S Y M O R O+

Symbolic Modeling of Robots

USER'S GUIDE

Wisama KHALIL, Philippe LEMOINE

October 2003

Table of Contents

Appendix 4.1 : General method of calculating the inverse geometric model 4-16

References

Introduction

The software package **SYMORO+** "SYmbolic MOdelling of RObots", generates automatically the symbolic models of robots. It provides to the user efficient tools to generate the different models needed in the control, simulation and design of robots such as "direct geometric model, inverse geometric model, direct kinematic model, inverse kinematic model, direct dynamic model, inverse dynamic model and the dynamic parameters identification model".

The first version of SYMORO was announced on 1986 for PC computers and was written in FORTRAN [Khalil 89a, 89b]. The current version is written in C++ and makes use of the Mathematica® system, from *Wolfram Research Inc.*, which must be installed on your computer to run SYMORO+. The software is developed by the robotics team of the IRCCyN "Institut de Recherche en Communications et Cybérnetique de Nantes - France".

The main characteristics of **SYMORO+** are:

1- It generates under symbolic form the different models of robots.
2- A unified notation is used to describe and model the three types of structures: simple open loop robots, tree structure robots and closed loop robots. Mobile robots and systems with lumped elasticity can be also considered.
3- The programs are interactive and make use of the Windows® (98/Me/NT4/2000/XP/ Vista) environment.
4- The generated dynamic model requires a reduced number of operations for on-line applications, such that it can be used for dynamic control of robots.
5- The maximum number of moving links is 80.
6- The maximum number of joints is 100.
7- General algorithm is provided to get the inverse geometric model.
8- The user can specify the data either symbolically or numerically.
9- The user disposes of all the capacity of Mathematica®, either to exploit the results of SYMORO+ or to program new functions.
10- The optimized programs can be given in FORTRAN, Mathematica, C, Matlab and Maple forms.

This manual is divided into two parts:

The first part, chapters 1,…,4 presents the theoretical basis of the algorithms used in SYMORO+. The user must be familiar with this part in order to use efficiently the different functions of SYMORO+. This part has been mainly inspired from the books of Khalil and Dombre [Khalil 99c, Khalil 02a], and from recent publications of the robotics team of IRCCyN "Institut de Recherche en Communications et Cybernétique de Nantes". In chapter 1, we present the notations used to describe the geometry of the robots. Chapter 2, introduces briefly the direct and inverse geometric and kinematic models. Chapter 3 presents the dynamic model of simple open loop robots and its applications in simulation, identification and control. Chapter 4 presents the dynamic model of tree structure and closed-loop robots.

The second part, Chapter 5, gives the installation instructions and describes the functions of SYMORO+. This part can be found while running SYMORO+ by using the Help functions.

Acknowledgements

We would like to thank all the members of the robotics team of the IRCCyN who have contributed by their research work in developing most of the algorithms presented in SYMORO+ and specially C.Chevallereau, F.Bennis and D.Creusot who have participated in programming this software package.

Nantes, October 2003

PART I

THEORETICAL SUPPORT

Chapter 1

Description of the geometric structure of robots

1.1. Introduction

This chapter presents the notation that are used to define the geometric structure of robots. This method can be used to describe simple open loop (serial), tree structure and closed loop robots [Khalil 85b, Khalil 86a].

It is to be noted that in the case of simple open loop robots our method has all the advantages of the classical Denavit and Hartenberg method [Denavit 55], which is only used for simple open loop robots. At the end of this chapter some examples concerning the description of simple open loop and closed loop robots are given.

1.2. General rules

The method is based on the following general rules:

- link j is denoted L_j,
- the variable of joint j is denoted q_j,
- the links are assumed to be perfectly rigid. They are connected by revolute, R, or prismatic, P, joints. Complex joints such as spherical, cylindrical or universal joints can be represented by an appropriate combination of revolute and prismatic joints and assuming fictitious links with zero masse and length between each two joints. Elasticity can be modeled as joints, torsion spring is considered as revolute joint, and linear spring is supposed as prismatic joint,
- the frame coordinates j is defined fixed with respect to link j,
- the axis $\mathbf{z}_j$ of frame j is along the axis of joint j,
- the parameters defining frame j with respect to its antecedent frame will get the subscript (j).

1.3. Description of simple open loop robots

The system is composed of n+1 links, denoted L_0, …, L_n, and n joints. The link L_0 is the base of the robot while link L_n is the terminal link. Joint j connects link j with link j-1 (figure 1.1).

The frame j coordinates, which is fixed with link j, is defined such that:

- the $\mathbf{z}_j$ axis is along the axis of joint j,
- the $\mathbf{x}_j$ axis is along the common normal between $\mathbf{z}_j$ and $\mathbf{z}_{j+1}$. If the axes $\mathbf{z}_j$ and $\mathbf{z}_{j+1}$ are parallel, the choice of $\mathbf{x}_j$ is not unique.
- the origin O_j is the intersection of $\mathbf{z}_j$ and $\mathbf{x}_j$.

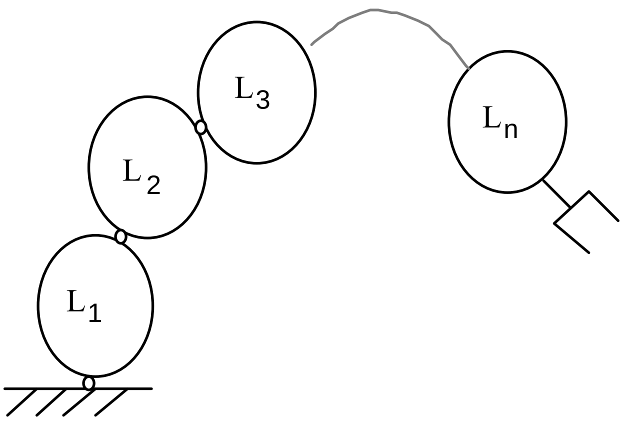


Figure 1.1. Simple open loop robot.

The transformation matrix from frame j-1 to frame j can be obtained as a function of the following four parameters (figure 1.2):

. α_j : angle between $\mathbf{z}_{j-1}$ and $\mathbf{z}_j$ around $\mathbf{x}_{j-1}$,
. d_j : distance between $\mathbf{z}_{j-1}$ and $\mathbf{z}_j$ along $\mathbf{x}_{j-1}$,
. θ_j : angle between $\mathbf{x}_{j-1}$ and $\mathbf{x}_j$ around $\mathbf{z}_j$,
. r_j : distance between $\mathbf{x}_{j-1}$ and $\mathbf{x}_j$ along $\mathbf{z}_j$.

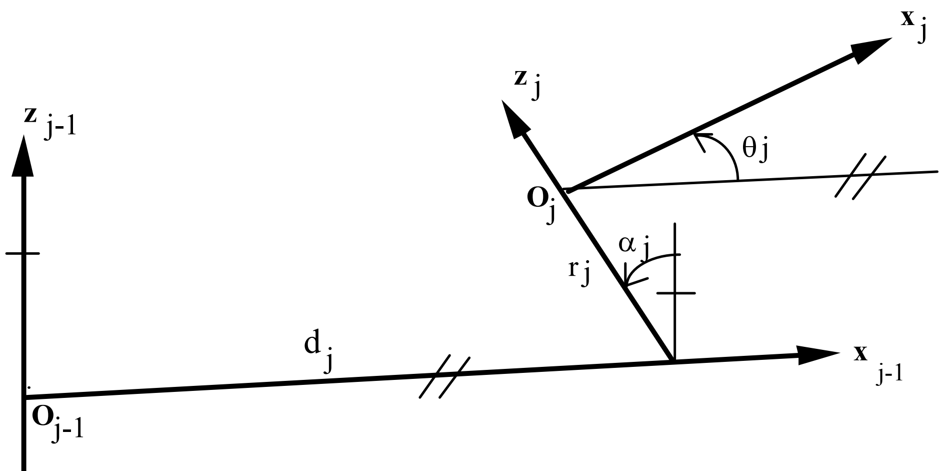


Figure 1.2. Geometric parameters of simple open loop robot.

The joint variable of joint j is θ_j if joint j is revolute or r_j if joint j is prismatic, thus we can write:

$$q_j = \bar{\sigma}_j \theta_j + \sigma_j r_j$$

where:

$\sigma_j = 0$ if joint j is revolute, $\sigma_j = 1$ if joint j is prismatic,

and

$\bar{\sigma}_j = 1$ if joint j is revolute, $\bar{\sigma}_j = 0$ if joint j is prismatic,

Similarly we note:

$$\bar{q}_j = \sigma_j \theta_j + \bar{\sigma}_j r_j$$

The value $\sigma_j = 2$ is used to indicate that frame j is fixed with respect to frame j-1. Thus both θ_j and r_j are constant, and $\bar{\sigma}_j$ is not defined.

The transformation matrix defining frame j with respect to frame j-1 is given as:

$$^{j-1}\mathbf{T}_j = \mathbf{Rot}\,(\mathbf{x}, \alpha_j)\ \mathbf{Trans}\,(\mathbf{x}, d_j)\ \mathbf{Rot}\,(\mathbf{z}, \theta_j)\ \mathbf{Trans}\,(\mathbf{z}, r_j)$$

$$= \begin{bmatrix} C\theta_j & -S\theta_j & 0 & d_j \\ C\alpha_j S\theta_j & C\alpha_j C\theta_j & -S\alpha_j & -r_j S\alpha_j \\ S\alpha_j S\theta_j & S\alpha_j C\theta_j & C\alpha_j & r_j C\alpha_j \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} & {}^{j-1}\mathbf{A}_j & & {}^{j-1}\mathbf{P}_j \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.1}$$

where:

$^{j-1}\mathbf{A}_j$ is the (3x3) matrix defining the orientation of frame j with respect to frame j-1.

$^{j-1}\mathbf{P}_j$ is the (3x1) vector defining the origin of frame j with respect to frame j-1.

C(.) and S(.) denote cos(.) and sin(.) respectively.

The transformation matrix defining frame j-1 with reference to frame j is the inverse of $^{j-1}\mathbf{T}_j$ and is given as:

$$^{j}\mathbf{T}_{j-1} = \mathbf{Trans}\,(\mathbf{z}, -r_j)\ \mathbf{Rot}\,(\mathbf{z}, -\theta_j)\ \mathbf{Trans}\,(\mathbf{x}, -d_j)\ \mathbf{Rot}\,(\mathbf{x}, -\alpha_j)$$

$$= \begin{bmatrix} & & & -d_j C\theta_j \\ & {}^{j-1}\mathbf{A}_j^T & & d_j S\theta_j \\ & & & -r_j \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.2}$$

Remarks:

- frame 0, is chosen to be aligned with frame 1 when $q_1 = 0$. This means that $\mathbf{z}_0$ is aligned with $\mathbf{z}_1$, whereas the origin O_0 is coincident with the origin O_1 if joint 1 is revolute, and $\mathbf{x}_0$ is parallel to $\mathbf{x}_1$ if joint 1 is prismatic. This choice makes $\alpha_1 = 0$, $d_1 = 0$ and $\bar{q}_1 = 0$;

- in a similar way, the choice of the $\mathbf{x}_n$ axis to be aligned with $\mathbf{x}_{n-1}$ when $q_n = 0$ makes $\overline{q}_n = 0$;
- if joint j is prismatic, the $\mathbf{z}_j$ axis must be taken to be parallel to the joint axis but can have any position in space. So, we place it in such a way that $d_j = 0$ or $d_{j+1} = 0$;
- if $\mathbf{z}_j$ is parallel to $\mathbf{z}_{j+1}$, we place $\mathbf{x}_j$ in such a way that $r_j = 0$ or $r_{j+1} = 0$;
- assuming that each joint is driven by an independent actuator, the vector of joint variables $\mathbf{q}$ can be obtained from the vector of encoder readings $\mathbf{q}_c$ using the relation:

  $$\mathbf{q} = \mathbf{K}\,\mathbf{q}_c + \mathbf{q}_0$$

  where $\mathbf{K}$ is an (n×n) constant matrix and $\mathbf{q}_0$ is an offset vector representing the robot configuration when $\mathbf{q}_c = \mathbf{0}$;
- these notations can be used to describe systems with lumped elasticity [Khalil 00], see example 1.5, where a torsional spring is considered as a revolute joint and a linear spring is considered as a prismatic joint.
- these notations can also be used to model mobile robots, where the motion of the robot in space is represented by virtual joints with zero-length massless links, see example 1.6 .

1.4. Description of tree structure robots

A tree structure robot is composed of n mobile links and n joints. The joints are either revolute or prismatic and assumed to be ideal (no backlash, no elasticity). A complex joint can be represented by an equivalent combination of revolute and prismatic joints with zero-length massless links.

The links are numbered consecutively from the base to the terminal links. Thus, link 0 is the fixed base and link n is one of the terminal links (figure 1.3). Joint j connects link j to link a(j), where a(j) denotes the link antecedent to link j, and consequently a(j) < j. We define a main branch as the set of links on the path between the base and a terminal link. Thus, a tree structure has as many main branches as the number of terminal links.

The topology of the system is defined by a(j) for j = 1, …, n. In order to compute the relationship between the locations of the links, we attach a frame R_i to each link i such that:

- $\mathbf{z}_i$ is along the axis of joint i;
- $\mathbf{x}_i$ is taken along the common normal between $\mathbf{z}_i$ and one of the succeeding joint axes, which are fixed on link i. Figure 1.4 shows the case where links j and k are articulated on link i.

Two cases are considered for computing the transformation matrix ${}^{i}\mathbf{T}_j$, which defines the location of frame j relative to frame i with i = a(j):

1) if $\mathbf{x}_i$ is along the common normal between $\mathbf{z}_i$ and $\mathbf{z}_j$, then ${}^{i}\mathbf{T}_j$ is the same as the transformation matrix between two consecutive frames of serial structure. It is obtained as a function of the four geometric parameters (α_j, d_j, θ_j, r_j) (equation (1.1)).

2) if $\mathbf{x}_i$ is not along the common normal between $\mathbf{z}_i$ and $\mathbf{z}_j$, then the matrix ${}^i\mathbf{T}_j$ must be defined using six geometric parameters. This case is illustrated in Figure 1.4, where $\mathbf{x}_i$ is along the common normal between $\mathbf{z}_i$ and $\mathbf{z}_k$. To obtain the six parameters defining frame j relative to frame R_i, we define $\mathbf{u}_j$ along the common normal between $\mathbf{z}_i$ and $\mathbf{z}_j$. The transformation from frame i to frame j can be obtained as a function of six parameters (γ_j, b_j, α_j, d_j, θ_j, r_j):

- γ_j: angle between $\mathbf{x}_i$ and $\mathbf{u}_j$ about $\mathbf{z}_i$
- b_j: distance between $\mathbf{x}_i$ and $\mathbf{u}_j$ along $\mathbf{z}_i$
- α_j: angle between $\mathbf{z}_i$ and $\mathbf{z}_j$ about $\mathbf{u}_j$
- d_j: distance between $\mathbf{z}_i$ and $\mathbf{z}_j$ along $\mathbf{u}_j$
- θ_j: angle between $\mathbf{u}_j$ and $\mathbf{x}_j$ about $\mathbf{z}_j$
- r_j: distance between $\mathbf{u}_j$ and $\mathbf{x}_j$ along $\mathbf{z}_j$

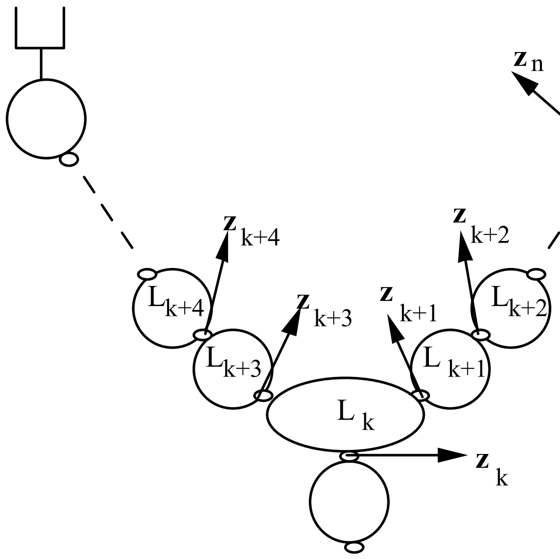


Figure 1.3. Tree structure robot.

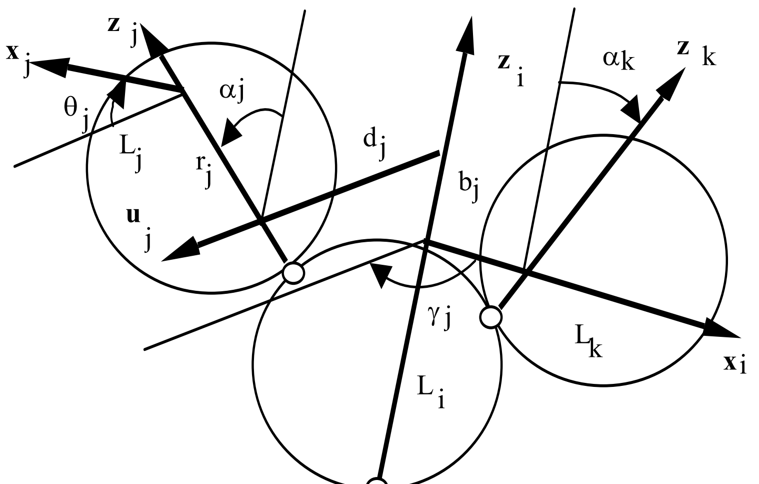


Figure 1.4. Geometric parameters for link having more than two joints.

The transformation matrix ${}^{i}\mathbf{T}_{j}$ is obtained as:

$${}^{i}\mathbf{T}_{j} = \mathbf{Rot}\,(\mathbf{z}, \gamma_j)\ \mathbf{Trans}(\mathbf{z}, b_j)\ \mathbf{Rot}\,(\mathbf{x}, \alpha_j)\ \mathbf{Trans}\,(\mathbf{x}, d_j)\ \mathbf{Rot}\,(\mathbf{z}, \theta_j)\ \mathbf{Trans}\,(\mathbf{z}, r_j)$$

$${}^{i}\mathbf{T}_{j} = \begin{bmatrix} C\gamma_jC\theta_j - S\gamma_jC\alpha_jS\theta_j & -C\gamma_jS\theta_j - S\gamma_jC\alpha_jC\theta_j & S\gamma_jS\alpha_j & d_jC\gamma_j + r_jS\gamma_jS\alpha_j \\ S\gamma_jC\theta_j + C\gamma_jC\alpha_jS\theta_j & -S\gamma_jS\theta_j + C\gamma_jC\alpha_jC\theta_j & -C\gamma_jS\alpha_j & d_jS\gamma_j - r_jC\gamma_jS\alpha_j \\ S\alpha_jS\theta_j & S\alpha_jC\theta_j & C\alpha_j & r_jC\alpha_j + b_j \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.3}$$

The matrix ${}^{j}\mathbf{T}_{i}$ which is the inverse of ${}^{i}\mathbf{T}_{j}$ is given as:

$${}^{j}\mathbf{T}_{i} = \begin{bmatrix} & & & -b_jS\alpha_jS\theta_j - d_jC\theta_j \\ & {}^{i}\mathbf{A}_{j}^{T} & & -b_jS\alpha_jC\theta_j + d_jS\theta_j \\ & & & -b_jC\alpha_j - r_j \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.4}$$

It is to be noted that equation (1.1) is a special case of equation (1.3) with $\gamma_j = 0$ and $b_j = 0$.

1.5. Description of closed loop structure robots

A closed chain structure consists of a set of rigid links connected to each other with joints where at least one closed loop exists. This structure enhances both the accuracy and the load-carrying capacity of the robot. The system is composed of L joints and n + 1 links, where link 0 is the fixed base and L > n. It may contain several terminal links. The number of independent closed loops is equal to:

$$B = L - n \tag{1.5}$$

The joints are either active or passive. The N active joints are provided with actuators. We assume that the number of actuated joints is equal to the number of degrees of freedom of the robot. Thus, the position and orientation of all the links can be determined as a function of the active joint variables.

We introduce the parameter μ_j such that:

- $\mu_j = 1$ if joint j is actuated (active joint);
- $\mu_j = 0$ if joint j is non-actuated (passive joint).

To determine the geometric parameters of a mechanism with closed chains, we proceed as follows:

a) construct an equivalent tree structure having n joints by virtually cutting each closed chain at one of its passive joints. Since a closed loop contains several passive joints, select the joint to be cut in such a way that the difference between the number of links of the two branches from the root of the loop[1] to the links connected to the cut joint is as small as possible. This choice reduces the computational complexity of the dynamic model [Kleinfinger 86a]. The geometric parameters of the equivalent tree structure are determined as described in the previous section;

b) the cut joints are numbered from n+1 to L. For each cut joint k, assign a frame R_k fixed on one of the links connected to this joint, for instance link j. The $\mathbf{z}_k$ axis is taken along the axis of joint k, and the $\mathbf{x}_k$ axis is aligned with the common normal between $\mathbf{z}_k$ and $\mathbf{z}_j$ (Figure 1.5). Let i = a(k) where link i denotes the other link of joint k. The transformation matrix from frame i to frame k can be obtained as a function of the usual six (or four) geometric parameters γ_k, b_k, α_k, d_k, θ_k, r_k, where q_k is equal to θ_k or r_k;

c) since frame k is fixed on link j, the transformation matrix between frames j and k is constant. To avoid any confusion, this transformation will be denoted by ${}^{j}\mathbf{T}_{k+B}$, with j = a(k+B). The geometric parameters defining this transformation will have as a subscript k + B. Note that frame k+B is aligned with frame k, and that both r_{k+B} and θ_{k+B} are zero.

In summary, the geometric description of a structure with closed loops is defined by an equivalent tree structure that is obtained by cutting each closed loop at one of its joints and by adding two frames at each cut joint. The total number of frames is equal to n + 2B and the geometric parameters of the last B frames are constants.

The (Lx1) joint variables vector **q** is written as:

$$\mathbf{q} = \begin{bmatrix} \mathbf{q}_a \\ \mathbf{q}_p \\ \mathbf{q}_c \end{bmatrix} \tag{1.6}$$

with:

- $\mathbf{q}_a$: vector containing the N active joint variables;
- $\mathbf{q}_p$: vector containing the $p = n - N$ passive joint variables of the equivalent tree structure;
- $\mathbf{q}_c$: vector containing the B variables of the cut joints. When a cut joint has several degrees of freedom (spherical, universal, …), we can consider all its joint variables to be belonging to $\mathbf{q}_c$.

[1] The root of a loop is the first common link when going from the links of the cut joint to the base of the robot.

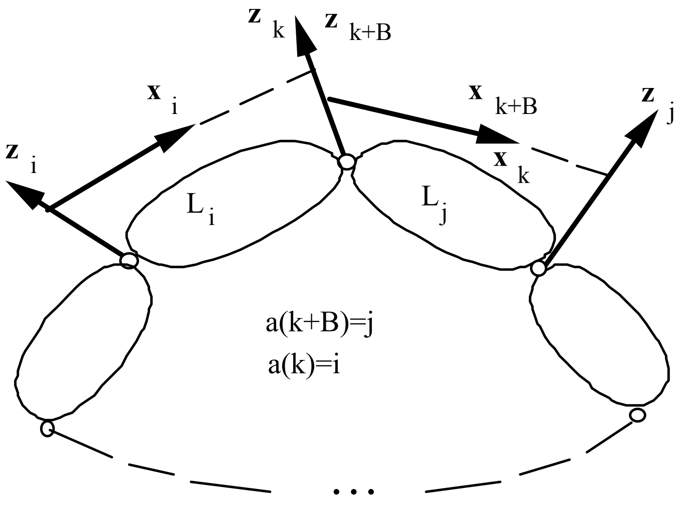


Figure 1.5. Frames on a cut joint.

Only the N active variables $\mathbf{q}_a$ are independent. Thus, there are $c = L - N$ independent constraint equations between the joint variables $\mathbf{q}$. These relations form the geometric constraint equations satisfying the closure of the loops. Since R_k and R_{k+B} are aligned, the geometric constraint equations for each loop can be written as:

$$ {}^{k+B}\mathbf{T}_j \ldots {}^{i}\mathbf{T}_k = \mathbf{I}_4 \tag{1.7}$$

where $\mathbf{I}_4$ is the (4x4) identity matrix.

For a spatial loop, the maximum number of independent geometric constraint equations is six, while for a planar loop this number reduces to three. The geometric constraint equations can be obtained from relation (1.7). They are represented by the nonlinear equation:

$$\boldsymbol{\phi}(\mathbf{q}) = \begin{bmatrix} \phi_1(\mathbf{q}) \\ \phi_2(\mathbf{q}) \\ \ldots \\ \phi_{(L-N)}(\mathbf{q}) \end{bmatrix} = \mathbf{0}_{(L-N)x1} \tag{1.8}$$

1.6. Transformation matrix between two frames

The transformation matrix from frames r1 to frame rc along the path: r1, r2,…, rc, can be obtained as:

$$ {}^{r1}\mathbf{T}_{rc} = {}^{r1}\mathbf{T}_{r2}\, {}^{r2}\mathbf{T}_{r3} \ldots {}^{rc-1}\mathbf{T}_{rc} \tag{1.9}$$

1.7. Examples

Example 1.1: Description of the Stanford manipulator

The first three links are of RRP structure while the last three constitute a spherical joint. The definition of the link frames are given in figure 1.6, the corresponding geometric parameters are given in table 1.1, note that Pi/2 means π/2 in Mathematica language.

j	σ_j	α_j	d_j	θ_j	r_j
1	0	0	0	t1	0
2	0	-Pi/2	0	t2	RL2
3	1	Pi/2	0	0	R3
4	0	0	0	t4	0
5	0	-Pi/2	0	t5	0
6	0	Pi/2	0	t6	0

Table 1.1. The geometric parameters of the Stanford manipulator.

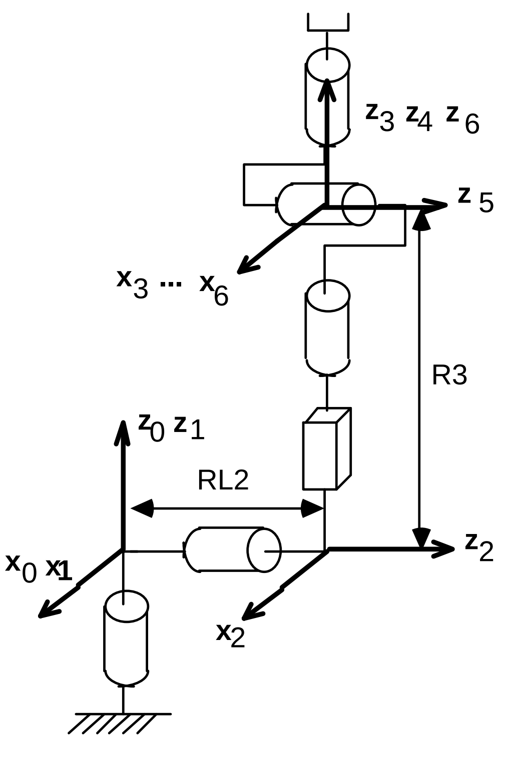


Figure 1.6. Stanford Manipulator.

Example 1.2: Description of the Stäubli-RX90 robot.

The frames of this six degrees of freedom robot are given in figure 1.7, and the parameters are shown in table 1.2 .

j	σ_j	α_j	d_j	θ_j	R_j
1	0	0	0	t1	0
2	0	Pi/2	0	t2	0
3	0	0	D3	t3	0
4	0	–Pi/2	0	t4	RL4
5	0	Pi/2	0	t5	0
6	0	–Pi/2	0	t6	0

Table 1.2. The geometric parameters of the Stäubli-RX90 robot.

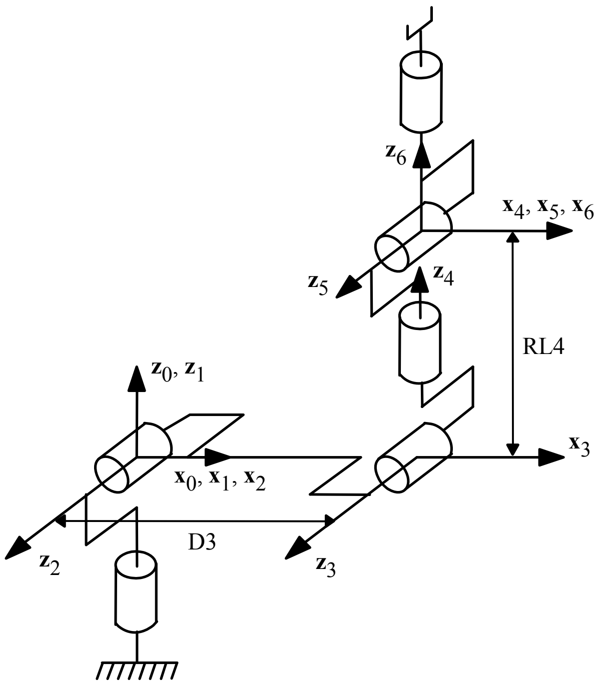


Figure 1.7. The Stäubli-RX90 robot.

Example 1.3: Description of the SR400 robot.

This robot has six degrees of freedom (figure 1.8), eight moving links and nine rotational joints. It has one closed loop of parallelogram type. The loop is supposed to be cut on joint 9, between links 8 and 3. Table 1.3 gives the geometric parameters of this robot.

j	a(j)	μ_j	σ_j	γ_j	b_j	α_j	d_j	θ_j	r_j
1	0	1	0	0	0	0	0	t1	0
2	1	1	0	0	0	–Pi/2	d2	t_2	0
3	2	0	0	0	0	0	d3	t_3	0
4	3	1	0	0	0	– Pi/2	d_4	t_4	RL4
5	4	1	0	0	0	Pi/2	0	t_5	0
6	5	1	0	0	0	– Pi/2	0	t_6	0
7	1	1	0	0	0	– Pi/2	d2	t_7	0
8	7	0	0	0	0	0	d_8	t_8	0
9	8	0	0	0	0	0	$d_9=d_3$	t_9	0
10	3	0	2	Pi/2	0	0	$d_{10}=-d_8$	0	0

Table 1.3. Geometric parameters of the SR400 robot.

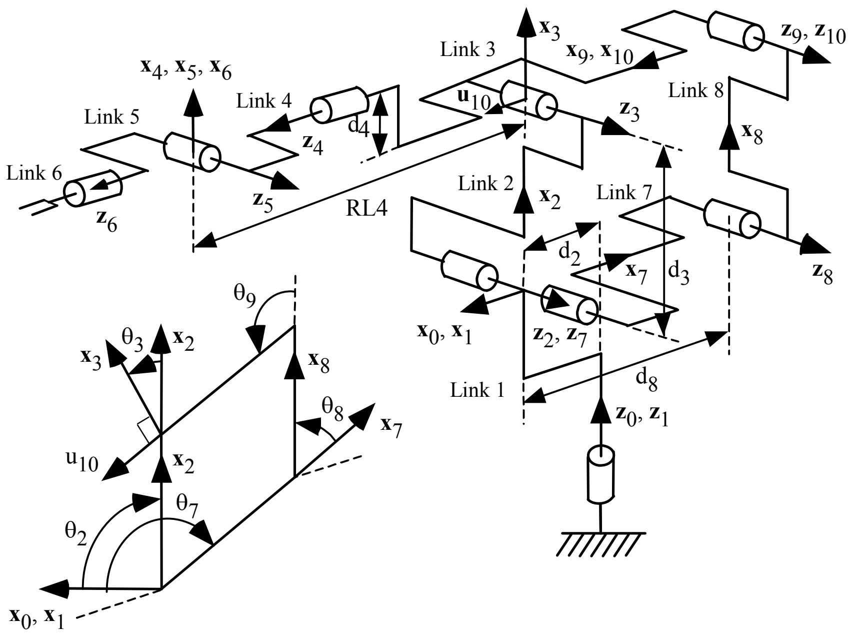


Figure 1.8. The SR400 robot.

Example 1.4: Description of the AKR-3000 robot.

The AKR robot is a six degrees of freedom structure (figure 1.9). It consists of ten moving links and twelve joints. It has two closed loops. The loops are supposed opened between links 5 and 7, and between links 2 and 6. The joints 2, 3, 4, 7, 11 and 12 are passive. The geometric parameters are given in table 1.4.

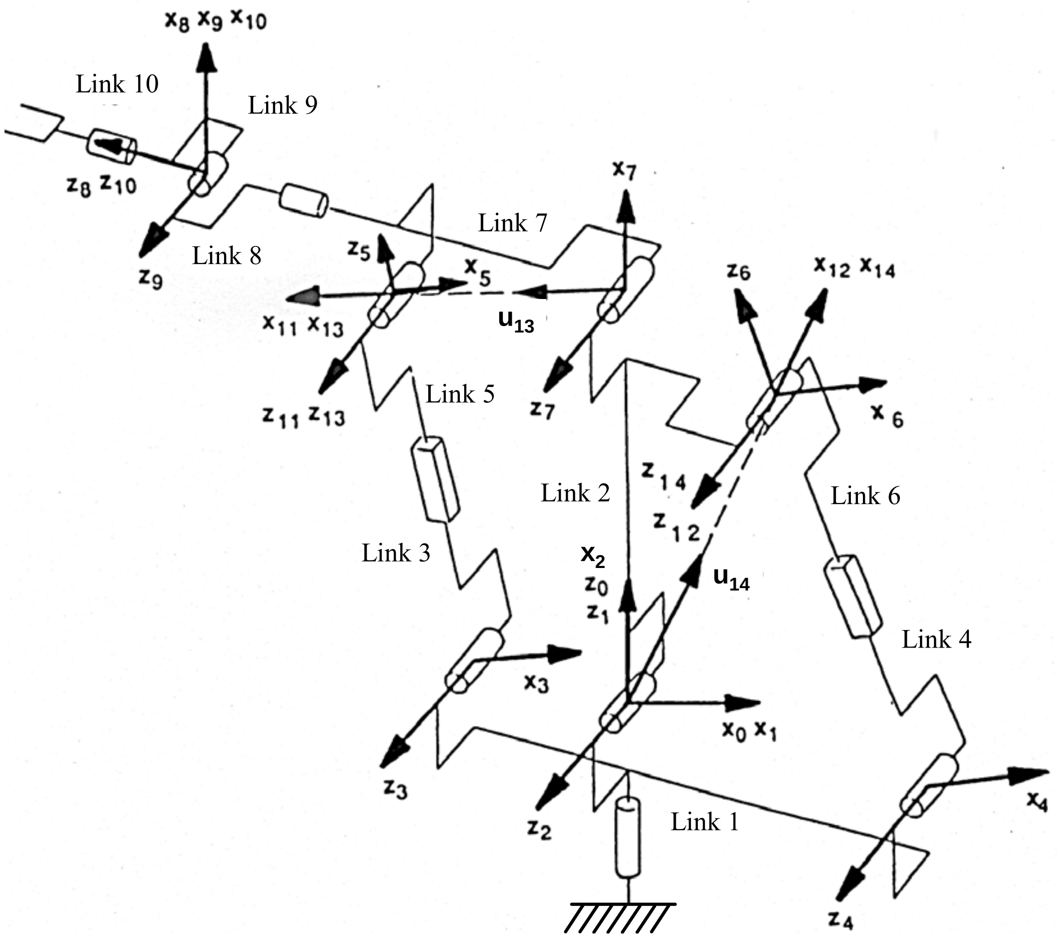


Figure 1.9. AKR 3000 robot.

j	a(j)	μj	σj	γj	bj	αj	dj	θj	rj
1	0	1	0	0	0	0	0	t1	0
2	1	0	0	0	0	Pi/2	0	t2	0
3	1	0	0	0	0	Pi/2	d3	t3	0
4	1	0	0	0	0	Pi/2	d4	t4	0
5	3	1	1	0	0	– Pi/2	0	0	r5
6	4	1	1	0	0	– Pi/2	0	0	r6
7	2	0	0	0	0	0	d7	t7	0
8	7	1	0	0	0	– Pi/2	0	t8	r8
9	8	1	0	0	0	Pi/2	0	t9	0
10	9	1	0	0	0	– Pi/2	0	t10	0
11	5	0	0	0	0	Pi/2	0	t11	0
12	6	0	0	0	0	Pi/2	0	t12	0
13	7	0	2	g13	0	0	d13	0	0
14	2	0	2	g14	0	0	d14	0	0

Table 1.4. Geometric parameters of the AKR-3000 robot.

Example 1.5: Description of a two degrees of freedom robot with lumped flexibility

As an example of a system with lumped elasticity, we consider the two axis high-speed machine shown in Figure 1.10 [Khalil 00]. The first axis is actuated using two motors that are mounted like a gantry type, which are acting as only one equivalent motor. The second axis is perpendicular to the first one, it is actuated with a classical synchronous motor, a gearbox, and gearwheel. We note that the elastic degree of freedom is considered as motorized articulation (μ_j=1). The corresponding motor torque is taken as ($-K_j$ q_j), with K_j is the joint stiffness and q_j is the joint variable. The direct dynamic model of this system can be obtained by the function “**direct dynamic model**” of the menu **dynamic.**

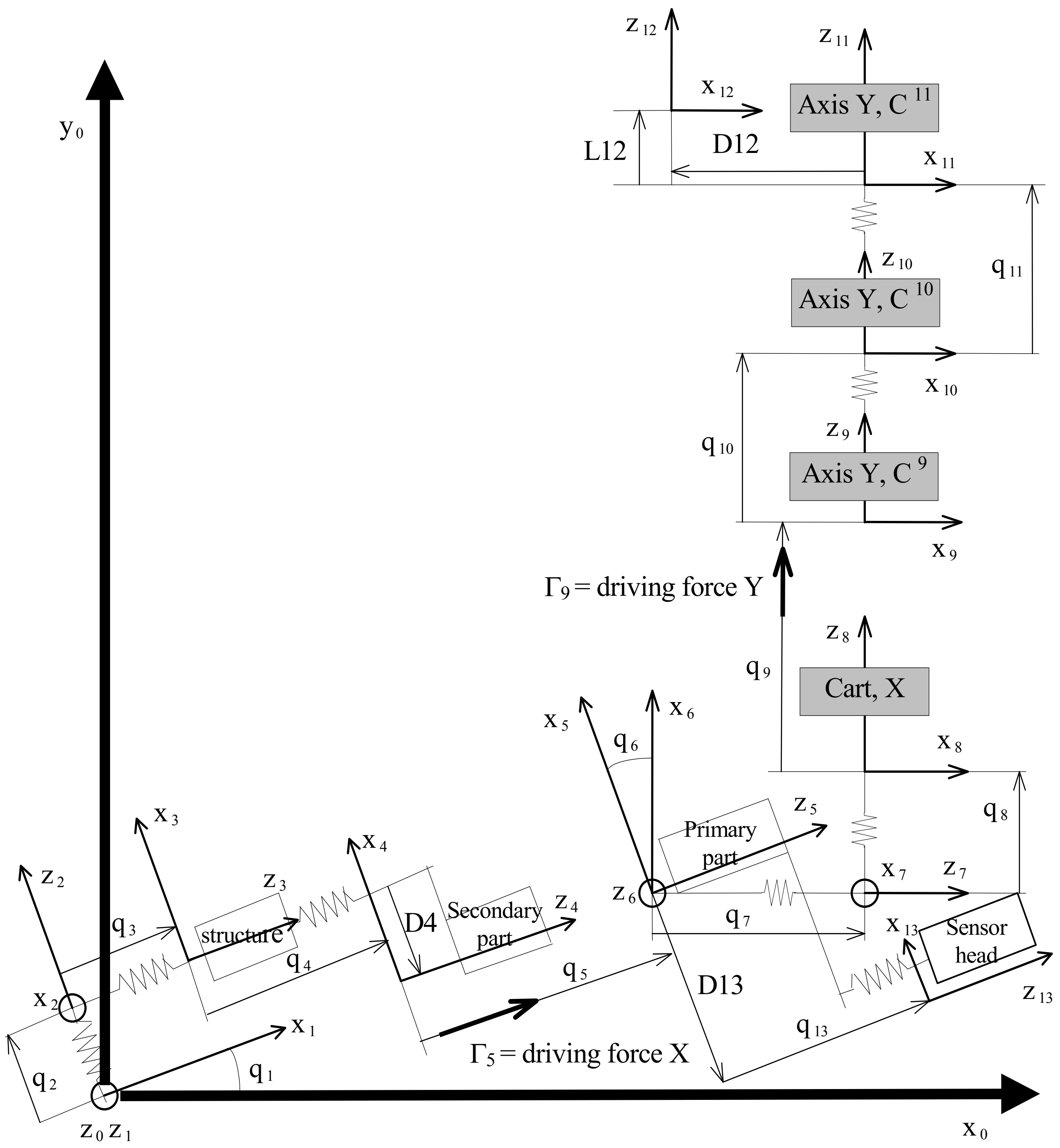


Figure 1.10. Description of a 2 axes high-speed machine.

The geometric parameters are given in table 1.5 .
Each generalized coordinate q_j, j=1,…, 13 represents a degree of freedom (d.o.f) such as:

- j = 1, 2, 3 represent the three elastic degrees of freedom (d.o.f) of the base,
- j = 4 represents the secondary part elastic d.o.f,
- j = 5 represents the rigid primary part d.o.f of axis X,
- j = 6, 7, 8 represent the elastic d.o.f of the cart supporting the axis Y,
- j = 9 represents the rigid d.o.f. of axis Y,
- j = 10, 11 represent the elastic d.o.f of the chain of the Y axis,
- j = 12 is to define a second frame which is attached to Y axis. Thus $\mu_{12} = 0$ and $\sigma_{12} = 2$.

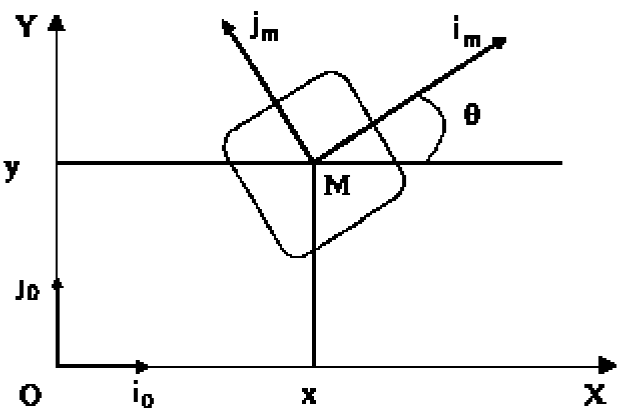
j = 13 represents the elastic d.o.f of the sensor head's scale. Body 5 is defined as the antecedent of links 6 and 13 with a(6) = a(13) = 5 .

j	a(j)	μ_j	σ_j	b_j	γ_j	α_j	d_j	θ_j	r_j
1	0	1	0	0	0	0	0	t1	0
2	1	1	1	0	0	-Pi/2	0	- Pi /2	r2
3	2	1	1	0	0	0	0	0	r3
4	3	1	1	0	0	0	D4	0	r4
5	4	1	0	0	0	0	0	0	r5
6	5	1	0	0	0	- Pi /2	0	t6	0
7	6	1	1	0	0	Pi /2	0	Pi /2	r7
8	7	1	1	0	0	0	0	Pi /2	r8
9	8	1	1	0	0	0	0	0	r9
10	9	1	1	0	0	0	0	0	r10
11	10	1	1	0	0	0	0	0	r11
12	11	0	2	0	0	0	D12	0	L12
13	5	1	1	0	0	0	D13	0	r13

Table 1.5. Geometric parameters of the high-speed machine.

Example 1.6: Geometric parameters of a planar mobile robot

The motion of the robot is supposed in a plane, it is composed of a body and three wheels, only the two rear wheels are motorized. Each of the motorized wheels has one degree of freedom around an axis fixed with the body of the robot (axis z5 and z6 in figure 1.11). They are called fixed wheels. The third, front wheel, is passive and has two degrees of freedom (axis z4 and z7 in figure 1.11), it is called a decentralized (castor) wheel. The robot can be considered as a tree structure robot, the contact with the soil can be represented by external forces on the wheels. The geometric parameters are shown in table 1.6. The first three degrees of freedom represent the motion of the body in the plane, two prismatic and one revolute. The dynamic parameters of the first two degrees of freedom are zero, the body of the robot constitutes the third link.



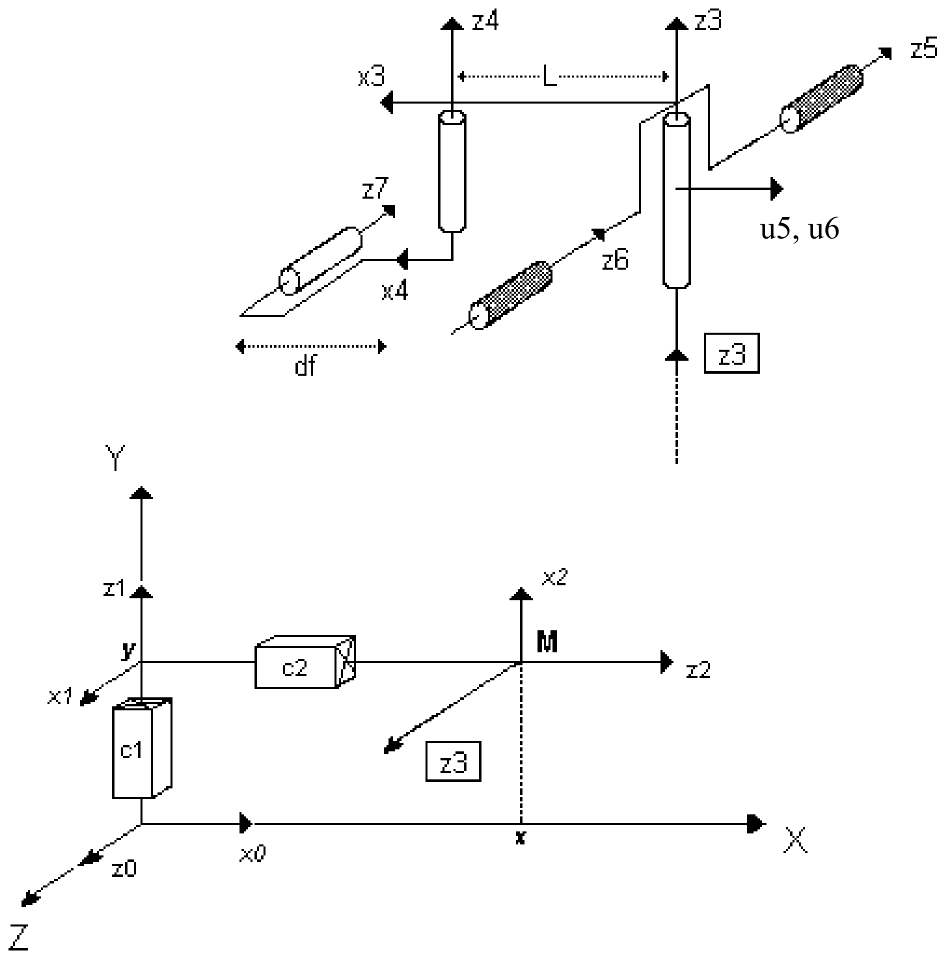


Figure 1.11. A planar mobile robot.

j	a(j)	νj	σj	γj	bj	αj	dj	θj	rj
1	0	1	1	0	0	- Pi/2	0	- Pi /2	y
2	1	1	1	0	0	- Pi /2	0	- Pi /2	x
3	2	1	0	0	0	- Pi /2	0	t3	0
4	3	1	0	0	0	0	L	t4=βd	0
5	3	1	0	Pi	0	Pi /2	0	t5=φ5	0
6	3	0	0	Pi	0	Pi /2	0	t6=φ6	0
7	4	0	0	0	0	Pi /2	df	t7=φ7	0

Table 1.6. Geometric parameters of the planar mobile robot.

1.8. Conclusion

This chapter presented the notation that is used in the description of the geometric structure of robots. This notation can treat the three types of structures: simple open loop, tree structure and closed loop robots. It can also treat systems with lumped flexibility or even mobile robots.

The user of the software package **SYMORO+** must be familiar with this notation in order to define the desired robot.

Chapter 2

Geometric and Kinematic Models

2.1. Introduction

This chapter presents the algorithms used in **SYMORO+** to obtain the geometric and kinematic models of robots. The following models are defined:

- the direct geometric model defining the location (position and orientation) of the end-effector (also called operational coordinates), as a function of the joint positions,
- the inverse geometric model defining the joint variables as a function of a desired end effector location,
- the direct kinematic model defining the velocities of the operational coordinates as a function of the joint velocities,
- the inverse kinematic model defining the joint velocities as a function of the operational coordinates velocities.

2.2. Direct geometric model

2.2.1. The transformation matrix of a robot

The transformation matrix of a robot is the matrix defining the location of the frame of the terminal link with respect to frame 0, which represents the base of the robot. In the case of simple open loop robots this transformation matrix is given as:

$$ {}^{0}\mathbf{T}_{n} = {}^{0}\mathbf{T}_{1}\,{}^{1}\mathbf{T}_{2} \ldots {}^{n-1}\mathbf{T}_{n} = \begin{bmatrix} {}^{0}\mathbf{s}_{n} & {}^{0}\mathbf{n}_{n} & {}^{0}\mathbf{a}_{n} & {}^{0}\mathbf{P}_{n} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.1} $$

with ${}^{0}\mathbf{s}_{n}$, ${}^{0}\mathbf{n}_{n}$, ${}^{0}\mathbf{a}_{n}$, represent the unit vectors along the (**x**, **y**, **z**) axes of frame n projected into frame 0, while ${}^{0}\mathbf{P}_{n}$ defines the origin of frame n with respect to frame 0.

The transformation matrix ${}^{f}\mathbf{T}_{E}$, defining the end-effector frame E with respect to a fixed world frame f, is given as:

$$ {}^{f}\mathbf{T}_{E} = \mathbf{Z}\,{}^{0}\mathbf{T}_{n}\,\mathbf{E} \tag{2.2}$$

where:

. $\mathbf{Z} = {}^{f}\mathbf{T}_{0}$ is the matrix defining the base frame 0 into the reference fixed frame f, it will be denoted also as ${}^{-1}\mathbf{T}_{0}$.

. $\mathbf{E} = {}^{n}\mathbf{T}_{E}$, is the fixed matrix defining the end-effector frame E into the frame of the terminal link n.

A matrix **E** fixed on the terminal link can be defined using the method given in chapter 1 using six constant geometric parameters (γ, b, α, d, θ, r), the corresponding σ will be set equal to 2.

2.2.2. Calculation of the direct geometric model

The direct geometric model is the set of relations from which the location of the end effector (operational coordinates) can be obtained in terms of the joint positions. The location of the end-effector will be defined by the coordinates:

$$\mathbf{X} = [x_1 \quad x_2 \; \dots \; x_m]^T$$

The joint positions will be defined by the joint variables:

$$\mathbf{q} = [q_1 \quad q_2 \dots \; q_n]^T$$

Thus the direct geometric model is written as:

$$\mathbf{X} = \mathbf{f}\,(\mathbf{q}) \tag{2.3}$$

Many possibilities can be used to define the coordinates of the vector **X** [Khalil 02a]. For example, using the Cartesian coordinates for the position and the director cosines to represent the orientation we obtain:

$$\mathbf{X} = [P_x \quad P_y \quad P_z \quad s_x \quad s_y \quad s_z \quad n_x \quad n_y \quad n_z \quad a_x \quad a_y \quad a_z]^T$$

or since **s** can be obtained as the vector product of **n** and **a** the vector **X** can be given as:

$$\mathbf{X} = [P_x \quad P_y \quad P_z \quad n_x \quad n_y \quad n_z \quad a_x \quad a_y \quad a_z]^T$$

For open loop robots the calculation of **X** means the calculation of the transformation matrix of the end-effectors. For robots with closed loops, the relation between the passive joint variables, existing in the robot transformation matrix, and the active variables not taking part in the robot transformation matrix must be also obtained. This can be done by solving the geometric constraint equations of the loops that are defined by:

$$ {}^{k+B}\mathbf{T}_j \;\ldots\; {}^{i}\mathbf{T}_k = \mathbf{I}_4 \qquad (2.4)$$

where frames k and k+B are the frames on each side of the cut joint of the loop, for k = n+1,...,L , where L is the total number of joints.

