Robotica (1997) volume 15, pp 153–161. © 1997 Cambridge University Press

SYMORO+: A system for the symbolic modelling of robots

Wisama Khalil and Denis Creusot

Ecole Centrale de Nantes, Laboratoire d'Automatique de Nantes URA C.N.R.S 823, P.B 92 101, 44321 Nantes cedex 3 (France), E-Mail: Wisama.Khalil@lan.ec-nantes.fr

SUMMARY

This paper presents the software package SYMORO+ for the automatic symbolic modelling of robots. This package permits to generate the direct geometric model, the inverse geometric model, the direct kinematic model, the inverse kinematic model, the dynamic model, and the inertial parameters identification models.

The structure of the robots can be serial, tree structure or containing closed loops. The package runs on Sun stations and PC computers, it has been developed under *MATHEMATICA* and *C* language. In this paper we give an overview of the algorithms used in the different models; the computational cost of the dynamic models of the PUMA robot are given.

KEYWORDS: Symbolic calculation; Robots; Symbolic modelling; SYMORO+.

1. INTRODUCTION

Many works have been devoted to the automatic generation of the symbolic modelling of robots. Most of these works were interested in the generation of some models or, more especially, in the dynamic model only. SYMORO+ generates almost all the symbolic models needed in the simulation, control, identification, and design of robots. This package is the result of the research work of the robotics team of the "Laboratoire d'Automatique de Nantes", in the field of modelling, control and identification of robots.

The programs generating the different models are developed using *MATHEMATICA*.[1] A graphic interface, written in language *C,* is developed such that the user can define his robot or the desired model using recent stations environment. The main interface page is seen in Figure 1, on which we distinguish the parameters of the robot and the following main menus: Robot, Geometric, Kinematic, Dynamic, Identification, Optimizer.

The functions and models corresponding to these menus are given in Table I.

The paper is organized as follows: Section 2 gives the parameters needed to define a robot, then Sections $3, \ldots, 7$ present the differnet models which can be generated by SYMORO+, an idea will be given to describe the algorithms which are used in generating the different models.

2. ROBOT DESCRIPTION

The definition of a robot is carried out using the menu **Robot**. The software package SYMORO+ can treat serial, tree structure or closed loop robots. The description of the geometry of the robot is carried out using the notations of Khalil and Kleinfinger.[2] At first the number of moving links (N_L), the number of joints (N_j) and the type of structure (serial, tree structure or closed) must be given. The following three sets of parameters (Figure 1) are then defined either numerically or symbolically:

2.1 The geometric parameters

These parameters define the kinematic of the structure, the type of joints and the location of the link frames with respect to its antecedent.[2] The coordinate frame j is assigned fixed with respect to link j. The $\mathbf{z}_j$ axis is along the axis of joint j, the $\mathbf{x}_j$ axis is along the common perpendicular of $\mathbf{z}_j$ and one of the following axis on the same link. The geometry of the robot is defined by the following parameters (for $j = 1, \ldots, N_f$):

—γ_j, b_j, α_j, d_j, θ_j, r_j defining frame j with respect to its antecedent frame $a(j)$, Figure 2, in serial robots γ_j, b_j are equal to zero. It is to be noted that the joint varaible q_j is equal to θ_j if j is rotational and equal to r_j for j translational,

—σ_j defining the type of joint j. $\sigma_j = 0$ for j rotational, $\sigma_j = 1$ for j translational,

—the antecedent frame $a(j)$,

—μ_j indicates if the joint j is motorized (active) or not (passive). In open loop robots (serial or tree structure robots) all the joints are supposed active.

It is to be noted that the number of frames N_f in serial or tree structure robot is equal to the number of moving links N_L. In the case of closed loop robot we suppose each closed loop opened in one of its passive joint to construct an equivalent tree structure, then we add two frames on one of the links surrounding the opened joints. If the number of the opened joint is k the number of the corresponding additional frames will be k and $k + B$, these two frames are aligned but their antecedent frames are not the same (Figure 3). Thus in the case of closed loop robots

$$N_f = N_L + 2B$$

where $B = N_j - N_L =$ number of closed loops.

2.2 Dynamic parameters

These parameters are composed of inertial and friction parameters. For each link the following parameters have to be defined:

