Legnani G, Zappa B, Righettini P. <A Homogeneous Matrix Approach to 3D Kinematics and Dynamics. ***Part 2:applications***. Mechanisms and Machine Theory (the scientific journal of IFToMM. Pergamon Press U.K.) vol.31, n.5, pp.589-605, 1996

A HOMOGENEOUS MATRIX APPROACH

TO 3D KINEMATICS AND DYNAMICS

Part 2: applications to chains of rigid bodies and serial manipulators.

Giovanni LEGNANI♥, Paolo RIGHETTINI♠, Bruno ZAPPA♠, Federico CASOLO♠

♥Universita' di Brescia - Dip. di Ing. Meccanica, Via Branze 38, 25123 Brescia, Italy.

giovanni.LEGNANI @ ING.UNIBS.IT

♠ Politecnico di Milano - Dip. Trasporti e Movimentazione - Pz.a L. Da Vinci 32, 20133 Milano, Italy

RIGHETTINI @ MECH.POLIMI.IT

SUMMARY.

In this paper we present applications of the new approach to the kinematic and dynamic analysis of systems of rigid bodies presented in part 1. An extension of the method to the Lagrangian formulation of the dynamics of chains of rigid bodies is also presented. The kinematic and dynamic analysis is performed for a generic serial manipulator either in open and closed loop. Two numerical examples concerning an open loop and a closed loop are presented too. Two software packages based on our approach are also briefly introduced.

1. INTRODUCTION.

This part of the paper is devoted to the presentation of the application of the matrix approach described in part 1 of the paper to open and closed chains of rigid bodies; moreover the dynamics is extended to the Lagrangian formulation. We present an analytical example that shows how to write the kinematic and dynamic matrices of the well known Standford Arm. A second example concerning a closed loop system is also developed. A further example refers to a numeric solution for the direct kinematics and the inverse dynamics of any serial manipulator using two standard libraries written in C and in C++ language. These software packages outline the good correspondence between the theoretical approach of the problem and its implementation in simulation programs. The notation used in the following paragraphs is explained in part 1 of the paper which is assumed known to the reader. Since, in the study of chains of rigid bodies, subscripts often assume standard values, it is possible to use an abbreviated notation which makes the notation more compact. In other words[1] some subscripts can be omitted and we assume that $\mathbf{L}_{i-1,i} = \mathbf{L}_{i-1,i(i-1)}$, $\mathbf{W}_{i-1,i} = \mathbf{W}_{i-1,i(i-1)}$, $\mathbf{H}_{i-1,i} = \mathbf{H}_{i-1,i(i-1)}$.

2. CHAINS OF RIGID BODIES.

2.1. GENERAL CONSIDERATIONS.

In agreement with the Denavit and Hartenberg approach, we suppose that all the links of the system are coupled to each other by one degree of freedom lower pairs (prismatic, revolute or screw pairs). Then if the system has joints with more then one degree of freedom we should simulate it by introducing *dummy* bodies with one degree of freedom. In relation to the serial manipulator of fig. 1, relative position, speed and acceleration matrices between two contiguous bodies *h-1* and *h* can be expressed as a function of the joint variable q_h and its first and second time derivatives $\dot{q}_h$ $\ddot{q}_h$, where *h* is the joint between the bodies. For each link h it is possible to define the position matrix $M_{h-1,h}$ describing its position with respect to the previous link; this matrix depends on the h-th joint variable q_h. More over each velocity and acceleration matrix relative to contiguous bodies can be expressed as a function of matrix **L**, which in this case is a sort of *generalized velocities ratio matrix*

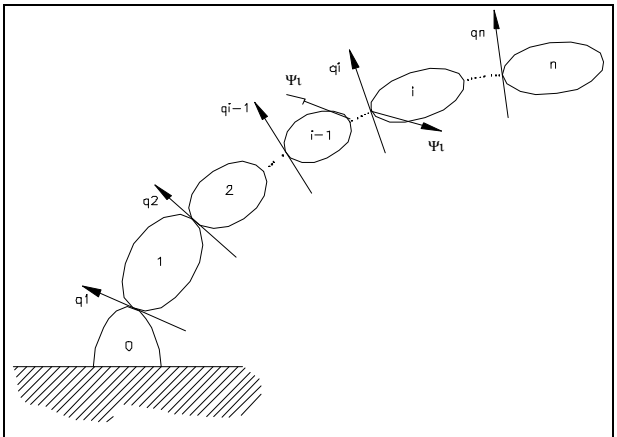

Fig. 1: scheme of a serial manipulator.

[1]See appendix B of part one.

$$\mathbf{W}_{h-1,h} = \mathbf{L}_{h-1,h}\dot{q}_h \ .$$

By remembering the relation between matrix **H** and **L** revealed in part1 we can write

$$\mathbf{H}_{h-1,h} = \mathbf{L}^2{}_{h-1,h}\dot{q}_h{}^2 + \mathbf{L}_{h-1,h}\ddot{q}_h$$

L , has similar properties (base reference transformation rule) to the correspondent matrix **W** and takes, for prismatic and screw pairs respectively, the two forms:

$${}^{p}\mathbf{L}_{i,j(k)} = \left|\begin{array}{ccc:c} & 0 & & \boldsymbol{u} \\ \hdashline 0 & 0 & 0 & 0 \end{array}\right| \qquad {}^{s}\mathbf{L}_{i,j(k)} = \left|\begin{array}{ccc:c} & \underline{\boldsymbol{u}} & & \boldsymbol{b} \\ \hdashline 0 & 0 & 0 & 0 \end{array}\right|$$

where $\boldsymbol{u}$ (unit vector) contains the direction cosines in *(k)* of the axis of the joint between the two bodies *i* and *j*, $\boldsymbol{b} = -\underline{\boldsymbol{u}}\,\boldsymbol{t} + p\boldsymbol{u}$, *p* is pitch of the pair, and $\boldsymbol{t}$ is the position in *(k)* of an arbitrary point of the axis. $\boldsymbol{u}$ and $\boldsymbol{t}$ can be immediately obtained from the appropriate position matrix. A revolute joint is a screw joint having a null pitch (*p*=0). If the reference frame of two subsequent bodies (*h-1* and *h*) are placed according to the Denavit and Hartenberg notation vector $\boldsymbol{u}$ assumes the simple form $\boldsymbol{u}^t = [0,0,1]^t$ and matrix **L** is simply:

$${}^{p}\mathbf{L}_{h-1,h} = \left|\begin{array}{ccc:c} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ \hdashline 0 & 0 & 0 & 0 \end{array}\right| \qquad {}^{s}\mathbf{L}_{h-1,h} = \left|\begin{array}{ccc:c} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & p \\ \hdashline 0 & 0 & 0 & 0 \end{array}\right| \qquad (1)$$

The position, speed and acceleration matrices of each body *i* can be found starting from the base of the manipulator and moving towards the end effector applying the motion composition rule

$$\mathbf{M}_{0,h} = \mathbf{M}_{0,h-1}\mathbf{M}_{h-1,h}$$

$$\mathbf{W}_{0,h} = \mathbf{W}_{0,h-1} + \mathbf{W}_{h-1,h(0)}$$

$$\mathbf{H}_{0,h} = \mathbf{H}_{0,h-1} + \mathbf{H}_{h-1,h(0)} + 2\mathbf{W}_{0,h-1}\mathbf{W}_{h-1,h(0)}$$

where $\mathbf{M}_{0,0} = \mathbf{[1]}$ (identity matrix) and $\mathbf{W}_{0,0} = \mathbf{H}_{0,0} = \mathbf{[0]}$ (null matrix). These equations can be generalized, for *i* ³ *2* as follows:

$$\mathbf{M}_{0,i} = \prod_{j=1}^{i} \mathbf{M}(q_j)_{j-1,j}$$

$$\mathbf{W}_{0,i} = \sum_{j=1}^{i} \mathbf{W}_{j-1,j(0)} = \sum_{j=1}^{i} \mathbf{L}_{j-1,j(0)}\dot{q}_j \qquad (2)$$

$$\mathbf{H}_{0,i} = \sum_{j=1}^{i} \mathbf{H}_{j-1,j(0)} + \sum_{r=2}^{i}\sum_{s=1}^{r-1} 2\mathbf{W}_{s-1,s(0)}\mathbf{W}_{r-1,r(0)} = \qquad (3)$$

$$= \sum_{j=1}^{i} \left[\left(\mathbf{L}_{j-1,j(0)}\dot{q}_j\right)^2 + \mathbf{L}_{j-1,j(0)}\ddot{q}_j\right] +$$

$$+ \sum_{r=2}^{i}\sum_{s=1}^{r-1} 2\mathbf{L}_{s-1,s(0)}\mathbf{L}_{r-1,r(0)}\dot{q}_s\dot{q}_r$$

