PERGAMON

Mechanism and Machine Theory 37 (2002) 739–755

MECHANISM AND MACHINE THEORY

www.elsevier.com/locate/mechmt

Serial-robot dynamics algorithms for moderately large numbers of joints

V. Mata [a,*], S. Provenzano [b], F. Valero [a], J.I. Cuadrado [a]

[a] *Departamento de Ingeniería Mecánica y de Materiales, Universidad Politécnica de Valencia, Camino de Vera s/n, 46022 Valencia, Spain*

[b] *Escuela de Ingeniería Mecánica, Universidad de Los Andes, Av. D. Tulio Febres Cordero, 5101, Mérida, Venezuela*

Received 8 May 2001; accepted 28 January 2002

Abstract

A method for solving the complete dynamic problem in robots with rigid links and ideal joints using the Gibbs–Appell equations as starting point is presented. The inverse dynamic problem is solved through a algorithm O(n), where tensor notation is used. The terms of the generalized inertia matrix are calculated by means of the Hessian of the Gibbs function with respect to generalized accelerations, and a recursive algorithm of order O(n^2) is developed. Proposed algorithms are computationally efficient for serial-robots with moderately large numbers of joints. The numerical stability of the proposed algorithms is analyzed and compared with those of other methods by the use of numerical examples. © 2002 Elsevier Science Ltd. All rights reserved.

Resumen

En este artículo se presenta un método para la resolución del problema dinámico en robots industriales con barras rígidas y pares cinemáticos ideales. Los algoritmos propuestos para la resolución del problema dinámico se han obtenido tomando como punto de partida las ecuaciones de Gibbs–Appell. El problema dinámico inverso se ha resuelto mediante un algoritmo de complejidad O(n), en su desarrollo se ha empleado notación tensorial. Los términos de la matriz de inercia generalizada se han obtenido mediante el Hessiano de la función de Gibbs, respecto a las aceleraciones generalizadas, habiéndose desarrollado un algoritmo recursivo O(n^2). Los algoritmos propuestos se han demostrado eficaces para robots con un número moderadamente grande de grados de libertad. Se ha analizado la estabilidad computacional de los algoritmos. © 2002 Elsevier Science Ltd. All rights reserved.

[*] Corresponding author: Tel.: +34-96-387-76-21; fax: +34-96-387-76-29.
E-mail address: vmata@mcm.upv.es (V. Mata).

1. Introduction

In the last three decades numerous investigators have used different principles of dynamics in order to obtain the equations that model the dynamic behavior of robot arms. The first formulations to be developed were based on a closed form representation of the equations, and the Lagrange–Euler (L–E) equations were used preferentially for this purpose. These formulations were found to be inefficient due to the high number of algebraic operations involved. A solution to this problem was found with the use of relationships present in the dynamic equations. The Newton–Euler (N–E) equations were found to be the most appropriate dynamic principle for this type of formulation and they have been used to develop the most efficient formulations that are known. Other formulations, based on Kane's equations, have yielded algorithms whose computational complexity is similar to that found in formulations based on N–E equations. The use of dynamic principles different from those employed in formulations based on L–E, N–E or Kane's equations has been minor and, furthermore, has produced formulations of high computational complexity. Currently it is believed that the use of diverse dynamic principles will lead to similar formulations of equivalent computational complexity. This has been partially proved by applying the appropriate relationships to the L–E equations in order to obtain an equivalent formulation to that given by the N–E equations, although a greater effort is required in order to reach the final equations [1]. It is for this reason that most of the formulations that produce efficient algorithms have been developed from the N–E equations. Featherstone and Orin [2] make a detailed review of these methods and of the derived algorithms.