$$[XX_j, XY_j, XZ_j, YY_j, YZ_j, ZZ_j, MX_j, MY_j, MZ_j, M_j, Ia_j],$$

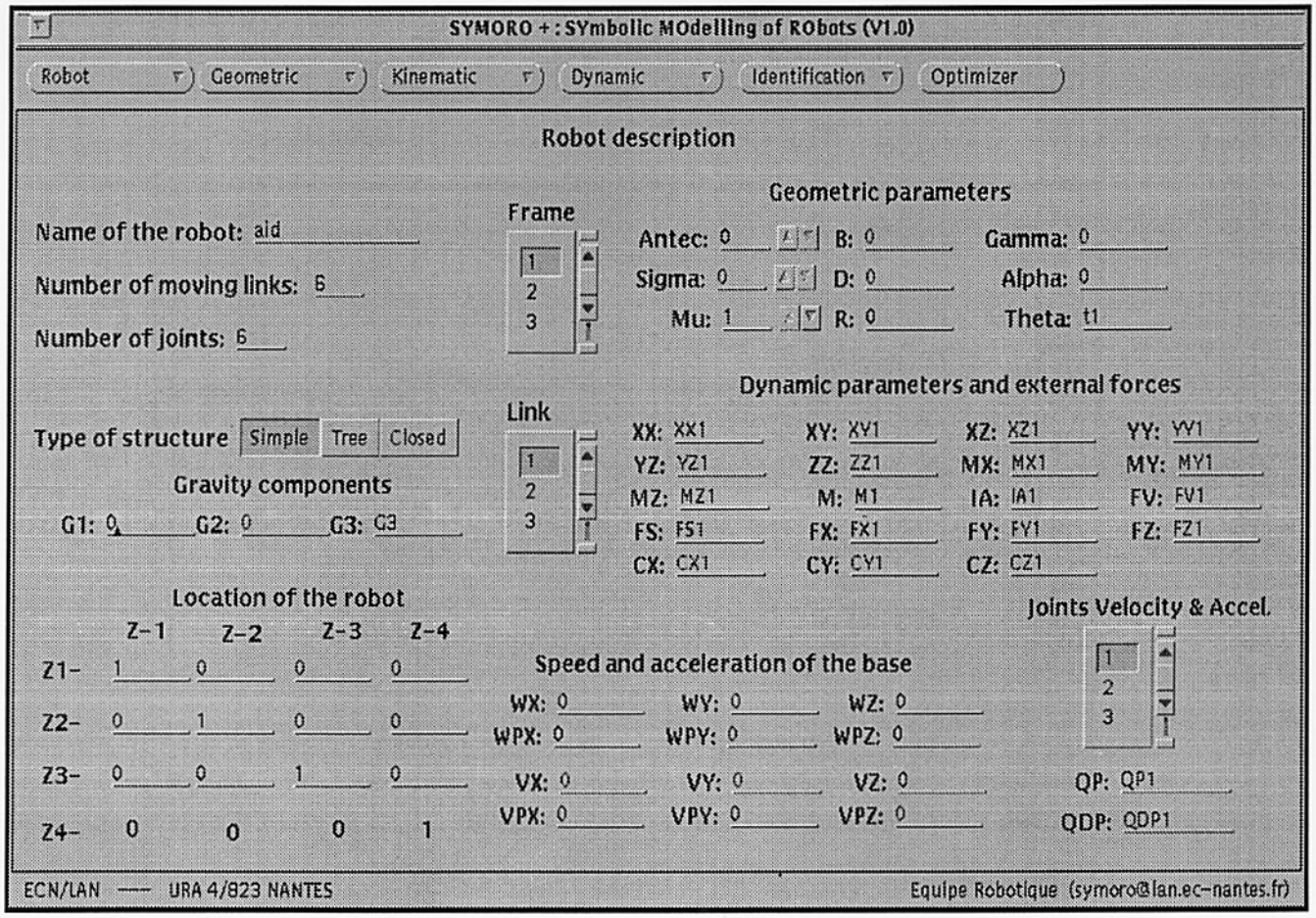

Fig. 1. Main page of SYMORO+.

and

$$[Fv_j, Fsj]$$

where:

—$(XX_j, \ldots, ZZ_j)$ are the elements of the inertia matrix ${}^j\mathbf{J}_j$, defining the inertia of link j, around the origin of frame j,
—(MX_j, MY_j, MZ_j) are the elements of ${}^j\mathbf{MS}_j$ defining the first moments of link j,
—M_j the mass of link j,
—Ia_j inertia of motor j referred to the joint side.
—Fv_j, Fs_j are the viscous and static friction coefficients of joint j.

3.2 General parameters

The following parameters can also be defined.

—The location of the base of the robot with respect to a general fixed frame (matrix **Z**).
—joint velocities (QPj) and accelerations ($QDPj$).
—external forces of each link on the environment (FXj, FYj, FZj)

Table I. Functions and models of the menus of SYMORO+

Robot	**Geometric**	**Kinematic**
New	Transformation matrix...	Jacobian matrix...
Open...	Fast geometric model...	Inverse Jacobian & determinant...
Save	I.G.M Pieper method	Velocity of links
Save as...	I.G.M Paul method	Acceleration of links
Quit	I.G.M general method	JPQP
	Constraint geometric eq. of loops	Constraint kinematic eq. of loops

Dynamic	**Identification**
I.D.M Newton Euler method	Base inertial parameters
I.D.M Lagrange method	Dynamic Identification model
Inertia matrix	Energy Identification model
Centrifugal, Coriolis & Gravity torque	Filtered dynamic Identification model
Direct dynamic model	

Optimizer

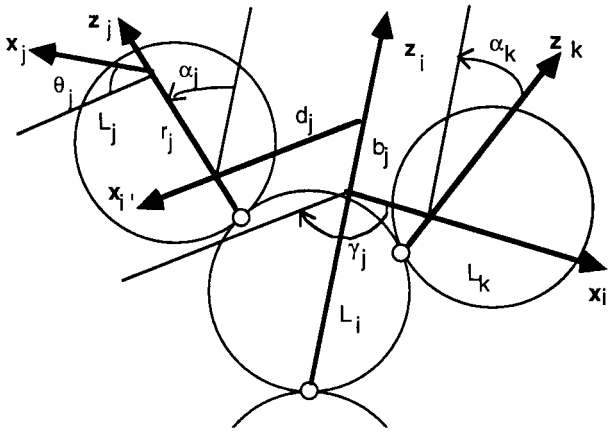

Fig. 2. Definition of link frames.

—external moments of each link on the environment (*CXj, CYj, CZj*).
—The speed and the acceleration of the base of the robot.
—the acceleration of gravity: $\mathbf{g} = [G1 \;\; G2 \;\; G3]^T$.

3. THE GEOMETRIC MODELS

The following models can be obtained under menu Geometric:

3.1 The direct geometric model

The direct geometric model, gives the position and orientation of the end effector as function of the motorized joints variables. It can be obtained by the multiplication of the transformation matrices along the path between the base of the robot and the terminal link.

A frame *j* will be defined with respect to frame *r,* by the following (4×4) transformation matrix:

$$ {}^{r}\mathbf{T}_j = \begin{bmatrix} {}^{r}\mathbf{A}_j & {}^{r}\mathbf{P}_j \\ 0 \;\; 0 \;\; 0 & 1 \end{bmatrix} \tag{1a} $$

where

- ${}^{r}\mathbf{A}_j$ defines the orientation of frame *j* with respect to frame *r.*
- ${}^{r}\mathbf{P}_j$ defines the position of the origin of frame *j* with respect to frame *i.*

The frame *j* coordinate will be defined with respect to frame *i*, with $i = a(j)$, by the following (4×4) matrix:

$$ {}^{i}\mathbf{T}_j = \begin{bmatrix} C\gamma_j C\theta_j - S\gamma_j C\alpha_j S\theta_j & -C\gamma_j S\theta_j - S\gamma_j C\alpha_j C\theta_j & S\gamma_j S\alpha_j & d_j C\gamma_j + r_j S\gamma_j S\alpha_j \\ S\gamma_j C\theta_j + C\gamma_j C\alpha_j S\theta_j & -S\gamma_j S\theta_j + C\gamma_j C\alpha_j C\theta_j & -C\gamma_j S\alpha_j & d_j S\gamma_j - r_j C\gamma_j S\alpha_j \\ S\alpha_j S\theta_j & S\alpha_j C\theta_j & C\alpha_j & r_j C\alpha_j + b_j \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1b} $$

where:

- $S = \sin$, $C = \cos$,

In the case of open loop robots the direct geometric model is defined by the transformation matrix of the terminal link referred to the base. In the case of closed loop robots the constraint equations defining the relation between the passive joint variables, in the path between the terminal link and the base, and the motorized joints outside this path can be obtained by solving the geometric constraint equations (Section 3.4), in order to obtain the location of the end effector as function of motorized joints only.

3.2 Fast geometric model

This function gives the transformation matrix between two frames in customized form, using intermediate variables in order to minimize its calculation cost.

3.3 The inverse geometric model

The inverse geometric model is the closed form solution giving all the configurations of the robot corresponding to a given location of the end effector. In our software package three methods are used:

- The first is derived from the work presented in references 3 and 4. This method gives the solution of six degrees of freedom robot provided that one of the following conditions is verified:

—the robot contains three translational joints.
—the robot has three rotational joints defining a spherical joint (their axes are intersecting).

