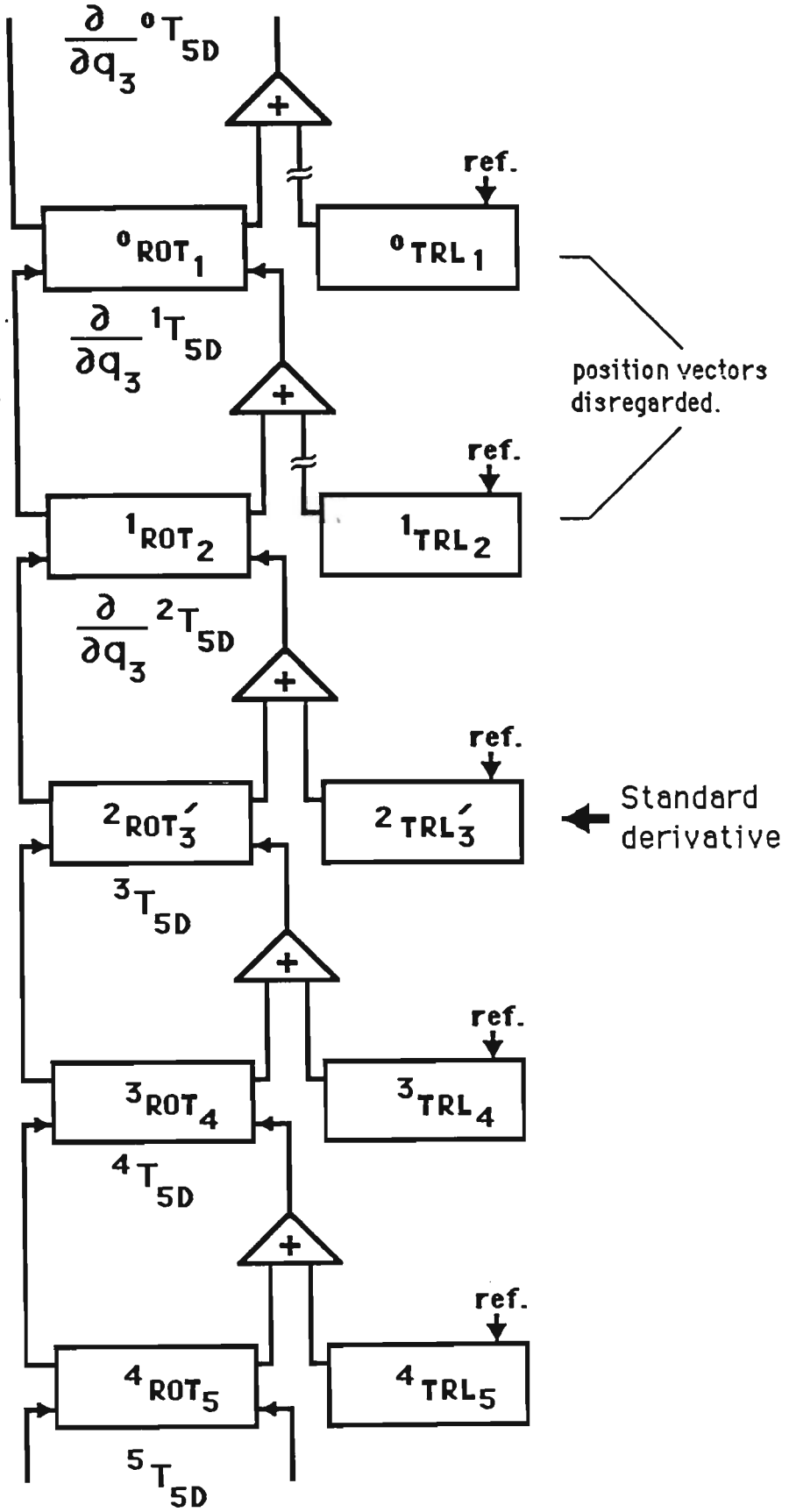


Fig.5.2.2: "Direct" first order derivatives.

5.3 THE SINUSOIDAL MATRIX PROCESSOR (SMP)

The LE-dynamic formulation can be found, for example, in [5,14,19]. For convenience of reference the dynamic formula [5] making use of the (4x4) transformation matrices and capable of describing the motion of a robot arm with n d.o.f. is given below:

$$F_i = \sum_{j=1}^{n} D_{ij}\,\ddot{q}_{ij} + \sum_{j=1}^{n}\sum_{k=1}^{n} D_{ijk}\,\dot{q}_j\dot{q}_k + D_i \quad ; \; i=1,\ldots,n \qquad \text{Eq.5.3.1}$$

In this formula, F_i represents the generalized force acting at joint i and the q_i and q_k are the j-th and k-th generalized coordinate, respectively. In this formula, their velocity and acceleration are present explicitely as can be seen. The F_i could not be computed in real-time, mainly because the coefficients D_{ij}, D_{ijk} and D_i are not constant quantities, but complicated functions of the varying q_i for i=1,2,...,n. They are:

$$D_{ij} = \sum_{p=\max(i,j)}^{n} \mathrm{Trace}(\mathbf{U}_{pj}\,\mathbf{J}_p\,\mathbf{U}^T_{pi}); \quad \mathbf{U}_{pj} = \delta / \delta q_j \; {}^{o}\mathbf{T}_p \qquad \text{Eq.5.3.2}$$

$$D_{ijk} = \sum_{p=\max(i,j,k)}^{n} \mathrm{Trace}(\mathbf{U}_{pjk}\,\mathbf{J}_p\,\mathbf{U}^T_{pi}); \quad \mathbf{U}_{pjk} = \delta^2 / \delta q_j \delta q_k \; {}^{o}\mathbf{T}_p \qquad \text{Eq.5.3.3}$$

$$D_i = \sum_{p=i}^{n} -m_p . \mathbf{g}^T . \mathbf{U}_{pi} . \mathbf{r}_p \qquad \text{Eq.5.3.4}$$

where:

m_p : Mass of link p

$\mathbf{r}_p$: Mass centre vector of link p with respect to link p coordinate frame, dimensioning (4x1) with the last element being 1.

$\mathbf{g}$: Gravity vector dimensioning (4x1) with the last element being 0.

$\mathbf{J}_p$: The p-th (4x4) symmetric pseudo inertia matrix.

A simple inspection (Appendix 8) shows that the computation of the first order and second order matrix derivatives $\mathbf{U}_{pj}$ and $\mathbf{U}_{pjk}$ claims the substantial part of the total computing time in arriving at the result for the F_i, given the desired motion of the robot arm. Therefore, it would be a substantial and potential time saving if the computing costs of n^4 and n^3 claimed by those matrix derivatives could be eliminated entirely. Indeed, for this purpose, we derive below a systematic approach to set up the RJDs in such a way that those partial matrix derivatives are provided. As is already apparent at this stage, there will be no conventional arithmetic expressions involved in obtaining those derivatives. In the first step, let us classify the p-class derivatives as all possible partial derivatives of the matrix composition $^{o}\mathbf{T}_p$. Thus, any $\mathbf{U}_{pj}$ or $\mathbf{U}_{pjk}$ belongs to the p-class derivatives. Let us first consider the first order

partial derivatives. We have to vary p=1 to n step +1 in order to obtain all possible $\mathbf{U}_{pj}$. Now given a class number p, the $\mathbf{U}_{pj}$ can be written as

$$\mathbf{U}_{pj} = \prod_{l=p}^{1} \{\mathbf{Q'}_l.\mathbf{A}_l\}$$

where

$$\mathbf{Q'}_l.\mathbf{A}_l = \begin{cases} \mathbf{A}_l & \text{for } l<>j \\ \mathbf{A'}_l & l= j \end{cases}$$

A graphical representation of this is shown in Fig. 5.3.1. Staying with the first order partial derivatives in this figure, the l-axis states how many elementary A-matrices we have in a particular matrix composition whereby the maximum number of each composition starts by one and, coming from one composition to the next adjacent composition, increases by one up to p. The j-axis indicates which matrix in the j-th composition is a standard derivative so that the composition produces the required partial derivative.

$$\mathbf{U}_{pj} = \mathbf{U}_{jj}.{}^{j}\mathbf{T}_p \qquad \text{for } 1=<j=<p \qquad \text{Eq.5.3.5}$$

Thus, in the (j,l)-plane we have matrices of the form:

$$\mathbf{A}_l^{(j)} = \begin{cases} \mathbf{A}_l & \text{for } l < j \\ \mathbf{A}'_l & l = j \end{cases} \qquad 1, j < p \qquad \text{Eq.5.3.6}$$

By inspection, all the $\mathbf{U}_{pj}$ with j>p are absent as desired.

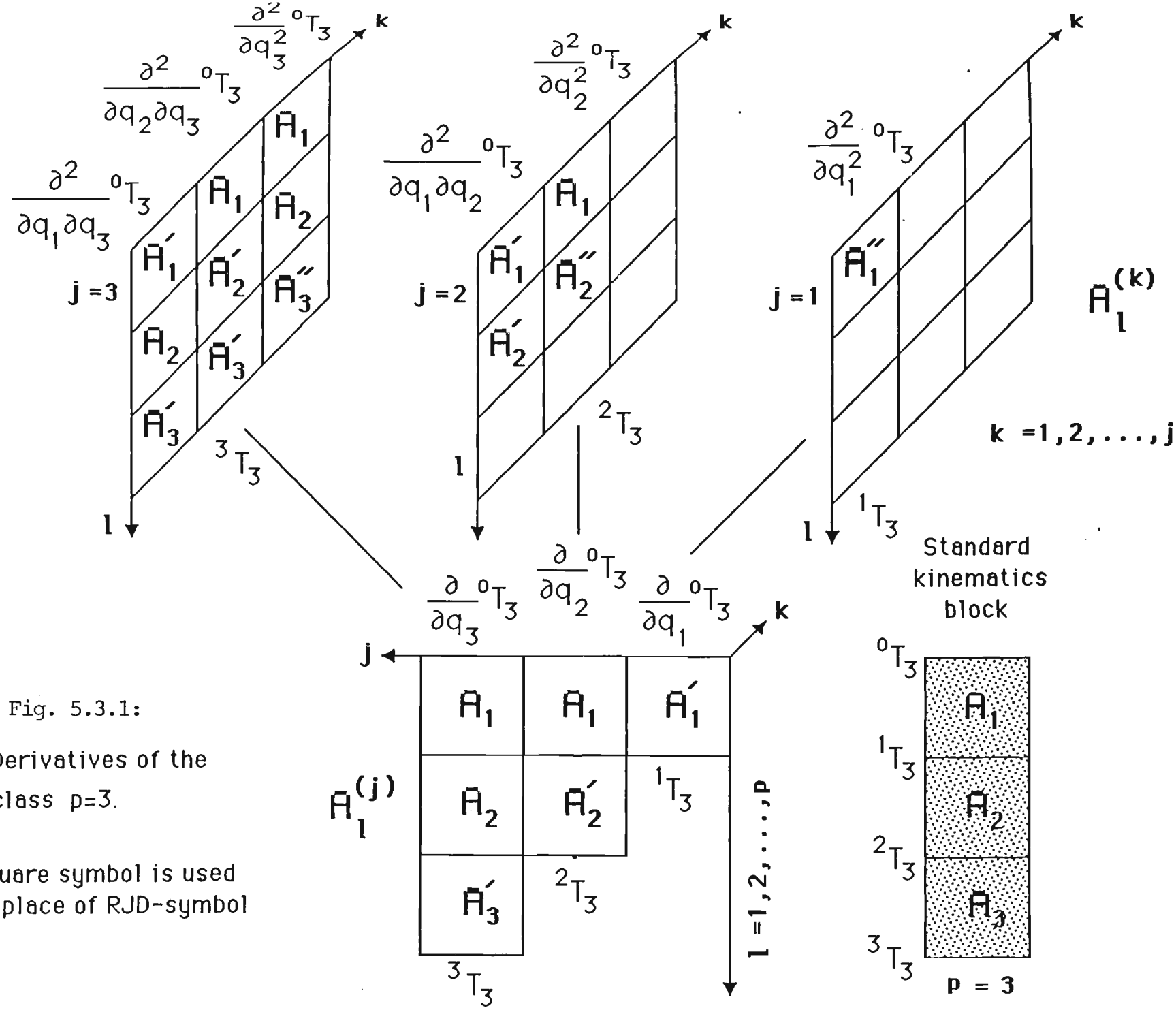


Fig. 5.3.1:

Derivatives of the class p=3.

Note: Square symbol is used in place of RJD-symbol

If we keep j constant and add the k-axis to the diagram of Fig. 5.3.1 we are in the position to systematically establish the second order partial derivatives needed in the LE dynamic equations simply by varying the index k. This is done as follows: In the (l,k)-plane, we first write all the matrices $\mathbf{A}_l^{(k)}$ in the diagram starting by k=1 to j step +1.

$$\mathbf{A}_l^{(k)} = \begin{cases} \mathbf{A}_l & \text{for } l \neq k \\ \mathbf{A}'_l & l = k \end{cases} \qquad l,k < j \qquad \text{Eq.5.3.7}$$

The indices k and l are varied from k,l=1 to j step +1. There will be a square diagram of dimension j^2 on the (l,k)-plane. Similarly as on the (j,l)-plane, there will be exactly a number j of the derivatives $\mathbf{A}'_i$ on the (l,k)-plane which are found in the diagonal of the diagram. However, in the last step on the (l,k)-plane, keeping j constant, it is essential that the derivative order of all the matrices $\mathbf{A}_j^{(k)}$ is increased by one. Now, we have to vary j. The same procedure above for every j is repeated until the index j has been varied from 1 to p step +1. As can easily be verified, the procedure does not consider $\mathbf{U}_{pj}$ for j>p, nor does it take $\mathbf{U}_{pjk}$ for max (j,k)>p into account, because $\mathbf{U}_{pj}=0$ and $\mathbf{U}_{pjk}=0$ in those cases. In addition, it also avoids the appearance of $\mathbf{U}_{pkj}$, since $\mathbf{U}_{pkj}=\mathbf{U}_{pjk}$.

Thus, the following derivatives are provided from the second order function blocks of the diagram:

$$\mathbf{U}_{pjk}=\mathbf{U}_{jjk}.{}^{j}\mathbf{T}_p \qquad \text{for } 1=<k=<j \qquad \text{Eq.5.3.8}$$

Fig. 5.3.1 illustrates the structure of the particular part of the complete Sinusoidal Matrix Processor (SMP) that provides all the first order and second order matrix derivatives of a particular derivative class "p". As is obvious, *the complete SMP* consists of all the classes p, for p=1 to n step +1 where we recall that n is the number of d.o.f. of the robot manipulator. From this structure, several SMP-variations can now be deduced. In the following text, matrices written within square brackets indicate the physical presence of the RJDs implementing them.

The first SMP-variation is derived by considering that all the relevant second order derivatives $\mathbf{U}_{pjk}$ are obtainable from the first order function blocks $[\mathbf{U}_{pj}]$. This is done by additionally performing differential operations on the k-th RJD of the matrix chain $[\mathbf{U}_{pj}]$ for k=1 to j step +1 as illustrated in Fig. 5.3.2. Fig 5.3.3 shows *the second SMP-variation* where all the first order $\mathbf{U}_{pj}$ and the second order $\mathbf{U}_{pjk}$ are obtained from the standard function blocks $[^{o}\mathbf{T}_{p}]$ in similar manner by additionally performing differential operations on the appropriate RJDs. *The third SMP-variation* is deduced from the second variation (see fig. 5.3.4) where the $^{o}\mathbf{T}_{p}$ are provided by disconnecting the RJDs representing $^{p}\mathbf{T}_{n}$ from the function blocks $[^{o}\mathbf{T}_{n}]$. Table 5.3.1 shows the RJD-costs for the particular variations.

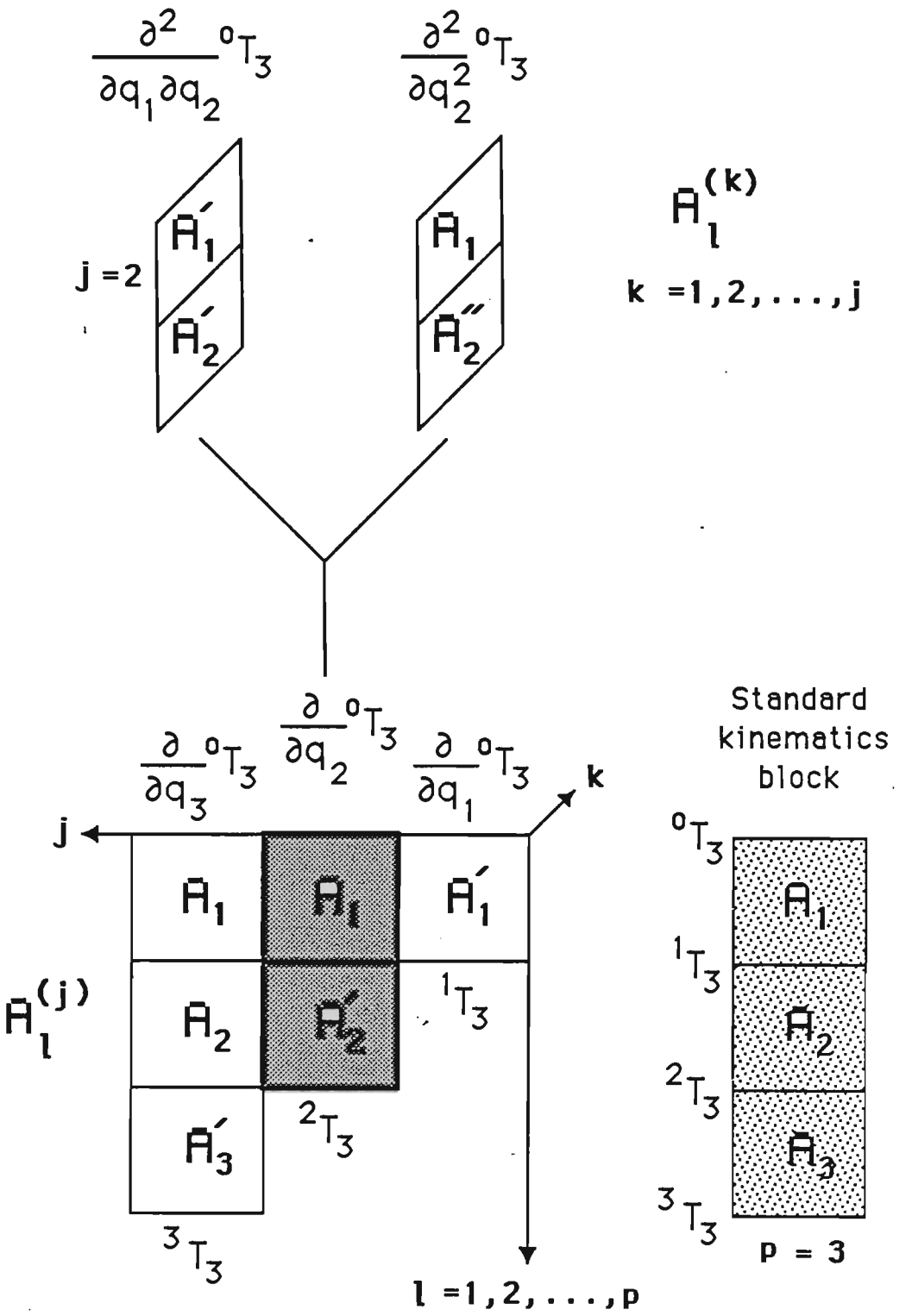


Fig.5.3.2: Second order derivatives obtainable from first order derivative function block.

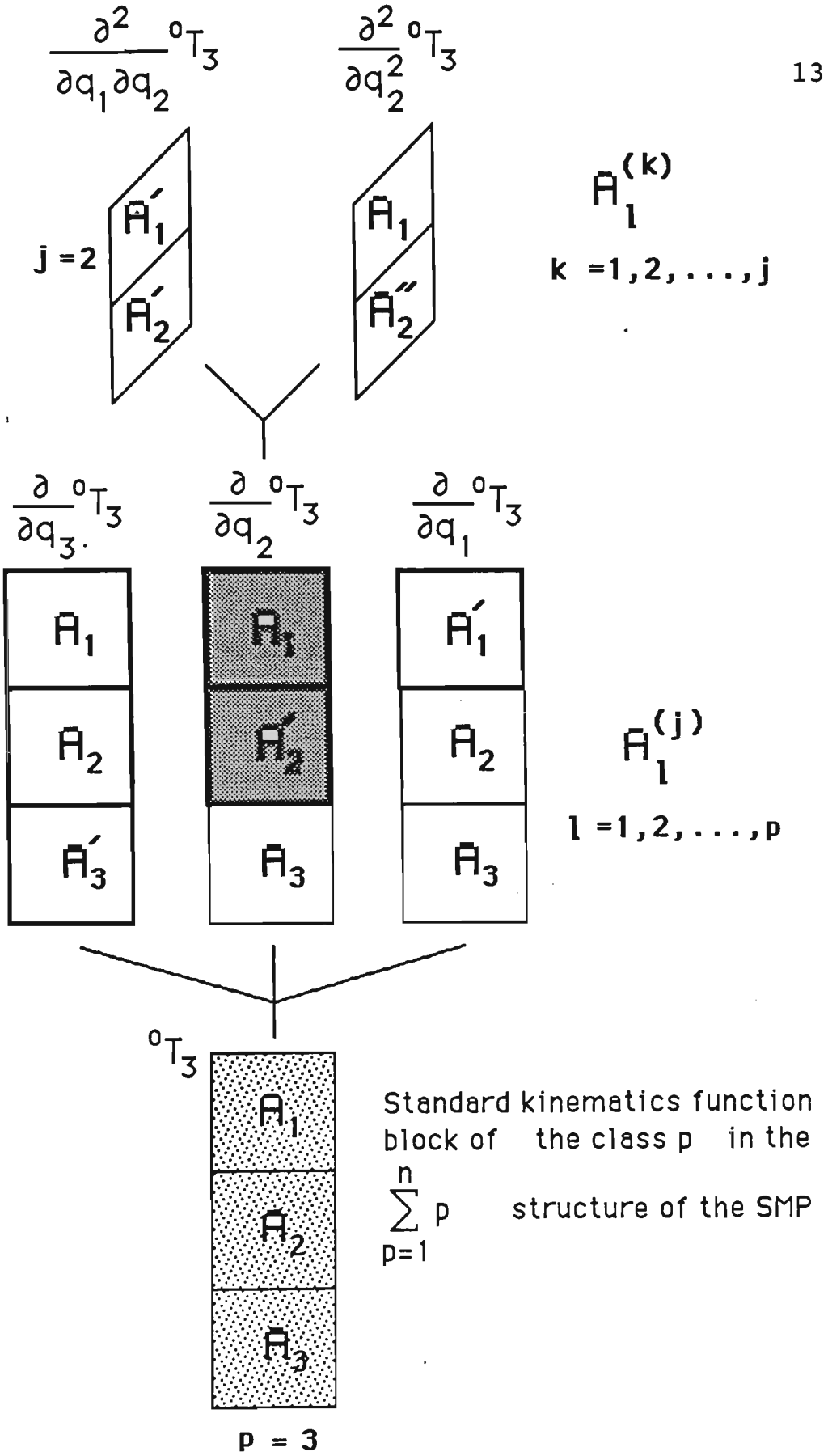


Fig.5.3.3: First order and second order derivatives obtainable from the standard kinematics function block.

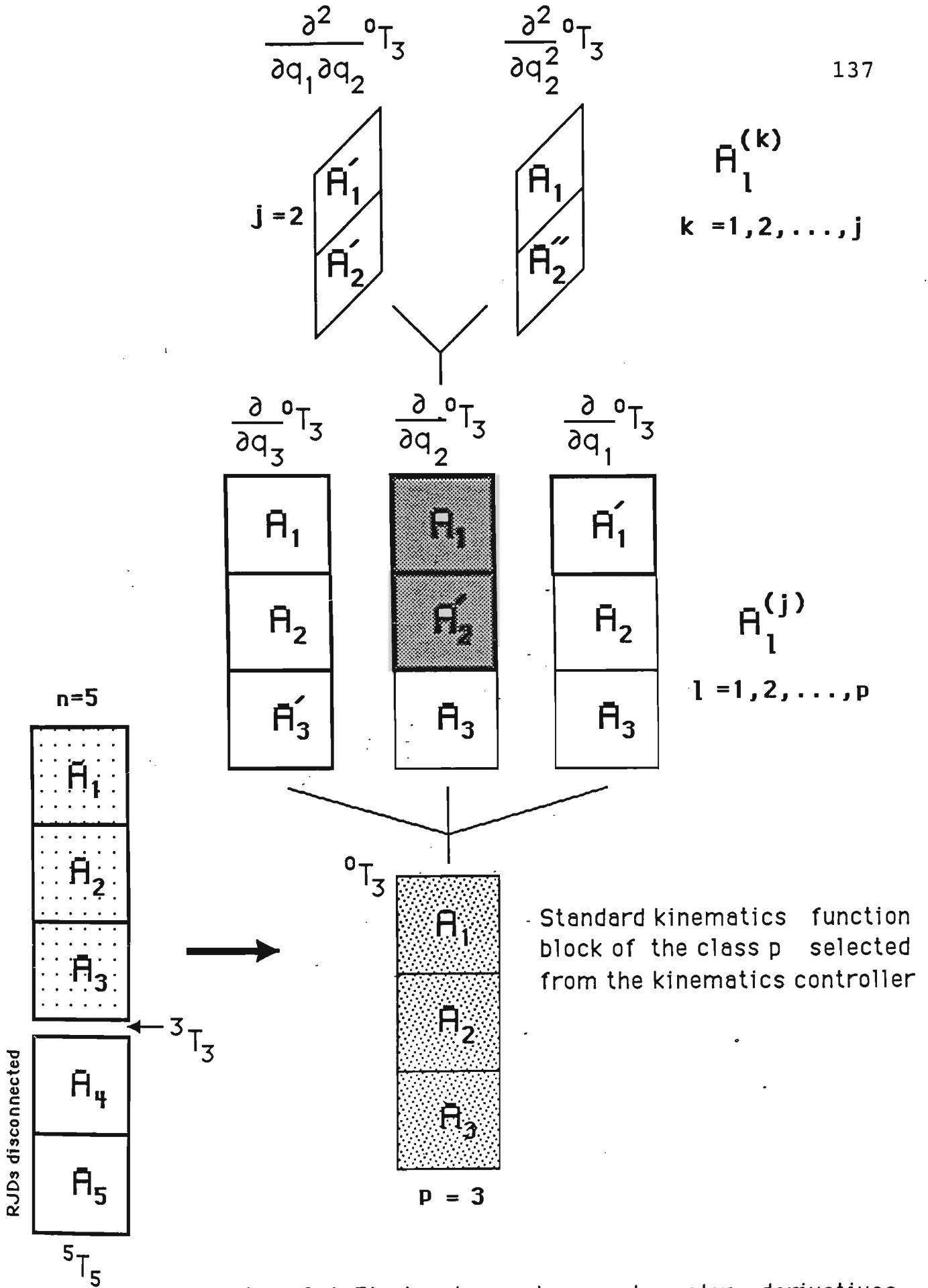


Fig.5.3.4: First order and second order derivatives obtainable from the "n structure" of the SMP

Table 5.3.1:

SMP version	Number of RJDs required as function of n	n=6
Complete	$\frac{1}{12}n^4 + \frac{1}{2}n^3 + \frac{17}{12}n^2 + n$	273
1st variation	$\frac{1}{6}n^3 + n^2 + \frac{5}{6}n$	77
2nd variation	$\frac{1}{2}n^2 + \frac{1}{2}n$	21
3rd variation	n	6

Table 5.3.2 compares the computation costs for various methods. Formulae in the SMP row are developed in Appendix 8. As is seen, for a robot arm with 6 d.o.f., the SMP method drastically reduces from 66271 to 8172 multiplications and from 51548 to 6718 additions while retaining the explicit states of the LE dynamic model.

Table 5.3.2

	Method	Multiplications	Additions
Non-Recursive	Uicker/Kahn	$32\frac{1}{2}n^4 + 86\frac{5}{12}n^3 + 171\frac{1}{4}n^2 + 53\frac{1}{3}n - 128$ **66,271**	$25n^4 + 66\frac{1}{3}n^3 + 129\frac{1}{2}n^2 + 42\frac{1}{3}n - 96$ **51,548**
	Raibert/Horn (Look up method)	$2n^3 + n^2$ **468**	$n^3 + n^2 + 2n$ **264**
	SMP	$30n^3 + 43n^2 + 24n$ **8,172** Max.	$\frac{1}{3}n^4 + 22\frac{5}{6}n^3 + 34\frac{1}{6}n^2 + 20\frac{2}{3}n$ **6,718** Max.
Recursive	Waters	$106\frac{1}{2}n^2 + 620\frac{1}{2}n - 512$ **7,051**	$82n^2 + 514n - 384$ **5,652**
	Hollerbach (4x4)	$830n - 592$ **4,388**	$675n - 464$ **3,586**
	Hollerbach (3x3)	$412n - 277$ **2,195**	$320n - 201$ **1,719**
	Newton-Euler	$150n - 48$ **852**	$131n - 48$ **738**

Note:

1.) Only the methods of Uicker/Kahn, Raibert/Horn and SMP retain the explicit-state of the dynamic model.
2.) Numbers written in bold face are computing costs for n=6; n is d.o.f. number
3.) Except for the methods of Raibert/Horn and SMP, all the other methods exhibit "negative computing costs" if one substitutes zero for n meaning that the relevant investigators have ignored some computing steps in arriving at those expressions.

Reductions from A.K. Bejczy, S. Lee "Robot Arm Dynamic Model Reduction For Control", the 22nd IEEE Conf. On Decision &Control, Dec. 1983, p.p. 1466-1476 have been taken into account in arriving at SMP computing costs.

Table from J.M. Hollerbach, "A Recursive Lagrangian Formulation of Manipulator Dynamics and a Comparative Study of Dynamics Formulation Complexity" IEEE Trans. SMC-10 (Nov. 1980), p.p. 730-736, has been adapted to include the SMP method.

5.4 SUMMARY

The inherent nature of the non-recursive dynamic equations potentially determines the structure of the SMP (i.e. there is no logical decision to make in the calculation of those equations). In fact, every one of the derivatives ${}^{o}\mathbf{U}_{pj}$ or ${}^{o}\mathbf{U}_{pjk}$ for $j,k=1,2,\ldots,p$ and $p=1,2,\ldots,n$ independently requires a number of p RJDs arranged in pipeline manner as stated by the formulas:

$${}^{o}\mathbf{U}_{pj} = {}^{o}\mathbf{A}_1.{}^{1}\mathbf{A}_2\ldots{}^{j-1}\mathbf{A}'_j\ldots{}^{p-1}\mathbf{A}_p$$

$${}^{o}\mathbf{U}_{pjk} = {}^{o}\mathbf{A}_1.{}^{1}\mathbf{A}_2\ldots{}^{j-1}\mathbf{A}'_j\ldots{}^{k-1}\mathbf{A}'_k\ldots{}^{p-1}\mathbf{A}_p$$

Because those partial derivatives ${}^{o}\mathbf{U}_{pj}$ do not depend on each other in the recursive sense, parallelism can be exploited. The same applies for ${}^{o}\mathbf{U}_{pjk}$. That is, the one RJD-path providing one particular partial derivative does not have to wait for the result of any other one, but indeed, all of those RJD-paths can provide the derivatives concurrently if one is prepared to employ an appropriate number of RJDs.

As is apparent, the SMP primarily implements sinewave simulation and requires the generalized coordinates $q_i(t)$ for $i=1,2,\ldots,n$ as the only control variables at any time point t. It provides in turn the partial derivatives mentioned above for that particular time instance t without involving any conventional arithmetic expressions.

The most powerful version of the SMP requires a number of RJDs of the order n^4. However, this RJD-number can be drastically reduced to the linear order n at the cost of more sophisticated control.

In this sense, we have presented several variations of the SMP. The associated number N of RJDs needed in the particular variations are as shown in Table 5.3.1. Using any one variation of the SMP, the substantial computing cost of n^4 and n^3 additions and multiplications in the calculation of $\mathbf{U}_{pj}$ and $\mathbf{U}_{pjk}$ is eliminated. The fact that no conventional arithmetic expressions are used is also useful because, for mechanisms with more than three d.o.f. the equations become so complex that it is difficult to manually derive them. The availability of the matrix derivatives on the SMP greatly simplifies the computer programming effort required to derive these equations.

The detailed hardware implementation and the question about the actual manipulation time will be the subjects of the next chapter. As is obvious, this manipulation time arises from the sequential nature of the formulas for the $\mathbf{U}_{pj}$ and $\mathbf{U}_{pjk}$, and mainly consists of the time it takes for the sinewaves to propagate through a series of p cascaded RJDs whenever the q_i for i=1,2,...,p are being changed.

Chapter 6

THE HARDWARE OF THE SINUSOIDAL MATRIX PROCESSOR

6.1 DEVELOPMENT OF HARDWARE FOR THE RJD

In addition to the software simulation results presented in chapter 4, several hardware simulations were used to show that the basic manipulation of ${}^{o}\mathbf{T}_n = \mathbf{A}_i.\mathbf{A}_2...\mathbf{A}_{n-1}.\mathbf{A}_n$ can be performed without involving any conventional arithmetic expressions and that the inverse kinematic problems can be solved using sinewave simulation.

Initially, several analogue versions of the RJD were built using the general purpose delay line SAD-1024-A from Analogue Devices as a basic phase shift element [31]. With this, however, the first version of the RJD faced four fundamental limitations:

1) Poor analogue magnitude resolution. That is, the SAD-1024-A allowed a signal amplitude of only 1 Volt.

2) The signal time delay was not a linear function of its control variable: the clock frequency f_c of the delay line. The functional relationship was:

$$\text{Delay} \; \alpha \; \frac{1}{f_c}$$

3) Accuracy and drift problems associated with most analogue techniques.

4) Resolution and solution time was limited by the maximum clock frequency f_c of the SAD-1024-A.

Ref032 and Ref033 showed that using conventional analogue circuitry, the above limitations could not be compensated for, unless the basic phase shift element was redesigned. Consequently, a dedicated phase shift version using digital circuitry was constructed[34]. This avoided the first three problems entirely and gave the designer greater control over the fourth.

The first all digital RJD implemented overcame accuracy and drift problems and could linearly adjust the phase shift angles in both positive and negative directions on the time axis. The first digital version also successfully demonstrated the fundamental functions of phase-shifting the operating sinewaves for three-dimensional problems.

Based on this digital design, a computer controlled SMP of the third variation was built. This was used to find partial derivatives and is described in detail in Ref045.

This chapter summarises the design of this SMP version and the results obtained with it.

6.2 IMPLEMENTATION OF COMPUTER CONTROLLED DIGITAL SMP

The realization of an ideal phase shifter represents one of the most important aspects of the RJD-technique as the success of the whole concept will rely on its operation. It must be "ideal" in the sense that 1) for any phase angle, the phase shifter should not introduce any kind of distortion on the operating sinewaves and 2) that the phase angles or equivalently the time delays should be continuously and linearly adjustable in both positive and negative directions. The digital technique offers a means of realizing such a phase shifter.

The digital phase shifter operates by first storing one cycle of the (sampled and quantized) incoming sinewave S(X), for X=0 to 2^m-1, in a 2^m Random Access Memory (RAM) chip where m is the number of the RAM's address bits, and then reading the chip with an address incremented by an amount (here called OFFSET) corresponding to the desired phase angle. The data of R(X)=S(X-OFFSET) is produced whereby the address overflow and underflow have the effect of phase shifting S(X) resulting in the "phase shifted sinewave" R(X).

Figure 6.2.1 shows the overall system block diagram of the SMP of the third variation. It consists of the following particular function blocks:

- The RJDs themselves that make use of the digital phase shifters mentioned above.

- The Master Timing and Control function block provides the m-bit address bus and the control bus necessary for the basic control and timing actions to be taken in the system.

- The computer interface that makes the joint variables of the RJDs adjustable from a host computer. Through the interface, data which represent the matrix elements manipulated by the RJDs can be read back to the computer when available on the r-bit data bus.

As can be seen from this figure, the system does have modular structure as far as the set-up of the RJDs is concerned. The data flow is connected as required in cascade such that the data outputs of the one RJD serve as data inputs for the next adjacent one. The input data for the n-th RJD representing the unit matrix ${}^{n}\mathbf{T}_{n}$ are arranged in the n-th RJD itself whereby the initial reference sinewaves are prepared in EPROMs (Eraseable and Programmable Read Only Memory) rather than RAMs. This modular or parallel structure also indicates that the RAM-addresses in the RJDs are clocked simultaneously every "machine cycle". However, as is obvious, the sinewave samples are fed through the system in sequential manner. This limits the system response time to the time it takes to stop erroneous data being clocked through the system whenever the external joint

variables are being changed. This time is critical to the SMP performance and so is discussed in detail in the following section.