The acceleration equation can be rewritten separating the effect of the joint velocities and accelerations

$$\mathbf{H}_{0,i} = \sum_{j=1}^{i} \left[\mathbf{L}_{j-1,j(0)}\ddot{q}_j\right] + \tilde{\mathbf{H}}$$

$$\tilde{\mathbf{H}} = \sum_{j=1}^{i} \left[\left(\mathbf{L}_{j-1,j(0)}\dot{q}_j\right)^2\right] + \qquad (3bis)$$

$$+ \sum_{r=2}^{i}\sum_{s=1}^{r-1} 2\mathbf{L}_{s-1,s(0)}\mathbf{L}_{r-1,r(0)}\dot{q}_s\dot{q}_r$$

In other words, $\tilde{\mathbf{H}}$ contains the Coriolis and the centrifugal terms

The angular velocity and acceleration of the end-effector can be extracted by the 3×3 submatrix of $W_{0,n}$ and $H_{0,n}$ (see equations (6) and (7) of part I) while its linear velocity and acceleration can be evaluated as

$$\dot{P}^* = W_{0,n}P^* \qquad \ddot{P}^* = H_{0,n}P^*$$

where P* is the last column of $M_{0,n}$.

2.2. CLOSED LOOP SYSTEMS.

If the system contains closed loops, three constraint matrix equations (for position, velocity and acceleration) can be written for each loop. These equations can be obtained by thinking of a loop as an open chain with the first and the last body coinciding ($n \equiv 0$):

$$\mathbf{M}_{0,n} = \mathbf{M}_{0,1}\mathbf{M}_{1,2}\ldots\ldots\mathbf{M}_{n-1,n} = \mathbf{M}_{0,0} = \mathbf{[1]} \qquad (4)$$

$$\mathbf{W}_{0,n} = \mathbf{W}_{0,1} + \mathbf{W}_{1,2(0)} + \ldots\ldots\ldots + \mathbf{W}_{n-1,n(0)} = \mathbf{W}_{0,0} = \mathbf{[0]} \qquad (5)$$

$$\mathbf{H}_{0,n} = \mathbf{H}_{0,1} + \mathbf{H}_{1,2(0)} + 2\mathbf{W}_{0,1}\mathbf{W}_{1,2(0)} + \ldots\ldots = \mathbf{H}_{0,0} = \mathbf{[0]} \qquad (6)$$

where **[1]** is the identity matrix, and **[0]** the null one and all **W** and **H** matrices must be evaluated in the same reference.

Due to the particular structure of the matrices involved (only 6 elements of each are independent), each of the equations (4,5,6) is equivalent to a scalar system of 6 equations with the *n* unknowns ($q_i, \dot{q}_i, \ddot{q}_i$). If the loop contains less than 6 joints and in special cases, some of the equations can depend on each other.

The position equation is non linear while the others are linear in $\dot{q}_i$ or $\ddot{q}_i$. Equations (5, 6) can easily be built using **L** matrices, for example the former should be writen in frame (0) as

$$\sum_{i=1}^{n} \mathbf{L}_{i-1,i(0)}\dot{q}_i = \mathbf{[0]}. \qquad (7)$$

Remembering that the independent components of matrix $\mathbf{L}_i$ are

$$\mathbf{L}_i = \left|\begin{array}{ccc:c} 0 & -u_{z_i} & u_{y_i} & b_{x_i} \\ u_{z_i} & 0 & -u_{x_i} & b_{y_i} \\ -u_{y_i} & u_{x_i} & 0 & b_{z_i} \\ \hdashline 0 & 0 & 0 & 0 \end{array}\right|$$

the linear system $\mathbf{A}\dot{\mathbf{q}} = \mathbf{[0]}$ obtained from Eq. 7 is

$$\mathbf{A}\dot{\mathbf{q}}=\begin{vmatrix} u_{x_1} & \dots & u_{x_i} & \dots & u_{x_n} \\ u_{y_1} & & u_{y_i} & & u_{y_n} \\ u_{z_1} & & u_{z_i} & & u_{z_n} \\ b_{x_1} & & b_{x_i} & & b_{x_n} \\ b_{y_1} & & b_{y_i} & & b_{y_n} \\ b_{z_1} & & b_{z_i} & & b_{z_n} \end{vmatrix} \begin{vmatrix} \dot{q}_1 \\ \dots \\ \dot{q}_i \\ \dots \\ \dot{q}_n \end{vmatrix} = \begin{vmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{vmatrix}$$

We have six equations for each loop and a number of unknown equal to the number of the joints. The degree of mobility of the loop (i.e. d.o.f.) is the difference between the number of the joints and the independent equations (the rank of matrix **A**).

The acceleration equation of each loop can be built by eq. 6 using **L** matrices (eq. 3). It is represented by the system $\mathbf{A}\ddot{\mathbf{q}}=\mathbf{b}$ where the coefficient matrix **A** is identical to the coefficient matrix **A** of the velocity equation system. In fact this equation system can be also obtained deriving the velocity equation system with respect to time $\mathbf{A}\dot{\mathbf{q}}=0$ obtaining $\mathbf{A}\ddot{\mathbf{q}}=-\dot{\mathbf{A}}\dot{\mathbf{q}}$, indeed $\mathbf{b}=-\dot{\mathbf{A}}\dot{\mathbf{q}}$.

Moreover the coefficient matrix **A** obtained in this way from Eq. 5, 6 is identical; **b** contains the Coriolis and the centrifugal terms extracted from $\tilde{\mathbf{H}}$ of eq. (3bis). In fact eq. (6) can be rewritten as $\sum \mathbf{L}\ddot{q}=-\tilde{\mathbf{H}}$ Finally the position system must be solved before the others, while the velocities system must be solved before that of the accelerations.

2.3. NEWTON-EULER DYNAMICS.

The dynamics of an open chain of rigid bodies can be developed starting from the above equation and applying the usual principles (virtual works, momentum and angular momentum conservation, etc.). For example the joint action $\mathbf{Y}_i$ between the bodies i-1 and i of an open chain (see fig. 1) can be simply obtained by a dynamic equilibrium. In other words $\mathbf{Y}_i$ is just the sum of all the actions (including weight and inertia) applied to body i,i+1,..n:

$$\Psi_{i(0)}=\sum_{j=i}^{n}\left[skew\left(\mathbf{H}_{0,j}\mathbf{J}_{j(0)}\right)+\hat{\Phi}_{j(0)}+\Phi_{j(0)}\right]$$

where $\mathbf{H}_{0,j}$ is the absolute acceleration of the body j, $\mathbf{J}_{j(0)}$ is its pseudo-inertial tensor referred to an inertial frame (0), $\hat{\Phi}_{j(0)}$ is the weight action matrix, and $\mathbf{F}_{j(0)}$ is the resultant external action applied to the body, and the skew operator is defined in part one, paragaph 4.6. The joint actions can be also evaluated iteratively starting from joint n using the recursive formula

$$\mathbf{Y}_{i-1(0)}=\mathbf{Y}_{i(0)}+skew(\mathbf{H}_{0,i}\mathbf{J}_{i(0)})+skew(\mathbf{H}_{g(0)}\mathbf{J}_{i(0)})+\Phi_{i(0)}$$

3. LAGRANGIAN DYNAMICS.

The matrix approach introduced allows the writing of the dynamics equation of a system of rigid bodies following Lagrange's method by means of the general equation

$$\frac{d}{dt}\frac{\P L}{\P\dot{q}_i}-\frac{\P L}{\P q_i}=Q_i$$

where the Lagrange quantity L is calculated as the difference between the kinetic and potential energy ($L = t\text{-}v$) and Q_i is the generalized components of force along q_i.