The vector of the joint variables of the equivalent tree structure system is given as:

$$\mathbf{q}_{tr} = [\ \mathbf{q}_a^T \quad \mathbf{q}_p^T]^T \qquad (2.5)$$

where:

$\mathbf{q}_a$ is the (Nx1) vector of the active joint positions, these joints have $\mu_j = 1$

$\mathbf{q}_p$ is the (n-Nx1) vector of the passive joint positions, these joints have $\mu_j = 0$.

Remark:

For a parallelogram closed loop composed of the links L_{k1}, L_{k2}, L_{k3} and L_{k4}, and supposing that the links L_{k1} and L_{k3} are parallel then the (3x3) orientation matrix between frames k_1 and k_3 is constant. Similarly the rotation matrix between frames k_2 and k_4 is constant. Thus the constraint geometric equations are directly given by the following equations:

$${}^{k1}\mathbf{A}_{k3} = C1$$

and $${}^{k2}\mathbf{A}_{k4} = C2$$

with C1 and C2 are constants, which are determined by using an initial configuration of the robot.

2.2.3. The fast direct geometric model

The control of the robots needs a fast calculation of the models. An efficient method to reduce its computation cost is to use a symbolic customized algorithm that uses intermediate variables and not an expanded symbolic model. Fast direct geometric model means to calculate the transformation matrix between two frames using such customized technique.

2.2.4. The direct geometric model in SYMORO+

The menu geometric of **SYMORO+** computes the transformation matrix ${}^{j}\mathbf{T}_i$ between any two frames i and j on the links of the robot, either in expanded form or using intermediate variables (fast direct geometric model). The definition of the elements of the base matrix **Z** can be carried out on the main page of SYMORO, the reference frame is numbered -1, the definition of **E** can be carried out using the geometric parameters (γ, b, α, d, θ, r) assuming fixed frame with $\sigma = 2$.

The solution of the constraint geometric equation is obtained for closed loop robots in this menu also. This function will give the non-motorized variables, with $\mu = 0$, as a function of the motorized ones, with $\mu = 1$. At the moment the solution is given for loops with three passive joints using the method developed in [Bennis 91b, Bennis 94].

2.3. The inverse geometric model

2.3.1. Introduction

The inverse geometric model, IGM, gives all the robot configurations corresponding to a given location of the end effector (operational coordinates). This model is generally called the closed form inverse kinematic model. Three methods are available in **SYMORO+:**

- the Paul method [Paul 81] that can provide the inverse geometric model of most industrial robots,
- the Pieper method [Pieper 68] which ensures the solution of six degrees of freedom robots provided that they have either three revolute joints forming a spherical joint (their axes are intersecting), or if the robot has three prismatic joints.
- a general method based on the method of Raghavan and Roth [Raghavan 89, Mavroidis 93].

2.3.2. Position of the problem

Let ${}^f\mathbf{T}_E{}^d$ be the desired transformation matrix defining the location of the end-effector with respect to the fixed frame. In general, we can write ${}^f\mathbf{T}_E{}^d$ as:

$$ {}^f\mathbf{T}_E{}^d = \mathbf{Z}\ {}^0\mathbf{T}_n\ \mathbf{E} \tag{2.6} $$

where:
- $\mathbf{Z}$ defines the robot base frame with respect to the fixed frame,
- ${}^0\mathbf{T}_n$ is the robot transformation matrix,
- $\mathbf{E}$ is the transformation matrix defining the end-effector frame $\mathbf{E}$ with respect to the terminal frame n.

Denoting $\mathbf{U}_0 = {}^0\mathbf{T}_n$ and assuming that n = 6, from (2.6) we obtain:

$$ \mathbf{U}_0 = \mathbf{Z}^{-1}\ {}^f\mathbf{T}_E{}^d\ \mathbf{E}^{-1} \tag{2.7} $$

The problem of the inverse geometric model is thus reduced to obtain the joint variables such that ${}^0\mathbf{T}_n$ is equal to $\mathbf{U}_0$.

2.3.3. Calculation of the inverse geometric model by the method of Paul

The transformation matrix of the robot is given as:

$$ {}^{0}\mathbf{T}_{n} = {}^{0}\mathbf{T}_{1}\, {}^{1}\mathbf{T}_{2} \ldots {}^{n-1}\mathbf{T}_{n} \tag{2.8} $$

denoting:

$$ \mathbf{U}_{0} = {}^{0}\mathbf{T}_{1}\, {}^{1}\mathbf{T}_{2} \ldots {}^{n-1}\mathbf{T}_{n} \tag{2.9} $$

and:

$$ \mathbf{U}_{0} = \begin{bmatrix} sx & nx & ax & Px \\ sy & ny & ay & Py \\ sz & nz & az & Pz \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{s} & \mathbf{n} & \mathbf{a} & \mathbf{P} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.10} $$

The matrix $\mathbf{U}_0$ is given: it defines the desired location of the terminal link frame with respect to frame 0.

Computing the joint variables from equation (2.9) as a function of the elements of **s**, **n**, **a**, and **P** is not easy. New set of equations can be obtained by premultiplying successively the two sides of equation (2.9) by the matrices ${}^{j}\mathbf{T}_{j-1}$, for j = 1,…, n–1, this operation permits to isolate and identify the joint variables one after another.

For a six degree of freedom robot we proceed as follows:
- premultiplying the two sides of equation (2.9) by ${}^{1}\mathbf{T}_{0}$ gives:

$$ {}^{1}\mathbf{T}_{0}\,\mathbf{U}_{0} = {}^{1}\mathbf{T}_{2}\, {}^{2}\mathbf{T}_{3}\, {}^{3}\mathbf{T}_{4}\, {}^{4}\mathbf{T}_{5}\, {}^{5}\mathbf{T}_{6} $$

the right hand side is a function of the variables q_2, …, q_6, while the left hand side is a function of q_1.
- q_1 can be obtained by identifying one or two elements of both sides of the previous equation.
- Similarly, from equation (2.9), the following equations are obtained:

$$ \begin{aligned} \mathbf{U}_{0} &= {}^{0}\mathbf{T}_{1}\, {}^{1}\mathbf{T}_{2}\, {}^{2}\mathbf{T}_{3}\, {}^{3}\mathbf{T}_{4}\, {}^{4}\mathbf{T}_{5}\, {}^{5}\mathbf{T}_{6} \\ \mathbf{U}_{1} &= {}^{1}\mathbf{T}_{2}\, {}^{2}\mathbf{T}_{3}\, {}^{3}\mathbf{T}_{4}\, {}^{4}\mathbf{T}_{5}\, {}^{5}\mathbf{T}_{6} \\ \mathbf{U}_{2} &= {}^{2}\mathbf{T}_{3}\, {}^{3}\mathbf{T}_{4}\, {}^{4}\mathbf{T}_{5}\, {}^{5}\mathbf{T}_{6} \\ \mathbf{U}_{3} &= {}^{3}\mathbf{T}_{4}\, {}^{4}\mathbf{T}_{5}\, {}^{5}\mathbf{T}_{6} \\ \mathbf{U}_{4} &= {}^{4}\mathbf{T}_{5}\, {}^{5}\mathbf{T}_{6} \\ \mathbf{U}_{5} &= {}^{5}\mathbf{T}_{6} \end{aligned} \tag{2.11} $$

with $\mathbf{U}_j = {}^{j}\mathbf{T}_{j-1}\,\mathbf{U}_{j-1}$

Remark:

The previous set of equations is called in forward direction. In some cases, it is preferable to begin the solution by identifying at first q_n, and to end with q_1. This can be done by the use of backward direction equations that are obtained by postmultiplying the two sides of (2.9) by ${}^j\mathbf{T}_{j-1}$ for j = n,…, 2.

The solution of a big number of industrial robots revealed that about ten types of equations might be encountered when using this method [Khalil 86b]. These types of equations, see table 2.1, are used by **SYMORO+**, forward equations and backward equations will be used in the case of 6 degree of freedom robots.

2.3.4. Robots with more than six degrees of freedom

When the robot has more than six degrees of freedom, the equations to be solved have more unknowns than the number of parameters defining the desired location. (n–6) supplementary relations must be defined in order to obtain the solution. We have to assume (n–6) joint variables as known. The choice of these joints depends on the task and the structure of the robot, the user can do this by changing the corresponding σ into 2 to indicate that these frames are fixed (known).

Type 1	$X\ r_i = Y$
Type 2	$X\ S\theta_i + Y\ C\theta_i = Z$
Type 3	$X1\ S\theta_i = Y1$ $X2\ C\theta_i = Y2$
Type 4	$X1\ r_j\ S\theta_i = Y1$ $X2\ r_j\ C\theta_i = Y2$
Type 5	$X1\ S\theta_i = Y1 + Z1\ r_j$ $X2\ C\theta_i = Y2 + Z2\ r_j$
Type 6	$W\ S\theta_j = X\ C\theta_i + Y\ S\theta_i + Z1$ $W\ C\theta_j = X\ S\theta_i - Y\ C\theta_i + Z2$
Type 7	$W1\ C\theta_j + W2\ S\theta_j = X\ C\theta_i + Y\ S\theta_i + Z1$ $W1\ S\theta_j - W2\ C\theta_j = X\ S\theta_i - Y\ C\theta_i + Z2$
Type 8	$X\ C\theta_i + Y\ C(\theta_i + \theta_j) = Z1$ $X\ S\theta_i + Y\ S(\theta_i + \theta_j) = Z2$
Type 9	$X\ C\theta_i = Y$
Type 10	$X\ S\theta_i = Y$

Table 2.1. Type of equations used in Paul's method.

2.3.5. Robots having less than six degrees of freedom

If the number of joints is less than six, the dimension of the robot working space is less than six. It is not possible to place the end-effector frame E at an arbitrarily location ${}^{f}\mathbf{T}_{E}{}^{d}$, except in some particular locations corresponding to the reduced working space. In general, only some elements of ${}^{n}\mathbf{T}_{E}$ can be specified thus equation (2.6) must be used develop the IGM and not equation (2.7). For instance, for 5 d.o.f. robots (where the robot can place an axis in space) the vectors **a** and **P** of the matrix **E** of the end-effector may be used in solving the inverse geometric model. Similarly, in the case of 3 d.o.f. positional robots only the vector **P** may be used. The elements to be taken into account have to be specified by the user in a special window.

2.3.6. The Pieper method

This method can obtain the general solution of six degrees of freedom robot provided that one of the following conditions is verified [Pieper 68]:

- the robot contains three revolute joints forming a spherical joint (their axes are intersecting),
- the robot has three prismatic joints.

The program of SYMORO can obtain the I.G.M of the 52 structures that verify one of these conditions [Khalil 91, Bennis 91b]. We present here the case of six degree of freedom robots whose joints 4, 5 and 6 form a spherical joint (figure 2.1). In this case we have:

$$\begin{cases} d_5 = r_5 = d_6 = 0 \\ \sigma_4 = \sigma_5 = \sigma_6 = 0 \\ S\alpha_5 \; S\alpha_6 \neq 0 \quad \text{(non degenerating robot)} \end{cases}$$

The position of the center of the spherical joint is a function of the joint variables q_1, q_2 and q_3. So this type of structure transforms the inverse geometric model problem into two problems: a position problem and an orientation problem.

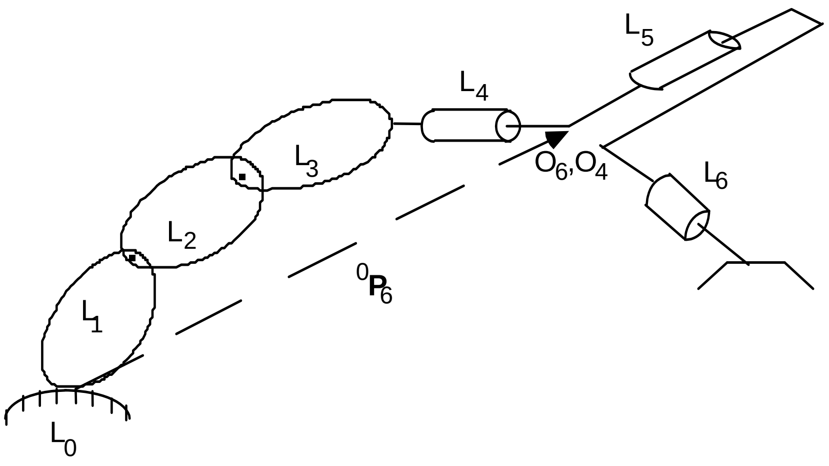


Figure 2.1. A six degree of freedom robot with spherical joint.

a) The position equations

Since ${}^0\mathbf{P}_6 = {}^0\mathbf{P}_4$, then the 4th column of the transformation ${}^0\mathbf{T}_1\, {}^1\mathbf{T}_2\, {}^2\mathbf{T}_3\, {}^3\mathbf{T}_4$ is equal to the 4th column of $\mathbf{U}_0$, such that:

$$\begin{bmatrix} Px \\ Py \\ Pz \\ 1 \end{bmatrix} = {}^0\mathbf{T}_1\, {}^1\mathbf{T}_2\, {}^2\mathbf{T}_3\, {}^3\mathbf{T}_4 \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \qquad (2.12)$$

The solution of equation (2.12) gives (q_1, q_2, q_3).

b) The orientation equations

Since the orientation of $\mathbf{U}_0$ is given as:

$$[\mathbf{s} \quad \mathbf{n} \quad \mathbf{a}] = {}^0\mathbf{A}_6$$

post multiplying the two sides by ${}^3\mathbf{A}_0$ we obtain:

$${}^3\mathbf{A}_0\, [\mathbf{s} \quad \mathbf{n} \quad \mathbf{a}] = {}^3\mathbf{A}_6 \qquad (2.13)$$

Since the variables (q_1, q_2, q_3) have been calculated, then ${}^3\mathbf{A}_0$ is known. Thus the left-hand side is known. The right-hand side is a function of $(\theta_4, \theta_5, \theta_6)$, which can be obtained by solving equation (2.13). The types of equations of the inverse geometric model that may be obtained using the Pieper method are given in table 2.2.

Type 1	$a\, r_i + b = 0$
Type 2	$a\, r_i^2 + b\, r_i + c = 0$
Type 3	$a\, C\theta_i + b\, S\theta_i + c = 0$
Type 4	$a\, C\theta_i + b\, S\theta_i + c = 0$ $a'\, C\theta_i + b'\, S\theta_i + c' = 0$
Type 5	$a_4\, r_i^4 + a_3\, r_i^3 + a_2\, r_i^2 + a_1\, r_i + a_0 = 0$
Type 6	$a_4\, S\theta_i^2 + a_3\, CS\theta_i + a_2\, C\theta_i + a_1\, S\theta_i + a_0 = 0$

Table 2.2 : Type of equations of Pieper method.

2.3.7. General method of the I.G.M.

The outline of this method, which is proposed by Raghavan & Roth, is given in appendix 1.

2.3.8. Robots with closed loops

The inverse geometric model of the direct (shortest) path between the base and the end-effector must be at first calculated, this can be obtained as in the case of simple open loop robots. Then the relations between the passive joint variables of the direct path in terms of the active joint variables outside this path must be obtained. If the loop contains only 3 non-motorized variables the explicit solution can be given by SYMORO+, the solution of robots with 4 passive joints is possible but not yet programmed.

Remark:

We can define the link frames such that the parameters b_j and γ_j of the frames of the direct path between the base of the robot and the terminal link are zero, in this case this path can be considered as a simple open loop. If some of these parameters (b_j and γ_j) are not zero, the program carry out the following variables change: $r_{i'} = r_i + b_j$ and $\theta_{i'} = \theta_i + \gamma_j$, with $i = a(j)$, such that the product of the matrices along the path from the base to the terminal link can be seen as an open loop robot. This because:

$$
\begin{aligned}
{}^{a(i)}\mathbf{T}_i\, {}^{i}\mathbf{T}_j &= \mathbf{Rot}(\mathbf{x},\alpha_i)\ \mathbf{Trans}(\mathbf{x},d_i)\ \mathbf{Rot}(\mathbf{z},\theta_i)\ \mathbf{Trans}(\mathbf{z},r_i)\ \mathbf{Rot}(\mathbf{z},\gamma_j)\ \mathbf{Trans}(\mathbf{z},b_j)\ \mathbf{Rot}(\mathbf{x},\alpha_j) \\
&\quad \mathbf{Trans}(\mathbf{x},d_j)\ \mathbf{Rot}(\mathbf{z},\theta_j)\ \mathbf{Trans}(\mathbf{z},r_j) \\
&= \mathbf{Rot}(\mathbf{x},\alpha_i)\ \mathbf{Trans}(\mathbf{x},d_i)\ \mathbf{Rot}(\mathbf{z},\theta_{i'})\ \mathbf{Trans}(\mathbf{z},r_{i'})\ \mathbf{Rot}(\mathbf{x},\alpha_j)\ \mathbf{Trans}(\mathbf{x},d_j)\ \mathbf{Rot}(\mathbf{z},\theta_j) \\
&\quad \mathbf{Trans}\ (\mathbf{z},r_j)
\end{aligned}
$$

with $r_{i'} = r_i + b_j$ and $\theta_{i'} = \theta_i + \gamma_j$

2.3.9. The inverse geometric model in SYMORO+

In the menu **Geometric** the methods of Paul, Pieper and the general method can be chosen to obtain the inverse geometric model of open loop robots or closed loop robots. The matrices **U** and **T** involved with the forward and backward direction of the Paul method can be also obtained such that the user can verify the solution of Paul's method or to obtain the solution by hand. This may be useful if the solution of the robot need new types of equations.

In this menu the explicit solution of the geometric constraint equations for closed loop robots with three passive joints, is also given [Bennis 94]. This function will provide the non-motorized variables, with μ=0, in terms of the motorized ones μ=1. To calculate the relation between motorized variables as a function of some non motorized joints, the parameters μ can be changed manually by the user and then recalculate the geometric constraint equations by SYMORO+, this situation is needed in the inverse geometric model of these robots.

2.4. The direct kinematic model

2.4.1. Introduction

The direct kinematic model defines the velocities of the operational coordinates as a function of the joint velocities. It is given by:

$$\dot{\mathbf{X}} = \mathbf{J}(\mathbf{q})\,\dot{\mathbf{q}} \tag{2.14}$$

where **J(q)** is the (mxn) Jacobian matrix, it is equal to $\frac{\partial \mathbf{X}}{\partial \mathbf{q}}$.

It is to be noted that the same Jacobian matrix appears in the direct differential model:

$$\mathbf{dX} = \mathbf{J}(\mathbf{q})\,\mathbf{dq} \tag{2.15}$$

where **dX** and **dq** are the differential vectors of the operational and joint coordinates.

2.4.2. The basic Jacobian matrix

The elements of the Jacobian matrix can be obtained by differentiating the direct geometric model **X** = **f** (**q**) as follows:

$$J_{ij} = \frac{\partial f_i(\mathbf{q})}{\partial q_j} \qquad i = 1, \ldots, m; \;\; j = 1, \ldots, n \tag{2.16}$$

where J_{ij} defines the ith row, jth column element of **J**.

We use a direct method to compute the Jacobian matrix. This Jacobian matrix is called the basic Jacobian, it gives the linear and angular velocities of frame n ($\mathbf{V}_n$ and ω_n) as a function of the joint velocities $\dot{\mathbf{q}}$. This relation is given as:

$$\begin{bmatrix} \mathbf{V}_n \\ \omega_n \end{bmatrix} = \mathbf{J}_n \, \dot{\mathbf{q}} \tag{2.17}$$

This basic Jacobian matrix also appears in the differential model that gives the differential translational and rotational vectors of frame n ($\mathbf{dP}_n$, δ_n) as a function of $\mathbf{dq}$.

$$\begin{bmatrix} \mathbf{dP}_n \\ \delta_n \end{bmatrix} = \mathbf{J}_n \, \mathbf{dq}$$

The Jacobian matrix corresponding to any representation of the operational coordinates $\mathbf{X}$ can be obtained from the basic Jacobian [Khalil 99c, Khalil 02a].

If $\mathbf{V}_n$ and ω_n are defined with respect to a given frame r, ${}^r\mathbf{V}_n$ and ${}^r\omega_n$, the corresponding Jacobian matrix will be denoted by ${}^r\mathbf{J}_n$. In most applications r = 0 or r = n.

2.4.3. Computation of the basic Jacobian matrix

We present here the case of simple open loop structure, when the system is tree structure, the same procedure can be applied by treating separately each branch from the robot base to the end-effectors. The case of closed loop robots is treated in section 2.6 .

Since the joint variable j represents either a translation or rotation, along the vector $\mathbf{z}_j$, thus we can write:

$$\begin{bmatrix} \mathbf{V}_n \\ \omega_n \end{bmatrix} = \begin{bmatrix} \sigma_1 \mathbf{a}_1 + \bar{\sigma}_1 (\mathbf{a}_1 \mathrm{x} \mathbf{L}_{1,n}) & \dots & \sigma_n \mathbf{a}_n + \bar{\sigma}_n (\mathbf{a}_n \mathrm{x} \mathbf{L}_{n,n}) \\ \bar{\sigma}_1 \mathbf{a}_1 & \dots & \bar{\sigma}_n \mathbf{a}_n \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \vdots \\ \dot{q}_n \end{bmatrix}$$

$$= \mathbf{J}_n \, \dot{\mathbf{q}} \tag{2.18}$$

with $\mathbf{L}_{k,n}$ is the vector between the origin of frames k and n, $\mathbf{O}_k\mathbf{O}_n$, while $\mathbf{a}_j$ is the unit vector along $\mathbf{z}_j$.

Projecting the elements of $\mathbf{J}_n$ with respect to an arbitrarily frame i, gives the Jacobian matrix ${}^i\mathbf{J}_n$.

We note that the frame transformation of the Jacobian matrix can be obtained as:

$${}^j\mathbf{J}_n = \begin{bmatrix} {}^j\mathbf{A}_i & \mathbf{0}_3 \\ \mathbf{0}_3 & {}^j\mathbf{A}_i \end{bmatrix} {}^i\mathbf{J}_n \tag{2.19}$$

with $\mathbf{0}_3$ is the (3x3) zero matrix

Remark:

The Jacobian matrix $\mathbf{J}_E$ giving the velocity of the end-effector frame E as a function of $\dot{\mathbf{q}}$ can be obtained using equation (2.18), after replacing $\mathbf{L}_{k,n}$ by $\mathbf{L}_{k,E}$, or by using the following relation:

$$ {}^{i}\mathbf{J}_E = \begin{bmatrix} \mathbf{I}_3 & -{}^{i}\hat{\mathbf{L}}_{E,n} \\ \mathbf{0}_3 & \mathbf{I}_3 \end{bmatrix} {}^{i}\mathbf{J}_n \tag{2.20} $$

with $\mathbf{I}_3$ is the (3x3) unit matrix, and $(\hat{\ })$ represents the (3x3) anti-symmetric matrix of the vector product such that: $\hat{\mathbf{V}}\,\mathbf{W} = \mathbf{V}\mathrm{x}\mathbf{W}$.

2.4.4. Decomposition of the Jacobian matrix

It has been shown that the Jacobian matrix can be decomposed into 3 matrices [Renaud 80a, 80b]: the first and the second are of full rank, while the third has the same rank as ${}^{i}\mathbf{J}_n$, but contains simpler elements, such that its inverse could be obtained analytically. In this method the Jacobian of frame n is calculated as a function of another Jacobian $\mathbf{J}_{n,j}$ defining the velocities of a frame fixed with the terminal link but confounded instantaneously with the intermediate frame j. The Jacobian matrix $\mathbf{J}_{n,j}$ is defined as:

$$ \mathbf{J}_{n,j} = \begin{bmatrix} \sigma_1\mathbf{a}_1+\bar{\sigma}_1(\mathbf{a}_1\mathrm{x}\mathbf{L}_{1,j}) & \sigma_2\mathbf{a}_2+\bar{\sigma}_2(\mathbf{a}_2\mathrm{x}\mathbf{L}_{2,j}) & \cdots & \sigma_n\mathbf{a}_n+\bar{\sigma}_n(\mathbf{a}_n\mathrm{x}\mathbf{L}_{n,j}) \\ \bar{\sigma}_1\mathbf{a}_1 & \bar{\sigma}_2\mathbf{a}_2 & \cdots & \bar{\sigma}_n\mathbf{a}_n \end{bmatrix} \tag{2.21} $$

$\mathbf{J}_n$ can be obtained from $\mathbf{J}_{n,j}$ using the following relation:

$$ \mathbf{J}_n = \begin{bmatrix} \mathbf{I}_3 & -\hat{\mathbf{L}}_{j,n} \\ \mathbf{0}_3 & \mathbf{I}_3 \end{bmatrix} \mathbf{J}_{n,j} \tag{2.22} $$

Projecting the elements of (2.21) with respect to frame i, we obtain:

$$ {}^{i}\mathbf{J}_n = \begin{bmatrix} \mathbf{I}_3 & -{}^{i}\hat{\mathbf{L}}_{j,n} \\ \mathbf{0}_3 & \mathbf{I}_3 \end{bmatrix} {}^{i}\mathbf{J}_{n,j} \tag{2.23} $$

The matrix ${}^{r}\mathbf{J}_n$ is thus calculated by the product of the following three matrices:

$$ {}^{r}\mathbf{J}_n = \begin{bmatrix} {}^{r}\mathbf{A}_i & \mathbf{0}_3 \\ \mathbf{0}_3 & {}^{r}\mathbf{A}_i \end{bmatrix} \begin{bmatrix} \mathbf{I}_3 & -{}^{i}\hat{\mathbf{L}}_{j,n} \\ \mathbf{0}_3 & \mathbf{I}_3 \end{bmatrix} {}^{i}\mathbf{J}_{n,j} \tag{2.24} $$

with ${}^{i}\mathbf{L}_{j,n} = {}^{i}\mathbf{A}_j\,{}^{j}\mathbf{P}_n$

We note that the first two matrices are of full rank.

In general, the values of j and i, defining the shifting intermediate frame and the projection frame, leading to the simplest ${}^{i}\mathbf{J}_{n,j}$ are given by:

$$j = p + 1,\ \ i = p, \qquad \text{with } p = [\text{integer of } n/2]$$

Thus for a robot with 6 degrees of freedom the matrix ${}^{3}\mathbf{J}_{6,4}$ is, in general, the simplest matrix. If the last three joints form a spherical joint then $\mathbf{L}_{6,4}$ is zero, consequently: ${}^{3}\mathbf{J}_{6,4} = {}^{3}\mathbf{J}_{6,6} = {}^{3}\mathbf{J}_{6}$.

2.4.5. Calculation of ${}^{i}\mathbf{J}_{n,j}$ matrix

The kth column of ${}^{i}\mathbf{J}_{n,j}$ is given by:
- if joint k is revolute:

$$
\begin{aligned}
{}^{i}\mathbf{j}_{n,jk} &= \begin{bmatrix} {}^{i}(\mathbf{a}_k \text{ x } \mathbf{L}_{k,j}) \\ {}^{i}\mathbf{a}_k \end{bmatrix} \\
&= \begin{bmatrix} {}^{i}\mathbf{A}_k\, {}^{k}\hat{\mathbf{a}}_k\, {}^{k}\mathbf{P}_j \\ {}^{i}\mathbf{a}_k \end{bmatrix} = \begin{bmatrix} -{}^{i}\mathbf{s}_k\, {}^{k}P_{y_j} + {}^{i}\mathbf{n}_k\, {}^{k}P_{x_j} \\ {}^{i}\mathbf{a}_k \end{bmatrix}
\end{aligned}
\tag{2.25}
$$

- if joint k is prismatic:

$$
{}^{i}\mathbf{j}_{n,jk} = \begin{bmatrix} {}^{i}\mathbf{a}_k \\ \mathbf{0} \end{bmatrix}
\tag{2.26}
$$

Remark: to calculate the Jacobian matrix ${}^{i}\mathbf{J}_E$ we replace ${}^{j}\mathbf{P}_n$ in relation (2.23) by ${}^{j}\mathbf{P}_E$.

2.4.6. Kinematic models as a function of the operational coordinates

Let $\mathbf{X} = \begin{bmatrix} \mathbf{X}_p \\ \mathbf{X}_r \end{bmatrix}$ be the vector representing the operational coordinates. For a given representation of $\mathbf{X}$, we can write [Khalil 02a]:

$$
\begin{bmatrix} \dot{\mathbf{X}}_p \\ \dot{\mathbf{X}}_r \end{bmatrix} = \begin{bmatrix} \Omega_p & \mathbf{0} \\ \mathbf{0} & \Omega_r \end{bmatrix} \begin{bmatrix} {}^{0}\mathbf{V}_n \\ {}^{0}\omega_n \end{bmatrix}
\tag{2.27}
$$

$\mathbf{X}_p$ and $\mathbf{X}_r$ define respectively the position and orientation vectors with respect to frame 0.

From relation (2.27), we obtain:

$$\begin{bmatrix} \dot{\mathbf{X}}_p \\ \dot{\mathbf{X}}_r \end{bmatrix} = \mathbf{J}_x \, \dot{\mathbf{q}} = \begin{bmatrix} \mathbf{\Omega}_p & \mathbf{0} \\ \mathbf{0} & \mathbf{\Omega}_r \end{bmatrix} {}^0\mathbf{J}_n \, \dot{\mathbf{q}}$$

where:

$$\mathbf{J}_x = \begin{bmatrix} \mathbf{\Omega}_p & \mathbf{0} \\ \mathbf{0} & \mathbf{\Omega}_r \end{bmatrix} {}^0\mathbf{J}_n$$

The matrices $\mathbf{\Omega}_p$ and $\mathbf{\Omega}_r$ are given in [Khalil 02a] for different representations of the operational coordinates **X**.

2.4.7. Practical calculation of the velocities of the links

The calculation of $\mathbf{V}_j$, ω_j can be carried out by first computing $\mathbf{J}_j$, then applying equation (2.17). In general it is more efficient, from the number of operations point of view, to use the following recursive relations (which will be developed in Chapter 3).

$$\begin{aligned} &\text{for } j = 1,\ldots,n \\ &{}^j\omega_i = {}^j\mathbf{A}_i \, {}^i\omega_i \\ &{}^j\omega_j = {}^j\omega_i + \bar{\sigma}_j \, \dot{q}_j \, {}^j\mathbf{a}_j \\ &{}^j\mathbf{V}_j = {}^j\mathbf{A}_i \, ({}^i\mathbf{V}_i + {}^i\omega_i \times {}^i\mathbf{P}_j) + \sigma_j \, \dot{q}_j \, {}^j\mathbf{a}_j \end{aligned} \qquad (2.28)$$

where i = a(j) defines the link antecedent to i, and ${}^j\mathbf{a}_j$ is the unit vector $[0 \;\; 0 \;\; 1]^T$
The initial values $\mathbf{V}_0$, ω_0 define the base velocities.

2.5. Second order kinematic model

The second order kinematic model gives the acceleration of the operational coordinates in terms of the joint positions, velocities and accelerations, it can be obtained by differentiating the kinematic model (2.14) as:

$$\ddot{\mathbf{X}} = \mathbf{J} \, \ddot{\mathbf{q}} + \dot{\mathbf{J}} \, \dot{\mathbf{q}} \qquad (2.29)$$

where:

$$\dot{\mathbf{J}} \, (\mathbf{q}, \dot{\mathbf{q}}) = \frac{d}{dt} \mathbf{J}(\mathbf{q}) \qquad (2.30)$$

As in the case of the kinematic model, $\ddot{\mathbf{X}}$ can be taken as the vectors of $\dot{\mathbf{V}}_n$ and $\dot{\omega}_n$. Thus, the second order kinematic model can be defined as:

$$\begin{bmatrix} \dot{\mathbf{V}}_n \\ \\ \dot{\omega}_n \end{bmatrix} = \mathbf{J}_n \ddot{\mathbf{q}} + \dot{\mathbf{J}}_n \dot{\mathbf{q}} \tag{2.31}$$

2.5.1. Practical calculation of link accelerations

The calculation of $\dot{\mathbf{V}}_j$ and $\dot{\omega}_j$ can be carried out by first calculating **J**, then using equation (2.31). In general, it is more efficient from the number of operations point of view, to use the following recursive relations, which will be detailed in Chapter 3.

for j = 1,...,n

$$\begin{aligned} {}^j\dot{\omega}_j &= {}^j\mathbf{A}_i\, {}^i\dot{\omega}_i + \bar{\sigma}_j \;(\ddot{q}_j\, {}^j\mathbf{a}_j + {}^j\omega_i \text{ x } \dot{q}_j\, {}^j\mathbf{a}_j) \\ {}^j\mathbf{U}_j &= {}^j\hat{\dot{\omega}}_j + {}^j\hat{\omega}_j\, {}^j\hat{\omega}_j \\ {}^j\dot{\mathbf{V}}_j &= {}^j\mathbf{A}_i\, [\, {}^i\dot{\mathbf{V}}_i + {}^i\mathbf{U}_i\, {}^i\mathbf{P}_j\,] + \sigma_j\, [\, \ddot{q}_j\, {}^j\mathbf{a}_j + 2\, {}^j\omega_i \text{ x } \dot{q}_j\, {}^j\mathbf{a}_j\,] \end{aligned} \tag{2.32}$$

where: $i = a(j)$, ${}^j\omega_i$ and ${}^j\omega_j$ are calculated using equations (2.28).

In some applications (task space control), we need to calculate the vector $\dot{\mathbf{J}}\,\dot{\mathbf{q}}$. Instead of differentiating **J** with respect to time and multiplying the result by $\dot{\mathbf{q}}$, which will be time consuming, we can use the recursive equations (2.32) and by setting $\ddot{\mathbf{q}}$ equal to zero [Khalil 87b].

2.6. Kinematic model of closed loop robots

This model defines the operational velocity as a function of the velocities of motorized joints. At first, the kinematic model (using the Jacobian matrix) of the direct (shortest) path between the base and the end-effector must be calculated, this can be obtained as in the case of simple open loop robots. Then the relations between the passive joint velocities of the direct path as a function of the active joint velocities outside this path must be obtained. One of the following solutions can be considered:

a- By differentiating the geometric constraint equations, this solution is used in SYMORO+ for parallelogram loops.

b- By equating the velocities on the terminal frame of each loop using the two branches of the loop. From figure 2.2, we obtain:

$$\begin{bmatrix} \mathbf{V}_k \\ \omega_k \end{bmatrix} - \begin{bmatrix} \mathbf{V}_{k+B} \\ \omega_{k+B} \end{bmatrix} = \mathbf{0} \tag{2.33}$$

For each loop we obtain:

$$\begin{bmatrix} \mathbf{V}_k \\ \omega_k \end{bmatrix} = \begin{bmatrix} \sigma_e \mathbf{a}_e + \bar{\sigma}_e(\mathbf{a}_e x \mathbf{L}_{e,k}) & \dots & \sigma_k \mathbf{a}_k + \bar{\sigma}_k(\mathbf{a}_k x \mathbf{L}_{k,k}) \\ \bar{\sigma}_e \, \mathbf{a}_e & \dots & \bar{\sigma}_k \, \mathbf{a}_k \end{bmatrix} \begin{bmatrix} \dot{q}_e \\ : \\ \dot{q}_k \end{bmatrix} \tag{2.34a}$$

$$\begin{bmatrix} \mathbf{V}_{k+B} \\ \omega_{k+B} \end{bmatrix} = \begin{bmatrix} \sigma_d \mathbf{a}_d + \bar{\sigma}_d(\mathbf{a}_d x \mathbf{L}_{d,k+B}) & \dots & \sigma_j \mathbf{a}_j + \bar{\sigma}_j(\mathbf{a}_j x \mathbf{L}_{j,k+B}) \\ \bar{\sigma}_d \, \mathbf{a}_d & \dots & \bar{\sigma}_j \, \mathbf{a}_j \end{bmatrix} \begin{bmatrix} \dot{q}_d \\ : \\ \dot{q}_j \end{bmatrix} \tag{2.34b}$$

where e and d are the first joints, from the root of the loop, on each branch of the loop.

Projecting the vectors of equations (2.34a) and (2.34b) into frames k and k+B respectively, we obtain by the use of equation (2.33) that:

$$^{k}\mathbf{J}_k \, \dot{\mathbf{q}}_1 - {}^{k+B}\mathbf{J}_{k+B} \, \dot{\mathbf{q}}_2 = \mathbf{0} \tag{2.35}$$

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

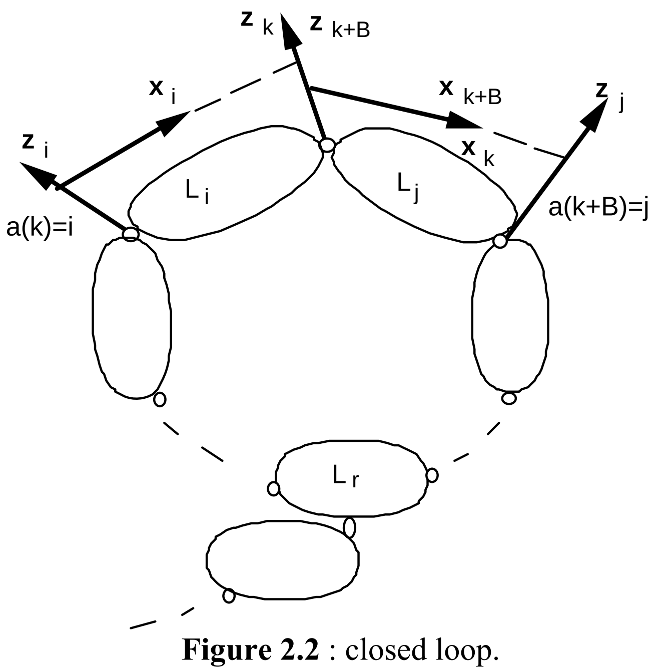


Figure 2.2 : closed loop.

Doing this procedure for all the loops, and gathering all the equations after eliminating the trivial rows with zeros elements, we obtain the following general relation [Zghaib 92]:

$$\begin{bmatrix} \mathbf{W_a} & \mathbf{W_p} & \mathbf{0} \\ \mathbf{W_{ac}} & \mathbf{W_{pc}} & \mathbf{W_c} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{q}}_a \\ \dot{\mathbf{q}}_p \\ \dot{\mathbf{q}}_c \end{bmatrix} = \mathbf{0} \tag{2.36}$$

where:

$\dot{\mathbf{q}}_a$ is the vector of the velocity of the N active (motorized) joints of the robot,

$\dot{\mathbf{q}}_p$ is the vector of the velocity of the n-N = p passive joints of the equivalent tree structure,

$\dot{\mathbf{q}}_c$ is the vector of the velocity of the B opened joint.

From the first row of equation (2.36), we can write:

$$\mathbf{W}_p \, \dot{\mathbf{q}}_p = -\mathbf{W}_a \, \dot{\mathbf{q}}_a \tag{2.37}$$

with:

$\mathbf{W}_a$ of dimension (p x N), and $\mathbf{W}_p$ of dimension (p x p).

If the system is compatible, $\mathbf{W}_p$ will be of rank p, outside singular positions. Thus using (2.37) we obtain:

$$\dot{\mathbf{q}}_p = \mathbf{W} \, \dot{\mathbf{q}}_a \tag{2.38}$$

with:

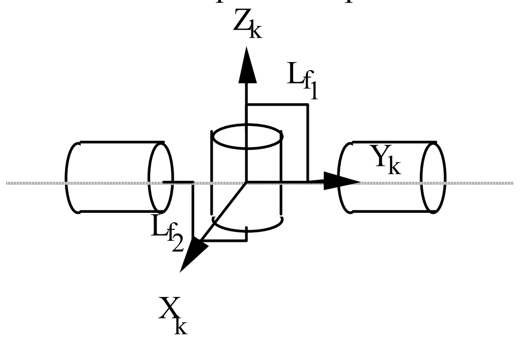
$$\mathbf{W} = -\mathbf{W}_p^{-1} \, \mathbf{W}_a \tag{2.39}$$

Remark:

1. SYMORO+ can take into account that the opened joint is a complex one, with more than one degree of freedom, such as spherical, universal joint, and cylindrical joints. In this case all the opened variables of this complex joints are included in $\dot{\mathbf{q}}_c$, for instance in the case of opened spherical joint, there will be three corresponding components in $\dot{\mathbf{q}}_c$, while $\dot{\mathbf{q}}_p$ will contain less components than the passive joints of the equivalent tree structure.

2. The description of a spherical joint will be done using three intersecting revolute joints, separated by virtual links of zero inertial parameters (mass, first moments and inertia matrix). To obtain the constraint equations as given in remark 1, the user has to define the frames such that the fictitious links be on each side of the opened joint, that is to say to carry out the opening on the middle joint (figure 2.3). In the case of universal joint (two intersecting rotational joints) or cylindrical joint (a revolute joint and a prismatic joint along the same axis), both variables must be on the same branch (figure 2.4). The speed and the accelerations of these joints will not appear on the dynamic model, but its positions may appear in $\mathbf{W_a}$, $\mathbf{W_p}$. The value of $\mathbf{q}_c$ can be obtained by integrating the second row of equation (2.36).

3. Depending on the type of the loop, spatial or planar, and the type of the supposed opened joint, redundant rows in equation (2.36) can be automatically eliminated. For example, in the case of a planar loop and rotational opened joint the rows (3,4,5) must be eliminated.



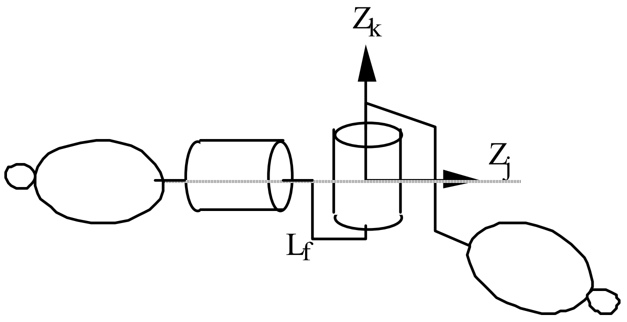


Figure 2.3 : definition of a spherical joint

Figure 2.4 : definition of a universal joint

2.7. The accelerations constraint equations of the loops

The acceleration constraint equations will be needed also in developing the dynamic model of closed loops, it can be obtained by differentiating equation (2.36), which yields:

$$\begin{bmatrix} \mathbf{W_a} & \mathbf{W_p} & \mathbf{0} \\ \mathbf{W_{ac}} & \mathbf{W_{pc}} & \mathbf{Wc} \end{bmatrix} \begin{bmatrix} \ddot{q}_a \\ \ddot{q}_p \\ \ddot{q}_c \end{bmatrix} + \begin{bmatrix} \mathbf{Psi} \\ \mathbf{Phi} \end{bmatrix} = \mathbf{0} \qquad (2.40a)$$

where **Psi** and **Phi** contain the quadratic joint velocities elements, they can be obtained for each loop from the following equation:

$$\begin{bmatrix} \dot{\mathbf{V}}_k(\ddot{\mathbf{q}}_1\mathbf{=0}) \\ \dot{\omega}_k(\ddot{\mathbf{q}}_1\mathbf{=0}) \end{bmatrix} - \begin{bmatrix} \dot{\mathbf{V}}_{k+B}(\ddot{\mathbf{q}}_2\mathbf{=0}) \\ \dot{\omega}_{k+B}(\ddot{\mathbf{q}}_2\mathbf{=0}) \end{bmatrix} = {}^{k}\dot{\mathbf{J}}_k\,\dot{\mathbf{q}}_1 - {}^{k+B}\dot{\mathbf{J}}_{k+B}\,\dot{\mathbf{q}}_2 = \mathbf{0} \qquad (2.40b)$$

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

The vectors **Psi** and **Phi** can be calculated efficiently using recursive acceleration relations and after setting the accelerations to zero, as in the case of calculating $\dot{\mathbf{J}}\dot{\mathbf{q}}$ presented in section 2.5, and by taking into account the order of the rows considered in equation (2.36).

2.8. The direct kinematic model in SYMORO+

The menu **Kinematic** of **SYMORO+** calculates ${}^{i}\mathbf{J}_{r,j}$, $-{}^{i}\hat{\mathbf{L}}_{j,r}$ for open loop or closed loop structures, with i, j, r = 1,...,n.

In this menu the link velocities can be calculated using recursive customized method (using intermediate variables to reduce the number of operations). The calculation of $\dot{\mathbf{J}}\dot{\mathbf{q}}$, denoted JPQP, is also given in customized form. The velocities and accelerations constraint equations of closed loops can also be obtained.

2.9. The inverse kinematic model

The inverse kinematic model gives, for a given configuration $\mathbf{q}$, the joint velocities $\dot{\mathbf{q}}$ corresponding to a desired $\dot{\mathbf{X}}$.

The solution can be obtained either symbolically or numerically:

- the symbolic solution has the advantage of reducing the computational cost, but it must treat on a case by case basis the singular positions [Chevallereau 87, Chevallereau 88a],
- the numerical methods are more general, the most in common use are those based on the use of the pseudo-inverse solution: the corresponding algorithm can take into account automatically the regular, singular and redundant cases. Its calculation cost is relatively more important than the symbolic methods.

2.9.1. General form of the kinematic inverse model

From equation (2.27), we obtain:

$$\dot{\mathbf{X}} = \begin{bmatrix} \Omega_p & \mathbf{0}_3 \\ \mathbf{0}_3 & \Omega_r \end{bmatrix} \begin{bmatrix} {}^{0}\mathbf{A}_i & \mathbf{0}_3 \\ \mathbf{0}_3 & {}^{0}\mathbf{A}_i \end{bmatrix} \begin{bmatrix} \mathbf{I}_3 & -{}^{i}\hat{\mathbf{L}}_{j,n} \\ \mathbf{0}_3 & \mathbf{I}_3 \end{bmatrix} {}^{i}\mathbf{J}_{n,j}\,\dot{\mathbf{q}} \tag{2.41}$$

or in a more compact form:

$$\dot{\mathbf{X}} = {}^{0}\mathbf{J}_n\,\dot{\mathbf{q}} \tag{2.42}$$

Owing to the simplicity of the elements of the matrix ${}^i\mathbf{J}_{n,j}$ compared to those of ${}^0\mathbf{J}_n$, it is more convenient to obtain the inverse kinematic model from the equation:

$$ {}^i\dot{\mathbf{X}}_j = {}^i\mathbf{J}_{n,j}\ \dot{\mathbf{q}} \tag{2.43}$$

with:

$$ {}^i\dot{\mathbf{X}}_j = \begin{bmatrix} \mathbf{I}_3 & {}^i\hat{\mathbf{L}}_{j,n} \\ \mathbf{0}_3 & \mathbf{I}_3 \end{bmatrix} \begin{bmatrix} {}^i\mathbf{A}_0 & \mathbf{0}_3 \\ \mathbf{0}_3 & {}^i\mathbf{A}_0 \end{bmatrix} \begin{bmatrix} \mathbf{\Omega}_p^{-1} & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{\Omega}_r^{+} \end{bmatrix} \dot{\mathbf{X}} \tag{2.44}$$

the matrices $\mathbf{\Omega}_p^{-1}$ et $\mathbf{\Omega}_r^{+}$ are given in [Khalil 02a] for different representations of $\mathbf{X}$.

Thus, the problem of the inverse kinematic model is reduced to the inversion of ${}^i\mathbf{J}_{n,j}$.

Remark: if n < 6, we cannot use systematically the matrix ${}^i\mathbf{J}_{n,j}$. The singularities in this case do not take into account the representation of the operational coordinates [Borrel 86].

2.9.2. The inverse kinematic model in the regular case

In this case, the Jacobian matrix is square and its determinant is not zero. We calculate ${}^i\mathbf{J}_j^{-1}$, the inverse of ${}^i\mathbf{J}_j$, the user calculates then $\dot{\mathbf{q}}$ using the relation:

$$ \dot{\mathbf{q}} = {}^i\mathbf{J}_j^{-1}\ {}^i\dot{\mathbf{X}}_j \tag{2.45}$$

When the matrix $\mathbf{J}$ has the following form:

$$ \mathbf{J} = \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{B} & \mathbf{C} \end{bmatrix} \tag{2.46}$$

where the matrices $\mathbf{A}$ and $\mathbf{C}$ are regular, we can see that:

$$ \mathbf{J}^{-1} = \begin{bmatrix} \mathbf{A}^{-1} & \mathbf{0} \\ -\mathbf{C}^{-1}\mathbf{B}\mathbf{A}^{-1} & \mathbf{C}^{-1} \end{bmatrix} \tag{2.47}$$

The solution of this problem will be reduced to the inversion of the matrices $\mathbf{A}$ and $\mathbf{C}$.

When the robot has six degrees of freedom with the last three joints forming a spherical joint, the $\mathbf{J}$ matrix will be like that of equation (2.46), with $\mathbf{A}$ and $\mathbf{C}$ are of dimension (3x3) [Gorla 84].

2.9.3. Kinematic model of redundant robots

A robot is said to be redundant if it has more degrees of freedom than the dimension of the operational space of its terminal link. The extra degrees of freedom permit to increase the working space of the robot and to satisfy supplementary criteria such as:

– avoiding obstacles [Baillieul 85],
– avoiding singular configurations [Yoshikawa 84],
– avoiding joint limits [Fournier 80, Klein 84],
– minimizing the joint torques [Baillieul 84, Hollerbach 85].

For such robots the matrix **J** is of (m x n) dimension with n > m. The number (n – m) represents the redundancy degree. Many methods have been proposed to solve such systems:

a) adding (n – m) relations such that **J**, becomes square. These relations may be trivial such as fixing some joint variables. They can be chosen to satisfy optimization criteria [Baillieul 85, Chang 86].

b) calculating a particular solution considering m primary joints, then calculating all the joint variables in order to satisfy a given optimization criterion [Chevallereau 88a]. This method gives in general solutions with reduced number of operations.

c) using the pseudo-inverse solution [Penrose 55] combined with the homogeneous solution, it is given as [Fournier 80]:

$$\mathbf{dq} = \mathbf{J}^{+}\,\mathbf{dX} + (\mathbf{I} - \mathbf{J}^{+}\,\mathbf{J})\,\mathbf{Z} \qquad (2.48)$$

where $\mathbf{J}^{+}$ is the pseudo-inverse of **J** while **Z** represents an arbitrary vector of R^{n}.

The first term of (2.48) gives the solution minimizing the Euclidien norm $||\mathbf{dq}||^2$. The second term, called the homogeneous solution or the optimization term, belongs to the kernel of **J** and does not affect the value of **X**. The homogeneous term can be used to optimize a desired criterion. Let $\phi(\mathbf{q})$ be a positive definite scalar function of the robot configuration **q** and let $\nabla\phi$ be the gradient of this function. Taking **Z** such that $\mathbf{Z} = -\alpha\nabla\phi$, with α positive, minimizes the function $\phi(\mathbf{q})$, while taking $\mathbf{Z} = \alpha\nabla\phi$ maximizes this function. Thus, we can write:

$$\mathbf{dq} = \mathbf{J}^{+}\,\mathbf{dX} + \alpha(\mathbf{I} - \mathbf{J}^{+}\,\mathbf{J})\,\nabla\phi \qquad (2.49)$$

with:

$$\nabla\phi = [\frac{\partial\phi}{\partial q_1} \;\cdots\; \frac{\partial\phi}{\partial q_n}]^T \qquad (2.50)$$

The value of α allows us to realize a trade-off between the minimization of $||\mathbf{dq}||^2$ and the optimization of $\phi(\mathbf{q})$.

2.9.4. The inverse kinematic model in SYMORO+

The menu **Kinematic** of **SYMORO+** gives, the symbolic inverse of a Jacobian matrix. It is to be noted that when the Jacobian matrix is complicated (containing many non zero elements), the symbolic inverse would be difficult to realize and will be certainly less efficient than the numerical method, from the computation cost point of view. The calculation of the determinant can also be obtained to study singular positions. The user has the possibility to inverse, or calculate the determinant, of a sub-matrix of the Jacobian matrix. At the moment no symbolic solution is given for redundant robots.

