

Document downloaded from:

<http://hdl.handle.net/10251/202038>

This paper must be cited as:

Valera Fernández, Á.; Mata Amela, V.; Vallés Miquel, M.; Valero, F.; Rosillo, N.; Benimeli, F. (2003). Solving the Inverse Dynamic Control for Low Cost Real-Time Industrial Robot Control Applications. *Robotica*. 21(3):261-269. <https://doi.org/10.1017/S0263574702004769>



The final publication is available at

<https://doi.org/10.1017/S0263574702004769>

Copyright Cambridge University Press

Additional Information

# Solving the Inverse Dynamic Control for Low Cost Real-Time Industrial Robot Control Applications

Valera, A. <sup>†</sup>; Mata, V. <sup>††</sup>; Vallés, M. <sup>†</sup>; Valero, F. <sup>††</sup>, Rosillo, N. <sup>††</sup>, Benimeli, F. <sup>††</sup>

<sup>†</sup> Departamento de Ingeniería de Sistemas y Automática

<sup>††</sup> Departamento de Ingeniería Mecánica y de Materiales  
Universidad Politécnica de Valencia (SPAIN)

**Abstract.** This work deals with the real-time robot control implementation. In this paper, one algorithm for solving Inverse Dynamic Problem based on the Gibbs-Appell equations is proposed and verified. It is developed using mainly vectorial variables and the equations are expressed in a recursive form, and it has a computational complexity of  $O(n)$ . This algorithm will be compared with one based on Newton-Euler equations of motion, formulated in a similar way, and using mainly vectors in its recursive formulation. This algorithm was implemented in an industrial robot: the PUMA manipulator. For the robot control a new and open architecture based on PC had been implemented. The architecture used has two main advantages: first it provides a total open control architecture, and second it is not expensive. Because the controller is based on PC, any control technique can be programmed and implemented, and in this way the PUMA can work on high level tasks such as automatic trajectory generation, task planning, control by artificial vision, etc.

**Keywords:** Robotic Manipulators, Inverse Dynamic Problem, Gibbs-Appell formulation, Robot Control, Computer Control, Digital Computer Applications

## 1. Introduction

One of the main problems that can be found when attempting to establish the control of an industrial robotic system is in its own control unit, since this unit is generally a totally closed subsystem. Due to the fact that it uses its own operating system and since it is impossible to modify this control system (not even the gain values), users have serious problems to implement conventional as well as advanced control strategies such force control, cooperative control of several robots. It is also important programming automatic trajectory generation, control based on external sensing (such as vision), control strategies etc.

In order to solve these kind of problems we can find some solutions at (Moreira, *et al.*, 1996), where the control hardware is modified, but generally it is neither trivial nor a cheap task. In (Valera, *et al.*, 1998) shows a solution for these drawbacks since it presents a very simple, economical and total open architecture for the robot control.

This paper shows the real-time robot control using this architecture. The control algorithms are based on the Inverse Dynamic Problem (IDP) implementation using the Gibbs and Appell notation. Literature about the IDP in robots is vast. The interest of its potential applications (verification that the torques needed to execute a proposed trajectory do not exceed the capabilities of the actuators, IDP as part of the Inverse Dynamics Control etc.) has contributed to it.

In order to increase their computational efficiency, many algorithms for solving the IDP have been proposed in the last thirty years. These algorithms are based on different Principles of Dynamics (Lagrange, Newton-Euler, Kane), the equations of motion are expressed in a closed-form or recursive formulation, and using different types of variables to express physical quantities. These algorithms can be implemented by means of symbolic programs or strictly numerical ones. Finally, according with the computer architecture where they will be processed, the algorithms can be sequential or parallel.

In the following table are shown some of the proposed algorithms for solving the IDP on robots with rigid links and ideal joints.

Table 1. Several algorithms for solving the IDP

Authors	Dynamic Principle	Formulation	Type of variables	Type of resolution	Type of processing
(Hollerbach, 1980)	Lagrange-Euler	Recursive	Matricial	Numerical	Sequential
(Luh et al, 1980)	Newton-Euler	Recursive	Vectorial	Numerical	Sequential
(Angeles, et al, 1989)	Kane	Recursive	Tensorial	Numerical	Sequential
(Balafoutis and Patel, 1991)	Newton-Euler	Recursive	Tensorial	Numerical	Sequential
(Khalil and Kleifinger, 1987)	Newton-Euler	Recursive	Vectorial	Symbolic	Sequential
(Lee and Chang, 1986)	Newton-Euler	Recursive	Vectorial	Numerical	Parallel

Custom-made algorithms, which take advantage of the special characteristics of particular industrial robots, must be specially mentioned. Examples of these are proposed in (Murray and Newman, 1988), based on the Newton-Euler formulation, expressed in a recursive way and with vectorial variables.

Several authors (Balafoutis and Patel, 1991) have made evident that the computational efficiency of the dynamic algorithms depends fundamentally on the way the calculations are arranged rather than the dynamic principle in which they are based. This idea was already proposed by (Hollerbach, 1980), in which the dynamic problem was reformulated for robots by using the Principle of Lagrange in a recursive way and using rotation matrix 3x3 instead of a 4x4 homogeneous transformation matrix. By this method, the computational complexity could be reduced from  $O(n^4)$  to  $O(n)$ . Nevertheless, the computational complexity of the Hollerbach algorithm was as three times larger than the Luh, Walker and Paul algorithm.

On the other hand, it must be pointed out that important differences can be noticed about the computational efficiency assigned to algorithms of the same characteristics applied to robots of the same type. An algorithm based on Newton-Euler formulation, implemented in a recursive way, using vectorial variables and solved in a numerical and sequential way, can be found in (Luh, et al, 1980) which had been assigned by Hollerbach a computational complexity of  $150n-48$  multiplications and  $131n-48$  additions, where  $n$  is the number of degrees of freedom of the robot. (Fu, et al, 1987) provided a version of the same algorithm with a complexity of  $117n-24$  multiplications and  $103n-21$  additions. (Zomaya, 1992) gave a complexity of  $150n$  multiplications and  $116n$  additions. Finally, (Craig, 1986) gave a complexity of  $126n-99$  multiplications and  $106n-92$  additions.