The kinetic energy of a body *h* is expressed as a function of velocity and inertia matrix by means of the trace operator (see note 3 of part one)

$$t_h=\frac{1}{2}Trace(\mathbf{W}_{0,h}\mathbf{J}_{h(0)}\mathbf{W}_{0,h}^{t}) \tag{8}$$

while the potential energy due to the gravitational effect is expressed by

$$v_{ph}=Trace(-\mathbf{H}_{g(0)}\mathbf{J}_{h(0)}) \tag{9}$$

We will develop Lagrange's formulation for the chain of rigid bodies dividing the effects due to the kinetic and potential energy. For the solution of the former we must obtain the derivatives of matrices used in relation (8) with respect to time *t*, to the coordinate q_m and $\dot{q}_m$. The basic step is to derive the position matrix from which we can obtain the other derivatives. Starting from $\mathsf{P}_0=\mathbf{M}_{0,1}\mathsf{P}_1$ the time derivative yields $\frac{d\mathsf{P}_0}{dt}=\frac{d\mathbf{M}_{0,1}}{dt}\mathsf{P}_1$ where P_1 is constant with respect to *t* (i.e. P_1 is embedded on frame 1). Recalling the relation between the velocity of the point of the body and matrix **W**, and the transformation of frame reference of the point we can write $\frac{d\mathsf{P}_0}{dt}=\mathbf{W}_{0,1}\mathsf{P}_0=\frac{d\mathbf{M}_{0,1}}{dt}\mathbf{M}_{1,0}\mathsf{P}_0$ so that $\frac{d\mathbf{M}_{0,1}}{dt}=\mathbf{W}_{0,1}\mathbf{M}_{0,1}$. The derivative of the position matrix with respect to q_m can be found remembering that matrix **L** represents an infinitesimal transformation and each column of matrix **M** is a particular point (points at infinity of the axis and the origin) of the frame, so we can write

$$\frac{d\mathbf{M}_{i,j}}{dq_m}=\begin{cases}\mathbf{L}_{m-1,m}\mathbf{M}_{i,j} & \text{if } i<m\le j\\ \mathbf{[0]} & \text{otherwise}\end{cases}$$

On these bases we can prove (see appendix) :

$$\frac{d\mathbf{W}_{0,h}}{dt}=\sum_{i=1}^{h}(\mathbf{W}_{0,i-1}\mathbf{W}_{i-1,i(0)}-\mathbf{W}_{i-1,i(0)}\mathbf{W}_{0,i-1})+\sum_{i=1}^{h}\mathbf{L}_{i-1,i(0)}\ddot{q}_i$$

$$\frac{d\mathbf{W}_{0,h}}{dq_m}=\begin{cases}\mathbf{[0]} & \text{if } m>h\\ \mathbf{L}_{m-1,m(0)}\mathbf{W}_{m,h(0)}-\mathbf{W}_{m,h(0)}\mathbf{L}_{m-1,m(0)} & \text{if } m\le h\end{cases}$$

$$\frac{d\mathbf{J}_{h(0)}}{dt}=\mathbf{W}_{0,h}\mathbf{J}_{h(0)}+\mathbf{J}_{h(0)}\mathbf{W}_{0,h}^{t}$$

$$\frac{d\mathbf{J}_{h(0)}}{dq_m}=\begin{cases}\mathbf{[0]} & \text{if } m>h\\ \mathbf{L}_{m-1,m(0)}\mathbf{J}_{h(0)}+\mathbf{J}_{h(0)}\mathbf{L}_{m-1,m(0)}^{t} & \text{if } m\le h\end{cases}$$

so that

$$\frac{d}{dt}\frac{\P t_h}{\P\dot{q}_m}-\frac{\P t_h}{\P q_m}=Trace[(\tilde{\mathbf{H}}_{0,h}+\sum_{i=1}^{h}\mathbf{L}_{i(0)}\ddot{q}_i)\mathbf{J}_{h(0)}\mathbf{L}_{m(0)}^{t}]$$

where $\tilde{\mathbf{H}}$ is the acceleration matrix calculated for $\ddot{q} = 0$, which depends on position and velocity and represents the Coriolis and centrifugal effects:

$$\tilde{\mathbf{H}}_{0,h} = \sum_{j=1}^{h} \mathbf{L}^2{}_{j-1,j} \dot{q}_j{}^2 + \sum_{r=2}^{h} \sum_{s=1}^{r-1} 2\mathbf{L}_{s-1,s}\mathbf{L}_{r-1,r}\dot{q}_r\dot{q}_s =$$

$$= \sum_{i=1}^{h} (\mathbf{W}_{0,i-1}\mathbf{W}_{i(0)} - \mathbf{W}_{i(0)}\mathbf{W}_{0,i-1}) + \mathbf{W}_{0,h}{}^2$$

The solution of the last is

$$\frac{d}{dt}\frac{\P v}{\P \dot{q}_m} - \frac{\P v}{\P q_m} = Trace[-\mathbf{H}_{g(0)}\mathbf{J}_{h(0)}\mathbf{L}^t_{m(0)}]$$

The complete dynamic equation is given by the sum of the two preceding relations.

For the determination of the generalised components of force from the action matrix **F** we introduce a pseudo scalar product $\otimes$ so that

$$Q_{m(0)} = (\sum_{i=1}^{N} \mathbf{F}_{i(0)}) \otimes \mathbf{L}_{m(0)} . \qquad (10)$$

The pseudo scalar product between two 4x4 matrices is defined as follows

$$\mathbf{A} \otimes \mathbf{B} = \mathbf{A}[3,2] * \mathbf{B}[3,2] + \mathbf{A}[1,3] * \mathbf{B}[1,3] + \mathbf{A}[2,1] * \mathbf{B}[2,1] +$$
$$+ \mathbf{A}[1,4] * \mathbf{B}[1,4] + \mathbf{A}[2,4] * \mathbf{B}[2,4] + \mathbf{A}[3,4] * \mathbf{B}[3,4]$$

where $\mathbf{A}[i,j]$ and $\mathbf{B}[i,j]$ are the elements of position *i,j* of matrices **A** and **B**. In the case of the eq. (10) it yields:

$$Q_m = c_x u_x + c_y u_y + ... + f_z b_z$$

that is, the components of the resulting action $\sum \mathbf{F}_i$ onto the displacement permitted by the joint *m* described by matrix $\mathbf{L}_m$.

For open chain systems the summation in eq. (10) starts from *m* instead of 1 because only the action applied to the link with label greater then *m* work for an infinitesimal variation of q_m.

It is possible to show that the dynamic equation of the system can be written in the following matrix form

$$\mathbf{M}\ddot{q} + \mathbf{C}(q, \dot{q}) = \mathbf{F}(t)$$

where $\mathbf{M}$ is the mass matrix, **C** a vector holding the weight, centrifugal and Coriolis effects, **F** a vector containing the components on the joint coordinates of the forces and torques applied to the manipulator (including the actuators actions). **F** depend on the time. The elements of the mass matrix (which is symmetric and positive defined) are given by the relation

$$\mathbf{M}[i,m] = Trace[\sum_{h=max(i,m)}^{N} \mathbf{L}_{i(0)}\mathbf{J}_{h(0)}\mathbf{L}^t_{m(0)}]$$

while the elements of vector **C** and **F** are

$$\mathbf{C}[m] = Trace\left[\sum_{h=m}^{N} (\tilde{\mathbf{H}}_{\mathbf{0,h}} - \mathbf{H}_{\mathbf{g(0)}})\mathbf{J}_{h(0)}\mathbf{L}^t_{m(0)}\right] - L_{m(0)} \otimes \sum \Phi_{m(0)}$$

$\mathbf{F}[i]$ = action (force or torque) on i-th joint.

A comparison between this methodology with those presented in [40] allows a better understanding of the meaning of coefficient D_i, D_{ij}, D_{ijk} there presented as the result of mathematical derivation and whose meaning can be explained in term of **W**, **H**, **L** and **J** matrices.

4. OPEN LOOP EXAMPLE.

In order to show the practical use of our methodology, let us consider the problem of writing the kinematic and dynamic equations of the STANDFORD ARM, having six degrees of freedom with five revolute joints and one prismatic, whose initial position is drawn in fig. 2. The kinematics section describes how you can write the position matrices, the relative and absolute velocity and acceleration matrices, while the dynamics section describes how to write the inertia matrix and the dynamic equilibrium. We describe the joint space coordinates by the vector $\mathbf{Q} = \left[J_1, J_2, d_3, J_4, J_5, J_6\right]^t$.