- The second method is derived from the work given in references 5 and 6. This method can provide the solution of most of the current industrial robots.
- The third method is derived from the general method of Raghavan and Roth,[7,8] where the solution of one variable is given as a polynomial equation of degree 16 at most, then the other variables can be obtained.

In the case of closed loop robots, the programs (Pieper, Paul, or general method) give the solution of the joints on the direct path between the base of the robot and the end effector. To get the values of the motorized joint positions outside this path the geometric constraint equations of the loops must be resolved, as seen in the following section.

3.4 Geometric constraint equations of loops

The direct and inverse geometric models of closed loop robots can be considered as the direct and inverse geometric robot of the serial path from the base to the terminal link plus the solution of the geometric constraint equations of the closed loops of the robot.

We make use of the method developed in reference 4. The solution is ensured analytically for each closed loop with less than five passive joints. The geometric constraint equations can be obtained by solving the transformation matrix along the loop (Figure 3) which is equal to the identity matrix:

$$ {}^{B}\mathbf{T}_i \cdots {}^{j}\mathbf{T}_{k+B} = \mathbf{I}_4 \tag{2} $$

with $\mathbf{I}_4$ is the 4×4 identity matrix.

4. THE KINEMATIC MODELING

The direct kinematic model defines the velocity of the end effector as function of the joint velocities. It can be written as:

$$ \begin{bmatrix} {}^{r}\mathbf{V}_n \\ {}^{r}\omega_n \end{bmatrix} = {}^{r}\mathbf{J}_n \dot{\mathbf{q}} \tag{3} $$

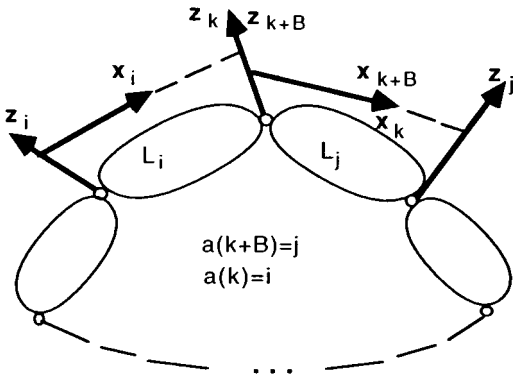

Fig. 3. Frames associated with the opened joint.

where:

$\dot{\mathbf{q}}$ is the vector of joint velocities,

${}^r\mathbf{V}_n$, ${}^r\omega_n$ define the translational and angular velocities of frame n referred to frame r,

${}^r\mathbf{J}_n$ is the Jacobian matrix of link n, referred to frame r.

The following models can be generated from the menu kinematic:

4.1 Calculation of the Jacobian matrix

SYMORO+ calculates the Jacobian matrix using the method developed by Renaud.[9] It give the Jacobian matrix as the product of three matrices, two of them are of full rank and the third contains simple terms such that:

$$ {}^r\mathbf{J}_n = \begin{bmatrix} {}^r\mathbf{A}_i & \mathbf{0}_3 \\ \mathbf{0}_3 & {}^r\mathbf{A}_i \end{bmatrix} \begin{bmatrix} \mathbf{I}_3 & -{}^i\hat{\mathbf{L}}_{j,n} \\ \mathbf{0}_3 & \mathbf{I}_3 \end{bmatrix} {}^i\mathbf{J}_{n,j} \tag{4} $$

with $\mathbf{0}_3$ and $\mathbf{I}_3$ denotes the zero and identity (3×3) matrix respectively, the symbol $\hat{}$ means the 3×3 matrix of the vector product; such that $\mathbf{a}\times\mathbf{b} = \hat{\mathbf{a}}\mathbf{b}$.

The vectoriel matrix $\mathbf{J}_{n,j}$ is given as:

$$ \mathbf{J}_{n,j} = \begin{bmatrix} \sigma_f\mathbf{a}_f + \bar{\sigma}_f(\mathbf{a}_f \times \mathbf{L}_{f,j}) & \cdots & \sigma_n\mathbf{a}_n + \bar{\sigma}_n(\mathbf{a}_n \times \mathbf{L}_{n,j}) \\ \bar{\sigma}_f\mathbf{a}_f & \cdots & \bar{\sigma}_n\mathbf{a}_n \end{bmatrix} \tag{5} $$

where

f is the first joint on the path between the base and link n, such that $f = 1$ in the case of serial robots.

$\mathbf{a}_j$ is the unit vector along the axis $\mathbf{z}_j$

$\mathbf{L}_{i,j}$ is the position vector connecting the origin of frame i to that of frame j

$\bar{\sigma}_j$ is equal to 1 if j is rotational, and equal to 0 if j is translational

4.2 Calculation of the velocities of the links

The translational and angular velocities of links can be calculated using relation (3), they can be calculated more efficiently, from the number of operations point of view, by the following recursive algorithm, for $j = 1, \ldots, N_L$,

$$ {}^j\omega_i = {}^j\mathbf{A}_i\,{}^i\omega_i \tag{6a} $$

$$ {}^j\omega_j = {}^j\omega_i + \bar{\sigma}_j\dot{q}_j\,{}^j\mathbf{a}_j \tag{6b} $$

$$ {}^j\mathbf{V}_j = {}^j\mathbf{A}_i({}^i\mathbf{V}_i + {}^i\omega_i \times {}^i\mathbf{P}_j) + \sigma_j\dot{q}_j\,{}^j\mathbf{a}_j \tag{6c} $$

where $i = a(j)$ denotes the link antecedent to j, and ${}^j\mathbf{a}_j$ is the unit vector $[0\ 0\ 1]^T$. The initial values ${}^0\omega_0$ and ${}^0\mathbf{V}_0$ are the velocities of the base.

4.3 Calculation of links accelerations

Differentiating equation (3), we get the translational and rotational acceleration of link n as:

$$ \begin{bmatrix} \dot{\mathbf{V}}_n \\ \dot{\omega}_n \end{bmatrix} = \dot{\mathbf{J}}_n\dot{\mathbf{q}} + \mathbf{J}_n\ddot{\mathbf{q}} \tag{7} $$

The calaulation of link accelerations using (7) is time consuming, it is more efficient to calculate the link accelerations by the following recursive equations, which can be obtained by differentiating equations (6):

$$ {}^j\dot{\omega}_j = {}^j\mathbf{A}_i\,{}^i\dot{\omega}_i + \bar{\sigma}_j(\ddot{q}_j\,{}^j\mathbf{a}_j + {}^j\omega_i \times \dot{q}_j\,{}^j\mathbf{a}_j) \tag{8} $$

$$ {}^j\mathbf{U}_j = {}^j\hat{\dot{\omega}}_j + {}^j\hat{\omega}_j\,{}^j\hat{\omega}_j \tag{9} $$

$$ {}^j\dot{\mathbf{V}}_j = {}^j\mathbf{A}_i[{}^i\dot{\mathbf{V}}_i + {}^i\mathbf{U}_i\,{}^i\mathbf{P}_j] + \sigma_j[\ddot{q}_j\,{}^j\mathbf{a}_j + 2{}^j\omega_i \times \dot{q}_j\,{}^j\mathbf{a}_j] \tag{10} $$

where: $i = a(j)$, ${}^j\omega_i$ and ${}^j\omega_j$ are calculated using equations (6).