The Gibbs–Appell (G–A) equations are one of the principles that have been used the least for the resolution of the dynamic problem of manipulating robots. Discovered independently by Gibbs in 1879 and Appell in 1899, it is said that "they probably are the simplest and most comprehensive form of motion equation ever discovered" [3]. The simple form with which these equations deal with mechanical systems subjected to holonomic and non-holonomic type constraints is also emphasized in the specialized technical literature. Surprisingly, a bibliographical review of the literature on this area reveals the limited use of the G–A equations in modern dynamics. A few years ago, the supposed relationship of the G–A equations and Kane's dynamic equations caused a good number of works and commentaries on the matter (see [4–7]). More recently, Udwadia and Kalaba published a work in which new forms of the G–A equations were developed [8]. In the field of robotics, Popov proposed a method, later developed by Vukobratovic and Potkonjak [9], in which the G–A equations were used to develop a closed form representation of high computational complexity. This method was used by Desoyer and Lugner [10] to solve, by means of a recursive formulation $O(n^2)$, the inverse dynamic problem using the Jacobian matrix of the manipulator, with the purpose of avoiding the explicit development of the partial derivatives. Another approach was suggested by Vereshcahagin [11] that proposed manipulator motion equations from Gauss' principle and Gibbs' function. This approach was used by Rudas and Toth [12] to solve the inverse dynamic problem of robots. Recently, Mata et al. [13] presented a formulation of order $O(n)$ which solves the inverse dynamic problem and establishes recursive relations that involve a reduced number of algebraic operations.

The algorithms that model the dynamic behavior of manipulators are divided in two types: algorithms that solve the inverse dynamic problem and those that give solution to the forward dynamic problem. In the former, the forces exerted by the actuators are obtained algebraically for certain configurations of the manipulator (position, velocity and acceleration). On the other hand, the forward dynamic problem computes the acceleration of the joints of the manipulator once the forces exerted by the actuators are given. This problem is part of the process that must be followed to perform the simulation of the dynamic behavior of the manipulator. This process is completed after it calculates the velocity and position of the joints by means of a process of numerical integration in which the acceleration of the joints and the initial configuration are data input to the problem.

The first efficient recursive algorithm for the solution of the inverse dynamic problem was proposed by Luh et al. [14]. This algorithm, based on the N–E equations, has been improved repeatedly throughout the years [2]. Other authors have developed efficient recursive algorithms to solve the inverse dynamic problem based on other principles of dynamics. As examples of these we have the work of Hollerbach [15] that uses the L–E equations; and those of Kane and Levinson [16], and Angeles et al. [17], which use Kane's equations. On the other hand, the algorithms developed to solve the forward dynamic problem use, regardless of the dynamics principle from which they are derived, one of the following approaches:

- Calculation of the accelerations of the joints by means of the proposal and solution of a system of simultaneous equations.
- Recursive calculation of the acceleration of the joints, propagating motion and constraint forces throughout the mechanism.

The algorithms derived from the methods that use the first approach require the calculation of the generalized inertia matrix and the bias vector. The generalized inertia matrix is also used in advanced control schemes, as well as in parameter estimation procedures. For this reason its calculation, by means of simple and efficient procedures, is beneficial to other fields and not only to mechanical systems motion simulation. The generalized inertia matrix can be obtained through the Hessian of the kinetic energy of the mechanical system with respect to generalized velocities; however, the most computationally efficient algorithms are not based on this procedure. The best known method that follows this first approach was proposed by Walker and Orin [18] which developed (using N–E equations) the method of the composed rigid body (CRBA), in which the generalized inertia matrix is obtained recursively with a complexity $O(n^2)$. Angeles and Ma [19] proposed another method that follows this approach which is based on the calculation of the natural orthogonal complement (NOC) of the manipulator kinematic constraint equations with a complexity $O(n^3)$, using Kane's equations to obtain the bias vector.