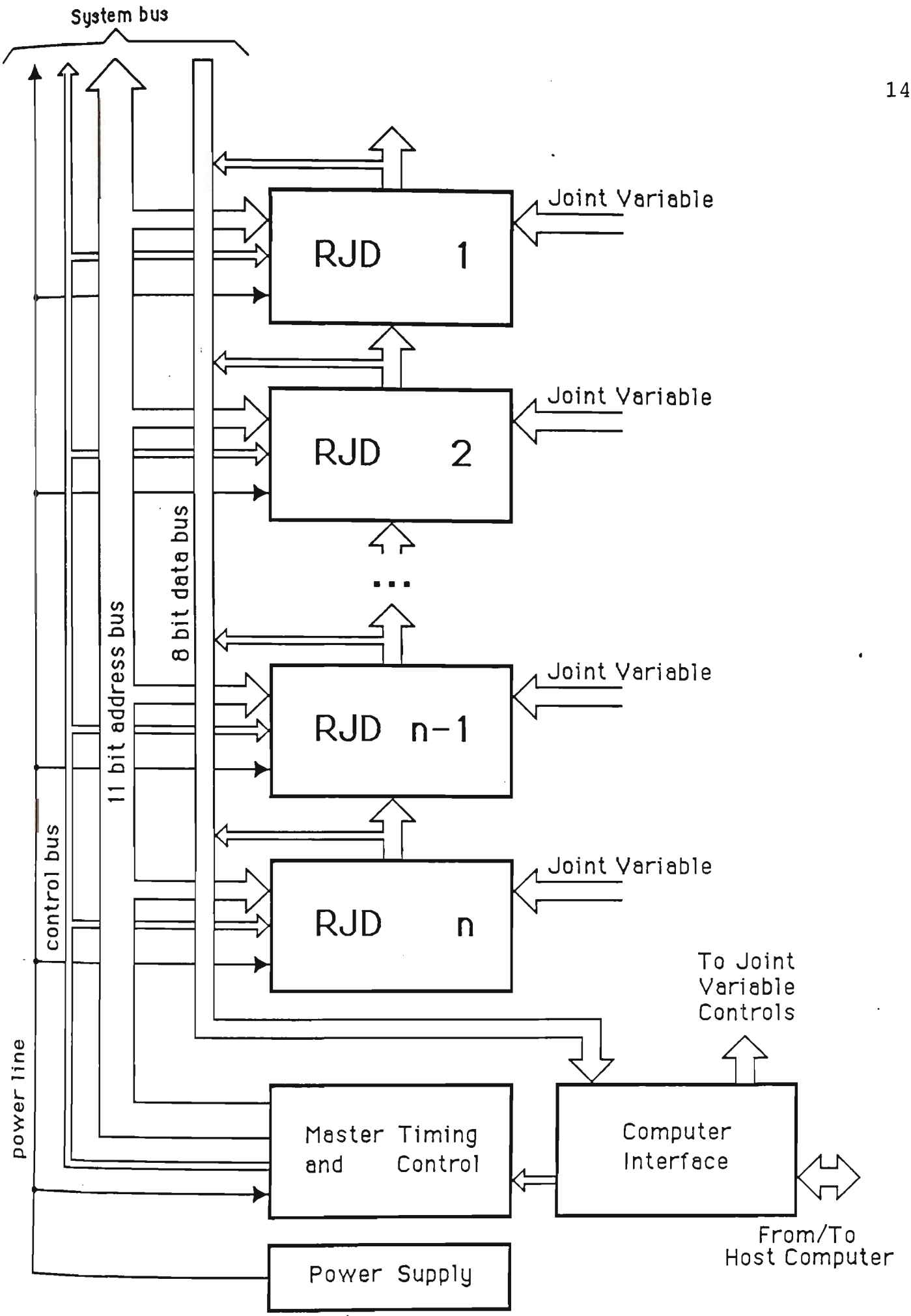

System bus
RJD 1
Joint Variable
RJD 2
Joint Variable
...
RJD n-1
Joint Variable
RJD n
Joint Variable
8 bit data bus
11 bit address bus
control bus
power line
To Joint Variable Controls
Master Timing and Control
Computer Interface
From/To Host Computer
Power Supply

Fig.6.2.1

6.3 SYSTEM MANIPULATION TIME

The SMP system performs two main distinct types of operation.

The first type involves additions (or subtractions). These operations are asynchronous and make use of passive combinational logic to provide the results before the end of the write half machine cycle. Because these operations take place within one machine cycle, no propagation delay is seen by the remainder of the system.

The second type of operation is the phase shift operation where the action of reading data from and writing data into the RAMs takes place synchronously every machine cycle. The most significant delays in the RJD system are caused by these phase shift operations because the data are fed through the system in a pipeline manner.

The basic formula for the period of the sinewave carriers is:

$$T_p = T_M.2^m$$

where

T_M = Machine cycle time

($T_M = 1/f_c$ where f_c is the system clock frequency)

m = Address bit number determining angle resolution.
At present m=11

T_P = Period of carrier.

The address offset representing the phase shift angle θ to be introduced to the sinewaves is presented in 11 bit (m=11) two's complement value such that the rotate function block has a time delay given by:

$T_D = k.T_M$ for $0 =< k =< 2^{m-1}-1$

$T_D = k.T_M + 2^m.T_M$ for $-2^{m-1} =< k < 0$

where k is delay angle in multiples of angle resolution.

The time it takes for the sinewaves to propagate through one RJD-stage is then three sinewave periods in the worst case as shown in Appendix 2.

The SMP requires one set of control variables and one set of input data, and provides in turn one set of manipulated data representing the matrix derivatives of interest. In particular, the x, y and z components of each vector **n**, **o** and **p** are provided from the SMP individually and can be selected to appear on the 8 bit system data bus.

The system timing then determines the design of the computer interface of Fig. 6.2.1. A detailed description of this interface is presented in Ref045. This interface was connected to an Apple IIe which allowed the software control of the SMP using Pascal and Assembly Language.

In particular, the computer set the joint variables in terms of integer offsets and sent them to the relevant RJDs of the SMP (output operation). The phase shifts were then performed by the SMP independently of the computer. The results at the outputs of the SMP phase shift system representing the matrix derivatives were then selected and read by the computer.

The "worst case" time it takes for the sinewaves to propagate through one RJD-stage, as shown in Appendix 2 is

$$C = 3.T_p$$

The carrier period T_p in terms of machine cycle T_M is:

$$T_p = 2^m.\ T_M$$

where m is address bit resolution and $T_M = 1/f_c$ with f_c being the system clock frequency.

The "worst case" assumes a change of consecutive values of the relative rotation angles of 360^0.

The manipulation time for the various variations of the SMP are of interest. The manipulation time for the complete SMP of Fig. 5.3.1 is C.n as all the U_{pj} and U_{pjk} are provided at the same time. This linear order C.n is provided at the cost of order n^4 RJDs involved in this complete version as shown in Table 6.3.1.

Table 6.3.1

SMP version	Number of RJDs required as function of n	n=6	Manipulation time
Complete	$\frac{1}{12}n^4 + \frac{1}{2}n^3 + \frac{17}{12}n^2 + n$	273	$\sim C.n$
1st variation	$\frac{1}{6}n^3 + n^2 + \frac{5}{6}n$	77	$\sim C.n^2$
2nd variation	$\frac{1}{2}n^2 + \frac{1}{2}n$	21	$\sim C.n^3$
3rd variation	n	6	$\sim C.n^4$

This table compares the manipulation times for the three variations derived in the previous chapter. It also illustrates the trade-off between manipulation speed and amount of hardware employed.

In particular, for the first SMP variation of Fig.5.3.2 where all $\mathbf{U}_{pj}$ are directly available, in order to obtain $\mathbf{U}_{pjk}$, additional differential operations must be performed on the k-th RJD of the path providing $\mathbf{U}_{pj}$. The manipulation time of $C.n^2$ then arises from the following number of operations applied to the SMP which states how often the variables are adjusted on the same hardware:

$$\sum_{k=1}^{j} k$$

For the second SMP variation of Fig 5.3.3 where all $\mathbf{T}_p$ are directly available, the manipulation time of $C.n^3$ arises from:

$$\sum_{j=1}^{p} \sum_{k=1}^{j} k$$

The third SMP variation of Fig. 5.3.4 requires the manipulation time of $C.n^4$ which arises from:

$$\sum_{p=1}^{n} \sum_{j=1}^{p} \sum_{k=1}^{j} k$$

The formulae from Table 6.3.1 give a numerical comparison of the manipulation time for n=6 as a function of the bit

resolution m and the clock frequency f_c. As can be seen from Table 6.3.2 for real-time applications, the third SMP-variation will not have much practical use for f_c=5MHz or lower. However, even the order of about one second manipulation time of this *(sequentially processing)* third variation (m=11, f_c=2.5 MHz) is considerably lower than the order of several seconds required by a *(sequential digital)* computer when conventional arithmetic calculations are involved. The decisive reductions of the processing time come about from the exploitation of parallelism, particularly in the complete SMP-version, and from the innovative concept of abolishing the use of conventional arithmetic expressions.

The hardware for the third SMP-variation also confirmed that the software control (PASCAL and assembly language) only needed to be simple in nature as most of the complex control functions were performed on the SMP-hardware [45].

Table 6.3.2: Worst case manipulation time in milliseconds (n=6)

m	f_c / MHz	complete SMP	1st var.	2nd var.	3rd var.
8	2.5	1,84	6,45	27,96	135,48
	5	0,92	3,23	13,98	67,74
9	2.5	3,69	12,90	55,91	270,95
	5	1,84	6,45	27,96	135,48
10	2.5	7,37	25,80	111,82	541,90
	5	3,69	12,90	55,91	270,95
11	2.5	14,75	51,61	223,64	1083,80
	5	7,37	25,80	111,82	541,90

6.4 SUMMARY

The digital phase shifter operates by first storing one cycle of the incoming sinewave in the Random Access Memory (RAM) chip, and then reading the chip with an address incremented by an amount corresponding to the desired phase shift angle. It represents the fundamental element of the hardware implementing the sinewave concept.

The principles guiding the construction of hardware for the computer controlled SMP were detailed, and the relationship between resolution and clock frequency for the complete SMP and its three variations discussed and tabulated in Tables 6.3.1 and 6.3.2.

The SMP-prototype showed that relevant derivatives could be obtained, without involving any conventional arithmetic expressions, extremely rapidly.

The times detailed in Table 6.3.2, and confirmed for the particular SMP implemented in hardware, show that the SMP method is capable of providing all relevant derivatives sufficiently rapidly to allow practical real-time control of 6 degree of freedom manipulators. For example, from Table 6.3.2 it can be seen that an SMP of 2nd variation and 10 bit resolution will, in priciple, provide all relevant derivatives in 56 msecs.

Chapter 7

SUMMARY AND CONCLUSIONS

7.1 Conclusions

In this thesis is presented a new method for manipulating the matrices involved in the kinematic and dynamic descriptions of robot manipulators. This method offers some conceptual and practical advantages in kinematics descriptions, and is of particular use in dynamics in cases where it is desirable for two major requirements to be satisfied simultaneously:

1) Real-time computing of the dynamic equations involved in the control scheme.

2) Retaining the explicit-state dynamic model of the robot manipulator for the purpose of applying advanced control laws/strategies.

Chapter 2 presented an approach using phase-shifted sinewaves to exploit the partial orthonormality of the "A-matrices" in arriving at the results for their chain product ${}^{o}\mathbf{T}_{nD} = \mathbf{A}_1 \mathbf{A}_2 \ldots \mathbf{A}_n$ without involving conventional arithmetic expressions. The exploitation of the partial orthonormality inevitably led to the sub-partitioning representation of the general homogeneous transformation matrix "T" as follows:

$$\mathbf{T} = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{ROT} & \mathbf{TRL} \\ 0 & 1 \end{bmatrix}$$

with **ROT** being of dimension (3x3) and orthonormal.

This representation, in turn, led to the following expressions for the intermediate results:

$$^{i-1}\mathbf{ROT}_{nD} = {}^{i-1}\mathbf{ROT}_{i} . {}^{i}\mathbf{ROT}_{nD} \qquad \text{Eq.2.2.7}$$

$$^{i-1}\mathbf{TRL}_{nD} = {}^{i-1}\mathbf{ROT}_{i} . {}^{i}\mathbf{TRL}_{nD} + {}^{i-1}\mathbf{TRL}_{i} \qquad \text{Eq.2.2.8}$$

for i = n,n-1,...,2,1

designated as the "direct equations"

The intermediate results of the "inverse" multiplication

$$^{n}\mathbf{T}_{nI} = \mathbf{A}_n^{-1} \; \mathbf{A}_{n-1}^{-1} \; \ldots \; \mathbf{A}_2^{-1} \; \mathbf{A}_1^{-1} . \; {}^{o}\mathbf{T}_{nI}$$

(where $^{o}\mathbf{T}_{nI}$ is pre-specified and may have any value)

were found as:

$$^{i}\mathbf{ROT}_{nI} = (^{i-1}\mathbf{ROT}_{i})^{-1} . \; {}^{i-1}\mathbf{ROT}_{nI} \qquad \text{Eq.2.2.9}$$

$$^{i}\mathbf{TRL}_{nI} = (^{i-1}\mathbf{ROT}_{i})^{-1} . \quad (^{i-1}\mathbf{TRL}_{nI} - ^{i-1}\mathbf{TRL}_{i}) \qquad \text{Eq.2.2.10}$$

for $i = 1,2,\ldots,n-1,n$

These equations constituted the RJD symbolically shown in Fig. 2.2.1 which has a fundamental importance in the new matrix manipulation method.

In the sinewave implementation, the RJD of this figure was realized by phase shifters so that the basic equations for the manipulation of the T-matrices in both direct and inverse direction were implemented simply by cascading an appropriate number of phase shifter stages and so matrix multiplications were performed without using any conventional arithmetic expressions.

This basic manipulation was initially restricted to two-dimensional problems due to sinewaves being capable of carrying only two pieces of information: the magnitude and the relative phase.

However, using the same concept of phase-shifting the orthonormal sinewave references, modulated by the relevant matrix elements, the expansion of the 2D-RJD into three dimensions was presented in Chapter 3.

In particular, the elementary A-matrix was implemented in three dimensions according to the following equation:

$$\mathbf{A}=ROT_z(\theta).Trans_z(d).Trans_x(a).ROT_x(\alpha) \qquad \text{Eq.3.1.1.b}$$

where the four particular matrix multiplicands represented one or two dimensional problems such that the A-matrix was realized, in essence, by a combination of sinewaves to be manipulated. That is, $ROT_x(\alpha)$ would not affect the sinewaves representing the x-components which, however, would be manipulated by the phase shifters representing $ROT_z(\theta)$. Thus, the 3D-RJD was then presented in Fig.3.4.4.

Chapter 2 and Chapter 3 had made use of the cascade characteristic of the general RJD. However, Chapter 4 inevitably arose from the balance characteristic of the RJD.

This chapter discussed the use of RJDs in finding a general, systematic solution procedure for the inverse kinematic problem. In particular, the procedure enables the systematic determination of a set of n simultaneous equations solvable for the robot's n independent joint variables without the need for an element by element comparison in a matrix equation to be performed on an intuitive basis. Two solution techniques based on the same procedure were demonstrated: 1) Arithmetic expressions for the solutions were derived. 2) Numerical solutions using sinewave simulation without using conventional arithmetic expressions for the solutions were also obtained . Solutions would also be derived even for the

general case of $\cos\alpha \neq \pm 1$ or $\cos\alpha \neq 0$. Degeneracy introduces no problem in finding the solutions.

This procedure does not provide lower computing costs than those of methods prior to its existence with the number of additions and multiplications in all methods being comparable. However, it offers a general non-intuitive means of deriving the solutions.

Chapter 5 described a new method of obtaining derivatives. It was first shown that the first order derivatives of the elementary rotational A-matrices could be obtained simply by introducing an extra phase shift of 90^0 to the relevant sinewaves. The second order derivatives were then obtained by introducing a further 90^0 to the sinewaves representing the first order derivative.

The sinusoidal matrix processor (SMP) was then defined, whose function was to provide all the relevant $\mathbf{U}_{pj}$ and $\mathbf{U}_{pjk}$ representing the first and second order derivatives respectively involved in the LE dynamic calculations. Several SMP variations were then presented and the associated numbers of RJD's needed for each variation given in Table 5.3.2.

Thus, using any one variation of the SMP, the substantial computing cost of n^4 and n^3 additions and multiplications involved in the calculation of $\mathbf{U}_{pj}$ and $\mathbf{U}_{pjk}$ is eliminated, as indicated in Appendix 8.

In Chapter 6 a hardware implementation of the SMP together with its computer control was discussed.

There is a trade off to be made between SMP complexity and time to obtain a solution. The manipulation time of an SMP is the time it takes for the sinewaves to propagate through a number of cascaded RJDs.

The times detailed in Table 6.3.2 and confirmed for the particular SMP implemented in hardware, show that the SMP method is capable of providing all relevant derivatives sufficiently rapidly to allow practical real time control of 6 degree of freedom manipulators. For example, from Table 6.3.2 it can be seen that an SMP of 2nd variation and 10 bit resolution will in principle provide all relevant derivatives in 56 msecs.

Therefore, in principle the SMP method provides the ability of real-time calculation of the inverse dynamics while retaining the explicit state structure of the dynamic model. Thus, this provides a potential method of enabling the control of robot manipulators using explicit state variable theory.

In obtaining the first order and second order partial matrix derivatives $\mathbf{U}_{pj}$, $\mathbf{U}^{T}_{pi}$ and $\mathbf{U}_{pjk}$ which are involved in the explicit state Lagrangian dynamic equations, no conventional arithmetic expressions are used. This is also useful because, for mechanisms with more than three degrees of freedom, the equations become so complex that it is difficult to manually derive them. The availability of the matrix derivatives $\mathbf{U}_{pj}$ $\mathbf{U}^{T}_{pi}$ and $\mathbf{U}_{pjk}$ on the Sinusoidal Matrix Processor greatly simplifies computer programming efforts to implement these equations.

7.2 Recommendations for Future Work

The existence of the novel method for deriving inverse kinematic solutions presented under the Chapter 4 gives rise to the idea of exploiting the "solution sequences" in kinematic control. It is recommended that the use of such solution sequences be explored for higher order kinematic problems involving velocities and accelerations.

Table 5.3.2 shows that the concept using the SMP still requires a number of multiplications and additions of the orders n^3 and n^4, respectively, even though there are small polynomial coefficients in the expressions for the computing costs in comparison with those of the other non-recursive methods. These computation costs mainly emerge from the calculations of the "Trace" operations involved in the dynamic equations. Therefore, further works should be addressed to the problem of reducing these remaining computing costs by, for example using some real-time hardware or real-time computation algorithms. It seems worthwhile studying this further since the "Trace" operations are extremely "structured" such as $\text{Trace}\{\mathbf{U}_{pj}.\mathbf{J}_p.\mathbf{U}^T_{pi}\}$ and $\text{Trace}\{\mathbf{U}_{pjk}.\mathbf{J}_p.\mathbf{U}^T_{pi}\}$. Also the (4x4) "pseudo inertia matrices" $\mathbf{J}_p$ are diagonal symetric.

A very large scale integration (VLSI) implementation of the RJD could also be investigated to further reduce computation times.

Indeed, the inherent and highly-structured nature of the Robot Joint Descriptors makes them suitable for VLSI implementation. That is, in particular, every one RJD basically consists of three main stages: Vector combiners, phase shifters and vector splitters. A VLSI implementation would, for example, reduce the chip count and increase the clock frequency f_c meaning faster manipulation time. From experimental results obtained to date it is felt that the angle resolution of m=11 or even lower would be sufficient. However, we recommend that quantitative investigations on the data and address (or angle) resolution be carried out. Those investigations would be useful in finding specifications for a VLSI implementation. In a further step of specialization, we recommend that the non-linear quantization of the relative rotation angles be explored. This recommendation arises from the idea of making use of the full m-bit resolution over one certain "working range" rather than over the full range of 360^0 of the rotation angles which would result in a higher accuracy for a lower bit number m.

Throughout the development of the SMP we have exclusively made use of sinewave signals as orthonormal carriers for the simulation of the A and T-matrices. However other non-sinusoidal carriers, for example digital square waves could be investigated. Thus, it might be worth investigating the use of a set of square wave signals as a particular set of general orthogonal carriers.

REFERENCES

Ref001: J. Denavit, R.S. HARTENBERG; 'A Kinematic Notation For Lower-Pair Mechanisms Based On Matrices', Journal of APplied Mechanics, June 1955, p.p. 215-221.

Ref002: K. JENSEN, N. WIRTH; 'PASCAL User Manual and Report', Springer-Verlag, 1974.

Ref003: T. LOZANO-PEREZ; 'Robot Programming', Proceedings of the IEEE, Vol. 71, No.7, July 1983, p.p. 821-841.

Ref004: C.S.G. LEE; 'Robot Arm Kinematics, Dynamics and Control', IEEE Computer, December 82, p.p. 62-80.

Ref005: R.P. PAUL; 'Robot Manipulator: Mathematics, Programming and Control', MIT Press, cambridge, Massachusetts, 1981.

Ref006: R.P. PAUL, B. SHIMANO, G.E. MAYER; 'Kinematic Control Equations for Simple Manipulators', IEEE Trans. SMC, Vol. SMC-11, No. 6, June 1981, p.p. 449-455.

Ref007: R.P. PAUL, B. SHIMANO, G.E. MAYER; 'Differential Kinematic Control Equations for Simple Manipulators', IEEE Trans. SMC, Vol. SMC-11, No.6, June 1981, p.p. 456-460.

Ref008: J.M.HOLLERBACH; 'A Recursive Lagrangian Formulation of Manipulator Dynamics Formulation Complexity', IEEE Trans. SMC, VOl. SMC-10, No.11, November 1980, p.p. 730-736.

Ref009: W.M. SILVER; 'On the Equivalence of Lagrangian and Newton-Euler Dynamics for Manipulators', Proceedings of the 1981 Joint Automatic Control Conference, Charlottesville, VA, paper TA-2A.

Ref010: J.Y.S.LUH, C.S.LIN; 'Automatic Generation of Dynamic Equations for MEchanical Manipulators', Proceedings of the 1981 Joint Automatic Control Conference, Charlottesville, VA, paper TA-2D.

Ref011: K.TAKASE, R.P. PAUL, E.J. BERG; 'A Structured Approach to Robot Programming and Teaching', IEEE Trans. SMC, Vol. SMC-11, No.4, April 1981, p.p. 274-289.

Ref012: V.CVETKOVIC, M. VUKOBRATOVIC; 'Computer-Oriented Algorithm for Modeling Active Spatial Mechanisms for Robotics Applications', IEEE Trans. SMC, Vol. SMC-12, No.6, November/December 1982, p.p. 838-847.

Ref013: D.E. WHITNEY; 'The Mathematics of Coordinated Control of Prosthetic Arms and Manipulators'; Journal of Dynamic Systems, Measurement and Control, December 1972, p.p. 303-309.

Ref014: E.POPOV; 'Modern Robot Engineering'; english trans. MIR Publishers, Moscow, 1982; Chapter 2 - Mathematical Models of Robot Dynamics by V.S. MEDVEDEV, p.p.23-61.

Ref015: C.S.G. LEE, M.J. CHUNG; 'An Adaptive Control Strategy for Computer-Based Manipulators', Proc. 21st IEEE Conf. Decision and Control, December 1982, p.p. 95-100.

Ref016: J.Y.S.LUH, M.W.WALKER, R.P.C.PAUL; 'Resolved-Acceleration Control of Mechanical Manipulators', IEEE Trans. AC, Vol.AC-25, No.3, June 1980,p.p. 468-474.

Ref017: G.W.WU, R.P.C. PAUL; 'Resolved Motion Force Control of Mechanical Manipulators', IEEE Trans. SMC, Vol.SMC-12, No.3, May/June 1982, p.p. 266-275.

Ref018: A.K. BEJCZY, S.LEE; 'Robot Arm Dynamic Model Reduction for Control', The 22nd IEEE Conf. on Decision and Control, San Antonio, December 1983, p.p. 1466-1476.

Ref019: M.VUKOBRATOVIC, V.POTKONJAK; 'Dynamics of Manipulation Robots, Theory and APplication', Springer Verlag, Berlin Heidelberg New York 1982.

Ref020: M.VUKOBRATOVIC, D.STOKIC; 'Control of Manipulation Robots, Theory and Application', Springer Verlag, Berlin Heidelberg New York 1982.

Ref021: S.DUBOWDKY, D.T. DesFORGES; 'The Application of Model-Referenced Adaptive Control to Robotic Manipulators'; Journal of Dynamic Systems, Measurements and Control, Vol. 101, September 1979, p.p. 225-232.

Ref022: M.BRADY, J.M. HOLLERBACH, T.JOHNSON, T.LOZANO-PEREZ, M.T. MASON; 'Robot Motion: Planning and Control'; The MIT Press, Cambridge, Massachusetts and London, 1983; Chapter 2 - Dynamics, p.p. 51-71.

Ref023: J.Y.S. LUH, M.W.WALKER, R.P.C. PAUL; 'On-line computational Scheme for Mechanical Manipulators'; Journal of Dynamics Systems, Measurement, and Control 102, 1980; p.p. 69-76.

Ref024: M.W.WALKER, D.E.ORIN; 'Efficient Dynamic Computer Simulation of Robotic Mechanisms'; Journal of Dynamics Systems, Measurement, and Control 104, 1982; p.p. 205-211.

Ref025: M.BRADY, J.M.HOLLERBACH, T.JOHNSON, T. LOZANO-PEREZ, M.T. MASON; ' Robot Motion: Planning and Control '; The MIT Press, Cambridge, Massachusetts and London, 1983; Chapter 3 - Feedback Control, p.p. 127-146.

Ref026: E. FREUND; 'Fast Nonlinear Control with Arbitrary Pole-Placement for Industrial Robots and Manipulators'; Robotics Research 1,1, 1982; p.p. 65-78.

Ref027: D.F. GOLLA, S.C.GARG; 'Linear State-Feedback Control of Manipulators'; Mech. Machine Theory 16, 1981; p.p. 93-103.

Ref028: K.D.YOUNG; 'Controller Design for a Manipulator Using Theory of Variable Structure Systems.' IEEE Trans. on Systems, Man and Cybernetics, SMC-8, 2(Feb.1978); p.p. 101-109.

Ref029: APPLE PASCAL, Operating System Reference Manual. Apple Computer Inc., 1980.

Ref030: APPLE PASCAL, Language Reference Manual. Apple Computer Inc., 1980.

Ref031: M. COMENSOLI, 'Design of analogue two-dimensional Robot Joint Descriptor', ELEC 457 Thesis, Wollongong University, Department of Electrical and Computer Engineering, 1983.

Ref032: D.McLEAN, 'Design of analogue three-dimensional Robot Joint Descriptor', ELEC 457 Thesis, Wollongong University, Department of Electrical and Computer Engineering, 1984.

Ref033: A.MOURAD, 'Investigation on Feasibility of analogue Implementation of the three-dimensional Robot Joint Descriptor', ELEC 457 Thesis, Wollongong University, Department of Electrical and Computer Engineering, 1984.

Ref034: G.SAMWAYS, 'Digital Implementation of the three-dimensional Robot Joint Descriptor based on Homogeneous Transformation Theory', ELEC 457 Thesis, Wollongong University, Department of Electrical and Computer Engineering, 1984.

Ref035: S. FORST, 'Robot Kinematics: Investigation of the Error Referenced Solving Strategy by Software Interfacing and Simulation', ELEC 457 Thesis, Wollongong University, Department of Electrical and Computer Engineering, 1984.

Ref036: R.H. LATHROP; 'Parallelism in Manipulator Dynamics'; The International Journal of Robotics Research, Vol.4, No.2, 1985; p.p. 80-102.

Ref037: J.G. NASH; 'A Systolic/Cellular Computer Architecture for Linear Algebraic Operations'; 1985 IEEE International Conference on Robotics and Automation, March 25-28, St Louis-Missouri, IEEE Computer Society Press; p.p.799-784.

Ref038: D.E.ORIN, H.H.CHAO, K.W.OLSON, W.W.SCHRADER; 'Pipeline/Parallel Algorithm for the Jacobian and Inverse Dynamics Computations'; 1985 IEEE International Conference on Robotics and Automation, March 25-28, St Louis-Missouri, IEEE Computer Society Press; p.p. 785-789.

Ref039: K.SCHWAN, T.BIHARI, B.W.WEIDE, G.TAULBEE; 'GEM: Operating System Primitives for Robots and Real-Time Control Systems'; 1985 IEEE International Conference on Robotics and Automation, March 25-28, St Louis-Missouri, IEEE Computer Society Press; p.p. 807-813.

Ref040: R.NIGAM, C.S.G.LEE; 'A Multiprocessor-Based Controller for the Control of Mechanical Manipulators'; 1985 IEEE International Conference on Robotics and Automation, March 25-28, St Louis-Missouri, IEEE Computer Society Press; p.p. 815-148.

Ref041: C.D.COOK, T. VU-DINH; 'A General Purpose Robot Controller'; The 1st International Symposium on Design and Synthesis, 11-13 July 1984, Tokyo; p.p. 143-148.

Ref042: C.D.COOK, T.VU-DINH; 'A new General Algorithm for Describing Manipulator Kinematics'; Transactions of the Institution of Engineers, Australia, Vol. ME10, No.3, p.p.169-174, Sept.85. (Paper's reference code no. M1298).

Ref043: K.C. GUPTA: "Kinematic Analysis of Manipulators Using the Zero Reference Position Description" The International Journal of Robotics Research, Vol.5, No.2, 1986. p.p. 5-13.

Ref044: R.P. PAUL, H.ZHANG; 'Computationally Efficient Kinematics for Manipulators with Spherical Wrists Based on the Homogeneous Transformation Representation' The International Journal of Robotics Research, Vol.5, No.2, 1986. p.p.32-44.

Ref045: T.VU-DINH, C.D.COOK; 'Implementation of the Sinusoidal Matrix Processor', AEAC Research Report, presented to CSIRO, Div. of Manufactoring, Melbourne, March 86.

APPENDIX 1

Proof of ${}^{m}\mathbf{T}_{n} = \mathbf{A}_{m+1}\ \mathbf{A}_{m+2}\ldots\mathbf{A}_{n-1}\ \mathbf{A}_{n}$ for n>m will yield:

$$ {}^{m}\mathbf{ROT}_{n} = \mathbf{ROT}_{m+1}.\mathbf{ROT}_{m+2} \ \ldots\ \mathbf{ROT}_{n-1}.\mathbf{ROT}_{n} \qquad \text{Eq.1} $$

$$ {}^{m}\mathbf{TRL}_{n} = {}^{m}\mathbf{ROT}_{n-1}.{}^{n-1}\mathbf{TRL}_{n} + {}^{m}\mathbf{TRL}_{n-1} \qquad \text{Eq.2} $$

The relationships Eq.1 and Eq.2 hold for the two general multiplicands ${}^{m}\mathbf{A}_{m+1}$ and ${}^{m+1}\mathbf{A}_{m+2}$ as can be shown by direct substitution:

$$ {}^{m}\mathbf{T}_{m+2} = \begin{bmatrix} {}^{m}\mathbf{ROT}_{m+1}.{}^{m+1}\mathbf{ROT}_{m+2} & {}^{m}\mathbf{ROT}_{m+1}.{}^{m+1}\mathbf{TRL}_{m+2} + {}^{m}\mathbf{TRL}_{m+1} \\ 0 & 1 \end{bmatrix} \qquad \text{Eq.3} $$

Now, let us write for ${}^{m}\mathbf{T}_{n-1}$

$$ {}^{m}\mathbf{T}_{n-1} = \begin{bmatrix} {}^{m}\mathbf{ROT}_{n-1} & {}^{m}\mathbf{TRL}_{n-1} \\ 0 & 1 \end{bmatrix} \qquad \text{Eq.4} $$

we assume

$$ {}^{m}\mathbf{T}_{n-1} = \begin{bmatrix} {}^{m}\mathbf{ROT}_{m+1}\ldots{}^{n-2}\mathbf{ROT}_{n-1} & {}^{m}\mathbf{ROT}_{n-2}.{}^{n-2}\mathbf{TRL}_{n-1} + {}^{m}\mathbf{TRL}_{n-2} \\ 0 & 1 \end{bmatrix} \qquad \text{Eq.5} $$

and find expression for ${}^{m}\mathbf{T}_{n}$.

We apply:

$$ {}^{m}\mathbf{T}_{n} = {}^{m}\mathbf{T}_{n-1} \cdot {}^{n-1}\mathbf{A}_{n} $$

$$ = {}^{m}\mathbf{T}_{n-1} \cdot \begin{bmatrix} {}^{n-1}\mathbf{ROT}_{n} & {}^{n-1}\mathbf{TRL}_{n} \\ 0 & 1 \end{bmatrix} \qquad \text{Eq.6} $$

Substituting for ${}^{m}\mathbf{T}_{n-1}$ from Eq.5 we obtain

$$ {}^{m}\mathbf{ROT}_{n} = ({}^{m}\mathbf{ROT}_{m+1} \ \ldots \ {}^{n-2}\mathbf{ROT}_{n-1}) . \ {}^{n-1}\mathbf{ROT}_{n} \qquad \text{Eq.7} $$

$$ {}^{m}\mathbf{TRL}_{n} = ({}^{m}\mathbf{ROT}_{m+1} \ \ldots \ {}^{n-2}\mathbf{ROT}_{n-1}) . {}^{n-1}\mathbf{TRL}_{n} $$

$$ + \ {}^{m}\mathbf{ROT}_{n-2} . {}^{n-2}\mathbf{TRL}_{n-1} + {}^{m}\mathbf{TRL}_{n-2} \qquad \text{Eq.8} $$

By inspection, the rotation part of ${}^{m}\mathbf{T}_{n}$, Eq.1 is proved.

Minor re-arrangement of the position part and comparison with Eq.4 and Eq.5 show that Eq.2 is also proved.

Equation Eq.2 can be modified to:

$$ {}^{m}\mathbf{TRL}_{n} = {}^{m}\mathbf{TROT}_{k} . {}^{k}\mathbf{TRL}_{n} + {}^{m}\mathbf{TRL}_{k} \quad \text{for } m<k<n \qquad \text{Eq.9} $$

This is already proved for n=k+1, so we assume:

$$ {}^{m}\mathbf{TRL}_{k+i} = {}^{m}\mathbf{ROT}_{k} . {}^{k}\mathbf{TRL}_{k+i} + {}^{m}\mathbf{TRL}_{k} \qquad \text{Eq.10} $$

and will prove:

$$^{m}\mathbf{TRL}_{k+i+1} = {}^{m}\mathbf{ROT}_{k} \cdot {}^{k}\mathbf{TRL}_{k+i+1} + {}^{m}\mathbf{TRL}_{k} \qquad \text{Eq.11}$$

From Eq.2 we have

$$^{m}\mathbf{TRL}_{k+i+1} = {}^{m}\mathbf{ROT}_{k+i} \cdot {}^{k+i}\mathbf{TRL}_{k+i+1} + {}^{m}\mathbf{TRL}_{k+i}$$

Substitiuting Eq.10 for $^{m}\mathbf{TRL}_{k+i}$ we have

$$\begin{aligned} ^{m}\mathbf{TRL}_{k+i+1} &= {}^{m}\mathbf{ROT}_{k+i} \cdot {}^{k+i}\mathbf{TRL}_{k+i+1} + {}^{m}\mathbf{ROT}_{k} \cdot {}^{k}\mathbf{TRL}_{k+i} + {}^{m}\mathbf{TRL}_{k} \\ &= {}^{m}\mathbf{ROT}_{k} \cdot [{}^{k}\mathbf{ROT}_{k+i} \cdot {}^{k+i}\mathbf{TRL}_{k+i+1} + {}^{k}\mathbf{TRL}_{k+i}] + {}^{m}\mathbf{TRL}_{k} \\ &= {}^{m}\mathbf{ROT}_{k} \cdot {}^{k}\mathbf{ROT}_{k+i+1} + {}^{m}\mathbf{TRL}_{k} \end{aligned}$$

Hence, Eq.9 is proved.

APPENDIX 2

VECTOR COMBINER, VECTOR SPLITTER WITH SINUSOIDAL CARRIERS OF THE SAME FREQUENCY

Given the components u_x and u_y of the vector **u**, the function of the vector combiner is to provide **u** in the form of the signal u(t), appropriately modulated and so suitable for further processing.

$$\mathbf{u} = [u_x , u_y , X , X]^T \qquad \text{Eq.1}$$

"X" means irrelevant for the current consideration.

Let us assume that we have chosen the x-carrier to be $c_x(t)=\cos(2.\Pi.f.t)$. Thus, it follows that the y-carrier will have to be $c_y(t)=\sin(2.\Pi.f.t)$. We note that both carriers do have here the same constant frequency f.

It follows for u(t) (exactly according to Eq.2.3.3 of section 2.3).

$$u(t) = u_x.\cos(2.\Pi.f.t). + u_y.\sin(2.\Pi.f.t). \qquad \text{Eq.2}$$

Of course, u(t) has the same frequency f as the carriers. And even this equation prescribes the way to process the signals u_x and u_y, if they are available in form of some

scalar (or DC) quantities. Fig.1 illustrates the implementation.

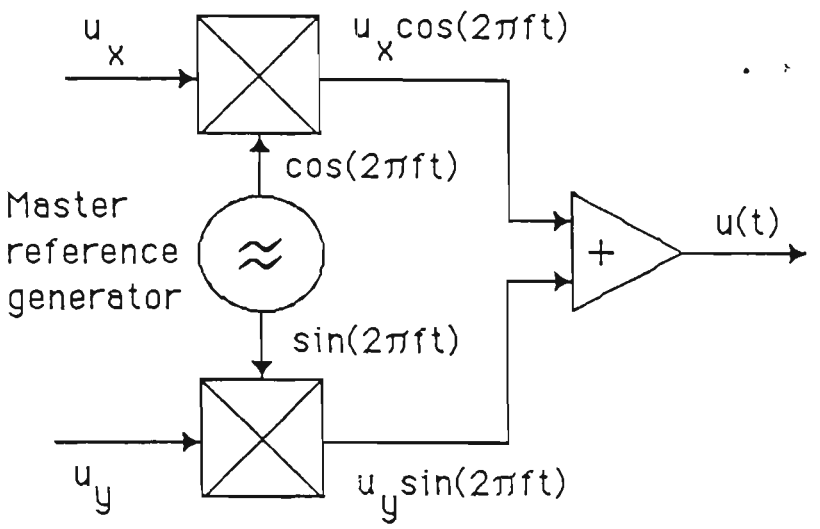


Fig. 1. : The "scalar" vector combiner.

However, the components of **u** may be available in the form of vectorial quantities rather than scalar.

$$u_{xv}(t) = u_x.\cos(2.\Pi.f.t) \qquad \text{Eq.3.x}$$

$$u_{yv}(t) = u_y.\cos(2.\Pi.f.t) \qquad \text{Eq.3.y}$$

Note that we have assumed for $u_{xv}(t)$ and $u_{yv}(t)$ having the same phase with the x-carrier.

In this case, the vector combiner will be as illustrated in Fig.2.

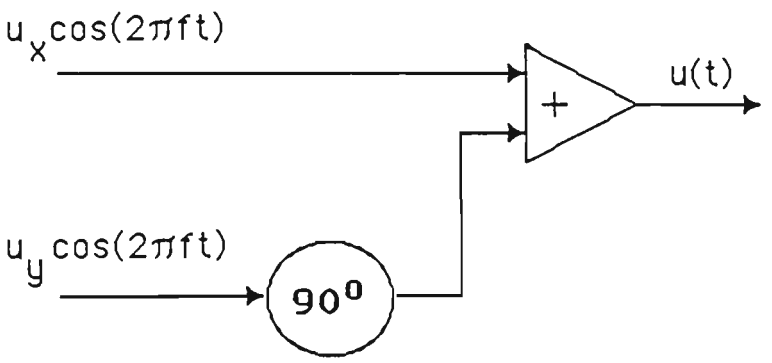


Fig. 2. : The "vectorial" vector combiner.

In this figure, little consideration reveals that the equation Eq.2 is, indeed, implemented.