4.4 Calculation of $\dot{\mathbf{J}}\dot{\mathbf{q}}$ of the links

In some applications (task space control), we need to calculate the vector $\dot{\mathbf{J}}\dot{\mathbf{q}}$. From equation (7) it can be seen that $\dot{\mathbf{J}}\dot{\mathbf{q}}$ can be obtained by the use of the recursive equations $(8, \ldots, 10)$ and by assuming $\ddot{q}_j$ equal zero.[10]

4.5 Kinematic constraint equations

This function gives the relation between the velocities of the passive joints as function of the velocities of the motorized joints. It can be calculated by equating the velocities on the terminal frame of each loop using the two sides of the loop. From Figure 3, we get:

$$ \begin{bmatrix} \mathbf{V}_k \\ \omega_k \end{bmatrix} = {}^k\mathbf{J}_k\dot{\boldsymbol{q}}_1 = {}^{k+B}\mathbf{J}_{k+B}\dot{\boldsymbol{q}}_2 \tag{11} $$

where $\dot{\mathbf{q}}_1$ and $\dot{\mathbf{q}}_2$ denote the velocities of the joints from the root of the loop to the opened joint, along each side of the loop.

As function of the type of the loop, spatial or planar, and the type of the opened joint, redundant rows in equation (11) can be automatically eliminated.[11]

Doing this procedure on all the loops we obtain the kinematic constraint equation as:

$$ \begin{bmatrix} \mathbf{W}a & \mathbf{W}p & \mathbf{0} \\ \mathbf{W}ac & \mathbf{W}pc & \mathbf{W}c \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{q}}_a \\ \dot{\boldsymbol{q}}_p \\ \dot{\boldsymbol{q}}_c \end{bmatrix} = \mathbf{0} \tag{12} $$

where $\dot{\mathbf{q}}_a$, $\dot{\boldsymbol{q}}_p$, $\dot{\boldsymbol{q}}_c$ represent the velocity of active (motorized), passive, and opened joints, respectively.

From the first row of equation (12) we obtain:

$$ \dot{\mathbf{q}}_p = -\mathbf{W}_p^{-1}\mathbf{W}_a\dot{\boldsymbol{q}}_a = \mathbf{W}\dot{\mathbf{q}}_a \tag{13} $$

The acceleration constraint equation can also be given by differentiating equation (12) with respect to the time as:

$$\begin{bmatrix} \mathbf{W_a} & \mathbf{W_p} & \mathbf{0} \\ \mathbf{W_{ac}} & \mathbf{W_{pc}} & \mathbf{Wc} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}}_a \\ \ddot{\mathbf{q}}_p \\ \ddot{\mathbf{q}}_c \end{bmatrix} + \begin{bmatrix} \psi \\ \Phi \end{bmatrix} = \mathbf{0} \tag{14}$$

In fact ψ and Φ are calculated using a recursive method similar to that calculating $\dot{\mathbf{J}}\dot{\mathbf{q}}$ as given in Section 4.4.

4.6 Inverse kinematic model

The inverse kinematic model gives $\dot{\mathbf{q}}$ as function of the link velocities $\mathbf{V}_n$, ω_n. The solution implies to calculate the inverse of the Jacobian matrix, which is reduced to the inversion of ${}^i\mathbf{J}_{n,j}$. The determinant is also provided to study the singular configurations of the robot.

5. THE DYNAMIC MODELLING

The following models can be obtained in the menu dynamic:

5.1 Inverse dynamic model (I.D.M)

The inverse dynamic model gives the motor torques or forces as a function of the joint positions, velocities and accelerations. The calculation of this model represent the computed control law which decouples and linearizes the equation of motion of the robots. The model generated by our program has a reduced number of operations, such that it can be used for dynamic control applications, and this is thanks to the use of the base inertial parameters (see Section 6) and a customized Newton Euler algorithm which is linear in the inertial parameters.[12]

The I.D.M using Newton Euler algorithm is given as follows:

The following notations will be used:

$\mathbf{F}_j$ total forces on link j,

$\mathbf{M}_j$ total moments on link j around the origin o_j.

$\mathbf{f}_j$ force on link j due to motor j and its antecedent link i,

$\mathbf{m}_j$ moment on link j due to motor j and the antecedent link i.

$\mathbf{f}_{e,j}$ the force of link j on the environment,

$\mathbf{m}_{e,j}$ the moments of link j on the environment,

Γ motor torques or forces vector,

$\mathbf{g}$ acceleration of gravity,

The algorithm consists of two recursive calculations:

a The forward calculation, for $j = 1, \ldots, N_L$, is given as:

$${}^j\mathbf{F}_j = M_j{}^j\dot{\boldsymbol{V}}_j + {}^j\mathbf{U}_j{}^j\mathbf{MS}_j \tag{15}$$

$${}^j\mathbf{M}_j = {}^j\mathbf{J}_j{}^j\dot{\omega}_j + {}^j\omega_j \times ({}^j\mathbf{J}_j{}^j\omega_j) + {}^j\mathbf{MS}_j \times {}^j\dot{\boldsymbol{V}}_j \tag{16}$$

where ${}^j\omega_j$, ${}^j\dot{\omega}_j$, ${}^j\dot{\boldsymbol{V}}_j$, ${}^j\mathbf{U}_j$ are calculated using equations (6), (8), (9), (10).

To take into account the gravity forces, the gravity acceleration will be subtracted from the translationsl acceleration of the base such that:

$${}^0\dot{\mathbf{V}}_0 = {}^0\dot{\mathbf{V}}_b - \mathbf{g}, \quad {}^0\dot{\omega}_0 = {}^0\dot{\omega}_b, \quad {}^0\omega_0 = {}^0\omega_b$$

b By studying the equilibrium of each link, the backward recursive calculation, for $j = N_L, \ldots, 1$, gives:

$${}^j\mathbf{f}_j = {}^j\mathbf{F}_j + \sum_k {}^j\mathbf{f}_k + {}^j\mathbf{f}_{e,j} \tag{17}$$

$${}^i\mathbf{f}_j = {}^i\mathbf{A}_j{}^j\mathbf{f}_j \tag{18}$$

$${}^j\mathbf{m}_j = {}^j\mathbf{M}_j + \sum {}^j\mathbf{A}_k{}^k\mathbf{m}_k + \sum {}^j\mathbf{P}_k \times {}^j\mathbf{f}_k + {}^j\mathbf{m}_{e,j} \tag{19}$$

projecting ${}^i\mathbf{f}_j$ or ${}^j\mathbf{m}_j$ on the point axis and taking into account the inertia and frictions of the actuators, we get the motor torque as:

$$\Gamma_j = (\sigma_j{}^j\mathbf{f}_j + \bar{\sigma}_j{}^j\mathbf{m}_j)^{Tj}\mathbf{a}_j + Ia_j\ddot{q}_j + F_{vj}\dot{q}_j + F_{sj}\,\mathrm{sign}\,(\dot{q}_j) \tag{20}$$

the sum sign in equations (17) and (19) is on k, which denote all the links successors of j such that $a(k) = j$.