4.1. KINEMATICS.

In this example the reference frames of the links are placed according to Denavit and Hartenberg. Then the matrices that describe the relative position matrices of the links are

$$\mathbf{M}_{0,1} = \left|\begin{array}{ccc|c} C_1 & 0 & -S_1 & 0 \\ S_1 & 0 & C_1 & 0 \\ 0 & -1 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right| \quad \mathbf{M}_{1,2} = \left|\begin{array}{ccc|c} C_2 & 0 & S_2 & 0 \\ S_2 & 0 & -C_2 & 0 \\ 0 & 1 & 0 & d_2 \\ \hline 0 & 0 & 0 & 1 \end{array}\right|$$

$$\mathbf{M}_{2,3} = \left|\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ \hline 0 & 0 & 0 & 1 \end{array}\right| \quad \mathbf{M}_{3,4} = \left|\begin{array}{ccc|c} C_4 & 0 & -S_4 & 0 \\ S_4 & 0 & C_4 & 0 \\ 0 & -1 & 0 & d_4 \\ \hline 0 & 0 & 0 & 1 \end{array}\right|$$

$$\mathbf{M}_{4,5} = \left|\begin{array}{ccc|c} C_5 & 0 & S_5 & 0 \\ S_5 & 0 & -C_5 & 0 \\ 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right| \quad \mathbf{M}_{5,6} = \left|\begin{array}{ccc|c} C_6 & -S_6 & 0 & 0 \\ S_6 & C_6 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right|$$

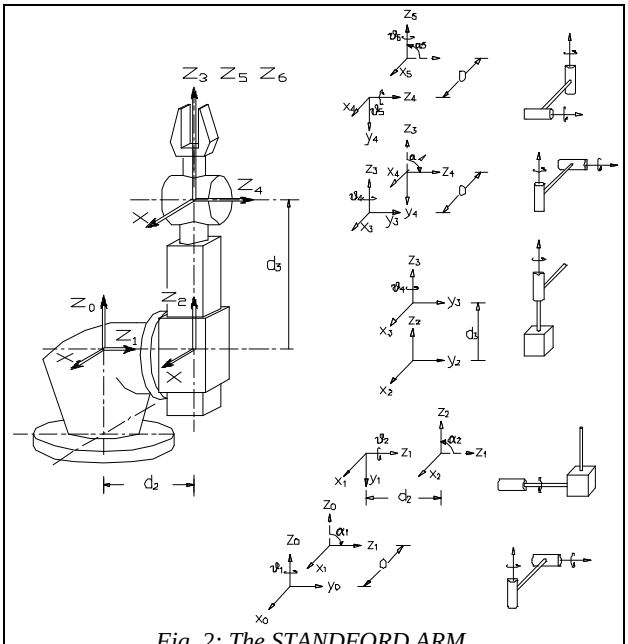

Fig. 2: The STANDFORD ARM.

where $C_i = cos(J_i)$ and $S_i = sin(J_i)$. The absolute position matrix of any link of the robot can be evaluated as usual as

$$\mathbf{M}_{0,i} = \mathbf{M}_{0,1}\mathbf{M}_{1,2}....\mathbf{M}_{i-1,i}$$

Matrices **L** in relative frame have the form ${}^{p}\mathbf{L}$ for prismatic joints and ${}^{r}\mathbf{L}$ for revolute joints as shown in eq.(1). Remembering the relation between matrices **W**, **H** and **L** the relative velocity and acceleration matrices between contiguous links with revolute pairs, joints *i*=1,2,4,5,6 in our example, are

$$\mathbf{W}_{i-1,i(i-1)} = \left|\begin{array}{ccc:c} 0 & -\dot{q}_i & 0 & 0 \\ \dot{q}_i & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \hdashline 0 & 0 & 0 & 0 \end{array}\right| \quad \mathbf{H}_{i-1,i(i-1)} = \left|\begin{array}{ccc:c} -\dot{q}_i{}^2 & -\ddot{q}_i & 0 & 0 \\ \ddot{q}_i & -\dot{q}_i{}^2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \hdashline 0 & 0 & 0 & 0 \end{array}\right|$$

and for the prismatic pair, joint 3 in Fig 2, are

$$\mathbf{W}_{2,3(2)} = \left|\begin{array}{ccc:c} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \dot{q}_3 \\ \hdashline 0 & 0 & 0 & 0 \end{array}\right| \qquad \mathbf{H}_{2,3(2)} = \left|\begin{array}{ccc:c} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \ddot{q}_3 \\ \hdashline 0 & 0 & 0 & 0 \end{array}\right|$$

where q_i is the *i-th* free coordinate of the joint and $\dot{q}, \ddot{q}$ its time derivatives.

To obtain the absolute velocity and acceleration matrices of the links we use the velocity composition rule and the Coriolis theorem,

$$\mathbf{W}_{0,j(0)} = \mathbf{W}_{0,j-1(0)} + \mathbf{W}_{j-1,j(0)} \qquad (11)$$

$$\mathbf{H}_{0,j(0)} = \mathbf{H}_{0,j-1(0)} + 2\mathbf{W}_{0,j-1(0)}\mathbf{W}_{j-1,j(0)} + \mathbf{H}_{j-1,j(0)}$$

in which the relative velocity and acceleration matrices are referred to the absolute frame (0). We can easily calculate these matrices by means of matrices **L**

$$\mathbf{W}_{i-1,i(0)} = \mathbf{L}_{i-1,i(0)}\dot{q}_i$$

$$\mathbf{H}_{i-1,i(0)} = \mathbf{L}_{i-1,i(0)}\ddot{q}_i + \mathbf{L}^2_{i-1,i(0)}\dot{q}_i{}^2$$

Matrices **L** can be referred in the absolute frame (0) using the following relation

$$\mathbf{L}_{i-1,i(0)} = \mathbf{M}_{0,i-1}\mathbf{L}_{i-1,i}\mathbf{M}^{-1}_{0,i-1} \ .$$

For link 1 $\mathbf{M}_{0,0} = [\mathbf{I}]$ (the identity matrix), so the absolute velocity and acceleration matrices are the same as the relative ones, $\mathbf{W}_{0,1(0)} = \mathbf{W}_{0,1}$ and $\mathbf{H}_{0,1(0)} = \mathbf{H}_{0,1}$.

The relative velocity and acceleration matrices between links 1 and 2 can be referred to the absolute frame by the relation

$$\mathbf{W}_{1,2(0)} = \mathbf{M}_{0,1}\mathbf{W}_{1,2}\mathbf{M}^{-1}_{0,1} = \mathbf{L}_{1,2(0)}\dot{q}_2$$

$$\mathbf{L}_{1,2(0)} = \mathbf{M}_{0,1}\mathbf{L}_{1,2}\mathbf{M}^{-1}_{0,1} = \left|\begin{array}{ccc:c} 0 & 0 & C_1 & 0 \\ 0 & 0 & S_1 & 0 \\ -C_1 & -S_1 & 0 & 0 \\ \hdashline 0 & 0 & 0 & 0 \end{array}\right|$$

$$\mathbf{H}_{1,2(0)} = \mathbf{L}_{1,2(0)}\ddot{q}_2 + \mathbf{L}_{1,2(0)}{}^2\dot{q}_2{}^2 =$$

$$= \left|\begin{array}{ccc:c} 0 & 0 & C_1 & 0 \\ 0 & 0 & S_1 & 0 \\ -C_1 & -S_1 & 0 & 0 \\ \hdashline 0 & 0 & 0 & 0 \end{array}\right|\ddot{q}_2 + \left|\begin{array}{ccc:c} -C_1{}^2 & -C_1S_1 & 0 & 0 \\ -C_1S_1 & -S_1{}^2 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ \hdashline 0 & 0 & 0 & 0 \end{array}\right|\dot{q}_2^2$$

The evaluation of the velocity and acceleration matrices of the other links (3, 4, 5 and 6) is executed in the same way. As a further explanation, let us show matrices $\mathbf{L}_{2,3(0)}, \mathbf{W}_{2,3(0)}, \mathbf{H}_{2,3(0)}$ which are

$$\mathbf{L}_{2,3(0)} = \left|\begin{array}{ccc:c} 0 & 0 & 0 & C_1S_1 \\ 0 & 0 & 0 & S_1S_2 \\ 0 & 0 & 0 & C_2 \\ \hdashline 0 & 0 & 0 & 0 \end{array}\right|$$

$$\mathbf{W}_{2,3(0)} = \mathbf{L}_{2,3(0)}\dot{q}_3 \qquad \mathbf{H}_{2,3(0)} = \mathbf{L}_{2,3(0)}\ddot{q}_3$$

The formula for $\mathbf{H}_{2,3(0)}$ is very simple because the 3^{rd} joint is prismatic and so $\mathbf{L}^2_{2,3(0)}$ reduces to the null matrix.

The absolute velocity and acceleration matrix of the other links of the manipulator can be easily obtained by using formulas (11) recursively.