Assuming that the vector **u** or, equivalently, the signal u(t) is rotated or delayed, respectively, by an angle θ to yield the new vector **v**, or equivalently, the signal v(t). This rotating/delaying process may better be described using the figure Fig.3, where we should mention the angle φ which is the angle between the vector **v** and the x-reference.

Now, we have the reverse-order problem. That is, given v(t), we wish to discriminate v_x and v_y from it. This will be the function of the vector splitter.

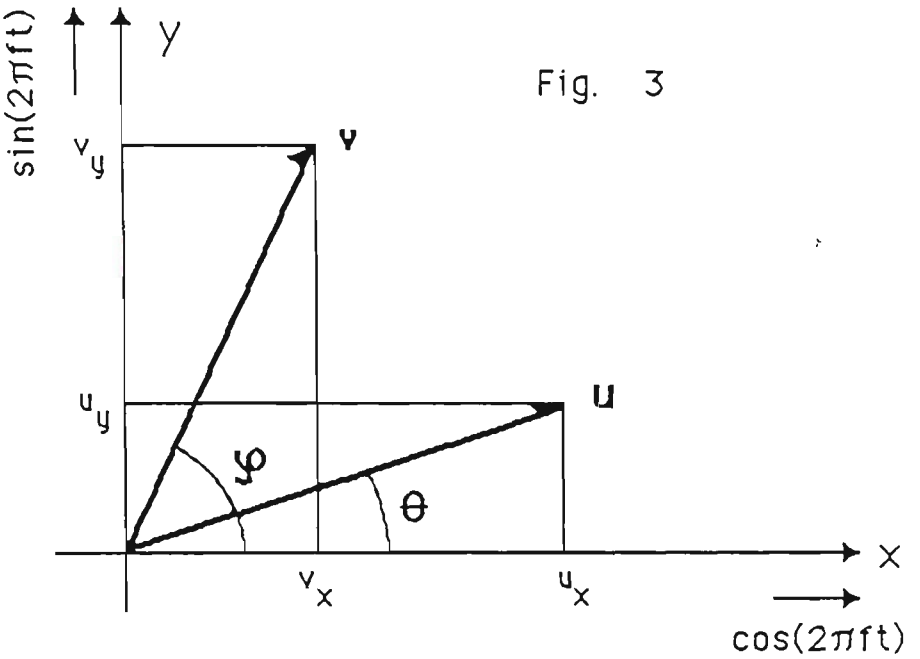
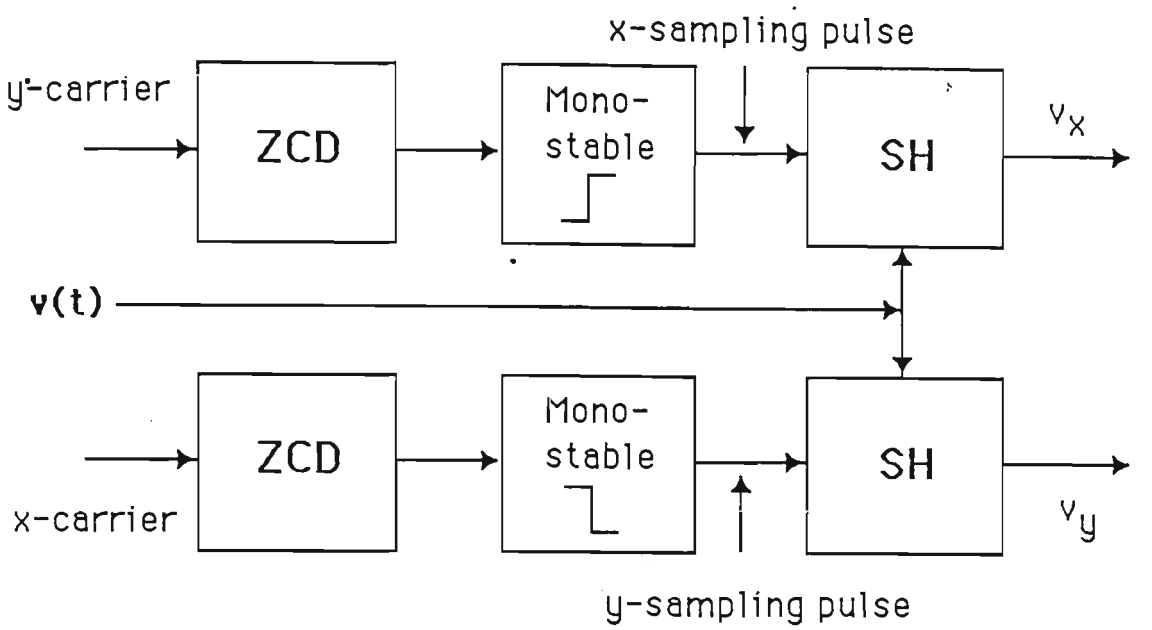


Fig. 3

The time signal expression for the vector **v** is $v(t)=\cos(2.\Pi.f.t-y)$. As is obvious the scalar quantity v_x will be observed at the maximum of the x-carrier; and the scalar quantity v_y will be observed at the maximum of the y-carrier.

Fig.4 illustrates the vector splitter designed according to what has just been said.



ZCD : Zero Crossing Detector

: Trigger on positive edge

: Trigger on negative edge

SH : Sample and Hold

Fig. 4 : The "scalar" vector splitter.

The vector splitter of Fig.4 provides scalar quantities v_x and v_y. However, for some reasons, we may be forced to provide them in modulated form. This can be achieved by applying the following equations:

$$v_x(t) = [v(t) + v(-t)]/2 \qquad \text{Eq.4.x}$$

$$v_y(t) = [v(t) - v(-t)]/2 \qquad \text{Eq.4.y}$$

where we make use of the x-carrier being symmetrical-even time function and the y-carrier being symmetrical-odd time function. The vector splitter is then as shown in Fig.5.

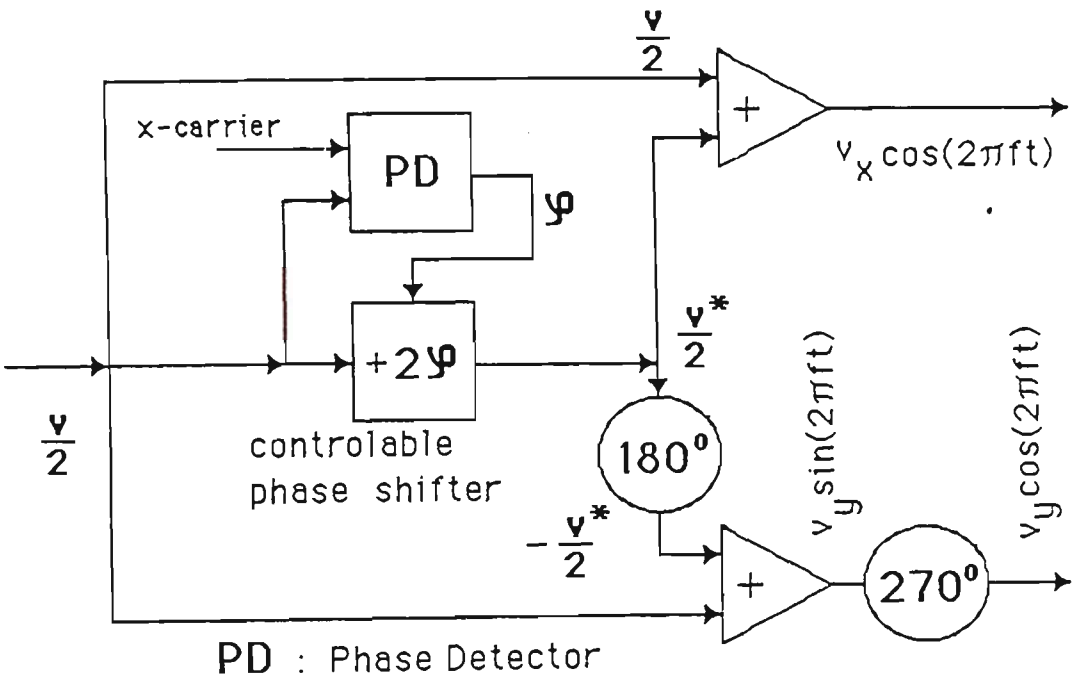


Fig. 5 : The "vectorial" vector splitter.

NOTE:

A variable phase shifter will claim in general one sinewave period delay maximum. The vectorial vector combiner of Fig.2 claims a delay corresponding to the phase shift of 90^0 or 1/4 sinewave period.

The controllable phase shifter of the vectorial vector splitter Fig.5 claims 360^0 delay maximum, while its y-output claims a phase shift of 270^0. Thus the total phase shift in this is 7/4 sinewave period maximum.

Thus, a serial combination of vector combiner, variable phase shifter, and vector splitter will claim 3 sinewave periods maximum.

APPENDIX 3

MULTIPLICATION OF ORTHONORMAL MATRICES USING ANALOGUE SIMULATION TECHNIQUE

Let us consider the ideal phase shifter of Fig.1

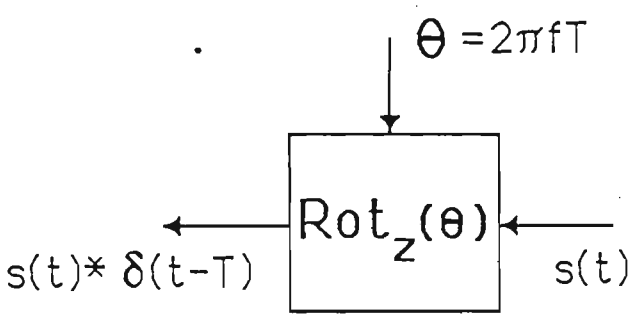


Fig. 1. : The ideal elementary rotate block.

In this figure, the notation $\delta(t)$ represents the Dirac-pulse (which we make use of for convenience of explanation), whereby the argument $(t-T)$ of $\delta(t-T)$ in connection with the convolution expression $s(t)*\delta(t-T)$ indicates a phase or time shift operation applied on the signal $s(t)$ to yield $s(t-T)$.

$$s(t)*\delta(t-T) = s(t-T)$$

where the time shift T corresponds to the rotating angle θ as $\theta = 2.\Pi.f.t$

The ideal translate block is presented in Fig.2. Despite the subfunctions of $Trans_x(a)$ and $Rot_z(\theta)$ in it, the ideal translate block as presented in this figure will play the role of an elementary function.

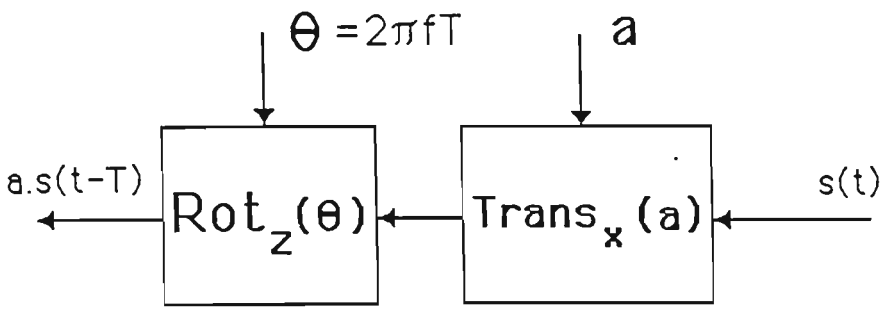


Fig.2. : The ideal translate block.

Now, consider a robot arm consisting of two links, whose (two-dimensional) geometric configuration is defined by the following two homogeneous transformation matrices:

$$ {}^{0}\mathbf{A}_1 = \begin{bmatrix} \cos(\theta_1) & -\sin(\theta_1) & 0 & a_1.\cos(\theta_1) \\ \sin(\theta_1) & \cos(\theta_1) & 0 & a_1.\sin(\theta_1) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{Eq.1} $$

$$ {}^{1}\mathbf{A}_2 = \begin{bmatrix} \cos(\theta_2) & -\sin(\theta_2) & 0 & a_2.\cos(\theta_2) \\ \sin(\theta_2) & \cos(\theta_2) & 0 & a_2.\sin(\theta_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{Eq.2} $$

where we have assumed θ_1 and θ_2 being variables, and a_1 and a_2 being constant parameters. Thus, we are dealing with a

two-dimensional revolute robot arm with two degrees of freedom.

We are interested in evaluating $\mathbf{T}_2 = \mathbf{A}_1\ \mathbf{A}_2$, however, by not using arithmetic expressions.

The "direct equations"

$$^{m}\mathbf{ROT}_{nD} = {}^{m}\mathbf{ROT}_{m+1}.{}^{m+1}\mathbf{ROT}_{nD} \qquad \text{Eq.3}$$

$$^{m}\mathbf{TRL}_{nD} = {}^{m}\mathbf{ROT}_{m+1}.{}^{m+1}\mathbf{TRL}_{nD} + {}^{m}\mathbf{TRL}_{m+1} \qquad \text{Eq.4}$$

are here applied for n=2 (two degrees of freedom). For m=2, the starting index:

$^{2}\mathbf{ROT}_{2D}$ = (3x3) unit matrix

$^{2}\mathbf{TRL}_{2D}$ = (3x1) null vector

In two dimensions a single reference sinewave signal is sufficient to implement the (3x3) unit matrix. However, the (3x1) null vector is implemented by using another sinewave reference of zero magnitude.

For m=1, we have

$$^{1}\mathbf{ROT}_{2D} = {}^{1}\mathbf{ROT}_{2}.{}^{2}\mathbf{ROT}_{2D} = {}^{1}\mathbf{ROT}_{2}$$

$$^{1}\mathbf{TRL}_{2D} = {}^{1}\mathbf{ROT}_{2}.{}^{2}\mathbf{TRL}_{2D} + {}^{1}\mathbf{TRL}_{2} = {}^{1}\mathbf{TRL}_{2}$$

which indicate

$$ {}^{1}\mathbf{ROT}_{2D} = \begin{bmatrix} \cos(\theta_2) & -\sin(\theta_2) & 0 \\ \sin(\theta_2) & \cos(\theta_2) & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \text{Eq.5} $$

and

$$ {}^{1}\mathbf{TRL}_{2D} = \begin{bmatrix} a_2.\cos(\theta_2) \\ a_2.\sin(\theta_2) \\ 0 \end{bmatrix} \qquad \text{Eq.6} $$

In two dimensions, alone the normal vector of ${}^{1}\mathbf{ROT}_{2D}$ is sufficient to describe the orientation of the matrix ${}^{1}\mathbf{T}_{2D}={}^{1}\mathbf{A}_{2}.{}^{2}\mathbf{T}_{2D}$.

Thus, we employ the elementary orientation block of Fig.1 to implement Eq. 5 and the elementary translate block of Fig.2 the Eq.6. The arrangement is shown in Fig.3 as the whole, where we may assume that the reference sinewave signal is available in form of an analogue signal.

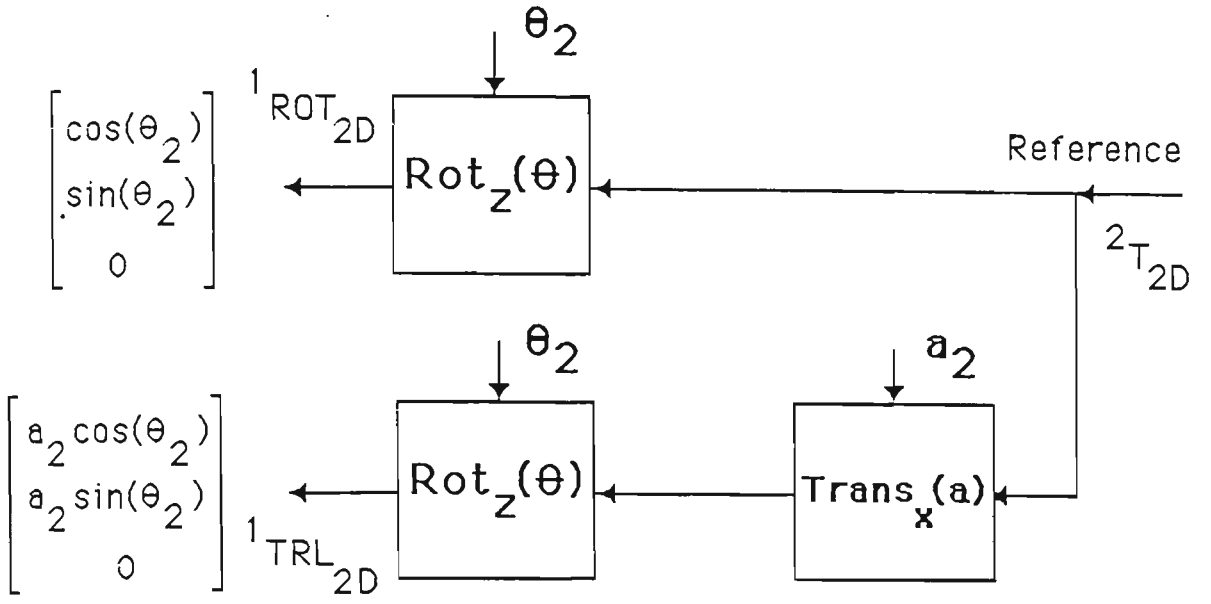


Fig. 3. : Analogue multiplication of $^{1}T_{2D} = {}^{1}A_{2}.{}^{2}T_{2D}$

Examining this figure, we realize that the $^{1}ROT_{2D}$ output is represented by the reference sinewave signal, appropriately delayed by the phase angle θ_2 such that it contains the x and y information $x=\cos(\theta_2)$ and $y=\sin(\theta_2)$ of the normal vector, see Eq.5. Similarly, the $^{1}TRL_{2D}$ output; however, this position signal has the magnitude a_2 as shown.

Proceeding the "direct equations" further to the last step for the index m=0, we obtain:

$$^{\circ}ROT_{2D} = {}^{\circ}ROT_{1}.{}^{1}ROT_{2D}$$

$$^{\circ}TRL_{2D} = {}^{\circ}ROT_{1}.{}^{1}TRL_{2D} + {}^{\circ}TRL_{1}$$

which, for ease of reference, are comprehensively written as follows:

$$^{o}\mathbf{ROT}_{2D}=\begin{bmatrix}\cos(\theta_1) & -\sin(\theta_1) & 0\\ \sin(\theta_1) & \cos(\theta_1) & 0\\ 0 & 0 & 1\end{bmatrix}\bullet\begin{bmatrix}\cos(\theta_2) & -\sin(\theta_2) & 0\\ \sin(\theta_2) & \cos(\theta_2) & 0\\ 0 & 0 & 1\end{bmatrix}$$

$$=\begin{bmatrix}\cos(\theta_1+\theta_2) & -\sin(\theta_1+\theta_2) & 0\\ \sin(\theta_1+\theta_2) & \cos(\theta_1+\theta_2) & 0\\ 0 & 0 & 1\end{bmatrix} \qquad \text{Eq.7}$$

$$^{o}\mathbf{TRL}_{2D}=\begin{bmatrix}\cos(\theta_1) & -\sin(\theta_1) & 0\\ \sin(\theta_1) & \cos(\theta_1) & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}a_2.\cos(\theta_2)\\ a_2.\sin(\theta_2)\\ 0\end{bmatrix}$$

$$+\begin{bmatrix}a_1.\cos(\theta_1)\\ a_1.\sin(\theta_1)\\ 0\end{bmatrix}$$

$$=\begin{bmatrix}a_2.\cos(\theta_1+\theta_2) & + & a_1.\cos(\theta_1)\\ a_2.\sin(\theta_1+\theta_2) & + & a_1.\sin(\theta_1)\\ & 0 & \end{bmatrix} \qquad \text{Eq.8}$$

Now, we are standing at the deciding point of the whole investigation. Eq.3 and Eq.4 require the outputs $^{1}\mathbf{T}_{2D}$ from

the function blocks representing ${}^1\mathbf{A}_2$ to serve as the inputs for the function blocks representing ${}^\circ\mathbf{A}_1$. Thus, to implement ${}^\circ\mathbf{T}_{2D}={}^\circ\mathbf{A}_1.{}^1\mathbf{T}_{2D}$, we will employ the same arrangment of Fig.3, however, this time the parameter/variable a & θ must appropriately be set to a_1 and θ_1, respectively, as required for this particular matrix ${}^\circ\mathbf{A}_1$. The Fig.4 illustrates this.

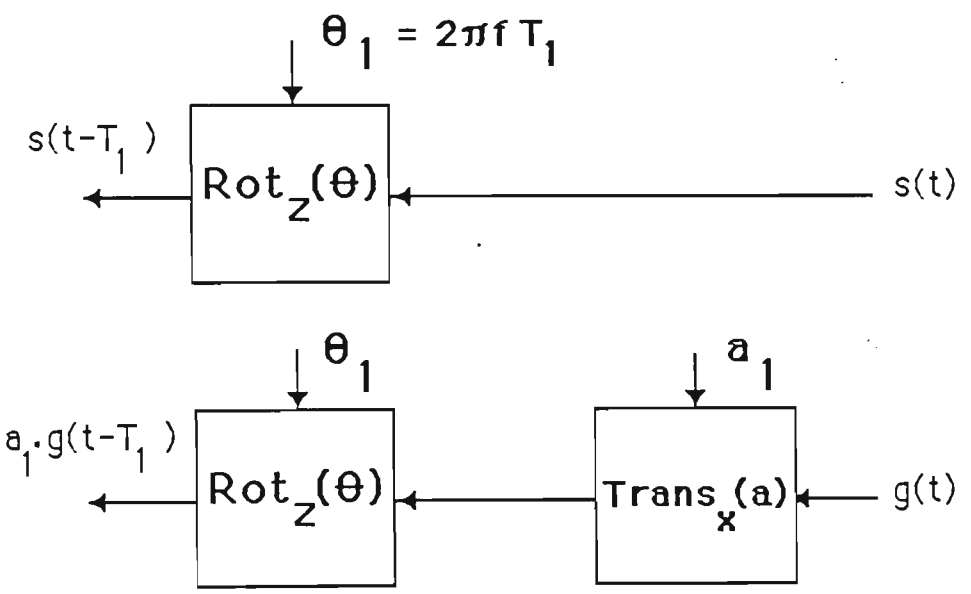


Fig. 4.

In this figure, whatever signal s(t) is applied on the input of the orientation part, the orientation ouput will be:

$$s(t)*\delta(t-T_1) = s(t-T_1) \qquad \text{Eq.9}$$

similarly, we have for the position part

$$a_1.g(t)*\delta(t-T_1) = a_1.g(t-T_1) \qquad \text{Eq.10}$$

Now, assuming that s(t) is the sinusoidal signal representing the normal vector of $^1\mathbf{ROT}_{2D}$, which itself is the delayed reference ref(t) by an angle $\theta_2=2.\Pi.f.T_2$ such that we may write for s(t) as follows:

$$s(t) = ref(t)*\delta(t-T_2) \qquad \text{Eq.11}$$

The substitution of s(t) into Eq.9 yields for the output signal of the orientation part of Fig.4:

$$ref(t)*\delta(t-T_2)*\delta(t-T_1) = ref(t-[T_2+T_1]) \qquad \text{Eq.12}$$

Similarly, if we had assumed s(t) being the signal which represents $^1\mathbf{TRL}_{2D}$, we would have found:

$$a_2.ref(t)*\delta(t-T_2)*\delta(t-T_1) = a_2.ref(t-[T_2+T_1]) \qquad \text{Eq.13}$$

However , if g(t) = ref(t), Eq.10 is written as:

$$a_1.ref(t)*\delta(t-T_1) = a_1.ref(t-T_1) \qquad \text{Eq.14}$$

In the usual cosine notation of the reference, we have for Eq.12 through Eq.14:

$$ref(t-[T_2+T_1]) = \cos(2.\Pi.f.t-[\theta_2+\theta_1]) \qquad \text{Eq.12b}$$

$$a_2.ref(t-[T_2+T_1]) = a_2.\cos(2.\Pi.f.t-[\theta_2+\theta_1]) \qquad \text{Eq.13b}$$

$$a_1.ref(t- \quad T_1) = a_1.\cos(2.\Pi.f.t- \quad \theta_1) \qquad \text{Eq.14b}$$

By inspection, Eq.12.b represents the normal vector of Eq.7 as its x/y components are observed at $2.\Pi.f.t=0^0$ and $2.\Pi.f.t=90^0$, respectively.

Eq.13b represents the result of the manipulation $^{o}\mathbf{ROT}_1.^{1}\mathbf{TRL}_{2D}$ providing $x=a_2.\cos(\theta_1+\theta_2)$ and $y=a_2.\sin(\theta_1+\theta_2)$, see Eq.8.

The signal of Eq.14b represents $^{o}\mathbf{TRL}_1$ and provides $x=a_1.\cos(\theta_1)$ and $y=a_1.\sin(\theta_1)$.

Putting Fig.3 and Fig.4 together, we are now in the position to present the arrangement that implements the matrix multiplications $^{o}\mathbf{T}_{2D} = {}^{o}\mathbf{A}_1.{}^{1}\mathbf{A}_2.{}^{2}\mathbf{T}_{2D}$ shown in Fig.5.

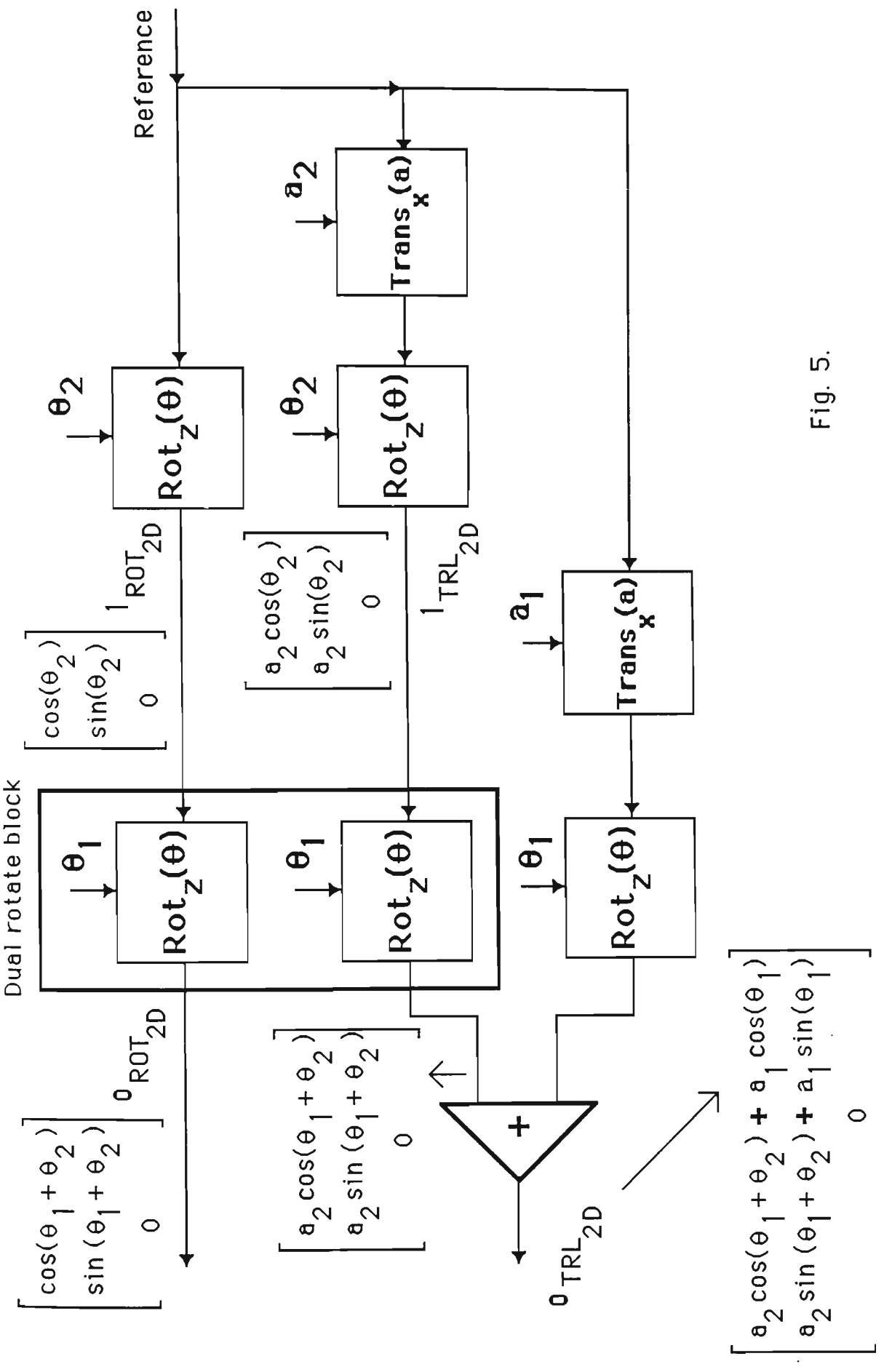

Reference
a_2
$Trans_x(a)$
θ_2
$Rot_z(\theta)$
θ_2
$Rot_z(\theta)$
$^1_{ROT_{2D}}$
$\begin{bmatrix} \cos(\theta_2) \\ \sin(\theta_2) \\ 0 \end{bmatrix}$
$\begin{bmatrix} a_2\cos(\theta_2) \\ a_2\sin(\theta_2) \\ 0 \end{bmatrix}$
$^1_{TRL_{2D}}$
a_1
$Trans_x(a)$
Dual rotate block
θ_1
$Rot_z(\theta)$
θ_1
$Rot_z(\theta)$
θ_1
$Rot_z(\theta)$
$^0_{ROT_{2D}}$
$\begin{bmatrix} \cos(\theta_1+\theta_2) \\ \sin(\theta_1+\theta_2) \\ 0 \end{bmatrix}$
$\begin{bmatrix} a_2\cos(\theta_1+\theta_2) \\ a_2\sin(\theta_1+\theta_2) \\ 0 \end{bmatrix}$
+
$^0_{TRL_{2D}}$
$\begin{bmatrix} a_2\cos(\theta_1+\theta_2)+a_1\cos(\theta_1) \\ a_2\sin(\theta_1+\theta_2)+a_1\sin(\theta_1) \\ 0 \end{bmatrix}$

Fig. 5.

APPENDIX 4

Alternative graphical representation for the T-matrices

This appendix refers to the T-matrices of section 3.2.2.

Rotating the normal vector $\mathbf{n}_5$ about the z_4 axis, we have it expressed in the 4-th coordinate frame:

$+C_5$ on the X_5 axis

$+S_5$ on the Y_5 axis

Similarly, rotating the orientation vector $\mathbf{o}_5$ about the z_4 axis, we have it expressed inthe 4-th coordinate frame:

$-S_5$ on the X_5 axis

$+C_5$ on the Y_5 axis

This can be expressed in "tree-format" as shown in Fig.1, then generalized to Fig.2, which also applies for position vectors.

Using these definitions, the computation of the T-matrices is performed straight-forwardly as shown in Fig.3 through Fig.6.

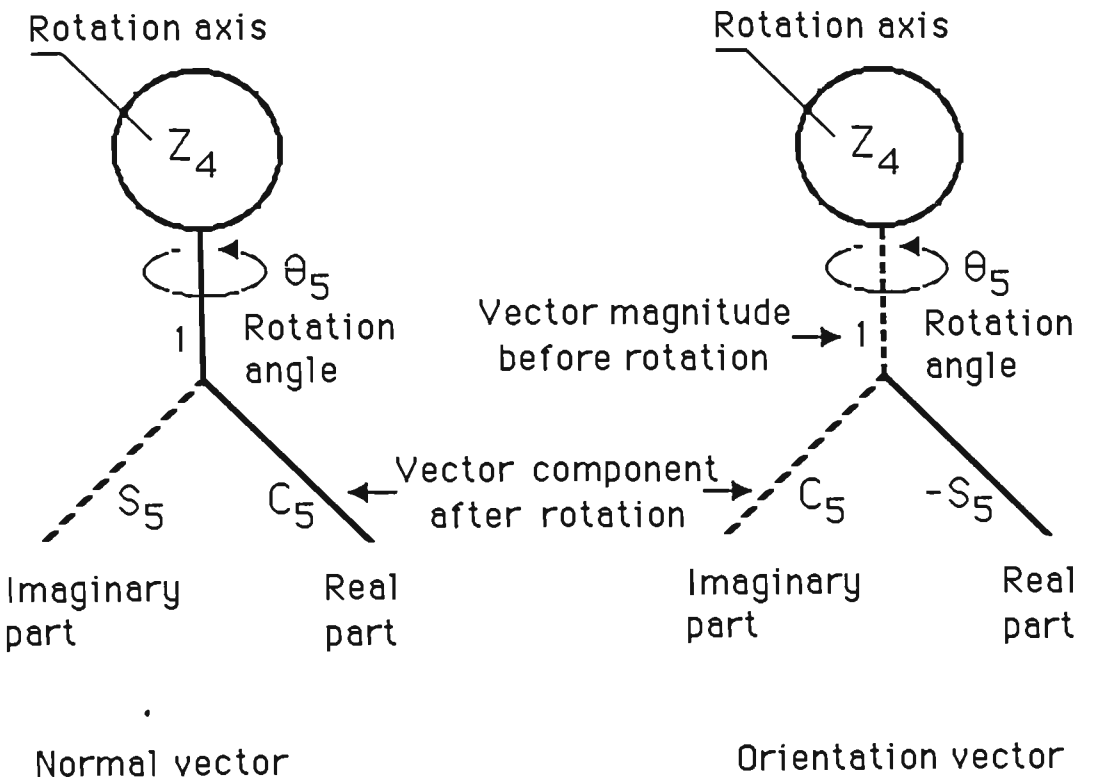


Fig. 1.

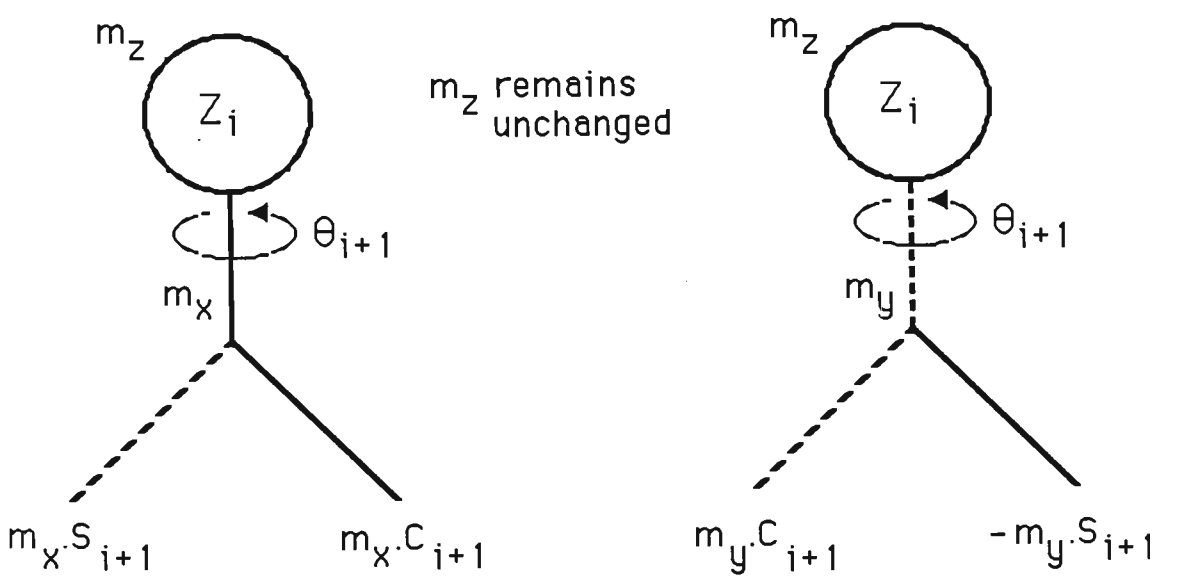


Fig. 2.: Generalization of figure 1.

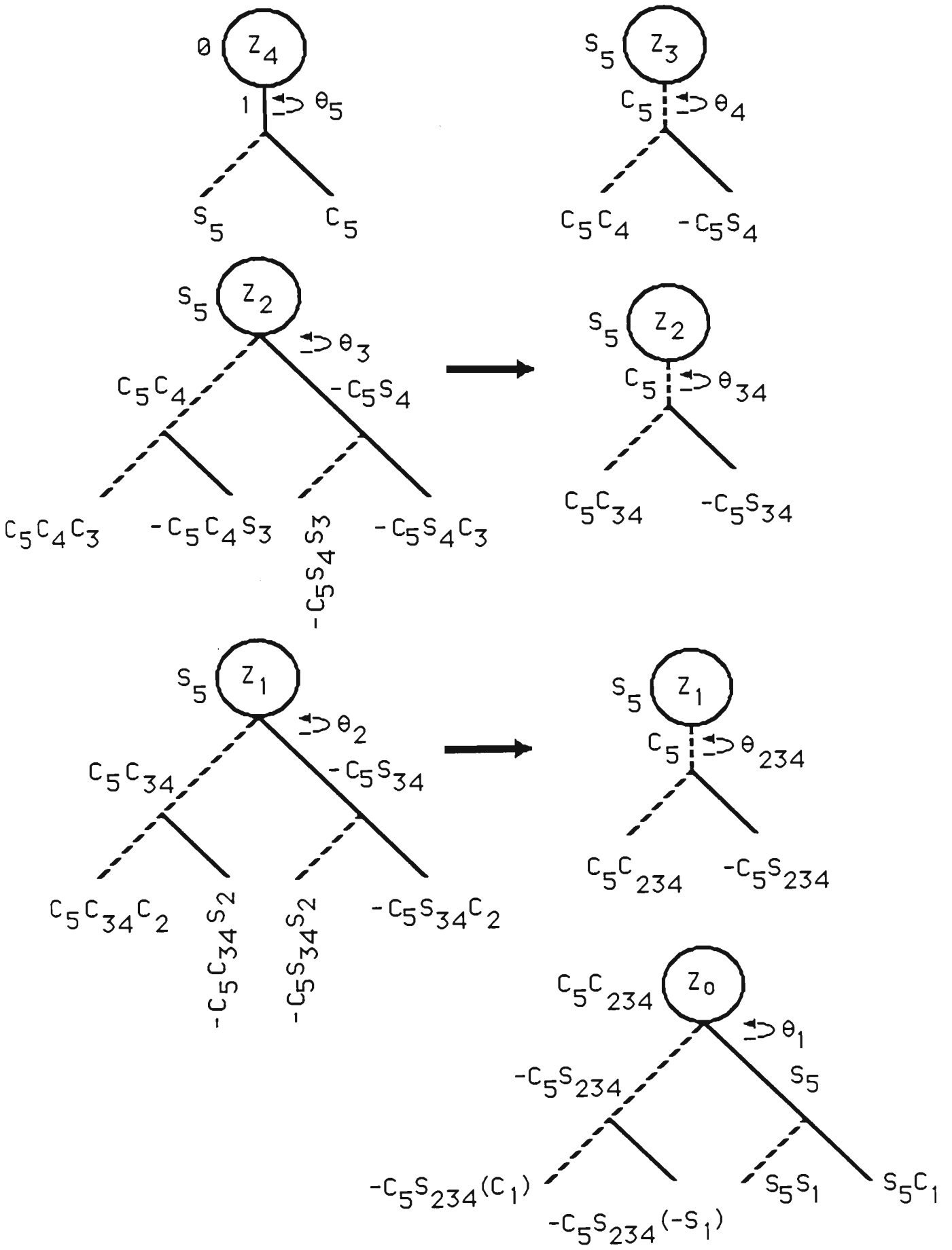


Fig. 3. : Normal vector trees.

