Robotica (2002) volume 20, pp. 59–67. © 2002 Cambridge University Press
DOI: 10.1017/S0263574701003502 Printed in the United Kingdom

Inverse dynamic problem in robots using Gibbs-Appell equations

V. Mata*, S. Provenzano†, J.L. Cuadrado* and F. Valero*

** Departamento de Ingeniería Mecánica y de Materiales, Universidad Politécnica de Valencia, Camíno de Vera s/n, 46022 Valencia (Spain)*
† Escuela de Ingeniería Mecánica, Universidad de Los Andes (Venezuela)

(Received in Final Form: March 26, 2001)

SUMMARY
In this paper, two algorithms for solving the Inverse Dynamic Problem based on the Gibbs-Appell equations are proposed and verified. Both are developed using mainly vectorial variables, and the equations are expressed in a recursive form. The first algorithm has a computational complexity of $O(n^2)$ and is the least efficient of the two; the second algorithm 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. The $O(n)$ proposed algorithm will be used to solve the Inverse Dynamic Problem in a PUMA industrial robot.

KEYWORDS: Robotics; Inverse dynamic problem; Gibbs-Appell formulation.

1. INTRODUCTION

The literature about the Inverse Dynamic Problem (IDP) in robots is vast. The interest of its potential applications has contributed to it. These applications may be classified into three main areas as follows:

- Industrial robot dynamic control with significant gains in robot performance, speed, and accuracy.
- Verification that the torques/forces needed to execute a proposed trajectory do not exceed the capabilities of the actuators.
- The Inverse Dynamic Problem as part of the Forward Dynamic Problem.

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, for instance, the angular velocity could be represented by the time derivative of a rotation matrix, by a vector, and can also be described by a second order skew-symmetric Cartesian tensor. These algorithms can be implemented by means of symbolic programs or strictly numerical ones. Finally, according to the computer architecture where they will be processed, the algorithms can be sequential or parallel. In Table I are shown some of the proposed algorithms for solving the IDP on robots with rigid links and ideal joints.

Custom-made algorithms, which take advantage of the special characteristics of particular industrial robots, must be particularly mentioned. Examples of these are proposed in Khosla and Neuman[7] and Murray and Neuman,[8] both based on the Newton-Euler formulation, expressed in a recursive way and with vectorial variables. However, the first one is numerical and the second is symbolic.

Several authors[4] showed that the computational efficiency of the dynamic algorithms depends fundamentally on the way the calculations are arranged rather than on the dynamic principle in which they are based. This idea has already been proposed by Hollerbach,[1] in which the dynamic problem was reformulated for robots by using the Principle of Lagrange in a recursive way and using rotation matrix 3×3 instead of a 4×4 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 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

Table I. Several algorithms for solving the IDP.

Authors	Dynamic Principle	Formulation	Type of variables	Type of resolution	Type of processing
Hollerbach[1]	Lagrange-Euler	Recursive	Matricial	Numerical	Sequential
Luh et al.[2]	Newton-Euler	Recursive	Vectorial	Numerical	Sequential
Angeles et al.[3]	Kane	Recursive	Tensorial	Numerical	Sequential
Balafoutis-Patel[4]	Newton-Euler	Recursive	Tensorial	Numerical	Sequential
Khalil-Kleifinger[5]	Newton-Euler	Recursive	Vectorial	Symbolic	Sequential
Lee and Chang[6]	Newton-Euler	Recursive	Vectorial	Numerical	Parallel

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, Walker and Paul[2] 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, Gonzalez and Lee[9] provided a version of the same algorithm with a complexity of 117n-24 multiplications and 103n-21 additions; Zomaya[10] gave a complexity of 150n multiplications and 116n additions. Finally, Craig[11] 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, and according to Pars[12] . . . *provide what is probably the simplest and most comprehensive form of the equations of motion so far discovered*. However, in the robot dynamics field there are few published references to works based on them. Renaud[13] 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 Potkonjak[14] developed a closed-form algorithm with $O(n^3)$ computational complexity. Desoyer and Lugner[15] 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 our work the Gibbs-Appell equations are applied to solving the inverse dynamic problems of robots that have rigid links and ideal pairs. Two algorithm are proposed, the first one has a computational complexity of $O(n^2)$, and the second of $O(n)$. In both cases, the algorithms are 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 the best of these two algorithms 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 algorithms are developed. In Section 3, the algorithms are formalised and an analysis of their computational complexity is provided, comparing them with the Luh, Walker and Paul algorithm. In Section 4, one of the developed algorithms is applied to a PUMA-type industrial robot. Finally, in Section 5 we summarise 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,[3] which considers four parameters θ_i, α_i, a_i, and d_i, as 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 ${}^{i-1}\mathbf{R}_i$ and the position vector ${}^{i-1}\vec{r}_{O_{i-1},O_i}$.

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

$$ {}^{i-1}\vec{r}_{O_{i-1},O_1} = \begin{bmatrix} a_i \\ d_i \sin\alpha_i \\ -d_i \cos\alpha_i \end{bmatrix} $$

The Gibbs-Appell dynamic equations are derived from the Gibbs function definition (also known as the energy of the accelerations). When we write the original form for an arbitrary solid composed of n-elemental particles with masses m_i an acceleration a_i, the Gibbs function 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

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

where m_i is the mass of the i-th link, $\mathbf{I}_{G_i}$ is the 3×3 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

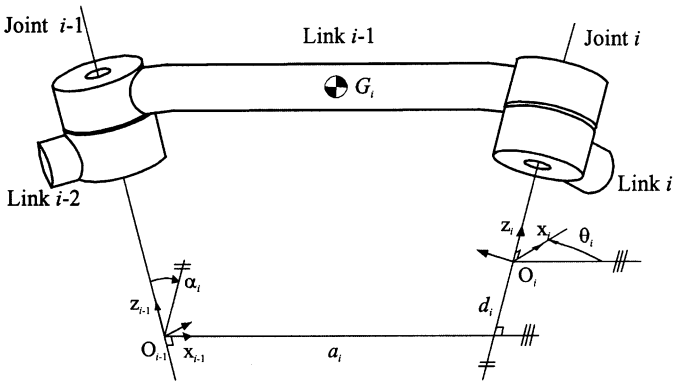

Fig. 1. Modified Denavit-Hartenberg notation.