The observed differences come fundamentally from the criteria used for counting operations, for instance, if operations, that involve multiplications by variables with 0 or 1 values, are detected. Therefore, it seems necessary to indicate clearly the criteria that are going to be used for counting the operations when comparing the efficiency of algorithms.

The Gibbs-Appell equations were introduced by Gibbs in 1879 and formalised by Appell twenty years later. However, in the robot dynamics field there are few published references to works based on them. (Renaud, 1975) stands out among the first references to the application of the Gibbs-Appell equations to dynamic modelling of robots, in which he made remarkable commentaries on the previous work of E.P. Popov. (Vukobratovic and Kircanski, 1985) developed a closed-form algorithm with  $O(n^3)$  computational complexity. (Desoyer and Lugner, 1989) developed a recursive algorithm for solving the IDP in robots using the Jacobian matrix in order to avoid algebraic or numerical derivatives. The computational complexity of the proposed algorithm is  $O(n^3)$ .

In the present work the Gibbs-Appell equations are applied to solving the inverse dynamic problems of robots that have rigid links and ideal pairs. The algorithm proposed has a computational complexity of  $O(n)$ . This algorithm is formulated in a recursive way, using vectors to express most of the physical magnitudes involved in them (angular velocity, angular acceleration, etc.). In order to achieve a higher computational efficiency, the involved magnitudes in the Gibbs function are expressed with respect to local reference systems in the links. The computational efficiency of this algorithm will be compared with that of the Luh, Walker and Paul algorithm. That can be done since the same type of formulation is used in both (recursive) and the common physical magnitudes are

expressed in the same way. It must be stated that the criteria to evaluate the number of operations will be the same in both algorithms.

This paper is organised as follows. In section 2, the proposed algorithm is developed and an analysis, comparing them with the Luh, Walker and Paul algorithm, of its computational complexity is provided. Section 3 presents a proposed control architecture for industrial robots based on PC. In section 4, the inverse dynamic control is addressed and is applied to a PUMA-type industrial robot. Finally, section 5 summarises the development of the paper and suggest directions for future research.

## 2. The Gibbs-Appel Formulation Applied to the Inverse Dynamic Problem in Robots

In this section, the Gibbs-Appell equations are described, and two different formulations are presented to solve the Inverse Dynamic Problem on robot manipulators. The robots are modelled following the Denavit-Hartenberg modified notation, which considers four parameters  $\theta_i$ ,  $\alpha_i$ ,  $a_i$  and  $d_i$ , as it is shown in Figure 1. In the mentioned notation, the reference system corresponding to link  $i$  is located on joint  $i$ , and the z-axis is located on the axis in the same node, which connects links  $i-1$  and  $i$ .

The reference system  $i$  is related to the  $i-1$  reference system by means of the rotation matrix  ${}^{i-1}R_i$  and the position vector  ${}^{i-1}\vec{r}_{O_{i-1},O_i}$ :

$${}^{i-1}R_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 \\ \cos\alpha_i \cdot \sin\theta_i & \cos\alpha_i \cdot \cos\theta_i & -\sin\alpha_i \\ \sin\alpha_i \cdot \sin\theta_i & \sin\alpha_i \cdot \cos\theta_i & \cos\alpha_i \end{bmatrix} \quad {}^{i-1}\vec{r}_{O_{i-1},O_i} = \begin{bmatrix} a_i \\ d_i \sin\alpha_i \\ -d_i \cos\alpha_i \end{bmatrix}$$

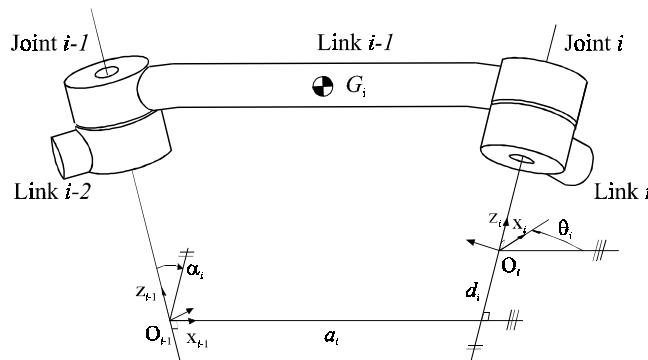


Figure 1. Modified Denavit-Hartenberg notation

The Gibbs-Appell dynamic equations come from the Gibbs function definition (also known as the energy of the accelerations). When written in its original form for an arbitrary solid composed of  $n$ -elemental particles with masses  $m_i$  an acceleration  $a_i$  is (considering an inertial reference system):

$$G = \frac{1}{2} \sum_{i=1}^n m_i a_i^2$$

The Gibbs function for the  $i$ -th rigid solid is given by

$$G_i = \frac{1}{2} m_i (\ddot{\vec{r}}_{G_i})^T \cdot \ddot{\vec{r}}_{G_i} + \frac{1}{2} (\dot{\vec{\omega}}_i)^T \cdot \mathbf{I}_{G_i} \cdot \dot{\vec{\omega}}_i + (\dot{\vec{\omega}}_i)^T \cdot [\vec{\omega}_i \wedge (\mathbf{I}_{G_i} \cdot \vec{\omega}_i)] \quad (1)$$

where  $m_i$  is the mass of the  $i$ -th link,  $\mathbf{I}_{G_i}$  is the 3x3 matrix representing the centroidal matrix of inertia of the  $i$ -th link,  $\vec{\omega}_i$  and  $\dot{\vec{\omega}}_i$  are the three-dimensional vectors representing the angular velocity and acceleration of the  $i$ -th link and  $\ddot{\vec{r}}_{G_i}$  is the three-dimensional vector representing the acceleration of the mass centre of the  $i$ -th link. An inertial reference system is considered to express these magnitudes.