Chapter 3

Dynamic model of simple open loop robots

3.1. Introduction

The dynamic model provides the joint torques and forces as a function of joint positions, velocities, accelerations and of the effort exerted by the end-effector on the environment. It is represented by the following relation:

$$\mathbf{\Gamma} = \mathbf{f}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}, \mathbf{F_t}) \tag{3.1}$$

where:

$\mathbf{\Gamma}$	joint torques and forces vector,
$\mathbf{q}$	joint positions vector,
$\dot{\mathbf{q}}$	joint velocities vector,
$\ddot{\mathbf{q}}$	joint accelerations vector,
$\mathbf{F_t}$	external wrench (forces and torques), exerted by the end-effector on the environment.

The relation (3.1) is called the inverse dynamic model or simply the dynamic model. The direct dynamic model, used in the simulation, gives the joint accelerations as a function of joint positions, velocities and torques. It is represented by the following relation:

$$\ddot{\mathbf{q}} = \mathbf{g}(\mathbf{q}, \dot{\mathbf{q}}, \Gamma, \mathbf{F_t}) \tag{3.2}$$

Many formulations are used to compute the dynamic model, the most popular are:

a) the Lagrange equations [Uicker 69, Khalil 76, Renaud 75, Hollerbach 80, Megahed 84, Paul 81, Renaud 85],

b) the Newton-Euler formulation [Armstrong 79, Luh 79, Khalil 85a, Khalil 87c, Renaud 87]. These methods are more efficient in computing the inverse dynamic model used in control applications.

In this chapter, we present these two formulations for simple open loop robots. At the end, the main applications of the dynamic model will be presented. The dynamic model of tree structure robots and closed loop robots will be treated in Chapter 4.

3.2. Notations

The main notations used in this chapter are compiled below:

$\mathbf{a}_j$	unit vector along the $\mathbf{z}_j$ axis,
$\mathbf{F}_j$	total external forces exerted on link j,
$\mathbf{f}_j$	force exerted on link j by link j-1,
$\mathbf{f}_{tj}$	force exerted by link j on the environment,
Fs_j	Coulomb friction parameter of joint j,
Fv_j	viscous friction parameter of joint j,
$\mathbf{g}$	acceleration of gravity,
G_j	center of mass of link j,
$\mathbb{I}_j$	inertia tensor of link j with respect to a frame parallel to frame j and its origin is G_j,
Ia_j	inertia of the rotor of motor j referred to the joint side,
${}^j\mathbf{J}_j$	inertia tensor of link j with respect to frame j, it will be denoted by:

$$
{}^j\mathbf{J}_j = \begin{bmatrix} XX_j & XY_j & XZ_j \\ XY_j & YY_j & YZ_j \\ XZ_j & YZ_j & ZZ_j \end{bmatrix}
$$

$\mathbf{L}_j$	vector $\mathbf{O_iO_j}$, where O_i is the origin of frame i, antecedent of frame j, and O_j is the origin of frame j, in simple open loop robots i = j-1.
M_j	mass of link j,
$\mathbf{MS}_j$	first moments vector of link j around the origin of frame j, it is equal to M_j $\mathbf{S}_j$. The components of ${}^j\mathbf{MS}_j$ will be denoted by $[\ MX_j \quad MY_j \quad MZ_j\ ]^T$,
$\mathbf{N}_j$	total external moments on link j about G_j,
$\mathbf{n}_j$	moment exerted on link j by link j-1and by the motor j,
$\mathbf{n}_{tj}$	moment about O_j exerted by link j on the environment,
$\mathbf{S}_j$	vector defining the center of mass of link j with respect to O_j, equal to $\mathbf{O_jG_j}$,
$\mathbf{V}_j$	linear velocity of point O_j,
$\dot{\mathbf{V}}_j$	linear acceleration of point O_j,
$\mathbf{Vg}_j$	linear velocity of G_j,
$\dot{\mathbf{Vg}}_j$	linear acceleration of G_j,
ω_j	angular velocity of link j,
$\dot{\omega}_j$	angular acceleration of link j.

3.3. Lagrange equation

3.3.1. Introduction

The dynamic model of a robot with several degrees of freedom represents a complicated system. The Newton-Euler method developed in § 3.4 presents an efficient and systematic approach to solving this problem. In this section, we develop a simple Lagrange method to present the general form of the dynamic model of robots and to study its properties. Firstly, we consider an ideal system without friction or elasticity, exerting neither forces nor moments on the environment. These phenomena will be covered in § 3.3.2.3 through 3.3.2.6.

The Lagrange formulation describes the behavior of a dynamic system in terms of work and energy stored in the system. The Lagrange equations are commonly written in the form:

$$\Gamma_i = \frac{d}{dt}\frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i} \qquad i = 1, \ldots, n \tag{3.3}$$

where:

L is the Lagrangian of the robot, given as E – U,
E the kinetic energy of the robot,
U the potential energy of the robot.

3.3.2. General form of the dynamic model

The kinetic energy, denoted by E, is a quadratic function of joint velocities, such that:

$$E = \frac{1}{2}\,\dot{\mathbf{q}}^T\,\mathbf{A}\,\dot{\mathbf{q}} \tag{3.4}$$

where **A** is the (n x n) matrix of the kinetic energy, its (i,j) element is denoted A_{ij}, it is also called the inertia matrix of the robot. Its elements are functions of the joint variables **q**.

The potential energy, denoted by U, is a function of **q**. Using (3.3) and (3.4) the input torque vector Γ can be obtained as:

$$\mathbf{\Gamma} = \mathbf{A}\,\ddot{\mathbf{q}} + \mathbf{B}\,\dot{\mathbf{q}}\dot{\mathbf{q}} + \mathbf{C}\,\dot{\mathbf{q}}^2 + \mathbf{Q} \tag{3.5}$$

with:

- **B** (n x n (n – 1)/2) matrix containing the elements of Coriolis torques, its general element is represented by $B_{i,jk}$,
- **C** (n x n) matrix containing the elements of centrifugal torques, its general element is represented by C_{ij},

$\dot{\mathbf{q}}\dot{\mathbf{q}} = [\, \dot{q}_1\dot{q}_2 \;\cdots\; \dot{q}_1\dot{q}_n \;\; \dot{q}_2\dot{q}_3 \;\cdots\; \dot{q}_{n-1}\dot{q}_n \,]^T$,

$\dot{\mathbf{q}}^{\mathbf{2}} = [\, \dot{q}_1^2 \;\ldots\; \dot{q}_n^2 \,]^T$,

$\mathbf{Q} = [\, Q_1 \;\ldots\; Q_n \,]^T$, gravity torques vector.

The elements of **B**, **C** and **Q** are obtained using the following relations:

$$B_{i,jk} = \frac{\partial A_{ij}}{\partial q_k} + \frac{\partial A_{ik}}{\partial q_j} - \frac{\partial A_{jk}}{\partial q_i} \tag{3.6}$$

$$C_{ij} = \frac{\partial A_{ij}}{\partial q_j} - \frac{1}{2}\frac{\partial A_{jj}}{\partial q_i} \tag{3.7}$$

$$Q_i = \frac{\partial U}{\partial q_i} \tag{3.8}$$

The elements of **A**, **B**, **C**, and **Q** are functions of the geometric and inertial parameters of the links. The dynamic equations form a system of n coupled and non-linear second order differential equations.

The dynamic model of a robot is linear in the elements of the inertial parameters M_j, ${}^j\mathbf{MS}_j$ and ${}^j\mathbf{J}_j$ [Khalil 85c]. This property will be used to determine the minimum inertial parameters [Khalil 87 c], also known as the base inertial parameters, which constitute the identifiable inertial parameters, and will be used to reduce the computational cost of the dynamic model.

3.3.2.1. Calculation of the kinetic energy

The kinetic energy of the robot is given as:

$$E = \sum_{j=1}^{n} E_j \tag{3.9}$$

where E_j denotes the kinetic energy of link j, calculated by the following relation:

$$E_j = \frac{1}{2}(\, \omega_j^T \,\mathbb{I}_j\, \omega_j + M_j \,\mathbf{Vg}_j^T\, \mathbf{Vg}_j \,) \tag{3.10}$$

or by:

$$E_j = \frac{1}{2}[\, \omega_j^T \,\mathbf{J}_j\, \omega_j + M_j \,\mathbf{V}_j^T \mathbf{V}_j + 2\,\mathbf{MS}_j^T\,(\, \mathbf{V}_j \times \omega_j \,)\,] \tag{3.11}$$

where:

$$\mathbf{J}_j = \mathbb{I}_j - M_j\, \hat{\mathbf{S}}_j\, \hat{\mathbf{S}}_j \tag{3.12}$$

Equation (3.11) is linear in the elements of M_j, $\mathbf{MS}_j$ and $\mathbf{J}_j$. The velocity vectors $\mathbf{V}_j$ and ω_j are computed by the following recursive relations:

$$\omega_j = \omega_{j-1} + \bar{\sigma}_j \dot{q}_j \mathbf{a}_j \tag{3.13}$$

$$\mathbf{V}_j = \mathbf{V}_{j-1} + \omega_{j-1} \times \mathbf{L}_j + \sigma_j \dot{q}_j \mathbf{a}_j \tag{3.14}$$

For a fixed base robot, the initial conditions are:

$$\mathbf{V}_0 = \mathbf{0} \quad \text{and} \quad \omega_0 = \mathbf{0} \tag{3.15}$$

Projecting all the vectors of equation (3.11) into frame j, we obtain:

$$E_j = \frac{1}{2} [{}^j\omega_j^T \, {}^j\mathbf{J}_j \, {}^j\omega_j + M_j \, {}^j\mathbf{V}_j^T \, {}^j\mathbf{V}_j + 2 \, {}^j\mathbf{MS}_j^T ({}^j\mathbf{V}_j \times {}^j\omega_j)] \tag{3.16}$$

$${}^j\omega_j = {}^j\mathbf{A}_{j-1} \, {}^{j-1}\omega_{j-1} + \bar{\sigma}_j \dot{q}_j \, {}^j\mathbf{a}_j \tag{3.17}$$

$${}^j\mathbf{V}_j = {}^j\mathbf{A}_{j-1} ({}^{j-1}\mathbf{V}_{j-1} + {}^{j-1}\omega_{j-1} \times {}^{j-1}\mathbf{P}_j) + \sigma_j \dot{q}_j \, {}^j\mathbf{a}_j \tag{3.18}$$

where:

$${}^j\mathbf{a}_j = [\,0 \quad 0 \quad 1\,]^T$$

3.3.2.2. Calculation of the potential energy

The potential energy is given by:

$$U = -\sum_{j=1}^{n} M_j \, \mathbf{g}^T (\mathbf{L}_{0,j} + \mathbf{S}_j) = -\sum_{j=1}^{n} {}^0\mathbf{g}^T (M_j \, {}^0\mathbf{P}_j + {}^0\mathbf{A}_j \, {}^j\mathbf{MS}_j)$$

$$= -\sum_{j=1}^{n} {}^0\mathbf{g}^T \, {}^0\mathbf{T}_j \begin{bmatrix} {}^j\mathbf{MS}_j \\ M_j \end{bmatrix} \tag{3.19}$$

We note that U is linear in M_j and in the elements of ${}^j\mathbf{MS}_j$.

Since the kinetic and potential energies are linear in the inertial parameters, then using (3.3) it can be seen that the dynamic model is also linear in these parameters.

3.3.2.3. Taking into account the effect of friction

In general, the friction is difficult to model. We can approximate its effect by adding to the ith component of the right hand side of equation (3.5) the following terms:

$$\Gamma_{fj} = Fs_j \, \text{sign}(\dot{q}_j) + Fv_j \, \dot{q}_j \tag{3.20}$$

where Fs_j is the Coulomb friction parameter, and Fv_j is the viscous friction parameter.

For a more detailed study on friction, the reader can consult [Armstrong 91].

3.3.2.4. Taking into account the rotor inertia

In general, the kinetic energy of the rotor of motor i is calculated by $Ia_i\,\dot{q}_i^2/2$. In order to consider the rotor inertia in the dynamic model of the robot, we add the inertia (or mass) Ia_i to the A_{ii} element of the matrix **A**. Note that such modeling neglects the gyroscopic effects of the rotors, which take place when the actuator is fixed on a moving link. However, this approximation is justified for high gear transmission ratios. For more accurate modeling of the rotors the reader is referred to [Llibre 83], [Chedmail 86], [Murphy 93], [Sciavicco 94].

3.3.2.5. Taking into account the transmission flexibility

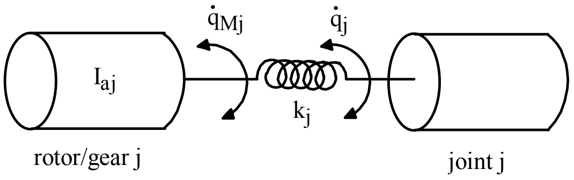


Figure 3.1 Modeling of joint flexibility

If the transmission system contains flexibility, the potential energy of the corresponding spring can be given as:

$$U_k = \frac{1}{2}(\mathbf{q} - \mathbf{q_M})^T \mathbf{k}\, (\mathbf{q} - \mathbf{q_M})$$

Adding this energy to the Lagrangian of the system, the dynamic model of the robot will be obtained as:

$$\mathbf{A}\,\ddot{\mathbf{q}} + \mathbf{B}\,\dot{\mathbf{q}}\dot{\mathbf{q}} + \mathbf{C}\,\dot{\mathbf{q}}^2 + \mathbf{Q} + \mathbf{k}\,(\mathbf{q} - \mathbf{q_M}) = \mathbf{0}$$

$$\mathbf{I_a}\,\ddot{\mathbf{q}}_\mathbf{M} + \mathbf{F_V}\,\dot{\mathbf{q}}_\mathbf{M} - \mathbf{k}\,(\mathbf{q} - \mathbf{q_M}) = \mathbf{\Gamma}$$

with:

$$\mathbf{I_a} = \rho^2\,\mathbf{J_m} \quad , \quad \mathbf{F_V} = \rho^2\,\mathbf{F_{Vm}} \quad , \quad \mathbf{\Gamma} = \rho\,\boldsymbol{\tau}_\mathbf{m} \quad \text{and} \quad \mathbf{q_M} = \rho^{-1}\,\mathbf{q_m}$$

- $\mathbf{F_V}$ is a diagonal (nxn) matrix representing the parameters of viscous friction referred to the joint side,
- **k** is a diagonal (nxn) matrix representing the flexibility stiffens referred to the joint side,
- ρ is a diagonal (nxn) matrix representing the transmission gear ratio,
- $\boldsymbol{\tau}_\mathbf{m}$, $\mathbf{q_m}$, $\mathbf{J_m}$, $\mathbf{F_{Vm}}$ are the motor torques, motor variables, rotor inertias, motor friction parameters referred to motor side.

A general and systematic method to model systems with lumped elasticity is presented in [Khalil 00].

3.3.2.6. Taking into account the forces exerted by the terminal-effector on the environment

Let $\mathbf{F}_{tn}$ represents the wrench (forces and moments) exerted by the terminal-effector on the environment, such that:

$$\mathbf{F}_{tn} = \begin{bmatrix} \mathbf{f}_{tn} \\ \\ \mathbf{n}_{tn} \end{bmatrix} = [\, f_x \;\; f_y \;\; f_z \;\; n_x \;\; n_y \; n_z \,]^T$$

From the virtual work principle, we obtain:

$$\Gamma_e^T \, \mathbf{dq} = \mathbf{F}^T{}_{tn} \begin{bmatrix} \mathbf{dP_n} \\ \\ \delta_n \end{bmatrix} \tag{3.21}$$

where Γ_e denotes the joint torques or forces corresponding to the external wrench, and $\mathbf{dP}_n$ and δ_n are the differential translational and rotational vector of the end-effector frame.

Using the direct differential model, giving $\mathbf{dP}_n$ and δ_n as a function of **dq**, we obtain:

$$\Gamma_e = \mathbf{J}_n{}^T \, \mathbf{F}_{tn} \tag{3.22}$$

The dynamic model equation (3.5) can take into account the external forces by adding to its right hand side the term Γ_e.

3.3.3. Properties of the dynamic model

The following properties can be deduced from the foregoing sections:

1- The inertia matrix is symmetric and positive definite,

2- The dynamic model can be represented as a linear function of the inertial and friction parameters. In fact, the dynamic model can be written as:

$$\Gamma = \mathbf{\Phi}\,(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})\,\Theta \tag{3.23}$$

where:

Θ is the vector of dynamic parameters of the robot:

$$\Theta = [\Theta^{1T} \;\; \Theta^{2T} \;\; \cdots \;\; \Theta^{nT}]^T \tag{3.24}$$

Θ^j is the vector of the “standard” dynamic parameters of link j:

$$\Theta^j = [\, XXj \;\; XYj \;\; XZj \;\; YYj \;\; YZj \;\; ZZj \;\; MXj \;\; MYj \;\; MZj \;\; M_j \;\; Iaj \;\; Fs_j \;\; Fv_j]^T \tag{3.25}$$

This property will be used to determine the minimum inertial parameters, which represent the identifiable inertial parameters, and will be used to reduce the computational cost of the dynamic model [Khalil 87c].

3- It can be proved that the matrix $[\frac{d}{dt}\mathbf{A} - 2\,\mathbf{h}(\mathbf{q},\dot{\mathbf{q}})]$ is anti-symmetric [Koditschek 84, Arimoto 84], where $\mathbf{h}\,\dot{\mathbf{q}} = \mathbf{B}\,\dot{\mathbf{q}}\dot{\mathbf{q}} + \mathbf{C}\,\dot{\mathbf{q}}^2$, represents a special form of the Coriolis and centrifugal forces.

4- The torque of joint j is a function of the inertial parameters of the links j, j+1,…,n.

5- The elements of **A** are bounded in **q.**

3.3.4. Lagrange method in SYMORO+

The menu **Dynamic** contains a function which computes the elements of the matrices **A, C, B** and **Q**, in customized form (using intermediate variables) or in expanded form. Appendix 2 presents the algorithm calculating these matrices in customized form (using intermediate variables). We note that the calculation of the models by customized form reduces considerably the computation cost (number of mathematical operations) of the model obtained.

The calculation of the matrix **A** and the coriolis, centrifugal and gravity torques are provided also in customized form by another function.

3.4. Newton-Euler method

The Newton-Euler method permits to calculate the dynamic model as defined in equation (3.1) without calculating beforehand the matrices **A**, **B**, **C** and **Q**. The efficient method proposed by Luh, Walker and Paul [Luh 79] constitutes an important step towards the computation of the dynamic model for the control applications. This method is based on two recursive calculations. The forward calculation, from the base to the terminal link, which determines the velocity, the accelerations and total forces and moments on each link, while the backward calculations, from link n to link 1, gives the joint torques by the use of the equilibrium equations of each link.

3.4.1. A Newton-Euler method linear in the inertial parameters

The algorithm of Luh, Walker and Paul [Luh 79, Luh 80b] gives the dynamic model as a function of the inertial parameters M_j, $\mathbf{S}_j$ and $\mathbb{I}_j$. In this section, we present a Newton-Euler algorithm based on the double recursive calculations of Luh, but it uses as inertial parameters the elements of M_j, $\mathbf{MS}_j$ and $\mathbf{J}_j$ [Khalil 87c, Khosla 86]. The dynamic model obtained is thus linear in the inertial parameters. This model can be used in the identification of the dynamic parameters and can be computed using the minimum inertial parameters (the base inertial parameters).

a) The forward recursive equations are given by:

The initial conditions for a robot with fixed base are:

$$\omega_0 = \mathbf{0} \ , \ \dot{\omega}_0 = \mathbf{0} \ \text{ and } \ \dot{\mathbf{V}}_0 = \mathbf{0} \tag{3.26}$$

The angular velocity of link j is given as:

$$\omega_j = \omega_{j-1} + \bar{\sigma}_j \dot{q}_j \mathbf{a}_j \tag{3.27}$$

The linear velocity of O_j is:

$$\mathbf{V}_j = \mathbf{V}_{j-1} + \omega_{j-1} \text{ x } \mathbf{L}_j + \sigma_j \dot{q}_j \mathbf{a}_j \tag{3.28}$$

The derivation of (3.27) and (3.28) gives the angular and linear acceleration as:

$$\dot{\omega}_j = \dot{\omega}_{j-1} + \bar{\sigma}_j (\ddot{q}_j \mathbf{a}_j + \omega_{j-1} \text{ x } \dot{q}_j \mathbf{a}_j) \tag{3.29}$$

$$\dot{\mathbf{V}}_j = \dot{\mathbf{V}}_{j-1} + \dot{\omega}_{j-1} \text{ x } \mathbf{L}_j + \omega_{j-1} \text{ x } (\omega_{j-1} \text{ x } \mathbf{L}_j) + \sigma_j (\ddot{q}_j \mathbf{a}_j + 2 \, \omega_{j-1} \text{ x } \dot{q}_j \mathbf{a}_j) \tag{3.30}$$

The center of mass linear acceleration is given as:

$$\dot{\mathbf{V}}\mathbf{g}_j = \dot{\mathbf{V}}_j + \dot{\omega}_j \text{ x } \mathbf{S}_j + \omega_j \text{ x } (\omega_j \text{ x } \mathbf{S}_j) \tag{3.31}$$

The total forces vector on link j is given by:

$$\mathbf{F}_j = M_j \dot{\mathbf{V}}\mathbf{g}_j \tag{3.32}$$

The total moment about the center of mass of link j is:

$$\mathbf{N}_j = \mathbb{I}_j \dot{\omega}_j + \omega_j \text{ x } (\mathbb{I}_j \omega_j) \tag{3.33}$$

Equations (3.32) and (3.33) could also be written as [Kleinfinger 86a]:

$$\mathbf{F}_j = M_j \dot{\mathbf{V}}_j + \dot{\omega}_j \text{ x } \mathbf{MS}_j + \omega_j \text{ x } (\omega_j \text{ x } \mathbf{MS}_j) \tag{3.34}$$

$$\mathbf{N}_j = \mathbf{No}_j - \mathbf{S}_j \text{ x } [\dot{\omega}_j \text{ x } \mathbf{MS}_j + \omega_j \text{ x } (\omega_j \text{ x } \mathbf{MS}_j)] \tag{3.35}$$

with:

$$\mathbf{No}_j = \mathbf{J}_j \dot{\omega}_j + \omega_j \text{ x } (\mathbf{J}_j \omega_j) \tag{3.36}$$

b) The backward recursive equations are given as:

The equilibrium of the forces on link j gives:

$$\mathbf{F}_j = \mathbf{f}_j - \mathbf{f}_{j+1} + M_j\, \mathbf{g} \tag{3.37}$$

Calculating the moment about the center of mass of link j we obtain:

$$\mathbf{N}_j = \mathbf{n}_j - \mathbf{n}_{j+1} + (\mathbf{S}_j - \mathbf{L}_{j+1}) \times \mathbf{f}_{j+1} - \mathbf{S}_j \times \mathbf{f}_j \tag{3.38}$$

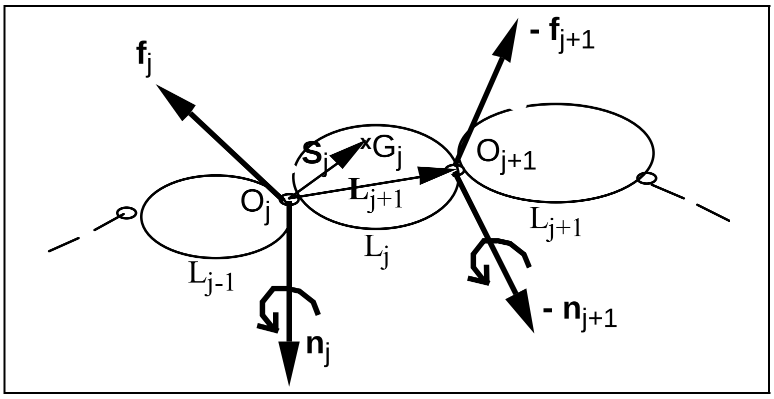


Figure 3.2. Forces and moments on link j

The gravity forces can be taken into account automatically by assuming that:

$$\dot{\mathbf{V}}_0 = -\,\mathbf{g} \tag{3.39}$$

Thus equations (3.37) and (3.38) will be transformed into the following:

$$\mathbf{f}_j = \mathbf{F}_j + \mathbf{f}_{j+1} \tag{3.40}$$

$$\mathbf{n}_j = \mathbf{N}_j + \mathbf{n}_{j+1} + \mathbf{L}_{j+1} \times \mathbf{f}_{j+1} + \mathbf{S}_j \times \mathbf{F}_j \tag{3.41}$$

These recursive equations will be initialized by the forces and moments of the terminal link on the environment $\mathbf{f}_{n+1}$ and $\mathbf{n}_{n+1}$.
Using (3.34) in (3.35), we obtain:

$$\mathbf{n}_j = \mathbf{N}_j + \mathbf{n}_{j+1} + \mathbf{L}_{j+1} \times \mathbf{f}_{j+1} + \mathbf{MS}_j \times \dot{\mathbf{V}}_j + \mathbf{S}_j \times [\dot{\omega}_j \times \mathbf{MS}_j + \omega_j \times (\omega_j \times \mathbf{MS}_j)] \tag{3.42}$$

and

$$\mathbf{n}_j = \mathbf{No}_j + \mathbf{n}_{j+1} + \mathbf{L}_{j+1} \times \mathbf{f}_{j+1} + \mathbf{MS}_j \times \dot{\mathbf{V}}_j \tag{3.43}$$

The torque (or the force) Γ_j, is obtained by projecting $\mathbf{n}_j$ (or $\mathbf{f}_j$) on the joint axis ($\mathbf{z}_j$), and by adding the friction and the motor inertia effects:

$$\Gamma_j = (\sigma_j\, \mathbf{f}_j + \bar{\sigma}_j\, \mathbf{n}_j)^T\, \mathbf{a}_j + Fs_j\, \mathrm{sign}(\dot{q}_j) + Fv_j\, \dot{q}_j + Ia_j\, \ddot{q}_j \tag{3.44}$$

3.4.2. Practical calculation of Newton-Euler algorithm

Projecting the previous equations into the link frames [Luh 80b], and defining the (3x3) matrix ${}^{j}\mathbf{U}_j$ such that:

$$ {}^{j}\mathbf{U}_j = {}^{j}\hat{\dot{\omega}}_j + {}^{j}\hat{\omega}_j\, {}^{j}\hat{\omega}_j \tag{3.45}$$

The algorithm of Newton-Euler will be given by [Khalil 87c] .

The forward recursive equations are given by, for j =1, ..., n:

$$ {}^{j}\omega_{j-1} = {}^{j}\mathbf{A}_{j-1}\, {}^{j-1}\omega_{j-1} \tag{3.46}$$

$$ {}^{j}\omega_{j} = {}^{j}\omega_{j-1} + \bar{\sigma}_j\, \dot{q}_j\, {}^{j}\mathbf{a}_j \tag{3.47}$$

$$ {}^{j}\dot{\omega}_{j} = {}^{j}\mathbf{A}_{j-1}\, {}^{j-1}\dot{\omega}_{j-1} + \bar{\sigma}_j\, (\ddot{q}_j\, {}^{j}\mathbf{a}_j + {}^{j}\omega_{j-1} \text{ x } \dot{q}_j\, {}^{j}\mathbf{a}_j) \tag{3.48}$$

$$ {}^{j}\dot{\mathbf{V}}_{j} = {}^{j}\mathbf{A}_{j-1}\, ({}^{j-1}\dot{\mathbf{V}}_{j-1} + {}^{j-1}\mathbf{U}_{j-1}\, {}^{j-1}\mathbf{P}_j) + \sigma_j\, (\, \ddot{q}_j\, {}^{j}\mathbf{a}_j + 2\, {}^{j}\omega_{j-1} \text{ x } \dot{q}_j\, {}^{j}\mathbf{a}_j) \tag{3.49}$$

$$ {}^{j}\mathbf{F}_{j} = M_j\, {}^{j}\dot{\mathbf{V}}_j + {}^{j}\mathbf{U}_j\, {}^{j}\mathbf{MS}_j \tag{3.50}$$

$$ {}^{j}\mathbf{No}_{j} = {}^{j}\mathbf{J}_j\, {}^{j}\dot{\omega}_j + {}^{j}\omega_j \text{ x } ({}^{j}\mathbf{J}_j\, {}^{j}\omega_j) \tag{3.51}$$

with $\omega_0 = \mathbf{0}$, $\dot{\omega}_0 = \mathbf{0}$ et $\dot{\mathbf{V}}_0 = -\,\mathbf{g}$

The backward recursive equations are given as, for j =n, ..., 1:

$$ {}^{j}\mathbf{f}_{j} = {}^{j}\mathbf{F}_j + {}^{j}\mathbf{f}_{j+1} \tag{3.52}$$

$$ {}^{j-1}\mathbf{f}_{j} = {}^{j-1}\mathbf{A}_j\, {}^{j}\mathbf{f}_j \tag{3.53}$$

$$ {}^{j}\mathbf{n}_{j} = {}^{j}\mathbf{No}_j + {}^{j}\mathbf{A}_{j+1}\, {}^{j+1}\mathbf{n}_{j+1} + {}^{j}\mathbf{P}_{j+1} \text{ x } {}^{j}\mathbf{f}_{j+1} + {}^{j}\mathbf{MS}_j \text{ x } {}^{j}\dot{\mathbf{V}}_j \tag{3.54}$$

$$ \Gamma_{j} = (\sigma_j\, {}^{j}\mathbf{f}_j + \bar{\sigma}_j\, {}^{j}\mathbf{m}_j)^T\, {}^{j}\mathbf{a}_j + F_{sj}\, \text{sign}(\dot{q}_j) + F_{vj}\, \dot{q}_j + Ia_j\, \ddot{q}_j \tag{3.55}$$

noting

$$ {}^{j}\omega_{j} = [\, \omega_{1,j} \quad \omega_{2,j} \quad \omega_{3,j}\,]^T \tag{3.56}$$

$$ {}^{j}\dot{\omega}_{j} = [\, \dot{\omega}_{1,j} \quad \dot{\omega}_{2,j} \quad \dot{\omega}_{3,j}\,]^T \tag{3.57}$$

${}^{j}\mathbf{U}_j$ can be calculated using (3.45) as:

$$ {}^{j}\mathbf{U}_j = \begin{bmatrix} -\omega_{3,j}^2 - \omega_{2,j}^2 & \omega_{1,j}\,\omega_{2,j} - \dot{\omega}_{3,j} & \omega_{1,j}\,\omega_{3,j} + \dot{\omega}_{2,j} \\ \omega_{1,j}\,\omega_{2,j} + \dot{\omega}_{3,j} & -\omega_{1,j}^2 - \omega_{3,j}^2 & \omega_{2,j}\,\omega_{3,j} - \dot{\omega}_{1,j} \\ \omega_{1,j}\,\omega_{3,j} - \dot{\omega}_{2,j} & \omega_{2,j}\,\omega_{3,j} + \dot{\omega}_{1,j} & -\omega_{1,j}^2 - \omega_{2,j}^2 \end{bmatrix} \tag{3.58}$$

3.5. Reducing the computation cost of the dynamic model

Many research works have been devoted to reduce the calculation cost of the dynamic model such that its calculation on-line may be possible with servo rate. We think that this problem can be now considered as solved. We present in this section the approaches used in **SYMORO+** in order to reduce the computational cost of the dynamic model.

3.5.1. The use of iterative symbolic method (customized method)

3.5.1.1. Principle of the method

The method consists of developing the previous equations, of section 3.4.2, to determine the components of each vector in terms of the elements of the right hand side vectors and matrices, and by substituting the constant values that are equal to 0, 1 or –1. As a big number of the geometric parameters are zero for the distances, or either 0 or ±1 for the sinus and cosines of angles, the model obtained will be more efficient than the general numerical algorithm [Kanade 84, Khalil 85a]. The method is based on the following rules:

1) use an intermediate variable for each component containing a mathematical operation and which will be used in subsequent calculation.

2) elimination of the intermediate variables which are not used in the computation of the desired motor torques. In **SYMORO+** this step is carried out by the function **Optimizer.**

The iterative symbolic model, customized calculation method, will reduce the computational cost of the dynamic model by about 50 % in many cases [Kleinfinger 86a].

3.5.1.2. Definition of the intermediate variables

In general, the dynamic model will be given in terms of the following intermediate variables:

* Variables resulting from the transformation matrices

We note:

$A21j = C\alpha_j\, Sj,$ $\qquad A31j = S\alpha_j\, Sj$

$A22j = C\alpha_j\, Cj,$ $\qquad A32j = S\alpha_j\, Cj$

From equation (1.1) we obtain:

$$^{j-1}\mathbf{A}_j = \begin{bmatrix} Cj & -Sj & 0 \\ A21j & A22j & -S\alpha_j \\ A31j & A32j & C\alpha_j \end{bmatrix}$$

with Cj and Sj denote $\cos(\theta_j)$ and $\cos(\theta_j)$ respectively.

If joint j is prismatic, the elements of ${}^{j-1}\mathbf{A}_j$ are constant. Furthermore if $\alpha_j = k(\pi / 2)$, with k integer, there will be no intermediate variables for the corresponding transformation matrix.

* Variables resulting from ${}^{j-1}\mathbf{P}_j$

$$\text{LOO2j} = -r_j \, S\alpha j \,, \quad \text{LOO3j} = r_j \, C\alpha_j$$

From relation (1.1), we can write:

$${}^{j-1}\mathbf{P}_j = [d_j \quad \text{LOO2j} \quad \text{LOO3j}]^T$$

If joint j is revolute, the elements LOOij will be constant and can be calculated off-line.

* Variables resulting from ${}^j\omega_j$

${}^j\omega_{j-1} = [\text{WI1j} \quad \text{WI2j} \quad \text{WI3j}]^T$ calculated by equation (3.46)

${}^j\omega_j = [\text{W1j} \quad \text{W2j} \quad \text{W3j}\ ]^T$ calculated by equation (3.47)

$\dot{q}_j$ is denoted QPj.

* Variables resulting from ${}^j\dot{\omega}_j$

${}^j\dot{\omega}_j = [\text{WP1j} \quad \text{WP2j} \quad \text{WP3j}]^T$ calculated from equation (3.48)

$\ddot{q}_j$ is denoted QDPj.

* Variables resulting from ${}^j\mathbf{U}_j$

The matrix ${}^j\mathbf{U}_j$ can be written as:

$${}^j\mathbf{U}_j = \begin{bmatrix} \text{U11j} & \text{U12j} & \text{U13j} \\ \text{U21j} & \text{U22j} & \text{U23j} \\ \text{U31j} & \text{U32j} & \text{U33j} \end{bmatrix}$$

From equation (3.58), we obtain:

$${}^j\mathbf{U}_j = \begin{bmatrix} \text{DV33j} + \text{DV22j} & \text{DV12j} - \text{WP3j} & \text{DV13j} + \text{WP2j} \\ \text{DV12j} + \text{WP3j} & \text{DV11j} + \text{DV33j} & \text{DV23j} - \text{WP1j} \\ \text{DV13j} - \text{WP2j} & \text{DV23j} + \text{WP1j} & \text{DV11j} + \text{DV22j} \end{bmatrix}$$

where:

$$\text{DV11j} = -\,\text{W1j}\,\text{W1j} \,, \quad \text{DV22j} = -\,\text{W2j}\,\text{W2j} \,, \quad \text{DV33j} = -\,\text{W3j}\,\text{W3j},$$
$$\text{DV12j} = \text{W1j}\,\text{W2j} \,, \quad \text{DV13j} = \text{W1j}\,\text{W3j} \,, \quad \text{DV23j} = \text{W2j}\,\text{W3j}.$$

* Variables resulting from ${}^{j}\dot{\mathbf{V}}_{j}$

We note:

$$ {}^{j-1}\dot{\mathbf{V}}_{j-1} + {}^{j-1}\mathbf{U}_{j-1}\, {}^{j-1}\mathbf{P}_{j} = \begin{bmatrix} \mathrm{VSP1j} \\ \mathrm{VSP2j} \\ \mathrm{VSP3j} \end{bmatrix} $$

then we calculate ${}^{j}\dot{\mathbf{V}}_{j}$ using equation (3.49):

$$ {}^{j}\dot{\mathbf{V}}_{j} = [\mathrm{VP1j} \quad \mathrm{VP2j} \quad \mathrm{VP3j}]^{T} $$

* Variables resulting from ${}^{j}\mathbf{F}_{j}$

It will be calculated using equation (3.50), we denote:

$$ {}^{j}\mathbf{F}_{j} = [\mathrm{F1j} \quad \mathrm{F2j} \quad \mathrm{F3j}]^{T} $$

The mass of link j is denoted Mj, and the components of the first moments of link j are denoted MXj, MYj, and MZj.

* Variables resulting from ${}^{j}\mathbf{No}_{j}$

We suppose that:

$$ {}^{j}\mathbf{J}_{j} = \begin{bmatrix} \mathrm{XXj} & \mathrm{XYj} & \mathrm{XZj} \\ \mathrm{XYj} & \mathrm{YYj} & \mathrm{YZj} \\ \mathrm{XZj} & \mathrm{YZj} & \mathrm{ZZj} \end{bmatrix} $$

and:

$$ \mathrm{PIS1j} = \mathrm{ZZj} - \mathrm{YYj}\ ,\ \mathrm{PIS2j} = \mathrm{XXj} - \mathrm{ZZj}\ ,\ \mathrm{PIS3j} = \mathrm{YYj} - \mathrm{XXj} $$

From equation (3.51), we obtain ${}^{j}\mathbf{No}_{j}$ as function of the elements of ${}^{j}\mathbf{U}_{j}$ and of DVikj:

$$ {}^{j}\mathbf{No}_{j} = [\mathrm{NO1j} \quad \mathrm{NO2j} \quad \mathrm{NO3j}]^{T} $$

$$ = $$

$$ \begin{bmatrix} \mathrm{WP1j} & \mathrm{XXj+DV23j} & \mathrm{PIS1j+U21j} & \mathrm{XZj-U31j} & \mathrm{XYj+(DV33j-DV22j)} & \mathrm{YZj} \\ \mathrm{WP2j} & \mathrm{YYj+DV13j} & \mathrm{PIS2j+U32j} & \mathrm{XYj-U12j} & \mathrm{YZj+(DV11j-DV33j)} & \mathrm{XZj} \\ \mathrm{WP3j} & \mathrm{ZZj+DV12j} & \mathrm{PIS3j+U13j} & \mathrm{YZj-U23j} & \mathrm{XZj+(DV22j-DV11j)} & \mathrm{XYj} \end{bmatrix} $$

The PISkj are constant and can be calculated off-line.

* Variables resulting from ${}^{j}\mathbf{f}_j$ and ${}^{j}\mathbf{n}_j$

The external forces and moments exerted by link j on the environment are denoted:

$$
\begin{aligned}
{}^{n}\mathbf{f}_{tj} &= [\, FXj \quad FY_j \quad FZj \,]^T \\
{}^{n}\mathbf{n}_{tj} &= [\, CXj \quad CYj \quad CZj \,]^T
\end{aligned}
$$

We calculate first ${}^{j}\mathbf{f}_j$ and ${}^{j}\mathbf{n}_j$ using equations (3.52) and (3.54), the following variables will be defined:

$$
\begin{aligned}
{}^{j}\mathbf{f}_j &= [\, E1j \quad E2j \quad E3j \,]^T \\
{}^{j}\mathbf{n}_j &= [\, N1j \quad N2j \quad N3j \,]^T
\end{aligned}
$$

${}^{j-1}\mathbf{f}_j$ is then calculated using equation (3.53), with the following notations:

$$
{}^{j-1}\mathbf{A}_j\, {}^{j}\mathbf{f}_j = \begin{bmatrix} SDJ1j \\ SDJ2j \\ SDJ3j \end{bmatrix}
$$

* Variables resulting from Γ_j

Equation (3.55) gives Γ_j, it will be denoted GAMj.

3.5.2. The use of the minimum inertial parameters

The minimum set of inertial parameters, also known as the base inertial parameters, represent the independent parameters from which the dynamic model can be calculated. They can be deduced from the standard parameters, the elements of $\mathbf{J}_j$, $\mathbf{MS}_j$, M_j by eliminating those that have no effect on the dynamic model and by grouping some others. In the following we present a method to determine this set of minimum inertial parameters, its use in the dynamic model will contribute in reducing its computational cost. Furthermore, the determination of these parameters is essential for the evaluation of the inertial parameters by identification, in fact they also represent the only identifiable parameters.

3.5.2.1. Study of the grouping and elimination of inertial parameters

Since the dynamic model is linear in the inertial parameters, we can write that:

$$
\Gamma = \sum_{i=1}^{c} (\mathbf{A}^i\, \ddot{\mathbf{q}} + \mathbf{B}^i\, \dot{\mathbf{q}}\dot{\mathbf{q}} + \mathbf{C}^i\, \dot{\mathbf{q}}^2 + \mathbf{Q}^i)\, K_i = \sum_{i=1}^{c} \mathbf{D}^i K_i = \mathbf{D}\,\mathbf{K} \qquad (3.59)
$$

where:

$\mathbf{D}$ is (n x c) matrix, function of $\mathbf{q}$, $\dot{\mathbf{q}}$ and $\ddot{\mathbf{q}}$,
$\mathbf{K}$ is (c x 1) vector containing the inertial parameters, with c = 11n, corresponding to six elements of the inertia tensor, three elements for the first moments, the mass and the rotor motor inertia per link,
$\mathbf{D}^i$ is the i^th column of $\mathbf{D}$,

$$\mathbf{A}^i = \frac{\partial \mathbf{A}}{\partial K_i}\ ,\quad \mathbf{B}^i = \frac{\partial \mathbf{B}}{\partial K_i}\ ,\quad \mathbf{C}^i = \frac{\partial \mathbf{C}}{\partial K_i}\ ,\quad \mathbf{Q}^i = \frac{\partial \mathbf{Q}}{\partial K_i} \tag{3.60}$$

From equation (3.59) we deduce that:

if $\mathbf{D}^i = \mathbf{0}$, the parameter K_i has no effect on $\mathbf{\Gamma}$. (3.61)

The condition (3.61) means that the parameter K_i can be put equal to zero without affecting the value of $\mathbf{\Gamma}$.

From equation (3.59) we deduce that if some columns of $\mathbf{D}$ are dependent such that:

$$\mathbf{D}^i = t_{i1}\,\mathbf{D}^{i1} + \ldots + t_{ir}\,\mathbf{D}^{ir} \tag{3.62}$$

with t_{ik} are constant.

Then, the same $\mathbf{\Gamma}$ is obtained if we use the parameters K_i, K_{i1}, …, K_{ir}, or if K_i is eliminated and the parameters K_{ij} are replaced by KR_{ij}, for j = 1, …, r.
where:

$$KR_{ij} = K_{ij} + t_{ij}\,K_i \qquad \text{with } j = 1, \ldots, r. \tag{3.63}$$

Using the previous conditions (3.61) and (3.62), we can determine a set of independent parameters $\mathbf{K_m}$, equivalent to $\mathbf{K}$.

A Lagrangian model as (3.59) will be more efficient if it is computed using $\mathbf{K_m}$ instead of $\mathbf{K}$, because the calculation of $\mathbf{D}^i$ corresponding to the eliminated parameters will be avoided. The calculation of Newton-Euler algorithm in customized form using $\mathbf{K_m}$ will be also more efficient than the use of $\mathbf{K}$ [Khalil 86c, Khalil 87c].

The determination of $\mathbf{K_m}$ is based on the study of the linear combinations of the columns $\mathbf{D}^i$. But, from the relations (3.6) and (3.7), we deduce that the linear relations between the different $\mathbf{A}^i$ will be valid also on the corresponding $\mathbf{B}^i$ and $\mathbf{C}^i$. Thus the foregoing conditions can be rewritten as:

– the necessary conditions to eliminate a parameter K_i are:

$$\mathbf{A}^i = \mathbf{0} \tag{3.64}$$

and,

$$\mathbf{Q}^i = \mathbf{0} \tag{3.65}$$

– the necessary conditions to group a parameter K_i with the parameters K_{ij}, for j = 1, …, r are:

$$\mathbf{A}^i = t_{i1}\,\mathbf{A}^{i1} + \dots + t_{ir}\,\mathbf{A}^{ir} \tag{3.66}$$

and,

$$\mathbf{Q}^i = t_{i1}\,\mathbf{Q}^{i1} + \dots + t_{ir}\,\mathbf{Q}^{ir} \tag{3.67}$$

3.5.2.2. Direct symbolic method for the calculation of minimum inertial parameters

The foregoing method for the determination of the minimum inertial parameters is difficult to carry out when n>3 owing to the complexity of the expressions of $\mathbf{A}^i$ and $\mathbf{Q}^i$. We have resolved this problem by proposing the following direct symbolic and recursive method [Gautier 88a] and by a numerical method [Gautier 91]. In the following we present the symbolic method, the numerical method is presented in Appendix 3, both methods are implemented in **SYMORO+**.

Since the kinetic energy and the potential energy are linear in the inertial parameters. Thus we can write:

$$H = E + U = \sum_{i=1}^{m} \frac{\partial H}{\partial K_i} K_i = \sum_{i=1}^{m} h_i K_i \tag{3.68}$$

– the necessary and sufficient conditions to eliminate the parameter K_i is given by:

$$h_i = \text{constant} \tag{3.69}$$

– the necessary and sufficient condition to group the parameter K_i is:

$$h_i = t_{i1}\,h_{i1} + \dots + t_{ir}\,h_{ir} = \sum_{k=1}^{r} t_{ik}\,h_{ik} \tag{3.70 a}$$

with t_{ik} constant,

The grouped parameters will be given as:

$$KR_{ij} = K_{ij} + t_{ij}\,K_i \quad \text{with } j = 1, \dots, r. \tag{3.70 b}$$

Let the inertial parameters of link j be given by the vector $\mathbf{K}^j$, without considering the rotor inertia,

$$\mathbf{K}^j = [\,XX_j\ \ XY_j\ \ XZ_j\ \ YY_j\ \ YZ_j\ \ ZZ_j\ \ MX_j\ \ MY_j\ \ MZ_j\ \ M_j\,]^T \tag{3.71}$$

let us define $\mathbf{h}^j$ as follows:

$$\mathbf{h}^j = \left[\frac{\partial H}{\partial XX_j}\ \frac{\partial H}{\partial XY_j}\ \frac{\partial H}{\partial XZ_j}\ \frac{\partial H}{\partial YY_j}\ \frac{\partial H}{\partial YZ_j}\ \frac{\partial H}{\partial ZZ_j}\ \frac{\partial H}{\partial MX_j}\ \frac{\partial H}{\partial MY_j}\ \frac{\partial H}{\partial MZ_j}\ \frac{\partial H}{\partial M_j}\right]$$

thus:

$$\mathbf{h}^j = [h_{XXj}\ h_{XYj}\ \ h_{XZj}\ h_{YYj}\ \ h_{YZj}\ h_{ZZj}\ \ h_{MXj}\ h_{MYj}\ \ h_{MZj}\ h_{Mj}] \tag{3.72}$$

where (from the energy expressions):

$$\begin{aligned} &h_{XXj} = \frac{1}{2}\ \omega_{1,j}{}^2, \quad h_{XYj} = \omega_{1,j}\omega_{2,j}, \quad h_{XZj} = \omega_{1,j}\omega_{3,j} \\ &h_{YYj} = \frac{1}{2}\ \omega_{2,j}{}^2, \quad h_{YZj} = \omega_{2,j}\omega_{3,j}, \quad h_{ZZj} = \frac{1}{2}\ \omega_{3,j}{}^2 \\ &h_{MXj} = \omega_{3,j}\, V_{2,j} - \omega_{2,j}\, V_{3,j} - \mathbf{g}^T\ {}^0\mathbf{s}_j, \quad h_{MYj} = \omega_{1,j}\, V_{3,j} - \omega_{3,j}\, V_{1,j} - \mathbf{g}^T\ {}^0\mathbf{n}_j \\ &h_{MZj} = \omega_{2,j}\, V_{1,j} - \omega_{1,j}\, V_{2,j} - \mathbf{g}^T\ {}^0\mathbf{a}_j, \quad h_{Mj} = \frac{1}{2}{}^j\mathbf{V}_j{}^T\, {}^j\mathbf{V}_j - \mathbf{g}^T\ {}^0\mathbf{P}_j \end{aligned} \tag{3.73}$$

It can be seen also that:

$$h_{Iaj} = \frac{1}{2}\dot{q}_j{}^2 \tag{3.74}$$

In the previous expressions we noted:

$${}^j\boldsymbol{\omega}_j = [\omega_{1,j}\ \ \omega_{2,j}\ \ \omega_{3,j}]^T \tag{3.75}$$

$${}^j\mathbf{V}_j = [V_{1,j}\ \ V_{2,j}\ \ V_{3,j}]^T \tag{3.76}$$

Calculating $\mathbf{h}^j$ as a function of $\mathbf{h}^{j-1}$, it can be seen that [Gautier 90a,b]:

$$\mathbf{h}^j = \mathbf{h}^{j-1}\ {}^{j-1}\lambda_j\,(q_j) + \dot{q}_j\ \mathbf{f}^j\,(\mathbf{q}, \dot{\mathbf{q}}) \tag{3.77}$$

with ${}^{j-1}\lambda_j$ is a (10x10) matrix and $\mathbf{f}^j$ is of dimension (1x10).

The elements of ${}^{j-1}\lambda_j$ are given in table 3.1, in terms of the geometric parameters (d_j, r_j, θ_j, α_j).