https://doi.org/10.1017/S0263574701003502 Published online by Cambridge University Press

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:

$$G_i=\frac{1}{2}\,m_i({}^0\mathbf{R}_i\cdot {}^i\ddot{\vec{r}}_{G_i})^{\mathrm{T}}\cdot {}^0\mathbf{R}_i\cdot {}^i\ddot{\vec{r}}_{G_i}$$
$$+\frac{1}{2}\,({}^0\mathbf{R}_j\cdot {}^i\dot{\vec{\omega}}_i)^{\mathrm{T}}\cdot {}^0\mathbf{R}_i\cdot {}^i\mathbf{I}_{G_i}\cdot ({}^0\mathbf{R}_i)^{\mathrm{T}}\cdot ({}^0\mathbf{R}_j\cdot {}^i\dot{\vec{\omega}}_i)$$
$$+({}^0\mathbf{R}_i\cdot {}^i\dot{\vec{\omega}}_i)^{\mathrm{T}}\cdot\{{}^0\mathbf{R}_i\cdot {}^i\dot{\vec{\omega}}_i\wedge[{}^0\mathbf{R}_i\cdot {}^i\mathbf{I}_{G_i}\cdot ({}^0\mathbf{R}_i)^{\mathrm{T}}\cdot {}^0\mathbf{R}_i\cdot {}^i\vec{\omega}_i]\} \tag{2}$$

the expression (2) could be rewritten as follows:

$$G_i=\frac{1}{2}\,m_i({}^i\ddot{\vec{r}}_{G_i})^{\mathrm{T}}\cdot ({}^0\mathbf{R}_i)^{\mathrm{T}}\cdot {}^0\mathbf{R}_i\cdot {}^i\ddot{\vec{r}}_{G_i}$$
$$+\frac{1}{2}\,({}^i\dot{\vec{\omega}}_i)^{\mathrm{T}}\cdot ({}^0\mathbf{R}_j)^{\mathrm{T}}\cdot {}^0\mathbf{R}_i\cdot {}^i\mathbf{I}_{G_i}\cdot ({}^0\mathbf{R}_i)^{\mathrm{T}}\cdot ({}^0\mathbf{R}_j\cdot {}^i\dot{\vec{\omega}}_i)$$
$$+({}^i\dot{\vec{\omega}}_i)^{\mathrm{T}}\cdot ({}^0\mathbf{R}_i)^{\mathrm{T}}\cdot\{{}^0\mathbf{R}_i\cdot {}^i\dot{\vec{\omega}}_i\wedge[{}^0\mathbf{R}_i\cdot {}^i\mathbf{I}_{G_i}\cdot ({}^0\mathbf{R}_i)^{\mathrm{T}}\cdot {}^0\mathbf{R}_i\cdot {}^i\vec{\omega}_i]\} \tag{3}$$

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

$$G_i=\frac{1}{2}\,m_i({}^i\ddot{\vec{r}}_{G_i})^{\mathrm{T}}\cdot {}^i\ddot{\vec{r}}_{G_i}$$
$$+\frac{1}{2}\,({}^i\dot{\vec{\omega}}_i)^{\mathrm{T}}\cdot {}^i\mathbf{I}_{G_i}\cdot {}^i\dot{\vec{\omega}}_i+({}^i\dot{\vec{\omega}}_i)^{\mathrm{T}}\cdot[{}^i\dot{\vec{\omega}}_i\wedge {}^i\mathbf{I}_{G_i}\cdot {}^i\vec{\omega}_i] \tag{4}$$

where ${}^i\ddot{\vec{r}}_{G_i}$, ${}^i\vec{\omega}_i$, ${}^i\dot{\vec{\omega}}_i$ 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

$$G=\sum_{i=1}^{n} G_i \qquad (i=1,2\ldots n) \tag{5}$$

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, τ_j

$$\tau_j=\sum_{i=j}^{n}\frac{\partial G_i}{\partial \ddot{q}_i} \qquad (j=1,2\ldots n) \tag{6}$$

that is,

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

In the following discussion, two procedures, with different computational efficiency for obtaining the generalised forces, will be developed.

First Method

The first formulation for the solution of the Inverse Dynamic Problem in robots would be obtained by developing every term of the expression (7) and adding them up. 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:

$${}^i\vec{\omega}_i={}^i\mathbf{R}_{i-1}\cdot {}^{i-1}\vec{\omega}_{i-1}+\rho_i\,{}^i\vec{z}_i\cdot\dot{q}_i \tag{8}$$

$${}^i\dot{\vec{\omega}}_i={}^i\mathbf{R}_{i-1}\cdot {}^{i-1}\dot{\vec{\omega}}_{i-1}+\rho_i[{}^i\vec{z}_i\cdot\ddot{q}_i+{}^i\mathbf{R}_{i-1}\cdot {}^{i-1}\vec{\omega}_{i-1}\wedge({}^i\vec{z}_i\cdot\dot{q}_i)] \tag{9}$$

$${}^i\ddot{\vec{r}}_{O_i}={}^i\mathbf{R}_{i-1}[{}^{i-1}\ddot{\vec{r}}_{O_{i-1}}+{}^{i-1}\dot{\vec{\omega}}_{i-1}\wedge({}^{i-1}\vec{\omega}_{i-1}\wedge {}^{i-1}\vec{r}_{O_{i-1},O_i})+{}^{i-1}\dot{\vec{\omega}}_{i-1}$$
$$\wedge {}^{i-1}\vec{r}_{O_{i-1},O_i}]+(1-\rho_i)[{}^i\vec{z}_i\cdot\ddot{q}_i+2({}^i\vec{\omega}_i\wedge {}^i\vec{z}_i\cdot\dot{q}_i)] \tag{10}$$

$${}^i\ddot{\vec{r}}_{G_i}={}^i\ddot{\vec{r}}_{O_i}+{}^i\dot{\vec{\omega}}_i\wedge({}^i\vec{\omega}_i\wedge {}^i\vec{r}_{O_i,G_i})+{}^i\dot{\vec{\omega}}_i\wedge {}^i\vec{r}_{O_i,G_i} \tag{11}$$

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

Next, the derivatives with respect to the generalised acceleration and the acceleration of the centre of mass of the *i*-th link will be obtained.