It is possible for these tensorial and vectorial magnitudes to be expressed considering a reference system located in the  $i$ -th link, so that the previous expression could be expressed as follows:

$$G_i = \frac{1}{2} m_i ({}^0 \mathbf{R}_i \cdot {}^i \ddot{\mathbf{r}}_{G_i})^T \cdot {}^0 \mathbf{R}_i \cdot {}^i \ddot{\mathbf{r}}_{G_i} + \frac{1}{2} ({}^0 \mathbf{R}_i \cdot {}^i \dot{\bar{\omega}}_i)^T \cdot {}^0 \mathbf{R}_i \cdot {}^i \mathbf{I}_{G_i} \cdot ({}^0 \mathbf{R}_i)^T \cdot ({}^0 \mathbf{R}_i \cdot {}^i \dot{\bar{\omega}}_i) + ({}^0 \mathbf{R}_i \cdot {}^i \dot{\bar{\omega}}_i)^T \cdot \left\{ {}^0 \mathbf{R}_i \cdot {}^i \bar{\omega}_i \wedge \left[ {}^0 \mathbf{R}_i \cdot {}^i \mathbf{I}_{G_i} \cdot ({}^0 \mathbf{R}_i)^T \cdot {}^0 \mathbf{R}_i \cdot {}^i \bar{\omega}_i \right] \right\} \quad (2)$$

the expression (2) could be rewritten as follows:

$$G_i = \frac{1}{2} m_i ({}^i \ddot{\mathbf{r}}_{G_i})^T \cdot ({}^0 \mathbf{R}_i)^T \cdot {}^0 \mathbf{R}_i \cdot {}^i \ddot{\mathbf{r}}_{G_i} + \frac{1}{2} ({}^i \dot{\bar{\omega}}_i)^T \cdot ({}^0 \mathbf{R}_i)^T \cdot {}^0 \mathbf{R}_i \cdot {}^i \mathbf{I}_{G_i} \cdot ({}^0 \mathbf{R}_i)^T \cdot {}^0 \mathbf{R}_i \cdot {}^i \dot{\bar{\omega}}_i + ({}^i \dot{\bar{\omega}}_i)^T \cdot ({}^0 \mathbf{R}_i)^T \cdot \left\{ {}^0 \mathbf{R}_i \cdot {}^i \bar{\omega}_i \wedge \left[ {}^0 \mathbf{R}_i \cdot {}^i \mathbf{I}_{G_i} \cdot ({}^0 \mathbf{R}_i)^T \cdot {}^0 \mathbf{R}_i \cdot {}^i \bar{\omega}_i \right] \right\} \quad (3)$$

and taking into account the orthogonal nature of the rotation matrix, the scalar  $G_i$  would be given by:

$$G_i = \frac{1}{2} m_i ({}^i \ddot{\mathbf{r}}_{G_i})^T \cdot {}^i \ddot{\mathbf{r}}_{G_i} + \frac{1}{2} ({}^i \dot{\bar{\omega}}_i)^T \cdot {}^i \mathbf{I}_{G_i} \cdot {}^i \dot{\bar{\omega}}_i + ({}^i \dot{\bar{\omega}}_i)^T \cdot \left[ {}^i \bar{\omega}_i \wedge {}^i \mathbf{I}_{G_i} \cdot {}^i \bar{\omega}_i \right] \quad (4)$$

where  ${}^i \ddot{\mathbf{r}}_{G_i}$ ,  ${}^i \bar{\omega}_i$ ,  ${}^i \dot{\bar{\omega}}_i$  and  ${}^i \mathbf{I}_{G_i}$  are expressed in the  $i$ -th reference system.

For a system consisting of  $n$ -bodies, the Gibbs function of the system would be given by

$$G = \sum_{i=1}^n G_i \quad (i = 1, 2 \dots n) \quad (5)$$

The Gibbs-Appell equations of motion are obtained from deriving the Gibbs function with respect to the generalised accelerations  $\ddot{q}_j$ , obtaining in this way the generalised inertial forces that are to equate to the generalised external forces,  $\tau_j$

$$\tau_j = \sum_{i=j}^n \frac{\partial G_i}{\partial \ddot{q}_j} \quad (j = 1, 2 \dots n) \quad (6)$$

that is,

$$\tau_j = \sum_{i=j}^n \left\{ m_i ({}^i \ddot{\mathbf{r}}_{G_i})^T \cdot \frac{\partial {}^i \ddot{\mathbf{r}}_{G_i}}{\partial \ddot{q}_j} + ({}^i \dot{\bar{\omega}}_i)^T \cdot {}^i \mathbf{I}_{G_i} \cdot \frac{\partial {}^i \dot{\bar{\omega}}_i}{\partial \ddot{q}_j} + \left( \frac{\partial {}^i \dot{\bar{\omega}}_i}{\partial \ddot{q}_j} \right)^T \cdot \left[ {}^i \bar{\omega}_i \wedge ({}^i \mathbf{I}_{G_i} \cdot {}^i \bar{\omega}_i) \right] \right\} \quad (7)$$

The formulation for the solution of the Inverse Dynamic Problem in robots would be obtained by reorganizing and identifying two different terms in expression (7) as follows:

$$\tau_j = \underbrace{\sum_{i=j}^n \left\{ \left( \frac{\partial {}^i \dot{\bar{\omega}}_i}{\partial \ddot{q}_j} \right)^T \cdot \left[ {}^i \mathbf{I}_{G_i} \cdot {}^i \dot{\bar{\omega}}_i + {}^i \bar{\omega}_i \wedge ({}^i \mathbf{I}_{G_i} \cdot {}^i \bar{\omega}_i) \right] \right\}}_{A_j} + \underbrace{\sum_{i=j}^n \left\{ \left( \frac{\partial {}^i \ddot{\mathbf{r}}_{G_i}}{\partial \ddot{q}_j} \right)^T \cdot m_i \cdot {}^i \ddot{\mathbf{r}}_{G_i} \right\}}_{B_j} \quad (8)$$

It is remarkable that expression below is similar to the proposed by (Angeles, et al, 1989) for solving the IDP based on the Kane's dynamic formulation.