4.2. DYNAMICS.

Remembering the first part of the paper, the inertial action matrix $\tilde{\Phi}_{i(0)}$ of the link i in absolute frame is found using the dynamic equilibrium equation, that reads

$$\tilde{\Phi}_{i(0)} = -Skew(\mathbf{H}_{0,i(0)}\mathbf{J}_{i(0)})$$

where $\mathbf{H}_{0,i(0)}$ is the absolute acceleration matrix (calculated in the preceding section), and $\mathbf{J}_{i(0)}$ is the absolute inertia matrix which is obtained from the inertia matrix of the link i in the local frame by the transformation $\mathbf{J}_{i(0)} = \mathbf{M}_{0,i}\mathbf{J}_{i(i)}\mathbf{M}_{0,i}{}^{t}$. It is important to note that the inertia of the link is constant if evaluated with respect to its local frame but it varies if expressed in the global reference frame. In other words $\mathbf{J}_{i(i)}$ in constant while $\mathbf{J}_{i(0)}$ depend on the robot motion. Let us consider the problem of building the inertia matrix of any link i in the local frame.

$$\mathbf{J}_{i(i)} = \left|\begin{array}{ccc:c} I_{xx} & I_{xy} & I_{xz} & mx_g \\ I_{yx} & I_{yy} & I_{yz} & my_g \\ I_{zx} & I_{zy} & I_{zz} & mz_g \\ \hdashline mx_g & my_g & mz_g & m \end{array}\right|$$

This matrix can immediately be built knowing the mass *m*, the "*usual*" inertia moments J_x, J_y, J_z, J_{xy}, J_{yz}, J_{xz} referred to the center of mass and the position x_g, y_g, z_g of the center of mass of the link.

The weight action matrix $\hat{\Phi}$ may be evaluated by means of the gravity acceleration matrix, inertia matrix and the *Skew* operator by the formula

$$\hat{\Phi}_{i(0)} = Skew(\mathbf{H}_{g(0)}\mathbf{J}_{i(0)}).$$

The total action matrix $\mathbf{F}_{0,i}$ on each link can be found starting from the end effector, the only link on which the external force acts (known), calculating the inertial action

matrix, summing the total action matrix of the successive link and summing the weight action matrix

$$\Phi_i = \tilde{\Phi}_i + \Phi_{i+1} + \hat{\Phi}_i$$

It's very hard to write the dynamic relation of the manipulator in fig. 2 in symbolic form, so this part is presented only in numeric form in section 6 which outlines the easy implementation of the presented formulation.

5. CLOSED LOOP EXAMPLE.

In this section we present an application of the methodology applied to the closed loop system in fig. 3, writing the kinematic and dynamic equations. We describe the joint space coordinate by the vector $\mathbf{Q} = [a_1, a_2, a_3, a]$. Using the local frame of the fourth link shown in fig. 3 and letting $c_i = cos(a_i)$ and $s_i = sin(a_i)$ the relative position matrices are:

$$\mathbf{M}_{0,1} = \left|\begin{array}{ccc:c} c_1 & -s_1 & 0 & l_1c_1 \\ s_1 & c_1 & 0 & l_1s_1 \\ 0 & 0 & 1 & 0 \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right| \quad \mathbf{M}_{1,2} = \left|\begin{array}{ccc:c} c_2 & -s_2 & 0 & l_2c_2 \\ s_2 & c_2 & 0 & l_2s_2 \\ 0 & 0 & 1 & 0 \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right|$$

$$\mathbf{M}_{2,3} = \left|\begin{array}{ccc:c} c_3 & -s_3 & 0 & 0 \\ s_3 & c_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right| \quad \mathbf{M}_{3,4} = \left|\begin{array}{ccc:c} 1 & 0 & 0 & -a \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right|$$

Applying the position equation 4 yields:

$$\left|\begin{array}{ccc:c} cos(\alpha_1+\alpha_2+\alpha_3) & -sin(\alpha_1+\alpha_2+\alpha_3) & 0 & l_1 cos(\alpha_1)+l_2 cos(\alpha_1+\alpha_2)-a \\ sin(\alpha_1+\alpha_2+\alpha_3) & cos(\alpha_1+\alpha_2+\alpha_3) & 0 & l_1 sin(\alpha_1)+l_2 sin(\alpha_1+\alpha_2) \\ 0 & 0 & 1 & 0 \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right| =$$

$$= \left|\begin{array}{ccc:c} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right| .$$

This matrix system is equivalent to

$$\begin{cases} \cos(a_1 + a_2 + a_3) = 1 \\ sin(a_1 + a_2 + a_3) = 0 \\ l_1 \cos(a_1) + l_2 \cos(a_1 + a_2) - a = 0 \\ l_1 sin(a_1) + l_2 sin(a_1 + a_2) = 0 \end{cases}$$

The first and the second equations are equivalent to the relation

$$a_1 + a_2 + a_3 = 2kp$$

so that the system is described by 3 independent equations with 4 unknowns (a_1, a_2, a_3, a), indeed the system has just one degree of freedom.

For the analysis of velocity and acceleration of the system we build the **L** matrices in local frame:

$$\mathbf{L}_{0,1} = \mathbf{L}_{1,2} = \mathbf{L}_{2,3} = \left|\begin{array}{ccc:c} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \hdashline 0 & 0 & 0 & 0 \end{array}\right| \quad \mathbf{L}_{3,0} = \left|\begin{array}{ccc:c} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \hdashline 0 & 0 & 0 & 0 \end{array}\right|$$

Matrices **L** can be referred in the absolute frame (0) using the following relation

$$\mathbf{L}_{i-1,i(0)} = \mathbf{M}_{0,i-1}\mathbf{L}_{i-1,i}\mathbf{M}_{0,i-1}^{-1}$$

Appling the equation 7 to this closed loop system we have

$$\mathbf{L}_{0,1(0)}\dot{a}_1 + \mathbf{L}_{1,2(0)}\dot{a}_2 + \mathbf{L}_{2,3(0)}\dot{a}_3 + \mathbf{L}_{3,0(0)}\dot{a} = \mathbf{[0]}$$

from which, remembering that **L** matrices have six independent elements, the system $\mathbf{A}\dot{q} = \mathbf{[0]}$ of paragraph 2.2 reduces to

$$\left|\begin{array}{cccc:c} 0 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 & \dot{a}_1 \\ 1 & 1 & 1 & 0 & \dot{a}_2 \\ 0 & l_1 sin(a_1) & l_1 sin(a_1) + l_2 sin(a_1 + a_2) & 1 & \dot{a}_3 \\ 0 & -l_1 \cos(a_1) & -l_1 \cos(a_1) - l_2 \cos(a_1 + a_2) & 0 & \dot{a} \\ 0 & 0 & 0 & 0 & \end{array}\right| = \left|\begin{array}{c} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{array}\right|$$

is equivalent to 3 equations with 4 unknowns. Note that the position system equations outline the geometrical relations (see fig. 3)

$$l_1 sin(a_1) + l_2 sin(a_1 + a_2) = 0$$

$$l_1 cos(a_1) + l_2 cos(a_1 + a_2) = a$$

that can be used to simplify matrix **A**.

The accelerations of the bodies of the mechanism can be obtained by equations 6 and 3. For the mechanism of fig. 3 we have

$$\mathbf{L}_{0,1(0)}\ddot{a}_1 + \mathbf{L}_{1,2(0)}\ddot{a}_2 + \mathbf{L}_{2,3(0)}\ddot{a}_3 + \mathbf{L}_{3,0(0)}\ddot{a} + \sum_{i=1}^{4}(\mathbf{L}_{i-1,i(0)}\dot{q}_i)^2 + 2\sum_{r=2}^{4}\sum_{s=1}^{r-1}\mathbf{L}_{s-1,s(0)}\mathbf{L}_{r-1,r(0)}\dot{q}_s\dot{q}_r = \mathbf{[0]}$$

and the acceleration equations system, $\mathbf{A}\ddot{\mathbf{q}} = \mathbf{b}$ becames