On the other hand, algorithms derived from methods that use the second approach usually have a complexity $O(n)$. These algorithms do not obtain the generalized inertia matrix, and for this reason their application is limited to system motion simulations. The best known method among those that use the second approach is the articulated body (ABA) method developed by Featherstone [20]. The number of required algebraic operations is inferior to those needed in the CRBA method, but only for systems that contain nine or more bodies. In [21], Saha performed symbolically the Gaussian elimination to obtain the decomposition of the generalized inertia matrix. As an application of this decomposition, he proposed an $O(n)$ forward dynamic algorithm with a computational complexity very similar to that of [20].

The differences between the CRBA and ABA methods have been studied by some investigators during the last decade. Ascher et al. [22] demonstrated, by solving the forward dynamic problem in a system made up of two bodies with a high condition number, that the precision of the results obtained with the ABA method is higher than in the CRBA method, due to inherent truncating problems in the CRBA method.

In this work the solution of the forward and inverse dynamic problems of robot manipulators are considered using the G–A equations. The resolution of the forward dynamic problem is done by means of a formulation that follows the first approach. In order to obtain the generalized inertia matrix, a recursive algorithm $O(n^2)$ has been developed. The numerical results obtained by this algorithm are compared with those of the ABA and CRBA methods, with the purpose of determining their numerical precision. A modified Denavit–Hartenberg (D–H) notation is used in the kinematic recursive equations [19]. The work is organized as follows:

- In Section 2 a recursive formulation is developed to solve the inverse dynamic problem.
- In Section 3 a formulation is developed to solve the forward dynamic problem.
- The algorithms derived from the proposed formulations are shown in Section 4.
- In Section 5 an example is used to analyze the numerical errors that the method produces in the solution of the proposed forward dynamic problem; additionally, as an example and validation, we perform a simulation of the motion of a PUMA robot that follows a pre-established trajectory.
- Finally, in Section 6, the conclusions of the work are presented.

2. The inverse dynamic problem

The G–A dynamic equations [3] come from the definition of Gibbs' function which, written in general form for an arbitrary body composed of n particles of mass m_i and acceleration a_i in an inertial coordinate frame, is:

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

It is necessary to obtain this function for a system of rigid bodies expressed in the reference systems that are local to the links, so the rotation matrices are used $({}^{i-1}R_i)$

$$\begin{aligned} G_i = {} & \tfrac{1}{2} m_i \left({}^0\mathbf{R}_i {}^i\ddot{\vec{r}}_i\right)^{\mathrm{T}} {}^0\mathbf{R}_i {}^i\ddot{\vec{r}}_i + \tfrac{1}{2}\left({}^0\mathbf{R}_i {}^i\dot{\vec{\omega}}_i\right)^{\mathrm{T}} {}^0\mathbf{R}_i {}^i\mathbf{I}_i \left({}^0\mathbf{R}_i\right)^{\mathrm{T}} \left({}^0\mathbf{R}_i {}^i\dot{\vec{\omega}}_i\right) \\ & + \left({}^0\mathbf{R}_i {}^i\dot{\vec{\omega}}_i\right)^{\mathrm{T}} \left\{ {}^0\mathbf{R}_i {}^i\vec{\omega}_i \left[{}^0\mathbf{R}_i {}^i\mathbf{I}_i \left({}^0\mathbf{R}_i\right)^{\mathrm{T}} {}^0\mathbf{R}_i {}^i\vec{\omega}_i \right] \right\} + f_i(\vec{\omega}_i) \end{aligned} \tag{2}$$

where ${}^i\ddot{\vec{r}}_i$ is the acceleration of the center of gravity for body ith, ${}^i\vec{\omega}_i$ its angular velocity, ${}^i\dot{\vec{\omega}}_i$ its angular acceleration and ${}^i\mathbf{I}_i$ the inertial tensor referred to its center of mass, all of them expressed in the local reference system. The function $f_i(\vec{\omega}_i)$ comprises all expressions that do not include generalized accelerations, there is no need to know it (see Eq. (5)).