For obtaining the generalised forces, the angular velocities, angular accelerations, the accelerations of the origin of reference system of links and the accelerations of the centre of masses of the links can be obtained using the following known recursive expressions:

$${}^i \bar{\omega}_i = {}^i R_{i-1} \cdot {}^{i-1} \bar{\omega}_{i-1} + {}^i \dot{\mathbf{z}}_i \cdot \dot{q}_i \quad (9)$$

$${}^i \dot{\bar{\omega}}_i = {}^i R_{i-1} \cdot {}^{i-1} \dot{\bar{\omega}}_{i-1} + {}^i \bar{z}_i \cdot \ddot{q}_i + {}^i R_{i-1} \cdot {}^{i-1} \bar{\omega}_{i-1} \wedge ({}^i \bar{z}_i \cdot \dot{q}_i) \quad (10)$$

$${}^i \ddot{\bar{r}}_{O_i} = {}^i R_{i-1} \left[ {}^{i-1} \ddot{\bar{r}}_{O_{i-1}} + {}^{i-1} \bar{\omega}_{i-1} \wedge ({}^{i-1} \bar{\omega}_{i-1} \wedge {}^{i-1} \bar{r}_{O_{i-1}, O_i}) + {}^{i-1} \dot{\bar{\omega}}_{i-1} \wedge {}^{i-1} \bar{r}_{O_{i-1}, O_i} \right] \\ + (1 - \rho_i) \left[ {}^i \bar{z}_i \cdot \ddot{q}_i + 2({}^i \bar{\omega}_i \wedge {}^i \bar{z}_i \cdot \dot{q}_i) \right] \quad (11)$$

$${}^i \ddot{\bar{r}}_{G_i} = {}^i \ddot{\bar{r}}_{O_i} + {}^i \bar{\omega}_i \wedge ({}^i \bar{\omega}_i \wedge {}^i \bar{r}_{O_i, G_i}) + {}^i \dot{\bar{\omega}}_i \wedge {}^i \bar{r}_{O_i, G_i} \quad (12)$$

where  ${}^i \bar{z}_i = [0 \ 0 \ 1]^T$ , and the variable  $\rho_i$  allows us to distinguish between the revolute joints ( $\rho_i=1$ ) and the prismatic ones ( $\rho_i=0$ ).

Developing the  $A_j$  term:

$$A_j = \sum_{i=j}^n \left\{ \left( {}^i R_j \cdot \frac{\partial {}^j \dot{\bar{\omega}}_j}{\partial \ddot{q}_j} \right)^T \cdot \left[ {}^i I_{G_i} \cdot {}^i \dot{\bar{\omega}}_i + {}^i \bar{\omega}_i \wedge ({}^i I_{G_i} \cdot {}^i \bar{\omega}_i) \right] \right\} \quad (13)$$

To obtain the  $\frac{\partial {}^i \dot{\bar{\omega}}_i}{\partial \ddot{q}_j}$  term, we start from expression (10). This derivative could be obtained by a recursive procedure as follows:

$$\begin{aligned} \text{If } i < j \quad \frac{\partial {}^i \dot{\bar{\omega}}_i}{\partial \ddot{q}_j} &= [0 \ 0 \ 0]^T \\ \text{If } i > j \quad \frac{\partial {}^i \dot{\bar{\omega}}_i}{\partial \ddot{q}_j} &= {}^i R_{i-1} \cdot {}^{i-1} R_{i-2} \cdots {}^{j+1} R_j \cdot \frac{\partial {}^j \dot{\bar{\omega}}_j}{\partial \ddot{q}_j} \\ \text{If } i = j \quad \frac{\partial {}^i \dot{\bar{\omega}}_i}{\partial \ddot{q}_j} &= {}^i \bar{z}_i \end{aligned} \quad (14)$$

Taking into account the expressions in (14), the  $A_j$  term could be rewritten as follows:

$$A_j = \left( \frac{\partial {}^j \dot{\bar{\omega}}_j}{\partial \ddot{q}_j} \right)^T \cdot \sum_{i=j}^n \left\{ {}^j R_i \cdot \left[ {}^i I_{G_i} \cdot {}^i \dot{\bar{\omega}}_i + {}^i \bar{\omega}_i \wedge ({}^i I_{G_i} \cdot {}^i \bar{\omega}_i) \right] \right\} \quad (15)$$

In this last expression, it can be seen that there are concurrent terms which could reduce the calculation complexity. Next, an expression that allows the terms to be obtained in a backward recursive way is presented:

$$A_j = \left( \frac{\partial {}^j \dot{\bar{\omega}}_j}{\partial \ddot{q}_j} \right)^T \cdot {}^j \bar{\alpha}_j \quad (16)$$

where

$${}^j \bar{\alpha}_j = {}^j I_{G_j} \cdot {}^j \dot{\bar{\omega}}_j + {}^j \bar{\omega}_j \wedge ({}^j I_{G_j} \cdot {}^j \bar{\omega}_j) + {}^j R_{j+1} \cdot {}^{j+1} \bar{\alpha}_{j+1} \quad (17)$$

$B_j$  term can be obtained using the development of  $\frac{\partial {}^i \ddot{\bar{r}}_{G_i}}{\partial \ddot{q}_j}$ . This term comes from deriving expression (12), obtaining

$$\frac{\partial {}^i \ddot{\vec{r}}_{G_i}}{\partial \ddot{q}_j} = \frac{\partial {}^i \ddot{\vec{r}}_{O_i}}{\partial \ddot{q}_j} + \frac{\partial {}^i \dot{\vec{\omega}}_i}{\partial \ddot{q}_j} \wedge {}^i \vec{r}_{O_i, G_i} \quad (18)$$

The  $\frac{\partial {}^i \ddot{\vec{r}}_{O_i}}{\partial \ddot{q}_j}$  term for revolute joints is obtained using the expression (11):

$$\frac{\partial {}^i \ddot{\vec{r}}_{O_i}}{\partial \ddot{q}_j} = {}^i \mathbf{R}_{i-1} \cdot \left( \frac{\partial {}^{i-1} \ddot{\vec{r}}_{O_{i-1}}}{\partial \ddot{q}_j} + \frac{\partial {}^{i-1} \dot{\vec{\omega}}_{i-1}}{\partial \ddot{q}_j} \wedge {}^{i-1} \vec{r}_{O_{i-1}, O_i} \right) \quad (19)$$

In a similar way, for prismatic joints we could obtain:

$$\frac{\partial {}^i \ddot{\vec{r}}_{O_i}}{\partial \ddot{q}_j} = {}^i \mathbf{R}_{i-1} \cdot \frac{\partial {}^{i-1} \ddot{\vec{r}}_{O_{i-1}}}{\partial \ddot{q}_j} + {}^i \mathbf{R}_{i-1} \cdot \frac{\partial {}^{i-1} \dot{\vec{\omega}}_{i-1}}{\partial \ddot{q}_j} \wedge {}^{i-1} \vec{r}_{O_{i-1}, O_i} + {}^i \vec{z}_i \cdot \frac{\partial \ddot{q}_i}{\partial \ddot{q}_j} \quad (20)$$