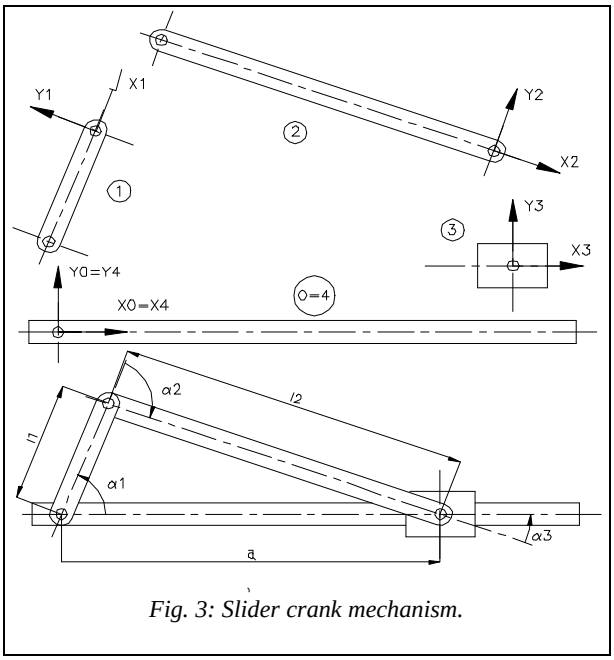

Fig. 3: Slider crank mechanism.

$$\begin{vmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & l_1 sin(a_1) & l_1 sin(a_1)+l_2 sin(a_1+a_2) & 1 \\ 0 & -l_1\cos(a_1) & -l_1\cos(a_1)-l_2\cos(a_1+a_2) & 0 \\ 0 & 0 & 0 & 0 \end{vmatrix} \begin{vmatrix} \ddot{a}_1 \\ \ddot{a}_2 \\ \ddot{a}_3 \\ \ddot{a} \end{vmatrix} =$$

$$= \begin{vmatrix} 0 \\ 0 \\ 0 \\ -2l_1\cos(a_1)\dot{a}_1\dot{a}_2 - 2a(\dot{a}_1+\dot{a}_2)\dot{a}_3 \\ -2l_1 sin(a_1)\dot{a}_1\dot{a}_2 - 2\dot{a}(\dot{a}_1+\dot{a}_2+\dot{a}_3) \\ 0 \end{vmatrix}$$

where the matrix coefficient is equal to the matrix coefficient of the velocity equations system, and the first time derivatives of the free coordinates in the right side are calculated by the velocity equations system.

6. NUMERICAL APPLICATION.

The proposed methodology appears to be very suitable for computer applications because the operations required for coding a program can be defined very easily. For this reason a software library called SPACE_LIB has been realised [21]. A complete sample program (fig.4) for the direct kinematics and the inverse dynamics of any serial manipulator highlights the power of the methodology.

The program is presented just with the aim of showing how to use the presented methodology. The simulation programs can be written using few standard statements as well a few calls to standard general purpose functions.

The program, whose source code consists of less then 100 lines of listing, is composed of four parts.

The first part is devoted to the declaration of the variables and to the input data phase (kinematic and dynamic characteristics of the robot in hand).

The second part is the *Kinematics* part, consisting of a simple loop performing the following iterative operations:

- reads the joints motion (step 2), builds relative position matrices **A** (step 3) and the relative velocity and acceleration matrices by means **L** matrix (step 4)
- evaluates the absolute position **M0** of each link (according to D.&H. method) using the formula $\mathbf{M0}_{0,i} = \mathbf{M0}_{0,i-1}\mathbf{A}_{i-1,i}$ (step 5)
- transforms the relative velocity and acceleration matrices from local to the absolute frame (0) (step 6-7)

$$\mathbf{W}_{i-1,i(0)} = \mathbf{M0}_{0,i}\mathbf{W}_{i-1,i}\mathbf{M0}_{0,i}^{-1}$$

$$\mathbf{H}_{i-1,i(0)} = \mathbf{M0}_{0,i}\mathbf{H}_{i-1,i}\mathbf{M0}_{0,i}^{-1}$$

- evaluates the absolute speed of each link by summing the drag and the relative speed of each link (step 8)

$$\mathbf{W}_{0,i} = \mathbf{W}_{0,i-1} + \mathbf{W}_{i-1,i(0)}$$

- evaluates of the absolute acceleration of each link by means the Coriolis' theorem (step 9)

$$\mathbf{H}_{0,i} = \mathbf{H}_{0,i-1} + \mathbf{H}_{i-1,i(0)} + 2\mathbf{W}_{0,i-1}\mathbf{W}_{i-1,i(0)}$$

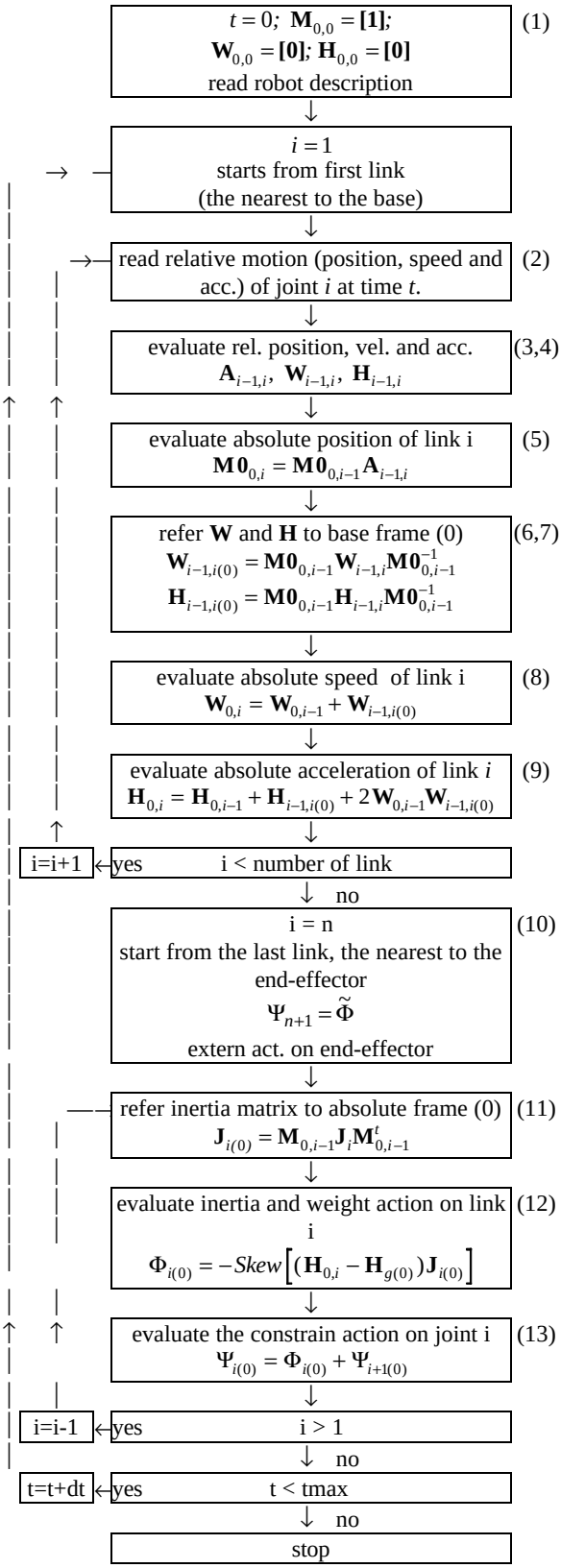

Fig. 4: Flow chart of a program for the direct kinematics and inverse dynamics of a serial manipulator. Numbers in parentheses refers to the program list of fig.6.

The third part is the *Dynamic* analysis, which consists of the evaluation of the inertial actions applied to each link and of the joint reactions, simply by summing all the actions applied to the links which follow the considered joint.

The fourth part is devoted to the output of the calculated matrices.

Another software library, CHAIN++ [24], has been developed for writing simulation programs in the C++ language. This *engine* defines several mathematical classes that describe the matrices presented in this paper and operations on them. Moreover it defines the class *LinkObject*, that encapsulates all the matrices needed to define the data structure describing a rigid body and has the methods working on it, and the class *ChainObject* a collection of objects *LinkObject*. In fig. 5 we present an excerpt of a C++ program, performing the kinematic loop previously described, that outlines the representation of the matrix operation and the compact code obtained.

To people not familiar to C++ language we can say that this language offers the possibility to define *new* types of variables and the modalities to operate on them. For example we have defined a number of special types of 4x4 matrices and special functions to operate on them for standard operation required by our methodology (e.g. transformations). More over the meaning of some standard operators like '+' or '*' has been "overridden" to have the possibility of using it also in matrix operations.