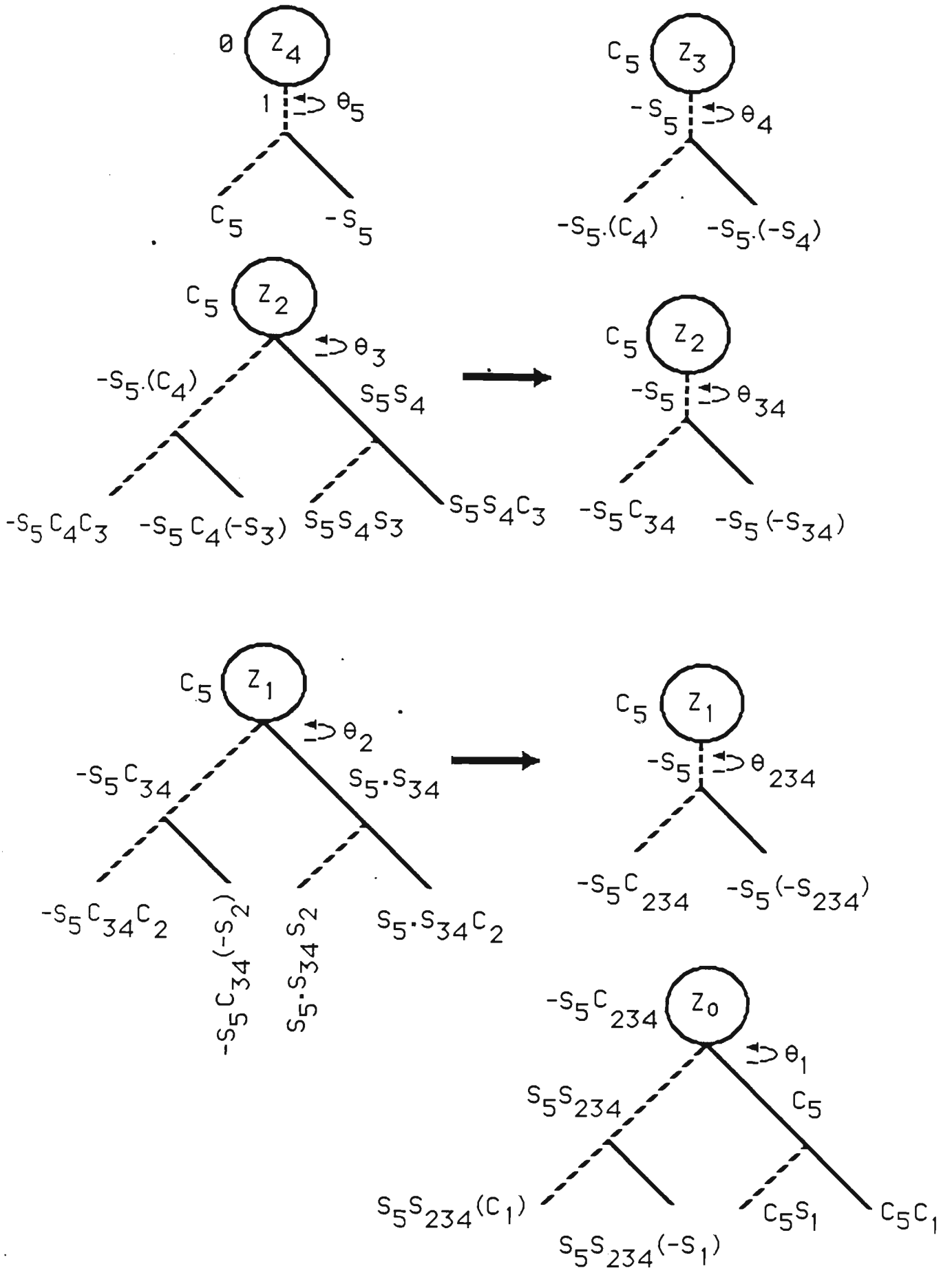


Fig. 4. : Orientation vector trees.

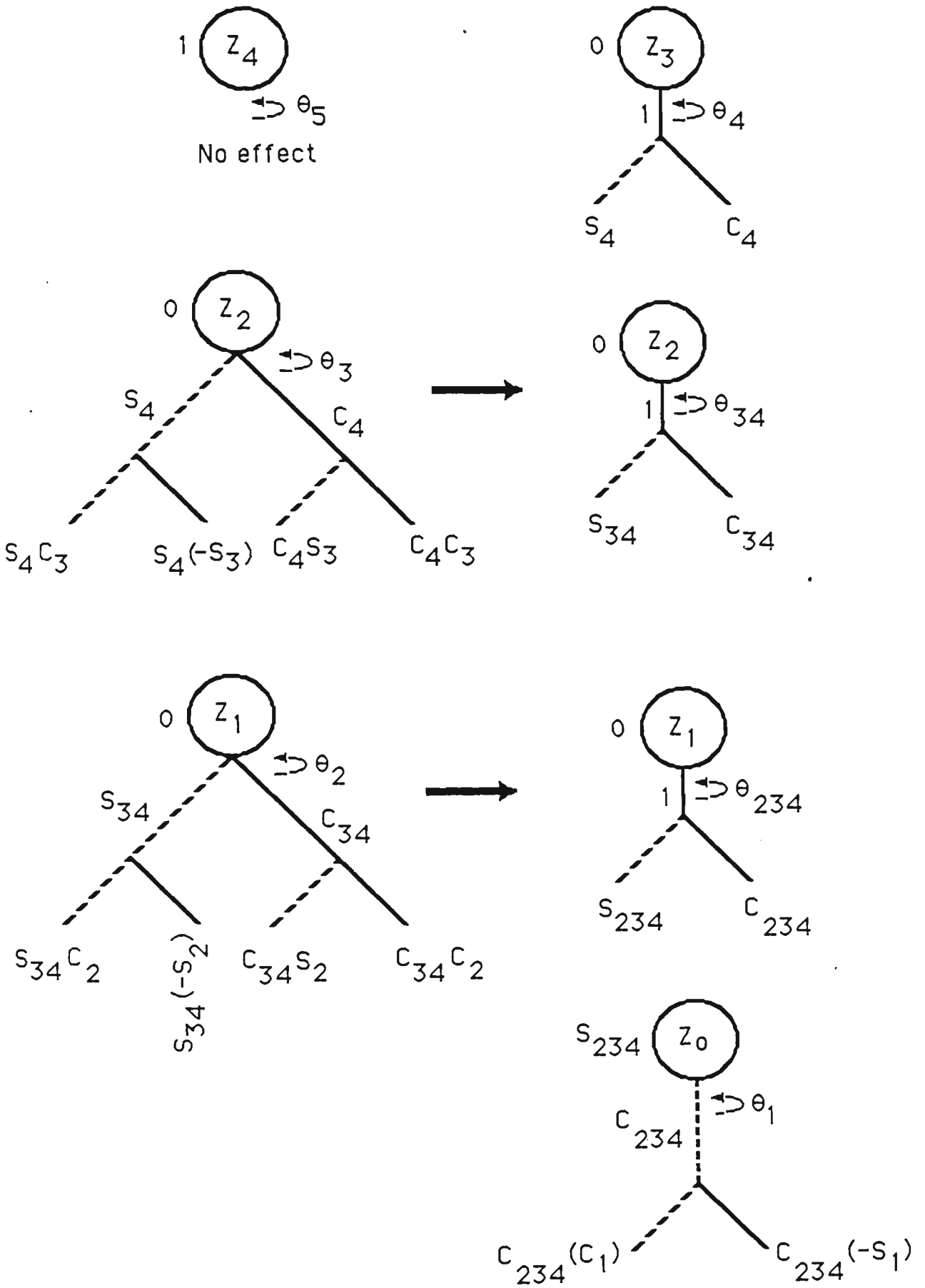
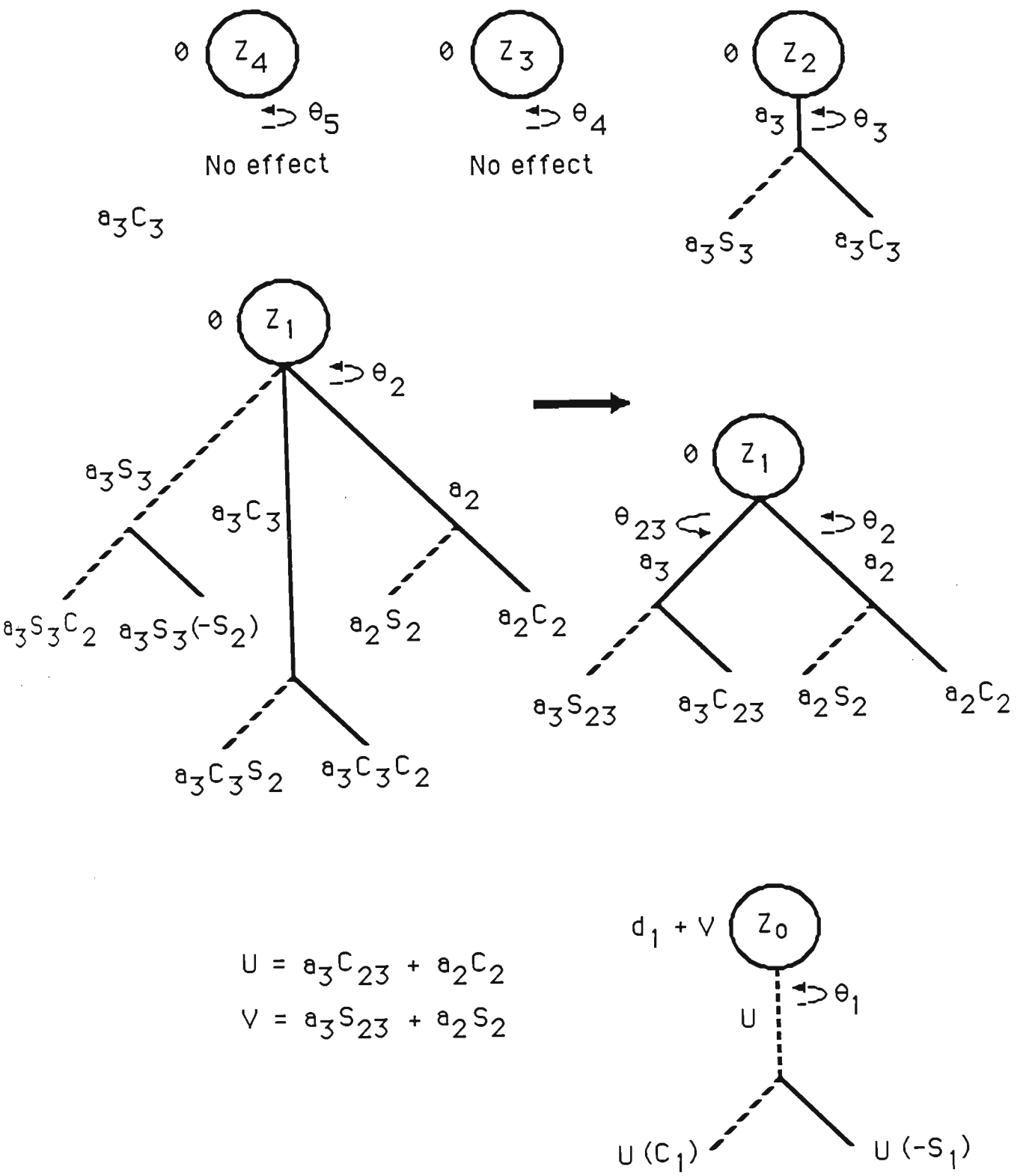


Fig. 5. : Approach vector trees.



$$U = a_3C_{23} + a_2C_2$$
$$V = a_3S_{23} + a_2S_2$$

Fig. 6. : Position vector trees.

APPENDIX 5

Arithmetic expressions for the inverse kinematic solutions for the RM-501.

This appendix refers to the T-matrices of section 3.2.2.

Manipulation of the position vectors from the direct manipulation path.

k=n

where n=5 for the RM-501,

$^5\mathbf{TRL}_{5D} = 0$ null-vector

Therefore, the magnitude square $^5f_{5D}$ of this vector is identically zero

$^5f_{5D} = 0$ No Variable

k=4

$^4\mathbf{TRL}_{5D}=^4\mathbf{ROT}_5 . ^5\mathbf{TRL}_{5D}+^4\mathbf{TRL}_5=^4\mathbf{TRL}_5=0$

$^4f_{5D} = 0$ No variable

k=3

$^{3}TRL_{5D} = {}^{3}ROT_{4} . {}^{4}TRL_{5D} + {}^{3}{}_{4} = {}^{3}TRL_{4} = 0$

$^{3}f_{5D} = 0$ No variable

k=2

$^{2}TRL_{5D} = {}^{2}ROT_{3} . {}^{3}TRL_{5D} + {}^{2}TRL_{3} = {}^{2}TRL_{3}$

$$= \begin{bmatrix} a_3.C_3 \\ a_3.S_3 \\ 0 \end{bmatrix}$$

$^{2}f_{5D} = (a_3.C_3)^2 + (a_3.S_3)^2 + (0)^2$

$= (a_3)^2 = \text{constant}$

No variable

k=1

$^{1}TRL_{5D} = {}^{1}ROT_{2} . {}^{2}TRL_{5D} + {}^{1}TRL_{2}$

$$= \begin{bmatrix} C_2 & -S_2 & 0 \\ S_2 & C_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} a_3.C_3 \\ a_3.S_3 \\ 0 \end{bmatrix} + \begin{bmatrix} a_2.C_2 \\ a_2.S_2 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} a_3.C_{23} + a_2.C_2 \\ a_3.S_{23} + a_2.S_2 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} u \\ v \\ w \end{bmatrix} \quad \text{(substitution for reference purposes)}$$

$${}^1f_{5D} = u^2 + v^2 + w^2$$

$$= (a_3.C_{23} + a_2.C_2)^2 + (a_3.S_{23} + a_2.S_2)^2$$

$$= (a_3.C_{23})^2 + 2.a_2.a_3.C_2.C_{23} + (a_2.C_2)^2$$
$$+ (a_3.S_{23})^2 + 2.a_2.a_3.S_2.S_{23} + (a_2.S_2)^2$$
$$= (a_2)^2 + (a_3)^2 + 2.a_2.a_3.(C_2.C_{23} + S_2.S_{23})$$
$$= (a_2)^2 + (a_3)^2 + 2.a_2.a_3.C_3 \qquad \text{Eq.1d}$$

$$= {}^1f_{5D}(\theta_3)$$

One Variable

<u>k=0</u>

$${}^{\circ}\mathbf{TRL}_{5D} = {}^{\circ}\mathbf{ROT}_1.{}^1\mathbf{TRL}_{5D} + {}^{\circ}\mathbf{TRL}_1$$

$$= \begin{bmatrix} -S_1 & 0 & C_1 \\ C_1 & 0 & S_1 \\ 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} u \\ v \\ w \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ d_1 \end{bmatrix}$$

$$= \begin{bmatrix} -S_1.u \\ C_1.u \\ v+d_1 \end{bmatrix}$$

$$^{o}f_{5D} = (-S_1.u)^2 + (C_1.u)^2 + (v+d_1)^2$$

$$= u^2 + v^2 + 2.d_1.v + (d_1)^2$$

$$= {}^{1}f_{5D}(\theta_3) + 2d_1.(a_3.S_{23} + a_2.S_2) + (d_1)^2 \qquad \text{Eq.2d}$$

$$= {}^{o}f_{5D}(\theta_2, \theta_3) \qquad \text{Two variables}$$

Manipulation of the position vectors from the inverse manipulation path.

k=0

$$^{o}\mathbf{TRL}_{5I} = [p_x, p_y, p_z]^T$$

which is given as desired position of the manipulator, thus:

$$^{o}f_{5I} = (p_x)^2 + (p_y)^2 + (p_z)^2 = \text{constant} \qquad \text{Eq.2i}$$

No variable

k=1

$$^{1}\mathbf{TRL}_{5I} = (^{o}\mathbf{ROT}_1)^{-1}.(^{o}\mathbf{TRL}_{5I} - {}^{o}\mathbf{TRL}_1)$$

$$= \begin{bmatrix} -S_1 & C_1 & 0 \\ 0 & 0 & 1 \\ C_1 & S_1 & 0 \end{bmatrix} \cdot \begin{bmatrix} p_x \\ p_y \\ p_z - d_1 \end{bmatrix}$$

$$= \begin{bmatrix} -S_1.p_x + C_1.p_y \\ p_z - d_1 \\ C_1.p_x + S_1.p_y \end{bmatrix} = \begin{bmatrix} u \\ v \\ w \end{bmatrix}$$

$$^1f_{5I} = u^2 + v^2 + w^2$$

$$= (-S_1.p_x + C_1.p_y)^2 + (p_z - d_1)^2 + (C_1.p_x + S_1.p_y)^2$$

$$= (p_x)^2 + (p_y)^2 + (p_z - d_1)^2 = \text{constant} \qquad \text{Eq.1i}$$

No variable

<u>k=2</u>

$$^2\mathbf{TRL}_{5I} = (^1\mathbf{ROT}_2)^{-1}.(^1\mathbf{TRL}_{5I} - {}^1\mathbf{TRL}_2)$$

$$= \begin{bmatrix} C_2 & S_2 & 0 \\ -S_2 & C_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u - a_2.C_2 \\ v - a_2.S_2 \\ w \end{bmatrix}$$

$$= \begin{bmatrix} C_2.(u - a_2.C_2) + S_2.(v-a_2.S_2) \\ -S_2.(u - a_2.C_2) + C_2.(v-a_2.S_2) \\ w \end{bmatrix} = \begin{bmatrix} e \\ f \\ w \end{bmatrix}$$

$$^2f_{5I} = e_2 + f^2 + w^2$$

$$= (C_2.(u-a_2.C_2))^2+2.C_2.S_2(u-a_2.C_2).(v-a_2.S_2)$$
$$+ (S_2.(v-a_2.S_2))^2$$
$$+ (S_2.(u-a_2.C_2))^2-2.C_2.S_2.(u-a_2.C_2).(v-a_2.S_2)$$
$$+ (C_2.(v-a_2.S_2))^2$$
$$+ w^2$$

$$= (u-a_2.C_2)^2+(v-a_2.S_2)^2 + w^2$$
$$= u^2-2.a_2.C_2.u + (a_2.C_2)^2$$
$$+ v^2-2.a_2.S_2.v + (a_2.S_2)^2 + w^2$$
$$= u^2 + v^2 +w^2 + (a_2)^2 -2.a_2.(u.C_2 + v.S_2)$$
$$= {}^1f_{5I} + (a_2)^2 - 2.a_2.(u.C_2 + v.S_2)$$
$$= {}^2f_{5I}(\theta_1,\theta_2)$$

Two variables considering that u is function of θ_1 (see above).

<u>k=3</u>

$$^3\mathbf{TRL}_{5I} = (^2\mathbf{ROT}_3)^{-1}.(^2\mathbf{TRL}_{5I} - {}^2\mathbf{TRL}_3)$$

$$\begin{bmatrix} C_3 & S_3 & 0 \\ -S_3 & C_3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} e - a_3.C_3 \\ f - a_3.S_3 \\ w \end{bmatrix}$$

$$= \begin{bmatrix} C_3.(e-a_3.C_3) + S_3.(f-a_3.S_3) \\ -S_3.(e-a_3.C_3) + C_3.(f-a_3.S_3) \\ w \end{bmatrix} = \begin{bmatrix} q \\ r \\ w \end{bmatrix}$$

These expressions have exactly the same forms as those under k=2. Thus, we can write straight-forward for ${}^3f_{5I}$ as follows:

$${}^3f_{5I} = q^2 + r^2 + w^2$$
$$= e^2 + f^2 + w^2 + (a_3)^2 - 2.a_3.(e.C_3 + f.S_3)$$
$$= {}^2f_{5I}(\theta_1,\theta_2) + (a_3)^2 - 2.a_3.(e.C_3 + f.S_3)$$
$$= {}^3f_{5I}(\theta_1,\theta_2,\theta_3)$$

Three variables

<u>k=4</u>

$${}^4\mathbf{TRL}_{5I} = ({}^3\mathbf{ROT}_4)^{-1}.({}^3\mathbf{TRL}_{5I} - {}^3\mathbf{TRL}_4)$$

Because ${}^3\mathbf{TRL}_4 = 0$ identically, see the corresponding A-matrix, it does not contribut e to ${}^4f_{5I}$. In addition, the rotation $({}^3\mathbf{ROT}_4)^{-1}$ does not affect the magnitude of the vector ${}^3\mathrm{TRL}_{5I}$ in any way, so we have:

$${}^4f_{5I} = {}^3f_{5I}$$
$$= {}^4f_{5I}(\theta_1,\theta_2,\theta_3)$$

Three variables

<u>k=5</u>

$${}^5\mathbf{TRL}_{5I} = ({}^4\mathbf{ROT}_5)^{-1}.({}^4\mathbf{TRL}_{5I} - {}^4\mathbf{TRL}_5)$$

Because of the same reasons, even that the rotation $({}^{4}\mathbf{ROT}_{5})^{-1}$ does not affect the magnitude of ${}^{4}\mathbf{TRL}_{5I}$ and that ${}^{4}\mathbf{TRL}_{5} = 0$ identically, the quantity ${}^{5}f_{5I}$ will be as follows:

${}^{5}f_{5I} = {}^{4}f_{5I} = {}^{5}f_{5I}(\theta_1, \theta_2, \theta_3)$ Three variables

The following comparis on table offers an overall-insight into the problem.

DIRECT MANIPULATION	INVERSE MANIPULATION
${}^{0}f_{5D} = {}^{0}f_{5D}(\theta_2, \theta_3)$	${}^{0}f_{5I} = (P_X)^2 + (P_Y)^2 + (P_Z)^2$
Two variables	No variable
${}^{1}f_{5D} = {}^{1}f_{5D}(\theta_3)$	${}^{1}f_{5I} = (p_x)^2 + (p_y)^2 + (p_z - d_1)^2$
One variable	No variable
${}^{2}f_{5D} = (a_3)^2 = \text{constant}$	${}^{2}f_{5I} = {}^{2}f_{5I}(\theta_1, \theta_2)$
No variable	Two variables
${}^{3}f_{5D} = 0$	${}^{3}f_{5I} = {}^{3}f_{5I}(\theta_1, \theta_2, \theta_3)$
No variable	Three variables
${}^{4}f_{5D} = 0$	${}^{4}f_{5I} = {}^{4}f_{5I}(\theta_1, \theta_2, \theta_3)$
No variable	Three variables
${}^{5}f_{5D} = 0$	${}^{5}f_{5I} = {}^{5}f_{5I}(\theta_1, \theta_2, \theta_3)$
No variable	Three variables

Solving sequence

We summarize the solving procedure for the positional variables as follows:

1) Set ${}^1f_{5D} = {}^1f_{5I}$ to find θ_3.

2) Set ${}^\circ f_{5D} = {}^\circ f_{5I}$ to find θ_2.

3) To find θ_1, equalize any other remaining pair ${}^kf_{5D}$ and ${}^kf_{5I}$, that involve θ_1 in their functional relationships. In this case, the possible pairs are ${}^kf_{5D}$ and ${}^kf_{5I}$, for k=2,3,4 or 5.

1) Set ${}^1f_{5D} = {}^1f_{5I}$ to find θ_3.

From Eq.1.d and Eq.1.i we have

$$(a_2)^2+(a_3)^2+2.a_2.a_3.C_3 = (p_x)^2+(p_y)^2+(p_z-d_1)^2$$

implying

$$C_3 = \frac{(p_x)^2 + (p_y)^2 + (p_z-d_1)^2 - [(a_2)^2 +(a_3)^2]}{2.a_2.a_3}$$

Taking

$$S_3 = +SQRT(1-(C_3)^2) \text{ or } S_3 = -SQRT(1-(C_3)^2)$$

we find two solutions for θ_3:

$$\theta_{3a} = \text{ATAN2}(+S_3, C_3)$$

as well as

$$\theta_{3b} = \text{ATAN2}(-S_3, C_3)$$

The function ATAN2 represents the arctan-function requiring two real arguments and providing $-\Pi =< \theta < +\Pi$.

As S_3 and C_3 cannot vanish at the same time, there are no singularities which must be considered.

2) Set ${}^{o}f_{5D} = {}^{o}f_{5I}$ to find θ_2.

From Eq.2.d and Eq. 2.i we have

$${}^{1}f_{5D} + 2d_1.(a_3.S_{23} + a_2.S_2) + (d_1)^2 = (p_x)^2 + (p_y)^2 + (p_z)^2$$

implying

$$a_3.S_{23} + a_2.S_2 = \frac{(p_x)^2 + (p_y)^2 + (p_z)^2 - {}^{1}f_{5D} - (d_1)^2}{2d_1}$$

Considering that we have just equalized ${}^{1}f_{5D}$ with ${}^{1}f_{5I}$ in the first step above

$${}^{1}f_{5D} = {}^{1}f_{5I} = (p_x)^2 + (p_y)^2 + (p_z - d_1)^2$$

Thus, after substitution and some manipulations, we have:

$$a_3.S_{23} + a_2.S_2 = p_z - d_1$$

The left hand side yields:

$$a_3.S_{23} + a_2.S_2 = (a_3.C3 + a_2).S_2 + a_3.S_3.C_2$$

The substitution

$$r.\sin(\gamma) = a_3.S_3$$

$$r.\cos(\gamma) = a_3.C_3 + a_2$$

unambiguously implies

$$r = +SQRT\ [\ (a_3.S_3)^2 + (a_3.C_3 + a_2)^2\ ] > 0$$

and

$$\gamma = ATAN2\ [\ (a_3.S_3\ ,\ a_3.C_3 + a_2\ ]$$

Now, we have:

$$r.\cos(\gamma).S_2 + r.\sin(\gamma).C_2 = p_z - d_1$$

implying

$$\sin(\gamma+\theta_2) = (p_z - d_1)/r$$

Taking

$$\cos(\gamma+\theta_2) = + \text{SQRT}(1 - [(p_z - d_1)/r]^2)$$

or

$$\cos(\gamma+\theta_2) = - \text{SQRT}(1 - [(p_z - d_1)/r]^2)$$

we find two solutions for θ_2:

$$\theta_{2a} = \text{ATAN2}[(p_z - d_1)/r, +\text{SQRT}(1 - [(p_z - d_1)/r]^2)] - \gamma$$

$$\theta_{2b} = \text{ATAN2}[(p_z - d_1)/r, -\text{SQRT}(1 - [(p_z - d_1)/r]^2)] - \gamma$$

3) Set ${}^3f_{5D} = {}^3f_{5I}$ to find θ_1.

We refer to the description of the manipulation of the position vectors from the inverse manipulation path presented above, where we have defined the variables u,v and e,f, which we again make use of in the following.

The manipulation of the position vectors from the inverse manipulation path provides:

$$^3f_{5I} = {}^2f_{5I}\ (\theta_1,\theta_2)$$
$$+(a_3)^2 - 2.a_3.(e.C_3 + f.S_3)$$

Substitution of $^2f_{5I}(\theta_1,\theta_2)$ in this yields

$$^3f_{5I} = {}^1f_{5I} +$$
$$+ (a_2)^2 - 2.a_2.(u.C_2 + v.S_2)$$
$$+ (a_3)^2 - 2.a_3.(e.C_3 + f.S_3)$$

In the next step, substitution of $^1f_{5I}$ yields

$$^3f_{5I} = (p_x)^2 + (p_y)^2 + (p_z - d_1)^2$$
$$+ (a_2)^2 - 2.a_2.(u.C_2 + v.S_2)$$
$$+ (a_3)^2 - 2.a_3.(e.C_3 + f.S_3)$$

The comparison table indicates that $^3f_{5D} = 0$. Now, equalizing both $^3f_{5D}$ and $^3f_{5I}$

$$0 = (p_x)^2 + (p_y)^2 + (p_z - d_1)^2$$
$$+ (a_2)^2 - 2.a_2.(u.C_2 + v.S_2)$$
$$+ (a_3)^2 - 2.a_3.(e.C_3 + f.S_3)$$

In general, the right hand side $^3f_{5I}$ is some positive quantity which we would like to see equal zero as the equation constraint requires. It will, indeed, be zero if there exist solutions for θ_1. In other words, the minimal extremes of this function coincide with its nullpoints if

there exist some. That means that the solutions for θ_1 are to find in the minima of the function ${}^3f_{5I}$.

Taking partial derivative of ${}^3f_{5I}$ with respect to θ_1, all the constants in the expression become zero, we have:

$$\frac{d}{d\theta_1}{}^3f_{5I} = -2.a_2.\left[C_2.\frac{du}{d\theta_1} + S_2.\frac{dv}{d\theta_1} \right]$$

$$-2.a3.\left[C_3.\frac{de}{d\theta_1} + S_3.\frac{df}{d\theta_1} \right]$$

considering that

$$u = -S_1.p_x + C_1.p_y$$

$$v = p_z - d_1$$

$$e = C_2.(u - a_2.C_2) + S_2.(v - a_2.S_2)$$

$$f = -S_2.(u - a_2.C_2) + C_2.(v - a_2.S_2)$$

we find:

$$\frac{du}{d\theta_1} = -p_x.C_1 - p_y.S_1 \qquad \frac{dv}{d\theta_1} = 0$$

$$\frac{de}{d\theta_1} = C_2 \frac{du}{d\theta_1} \qquad \frac{df}{d\theta_1} = -S_2 . \frac{du}{d\theta_1}$$

Using these expressions, we have after some manipulations:

$$\frac{d}{d\theta_1} {}^3f_{5I} = [-2.a_2.C_2 - 2.a_3.C_{23}] . \frac{du}{d\theta_1} = 0$$

Assuming $-2.a_2.C_2-2.a_3.C_{23} <> 0$, if we require the derivative of u to be zero, we have:

$$-p_x.C_1 - p_y.S_1 = 0 \quad .$$

$$==> S_1/C_1 = - p_x/p_y \qquad \text{Eq.3}$$

implying:

$$\theta_{1a} = \text{ATAN2}(-p_x, p_y)$$

as well as

$$\theta_{1b} = \text{ATAN2}(p_x, -p_y)$$

In addition, we require that the following inequality for the second order derivative of ${}^3f_{5I}$ is satisfied, when the particular θ_{1a} or θ_{1b} is substituted in it:

$$\frac{(d)^2}{(d\theta_1)^2}\,{}^3f_{5I} = -2.[\, a_2.C_2 + a_3.C_{23}\,].\,\frac{(d)^2u}{(d\theta_1)^2} > 0$$

where

$$\frac{(d)^2u}{(d\theta_1)^2} = +\,p_x.S_1 - p_y.C_1$$

in order to ensure that we have found the minimum of ${}^3f_{5I}$ meaning solution for θ_1.

Degeneracy

The particular case of $-2.a_2.C_2-2.a_3.C_{23} = 0$ indicating

$$\frac{S_2}{C_2} = \frac{a_2 + a_3.C_3}{a_3.S_3}$$

means that the derivative of ${}^3f_{5I}$ is identically zero, irrespective of whichever value we have for θ_1, and indicates that any arbitrary θ_1 is solution. This is known as manipulator's degeneracy with it losing one degree of freedom. This is illust rated in Fig.1. As can be seen, this is the case when $p_x=p_y=0$ which represents singularity in the arguments of ATAN2 above.

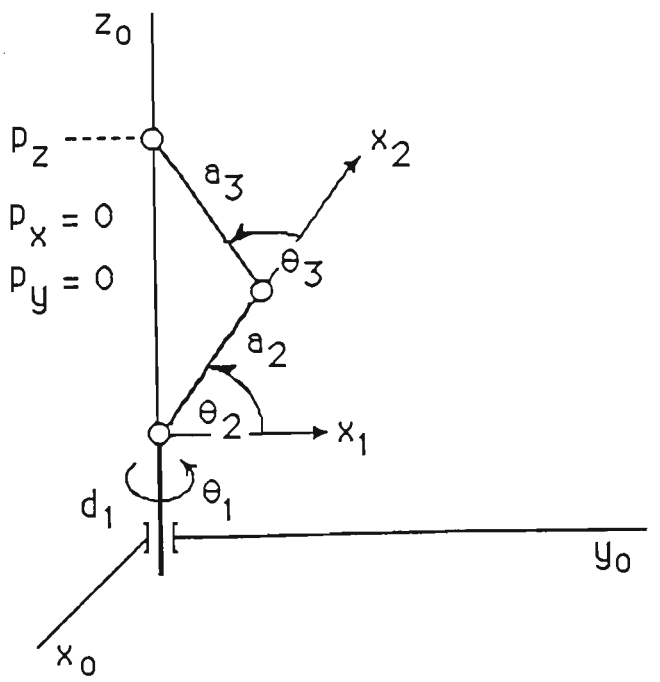


Fig. 1. : Illustration of manipulator's degeneracy.

DERIVATION OF ORIENTATION SOLUTIONS

Method using the Trace Vector of the Error Matrix

The error matrix ${}^5\mathbf{E}_{5I}$ available at the end output of the inverse manipulation path of the kinematic controller is mathematically determined by:

$$ {}^5\mathbf{E}_{5I} = ({}^\circ\mathbf{T}_5)^{-1} . {}^\circ\mathbf{T}_{5I} \qquad \text{Eq.4} $$

where ${}^\circ T_{5I}$ represents the desired location of the robot end-effector and $({}^\circ\mathbf{T}_5)^{-1}$ is the result of the inverse manipulation if ${}^0\mathbf{T}_{5I}$ has the value of the unit matrix.

Eq.4 implies:

$$ {}^5\mathbf{ROT}_{5I} = {}^5\mathbf{ROT}_\circ . {}^\circ\mathbf{ROT}_{5I} \qquad \text{Eq.5} $$

$$ {}^5\mathbf{ROT}_{5I} = \begin{bmatrix} N_x & N_y & N_z \\ O_x & O_y & O_z \\ A_x & A_y & A_z \end{bmatrix} \cdot \begin{bmatrix} n_x & o_x & a_x \\ n_y & o_y & a_y \\ n_z & o_z & a_z \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \text{Eq.6} $$

Of which we are only interested in the trace vector which is:

$$ \text{Trace} = \begin{bmatrix} N_x.n_x + N_y.n_y + N_z.n_z \\ O_x.o_x + O_y.o_y + O_z.o_z \\ A_x.a_x + A_y.a_y + A_z.a_z \end{bmatrix} \qquad \text{Eq.7} $$

The error matrix will become unit matrix, as soon as the inverse kinematic solutions have been found. Thus, the aim

of the solving process is to make the trace vector equal to $[1,1,1]^T$ by equalizing all the three elements of "Trace" in Eq.7 to one, taking into account that all the ROT-submatrices inthe system are orthonormal. In this way, three equations are available for obtaining the orientation-sensitive variables.

At first, it seems that the three so-obtained equations might not suffice to derive the solutions, whenever the number of the orientation-sensitive variables exceeds 3. In such cases, the solution derivation will be, indeed, not possible if one attempts to proceed the "ordinary way" of substituting and eliminating variables. Saying this, we have not considered yet that we are dealing with the trace vector of the error matrix.

Let us write that trace vector as follows, assuming that we have already found solution for the position-sensitive variables q_k for $k=1,2,...,j$.

$$\text{Trace} = \begin{bmatrix} \text{trace\_x}(q_{j+1}, q_{j+2}, \ldots, q_n) \\ \text{trace\_y}(q_{j+1}, q_{j+2}, \ldots, q_n) \\ \text{trace\_z}(q_{j+1}, q_{j+2}, \ldots, q_n) \end{bmatrix} \qquad \text{Eq.8}$$

Because of the orthogonality (just mentioned above), the elements of "Trace" vary between -1 and +1. Let us consider the vector "Trace" of Eq.8. The **diff_trace_square** value is not necessary for this particular solution derivation due to the relative geometric simplicity of the RM-501. Despite this, we do not rely on geometric intuitions as shown in the following:

$$^{o}ROT_5 = \begin{bmatrix} S_1 \cdot S_{234} \cdot C_5 + C_1 \cdot S_5 & -S_1 \cdot S_{234} \cdot S_5 + C_1 \cdot C_5 & -S_1 \cdot C_{234} \\ -C_1 \cdot S_{234} \cdot C_5 + S_1 \cdot S_5 & C_1 \cdot S_{234} \cdot S_5 + S_1 \cdot C_5 & C_1 \cdot C_{234} \\ C_{234} \cdot C_5 & -C_{234} \cdot S_5 & S_{234} \end{bmatrix} \qquad \text{Eq.9}$$

the vector "Trace" will be:

$$\text{Trace} = \begin{bmatrix} [S_1 \cdot S_{234} \cdot C_5 + C_1 \cdot S_5] \cdot n_x + [-C_1 \cdot S_{234} \cdot C_5 + S_1 \cdot S_5] \cdot n_y \\ + [C_{234} \cdot C_5] \cdot n_z \\ \\ [S_1 \cdot S_{234} \cdot S_5 + C_1 \cdot C_5] \cdot o_x + [+C_1 \cdot S_{234} \cdot S_5 + S_1 \cdot C_5] \cdot o_y \\ + [-C_{234} \cdot S_5] \cdot o_z \\ \\ [-S_1 \cdot C_{234}] \cdot a_x + [C_1 \cdot C_{234}] \cdot a_y \\ + [S_{234}] \cdot a_z \end{bmatrix} \qquad \text{Eq.10}$$

By equalizing "Trace" with the vector $[1,1,1]^T$ we would obtain three equations solvable for θ_4 and θ_5. On the other hand, we know that the vector elements cannot have values greater than one. Thus, let us proceed the following steps.