Simplifying, Gibbs' function is reduced to the following expression:

$$G_i = \tfrac{1}{2} m_i \left({}^i\ddot{\vec{r}}_i\right)^{\mathrm{T}} {}^i\ddot{\vec{r}}_i + \tfrac{1}{2}\left({}^i\dot{\vec{\omega}}_i\right)^{\mathrm{T}} {}^i\mathbf{I}_i {}^i\dot{\vec{\omega}}_i + \left({}^i\dot{\vec{\omega}}_i\right)^{\mathrm{T}} \left({}^i\vec{\omega}_i \times {}^i\mathbf{I}_i {}^i\vec{\omega}_i\right) \tag{3}$$

For a system composed of n bodies, the total Gibbs' function is given by

$$G = \sum_{i=1}^{n} G_i \quad (i = 1, 2, \ldots, n) \tag{4}$$

The G–A equations are obtained by taking the derivative of Gibbs' function with respect to the generalized accelerations $\ddot{q}_j$:

$$\tau_j = \sum_{i=j}^{n} \frac{\partial G_i}{\partial \ddot{q}_j} \quad (j = 1, 2, \ldots, n) \tag{5}$$

therefore

$$\tau_j = \sum_{i=j}^{n} \left\{ m_i \left(\frac{\partial {}^i\ddot{\vec{r}}_i}{\partial \ddot{q}_j}\right)^{\mathrm{T}} {}^i\ddot{\vec{r}}_i + \left(\frac{\partial {}^i\dot{\vec{\omega}}_i}{\partial \ddot{q}_j}\right)^{\mathrm{T}} {}^i\mathbf{I}_i {}^i\dot{\vec{\omega}}_i + \left(\frac{\partial {}^i\dot{\vec{\omega}}_i}{\partial \ddot{q}_j}\right)^{\mathrm{T}} \left[{}^i\vec{\omega}_i \times \left({}^i\mathbf{I}_i {}^i\vec{\omega}_i\right)\right] \right\} \tag{6}$$

where $\vec{\tau}$ is the generalized force vector. The calculation of the partial derivatives in the previous expression is simplified if the following series of suppositions are into:

$$\frac{\partial {}^i\dot{\vec{\omega}}_i}{\partial \ddot{q}_j} = \mathbf{0}, \quad \text{for } i < j \tag{7a}$$

$$\frac{\partial {}^i\dot{\vec{\omega}}_i}{\partial \ddot{q}_j} = {}^i\mathbf{R}_{i-1} {}^{i-1}\mathbf{R}_{i-2} \cdots {}^{j-1}\mathbf{R}_j \frac{\partial {}^j\dot{\vec{\omega}}_j}{\partial \ddot{q}_j}, \quad \text{for } i > j \tag{7b}$$

$$\frac{\partial {}^i\dot{\vec{\omega}}_i}{\partial \ddot{q}_j} = {}^i\vec{z}_i, \quad \text{for } i = j \tag{7c}$$

$$\frac{\partial {}^i\ddot{\vec{r}}_i}{\partial \ddot{q}_k} = \frac{\partial {}^i\dot{\vec{\omega}}_i}{\partial \ddot{q}_k} \times {}^i\vec{r}_{k,i} \text{ or, in tensor notation, } \frac{\partial {}^i\ddot{\vec{r}}_i}{\partial \ddot{q}_k} = \left(\frac{\partial {}^i\dot{\vec{\omega}}_i}{\partial \ddot{q}_k}\right) {}^i\tilde{r}_{k,i} \tag{7d}$$

where ${}^i\vec{r}_{k,i}$ is the position vector that goes from the origin of the kth body's local reference system to ith body's center of gravity, ${}^i\tilde{r}_{k,i}$ is the associated skew-symmetric tensor, and ${}^i\vec{z}_i = [0 \quad 0 \quad 1]^{\mathrm{T}}$. In these suppositions, it is assumed that the manipulator is made up of revolute joints; in case joint i is prismatic, the last expression contains an additional term:

$$\frac{\partial {}^i\ddot{\vec{r}}_i}{\partial \ddot{q}_k} = \frac{\partial {}^i\dot{\vec{\omega}}_i}{\partial \ddot{q}_k} \times {}^i\vec{r}_{k,i} + {}^i\vec{z}_i, \text{ or, in tensor notation, } \frac{\partial {}^i\ddot{\vec{r}}_i}{\partial \ddot{q}_k} = \left(\frac{\partial {}^i\dot{\vec{\omega}}_i}{\partial \ddot{q}_k}\right)^{\mathrm{T}} {}^i\tilde{r}_{k,i} + {}^i\vec{z}_i \tag{7e}$$

In Eq. (6) two terms are recognized: the first one is the force that acts on the body's center of mass, and the second term corresponds to the moment that acts on the body with respect to the center of mass. In tensor notation the simplification of the second term is feasible, which reduces, in addition, the number of necessary algebraic operations for its computation. Taking into account the mentioned simplification, Eq. (6) can be written as follows:

$$\tau_j = \sum_{i=j}^{n} \left[\left(\frac{\partial^i \ddot{\vec{r}}_i}{\partial \ddot{q}_j} \right)^{\mathrm{T}} {}^i\vec{f}_i \right] + \sum_{i=j}^{n} \left[\left(\frac{\partial^i \dot{\vec{\omega}}_i}{\partial \ddot{q}_j} \right)^{\mathrm{T}} {}^i\vec{t}_i \right] \tag{8}$$

where,

$${}^i\vec{f}_i = m_i \left({}^i\ddot{\vec{r}}_i + {}^i\vec{g} \right) \tag{9}$$

In the last expression, ${}^i\vec{g}$ denotes the acceleration of gravity expressed in the local coordinate system bound to body i. The moment that acts on body i with respect to its center of mass can be simplified with the aid of tensor algebra: [1]

$${}^i\tilde{t}_i = {}^i\Omega_i {}^i\mathbf{J}_i - ({}^i\Omega_i {}^i\mathbf{J}_i)^{\mathrm{T}} \tag{10}$$

where, ${}^i\tilde{t}_i$ is skew-symmetric tensor representation of the ${}^i\vec{t}_i$ vector, ${}^i\mathbf{J}_i$ is Euler's tensor for a rigid body whose computation is made off-line by means of the following expression,

$${}^i\mathbf{J}_i = \tfrac{1}{2}\mathrm{trace}[{}^i\mathbf{I}_i]\mathbf{1}_{3\times3} - {}^i\mathbf{I}_i \tag{11}$$

and ${}^i\Omega_i$ is the angular acceleration tensor,

$${}^i\Omega_i = {}^i\tilde{\dot{\omega}}_i + {}^i\tilde{\omega}_i^2 \tag{12}$$

Eq. (8) can be further simplified by using the following expression:

$$\left(\frac{\partial^i \ddot{\vec{r}}_i}{\partial \ddot{q}_j} \right)^{\mathrm{T}} = \left(\frac{\partial^i \dot{\vec{\omega}}_i}{\partial \ddot{q}_j} \right)^{\mathrm{T}} {}^i\tilde{r}_{j,i}$$

Replacing in Eq. (8) and simplifying:

$$\tau_j = \sum_{i=j}^{n} \left[\left(\frac{\partial^i \dot{\vec{\omega}}_i}{\partial \ddot{q}_j} \right)^{\mathrm{T}} \left({}^i\tilde{r}_{j,i} {}^i\vec{f}_i + {}^i\vec{t}_i \right) \right] \tag{13}$$