To develop the $\frac{\partial {}^i\dot{\vec{\omega}}_i}{\partial \ddot{q}_j}$ term, we start from expression (9). This derivative could be obtained by a recursive procedure as follows:

$$\text{If } i<j \qquad \frac{\partial {}^i\dot{\vec{\omega}}_i}{\partial \ddot{q}_j}=[0\ 0\ 0]^{\mathrm{T}}$$
$$\text{If } i>j \qquad \frac{\partial {}^i\dot{\vec{\omega}}_i}{\partial \ddot{q}_j}={}^i\mathbf{R}_{i-1}\cdot {}^{i-1}\mathbf{R}_{i-2}\cdots {}^{j-1}\mathbf{R}_j\cdot\frac{\partial {}^j\dot{\vec{\omega}}_j}{\partial \ddot{q}_j} \tag{12}$$
$$\text{If } i=j \qquad \frac{\partial {}^i\dot{\vec{\omega}}_i}{\partial \ddot{q}_j}={}^i\vec{z}_i$$

The development of the $\frac{\partial {}^i\ddot{\vec{r}}_{G_i}}{\partial \ddot{q}_j}$ term comes from deriving expression (11), then obtaining

$$\frac{\partial {}^i\ddot{\vec{r}}_{G_i}}{\partial \ddot{q}_j}=\frac{\partial {}^i\ddot{\vec{r}}_{O_i}}{\partial \ddot{q}_j}+\frac{\partial {}^i\dot{\vec{\omega}}_i}{\partial \ddot{q}_j}\wedge {}^i\vec{r}_{O_i,G_i} \tag{13}$$

https://doi.org/10.1017/S0263574701003502 Published online by Cambridge University Press

The $\frac{\partial\, {}^{i}\ddot{\vec{r}}_{O_i}}{\partial \ddot{q}_j}$ term for revolute joints is obtained using the expression (10) which leads to:

$$\frac{\partial\, {}^{i}\ddot{\vec{r}}_{O_i}}{\partial \ddot{q}_j} = {}^{i}\mathbf{R}_{i-1} \cdot \left(\frac{\partial\, {}^{i-1}\ddot{\vec{r}}_{O_{i-1}}}{\partial \ddot{q}_j} + \frac{\partial\, {}^{i-1}\dot{\vec{\omega}}_{i-1}}{\partial \ddot{q}_j} \wedge {}^{i-1}\vec{r}_{O_{i-1},O_i} \right) \quad (14)$$

This expression could also be determined by a recursive procedure similar to the one used earlier:

$$\text{If } i \leq j \qquad \frac{\partial\, {}^{i}\ddot{\vec{r}}_{O_i}}{\partial \ddot{q}_j} = [0\ 0\ 0]^{\mathrm{T}} \quad (15)$$

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

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

$$\frac{\partial\, {}^{i}\ddot{\vec{r}}_{O_i}}{\partial \ddot{q}_j} = {}^{i}\mathbf{R}_{i-1} \cdot \frac{\partial\, {}^{i-1}\ddot{\vec{r}}_{O_{i-1}}}{\partial \ddot{q}_j} + {}^{i}\mathbf{R}_{i-1} \cdot \frac{\partial\, {}^{i-1}\dot{\vec{\omega}}_{i-1}}{\partial \ddot{q}_j} \wedge {}^{i-1}\vec{r}_{O_{i-1},O_i} + {}^{i}\vec{z}_i \cdot \frac{\partial \ddot{q}_i}{\partial \ddot{q}_j} \quad (16)$$

Notice that there is an additional term in expression (16) in relation with (14), which must be included if the *i*-th joint is a prismatic one. This term is:

$${}^{i}\vec{z}_i \cdot \frac{\partial \ddot{q}_i}{\partial \ddot{q}_j} \quad (17)$$

that, depending on the type of link which it is applied to, would be given by:

$$\text{If } i = j \qquad {}^{i}\vec{z}_i \quad (18)$$

$$\text{If } i \neq j \qquad [0\ 0\ 0]^{\mathrm{T}} \quad (19)$$

Terms $\frac{\partial\, {}^{i}\ddot{\vec{r}}_{G_i}}{\partial \dot{q}_j}$ could be obtained from expression (13). Finally, expression (7) allows us to obtain the generalised forces.

As will be shown in the next section, this development of the Gibbs-Appell equations will lead to an algorithm to solve the IDP with a computational complexity of $O(n^2)$.

Second Method

The second 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:

$$\tau_j = \underbrace{\sum_{i=j}^{n} \left\{ \left(\frac{\partial\, {}^{i}\dot{\vec{\omega}}_i}{\partial \ddot{q}_j} \right)^{\mathrm{T}} \cdot [{}^{i}\mathbf{I}_{G_i} \cdot {}^{i}\dot{\vec{\omega}}_i + {}^{i}\vec{\omega}_i \wedge ({}^{i}\mathbf{I}_{G_i} \cdot {}^{i}\vec{\omega}_i)] \right\}}_{A_j} + \underbrace{\sum_{i=j}^{n} \left[\left(\frac{\partial\, {}^{i}\ddot{\vec{r}}_{G_i}}{\partial \ddot{q}_j} \right)^{\mathrm{T}} \cdot m_i \cdot {}^{i}\ddot{\vec{r}}_{G_i} \right]}_{B_j} \quad (20)$$

It is remarkable that expression below is similar to that proposed by Angeles, Ma and Rojas in reference [3] for solving the IDP based on the Kane's dynamic formulation.