Taking partial derivative of the third component with respect to θ_4 and requiring it to be zero, we have:

$$-S_{234}.[-S_1.a_x + C_1.a_y] + C_{234}.a_z = 0$$

meaning solution for θ_4

$$\theta_{4a} = ATAN2[+a_z , -S_1.a_x + C_1.a_y] - [\theta_2 + \theta_3]$$

$$\theta_{4b} = ATAN2[-a_z , +S_1.a_x - C_1.a_y] - [\theta_2 + \theta_3]$$

Similarly, taking partial derivative of the first component with respect to θ_5 and requiring it to be zero, we have:

$$-S_5.[S_1.S_{234}.n_x - C_1.S_{234}.n_y + C_{234}.n_z] + C_5.[C_1.n_x + S_1.n_y] = 0$$

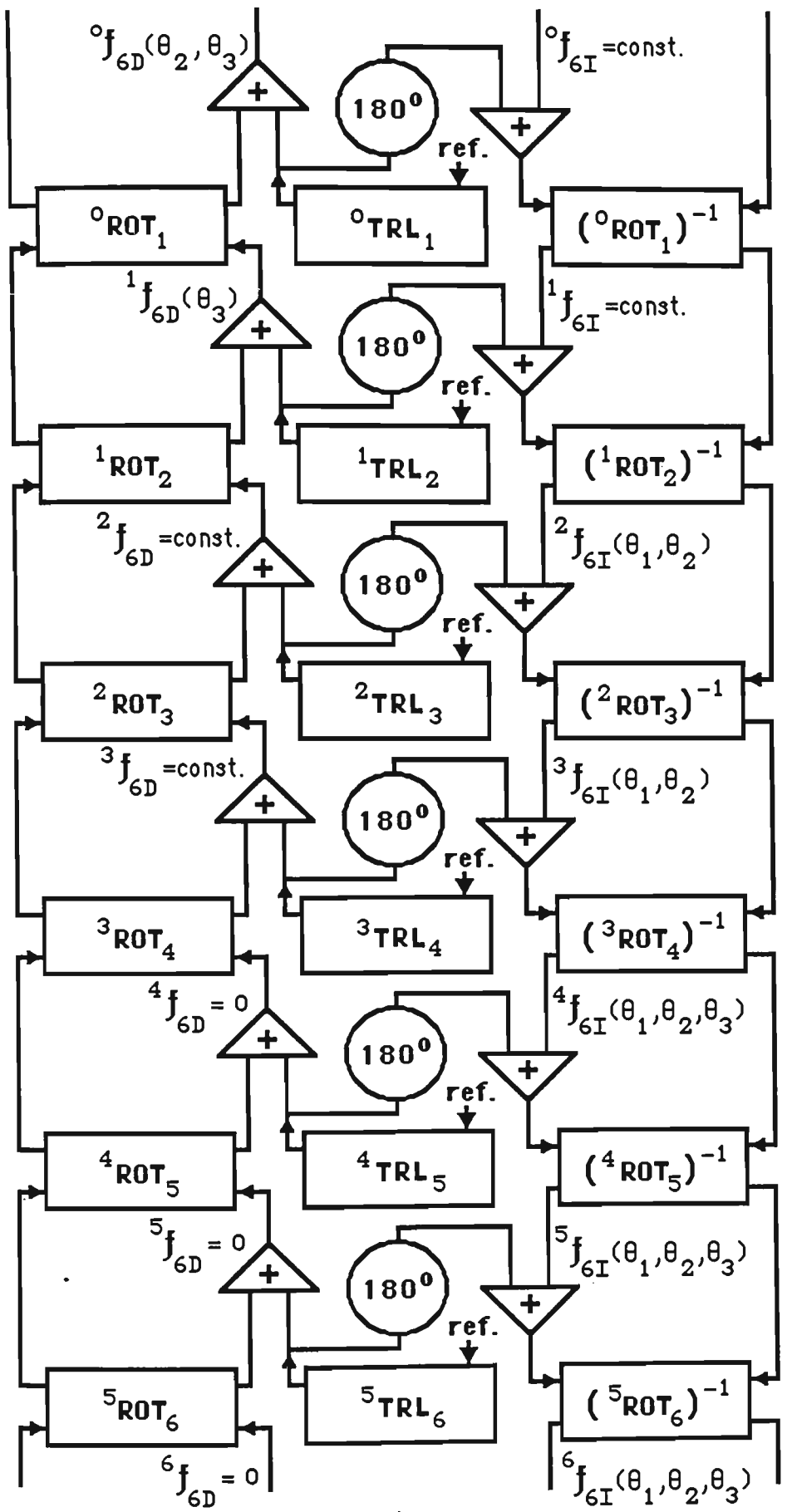
meaning solution for θ_5

$$\theta_{5a} = ATAN2[C_1.n_x + S_1.n_y , S_1.S_{234}.n_x - C_1.S_{234}.n_y + C_{234}.n_z]$$

$$\theta_{5b} = \theta_{5a} \pm 180^0$$

APPENDIX 6

PUMA RRR-(RRR)



$$^{0}A_1=\begin{bmatrix} C_1 & 0 & -S_1 & 0 \\ S_1 & 0 & C_1 & 0 \\ 0 & -1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^{1}A_2=\begin{bmatrix} C_2 & -S_2 & 0 & a_2C_2 \\ S_2 & C_2 & 0 & a_2S_2 \\ 0 & 0 & 1 & d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^{2}A_3=\begin{bmatrix} C_3 & 0 & S_3 & 0 \\ S_3 & 0 & -C_3 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

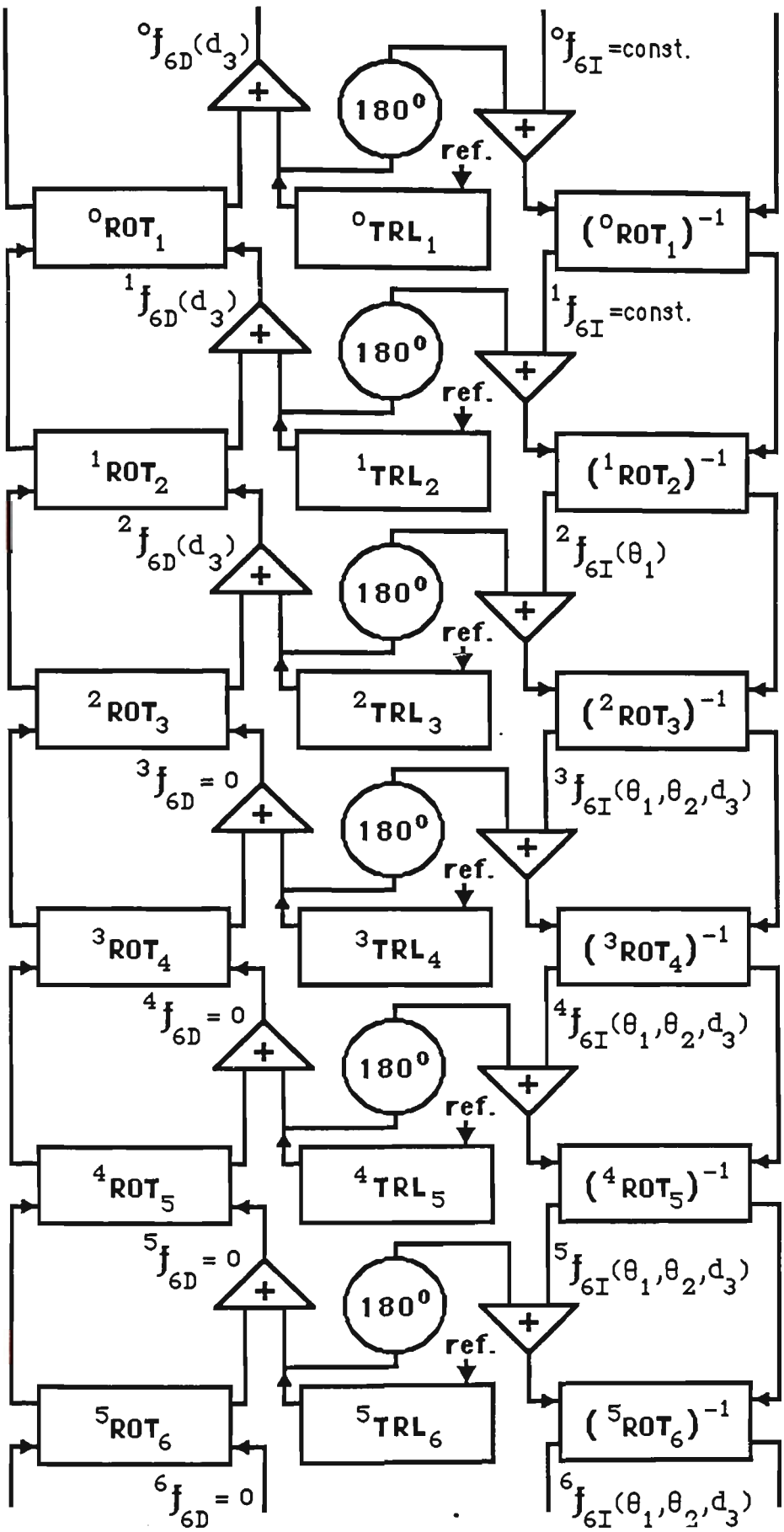
$$^{3}A_4=\begin{bmatrix} C_4 & 0 & -S_4 & 0 \\ S_4 & 0 & C_4 & 0 \\ 0 & -1 & 0 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^{4}A_5=\begin{bmatrix} C_5 & 0 & S_5 & 0 \\ S_5 & 0 & -C_5 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^{5}A_6=\begin{bmatrix} C_6 & -S_6 & 0 & 0 \\ S_6 & C_6 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Solving sequence	Position			Orientation
Variable(s)	θ_3	θ_2	θ_1	$[\theta_4,\theta_5,\theta_6]$ arbitrary order
Error reference	$^{1}f_{6I}$	$^{0}f_{6I}$	$^{6}f_{6D}$	diff_trace_square

STANFORD RRT-(RRR)



$$^{0}A_1 = \begin{bmatrix} C_1 & 0 & -S_1 & 0 \\ S_1 & 0 & C_1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^{1}A_2 = \begin{bmatrix} C_2 & 0 & S_2 & 0 \\ S_2 & 0 & -C_2 & 0 \\ 0 & 1 & 0 & d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^{2}A_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

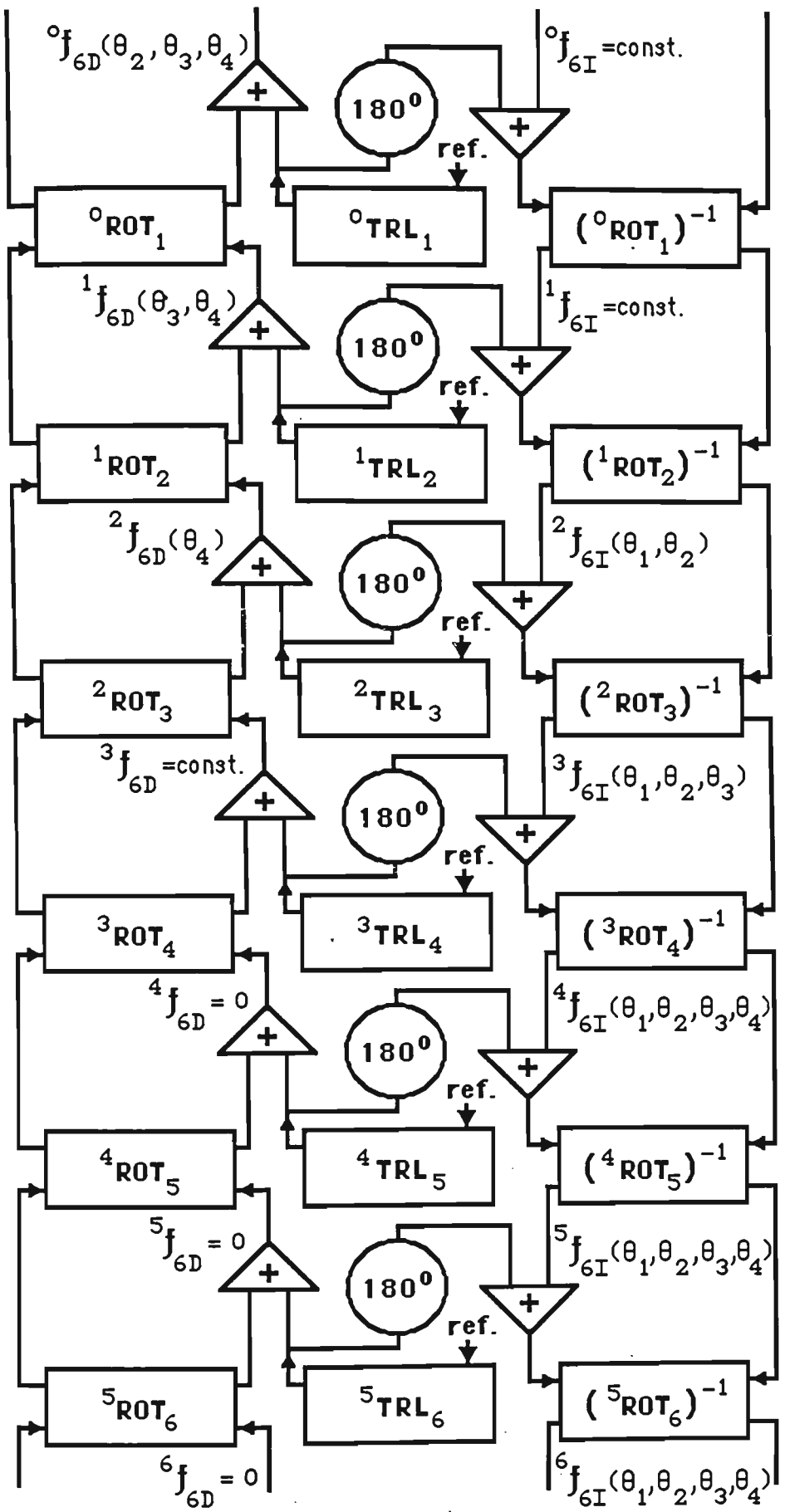
$$^{3}A_4 = \begin{bmatrix} C_4 & 0 & -S_4 & 0 \\ S_4 & 0 & C_4 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^{4}A_5 = \begin{bmatrix} C_5 & 0 & S_5 & 0 \\ S_5 & 0 & -C_5 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^{5}A_6 = \begin{bmatrix} C_6 & -S_6 & 0 & 0 \\ S_6 & C_6 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Solving sequence	Position			Orientation
Variable(s)	d_3	θ_1	θ_2	$[\theta_4, \theta_5, \theta_6]$ arbitrary order
Error reference	$^{0}f_{6I}$	$^{2}f_{6D}$	$^{6}f_{6D}$	diff_trace_square

ELBOW RRRR-(RR)



$$^0A_1=\begin{bmatrix} C_1 & 0 & S_1 & 0 \\ S_1 & 0 & -C_1 & 0 \\ 0 & 1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^1A_2=\begin{bmatrix} C_2 & -S_2 & 0 & a_2C_2 \\ S_2 & C_2 & 0 & a_2S_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^2A_3=\begin{bmatrix} C_3 & -S_3 & 0 & a_3C_3 \\ S_3 & C_3 & 0 & a_3S_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^3A_4=\begin{bmatrix} C_4 & 0 & -S_4 & a_4C_4 \\ S_4 & 0 & C_4 & a_4S_4 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^4A_5=\begin{bmatrix} C_5 & 0 & S_5 & 0 \\ S_5 & 0 & -C_5 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^5A_6=\begin{bmatrix} C_6 & -S_6 & 0 & 0 \\ S_6 & C_6 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Solving sequence	Position			Orientation
Variable(s)	$[\theta_3,\theta_4]$	θ_2	θ_1	$[\theta_5,\theta_6]$ arbitrary order
Error reference	$^1f_{6I}$	$^0f_{6I}$	$^6f_{6D}$	diff_trace_square