The vector $\mathbf{f}^j$ is given as:

- for j revolute:

$$\mathbf{f}^j = \begin{bmatrix} 0 & 0 & \omega_{1,j} & 0 & \omega_{2,j} & \omega_{3,j} - \frac{1}{2}\dot{q}_j & V_{2,j} & -V_{1,j} & 0 & 0 \end{bmatrix} \tag{3.78}$$

- for j prismatic:

$$\mathbf{f}^j = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & -\omega_{2,j} & \omega_{1,j} & 0 & V_{3,j} - \frac{1}{2}\dot{q}_j \end{bmatrix} \tag{3.79}$$

Remark: It is to be noted that the (10x10) matrix ${}^{j-1}\lambda_j$ permits to express the inertial parameters of link j into frame j-1, such that [Khalil 90a]:

$${}^{j-1}\mathbf{K}^j = {}^{j-1}\lambda_j\ {}^j\mathbf{K}^j \tag{3.80}$$

Table 3.1: *Expression of the elements of the matrix* ${}^{j-1}\lambda_j$

$CC\theta$	$-2CS\theta$	0	$SS\theta$	0	0	0	0	$2r$	r^2
$CS\theta C\alpha$	$(CC\theta-SS\theta)C\alpha$	$-C\theta S\alpha$	$-CS\theta C\alpha$	$S\theta S\alpha$	0	$-dS\theta C\alpha+rC\theta S\alpha$	$-dC\theta C\alpha-rS\theta S\alpha$	$dS\alpha$	$drS\alpha$
$CS\theta S\alpha$	$(CC\theta-SS\theta)S\alpha$	$C\theta C\alpha$	$-CS\theta S\alpha$	$-S\theta C\alpha$	0	$-dS\theta S\alpha-rC\theta C\alpha$	$-dC\theta S\alpha+rS\theta C\alpha$	$-dC\alpha$	$-drC\alpha$
$SS\theta CC\alpha$	$2CS\theta CC\alpha$	$-2S\theta CS\alpha$	$CC\theta CC\alpha$	$-2C\theta CS\alpha$	$SS\alpha$	$2(dC\theta+rS\theta CS\alpha)$	$2(-dS\theta+rC\theta CS\alpha)$	$2rCC\alpha$	$d^2+r^2CC\alpha$
$SS\theta CS\alpha$	$2CS\theta CS\alpha$	$S\theta(CC\alpha-SS\alpha)$	$CC\theta CS\alpha$	$C\theta(CC\alpha-SS\alpha)$	$-CS\alpha$	$rS\theta(SS\alpha-CC\alpha)$	$rC\theta(SS\alpha-CC\alpha)$	$2rCS\alpha$	$r^2CS\alpha$
$SS\theta SS\alpha$	$2CS\theta SS\alpha$	$2S\theta CS\alpha$	$CC\theta SS\alpha$	$2C\theta CS\alpha$	$CC\alpha$	$2(dC\theta-rS\theta CS\alpha)$	$2(-dS\theta-rC\theta CS\alpha)$	$2rSS\alpha$	$d^2+r^2SS\alpha$
0	0	0	0	0	0	$C\theta$	$-S\theta$	0	d
0	0	0	0	0	0	$S\theta C\alpha$	$C\theta C\alpha$	$-S\alpha$	$-rS\alpha$
0	0	0	0	0	0	$S\theta S\alpha$	$C\theta S\alpha$	$C\alpha$	$rC\alpha$
0	0	0	0	0	0	0	0	0	1

The subscript j of the geometric parameters has been omitted.
The following notations are used: $CS^* = \cos(^*)\sin(^*)$, $CC^* = \cos^2(^*)$, $SS^* = \sin^2(^*)$

3.5.2.2.1. General grouping relations of the inertial parameters

Using relation (3.77), the following cases are considered:

1- If joint j is revolute: The following three relations are always satisfied:

a- $h_{Mj} = \mathbf{h}^{j-1}\ {}^{j-1}\lambda_j^{10}$ (3.81-a)

b- $h_{MZj} = \mathbf{h}^{j-1}\ {}^{j-1}\lambda_j^{9}$ (3.81-b)

c- $h_{XXj} + h_{YYj} = \mathbf{h}^{j-1}\ {}^{j-1}\lambda_j^{1+4}$ (3.81-c)

where:

$${}^{j-1}\lambda_j^{1+4} = {}^{j-1}\lambda_j^{1} + {}^{j-1}\lambda_j^{4} \tag{3.82}$$

where ${}^{j-1}\lambda_j^{m}$ gives the m^th column of ${}^{j-1}\lambda_j$ as a function of the geometric parameters (α_j, d_j, θ_j, r_j) defining frame j with respect to frame j-1. The expressions of the columns (1+4), 9, and 10 of ${}^{j-1}\lambda_j$ are constant and given as follows:

$${}^{j-1}\lambda_j^{10} = [r_j^2 \quad d_j r_j S\alpha_j \quad -d_j r_j C\alpha_j \quad (d_j^2+r_j^2 CC\alpha_j) \quad r_j^2 CS\alpha_j \quad (d_j^2+r_j^2 SS\alpha_j) \quad d_j \quad -r_j S\alpha_j \quad r_j C\alpha_j \quad 1]^T \tag{3.83}$$

$${}^{j-1}\lambda_j^{9} = [2r_j \quad d_j S\alpha_j \quad -d_j C\alpha_j \quad 2r_j CC\alpha_j \quad 2r_j CS\alpha_j \quad 2r_j SS\alpha_j \quad 0 \quad -S\alpha_j \quad C\alpha_j \quad 0]^T \tag{3.84}$$

$${}^{j-1}\lambda_j^{1+4} = [1 \quad 0 \quad 0 \quad CC\alpha_j \quad CS\alpha_j \quad SS\alpha_j \quad 0 \quad 0 \quad 0 \quad 0]^T \tag{3.85}$$

From equations (3.81), we deduce that three parameters can be grouped, from the standard parameters. The choice of the three parameters to be eliminated is not unique. We choose to group the parameters M_j, MZ_j, YY_j. This choice is always possible because the corresponding coefficients are equal to one. The grouping relations are obtained using (3.70) and (3.81) as follows:

$$XXR_j = XX_j - YY_j \tag{3.86-a}$$

and

$$\mathbf{KR}^{j-1} = \mathbf{K}^{j-1} + YY_j\ {}^{j-1}\lambda_j^{1+4} + MZ_j\ {}^{j-1}\lambda_j^{9} + M_j\ {}^{j-1}\lambda_j^{10} \tag{3.86-b}$$

2- If joint j is prismatic: the following six relations are obtained:

a- $h_{XXj} = \mathbf{h}^{j-1}\ {}^{j-1}\lambda_j^{1}$ (3.87-a)

b- $h_{XYj} = \mathbf{h}^{j-1}\ {}^{j-1}\lambda_j^{2}$ (3.87-b)

$\ldots = \ldots$

f- $h_{ZZj} = \mathbf{h}^{j-1}\ {}^{j-1}\lambda_j^{6}$ (3.87-f)

The (10x1) vectors ${}^{j-1}\lambda_j^1, \ldots, {}^{j-1}\lambda_j^6$ can be deduced from table 3.3, they are functions of the constant geometric parameters, thus six parameters can be grouped, we choose to group the parameters of the inertia matrix of link j on those of link j-1. The grouping relations are:

$$\mathbf{KR}^{j-1} = \mathbf{K}^{j-1} + XX_j\, {}^{j-1}\lambda_j^1 + YY_j\, {}^{j-1}\lambda_j^2 + \ldots + ZZ_j\, {}^{j-1}\lambda_j^6 \tag{3.88}$$

We note that equation (3.88) is equivalent to:

$${}^{j-1}\mathbf{JR}_{j-1} = {}^{j-1}\mathbf{J}_{j-1} + {}^{j-1}\mathbf{A}_j\, {}^{j}\mathbf{J}_j\, {}^{j}\mathbf{A}_{j-1} \tag{3.89}$$

This relation can be directly deduced using the fact that when joint j is prismatic, the angular velocity of link j is equal to the angular velocity of link j-1.

The following results are deduced from the foregoing formulas and from the conditions (3.70), some of them will be given without demonstration.

Theorem 1: If joint j is revolute, the parameters (YY_j, MZ_j, M_j) can be grouped with the parameters of links j and j-1. The grouped parameters are given as:

$$\begin{aligned}
XXR_j &= XX_j - YY_j \\
XXR_{j-1} &= XX_{j-1} + YY_j + r_j^2\, M_j + 2\, r_j\, MZ_j \\
XYR_{j-1} &= XY_{j-1} + d_j\, S\alpha_j\, MZ_j + d_j\, r_j\, S\alpha_j\, M_j \\
XZR_{j-1} &= XZ_{j-1} - d_j\, C\alpha_j\, MZ_j - d_j\, r_j\, C\alpha_j\, M_j \\
YYR_{j-1} &= YY_{j-1} + CC\alpha_j\, YY_j + 2\, r_j\, CC\alpha_j\, MZ_j + (d_j^2 + r_j^2\, CC\alpha_j)\, M_j \\
YZR_{j-1} &= YZ_{j-1} + CS\alpha_j\, YY_j + 2\, r_j\, CS\alpha_j\, MZ_j + r_j^2\, CS\alpha_j\, M_j \\
ZZR_{j-1} &= ZZ_{j-1} + SS\alpha_j\, YY_j + 2\, r_j\, SS\alpha_j\, MZ_j + (d_j^2 + r_j^2\, SS\alpha_j)\, M_j \\
MXR_{j-1} &= MX_{j-1} + d_j\, M_j \\
MYR_{j-1} &= MY_{j-1} - S\alpha_j\, MZ_j - r_j\, S\alpha_j\, M_j \\
MZR_{j-1} &= MZ_{j-1} + C\alpha_j\, MZ_j + r_j\, C\alpha_j\, M_j \\
MR_{j-1} &= M_{j-1} + M_j
\end{aligned} \tag{3.90}$$

Theorem 2: If joint j is prismatic, the parameters of the inertia tensor $\mathbf{J}_j$ can be regrouped to the elements of the inertia tensor $\mathbf{J}_{j-1}$, the grouped parameters are given as:

$$\begin{aligned}
XXR_{j-1} &= XX_{j-1} + CC\theta_j\, XX_j - 2\, CS\theta_j\, XY_j + SS\theta_j\, YY_j \\
XYR_{j-1} &= XY_{j-1} + CS\theta_j\, C\alpha_j\, XX_j + (CC\theta_j - SS\theta_j)\, C\alpha_j\, XY_j - C\theta_j\, S\alpha_j\, XZ_j \\
&\quad - CS\theta_j\, C\alpha_j\, YY_j + S\theta_j\, S\alpha_j\, YZ_j \\
XZR_{j-1} &= XZ_{j-1} + CS\theta_j\, S\alpha_j\, XX_j + (CC\theta_j - SS\theta_j)\, S\alpha_j\, XY_j + C\theta_j\, C\alpha_j\, XZ_j \\
&\quad - CS\theta_j\, S\alpha_j\, YY_j - S\theta_j\, C\alpha_j\, YZ_j \\
YYR_{j-1} &= YY_{j-1} + SS\theta_j\, CC\alpha_j\, XX_j + SS\alpha_j\, ZZ_j + 2CS\theta_j\, CC\alpha_j\, XY_j - 2S\theta_j\, CS\alpha_j\, XZ_j \\
&\quad + CC\theta_j\, CC\alpha_j\, YY_j - 2C\theta_j\, CS\alpha_j\, YZ_j
\end{aligned}$$

$$\begin{aligned} YZR_{j-1} &= YZ_{j-1} + SS\theta_j\, CS\alpha_j\, XX_j - CS\alpha_j\, ZZ_j + 2CS\theta_j\, CS\alpha_j\, XY_j \\ &\quad + S\theta_j\, (CC\alpha_j - SS\alpha_j)\, XZ_j + CC\theta_j\, CS\alpha_j\, YY_j + C\theta_j\, (CC\alpha_j - SS\alpha_j)\, YZ_j \\ ZZR_{j-1} &= ZZ_{j-1} + SS\theta_j\, SS\alpha_j\, XX_j + 2CS\theta_j\, SS\alpha_j\, XY_j + 2S\theta_j\, CS\alpha_j\, XZ_j \\ &\quad + CC\theta_j\, SS\alpha_j\, YY_j + 2C\theta_j\, CS\alpha_j\, YZ_j + CC\alpha_j\, ZZ_j \end{aligned} \tag{3.91}$$

Theorem 3: If r_1 is the first rotational joint, and r_2 is the first rotational joint not parallel to r_1, we find that the previous general regrouping relations are the only possible grouping of the parameters of links $r_2,\ldots,n$. For the prismatic links between links r_1 and r_2 particular grouping of parameters may take place, the following two cases are considered [Khalil 94a]:

a- The axis of j is not parallel to the axis of r1 for (r1 < j < r2)

It can be seen that:

$$a_{xr1}\ h_{MXj} + {}^{j}a_{yr1}\ h_{MYj} + {}^{j}a_{zr1}\ h_{MZj} = \text{constant} \tag{3.92}$$

The grouping relation or elimination are given in table 3.2 in terms of the elements ${}^{j}a_{xr1}$, ${}^{j}a_{yr1}$ and ${}^{j}a_{zr1}$.

${}^{j}a_{zr1} \neq 0$	${}^{j}a_{zr1} = 0, {}^{j}a_{xr1}\, {}^{j}a_{yr1} \neq 0$	${}^{j}a_{zr1} = 0, {}^{j}a_{xr1} = 0$	${}^{j}a_{zr1} = 0, {}^{j}a_{yr1} = 0$
$MXR_j = MX_j - \frac{{}^{j}a_{xr1}}{{}^{j}a_{zr1}} MZ_j$ $MYR_j = MY_j - \frac{{}^{j}a_{yr1}}{{}^{j}a_{zr1}} MZ_j$	$MXR_j = MX_j - \frac{{}^{j}a_{xr1}}{{}^{j}a_{yr1}} MY_j$	$MY_j = 0$	$MX_j = 0$

Table 3.2

Thus, in this case, one parameter will be always grouped or has no effect on the dynamic model.

b- The axis of j is parallel to the axis of r1 for (r1 < j < r2)

It can be seen that the parameter MZj has no effect on the dynamic model, and the following linear relation is obtained:

$$\mathbf{h}_{MSj}^{T} = {}^{j}\mathbf{A}_{j-1}\ \mathbf{h}_{MSj-1}^{T} - [\ 2P_x\, h_{ZZi} \quad 2P_y\, h_{ZZi} \quad 0\ ]^{T} \tag{3.93}$$

with:

$${}^{j}\mathbf{P}_{j-1} = [\ P_x \quad P_y \quad P_z\ ]^{T} = [\ -d_j\, C\theta_j \quad d_j\, S\theta_j \quad -r_j\ ]^{T}$$

We can group the parameters MX_j and MY_j with the parameters of the first moments of link j-1 and with the parameter ZZ_s of link s. The grouped parameters are given as:

$$\begin{aligned}
MXR_{j-1} &= MX_{j-1} + C\theta_j\, MX_j - S\theta_j\, MY_j \\
MYR_{j-1} &= MY_{j-1} + S\theta_j\, C\alpha_j\, MX_j + C\theta_j\, C\alpha_j\, MY_j \\
MZR_{j-1} &= MZ_{j-1} + S\theta_j\, S\alpha_j\, MX_j + C\theta_j\, S\alpha_j\, MY_j \\
ZZR_s &= ZZ_s + 2\, d_j\, C\theta_j\, MX_j - 2\, d_j\, S\theta_j\, MY_j
\end{aligned} \tag{3.94}$$

where s is the first revolute joint to link j from the base side.

Theorem 4: The inertia Ia_1 will be grouped with ZZ_1 if joint 1 is rotational, or to M_1 if joint 1 is prismatic and its joint axis is perpendicular to gravity (whereas, the inertia Ia_2 will be grouped with ZZ_2 if $\sigma_2 = 0$ and $\alpha_2 = \pm 90°$).

Theorem 5: From the previous theorems, it can be seen that the number of minimum inertial parameters of links m_m is given as:

$$m_m \leq [7\, \Sigma\bar{\sigma}_i + 4\, \Sigma\sigma_i - 3 - 2\bar{\sigma}_1]$$

where $\Sigma\bar{\sigma}_i$ and $\Sigma\sigma_i$ gives the number of revolute and prismatic joints respectively.

3.5.2.3. Practical use of the grouping relations

The algorithm of the determination of the minimum inertial parameters will be carried out using the following rules [Khalil 94a]:

1- Use the general grouping relations, theorem 1 and 2, to eliminate the following parameters:
   a- YY_j , MZ_j , M_j , if joint j is revolute for j = n,…,1
   b- XX_j , XY_j , XZ_j , YY_j , YZ_j , ZZ_j , if joint j is prismatic for j = n,…,1
2- Eliminate MZ_j and group MX_j and MY_j using (3.94) if j is prismatic and $\mathbf{a}_j$ // $\mathbf{a}_{r1}$ for $r_1 < j < r_2$.
3- Group or eliminate one of the parameters MX_j , MY_j , MZ_j, if $\mathbf{a}_{r1}$ is not parallel to $\mathbf{a}_j$, and j is prismatic and $r_1 < j < r_2$, using relation (3.92) or table 3.2 .
4- Eliminate XX_j , XY_j , XZ_j , YZ_j if j is revolute and $r_1 \leq j < r_2$ (the axes of these joints are parallel to r_1). We note that YY_1 has also been eliminated in rule 1 .
5- Eliminate MX_j , MY_j, they have no effect when j is revolute for ($r_1 \leq j < r_2$) and ($\mathbf{a}_j$ is along $\mathbf{a}_{r1}$) and ($\mathbf{a}_{r1}$ // $\mathbf{a}_i$ // $\mathbf{g}$ for all i < j). We note that MZ_j has also been eliminated in rule 1 .
6- Eliminate MX_j, MY_j, MZ_j, they have no effect, when $j < r_1$ (they represent the prismatic links before r_1, where $\boldsymbol{\omega}_j = \mathbf{0}$).

Taking into account the general grouping parameters, we find that the number of operations of the dynamic model in the case of general robot of n revolute joints is 92n–127 multiplications and 81n–117 additions (n >2), which gives 425 multiplications and 369 additions if n = 6. In the case of industrial robots this computation cost will be reduced considerably [Khalil 87c].

3.5.3. The Newton-Euler dynamic model in SYMORO+

The algorithm presented in section (3.5.1) can be obtained using the menu **Dynamic** and the function "Newton-Euler". The base inertial parameters are given by the function "base inertial parameters" of the menu **Identification**, which contains two algorithms: a symbolic method (section 3.5.2.2) and a numerical method (see Appendix 3). In the numerical method the acceleration of gravity and the constant geometric parameters must be given numerically either directly while defining the robot or in a particular file.

3.6. Main applications of the dynamic model

The following main applications of the dynamic model will be presented in this section:
- the simulation,
- the choice of robot motors,
- the identification of the inertial parameters of robot,
- the control.

3.6.1. Simulation

The direct dynamic model is used to simulate the dynamics of a robot. Using equation (3.5) and taking into account equation (3.22) we obtain:

$$\ddot{\mathbf{q}} = \mathbf{A}^{-1} [\mathbf{\Gamma} - \mathbf{H}] \qquad (3.95)$$

with:

$$\mathbf{H} = (\mathbf{B}\, \dot{\mathbf{q}}\dot{\mathbf{q}} + \mathbf{C}\, \dot{\mathbf{q}}^{\,2} + \mathbf{Q} + \mathbf{J}^{T}\, \mathbf{F}_{tn})$$

To realize such simulation the matrix **A** and the vector **H** are needed. The matrix **A** can be obtained from the menu **Dynamic** of **SYMORO+ using** the function "inertia matrix"**,** the calculation of **H** is also calculated in the menu **Dynamic** by the function "calculation of Coriolis and gravity forces".

Remark:

1- For numerical reasons the calculation of $\ddot{\mathbf{q}}$ is calculated by the solution of the system of equations: $\mathbf{A}\, \ddot{\mathbf{q}} = [\mathbf{\Gamma} - \mathbf{H}]$ using Cholesky method, and not by inverting **A** as given in (3.95).

2- Another function that provides $\ddot{\mathbf{q}}$, for open loop or tree structure robots, without inverting **A** is also available in the menu **Dynamic**, function "direct dynamic model". This program is developed using a method based on the work of [Armstrong 79, Featherstone 83, Brandl 86], this method is presented in Appendix 4.

Calculation of the inertia matrix:

The matrix **A** can be calculated by the algorithm of Newton-Euler, by noting from the relation (3.5) , [Walker 82], that the ith column is equal to **Γ** if:

$$\dot{\mathbf{q}} = \mathbf{0},\ \ \mathbf{g} = \mathbf{0},\ \ \ddot{\mathbf{q}} = \mathbf{e}_i,\ \ \mathbf{f}_{tn} = \mathbf{0},\ \ \mathbf{n}_{tn} = \mathbf{0} \tag{3.96}$$

$\mathbf{e}_i$ is the unit (n x 1) vector, whose elements are zero except the ith element which is equal to 1. To increase the efficiency of this method, we can take into account that the links from j to n can be considered as a single composite link which is called the generalized link j. The inertial parameters of the generalized links can be obtained by the following recursive algorithm [Khalil 90b] (for j = n, …,2 and a(j) ≠ 0), where a(j)=j-1, in the case of open loop robots:

$$^{a(j)}\mathbf{J}^{+}_{a(j)} = {}^{a(j)}\mathbf{J}^{+}_{a(j)} + {}^{a(j)}\mathbf{A}_j\, {}^{j}\mathbf{J}^{+}_{j}\, {}^{j}\mathbf{A}_{a(j)} - [{}^{a(j)}\hat{\mathbf{P}}_j\, {}^{a(j)}\hat{\mathbf{MS}}^{+}_j + ({}^{a(j)}\hat{\mathbf{P}}_j\, {}^{a(j)}\hat{\mathbf{MS}}^{+}_j)^T] + {}^{a(j)}\hat{\mathbf{P}}_j\, {}^{a(j)}\hat{\mathbf{P}}^{T}_j\, M^{+}_j \tag{3.97}$$

$$^{a(j)}\mathbf{MS}^{+}_{a(j)} = {}^{a(j)}\mathbf{MS}^{+}_{a(j)} + {}^{a(j)}\mathbf{A}_j\, {}^{j}\mathbf{MS}^{+}_j + {}^{a(j)}\mathbf{P}_j\, M^{+}_j \tag{3.98}$$

$$M^{+}_{a(j)} = M^{+}_{a(j)} + M^{+}_j \tag{3.99}$$

with:

$$^{a(j)}\mathbf{MS}^{+}_j = {}^{a(j)}\mathbf{A}_j\, {}^{j}\mathbf{MS}^{+}_j \tag{3.100}$$

where:

M^{+}_j , ${}^{j}\mathbf{J}^{+}_j$, ${}^{j}\mathbf{MS}^{+}_j$ are initialized by M_j , ${}^{j}\mathbf{J}_j$, ${}^{j}\mathbf{MS}_j$

Calculation of the j th column of the inertia matrix:

Using the values (3.96) in the Newton-Euler algorithm the recursive forward calculation will be reduced to:

$$\begin{aligned} {}^{j}\mathbf{F}_j &= \sigma_j\,[\,0 \quad 0 \quad M_j^{+}\,]^T + \bar{\sigma}_j\,[\,-MY_j^{+} \quad MX_j^{+} \quad 0\,]^T \\ {}^{j}\mathbf{No}_j &= \bar{\sigma}_j\,[\,XZ_j^{+} \quad YZ_j^{+} \quad ZZ_j^{+}\,]^T \end{aligned} \tag{3.101}$$

The recursive backward calculation will be reduced to:

$$\begin{aligned} {}^{j}\mathbf{f}_j &= {}^{j}\mathbf{F}_j \\ {}^{j}\mathbf{n}_j &= \sigma_j\,[\,MY_j^{+} \quad -MX_j^{+} \quad 0\,]^T + \bar{\sigma}_j\,[\,XZ_j^{+} \quad YZ_j^{+} \quad ZZ_j^{+}\,]^T \\ A_{j,j} &= \sigma_j\, M_j^{+} + \bar{\sigma}_j\, ZZ_j^{+} + Ia_j \end{aligned} \tag{3.102}$$

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

To calculate the other elements of the column j, ($A_{a(j),j}$, ... , $A_{sj(0),j}$), where sj(0) indicates the link succeeding link 0 on the path between link 0 and link j, we use the following relations for k=j, a(j), a(a(j)),..., sj(0):

$$
\begin{aligned}
{}^{a(k)}\mathbf{f}_{a(k)} &= {}^{a(k)}\mathbf{A}_k \, {}^{k}\mathbf{f}_k \\
{}^{a(k)}\mathbf{n}_{a(k)} &= {}^{a(k)}\mathbf{A}_k \, {}^{k}\mathbf{n}_k + {}^{a(k)}P_k \times {}^{a(k)}\mathbf{f}_{a(k)} \\
A_{a(k),j} &= [\sigma_j \, {}^{a(k)}\mathbf{f}_{a(k)} + \bar{\sigma}_j \, {}^{a(k)}\mathbf{n}_{(ak)}]^T \, {}^{a(k)}\mathbf{a}_{a(k)}
\end{aligned}
\tag{3.103}
$$

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

This algorithm is programmed using customized method and can be computed using the base inertial parameters.

Calculation of H:

The calculation of the vector ($\mathbf{H} = \mathbf{B}\,\dot{\mathbf{q}}\dot{\mathbf{q}} + \mathbf{C}\,\dot{\mathbf{q}}^2 + \mathbf{Q} + \mathbf{J}^T\,\mathbf{F}_{tn}$) can be obtained with the Newton-Euler method, by noting that $\mathbf{H} = \mathbf{\Gamma}$ if:

$$\ddot{\mathbf{q}} = \mathbf{0} \tag{3.104}$$

Remark: from equation (3.95), the state space equations of a robot can be written as:

$$\frac{d}{dt}\begin{bmatrix}\mathbf{q}\\ \dot{\mathbf{q}}\end{bmatrix} = \begin{bmatrix}\dot{\mathbf{q}}\\ -\mathbf{A}^{-1}\,\mathbf{H}\end{bmatrix} + \begin{bmatrix}\mathbf{0}\\ \mathbf{A}^{-1}\end{bmatrix}\mathbf{\Gamma} \tag{3.105}$$

and:

$$\mathbf{y} = \mathbf{q} \quad \text{or} \quad \mathbf{y} = \mathbf{X} \tag{3.106}$$

In this model the state variables are given by the vector $[\mathbf{q}^T \;\; \dot{\mathbf{q}}^T]^T$, while the output is taken as $\mathbf{y} = \mathbf{q}$ if the output is in the joint space, while $\mathbf{y} = \mathbf{X}$ gives the output in the operational space.

3.6.2. Calculation of motor powers

The dynamic model can be used to determine the motor power of the joints. Given a trajectory specifying the family of tasks to be executed by the robot, we use the inverse dynamic model, to calculate the maximum motor torques. The calculation is carried out beginning by joint n and going back to joint 1. We thus take into account that the torque of motor j is a function of the inertial parameters of link j+1,…,n. After determining a motor j, the inertial parameters of link j-1 must be updated.

A more practical calculation can be carried out by taking into account the thermal model of the motor, to verify if the motor can be driven on the impulse torque [Chedmail 86a, Potkonjak 86].

3.6.3. Identification of the inertial parameters

In order to use the dynamic model in the control of robots, the numerical values of the inertial parameters must be known. Many techniques can be used to obtain these values:

a) The measurement [Armstrong 86]: the main difficulty is the need to take into pieces the different links of the robot.

b) The calculation: by considering the geometry and the material of each link we can calculate the inertial parameters. The use of CAD systems will facilitate this procedure.

c) The identification: we can make use of the identification techniques to determine the values of the inertial parameters [Ferreira 84, Mayeda 84, An 85, Atkeson 85, Khosla 85, Gautier 86, Aldon 86, Olsen 86, Bouzouia 89, Kawasaki 88, Khalil 02a].

3.6.3.1. Dynamic identification model

The solution is based on the use of an identification model linear in the inertial parameters of the dynamic model as given in (3.23). Taking into account the minimum inertial parameters, this model can be written as:

$$\Gamma = \mathbf{D_m}\, \boldsymbol{\Theta}_\mathbf{m} \tag{3.107}$$

where $\boldsymbol{\Theta}_\mathbf{m}$ is the minimum inertial parameters vector.

This model permits to use the least-squares solution to identify the inertial parameters. For this application the grouping formulas of the inertial parameters are not necessary to be known. But it is essential to know the parameters to be eliminated because they have no effect on the dynamic model or because they can be grouped, the grouped values will be directly identified.

The calculation of the i^{th} column of $\mathbf{D_m}$ corresponding to the parameter K_i, denoted $\mathbf{D}^i$, can be calculated by the algorithm of Newton-Euler. In fact, it is easy to see that if we set in the dynamic model the parameter $K_i = 1$ while all the other parameters are zero, then:

$$\Gamma = \mathbf{D}^i$$

The function “Dynamic identification model” in the menu **Identification** gives the columns $\mathbf{D}^i$ in customized form. The element j of $\mathbf{D}^i$, corresponding to the parameter K_i, is denoted DGjK$_i$. In this algorithm, the symbolic iterative calculation is used and we take into account that the forward recursive calculation of the velocities and accelerations is common to all of the columns $\mathbf{D}^i$.

Remarks:

a) The parameters of the links without the payload can be identified only once off-line. The parameters depending on the load must be identified after each changing of the load.

b) Owing to the big number of parameters to be identified. The parameters of each link can be identified separately and recursively from link n to link 1, using the fact that the dynamic model can be put on the following triangular form [Khosla 86, Aubin 91]:

$$\begin{bmatrix} \Gamma_1 \\ \Gamma_2 \\ \vdots \\ \Gamma_{n-1} \\ \Gamma_n \end{bmatrix} = \begin{bmatrix} \Phi_{11} & \Phi_{12} & \cdots & \Phi_{1n} \\ \mathbf{0} & \Phi_{22} & \cdots & \Phi_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{0} & \ldots & \Phi_{n-1,n-1} & \Phi_{n-1,n} \\ \mathbf{0} & \ldots & \mathbf{0} & \Phi_{nn} \end{bmatrix} \begin{bmatrix} \Theta^1 \\ \Theta^2 \\ \vdots \\ \Theta^{n-1} \\ \Theta^n \end{bmatrix} \tag{3.108}$$

where Θ^j is the vector containing the minimum inertial parameters of link j.

c) The choice of the robot trajectory during the identification is very important in order to excite the different parameters [Armstrong 87, Gautier 92, Khalil 02a].

3.6.3.2. Energy identification model

Since the matrix $\mathbf{D_m}$ is a function of the joint acceleration $\ddot{q}_i$, which will be noisy, the identification result would be biased. By simulation and taking into account the characteristics of the sensors one can verify if the bias can be accepted. A solution for this problem is to use a model based on the difference of the energy that is a function of $\mathbf{q}$ and $\dot{\mathbf{q}}$ only [Gautier 88b]. This model, without taking into account the friction to simplify the writing[1], is:

$$\int_{t_1}^{t_2} \mathbf{\Gamma}^T \, \dot{\mathbf{q}} \, dt = H(t_2) - H(t_1) \tag{3.109}$$

with $H(t_i)$ is the total energy at time t_i, with:

$$H(t_i) = E(t_i) + U(t_i) \tag{3.110}$$

As H is linear in the inertial parameters, we can write:

$$H(t_2) - H(t_1) = [\mathbf{h}(t_2) - \mathbf{h}(t_1)] \, \mathbf{K_m} \tag{3.111}$$

with $\mathbf{K_m}$ is the minimum inertial parameters vector.

[1] In general the friction coefficients are identified separately [Held 88, Dahl 77].

Equation (3.109) can be rewritten:

$$\int_{t_1}^{t_2} \mathbf{\Gamma}^T \dot{\mathbf{q}} = \Delta \mathbf{h} \; \mathbf{K_m} \tag{3.112}$$

The derivative of relation (3.112) gives what is known by the power model [Gautier 97 or Gautier 96], which is written as:

$$\mathbf{\Gamma}^T \dot{\mathbf{q}} = d/dt(\Delta \mathbf{h}) \, \mathbf{K_m} \tag{3.113}$$

Equations (3.112) and (3.113) represent a linear system in the inertial parameters. To identify $\mathbf{K_m}$, a sufficient number of equations can be obtained by calculating (3.112) between different intervals of time or calculating (3.113) at sufficient number of configurations. The identification can be obtained using the least-squares technique.

The menu **Identification** of **SYMORO+** gives the expressions of the coefficients h_i, in customized form.

3.6.3.3. Filtered dynamic identification model

Another identification model that is not a function of the joint acceleration is known as the filtered dynamic model. Previous work proposed to get the filtered dynamic model from the dynamic model [Aubin 91, Middleton 86, Slotine 87], this procedure is very complicated and contains many redundant calculations. We propose to make use of the Lagrangian equation, this algorithm is detailed in Appendix 5.

Since Lagrange Equation is given as:

$$\mathbf{\Gamma} = \frac{d}{dt}\frac{\partial L}{\partial \dot{\mathbf{q}}} - \frac{\partial L}{\partial \mathbf{q}} \tag{3.114}$$

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.

Thus:

$$\mathbf{\Gamma} = \frac{d}{dt}\left(\frac{\partial E}{\partial \dot{\mathbf{q}}}\right) - \frac{\partial E}{\partial \mathbf{q}} + \frac{\partial U}{\partial \mathbf{q}} \tag{3.115}$$

We obtain:

$$\Gamma = [\frac{d}{dt}\mathbf{D_1}(\mathbf{q},\dot{\mathbf{q}}) + \mathbf{D_2}(\mathbf{q},\dot{\mathbf{q}})]\,\mathbf{K} \qquad (3.116)$$

where, **K** is the inertial parameters vector of all the links, $\mathbf{D_1}$ and $\mathbf{D_2}$ are of dimension nx10n.

Applying a numerical filter, denoted F, on equation (3.116) gives:

$$\Gamma_f = \mathbf{D_{1df}}(\mathbf{q},\dot{\mathbf{q}})\,\mathbf{K} + \mathbf{D_{2f}}(\mathbf{q},\dot{\mathbf{q}})\,\mathbf{K} \qquad (3.117)$$

where the subscribt “f” means the application of the original 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:

$$F(s) = \frac{a^2}{(s+a)^2}\ ,\quad F'(s) = \frac{a^2\,s}{(s+a)^2} \qquad (3.118)$$

Equation (3.116) can be used to identify the dynamic parameters, it is independent of the joint accelerations. It is what we mean by the filtered dynamic model. Its main advantage is that the filtering of both terms of the model gives what is called parallel filtering of the identification model. This process is very important to the success of the identification. This parallel filtering must also be done on the identification dynamic model to get good results. For practical identification considerations consult the references [Gautier 95a, Gautier95b, Khalil 02a].

The menu **Identification** of **SYMORO+** gives the expressions of the coefficients **D1** and **D2** in customized form.

3.6.4. Control

The most important application of the dynamic model is its use to realize more sophisticated control law of robots in order to increase its performance in speed and accuracy. We present in this section the use of the dynamic model in a decoupling non-linear control strategy.

3.6.4.1. Introduction

The use of the decoupled non-linear control has been considered as a good approach to control the robots [Bejczy 74, Zabala 78, Raibert 78, Khalil 78, Khalil 79, Khatib 80, Luh 80a, Freund 82, Khosla 86, Khalil 87b]. This control is also known as *dynamic control* or as *computed torque control*. Theoretically, it ensures the decoupling and linearization of the equations of motion of the robot. When this type of control has been proposed, two problems have delayed its application on real robots:

a) the complexity of the computation of the dynamic model on line,
b) the numerical values of the inertial and friction parameters are not given by the robot manufacturers.

These difficulties can be considered as being solved. In fact the calculation of the dynamic model on line is now possible thanks to efficient modeling methods and to the progress in the technology of computers, also identification techniques (section 3.6.3) permit to determine the values of inertial and friction parameters.

3.6.4.2. Decoupling nonlinear control in the joint space

3.6.4.2.1. Principle of the control:

The dynamic model can be written as:

$$\boldsymbol{\Gamma} = \mathbf{A}(\mathbf{q})\,\ddot{\mathbf{q}} + \mathbf{H}(\mathbf{q},\dot{\mathbf{q}}) \tag{3.119}$$

where:

$$\mathbf{H} = \mathbf{B}\,\dot{\mathbf{q}}\,\dot{\mathbf{q}} + \mathbf{C}\,\dot{\mathbf{q}}^2 + \mathbf{Q} \tag{3.120}$$

Supposing that the joint positions and velocities are measurable without noise. Then if we take the control input as [Khalil 79]:

$$\boldsymbol{\Gamma} = \hat{\mathbf{A}}(\mathbf{q})\mathbf{W}(t) + \hat{\mathbf{H}}(\mathbf{q},\dot{\mathbf{q}}) \tag{3.121}$$

where $\hat{\mathbf{A}}(\mathbf{q})$ and $\hat{\mathbf{H}}(\mathbf{q},\dot{\mathbf{q}})$ denote the estimation of **A** and **H** respectively.

Using (3.119) and supposing that the model is exact, the system will be represented by:

$$\ddot{\mathbf{q}} = \mathbf{W}(t) \tag{3.122}$$

W(t) can be considered as a new control vector. We are now dealing with a control problem of n second order linear, decoupled systems. Many possibilities can be chosen to define **W**(t) [Chevallereau 88b, Khalil 002a], for example in the case of a completely specified trajectory, where $\ddot{\mathbf{q}}^{\mathbf{d}}(t)$, $\dot{\mathbf{q}}^{\mathbf{d}}(t)$ and $\mathbf{q}^{\mathbf{d}}(t)$ define the desired acceleration, velocity, and position in the joint space. **W**(t) can be calculated as:

$$\mathbf{W}(t) = \ddot{\mathbf{q}}^{\mathbf{d}} + \mathbf{K_d}\,(\dot{\mathbf{q}}^{\mathbf{d}} - \dot{\mathbf{q}}) + \mathbf{K_p}\,(\mathbf{q}^{\mathbf{d}} - \mathbf{q}) \tag{3.123}$$

using (3.122), the closed loop system will be given as:

$$\ddot{\mathbf{e}} + \mathbf{K_d}\,\dot{\mathbf{e}} + \mathbf{K_p}\,\mathbf{e} = \mathbf{0} \tag{3.124}$$

Taking $\mathbf{K_d}$ and $\mathbf{K_p}$ positive the error **e** will be asymptotically zero. The block diagram of this control law is given in figure 3.3:

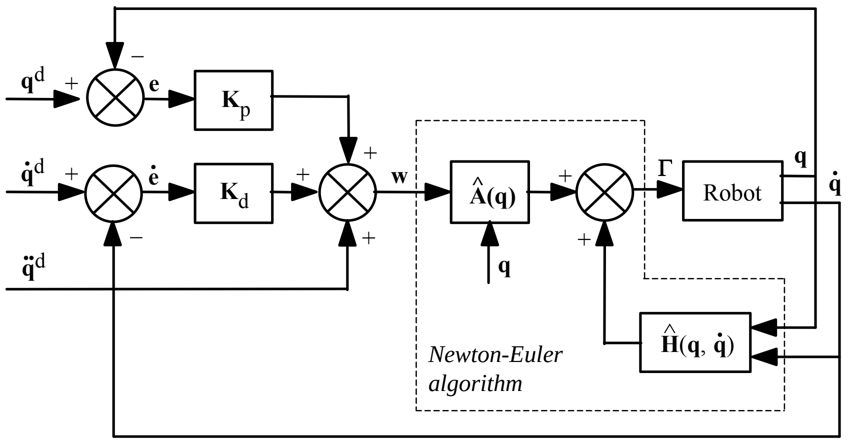


Figure 3.3. Computed torque of the tracking control scheme in the joint space

3.6.4.2.2 Predictive control law [Khalil 78]

In this case the estimates of $\hat{\mathbf{A}}$ and $\hat{\mathbf{H}}$ are no longer computed with the current values of $\mathbf{q}$ and $\dot{\mathbf{q}}$, but rather with the desired values $\mathbf{q}^d$ and $\dot{\mathbf{q}}^d$. Thus, the control law is written as:

$$\Gamma = \hat{\mathbf{A}}(\mathbf{q}^d)\,\mathbf{W}(t) + \hat{\mathbf{H}}(\mathbf{q}^d,\dot{\mathbf{q}}^d) \tag{3.125}$$

where $\mathbf{W}(t)$ is defined by (3.123).

3.6.4.2.3. Practical calculation of the control law

The foregoing control laws can be calculated using the algorithm of Newton-Euler (menu **Dynamic**), without calculating separately the coefficients **A** and **H**. In fact, from the relations (3.119) and (3.120), we obtain that:

- the calculation of the control law of figure 3.3 can be obtained by the algorithm of Newton-Euler by taking as input arguments:
  * joint positions, equal to the current joint positions $\mathbf{q}$,
  * joint velocities, equal to the current joint velocities $\dot{\mathbf{q}}$,
  * joint accelerations, equal to $\mathbf{W}(t)$.
- the calculation of the predective control law can be obtained by the algorithm of Newton-Euler by taking as input arguments:
  * joint positions, equal to the desired joint positions $\mathbf{q}^{\mathbf{d}}$,
  * joint velocities, equal to the desired joint velocities $\dot{\mathbf{q}}^{\mathbf{d}}$,
  * joint accelerations, equal to $\mathbf{W}(t)$.

The computation cost of the decoupling nonlinear control in the joint space is thus equivalent to the calculation of the inverse dynamic model.

To study the robustness of this control law, the reader can consult the references [Samson 87a, Samson 87b, Craig 86, Khalil 02a, Samson 91].

3.6.4.3. Decoupling nonlinear control in the operational space

3.6.4.3.1. Introduction

If the desired trajectory is given in the operational space, one of the following strategies can be used:

a) - transform the coordinates of the desired trajectory from operational space into joint space, then the control in the joint space can be used.
b) - consider directly the dynamic equations in the operational space.

In the following we develop the second approach.

By differentiating the kinematic model, $\dot{\mathbf{X}} = \mathbf{J}\,\dot{\mathbf{q}}$, we obtain:

$$\ddot{\mathbf{q}} = \mathbf{J}^{-1}(\ddot{\mathbf{X}} - \dot{\mathbf{J}}\,\dot{\mathbf{q}}) \tag{3.126}$$

where:

$$\dot{\mathbf{J}}\,(\mathbf{q},\dot{\mathbf{q}}) = \frac{d}{dt}\mathbf{J}(\mathbf{q})$$

Substituting $\ddot{\mathbf{q}}$ from relation (3.126) into (3.119) we obtain:

$$\Gamma = \mathbf{A}\,\mathbf{J}^{-1}[\ddot{\mathbf{X}} - \dot{\mathbf{J}}\,\dot{\mathbf{q}}] + \mathbf{H} \tag{3.127}$$

Similarly to the joint space control, the decoupling control law can be obtained as:

$$\Gamma = \mathbf{A}\,\mathbf{J}^{-1}(\mathbf{W} - \dot{\mathbf{J}}\,\dot{\mathbf{q}}) + \mathbf{H} \tag{3.128}$$

As in the case of joint space control many possibilities can be used for this control law [Chevallereau 88b]. If the desired trajectory is completely specified, then we take:

$$\mathbf{W}(t) = \ddot{\mathbf{X}}^{\mathbf{d}} + \mathbf{K_d}\,(\dot{\mathbf{X}}^{\mathbf{d}} - \dot{\mathbf{X}}) + \mathbf{K_p}\,(\mathbf{X}^{\mathbf{d}} - \mathbf{X}) \tag{3.129}$$

Using this control law, and supposing that the model is exact and there is no initial error, the robot will be driven by the following equation:

$$\ddot{\mathbf{e}}_{\mathbf{x}} + \mathbf{K_d}\,\dot{\mathbf{e}}_{\mathbf{x}} + \mathbf{K_p}\,\mathbf{e}_{\mathbf{x}} = \mathbf{0} \tag{3.130}$$

where $\mathbf{e_x} = \mathbf{X}^{\mathbf{d}} - \mathbf{X}$

The corresponding block diagram is given in figure 3.4. The input vector Γ can be computed by the Newton-Euler dynamic model using the following arguments:

∗ joint positions, equal to the current joint positions $\mathbf{q}$,

∗ joint velocities, equal to the current joint velocities $\dot{\mathbf{q}}$,

∗ joint accelerations, equal to: $\ddot{\mathbf{q}} = \mathbf{J}^{-1}(\mathbf{W}(t) - \dot{\mathbf{J}}\,\dot{\mathbf{q}})$ (3.131)

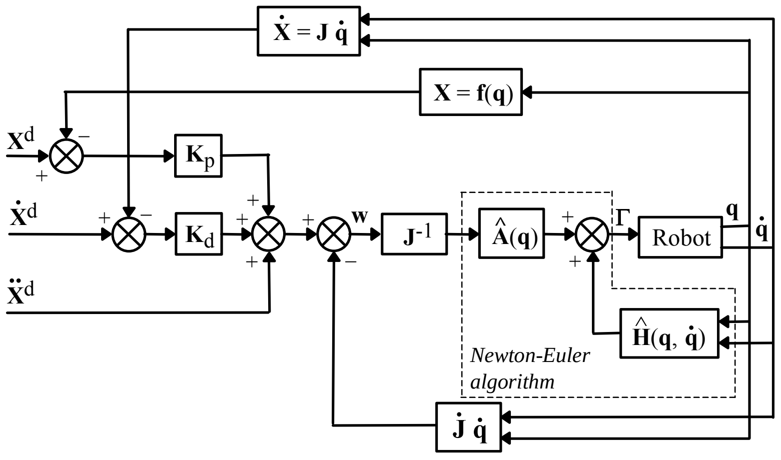


Figure 3.4. Control scheme in the operational space

3.6.4.3.2. Optimization of the computational cost of the operational space control law [Khalil 87b]

We present in this section the calculations needed to realize the control law in the operational space. The proposed algorithm deals with the inverse dynamic and inverse kinematic calculations simultaneously in order to eliminate many redundant calculations.

a) Calculation of the situation error $\mathbf{e_X}$

The current situation of the terminal link is given by the matrix ${}^0\mathbf{T}_n$.

The desired position and orientation are supposed given by the vector ${}^0\mathbf{P}_n{}^d$ and by the director cosines matrix $[{}^0\mathbf{s}_n{}^\mathbf{d}\quad {}^0\mathbf{n}_n{}^\mathbf{d}\quad {}^0\mathbf{a}_n{}^\mathbf{d}]$ respectively. The situation error $\mathbf{e_X}$ represent the position error $\mathbf{dX_p}$ and the orientation error $\mathbf{dX_r}$ such that:

$$\mathbf{e_X} = \begin{bmatrix} \mathbf{dX_p} \\ \mathbf{dX_r} \end{bmatrix}$$

where [Khalil 02a]:

$$\mathbf{dX}_p = {}^0\mathbf{dP}_n = {}^0\mathbf{P}_n{}^{\mathbf{d}} - {}^0\mathbf{P}_n \tag{3.132}$$

$$\mathbf{dX}_r = \frac{1}{2}[{}^0\mathbf{s}_n \times {}^0\mathbf{s}_n{}^d + {}^0\mathbf{n}_n \times {}^0\mathbf{n}_n{}^d + {}^0\mathbf{a}_n \times {}^0\mathbf{a}_n{}^d] \tag{3.133}$$

b) Calculation of the current operational velocity $\dot{\mathbf{X}}$

The current operational velocity is composed of the linear velocity ${}^0\mathbf{V}_n$ and the angular velocity ${}^0\omega_n$ of frame n, thus:

$$\dot{\mathbf{X}} = \begin{bmatrix} {}^0\mathbf{V}_n \\ {}^0\omega_n \end{bmatrix} \tag{3.134}$$

The calculation of ${}^0\mathbf{V}_n$ and ${}^0\omega_n$ can be carried out using the recursive algorithm of equations (2.28) (for j = 1, …, n). The values of $\omega_1, \ldots, \omega_n$ will be also used in the dynamic model calculations.

Remark: If the origin O_n is confounded with the origin O_k (with k < n), then $\mathbf{V}_n = \mathbf{V}_k$. The recursive calculation of the linear velocity can stop at j = k.

c) Calculation of $\dot{\mathbf{J}}\,\dot{\mathbf{q}}$

The calculation of this term by differentiating the Jacobian matrix will need a prohibitive number of operations, we propose to calculate it using a recursive algorithm whose intermediate calculations will be used at the dynamic model calculation. From the direct kinematic model we obtain:

$$\ddot{\mathbf{X}}_n = \begin{bmatrix} \dot{\mathbf{V}}_n \\ \dot{\omega}_n \end{bmatrix} = \mathbf{J(q)}\,\ddot{\mathbf{q}} + \dot{\mathbf{J}}\,(\mathbf{q}, \dot{\mathbf{q}})\,\dot{\mathbf{q}} \tag{3.135}$$

From (3.135), it is seen that $\dot{\mathbf{J}}\,(\mathbf{q}, \dot{\mathbf{q}})\,\dot{\mathbf{q}}$ is equal to $\ddot{\mathbf{X}}_n$ when setting $\ddot{\mathbf{q}} = \mathbf{0}$, which gives:

$${}^j\Psi_j = {}^j\mathbf{A}_{j-1}\,{}^{j-1}\Psi_{j-1} + \bar{\sigma}_j\,({}^j\omega_{j-1} \times \dot{q}_j\,{}^j\mathbf{a}_j) \tag{3.136}$$

$${}^j\mathbf{U}_j^{*} = {}^j\hat{\Psi}_j + {}^j\hat{\omega}_j\,{}^j\hat{\omega}_j \tag{3.137}$$

$${}^j\Phi_j = {}^j\mathbf{A}_{j-1}\,({}^{j-1}\Phi_{j-1} + {}^{j-1}\mathbf{U}_{j-1}^{*}\,{}^{j-1}\mathbf{P}_j) + 2\,\sigma_j\,{}^j\omega_{j-1} \times (\dot{q}_j\,{}^j\mathbf{a}_j) \tag{3.138}$$

In this case, where ($\ddot{\mathbf{q}} = \mathbf{0}$), $\dot{\mathbf{V}}_j$ and $\dot{\omega}_j$ are denoted $\mathbf{\Phi}_j$ and $\mathbf{\Psi}_j$ respectively.

The initial values are taken as ${}^0\mathbf{\Psi}_0 = \mathbf{0}$ and ${}^0\mathbf{\Phi}_0 = \mathbf{0}$.

If O_k is confounded with O_n, then $\mathbf{\Phi}_k = \mathbf{\Phi}_n$, the calculation of $\mathbf{\Phi}_j$ can be stopped at $j = k$.

d) Calculation of $\mathbf{J(q)}^{-1}\mathbf{y}$, where $\mathbf{y}$ is the term between brackets of the relation (3.128)

This problem has been treated in Chapter 2 in the inverse kinematic model.

e) Calculation of the dynamic model

We will modify the Newton-Euler algorithm, to take into account the results obtained while calculating $\mathbf{\Phi}_j$ and $\mathbf{\Psi}_j$. The equations calculating $\dot{\mathbf{V}}_j$ and $\dot{\omega}_j$ will be replaced by:

$$ {}^j\dot{\mathbf{V}}_j = {}^j\mathbf{\Phi}_j + {}^j\mathbf{b}_j \tag{3.139}$$

$$ {}^j\dot{\omega}_j = {}^j\mathbf{\Psi}_j + {}^j\mathbf{e}_j \tag{3.140}$$

where ${}^j\mathbf{b}_j$ and ${}^j\mathbf{e}_j$ represent ${}^j\dot{\mathbf{V}}_j$ and ${}^j\dot{\omega}_j$ respectively when $\dot{\mathbf{q}} = \mathbf{0}$:

$$ {}^j\mathbf{b}_j = {}^j\mathbf{A}_{j-1}({}^{j-1}\mathbf{b}_{j-1} + {}^{j-1}\hat{\mathbf{e}}_{j-1}\, {}^{j-1}\mathbf{P}_j) + \sigma_j\, \ddot{q}_j\, {}^j\mathbf{a}_j \tag{3.141}$$

$$ {}^j\mathbf{e}_j = {}^j\mathbf{A}_{j-1}\, {}^{j-1}\mathbf{e}_{j-1} + \bar{\sigma}_j\, \ddot{q}_j\, {}^j\mathbf{a}_j \tag{3.142}$$

with the initial conditions, ${}^0\mathbf{b}_0 = -\,\mathbf{g}$ and ${}^0\mathbf{e}_0 = \mathbf{0}$.

The matrix ${}^j\mathbf{U}_j$, defined by equation (3.45), is calculated for $j = 1,\ldots,k$ with O_k is confounded with O_n , by:

$$ {}^j\mathbf{U}_j = {}^j\mathbf{U}_j^* + {}^j\hat{\mathbf{e}}_j \tag{3.143}$$

For $j > k$, the matrix ${}^j\mathbf{U}_j$ is obtained using relation (3.45).

3.6.4.3.3. The operational dynamic control in **SYMORO+**

Most of the elements of the operational control law can be obtained using the menu **Kinematic** (${}^j\mathbf{V}_j$, ${}^j\mathbf{\Phi}_j$, ${}^j\omega_j$, ${}^j\mathbf{\Psi}_j$) and **Dynamic**.

3.7. Conclusion

In this chapter, we presented the calculation of the dynamic model of robots using the Lagrange method and the Newton-Euler recursive method. The Newton-Euler method has been optimized to reduce its computation cost. An important section has been devoted for the presentation of the main applications of the dynamic model.

In the next chapter the dynamic model of tree structure and closed loop robots will be presented.

Chapter 4

Dynamic model of tree structure and closed robots

4.1. Introduction

We present in this chapter the dynamic model of tree structure and closed loop robots, the base inertial parameters of these robots will be also presented.

4.2. Dynamic model of tree structure robots

We note that the models and methods presented for simple open loop robots are available in SYMORO+ for tree structure robots. We just present, here the inverse dynamic model by Newton-Euler.