Notice that there is an additional term in expression (20) in relation with (19), which must be included if the  $i$ -th joint is a prismatic one:

$${}^i \vec{z}_i \cdot \frac{\partial \ddot{q}_i}{\partial \ddot{q}_j} \quad (21)$$

To develop the  $B_j$  term, considering expressions (18) and (19), the following expression is used

$$\frac{\partial {}^i \ddot{\vec{r}}_{G_i}}{\partial \ddot{q}_j} = \frac{\partial {}^i \dot{\vec{\omega}}_i}{\partial \ddot{q}_j} \wedge {}^i \vec{r}_{O_{j-1}, G_i}$$

This expression, when substituted in the above description of every  $B_j$  term, would give:

$$B_j = \sum_{i=j}^n \left\{ \left( \frac{\partial {}^i \dot{\vec{\omega}}_i}{\partial \ddot{q}_j} \right)^T \cdot \left( {}^i \vec{r}_{O_{j-1}, G_i} \wedge m_i {}^i \ddot{\vec{r}}_{G_i} \right) \right\} \quad (22)$$

Applying vectorial products properties and using again expression (14), would give:

$$B_j = - \left( \frac{\partial {}^j \dot{\vec{\omega}}_j}{\partial \ddot{q}_j} \right)^T \cdot {}^j \vec{\beta}_j \quad (23)$$

where

$${}^j \vec{\beta}_j = \sum_{i=j}^n \left[ {}^j \mathbf{R}_i \cdot \left( m_i {}^i \ddot{\vec{r}}_{G_i} \wedge {}^i \vec{r}_{O_{j-1}, G_i} \right) \right] \quad (24)$$

This expression could be calculated in a recursive way as follows:

$${}^j \vec{\beta}_j = m_j {}^j \ddot{\vec{r}}_{G_j} \wedge {}^j \vec{r}_{O_{j-1}, G_j} + {}^j \vec{\varphi}_j \wedge {}^j \vec{r}_{O_j, O_{j+1}} + {}^j \mathbf{R}_{j+1} \cdot {}^{j+1} \vec{\beta}_{j+1} \quad (25)$$

where

$${}^j \vec{\varphi}_j = {}^j \mathbf{R}_{j+1} \cdot \left( m_{j+1} {}^{j+1} \ddot{\vec{r}}_{G_{j+1}} + {}^{j+1} \vec{\varphi}_{j+1} \right) \quad (26)$$

This formulation leads to an algorithm to solve the Inverse Dynamic Problem in robots of computational complexity of  $O(n)$ . The algorithm description and the analysis of its computational complexity will be shown now. It's composed by 6 steps (note that in order to compare the computational complexity with other algorithms, only revolute joints are considered and the robot base is considered fixed).

In the step 1, there are the computation of the rotation matrices  ${}^{i-1}R_i$  and translation vectors  ${}^{i-1}\vec{r}_{O_{i-1},O_i}$ . The velocities and accelerations are computed in the step 2. Step 3 derives  ${}^i\dot{\omega}_i$  with respect to the generalised accelerations  $\ddot{q}_i$ , and the  $A_i$  and  $B_i$  terms are obtained in the steps 4 and 5. Finally, step 6 computes the generalised forces  $\tau_i$ .

The computational complexity of the proposed algorithm is summarised in Table 2. Furthermore, the computational complexity of the Luh, Walker and Paul algorithm is reported in Table 3.

Table 2. Proposed algorithm complexity

Step	Complexity	
	( $\times$ )	(+)
Step 1	4n	0
Step 2	62n-75	46n-61
Step 3	0	0
Step 4	24n-23	18n-18
Step 5	31n-38	26n-37
Step 6	0	n

Table 3. Computational complexity for robots with  $n \geq 3$

Algorithm	Complexity	$n = 6$
Luh, Walker y Paul	( $\times$ ) 121n - 112	614
	(+) 90n - 82	458
Proposed algorithm	( $\times$ ) 121n - 136	590
	(+) 91n - 116	430

As can be appreciated from Table 3, the computational complexity of the proposed algorithm is very close to the Newton-Euler based algorithm. It is also remarkable that using the Denavit-Hartenberg notation under Paul's convention, no substantial differences are observed, being in this case the computational complexity  $133n - 74$  ( $\times$ ) and  $97n - 71$  (+) for a robot with only rotational joints, so that, for a six degree of freedom robot, the computational complexity would be 724 ( $\times$ ) and 511 (+).

### 3. A Proposed Control Architecture for Industrial Robots

Increasing demands on the robot systems performance has led to the development of advanced control methods. If the robotic system has an open control stage, the implementation and the use of any of the control methods are very easy since it can use any high level programming language (for instance the C language has been used to program the new control system). In this work a new control architecture was used. This control unit is based on PC, and was implemented to control a 6-joint industrial robot, the PUMA 600.

The original control module consists of one processor LS-11/ 02, which interprets VAL-II statements and generates trajectories that are sent to the six digital servos (each of them containing a Rockwell 6503 processor). These digital servos generate analog signals through a series of Digital to Analog converters, which are sent to the amplifiers panels connected to the arm. The control loop is closed with the potentiometers and the encoders. Each robot joint has associated a potentiometer that gives the joint absolute position, while the signals of the incremental encoders (200 counts per revolution for the 2 first joints and 250 for the remainder) to provide a more precise motion measurement.

In order to avoid the original control unit limitations (high level tasks implementation, modification of the control algorithm etc.), the control stage has been substituted by a module based on PC. In this way the PC has access to the positions and generates the adequate control actions in order to move the different elements of the robot. This proposed architecture is specified in Figure 2.