If an algorithm for the computation of the generalized forces is formulated using Eq. (13), it will have a computational complexity $\mathrm{O}(n^2)$. The procedure to follow in order to obtain an algorithm that involves a complexity $\mathrm{O}(n)$ should find a recurrent relationship that eliminates the summation in the equation. The following expressions possess a recurrent relationship that is able to effectively reduce the computational complexity of the resulting algorithm:

$$\tau_j = \left(\frac{\partial^j \dot{\vec{\omega}}_j}{\partial \ddot{q}_j} \right)^{\mathrm{T}} {}^j\vec{\chi}_j \tag{14}$$

where,

$${}^j\vec{\chi}_j = {}^j\tilde{r}_{j,j} {}^j\vec{f}_j + {}^j\tilde{s}_{j,j+1} {}^j\vec{\phi}_j + {}^j\vec{t}_j + {}^j\mathbf{R}_{j+1} {}^{j+1}\vec{\chi}_{j+1} \tag{15}$$

[1] A proof of this statement and a definition of Euler's tensor for a rigid body, can be found in Ref. [23], pp. 109–111.

and

$$ {}^{j}\vec{\phi}_j = {}^{j}\mathbf{R}_{j+1}\left({}^{j+1}\vec{f}_{j+1} + {}^{j+1}\vec{\phi}_{j+1}\right) \tag{16}$$

In (15) ${}^{j}\tilde{s}_{j,k}$ is the skew-symmetric tensor associated with the position vector ${}^{j}\vec{s}_{j,k}$ that goes from the origin of the *j*th body's local reference system to the *k*th. Expressions (14)–(16) provide the basis for the elaboration of the algorithm presented in Section 4.1, which solves the inverse dynamic problem of robot manipulators with a complexity of order O(n).

3. The forward dynamic problem

In the previous section, the G–A equations are described as function of quantities expressed in Cartesian coordinates. It is possible to write the G–A equations based on generalized coordinates and their time derivatives. This formulation was obtained by Vukobratovic and Potkonjak [9]. These equations are contained in the following matrix expression:

$$\mathbf{D}(\vec{q})\ddot{\vec{q}} = \vec{\tau} - \vec{C}(\vec{q},\dot{\vec{q}}) + \vec{G}(\vec{q}) \tag{17}$$

where the generalized inertia matrix (**D**) and the bias vector ($\vec{C} + \vec{G}$), can be calculated by means of recursive algorithms developed separately, and $\vec{\tau}$ represents the generalized forces vector. This equation, similar to that obtained from other dynamics principles, constitutes the basis of the method for the resolution of the forward dynamic problem.

In this work, the bias vector is calculated by using the algorithm developed here to solve the inverse dynamic problem, in which the terms corresponding to the joint accelerations have been eliminated, according to the method proposed by Walker and Orin [18].

In order to obtain the generalized inertia matrix, a new method based on the G–A equations is developed. The development of the method begins with Eq. (17), which can be written in the following way:

$$\frac{\partial G}{\partial \ddot{\vec{q}}} = \mathbf{D}(\vec{q})\ddot{\vec{q}} + \vec{C}(\vec{q},\dot{\vec{q}}) - \vec{G}(\vec{q}) \tag{18}$$

If we take the partial derivative of the previous equation once again, an expression is obtained for the calculation of the generalized inertia matrix by means of Gibbs' function Hessian matrix:

$$D_{jk} = \frac{\partial^2 G}{\partial \ddot{q}_j \partial \ddot{q}_k} \tag{19}$$

where **D** is the system's generalized inertia matrix. From the previous expression a simple algorithm of order O(n^3) for the calculation of **D** is obtained in a natural way, although it is possible to reduce it to a recursive algorithm of order O(n^2). The procedure used to find the expression that allows us to calculate the coefficients of the generalized inertia matrix, by means of a process of successive partial derivatives of Gibbs' function, is simplified taken into account:

$$\frac{\partial}{\partial \ddot{q}_j}\frac{\partial {}^{i}\dot{\vec{\omega}}_i}{\partial \ddot{q}_k} = \frac{\partial}{\partial \ddot{q}_j}{}^{i}\vec{z}_i = 0$$