APPENDIX 7

```
Printer #1 on
```

```
 1  000000      (*************************************************************************)
 2  000000
 3  000000      (*
 4  000000         ------------------------
 5  000000         ELEMENT     RM50100-CM              ; EXTERNAL ELEMENTS: SEE BELOW
 6  000000         ------------------------
 7  000000
 8  000000         SOLVING ROBOT KINEMATICS INVERSE PROBLEM
 9  000000         WITHOUT EXPLICIT ARITHMETIC EXPRESSIONS.
10  000000         (This File Element: Global/Common Definitions)
11  000000
12  000000         SECOND  UNIVAC VERSION              10/08/84          BY T.Vd./
13  000000         LAST    IMPLEMENTATION              10/31/84
14  000000                                            (mm/dd/yy)
15  000000
16  000000
17  000000                                                  UNIVERSITY OF WOLLONGONG
18  000000                                            DEPARTMENT OF ELECTRICAL ENGINEERING
19  000000
20  000000
21  000000         EXTERNAL ELEMENTS:
22  000000         ------------------
23  000000
24  000000                     03) RM50100-3A
25  000000                     02) RM50100-2A
26  000000                     01) RM50100-1A
27  000000
28  000000         Total number of external elements:          03
29  000000
30  000000      *)
31  000000
32  000000      (*************************************************************************)
33  000000      ENVIRONMENT
34  000000      DECLARATIONS ;
35  000245      (*************************************************************************)
36  000245      CONST
37  000245              PI      =    3.1415926536 ;
38  000245              IRE     =  360 ;          (* Variable Resolution                  *)
39  000245              N       =    5 ;          (* Robot's Number of Degrees of Freedom *)
40  000245              M       =    3 ;          (* No. of D.O.F. without the End-Effector *)
41  000245
42  000245      (*************************************************************************)
43  000245      TYPE
44  000245
45  000245              (* Cartesian Co-Ordinates and Projection Factors ..... *)
46  000245 [00]         VECTOR  =  RECORD
47  000245 [01]                       X,Y,Z,W : REAL
48  000245 [01]                   END ;
49  000245 [00]
50  000245              (* Normal, Orientation, Approach & Position Vector ..... *)
51  000245 [00]         MATRIX  =  RECORD
52  000245 [01]                       N,O,A,P : VECTOR
53  000245 [01]                   END ;
54  000245 [00]
55  000245              (* Robot with    End-Effector ..................... *)
56  000245              ARM     =  ARRAY[0..N]   OF MATRIX   ;
57  000247
58  000247              (* Robot without End-Effector ....................... *)
59  000247              WRIST   =  ARRAY[0..M]   OF MATRIX   ;
60  000251
61  000251              (* End-Effector ..................................... *)
62  000251              HAND    =  ARRAY[M..N]   OF MATRIX   ;
63  000253
64  000253              (* Robot's Geometry Parameters ...................... *)
65  000253 [00]         PARAM   =  RECORD
66  000253 [01]                    IOFST,ITHETA : INTEGER    ;
67  000253                           ALFA,AX,DZ : REAL
68  000253 [01]                   END ;
69  000253 [00]
70  000253              (* System Error ..................................... *)
71  000253              TYPER   =  ARRAY[1..IRE] OF REAL    ;
72  000255              TYPEI   =  ARRAY[1..N]   OF INTEGER ;
73  000257 [00]         ERRDAT  =  RECORD
74  000257 [01]                     ERPAR       : TYPEI                    ;
75  000257                          ERVAR       : TYPER                    ;
76  000257 [01]                   END ;
77  000257 [00]
```

```
 78  000257      (*****************************************************************************)
 79  000257      VAR
 80  000257
 81  000257              (* GLOBAL PARAMETERS ------------------------------------------- *)
 82  000257
 83  000257              (* Robot's Geometry Parameters ..................................... *)
 84  000257              LINK     :  ARRAY[1..N]   OF PARAM     ;
 85  000312
 86  000312              (* Reference Sinewave Signal ....................................... *)
 87  000312              REF      :  ARRAY[1..IRE] OF REAL      ;
 88  001064
 89  001064              (* GLOBAL VARIABLES -------------------------------------------- *)
 90  001064
 91  001064              (* A-matrices ... AH homogeneous Matrix ... HA its Inverse ...... *)
 92  001064              AH, HA   :  ARRAY[1..N]   OF MATRIX    ;
 93  001326
 94  001326              (* T-Matrices ...................................................... *)
 95  001326              (* Convention of Variable-Names: 'F' Forward , 'B' Backward ..... *)
 96  001326              TNF,TNB  :  ARM                        ;
 97  001626              TMF,TMB  :  WRIST                      ;
 98  002026              EMF,EMB  :  HAND                       ;
 99  002166
100  002166              (* Euler-Angles .................................................... *)
101  002166              EULER0 ,
102  002166              EULERY ,
103  002166              EULER1   :  INTEGER                    ;
104  002171
105  002171              (* Error References ........... Forward/Backward ................. *)
106  002171              ERB,ERF  :  ARRAY[0..N]   OF REAL      ;
107  002207              FEHLER   :  ARRAY[1..N]   OF ERRDAT    ;
108  005652
109  005652              (* Components of Backward Orientator Input Position Path ........ *)
110  005652              G        :  VECTOR                     ;
111  005656
112  005656              (* Unit Matrix, Renaming Forward Input, Backward Output ......... *)
113  005656              U,TF,TB  :  MATRIX                     ;
114  005736
115  005736              (* Help Variables .................................................. *)
116  005736              JA       :  CHAR                       ;
117  005737              I,J,K,L  :  INTEGER                    ;
118  005743
```

```
119  005743      (*****************************************************************************)
120  005743      (* ROUTINE HEAD LIST  /empty/                                                *)
121  005743      (*****************************************************************************)
122  005743
123  005743      (* ------------------------------------ FROM FILE ELEMENT RM50100-3A   *)
124  005743
125. 005743      PROCEDURE
126  005743      EQMTRX (VAR TI,OT:MATRIX) ,                                    (* 07/10/84 *)
127  005743
128  005743      FUNCTION
129  005743      SINUS  (T:REAL) : REAL ,                                       (* 07/10/84 *)
130  005743
131  005743      (* ------------------------------------ FROM FILE ELEMENT RM50100-2A   *)
132  005743
133  005743      FUNCTION
134  005743      WAVE01 (I,JANGLE:INTEGER ; R,S:REAL) : REAL ,                  (* 07/03/84 *)
135  005743
136  005743      PROCEDURE
137  005743      ROTATE (K:CHAR ; IANGLE:INTEGER ; VAR SI,SO:VECTOR) ,          (* 07/03/84 *)
138  005743
139  005743      PROCEDURE
140  005743      ROTMAT (K:CHAR ; IANGLE:INTEGER ; VAR TI,OT:MATRIX) ,          (* 07/05/84 *)
141  005743
142  005743      PROCEDURE
143  005743      REVECT (I:INTEGER ; K:CHAR ; VAR V:VECTOR) ,                   (* 07/05/84 *)
144  005743
145  005743      PROCEDURE
146  005743      REMTRX (I:INTEGER ; K:CHAR ; VAR TM:MATRIX) ,                  (* 07/05/84 *)
147  005743
148  005743      PROCEDURE
149  005743      ATRANS (I:INTEGER) ,                                           (* 07/06/84 *)
150  005743
151  005743      PROCEDURE
152  005743      ARJD00 (I:INTEGER ; K:CHAR ; VAR TI,OT:MATRIX) ,               (* 07/06/84 *)
153  005743
154  005743      PROCEDURE
155  005743      STATUS (I,JB,JF,MN:INTEGER) ,                                  (* 07/08/84 *)
156  005743
157  005743      PROCEDURE
158  005743      SOLV00 (I,J,MN:INTEGER ; C:CHAR) ,                             (* 09/26/84 *)
159  005743
160  005743      (* ------------------------------------ FROM FILE ELEMENT RM50100-1A   *)
161  005743
162  005743      PROCEDURE
163  005743      PRINTT (I,L:INTEGER ; F,B:MATRIX) ,                            (* 07/09/84 *)
164  005743
165  005743      PROCEDURE
166  005743      PPRINT (J:INTEGER) ,                                           (* 07/09/84 *)
167  005743
168  005743      PROCEDURE
169  005743      INITIA ,                                                       (* 07/09/84 *)
170  005743
171  005743      (* ------------------------------------ FROM FILE ELEMENT DES-PLOT00   *)
172  005743
173  005743      FORTRAN77 PROCEDURE
174  005743      STPLOT ,                                                       (* 10/29/84 *)
175  005743
176  005743      FORTRAN77 PROCEDURE
177  005743      PLOT (VAR DATA:TYPER; NN:INTEGER; VAR IP:TYPEI; MM,J1,J2,J3,J4,J5:INTEGER) ,
178  005743                                                                     (* 10/29/84 *)
179  005743
180  005743      FORTRAN77 PROCEDURE
181  005743      ENDPLT                                                         (* 10/29/84 *)
182  005743
183  005743                                                                     (* mm/dd/yy *)
184  005743
185  005743      (*****************************************************************************)
186  005743      .                          (* TERMINATION PERIOD  end of element RM50100-CM   *)
187  005743      (*****************************************************************************)
Compilation completed - no errors found.
FURPUR 28R3A W1 S74T11 07/11/85 13:57:43
RO7TVD*G$FILE$(1)
```

```
*THIS FILE CONTAINS DELETED TEXT TOTALING     4 TRACKS
)@PAS,S F.RM50100-3A,F.RM50100-3A,,fKF.RM50100-CM

University of Wisconsin  P A S C A L  (8:A:1)  07/11/85 14:00:05 (3)
)@EOF

     1  000000  (*************************************************************************************)
     2  000000
     3  000000  (*
     4  000000     ------------------------
     5  000000     ELEMENT     RM50100-3A                ; EXTERNAL ELEMENTS: SEE BELOW
     6  000000     ------------------------
     7  000000
     8  000000     SOLVING ROBOT KINEMATICS INVERSE PROBLEM
     9  000000     WITHOUT EXPLICIT ARITHMETIC EXPRESSIONS.
    10  000000     (This File Element: Auxiliary Help Func./Proc.)
    11  000000
    12  000000     SECOND  UNIVAC VERSION                 10/08/84              BY T.Vd./
    13  000000     LAST    IMPLEMENTATION                 10/22/84
    14  000000                                            (mm/dd/yy)
    15  000000
    16  000000
    17  000000                                                  UNIVERSITY OF WOLLONGONG
    18  000000                                            DEPARTMENT OF ELECTRICAL ENGINEERING
    19  000000
    20  000000
    21  000000
    22  000000                USER PROCEDURES AND FUNCTIONS CALLED:
    23  000000                ------------------------------------
    24  000000                (Listed     in     appearing     order)
    25  000000
    26  000000                RELATIVE            : EXTERNAL/T/C :                :
    27  000000                LEVEL  #            :  T: TEXT     : PROC./FUNC.  :     LANGUAGE
    28  000000          1          2       3,4,5  :  C: CODE     :              :
    29  000000        .........:.........:.........:.............:..............:.................
    30  000000        EQMTRX                      :     NO        :  PROCEDURE   :      PASCAL
    31  000000        .........:.........:.........:.............:..............:.................
    32  000000        SINUS                       :     NO        :  FUNCTION    :      PASCAL
    33  000000        .........:.........:.........:.............:..............:.................
    34  000000                  SIN               :     NO        :  FUNCTION    : LIB PASCAL FUNC.
    35  000000        .........:.........:.........:.............:..............:.................
    36  000000        * indicates local Proc./Func.
    37  000000
    38  000000
    39  000000        TOTAL NUMBER OF PROCEDURES          :       01
    40  000000                        FUNCTIONS           :       01
    41  000000                        ----------                  ----
    42  000000                        T O T A L           :       02          C A L L E D
    43  000000
    44  000000        EXTERNAL ELEMENT(S)                 :            (NONE)
    45  000000
    46  000000        LOADING  ENVIRONMENT                :            RM50100-CM
    47  000000
    48  000000        NUMBER OF PROCEDURES AND FUNCTIONS
    49  000000        CONTAINED IN THE CURRENT  ELEMENT   :       02
    50  000000
    51  000000        NUMBER OF PROCEDURES AND FUNCTIONS
    52  000000        CALLED  FROM THE EXTERNAL ELEMENT(S):       00
    53  000000
    54  000000        NUMBER OF PROCEDURES AND FUNCTIONS
    55  000000        CALLED  FROM PASCAL LIBRARY         :       01
    56  000000        *)
    57  000000
```

```
58  000000       (*****************************************************************************)
59  000000       ENVIRONMENT
60  005743       ROUTINES ( EQMTRX , SINUS  ) ;
61  000016       (*****************************************************************************)
62  000016       PROCEDURE
63  000016       EQMTRX (VAR TI,OT:MATRIX) ;                                        (* 07/10/84 *)
64  000000
65  000000       (*
66  000000         PURPOSES:
67  000000         --------
68  000000         ... EQUALIZING MATRICES ELEMENT BY ELEMENT
69  000000         PARAMETERS IN APPEARING ORDER:
70  000000         ------------------------------
71  000000             TI     :  INPUT MATRIX                                      INPUT
72  000000             OT     : OUTPUT MATRIX                                     OUTPUT
73  000000         GLOBAL PARAMETERS:
74  000000         ------------------
75  000000                      (NONE)
76  000000         GLOBAL  VARIABLES:
77  000000         ------------------
78  000000                      (NONE)
79  000000         LOCAL   VARIABLES:
80  000000         ------------------
81  000000                      (NONE)
82  000000         PROCEDURES AND/OR FUNCTIONS CALLED:
83  000000         -----------------------------------
84  000000                      (NONE)
85  000000         COMMENTS:
86  000000         --------
87  000000         ... TYPE 'MATRIX' MUST BE DEFINED IN THE APPROPRIATE LOADING ENVIRONMENT
88  000000       *)
89  000007
90  000007 [00] BEGIN (*EQMTRX*)
91  000000 [01]
92  000005         OT.N.X:=TI.N.X ; OT.O.X:=TI.O.X ; OT.A.X:=TI.A.X ; OT.P.X:=TI.P.X  ;
93  000025         OT.N.Y:=TI.N.Y ; OT.O.Y:=TI.O.Y ; OT.A.Y:=TI.A.Y ; OT.P.Y:=TI.P.Y  ;
94  000045         OT.N.Z:=TI.N.Z ; OT.O.Z:=TI.O.Z ; OT.A.Z:=TI.A.Z ; OT.P.Z:=TI.P.Z  ;
95  000065         OT.N.W:=TI.N.W ; OT.O.W:=TI.O.W ; OT.A.W:=TI.A.W ; OT.P.W:=TI.P.W  ;
96  000105
97  000007 [01] END   (*EQMTRX*)  ;
98  000016 [00]
99  000016       (*****************************************************************************)
100 000016       FUNCTION
101 000016       SINUS  (T:REAL) : REAL  ;                                          (* 07/10/84 *)
102 000000
103 000000       (*
104 000000         PURPOSES:
105 000000         --------
106 000000         ... RETURN SINE VALUE TO CALLING ROUTINE
107 000000         PARAMETERS IN APPEARING ORDER:
108 000000         ------------------------------
109 000000            SINUS  : FUNCTION VALUE RETURNED                             OUTPUT
110 000000              T    : FUNCTION ARGUMENT                                    INPUT
111 000000         GLOBAL PARAMETERS:
112 000000         ------------------
113 000000                      (NONE)
114 000000         GLOBAL  VARIABLES:
115 000000         ------------------
116 000000                      (NONE)
117 000000         LOCAL   VARIABLES:
118 000000         ------------------
119 000000                      (NONE)
120 000000         PROCEDURES AND/OR FUNCTIONS CALLED:
121 000000         -----------------------------------
122 000000           'SIN' : STANDARD LIBRARY FUNCTION
123 000000         COMMENTS:
124 000000         --------
125 000000         ... STANDARD FUNCTIONS CANNOT BE USED AS ACTUAL PARAMETERS (UW-PASCAL)
126 000000             THAT IS WHY ...
127 000000       *)
128 000006
129 000006 [00] BEGIN (*SINUS *)
130 000107 [01]
131 000114               SINUS:=SIN(T)
132 000117
133 000006 [01] END   (*SINUS *) ;
134 000016 [00]
135 000016       (*****************************************************************************)
136 000016       .                          (* TERMINATION PERIOD  end of element RM50100-3A   *)
137 000016       .(*****************************************************************************)
Compilation completed - no errors found.
```

```
)@PAS,S F.RM50100-2A,F.RM50100-2A,,F.RM50100-CM

University of Wisconsin  P A S C A L  (8:A:1)  07/11/85 14:03:00 (25)
)@EOF

000000 (*************************************************************************)
000000
000000 (*
000000    ----------------------
000000    ELEMENT    RM50100-2A              ; EXTERNAL ELEMENTS: SEE BELOW
000000    ----------------------
000000
000000    SOLVING ROBOT KINEMATICS INVERSE PROBLEM
000000    WITHOUT EXPLICIT ARITHMETIC EXPRESSIONS.
000000    (This File Element: Heart of the Simulation)
000000
000000    SECOND  UNIVAC VERSION              10/08/84          BY T.Vd./
000000    LAST    IMPLEMENTATION              10/31/84
000000                                        (mm/dd/yy)
000000
000000
000000                                          UNIVERSITY OF WOLLONGONG
000000                                        DEPARTMENT OF ELECTRICAL ENGINEERING
000000
000000
000000
000000          USER PROCEDURES AND FUNCTIONS CALLED:
000000          ------------------------------------
000000          (Listed    in    appearing    order)
000000
000000          RELATIVE          : EXTERNAL/T/C :                :
000000          LEVEL  #          :  T: TEXT     : PROC./FUNC.   :    LANGUAGE
000000     1       2       3,4,5  :  C: CODE     :               :
000000 .....:.....:.....:.....:.....:.....:.....:.....:.....:.....:.....:.....
000000 WAVE01                     :     NO       : FUNCTION      :     PASCAL
000000 .....:.....:.....:.....:.....:.....:.....:.....:.....:.....:.....:.....
000000 ROTATE                     :     NO       : PROCEDURE     :     PASCAL
000000 .....:.....:.....:.....:.....:.....:.....:.....:.....:.....:.....:.....
000000          wave01            :              :               :
000000 .....:.....:.....:.....:.....:.....:.....:.....:.....:.....:.....:.....
000000 ROTMAT                     :     NO       : PROC.         :     PASCAL
000000 .....:.....:.....:.....:.....:.....:.....:.....:.....:.....:.....:.....
000000          rotate            :              :               :
000000 .....:.....:.....:.....:.....:.....:.....:.....:.....:.....:.....:.....
000000                wave01      :              :               :
000000 .....:.....:.....:.....:.....:.....:.....:.....:.....:.....:.....:.....
000000 REVECT                     :     NO       : PROC.         :     PASCAL
000000 .....:.....:.....:.....:.....:.....:.....:.....:.....:.....:.....:.....
000000          rotate            :              :               :
000000 .....:.....:.....:.....:.....:.....:.....:.....:.....:.....:.....:.....
000000          TRUNC             :     NO       : FUNC.         : LIB PASCAL FUNC.
000000 .....:.....:.....:.....:.....:.....:.....:.....:.....:.....:.....:.....
000000 REMTRX                     :     NO       : PROC.         :     PASCAL
000000 .....:.....:.....:.....:.....:.....:.....:.....:.....:.....:.....:.....
000000          revect            :              :               :
000000 .....:.....:.....:.....:.....:.....:.....:.....:.....:.....:.....:.....
000000          EQMTRX            :3A) YES/C     : PROC.         :     PASCAL
000000 .....:.....:.....:.....:.....:.....:.....:.....:.....:.....:.....:.....
000000 ATRANS                     :     NO       : PROC.         :     PASCAL
000000 .....:.....:.....:.....:.....:.....:.....:.....:.....:.....:.....:.....
000000          revect            :              :               :
000000 .....:.....:.....:.....:.....:.....:.....:.....:.....:.....:.....:.....
000000          remtrx            :              :               :
000000 .....:.....:.....:.....:.....:.....:.....:.....:.....:.....:.....:.....
000000                revect      :              :               :
000000 .....:.....:.....:.....:.....:.....:.....:.....:.....:.....:.....:.....
000000                eqmtrx      :              :               :
000000 .....:.....:.....:.....:.....:.....:.....:.....:.....:.....:.....:.....
000000          rotate            :              :               :
000000 .....:.....:.....:.....:.....:.....:.....:.....:.....:.....:.....:.....
000000                wave01      :              :               :
000000 .....:.....:.....:.....:.....:.....:.....:.....:.....:.....:.....:.....
000000 ARJDOO                     :     NO       : PROC.         :     PASCAL
000000 .....:.....:.....:.....:.....:.....:.....:.....:.....:.....:.....:.....
000000          remtrx            :              :               :
000000 .....:.....:.....:.....:.....:.....:.....:.....:.....:.....:.....:.....
000000                revect      :              :               :
000000 .....:.....:.....:.....:.....:.....:.....:.....:.....:.....:.....:.....
000000                eqmtrx      :              :               :
000000 .....:.....:.....:.....:.....:.....:.....:.....:.....:.....:.....:.....
```

```
000000
000000          rotate                :              :              :
000000 .........:.........:.........:..............:..............:..............
000000                          wave01  :              :              :
000000 .........:.........:.........:..............:..............:..............
000000 STATUS                         :      NO      :  PROC.       :     PASCAL
000000 .........:.........:.........:..............:..............:..............
000000          atrans                :              :              :
000000 .........:.........:.........:..............:..............:..............
000000                          revect  :              :              :
000000 .........:.........:.........:..............:..............:..............
000000                          remtrx  :              :              :
000000                          revect(4):             :              :
000000                          eqmtrx(4):             :              :
000000 .........:.........:.........:..............:..............:..............
000000                          rotate  :              :              :
000000                          wave01(4):             :              :
000000 .........:.........:.........:..............:..............:..............
000000          arjd00                :              :              :
000000 .........:.........:.........:..............:..............:..............
000000                          remtrx  :              :              :
000000                          revect(4):             :              :
000000                          eqmtrx(4):             :              :
000000 .........:.........:.........:..............:..............:..............
000000                          rotate  :              :              :
000000                          wave01(4):             :              :
000000 .........:.........:.........:..............:..............:..............
000000 SOLV00                         :      NO      :  PROC.       :     PASCAL
000000 .........:.........:.........:..............:..............:..............
000000          *sqrmag               :      NO      :  FUNC.       :     PASCAL
000000 .........:.........:.........:..............:..............:..............
000000          *poserr               :      NO      :  FUNC.       :     PASCAL
000000 .........:.........:.........:..............:..............:..............
000000          *orierr               :      NO      :  FUNC.       :     PASCAL
000000 .........:.........:.........:..............:..............:..............
000000          *error0               :      NO      :  FUNC.       :     PASCAL
000000 .........:.........:.........:..............:..............:..............
000000             ABS                :      NO      :  FUNCTION    : LIB PASCAL FUNC
000000 .........:.........:.........:..............:..............:..............
000000 * indicates local Proc./Func.
000000
000000
000000 TOTAL NUMBER OF PROCEDURES          :      09
000000              FUNCTIONS              :      03
000000              ----------             :     ----
000000              T O T A L              :      12          C A L L E D
000000
000000 EXTERNAL ELEMENT(S)                 :      3A) RM50100-3A
000000
000000 LOADING  ENVIRONMENT                :          RM50100-CM .
000000
000000 NUMBER OF PROCEDURES AND FUNCTIONS
000000 CONTAINED IN THE CURRENT  ELEMENT   :      09
000000
000000 NUMBER OF PROCEDURES AND FUNCTIONS
000000 CALLED  FROM THE EXTERNAL ELEMENT(S):      01
000000
000000 NUMBER OF PROCEDURES AND FUNCTIONS
000000 CALLED  FROM PASCAL LIBRARY         :      02
000000 *)
000000
```

```
136  000000       (*****************************************************************************)
137  000000       ENVIRONMENT
138  005743       ROUTINES ( WAVE01 , ROTATE , ROTMAT , REVECT ,
139  005743                  REMTRX , ATRANS , ARJD00 , STATUS , SOLV00 ) ;
140  000016       (*****************************************************************************)
141  000016       FUNCTION
142  000016       WAVE01 (I,JANGLE:INTEGER ; R,S:REAL) : REAL ;                    (* 07/03/84 *)
143  000000
144  000000       (*
145  000000         PURPOSES:
146  000000         --------
147  000000         ... SAMPLING OUTPUT SIGNAL FROM THE PHASE SHIFTER
148  000000         PARAMETERS IN APPEARING ORDER:
149  000000         ------------------------------
150  000000           WAVE01   : SIGNAL SAMPLE                                    OUTPUT
151  000000              I     : SAMPLING POINT                                    INPUT
152  000000           JANGLE   : DELAY ANGLE                                       INPUT
153  000000              R     : REAL       SIGNAL PART WITH RESP. TO REF.         INPUT
154  000000              S     : IMAGINARY SIGNAL PART WITH RESP. TO REF.          INPUT
155  000000         GLOBAL PARAMETERS:
156  000000         ------------------
157  000000             REF    : ARRAY CONTAINING REFERENCE SIGNAL                 INPUT
158  000000             IRE    : VARIABLE RESOLUTION                               INPUT
159  000000         GLOBAL  VARIABLES:
160  000000         ------------------
161  000000                     (NONE)
162  000000         LOCAL   VARIABLES:
163  000000         ------------------
164  000000              J     : (SEE BELOW)
165  000000         PROCEDURES AND/OR FUNCTIONS CALLED:
166  000000         -----------------------------------
167  000000                     (NONE)
168  000000         COMMENTS:
169  000000         --------
170  000000         ... SIGNAL TIME RESOLUTION IS 'IRE', SAMPLING AT 'IRE/4' AND 'IRE/2'
171  000000       *)
172  000011
173  000011       VAR J,K,L : INTEGER ;                                  (* LOCAL VARIABLES *)
174  000014
175  000014 [00] BEGIN (*WAVE01*)
176  000000 [01]
177  000005         K := (3*IRE) DIV 4 ;
178  000007         L := (1*IRE) DIV 4 ;
179  000011
180  000011         J:=I-JANGLE      ;
181  000014         IF J<1 THEN J:=J+IRE ;
182  000022              IF J<L+1
183  000030              THEN WAVE01:=R*REF[J] + S*REF[J+K]
184  000054              ELSE WAVE01:=R*REF[J] + S*REF[J-L]
185  000065
186  000014 [01] END    (*WAVE01*)   ;
187  000016 [00]
```

```
188 000016       (*************************************************************************)
189 000016       PROCEDURE
190 000016       ROTATE (K:CHAR ; IANGLE:INTEGER ; VAR SI,SO:VECTOR) ;          (* 07/03/84 *)
191 000000
192 000000       (*
193 000000         PURPOSES:
194 000000         --------
195 000000         ... ELEMENTARY   O R I E N T A T O R
196 000000         ... VECTOR COMBINER ... DELAY ... VECTOR SPLITTER
197 000000         PARAMETERS IN APPEARING ORDER:
198 000000         ------------------------------
199 000000           K       : MODE "X" , "Y" OR "Z"    (ROTATING AXIS)          INPUT
200 000000           IANGLE  : ROTATING ANGLE CALL-BY-VALUE                      INPUT
201 000000           SI      : VECTOR BEFORE DELAY (ROTATING)                    INPUT
202 000000           SO      : VECTOR AFTER  DELAY                              OUTPUT
203 000000         GLOBAL PARAMETERS:
204 000000         ------------------
205 000000             IRE   : VARIABLE RESOLUTION                              INPUT
206 000000         GLOBAL  VARIABLES:
207 000000         ------------------
208 000000                     (NONE)
209 000000         LOCAL    VARIABLES:
210 000000         ------------------
211 000000           JANGLE  : ROTATING ANGLE PASSED TO 'WAVE01'
212 000000         R,S,UU,VV : MANIPULATED VECTOR COMPONENTS
213 000000                     (SEE BELOW)
214 000000         PROCEDURES AND/OR FUNCTIONS CALLED:
215 000000         -----------------------------------
216 000000          'WAVE01' : REAL FUNCTION CONTAINED IN THE CURRENT ELEMENT
217 000000         COMMENTS:
218 000000         --------
219 000000         ... SIGNAL TIME RESOLUTION IS 'IRE', SAMPLING AT 'IRE/4' AND 'IRE/2'
220 000000         ... TYPE 'VECTOR' MUST BE DEFINED IN THE APPROPRIATE LOADING ENVIRONMENT
221 000000         ... ASSUMING RIGHT-HANDED CO-ORDINATE SYSTEMS
222 000000       *)
223 000011
224 000011       VAR   JANGLE,KX,KY: INTEGER ;                    (* LOCAL VARIABLES *)
225 000014           R,S,  UU,VV   : REAL    ;
226 000020
227 000020 [00] BEGIN (*ROTATE*)
228 000101 [01]
229 000106 [01]   CASE K OF 'Z' : BEGIN
230 000110 [03]                     R:=SI.X ; S:=SI.Y    (* EIGEN-VALUE IS Z *)
231 000116 [03]                   END ;
232 000117 [02]             'Y' : BEGIN
233 000117 [03]                     R:=SI.Z ; S:=SI.X    (* EIGEN-VALUE IS Y *)
234 000125 [03]                   END ;
235 000126 [02]             'X' : BEGIN
236 000126 [03]                     R:=SI.Y ; S:=SI.Z    (* EIGEN-VALUE IS X *)
237 000134 [03]                   END ;
238 000145 [02]   END (*CASE 1*) ;
239 000145 [01]
240 000145        UU:=0.0 ; VV:=0.0 ; KX:=IRE DIV 4 ; KY:=IRE DIV 2 ;
241 000153
242 000153        IF R*R + S*S > 0.0 THEN
243 000163 [01]     BEGIN
244 000163 [02]       JANGLE:=IANGLE ;
245 000165            IF JANGLE<0 THEN JANGLE:=IRE + JANGLE ;
246 000174            JANGLE:=JANGLE MOD IRE                ;
247 000203            UU:=WAVE01(KX,JANGLE,R,S) ;
248 000217            VV:=WAVE01(KY,JANGLE,R,S) ;
249 000233 [02]     END ;
250 000233 [01]
251 000233 [01]   CASE K OF 'Z' : BEGIN
252 000235 [03]                     SO.X:=UU ; SO.Y:=VV ; SO.Z:=SI.Z
253 000247 [03]                   END ;
254 000250 [02]             'Y' : BEGIN
255 000250 [03]                     SO.Z:=UU ; SO.X:=VV ; SO.Y:=SI.Y
256 000262 [03]                   END ;
257 000263 [02]             'X' : BEGIN
258 000263 [03]                     SO.Y:=UU ; SO.Z:=VV ; SO.X:=SI.X
259 000275 [03]                   END ;
260 000306 [02]   END (*CASE 2*) ;
261 000306 [01]
262 000020 [01] END   (*ROTATE*) ;
263 000016 [00]
```

```
264 000016       (*************************************************************************)
265 000016       PROCEDURE
266 000016       ROTMAT (K:CHAR ; IANGLE:INTEGER ; VAR TI,OT:MATRIX) ;          (* 07/05/84 *)
267 000000
268 000000       (*
269 000000         PURPOSES:
270 000000         --------
271 000000         ... ROTATE THE WHOLE COORDINATE SYSTEM (HOMOGENEOUS TRANSFORMATION MATRIX)
272 000000             WITHOUT REGARD TO THE OFFSET FROM THE   P O S I T A T O R
273 000000         PARAMETERS IN APPEARING ORDER:
274 000000         -----------------------------
275 000000           K        : MODE "X" , "Y" OR "Z"    (ROTATING AXIS)              INPUT
276 000000           IANGLE   : ROTATING ANGLE CALL-BY-VALUE                          INPUT
277 000000           TI       : MATRIX BEFORE DELAY (ROTATING)                        INPUT
278 000000           OT       : MATRIX AFTER  DELAY                                  OUTPUT
279 000000         GLOBAL PARAMETERS:
280 000000         -----------------
281 000000                     (NONE)
282 000000         GLOBAL  VARIABLES:
283 000000         -----------------
284 000000                     (NONE)
285 000000         LOCAL   VARIABLES:
286 000000         -----------------
287 000000                     (NONE)
288 000000         PROCEDURES AND/OR FUNCTIONS CALLED:
289 000000         ----------------------------------
290 000000          'ROTATE' : VECTOR-MANIPULATING PROCEDURE CONTAINED IN THE CURRENT ELEMENT
291 000000         COMMENTS:
292 000000         --------
293 000000         ... TYPE 'MATRIX' MUST BE DEFINED IN THE APPROPRIATE LOADING ENVIRONMENT
294 000000       *)
295 000011
296 000011 [00] BEGIN (*ROTMAT*)
297 000310 [01]
298 000315         ROTATE (K,IANGLE,TI.N,OT.N) ;
299 000332         ROTATE (K,IANGLE,TI.O,OT.O) ;
300 000347         ROTATE (K,IANGLE,TI.A,OT.A) ;
301 000364         ROTATE (K,IANGLE,TI.P,OT.P)
302 000401
303 000011 [01] END   (*ROTMAT*)  ;
304 000016 [00]
305 000016       (*************************************************************************)
306 000016       PROCEDURE
307 000016       REVECT (I:INTEGER ; K:CHAR ; VAR V:VECTOR) ;                   (* 07/05/84 *)
308 000000
309 000000       (*
310 000000         PURPOSES:
311 000000         --------
312 000000         ... ROTATE ABOUT X BY ALPHA (TWIST ANGLE)
313 000000         PARAMETERS IN APPEARING ORDER:
314 000000         -----------------------------
315 000000              I     : CASCADE NUMBER                                      INPUT
316 000000              K     : MODE       'F': FORWARD      'B': BACKWARD          INPUT
317 000000              V     : VECTOR WHOSE COMPONENTS ARE TO BE RE-NAMED      INPUT/OUTPUT
318 000000         GLOBAL PARAMETERS:
319 000000         -----------------
320 000000         LINK[I].xx: ROBOT'S LINK PARAMETERS                            INPUT
321 000000         GLOBAL  VARIABLES:
322 000000         -----------------
323 000000                      (NONE)
324 000000         LOCAL   VARIABLES:
325 000000         -----------------
326 000000              II    : TEMPORARY STORAGE VARIABLE (SEE BELOW)
327 000000              UT    : TEMPORARY STORAGE VARIABLE (SEE BELOW)
328 000000         PROCEDURES AND/OR FUNCTIONS CALLED:
329 000000         ----------------------------------
330 000000         'ROTATE'    : PROCEDURE CONTAINED IN THE CURRENT ELEMENT
331 000000         'TRUNC '    : STANDARD PASCAL LIBRARY FUNCTION
332 000000          COMMENTS:
333 000000          --------
334 000000          ... TYPE 'VECTOR' MUST BE DEFINED IN THE APPROPRIATE LOADING ENVIRONMENT
335 000000          ... TAKES ONLY THE TWIST ANGLE ALPHA INTO ACCOUNT
336 000000          ... THETA OFFSET MUST BE CONSIDERED IN 'ROTATE' WHEN ROTATE ABOUT Z
337 000000       *)
338 000010                                                                 (* LOCAL VARIABLES *)
339 000010       VAR II     : INTEGER ;
340 000011           UT     : VECTOR  ;
341 000015
342 000015 [00] BEGIN (*REVECT*)
343 000403 [01]
344 000410         II := TRUNC (LINK[I].ALFA) ;
345 000423
346 000423         IF K='F' THEN ROTATE ('X', II,V,UT)
347 000437                  ELSE ROTATE ('X',-II,V,UT) ;
348 000451         V.X:=UT.X ; V.Y:=UT.Y ; V.Z:=UT.Z ;
349 000462
350 000015 [01] END   (*REVECT*) ;
```

```
000016 [00]
000016      (*************************************************************************)
000016      PROCEDURE
000016      REMTRX (I:INTEGER ; K:CHAR ; VAR TM:MATRIX) ;                    (* 07/05/84 *)
000000
000000      (*
000000        PURPOSES:
000000        --------
000000        ... RE-NAME T-MATRICES
000000        PARAMETERS IN APPEARING ORDER:
000000        ------------------------------
000000             I     : CASCADE NUMBER                                          INPUT
000000             K     : MODE       'F': FORWARD      'B': BACKWARD            INPUT
000000            TM     : MATRIX WHICH IS TO BE MANIPULATED                INPUT/OUTPUT
000000                     IF K='F' --) TM IS  INPUT & WILL NOT BE CHANGED
000000                     IF K='B' --) TM IS OUTPUT & WILL     BE CHANGED
000000        GLOBAL PARAMETERS:
000000        ------------------
000000                     (NONE)
000000        GLOBAL  VARIABLES:
000000        ------------------
000000            TF     : MATRIX AS  INPUT OF  FORWARD CASCADE (    RE-NAMED)    OUTPUT
000000            TB     : MATRIX AS OUTPUT OF BACKWARD CASCADE (NOT RE-NAMED)    OUTPUT
000000        LOCAL   VARIABLES:
000000        ------------------
000000                     (NONE)
000000        PROCEDURES AND/OR FUNCTIONS CALLED:
000000        -----------------------------------
000000         'EQMTRX' : EXTERNAL PROCEDURE CONTAINED IN THE ELEMENT 'RM50100-3A'
000000         'REVECT' : VECTOR-RENAMING PROCEDURE CONTAINED IN THE CURRENT ELEMENT
000000        COMMENTS:
000000        --------
000000        ... TYPE 'MATRIX' MUST BE DEFINED IN THE APPROPRIATE LOADING ENVIRONMENT
000000      *)
000010
000010 [00] BEGIN (*REMTRX*)
000464 [01]
000471 [01]   CASE K OF 'F' : BEGIN
000473 [03]                    EQMTRX (TM,TF)    ;
000501                         REVECT(I,K,TF.N)  ;
000513                         REVECT(I,K,TF.O)  ;
000525                         REVECT(I,K,TF.A)  ;
000537                         REVECT(I,K,TF.P)
000551 [03]                  END (* FORWARD RENAMING*) ;
000552 [02]            'B' : BEGIN
000552 [03]                    EQMTRX (TB,TM)    ;
000560                         REVECT(I,K,TM.N)  ;
000573                         REVECT(I,K,TM.O)  ;
000606                         REVECT(I,K,TM.A)  ;
000621                         REVECT(I,K,TM.P)
000634 [03]                  END (*BACKWARD RENAMING*) ;
000647 [02]   END (*CASE*) ;
000647 [01]
000010 [01] END   (*REMTRX*) ;
```

```
405  000016 [00]
406  000016       (*****************************************************************************)
407  000016       PROCEDURE
408  000016       ATRANS (I:INTEGER) ;                                                (* 07/06/84 *)
409  000000
410  000000       (*
411  000000         PURPOSES:
412  000000         --------
413  000000         ... DETERMINING ACTUAL ELEMENTS OF THE I-TH A-MATRIX & ITS INVERSE
414  000000         PARAMETERS IN APPEARING ORDER:
415  000000         ------------------------------
416  000000              I     : CASCADE NUMBER                                        INPUT
417  000000         GLOBAL PARAMETERS:
418  000000         -----------------
419  000000         LINK[i].xx: ROBOT LINK PARAMETERS                                 INPUT
420  000000         GLOBAL  VARIABLES:
421  000000         ------------------
422  000000              U     : UNIT MATRIX, HERE AS HELP VARIABLE                    INPUT
423  000000             AH     : HOMOGENEOUS TRANSFORMATION MATRIX OF THE i-th CASCADE OUTPUT
424  000000             HA     : ITS INVERSE                                           OUTPUT
425  000000         LOCAL   VARIABLES:
426  000000         -----------------
427  000000                    (NONE)
428  000000         PROCEDURES AND/OR FUNCTIONS CALLED:
429  000000         ----------------------------------
430  000000          'REVECT' : VECTOR-RENAMING     PROCEDURE CONTAINED IN THE CURRENT ELEMENT
431  000000          'REMTRX' : MATRIX-RENAMING     PROCEDURE CONTAINED IN THE CURRENT ELEMENT
432  000000          'ROTATE' : ORIENTATOR          PROCEDURE CONTAINED IN THE CURRENT ELEMENT
433  000000         COMMENTS:
434  000000         --------
435  000000         ... NO ARITHMETIC CALCULATIONS, BUT PURE PHASE SHIFTING
436  000000       *)
437  000006
438  000006 [00] BEGIN (*ATRANS*)
439  000651 [01]
440  000656         U.P.X:=LINK[I].AX ; U.P.Y:=0.0 ; U.P.Z:=LINK[I].DZ ;
441  000701         REVECT (I,'B',U.P) ;
442  000710         REMTRX (I,'F',U  ) ;
443  000717         ROTATE ('Z',LINK[I].ITHETA+LINK[I].IOFST,TF.N,AH[I].N) ;
444  000756         ROTATE ('Z',LINK[I].ITHETA+LINK[I].IOFST,TF.O,AH[I].O) ;
445  001015         ROTATE ('Z',LINK[I].ITHETA+LINK[I].IOFST,TF.A,AH[I].A) ;
446  001054         ROTATE ('Z',LINK[I].ITHETA+LINK[I].IOFST,TF.P,AH[I].P) ;
447  001113
448  001113         U.P.X:=-AH[I].P.X ; U.P.Y:=-AH[I].P.Y ; U.P.Z:=-AH[I].P.Z ;
449  001151
450  001151         ROTATE ('Z',-LINK[I].ITHETA-LINK[I].IOFST, U.P,  TB.P) ;
451  001202         ROTATE ('Z',-LINK[I].ITHETA-LINK[I].IOFST, U.A,  TB.A) ;
452  001233         ROTATE ('Z',-LINK[I].ITHETA-LINK[I].IOFST, U.O,  TB.O) ;
453  001264         ROTATE ('Z',-LINK[I].ITHETA-LINK[I].IOFST, U.N,  TB.N) ;
454  001315         REMTRX (I,'B',HA[I]) ;
455  001333
456  001333         U.P.X:=0.0 ; U.P.Y:=0.0 ; U.P.Z:=0.0
457  001335
458  000006 [01] END   (*ATRANS*) ;
```

```
459  000016 [00]
460  000016       (*************************************************************************)
461  000016       PROCEDURE
462  000016       ARJDOO (I:INTEGER ; K:CHAR ; VAR TI,OT:MATRIX) ;              (* 07/06/84 *)
463  000000
464  000000       (*
465  000000         PURPOSES:
466  000000         --------
467  000000         ... THE I-TH ROBOT JOINT DESCRIPTOR IN THE I-TH CASCADE
468  000000                ^^^^                                 ^^^^
469  000000         PARAMETERS IN APPEARING ORDER:
470  000000         ------------------------------
471  000000              I     : CASCADE NUMBER                                  INPUT
472  000000              K     : COMPUTING MODE  ('F' FORWARD, 'B' BACKWARD)     INPUT
473  000000             TI     :  INPUT MATRIX                                   INPUT
474  000000             OT     : OUTPUT MATRIX                                  OUTPUT
475  000000         GLOBAL PARAMETERS:
476  000000         ------------------
477  000000         LINK[i].ITHETA    : DELAY ANGLE  OF THE i-th CASCADE          INPUT
478  000000         GLOBAL  VARIABLES:
479  000000         ------------------
480  000000            TF             :  INPUT  FORWARD MATRIX (    re-named)    INPUT
481  000000            TB             : OUTPUT BACKWARD MATRIX (not re-named)    OUTPUT
482  000000            AH[i]          : THE A-MATRIX OF THE i-th CASCADE          INPUT
483  000000            G              : HELP VARIABLE AS TEMPORARY STORAGE        INPUT
484  000000         LOCAL   VARIABLES:
485  000000         ------------------
486  000000                       (NONE)
487  000000         PROCEDURES AND/OR FUNCTIONS CALLED:
488  000000         -----------------------------------
489  000000          'REMTRX' : MATRIX-RENAMING     PROCEDURE CONTAINED IN THE CURRENT ELEMENT
490  000000          'ROTATE' : ORIENTATOR          PROCEDURE CONTAINED IN THE CURRENT ELEMENT
491  000000         COMMENTS:
492  000000         --------
493  000000         ... NO ARITHMETIC CALCULATIONS, BUT PURE PHASE SHIFTING
494  000000       *)
495  000011
496  000011 [00] BEGIN (*ARJDOO*)
497  000011 [01]
498  000011          (* DETERMINING ACTUAL ELEMENTS OF THE T-MATRIX                 *)
499  001340
500  001345 [01]    CASE K OF 'F' : BEGIN
501  001347 [03]
502  001347                         REMTRX (I,'F',TI) ; (* SAVE TI & RENAME TF     *)
503  001356
504  001356                         (*   F O R W A R D   *)
505  001356                         ROTATE ('Z', LINK[I].ITHETA+LINK[I].IOFST,TF.N,OT.N) ;
506  001407                         ROTATE ('Z', LINK[I].ITHETA+LINK[I].IOFST,TF.O,OT.O) ;
507  001440                         ROTATE ('Z', LINK[I].ITHETA+LINK[I].IOFST,TF.A,OT.A) ;
508  001471                         ROTATE ('Z', LINK[I].ITHETA+LINK[I].IOFST,TF.P,OT.P) ;
509  001522                         OT.P.X:=OT.P.X + AH[I].P.X              ;
510  001536                         OT.P.Y:=OT.P.Y + AH[I].P.Y              ;
511  001552                         OT.P.Z:=OT.P.Z + AH[I].P.Z              ;
512  001566
513  001566 [03]                    END ;
514  001567 [02]
515  001567 [02]                'B' : BEGIN
516  001567 [03]
517  001567                         (* B A C K W A R D   *)
518  001567                            G.Z:=TI.P.Z - AH[I].P.Z              ;
519  001602                            G.Y:=TI.P.Y - AH[I].P.Y              ;
520  001615                            G.X:=TI.P.X - AH[I].P.X              ;
521  001630                         ROTATE ('Z',-LINK[I].ITHETA-LINK[I].IOFST,   G,TB.P) ;
522  001661                         ROTATE ('Z',-LINK[I].ITHETA-LINK[I].IOFST,TI.A,TB.A) ;
523  001713                         ROTATE ('Z',-LINK[I].ITHETA-LINK[I].IOFST,TI.O,TB.O) ;
524  001745                         ROTATE ('Z',-LINK[I].ITHETA-LINK[I].IOFST,TI.N,TB.N) ;
525  001777
526  001777                         REMTRX (I,'B',OT) ; (* SAVE TB & RENAME OT     *)
527  002006
528  002006 [03]                    END ;
529  002007 [02]
530  002021 [02]    END (*CASE*)    ;
531  002021 [01]
532  000011 [01] END    (*ARJDOO*) ;
533  000016 [00]
```

```
534  000016      (*************************************************************************)
535  000016      PROCEDURE
536  000016      STATUS (I,JB,JF,MN:INTEGER) ;                                  (* 07/08/84 *)
537  000000
538  000000      (*
539  000000        PURPOSES:
540  000000        --------
541  000000        ... ACTUAL STATUS OF THE GENERAL PURPOSE ROBOT CONTROLLER
542  000000        PARAMETERS IN APPEARING ORDER:
543  000000        ------------------------------
544  000000            I      : CASCADE WHICH IS CHANGING                          INPUT
545  000000                     (IF I=0, NO COMPUTATION OF THE A-MATRIX & ITS INVERSE)
546  000000            JB     : BACKWARD STARTING CASCADE                          INPUT
547  000000            JF     :  FORWARD STARTING CASCADE                          INPUT
548  000000            MN     : MUST BE EITHER 'M' OR 'N'                          INPUT
549  000000        GLOBAL PARAMETERS:
550  000000        -----------------
551  000000            M      : NUMBER OF D.O.F. WITHOUT THE END-EFFECTOR           INPUT
552  000000            N      : NUMBER OF D.O.F. WITH    THE END-EFFECTOR           INPUT
553  000000                     (D.O.F.: DEGREES OF FREEDOM)
554  000000        GLOBAL  VARIABLES:
555  000000        -----------------
556  000000           TMB     : T-MATRICES OF BACKWARD COMPUTING PATH          INPUT/OUTPUT
557  000000           TMF     : T-MATRICES OF  FORWARD COMPUTING PATH          INPUT/OUTPUT
558  000000                     ('M' INDICATES ROBOT WITHOUT THE END-EFFECTOR)
559  000000           TNB     : T-MATRICES OF BACKWARD COMPUTING PATH          INPUT/OUTPUT
560  000000           TNF     : T-MATRICES OF  FORWARD COMPUTING PATH          INPUT/OUTPUT
561  000000                     ('N' INDICATES ROBOT WITH    THE END-EFFECTOR)
562  000000        LOCAL   VARIABLES:
563  000000        -----------------
564  000000            K      : LOCAL COUNT VARIABLE
565  000000        PROCEDURES AND/OR FUNCTIONS CALLED:
566  000000        ----------------------------------
567  000000         'ATRANS' : COMPUTATION OF THE i-th A-MATRIX & ITS INVERSE
568  000000         'ARJDOO' : COMPUTATION OF THE T-MATRICES
569  000000                    (PROCEDURES CONTAINED IN THE CURRENT ELEMENT)
570  000000        COMMENTS:
571  000000        --------
572  000000        ... THE A-MATRIX (AH) AND ITS INVERSE (HA) ARE COMPUTED IN 'ATRANS'
573  000000      *)
574  000011
575  000011      VAR   K    : INTEGER ;                                  (* LOCAL VARIABLES *)
576  000012
577  000012 [00] BEGIN (*STATUS*)
578  002023 [01]
579  002030        IF I<>0 THEN ATRANS(I) ;
580  002037
581  002037        IF MN=M THEN
582  002042
583  002042 [01]             BEGIN (*     DECOUPLING *)
584  002042 [02]                  FOR K:=JB      TO MN DO
585  002051                            BEGIN   ARJDOO (K,'B',TMB[K-1],TMB[ K ])   END ;
586  002106                       FOR K:=JF DOWNTO  1 DO
587  002115                            BEGIN   ARJDOO (K,'F',TMF[ K ],TMF[K-1])   END ;
588  002152 [02]             END   (*     DECOUPLING *) ;
589  002152 [01]
590  002152        IF MN=N THEN
591  002155
592  002155 [01]             BEGIN (* NON-DECOUPLING *)
593  002155 [02]                  FOR K:=JB      TO MN DO
594  002164                            BEGIN   ARJDOO (K,'B',TNB[K-1],TNB[ K ])   END ;
595  002221                       FOR K:=JF DOWNTO  1 DO
596  002230                            BEGIN   ARJDOO (K,'F',TNF[ K ],TNF[K-1])   END ;
597  002265 [02]             END   (* NON-DECOUPLING *) ;
598  002265 [01]
599  000012 [01] END   (*STATUS*) ;
600  000016 [00]
```

```
601  000016      (*************************************************************************)
602  000016      PROCEDURE
603  000016      SOLV00 (I,J,MN:INTEGER ; C:CHAR) ;                                (* 09/26/84 *)
604  000000
605  000000      (*
606  000000        PURPOSES:
607  000000        --------
608  000000        ... SOLVING STRATEGY
609  000000        PARAMETERS IN APPEARING ORDER:
610  000000        ------------------------------
611  000000            I      : CASCADE WHICH IS CHANGING                          INPUT
612  000000            J      : ERROR REFERENCE NUMBER                             INPUT
613  000000            MN     : MUST BE EITHER 'M' OR 'N'                          INPUT
614  000000            C      : CONTROL PARAMETER                                  INPUT
615  000000                     C='F' Fast computation
616  000000                     C='A' Consideration of  All  possible variable's values
617  000000        GLOBAL PARAMETERS:
618  000000        -----------------
619  000000            M      : NUMBER OF D.O.F. WITHOUT THE END-EFFECTOR          INPUT
620  000000            N      : NUMBER OF D.O.F. WITH    THE END-EFFECTOR          INPUT
621  000000                     (D.O.F.: DEGREES OF FREEDOM)
622  000000            IRE    : VARIABLE'S RESOLUTION                              INPUT
623  000000        GLOBAL  VARIABLES:
624  000000        ------------------
625  000000           TMB     : T-MATRICES OF BACKWARD COMPUTING PATH          INPUT/OUTPUT
626  000000           TMF     : T-MATRICES OF  FORWARD COMPUTING PATH          INPUT/OUTPUT
627  000000                     ('M' INDICATES ROBOT WITHOUT THE END-EFFECTOR)
628  000000           TNB     : T-MATRICES OF BACKWARD COMPUTING PATH          INPUT/OUTPUT
629  000000           TNF     : T-MATRICES OF  FORWARD COMPUTING PATH          INPUT/OUTPUT
630  000000                     ('N' INDICATES ROBOT WITH    THE END-EFFECTOR)
631  000000           ERB     : ERROR REFERENCE BACKWARD                           OUTPUT
632  000000           ERF     : ERROR REFERENCE  FORWARD                           OUTPUT
633  000000           FEHLER  : SYSTEM ERROR ARRAY                                 OUTPUT
634  000000        LOCAL   VARIABLES:
635  000000        ------------------
636  000000                     (SEE BELOW)
637  000000        PROCEDURES AND/OR FUNCTIONS CALLED:
638  000000        -----------------------------------
639  000000          'STATUS' : CURRENT STATE COMPUTATION OF THE GPRC
640  000000                     (PROCEDURE  CONTAINED IN THE CURRENT ELEMENT)
641  000000
642  000000                   * Local Functions called *
643  000000          'SQRMAG' : Magnitude square calculation
644  000000          'POSERR' : Position    error
645  000000          'ORIERR' : Orientation error
646  000000          'ERROR0' : Routine that decides which error is to compute
647  000000
648  000000          ' ABS  ' : Standard Pascal Library Function
649  000000        COMMENTS:
650  000000        ---------
651  000000        ... ERROR REFERENCE SOLVING STRATEGY
652  000000      *)
653  000011
654  000011      VAR IANGLE,ISTEP        : INTEGER ;                    (* LOCAL VARIABLES *)
655  000013          ECRENT , ENEXT      : REAL    ;
656  000015
```

```
000015       (*-------------------------------------------------------------------------*)
000015       FUNCTION
000015       SQRMAG (V:VECTOR) : REAL ;                                    (* 09/26/84 *)
000011
002267 [00] BEGIN (*SQRMAG*)
002277 [01]         SQRMAG := V.X*V.X + V.Y*V.Y + V.Z*V.Z
000011 [01] END   (*SQRMAG*) ;
000015 [00]
000015       (*-------------------------------------------------------------------------*)
000015       FUNCTION
000015       POSERR (IANGLE,I,J:INTEGER) : REAL ;                          (* 09/26/84 *)
000010
000010       VAR   EEE   :   REAL ;
000011
000011 [00] BEGIN (*POSERR              position error*)
002313 [01]
002315           LINK[I].ITHETA:=IANGLE           ;
002326           STATUS (I,I,I,MN)                ;
002336
002336           IF MN=N THEN
002341 [01]          BEGIN
002341 [02]              ERB[J] := SQRMAG (TNB[J].P) ;
002366                   ERF[J] := SQRMAG (TNF[J].P) ;
002413 [02]          END ;
002413 [01]
002413           IF MN=M THEN
002416 [01]          BEGIN
002416 [02]              ERB[J] := SQRMAG (TMB[J].P) ;
002443                   ERF[J] := SQRMAG (TMF[J].P) ;
002470 [02]          END ;
002470 [01]
002470           EEE    := ERF[J] - ERB[J] ;
002505           (* IF  EEE < 0.0  THEN  EEE := -EEE ; *)
002505           POSERR:= EEE ;
002510
000011 [01] END   (*POSERR*) ;
000015 [00]
000015       (*-------------------------------------------------------------------------*)
000015       FUNCTION
000015       ORIERR (IANGLE,I,J:INTEGER) : REAL ;                          (* 09/26/84 *)
000010
000010       VAR   V     :   VECTOR ;
000014
000014 [00] BEGIN (*ORIERR           orientation error*)
002512 [01]
002514           LINK[I]..ITHETA:=IANGLE           ;
002525           STATUS (I,I,I,MN)                 ;
002535
002535           IF MN=N THEN
002540 [01]          BEGIN
002540 [02]              V.X:=1.0-TNB[J].N.X ;
002552                   V.Y:=1.0-TNB[J].O.Y ;
002564                   V.Z:=1.0-TNB[J].A.Z ;
002576 [02]          END ;
002576 [01]
002576           IF MN=M THEN
002601 [01]          BEGIN
002601 [02]              V.X:=1.0-TMB[J].N.X ;
002613                   V.Y:=1.0-TMB[J].O.Y ;
002625                   V.Z:=1.0-TMB[J].A.Z ;
002637 [02]          END ;
002637 [01]
002637           ORIERR:=SQRMAG(V) ;
002651
000014 [01] END   (*ORIERR*) ;
000015 [00]
000015       (*-------------------------------------------------------------------------*)
000015       FUNCTION
000015       ERRORO (IANGLE,I,J:INTEGER) : REAL ;                          (* 09/27/84 *)
000010
000010       VAR FFF : REAL ;
000011
002653 [00] BEGIN (*ERRORO*)
002655 [01]          IF (J=MN) AND (I)3) THEN FFF:= ORIERR (IANGLE,I,J)
002677                                   ELSE FFF:= POSERR (IANGLE,I,J) ;
002712               ERRORO:=FFF ;
000011 [01] END   (*ERRORO*) ;
000015 [00]
000015       (*-------------------------------------------------------------------------*)
```

```
736  000015
737  000015 [00] BEGIN (*SOLV00*)
738  002717 [01]
739  002724         FOR ISTEP:=1 TO  N  DO
740  002727 [01]      BEGIN
741  002727 [02]            FEHLER[I].ERPAR[ISTEP] := LINK[ISTEP].ITHETA
742  002747 [02]      END ;
743  002747 [01]
744  002747
745  002747
746  002747
747  002747
748  002747              ISTEP := 1                                    ;
749  002751              IANGLE:= LINK[I].ITHETA                       ;
750  002762              ECRENT:= ERRORO (IANGLE,I,J)                  ;
751  002775              IANGLE:=IANGLE+ISTEP                          ;
752  003000              ENEXT := ERRORO (IANGLE,I,J)                  ;
753  003013              IF (ABS(ENEXT) ) ABS(ECRENT)) THEN
754  003020 [01]             BEGIN
755  003020 [02]               ISTEP := -ISTEP                         ;
756  003023                    ENEXT :=  ECRENT                        ;
757  003025                    IANGLE:=  IANGLE+ISTEP                  ;
758  003030 [02]             END                                       ;
759  003030 [01]
760  003030 [01]         REPEAT
761  003030 [02]           ECRENT:=ENEXT                               ;
762  003032                IANGLE:=IANGLE+ISTEP                        ;
763  003035                ENEXT :=ERRORO (IANGLE,I,J)                 ;
764  003050 [02]         UNTIL (ABS(ECRENT) (= ABS(ENEXT))             ;
765  003055 [01]
766  003055              LINK[I].ITHETA:=IANGLE-ISTEP                  ;
767  003067              STATUS (I,I,I,MN)                             ;
768  003077
769  003077
770  003077
771  003077         IF C='A' THEN
772  003102
773  003102 [01]      BEGIN
774  003102 [02]
775  003102              IANGLE:=LINK[I].ITHETA ;
776  003113
777  003113              FOR ISTEP:=1 TO IRE DO
778  003116 [02]           BEGIN
779  003116 [03]             FEHLER[I].ERVAR[ISTEP] := ERRORO (ISTEP,I,J) ;
780  003147 [03]           END ;
781  003147 [02]
782  003147              LINK[I].ITHETA:=IANGLE ;
783  003160              STATUS (I,I,I,MN)      ;
784  003170
785  003170 [02]       END ;
786  003170 [01]
787  003170
788  000015 [01] END   (*SOLV00*) ;
789  000016 [00]
790  000016       (*****************************************************************************)
791  000016       (*                          TERMINATION PERIOD  end of element RM50100-2A    *)
792  000016       (*****************************************************************************)
Compilation completed - no errors found.
```