4.2.1. Newton-Euler equations

The forward recursive equations of the Newton-Euler inverse dynamic model (Chapter 3) can be generalized for tree structured robots by replacing j – 1 by i, with i = a(j):

$$^{j}\omega_{i} = {}^{j}\mathbf{A}_{i}\, {}^{i}\omega_{i} \tag{4.1}$$

$$^{j}\omega_{j} = {}^{j}\omega_{i} + \bar{\sigma}_{j}\, \dot{q}_{j}\, {}^{j}\mathbf{a}_{j} \tag{4.2}$$

$$^{j}\dot{\omega}_{j} = {}^{j}\mathbf{A}_{i}\, {}^{i}\dot{\omega}_{i} + \bar{\sigma}_{j}\, (\ddot{q}_{j}\, {}^{j}\mathbf{a}_{j} + {}^{j}\omega_{i} \text{ x } \dot{q}_{j}\, {}^{j}\mathbf{a}_{j}) \tag{4.3}$$

$$^{j}\dot{\mathbf{V}}_{j} = {}^{j}\mathbf{A}_{i}\, ({}^{i}\dot{\mathbf{V}}_{i} + {}^{i}\mathbf{U}_{i}\, {}^{i}\mathbf{P}_{j}) + \sigma_{j}\, (\ddot{q}_{j}\, {}^{j}\mathbf{a}_{j} + 2\, {}^{j}\omega_{i} \text{ x } \dot{q}_{j}\, {}^{j}\mathbf{a}_{j}) \tag{4.4}$$

$$^{j}\mathbf{F}_{j} = M_{j}\, {}^{j}\dot{\mathbf{V}}_{j} + {}^{j}\mathbf{U}_{j}\, {}^{j}\mathbf{MS}_{j} \tag{4.5}$$

$$^{j}\mathbf{No}_{j} = {}^{j}\mathbf{J}_{j}\, {}^{j}\dot{\omega}_{j} + {}^{j}\omega_{j} \text{ x } ({}^{j}\mathbf{J}_{j}\ {}^{j}\omega_{j}) \tag{4.6}$$

The backward recursive relations must take into account that a link j may have many links articulated on it (figure 4.1). Let k define a link such that a(k) = j.

Thus the backward recursive relations can be written as [Khalil 02a], for j = n,…,1:

$$^{j}\mathbf{f}_{j} = {}^{j}\mathbf{F}_{j} + \sum_{k/a(k)=j} {}^{j}\mathbf{f}_{k} \tag{4.7}$$

$$^{i}\mathbf{f}_{j} = {}^{i}\mathbf{A}_{j}\ {}^{j}\mathbf{f}_{j} \tag{4.8}$$

$$ {}^{j}\mathbf{n}_j = {}^{j}\mathbf{No}_j + \sum_{k/a(k)=j} ({}^{j}\mathbf{A}_k\, {}^{k}\mathbf{n}_k + {}^{j}\mathbf{P}_k \times {}^{j}\mathbf{f}_k) + {}^{j}\mathbf{MS}_j \times {}^{j}\dot{\mathbf{V}}_j \tag{4.9} $$

$$ \Gamma_j = (\sigma_j\, {}^{j}\mathbf{f}_j + \bar{\sigma}_j\, {}^{j}\mathbf{n}_j)^T\, {}^{j}\mathbf{a}_j + F_{sj}\, \mathrm{sign}(\dot{q}_j) + F_{vj}\, \dot{q}_j + Ia_j\, \ddot{q}_j \tag{4.10} $$

If link j is a terminal link the elements ${}^{j}\mathbf{n}_k$ and ${}^{j}\mathbf{f}_k$ will be replaced by ${}^{j}\mathbf{n}_{tj}$ and ${}^{j}\mathbf{f}_{tj}$, representing the moments and forces exerted by link j on the environment, while ${}^{j}\mathbf{P}_k$ will be zero.

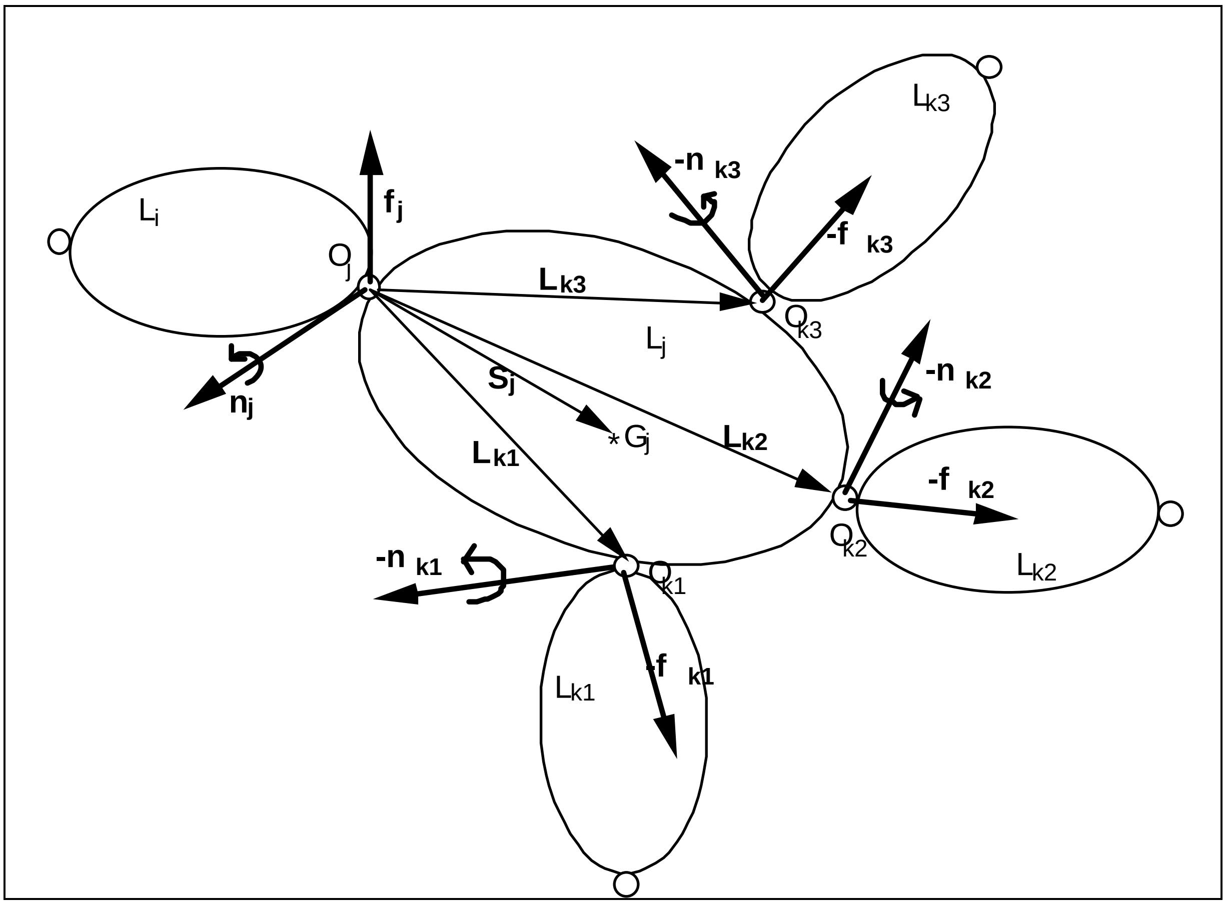


Figure 4.1. Efforts on a tree structure robot.

4.2.2. Intermediate variables used in the iterative symbolic model

The dynamic model using the Newton-Euler method of simple open loop robots, given in Chapter 3, is a special case of the tree structure robots, so only the algorithm of tree structure robots is implemented in **SYMORO+**. The variables used in this program are the same as those presented in Chapter 3. Thus we only present the intermediate variables corresponding to the transformation matrices.

** Intermediate variables corresponding to the transformation matrices*

From equation (1.3) we obtain:

$$\begin{array}{lll} A11j = Cj\, C\gamma_j - Sj\, C\alpha_j\, S\gamma_j, & A21j = Cj\, S\gamma_j + Sj\, C\alpha_j\, C\gamma_j, & A31j = Sj\, S\alpha_j \\ A12j = -Sj\, C\gamma_j - Cj\, C\alpha_j\, S\gamma_j, & A22j = -\, Sj\, S\gamma_j + Cj\, C\alpha_j\, C\gamma_j, & A32j = Cj\, S\alpha_j \\ A13j = S\gamma_j\, S\alpha_j, & A23j = -\, C\gamma_j\, S\alpha_j & \end{array}$$

Thus:

$$^{i}\mathbf{A}_j = \begin{bmatrix} A11j & A12j & A13j \\ A21j & A22j & A23j \\ A31j & A32j & C\alpha_j \end{bmatrix}$$

with Cj and Sj denote $\cos(\theta_j)$ and $\sin(\theta_j)$ respectively.

If joint j is prismatic, all the elements of this matrix are constant and can be calculated off-line, whereas only the elements A13j and A23j are constant if joint j is revolute.

** Intermediate variables corresponding to $^{i}\mathbf{P}_j$*

From equation (1.3), we obtain:

$$\begin{aligned} LOO1j &= d_j\, C\gamma_j + r_j\, S\gamma_j\, S\alpha_j = d_j\, C\gamma_j + r_j\, A13j \\ LOO2j &= d_j\, S\gamma_j - r_j\, C\gamma_j\, S\alpha_j = d_j\, S\gamma_j + r_j\, A23j \\ LOO3j &= r_j\, C\alpha_j + b_j \end{aligned}$$

and:

$$^{i}\mathbf{P}_j = [LOO1j \quad LOO2j \quad LOO3j]^T$$

If joint j is revolute, the elements LOOij will be constant and can be calculated off-line.

4.2.3. Calculation of the minimum set of inertial parameters

The method presented in section 3.5.2 can be extended to tree structure robots [Khalil 89c]. The relations (3.81) and (3.87) are valid, but the elements of $^{i}\lambda_j$ are calculated in terms of the 6 geometric parameters b_j, γ_j, α_j, d_j, θ_j, r_j . Thus:

The columns $^{i}\lambda_j^{10}$, $^{i}\lambda_j^{9}$, $^{i}\lambda_j^{1+4}$ are constant, for j revolute, and are given as [Bennis 91b]:

$$^{i}\lambda_j^{10} = \left[\, P_y{}^2+P_z{}^2 \quad -P_xP_y \quad -P_xP_z \quad P_x{}^2+P_z{}^2 \quad -P_yP_z \quad P_x{}^2+P_y{}^2 \quad P_x \quad P_y \quad P_z \quad 1 \,\right]^T \tag{4.11}$$

$$ {}^{i}\lambda_j^9 = [\; 2P_zC\alpha - 2P_yC\gamma S\alpha \quad P_xC\gamma S\alpha - P_yS\gamma S\alpha \quad -P_xC\alpha - P_zS\gamma S\alpha \quad 2P_xS\gamma S\alpha + 2P_zC\alpha $$
$$ -P_yC\alpha + P_zC\gamma S\alpha \quad 2P_xS\gamma S\alpha - 2P_yC\gamma S\alpha \quad S\gamma S\alpha \quad -C\gamma S\alpha \quad C\alpha \quad 0 \;]^T \tag{4.12} $$

$$ {}^{i}\lambda_j^{1+4} = [\; 1-SS\gamma SS\alpha \quad SC\gamma SS\alpha \quad -S\gamma SC\alpha \quad 1-CC\gamma SS\alpha \quad C\gamma CS\alpha \quad SS\alpha \quad 0 \quad 0 \quad 0 \quad 0 \;]^T \tag{4.13} $$

where:

$SS(.) = \text{Sin}^2(.)$, $CC(.) = \text{Cos}^2(.)$, $CS(.) = \text{Cos}(.)\text{Sin}(.)$

P_x, P_y, P_z are the components of ${}^{i}\mathbf{P}_j$, they are calculated from equation (1.3). To simplify the writing the subscript j is not written for the parameters: b_j, γ_j, d_j, α_j, r_j, θ_j in the previous equations.

The grouping relations of section 3.5.2.2 can be generalized by the following theorem [Khalil 89c].

Theorem 1: If joint j is revolute, the parameters M_j, MZ_j, YY_j can be grouped with the parameters of links j and i, with i = a(j). The grouping relations are given as:

$$
\begin{aligned}
XXR_j &= XX_j - YY_j \\
XXR_i &= XX_i + YY_j\,(1-SS\gamma_jSS\alpha_j) + M_j\,(P_y^2+P_z^2) + 2\,MZ_j\,(P_zC\alpha_j - P_yC\gamma_jS\alpha_j) \\
XYR_i &= XY_i + YY_j\,(CS\gamma_jSS\alpha_j) + M_j\,(-P_xP_y) + MZ_j\,(P_xC\gamma_jS\alpha_j - P_yS\gamma_jS\alpha_j) \\
XZR_i &= XZ_i - YY_j\,(S\gamma_jCS\alpha_j) + M_j\,(-P_xP_z) + MZ_j\,(-P_xC\alpha_j - P_zS\gamma_jS\alpha_j) \\
YYR_i &= YY_i + YY_j\,(1-CC\gamma_jSS\alpha_j) + M_j\,(P_x^2+P_z^2) + 2\,MZ_j\,(P_xS\gamma_jS\alpha_j + P_zC\alpha_j) \\
YZR_i &= YZ_i + YY_j\,(C\gamma_jCS\alpha_j) + M_j\,(-P_yP_z) + MZ_j\,(-P_yC\alpha_j + P_zC\gamma_jS\alpha_j) \\
ZZR_i &= ZZ_i + YY_j\,SS\alpha_i + M_j\,(P_x^2+P_y^2) + 2\,MZ_j\,(P_xS\gamma_jS\alpha_j - P_yC\gamma_jS\alpha_j) \\
MR_i &= M_i + M_j \\
MXR_i &= MX_i + M_j\,P_x + MZ_j\,(S\gamma_jS\alpha_j) \\
MYR_i &= MY_i + M_j\,P_y - MZ_j\,(C\gamma_jS\alpha_j) \\
MZR_i &= MZ_i + M_j\,P_z + MZ_j\,C\alpha_j
\end{aligned} \tag{4.14}
$$

with P_x, P_y and P_z are the coordinates of ${}^{i}\mathbf{P}_j$, obtained from equation (1.3) as:

$$ {}^{i}\mathbf{P}_j = \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix} = \begin{bmatrix} d_j\,C\gamma_j + r_j\,S\gamma_j\,S\alpha_j \\ d_j\,S\gamma_j - r_j\,C\gamma_j\,S\alpha_j \\ r_j\,C\alpha_j + b_j \end{bmatrix} \tag{4.15} $$

Theorem 2: If joint j is prismatic, the elements of the inertia tensor ${}^{j}\mathbf{J}_j$ can be grouped with the parameters of the inertia tensor ${}^{i}\mathbf{J}_i$ using the following relation:

$$ {}^{i}\mathbf{JR}_i = {}^{i}\mathbf{J}_i + {}^{i}\mathbf{A}_j\,{}^{j}\mathbf{J}_j\,{}^{j}\mathbf{A}_i \tag{4.16} $$

Theorem 3: Particular grouping relations:

Two cases are considered [Khalil 94c]:

a- if the axis of the prismatic joint j, for r1 < j < r2, is not parallel to the r1 axis:

The following relation between the coefficients h_{MXj}, h_{MYj} and h_{MZj} of link j is verified:

$$ {}^{j}a_{xr1}\ h_{MXj} + {}^{j}a_{yr1}\ h_{MYj} + {}^{j}a_{zr1}\ h_{MZj} = \text{constant} \tag{4.17}$$

where ${}^{j}\mathbf{a}_{r1}$ is the unit vector along the r1 axis referred to frame j:

$$ {}^{j}\mathbf{a}_{r1} = [{}^{j}a_{xr1} \quad {}^{j}a_{yr1} \quad {}^{j}a_{zr1}]^T $$

By the use of relations (3.70), we obtain the grouping relations or elimination given in table 4.1 . It can be seen that, in this case, one element of the first moments $\mathbf{MS}_j$ will be eliminated either by grouping or because it has no effect.

${}^{j}a_{zr1} \neq 0$	${}^{j}a_{zr1} = 0,\ {}^{j}a_{xr1}\,{}^{j}a_{yr1} \neq 0$	${}^{j}a_{zr1} = 0,\ {}^{j}a_{xr1} = 0$	${}^{j}a_{zr1} = 0,\ {}^{j}a_{yr1} = 0$
$MXR_j = MX_j - \frac{{}^{j}a_{xr1}}{{}^{j}a_{zr1}} MZ_j$ $MYR_j = MY_j - \frac{{}^{j}a_{yr1}}{{}^{j}a_{zr1}} MZ_j$	$MXR_j = MX_j - \frac{{}^{j}a_{xr1}}{{}^{j}a_{yr1}} MY_j$	$MY_j = 0$	$MX_j = 0$

Table 4.1

b) if the axis of the prismatic joint j is parallel to the axis of r1 for (r1 < j < r2):

It can be seen from (3.73) that MZ_j has no effect on the dynamic model, while the parameters MX_j and MY_j can be grouped with the parameters of the first moments of link i, with i = a(j), and with the parameter ZZ_s, with s is the nearest revolute joint for j from the base side. The grouped parameters are given by:

$$\begin{aligned}
MXR_i &= MX_i + (C\gamma_j C\theta_j - S\gamma_j C\alpha_j S\theta_j)\, MX_j - (C\gamma_j S\theta_j + S\gamma_j C\alpha_j C\theta_j)\, MY_j \\
MYR_i &= MY_i + (S\gamma_j C\theta_j + C\gamma_j C\alpha_j S\theta_j)\, MX_j + (-S\gamma_j S\theta_j + C\gamma_j C\alpha_j C\theta_j)\, MY_j \\
MZR_i &= MZ_i + S\theta_j\, S\alpha_j\, MX_j + C\theta_j\, S\alpha_j\, MY_j \\
ZZR_s &= ZZ_s + 2\,(d_j C\theta_j + b_j S\theta_j S\alpha_j)\, MX_j - 2\,(d_j S\theta_j + b_j C\theta_j S\alpha_j)\, MY_j
\end{aligned} \tag{4.18}$$

Theorem 4: From the previous theorems, we deduce that the number of minimum inertial parameters is equal to or less than:

$$ m_m = 7 \sum_{i=1}^{n} \bar{\sigma}_i + 4 \sum_{i=1}^{n} \sigma_i - 3 \sum_{k} \sigma_k - 5 \sum_{k} \bar{\sigma}_k\,, \text{ with } a(k) = 0 $$

The function “base inertial parameters” (symbolic method) in menu **Identification** permits to calculate automatically the minimum inertial parameters of tree structure robots.

4.3. Dynamic model of closed loop robots

4.3.1. Introduction

Many methods have been proposed to calculate the dynamic equations of closed loop robots [Chace 67, Uicker 69, Chace 71, Wittenburg 77, Megahed 84, Touron 84, Luh 85, Kleinfinger 86b, Giordano 86]. We present here a method based on the calculation of the dynamic model of an equivalent tree structure and a Jacobian matrix between the variables of the tree structure and the motorized joints [Kleinfinger 86b, Khalil 99, Khalil 02a].

4.3.2. Description of the system

The system is composed of n moving links and N_j joints, it can be described by the method given in the first chapter. We recall that the number of closed loops is equal to $B = N_j - n$. The number of motorized joints is given by $N < N_j$. It is supposed that the structure is compatible with the geometric constraint of the loops, such that the number of degrees of freedom of the system is equal to the motorized joints N. This condition is verified on all the industrial robots with closed loops structure.

The loops are supposed opened in one of its joints, which lead to an equivalent tree structure system. The opened joints must be chosen among the non-motorized (passive) joints [Smith 73, Wittenburg 77]. We choose to open a joint such that the difference between the number of links from the base of the loop to the links on each side of the opened joint is minimum. This choice reduces computational complexity [Kleinfinger 86a].

We denote:
- $\mathbf{q}_{ar}$ the vector containing the n joint variables of the equivalent tree structure,
- $\mathbf{q}_a$ the vector containing the N motorized (active) joint variables of the equivalent tree structure,
- $\mathbf{q}_p$ the vector containing the $p = n - N$ passive joint variables of the equivalent tree structure.

Thus:

$$\mathbf{q}_{ar} = \begin{bmatrix} \mathbf{q}_a \\ \mathbf{q}_p \end{bmatrix} \tag{4.19}$$

There are p independent relations between the variables $\mathbf{q}_a$ and $\mathbf{q}_p$. These relations constitute the constraint geometric equations of the closed loops. They are given by:

$$\mathbf{0} = \phi(\mathbf{q}_{ar}) = \begin{bmatrix} \phi_1(\mathbf{q}_{ar}) \\ \cdots \\ \phi_p(\mathbf{q}_{ar}) \end{bmatrix} \tag{4.20}$$

The original system is represented by the equivalent tree structure and the constraint geometric equations (4.20).

4.3.3. Calculation of $\mathbf{q}_p$ using the constraint equations

The more convenient case is to obtain explicit relations giving the passive variables in terms of the active variables such that:

$$\mathbf{q}_p = \mathbf{f}(\mathbf{q}_a) \tag{4.21}$$

Such relation can be obtained in menu **geometric** if the loop contains three passive joints. If explicit relations cannot be obtained, we make use of the constraint kinematic equations of loops given in menu **kinematic**. From equation (4.20), we obtain:

$$\mathbf{0} = \frac{\partial \phi}{\partial \mathbf{q}_{ar}} \mathbf{dq}_{ar} = [\frac{\partial \phi}{\partial \mathbf{q}_a} \; \frac{\partial \phi}{\partial \mathbf{q}_p}] \begin{bmatrix} \mathbf{dq}_a \\ \mathbf{dq}_p \end{bmatrix} \tag{4.22}$$

from which:

$$\frac{\partial \phi}{\partial \mathbf{q}_p} \mathbf{dq}_p = - \frac{\partial \phi}{\partial \mathbf{q}_a} \mathbf{dq}_a \tag{4.23}$$

Equation (4.23) can be written as:

$$\mathbf{W}_p \, \mathbf{dq}_p = - \mathbf{W}_a \, \mathbf{dq}_a \tag{4.24}$$

with:

$$\mathbf{W}_a = \frac{\partial \phi}{\partial \mathbf{q}_a} \qquad \text{of dimension (p x N)} \tag{4.25}$$

$$\mathbf{W}_p = \frac{\partial \phi}{\partial \mathbf{q}_p} \qquad \text{of dimension (p x p)} \tag{4.26}$$

For a compatible system, $\mathbf{W}_p$ is of rank p, except in some singular positions. Equation (4.24) becomes:

$$\mathbf{dq}_p = \mathbf{W} \, \mathbf{dq}_a \tag{4.27}$$

with:

$$\mathbf{W} = -\mathbf{W}_p^{-1}\mathbf{W}_a = \frac{\partial \mathbf{q}_p}{\partial \mathbf{q}_a} \tag{4.28}$$

Equation (4.27) gives $\mathbf{dq}_p$ corresponding to a given $\mathbf{dq}_a$.

It is to be noted that the matrices $\mathbf{W}_p$ and $\mathbf{W}_a$ are calculated by SYMORO+ using the kinematic constraint equations as given in section (2.6).

4.3.4. Calculation of the dynamic model

For a system with constraints between the variables, the Lagrange dynamic model is given as:

$$\Gamma_i = \frac{d}{dt}\frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i} + \sum_k \lambda_k \frac{\partial \phi_k}{\partial q_i} \quad \text{for } i = 1, ..., n \tag{4.29}$$

where λ_k is the kth Lagrange multiplier, and L is the Lagrangian of the equivalent tree structure.

We can rewrite these equations, in matrix form as:

$$\mathbf{\Gamma} = \mathbf{\Gamma}_{ar} + \mathbf{\Gamma}_c \tag{4.30}$$

where $\mathbf{\Gamma}$ represent the torque of the joints of the closed loop system and that:

$$\mathbf{\Gamma}_{ar} = \mathbf{A}\ddot{\mathbf{q}} + \mathbf{B}\dot{\mathbf{q}}\dot{\mathbf{q}} + \mathbf{C}\dot{\mathbf{q}}^2 + \mathbf{Q} \tag{4.31}$$

$$\mathbf{\Gamma}_c = [\frac{\partial \phi}{\partial \mathbf{q}_{ar}}]^T \lambda \tag{4.32}$$

$\mathbf{\Gamma}_{ar}$ is calculated by the Newton-Euler inverse dynamic model of the equivalent tree structure system. It can be obtained using the menu **dynamic,** function “Newton Euler method”. While λ is the vector of Lagrange multiplier.

We can write:

$$\mathbf{\Gamma}_{ar} = \begin{bmatrix} \mathbf{\Gamma}_a \\ \mathbf{\Gamma}_p \end{bmatrix} \tag{4.33}$$

where $\mathbf{\Gamma}_a$ and $\mathbf{\Gamma}_p$ denote the torques vectors corresponding to active and passive joints respectively.

$\mathbf{\Gamma}_c$ can be calculated using equation (4.32) as:

$$\mathbf{\Gamma}_c = \begin{bmatrix} \frac{\partial \phi}{\partial \mathbf{q}_a} & \frac{\partial \phi}{\partial \mathbf{q}_p} \end{bmatrix}^T \lambda$$

$$= [\mathbf{W}_a \quad \mathbf{W}_p]^T \lambda \tag{4.34}$$

Since the torque of the passive joints are zero, the vector $\mathbf{\Gamma}$, of dimension (nx1), can be partitioned as follows:

$$\mathbf{\Gamma} = \begin{bmatrix} \mathbf{\Gamma}_m \\ \mathbf{0} \end{bmatrix} \tag{4.35}$$

where $\mathbf{\Gamma}_m$ contains the torques of the N motorized joints.

Equation (4.30) becomes:

$$\mathbf{\Gamma} = \begin{bmatrix} \mathbf{\Gamma}_m \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{\Gamma}_a \\ \mathbf{\Gamma}_p \end{bmatrix} + \begin{bmatrix} \mathbf{W}_a^T \boldsymbol{\lambda} \\ \mathbf{W}_p^T \boldsymbol{\lambda} \end{bmatrix} \tag{4.36}$$

Thus, we have a system of n equations, where the unknowns are $\mathbf{\Gamma}_m$ and $\boldsymbol{\lambda}$. From the second row of (4.36), we obtain:

$$\boldsymbol{\lambda} = -[\mathbf{W}_p^T]^{-1} \mathbf{\Gamma}_p \tag{4.37}$$

Using (4.37) into (4.36):

$$\mathbf{\Gamma}_m = \mathbf{\Gamma}_a - \mathbf{W}_a^T [\mathbf{W}_p^T]^{-1} \mathbf{\Gamma}_p \tag{4.38}$$

Using equation (4.28), we obtain:

$$\mathbf{\Gamma}_m = \mathbf{\Gamma}_a + \mathbf{W}^T \mathbf{\Gamma}_p = [\,\mathbf{I} \quad \mathbf{W}^T\,] \begin{bmatrix} \mathbf{\Gamma}_a \\ \mathbf{\Gamma}_p \end{bmatrix} \tag{4.39}$$

which can be rewritten as:

$$\begin{aligned} \mathbf{\Gamma}_m &= [\,\mathbf{I} \quad \mathbf{W}^T\,]\, \mathbf{\Gamma}_{ar} \\ &= [\,[\frac{\partial \mathbf{q}_a}{\partial \mathbf{q}_a}]^T \quad [\frac{\partial \mathbf{q}_p}{\partial \mathbf{q}_a}]^T\,]\, \mathbf{\Gamma}_{ar} \\ &= \mathbf{G}^T \mathbf{\Gamma}_{ar} \end{aligned} \tag{4.40}$$

$\mathbf{G}$, equal to $\frac{\partial \mathbf{q}_{ar}}{\partial \mathbf{q}_a}$, represents the Jacobian of $\mathbf{q}_{ar}$ with respect to $\mathbf{q}_a$.

Equations (4.38), (4.39) or (4.40) constitute the dynamic model of the closed loop system.

4.3.5. Determination of the constraint equations

The geometric constraint equations are obtained by the solution of the transformation matrix around the closed loops (see section 2.3.7).

4.3.5.1. Velocity constraint equations of loops

The matrices $\mathbf{W}_p$ and $\mathbf{W}_a$ can be obtained either by differentiating the geometric constraint equations or by calculating the kinematic constraint equations of the closed loops. This last method is presented in section 2.7, it leads to obtain the following relation:

$$\begin{bmatrix} \mathbf{W_a} & \mathbf{W_p} & \mathbf{0} \\ \mathbf{W_{ac}} & \mathbf{W_{pc}} & \mathbf{Wc} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{q}}_a \\ \dot{\mathbf{q}}_p \\ \dot{\mathbf{q}}_c \end{bmatrix} = \mathbf{0} \tag{4.41}$$

where $\dot{\mathbf{q}}_c$ defines the velocity of the opened joints. **SYMORO+** can take into account complex joints such as spherical, universal and cylindrical joints.

The first row of (4.41) gives:

$$\dot{\mathbf{q}}_p = \mathbf{W}\,\dot{\mathbf{q}}_a \tag{4.42}$$

with:

$$\mathbf{W} = -\mathbf{W}_p^{-1}\,\mathbf{W}_a$$

4.3.5.2. The acceleration constraint equations

These equations are presented in section 2.7, they are given by:

$$\begin{bmatrix} \mathbf{W_a} & \mathbf{W_p} & \mathbf{0} \\ \mathbf{W_{ac}} & \mathbf{W_{pc}} & \mathbf{Wc} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}}_a \\ \ddot{\mathbf{q}}_p \\ \ddot{\mathbf{q}}_c \end{bmatrix} + \begin{bmatrix} \mathbf{Psi} \\ \mathbf{Phi} \end{bmatrix} = \mathbf{0} \tag{4.43}$$

If the loops are composed of parallelograms the elements **Psi** and **Phi** will be zero.

4.4. Calculation of the direct (simulation) dynamic model

To simulate a robot the calculation of the inertia matrix and the vector of Coriolis, centrifugal and gravity torques are needed, see section 3.6.1, such that the vector $\ddot{\mathbf{q}}_a$ can be calculated using the input torque and the state of the robot.

The dynamic model of the closed loop is written as:

$$\Gamma_m = [\,\mathbf{I} \quad \mathbf{W}^T\,]\,\mathbf{A_{ar}} \begin{bmatrix} \ddot{q}_a \\ \\ \ddot{q}_p \end{bmatrix} + [\,\mathbf{I} \quad \mathbf{W}^T\,]\,\mathbf{H_{ar}} \tag{4.44}$$

where $\mathbf{A_{ar}} = \begin{bmatrix} \mathbf{A_{aa}} & \mathbf{A_{ap}} \\ \\ \mathbf{A_{pa}} & \mathbf{A_{pp}} \end{bmatrix}$, $\mathbf{H_{ar}} = \begin{bmatrix} \mathbf{H_a} \\ \\ \mathbf{H_p} \end{bmatrix}$ represent the inertia matrix and the Coriolis, centrifugal, and gravity torques of the equivalent tree structure robot. Thus we obtain:

$$\Gamma_m = (\mathbf{A_{aa}}\,\ddot{q}_a + \mathbf{A_{ap}}\,\ddot{q}_p) + \mathbf{W}^T(\mathbf{A_{pa}}\,\ddot{q}_a + \mathbf{A_{pp}}\,\ddot{q}_p) + \mathbf{H_a} + \mathbf{W}^T\,\mathbf{H_p} \tag{4.45}$$

Using equation (4.43), we obtain:

$$\ddot{q}_p = \mathbf{W}\,\ddot{q}_a - [\,\mathbf{W_p}\,]^{-1}\,\mathbf{Psi} \tag{4.46}$$

$$\begin{aligned} \Gamma_m = {} & \mathbf{A_{aa}}\,\ddot{q}_a + \mathbf{A_{ap}}\,(\mathbf{W}\,\ddot{q}_a - [\,\mathbf{Wp}\,]^{-1}\,\mathbf{Psi}) + \mathbf{W}^T(\mathbf{A_{pa}})\,\ddot{q}_a + \\ & \mathbf{W}^T\,\mathbf{A_{pp}}\,(\mathbf{W}\,\ddot{q}_a - [\,\mathbf{Wp}\,]^{-1}\,\mathbf{Psi}) + \mathbf{H_a} + \mathbf{W}^T\,\mathbf{H_p} \end{aligned} \tag{4.47}$$

$$\Gamma_m = \mathbf{A_c}\,\ddot{q}_a + \mathbf{H_c}$$

with:

$$\mathbf{A_c} = \mathbf{A_{aa}} + \mathbf{A_{ap}}\,\mathbf{W} + \mathbf{W}^T\,\mathbf{A_{pa}} + \mathbf{W}^T\,\mathbf{A_{pp}}\,\mathbf{W} \tag{4.48}$$

$$\mathbf{H_c} = \mathbf{H_a} + \mathbf{W}^T\,\mathbf{H_p} - (\mathbf{A_{ap}} + \mathbf{W}^T\,\mathbf{A_{pp}})\,[\,\mathbf{Wp}\,]^{-1}\,\mathbf{Psi} \tag{4.49}$$

where $\mathbf{A_c}$ and $\mathbf{H_c}$ represent the inertia matrix and the Coriolis, centrifugal and gravity torques of the closed loop robot respectively.

Remarks:

- the accelerations of the passive variables can be obtained using the first row of equation (4.43).
- the accelerations of the opened variables can be obtained by integrating the second row of equation (4.43).

4.5. Minimum inertial parameters of closed loops

The relations (3.70) can be used to determine the base inertial parameters of the closed loops. The h_i functions of the closed loop structure can be obtained from the h_i functions of the corresponding tree structure by expressing them in terms of the active joint positions and velocities and not as a function of the tree structure variables. Doing this symbolically will be complicated, in [Bennis 90, Bennis 91a] the problem has been solved for parallelogram closed loops without calculating the energy functions h_i, in [Khalil 94c] partial results concerning general closed loops are given.

4.5.1. Case of general loops

General linear relations between the energy functions, h_i , of the links of general closed loops can be obtained without solving the closed loop geometric equation. These relations will permit to completely determine the base parameters in most cases.

Since the frames k and k+B, of an opened joint k, are aligned, then they have the same linear and angular velocities, thus we can write:

$$\mathbf{h}^k = \mathbf{h}^{k+B} \tag{4.50}$$

We note that a(k) = i, and a(k+B) = j.

Two cases are considered:

1- if joint k is revolute:

Let a(k) be equal to i and a(k+B) be equal to j, see section 1.5, then using equation (3.81) we obtain:

$$\text{a- } \mathbf{h}^j \, {}^j\lambda_{k+B}^{10} = \mathbf{h}^i \, {}^i\lambda_k^{10} \tag{4.51-a}$$

$$\text{b- } \mathbf{h}^j \, {}^j\lambda_{k+B}^{9} = \mathbf{h}^i \, {}^i\lambda_k^{9} \tag{4.51-b}$$

$$\text{c- } \mathbf{h}^j \, {}^j\lambda_{k+B}^{1+4} = \mathbf{h}^i \, {}^i\lambda_k^{1+4} \tag{4.51-c}$$

The vectors ${}^j\lambda_{k+B}^{10}$, ${}^j\lambda_{k+B}^{9}$, ${}^j\lambda_{k+B}^{1+4}$ are expressed in terms of the constant geometric parameters defining frame k+B, while the vectors ${}^i\lambda_k^{10}$, ${}^i\lambda_k^{9}$, ${}^i\lambda_k^{1+4}$ are expressed in terms of the constant geometric parameters defining frame k.

2- if joint k is prismatic:

We can write the following six relations:

$$\text{a- } \mathbf{h}^j \, {}^j\lambda_{k+B}^{1} = \mathbf{h}^i \, {}^i\lambda_k^{1} \tag{4.52-a}$$

$$\dots \quad \dots \quad = \quad \dots \qquad \dots$$

$$\text{f- } \mathbf{h}^j \, {}^j\lambda_{k+B}^{6} = \mathbf{h}^i \, {}^i\lambda_k^{6} \tag{4.52-f}$$

The vectors $({}^j\lambda_{k+B}^{1}, \ldots, {}^j\lambda_{k+B}^{6})$ and $({}^i\lambda_k^{1}, \ldots, {}^i\lambda_k^{6})$ are expressed in terms of the constant parameters defining frame k+B, and frame k respectively.

From equations (4.51) and (4.52) we deduce that there are three linear relations between the elements of $\mathbf{h}^{\mathbf{i}}$ and $\mathbf{h}^{\mathbf{j}}$ (if joint k is rotational) and six if joint j is prismatic. The parameters to be eliminated and grouped can be performed using (3.70), the choice is not unique but at any case we have to choose the parameters which will not be grouped using the tree structure relations. The general rule is to group the parameters of the link of largest index, and for the parameters of the same link we have to choose the parameter of largest index with respect to the order of equation (3.71).

The algorithm used in SYMORO+ is given in [Khalil 94c], it gives most of the base inertial parameters. The numerical method, whose outline is given in Appendix 3, and is available also in SYMORO+, must be used to verify the result [Gautier 90b].

4.5.2. Base inertial parameters of parallelogram closed loops

Such structure is commonly used in industrial robots, suppose that the links L_{k1}, L_{k2}, L_{k3} and L_{k4} constitute a parallelogram. In this case we consider at first the general grouping relations that will lead to group MX_{k4} with the parameters of links k3 and k4. Another more complicated relations that lead to eliminate MX_{k4} using the particular equations of the parallelogram loop is developed in [Bennis 91a].

Using the particularity of parallelogram:

Since the links L_{k1} and L_{k3} are parallel then the (3x3) orientation matrix between frames k_1 and k_3 is constant. Similarly the rotation matrix between frames k_2 and k_4 is constant.

$$^{k1}\mathbf{A}_{k3} = \text{constant} \tag{4.53-a}$$

and $$^{k2}\mathbf{A}_{k4} = \text{constant} \tag{4.53-b}$$

Using the fact that the parallel links have the same angular velocity, thus:

$$\omega_{k1} = \omega_{k3} \tag{4.54-a}$$

and $$\omega_{k2} = \omega_{k4} \tag{4.54-b}$$

We can group the inertia matrix of link k_4 with the inertia matrix of link k_2 and the inertia matrix of link k_3 with the inertia matrix of link k_1, using the following relations:

$$^{k1}\mathbf{JR}_{k1} = {}^{k1}\mathbf{J}_{k1} + {}^{k1}\mathbf{A}_{k3}\, {}^{k3}\mathbf{J}_{k3}\, {}^{k3}\mathbf{A}_{k1} \tag{4.55}$$

$$^{k2}\mathbf{JR}_{k2} = {}^{k2}\mathbf{J}_{k2} + {}^{k2}\mathbf{A}_{k4}\, {}^{k4}\mathbf{J}_{k4}\, {}^{k4}\mathbf{A}_{k2} \tag{4.56}$$

4.5.3. Algorithm of the symbolic determination of base parameters

The following rules permit to define most of the parameters that will be grouped or eliminated. We have to verify if there are more parameters to be eliminated by the use of the numerical method [Khalil 94c].

For j = n, ..., 1 do:

1- If link j represents: (k_4 or k_3 in a parallelogram closed loop) or (the link of biggest index and adjacent to an opened joint in a general closed loop), apply the corresponding grouping relations.

2- Use the general grouping relations (theorem 1 and 2) to group the following parameters:
   i- YY_j , MZ_j , M_j if joint j is revolute,
   ii- XX_j, XY_j, XZ_j, YY_j, YZ_j, ZZ_j, if joint j is prismatic for j = n, ..., 1.

3- If joint j is prismatic and $\mathbf{a}_j$ // $\mathbf{a}_{r1}$ when $r_1 < j < r_2$, then eliminate MZ_j and group MX_j and MY_j using (4.17).

4- If joint j is prismatic, $\mathbf{a}_j$ is not parallel to $\mathbf{a}_{r1}$, and $r_1 < j < r_2$, then group or eliminate one of the parameters MX_j , MY_j , MZ_j , using table 4.1.

5- If joint j is revolute, and $r_1 \leq j < r_2$ (the axes of these joints are parallel to r_1), then eliminate XX_j , XY_j , XZ_j , YZ_j . It is to be noted that YY_1 has also been eliminated in step 2.

6- If joint j is revolute, $r_1 \leq j < r_2$, $\mathbf{a}_j$ is along $\mathbf{a}_{r1}$ and ($\mathbf{a}_{r1}$ // $\mathbf{a}_i$ // $\mathbf{g}$ for all i < j), then eliminate MX_j , MY_j they have no effect. It is to be noted that MZ_j has also been eliminated in step 1.

7- When $j < r_1$ (they represent the prismatic links before r_1, thus $\omega_j = \mathbf{0}$), then eliminate MX_j , MY_j , MZ_j they have no effect.

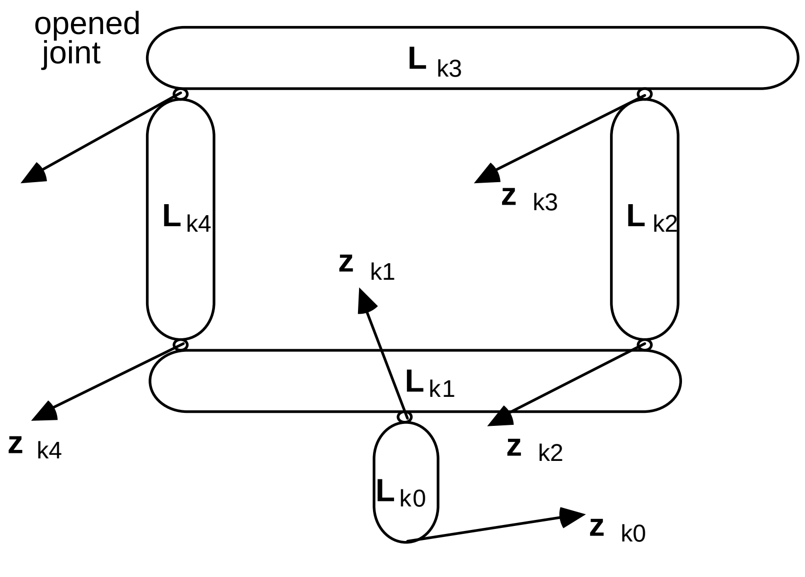


Figure 4.1a : Example of a parallelogram loop.

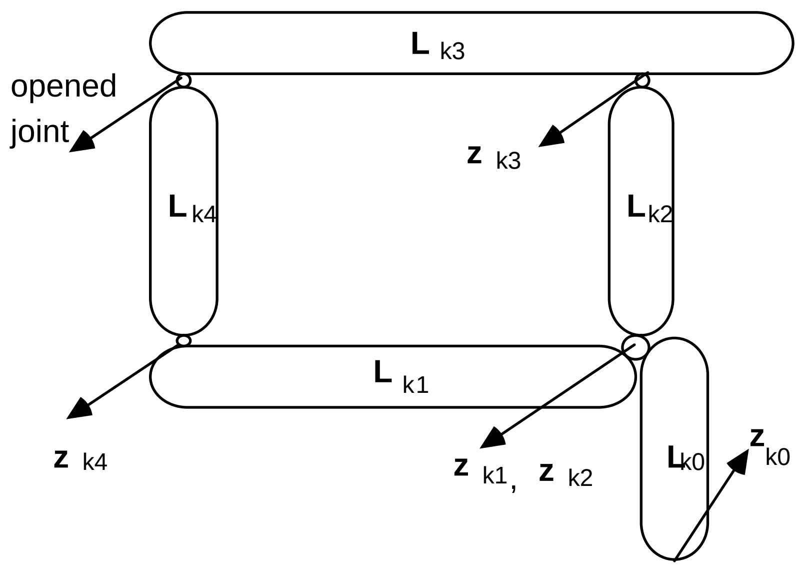


Figure 4.1b : Example of a parallelogram loop.

4.6. Conclusion

In this chapter, we have considered the generation of the dynamic model of complex structure robots. The models of tree structure robots are automatically generated by the menu **dynamic of SYMORO+**. For closed loop robots, the model is obtained by multiplying the dynamic model of an equivalent tree structure by a Jacobian matrix. This matrix represents the Jacobian of the tree structure variables with respect to the motorized joint variables, it can be obtained using the menu **kinematic,** or by differentiating the explicit geometric constraint equations that may be obtained by the menu **geometric**. The base inertial parameters of the links of tree structure robots will be completely determined by the symbolic or numerical method. In the case of closed loop robots, most of the base parameters will be obtained automatically by the symbolic method, all of them will be obtained by the numerical method. We note that the dynamic models of parallel robots can be obtained by the SYMORO+ method of closed loop robots, but a particular method for these structures is developed in [Khalil 02b].

Appendix 4.1 : General method of calculating the inverse geometric model

A complete description of the general method for calculating the inverse geometric model using classical Denavit and Hartenberg notation can be found in [Raghavan 89, Mavroidis 93]. Here we describe the outline of this method for 6R robots using our notation of robot description. The main steps of the method are:

1) The homogeneous matrix transformation equation of a 6R degree of freedom serial manipulator is written under the form:

$$ {}^{0}\mathbf{T}_1\, {}^{1}\mathbf{T}_2\, {}^{2}\mathbf{T}_3\, {}^{3}\mathbf{T}_4 = \mathbf{U}_0\, {}^{6}\mathbf{T}_5\, {}^{5}\mathbf{T}_4 \qquad \text{(A1-1)}$$

Expanding this equation, we find that its elements are functions of the following variables:

$$\begin{bmatrix} \theta1,\theta2,\theta3,\theta4 & \theta1,\theta2,\theta3,\theta4 & \theta1,\theta2,\theta3 & \theta1,\theta2,\theta3 \\ \theta1,\theta2,\theta3,\theta4 & \theta1,\theta2,\theta3,\theta4 & \theta1,\theta2,\theta3 & \theta1,\theta2,\theta3 \\ \theta1,\theta2,\theta3,\theta4 & \theta1,\theta2,\theta3,\theta4 & \theta1,\theta2,\theta3 & \theta1,\theta2,\theta3 \end{bmatrix} = \begin{bmatrix} \theta5,\theta6 & \theta5,\theta6 & \theta5,\theta6 & \theta5,\theta6 \\ \theta5,\theta6 & \theta5,\theta6 & \theta5,\theta6 & \theta5,\theta6 \\ \theta5,\theta6 & \theta5,\theta6 & \theta5,\theta6 & \theta5,\theta6 \end{bmatrix} \qquad \text{(A1-2)}$$

2) The following 14 scalar equations that are devoid of θ_4 are calculated analytically:

$$\mathbf{a}_1 = \mathbf{a}_2 \qquad \text{(A1-3a)}$$
$$\mathbf{P}_1 = \mathbf{P}_2 \qquad \text{(A1-3b)}$$
$$(\mathbf{a.P})_1 = (\mathbf{a.P})_2 \qquad \text{(A1-3c)}$$
$$(\mathbf{P.P})_1 = (\mathbf{P.P})_2 \qquad \text{(A1-3d)}$$
$$(\mathbf{a}\mathrm{x}\mathbf{P})_1 = (\mathbf{a}\mathrm{x}\mathbf{P})_2 \qquad \text{(A1-3e)}$$
$$\{\mathbf{a(P.P) - 2P(a.P)}\}_1 = \{\mathbf{a(P.P) - 2P(a.P)}\}_2 \qquad \text{(A1-3f)}$$

where**:**
- **a** and **P** are the vectors formed by the first three elements of columns 3 and 4 respectively of equation (A1-1)
- the subscripts "1" and "2" mean the left and right hand side of equation (A1-1) respectively.

We note that the right hand side equations of (A1-3) are linear in S5S6, S5C6, C5S6, C5C6, S5, C5, S6 and C6 while the left hand side are linear of S2S3, S2C3, C2S3, C2C3 , S2 , C2 , S3, C3, and 1, with coefficients which are functions of S1 and C1.

3) These equations form the following system:

$$\mathbf{A\,X1 = B\,Y} \qquad \text{(A1-4)}$$

where:

A is a (14x9) matrix whose elements are functions of S1 and C1,
B is a (14x8) matrix whose elements are constant,

$$\mathbf{X1} = [\text{ S2S3, S2C3, C2S3, C2C3, S2, C2, S3, C3, 1 }]^T \qquad \text{(A1-5)}$$
$$\mathbf{Y} = [\text{ S5S6, S5C6, C5S6, C5C6, S5, C5, S6, C6 }]^T \qquad \text{(A1-6)}$$

4) **Elimination of θ5 and θ6:** using 8 equations out of the 14 equations of (A1-4) to eliminate **Y**; the resulting system of 6 equations takes the form:

$$\mathbf{D\,X1} = \mathbf{0} \tag{A1-7}$$

where the elements of the (6x9) matrix **D** are functions of S1, C1, the constant geometric parameters and $\mathbf{U}_0$.

5) **Elimination of θ2 and θ3:** Using the following substitutions into equation (A1-7):

$$Si = 2x_i / (1 + x_i^2) \qquad \text{and} \qquad Ci = (1 - x_i^2) / (1 + x_i^2)$$

where $x_i = \tan(\theta_i/2)$, we obtain the following system:

$$\mathbf{E\,X2} = \mathbf{0} \tag{A1-8}$$

where **E** is a (6 x 9) matrix, whose elements are quadratic functions of x_1 , and

$$\mathbf{X2} = [x_2^2x_3^2, x_2^2x_3, x_2^2, x_2x_3^2, x_2x_3, x_2, x_3^2, x_3, 1]^T \tag{A1-9}$$

6) Multiplying the 6 equations of (A1-8) by x_2, we obtain new 6 supplementary equations with only 3 additional variables:

$$\mathbf{E\,X3} = \mathbf{0} \tag{A1-10}$$

where:

$$\mathbf{X3} = [x_2^3x_3^2, x_2^3x_3, x_2^3, x_2^2x_3^2, x_2^2x_3, x_2^2, x_2x_3^2, x_2x_3, x_2]^T \tag{A1-11}$$

Grouping equations (A1-8) and (A1-10), we obtain the following system of equations of dimension (12x12):

$$\mathbf{\Sigma\,X} = \mathbf{0} \tag{A1-12}$$

where **Σ** is a (12 x 12) matrix, whose elements are quadratic functions of x_1.

Σ has the following form:

$$\Sigma = \begin{bmatrix} \mathbf{E} & \mathbf{0}_{6x3} \\ \mathbf{0}_{6x3} & \mathbf{E} \end{bmatrix} \tag{A1-13}$$

$$\mathbf{X} = [x_2^3x_3^2, x_2^3x_3, x_2^3, x_2^2x_3^2, x_2^2x_3, x_2^2, x_2x_3^2, x_2x_3, x_2, x_3^2, x_3, 1]^T \tag{A1-14}$$

7) The condition:

$$\det \Sigma = 0 \tag{A1-15}$$

gives the characteristic polynomial of the manipulator in x_1. For a **6R** general manipulator the degree of this polynomial is 16.

8) Substituting back each real solution of θ_1 in step 6, we can calculate a unique value for θ_2 and θ_3 by solving a linear system of equations. Further substitution of θ_1, θ_2, θ_3 in step 6 gives a unique value for θ_5, θ_6. Finally, θ_4 follows from two elements of the first two columns of equation (A1-1).

It has been shown that the same method can be applied to solve the inverse kinematics problem of general **5R1P, 4R2P,** and **3R3P** robots. The only difference is that, in the elements of vectors $\mathbf{X_1}$ and **Y** of equation (A1-4), for a prismatic joint i, c_i is replaced by r_i and s_i is replaced by r_i^2 .

General Algorithm:

There are many different ways of writing the loop closure equation such as:

$$ {}^4T_5\, {}^5T_6\, {}^6T_7\, {}^0T_1 = {}^4T_3\, {}^3T_2\, {}^2T_1 \quad \text{(A1-16a)}$$

$$ {}^5T_6\, {}^6T_7\, {}^0T_1\, {}^1T_2 = {}^5T_4\, {}^4T_3\, {}^3T_2 \quad \text{(A1-16b)}$$

$$ {}^6T_7\, {}^0T_1\, {}^1T_2\, {}^2T_3 = {}^6T_5\, {}^5T_4\, {}^4T_3 \quad \text{(A1-16c)}$$

$$ {}^0T_1\, {}^1T_2\, {}^2T_3\, {}^3T_4 = {}^7T_6\, {}^6T_5\, {}^5T_4 \quad \text{(A1-16d)}$$

$$ {}^1T_2\, {}^2T_3\, {}^3T_4\, {}^4T_5 = {}^1T_0\, {}^7T_6\, {}^6T_5 \quad \text{(A1-16e)}$$

$$ {}^2T_3\, {}^3T_4\, {}^4T_5\, {}^5T_6 = {}^2T_1\, {}^1T_0\, {}^7T_6 \quad \text{(A1-16f)}$$

with ${}^7T_6 = U_0$ and ${}^6T_7 = U_0^{-1}$

These equations give the matrix Σ and the characteristic polynomial as a function of xi, with i = 1, …, 6, respectively.