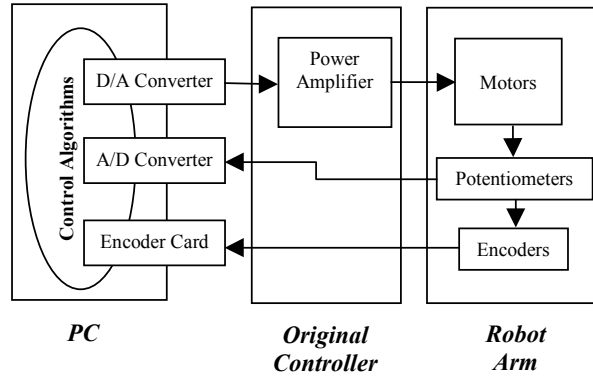


Figure 2. New control architecture used in this work

In order to implement this control architecture, four data acquisition cards have been used: a card, *Advantech™ PCL-818*, has been used in order to obtain the analog outputs; another one, *PCL-726*, has been used for supplying the control actions; and the two others, *PCL-833*, have been used for reading encoders. The proposed control architecture gives two advantages, first the simplicity and second the low cost (the total cost of all these cards doesn't exceed \$2.000 dollars). In addition, this open architecture gives a powerful platform for programming a higher level tasks.

Technically this control architecture works as follows:

- Analog inputs: these are the potentiometers signals that give the absolute position of each robot joint. Since it is had six elements in the robot it is necessary 6 analog inputs to obtain the potentiometers voltages (0-5volts). These analog inputs are provided by the PCL-818 card.
- Digital inputs: the robot supplies some *thermo signals* which indicate the joint motors overtemperature. These signals are digital, so it is necessary 6 digital inputs to read them. These digital inputs are also provided by the PCL-818 card.
- Encoder signals: the encoders used by the robot are incremental encoders, and each of them provides three signals: the channel A, the channel B and the index pulse. Since each PCL-833 card can control 3 encoders, it is needed 2 cards in order to get the robot position
- Analog outputs: once the 6 control actions have been calculated by the control strategy, it will be necessary to make a digital to analog conversion. Although the power amplifiers of the robot could be fed with 12 volts, we had limited it to 10 volts due to the robot and the PCL-726 card features, therefore 6 channels of the PCL-726 card are used.
- Digital outputs: in order to activate the robot tool it is necessary 2 digital outputs. This digital outputs are also provided by the PCL-726.

#### 4. Inverse Dynamic Control

With the control architecture depicted previously some controllers, based on inverse dynamic method, have been implemented. This control approach makes a regular static state feedback that transforms the nonlinear system into a linear one (this is known as the inverse dynamics or feedback linearization problem). Potentially this technique is very useful because it reduces the nonlinear control problem to the control problem of a linear system, for which many tools are available. Assuming the dynamic model as:

$$\dot{x}^{(n)} = f(x) + b(x)u$$

where  $f(x)$  is a nonlinear state function and  $u$  is the control input. If you use as the control input the expression:

$$u = \frac{1}{b} [v - f] \tag{27}$$

the nonlinearities will be cancelling, and the simple input-output relation will be obtained:

$$x^{(n)}=v$$

where  $v$  is a new input vector to be defined below. In the robot case, the dynamic model can be expressed with the next equation:

$$\tau = C(q, \dot{q})\dot{q} + G(q) + M(q)\ddot{q} \quad (28)$$

where  $C(q, \dot{q})$  is the vector of centrifugal and Coriolis terms,  $G(q)$  is the vector of gravity terms, and  $M(q)$  is the mass matrix. Working with this equation:

$$\ddot{q} = M^{-1}(q)(\tau - C(q, \dot{q})\dot{q} - G(q))$$

so, we have the next terms:

$$f(x) \equiv M^{-1}(q)(-C(q, \dot{q})\dot{q} - G(q))$$

$$b(x) \equiv M^{-1}(q)$$

using the general expression (27):

$$\tau_c = \frac{1}{M^{-1}(q)} \left[ v - M^{-1}(q)(-C(q, \dot{q})\dot{q} - G(q)) \right]$$

so, the controllers based on the inverse dynamics could be viewed such as particular cases of the next general control law:

$$\tau_c = C(q, \dot{q})\dot{q} + G(q) + M(q)v \quad (29)$$

The inverse dynamics control (29) shows how the nonlinearities such as Coriolis terms as well as gravity terms can be simply compensated by adding these forces to the control input. Depending on the  $v$  expression, different controllers can be obtained (Valera, 2000):

Table 4. Inverse dynamics controllers implemented

Controller	$v$
Point to point control	$-K_d\dot{q} - K_p e$
Tracking Control	$\ddot{q}_d - K_d\dot{q} - K_p e$
Tracking Control with integral action	$\ddot{q}_d - K_d\dot{q} - K_p e - K_i \int_0^t e(u) du$

The first controller implemented was the point to point controller. In this case a proportional and derivative terms compose the linear auxiliary control input  $v$ , and the robot system is exponentially stable by a suitable choice of the matrices  $K_d$  and  $K_p$ .

The second controller is very similar to the first, but in this case the robot must follow a given time-varying trajectory  $q_d(t)$  and its successive derivatives  $\dot{q}_d$  and  $\ddot{q}_d$  which respectively describe the desired velocity and acceleration. This tracking control is very simple but it has several drawbacks: any error in the robot dynamics estimation can cause a variation in the equilibrium point and therefore a position error. The second problem that can occur is related to the dead-zone phenomenon: in this case the static friction at the motor shafts can also provoke a position error. A practical solution to attempt to solve these problems is to insert an integral action in the control law. This is the case of the last controller, where it has been added the integral of the error.

All these controllers need the dynamic elements of the robot. In this work we used the Gibbs-Appell equations presented before to calculate the control action because is a very efficient way to

calculate these elements. In this way, some functions were programmed to calculate with the  $A_j$  term the mass matrix  $M(q)$  using equation (15), to obtain the vector bias  $C(q, \dot{q})\dot{q} + G(q)$  employing the  $B_j$  term of equation (23), etc.