)@EOF

```
000000 (*************************************************************************)
000000
000000 (*
000000    ------------------------
000000    ELEMENT     RM50100-1A              ; EXTERNAL ELEMENTS: SEE BELOW
000000    ------------------------
000000
000000    SOLVING ROBOT KINEMATICS INVERSE PROBLEM
000000    WITHOUT EXPLICIT ARITHMETIC EXPRESSIONS.
000000    (This File Element: Initializing ... Printing ...)
000000
000000    SECOND  UNIVAC VERSION                10/09/84          BY T.Vd./
000000    LAST    IMPLEMENTATION                10/22/84
000000                                         (mm/dd/yy)
000000
000000
000000                                              UNIVERSITY OF WOLLONGONG
000000                                          DEPARTMENT OF ELECTRICAL ENGINEERING
000000
000000
000000
000000             USER PROCEDURES AND FUNCTIONS CALLED:
000000             ------------------------------------
000000             (Listed    in    appearing    order)
000000
000000             RELATIVE          : EXTERNAL/T/C :                :
000000             LEVEL  #          :  T: TEXT     : PROC./FUNC.   :    LANGUAGE
000000    1          2       3,4,5   :  C: CODE     :               :
000000 .........:.........:.........:..............:...............:..............
000000 PRINTT                        :     NO       :  PROCEDURE    :     PASCAL
000000 .........:.........:.........:..............:...............:..............
000000 PPRINT                        :     NO       :  PROCEDURE    :     PASCAL
000000 .........:.........:.........:..............:...............:..............
000000           printt              :              :               :
000000 .........:.........:.........:..............:...............:..............
000000 INITIA                        :     NO       :  PROCEDURE    :     PASCAL
000000 .........:.........:.........:..............:...............:..............
000000          *init00              :     NO       :  PROC.        :     PASCAL
000000 .........:.........:.........:..............:...............:..............
000000           SINUS               :3A)  YES/C    :  FUNCTION     :     PASCAL
000000 .........:.........:.........:..............:...............:..............
000000           ATRANS              :2A)  YES/C    :  PROC.        :     PASCAL
000000 .........:.........:.........:..............:...............:..............
000000                     revect    :              :               :
000000 .........:.........:.........:..............:...............:..............
000000                     remtrx    :              :               :
000000 .........:.........:.........:..............:...............:..............
000000                     rotate    :              :               :
000000 .........:.........:.........:..............:...............:..............
000000           STATUS              :2A)  YES/C    :  PROC.        :     PASCAL
000000 .........:.........:.........:..............:...............:..............
000000                     atrans    :              :               :
000000 .........:.........:.........:..............:...............:..............
000000                     arjd00    :              :               :
000000 .........:.........:.........:..............:...............:..............
000000 * indicates local Proc./Func.
000000
000000
000000 TOTAL NUMBER OF PROCEDURES          :      05
000000                 FUNCTIONS           :      01
000000                 ----------                 ----
000000                 T O T A L           :      06          C A L L E D
000000
000000 EXTERNAL ELEMENT(S)                 :      3A) RM50100-3A
000000                                            2A) RM50100-2A
000000
000000 LOADING  ENVIRONMENT                :          RM50100-CM
000000
000000 NUMBER OF PROCEDURES AND FUNCTIONS
000000 CONTAINED IN THE CURRENT  ELEMENT   :      03
000000
000000 NUMBER OF PROCEDURES AND FUNCTIONS
000000 CALLED  FROM THE EXTERNAL ELEMENT(S):      03
000000
000000 NUMBER OF PROCEDURES AND FUNCTIONS
000000 CALLED  FROM PASCAL LIBRARY         :      00
000000 *)
```

```
78  000000
79  000000      (*************************************************************************)
80  000000      ENVIRONMENT
81  005743      ROUTINES ( PRINTT , PPRINT , INITIA ) ;
82  000016      (*************************************************************************)
83  000016      PROCEDURE
84  000016      PRINTT (I,L:INTEGER ; F,B:MATRIX) ;                          (* 07/09/84 *)
85  000000
86  000000      (*
87  000000        PURPOSES:
88  000000        --------
89  000000        ... PRINT ALL A[ i ] OR T[ i ] ELEMENTS, HERE REPRESENTED AS F & B
90  000000        PARAMETERS IN APPEARING ORDER:
91  000000        ------------------------------
92  000000             I     : CASCADE NUMBER                                  INPUT
93  000000             L     : CONTROL PARAMETER                               INPUT
94  000000                     if L=M or L=N   print   [ i ]T[ 1 ]    as T-Matrix
95  000000                     if        L=0   print   [i-1]A[ i ]    as A-Matrix
96  000000           F[i]    :  FORWARD PATH OUTPUT T-MATRIX OR A-MATRIX        INPUT
97  000000           B[i]    : BACKWARD PATH  INPUT T-MATRIX OR A-INVERSE       INPUT
98  000000        GLOBAL PARAMETERS:
99  000000        ------------------
100 000000        LINK[i].ITHETA : DELAY ANGLE OF THE i-th CASCADE               INPUT
101 000000        GLOBAL  VARIABLES:
102 000000        ------------------
103 000000                    (NONE)
104 000000        LOCAL   VARIABLES:
105 000000        ------------------
106 000000                    (NONE)
107 000000        PROCEDURES AND/OR FUNCTIONS CALLED:
108 000000        -----------------------------------
109 000000                    (NONE)
110 000000        COMMENTS:
111 000000        --------
112 000000        ... TYPE 'MATRIX' MUST BE DEFINED IN THE APPROPRIATE LOADING ENVIRONMENT
113 000000      *)
114 000047
115 000047 [00] BEGIN (*PRINTT*)
116 000000 [01]
117 000025        WRITE('.') ;
118 000030
119 000030        WRITELN ;
120 000032        IF L<>0 THEN WRITELN('[',  I:1,']T[',L:1,']') ;
121 000061 [01]   IF L=0  THEN BEGIN
122 000063 [02]                   WRITE('[',I-1:1,']A[',I:1,']') ;
123 000110                        WRITELN('    Theta[',I:1,'] = ',LINK[I].ITHETA:5) ;
124 000143                        WRITELN('           ----------------') ;
125 000152 [02]                 END ;
126 000152 [01]   WRITELN ( F.N.X:9:5 , F.O.X:9:5 , F.A.X:9:5 , F.P.X:8:2 , ' ' ,
127 000201                  B.N.X:9:5 , B.O.X:9:5 , B.A.X:9:5 , B.P.X:8:2 )          ;
128 000227        WRITELN ( F.N.Y:9:5 , F.O.Y:9:5 , F.A.Y:9:5 , F.P.Y:8:2 , ' ' ,
129 000256                  B.N.Y:9:5 , B.O.Y:9:5 , B.A.Y:9:5 , B.P.Y:8:2 )          ;
130 000304        WRITELN ( F.N.Z:9:5 , F.O.Z:9:5 , F.A.Z:9:5 , F.P.Z:8:2 , ' ' ,
131 000333                  B.N.Z:9:5 , B.O.Z:9:5 , B.A.Z:9:5 , B.P.Z:8:2 )          ;
132 000361        WRITELN ( F.N.W:9:5 , F.O.W:9:5 , F.A.W:9:5 , F.P.W:8:2 , ' ' ,
133 000410                  B.N.W:9:5 , B.O.W:9:5 , B.A.W:9:5 , B.P.W:8:2 )          ;
134 000436
135 000047 [01] END   (*PRINTT*)  ;
136 000016 [00]
137 000016      (*************************************************************************)
138 000016      PROCEDURE
139 000016      PPRINT (J:INTEGER) ;                                         (* 07/09/84 *)
140 000000
141 000000      (*
142 000000        PURPOSES:
143 000000        --------
144 000000        ... PRINT ALL A[ i ] AND T[ i ] MATRICES        --)   G P R C   STATUS
145 000000        PARAMETERS IN APPEARING ORDER:
146 000000        ------------------------------
147 000000             J     : CONTROL PARAMETER                               INPUT
148 000000                     if  J=M   print   Status   without the End-Effector
149 000000                     if  J=N   print   Status   with    the End-Effector
150 000000        GLOBAL PARAMETERS:
151 000000        ------------------
152 000000        LINK[i].IOFST  : DELAY ANGLE OFFSET                            INPUT
153 000000        LINK[i].ITHETA : DELAY ANGLE                                   INPUT
154 000000        LINK[i].ALFA   : TWIST ANGLE                                   INPUT
155 000000        LINK[i].AX     : "X" LINK LENGTH                               INPUT
156 000000        LINK[i].DZ     : "Z" LINK LENGTH                               INPUT
157 000000                         (ROBOT LINK PARAMETERS OF THE i-th CASCADE)
158 000000             M         : NUMBER OF D.O.F. WITHOUT THE END-EFFECTOR     INPUT
159 000000             N         : NUMBER OF D.O.F. WITH    THE END-EFFECTOR     INPUT
160 000000                         (D.O.F.: degrees of freedom)
```

```
161  000000          GLOBAL  VARIABLES:
162  000000          ----------------
163  000000             EULER0 ,
164  000000             EULERY ,
165  000000             EULER1       : EULER-ANGLES IN SEQUENCE                                  INPUTS
166  000000             TNB[0]       : DESIRED END-EFFECTOR'S ORIENTATION & POSITION      INPUT
167  000000                            IN FORM OF GLOBAL MATRIX
168  000000             AH, HA       : THE A-MATRICES & THEIR INVERSES     Printed Output/INPUT
169  000000            TMF,TMB       : THE T-MATRICES ...      DECOUPLED   Printed Output/INPUT
170  000000            TNF,TNB       : THE T-MATRICES ... NOT DECOUPLED    Printed Output/INPUT
171  000000          LOCAL   VARIABLES:
172  000000          -----------------
173  000000               I     : LOCAL COUNT VARIABLE
174  000000          PROCEDURES AND/OR FUNCTIONS CALLED:
175  000000          ----------------------------------
176  000000           'PRINTT' : PRINTING PROCEDURE CONTAINED IN THE CURRENT ELEMENT
177  000000          COMMENTS:
178  000000          --------
179  000000          ... TYPE 'MATRIX' MUST BE DEFINED IN THE APPROPRIATE LOADING ENVIRONMENT
180  000000        *)
181  000006
182  000006        VAR  I   : INTEGER ;                                        (* LOCAL VARIABLES *)
183  000007
184  000007 [00] BEGIN (*PPRINT*)
185  000440 [01]
186  000445          WRITELN ;
187  000447          WRITELN ('============================') ;
188  000456          WRITELN ('* G P R C     S T A T U S *') ;
189  000465          WRITELN ('============================') ;
190  000474          WRITELN ('Matrix System of  Robot 501') ;
191  000503          WRITELN ;
192  000505
193  000505          WRITELN
194  000505          ('  Theta Offset    Theta         Alfa        Ax         Dz          Cascade #') ;
195  000514          WRITELN
196  000514          ('    [Degrees]    [Degrees]     [Degrees]    [mm]       [mm]')       ;
197  000523          WRITELN ;
198  000525
199  000525          FOR I:=1 TO N DO
200  000530 [01]       BEGIN
201  000530 [02]         WRITELN (LINK[I].IOFST:10  ,LINK[I].ITHETA:10  ,
202  000550                       LINK[I].ALFA :15:2,LINK[I].AX    :10:2, LINK[I].DZ:10:2 , I:12) ;
203  000611              IF I=M THEN WRITELN ;
204  000620 [02]       END ;
205  000620 [01]
206  000620          WRITELN ;
207  000622          WRITELN ('Euler-Angles/[Degrees] : ',EULER0:10,EULERY:10,EULER1:10) ;
208  000645          WRITELN
209  000645          ('Position     /[  mm   ] : ',TNB[0].P.X:10:2,TNB[0].P.Y:10:2,TNB[0].P.Z:10:2);
210  000673
211  000673          WRITELN ; WRITELN ;
212  000677          WRITELN
213  000677          ('  F O R W A R D                                          B A C K W A R D') ;
214  000706
215  000706 [01]     IF J=M THEN BEGIN
216  000711 [02]                   FOR I:=1 TO M DO
217  000714 [02]                     BEGIN
218  000714 [03]                       PRINTT (I-1,M,TMF[I-1],TMB[I-1]) ;
219  000753                            PRINTT ( I ,0, AH[ I ], HA[ I ])
220  001003 [03]                     END ;
221  001003 [02]                       PRINTT ( M ,M,TMF[ M ],TMB[ M ])
222  001015 [02]                   END ;
223  001015 [01]
224  001015 [01]     IF J=N THEN BEGIN
225  001020 [02]                   FOR I:=1 TO N DO
226  001023 [02]                     BEGIN
227  001023 [03]                       PRINTT (I-1,N,TNF[I-1],TNB[I-1]) ;
228  001062                            PRINTT ( I ,0, AH[ I ], HA[ I ])
229  001112 [03]                     END ;
230  001112 [02]                       PRINTT ( N ,N,TNF[ N ],TNB[ N ])
231  001124 [02]                   END ;
232  001124 [01]
233  000007 [01] END   (*PPRINT*)  ;
234  000016 [00]
```

```
000016      (*************************************************************************)
000016      PROCEDURE
000016      INITIA ;                                                        (* 07/09/84 *)
000000
000000      (*
000000        PURPOSES:
000000        --------
000000        ... INITIALIZING GLOBAL PROGRAM PARAMETERS & VARIABLES
000000        PARAMETERS IN APPEARING ORDER:
000000        ------------------------------
000000                      (NONE)
000000        GLOBAL PARAMETERS:
000000        ------------------
000000        LINK[i].IOFST  : DELAY ANGLE OFFSET                              OUTPUT
000000        LINK[i].ITHETA : DELAY ANGLE                                     OUTPUT
000000        LINK[i].ALFA   : WRIST ANGLE                                     OUTPUT
000000        LINK[i].AX     : "X" LINK LENGTH                                 OUTPUT
000000        LINK[i].DZ     : "Z" LINK LENGTH                                 OUTPUT
000000                         (ROBOT LINK PARAMETERS OF THE i-th CASCADE)
000000             M         : NUMBER OF D.O.F. WITHOUT THE END-EFFECTOR         INPUT
000000             N         : NUMBER OF D.O.F. WITH    THE END-EFFECTOR         INPUT
000000                         (D.O.F.: degrees of freedom)
000000             PI        : PI CONSTANT                                      INPUT
000000             IRE       : VARIABLE RESOLUTION                              INPUT
000000        GLOBAL  VARIABLES:
000000        ------------------
000000           EULER0 ,
000000           EULERY ,
000000           EULER1       : EULER-ANGLES IN SEQUENCE                       OUTPUTS
000000           AH, HA       : THE A-MATRICES & THEIR INVERSES                OUTPUT
000000          TMF,TMB       : THE T-MATRICES ...      DECOUPLED              OUTPUT
000000          TNF,TNB       : THE T-MATRICES ... NOT DECOUPLED               OUTPUT
000000          EMF,EMB       : THE ROBOT HAND ...      DECOUPLED              OUTPUT
000000           TF, TB       : FORWARD..BACKWARD TEMPORARY STORAGE MATRICES   OUTPUT
000000             U          : UNITY MATRIX                                   OUTPUT
000000        LOCAL   VARIABLES:
000000        ------------------
000000             I     : LOCAL COUNT VARIABLE
000000             T     : LOCAL REAL RADIAN ANGLE
000000        PROCEDURES AND/OR FUNCTIONS CALLED:
000000        -----------------------------------
000000         'INIT00' :       LOCAL PROCEDURE IN THE FIRST RELATIVE CALLING LEVEL *
000000         'SINUS ' :   EXTERNAL FUNCTION  CONTAINED IN THE ELEMENT      'RM50100-3A'
000000         'ATRANS' :   EXTERNAL PROCEDURE CONTAINED IN THE ELEMENT      'RM50100-2A'
000000         'STATUS' :   EXTERNAL PROCEDURE CONTAINED IN THE ELEMENT      'RM50100-2A'
000000        COMMENTS:
000000        --------
000000                      (NONE)
000000      *)
000000
000005      (*-------------------------------------------------------------------------*)
000005      VAR  I   : INTEGER ;                                    (* LOCAL VARIABLES *)
000006           T,R: REAL    ;
000010      (*-------------------------------------------------------------------------*)
000010      PROCEDURE
000010      INIT00 (VAR F:MATRIX) ;                                 (* LOCAL PROCEDURE *)
000006
000006 [00] BEGIN (*INIT00*)
001126 [01]
001131           F.N.X:=1.0 ; F.O.X:=0.0 ; F.A.X:=0.0 ; F.P.X:=0.0 ;
001142           F.N.Y:=0.0 ; F.O.Y:=1.0 ; F.A.Y:=0.0 ; F.P.Y:=0.0 ;
001153           F.N.Z:=0.0 ; F.O.Z:=0.0 ; F.A.Z:=1.0 ; F.P.Z:=0.0 ;
001164           F.N.W:=0.0 ; F.O.W:=0.0 ; F.A.W:=0.0 ; F.P.W:=1.0 ;
001175
000006 [01] END   (*INIT00*) ;
000010 [00]
000010      (*-------------------------------------------------------------------------*)
```

```
302  000010
303  000010 [00] BEGIN (*INITIA*)
304  000010 [01]
305  001177         (* Assign Robot's Geometry Parameters ... Lengths/mm ... Angles/Degreés *)
306  001204         LINK[1].IOFST:=90 ; LINK[2].IOFST:=0 ; LINK[3].IOFST:=0 ;
307  001210         LINK[4].IOFST:=90 ; LINK[5].IOFST:=0 ;
308  001213
309  001213         LINK[1].ITHETA:=0;LINK[1].ALFA:=90.0;LINK[1].AX:=000.0;LINK[1].DZ:=250.0;
310  001221         LINK[2].ITHETA:=0;LINK[2].ALFA:=00.0;LINK[2].AX:=220.0;LINK[2].DZ:=000.0;
311  001226         LINK[3].ITHETA:=0;LINK[3].ALFA:=00.0;LINK[3].AX:=160.0;LINK[3].DZ:=000.0;
312  001233         LINK[4].ITHETA:=0;LINK[4].ALFA:=90.0;LINK[4].AX:=000.0;LINK[4].DZ:=000.0;
313  001240         LINK[5].ITHETA:=0;LINK[5].ALFA:=00.0;LINK[5].AX:=000.0;LINK[5].DZ:=000.0;
314  001244
315  001244         EULER0:=0 ; EULERY:=0 ; EULERY:=0 ; R:=0.5*IRE ;
316  001251
317  001251         (* Reference Sinewave with Resolution of 'IRE' Samples per Period       *)
318  001251         FOR I:=1 TO IRE DO
319  001254 [01]      BEGIN
320  001254 [02]        T      :=(PI*I)/R ;
321  001261            REF[I]:=SINUS(T)
322  001302 [02]      END ;
323  001302 [01]
324  001302         (* Initializing Variables & Parameters                                  *)
325  001302         INITOO (U) ; INITOO (TF) ; INITOO (TB) ;
326  001321         FOR I:=0 TO N DO      BEGIN  INITOO (TNF[I]) ; INITOO (TNB[I])  END ;
327  001360         FOR I:=0 TO M DO      BEGIN  INITOO (TMF[I]) ; INITOO (TMB[I])  END ;
328  001417         FOR I:=M TO N DO      BEGIN  INITOO (EMF[I]) ; INITOO (EMB[I])  END ;
329  001456         FOR I:=1 TO N DO      BEGIN  INITOO ( AH[I]) ; INITOO ( HA[I])  END ;
330  001515         FOR I:=1 TO N DO      BEGIN  ATRANS (I)                         END ;
331  001532         STATUS (0,1,M,M) ;
332  001542         STATUS (0,1,N,N)
333  001552
334  000010 [01] END   (*INITIA*)  ;
335  000016 [00]
336  000016       (*************************************************************************)
337  000016       .                          (* TERMINATION PERIOD  end of element RM50100-1A   *)
338  000016       (*************************************************************************)
ompilation completed - no errors found.
```

```
)@PAS,S F.RM50100-0A,F.RM50100-0A,,F.RM50100-CM
University of Wisconsin  P A S C A L  (8:A:1)  07/11/85 14:22:04 (15)
)@EOF
```

```
000000 (*************************************************************************************)
000000
000000 (*
000000    ------------------------
000000    ELEMENT     RM50100-0A              ; EXTERNAL ELEMENTS: SEE BELOW
000000    ------------------------
000000
000000    SOLVING ROBOT KINEMATICS INVERSE PROBLEM
000000    WITHOUT EXPLICIT ARITHMETIC EXPRESSIONS.
000000    (This File Element: Main Program.)
000000
000000    SECOND  UNIVAC VERSION              10/08/84            BY T.Vd./
000000    LAST    IMPLEMENTATION              10/31/84
000000                                        (mm/dd/yy)
000000
000000
000000                                              UNIVERSITY OF WOLLONGONG
000000                                          DEPARTMENT OF ELECTRICAL ENGINEERING
000000
000000    RELEVANCE :
000000
000000          THIS IS THE SIMULATION OF THE THREE-DIMENSIONAL ROBOT JOINT
000000          DESCRIPTORS  WHICH USE PHASE-SHIFTED SINEWAVES TO REPRESENT
000000          THE  HOMOGENEOUS  TRANSFORMATION  MATRICES  INVOLVED IN THE
000000          MANIPULATOR'S KINEMATICS EQUATIONS
000000
000000          FOR  ANY  ROBOT THE FOLLOWING RELATIONSHIPS ARE TRUE:
000000
000000          WORLD + ARM + TOOL =            GRIPPER
000000                  ARM + TOOL = - WORLD + GRIPPER
000000                  ARM        = - WORLD + GRIPPER - TOOL
000000
000000          WHERE
000000
000000          WORLD       :  TRANSFORM DESCRIBING ROBOT'S LOCATION  WITH
000000                         RESPECT  TO  SOME  FIXED WORLD'S COORDINATE
000000                         SYSTEM.
000000          ARM         :  ROBOT ARM MATRIX.
000000          TOOL        :  TRANSFORM DESCRIBING GRIPPER'S     LOCATION
000000                         WITH RESPECT TO ROBOT'S END CO-ORD.  SYSTEM
000000          GRIPPER     :  TRANSFORM DESCRIBING GRIPPER' LOCATION WITH
000000                         RESPECT TO WORLD.
000000
000000          THE ABOVE MATRIX EQUATION IS SOLVABLE FOR  THE  ARM MATRIX,
000000          ASSUMING THE RIGHT HAND SIDE MATRIX PRODUCT IS GIVEN.
000000
000000          THIS PROGRAM SIMULATES  THE SOLVING PROCESS USING THE ROBOT
000000          JOINT   DESCRIPTORS,   WHEREBY  A  REFERENCE  SINEWAVE   IS
000000          GENERATED JUST ONCE AND STORED IN A GLOBAL ARRAY FOR  LOOK-
000000          UP PURPOSES.  THE  ANGLES  ARE  LINEARLY  QUANTIZED  TO  AN
000000          INTEGER RESOLUTION OF 360 SAMPLES PER SIGNAL PERIOD.
000000
```

```
000000
000000          USER PROCEDURES AND FUNCTIONS CALLED:
000000          ------------------------------------
000000          (Listed    in    appearing    order)
000000
000000          RELATIVE          : EXTERNAL/T/C :                :
000000          LEVEL  #          :  T: TEXT     : PROC./FUNC.  :    LANGUAGE
000000     1        2       3,4,5 :  C: CODE     :              :
000000 .........:........:........:..............:..............:..............
000000 EQMTRX                     :3A)  YES/C    :  PROCEDURE   :     PASCAL
000000 .........:........:........:..............:..............:..............
000000 SINUS                      :3A)  YES/C    :  FUNCTION    :     PASCAL
000000 .........:........:........:..............:..............:..............
000000 WAVE01                     :2A)  YES/C    :  FUNCTION    :     PASCAL
000000 .........:........:........:..............:..............:..............
000000 ROTATE                     :2A)  YES/C    :  PROCEDURE   :     PASCAL
000000 .........:........:........:..............:..............:..............
000000          wave01            :              :              :
000000 .........:........:........:..............:..............:..............
000000 ROTMAT                     :2A)  YES/C    :  PROC.       :     PASCAL
000000 .........:........:........:..............:..............:..............
000000          rotate            :              :              :
000000 .........:........:........:..............:..............:..............
000000                    wave01  :              :              :
000000 .........:........:........:..............:..............:..............
000000 REVECT                     :2A)  YES/C    :  PROC.       :     PASCAL
000000 .........:........:........:..............:..............:..............
000000          rotate            :              :              :
000000 .........:........:........:..............:..............:..............
000000          TRUNC             :     NO       :  FUNC.       : LIB PASCAL FUNC.
000000 .........:........:........:..............:..............:..............
000000 REMTRX                     :2A)  YES/C    :  PROC.       :     PASCAL
000000 .........:........:........:..............:..............:..............
000000          revect            :              :              :
000000 .........:........:........:..............:..............:..............
000000          eqmtrx            :              :              :
000000 .........:........:........:..............:..............:..............
000000 ATRANS                     :2A)  YES/C    :  PROC.       :     PASCAL
000000 .........:........:........:..............:..............:..............
000000          revect            :              :              :
000000 .........:........:........:..............:..............:..............
000000          remtrx            :              :              :
000000 .........:........:........:..............:..............:..............
000000                    revect  :              :              :
000000 .........:........:........:..............:..............:..............
000000                    eqmtrx  :              :              :
000000 .........:........:........:..............:..............:..............
000000          rotate            :              :              :
000000 .........:........:........:..............:..............:..............
000000                    wave01  :              :              :
000000 .........:........:........:..............:..............:..............
000000 ARJD00                     :2A)  YES/C    :  PROC.       :     PASCAL
000000 .........:........:........:..............:..............:..............
000000          remtrx            :              :              :
000000 .........:........:........:..............:..............:..............
000000                    revect  :              :              :
000000 .........:........:........:..............:..............:..............
000000                    eqmtrx  :              :              :
000000 .........:........:........:..............:..............:..............
000000          rotate            :              :              :
000000 .........:........:........:..............:..............:..............
000000                    wave01  :              :              :
000000 .........:........:........:..............:..............:..............
000000 STATUS                     :2A)  YES/C    :  PROC.       :     PASCAL
000000 .........:........:........:..............:..............:..............
000000          atrans            :              :              :
000000 .........:........:........:..............:..............:..............
000000                    revect  :              :              :
000000 .........:........:........:..............:..............:..............
000000                    remtrx  :              :              :
000000                    revect(4):             :              :
000000                    eqmtrx(4):             :              :
000000 .........:........:........:..............:..............:..............
000000                    rotate  :              :              :
000000                    wave01(4):             :              :
000000 .........:........:........:..............:..............:..............
000000          arjd00            :              :              :
000000 .........:........:........:..............:..............:..............
000000                    remtrx  :              :              :
000000                    revect(4):             :              :
000000                    eqmtrx(4):             :              :
000000 .........:........:........:..............:..............:..............
000000                    rotate  :              :              :
000000                    wave01(4):             :              :
```

```
000000 .........:.........:.........:.............:.............:.............
000000 SOLV00                      :2A)  YES/C     :  PROC.       :     PASCAL
000000 .........:.........:.........:.............:.............:.............
000000          *sqrmag            :2A)  YES/C     :  FUNC.       :     PASCAL
000000 .........:.........:.........:.............:.............:.............
000000          *poserr            :2A)  YES/C     :  FUNC.       :     PASCAL
000000 .........:.........:.........:.............:.............:.............
000000          *orierr            :2A)  YES/C     :  FUNC.       :     PASCAL
000000 .........:.........:.........:.............:.............:.............
000000          *error0            :2A)  YES/C     :  FUNC.       :     PASCAL
000000 .........:.........:.........:.............:.............:.............
000000 PRINTT                      :1A)  YES/C     :  PROCEDURE   :     PASCAL
000000 .........:.........:.........:.............:.............:.............
000000 PPRINT                      :1A)  YES/C     :  PROCEDURE   :     PASCAL
000000 .........:.........:.........:.............:.............:.............
000000           printt            :               :              :
000000 .........:.........:.........:.............:.............:.............
000000 INITIA                      :1A)  YES/C     :  PROCEDURE   :     PASCAL
000000 .........:.........:.........:.............:.............:.............
000000          *init00            :1A)  YES/C     :  PROC.       :     PASCAL
000000 .........:.........:.........:.............:.............:.............
000000           sinus             :               :              :
000000 .........:.........:.........:.............:.............:.............
000000           atrans            :               :              :
000000 .........:.........:.........:.............:.............:.............
000000                     revect   :               :              :
000000 .........:.........:.........:.............:.............:.............
000000                     remtrx   :               :              :
000000 .........:.........:.........:.............:.............:.............
000000                     rotate   :               :              :
000000 .........:.........:.........:.............:.............:.............
000000           status            :               :              :
000000 .........:.........:.........:.............:.............:.............
000000                     atrans   :               :              :
000000 .........:.........:.........:.............:.............:.............
000000                     arjd00   :               :              :
000000 .........:.........:.........:.............:.............:.............
000000 WRPOSI                      :     NO        :  PROCEDURE   :     PASCAL
000000 .........:.........:.........:.............:.............:.............
000000 BOTHOP                      :     NO        :  PROCEDURE   :     PASCAL
000000 .........:.........:.........:.............:.............:.............
000000 ONLYPO                      :     NO        :  PROCEDURE   :     PASCAL
000000 .........:.........:.........:.............:.............:.............
000000 ONLYOR                      :     NO        :  PROCEDURE   :     PASCAL
000000 .........:.........:.........:.............:.............:.............
000000 * indicates local Proc./Func.
000000
000000
000000 TOTAL NUMBER OF PROCEDURES          :       12
000000                 FUNCTIONS           :       03
000000                 ----------                  ----
000000                 T O T A L           :       15          C A L L E D
000000                                                (not including local Proc./Func.)
000000
000000 EXTERNAL ELEMENT(S)                 :       3A) RM50100-3A
000000                                             2A) RM50100-2A
000000                                             1A) RM50100-1A
000000
000000 LOADING  ENVIRONMENT                :           RM50100-CM
000000
000000 NUMBER OF PROCEDURES AND FUNCTIONS
000000 CONTAINED IN THE CURRENT  ELEMENT   :       04
000000
000000 NUMBER OF PROCEDURES AND FUNCTIONS
000000 CALLED  FROM THE EXTERNAL ELEMENT(S):       14
000000
000000 NUMBER OF PROCEDURES AND FUNCTIONS
000000 CALLED  FROM PASCAL LIBRARY         :       01
000000 *)
```

```
206  000000
207  000000      (*****************************************************************************)
208  000000      ENVIRONMENT
209  005743      PROGRAM RM50100 (INPUT,OUTPUT) ;
210  005743      (*****************************************************************************)
211  005743
212  005743      VAR   INDEX   : INTEGER ;
213  005744            C       : CHAR    ;
214  005745
215  005745      (*****************************************************************************)
216  005745      PROCEDURE
217  005745      WRPOSI ;                                                    (* 09/30/84 *)
218  000000
219  000000      (*
220  000000        PURPOSES:
221  000000        --------
222  000000        ... DESIRED POSITION & ORIENTATION OF GRIPPER --- WRIST POSITION
223  000000        PARAMETERS IN APPEARING ORDER:
224  000000        ------------------------------
225  000000                     (NONE)
226  000000        GLOBAL PARAMETERS:
227  000000        ------------------
228  000000         M , N     : NUMBER OF D.O.F WITHOUT/WITH END-EFFECTOR          INPUTS
229  000000        GLOBAL  VARIABLES:
230  000000        ------------------
231  000000             U     : UNIT MATRIX, HERE AS HELP VARIABLE                 INPUT
232  000000            EMB    : THE ROBOT'S ORIENTATIONAL SUBASSEMBLY BACKWARD     OUTPUT
233  000000            EMF    : THE ROBOT'S ORIENTATIONAL SUBASSEMBLY  FORWARD     OUTPUT
234  000000            TNB    : BACKWARD MATRICES NON-DECOUPLED                    OUTPUT
235  000000            TNF    :  FORWARD MATRICES NON-DECOUPLED                    OUTPUT
236  000000            TMB    : BACKWARD MATRICES     DECOUPLED                    OUTPUT
237  000000            TMF    :  FORWARD MATRICES     DECOUPLED                    OUTPUT
238  000000             TB    : HELP MATRIX                                        OUTPUT
239  000000             TF    : HELP MATRIX                                        OUTPUT
240  000000          EULER0 ,
241  000000          EULERY ,
242  000000          EULER1   : EULER-ANGLES                                      OUTPUTS
243  000000        LOCAL   VARIABLES:
244  000000        ------------------
245  000000                     (SEE BELOW)
246  000000        PROCEDURES AND/OR FUNCTIONS CALLED:
247  000000        -----------------------------------
248  000000          'ARJDOO' : PROCEDURE CONTAINED IN ELEMENT    RM50100-2A
249  000000          'ROTMAT' : PROCEDURE CONTAINED IN ELEMENT    RM50100-2A
250  000000        COMMENTS:
251  000000        --------
252  000000        ... NO ARITHMETIC CALCULATIONS, BUT PURE PHASE SHIFTING
253  000000      *)
254  000005
255  000005      VAR   I , K : INTEGER ;
256  000007
257  000007 [00] BEGIN (*WRPOSI*)
258  000000 [01]
259  000002        FOR K:=M+1     TO N   DO
260  000005              BEGIN  ARJDOO (K,'B',EMB[K-1],EMB[ K ])  END ;
261  000037        FOR K:=N   DOWNTO M+1 DO
262  000042              BEGIN  ARJDOO (K,'F',EMF[ K ],EMF[K-1])  END ;
263  Q00074
264  000074        WRITELN ;
265  000076
266  000076        WRITELN ('Euler-Angle Phi0  about Z-Axis ---)') ; READLN (EULER0) ;
267  000112        WRITELN ('Euler-Angle Theta       Y-Axis ---)') ; READLN (EULERY) ;
268  000126        WRITELN ('Euler-Angle Phi1        Z-Axis ---)') ; READLN (EULER1) ;
269  000142
270  000142        ROTMAT ('Z',EULER0,     U,     TB) ;
271  000152        ROTMAT ('Y',EULERY,    TB,     TF) ;
272  000162        ROTMAT ('Z',EULER1,    TF,TNB[0]) ;
273  000172
274  000172        ROTMAT ('Z',EULER0,EMB[N],     TB) ;
275  000202        ROTMAT ('Y',EULERY,    TB,     TF) ;
276  000212        ROTMAT ('Z',EULER1,    TF,TMB[0]) ;
277  000222
278  000222        WRITELN ;
279  000224
280  000224        WRITELN ('Enter P.X          [0 ... 380]  ---)') ; READLN (I) ;
281  000240        TNB[0].P.X:=I ; TMB[0].P.X:=TMB[0].P.X + TNB[0].P.X ;
282  000246        WRITELN ('Enter P.Y          [0 ... 380]  ---)') ; READLN (I) ;
283  000262        TNB[0].P.Y:=I ; TMB[0].P.Y:=TMB[0].P.Y + TNB[0].P.Y ;
284  000270        WRITELN ('Enter P.Z          [0 ... 630]  ---)') ; READLN (I) ;
285  000304        TNB[0].P.Z:=I ; TMB[0].P.Z:=TMB[0].P.Z + TNB[0].P.Z ;
286  000312
287  000007 [01] END   (*WRPOSI*) ;
288  005745 [00]
```

```
289 005745       (*****************************************************************************)
290 005745       PROCEDURE
291 005745       BOTHOP ;                                                          (* 10/05/84 *)
292 000005
293 000005 [00] BEGIN (*BOTHOP*)
294 000314 [01]
295 000316         STATUS (0,1,M,M) ;
296 000326         SOLVOO (3,1,M,C)   ;
297 000341         SOLVOO (2,0,M,C)   ;
298 000354         SOLVOO (1,M,M,C)   ;
299 000367
300 000367         STATUS (0,1,N,N) ;
301 000377         SOLVOO (4,N,N,C)   ;
302 000412         SOLVOO (5,N,N,C)   ; PPRINT (N) ;
303 000432
304 000005 [01] END   (*BOTHOP*) ;
305 005745 [00]
306 005745       (*****************************************************************************)
307 005745       PROCEDURE
308 005745       ONLYPO ;                                                          (* 10/05/84 *)
309 000005
310 000005 [00] BEGIN (*ONLYPO*)
311 000434 [01]
312 000436         STATUS (0,1,N,N) ;
313 000446         SOLVOO (4,1,N,C)   ;
314 000461         SOLVOO (3,1,N,C)   ;
315 000474         SOLVOO (2,0,N,C)   ;
316 000507         SOLVOO (1,N,N,C)   ; PPRINT (N) ;
317 000527
318 000005 [01] END   (*ONLYPO*) ;
319 005745 [00]
320 005745       (*****************************************************************************)
321 005745       PROCEDURE
322 005745       ONLYOR ;                                                          (* 10/05/84 *)
323 000005
324 000005 [00] BEGIN (*ONLYOR*)
325 000531 [01]
326 000533         STATUS (0,1,N,N) ;
327 000543         SOLVOO (1,N,N,C)   ;
328 000556         SOLVOO (2,N,N,C)   ;
329 000571         SOLVOO (3,N,N,C)   ;
330 000604         SOLVOO (4,N,N,C)   ;
331 000617         SOLVOO (5,N,N,C)   ; PPRINT (N) ;
332 000637
333 000005 [01] END   (*ONLYOR*) ;
334 005745 [00]
335 005745       (*****************************************************************************)
336 005745
337 005745 [00] BEGIN (* main program RM50100 *)
338 000641 [01]
339 000660         WRITELN ('R M 5 0 1 0 0   RUNNING ...') ;
340 000667
341 000667         INITIA ;
342 000672         WRPOSI ;
343 000675
344 000675         WRITELN ('F for Fast / A for All option  ---)') ; READLN (C) ; WRITELN(C) ;
345 000721         WRITELN ('Option 1 P, 2 O only, 3 both   ---)') ; READLN (INDEX) ;
346 000735         IF INDEX=1 THEN ONLYPO ;
347 000743         IF INDEX=2 THEN ONLYOR ;
348 000751         IF INDEX=3 THEN BOTHOP ;
349 000757
350 000757         WRITELN ;
351 000761         WRITELN ;
352 000763         WRITELN ('Best possible solution:') ;
353 000772         WRITELN ;
354 000774         FOR INDEX:=1 TO N DO    WRITELN (INDEX:10,LINK[INDEX].ITHETA:10) ;
355 001017 [01]    IF C='A' THEN BEGIN
356 001022 [02]
357 001022         STPLOT ;
358 001027         PLOT (FEHLER[3].ERVAR,IRE,FEHLER[3].ERPAR,N,3,1,N,LINK[3].ITHETA,1) ;
359 001116         PLOT (FEHLER[2].ERVAR,IRE,FEHLER[2].ERPAR,N,2,0,N,LINK[2].ITHETA,1) ;
360 001205         PLOT (FEHLER[1].ERVAR,IRE,FEHLER[1].ERPAR,N,1,N,N,LINK[1].ITHETA,1) ;
361 001274         PLOT (FEHLER[4].ERVAR,IRE,FEHLER[4].ERPAR,N,4,N,N,LINK[4].ITHETA,0) ;
362 001363         PLOT (FEHLER[5].ERVAR,IRE,FEHLER[5].ERPAR,N,5,N,N,LINK[5].ITHETA,0) ;
363 001452         ENDPLT ;
364 001457
365 001457 [02]                    END ;
366 001457 [01]
367 001457 [01] END   (* main program RM50100 *)
368 001457 [00]
369 001457       (*****************************************************************************)
370 001460       .                            (* TERMINATION PERIOD  end of element RM50100-0A   *)
371 001460       (*****************************************************************************)
Compilation completed - no errors found.
```

```
)3MAP,S ,RM50100

Collector 31R2B (841126 1925:45) 1985 Jul 11 Thu 1430:16
)IN F.RM50100-3A

        1.              IN F.RM50100-3A
)IN F.RM50100-2A

        2.              IN F.RM50100-2A
)IN F.RM50100-1A

        3.              IN F.RM50100-1A
)IN F.RM50100-0A

        4.              IN F.RM50100-0A
)LIB PASCAL*LIB

        5.              LIB PASCAL*LIB
)LIB AFTN*NEW-PLOT

        6.              LIB AFTN*NEW-PLOT
)CLASS FILE

        7.              CLASS FILE
)IN F.DES-PLOT00

        8.              IN F.DES-PLOT00
)END

        9.              END

AFCM status of output element is UNKNOWN
Quarter-word sensitive

ADDRESS LIMITS     001000 052023      21012 IBANK WORDS DECIMAL
                   053000 115401      17666 DBANK WORDS DECIMAL
STARTING ADDRESS   046746


          SEGMENT $MAIN$          001000 052023      053000 115401

RPLUS-DIAG                $(1)    001000 001547    $(2)    053000 053422    14 MAR 78   10:53:15
DUMP-UTIL                                          $(2)    053423 053470    28 JAN 81   13:53:22
TABLE$/SYS75              $(1)    001550 001747                             28 APR 83   02:46:39
CHSTR                     $(1)    001750 002132    $(0)    053471 053474    07 JAN 83   10:41:43
MEMINTERFACE              $(1)    002133 002244    $(2)    053475 053536    23 MAR 84   15:20:12
MEM$ERR(COMMONBLOCK)                               053537 053541
M$PKT$                                             $(0)    053542 053547    08 SEP 83   11:01:20
                                                   $(012)  MEM$ERR
F2ER                      $(1)    002245 003003    $(2)    053550 053623    29 FEB 84   14:53:24
CHACT                     $(1)    003004 003036                             07 JAN 83   10:40:34
F2FR1I/FTNINT             $(1)    003037 003101                             13 JAN 83   15:42:21
F2ATAN$/FTNINT            $(1)    003102 003176                             06 OCT 82   13:40:52
FTNPMD2                                            $(2)    053624 054345    23 FEB 83   09:37:20
                                                   $(012)  DMP$
F2SUBSTR                  $(1)    003177 003570    $(0)    054346 054360    07 JAN 83   10:42:17
F2MATHERR                 $(1)    003571 004103    $(2)    054361 054530    28 FEB 83   10:29:21
CRELAD$/SYS75             $(1)    004104 004511    $(2)    054531 055176    28 APR 83   02:42:48
F2FR1R1/FTNINT            $(1)    004512 004554                             13 JAN 83   15:42:23
CABSAD$/SYS75             $(1)    004555 005245    $(2)    055177 055645    28 APR 83   02:42:17
F2ALOG$/FTNINT            $(1)    005246 005352                             13 JAN 83   15:42:00
FTNPMD3                                            $(2)    055646 057024    05 MAY 82   14:35:45
                                                   $(012)  DMP$
INFOR$/SYS75              $(1)    005353 005755    $(2)    057025 057060    28 APR 83   02:43:39
FTNPMD5                                            $(2)    057061 060223    05 APR 83   15:01:10
FDASC$/SYS75              $(1)    005756 006152                             28 APR 83   02:43:16
CHREL                     $(1)    006153 006600    $(0)    060224 060234    07 JAN 83   10:41:07
FTNPMD4                                            $(2)    060235 061233    23 FEB 83   09:38:46
CHOPN                                              $(4)    061234 061436    14 AUG 80   16:23:31
CHCAT                     $(1)    006601 007041    $(0)    061437 061444    07 JAN 83   10:42:52
CHMOVE                    $(1)    007042 007315                             10 JAN 83   09:06:03
VFTNEQU$/FORFTN                                                              03 FEB 83   08:15:54
F2EXIT                    $(1)    007316 007316    $(2)    061445 061510    16 DEC 82   17:20:55
F2SINCOS$/FTNINT          $(1)    007317 007452                             13 JAN 83   15:42:25
F2ARGCK                   $(1)    007453 010130    $(0)    061511 061644    07 JAN 83   10:39:49
CBEP$$CML                                                                    19 FEB 81   10:55:57
FORCOM$/FORFTN                                     $(2)    061645 061653    26 OCT 82   11:02:56
F2SQRT$/FTNINT            $(1)    010131 010157                             13 JAN 83   15:42:31
F2VMOVE$                  $(1)    010160 012443    $(2)    061654 061727    12 SEP 83   17:52:14
```