Developing the A_j term:

$$A_j = \sum_{i=j}^{n} \left\{ \left({}^{i}\mathbf{R}_j \cdot \frac{\partial\, {}^{j}\dot{\vec{\omega}}_j}{\partial \ddot{q}_j} \right)^{\mathrm{T}} \cdot [{}^{i}\mathbf{I}_{G_i} \cdot {}^{i}\dot{\vec{\omega}}_i + {}^{i}\vec{\omega}_i \wedge ({}^{i}\mathbf{I}_{G_i} \cdot {}^{i}\vec{\omega}_i)] \right\} \quad (21)$$

and, taking into account the expressions in (12), this term could be rewritten as follows:

$$A_j = \left(\frac{\partial\, {}^{j}\dot{\vec{\omega}}_j}{\partial \ddot{q}_j} \right)^{\mathrm{T}} \cdot \sum_{i=j}^{n} \{ {}^{j}\mathbf{R}_i \cdot [{}^{i}\mathbf{I}_{G_i} \cdot {}^{i}\dot{\vec{\omega}}_i + {}^{i}\vec{\omega}_i \wedge ({}^{i}\mathbf{I}_{G_i} \cdot {}^{i}\vec{\omega}_i)] \} \quad (22)$$

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 reverse recursive way is presented:

$$A_j = \left(\frac{\partial\, {}^{j}\dot{\vec{\omega}}_j}{\partial \ddot{q}_j} \right)^{\mathrm{T}} \cdot {}^{j}\vec{\alpha}_j \quad (23)$$

where

$${}^{j}\vec{\alpha}_j = {}^{j}\mathbf{I}_{G_j} \cdot {}^{j}\dot{\vec{\omega}}_j + {}^{j}\vec{\omega}_j \wedge ({}^{j}\mathbf{I}_{G_j} \cdot {}^{j}\vec{\omega}_j) + {}^{j}\mathbf{R}_{j+1} \cdot {}^{j+1}\vec{\alpha}_{j+1} \quad (24)$$

which gives recursively the A_j terms.

To develop the B_j term, the following expression is used (which comes from considering expressions (13) and (14)).

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

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

$$B_j = \sum_{i=j}^{n} \left\{ \left(\frac{\partial\, {}^{i}\dot{\vec{\omega}}_i}{\partial \ddot{q}_j} \right)^{\mathrm{T}} \cdot ({}^{i}\vec{r}_{O_{j-1},G_i} \wedge m_i\, {}^{i}\ddot{\vec{r}}_{G_i}) \right\} \quad (25)$$

Applying vectorial products properties and using again expression (12), would give:

$$B_j \equiv \left(\frac{\partial\, {}^{j}\dot{\vec{\omega}}_j}{\partial \ddot{q}_j} \right)^{\mathrm{T}} \cdot {}^{j}\vec{\beta}_j \quad (26)$$

where

$${}^{j}\vec{\beta}_j = \sum_{i=j}^{n} [{}^{j}\mathbf{R}_i \cdot (m_i\, {}^{i}\ddot{\vec{r}}_{G_i} \wedge {}^{i}\vec{r}_{O_{j-1},G_i})] \quad (27)$$

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

$${}^{j}\vec{\beta}_j = m_j\, {}^{j}\ddot{\vec{r}}_{G_j} \wedge {}^{j}\vec{r}_{O_{j-1},G_j} + {}^{j}\vec{\phi}_j \wedge {}^{j}\vec{r}_{O_j,O_{j+1}} + {}^{j}\mathbf{R}_{j+1} \cdot {}^{j+1}\vec{\beta}_{j+1} \quad (28)$$

where

https://doi.org/10.1017/S0263574701003502 Published online by Cambridge University Press

$$^{j}\vec{\phi}_j = {}^{j}\mathbf{R}_{j+1} \cdot (m_{j+1}\,{}^{j+1}\ddot{\vec{r}}_{G_{j+1}} + {}^{j+1}\vec{\phi}_{j+1}) \tag{29}$$

In the next section, it will be shown that the use of this second formulation leads to an algorithm of computational complexity of $O(n)$ to solve the Inverse Dynamic Problem in robots.

3. DESCRIPTION OF ALGORITHMS AND ANALYSIS OF THEIR COMPUTATIONAL COMPLEXITY

In this section two algorithms for solving the Inverse Dynamic Problem are presented. These algorithms are based on the formulations stated in the previous section. In order to compare their computational complexity with other algorithms, only revolute joints are considered and the robot base is considered fixed.

Algorithm 1 *(based on Method 1)*

Note: For robots with only revolute joints, the vector $^{i-1}\vec{r}_{O_{i-1},O_i}$ could be computed off-line.

Step 1.–Computation of the velocities and accelerations

Initialise:

$${}^{0}\ddot{\vec{r}}_{O_0} = [0\ 0\ g]^{\mathrm{T}}$$

$${}^{0}\ddot{\vec{r}}_{G_0} = {}^{0}\ddot{\vec{r}}_{O_0}$$

do:

$${}^{1}\vec{\omega}_1 = {}^{1}\vec{z}_1 \cdot \dot{q}_1$$

$${}^{1}\dot{\vec{\omega}}_1 = {}^{1}\vec{z}_1 \cdot \ddot{q}_1$$

$${}^{1}\ddot{\vec{r}}_{O_1} = {}^{1}\mathbf{R}_0 \cdot {}^{0}\ddot{\vec{r}}_{O_0}$$

For $i = 2, 3, \ldots, n$ do:

$${}^{i}\vec{u}_i = {}^{i}\mathbf{R}_{i-1} \cdot {}^{i-1}\vec{\omega}_{i-1}$$

$${}^{i}\vec{\omega}_i = {}^{i}\vec{u}_i + {}^{i}\vec{z}_i \cdot \dot{q}_i$$

$${}^{i}\dot{\vec{\omega}}_i = {}^{i}\mathbf{R}_{i-1} \cdot {}^{i-1}\dot{\vec{\omega}}_{i-1} + {}^{i}\vec{z}_i \cdot \ddot{q}_i + {}^{i}\vec{u}_i \wedge ({}^{i}\vec{z}_i \cdot \dot{q}_i)$$

$${}^{i}\ddot{\vec{r}}_{O_i} = {}^{i}\mathbf{R}_{i-1}[{}^{i-1}\ddot{\vec{r}}_{O_{i-1}} + {}^{i-1}\vec{\omega}_{i-1} \wedge ({}^{i-1}\vec{\omega}_{i-1} \wedge {}^{i-1}\vec{r}_{O_{i-1},O_i}) + {}^{i-1}\dot{\vec{\omega}}_{i-1} \wedge \vec{r}_{O_{i-1},O_i}]$$