7. NOTES ON COMPUTATIONAL COMPLEXITY.

The presented methodology can be easily utilized to write simulation software. The resulting programs are generally quite compact if they are written using specialized libraries like spacelib or chain++. These libraries have been developed taking into account the special properties of each matrix obtaining a quite efficient code. For example there is a special function to invert the position matrices developed taking into accounts it particularities (see part 1, paragraph 2.2).

```
...
ifstream motion("data");
...
while(TRUE)
{
  // KINEMATIC LOOP
  for(int i=1;i<=nlink;i++)
  {
    // Read motion
    motion >> q >> qp >> qpp;
    if(motion.eof()) exit(0);

    // build rel. pos., vel., acc. matrices
    A[i].SetUp(jtype[i],theta[i],s[i],b[i],a[i],
       alfa[i],q);
    W[i].SetUp(jtype[i],qp);
    H[i].SetUp(jtype[i],qp,qpp);

    // evaluate abs. position matrix
    M0[i] = M0[i-1]*A[i];
    // evaluate rel. vel. and acc. matrices in (0)
    W0[i] = W[i].ChangeRef(M0[i-1]);
    H0[i] = H[i].ChangeRef(M0[i-1]);

    // evaluate abs. vel. matrix
    WA[i] = WA[i-1] + W0[i];
    // evaluate abs. acc. matrix (Coriolis' theorem)
    HA[i] = HA[i-1] + H0[i] + 2*WA[i-1]*W0[i];
  }

  // DYNAMIC LOOP
  .........
}
```

Fig. 5: C++ language example of kinematic loop.

Although a precise determination of the computational complexity has not been performed we guess that an average programmer can write programs whose computational complexity is close to the minimum. However as the presented examples suggest the time necessary to write the programs is quite short.

8. CONCLUSIONS.

The presented applications of the matrix approach show how it can be easily adopted to write the kinematics and dynamics equation of a chain of rigid bodies.

For simple mechanisms the equation can be directly developed analytically while for more complicated chains of rigid bodies they can be easily translated into application programs. With the use of a standard library (such as SPACELIB or CHAIN++) performing the basic matrix operation required by our notation, these application programs are very short and assume very simple forms.

In summary, the presented methodology is a convenient tool both for analytical and numerical analysis of the system of rigid bodies.

9. APPENDIX.

9.1. DERIVATIVES OF $\mathbf{M}^{-1}$ WITH RESPECT TO TIME.

The time derivatives of the invers of a position matrix can be found by deriving with respect to time the equation

$$\mathbf{M}\mathbf{M}^{-1} = \mathbf{[1]}$$

which becomes $\frac{d\mathbf{M}}{dt}\mathbf{M}^{-1} + \mathbf{M}\frac{d\mathbf{M}^{-1}}{dt} = \mathbf{[0]}$ and so

$$\frac{d\mathbf{M}^{-1}}{dt} = -\mathbf{M}^{-1}\frac{d\mathbf{M}}{dt}\mathbf{M}^{-1}$$

9.2. DERIVATIVES OF $\mathbf{L}_{i(0)}$ WITH RESPECT TO TIME.

Starting from the transformation of the reference of matrix $\mathbf{L}_{i(0)} = \mathbf{M}_{0,i-1}\mathbf{L}_i\mathbf{M}_{0,m-1}^{-1}$, and differentiating respect to time t we have $\frac{d\mathbf{L}_{i(0)}}{dt} = \frac{d\mathbf{M}_{0,i-1}}{dt}\mathbf{L}_i\mathbf{M}_{0,i-1}^{-1} + \mathbf{M}_{0,i-1}\mathbf{L}_i\frac{d\mathbf{M}_{0,i-1}^{-1}}{dt}$,

but $\frac{d\mathbf{M}_{0,i-1}}{dt} = \mathbf{W}_{0,i-1}\mathbf{M}_{0,i-1}$ and $\frac{d\mathbf{M}_{0,i-1}^{-1}}{dt} = -\mathbf{M}_{0,i-1}^{-1}\mathbf{W}_{0,i-1}$ so we can write $\frac{d\mathbf{L}_{i(0)}}{dt} = \mathbf{W}_{0,i-1}\mathbf{L}_{i(0)} - \mathbf{L}_{i(0)}\mathbf{W}_{0,i-1}$.

10. REFERENCES.

All the references are quoted at the end of part 1 of the paper.

```
/* program for direct kinematics and inverse dynamics of ANY serial robot v.2
Developed on MS-DOS operative system with Microsoft C compiler V. 5.10 */

#include <stdio.h>
#include <math.h>
#include "spacelib.h"

main(int argc,char *argv[])
{
  #define MAXLINK 10                                    /* max number of links */
  int nlink,jtype[MAXLINK];                            /* n.links;  joint type */
  float theta[MAXLINK],s[MAXLINK];                  /* Extended D.&H. parameters */
  float b[MAXLINK],a[MAXLINK],alfa[MAXLINK];
  float m,jxx,jxy,jxz,jyy,jyz,jzz,xg,yg,zg;             /* dynamics parameters */
  float q,qp,qpp;                                          /* joint variables */
  float gx,gy,gz;                                      /* gravity acceleration */
  float fx,fy,fz,cx,cy,cz;        /* external forces and torques on end-effector */
  FILE *data;                                /* file including robot parametres */
  FILE *motion;                              /* file including actuator motions */
  int i;                                                        /* counter */
  int ierr;                                                   /* error code */
  float t,dt;
  if(argc!=3) exit(1);                                 /* check of input data */
  data=fopen(argv[1],"r");    if(data==0) exit(2);
  motion=fopen(argv[2],"r");  if(motion==0) exit(3);
                                                       /* define 4x4 matrices */
  MAT4 A[MAXLINK], M0[MAXLINK];
  MAT4 W[MAXLINK], W0[MAXLINK], WA[MAXLINK];
  MAT4 H[MAXLINK], H0[MAXLINK], HA[MAXLINK];
  MAT4 J[MAXLINK], J0[MAXLINK];
  MAT4 FI[MAXLINK],ACT0[MAXLINK+1], EXT, Hg, Ht;
  MAT4 TMP;                                              /* temporary matrix */
                                                                 /* step (1) */
  idmat4(M0[0]); clear4(WA[0]); clear4(HA[0]);     /* INITIALIZATION of matrices */

                          /* READ ROBOT DESCRIPTION */

  fscanf(data,"%d",&nlink);                                    /* n. of links */
  for (i=1;i<=nlink;i++)                                      /* for each link */
  {  fscanf(data,"%d %f %f %f %f %f",                        /* D.&H. parameters */
           &jtype[i],&theta[i],&s[i],&b[i],&a[i],&alfa[i]);
    fscanf(data,"%f %f %f %f %f %f %f",                        /* dynamic data */
          &m,&jxx,&jxy,&jxz,&jyy,&jyz,&jzz);
    fscanf(data,"%f %f %f",&xg,&yg,&zg);
    jtoJ(m,jxx,jyy,jzz,jxy,jyz,jxz,xg,yg,zg,J[i]);       /* build inertia matrix */
  }
  fscanf(data,"%f %f %f",&gx,&gy,&gz);       /* read gravity acceleration vector */
  gtom(gx,gy,gz,Hg);                        /* build gravity acceleration matrix */

  for(t=0;;t+=dt)                                    /* for each instant of time */
  {
                              /* ***** KINEMATICS *****  */
  for (i=1;i<=nlink;i++)                                        /* for each link */
  { ierr=fscanf(motion,"%f %f %f",&q,&qp,&qpp);               /* read motions (2) */
    if (ierr!=3) goto end_motion;                  /* end of data in file MOTION */
                                          /* build relative position matrix (3) */
    dhtom(jtype[i],theta[i],s[i],b[i],a[i],alfa[i],q,A[i]);
    velacctoWH(jtype[i],qp,qpp,W[i],H[i]);                /* build relative velocity
                                        and acceleration matrix in local frame (4) */

    molt4(M0[i-1],A[i],M0[i]);          /* evaluate absolute position matrix (5) */

    trasf_mami(W[i],M0[i-1],W0[i]);           /* transform relative velocity matrix
                                             from local frame to base frame  (6) */
    trasf_mami(H[i],M0[i-1],H0[i]);       /* transform relative acceleration matrix
                                            from  local  frame  to  base frame  (7) */
    sum4(WA[i-1],W0[i],WA[i]);         /* evaluate absolute velocity matrix  (8) */
                                     /* evaluate absolute acceleration matrix (9) */
    coriolis(HA[i-1],H0[i],WA[i-1],W0[i],HA[i]);
  }
                                  /* ***** DYNAMICS ***** */
                                                    /* initializations  (10) */
                                               /* read external actions on end-effector */
  fscanf(data,"%f %f %f %f %f %f",&fx,&fy,&fz,&cx,&cy,&cz);
  actom(fx,fy,fz,cx,cy,cz,EXT);                    /* build external action matrix */
  trasf_mamt4(EXT,M0[nlink],ACT0[nlink+1]);          /* transforms external actions
                                                        from local to base frame */
  for(i=nlink;i>0;i--)                                          /* for each link */
  { trasf_mamt4(J[i],M0[i],J0[i]);                    /* transform inertia matrix
                                                  from local to base frame  (11) */
    rmolt4(HA[i],-1.,TMP);                /* change sign to find inertia action */
    sum4(TMP,Hg,Ht);                      /* evaluate total acceleration matrix */
    skew4(Ht,J0[i],FI[i]);                          /* evaluate the action matrix
                                                 due to inertia and weight (12) */
    sum4(FI[i],ACT0[i+1],ACT0[i]);          /* evaluate total action matrix (13) */
   }
                              /* ***** OUTPUT RESULTS ***** */
for(i=1;i<=nlink;i++)                                           /* for each link */
   { printf("\n\n Link %d \n\n",i);
     printm4("rel. pos. matrix",A[i]);
     printm4("abs. pos. matrix",M0[i]);
     printm4("rel. vel. matrix in frame (i)",W[i]);
     printm4("rel. vel. matrix in frame (0)",W0[i]);
     printm4("absolute velocity matrix in frame (0)",WA[i]);
     printm4("rel. acc. matrix in frame (i)",H[i]);
     printm4("rel. acc. matrix in frame (0)",H0[i]);
     printm4("absolute acceleration matrix in frame (0)",HA[i]);
     printm4("inertia matrix in frame (i)",J[i]);
     printm4("inertia matrix in frame (0)",J0[i]);
     printm4("total actions",FI[i]);
     printm4("action on joint i",ACT0[i]);
   }
}
end_motion: exit(0);}                                          /*  end main  */
```