For serial robot, the maximum number of operations of this algorithm is equal to $92N_L$-127 multiplications and $81N_L$-117 additions, for $N_L = 6$ these relations give 425 multiplications and 369 additions. This number will be further reduced for real robots, where many of the geometric parameters of distances are zero and those of angles are equal to $k\pi/2$, with k integer, and in particularly if the links are supposed symmetric. Table II gives the number of operations for the 6 d.o.f. PUMA robot (using base inertial parameters as defined in Section 6.4).

5.2 Dynamic model of closed loop robots

The joint positions of the equivalent tree structure of a closed loop robot can be written as:

$$\mathbf{q}_{tr} = \begin{bmatrix} \mathbf{q}_a \\ \mathbf{q}_p \end{bmatrix} \tag{21}$$

where $\mathbf{q}_a$, $\mathbf{q}_p$ represent the positions of active and passive joints respectively.

The dynamic model of a closed-loop structured robot can be obtained as a function of the corresponding tree structure dynamic model using the following relation:[13,14]

$$\Gamma_{\mathbf{m}} = \mathbf{G}^T\Gamma_{\mathbf{tr}} = \Gamma_a + \mathbf{W}^T\Gamma_p \tag{22}$$

where:

$\Gamma_{\mathbf{m}}$ is the $(m \times 1)$ vector of the torques of motorized joints,

$\mathbf{G} = (\partial\mathbf{q}_{tr}/\partial\mathbf{q}_a)$ is the Jacobian matrix of $\mathbf{q}_{tr}$ with respect to $\mathbf{q}_a$,

$\mathbf{W}$ is defined in equation (13).

$\Gamma_{\mathbf{tr}}$ is the $(N_L \times 1)$ vector of the joint torques (or forces) of the corresponding tree structure, which is supposed as

$$\Gamma_{tr} = \begin{bmatrix} \Gamma_a \\ \Gamma_p \end{bmatrix}$$

where Γ_a and Γ_p denote the vectors of the torques corresponding to active and passive joints respectively.

$\Gamma_{\mathbf{tr}}$ can be obtained using Newton-Euler algorithm as given in Section 5.1, while $\mathbf{G}$ can be obtained from the resolution of kinematic constraint equations.

Table II. Number of operations of the dynamic models of the 6.d.o.f PUMA robot

	Complete parameters		Symmetric links	
	Additions	Multiplications	Additions	Multiplications
I.D.M (N-E)	265	271	142	185
Inertia matrix	161	208	96	156
H	246	260	125	174
D.D.M	713	961	512	716
Lagrange Coeff.	428	687	247	495

5.3 Direct dynamic model (D.D.M)

The direct dynamic model is used to simulate the robot dynamics, it permits to calculate the joint accelerations as a function of the input torques or forces. Since the general form of the inverse dynamic model can be written as:

$$\Gamma = \mathbf{A}\ddot{\mathbf{q}} + \mathbf{B}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{Q} \tag{23}$$

where:

A is the inertia matrix of the robot,
B(**q**, $\dot{\mathbf{q}}$)$\dot{\mathbf{q}}$ gives the Coriolis and centrifugal forces,
Q is the gravity forces.

The direct dynamic model can be obtained as follows:

$$\ddot{\mathbf{q}} = \mathbf{A}^{-1}[\Gamma - \mathbf{H}] \tag{24}$$

where:

$$\mathbf{H} = \mathbf{B}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{Q} \tag{25}$$

Thus the calculation of **A**, and **H** represent the direct dynamic model.

5.3.1 Calculation of the vector H. The vector **H** can be calculated by the Newton-Euler method, given in Section 5.1 by noting that, see equation (23), it is equal to Γ under the condition that $\ddot{\mathbf{q}} = \mathbf{0}$. To calculate **H** efficiently a separate function is given to calculate it using a customized symbolic calculation. Table II gives the number of operations for calculating **H** for the PUMA robot (using base inertial parameters).

5.3.2 Calculation of the inertia matrix A. The calculation of the matrix **A** is given by an algorithm similar to the algorithm of Newton-Euler.[15] By using the fact that the j^{th} column of **A** is equal to Γ if:

$$\dot{\mathbf{q}} = \mathbf{0}, \quad \mathbf{g} = \mathbf{0}, \quad \ddot{\mathbf{q}} = \mathbf{e}_j \tag{26}$$

with $\mathbf{e}_j$ is the unit $(N_L \times 1)$ vector, whose elements are equal to zero except the j^{th} component which is equal to 1.

To increase the efficiency of this method, we make use of the symmetrical property of **A** and we use the generalized link inertial parameters. The generalized link j is defined as the fictitious link composed of link j and all the succeeding links articulated on it.[16] The algorithm is thus composed of two steps, the calculation of the inertial parameters of the generalized links and a customized Newton-Euler algorithm which take into account the conditions of equation (26).

The inertial parameters of the generalized link j are denoted as: M_j^+, ${}^j\mathbf{J}_j^+$, ${}^J\mathbf{MS}_j^+$, they will be calculated by the following algorithm:

Initialization: for $j = 1, \ldots, N_L$:

$${}^j\mathbf{J}_j^+ = {}^j\mathbf{J}_j, \quad {}^j\mathbf{MS}_j^+ = {}^j\mathbf{MS}_j, \quad M_j^+ = M_j$$

for $j = N_L, \ldots, 2$ and $a(j) \neq 0$:

$${}^{a(j)}\mathbf{MS}_j = {}^{a(j)}\mathbf{A}_j\,{}^j\mathbf{MS}_j \tag{27}$$

$$\begin{aligned} {}^{a(j)}\mathbf{J}_{a(j)}^+ = {}^{a(j)}\mathbf{J}_{a(j)}^+ &+ {}^{a(j)}\mathbf{A}_j\,{}^j\mathbf{J}_j\,{}^j\mathbf{A}_{a(j)} \\ &- [{}^{a(j)}\hat{\mathbf{P}}_j\,{}^{a(j)}\mathbf{MS}_j^+ + ({}^{a(j)}\hat{\mathbf{P}}_j\,{}^{a(j)}\mathbf{MS}_j^+)^T] \\ &+ {}^{a(j)}\hat{\mathbf{P}}_j\,{}^{a(j)}\hat{\mathbf{P}}_j^T M_j \end{aligned} \tag{28}$$