In general, for **6R** manipulators any of the equations (A1-16) can be used as starting equation, but owing to some conditions between the geometric parameters, or if the robot has prismatic joints, some of these equations cannot be used. For instance if the manipulator has a prismatic joint, the equation where this joint is equal to the last subscript of the left-hand side cannot be used. Mavroidis [Mavroidis 93] has given some conditions between the geometric parameters to avoid some of these equations, because the resulting Σ matrix will contain independent columns. Instead of eliminating such equations we propose to take them and to reduce the number of variables in **X1** and the dimension of **A**; this choice leads to reduce the dimension of Σ and, consequently, the characteristic polynomial which may be obtained completely in symbolic and reduced form [Khalil 94b].

Reducing the dimension of matrix Σ

The following cases are considered to reduce the dimension of the Σ matrix:

1) The matrix A contains dependent columns

If some columns of the matrix **A** are linearly dependent, then the matrix Σ will contain also dependent columns and all the characteristic polynomial coefficients turn to 0. In this case new reduced **A** and Σ can be obtained by eliminating the dependent columns and grouping the corresponding elements of the vector **X1**, such that the following reduced system of equations can be obtained:

$$\mathbf{A_R}(14xa)\, \mathbf{X1_R} = \mathbf{B}(14x8)\, \mathbf{Y} \quad \text{(A1-17)}$$

where a is the number of independent columns in the matrix **A**.

$\mathbf{A_R}$ is obtained from A by eliminating the zero columns and taking only the independent columns.

$\mathbf{X1_R}$ is the (ax1) vector, obtained from **X1** after grouping the elements corresponding to dependent columns on those corresponding to the independent columns.

For instance if **A**(; i) = p **A**(; j), and **A**(; m) = 0, where **A**(; j) is the j^{th} column of the matrix **A**. Then the reduced $\mathbf{A_R}$ is formed from A after eliminating the columns j and m, while $\mathbf{X1_R}$ is formed from **X1** after eliminating the m^{th} and j^{th} elements and replacing the i^{th} element by the sum **X1**(i) + p **X1**(j).

For the solution of the regrouped system two cases are considered:

1) If a is less than 6

   This means that he effective number of unknowns a is less than 6 then the calculation of **E** and **Σ** is not needed, the following system of equations can be used directly:

   $$\mathbf{C\,X1_R = 0} \qquad \text{(A1-18)}$$

   where the matrix **C** of dimension (axa), is obtained from **D** (step 2) after:

   a- eliminating arbitrarily (6-a) rows

   b- replacing $C(\theta_i)$ and $S(\theta_i)$ as function of x_i.

   As the last element of the system (A1-18) is unity, thus (det **C** = 0) gives the solution of the variable i. Substituting qi in (A1-18) permits to obtain the grouped vector $\mathbf{X1_R}$ by solving a system of linear equations with a-1 unknowns. The solution of the joint variables j and k will be consequently obtained. It is to be noted that since the elements of $\mathbf{X1_R}$ are linear combinations of the elements of **X1**, multiple solutions may be obtained in this step. The variables of the right hand side (R.H.S.) of equation (A1-18) will be obtained by solving a system of linear equations.

2) If a is greater than 6

   The number of independent columns of A is greater than 6. In this case the calculation of **Σ** (step 4) must be carried out. But its dimension will be reduced, thus arbitrarily rows can be eliminated to obtain a square matrix.

2) The robot has more than one prismatic joint

The following two cases have been given [Raghavan 90]:

- In the case of the **4R2P** manipulator, the characteristic polynomial can be obtained as the determinant of **Σ** either of 12x12 matrix if i is prismatic or 6x6 matrix if i is rotational.

- In the case of **3P3R** robots, the characteristic polynomial is obtained from the determinant of **Σ** either of 6x6 matrix if i is prismatic or 3x3 matrix if i is rotational.

Symbolic versus numerical calculations

For many robots it is impossible to obtain symbolically the characteristic polynomial. Thus where does symbolic computation stop and where do we start to continue the calculation with the numerical values of the desired situation? as there is no general answer for this question, we leave the user decide where to stop the symbolic computation. The following stop points can be defined:

i- calculation of equation (A1-4): $\mathbf{A\,X1 = B\,Y}$

ii- calculation of (A1-8): $\mathbf{E\,X2} = \mathbf{0}$

iii- calculation of (A1-13): $\mathbf{\Sigma} = \begin{bmatrix} \mathbf{E} & \mathbf{0}_{6x3} \\ \mathbf{0}_{6x3} & \mathbf{E} \end{bmatrix}$

iv- calculation of (A1-15): $\det \mathbf{\Sigma} = 0$

Appendix 4.2 : Lagrange dynamic model of simple open loop robots

A 4.2.1. Introduction

In this appendix, we present the calculation of the dynamic model of robots using the Lagrange method developed in [Renaud 85], this method does not need to carry out analytical derivation. The method has been programmed using customized calculation in order to reduce the computation cost of the generated model. The model is given using the notion of generalized links, the notion of augmented links as given in [Renaud 85] is not used because the use of the base inertial parameters is more efficient.

A 4.2.2. General form of dynamic model of Lagrange

The torque vector $\boldsymbol{\Gamma}$ can be obtained as:

$$\boldsymbol{\Gamma} = \mathbf{A}\,\ddot{\mathbf{q}} + \mathbf{B}\,\dot{\mathbf{q}}\dot{\mathbf{q}} + \mathbf{C}\,\dot{\mathbf{q}}^{2} + \mathbf{Q} \tag{A2.1}$$

where:

B (n x n(n – 1)/2) matrix containing the elements of Coriolis forces, its general element is represented by B(i,j,k),

C (n x n) matrix containing the elements of centrifugal forces, its general element is represented by C(i,j),

$\dot{\mathbf{q}}\dot{\mathbf{q}} = [\dot{q}_1\dot{q}_2 \;\cdots\; \dot{q}_1\dot{q}_n \;\; \dot{q}_2\dot{q}_3 \;\cdots\; \dot{q}_{n-1}\dot{q}_n]^T$,

$\dot{\mathbf{q}}^{2} = [\dot{q}_1^{2} \;\dots\; \dot{q}_n^{2}]^T$,

$\mathbf{Q} = [Q_1 \;\dots\; Q_n]^T$, gravity forces vector.

The elements of **B**, **C** and **Q** can be computed by the derivation of the expressions of the inertia matrix **A** and the potential energy as follows:

$$B(i,j,k) = \frac{\partial A(i,j)}{\partial q_k} + \frac{\partial A(i,k)}{\partial q_j} - \frac{\partial A(j,k)}{\partial q_i} \tag{A2.2}$$

$$C(i,j) = \frac{\partial A(i,j)}{\partial q_j} - \frac{1}{2}\frac{\partial A(j,j)}{\partial q_i} \tag{A2.3}$$

$$Q(i) = \frac{\partial U}{\partial q_i} \tag{A2.4}$$

The centrifugal and Coriolis terms on joint i can be also given as:

$$(\mathbf{B}\,\dot{\mathbf{q}}\dot{\mathbf{q}} + \mathbf{C}\,\dot{\mathbf{q}}^{2})_i = \sum_{k=1}^{n}\sum_{j=1}^{n} b(i,j,k)\,\dot{q}_j\dot{q}_k \tag{A2.5}$$

The coefficients $B_{i,j,k}$ and $C_{i,j}$ can be obtained as function of b(i,j,k) as follows:

$B(i,j,k) = 2\, b(i,j,k)$	if $i < k$
$B(i,j,k) = -2\, b(k,j,i)$	if $i > k$
$B(i,j,k) = 0$	if $i = k$
$C(i,j) = b(i,j,j)$	if $i < j$
$C(i,j) = -2\, b(j,j,i)$	if $i > j$
$C(i,j) = 0$	if $i = j$

Calculation of the elements of the matrix A

At first the inertial parameters of the generalized links are computed as given in section (3.6.1), then the vectors ${}^{j}\mathbf{P}_i$, ${}^{j}\mathbf{a}_i$, ${}^{j}\mathbf{u}_i$ are calculated for $i \leq j$ and if i is on the branch composed of the links j, a(j), a(a(j)), ..., 0. using the following algorithm:

Initialization:

$${}^{j}\mathbf{a}_j = [0 \quad 0 \quad 1]^T,\ {}^{j}\mathbf{P}_j = [0 \quad 0 \quad 0]^T$$

```
for j = 1, ..., n do
   for i = 1, ..., j-1 do
      ${}^{j}\mathbf{a}_i = {}^{j}\mathbf{A}_{a(j)}\, {}^{a(j)}\mathbf{a}_i$
      ${}^{j}\mathbf{P}_i = {}^{j}\mathbf{A}_{a(j)}\, {}^{a(j)}\mathbf{P}_i + {}^{j}\mathbf{P}_{a(j)}$
      ${}^{j}\mathbf{u}_i = {}^{j}\mathbf{a}_i \times {}^{j}\mathbf{L}_{i,j} = -{}^{j}\mathbf{a}_i \times {}^{j}\mathbf{P}_i$
   end do
end do
```

Determination of the elements of the inertia matrix

It can be shown that:

$$A(i,j) = ({}^{j}\mathbf{a}_i)_x\, XZ_j^{+} + ({}^{j}\mathbf{a}_i)_y\, YZ_j^{+} + ({}^{j}\mathbf{a}_i)_z\, ZZ_j^{+} - ({}^{j}\mathbf{u}_i)_x\, MY_j^{+} + ({}^{j}\mathbf{u}_i)_x\, MY_j^{+} \qquad \text{if } \sigma_i = \sigma_j = 0$$

$$A(i,j) = ({}^{j}\mathbf{a}_i)_z\, M_j^{+} \qquad \text{if } \sigma_i = \sigma_j = 1$$

$$A(i,j) = (\sigma_i\, \bar{\sigma}_j - \bar{\sigma}_i\, \sigma_j)\, (({}^{j}\mathbf{a}_i)_y\, MX_j^{+} - ({}^{j}\mathbf{a}_i)_x\, MY_j^{+} + ({}^{j}\mathbf{u}_i)_x\, MY_j^{+} + \bar{\sigma}_i\, \sigma_j\, ({}^{j}\mathbf{u}_i)_z\, M_j^{+}) \quad \text{if } \sigma_i \neq \sigma_j$$

M_j^{+}, ${}^{j}\mathbf{J}_j^{+}$, ${}^{j}\mathbf{MS}_j^{+}$ are the inertial parameters of the generalized link j (see section 3.6.1).

Determination of the gravity torques

$$Q(j) \;=\; -\,\mathbf{g}^T\,[\bar{\sigma}_j\;{}^0\mathbf{A}_j\,[\,-MY_j^{+}\quad MX_j^{+}\quad 0\,]^T + \sigma_j\,({}^0\mathbf{a}_j)_z\,M_j^{+}]$$

Calculation of the coefficients B(i,j,k)

```
for k = 2, …, n do
    for i = 1, …, k-1 do
        for j = 1, …, k do
            $R_k^{+} \;=\; 0.5\,[\,-XX_k^{+} + YY_k^{+} + ZZ_k^{+}\,]$

            $N_k^{+} \;=\; 0.5\,[\,-XX_k^{+} - YY_k^{+} + ZZ_k^{+}\,]$

            ${}^k\Phi_j \;=\; -XY_j^{+}\,({}^k\mathbf{a}_j)_x - YZ_j^{+}\,({}^k\mathbf{a}_j)_z + N_j^{+}\,({}^k\mathbf{a}_j)_y$

            ${}^k\Psi_j \;=\; -R_j^{+}\,({}^k\mathbf{a}_j)_x + XY_j^{+}\,({}^k\mathbf{a}_j)_y + XZ_j^{+}\,({}^k\mathbf{a}_j)_z$

            ${}^k\Theta_j \;=\; MX_j^{+}\,({}^k\mathbf{a}_j)_x + MY_j^{+}\,({}^k\mathbf{a}_j)_y$

            ${}^k\lambda_j \;=\; -MX_j^{+}\,({}^k\mathbf{a}_j)_z$

            ${}^k\upsilon_j \;=\; -MY_j^{+}\,({}^k\mathbf{a}_j)_z$
        end do
    end do
end do
```

The following cases are given:

1) if $\sigma_j = 1$ then

$$b(i,j,k) \;=\; 0$$

2) if $\sigma_i = 1,\ \sigma_j = 0,\ \sigma_k = 1$

$$b(i,j,k) \;=\; ({}^j\mathbf{a}_i)_x\,(M_j^{+}\,({}^k\mathbf{a}_j)_y - ({}^k\mathbf{a}_j)_y\,(M_k^{+}\,({}^k\mathbf{a}_j)_x)$$

3) if $\sigma_i = 1,\ \sigma_j = 0,\ \sigma_k = 0$

$$b(i,j,k) \;=\; ({}^j\mathbf{a}_i)_x\,({}^k\lambda_j) + ({}^k\mathbf{a}_j)_y\,({}^k\upsilon_j) + ({}^k\mathbf{a}_j)_z\,({}^k\theta_j)$$

4) if $\sigma_i = 0,\ \sigma_j = 0,\ \sigma_k = 1$

$$b(i,j,k) \;=\; [({}^k\mathbf{u}_i)_z\,M_k^{+} - ({}^k\lambda_i) - ({}^k\mathbf{a}_i)_x\,MZ_j^{+}]\,({}^k\mathbf{a}_j)_x$$

5) if $\sigma_i = 0,\ \sigma_j = 0,\ \sigma_k = 0$

$$b(i,j,k) \;=\; [({}^k\mathbf{a}_i)_x\,{}^k\Phi_j + ({}^k\mathbf{a}_i)_y\,{}^k\Psi_j + ({}^k\mathbf{u}_i)_x\,{}^k\lambda_j + ({}^k\mathbf{u}_i)_y\,{}^k\upsilon_j + ({}^k\mathbf{u}_i)_z\,{}^k\Theta_j$$

A 4.2.3. Notations used for intermediate variables

The following intermediate variables are used:

$$^{a(j)}\mathbf{A}_j = \begin{bmatrix} \text{A11j} & \text{A12j} & \text{A13j} \\ \text{A21j} & \text{A22j} & \text{A23j} \\ \text{A31j} & \text{A32j} & \text{A33j} \end{bmatrix}$$

$$^{a(j)}\mathbf{P}_j = [\text{LOO1j} \quad \text{LOO2j} \quad \text{LOO3j}]^T$$

$$^{j}\mathbf{P}_{a(j)} = [\text{JLA1j} \quad \text{JLA2j} \quad \text{JLA3j}]^T$$

$$^{a(j)}\mathbf{A}_j \, {}^{j}\mathbf{MS}_j^+ = [\text{AS1j} \quad \text{AS2j} \quad \text{AS3j}]^T$$

$$^{a(j)}\mathbf{A}_j \, {}^{j}\mathbf{J}_j^+ = \begin{bmatrix} \text{AJ11j} & \text{AJ12j} & \text{AJ13j} \\ \text{AJ21j} & \text{AJ22j} & \text{AJ23j} \\ \text{AJ31j} & \text{AJ32j} & \text{AJ33j} \end{bmatrix}$$

$$^{a(j)}\mathbf{A}_j \, {}^{j}\mathbf{J}_j^+ \, {}^{j}\mathbf{A}_{a(j)} = \begin{bmatrix} \text{AJA11j} & \text{AJA12j} & \text{AJA13j} \\ \text{AJA21j} & \text{AJA22j} & \text{AJA23j} \\ \text{AJA31j} & \text{AJA32j} & \text{AJA33j} \end{bmatrix}$$

$$^{j}\mathbf{MS}_j^+ = [\text{MXPj} \quad \text{MYPj} \quad \text{MZPj}]^T$$

$$^{j}\mathbf{P}_i = [\text{JP1ji} \quad \text{JP2ji} \quad \text{JP3ji}]^T$$

$$^{j}\mathbf{a}_i = [\text{VZ1ji} \quad \text{VZ2ji} \quad \text{VZ3ji}]^T$$

$$^{j}\mathbf{u}_i = [\text{PRV1ji} \quad \text{PRV2j} \quad \text{PRV3ji}]^T$$

$$^{0}\mathbf{A}_j = \begin{bmatrix} \text{PMA110j} & \text{AJA12j} & \text{PMA130j} \\ \text{PMA210j} & \text{PMA220j} & \text{PMA230j} \\ \text{PMA310j} & \text{PMA320j} & \text{PMA330j} \end{bmatrix}$$

$$R_j^+ = \text{RPj}$$

$$^{k}\Phi_j = \text{PHkj}$$

$$^{k}\lambda_j = \text{LAkj}$$

$^{k}\Psi_{j}$ = PSkj

$^{k}\Theta_{j}$ = TEkj

A 4.2.4. The Lagrange model of Renaud in SYMORO+

The foregoing algorithm is used for the open loop robots (simple or tree) it gives also the dynamic coefficients for the equivalent tree structure for closed loop robots. It is included in the menu **Dynamic.**

Appendix 4.3 : Numerical calculation of the base inertial parameters

The energy of the robot is linear in terms of the inertial parameters such that:

$$H = \mathbf{h}(\mathbf{q},\dot{\mathbf{q}})\,\mathbf{K} \qquad \text{(A3-1)}$$

where:
$\mathbf{K}$ is the vector of the inertial parameters = $[XX_1 \quad XY_1 \quad \ldots \quad ZZ_n \quad IA_n]^T$
$\mathbf{h}$ is the energy function row vector function of the joints position and velocities.
n is the number of links.

The base inertial parameters can be determined by studying the dependence of the functions h_i, elements of $\mathbf{h}$. Numerically this is equivalent to study the space span by the columns of a matrix $\mathbf{W}$ calculated from r random sequences (r > 11n) ($\mathbf{q}_a(i)$, $\dot{\mathbf{q}}_a(i)$) i = 1, ..., r [Bennis 92a], where $\mathbf{q}_a$ and $\dot{\mathbf{q}}_a$ represent the position and velocity of active joints. In open loop robots all the joints are supposed active.

$$\mathbf{W} = \begin{bmatrix} \mathbf{h}(1) \\ \mathbf{h}(2) \\ \cdots \\ \mathbf{h}(r) \end{bmatrix} \qquad \text{(A3-2)}$$

A column with a constant value corresponds to an inertial parameter that doesn't affect the dynamic model. At first we eliminate such columns and suppose that the number of the other columns is equal to c, and the reduced $\mathbf{W}$ will be of (rxc) dimension.

From a linear algebra point of view, the determination of the base inertial parameters, is a rank deficiency problem.

The study is then carried out in three steps:

a- Find the rank of $\mathbf{W}$, this gives the number of the base parameters. Let it be denoted by b.

b- Choose the base parameters from the standard ones. These parameters are those corresponding to b independent columns of **W.** The choice is not unique, we choose those corresponding to the first b independent columns.

c- Determine the grouping relations from the determination of c-b linear relations between the base columns and the dependent ones.

The solution of these three problems using the QR or the SVD decomposition has been presented in [Gautier 90a,Gautier 90b].

Remarks:

1- Since we start by eliminating the constant columns, the linear relations between columns containing such constant columns will not be determined. To overcome this problem, we have to construct **W** by the difference between two sets of random configurations, such that:

$$\mathbf{W} = \begin{bmatrix} \mathbf{h}(1) \\ \mathbf{h}(2) \\ \dots \\ \mathbf{h}(r) \end{bmatrix} - \begin{bmatrix} \mathbf{h}(r+1) \\ \mathbf{h}(r+2) \\ \dots \\ \mathbf{h}(r+r) \end{bmatrix}$$

2- In the case of closed loop robots the **h** expressions are determined in terms of the joint positions and velocities of the equivalent tree structure. In order that the **h** elements of the equivalent tree structure represent the closed loop structure, the functions h_i must be expressed in terms of the active joint positions and velocities.

The main problem is to obtain $\mathbf{q}_p$ and $\dot{\mathbf{q}}_p$ as a function of $\mathbf{q}_a$ and $\dot{\mathbf{q}}_a$ in order that the matrix **W** represents the closed loop structure. If explicit constraint positions and velocities relations are available, in this case the vector **h** of the tree structure can be obtained in terms of the active variables as:

$$\mathbf{q}_p = \mathbf{g}_p(\mathbf{q}_a) \tag{A3-3a}$$

$$\dot{\mathbf{q}}_p = \mathbf{g}_v(\mathbf{q}_a)\,\dot{\mathbf{q}}_a \tag{A3-3b}$$

$\mathbf{g}_v(\mathbf{q}_a)$ is the Jacobian matrix of $\mathbf{g}_p$ with respect to $\mathbf{q}_a$.

Independent variables $\mathbf{q}_a$ and $\dot{\mathbf{q}}_a$ can be chosen as random sequences, then $\mathbf{q}_p$ and $\dot{\mathbf{q}}_p$ can be obtained using (A3-3a), this is the case in the program of SYMORO+, when an explicit relation cannot be obtained, the problem must be solved numerically [Bennis 92a] using iterative methods such as Newton-Raphson method or nonlinear optimization method.

a- Calculation of the number of base parameters

The rank deficiency solution can be obtained using QR decomposition with pivoting or by SVD factorization. The QR decomposition is used in **SYMORO+.**

If **W** is rank deficient (b < c) then, using QR decomposition the (rxc) matrix **W** can be decomposed as [Gautier 91]:

$$\mathbf{Q}^T\,\mathbf{W} = \begin{bmatrix} \mathbf{R} \\ \mathbf{0} \end{bmatrix} \tag{A3-4}$$

Q is a (rxr) orthogonal matrix.
R is a (cxc) upper triangular and regular matrix.

The number of elements of the diagonal of **R** which are not equal to zero gives the rank of **W**, the elements which will be eliminated by grouping are those corresponding to the zero elements of **R.**

The numerical rank is defined with a tolerance $\tau \neq 0$, because of round off errors.

rank(**W**) = b number of $| R_{ii} | \geq \tau$.

τ can be chosen as:

$$\tau = r\, \varepsilon \,\mathrm{Max}\, (|R_{ii}|) \tag{A3-5}$$

where ε is the precision of the computer, max $(|R_{ii}|)$ is the biggest absolute value of the elements R_{ii}.

When the values $|R_{ii}|$ are clustered in two groups, one near max $(|R_{ii}|)$ and the others less than τ, there is no problem to determine the rank of **W**.

b- Relation between independent columns and dependent columns of W

From the previous step we define a permutation matrix **P** such that:

$$\mathbf{W\,P} = [\ \mathbf{W1} \quad \mathbf{W2}\ ] \tag{A3-6}$$

W1 contains the first independent columns. The QR decomposition of **W P** gives:

$$[\ \mathbf{W1} \ \ \mathbf{W2}\ ] = [\ \mathbf{Q1} \quad \mathbf{Q2}\ ] \begin{bmatrix} \mathbf{R1} & \mathbf{R2} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} = [\ \mathbf{Q1.R1} \quad \mathbf{Q1.R2}\ ] \tag{A3-7}$$

where **R1** is a (bxb) regular matrix. Then it comes:

$$\mathbf{W1} = \mathbf{Q1.R1} \text{ and } \mathbf{W2} = \mathbf{Q1.R2}$$

We deduce the relation:

$$\mathbf{W2} = \mathbf{W1\, R1^{-1}\, R2} \tag{A3-8}$$

The relation (A3-8) gives the c-b columns of **W2** as linear combinations of the b independent columns of **W1**.

The zero rows of $\mathbf{R1^{-1}\, R2}$ correspond to the independent columns of **W P**.

Remark:

If the last diagonal elements of **R** are equal to zero, the QR decomposition algorithm of Mathematica, does not give the corresponding rows. That is to say in this case the matrix **R** is not square.

c- Explicit relations for the base parameters

As the dynamic model can be written by:

$$\mathbf{W\,.\,K} = \mathbf{W1\,.\,KB} \tag{A3-9}$$

where **K** represents the inertial parameters, and **KB** represents the base inertial parameters, then

$$\mathbf{W1\,.\,KB} = [\ \mathbf{W1} \quad \mathbf{W2}\ ] \begin{bmatrix} \mathbf{K1} \\ \mathbf{K2} \end{bmatrix} \tag{A3-10}$$

Using (A3-8) into (A3-10), we obtain:

$$\mathbf{KB} = \mathbf{K1} + \mathbf{R1}^{-1}\,\mathbf{R2}\,\mathbf{K2} \tag{A3-11}$$

Relation (A3-11) gives the numerical values of the base parameters **KB** from those of the standard parameters.

The numerical method in SYMORO+ :

The numerical method is programmed in SYMORO+, in menu **identification**, and as an option of the function "base inertial parameters". When using this function the user must take care that:

i- the gravity and constant geometric (only geometric) parameters of the robot must be given numerically, this can be done directly while defining the robot, or in the file robot-name.cons. That is to say for a robot called SR400 the name of this file must be SR400.cons, it must be located in the directory of the robot SR400.

ii- for closed loop robots the geometric and kinematic constraint equations giving the positions and velocities of passive joints in terms of positions and velocities of active joints must be given in the file "robot-name.cons" (this file can be used also to define the numerical values of the geometric parameters as in the foregoing step). If the geometric constraint file is not given, only the base parameters of the equivalent tree structure will be determined. The geometric constraint relations can be obtained using SYMORO+ in menu geometric and menu kinematic, the file SR400.cons corresponding to the robot SR400, table 1.3, is given as follows (the part written in fold must be respected).

```
Constraint[nbs_Integer] : Module [{i,Pi = 3.14159265358979},
  For [i= 1,i ≤ nbs, i++,
  Q[[i,3]] = Pi/2 - Q[[i,2]]+ Q[[i,7]];
  Q[[i,8]] =  Q[[i,2]]- Q[[i,7]];
  QD[[i,3]] = - QD[[i,2]]+ QD[[i,7]];
  QD[[i,8]] =  QD[[i,2]]- QD[[i,7]];
  ];
]
```

where:

Q[[i,j]] denotes the ith random variable of joint j.

QD[[i,j]] denotes the ith random velocity of joint j.

Remark:

1- This program defines the following equations giving the position and velocities of passive joints as functions of active joints:

$$\theta_3 = \pi/2 - \theta_2 + \theta_7$$
$$\theta_8 = \theta_2 - \theta_7$$

and

$$\dot{\theta}_3 = -\dot{\theta}_2 + \dot{\theta}_7$$

$$\dot{\theta}_8 = \dot{\theta}_2 - \dot{\theta}_7$$

2- The values of the constant geometric parameters and the values of the gravity can be given also in this file before or after the constraint sub-program for instance the values of d8 can be given as:

   d8 = 0.5 ;

   This equation must be written before the instruction:
   Constraint[nbs_Integer] : Module [{i,Pi = 3.14159265358979**},**

Appendix 4.4 : Direct dynamic model of open loop robots

A 4.4.1. Introduction

In this appendix, we present the computation of the direct dynamic model, which gives the joint accelerations, without the inversion of the inertia matrix of the robot. This method is inspired from the work of [Armstrong 79, Featherstone 83, Brandl 86]. The method has been programmed using symbolic customized calculation in order to reduce the computation cost of the generated model. This algorithm can be used to simulate systems with lumped elasticity, where the torque of such joint is set equal to $-K_j\, q_j$, with K_j is the stiffness and q_j is the position variable with respect to the zero force position [Khalil 00].

A 4.4.2. Notations

In this method we make use of the (6x1) spatial vectors of velocities, accelerations and forces. The following notations will be used:

. ${}^j\mathbb{V}_j$ is a (6x1) vector containing the linear and angular velocity of link j, such that:

$$ {}^j\mathbb{V}_j = \begin{bmatrix} {}^j\mathbf{V}_j \\ {}^j\omega_j \end{bmatrix} \tag{A4-1} $$

the vector ${}^j\mathbb{V}_j$ (6x1) is known also as the kinematic screw of link j referred to frame j, on the spatial velocity vector.

. The (6x1) vector $\mathbb{a}_j$ is given as:

$$ \mathbb{a}_j = \begin{bmatrix} \sigma_j\, {}^j\mathbf{a}_j \\ \bar{\sigma}_j\, {}^j\mathbf{a}_j \end{bmatrix} \tag{A4-2} $$

. ${}^j\mathbb{J}_j$ is the (6x6) matrix of the spatial inertia matrix of link j, it is given as:

$$ {}^j\mathbb{J}_j = \begin{bmatrix} M_j \mathbf{I}_{3x3} & -{}^j\hat{\mathbf{MS}}_j \\ {}^j\hat{\mathbf{MS}}_j & {}^j\mathbf{J}_j \end{bmatrix} \tag{A4-3} $$

. $\mathbb{F}_j = \begin{bmatrix} \mathbf{f}_j \\ \mathbf{n}_j \end{bmatrix}$ = the forces and moments exerted by link j on link j+1 , thus:

$$ {}^j\mathbb{F}_j = \begin{bmatrix} {}^j\mathbf{f}_j \\ {}^j\mathbf{n}_j \end{bmatrix} = {}^j\mathbb{F}_j^r + (\Gamma_j - Fs_j \, \mathrm{sign}(\dot{q}_j) - Fv_j \, \dot{q}_j - Ia_j \, \ddot{q}_j) \, \mathbb{a}_j \qquad \text{(A4-4)} $$

. ${}^j\mathbb{F}_j^r = \begin{bmatrix} {}^j\mathbf{f}_j^r \\ {}^j\mathbf{n}_j^r \end{bmatrix}$ is the reaction forces and moments vector, it is to be noted that its components are orthogonal on the axis of joint j, that is $\mathbb{a}_j^T \, {}^j\mathbb{F}_j^r = 0$.

. ${}^j\mathbb{F}_j = {}^j\mathbb{F}_j^r + \tau_j \, \mathbb{a}_j - Ia_j \, \ddot{q}_j \, \mathbb{a}_j$

. $\tau_j = \Gamma_j - Fs_j \, \mathrm{sign}(\dot{q}_j) - Fv_j \, \dot{q}_j$ (A4-5)

. ${}^j\mathbb{F}_{tj} = \begin{bmatrix} {}^j\mathbf{f}_{tj} \\ {}^j\mathbf{n}_j \end{bmatrix}$ is the force exerted by the link j on the environment.

. ${}^j\mathbb{T}_{j-1}$ is the (6x6) transformation matrix between screws, it is given as:

$$ {}^j\mathbb{T}_i = \begin{bmatrix} {}^j\mathbf{A}_i & -{}^j\mathbf{A}_i \, {}^i\hat{\mathbf{P}}_j \\ \mathbf{0}_{3x3} & {}^j\mathbf{A}_i \end{bmatrix} \qquad \text{(A4-6)} $$

. The transformation of velocities between frames will be given as:

$$ {}^j\mathbb{V}_j = {}^j\mathbb{T}_i \, {}^i\mathbb{V}_j \qquad \text{(A4-7)} $$

. The transformation of forces is given as:

$$ {}^j\mathbb{F}_k = {}^k\mathbb{T}_j^T \, {}^k\mathbb{F}_k \qquad \text{(A4-8)} $$

A 4.4.3. Calculation of the joint accelerations

The recursive relations of velocities (3.17) and (3.18) can be rewritten by the following equation:

$$ {}^j\mathbb{V}_j = {}^j\mathbb{T}_i \, {}^i\mathbb{V}_i + \dot{q}_j \, \mathbb{a}_j \qquad \text{(A4-9)} $$

where i = a(j), it is equal to j-1 in the case of simple open loop robots.

The recursive relations of acceleration (3.48) and (3.49) can be rewritten by the following equation:

$$ {}^{j}\dot{\mathbb{V}}_{j} = {}^{j}\mathbb{T}_{i}\, {}^{i}\dot{\mathbb{V}}_{i} + \ddot{q}_{j}\, \mathbb{a}_{j} + \begin{bmatrix} {}^{j}\mathbf{A}_{i}({}^{i}\omega_{i} \text{ x } ({}^{i}\omega_{i} \text{ x } {}^{i}\mathbf{P}_{j})) + 2\sigma_{j}\, ({}^{j}\omega_{i} \text{ x } \dot{q}_{j}\, {}^{j}\mathbf{a}_{j}) \\ \bar{\sigma}_{j}\, {}^{j}\omega_{i} \text{ x } \dot{q}_{j}\, {}^{j}\mathbf{a}_{j} \end{bmatrix} \tag{A4-10} $$

It can also be rewritten as:

$$ {}^{j}\dot{\mathbb{V}}_{j} = {}^{j}\mathbb{T}_{i}\, {}^{i}\dot{\mathbb{V}}_{i} + \ddot{q}_{j}\, \mathbb{a}_{j} + {}^{j}\gamma_{j} \tag{A4-11} $$

where:

$$ {}^{j}\gamma_{j} = \begin{bmatrix} {}^{j}\mathbf{A}_{i}({}^{i}\omega_{i} \text{ x } ({}^{i}\omega_{i} \text{ x } {}^{i}\mathbf{P}_{j})) + 2\sigma_{j}\, ({}^{j}\omega_{i} \text{ x } \dot{q}_{j}\, {}^{j}\mathbf{a}_{j}) \\ \bar{\sigma}_{j}\, {}^{j}\omega_{i} \text{ x } \dot{q}_{j}\, {}^{j}\mathbf{a}_{j} \end{bmatrix} \tag{A4-12} $$

The equilibrium equation of link j can be written as:

$$ {}^{j}\mathbb{J}_{j}\, {}^{j}\dot{\mathbb{V}}_{j} = {}^{j}\mathbb{F}_{j} - {}^{k}\mathbb{T}_{j}^{T}\, {}^{k}\mathbb{F}_{k} - {}^{j}\mathbb{F}_{tj} - \begin{bmatrix} {}^{j}\omega_{j} \text{ x } ({}^{j}\omega_{j} \text{ x } {}^{j}\mathbf{MS}_{j}) \\ {}^{j}\omega_{j} \text{ x } ({}^{j}\mathbf{J}_{j}\, {}^{j}\omega_{j}) \end{bmatrix} \tag{A4-13} $$

with j = a(k), it will be equal to j+1 in the case of simple open loop robots, in the case of tree structure robot we have to do the sum on k.

Equation (A4-13) can be rewritten as:

$$ {}^{j}\mathbb{J}_{j}\, {}^{j}\dot{\mathbb{V}}_{j} = {}^{j}\mathbb{F}_{j} - {}^{k}\mathbb{T}_{j}^{T}\, {}^{k}\mathbb{F}_{k} + {}^{j}\boldsymbol{\beta}_{j} \tag{A4-14} $$

with:

$$ {}^{j}\boldsymbol{\beta}_{j} = - {}^{j}\mathbb{F}_{tj} - \begin{bmatrix} {}^{j}\omega_{j} \text{ x } ({}^{j}\omega_{j} \text{ x } {}^{j}\mathbf{MS}_{j}) \\ {}^{j}\omega_{j} \text{ x } ({}^{j}\mathbf{J}_{j}\, {}^{j}\omega_{j}) \end{bmatrix} \tag{A3-15} $$

Equations (A4-14) can be written for j = n as:

$$ {}^{n}\mathbb{J}_{n} \left\{ {}^{n}\mathbb{T}_{n-1}\, {}^{n-1}\dot{\mathbb{V}}_{n-1} + \ddot{q}_{n}\, \mathbb{a}_{n} + {}^{n}\gamma_{n} \right\} = {}^{n}\mathbb{F}_{n} + {}^{n}\boldsymbol{\beta}_{n} \tag{A4-16} $$

with:

$$ {}^{n}\boldsymbol{\beta}_{n} = - {}^{n}\mathbb{F}_{tn} - \begin{bmatrix} {}^{n}\omega_{n} \text{ x } ({}^{n}\omega_{n} \text{ x } {}^{n}\mathbf{MS}_{n}) \\ {}^{n}\omega_{n} \text{ x } ({}^{n}\mathbf{J}_{n}\, {}^{n}\omega_{n}) \end{bmatrix} $$

Multiplying the previous equation by $\mathbb{a}_n^T$ and noting that $\mathbb{a}_n^T\, {}^{n}\mathbb{F}_n^r = 0$, we obtain:

$$\ddot{q}_n = H_n^{-1} \{ - \mathbb{a}_n^T\, {}^{n}\mathbb{J}_n\, ({}^{n}\mathbb{T}_{n-1}\, {}^{n-1}\dot{\mathbb{V}}_{n-1} + {}^{n}\gamma_n) + \tau_n + \mathbb{a}_n^T\, {}^{n}\beta_n \} \tag{A4-17}$$

with $H_n = (\mathbb{a}_n^T\, {}^{n}\mathbb{J}_n\, \mathbb{a}_n + Ia_n)$

Substituting for $\ddot{q}_n$ from equation (A4-17) into (A4-16), we obtain:

$${}^{n}\mathbb{J}_n\, {}^{n}\mathbb{T}_{n-1}\, {}^{n-1}\dot{\mathbb{V}}_{n-1} + {}^{n}\mathbb{J}_n\, \mathbb{a}_n\, H_n^{-1} [- \mathbb{a}_n^T\, {}^{n}\mathbb{J}_n\, ({}^{n}\mathbb{T}_{n-1}\, {}^{n-1}\dot{\mathbb{V}}_{n-1} + {}^{n}\gamma_n) + \tau_n + \mathbb{a}_n^T\, {}^{n}\beta_n] + {}^{n}\mathbb{J}_n\, {}^{n}\gamma_n$$

$$= {}^{n}\mathbb{F}_n + {}^{n}\beta_n$$

Thus ${}^{n}\mathbb{F}_n$ can be obtained as function of ${}^{n-1}\dot{\mathbb{V}}_{n-1}$ as follows:

$${}^{n}\mathbb{F}_n = \begin{bmatrix} {}^{n}\mathbf{f}_n \\ {}^{n}\mathbf{n}_n \end{bmatrix} = {}^{n}\mathbf{K}_n\, {}^{n}\mathbb{T}_{n-1}\, {}^{n-1}\dot{\mathbb{V}}_{n-1} + {}^{n}\alpha_n \tag{A4-18}$$

with:

$${}^{n}\mathbf{K}_n = {}^{n}\mathbb{J}_n - {}^{n}\mathbb{J}_n\, \mathbb{a}_n\, H_n^{-1}\, \mathbb{a}_n^T\, {}^{n}\mathbb{J}_n \tag{A4-19}$$

$${}^{n}\alpha_n = {}^{n}\mathbf{K}_n\, {}^{n}\gamma_n + {}^{n}\mathbb{J}_n\, \mathbb{a}_n\, H_n^{-1} (\tau_n + \mathbb{a}_n^T\, {}^{n}\beta_n) - {}^{n}\beta_n \tag{A4-20}$$

Similarly for $j = n-1$, we obtain:

$${}^{n-1}\mathbb{J}_{n-1}\, {}^{n-1}\dot{\mathbb{V}}_{n-1} = {}^{n-1}\mathbb{F}_{n-1} + {}^{n}\mathbb{T}^T_{n-1}\, {}^{n}\mathbb{F}_n + {}^{n-1}\beta_{n-1}$$

with:

$${}^{n-1}\beta_{n-1} = - {}^{n-1}\mathbb{F}_{tn-1} - \begin{bmatrix} {}^{n-1}\omega_{n-1} \times ({}^{n-1}\omega_{n-1} \times {}^{n-1}\mathbf{MS}_{n-1}) \\ {}^{n-1}\omega_{n-1} \times ({}^{n-1}\mathbf{J}_{n-1}\, {}^{n-1}\omega_{n-1}) \end{bmatrix} \tag{A4-21}$$

which can be rewritten as:

$${}^{n-1}\mathbb{J}^*_{n-1} \{ {}^{n-1}\mathbb{T}_{n-2}\, {}^{n-2}\dot{\mathbb{V}}_{n-2} + \ddot{q}_{n-1}\, \mathbb{a}_{n-1} + {}^{n-1}\gamma_{n-1} \} = {}^{n-1}\mathbb{F}_{n-1} + {}^{n-1}\boldsymbol{\beta}^*_{n-1} \tag{A4-22}$$

with:

$${}^{n-1}\mathbb{J}^*_{n-1} = {}^{n-1}\mathbb{J}_{n-1} + {}^{n}\mathbb{T}^T_{n-1}\, {}^{n}\mathbf{K}_n\, {}^{n}\mathbb{T}_{n-1} \tag{A4-23}$$

$${}^{n-1}\boldsymbol{\beta}^*_{n-1} = {}^{n-1}\beta_{n-1} - {}^{n}\mathbb{T}^T_{n-1}\, {}^{n}\alpha_n \tag{A4-24}$$

Equation (A4-22) is similar to (A4-16), thus $\ddot{q}_{n-1}$ and ${}^{n-1}\mathbb{F}_{n-1}$ can be obtained in terms of ${}^{n-2}\dot{\mathbb{V}}_{n-2}$.

This procedure can be repeated for n-2,...,1 , in order to obtain at the end $\ddot{q}_1$ and ${}^0\mathbb{F}_0$ in terms of ${}^0\dot{\mathbb{V}}_0$, which is known (for fixed base we have: $\dot{\omega}_0 = \mathbf{0}$ and $\dot{\mathbf{V}}_0 = -\mathbf{g}$), thus $\ddot{q}_1$ will be determined.

The accelerations $\ddot{q}_2,...,\ddot{q}_n$ and ${}^j\mathbb{F}_j$ can be obtained by a forward recursive calculation for j = 2,…,n , by using the following relations:

$$ {}^j\mathbf{R}_j = {}^j\mathbb{T}_i\, {}^i\dot{\mathbb{V}}_i + {}^j\gamma_j $$

$$ {}^j\mathbf{v}_j = {}^j\mathbb{J}_j^* \,.\, \mathbb{a}_j $$

$$ D_j = [\, \mathbb{a}_j^T \,.\, {}^j\mathbf{v}_j + Ia_j \,]^{-1} $$

$$ w_j = \mathbb{a}_j^T\, \beta_j^* + \tau_j $$

$$ \ddot{q}_j = D_j\, (w_j - {}^j\mathbf{v}_j^T\, {}^j\mathbf{R}_j) $$

$$ {}^j\dot{\mathbb{V}}_j = {}^j\mathbf{R}_j + \mathbb{a}_j\, \ddot{q}_j $$

$$ {}^j\mathbb{F}_j = {}^j\mathbb{J}_j^*\, {}^j\dot{\mathbb{V}}_j - \beta_j^* $$

A 4.4.4. Notations used for the intermediate variables

The following intermediate variables are used:

${}^j\omega_i = [\mathrm{WI1j} \quad \mathrm{WI2j} \quad \mathrm{WI3j}]^T$	calculated from relation (3.46)
${}^j\omega_j = [\mathrm{W1j} \quad \mathrm{W2j} \quad \mathrm{W3j}]^T$	calculated from relation (3.47)

$$ {}^j\mathbf{A}_i\, {}^i\hat{\mathbf{P}}_j = \mathrm{JPRJ} $$

$$ {}^j\mathbb{T}_i = \begin{bmatrix} {}^j\mathbf{A}_i & -{}^j\mathbf{A}_i\, {}^i\hat{\mathbf{P}}_j \\ \mathbf{0}_3 & {}^j\mathbf{A}_i \end{bmatrix} $$

$$ {}^{a(j)}\mathbf{A}_j = \begin{bmatrix} \mathrm{A11j} & \mathrm{A12j} & \mathrm{A13j} \\ \mathrm{A21j} & \mathrm{A22j} & \mathrm{A23j} \\ \mathrm{A31j} & \mathrm{A32j} & \mathrm{A33j} \end{bmatrix} $$

${}^{j}\omega_j \times ({}^{j}\omega_j \times {}^{j}\mathbf{MS}_j)$ is denoted SWj, $({}^{j}\mathbf{J}_j\ {}^{j}\omega_j)$ is denoted JWj

${}^{j}\omega_j \times ({}^{j}\mathbf{J}_j\ {}^{j}\omega_j)$ is denoted KWj

${}^{j}\mathbf{A}_i\ ({}^{i}\omega_i \times ({}^{i}\omega_i \times {}^{i}\mathbf{P}_j))$ is denoted LWj

${}^{j}\mathbf{K}_j = {}^{j}\mathbb{J}_j^{*} - {}^{j}\mathbf{U}_j\ {}^{j}\mathbf{v}_j^{T}$ will be denoted GKj which is a 6x6 matrix

${}^{j}\mathbf{K}_j\ {}^{j}\gamma_j + {}^{j}\mathbf{U}_j\ w_j$ will be denoted VSj

${}^{j}\alpha_j$ will be denoted as APj = [AP1j … AP6j]

${}^{i}\mathbb{T}_j\ {}^{j}\mathbf{K}_j\ {}^{j}\mathbb{T}_i$ = TKTj (the general element (a,b) is denoted by TKTabj)

${}^{j}\mathbb{J}^{*}{}_j$ is denoted MJEj which is a 6x6 symmetric matrix, with the element (a,b) denoted by MJEabj

$\boldsymbol{\beta}^{*}{}_i$ will be denoted VBEj which is a (6x1) vector

${}^{j}\mathbf{R}_j$ is denoted VRj

$\ddot{q}_j$ is denoted QDPj

${}^{j}\dot{\omega}_j = [\text{WP1j} \quad \text{WP2j} \quad \text{WP3j}]^{T}$

${}^{j}\dot{\mathbf{V}}_j = [\text{VP1j} \quad \text{VP2j} \quad \text{VP3j}]^{T}$

${}^{j}\mathbf{f}_j = [\text{E1j} \quad \text{E2j} \quad \text{E3j}]^{T}$

${}^{j}\mathbf{n}_j = [\text{N1j} \quad \text{N2j} \quad \text{N3j}]^{T}$

Appendix 4.5 : Calculation of the filtered dynamic model of robots

This appendix presents the computation of the filtered dynamic model of robots, which is used in the adaptive control of robots and in the identification of its inertial parameters. The main advantage of this model is that it is not a function of the joint acceleration. For more details about the proposed method the reader can consult [Khalil 96, Restrepo 96].

Filtered dynamic model

The dynamics of a rigid robot with n joints can be described as:

$$\mathbf{\Gamma} = \mathbf{A(q)}\,\ddot{\mathbf{q}} + \mathbf{H}(\mathbf{q},\dot{\mathbf{q}}) \tag{A5-1}$$

where:

- $\mathbf{q}$ is the (nx1) joint positions vector,
- $\mathbf{\Gamma}$ is the (nx1) joint torques vector,
- $\mathbf{A}$ is the (nxn) robot inertia matrix that is symmetric and positive definite,
- $\mathbf{H}$ is the vector of centrifuge, Coriolis and gravitational forces or torques.

Previous works have proposed to obtain the filtered dynamic model from relation (A5-1) [Aubin 91, Middleton 86, Slotine 87], this procedure is very complicated and contains many redundant calculations. We propose to directly use the Lagrangian equation, which is given by:

$$\mathbf{\Gamma} = \frac{d}{dt}\frac{\partial L}{\partial \dot{\mathbf{q}}} - \frac{\partial L}{\partial \mathbf{q}} \tag{A5-2}$$

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.

Thus equation (A5-2), can be written as:

$$\mathbf{\Gamma} = \frac{d}{dt}\left(\frac{\partial E}{\partial \dot{\mathbf{q}}}\right) - \frac{\partial E}{\partial \mathbf{q}} + \frac{\partial U}{\partial \mathbf{q}} \tag{A5-3}$$

We obtain that:

$$\mathbf{\Gamma} = \left[\frac{d}{dt}\mathbf{D_1}(\mathbf{q},\dot{\mathbf{q}}) + \mathbf{D_2}(\mathbf{q},\dot{\mathbf{q}})\right]\mathbf{K} \tag{A5-4}$$

where **K** is the inertial parameters of all the links.

$$\frac{\partial E}{\partial \dot{\mathbf{q}}} = \mathbf{D_1}(\mathbf{q},\dot{\mathbf{q}})\,\mathbf{K}, \text{ and}$$

$$\mathbf{D_2}(\mathbf{q},\dot{\mathbf{q}})\,\mathbf{K} = -\frac{\partial E}{\partial \mathbf{q}} + \frac{\partial U}{\partial \mathbf{q}}$$

where $\mathbf{D_1}$ and $\mathbf{D_2}$ are of dimension (nx10n).

Applying a numerical filter, denoted **F**, of first or second order, on equation (A5-4) gives:

$$\Gamma_{\mathbf{f}} = \mathbf{D_{1df}}(\mathbf{q},\dot{\mathbf{q}})\,\mathbf{K} + \mathbf{D_{2f}}(\mathbf{q},\dot{\mathbf{q}})\,\mathbf{K} \qquad \text{(A5-6)}$$

where the subscribt "f" means the application of the original filter F, and "df" means the application of a filter F ' which associate to the filter F a derivative action. For example the use of a second order filter, the transfer functions F and F ' could be taken as:

$$F(s) = \frac{a2}{(s+a)2} \quad , \quad F'(s) = \frac{a^2 s}{(s+a)^2} \qquad \text{(A5-7)}$$

Equation (A5-6) defines the filtered dynamic model.

The calculation of the columns of $\mathbf{D_1}$ and $\mathbf{D_2}$ correspond to the calculation of the coefficients of each dynamic parameters in the following three terms:

$$\frac{\partial E}{\partial \dot{\mathbf{q}}},\ \frac{\partial E}{\partial \mathbf{q}},\ \frac{\partial U}{\partial \mathbf{q}}$$

We will start by presenting the calculation of the second term, which contains many intermediate variables that will be used in the other terms.

Calculation of $\frac{\partial \mathbf{E}}{\partial \mathbf{q}}$

The kinetic energy can be written as:

$$E = \mathbf{e}(\mathbf{q},\dot{\mathbf{q}})\,\mathbf{K} = \sum_{j=1}^{Np} e_j K_j \qquad \text{(A5-8)}$$

with Np the number of inertial parameters, K_j is a general dynamic parameter, e_j the coefficient of K_j in the kinetic energy.