For $i = 1, 2, \ldots, n$ do:

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

Complexity of Step 1

$62n - 75$ $(\times)$, $46n - 61$ $(+)$

Step 2.–Computation of the $\dfrac{\partial\, {}^{i}\dot{\vec{\omega}}_i}{\partial \ddot{q}_j}$ terms

For $i = 1, 2, \ldots, n$ do:

$$\frac{\partial\, {}^{i}\dot{\vec{\omega}}_i}{\partial \ddot{q}_i} = {}^{i}\vec{z}_i$$

For $i = 2, 3, \ldots, n$ do:

$$\frac{\partial\, {}^{i}\dot{\vec{\omega}}_i}{\partial \ddot{q}_{i-1}} = {}^{i}\mathbf{R}_{i-1} \cdot \frac{\partial\, {}^{i-1}\dot{\vec{\omega}}_{i-1}}{\partial \ddot{q}_{i-1}}$$

For $j = 1, 2, \ldots, n-1$ do:

For $i = j+2, \ldots, n$ do:

$$\frac{\partial\, {}^{i}\dot{\vec{\omega}}_i}{\partial \ddot{q}_j} = {}^{i}\mathbf{R}_{i-1} \cdot \frac{\partial\, {}^{i-1}\dot{\vec{\omega}}_{i-1}}{\partial \ddot{q}_j}$$

Complexity of Step 2

$4n^2 - 12n + 8$ $(\times)$, $2n^2 - 6n + 4$ $(+)$

Step 3.–Computation of the $\dfrac{\partial\, {}^{i}\ddot{\vec{r}}_{O_i}}{\partial \ddot{q}_j}$ terms

For $i = 2, 3, \ldots, n$ do:

$$\frac{\partial\, {}^{i}\ddot{\vec{r}}_{O_i}}{\partial \ddot{q}_{i-1}} = {}^{i}\mathbf{R}_{i-1} \cdot \frac{\partial\, {}^{i-1}\dot{\vec{\omega}}_{i-1}}{\partial \ddot{q}_{i-1}} \wedge {}^{i-1}\vec{r}_{O_{i-1},O_i}$$

For $j = 1, 2, \ldots, n-1$ do:

For $i = j+2, \ldots, n$ do:

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

Complexity of Step 3

$7n^2 - 16n + 9$ $(\times)$, $5n^2 - 13n + 8$ $(+)$

Step 4.–Computation of the $\dfrac{\partial\, {}^{i}\ddot{\vec{r}}_{G_i}}{\partial \ddot{q}_j}$ terms

For $i = 1, 2, \ldots, n$ do:

$$\frac{\partial\, {}^{i}\ddot{\vec{r}}_{G_i}}{\partial \ddot{q}_i} = \frac{\partial\, {}^{i}\dot{\vec{\omega}}_i}{\partial \ddot{q}_i} \wedge {}^{i}\vec{r}_{O_i,G_i}$$

For $j = 1, 2, \ldots, n-$ do:

For $i = j+1, \ldots, n$ do:

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

Complexity of Step 4

$3n^2 - 3n$ $(\times)$, $3n^2 - 3n$ $(+)$

Step 5.–Computation of the $\dfrac{\partial G_i}{\partial \ddot{q}_j}$ terms

For $i = 1, 2, \ldots, n$ do:

$$\frac{\partial G_i}{\partial \ddot{q}_i} = m_i \cdot \left[({}^{i}\ddot{\vec{r}}_{G_i})^{\mathrm{T}} \cdot \frac{\partial\, {}^{i}\ddot{\vec{r}}_{G_i}}{\partial \ddot{q}_i} \right] + ({}^{i}\dot{\vec{\omega}}_i)^{\mathrm{T}} \cdot {}^{i}\mathbf{I}_{G_i} \cdot \frac{\partial\, {}^{i}\dot{\vec{\omega}}_i}{\partial \ddot{q}_i} + \left(\frac{\partial\, {}^{i}\dot{\vec{\omega}}_i}{\partial \ddot{q}_i} \right)^{\mathrm{T}} \cdot [{}^{i}\vec{\omega}_i \wedge ({}^{i}\mathbf{I}_{G_i} \cdot {}^{i}\vec{\omega}_i)]$$

https://doi.org/10.1017/S0263574701003502 Published online by Cambridge University Press

For $j=1, 2, \ldots, n-1$ do:

For $i=j+1, \ldots, n$ do:

$$\frac{\partial G_i}{\partial \ddot{q}_j} = m_i \cdot \left[({}^i\ddot{\vec{r}}_{G_i})^{\mathrm{T}} \cdot \frac{\partial {}^i\ddot{\vec{r}}_{G_i}}{\partial \ddot{q}_j} \right] + ({}^i\dot{\vec{\omega}}_i)^{\mathrm{T}} \cdot {}^i\mathbf{I}_{G_i} \cdot \frac{\partial {}^i\dot{\vec{\omega}}_i}{\partial \ddot{q}_j} + \left(\frac{\partial {}^i\dot{\vec{\omega}}_i}{\partial \ddot{q}_j} \right)^{\mathrm{T}} \cdot [{}^i\vec{\omega}_i \wedge ({}^i\mathbf{I}_{G_i} \cdot {}^i\vec{\omega}_i)]$$

Complexity of Step 5

$\frac{1}{2}(15n^2-3n+30)$ $(\times)$, $6n^2-4n+12$ $(+)$

Step 6.–Computation of the generalised forces

For $j=1, 2, \ldots, n-1$ do:

For $i=j, j+1, \ldots, n$ do:

$$\tau_j = \tau_j + \frac{\partial G_i}{\partial \ddot{q}_j}$$

Complexity of Step 6

0 $(\times)$, $\frac{1}{2}(n^2-n)$ $(+)$

Algorithm 2 *(based on Method 2)*

Step 1.–Same as step 1 of algorithm 1.