$$\frac{\partial}{\partial \ddot{q}_j}\frac{\partial {}^i\ddot{\vec{s}}_i}{\partial \ddot{q}_k} = \frac{\partial}{\partial \ddot{q}_j}{}^i\mathbf{R}_{i-1}\left[\frac{\partial^{i-1}\ddot{\vec{s}}_{i-1}}{\partial \ddot{q}_k} + \frac{\partial^{i-1}\dot{\vec{\omega}}_{i-1}}{\partial \ddot{q}_k} \times {}^{i-1}\vec{s}_{i-1,i}\right] = 0 \tag{20}$$

$$\frac{\partial}{\partial \ddot{q}_j}\frac{\partial {}^i\ddot{\vec{r}}_i}{\partial \ddot{q}_k} = \frac{\partial}{\partial \ddot{q}_j}\left(\frac{\partial^{i}\ddot{\vec{s}}_{i}}{\partial \ddot{q}_k} + \frac{\partial^{i}\dot{\vec{\omega}}_{i}}{\partial \ddot{q}_k} \times {}^{i}\vec{r}_{i,i}\right) = 0$$

where, ${}^i\ddot{\vec{s}}_i$ is the acceleration of the origin of *i*th body's local reference system. In this case the expressions have the same form for prismatic and revolute couples. Using Eq. (20), the expression for the calculation of the coefficients of the generalized inertia matrix takes the following form:

$$D_{jk} = \sum_{i=j}^{n}\left[m_i\left(\frac{\partial^i\ddot{\vec{r}}_i}{\partial \ddot{q}_k}\right)^{\mathrm{T}}\frac{\partial^i\ddot{\vec{r}}_i}{\partial \ddot{q}_j} + \left(\frac{\partial^i\dot{\vec{\omega}}_i}{\partial \ddot{q}_k}\right)^{\mathrm{T}}{}^i\mathbf{I}_i\frac{\partial^i\dot{\vec{\omega}}_i}{\partial \ddot{q}_j}\right] \tag{21}$$

The use of the expression (21) to develop an algorithm applicable to manipulators, that is computationally achievable, requires the study of the recurrent relationships that exist between their elements, since the direct application of the expression would produce an algorithm with a high number of operations. Taking into account Eqs. (7a)–(7e), and the properties of cross products, then expression (21) can be written as:

$$D_{jk} = \sum_{i=j}^{n}\left[\left({}^i\mathbf{R}_k\frac{\partial^k\dot{\vec{\omega}}_k}{\partial \ddot{q}_k}\right)^{\mathrm{T}}\left({}^i\mathbf{I}_i - m_i{}^i\tilde{r}_{i,i}{}^i\tilde{r}_{j,i} - m_i{}^i\tilde{s}_{k,i}{}^i\tilde{s}_{j,i}\right)\frac{\partial^i\dot{\vec{\omega}}_i}{\partial \ddot{q}_j}\right] \tag{22}$$

In Section 4.2 an algorithm for the calculation of the generalized inertia matrix is presented based on this expression.

4. Proposed algorithms

4.1. An algorithm to solve the inverse dynamic problem

Now we shall present an algorithm that results from the expressions developed in Section 2. The algorithm only contemplates manipulators composed of connected rigid bodies by means of ideal revolute couples. Also, the premultiplication of a vector by a rotation matrix does not take place in the way indicated in the expressions, but trigonometric identities are used which allow a reduction of the number of operations. Additionally, the use of this method makes the calculation of the components of the rotation matrix unnecessary, producing additional savings (see [24] for example). In the following description, each specified algorithmic expression is accompanied by info that indicates the number of algebraic operations that are involved, showing separately the products (M) and the sums (A). At the end of each section there is a summary and the total complexity of the proposed algorithm is compared with other well-known algorithms.