The kinetic energy of a link j can be written as:

$$E_j = \mathbf{e_j}\,\mathbf{K}_j \qquad \text{(A5-9)}$$

with:

$$\mathbf{e_j} = [e_{XXj}\ \ e_{XYj}\ \ e_{XZj}\ \ e_{YYj}\ \ e_{YZj}\ \ e_{ZZj}\ \ e_{MXj}\ \ e_{MYj}\ \ e_{MZj}\ \ e_{Mj}] \qquad \text{(A5-10)}$$

Since E_j is given as:

$$E_j = \frac{1}{2}\left[{}^j\omega_j^T\, {}^j\mathbf{J}_j\, {}^j\omega_j + M_j\, {}^j\mathbf{V}_j^T\, {}^j\mathbf{V}_j + 2\, {}^j\mathbf{MS}_j^T\, ({}^j\mathbf{V}_j \times {}^j\omega_j)\right] \qquad \text{(A5-11)}$$

where, ${}^j\omega_j$ is the rotational velocity of link j and ${}^j\mathbf{V}_j$ represents the velocity of the origin of frame j, both referred to frame j. They will be represented by the vectors:

$$\begin{aligned} {}^j\omega_j &= [\omega_{1,j} \quad \omega_{2,j} \quad \omega_{3,j}]^T \\ {}^j\mathbf{V}_j &= [V_{1,j} \quad V_{2,j} \quad V_{3,j}]^T \end{aligned}$$

We can obtain the coefficients of the inertial parameters $XX_j,\ldots,M_j$, in the kinetic energy as:

$e_{XXj} = \frac{1}{2}\, \omega_{1,j}\, \omega_{1,j}$, $\qquad e_{XYj} = \omega_{1,j}\, \omega_{2,j}$

$e_{XZj} = \omega_{1,j}\, \omega_{3,j}$, $\qquad e_{YYj} = \frac{1}{2}\, \omega_{2,j}\, \omega_{2,j}$

$e_{YZj} = \omega_{2,j}\, \omega_{3,j}$, $\qquad e_{ZZj} = \frac{1}{2}\, \omega_{3,j}\, \omega_{3,j}$

$e_{MXj} = \omega_{3,j}\, V_{2,j} - \omega_{2,j}\, V_{3,j}$

$e_{MYj} = \omega_{1,j}\, V_{3,j} - \omega_{3,j}\, V_{1,j}$

$e_{MZj} = \omega_{2,j}\, V_{1,j} - \omega_{1,j}\, V_{2,j}$

$e_{Mj} = \frac{1}{2}\, {}^j\mathbf{V}_j^T\, {}^j\mathbf{V}_j$

Finally, the coefficients of $\frac{\partial E}{\partial \mathbf{q}}$, means to calculate the derivative of $e_{XXj},\ldots,e_{Mj}$:

$$\frac{\partial e_{XXj}}{\partial q_i} = \frac{\partial \omega_{1,j}}{\partial q_i}\omega_{1,j} \quad , \qquad \frac{\partial e_{XYj}}{\partial q_i} = \frac{\partial \omega_{1,j}}{\partial q_i}\omega_{2,j} + \omega_{1,j}\frac{\partial \omega_{2,j}}{\partial q_i}$$

$$\frac{\partial e_{XZj}}{\partial q_i} = \frac{\partial \omega_{1,j}}{\partial q_i}\omega_{3,j} + \omega_{1,j}\frac{\partial \omega_{3,j}}{\partial q_i} \quad , \qquad \frac{\partial e_{YYj}}{\partial q_i} = \frac{\partial \omega_{2,j}}{\partial q_i}\omega_{2,j}$$

$$\frac{\partial e_{YZj}}{\partial q_i} = \frac{\partial \omega_{2,j}}{\partial q_i}\omega_{3,j} + \omega_{2,j}\frac{\partial \omega_{3,j}}{\partial q_i} \quad , \qquad \frac{\partial e_{ZZj}}{\partial q_i} = \frac{\partial \omega_{3,j}}{\partial q_i}\omega_{3,j}$$

$$\frac{\partial e_{MXj}}{\partial q_i} = \frac{\partial \omega_{3,j}}{\partial q_i}V_{2,j} + \omega_{3,j}\frac{\partial V_{2,j}}{\partial q_i} - \frac{\partial \omega_{2,j}}{\partial q_i}V_{3,j} + \omega_{2,j}\frac{\partial V_{3,j}}{\partial q_i}$$

$$\frac{\partial e_{MYj}}{\partial q_i} = \frac{\partial \omega_{1,j}}{\partial q_i}V_{3,j} + \omega_{1,j}\frac{\partial V_{3,j}}{\partial q_i} - \frac{\partial \omega_{3,j}}{\partial q_i}V_{1,j} + \omega_{3,j}\frac{\partial V_{1,j}}{\partial q_i}$$

$$\frac{\partial e_{MZj}}{\partial q_i} = \frac{\partial \omega_{2,j}}{\partial q_i}V_{1,j} + \omega_{2,j}\frac{\partial V_{1,j}}{\partial q_i} - \frac{\partial \omega_{1,j}}{\partial q_i}V_{2,j} + \omega_{2,j}\frac{\partial V_{1,j}}{\partial q_i}$$

$$\frac{\partial e_{Mj}}{\partial q_i} = {}^j\mathbf{V}_j^T\frac{\partial\, {}^j\mathbf{V}_j}{\partial q_i}$$

These expressions need to calculate ${}^j\omega_j$ and ${}^j\mathbf{V}_j$ and its partial derivative with respect to q_i.

Calculation of $\dfrac{\partial\, {}^j\mathbf{V}_j}{\partial\, q_i}$ and $\dfrac{\partial\, {}^j\omega_j}{\partial\, q_i}$

The velocities ${}^j\mathbf{V}_j$ and ${}^j\omega_j$ can be obtained by:

$$ {}^j\mathcal{V}_j = \sum_{k=1}^{j} [\sigma_k\, {}^j\mathbf{a}_k - \bar{\sigma}_k({}^j\mathbf{a}_k \times {}^j\mathbf{P}_k)]\dot{q}_k \qquad , \qquad {}^j\omega_j = \sum_{k=1}^{j} \bar{\sigma}_k\, \mathbf{a}_k\, \dot{q}_k $$

where:

${}^j\mathbf{a}_k$ is the unit vector along the axis of joint k, or $\mathbf{z}_k$, projected into frame j,

${}^j\mathbf{P}k$ is the position vector $O_j\, O_k$, with O_j the origin of frame j, projected into frame j.

We obtain:

$$ \frac{\partial\, {}^j\mathbf{V}_j}{\partial\, q_i} = \sum_{k=1}^{j} \dot{q}_k \left\{\sigma_k \frac{\partial\, {}^j\mathbf{a}_k}{\partial\, q_i} - \bar{\sigma}_k\left(\frac{\partial\, {}^j\mathbf{a}_k}{\partial\, q_i} \times {}^j\mathbf{P}_k\right) - \bar{\sigma}_k\left({}^j\mathbf{a}_k \times \frac{\partial\, {}^j\mathbf{P}_k}{\partial\, q_i}\right)\right\} $$

$$ \frac{\partial\, {}^j\omega_j}{\partial\, q_i} = \sum_{k=1}^{j} \bar{\sigma}_k \frac{\partial\, {}^j\mathbf{a}_k}{\partial\, q_i}\dot{q}_k $$

In order to calculate $\dfrac{\partial\, {}^j\mathbf{a}_k}{\partial\, q_i}$ and $\dfrac{\partial\, {}^j\mathbf{P}_k}{\partial\, q_i}$, three cases are considered:

i- if $k < i \leq j$: It can be proved that:

$$ \frac{\partial\, {}^j\mathbf{a}_k}{\partial\, q_i} = -\bar{\sigma}_i\, ({}^j\mathbf{a}_i \times {}^j\mathbf{a}_k) $$

$$ \frac{\partial\, {}^j\mathbf{P}_k}{\partial\, q_i} = -\bar{\sigma}_i\, ({}^j\mathbf{a}_i \times {}^j\mathbf{P}_k - {}^j\mathbf{a}_i \times {}^j\mathbf{P}_i) - \sigma_i\, {}^j\mathbf{a}_i $$

thus,

$$ \frac{\partial\, {}^j\mathbf{V}_j}{\partial\, q_i} = \sum_{k=1}^{i-1} \dot{q}_k \{(\sigma_k\bar{\sigma}_i + \sigma_i\bar{\sigma}_k)\, ({}^j\mathbf{a}_i \times {}^j\mathbf{a}_k) - \bar{\sigma}_k\bar{\sigma}_i\, (({}^j\mathbf{a}_k \times {}^j\mathbf{P}_k) \times {}^j\mathbf{a}_i + {}^j\mathbf{a}_k \times ({}^j\mathbf{a}_i \times {}^j\mathbf{P}_i))\} $$

$$ \frac{\partial\, {}^j\omega_j}{\partial\, q_i} = \sum_{k=1}^{i-1} \bar{\sigma}_k\bar{\sigma}_i\, ({}^j\mathbf{a}_i \times {}^j\mathbf{a}_k)\, \dot{q}_k $$

ii- if $i \leq k \leq j$: It can be seen that:

$$ \frac{\partial\, {}^j\mathbf{a}_k}{\partial\, q_i} = \mathbf{0}_{3\text{x}1} \quad , \quad \frac{\partial\, {}^j\mathbf{P}_k}{\partial\, q_i} = \mathbf{0}_{3\text{x}1} $$

with $\mathbf{0}_{3\text{x}1}$ is the (3x1) zero vector.

which gives:

$$ \frac{\partial\, {}^j\mathbf{V}_j}{\partial\, q_i} = \mathbf{0}_{3\text{x}1} \quad , \quad \frac{\partial\, {}^j\omega_j}{\partial\, q_i} = \mathbf{0}_{3\text{x}1} $$

iii- if $k \le j < i$: It is easy to see that:

$$\frac{\partial\, {}^j\mathbf{V}_j}{\partial q_i} = \mathbf{0}_{3x1} \quad , \quad \frac{\partial\, {}^j\omega_j}{\partial q_i} = \mathbf{0}_{3x1}$$

Calculation of $\frac{\partial \mathbf{E}}{\partial \dot{\mathbf{q}}}$

The column corresponding to an inertial parameter in **D1,** can be calculated by differentiating the coefficient of this parameter in the kinetic energy with respect to the joint velocities. *We* deduce that:

$$\frac{\partial\, {}^j\mathbf{V}_j}{\partial \dot{q}_i} = \sigma_i {}^j\mathbf{a}_i - \bar{\sigma}_i({}^j\mathbf{a}_i \times {}^j\mathbf{P}_i) \quad , \text{ for } i \le j$$

$$\frac{\partial\, {}^j\mathbf{V}_j}{\partial \dot{q}_i} = \mathbf{0}_{3x1} \quad , \text{ for } i > j$$

$$\frac{\partial\, {}^j\omega_j}{\partial \dot{q}_i} = \bar{\sigma}_i {}^j\mathbf{a}_i \quad , \text{ for } i \le j$$

$$\frac{\partial\, {}^j\omega_j}{\partial \dot{q}_i} = \mathbf{0}_{3x1} \quad , \text{ for } i > j$$

Calculation of $\frac{\partial \mathbf{U}}{\partial \mathbf{q}}$

The potential energy of link j is given by:

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

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 with respect to frame zero.
- ${}^j\mathbf{MS}_j$ is the vector of the first moments of link j.
- $0\mathbf{A}j$ is the (3x3) transformation matrix giving the orientation of frame j with respect to frame 0, such that:

  $0\mathbf{A}j = [{}^0\mathbf{s}_j \quad {}^0\mathbf{n}_j \quad {}^0\mathbf{a}_j]$, with $\mathbf{s}_j$, $\mathbf{n}_j$, $\mathbf{a}_j$ are the unit vectors along the x, y, and z axis respectively.

Since the potential energy is linear relation in the inertial parameters, we can write:

$$U = \mathbf{u(q)}\, \mathbf{K} = \sum_{j=1}^{Np} u_j K_j$$

Thus:

$$u_{XXj} = u_{XYj} = \ldots = u_{ZZj} = 0$$
$$u_{MXj} = -\,{}^{0}\mathbf{g}^{T}\,{}^{0}\mathbf{s}_j$$
$$u_{MYj} = -\,{}^{0}\mathbf{g}^{T}\,{}^{0}\mathbf{n}_j$$
$$u_{MZj} = -\,{}^{0}\mathbf{g}^{T}\,{}^{0}\mathbf{a}_j$$
$$u_{Mj} = -\,{}^{0}\mathbf{g}^{T}\,{}^{0}\mathbf{P}_j$$

Thus $\frac{\partial U}{\partial q}$ can be obtained using the following relations [Khalil 96]:

$$\frac{\partial\, {}^{j}\mathbf{s}_j}{\partial q_i} = \bar{\sigma}_i\,({}^{0}\mathbf{a}_i \text{ x } {}^{0}\mathbf{s}_j)$$
$$\frac{\partial\, {}^{j}\mathbf{n}_j}{\partial q_i} = \bar{\sigma}_i\,({}^{0}\mathbf{a}_i \text{ x } {}^{0}\mathbf{n}_j)$$
$$\frac{\partial\, {}^{j}\mathbf{a}_j}{\partial q_i} = \bar{\sigma}_i\,({}^{0}\mathbf{a}_i \text{ x } {}^{0}\mathbf{a}_j)$$
$$\frac{\partial\, {}^{0}\mathbf{P}_j}{\partial q_i} = \bar{\sigma}_i\,({}^{0}\mathbf{a}_i \text{ x } {}^{0}\mathbf{P}_j - {}^{0}\mathbf{a}_i \text{ x } {}^{0}\mathbf{P}_i) + \sigma_i\,{}^{0}\mathbf{a}_i$$

Intermediate variables

SYMORO+ calculates this algorithm using customized symbolic calculation, it generates the elements of **D1** and **D2.** The i^{th} components (corresponding to joint i) of the vector corresponding to the inertial parameter Kj are denoted as DEPiKj and DLQiKj respectively.

Computational cost

Using the minimum inertial parameters (base inertial parameters), it can be seen that the maximum number of operations in the general case for n degrees of freedom robot are ($3n^3 + 12n^2 - 27n + 9$) additions and ($4n^3 + 23n^2 - 16n - 12$) multiplications.

For an industrial robot (whose many geometric parameters are zero, for the distances, or equal to k $\pi/2$ (with k integer) for the angles), the number of operations will be reduced considerably. For example, in the case of the 6 d.o.f Puma robot the filtered model needs 635 additions and 720 multiplications, while in the case of the 6 d.o.f. Stanford manipulator the filtered model needs 524 additions and 612 multiplications.

PART II

INSTALLATION AND USE OF SYMORO+

Chapter 5

Installation and Use of SYMORO+

5.1. Introduction

The software package SYMORO+ “SYmbolic MOdelling of Robots”, is an interactive software package for the automatic generation of symbolic models of robots. It provides the user with rapid, sure, and efficient tools to generate the different models needed in the control, simulation and design of robots (direct and inverse geometric models, direct and inverse kinematic models, dynamic models, identification models,...).

The first version of SYMORO was announced on 1986 for PC computers and was written in FORTRAN. The current version, SYMORO+, is available for PC computers running Windows® operating system. It is written in C++ and makes use of the Mathematica® system, from *Wolfram Research Inc.*, which must be installed on your computer to run SYMORO+.

This chapter contains the installation instructions, and presents the different menus and functions of SYMORO+. It is supposed that the user is familiar with the method of description of robots, chapter 1, and at least the definition and the application of the different models described in chapters 2, 3, and 4 of this guide.

An on-line help describing the functions of the different menus is also available.

5.2. Instructions for installation on PC Computers

5.2.1. System requirements

Screen resolution: This program requires 1024x768 or better resolution.
Operating system: Microsoft Windows® (98/Me/NT4/2000/XP/Vista).
Mathematica® from *Wolfram Research Inc.* version 3.0.1 or higher must be installed on your computer to run this program. The installed software must at least contain the following components: Kernel, MathLink Libraries and Standard Add-on Packages.

5.2.2. Disk space requirements

5 Mb of free space are required on the hard disk to install SYMORO+ and the SENTINEL DRIVER (this size does not include the Mathematica® package).

5.2.3. Installation of the SYMORO+ package

IMPORTANT: Your protected application has to communicate with the provided USB key to run correctly. So you must install the Sentinel Keys Driver provided on the distributed disk and connect the key to a USB port of your computer before running the SYMORO+ application. Please edit the "Readme.txt" file in the "SENTINEL" directory of the distributed disk to get useful information about the driver install process.

SYMORO+ is provided with a setup program. During the installation a directory “Math” which contains the Mathematica® programs and a directory “Robots” which contains the parameters of some known robots are copied to the hard disk.

To install the SYMORO+ package on Microsoft Windows® NT4/2000/XP/Vista systems, you must have administrator privileges.

It is recommended to cancel a previously installed version of SYMORO+ by using the “uninstall” procedure that was provided with it, in order to clean the Windows® registry.

Be aware that when installing the software in the same directory as an older version, all the files in the underlying directory tree "<OS drive>:\Program Files\IRCCyN\Symoro+\..." will be deleted by the setup program. If you want to preserve some files in the "…\Symoro+\Robots\…" directories you must make a copy of them.

To start the installation process run the file “setup.exe” on the distributed disk in the directory named “SYMORO” and follow the instructions on the screen. If the autorun is activated for your CD-ROM drive, the install process starts automatically when the disk is inserted.

When setup is complete, you may run the installed program by selecting the SYMORO+ icon in the program group.

Note that Mathematica® from *Wolfram Research Inc.* version 3.0.1 or higher must be installed on your computer to run this program. The list of the components to be installed is given in §"5.2.1. System requirements".

This program is not intended to run on a network configuration, so SYMORO+ and the SENTINEL package and its protection key must be installed on the computer that will be used by the final users.

Known problem with Mathematica® release 4.0 or higher: Sometime the installation of the Mathematica® package does not update the following registry key:

“HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\App Paths\MathKernel.exe”

So this error message appears on the screen when you start SYMORO+: “Mathematica references are not correct in the Windows Registry. Try to reinstall the Mathematica package”.

To solve this problem you must update the Windows® registry with the key above. To help you we provide a file named “MathKey.reg” located in the "SYMORO" directory of the distributed disk and also in the directory of the installed SYMORO+ software package. Edit this file and set the correct access path for the Mathematica® program file “MathKernel.exe”. Then just double-click the updated "MathKey.reg" file or drag and drop this file on the opened registry editor. (to open the registry editor execute the command “regedit”)

<u>IMPORTANT</u>: Upgrading from a version prior to 1.1

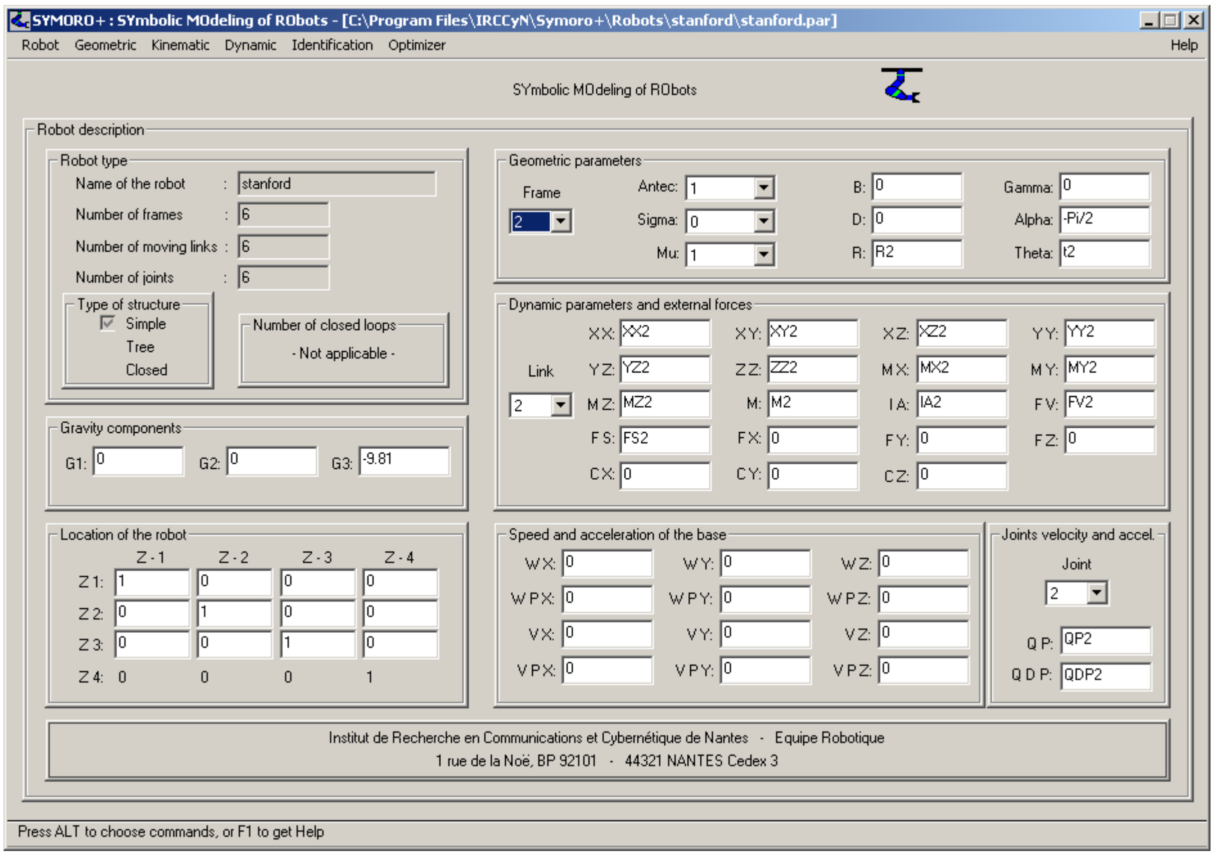
The *.ini files (located in the robot directories and containing some customized parameters) generated by a version prior to 1.1 are no longer readable by SYMORO+.

We recommend to delete these *.ini files in order to let SYMORO+ be able to load the corresponding *.par files (containing the robot definition). The specification of the points used to get the “Inverse Geometric Model” using the “General Numerical Method” will be lost and replaced by the default values in the new *.ini created files.

5.3. Using SYMORO+

The software package SYMORO+ is composed of:

- a set of programs, written in Mathematica, to generate the different models of robots,
- a graphical interface program, written in C++, such that the user can define his robot or activate the desired function using friendly interface. When starting SYMORO+, the main interface page, figure 5.1, will appear.


SYMORO+ : SYmbolic MOdeling of RObots - [C:\Program Files\IRCCyN\Symoro+\Robots\stanford\stanford.par]
Robot
Geometric
Kinematic
Dynamic
Identification
Optimizer
Help
SYmbolic MOdeling of RObots

Robot description

Robot type

Name of the robot :	stanford
Number of frames :	6
Number of moving links :	6
Number of joints :	6

Type of structure: [✓ Simple] Tree Closed

Number of closed loops: - Not applicable -

Gravity components

G1: 0 G2: 0 G3: -9.81

Location of the robot

	Z - 1	Z - 2	Z - 3	Z - 4
Z 1:	1	0	0	0
Z 2:	0	1	0	0
Z 3:	0	0	1	0
Z 4:	0	0	0	1

Geometric parameters

Frame: 2

Antec:	1	B:	0	Gamma:	0
Sigma:	0	D:	0	Alpha:	-Pi/2
Mu:	1	R:	R2	Theta:	t2

Dynamic parameters and external forces

Link: 2

XX:	XX2	XY:	XY2	XZ:	XZ2	YY:	YY2
YZ:	YZ2	ZZ:	ZZ2	MX:	MX2	MY:	MY2
MZ:	MZ2	M:	M2	IA:	IA2	FV:	FV2
FS:	FS2	FX:	0	FY:	0	FZ:	0
CX:	0	CY:	0	CZ:	0		

Speed and acceleration of the base

WX:	0	WY:	0	WZ:	0
WPX:	0	WPY:	0	WPZ:	0
VX:	0	VY:	0	VZ:	0
VPX:	0	VPY:	0	VPZ:	0

Joints velocity and accel.

Joint: 2

QP: QP2

QDP: QDP2

Institut de Recherche en Communications et Cybernétique de Nantes - Equipe Robotique
1 rue de la Noë, BP 92101 - 44321 NANTES Cedex 3

Press ALT to choose commands, or F1 to get Help

Figure 5.1. : The main interface page of SYMORO+

The following menus are given:

Robot, Geometric, Kinematic, Dynamic, Identification, Optimizer

The following functions are available as submenus:

Robot
New …
Open …
Save
Save as ...
Preferences ...
MRU file list
Exit

Geometric
Transformation matrix ...
Fast geometric model ...
I.G.M. Pieper method
I.G.M. General method (symbolic or numeric) ...
Transformation matrices for I.G.M.
I.G.M. Paul method ...
Constraint geometric equations of loops

Kinematic
Jacobian matrix …
Determinant of a Jacobian …
Inverse of a Jacobian …
Velocity of links
Acceleration of links
Calculation of JpQp
Constraint kinematic equations of loops
Resolution of constraint equations of loops

Dynamic
Inverse dynamic model
Lagrange method (customized or expanded)
Inertia matrix
Centrifugal, Coriolis & Gravity torques
Direct dynamic model

Identification
Base inertial parameters (symbolic or numeric)
Energy identification model
Dynamic identification model
Filtered dynamic identification model

Optimizer
Optimizer …

Remark:
The output of a function is given "by default" in the file:

robot-name/robot-name.extension

where "robot-name/" is the directory of the current opened robot description file, and "robot-name.extension" is the name of the file to be created. The user can change the directory and the name of the destination file excluding the extension part, which is appended automatically by SYMORO+. The file extensions generated by SYMORO+ are given in Appendix 5.1 .

5.4. Menu Robot

The aim of this menu is to define new robots or to open the description file of a previously defined robot. The user must be familiar with the description of the geometric parameters given in Chapter 1, and of the dynamic parameters needed for the dynamic model presented in Chapter 3.

The following functions are available:

5.4.1. New

This function permits to create a file containing the parameters of a robot. At first the number of joints ($N_j \leq 100$), the number of moving links ($n \leq 80$), the type of structure (open, tree, or closed), and the name of the robot will be specified, the program will initialize the other parameters by default values. Recall that for serial or tree structure robots $N_j = n$, while for closed loop robots $N_j > n$, the number of frames is equal to $Nf = [n + 2(Nj\text{-}n)]$. The default values suppose that all the joints are revolute and that all the geometric parameters are zero except the joint variables, the antecedent a(j) and μ(j). The default dynamic parameters are general. These initial values can be modified using the graphical interface. The following parameters and notations are defined (see Figure 5.1):

5.4.1.1. The geometric parameters

These parameters define the geometry of the structure, the type of joints and the location of the fixed frame of each link with respect to its antecedent frame using the method presented in Chapter 1.

Recall that 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 $j = 1,\ldots,N_f$), where N_f is the number of frames:

- γ_j, b_j, α_j, d_j, θ_j, r_j defining frame j with respect to its antecedent frame a(j), in serial robots γ_j and b_j are equal to zero.

- σ_j defining the type of joint j. Recall that $\sigma_j = 0$ for j revolute, $\sigma_j = 1$ for j prismatic, and $\sigma_j = 2$, if j represents a fixed frame with respect to its antecedent. This last case takes place for frames $N_j+1,...,N_f$ for closed loop robots and can be used to define an end-effector frame.

- a(j) the antecedent frame of frame j.

- μ_j indicates if the joint j is active (motorized) $\mu_j = 1$, or passive $\mu_j = 0$. In serial robots all the joints are supposed active.

We note that the number of frames, N_f, for serial or tree structure robot is equal to the number of moving links n. In the case of closed loop robot each closed loop is supposed opened in one of its passive joint to construct an equivalent tree structure, then two frames are added on the opened joints (see Chapter 1). Thus in the case of closed loop robots

$$N_f = n + 2B$$

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

5.4.1.2. Dynamic parameters

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

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

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, around the origin of frame j,
- M_j is the mass of link j,
- Ia_j is the inertia of rotor of motor j referred to the joint side,
- Fv_j , Fs are the viscous and Coulomb friction coefficients.

5.4.1.3. Additional parameters

The following parameters can also be defined:

- the location of the base of the robot with respect to a general fixed frame (using the 4x4 transformation matrix **Z**),
- the joint velocities (QPj) and accelerations (QDPj),
- the external forces and moments applied by each link on the environment (FX,…,CZ),
- the speed and the acceleration of the base of the robot (if it is mobile): (WX,…,WPZ,VX,…,VPZ),
- the acceleration of gravity: $\mathbf{g} = [G1\ \ G2\ \ G3\ ]^T$.

It is to be noted that the joint velocities (QP_j) and accelerations (QDP_j) are zero for a fixed frame ($\sigma_j = 2$). The velocities and accelerations of frames $N_{j+1},\ldots,N_f$ are not given, they are zero, because they represent fixed frames.

Symbols or numerical values can be given for the parameters, the angle with numerical data must be defined in radian in Mathematica form, for example 90° must be written as Pi/2.

The user can save the robot anywhere, but we recommend to create a directory called “robot-name/” under the user's robots directory, then to save the robot parameters in the file “**robot-name/robot-name.par**”. The extension “**.par**” is automatically appended by SYMORO+.

<u>Remark</u>: All the models of a given robot will be proposed to be saved by default in the directory “robot-name/”. To simplify the writing we will give in the following just the name of the created files. All the extensions are automatically appended by SYMORO+.

The file **robot-name.par** is presented as follows:

```
(* File robot-name.par *)

(* General parameters *)
(* Robotname = 'robot-name' *)
NF = number of frames
NL = number of links
NJ = number of joints
Type = 0 or 1 or 2 (* Simple or Tree or Closed structure respectively *)

(* Geometric parameters *)
Ant = {0,1,2,  …  }
Sigma  = {0,1,0,  …  }
B  = {b1,  …  ,bNF}
d  = {d1,  …  ,dNF}
R  = {r1,  …  ,rNF}
gamma  = {γ1,  …  ,γNF}
Alpha  = {0,-Pi/2,0,Pi/2,  …  }
Mu  = {1,1,0,  …  }
Theta  = {t1,  …  tNF}

(* Dynamic parameters and external forces *)
XX = {XX1,  …  ,XXNL}
XY  = {XY1,  …  ,XYNL}
 … =           …
MZ  = {MZ1,  …  ,MZNL}
M  = {M1,  …  ,MNL}
IA  = {IA1,  …  ,IANL}
FV  = {FV1,  …  ,FVNL}
FS  = {FS1,  …  ,FSNL}
FX  = {FX1,  …  ,FXNL}
… =            …
CZ  = {CZ1,  …  ,CZNL}
```

```
(* Joints velocity and acceleration *)
QP = {QP1,  …  ,QPNJ}
QDP  = {QDP1,  …  ,QDPNJ}

(* Speed and acceleration of the base *)
W0 = {W1,W2,W3}
WP0  = {WP1,WP2,WP3}
V0  = {V1,V2,V3}
VP0  = {VP1,VP2,VP3}

(* Matrix Z *)
Z = {Z11,Z12,Z13,Z14,Z21,Z22,Z23,Z24,Z31,Z32,
      Z33,Z34,0,0,0,1}

(* Acceleration of gravity *)
G = {G1,G2,G3}

(* End of definition *)
```

5.4.2. Open

This function permits to open the file containing the parameters of a previously defined robot, the directory of this robot will be selected by the user. The program displays the names of the files ***.par** contained in the directory, one of them can be selected by the user.

5.4.3. Save

This function is used to save the current parameters under the same file "robot-name.par". The parameters of the current robot can be changed at any time on the interface page, this changes will be taken into account by the different functions even if they are not yet saved.

5.4.4. Save as

This function is used to save the current parameters under another robot name. The new ***.par** file will be created by default in the current robot-name directory, but the user can choose another one.

5.4.5. Preferences

It permits to change some options like the number of Most Recently Used files to display in the Robot menu.

5.4.6. MRU file list

The Most Recently Used file list. The user can open a robot parameter file by clicking on its name in the list.

5.4.7. Exit

To quit SYMORO+.

5.5. Menu Geometric

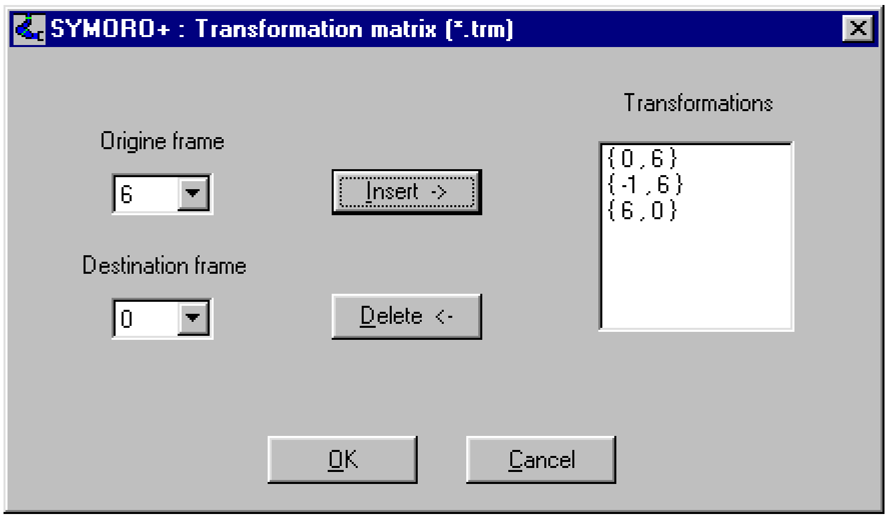
This menu generates the direct and inverse geometric models of open, tree, or closed loop robots, using the methods presented in Chapter 2.

The following functions are available:

5.5.1. Transformation matrix

This function generates the (4x4) transformation matrices between a set of specified pairs of frames T[i,j], T[m,k], ... which means ${}^{i}\mathbf{T}_{j}$, ${}^{m}\mathbf{T}_{k}$.

The definition of these frames will be carried out in the window shown in figure 5.3. The user defines an origin frame and a destination frame then insert this matrix definition by clicking on the button “Insert”.



{0,6}
{-1,6}
{6,0}

Figure 5.3 : Definition of the desired transformation matrices.

The default output file is: **"robot-name.trm"**. The (r,s) element of the matrix T[i,j] is denoted as TiTjrs.

Recall that a destination frame j will be defined with respect to an origin frame r, by the following (4x4) matrix:

$$ {}^{r}\mathbf{T}_{j} = \begin{bmatrix} {}^{r}\mathbf{A}_{j} & {}^{r}\mathbf{P}_{j} \\ 0\ 0\ 0 & 1 \end{bmatrix} $$

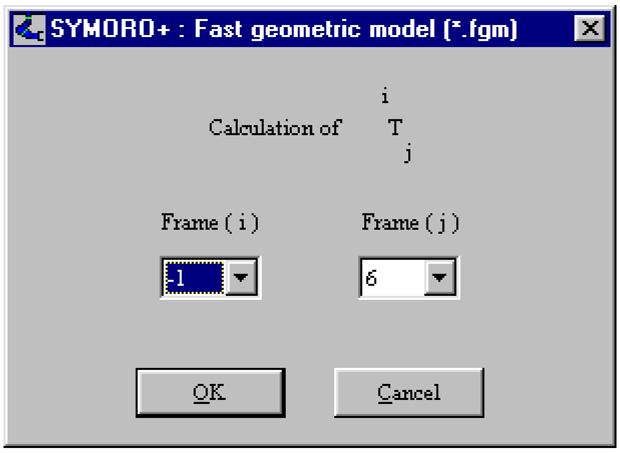
where: ${}^{r}\mathbf{A}_{j}$ is (3x3) orientation matrix, ${}^{r}\mathbf{P}_{j}$ (1x3) position vector.

For a simple open loop robot the matrix ${}^{0}\mathbf{T}_{n}$ defines the direct geometric model. For closed loop robots the constraint equations defining the relation between the unmotorized joints, 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 using the function "constraint geometric equations of loops" , which is described in section 5.5.7.

Recall that frame -1 indicates the general reference frame and frame 0 indicates the frame fixed with the base of the robot.

5.5.2. Fast geometric model

This function generates the transformation matrix between two frames, ${}^{i}\mathbf{T}_{j}$, using intermediate variables (customized method). The values i and j will be defined using the window shown in figure 5.4 .


SYMORO+ : Fast geometric model (*.fgm)

Calculation of $^{i}T_{j}$

Frame (i): -1

Frame (j): 6

Figure 5.4 : Definition of the desired transformation matrix for fast geometric model.

The default output file is: **“robot-name.fgm”**. The (r,s) element of the matrix T[i,j] is denoted by TiTjrs.

The output file can be optimized from the number of operations point of view using the menu **Optimizer,** which can also provide the optimized model in C, Matlab, Maple, Fortran or Mathematica format.

5.5.3. I.G.M. Pieper method

This function generates the inverse geometric model of 6 d.o.f. robots having three prismatic joints or three intersecting revolute joints forming a spherical joint. For such robots the I.G.M. solution exists. The default output file is: **“robot-name.pie”**.

The model is given as a set of equations (each of them containing only one unknown joint variable) which are a function of the desired location of the terminal link T[0,6], denoted as **[s n a P]**, and the already resolved variables. If the robot contains closed loops the program gives the inverse geometric model of the direct chain between the base and the terminal link. The theoretical basis of this method is given in chapter 2. The program can take into account the fixed terminal frames.

5.5.4. I.G.M. General method (symbolic or numeric)

This function gives the inverse geometric model of robots with 6 d.o.f. using the method of Raghavan and Roth [Raghavan 89]. If the robot contains closed loops the program gives the I.G.M. of the direct chain between the base and the terminal link. In the case of tree robots, the program gives the I.G.M. of all 6 d.o.f. chains between the base and the terminal links. This method gives the I.G.M. symbolically or numerically.
The default output file is **“robot-name.rag”** for the symbolic computation and **“robot-name.ragn”** for the numerical one.

The following intermediate results are given:

- vectors **X, Y** and the matrices **A, B** corresponding to equation (A1-4): **A X = B Y**
- vector **X1** and matrix Σ corresponding to the equation Σ **X1 = 0**
- factorized determinant of Σ that represents the polynomial equation of the first joint variable to be obtained.
- all the solutions of the joints for a given configuration (only if numerical computation).

As it is impossible to obtain symbolically the determinant of Σ for some robots because of the complexity of calculating a symbolic (12x12) determinant, the user specifies if he wants to obtain this determinant or not.

If a numerical result is required, the user will be asked to define the number of points of the end effector for which the program has to calculate the I.G.M. (maximum 10 points). For each point the following data must be defined:

* the position, which is given by the Cartesian coordinates of **P,**

* the orientation, which will be defined by:

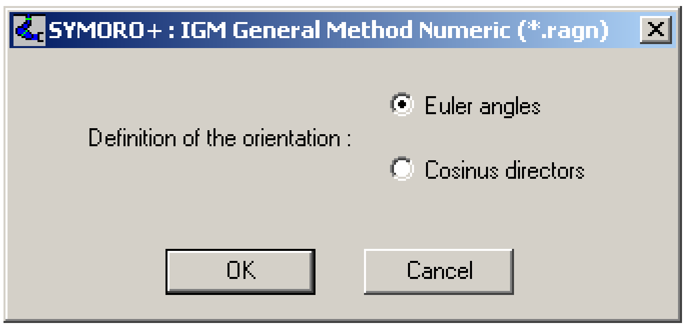
. either the cosines directors vectors **n** and **a**, the program calculates **s** as **nxa**.

. or, as shown in figure 5.5, the Euler angles ϕ, θ, ψ, in radians, the program calculates:

[s n a] using the relation **rot(z,ϕ) rot(x,θ) rot(z,ψ),** which gives:

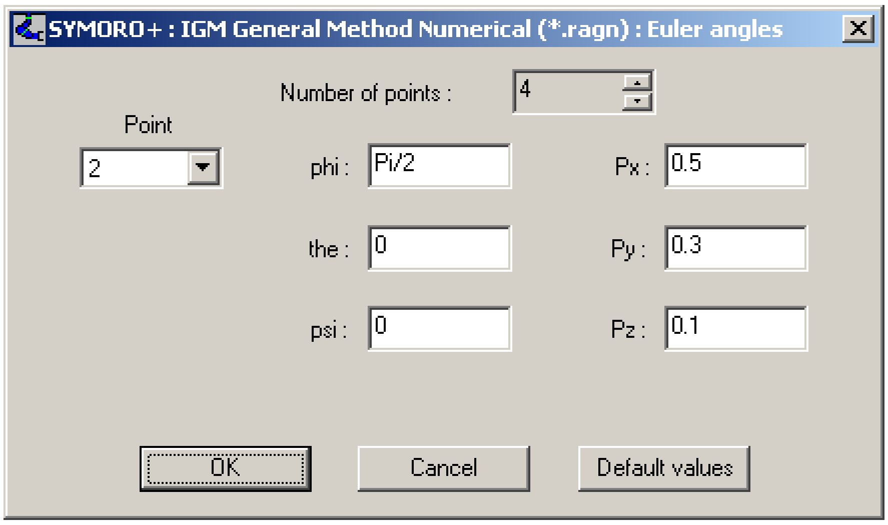
$$[\mathbf{s}\ \ \mathbf{n}\ \ \mathbf{a}] = \begin{bmatrix} C\phi C\psi - S\phi C\theta S\psi & -C\phi S\psi - S\phi C\theta C\psi & S\phi S\theta \\ S\phi C\psi + C\phi C\theta S\psi & -S\phi S\psi + C\phi C\theta C\psi & -C\phi S\theta \\ S\theta S\psi & S\theta C\psi & C\theta \end{bmatrix}$$

The numerical solution of the I.G.M. will be given by default in the file **"robot-name.ragn"**.


SYMORO+ : IGM General Method Numeric (*.ragn)

Definition of the orientation : [✓ Euler angles] Cosinus directors

OK Cancel


SYMORO+ : IGM General Method Numerical (*.ragn) : Euler angles

Number of points : 4

Point: 2

phi : Pi/2

the : 0

psi : 0

Px : 0.5

Py : 0.3

Pz : 0.1

Figure 5.5 : Inverse geometric model using general method.

Two error messages could occur for the numerical computation:
- if the geometric parameters are not defined numerical (except the joint variables).
- if the input elements, **n**, **a** or **P** of one of the points, are not numerical.

5.5.5. Transformation matrices for I.G.M.

This function computes the transformation matrices that can be used by the method of Paul to obtain the inverse geometric model. The default output file is: **"robot-name.mpi"**.
These matrices are:
$U_k = T[k,n]$, for k=0,...,n-1 , with $U_j = T[j,j-1]\ U_{j-1}$, in forward direction, and
$U_k = T[0,n-k]$, for k=0,...,n-1 , with $U_j = U_{j-1} T[n-j,n-j+1]$ in backward direction.

The aim of this function is to give the user the tool to obtain by hand the IGM by the method of Paul if the program cannot generate it automatically, or to verify the output of the inverse geometric model of Paul's method. The matrices of both directions will be given.

5.5.6. I.G.M. Paul method

This function generates the inverse geometric model by Paul's method. The default output file is: **"robot-name.pau"**.
This method can provide the solution of the I.G.M. of most of the current industrial robots. The theoretical basis of this method is given in chapter 2, it is summarized as follows:

The problem is to find the joint variables **q** for a given position and orientation of the end effector ${}^{f}\mathbf{T}_{E}{}^{d}$. That is to solve the following matrix equation:

$${}^{f}\mathbf{T}_{E}{}^{d} = {}^{0}\mathbf{T}_{6}\,(\mathbf{q})\ \mathbf{E}$$

For 6 d.o.f. robots, the problem is reduced to the solution of:

$$\mathbf{U}_0 = {}^0\mathbf{T}_6\,(\mathbf{q})$$

where $\mathbf{U}_0 = [\mathbf{s}\quad \mathbf{n}\quad \mathbf{a}\quad \mathbf{P}]$

Using the forward direction, we obtain the following set of equations:

$$\begin{aligned}
\mathbf{U}_1 &= {}^1\mathbf{T}_2\,{}^2\mathbf{T}_3\,{}^3\mathbf{T}_4\,{}^4\mathbf{T}_5\,{}^5\mathbf{T}_6 \\
\mathbf{U}_2 &= {}^2\mathbf{T}_3\,{}^3\mathbf{T}_4\,{}^4\mathbf{T}_5\,{}^5\mathbf{T}_6 \\
\ldots &= \quad \ldots \\
\mathbf{U}_5 &= {}^5\mathbf{T}_6
\end{aligned}$$

with:

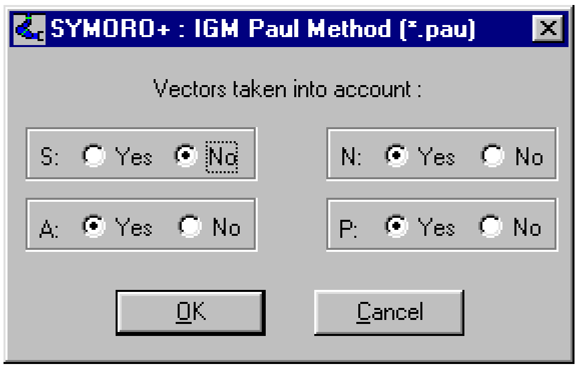
$$\mathbf{U}_j = {}^j\mathbf{T}_{j\text{-}1}\,\mathbf{U}_{j\text{-}1}$$

The inverse geometric model can be obtained by solving the previous equations. The use of this method on a big number of industrial robots has allowed us to identify ten types of equations which can be used to find the inverse geometric model of almost all industrial robots [Khalil 86b], these equations, see table 2.1, are identified and solved by SYMORO+.

If the solution fails, the program will try to solve the backward direction equations.

Remarks:

1. For a robot with less than 6 d.o.f., only some vectors of the matrix ${}^f\mathbf{T}_E{}^d$, denoted as **s, n, a, P** must be used in the solution (these vectors define the terminal-effector whose dimension < 6), the program will ask the user to define these vectors using the window of figure 5.6 . For example for 5 d.o.f. robot the user can choose the vectors **a, P,** while for 3 d.o.f. robot the user can choose only **P**. The backward direction equations are not used in this case.
2. For closed loop robots, the programs (the Pieper or Paul methods) give the solution of the joints on the direct path between the base of the robot and the end-effector. To obtain the solution of the motorized joints outside this path the geometric constraint equations of the loops must be solved (see section 5.5.7).



Vectors taken into account :

S: Yes [✓ No]
N: [✓ Yes] No
A: [✓ Yes] No
P: [✓ Yes] No

Figure 5.6 : Inverse geometric model using the Paul method.

5.5.7. Constraint geometric equations of loops

This function gives the solution of the geometric constraint equations of closed loop robots.

The geometric constraint equations can be obtained using the fact that the product of the transformation matrices along the loop (Figure 2.2) is equal to the identity matrix:

$$^{k}\mathbf{T}_{i} \dots {}^{j}\mathbf{T}_{k+B} = \mathbf{I}_4$$

with $\mathbf{I}_4$ is the 4x4 identity matrix.

The solution gives the passive joints (whose μ are equal to zero) as a function of the active joints (whose μ are equal to one), which is needed in the direct and inverse geometric models of such robots. If it is desired to obtain motorized joints as a function of some passive joints, we have to set μ of the supposed known variables to be equal to 1, and μ of the unknown to be equal to zero.

More than one solution may be given for each variable. The solution is ensured for closed loops with less than four non motorized joints (passive), the default output file is **“robot-name.cgel”**.

5.6. Menu Kinematic

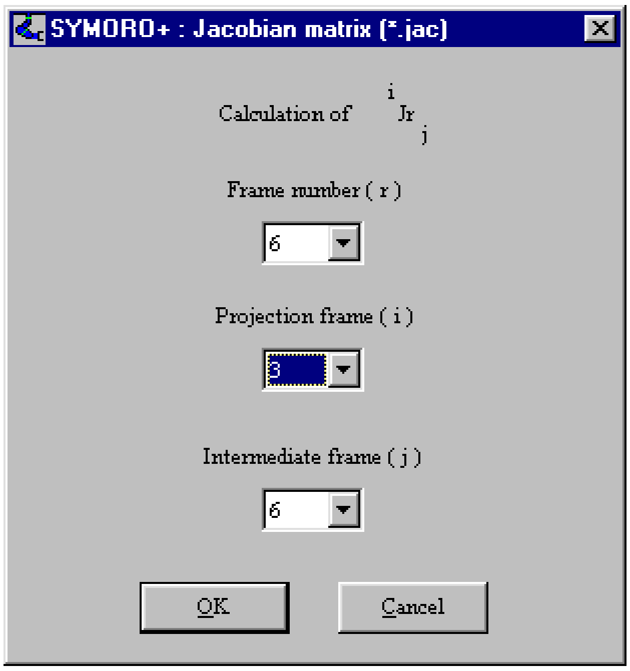
This menu generates the direct and inverse kinematic models of open, tree, or closed loop robots. The following functions are available:

5.6.1. Jacobian matrix

This function computes the Jacobian matrix $^{i}\mathbf{J}_{r,j}$ relating the velocities of link r, projected into frame i, using the intermediate frame j.

The result is given by the multiplication of two matrices **L*J**. The joint velocities vector from the base to the specified link will be given by the vector **VQP**. The default output file is: **“robot-name.jac”**.

A file containing the Jacobian matrix in MATHEMATICA format will be given in **“robot-name.Mjac”**; this file will be used to obtain the inverse and the determinant of the Jacobian matrix. The definition of i, j, and r is carried out using the window of figure 5.7.


SYMORO+ : Jacobian matrix (*.jac)

Calculation of ${}^{i}Jr_{j}$

Frame number (r): 6

Projection frame (i): 3

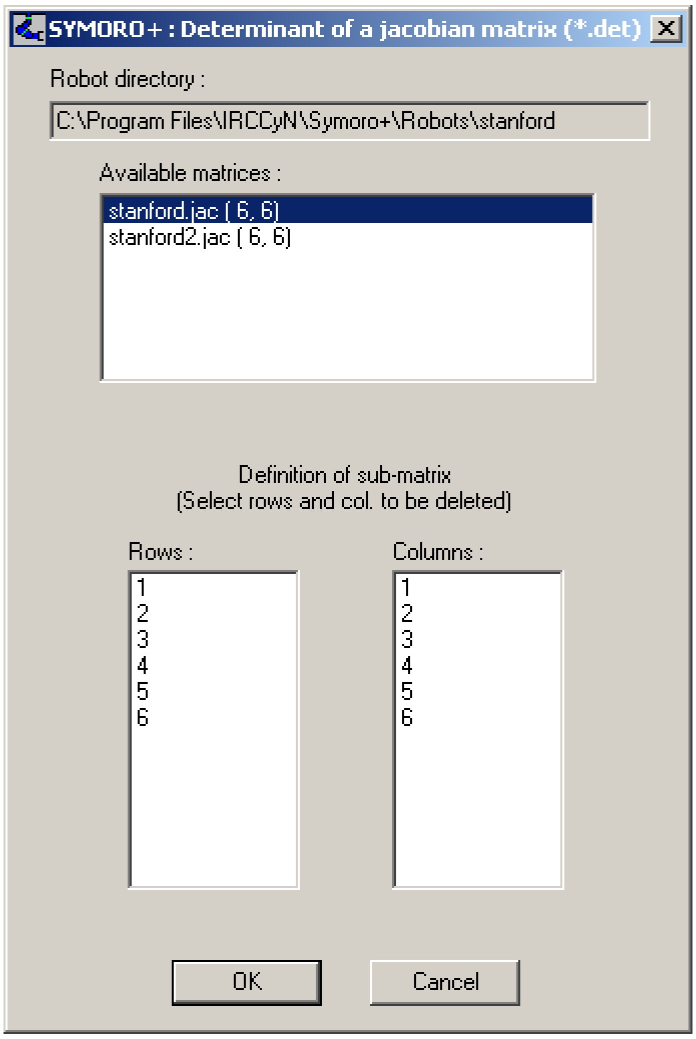
Intermediate frame (j): 6

Figure 5.7 : Definition of a Jacobian matrix.

5.6.2. Determinant of a Jacobian

This function gives the determinant of a Jacobian matrix. The program will display the names of the Jacobian matrices existing in the selected directory (having extension .jac), from which the user selects one.(the existing Jacobian matrices *.jac must have their corresponding Mathematica format files *.Mjac accessible in the same directory; these files are automatically generated by the "Jacobian matrix" function)

Some columns or rows can be eliminated from the selected matrix as shown in figure 5.8 . If the matrix is not square the determinant of the matrix $\mathbf{J}\ \mathbf{J}^T$ will be given. The default output file is **"robot-name.det"**.


SYMORO+ : Determinant of a jacobian matrix (*.det)

Robot directory :

C:\Program Files\IRCCyN\Symoro+\Robots\stanford

Available matrices :

stanford.jac (6, 6)
stanford2.jac (6, 6)

Definition of sub-matrix
(Select rows and col. to be deleted)

Rows : 1 2 3 4 5 6

Columns : 1 2 3 4 5 6

Figure 5.8 : Selection of a Jacobian matrix.

5.6.3. Inverse of a Jacobian

This function gives the inverse of a Jacobian matrix. The program will display the names of the Jacobian matrices existing in the selected directory (having extension .jac), from which the user selects one.(the existing Jacobian matrices *.jac must have their corresponding Mathematica format files *.Mjac accessible in the same directory; these files are automatically generated by the "Jacobian matrix" function)

Some columns or rows can be eliminated from the selected matrix in the same way as for the function "Determinant of a Jacobian" and shown in figure 5.8 . The matrix to be inverted must be square. The execution time of this function may take a long time in some cases (complicated Jacobian matrix). The default output file is: **"robot-name.inv"**.