Step 2.–Computation of the $\frac{\partial {}^i\dot{\vec{\omega}}_i}{\partial \ddot{q}_i}$ terms.

For $i=1, 2, \ldots, n$ do:

$$\frac{\partial {}^i\dot{\vec{\omega}}_i}{\partial \ddot{q}_i} = {}^i\vec{z}_i$$

Complexity of Step 2

0 $(\times)$, 0 $(+)$

Step 3.–Computation of the A_i terms. In this step the calculation of ${}^i\mathbf{R}_{i+1} \cdot {}^{i+1}\vec{\alpha}_{i+1}$ has been omitted and it has been added to the calculations involved in the B term, in order to achieve some savings in the number of operations. Since the A_i terms are scalar ones and as a consequence of the Denavit-Hartenberg notation, these are equal to the third component of the $\vec{\alpha}$ vectors. Calculations for $i=1$ can be simplified because only ${}^1\alpha_1{}^{(3)}$ is needed. Also, we take into account that ${}^1\vec{\omega}_1$ and ${}^1\dot{\vec{\omega}}_1$ have components one and two equal to zero.

For $i=n, n-1, \ldots, 1$ do:

$${}^i\vec{\alpha}_i = {}^i\mathbf{I}_{G_i} \cdot {}^i\dot{\vec{\omega}}_i + {}^i\vec{\omega}_i \wedge ({}^i\mathbf{I}_{G_i} \cdot {}^i\vec{\omega}_i)$$

For $i=1, 2, \ldots, n$ do:

$$A_i = \left(\frac{\partial {}^i\dot{\vec{\omega}}_i}{\partial \ddot{q}_i} \right)^{\mathrm{T}} \cdot {}^i\vec{\alpha}_i$$

Complexity of Step 3

$24n-23$ $(\times)$, $18n-18$ $(+)$

Step 4.–Computation of the B_i terms.

For $i=1, 2, \ldots, n$ do:

$${}^i\vec{k}_i = m_i \cdot {}^i\ddot{\vec{r}}_{G_i}$$

Do:

$${}^{n-1}\vec{\phi}_{n-1} = {}^{n-1}\mathbf{R}_n \cdot {}^n\vec{k}_n$$

For $i=n-2, n-3, \ldots, 1$ do:

$${}^i\vec{\phi}_i = {}^i\mathbf{R}_{i+1} \cdot ({}^{i+1}\vec{k}_{i+1} + {}^{i+1}\vec{\phi}_{i+1})$$

Do

$${}^n\vec{\beta}_n = {}^n\vec{k}_n \wedge {}^n\vec{r}_{O_n,G_n}$$

For $i=n-1, n-2, \ldots, 1$ do:

$${}^i\vec{\beta}_i = {}^i\vec{k}_i \wedge {}^i\vec{r}_{O_i,G_i} + {}^i\vec{\phi}_i \wedge {}^i\vec{r}_{O_i,O_{i+1}} + {}^i\mathbf{R}_{i+1} \cdot ({}^{i+1}\vec{\beta}_{i+1} - {}^{i+1}\vec{\alpha}_{i+1})$$

For $i=1, 2, \ldots, n$ do:

$$B_i = -\left(\frac{\partial {}^i\dot{\vec{\omega}}_i}{\partial \ddot{q}_i} \right)^{\mathrm{T}} \cdot {}^i\vec{\beta}_i$$

Complexity of Step 4

$31n-38$ $(\times)$, $26n-37$ $(+)$

Step 5.–Computation of the generalised forces

For $i=1, 2, \ldots, n$ do:

$$\tau_i = B_i + A_i$$

Complexity of Step 5.

0 $(\times)$, n $(+)$

The computational complexity of the two proposed algorithms are summarised in Table II. Furthermore, the computational complexity of the Luh, Walker and Paul algorithm is reported. In all three cases, the criteria for counting operations are the same and are indicated in Table III.

As can be appreciated from Table II, 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 $129n-74$ $(\times)$ and $97n-71$ $(+)$ for a robot with only rotational joints, so that, for a six degree of freedom robot, the computational complexity would be 700 $(\times)$ and 511 $(+)$.

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

Algorithm		Complexity	$n=6$
Luh, Walker and Paul	$(\times)$	$121n-112$	614
	$(+)$	$90n-82$	458
Algorithm 1	$(\times)$	$\frac{1}{2}(43n^2+59n-43)$	908
	$(+)$	$\frac{1}{2}(33n^2+39n-74)$	674
Algorithm 2	$(\times)$	$117n-136$	566
	$(+)$	$91n-116$	430

https://doi.org/10.1017/S0263574701003502 Published online by Cambridge University Press

Table III. Number of elementary operations considered

Operation*	Multiplications	Additions
${}^{i}\mathbf{I}_i \cdot \vec{a}$	9	6
${}^{i-1}\mathbf{R}_i \cdot \vec{a}$, ${}^{i}\mathbf{R}_{i-1} \cdot \vec{a}$	8	4
$\vec{a} \wedge \vec{b}$	6	3
$\vec{a} \wedge \vec{c}$	2	0
$\vec{a}^T \wedge \vec{b}$	3	2
$\vec{a}^T \wedge \vec{d}$	1	0
$\vec{a} \wedge \vec{b}^T$	9	0
$\vec{a} \wedge \vec{a}^T$	6	0

* Where: $\vec{a}$, $\vec{b}$, $\vec{c}$ and $\vec{d} \in \mathrm{R}^{3\times 1}$ with c_1, d_1 and $d_2 = 0$

4. NUMERICAL EXAMPLE

The presented algorithms have been developed and verified with the symbolic algebra software *MACSYMA*. Computer code has been written in *FORTRAN* for Algorithm 2. The robot we have applied the formulation to, is a PUMA 600 model, taken from the scientific literature. The Denavit-Hartenberg constant parameters are shown in Table IV.