On the other hand, because the robot parameters can vary along the time (deformation or wear on the elements, robot payload etc.), in order to implement the robot control, a precise real robot dynamic model is required. A dynamic parameters identification can be obtained rewritten the robot equations:

$$\tau_m = K_m(q, \dot{q}, \ddot{q}) \cdot \Phi_m$$

where  $\tau_m$  is the generalized torques,  $K_m$  is the observation matrix, and  $\Phi_m$  is the robot parameters

$$\Phi_m = \begin{bmatrix} m_m & mc_{x_m} & mc_{y_m} & mc_{z_m} & I_{xx_m} & I_{xy_m} & I_{xz_m} & I_{yy_m} & I_{yz_m} & I_{zz_m} \end{bmatrix}$$

Parameters vector  $\Phi_i$  can be obtained using least squares or any other matrix technique. In our work, QR decomposition, Singular Values decomposition, pseudo-inverse calculation or Ridge regression have been implemented. In this way, the robot parameters used are the following:

Table 5. Denavit-Hartenberg parameters

Joint	$\alpha$ (rad)	a (m)	D (m)
1	0	0	0
2	$-\pi/2$	0	0
3	0	0.432	-0.15
4	$\pi/2$	-0.02	-0.433
5	$-\pi/2$	0	0
6	$\pi/2$	0	0

Table 6. Masses of the links

Joint	Mass (kg)
1	10.521
2	15.781
3	8.767
4	1.052
5	1.052
6	0.351

Table 7. Coordinates in local reference system

Link $i$	Centre of masses (m) ${}^i\vec{r}_{O_i, G_i}$
1	$[0.0 \ -0.054 \ 0.0]^T$
2	$[0.1398 \ 0.0 \ 0.1491]^T$
3	$[-0.32 \cdot 10^{-3} \ -0.197 \ 0.0]^T$
4	$[0.0 \ 0.0 \ -0.057]^T$
5	$[0.0 \ -0.007 \ 0.0]^T$
6	$[0.0 \ 0.0 \ 0.0372]^T$

The inertial tensor of links ( $kg \cdot m^2$ ), defined with respect to parallel axes to the local links and passing through their centre of masses are:

$${}^1I_{G_1} = \begin{bmatrix} 1.612 & 0 & 0 \\ 0 & 0.5091 & 0 \\ 0 & 0 & 1.612 \end{bmatrix} \quad {}^2I_{G_2} = \begin{bmatrix} 0.4898 & 0 & 0 \\ 0 & 8.0783 & 0 \\ 0 & 0 & 8.2672 \end{bmatrix} \quad {}^3I_{G_3} = \begin{bmatrix} 3.3768 & 0 & 0 \\ 0 & 0.3009 & 0 \\ 0 & 0 & 3.3768 \end{bmatrix}$$

$${}^4I_{G_4} = \begin{bmatrix} 0.181 & 0 & 0 \\ 0 & 0.181 & 0 \\ 0 & 0 & 0.1273 \end{bmatrix} \quad {}^5I_{G_5} = \begin{bmatrix} 0.0735 & 0 & 0 \\ 0 & 0.0735 & 0 \\ 0 & 0 & 0.1273 \end{bmatrix} \quad {}^6I_{G_6} = \begin{bmatrix} 0.0071 & 0 & 0 \\ 0 & 0.0071 & 0 \\ 0 & 0 & 0.0141 \end{bmatrix}$$

Before the implementation in the new control unit, some simulations have been carried out with the symbolic algebra software MACSYMA. In the next example, a straight-line trajectory has been considered. The start point has coordinates  $[0.6 \ 0.175 \ 0.250]^T m$  and the end point  $[0.018 \ 0.757 \ 0]^T m$ , the constant orientation given by Euler angles  $ZYZ$  is  $(45^\circ \ 60^\circ \ 90^\circ)$ . The constant linear velocity prescribed for the robot end-effector is  $0.1 m/s$ , the total time needed for the prescribed robot motion is 8.6 seconds (s). In Figure 3 the initial configuration of the robot, an intermediate and the final one are depicted. The torques required in each joint are depicted in Figure 4 as a function of time.

Figure 5 will show positions of the real robot controlled by the inverse dynamic strategies with a ramp and splines references. In both cases the joint positions follow without problems the input references. Several types of computers have been used to obtain the execution times of these control algorithms. If a Pentium 150Mhz. is employed, the execution time was  $0.225 ms$ , and with a Pentium II 350Mhz. the time required was  $0.061 ms$ .