It is to be noted that if the Jacobian matrix is very complicated, its symbolic inverse is not practical to use, it is more efficient in this case to calculate the numerical value of the Jacobian matrix, and then invert it numerically.

5.6.4. Velocities of links

This function calculates the revolute and transnational velocities of links projected in the link frames. The result is given in customized form; the default output file is: **“robot-name.vel”**. The translational velocity of frame j is denoted as [V1j V2j V3j]. The revolute velocity of frame j is denoted as [W1j W2j W3j].

The output file can be optimized from the number of operations point of view using the menu **Optimizer,** which can also provide the optimized model in C, Matlab, Maple, Fortran or Mathematica format.

5.6.5. Accelerations of links

This function calculates the revolute and translational accelerations of links projected in the link frames. The result is given in customized form; the default output file is **“robot-name.acc”**. The translational acceleration of frame j is denoted as [VP1j VP2j VP3j]. The revolute acceleration of frame j is denoted as [WP1j WP2j WP3j].

The output file can be optimized from the number of operations point of view using the menu **Optimizer,** which can also provide the optimized model in C, Matlab, Maple, Fortran or Mathematica format.

5.6.6. Calculation of JpQp

This function calculates the vector $\dot{\mathbf{J}}\dot{\mathbf{q}}$, this vector is used to calculate efficiently the second order kinematic model. It is calculated from the link accelerations algorithm by assuming that the joint accelerations are equal to zero, thus this vector is also called partial translational and revolute accelerations of links. The j^{th} component of the partial translational acceleration of frame k is denoted as VPJjk. The j^{th} component of the partial revolute acceleration of frame k is denoted as WPJjk. The output vector is given in customized form in the default file **“robot-name.jpqp”**.

The output file can be optimized from the number of operations point of view using the menu **Optimizer,** which can also provide the optimized model in C, Matlab, Maple, Fortran or Mathematica format.

5.6.7. Constraint kinematic equations of loops

This function calculates the first and second order kinematic constraint equations of closed loop robots. These equations give the relation between joint velocities and accelerations of passive joints as function of the velocities and accelerations of active joints. Redundant equations are, in general, eliminated automatically. The default output file is **“robot-name.ckel”**.

The output file can be optimized from the number of operations point of view using the menu **Optimizer**.

The elements given in this file are needed in the calculation of the dynamic models of closed loop robots. Recall that the velocity constraint equation is given as:

$$\begin{bmatrix} \mathbf{W_a} & \mathbf{W_p} & \mathbf{0} \\ \mathbf{W_{ac}} & \mathbf{W_{pc}} & \mathbf{Wc} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{q}}_a \\ \dot{\mathbf{q}}_p \\ \dot{\mathbf{q}}_c \end{bmatrix} = \mathbf{0}$$

where $\dot{\mathbf{q}}_a$, $\dot{\mathbf{q}}_p$ represent the velocity of the active (motorized) and the passive joints of the robot respectively, while $\dot{\mathbf{q}}_C$ represents the virtually opened joint velocities. Recall that in the case of opening a complex joint the other degrees of freedom of this joint may be included in the vector $\dot{\mathbf{q}}_C$.

The acceleration constraint equation is given as:

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

where $\ddot{\mathbf{q}}_a$, $\ddot{\mathbf{q}}_p$ represent the acceleration of the active (motorized) and the passive joints of the robot respectively, while $\ddot{\mathbf{q}}_C$ represents the virtually opened joint accelerations.

A file containing the matrices $\mathbf{W_a}$ and $\mathbf{W_p}$ in Mathematica format will be given in the file **"robot-name.Mckel",** this file will be used by the function "Resolution of constraint equations of loops" to obtain the matrix **W** as: - $\mathbf{W_p^{-1}}$ $\mathbf{W_a}$.

5.6.8. Resolution of constraint equations of loops

This function provides the symbolic matrix **W** using the relation $\mathbf{W} = -\mathbf{W_p^{-1}}\ \mathbf{W_a}$. This matrix gives the passive joint velocities as function of the active (motorized) joint velocities. The input file **"robot-name.Mckel"** containing the constraint kinematic equations of loops in Mathematica format must be selected by the user. This file is automatically generated by the function "Constraint kinematic equations of loops".

The default output file is **"robot-name.ickel"**.

5.7. Menu Dynamic

This menu generates the direct and inverse dynamic models of open, tree, or closed loop robots. It is to be noted that all these models will have an efficient model from the number of operations point of view if the base inertial parameters (see menu identification) are used.

5.7.1. Inverse dynamic model

This function gives the inverse dynamic model of a robot using Newton Euler method, which is considered as the most efficient method to get the inverse dynamic model. The use of the base inertial parameters will reduce the number of operations of the generated model. The generated model is given in customized form in the default file **"robot-name.dyn"**. The torque or force of motor j is denoted GAMj. The component i of reaction forces and moments on link j, by its antecedent link a(j), is denoted as Eij and Nij respectively.

The output file can be optimized from the number of operations point of view using the menu **Optimizer,** which can also provide the optimized model in C, Matlab, Maple, Fortran or Mathematica format.

If the robot contains closed loops, the model corresponding to the equivalent tree structure will be given, the model can be completed using the constraint equations, developed in geometric and kinematic menus.

5.7.2. Lagrange method

This function gives the dynamic cefficients, the elements of the matrices **A, B, C** and **Q**, of the inverse dynamic model, under customized form in the default file **"robot-name.lag"**, or expanded form in the default file **"robot-name.Elag"**. The dynamic coefficients are denoted A(i,j), B(i,j,k), C(i,j) and Q(j).

In the case of customized model, the output file can be optimized from the number of operations point of view using the menu **Optimizer,** which can also provide the optimized model in C, Matlab, Maple, Fortran or Mathematica format.

It is to be noted that the expanded form may take very long time to be obtained (some hours), because of the use of trigonometric simplifications.

Recall that the Lagrange dynamic model is given as:

$$\mathbf{\Gamma} = \mathbf{A}\,\ddot{\mathbf{q}} + \mathbf{B}\,\dot{\mathbf{q}}\dot{\mathbf{q}} + \mathbf{C}\,\dot{\mathbf{q}}^2 + \mathbf{Q}$$

where:

- $\mathbf{\Gamma}$ is the vector of motor forces or torques.
- **A** is the robot inertia matrix.
- **B** is the (n x n (n–1)/2) matrix containing the elements of Coriolis forces, its general element is represented by $B_{i,jk}$.
- **C** is the (n x n) matrix containing the elements of centrifugal forces, its general element is represented by C_{ij} .

$\mathbf{\dot{q}\dot{q}}$ = $[\dot{q}_1\dot{q}_2 \; \cdots \; \dot{q}_1\dot{q}_n \;\; \dot{q}_2\dot{q}_3 \cdots \; \dot{q}_{n-1}\dot{q}_n]^T$

$\mathbf{\dot{q}^2}$ = $[\dot{q}_1{}^2 \; \ldots \; \dot{q}_n{}^2]^T$

$\mathbf{Q}$ = $[Q_1 \; \ldots \; Q_n]^T$, the gravity forces or torques vector.

5.7.3. Inertia matrix

This function calculates the inertia matrix of robots. If the robot contains closed loops, the program gives the inertia matrix of the equivalent tree structure, the inertia matrix corresponding to the active joints can be obtained using the kinematic constraint equations.

The matrix is given in customized form in the default file **"robot-name.inm"**. The (i,j) element of the inertia matrix is denoted as A(i,j).

It is to be noted that if only the inertia matrix is needed, its calculation in this function is more efficient than that calculated in the method of Lagrange.

The output matrix can be optimized from the number of operations point of view using the menu **Optimizer,** which can also provide the optimized model in C, Matlab, Maple, Fortran or Mathematica format.

5.7.4. Centrifugal, Coriolis & Gravity torques

This function gives the vector of the centrifugal, Coriolis and gravity forces or torques on the motors of robots. This vector is represented by:

$$\mathbf{H} = \mathbf{B}\,\mathbf{\dot{q}\dot{q}} + \mathbf{C}\,\mathbf{\dot{q}^2} + \mathbf{Q}$$

This vector is needed for the simulation of the robot (see section 3.6.1), it is calculated using Newton Euler algorithm after putting the joint accelerations equal to zero.

The vector is given in customized form in the default file **"robot-name.ccg"**. The torque or force of motor j is denoted Hj .

The output file can be optimized from the number of operations point of view using the menu **Optimizer,** which can also provide the optimized model in C, Matlab, Maple, Fortran or Mathematica format.

If the robot contains closed loops, the model corresponding to the equivalent tree structure will be given.

5.7.5. Direct dynamic model

This function gives the direct dynamic model (simulation model) of tree structure and open loop robots. It calculates the accelerations of the joints as function of the input torques or forces and the state of the robot using a recursive algorithm without inverting the inertia matrix.

The joint accelerations vector is given in customized form in the default file **"robot-name.ddm"**. The acceleration of joint j is denoted QDPj. The reaction forces and moments on links are calculated also in this function, the component i of reaction forces and moments on link j, by its antecedent link a(j), is denoted as Eij and Nij respectively.

This function can be used for systems with lumped elasticity, where the torque of such joint is set equal to -Kj*qj, with Kj is the stiffness and qj is the position variable with respect to the zero force position.

The output file can be optimized from the number of operations point of view using the menu **Optimizer,** which can also provide the optimized model in C, Matlab, Maple, Fortran or Mathematica format.

We note that this method is proved to be more stable numerically than the method based on the calculation of **A** and **H.**

5.8. Menu Identification

This menu determines the base inertial parameters (known also as minimum inertial parameters or identifiable inertial parameters) and generates the dynamic, energy, filtered dynamic identification models.

The following functions are available:

5.8.1. Base inertial parameters (symbolic or numerical)

Calculation of the base inertial parameters of open, tree structure or closed loop robots. These parameters are also known as the minimum inertial parameters.

The base inertial parameters are defined as the minimum parameters which can be used to compute the dynamic model. They represent the set of parameters which can be identified using the dynamic or energy model, thus its determination is essential to the identification of the inertial parameters of robots. The use of the base parameters in Newton-Euler algorithm, or inertia matrix model, or centrifugal, Coriolis, and gravity vector,… (of menu dynamic) leads to reduce the computational complexity of these models.

These parameters can be obtained from the classical inertial parameters by eliminating those having no effect on the dynamic model and by regrouping some others. Two methods are available:

- symbolic method: the base inertial parameters of simple open loop, or tree structure robots will be completely obtained. For closed robots the minimum inertial parameters will be completely given in most cases, the numerical method must be used to detect if there are more parameters to regroup or to eliminate.

The default output file is **“robot-namebase.par”** which is similar to **“robot-name.par”** but the base inertial parameters replace the standard ones. The expressions of the regrouping parameters are given in file **“robot-name.regp”**.

- **numerical method:** when using this function the user must take care about the following:

i- the acceleration of the gravity and the constant geometric (only geometric) parameters of the robot must be given numerically, this can be done directly while defining the robot, or in the file **"robot-name.cons"** which must be selected by the user.

ii- for closed loop robots the geometric and kinematic constraint equations (giving the positions and velocities of passive joints as function of the positions and velocities of active joints) of the robot must be provided using Mathematica language in the subroutine **"Constraint"** in the file **"robot-name.cons"** (this file can be used also to define the numerical values of the geometric parameters). If the constraint equations are not given, the parameters of the equivalent tree structure will be obtained. The constraint relations can be obtained using geometric and kinematic menus. The file **"SR400.cons"** corresponding to the robot SR400, table 1.3, is given as follows (what is written in bold must be copied for all robots):

```
Constraint[nbs_Integer]:= Module [{i,Pi = 3.14159265358979},
 For [i= 1,i ≤ nbs, i++,
 Q[[i,3]] = Pi/2 - Q[[i,2]] + Q[[i,7]];
 Q[[i,8]] =  Q[[i,2]] - Q[[i,7]];
 QD[[i,3]] = - QD[[i,2]] + QD[[i,7]];
 QD[[i,8]] =  QD[[i,2]] - QD[[i,7]];
 ];
]
```

where:

Q[[i,j]] denotes the ith random variable of joint j,

QD[[i,j]] denotes the ith random velocity of joint j,

nbs is the number of random points, it is defined in the main program.

Remark:

1. This program defines the following equations, which give the positions and velocities of the passive joint variables as a function of the active joint variables:

   $\theta_3 = \pi/2 - \theta_2 + \theta_7$

   $\theta_8 = \theta_2 - \theta_7$

   and

   $\dot{\theta}_3 = - \dot{\theta}_2 + \dot{\theta}_7$

   $\dot{\theta}_8 = \dot{\theta}_2 - \dot{\theta}_7$

2. The values of the constant geometric parameters and the values of the gravity acceleration can be given also in this file before or after the Constraint subroutine, for example the value of d8 can be given as:

   d8 = 0.5;

3. The default output file is **"robot-namebaseN.par"** which is similar to **"robot-name.par"** but the base inertial parameters replace the standard ones. The expressions of the regrouping parameters are given in the file **"robot-name.Nreg"**.

It is recommended to compute the dynamic models (menu dynamic) using the base inertial parameters **“robot-namebase.par”** in order to reduce the corresponding number of operations.

5.8.2. Energy identification model

Using the dynamic model in the identification needs to estimate or measure the joint accelerations. To overcome this difficulty a model based on the energy theorem has been proposed [Gautier 88b]. To simplify the writing we assume that the friction is neglected. From the energy theorem we obtain:

$$\int_{t_1}^{t_2} \dot{\mathbf{q}}^T \mathbf{\Gamma}\, dt = H(t_2) - H(t_1)$$

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

Since H is linear in the inertial parameters then:

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

where:

K denotes the vector of inertial parameters,

h denotes a row matrix.

This relation represents a linear equation in the inertial parameters of the links, it is function of the joint positions, velocities and of the input joint torques. To identify **K** a sufficient number of equations can be obtained by calculating this relation between different intervals of time. The least squares solution is generally used in the identification.

This function calculates the elements of the matrix **h**. The coefficient of the inertial parameter Ki will be denoted as HKi. The coefficients HKi are given in customized form in the default file **“robot-name.eim”**.

The output file can be optimized from the number of operations point of view using the menu **Optimizer,** which can also provide the optimized model in C, Matlab, Maple, Fortran or Mathematica format.

We note that these coefficients are also used in the power identification model, which is given as:

$$\dot{\mathbf{q}}^T \mathbf{\Gamma} = d/dt\, [\mathbf{h}(\mathbf{q},\dot{\mathbf{q}})]\, \mathbf{K}$$

The time derivative can be obtained numerically or by filtering.

5.8.3. Dynamic identification model

This function generates the identification model of the dynamic parameters (inertial and friction) using the dynamic model. Since the dynamic model is linear in the inertial parameters and function of the joint positions, velocities and accelerations, it can be written as:

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

where:

K is the vector of inertial parameters,

D is a (N_LxNp) matrix where N_L is the number of links, N_p is the number of parameters.

This function gives the symbolic expressions of the elements of the matrix **D**. The ith component (corresponding to joint i) of the vector corresponding to the inertial parameter Kj will be denoted as DGiKj. The coefficients DGiKj are given in customized form in the default file **“robot-name.dim”**.

The output file can be optimized from the number of operations point of view using the menu **Optimizer,** which can also provide the optimized model in C, Matlab, Maple, Fortran or Mathematica format.

5.8.4. Filtered dynamic identification model

This function generates the filtered dynamic parameters (inertial and friction) identification model using the Lagrange equation (see Appendix 5) [Khalil 96, Restrepo 96] that is represented by:

$$\Gamma = \frac{dE}{dt} - \frac{\partial E}{\partial q} + \frac{\partial U}{\partial q} + \mathbf{F}$$

where E and U are the kinetic and potential energy of the robot, while **F** is the friction forces.

This equation is linear in the dynamic parameters. The identification problem is based on the numerical solution of the linear least-squares problem:

$$\Gamma = [\frac{d}{dt}\mathbf{D1}(\mathbf{q},\dot{\mathbf{q}}) + \mathbf{D2}(\mathbf{q},\dot{\mathbf{q}})]\ \mathbf{K}$$

In appendix 5, we detail the calculation and the use of this identification model.

This function gives the coefficients of **D1** and **D2** . The ith component (corresponding to joint i) of the vectors corresponding to the inertial parameter Kj are denoted as DEPiKj and DLQiKj respectively. The generated DEPiKj and DLQiKj coefficients are given in customized form in the default file **“robot-name.fdi”**.

The output file can be optimized from the number of operations point of view using the menu **Optimizer,** which can also provide the optimized model in C, Matlab, Maple, Fortran or Mathematica Format.

5.9. Optimizer

This function is written in C, its aim is to optimize, from the number of operations point of view, a model given in customized form (using intermediate variables) by eliminating the intermediate variables that have no effect on the desired output variables. The user selects at first the extension of the file to be optimized (fgm, dyn, vel, acc, dim, lag, ddm, …), then the program displays the existing files in the current directory having this extension. After selecting one of these files, the variables contained in it are listed. (figure 5.9)

The user can select the desired output variables by the mouse or by writing the generic name of the variables, for instance writing GAM as desired output means to specify all the variables starting by GAM as desired output. All the selected output variables are listed and the user can then eliminate some of them or add new ones. Specifying * as generic name means selecting all the available variables.

When the optimization is done, the number and the names of the variables used in the resulting model are displayed. The number of additions (or subtractions) and multiplications (or divisions) in the optimized model are also given. The default name of the output file is given as:

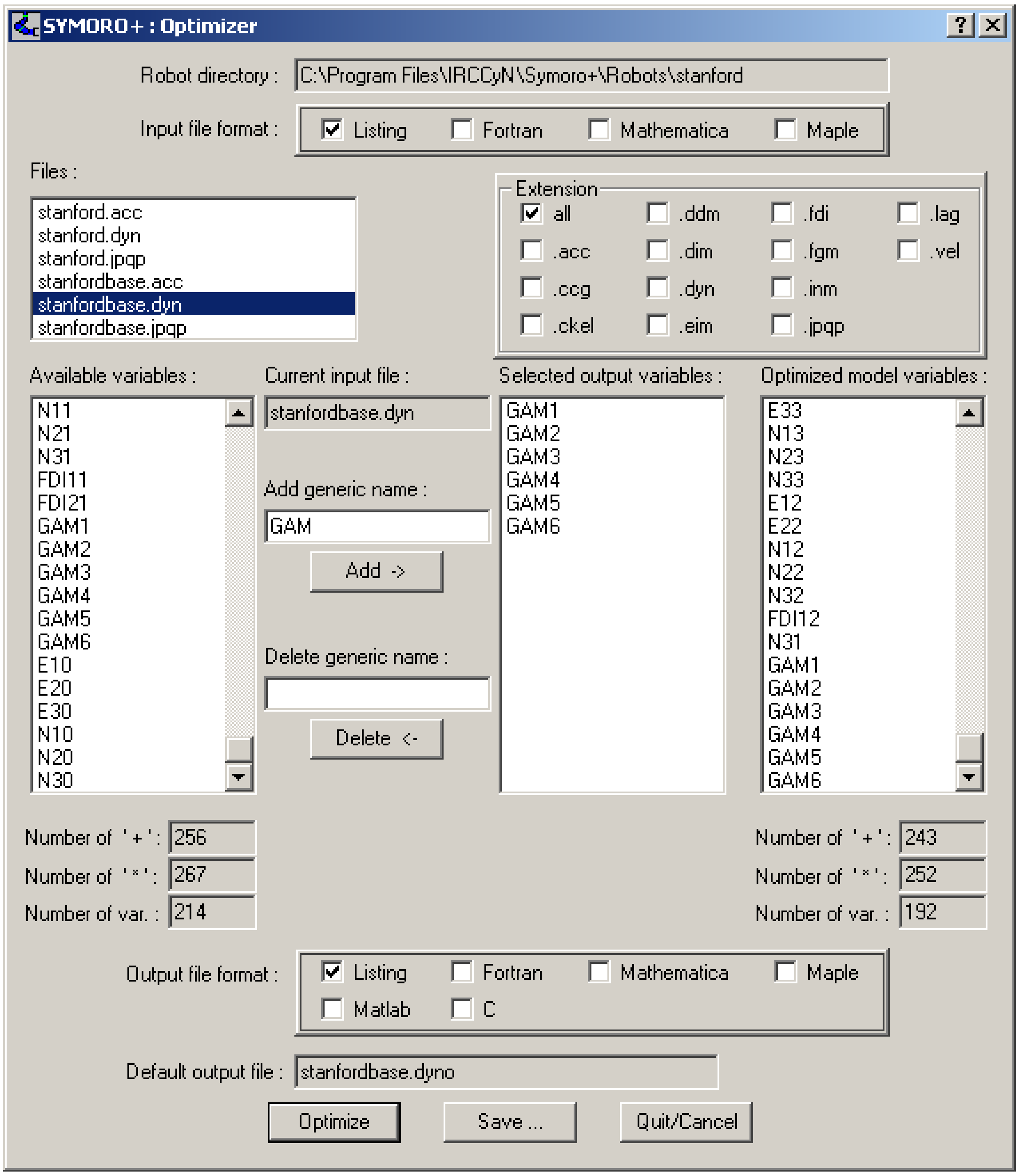
robot-name.**o,	for output file in listing form,
robot-name_**.f,	for output file in Fortran form,
robot-name_**.mh,	for output file in Mathematica form,
robot-name_**.m,	for output file in Matlab form,
robot-name_**.mp,	for output file in Maple form,
robot-name_**.c,	for output file in C form.

where ** means the extension of the input file that must contain a model in customized form.

The output files can be reprocessed by the optimizer to redefine the output variables or to change its format, except for the output files in C or Matlab format for which the content is not in customized form: in these files the model is given as a function which can be called as is in C or Matlab programs. (function parameters and local variables are not used in the generated function in order to minimize stack manipulation or stack overflow in the case of many variables)

To obtain an expanded Mathematica form of a given model, the user can activate Mathematica, give the desired customized Mathematica model as input file using the command "<< ", then by writing the name of the desired output variables, it will be calculated by Mathematica in expanded form.

Be aware that the input files of the optimizer which have been modified by hand may produce unpredictable behavior of the program if the contents are erroneous, specially when conversion to/from other format than listing is used.


SYMORO+ : Optimizer

Robot directory : C:\Program Files\IRCCyN\Symoro+\Robots\stanford

Input file format : [✓ Listing] Fortran Mathematica Maple

Files :

stanford.acc
stanford.dyn
stanford.jpqp
stanfordbase.acc
stanfordbase.dyn
stanfordbase.jpqp

Extension

[✓ all]	.ddm	.fdi	.lag
.acc	.dim	.fgm	.vel
.ccg	.dyn	.inm	
.ckel	.eim	.jpqp	

Available variables :

N11
N21
N31
FDI11
FDI21
GAM1
GAM2
GAM3
GAM4
GAM5
GAM6
E10
E20
E30
N10
N20
N30

Current input file : stanfordbase.dyn

Add generic name : GAM

Add ->

Delete generic name :

Delete <-

Selected output variables :

GAM1
GAM2
GAM3
GAM4
GAM5
GAM6

Optimized model variables :

E33
N13
N23
N33
E12
E22
N12
N22
N32
FDI12
N31
GAM1
GAM2
GAM3
GAM4
GAM5
GAM6

Number of '+': 256
Number of '*': 267
Number of var. : 214

Number of '+': 243
Number of '*': 252
Number of var. : 192

Output file format : [✓ Listing] Fortran Mathematica Maple Matlab C

Default output file : stanfordbase.dyno

Optimize Save ... Quit/Cancel

Figure 5.9 : Optimizer window

Appendix 5.1 : Output files of SYMORO+

This appendix gives the default name of the files generated by SYMORO+. They are classified by menu. The name of the robot is replaced here by * except in the menu “Optimizer”. The default directory for all the output files is the directory of the current opened parameter file (*.par).

Robot

*.par	“file containing the robot parameters”
*.ini	“file containing some customized parameters”

Geometric

*.trm	“transformation matrix”
*.fgm	“fast geometric matrix”
*.pau	“I.G.M. Paul's method”
*.mpi	“transformation matrices in Paul's method”
*.pie	“I.G.M. Pieper's method”
*.cgel	“constraint geometric equations of loops”
*.rag	“general inverse geometric model”
*.ragn	“general inverse geometric model, numerical method”

Kinematic

*.jac	“Jacobian matrix”
*.Mjac	“Jacobian matrix in Mathematica form”
*.det	“determinant of Jacobian matrix”
*.inv	“inverse Jacobian”
*.vel	“velocities of the links in customized form”
*.acc	“acceleration of the links in customized form”
*.jpqp	“JpQp of links in customized form”
*.ckel	“constraint kinematic equations of loops”
*.Mckel	“constraint kinematic equations of loops in Mathematica form”
*.ickel	“Resolution of constraint kinematic equations of loops, matrix **W**”

Dynamic

*.dyn	"inverse dynamic model using Newton Euler method in customized form"
*.inm	"inertia matrix customized form"
*.lag	"dynamic coefficients of Lagrange method in customized form"
*.Elag	"dynamic coefficients of Lagrange method in expanded form"
*.ccg	"centrifuge, Coriolis and gravity forces in customized form"
*.ddm	"direct dynamic model in customized form"

Identification

*base.par	"robot parameters using the base inertial parameters, using symbolic method"
*.reg	"the expressions of regrouped inertial parameters, using symbolic method"
*Nbase.par	"robot parameters containing the base inertial parameters, using numerical method"
*.Nreg	"the expressions of regrouped inertial parameters, using numeric method"
*.eim	"energy identification model in customized form"
*.dim	"dynamic identification model in customized form"
*.fdm	"filtered dynamic identification model in customized form"

Optimizer

The name of the output file depends on the desired output form:

robot-name.**o,	for output file in listing form,
robot-name_**.f,	for output file in Fortran form,
robot-name_**.mh,	for output file in Mathematica form,
robot-name_**.m,	for output file in Matlab form,
robot-name_**.mp,	for output file in Maple form,
robot-name_**.c,	for output file in C form.

where ** means the extension of the input file which must contain a model in customized form.

References

[Aldon 86] M.J.Aldon , “Identification des paramètres structuraux des robots manipulateurs”, *Proc. Outils Mathématiques pour la Modélisation et la Commande des Robots* , Paris, sept. 1986, pp. 243-296.

[An 85] C.H.An , C.G.Atkeson , J.M.Hollerbach , “Estimation of inertial parameters of rigid body links of manipulators”, *Proc. 24th IEEE Conf. on Decision and Control*, Fort-Lauderdale, dec. 1985, pp. 990-995.

[Arimoto 84] S.Arimoto , F.Miyazaki , “Stability and robustness of PID feedback control for robots manipulators of sensory capability”, *The 1st Int. Symp. of Robotics Research*, MIT Press, 1984.

[Armstrong 79] W.W.Armstrong , “Recursive solution to the equation of motion of an N-links manipulator”, *Proc. 5th World Congress on Theory of Machines and Mechanisms, Montréal, 1979*, pp. 1343-1346.

[Armstrong 86] B.Armstrong, O.Khatib, J.Burdick, “The explicit dynamic model and inertial parameters of the PUMA 560 Arm”, *Proc. IEEE Conf. on Robotics and Automation*, San Francisco, april 1986, pp. 510-518.

[Armstrong 87] B.Armstrong , “On finding “exciting” trajectories for identification experiments involving systems with non-linear dynamics”, *Proc. IEEE Conf. on Robotics and Automation*, Raleigh, march-april 1987, pp. 1131-1139.

[Armstrong 88] B.Armstrong, “Dynamics for robot control:Friction modeling and ensuring excitation during parameter identification”, Ph.D thesis, Dept. of Electrical Engineering, Stanford University, May 1988.

[Armstrong 91] B.Armstrong,“Control of Machines with frictions”, Kluwer Academic Publishers, 1991.

[Atkeson 85] C.G.Atkeson , C.H.An, J.M.Hollerbach , “Rigid body load identification for manipulators”, *Proc. 24th IEEE Conf. on Decision and Control*, Fort-Lauderdale, dec. 1985, pp. 996-1002.

[Aubin 91] A.Aubin, “Modélisation, identification, et commande du bras manipulateur TAM”, thèse de doctorat, INPG, Grenoble Nov. 1991.

[Baillieul 84] J.Baillieul, J.M.Hollerbach , R.Brockett , “Programming and control of kinematically redundant manipulators”, *Proc. 23rd Conf. on Decision and Control*, Las Vegas, dec.1984, pp. 768-774.

[Baillieul 85] J.Baillieul , “Kinematic programming alternatives for redundant manipulators”, *Proc. IEEE Conf. on Robotics and Automation*, St Louis, march 1985, pp. 722-728.

[Bejczy 74] A.K.Bejczy, “Robot arm dynamics and control”, NASA Technical Memorandum 33-669, Jet Propulsion Laboratory, Pasadena, 1974.

[Bennis 90] F Bennis, W.Khalil, “Minimum inertial parameters of robots with parallelogram closed-loops”, Proc. IEEE Conf. on Robotics and Automation, Cincinnati, pp. 1026-1031, 1990.

[Bennis 91a] F.Bennis, W.Khalil, “Minimum inertial parameters of robots with parallelogram closed-loops”, IEEE Trans. Syst. Man and Cybernetics, Vol.21, N° 2, pp.318-326. March/April 1991.

[Bennis 91b] F.Bennis, “Contribution à la modélisation géométrique et dynamique des robots à chaîne simple et complexe”, Thèse de doctorat, E.N.S.M, Nantes, 1991.

[Bennis 92a] F. Bennis, W. Khalil, M. Gautier, “Calculation of the Base Inertial Parameters of Closed-loops Robots”, IEEE Robotics and Automation Conference, Nice, May 1992.

[Bennis 93] F.Bennis, W.Khalil, “Modèle géométrique Inverse des robots à chaîne découplable: Application aux equations de containtes des boucles fermées”, Transactions of the Canadian Society for Mechanical Engineering, Vol. 17, n° 4, pp. 473-492, Dec. 1993.

[Borrel 86] P.Borrel, “Contribution à la modélisation géométrique des robots-manipulateurs ; application à la conception assistée par ordinateur”, Thèse d'Etat, USTL, Montpellier, 1986.

[Bouzouia 89] B.Bouzouia, “Commande de robots manipulateurs: identification des paramètres et étude de stratégies adaptatives”, Thèse de Doctorat, LAAS, Toulouse, 1989.

[Brandl 86] H.Brandl, R. Johanni, 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, December 1986, pp. 356-362.

[Chace 67] M.A.Chace, “Analysis of the time dependance of multi-freedoom mechanical system in relative coordinate”, *Trans. of ASME, J. of Engineering for Industry*, Vol. 89 , feb. 1967, pp. 119-125.

[Chace 71] M.A.Chace, Y.O.Bayazitoglu, “Development and application of a generalized d'Alembert force for multi-freedom mechanical system” , *Trans. ASME, J. of Engineering for Industry*, Vol. 93 , feb. 1971, pp. 317-327.

[Chang 86] P.H.Chang , “A closed form solution for the control of manipulators with kinematic redundancy”, *Proc. IEEE Conf. on Robotics and Automation*, San Francisco, april 1986, pp. 9-14.

[Chedmail 86a] P.Chedmail,M.Gautier,“Simulation de robot et choix optimum des actionneurs”, *Proc. IMACS-IFAC Symp.*, Villeneuve D'Ascq, june 1986, pp. 531-536.

[Chedmail 86b] P.Chedmail, Gautier M., Khalil W., “Automatic modelling of robots including parameters of links and actuators”, *Proc. IFAC Symp. on Theory of Robots*, Vienne, dec.1986, pp. 295-299.

[Chevallereau 87] C.Chevallereau, W.Khalil , “Efficient method for the calculation of the pseudo inverse kinematic problem”, *Proc. IEEE Conf. on Robotics and Automation*, Raleigh, march-april 1987, pp. 1842-1848.

[Chevallereau 88a] C.Chevallereau , W.Khalil , “A new method for the solution of the inverse kinematics of redundant robots”, *Proc. IEEE Conf. on Robotics and Automation*, Philadelphia, april 1988, pp. 37-42.

[Chevallereau 88b] C.Chevallereau , “Contribution à la commande des robots-manipulateurs dans l'espace opérationnel”, Thèse de Doctorat , ENSM, Nantes, may 1988.

[Craig 86] J.J.Craig , *Introduction to robotics: mechanics and control*, Addison Wesley Publishing Company, Reading, 1986.

[Dahl 77] P.R.Dahl ,"Measurements of solid friction parameters of ball bearings", Proc. of the 6th Annual Symposium on Incremental Motion Control Systems and Devices, University of Illinois 1977.

[Denavit 55] Denavit J., Hartenberg R.S., "A kinematic notation for lower pair mechanism based on matrices", *Trans. of ASME, J. of Applied Mechanics*, Vol. 22, june 1955, pp. 215-221.

[Featherstone 83] R.Featherstone, "Position and velocity transformations between robot end-effector coordinates and joint angles", *The Int. J. of Robotics Research*, Vol. 2(2), 1983, pp. 35-45.

[Ferreira 84] E.P.Ferreira , "Contribution à l'identification de paramètres et à la commande des robots manipulateurs", Thèse de Docteur-Ingénieur, Université P. Sabatier, Toulouse, July. 1984.

[Fournier 80] A.Fournier , "Génération de mouvements en robotique ; application des inverses généralisées et des pseudo-inverses", Thèse d'Etat, USTL, Montpellier, april 1980.

[Freund 82] E.Freund , "Fast nonlinear control with arbitrary pole placement for industrial robots and manipulators", *The Int. J. of Robotics Research*, Vol. 1(1), 1982, pp. 65-78.

[Gautier 86] M.Gautier , "Identification of robots dynamics", *Proc. IFAC Symp. on Theory of Robots*, Vienne, déc. 1986, pp. 351-356.

[Gautier 88a] M.Gautier , W.Khalil , "A direct determination of minimum inertial parameters of robots", *Proc. IEEE Conf. on Robotics and Automation,* Philadelphia, april 1988, pp. 1682-1687.

[Gautier 88b] M.Gautier , W.Khalil , "On the identification of the inertial parameters of robots", *Proc. 27th IEEE Conf. on Decision and Control*, Austin, dec. 1988.

[Gautier 90a] M. Gautier, "Contribution à la modélisation et à l'identification des robots", Thèse de Doctorat d'Etat, ENSM, Nantes, 22 may, 1990.

[Gautier 90b] M. Gautier, W. Khalil, "Direct calculation of minimum set of inertial parameters of serial robots", IEEE Trans. on Robotics and Automation, Vol. 6, N^o 3, pp. 368-373, 1990.

[Gautier 91] M. Gautier, "Numerical calculation of the base inertial parameters", Journal of Robotics Systems, Vol. 8, N° 4, pp. 485-506, 1991.

[Gautier 92] M.Gautier, W.Khalil,"Exciting trajectories for inertial parameters identification", International Journal of Robotics Research, vol 11, n° 4, Aug 1992, pp. 362-375.

[Giordano 86] M.Giordano, "Dynamic model of robots with a complex kinematic chain", *Proc.16th Int. Symp. on Industrial Robots*, Bruxelles, sept.-oct. 1986, pp. 377-388.

[Gorla 84] Gorla B., Renaud M., *Modèles des robots-manipulateurs ; application à leur commande*, Cepadues Editions, Toulouse, 1984.

[Ha 89] I.J. Ha, M.S. Ko, S.K. Kwon, "An efficient estimation algorithm for the model parameters of robotics manipulators", IEEE Trans. on Robotics and Automation, Vol. 5, n^o 3, pp. 386-394, 1989.

[Held 88] V. Held, C.Maron, "Estimation of friction characteristics, inertial and coupling coefficients in robotic joints based on current and speed measurements", Proc. SYROCO'88, pp. 86.1-86.6, 1988.

[Hollerbach 80] J.M.Hollerbach , “An iterative Lagrangian formulation of manipulators dynamics and a comparative study of dynamics formulation complexity”, *IEEE Trans. on Systems, Man, and Cybernetics*, Vol. SMC-10(11), 1980, pp. 730-736.

[Hollerbach 85] J.M.Hollerbach , K.C.Suh , “Redundancy resolution of manipulators through torque optimization”, *Proc. IEEE Conf. on Robotics and Automation*, St Louis, march 1985, pp. 1016-1021

[Kanade 84] T.Kanade, P.Khosla , N.Tanaka, “Real-time control of the CMU direct-drive arm II using customized inverse dynamics” , *Proc. 23th IEEE Conf. on Decision and Control*, Las Vegas, dec.1984, pp. 1345-1352.

[Kawasaki 88] H. Kawasaki, K.Nishimura,“Terminal-link parameter estimation and trajectory control of robotic manipulators”, IEEE J. Robotics and Automation, Vol. RA-4(5), pp. 485-490, 1988.

[Khalil 76] W.Khalil , “Modélisation et commande par calculateur du manipulateur MA-23 ; extension à la conception par ordinateur des manipulateurs”, Thèse de Docteur-Ingénieur, USTL, Montpellier, sept. 1976.

[Khalil 78] W.Khalil, “Contribution à la commande automatique des manipulateurs avec l'aide d'un modèle mathématique des mécanismes”, Thèse d'Etat, USTL, Montpellier, oct. 1978.

[Khalil 79] W.Khalil, A.Liegeois , A.Fournier, “Commande dynamique des robots”, *Revue RAIRO Automatique / Systems Analysis and Control*, Vol. 13(2), 1979, pp. 189-201.

[Khalil 85a] W.Khalil, J.F.Kleinfinger , “Une modélisation performante pour la commande dynamique de robots”, *Revue RAIRO, APII*, Vol. 6, 1985, pp. 561-574.

[Khalil 85b] W.Khalil, J.F.Kleinfinger, “Nouvelles notations pour décrire la structure géométrique des robots”, Rapport Interne n° 85-05, LAN, Nantes, 1985.

[Khalil 85c] W.Khalil W., Gautier M., “On the derivation of the dynamic models of robots”, *Proc. Int. Conf. on Advanced Robotics, ICAR'85*, Tokyo, sept. 85, p. 243-250.

[Khalil 86a] W.Khalil W., Kleinfinger J.-F., “A new geometric notation for open and closed-loop robots”, *Proc. IEEE Conf. on Robotics and Automation*, San Francisco, april 1986, p. 1174-1180.

[Khalil 86b] W.Khalil W., “On the explicit derivation of the inverse geometric models of robots”, *Proc. IMACS-IFAC Symp.*, Villeneuve D'Ascq, june 1986, pp. 541-546 .

[Khalil 86c] W.Khalil W., Kleinfinger J.-F., Gautier M., “Reducing the computational burden of the dynamic model of robots”, *Proc. IEEE Conf. on Robotics and Automation*, San Francisco, april 1986, pp. 525-531.

[Khalil 87a] W.Khalil W., Kleinfinger J.-F., Chevallereau C., “SYMORO: système pour la modélisation des robots”, Notice d'utilisation, LAN, Nantes, feb. 1987.

[Khalil 87b] W.Khalil W., C.Chevallereau , “An efficient algorithm for the dynamic control of robots in the cartesian space”, *Proc. 26th IEEE Conf. on Decision and Control*, Los Angeles, dec. 1987, pp. 582-588.

[Khalil 87c] W.Khalil, J.-F.Kleinfinger, “Minimum operations and minimum parameters of the dynamic model of tree structure robots”, *IEEE J. of Robotics and Automation*, Vol. RA-3(6), dec. 1987, pp. 517-526.

[Khalil 89a] W.Khalil, “SYMORO: système pour la modélisation des robots”, Support technique-Notice d'utilisation, LAN, Nantes, 1989.

[Khalil 89b] W.Khalil, F.Bennis, C.Chevallereau, J.F. Kleinfinger, “SYMORO: A softword package for the symbolic modelling of robots.”, 20th ISIR, Tokyo, pp. 1023-1030, October 1989.

[Khalil 89c] W. Khalil, F. Bennis, M. Gautier, “Calculation of the minimum inertial parameters of tree structure robots”, Proc. 4ième ICAR, Columbus, USA, Springer-Verlag, pp. 189-201, 1989.

[Khalil 90a] W. Khalil, F. Bennis, M. Gautier, “The use of the generalized links to determine the minimum inertial parameters of robots”, J. of Robotic Systems, 7 (2), pp. 225-242, 1990.

[Khalil 90b] W. Khalil, F. Bennis, “Calcul de la matrice d'inertie des robots à chaîne ouverte simple ou arborescente”, Rapport interne n° 90-09, LAN-ENSM, April 1990.

[Khalil 91] W. Khalil, F. Bennis, “Automatic generation of the inverse geometric model of robots”, Journal of Robotics and Autonomous Systems, Vol. 7, pp. 1-10, 1991.

[Khalil 94a] W. Khalil, F. Bennis, “Comments on Direct Calculation of minimum set of inertial Parameters of Serial Robots” IEEE Trans. on Robotics and Automation, Vol.10, N° 1, February 94, pp. 78-79.

[Khalil 94b] W. Khalil, D. Murareci, “On the general solution of the inverse kinematics of six-degrees-of-freedom manipulators”, 4th international Workshop on advances in robot kinematics, ARK, Ljubljana, July 1994.

[Khalil 94c] W. Khalil, F. Bennis, “Symbolic calculation of the base inertial parameters of closed loop robots” International Journal of Robotics Research, Vol. RA-10(1), 1994, pp. 78-79.

[Khalil 96] Khalil W., Restrepo P.P., “An efficient algorithm for the calculation of the filtered dynamic model of robots”, *IEEE International Conference on Robotics and Automation*, pp. 323-329, ISBN: 0-7803-2988-4/96, April 1996, Minneapolis, MN, USA.

[Khalil 97] Khalil W., Creusot D., “SYMORO+: a system for the symbolic modelling of robots”, *Robotica*, Vol. 15, 1997, p. 153-161.

[Khalil 99a] Khalil W., Lemoine Ph., “Gecaro: a system for the geometric calibration of robots”, *Revue APII-JESA*, Vol. 33(5-6), 1999, p. 717-739.

[Khalil 99b] Khalil W., Besnard S., “Self calibration of Stewart-Gough parallel robots without extra sensors”, *IEEE Trans. on Robotics and Automation*, Vol. RA-15(6), p. 1116-1121, December 1999.

[Khalil 99c] W.Khalil, E. Dombre, “Modélisation, identification et commande des robots”, Editions Hermès, Paris, 1999.

[Khalil 00] W.Khalil , M.Gautier, “Modeling of mechanical systems with lumped elasticity”, *Proc. IEEE Int. Conf. on Robotics and Automation*, San Francisco, April 2000, p. 3965-3970.

[Khalil 02a] W.Khalil , E.Dombre, “Modeling, identification and control of robots”, Hermès Science Penton, London, 2002.

[Khalil 02b] W. Khalil, S. Guegan, “ A novel solution for the dynamic modeling of Gough-Stewart manipulators”, IEEE Robotics and Automation Conference, Washington, mai 2002.

[Khatib 80] O.Khatib, “Commande dynamique dans l'espace opérationnel des robots-manipulateurs en présence d'obstacles”, Thèse de Docteur-Ingénieur, Ecole Nationale Supérieure de l'Aéronautique et de l'Espace, Toulouse, Dec. 1980.

[Khosla 85] P.K.Khosla,T.Kanade, “Parameter identification of robot dynamics”, *Proc. 24th IEEE Conf. on Decision and Control*, Fort-Lauderdale, dec. 1985, pp. 1754-1760.

[Khosla 86] P.K.Khosla , “Real-time control and identification of direct drive manipulators”, Ph. D. Thesis, Carnegie Mellon University, Pittsburgh, 1986.

[Klein 84] C.A.Klein , “Use of redundancy in the design of robotic systems”, *Proc. 2nd Int. Symp. of Robotic Research*, Kyoto, 1984, pp. 58-65.

[Kleinfinger 86a] J.-F.Kleinfinger, “Modélisation dynamique de robots à chaîne cinématique simple, arborescente ou fermée, en vue de leur commande”, Thèse de Doctorat, ENSM, Nantes, may 1986.

[Kleinfinger 86b] J.-F.Kleinfinger, W.Khalil, “Dynamic modelling of closed-chain robots”, *Proc. 16th Int. Symp. on Industrial Robots*, Bruxelles, sept.-oct. 1986, pp. 401-412.

[Koditschek 84] D.E.Koditschek , “Natural motion for robot arms”, *Proc. 23rd Conf. on Decision and Control*, Las Vegas, dec. 1984, pp. 737-735.

[Llibre 83] M.Llibre, R.Mampey , J.P.Chrétien, “Simulation de la dynamique des robots manipulateurs motorisés”, *Congrès AFCET: Productique et Robotique Intelligente*, Besançon, nov. 1983, pp. 197-207.

[Luh 79] J.Y.S.Luh , M.W.Walker , R.C.P.Paul , “On-line computational scheme for mechanical manipulators”, *Proc. 2nd IFAC/IFIP Symp. on Information Control Problems in Manufacturing Technology*, Stuttgart, 1979, pp. 165-172.

[Luh 80a] J.Y.S.Luh , M.W.Walker , R.C.P.Paul , “Resolved-acceleration control of mechanical manipulators”, *IEEE Trans. on Automatic Control*, Vol. AC-25(3), june 1980, pp. 468-474.

[Luh 80b] J.Y.S.Luh , M.W.Walker , R.C.P.Paul , “On-line computational scheme for mechanical manipulators”, *Trans. of ASME, J. of Dynamic Systems, Measurement, and Control*, Vol. 102(2), 1980, p. 69-76.

[Luh 85] J.Y.S.Luh , Y.F.Zheng, “Computation of input generalized forces for robots with closed kinematic chain mechanisms”, *IEEE J. of Robotics and Automation*, Vol. RA-1(2), 1985, pp. 95-103.

[Mayeda 84] H.Mayeda , K.Osuka , A.Kangawa , “A new identification method for serial manipulator arms”, *Proc. IFAC 9th World Congress*, Budapest, july. 1984, pp. 74-79.

[Mayeda 88] H. Mayeda, K. Yoshida, K. Osuka, “Base parameters of manipulator dynamics”, Proc. IEEE Conf. on Robotics and Automation, pp. 1367-1373, 1988.

[Mavroidis 93] C. Mavroidis, “ Résolution du problème géométrique inverse pour les manipulaterurs série à 6 d.d.l”, Thèse de doctorat, Paris 6, May 1993.

[Megahed 82] S.Megahed, M.Renaud, “Minimization of the computation time necessary for the dynamic control”, *12th Int. Symp. on Industrial Robots*, Paris, june 1982, pp. 469-478.

[Megahed 84] S.Megahed, “Contribution à la modélisation géométrique et dynamique des robots manipulateurs ayant une structure de chaîne cinématique simple ou complexe ; application à leur commande”, Thèse d'Etat, Université P. Sabatier, Toulouse, july. 1984.

[Murphy 93] Murphy S.H., Wen J.T.U, “Analysis of active manipulator elements in space manipulation”, *IEEE Trans. on Robotics and Automation*, Vol. RA-9(5), October 1993, p. 544-552.

[Olsen 86] H.B.Olsen , G.A.Bekey , “Identification of robot dynamics”, *Proc. IEEE Conf. on Robotics and Automation*, San Francisco, april 1986, pp. 1004-1010.

[Paul 81] R.C.P.Paul , *Robot manipulators: mathematics, programming and control*, MIT Press, 1981.

[Penrose 55] R.Penrose , “A generalized inverse for matrices”, *Proc. Cambridge Philos. Soc.*, Vol. 51, 1955, pp. 406-413.

[Pieper 68] D.L.Pieper , “The kinematics of manipulators under computer control”, Ph. D. Thesis, Stanford University, Stanford, 1968.

[Potkonjak 86] V.Potkonjak , “Thermal criterion for the selection of DC drives for industrial robots”, *Proc. 16th Int. Symp. on Industrial Robots*, Bruxelles, sept.-oct. 1986, pp.129-140.

[Raghavan 89] M. Raghavan, B. Roth, “Kinematic analysis of the 6R manipulator of general geometry”, The 5 th Int.Symp.on Robotics Research, Tokyo, 1989.

[Raghavan 90] M.Raghavan, B.Roth , “A General Solution for the Inverse Kinematics of all chains”, Romansy 90, Cracow, Poland 1990 .

[Raibert 78] M.H.Raibert, B.K.P.Horn , “Manipulator control using the configuration space method”, *The Industrial Robot*, Vol. 5(2), 1978, pp. 69-73.

[Renaud 75] M.Renaud , “Contribution à l'étude de la modélisation et de la commande des systèmes mécaniques articulés”, Thèse de Docteur-Ingénieur, Université P. Sabatier, Toulouse, dec. 1975.

[Renaud 80a] M.Renaud , “Contribution à la modélisation et à la commande dynamique des robots manipulateurs”, Thèse d'Etat, Université P. Sabatier, Toulouse, sept. 1980.

[Renaud 80b] M.Renaud , “Calcul de la matrice jacobienne nécessaire à la commande coordonnée d'un manipulateur”, *J. of Mechanism Machine Theory*, Vol. 15(1), 1980, pp. 81-91.

[Renaud 85] M.Renaud , “A near minimum iterative analytical procedure for obtaining a robot-manipulator dynamic model”, *IUTAM/IFToMM Symp. on Dynamics of Multi-body Systems*, Udine, 1985.

[Renaud 87] M.Renaud, “Quasi-minimal computation of the dynamic model of a robot manipulator utilizing the Newton-Euler formalism and the notion of augmented body”, *Proc. IEEE Conf. on Robotics and Automation*, Raleigh, march-april 1987, pp. 1677-1682.

[Samson 87a] C.Samson ,“ Une approche pour la synthèse et l'analyse de la commande des robots manipulateurs rigides”, Rapport INRIA n° 669, Rennes, 1987.

[Samson 87b] C. Samson, “Robust control of a class of non-linear systems and applications to robotics”.Int.Journal of adaptive control and signal processing, Vol.1,1987, pp. 49-68.

[Samson 91] C.Samson, M.Le Borgne, B.Espiau, *Robot Control*, Oxford University Press, Oxford, 1991.

[Sciavicco 94] Sciavicco L., Siciliano B., Villani L., “On dynamic modelling of gear-driven rigid robot manipulators”, *Proc. 4th IFAC Symp. on Robot Control, SYROCO'94*, Capri, Italy, September 1994, p. 543-549.

[Sheth 71] P.N.Sheth, J.J.Uicker, “A generalized symbolic notation for mechanism”, *Trans. of ASME, J. of Engineering for Industry*, Vol. 93, 1971, pp. 102-112.

[Smith 73] D.A.Smith, M.A.Chace, A.C.Rubens , “The automatic generation of a mathematical model for machinery systems”, *Trans. of ASME, J. of Engineering for Industry*, 1973, Vol. 95, pp. 629-635.

[Touron 84] P.Touron, “Modélisation de la dynamique des mécanismes polyarticulés ; application à la CAO et à la simulation de robots”, Thèse de Docteur-Ingénieur, USTL, Montpellier, july. 1984.

[Uicker 69] J.J.Uicker, “Dynamic behavior of spatial linkages”, *Trans. of ASME, J. of Engineering for Industry*, Vol. 91, 1969, pp. 251-258.

[Walker 82] M.W.Walker, D.E.Orin , “Efficient dynamic computer simulation of robotics mechanism”, *Trans. of ASME, J. of Dynamic Systems, Measurement, and Control*, Vol. 104, 1982, pp. 205-211.

[Wittenburg 77] J.Wittenburg, *Dynamics of system of rigid bodies*, B.G. Teubner, Stuttgart, 1977.

[Yoshikawa 84] T.Yoshikawa, “Analysis and control of robot manipulators with redundancy”, *The 1st Int. Symp. of Robotics Research*, MIT Press, 1984, pp. 735-748.

[Zabala 78] Iturralde J.Zabala, “Commande des robots-manipulateurs à partir de la modélisation de leur dynamique”, Thèse de Troisième Cycle, Université P. Sabatier, Toulouse, july. 1978

[Zghaib 92] W.Zghaib,“Génération Symbolique Automatique des Equations de la Dynamique des Systèmes mécaniques Complexes avec Contraintes Cinématiques”. ENSAM, Paris 1992.