The masses (*kg*.) of the links are as follows:

$$m_1 = 10.521, \quad m_2 = 15.781, \quad m_3 = 8.767, \quad m_4 = 1.052,$$
$$m_5 = 1.052, \quad m_6 = 0.351$$

The coordinates (*m*.) of the centre of masses in the local reference system of the links, are:

$${}^{1}\vec{r}_{O_1,G_1} = [0\ -0.054\ 0]^{\mathrm{T}}, \quad {}^{2}\vec{r}_{O_2,G_2} = [0.1398\ 0\ 0.14909]^{\mathrm{T}},$$
$${}^{3}\vec{r}_{O_3,G_3} = [-0.32 \,.\, 10^{-3}\ -0.197\ 0]^{\mathrm{T}}, \quad {}^{4}\vec{r}_{O_4,G_4} = [0\ 0\ -0.057]^{\mathrm{T}},$$
$${}^{5}\vec{r}_{O_5,G_5} = [0\ -0.007\ 0]^{\mathrm{T}}, \quad {}^{6}\vec{r}_{O_6,G_6} = [0\ 0\ 0.03725]^{\mathrm{T}}$$

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

Table IV. Denavit-Hartenberg parameters

Joint	α (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

$${}^{1}\mathbf{I}_{G_1} = \begin{bmatrix} 1.612 & 0 & 0 \\ 0 & 0.5091 & 0 \\ 0 & 0 & 1.612 \end{bmatrix},$$

$${}^{2}\mathbf{I}_{G_2} = \begin{bmatrix} 0.4898 & 0 & 0 \\ 0 & 8.0783 & 0 \\ 0 & 0 & 8.2672 \end{bmatrix},$$

$${}^{3}\mathbf{I}_{G_3} = \begin{bmatrix} 3.3768 & 0 & 0 \\ 0 & 0.3009 & 0 \\ 0 & 0 & 3.3768 \end{bmatrix},$$

$${}^{4}\mathbf{I}_{G_4} = \begin{bmatrix} 0.181 & 0 & 0 \\ 0 & 0.181 & 0 \\ 0 & 0 & 0.1273 \end{bmatrix},$$

$${}^{5}\mathbf{I}_{G_5} = \begin{bmatrix} 0.0735 & 0 & 0 \\ 0 & 0.0735 & 0 \\ 0 & 0 & 0.1273 \end{bmatrix},$$

$${}^{6}\mathbf{I}_{G_6} = \begin{bmatrix} 0.0071 & 0 & 0 \\ 0 & 0.0071 & 0 \\ 0 & 0 & 0.0141 \end{bmatrix},$$

In order to verify the proposed algorithm, a straight-line trajectory has been considered. The start point has coordinates $[0.600\ 0.175\ 0.250]^{\mathrm{T}}$ *m* and the end point $[0.018\ 0.757\ 0]^{\mathrm{T}}$ *m*, the constant orientation given by Euler angles ZYZ is (45° 60° 90°). 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 2 the initial configuration of the robot, an intermediate and the final one are depicted. The Inverse Dynamic Problem was solved at 0.1 *s* intervals. The torques required in each joint are depicted in Figure 3 as a function of time. Using a Personal Computer with a Pentium 200 Mhz Processor, the average CPU time required for calculating each Inverse Dynamic Problem was 0.15 *ms*.

5. CONCLUSIONS

In this paper, two algorithms for solving the Inverse Dynamic Problem based on the Gibbs-Appell equations

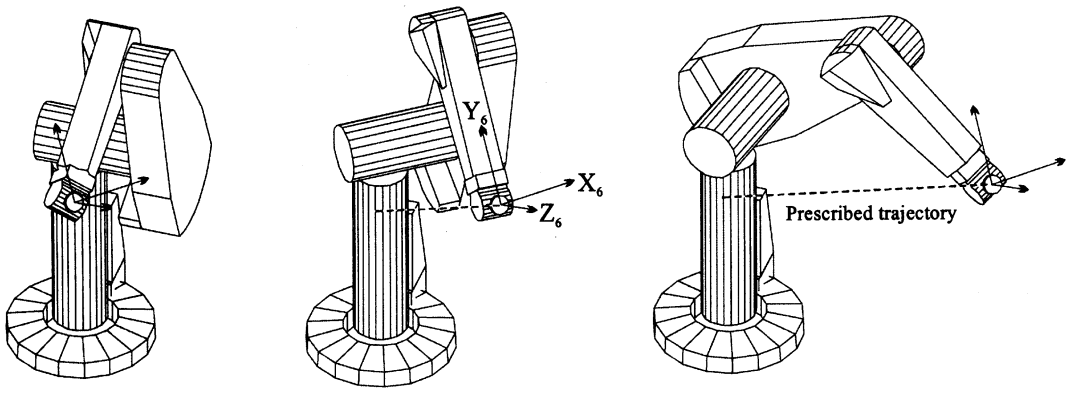

Fig. 2. Prescribed trajectory for the Puma robot.

https://doi.org/10.1017/S0263574701003502 Published online by Cambridge University Press