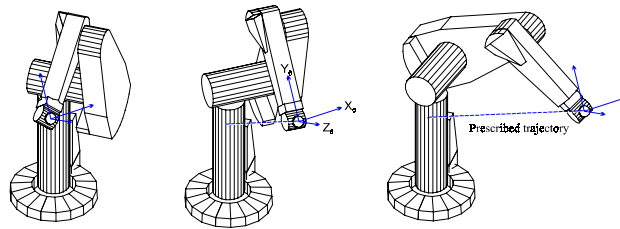


Figure 3. Prescribed trajectory for the Puma robot

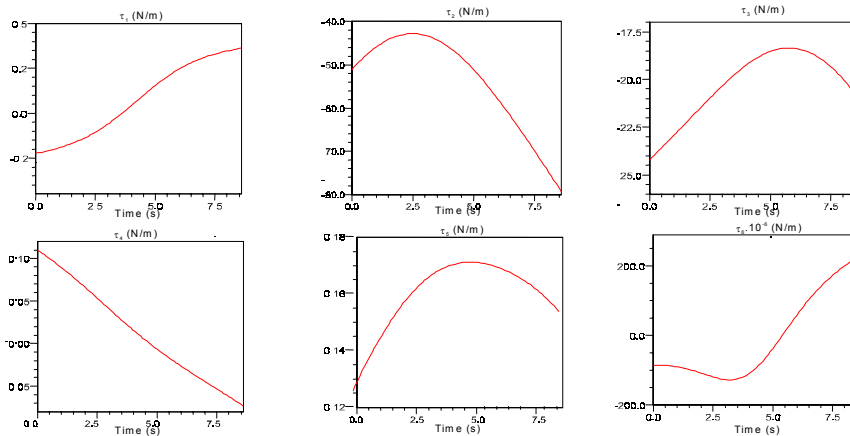


Figure 4. Torques required per joint

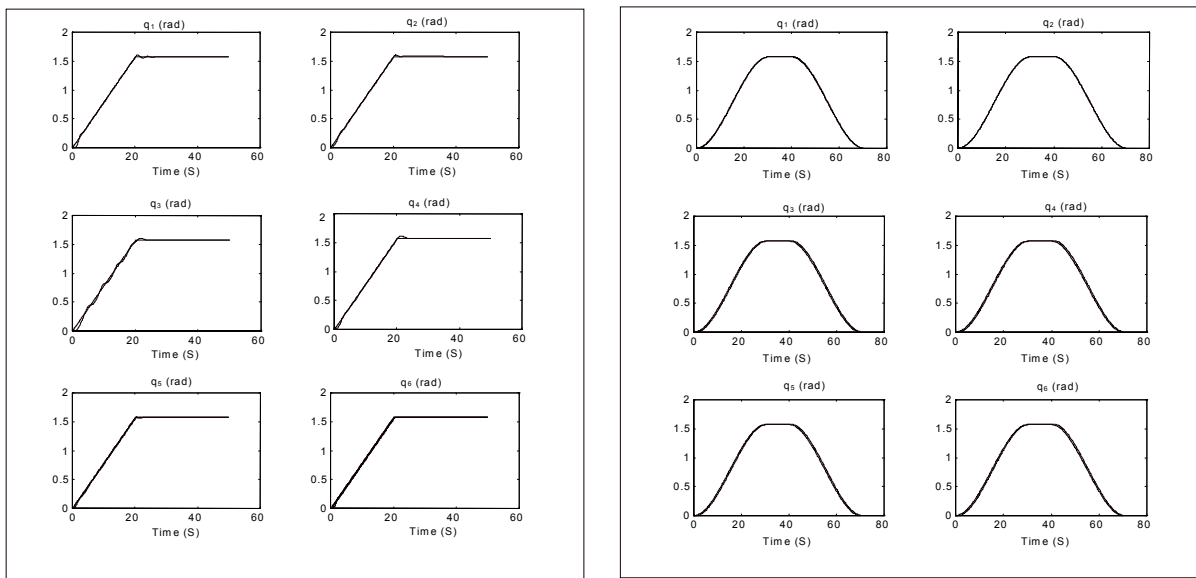


Figure 5. Real robot positions for ramp and splines references

## 5. Conclusions

This paper has shown the implementation of the inverse dynamics controllers using an open control system for the PUMA industrial robot arm. Due to the components used (PC and conventional data acquisition cards), the control stage is very economic and flexible. This flexibility allows for instance, programming and comparing advanced control strategies, control algorithms based on artificial vision as well as integrating the PUMA in a flexible manufacturing system, etc.

In this open control unit, several algorithms based on the Gibbs-Appell equations have been proposed and verified for solving the Inverse Dynamic Problem and the control problem. These algorithms have computational complexities slightly lower than the algorithm based on Newton-Euler

equations of motion, formulated in a similar way, and using mainly vectors in its recursive formulation. This fact confirms the conclusion from other authors who claim that the efficiency of the dynamic algorithms arises from the type of formulation used, rather than the Principle of Dynamics considered. In this way, it can be expected that further reductions in computational complexity may be achieved by using tensorial notation rather than vectorial notation. For instance, important savings could be obtained developing the term corresponding to the Moment of Inertia (Euler equation) in the  $A$  terms in a tensorial form.

The paper showed the robot response in the simulation and in the real execution, working with the point to point and the tracking problem. Several aspects of the overall robot response can be analysed easily. In addition, the PC environment has allowed us to connect the data structure with commercial CADCS packages such as MATLAB, MATHEMATICA etc.

## References

- Angeles, J., Ma, O., and Rojas, 1989. A. An Algorithm for the Inverse Dynamics of n-Axis General Manipulators Using Kane's Equations. *Computers Math. Applic.* Vol 17, n° 12: 1545-1561.
- Balafoutis, C.A., and Patel, R.A., 1991. *Dynamic Analysis of Robot Manipulators: A Cartesian Tensor Approach*. Boston: Kluwer Academic Press.
- Craig, J.J., 1986. *Introduction to Robotics: Mechanics and Control*. Addison-Wesley, Reading.
- Desoyer, K. and Lugner, P., 1989. Recursive formulation for the analytical or numerical application of the Gibbs-Appell method to the dynamics of robots. *Robotica*, Vol 7: 343-347.
- Fu, K.S., Gonzalez, R.C., and Lee, C.S.G., 1987. *Robotics: Control, Sensing, Vision, and Intelligence*. Singapore: McGraw-Hill.
- Hollerbach, J.M., 1980. A Recursive Lagrangian Formulation of Manipulator Dynamics and a Comparative Study of Dynamics Formulation Complexity. *IEEE Trans. on Systems, Man, and Cybernetics SMC-10*, 11: 730-736.
- Khalil, W., and Kleinfinger, J-F., 1987. Minimum Operations and Minimum parameters of the Dynamic Models of Tree Structure Robots. *IEEE J. of Robotics and Automation*, Vol 3, 6: 517-526.
- Lee, C.S.G. Lee and Chang, P.R., 1986. Efficient Parallel Algorithm For Robot Inverse Dynamics Computation. *IEEE Trans. on Systems, Man, and Cybernetics*, Vol 16, 4: 532-542
- Luh, J.Y.S., Walker, M.W., and Paul, R. P.C., 1980. On-line Computational Scheme for Mechanical Manipulators. *Journal of Dynamic Systems, Measurement, and Control*, Vol. 102: 69-76.
- Moreira N., Alvito, P. and Lima, P., 1996. First steps towards an open control architecture for a PUMA 560. *Proc. 2<sup>nd</sup> Portuguese Conf. on Automatic Control*. Porto, Portugal.
- Murray, J.J. and Neuman, C.P., 1988 Organizing Customized Robot Dynamics Algorithms for Efficient Numerical Evaluation. *IEEE Trans. on Systems, Man, and Cybernetics*, Vol 18, 1: 115-125.
- Renaud, M., 1975. Contribution a l'étude de la modélisation et de la commande des systèmes mécaniques articulés. Thèse doctoral. Université Paul Sabatier, Toulouse.
- Valera, A. 2000 *Análisis Comparativo de Técnicas de Control de Robots Rígidos y Flexibles*. Valencia. Editorial de la Universidad Politécnica de Valencia. ISBN 84-699-2911-9
- Valera, A., Vergara, J.V., Tornero, J., and García, E. 1998. Control a PUMA 500 Using a New Open Architecture. *Proc. 3<sup>rd</sup> Portuguese Conf. on Automatic Control*, Coimbra, Portugal, Vol. 2, pp. 795--798
- Vukobratovic, M., and Kircanski, N., 1985. *Real-Time Dynamics of Manipulation Robots*. Berlin: Springer-Verlag,
- Zomaya, A.Y., 1992. *Modelling and Simulation of Robot Manipulators: A Parallel Processing Approach*. Singapore: World Scientific Publishing Co.