$${}^{a(j)}\mathbf{MS}_{a(j)}^+ = {}^{a(j)}\mathbf{MS}_{a(j)}^+ + {}^{a(j)}\mathbf{A}_j\,{}^J\mathbf{MS}_j + {}^{a(j)}\mathbf{P}_j M_j \tag{29}$$

$$M_{a(j)}^+ = M_{a(j)}^+ + M_j \tag{30}$$

Using the conditions (26) in Newton Euler algorithm, the recursive forward calculation will be reduced to:

$${}^j\mathbf{F}_j = \sigma_j[0 \;\; 0 \;\; M_j^+]^T \bar{\sigma}_j[-MY_j^+ + MX_j^+ \;\; 0]^T \tag{31}$$

$$\begin{aligned} {}^j\mathbf{M}_j = \bar{\sigma}_j[XZ_j^+ \;\; YZ_j^+ \;\; ZZ_j^+] \\ + \sigma_j[MY_j^+ \;\; -MX_j^+ \;\; 0]^T \end{aligned} \tag{32}$$

The recursive backward calculation of the I.D.M will be reduced to

$${}^j\mathbf{f}_j = {}^j\mathbf{F}_j \tag{33}$$

$$\begin{aligned} {}^j\mathbf{m}_j = \sigma_j[MY_j^+ \;\; -MX_j^+ \;\; 0]^T \\ + \bar{\sigma}_j[XZ_j^+ \;\; YZ_j^+ \;\; ZZ_j^+]^T \end{aligned} \tag{34}$$

$$A_{j,j} = \sigma_j M_j^+ + \bar{\sigma}_j ZZ_j^+ + Ia_j \tag{35}$$

where $A_{j,j}$ is the (j, j) element of the matrix **A**.

To calculate the other elements of the column $j(A_{a(j),j}, \ldots, A_{sj(0),j})$, where $sj(0)$ indicates the succeeding link of the base on the path between link 0 and link j, we continue the backward calculations which gives the following relations for $k = j, a(j), a(a(j)), \ldots, sj(0)$:

$${}^{a(k)}\mathbf{f}_{a(k)} = {}^{a(k)}\mathbf{A}_k\,{}^k\mathbf{f}_k \tag{36}$$

$${}^{a(k)}\mathbf{m}_{a(k)} = {}^{a(k)}\mathbf{A}_k\,{}^k\mathbf{m}_k + {}^{a(k)}\mathbf{P}_k \times {}^{a(k)}\mathbf{f}_{a(k)} \tag{37}$$

$$A_{a(k),j} = {}^{a(k)}\mathbf{a}_{a(k)}^T({\sigma_j}^{a(k)}\mathbf{f}_{a(k)} + \bar{\sigma}_j\,{}^{a(k)}\mathbf{m}_{a(k)}) \tag{38}$$

$A_{i,j} = 0$ if link i is not on the path between link j and link 0.

Table II, gives the number of operations for calculating the inertia matrix for the PUMA robot (using base inertial parameters).

5.4 Direct dynamic model without calculating the inertia matrix

An algorithm derived from the work in references 17 and 18 is programmed in customized form to get directly the joint accelerations as function of joint positions, velocities and torques without calculating nor inverting the inertia matrix. This algorithm can be used only for open loop robots. Table II gives the number of operations for calculating the **D.D.M** for the PUMA robot (using base inertial parameters).

5.5 Dynamic model using Lagrange equation

This function calculates the elements of the matrices **A**, **B**, and **Q** of the dynamic model defined in equation (23). Two methods have been developed to get the elements of **B**: the first based on differentiating the inertia matrix of the robot obtained in Section 5.3; the other method uses the algorithm developed in reference 19. Table II gives the number of operations for calculating the coefficients of the matrices **A**, **B** and **Q** for the PUMA robot (using base inertial parameters).

6. THE IDENTIFICATION MODEL

In order to use the dynamic model in simulation or control we need to know the values of the dynamic (inertial and friction) parameters. The most appropriate method to evaluate these parameters is the use of the identification techniques. Many identification models which are linear in the dynamic parameters are proposed: the dynamic model, the filtered dynamic model, and the energy model. All these models are generated in SYMORO+ as given in the following sections.

6.1 The dynamic identification model

The dynamic model can be used to identify the inertial parameters.[20–22] Since the dynamic model is linear in the dynamic parameters, as defined in Section 2.2, it can be written as follows:

$$\Gamma = \mathbf{D}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})\mathbf{K} \tag{39}$$

where

K is the $(N_p \times 1)$ vector of dynamic parameters.

D is $(N_L \times N_p)$ matrix.

N_P is the number of inertial and friction parameters.

To identify **K** a sufficient number of equations can be obtained by calculating relation (39) at different times. The least squares solution is generally used in the solution, SYMORO+ gives the symbolic expressions of the elements of the matrix **D** using customized symbolic calculation. The calculation of the column j of the matrix **D** is obtained from the Newton-Euler algorithm of the inverse dynamic model, Section 5.1, with the assumption that the j^{th} inertial parameters, denoted as K_j, is equal to one while all the other inertial parameters are equal to zero. It is to be noted that the calculation of ${}^j\omega_j$, ${}^j\dot{\omega}_j$, ${}^j\dot{\boldsymbol{V}}_j$, ${}^j\mathbf{U}_j$ for $j = 1, \ldots, N_L$ is calculated only once for all the columns.

6.2 The filtered dynamic identification model

The filtered dynamic model is proposed in references 23, 24 and 25 in order to get a model which is not function of the joint accelerations. We have proposed and programmed a new method to get this model in SYMORO+.[26] The main idea of the proposed algorithm is derived from the Lagrangian equation which is given as:

$$\Gamma = \frac{d}{dt}\frac{\partial L}{\partial \dot{\boldsymbol{q}}} - \frac{\partial L}{\partial \mathbf{q}} \tag{40}$$

with:

L the Lagrangien of the system, equal to $E - U$,

E the kinetic energy of the system,

U the potential energy of the system.