Fig. 6: C language program for the direct kinematic and inverse dynamic analysis of ANY serial manipulator. Numbers in parenteses refer to the flow chart of Fig. 4

```
/* program for direct kinematics and inverse dynamics of ANY serial robot v.2 Developed on MS-DOS operative system
with Microsoft C compiler V. 5.10 */

#include <stdio.h>
#include <math.h>
#include "spacelib.h"

main(int argc,char *argv[])
{
  #define MAXLINK 10                                    /* max number of links */
  int nlink,jtype[MAXLINK];                            /* n.links;  joint type */
  float theta[MAXLINK],s[MAXLINK];                 /* Extended D.&H. parameters */
  float b[MAXLINK],a[MAXLINK],alfa[MAXLINK];
  float m,jxx,jxy,jxz,jyy,jyz,jzz,xg,yg,zg;             /* dynamics parameters */
  float q,qp,qpp;                                          /* joint variables */
  float gx,gy,gz;                                      /* gravity acceleration */
  float fx,fy,fz,cx,cy,cz;        /* external forces and torques on end-effector */
  FILE *data;                                /* file including robot parametres */
  FILE *motion;                              /* file including actuator motions */
  int i;                                                        /* counter */
  int ierr;                                                  /* error code */
  float t,dt;
  if(argc!=3) exit(1);                                 /* check of input data */
  data=fopen(argv[1],"r");    if(data==0) exit(2);
  motion=fopen(argv[2],"r");  if(motion==0) exit(3);
                                                       /* define 4x4 matrices */
  MAT4 A[MAXLINK], M0[MAXLINK];
  MAT4 W[MAXLINK], W0[MAXLINK], WA[MAXLINK];
  MAT4 H[MAXLINK], H0[MAXLINK], HA[MAXLINK];
  MAT4 J[MAXLINK], J0[MAXLINK];
  MAT4 FI[MAXLINK],ACT0[MAXLINK+1], EXT, Hg, Ht;
  MAT4 TMP;                                               /* temporary matrix */
                                                                /* step (1) */
  idmat4(M0[0]); clear4(WA[0]); clear4(HA[0]);     /* INITIALIZATION of matrices */

                                        /* READ ROBOT DESCRIPTION */

  fscanf(data,"%d",&nlink);                                    /* n. of links */
  for (i=1;i<=nlink;i++)                                     /* for each link */
  {  fscanf(data,"%d %f %f %f %f %f",                      /* D.&H. parameters */
          &jtype[i],&theta[i],&s[i],&b[i],&a[i],&alfa[i]);
    fscanf(data,"%f %f %f %f %f %f %f",                        /* dynamic data */
         &m,&jxx,&jxy,&jxz,&jyy,&jyz,&jzz);
    fscanf(data,"%f %f %f",&xg,&yg,&zg);
    jtoJ(m,jxx,jyy,jzz,jxy,jyz,jxz,xg,yg,zg,J[i]);       /* build inertia matrix */
  }
  fscanf(data,"%f %f %f",&gx,&gy,&gz);       /* read gravity acceleration vector */
  gtom(gx,gy,gz,Hg);                        /* build gravity acceleration matrix */
 for(t=0;;t+=dt)                                    /* for each instant of time */
  {
                                        /* ***** KINEMATICS *****  */
  for (i=1;i<=nlink;i++)                                       /* for each link */
  { ierr=fscanf(motion,"%f %f %f",&q,&qp,&qpp);               /* read motions (2) */
    if (ierr!=3) goto end_motion;                 /* end of data in file MOTION */
                                          /* build relative position matrix (3) */
    dhtom(jtype[i],theta[i],s[i],b[i],a[i],alfa[i],q,A[i]);
    velacctoWH(jtype[i],qp,qpp,W[i],H[i]);                /* build relative velocity
                                  and acceleration matrix in local frame (4) */

    molt4(M0[i-1],A[i],M0[i]);          /* evaluate absolute position matrix (5) */

    trasf_mami(W[i],M0[i-1],W0[i]);           /* transform relative velocity matrix
                                     from local frame to base frame  (6) */
    trasf_mami(H[i],M0[i-1],H0[i]);       /* transform relative acceleration matrix
                                    from  local  frame  to  base frame  (7) */
    sum4(WA[i-1],W0[i],WA[i]);         /* evaluate absolute velocity matrix  (8) */
                                /* evaluate absolute acceleration matrix (9) */
    coriolis(HA[i-1],H0[i],WA[i-1],W0[i],HA[i]);
  }
                                        /* ***** DYNAMICS ***** */
                                          /* initializations  (10) */
                                   /* read external actions on end-effector */
  fscanf(data,"%f %f %f %f %f %f",&fx,&fy,&fz,&cx,&cy,&cz);
  actom(fx,fy,fz,cx,cy,cz,EXT);                    /* build external action matrix */
  trasf_mamt4(EXT,M0[nlink],ACT0[nlink+1]);          /* transforms external actions
                                                     from local to base frame */
  for(i=nlink;i>0;i--)                                         /* for each link */
  { trasf_mamt4(J[i],M0[i],J0[i]);                      /* transform inertia matrix
                                                  from local to base frame  (11) */
    rmolt4(HA[i],-1.,TMP);                  /* change sign to find inertia action */
    sum4(TMP,Hg,Ht);                        /* evaluate total acceleration matrix */
    skew4(Ht,J0[i],FI[i]);                            /* evaluate the action matrix
                                                 due to inertia and weight (12) */
    sum4(FI[i],ACT0[i+1],ACT0[i]);          /* evaluate total action matrix (13) */
   }
                                        /* ***** OUTPUT RESULTS ***** */
for(i=1;i<=nlink;i++)                                          /* for each link */
   { printf("\n\n Link %d \n\n",i);
     printm4("rel. pos. matrix",A[i]);
     printm4("abs. pos. matrix",M0[i]);
     printm4("rel. vel. matrix in frame (i)",W[i]);
     printm4("rel. vel. matrix in frame (0)",W0[i]);
     printm4("absolute velocity matrix in frame (0)",WA[i]);
     printm4("rel. acc. matrix in frame (i)",H[i]);
```

```
     printm4("rel. acc. matrix in frame (0)",H0[i]);
     printm4("absolute acceleration matrix in frame (0)",HA[i]);
     printm4("inertia matrix in frame (i)",J[i]);
     printm4("inertia matrix in frame (0)",J0[i]);
     printm4("total actions",FI[i]);
     printm4("action on joint i",ACT0[i]);
   }
}
end_motion: exit(0);}                                          /*  end main  */
```