FTNPMD1	$(1)	012444 012444	$(2)	061730 063067	22 DEC 83	14:09:05
			$(036)	PMD$COM		
CBEP$$FTN			$(0)	063070 063073	03 OCT 83	14:12:15
F2AUTO	$(1)	012445 012734	$(0)	063074 063170	24 MAR 83	16:53:23
F2CLOSE	$(1)	012735 012772	$(2)	063171 063173	19 NOV 82	15:51:44
F2BDREQU$/FORFTN					20 DEC 82	08:15:20
FTNCMLERR(COMMONBLOCK)				063174 063174		
PMD$COM(COMMONBLOCK)				063175 063176		
F2INIT			$(2)	063177 063607	15 JUN 84	17:58:18
ERU$/SYS75R1					10 SEP 83	00:30:51
P$MDP-UTIL	$(1)	012773 013014			03 MAR 84	12:16:17
	$(3)	013015 013070				
P$WRE	$(1)	013071 013256	$(2)	063610 063622	03 MAR 84	12:17:26
	$(3)	013257 013267				
READ$INFOR	$(1)	013270 013337	$(2)	063623 063757	06 APR 84	09:04:26
	$(3)	013340 013356				
P$WRO	$(1)	013357 013440	$(2)	063760 063763	03 MAR 84	12:17:37
PASC$WBD	$(1)	013441 015300	$(0)	063764 064017	20 MAR 84	15:09:59
	$(5)	015301 015314	$(4)	064020 064060		
	$(7)	015315 015375	$(016)	PAS$$$		
	$(011)	015376 015546				
	$(013)	015547 015655				
	$(015)	015656 016023				
P$WRD-UTIL	$(1)	016024 016133	$(2)	064061 064061	03 MAR 84	12:17:22
	$(3)	016134 016137				
P$GETC	$(1)	016140 016242			03 MAR 84	12:15:56
	$(3)	016243 016243				
CALL$WBD	$(1)	016244 016407	$(2)	064062 064072	03 MAR 84	12:15:03
	$(3)	016410 016411				
P$WRF	$(1)	016412 016576	$(2)	064073 064105	03 MAR 84	12:17:29
	$(3)	016577 016607				
P$WRS	$(1)	016610 016761	$(2)	064106 064114	03 MAR 84	12:17:40
	$(3)	016762 016762				
CMLPASINT	$(1)	016763 017027	$(2)	064115 064117	03 MAR 84	12:15:09
	$(3)	017030 017041	$(4)	064120 064141		
P$WRC	$(1)	017042 017110	$(2)	064142 064146	03 MAR 84	12:17:19
COMMUN$AREA			$(2)	064147 064153	03 MAR 84	12:15:10
P$PUTC	$(1)	017111 017177	$(2)	064154 064155	03 MAR 84	12:16:30
P$INPF	$(1)	017200 017407	$(2)	064156 064167	03 MAR 84	12:16:11
	$(3)	017410 017412				
P$OPCL	$(1)	017413 017433			03 MAR 84	12:16:19
P$OUTF	$(1)	017434 017601	$(2)	064170 064171	03 MAR 84	12:16:21
	$(3)	017602 017605				
P$RDC	$(1)	017606 017627	$(2)	064172 064176	03 MAR 84	12:16:39
PW-A2	$(1)	017630 017657			03 MAR 84	12:18:44
P$WRI	$(1)	017660 017750	$(2)	064177 064217	03 MAR 84	12:17:33
	$(3)	017751 017751				
PAS$F2FCA			$(0)	064220 070143	03 MAR 84	12:17:46
PASFTNINIT	$(1)	017752 020016	$(2)	070144 070152	03 MAR 84	12:17:49
	$(3)	020017 020020				
P$RLN	$(1)	020021 020032			03 MAR 84	12:16:54
P$RDI	$(1)	020033 020134	$(2)	070153 070157	03 MAR 84	12:16:44
	$(3)	020135 020135				
P$TRNC	$(1)	020136 020164			03 MAR 84	12:17:16
	$(3)	020165 020171				
RUN$ERROR	$(1)	020172 020615	$(2)	070160 070234	20 MAR 84	15:06:12
	$(3)	020616 020655	$(4)	070235 070465		
	$(5)	020656 020707	$(6)	070466 070476		
	$(7)	020710 021766				
PROLOG	$(1)	021767 022271	$(2)	070477 070552	03 MAR 84	12:18:40
	$(3)	022272 022310				
P$ENTRY-EXIT	$(1)	022311 022501	$(2)	070553 070554	03 MAR 84	12:15:41
PAS$$$(COMMONBLOCK)						
DUMP-SCT			$(2)	070555 071056	01 DEC 82	09:42:19
OLD$CM(COMMONBLOCK)				071057 071062		
V$DATA(COMMONBLOCK)				071063 071125		
F2OLDCOMMON$	$(1)	022502 023276	$(0)	071126 071517	22 DEC 83	10:45:37
	$(3)	INFO-010-LC	$(014)	V$DATA		
	$(015)	OLD$CM				
F2CDCD$			$(2)	071520 071565	06 FEB 81	10:49:45
F2RTRN$			$(2)	071566 071570	06 FEB 81	10:49:35
FTNDUMP	$(1)	023277 026066	$(0)	071571 073614	30 DEC 82	16:18:28
	$(3)	INFO-010-LC	$(014)	DMP$		
F2ARG$	$(1)	026067 026070	$(2)	073615 073624	17 MAR 81	16:34:20
			$(044)	FTNCMLERR		

DMP$(COMMONBLOCK)				073625 073665		
F2ABS	$(1)	026071 026407			16 NOV 83	11:11:12
	$(3)	INFO-010-LC				
	$(5)	026410 026572				
	$(7)	026573 026744				
F2ACTIV$/FORFTN	$(1)	026745 027044	$(2)	073666 075010	17 OCT 83	13:09:17
	$(3)	027045 027144				
F2CON			$(2)	075011 075743	02 MAR 83	16:18:45
			$(036)	PMD$COM		
			$(044)	FTNCMLERR		
F2SCT			$(2)	075744 076246	27 JAN 84	16:08:34
PPLGTH	$(1)	027145 027533	$(0)	076247 076304	08 DEC 83	14:14:56
	$(3)	INFO-010-LC	$(4)	076305 076361		
	$(013)	PPCOMM				
PPCLIP	$(1)	027534 030226	$(0)	076362 076412	05 JAN 83	18:57:38
	$(3)	INFO-010-LC	$(4)	076413 076434		
	$(013)	PPCOMM	$(6)	076435 076471		
			$(012)	076472 076474		
PPSCAL	$(1)	030227 030450	$(0)	076475 076504	05 DEC 82	14:25:39
	$(3)	INFO-010-LC	$(4)	076505 076526		
	$(013)	PPCOMM	$(6)	076527 076562		
			$(012)	076563 076571		
PPSCLE	$(1)	030451 030733	$(0)	076572 076604	06 DEC 82	13:54:17
	$(3)	INFO-010-LC	$(4)	076605 076635		
	$(013)	PPCOMM	$(6)	076636 076651		
			$(012)	076652 076652		
PPCHIS(COMMONBLOCK)				076653 076655		
PPCAX(COMMONBLOCK)				076656 076661		
PPSTAN	$(1)	030734 031033	$(4)	076662 076677	08 DEC 82	09:23:20
	$(3)	INFO-010-LC	$(6)	076700 076717		
	$(013)	PPCAX	$(012)	076720 076735		
	$(015)	PPMCOM	$(014)	PPCHIS		
			$(016)	PPCOMM		
PPFRAM	$(1)	031034 031104	$(0)	076736 076750	05 DEC 82	14:26:01
	$(3)	INFO-010-LC	$(4)	076751 076756		
	$(013)	PPCOMM	$(6)	076757 076764		
			$(012)	076765 076766		
PRIMC	$(1)	031105 031677	$(0)	076767 077007	27 MAR 83	15:47:57
	$(3)	INFO-010-LC	$(4)	077010 077065		
	$(013)	PPMCOM	$(6)	077066 077137		
			$(012)	077140 077143		
			$(014)	PPCOMM		
PPHELP	$(1)	031700 032003	$(0)	077144 077145	05 DEC 82	14:26:35
	$(3)	INFO-010-LC	$(4)	077146 077304		
	$(013)	PPCOMM	$(6)	077305 077314		
			$(012)	077315 077315		
PPSDEF	$(1)	032004 032122	$(0)	077316 077316	05 DEC 82	14:26:30
	$(3)	INFO-010-LC	$(4)	077317 077345		
			$(6)	077346 077352		
PPPUSH	$(1)	032123 032321	$(0)	077353 077734	05 DEC 82	14:25:51
	$(3)	INFO-010-LC	$(4)	077735 077753		
	$(013)	PPCOMM	$(6)	077754 100006		
			$(012)	100007 100021		
PRIMB	$(1)	032322 033666	$(0)	100022 101476	09 DEC 82	15:40:13
	$(3)	INFO-010-LC	$(4)	101477 101565		
	$(013)	PPCOMM	$(6)	101566 101665		
			$(012)	101666 101703		
CVTTOU	$(1)	033667 033767	$(0)	101704 101705	05 DEC 82	14:26:09
	$(3)	INFO-010-LC	$(4)	101706 101725		
			$(012)	101726 101726		
PRIMA/FILE	$(1)	033770 034431	$(0)	101727 101760	07 JAN 83	14:29:57
	$(3)	INFO-010-LC	$(4)	101761 102170		
			$(6)	102171 102260		
			$(012)	102261 102271		
PPRNG	$(1)	034432 035012	$(0)	102272 102272	06 DEC 82	13:49:52
	$(3)	INFO-010-LC	$(4)	102273 102325		
	$(013)	PPCOMM	$(6)	102326 102435		
			$(012)	102436 102461		

PPTYPE(COMMONBLOCK)				102462	102462		
PPBGN	$(1)	035013 035164	$(0)	102463	102473	05 DEC 82	14:26:43
	$(035)	PPCOMM	$(4)	102474	102525		
	$(037)	INFO-010-LC	$(6)	102526	102610		
			$(012)	102611	102647		
			$(034)	102650	102723		
			$(036)	PPTYPE			
PPMODE	$(1)	035165 035342	$(0)	102724	102726	05 DEC 82	14:26:15
	$(3)	INFO-010-LC	$(4)	102727	102735		
	$(013)	PPCOMM	$(6)	102736	103037		
			$(012)	103040	103064		
PPTEXT	$(1)	035343 035606	$(0)	103065	103120	05 DEC 82	14:26:37
	$(3)	INFO-010-LC	$(4)	103121	103161		
	$(013)	PPCOMM	$(6)	103162	103257		
			$(012)	103260	103275		
PPLINE	$(1)	035607 035636	$(0)	103276	103301	05 DEC 82	14:25:52
	$(3)	INFO-010-LC	$(4)	103302	103307		
			$(6)	103310	103315		
			$(012)	103316	103317		
PPMCOM(COMMONBLOCK)				103320	103325		
PPSET	$(1)	035637 036142	$(0)	103326	103330	08 DEC 82	09:19:50
	$(3)	INFO-010-LC	$(4)	103331	103373		
	$(013)	PPMCOM	$(6)	103374	103506		
			$(012)	103507	103550		
			$(014)	PPCOMM			
PPCOMM(COMMONBLOCK)				103551	103606		
PPDRAW	$(1)	036143 037210	$(0)	103607	103647	06 SEP 84	12:15:26
	$(3)	INFO-010-LC	$(4)	103650	103741		
	$(013)	PPCOMM	$(6)	103742	104107		
			$(012)	104110	104123		
RM50100-3A	$(1)	037211 037333	$(0)	104124	104143	11 JUL 85	14:01:51
	$(5)	037334 037336	$(4)	104144	104160		
	$(7)	037337 037340	$(016)	PAS$$$			
	$(011)	037341 037355					
	$(013)	037356 037364					
	$(015)	037365 037366					
RM50100-2A	$(1)	037367 042560	$(0)	104161	104200	11 JUL 85	14:14:40
	$(5)	042561 042563	$(4)	104201	104271		
	$(7)	042564 042640	$(016)	PAS$$$			
	$(011)	042641 043007					
	$(013)	043010 043170					
	$(015)	043171 043224					
RM50100-1A	$(1)	043225 045000	$(0)	104272	104311	11 JUL 85	14:21:03
	$(5)	045001 045003	$(4)	104312	104336		
	$(7)	045004 045027	$(016)	PAS$$$			
	$(011)	045030 045054					
	$(013)	045055 045075					
	$(015)	045076 045265					
RM50100-0A	$(1)	045266 046771	$(0)	104337	112356	11 JUL 85	14:26:47
	$(5)	046772 046774	$(4)	112357	112404		
	$(7)	046775 047014	$(6)	112405	112502		
	$(011)	047015 047051	$(016)	PAS$$$			
	$(013)	047052 047104					
	$(015)	047105 047235					
DES-PLOT00	$(1)	047236 052023	$(0)	112503	113713	05 NOV 84	15:19:22
	$(3)	INFO-010-LC	$(4)	113714	114315		
			$(6)	114316	115161		
			$(012)	115162	115401		

Common Banks referenced

ALOG$	0400001	FCORE$	0400025

END MAP. ERRORS: 0 TIME: 51.561 STORAGE: 21612/24/035177/0120277

```
==========================
* G P R C     S T A T U S *
==========================
Matrix System of  Robot 501
```

Theta Offset [Degrees]	Theta [Degrees]	Alfa [Degrees]	Ax [mm]	Dz [mm]	Cascade #
90	16	90.00	0.00	250.00	1
0	10	0.00	220.00	0.00	2
0	-1	0.00	160.00	0.00	3
90	81	90.00	0.00	0.00	4
0	-46	0.00	0.00	0.00	5
Euler-Angles/[Degrees] :		240	0	0	
Position /[mm] :		-80.00	280.00	300.00	

F O R W A R D

```
[0]T[5]
 -0.50000  0.86603  0.00000 -103.28
 -0.86603 -0.50000 -0.00000  360.17
  0.00000 -0.00000  1.00000  313.23
  0.00000  0.00000  0.00000    1.00
```

```
[0]A[1]     Theta[1] =     16
            ----------------
 -0.27564  0.00000  0.96126    0.00
  0.96126  0.00000  0.27564    0.00
  0.00000  1.00000 -0.00000  250.00
  0.00000  0.00000  0.00000    1.00
```

```
[1]T[5]
 -0.69466 -0.71934 -0.00000  374.69
 -0.00000 -0.00000  1.00000   63.23
 -0.71934  0.69466 -0.00000   -0.00
  0.00000  0.00000  0.00000    1.00
```

```
[1]A[2]     Theta[2] =     10
            ----------------
  0.98481 -0.17365  0.00000  216.66
  0.17365  0.98481 -0.00000   38.20
  0.00000  0.00000  1.00000    0.00
  0.00000  0.00000  0.00000    1.00
```

```
[2]T[5]
 -0.68410 -0.70841  0.17365  159.98
  0.12063  0.12491  0.98481   -2.79
 -0.71934  0.69466 -0.00000    0.00
  0.00000  0.00000  0.00000    1.00
```

```
[2]A[3]     Theta[3] =     -1
            ----------------
  0.99985  0.01745 -0.00000  159.98
 -0.01745  0.99985 -0.00000   -2.79
  0.00000  0.00000  1.00000    0.00
  0.00000  0.00000  0.00000    1.00
```

```
[3]T[5]
 -0.68611 -0.71048  0.15643    0.00
  0.10867  0.11253  0.98769    0.00
 -0.71934  0.69466 -0.00000    0.00
  0.00000  0.00000  0.00000    1.00
```

```
[3]A[4]     Theta[4] =     81
            ----------------
 -0.98769  0.00000  0.15643    0.00
  0.15643  0.00000  0.98769    0.00
  0.00000  1.00000 -0.00000    0.00
  0.00000  0.00000  0.00000    1.00
```

```
[4]T[5]
  0.69466  0.71934 -0.00000    0.00
 -0.71934  0.69466 -0.00000    0.00
  0.00000  0.00000  1.00000    0.00
  0.00000  0.00000  0.00000    1.00
```

```
[4]A[5]     Theta[5] =    -46
            ----------------
  0.69466  0.71934 -0.00000    0.00
 -0.71934  0.69466 -0.00000    0.00
  0.00000  0.00000  1.00000    0.00
  0.00000  0.00000  0.00000    1.00
```

```
[5]T[5]
  1.00000  0.00000  0.00000    0.00
  0.00000  1.00000  0.00000    0.00
  0.00000  0.00000  1.00000    0.00
  0.00000  0.00000  0.00000    1.00
```

B A C K W A R D

```
 -0.50000  0.86603  0.00000  -80.00
 -0.86603 -0.50000 -0.00000  280.00
  0.00000 -0.00000  1.00000  300.00
  0.00000  0.00000  0.00000    1.00
```

```
 -0.27564  0.96126  0.00000   -0.00
  0.96126  0.27564  0.00000 -250.00
  0.00000  0.00000  0.00000    1.00
```

```
 -0.69466 -0.71934 -0.00000  291.20
 -0.00000 -0.00000  1.00000   50.00
 -0.71934  0.69466 -0.00000    0.28
  0.00000  0.00000  0.00000    1.00
```

```
  0.98481  0.17365  0.00000 -220.00
  0.17365  0.98481 -0.00000   -0.00
  0.00000  0.00000  1.00000   -0.00
  0.00000  0.00000  0.00000    1.00
```

```
 -0.68411 -0.70841  0.17365   75.46
  0.12063  0.12491  0.98481   -1.33
 -0.71934  0.69466  0.00000    0.28
  0.00000  0.00000  0.00000    1.00
```

```
  0.99985 -0.01745  0.00000 -160.00
  0.01745  0.99985 -0.00000   -0.00
  0.00000  0.00000  1.00000   -0.00
  0.00000  0.00000  0.00000    1.00
```

```
 -0.68611 -0.71048  0.15643  -84.53
  0.10867  0.11253  0.98769   -0.01
 -0.71934  0.69466  0.00000    0.28
  0.00000  0.00000  0.00000    1.00
```

```
 -0.98769  0.00000  0.15643    0.00
  0.15643  0.00000  0.98769    0.00
  0.00000  1.00000 -0.00000    0.00
  0.00000  0.00000  0.00000    1.00
```

```
  0.69466  0.71934  0.00000   83.48
 -0.71934  0.69466  0.00000    0.28
  0.00000  0.00000  1.00000  -13.23
  0.00000  0.00000  0.00000    1.00
```

```
  0.69466 -0.71934  0.00000    0.00
  0.71934  0.69466 -0.00000    0.00
  0.00000  0.00000  1.00000    0.00
  0.00000  0.00000  0.00000    1.00
```

```
  1.00000 -0.00000  0.00000   57.79
  0.00000  1.00000  0.00000   60.25
  0.00000 -0.00000  1.00000  -13.23
  0.00000  0.00000  0.00000    1.00
```

```
==========================
* G P R C     S T A T U S *
==========================
Matrix System of  Robot 501
```

Theta Offset [Degrees]	Theta [Degrees]	Alfa [Degrees]	Ax [mm]	Dz [mm]	Cascade #
90	16	90.00	0.00	250.00	1
0	-22	0.00	220.00	0.00	2
0	79	0.00	160.00	0.00	3
90	1	90.00	0.00	0.00	4
0	0	0.00	0.00	0.00	5
Euler-Angles/[Degrees] :		240	0	0	
Position /[mm] :		-80.00	280.00	300.00	

F O R W A R D

[0]T[5]

0.23375	0.96126	-0.14607	-80.24	-0.50000	0.86603	0.00000	-80.00
-0.81520	0.27564	0.50939	279.85	-0.86603	-0.50000	0.00000	280.00
0.52992	-0.00000	0.84805	301.77	0.00000	-0.00000	1.00000	300.00
0.00000	0.00000	0.00000	1.00	0.00000	0.00000	0.00000	1.00

[0]A[1] Theta[1] = 16

-0.27564	0.00000	0.96126	0.00	-0.27564	0.96126	0.00000	-0.00
0.96126	0.00000	0.27564	0.00	-0.00000	-0.00000	1.00000	-250.00
0.00000	1.00000	-0.00000	250.00	0.96126	0.27564	0.00000	-0.00
0.00000	0.00000	0.00000	1.00	0.00000	0.00000	0.00000	1.00

[1]T[5]

-0.84805	0.00000	0.52992	291.12	-0.69466	-0.71934	-0.00000	291.20
0.52992	-0.00000	0.84805	51.77	0.00000	-0.00000	1.00000	50.00
0.00000	1.00000	-0.00000	0.00	-0.71934	0.69466	0.00000	0.28
0.00000	0.00000	0.00000	1.00	0.00000	0.00000	0.00000	1.00

[1]A[2] Theta[2] = -22

0.92718	0.37461	-0.00000	203.98	0.92718	-0.37461	0.00000	-220.00
-0.37461	0.92718	-0.00000	-82.41	0.37461	0.92718	-0.00000	-0.00
0.00000	0.00000	1.00000	0.00	0.00000	0.00000	1.00000	-0.00
0.00000	0.00000	0.00000	1.00	0.00000	0.00000	0.00000	1.00

[2]T[5]

-0.98481	0.00000	0.17365	30.53	-0.64408	-0.66696	-0.37461	31.27
0.17365	0.00000	0.98481	157.06	-0.26022	-0.26947	0.92718	155.45
0.00000	1.00000	-0.00000	0.00	-0.71934	0.69466	0.00000	0.28
0.00000	0.00000	0.00000	1.00	0.00000	0.00000	0.00000	1.00

[2]A[3] Theta[3] = 79

0.19081	-0.98163	0.00000	30.53	0.19081	0.98163	0.00000	-160.00
0.98163	0.19081	-0.00000	157.06	-0.98163	0.19081	-0.00000	-0.00
0.00000	0.00000	1.00000	0.00	-0.00000	0.00000	1.00000	-0.00
0.00000	0.00000	0.00000	1.00	0.00000	0.00000	0.00000	1.00

[3]T[5]

-0.01745	0.00000	0.99985	0.00	-0.37834	-0.39178	0.83867	-1.44
0.99985	-0.00000	0.01745	0.00	0.58259	0.60329	0.54464	-1.03
0.00000	1.00000	-0.00000	0.00	-0.71934	0.69466	0.00000	0.28
0.00000	0.00000	0.00000	1.00	0.00000	0.00000	0.00000	1.00

[3]A[4] Theta[4] = 1

-0.01745	0.00000	0.99985	0.00	-0.01745	0.99985	0.00000	0.00
0.99985	0.00000	0.01745	0.00	-0.00000	-0.00000	1.00000	0.00
0.00000	1.00000	-0.00000	0.00	0.99985	0.01745	0.00000	0.00
0.00000	0.00000	0.00000	1.00	0.00000	0.00000	0.00000	1.00

[4]T[5]

1.00000	-0.00000	0.00000	0.00	0.58910	0.61003	0.52992	-1.01
0.00000	1.00000	-0.00000	0.00	-0.71934	0.69466	0.00000	0.28
0.00000	0.00000	1.00000	0.00	-0.36811	-0.38119	0.84805	-1.46
0.00000	0.00000	0.00000	1.00	0.00000	0.00000	0.00000	1.00

[4]A[5] Theta[5] = 0

1.00000	-0.00000	0.00000	0.00	1.00000	-0.00000	0.00000	0.00
0.00000	1.00000	-0.00000	0.00	0.00000	1.00000	-0.00000	0.00
0.00000	0.00000	1.00000	0.00	0.00000	0.00000	1.00000	0.00
0.00000	0.00000	0.00000	1.00	0.00000	0.00000	0.00000	1.00

[5]T[5]

1.00000	0.00000	0.00000	0.00	0.58910	0.61003	0.52992	-1.01
0.00000	1.00000	0.00000	0.00	-0.71934	0.69466	0.00000	0.28
0.00000	0.00000	1.00000	0.00	-0.36811	-0.38119	0.84805	-1.46
0.00000	0.00000	0.00000	1.00	0.00000	0.00000	0.00000	1.00

```
==========================
* G P R C     S T A T U S *
==========================
Matrix System of  Robot 501
```

Theta Offset [Degrees]	Theta [Degrees]	Alfa [Degrees]	Ax [mm]	Dz [mm]	Cascade #
90	16	90.00	0.00	250.00	1
0	-22	0.00	220.00	0.00	2
0	79	0.00	160.00	0.00	3
90	33	90.00	0.00	0.00	4
0	-46	0.00	0.00	0.00	5
Euler-Angles/[Degrees] :		240	0	0	
Position /[mm] :		-80.00	280.00	300.00	

F O R W A R D B A C K W A R D

[0]T[5]

-0.50000	0.86603	0.00000	-80.24	-0.50000	0.86603	0.00000	-80.00
-0.86603	-0.50000	-0.00000	279.85	-0.86603	-0.50000	0.00000	280.00
0.00000	-0.00000	1.00000	301.77	0.00000	-0.00000	1.00000	300.00
0.00000	0.00000	0.00000	1.00	0.00000	0.00000	0.00000	1.00

[0]A[1] Theta[1] = 16

-0.27564	0.00000	0.96126	0.00	-0.27564	0.96126	0.00000	-0.00
0.96126	0.00000	0.27564	0.00	-0.00000	-0.00000	1.00000	-250.00
0.00000	1.00000	-0.00000	250.00	0.96126	0.27564	0.00000	-0.00
0.00000	0.00000	0.00000	1.00	0.00000	0.00000	0.00000	1.00

[1]T[5]

-0.69466	-0.71934	-0.00000	291.12	-0.69466	-0.71934	-0.00000	291.20
-0.00000	-0.00000	1.00000	51.77	0.00000	-0.00000	1.00000	50.00
-0.71934	0.69466	-0.00000	0.00	-0.71934	0.69466	0.00000	0.28
0.00000	0.00000	0.00000	1.00	0.00000	0.00000	0.00000	1.00

[1]A[2] Theta[2] = -22

0.92718	0.37461	-0.00000	203.98	0.92718	-0.37461	0.00000	-220.00
-0.37461	0.92718	-0.00000	-82.41	0.37461	0.92718	-0.00000	-0.00
0.00000	0.00000	1.00000	0.00	0.00000	0.00000	1.00000	-0.00
0.00000	0.00000	0.00000	1.00	0.00000	0.00000	0.00000	1.00

[2]T[5]

-0.64408	-0.66696	-0.37461	30.53	-0.64408	-0.66696	-0.37461	31.27
-0.26022	-0.26947	0.92718	157.06	-0.26022	-0.26947	0.92718	155.45
-0.71934	0.69466	-0.00000	0.00	-0.71934	0.69466	0.00000	0.28
0.00000	0.00000	0.00000	1.00	0.00000	0.00000	0.00000	1.00

[2]A[3] Theta[3] = 79

0.19081	-0.98163	0.00000	30.53	0.19081	0.98163	0.00000	-160.00
0.98163	0.19081	-0.00000	157.06	-0.98163	0.19081	-0.00000	-0.00
0.00000	0.00000	1.00000	0.00	-0.00000	0.00000	1.00000	-0.00
0.00000	0.00000	0.00000	1.00	0.00000	0.00000	0.00000	1.00

[3]T[5]

-0.37834	-0.39178	0.83867	0.00	-0.37834	-0.39178	0.83867	-1.44
0.58259	0.60329	0.54464	0.00	0.58259	0.60329	0.54464	-1.03
-0.71934	0.69466	-0.00000	0.00	-0.71934	0.69466	0.00000	0.28
0.00000	0.00000	0.00000	1.00	0.00000	0.00000	0.00000	1.00

[3]A[4] Theta[4] = 33

-0.54464	0.00000	0.83867	0.00	-0.54464	0.83867	0.00000	0.00
0.83867	0.00000	0.54464	0.00	-0.00000	-0.00000	1.00000	0.00
0.00000	1.00000	-0.00000	0.00	0.83867	0.54464	0.00000	0.00
0.00000	0.00000	0.00000	1.00	0.00000	0.00000	0.00000	1.00

[4]T[5]

0.69466	0.71934	-0.00000	0.00	0.69466	0.71934	0.00000	-0.08
-0.71934	0.69466	-0.00000	0.00	-0.71934	0.69466	0.00000	0.28
0.00000	0.00000	1.00000	0.00	0.00000	-0.00000	1.00000	-1.77
0.00000	0.00000	0.00000	1.00	0.00000	0.00000	0.00000	1.00

[4]A[5] Theta[5] = -46

0.69466	0.71934	-0.00000	0.00	0.69466	-0.71934	0.00000	0.00
-0.71934	0.69466	-0.00000	0.00	0.71934	0.69466	-0.00000	0.00
0.00000	0.00000	1.00000	0.00	0.00000	0.00000	1.00000	0.00
0.00000	0.00000	0.00000	1.00	0.00000	0.00000	0.00000	1.00

[5]T[5]

1.00000	0.00000	0.00000	0.00	1.00000	-0.00000	0.00000	-0.26
0.00000	1.00000	0.00000	0.00	0.00000	1.00000	0.00000	0.13
0.00000	0.00000	1.00000	0.00	0.00000	-0.00000	1.00000	-1.77
0.00000	0.00000	0.00000	1.00	0.00000	0.00000	0.00000	1.00

APPENDIX 8

8.1) Estimation of computing costs for $\mathbf{U}_{pj}$ and $\mathbf{U}_{pjk}$

The manipulation of

$$\mathbf{U}_{pj} = {}^{o}\mathbf{A}_1 . {}^{1}\mathbf{A}_2 \ldots {}^{j-1}\mathbf{A'}_j \ldots {}^{p-2}\mathbf{A}_{p-1} . {}^{p-1}\mathbf{A}_p \quad \text{for } j=<p$$

involves (p-1) matrix multiplications such that the computation of $\mathbf{U}_{pj}$ requries $k_A(p-1)$ additions and $k_M(p-1)$ multiplications where $k_A=48$ and $k_M=64$ are respectively the number of additions and multiplications required for a single matrix multiplication if one excludes the computing cost of ${}^{j-1}\mathbf{A'}_j=\mathbf{Q'} . {}^{j-1}\mathbf{A}_j$. Varying j from 1 to p step 1 and adding up the p particular computing costs, those numbers increase to k.p.(p-1). This number is the number of arithmetic operations required for the computation of all the matrix compositions $\mathbf{U}_{pj}$ of the particular first order derivative class involving only p A-matrices.

(Set $k=k_A$ in the case of additions, $k=k_M$ in the case of multiplications)

The total computing cost is found by varying the index p of the summation

$$\sum_{p=1}^{n} kp(p-1) = k[1/6\ n(n+1)(2n+1) - 1/2\ n(n+1)]$$

which as is seen, yields the order n^3

Similarly, the computation of $\mathbf{U}_{pjk}$ requires the following number of additions and multiplications, which is of order n^4:

$$\sum_{p=1}^{n} kp^2(p-1) = k[1/4\ n^2(n+1)^2 - 1/6\ n(n+1)(2n+1)]$$

8.2.) Number of RJDs for the various SMP versions

We refer to Fig. 5.3.1 to evaluate the RJD-cost of the SMP. Varying j=1 to p, we obtain the number of the RJDs necessary for establishing the 2nd order derivatives of the class p which is:

$$N_{2nd} = \sum_{j=1}^{p} \{j^2\} = p.(p+1).(2p+1)/6$$

The total number of RJDs evaluated for a class p is given as follows:

Function Block		Number of RJDs required
SKB	(Standard kin. block)	p
FOPD	(1st order part.deriv)	p.(p+1)/2
SOPD	(2nd order part.deriv)	p.(p+1).(2p+1)/6

We have to vary p=1 to n, in order to take all possible classes p into consideration. Doing this, we find the total number of RJDs needed for a robot with n d.o.f.

$$N_{complete} = n^4/12 + n^3/2 + 17n^2/12 + n$$

The total number of RJDs required by the first variation of the SMP (Fig.5.3.2) evaluated for a class p is given as follows:

Function Block		Number of RJDs Required
SKB	(Standard Kin. Block)	p
FOPD	(1st order part.deriv)	p.(p+1)/2

We have to vary p=1 to n, in order to take all possible classes p into consideration. Doing this, we find the total number of RJDs needed for a robot with n d.o.f.:

$$N_{1st\ variation} = n^3/6 + n^2 + 10n/12$$

The total number of RJDs for the 2nd variation of Fig.5.3.3, evaluated for a class p, is given as follows:

Function Block	Number of RJDs Required
SKB (Standard kin. block)	p

We have to vary p=1 to n, in order to take all possible classes p into consideration. Doing this, we find the total number of RJDs needed for a robot with n d.o.f.

$$N_{\text{2nd variation}} = n(n+1)/2$$

The total number of RJDs of the 3rd variation Fig.5.3.4 is

$$N_{\text{3rd variation}} = n$$

8.3.) Computing costs of SMP method

A rough estimation finds that the computation of

$$D_{ij} = \sum_{p=\max(i,j)}^{n} \text{Trace}\{\mathbf{U}_{pj}.\mathbf{J}.\mathbf{U}^{T}_{pi}\}$$

requires maximum 84 multiplications and (64+n) additions for given i and j.

The computation of

$$D_{ijk} = \sum_{\max(i,j,k)}^{n} \text{Trace}\{\mathbf{U}_{pjk}.\mathbf{J}.\mathbf{U}^{T}_{pjk}\}$$

claims the same computation costs for given i,j and k.

Similarly, $D_i = \sum_{p=i}^{n} -m_p \mathbf{g}^T \mathbf{U}_{pi} \mathbf{r}_p$

claims less than 13 multiplications and (10+n) additions. Following an investigation by Lee[18], where the number of D_i, D_{ij} and D_{ijk} to be computed has been found as shown below

Term	Number of terms to be comuted
D_i	n
D_{ij}	1/2n (n+1)
D_{ijk}	1/3n(n^2-1)

The aggregate number of multiplications are then:

$$\frac{84}{3} n^3 + 42n^2 + 24n$$

The aggregate number of additions are:

$$\frac{1}{3} n^4 + \frac{131}{6} n^3 + \frac{199}{6} n^2 + \frac{62}{3} n$$

These numbers are computating costs required in the calculation of the terms D_i, D_{ij} and D_{ijk}.

In arriving at the results for the F_i for i= 1,2,...,n, the following remaining costs must be taken into account

$2n^3 + n^2$ multiplications

$n^3 + n^2$ additions

so that in total, we have

$\frac{1}{3} n^4 + \frac{137}{6} n^3 + \frac{205}{6} n^2 + \frac{62}{3} n$ additions and

$30n^3 + 43n^2 + 24n$ multiplications