Equation (40), can be written as:

$$\Gamma = \frac{d\mathbf{D}_1(\mathbf{q}, \dot{\mathbf{q}})\mathbf{K}}{dt} + \mathbf{D}_2(\mathbf{q}, \dot{\mathbf{q}})\mathbf{K} \tag{41}$$

where

$$\mathbf{D}_1(\mathbf{q}, \dot{\mathbf{q}})\mathbf{K} = \frac{\partial E}{\partial \dot{\boldsymbol{q}}} \tag{42}$$

and

$$\mathbf{D}_2(\mathbf{q}, \dot{\mathbf{q}})\mathbf{K} = -\frac{\partial E}{\partial \mathbf{q}} + \frac{\partial U}{\partial \mathbf{q}} \tag{43}$$

Applying a numerical filter, denoted F, of second order or higher, on the two sides of equation (41) gives:

$$\Gamma_{\mathbf{f}} = \mathbf{D}_{1df}(\mathbf{q}, \dot{\mathbf{q}})\mathbf{K} + \mathbf{D}_{2f}(\mathbf{q}, \dot{\mathbf{q}})\mathbf{K} \tag{44}$$

where the subscript "f" means the application of the filter F, and "df" means the application of a filter F' which associate to the filter F a derivative action. For instance in the case of using second order filter F and F' could be taken as:

$$F(s) = \frac{a^2}{(s+a)^2} \tag{45}$$

$$F'(s) = \frac{a^2 s}{(s+a)^2} \tag{46}$$

SYMORO+ gives the symbolic expressions of the elements of the matrices $\mathbf{D}_1$ and $\mathbf{D}_2$ using customized symbolic calculation using the algorithm presented in reference 26.

6.3 The energy identification model

Using the dynamic model in the identification need to estimate or measure the joint accelerations. To overcome this difficulty a model based on the energy theorem has been proposed.[27]

To simplify the writing we assume that the friction is neglected. From the energy theorem we get:

$$\int_{t_1}^{t_2} \dot{\boldsymbol{q}}^T \Gamma \, dt = H(t_2) - H(t_1) = \Delta\mathbf{h}(\mathbf{q}, \dot{\mathbf{q}})\mathbf{K} \tag{47}$$

where:

$H(t_i)$ is the total energy (kinetic and potential) at time

t_i, it is linear in the inertial parameters and can be written as:

$$H = \mathbf{h}(\mathbf{q}, \dot{\mathbf{q}})\mathbf{K} \tag{48}$$

$\mathbf{h}$ is a row matrix function of $\mathbf{q}$ and $\dot{\mathbf{q}}$.

Since the kinetic and potential energies of link j can be written as:

$$E_j = \tfrac{1}{2}[{}^j\omega_j^{Tj}\mathbf{J}_j{}^j\omega_j + M_j{}^j\mathbf{V}_j^{Tj}\mathbf{V}_j + 2{}^j\mathbf{MS}_j^T({}^j\mathbf{V}_j \times {}^j\omega_j)] \tag{49}$$

$$U_j = -{}^0\mathbf{g}^T({}^0\mathbf{P}_j + {}^0\mathbf{A}_j{}^j\mathbf{MS}_j) \tag{50}$$

where,

${}^0\mathbf{g}$ is the acceleration of gravity referred to frame zero.

${}^0\mathbf{P}_j$ is the position of the origin of frame j w.r.t. frame zero.

${}^0\mathbf{A}_j$ is the 3×3 transformation matrix giving the orientation of frame j w.r.t. frame 0, such that:

$${}^0\mathbf{A}_j = [{}^0\mathbf{s}_j \quad {}^0\mathbf{n}_j \quad {}^0\mathbf{a}_j] \tag{51}$$

with $\mathbf{s}_j$, $\mathbf{n}_j$, $\mathbf{a}_j$ are the unit vectors along the x, y, and z axis, respectively.

The coefficients of the inertial parameters in the vector $\mathbf{h}$ can be given as follows:

$$\begin{aligned}
h_{XX_j} &= \tfrac{1}{2}\omega_{1,j}\omega_{1,j}\\
h_{XY_j} &= \omega_{1,j}\omega_{2,j}\\
h_{XZ_j} &= \omega_{1,j}\omega_{3,j}\\
h_{YY_j} &= \tfrac{1}{2}\omega_{2,j}\omega_{2,j}\\
h_{YZ_j} &= \omega_{2,j}\omega_{3,j}\\
h_{ZZ_j} &= \tfrac{1}{2}\omega_{3,j}\omega_{3,j}\\
h_{MX_j} &= \omega_{3,j}V_{2,j} - \omega_{2,j}V_{3,j} - {}^0\mathbf{g}^{T0}\mathbf{s}_j\\
h_{MY_j} &= \omega_{1,j}V_{3,j} - \omega_{3,j}V_{1,j} - {}^0\mathbf{g}^{T0}\mathbf{n}_j\\
h_{MZ_j} &= \omega_{2,j}V_{1,j} - \omega_{1,j}V_{2,j} - {}^0\mathbf{g}^{T0}\mathbf{a}_j\\
h_{M_j} &= \tfrac{1}{2}{}^j\mathbf{V}_j^{Tj}\mathbf{V}_j - {}^0\mathbf{g}^{T0}\mathbf{P}_j
\end{aligned} \tag{52}$$

where, ${}^j\omega_j$ and ${}^j\mathbf{V}_j$ are represented by the vectors:

$${}^j\omega_j = [\omega_{1,j} \quad \omega_{2,j} \quad \omega_{3,j}]^T \tag{53}$$

$${}^j\mathbf{V}_j = [V_{1,j} \quad V_{2,j} \quad V_{3,j}]^T \tag{54}$$

To identify $\mathbf{K}$ a sufficient number of equations can be obtained by calculating relation (47) between different intervals of time. The least squares solution is generally used in this identification.

SYMORO+ calculates the elements of the matrix $\mathbf{h}$ using customized symbolic calculation.

6.4 The base inertial parameters

The base inertial parameters are defined as the minimum parameters which can be used to get the dynamic model. They represent the set of parameters which can be identified using the dynamic or energy model, thus its determination is essential for the identification of the inertial parameters of robots. They constitute also the parameters to be adapted during an adaptive dynamic control strategy. The use of the base parameters in all the dynamic models presented in Section (5) leads to reduce their computation complexity.[12]

These parameters can be obtained from the classical inertial parameters by eliminating those which have no effect on the dynamic model and by regrouping some others. In references 28–33 we have presented symbolic and numerical methods to calculate these parameters for serial, tree structured, or closed loop robots. These algorithms have been programmed in SYMORO+. The program generates a new file of inertial parameters which can be used directly in the Dynamic or Identification menus.

7. OPTIMIZATION OF THE MODELS

The models obtained under customized symbolic form can be optimized from the number of operations point of view using the optimizer menu. The optimized output file can be given in either of the following format: *Fortran, Mathematica, C, Maple, Matlab,* such that the generated model can be used directly by these systems.

8. CONCLUSION

This paper presents the software package of symbolic modelling of robots SYMORO+. This package generates all the models needed in the simulation, identification, control and design of robots. The algorithms used in deriving these models have been presented, they have been programmed using *MATHEMATICA* system. A friendly user interface is developed in *C*. The paper gives also an overview of the best algorithms which can be used in modelling the robots. Current extension is concerned with the modelling of flexible robots[34] structure and walking robots.

Acknowledgement

This work has been supported by the "CRITT Productique de Pays de Loire", I would like to thank also all the members of the robotics team of the "Laboratoire d'Automatique de Nantes" who have participated in this project, either by developing new algorithms in the modelling of robots or by testing this package. Particular thanks to C. Chevallereau, F. Bennis, Ph. Seigle, P. Restrepo, and D. Murareci who have been involved in programming some modules of SYMORO+.