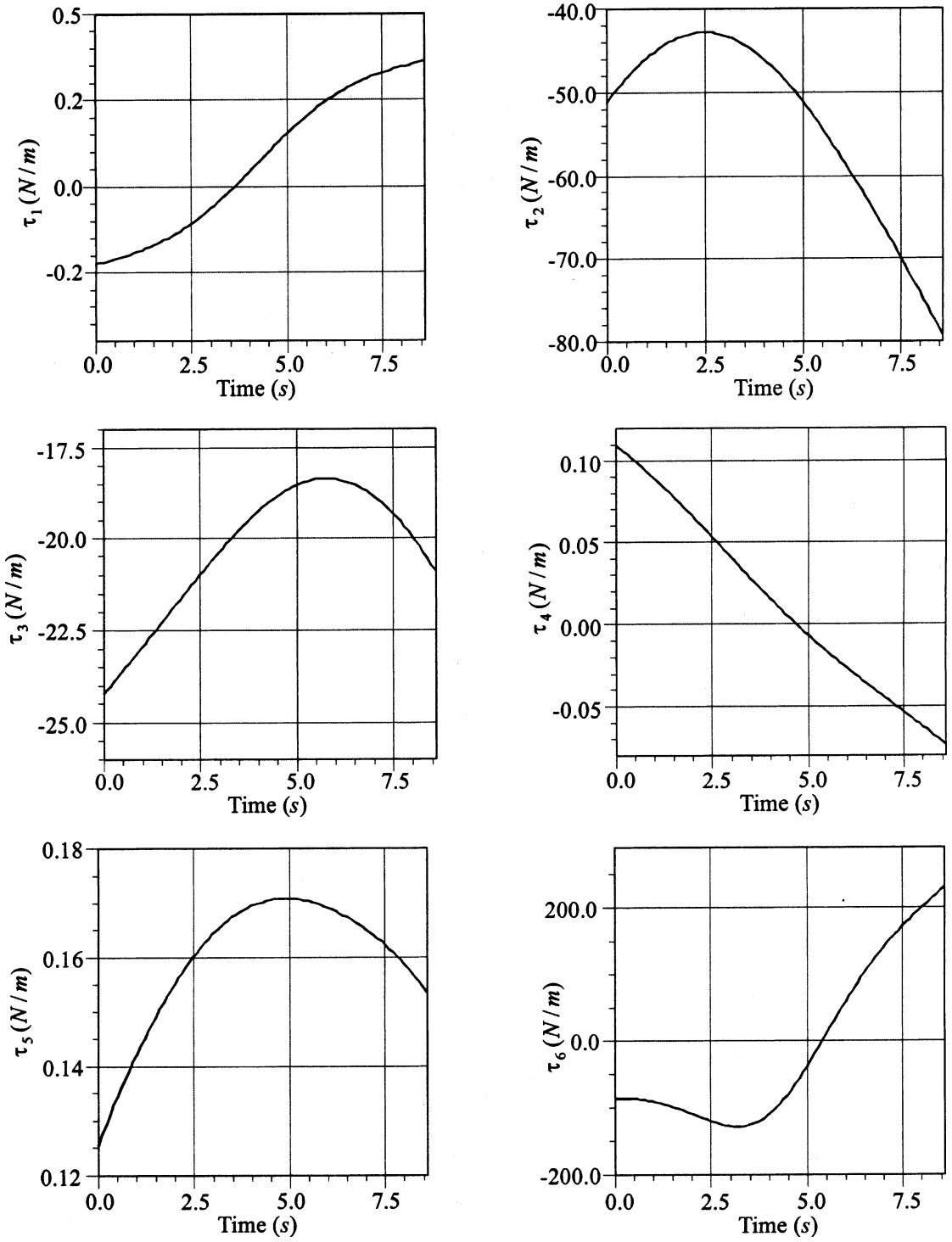

Fig. 3. Torques required per joint.

have been proposed and verified. The most efficient of the proposed algorithms has a computational complexity 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.

ACKNOWLEDGEMENTS

This work has been partially funded by contract number TAP95-0883-C03-0 from *Plan Nacional de Investigación* (Spanish National Research Programme).

References

1. J.M. Hollerbach, "A Recursive Lagrangian Formulation of Manipulator Dynamics and a Comparative Study of Dynam-

https://doi.org/10.1017/S0263574701003502 Published online by Cambridge University Press

ics Formulation Complexity", *IEEE Trans. on Systems, Man, and Cybernetics* **SMC-10**(11), 730-736 (1980).
2. J.Y.S Luh, M.W. Walker and R. Paul "P.C. On-line Computational Scheme for Mechanical Manipulators. Journal of Dynamic Systems", *J. Dynamic Systems, Measurement, and Control* **102**, 69-76 (1980).
3. J. Angeles, O. Ma and A. Rojas, "An Algorithm for the Inverse Dynamics of n-Axis General Manipulators Using Kane's Equations", *Computers Math. Applic.* **17**(12), 1545-1561 (1989).
4. C.A. Balafoutis and R.A. Patel, *Dynamic Analysis of Robot Manipulators: A Cartesian Tensor Approach* (Kluwer Academic Press, Boston, 1991).
5. W. Khalil and J.F. Kleinfinger, "Minimum Operations and Minimum Parameters of the Dynamic Models of Tree Structure Robots", *IEEE Journal of Robotics and Automation* **3**(6), 517-526 (1987).
6. C.S.G. Lee and P.R. Chang, "Efficient Parallel Algorithm For Robot Inverse Dynamics Computation", *IEEE Trans. on Systems, Man, and Cybernetics* **16**(4), 532-542 (1986).
7. P.K. Khosla and C.P. Neuman, "Computational Requirements of Customized Newton-Euler Algorithms", *Journal of Robotic Systems* **2**(3), 309-327 (1985).
8. J.J. Murray and C.P. Neuman, "Organizing Customized Robot Dynamics Algorithms for Efficient Numerical Evaluation", *IEEE Trans. on Systems, Man, and Cybernetics* **18**(1), 115-125 (1988).
9. K.S. Fu, R.C. Gonzalez and C.S.G. Lee, *Robotics: Control, Sensing, Vision, and Intelligence* (McGraw-Hill, Singapore, 1987).
10. A.Y. Zomaya, *Modelling and Simulation of Robot Manipulators: A Parallel Processing Approach*, (World Scientific Publishing Co., Singapore, 1992).
11. J.J. Craig, *Introduction to Robotics: Mechanics and Control* (Addison-Wesley, Reading, 1986).
12. L.A. Pars, *A Treatise on Analytical Dynamics* (Ox Bow Press, Woodbridge, Connecticut 1979).
13. M. Renaud, "Contribution a l'etude de la modélisation et de la commande des systèmes mécaniques articulés", *Thèse doctoral* (Université Paul Sabatier, Toulouse, 1975).
14. M.Vukobratovic and V. Potkonjak, *Applied dynamics and CAD of manipulation robots* (Springer-Verlag, Berlin, 1985).
15. K. Desoyer and P. Lugner, "Recursive formulation for the analytical or numerical application of the Gibbs-Appell method to the dynamics of robots", *Robotica* **7**, Part 4, 343-347 (1989).

https://doi.org/10.1017/S0263574701003502 Published online by Cambridge University Press