References

1. S. Wolfram, *Mathematica, A System For Doing Mathematics* (Addison Wesley Publishing Company, Reading, Mass., 1992).
2. W. Khalil and J.-F. Kleinfinger, "A new geometric notation for open and closed-loop robots" *Proc. IEEE Conf. on Robotics and Automation,* San Francisco (1986) pp. 1174–1180.
3. D.L. Pieper, "The kinematics of manipulators under computer control" *Ph. D. Thesis* (Stanford University, 1968).
4. F. Bennis and W. Khalil, "Modèle géométrique inverse des robots à structure découplable: Application à la résolution des équations de contrainte des boucles fermées" *CSME FORUM SCGM,* Montréal (1992) pp. 577–582.
5. R.C.P. Paul, *Robot Manipulators*: *mathematics, programming and control* (MIT Press, Boston, Mass., 1981).
6. W. Khalil, "On the explicit derivation of the inverse geometric models of robots" *Proc. IMACS-IFAC Symp., Villeneuve D'Ascq* (1986) pp. 541–546.
7. M. Raghavan and B. Roth, "Kinematic analysis of the 6R

manipulator of general geometry'' *Proc. of the 5th Int. Symp. on Robotics Research,* MIT Press, (1990) pp. 263–270.
8. W. Khalil and D. Murareci, ''On the general solution of the inverse kinematics of six-degrees-of-freedom manipulators'' *Proc. Int. Workshop on Advances in Robot Kinematics, ARK 94* (1994) pp. 309–318.
9. M. Renaud, ''Calcul de la matrice jacobienne nécessaire à la commande coordonnée d'un manipulateur'' *J. of Mechanism Machine Theory* **15**(1), 81–91 (1980).
10. W. Khalil and C. Chevallereau, ''An efficient algorithm for the dynamic control of robots in the cartesian space'' *26th IEEE CDC,* Los Angeles (1987) pp. 582–587.
11. W. Zghaib, ''Génération Symbolique Automatique des Equations de la Dynamique des Systèmes mécaniques Complexes avec Contraintes Cinématiques'' *Ph.D thesis* (ENSAM, Paris, 1992).
12. W. Khalil and J.-F. Kleinfinger, ''Minimum operations and minimum parameters of the dynamic model of tree structure robots'' *IEEE J. of Robotics and Automation* **RA-3**(6), 517–526 (1987).
13. J.-F. Kleinfinger and W. Khalil, ''Dynamic modelling of closed-chain Robots'', *Proc 16th Int. Symp. on Industrial robots, Bruxelles* (1986) pp. 401–412.
14. E. Dombre and W. Khalil, *Modélisation et Commande des Robots* (Edition Hermes, Paris, 1988).
15. M.W. Walker and D.E. Orin, ''Efficient dynamic computer simulation of robotics mechanism'' *Trans. of ASME, J. of Dynamic Systems, Measurement, and Control* **104,** 205–211 (1982).
16. W. Khalil, F. Bennis and M. Gautier, ''The use of the generalized links to determine the minimum inertial parameters of robots'' *Journal of Robotic Systems* **7,** No. 2, 225–242 (1990).
17. R. Featherstone, ''The calculation of robot dynamics using articulated body inertias'' *Int. J. Robotics Research* **2,** No. 1, 13–30 (1984).
18. H. Brandl, R. Johanni and M. Otter, ''A very efficient algorithm for the simulation of robots and multibody systems without inversion of the mass matrix'' *IFAC Theory of Robots,* Vienne (1986) pp. 356–362.
19. M. Renaud, ''Iterative analytical computation of the dynamic of a robot manipulator'' *Report No. 86014,* LAAS, Toulouse, France (1986).
20. C.H. An, C.G. Atkeson and J.M. Hollerbach, ''Estimation of inertial parameters of rigid body links of manipulators'' *Proc. 24th Conf. on Decision and Control* (1985) pp. 990–995.
21. P.K. Khosla and T. Kanade, ''Parameter identification of robot dynamics'' *Proc. 24th Conf. on Decision and Control* (1985) pp. 1754–1760.
22. M. Gautier, W. Khalil and P. Restrepo, ''Identification of the dynamic parameters of a closed loop robot'', *IEEE Robotics and Automation Conference Japan* (1995) pp. 3045–3050.
23. R.H. Middleton and G. Goodwin, ''Adaptive computed torque control for rigid link manipulators'' *Proc. IEEE Conf. Decision and Control, Athens* (1986) *pp.* 68–73.
24. P. Hsu, M. Bodson, S. Sastry and B. Paden, ''Adaptive identification and control for manipulators without using joint accelerations *Proc. IEEE RAC, Raleigh* (1987) pp. 1210–1215.
25. C. Canudas de Wit and A. Aubin, ''Parameters identification of robots manipulators via sequential hybrid estimation algorithms'' *Proc. IFAC'90 Congress, Tallin* (1990) pp. 178–183.
26. W. Khalil and P. Restrepo, ''An efficient algorithm for the calculation of the filtered dynamic model of robots'' *IEEE Conference on Robotics and Automation, Minneapolis* (April, 1996) pp. 323–328.
27. M. Gautier and W. Khalil, ''On the identification of the inertial parameters of robots'' *Proc. 27th CDC* (1988) pp. 2264–2269.
28. M. Gautier and W. Khalil, ''A direct determination of minimum inertial parameters of robots'' *Proc. IEEE Conf. on Robotics and Automation, Philadelphia* (1988) pp. 1682–1687.
29. W. Khalil, M. Gautier and F. Bennis, ''Calculation of the minimum inertial parameters of tree structure robots'' *4th ICAR,* Columbus, Ohio, USA (1989) pp. 188–201.
30. F. Bennis and W. Khalil, ''Minimum inertial parameters of robots with parallelogram closed-loops'' *IEEE Conference of Robotics and Automation, Cincinnati* (1990) pp. 1026–1031.
31. M. Gautier, ''Numerical calculation of the base inertial parameters'' *Proc. IEEE Conf. on Robotics and Automation, Cincinnati* (1990) pp. 1020–1025.
32. W. Khalil and F. Bennis, ''Comments on ''Direct Calculation of Minimum Set of Inertial Parameters of Serial Robots'' *IEEE Transactions Robotics and Automation* (1994) pp. 78–79.
33. W. Khalil and F. Bennis, ''Symbolic calculation of the base inertial parameters of closed loop robots'' *Int. J. Robotics Research* **14,** No. 2, 112–128 (1995).
34. W. Khalil and F. Boyer, ''An efficient calculation of the computed torque control of flexible maniplulators'' *IEEE Robotics and Automation Conference* Japan (1995) pp. 609–614.