Step 1: In this algorithm the acceleration of gravity is introduced by means of the acceleration of the base of the robot, such as proposed by Luh et al. [14], since this procedure improves the computational complexity of the formulation.

```
Initialize
$^0\ddot{\vec{s}}_0 = -g\,^0\vec{z}_0 \quad (g = \text{gravity})$
$^0\ddot{\vec{r}}_0 = {}^0\ddot{\vec{s}}_0$
For $i = 1, 2, \ldots, n$, do
    $^i\vec{\omega}_{i-1} = {}^i\mathbf{R}_{i-1}{}^{i-1}\vec{\omega}_{i-1}$    8M, 4A
    $^i\vec{\omega}_i = {}^i\vec{\omega}_{i-1} + {}^i\vec{z}_i\dot{q}_i$    0M, 1A
    $^i\dot{\vec{\omega}}_i = {}^i\mathbf{R}_{i-1}{}^{i-1}\dot{\vec{\omega}}_{i-1} + {}^i\vec{z}_i\ddot{q}_i + {}^i\tilde{\omega}_{i-1}({}^i\vec{z}_i\dot{q}_i)$    10M, 7A
    $^i\Omega_i = {}^i\tilde{\dot{\omega}}_i + {}^i\tilde{\omega}_i^2$    6M, 9A
    $^i\ddot{\vec{s}}_i = {}^i\mathbf{R}_{i-1}[{}^{i-1}\ddot{\vec{s}}_{i-1} + {}^{i-1}\Omega_{i-1}{}^{i-1}\vec{s}_{i-1,i}]$    17M, 13A
    $^i\ddot{\vec{r}}_i = {}^i\ddot{\vec{s}}_i + {}^i\Omega_i^i\vec{r}_{i,i}$    9M, 9A
```

Step 2: In this step, Eqs. (9) and (10) of Section 2 are used, with a difference for the first of these expressions where the acceleration of gravity has been eliminated because it was introduced in the previous step.

```
For $i = 1, 2, \ldots, n$, do
    $^i\vec{f}_i = m_i^i\ddot{\vec{r}}_i$    3M, 0A
    $^i\tilde{t}_i = {}^i\Omega_i{}^i\mathbf{J}_i - ({}^i\Omega_i{}^i\mathbf{J}_i)^{\mathrm{T}}$    15M, 15A
```

Step 3: In this step the generalized force or torque exerted by the actuators is calculated by applying expressions (14)–(16) by means of a regressive recursion. In the last recursion, the calculation of all the components of $^1\vec{\chi}_1$ is not necessary, since only the third component is required. Therefore, only the calculation of this component is considered.

```
For $i = n-1, n-2, \ldots, 1$, do
    $^i\vec{\phi}_i = {}^i\mathbf{R}_{i+1}\left({}^{i+1}\vec{f}_{i+1} + {}^{i+1}\vec{\phi}_{i+1}\right)$    8M, 7A
For $i = n, n-1, \ldots, 1$, do
    $^j\vec{\chi}_j = {}^j\tilde{r}_{j,j}{}^j\vec{f}_j + {}^j\tilde{s}_{j,j+1}{}^j\vec{\phi}_j + {}^j\vec{t}_j + {}^j\,\mathbf{R}_{j+1}{}^{j+1}\vec{\chi}_{j+1}$    20M, 19A
For $i = 1, 2, \ldots, n$, do
    $\tau_j = \left(\frac{\partial^j\dot{\vec{\omega}}_j}{\partial\ddot{q}_j}\right)^{\mathrm{T}}{}^j\vec{\chi}_j$    0M, 0A
```

Table 1 shows the computational complexity of the proposed algorithm and a comparison with the complexity of other well-known algorithms.

Table 1
Inverse dynamic problem, complexity of the algorithms

Authors	Principle	Products ($n = 6$)	Sums ($n = 6$)
Luh et al. [14]	N–E	$150n - 48$ (852)	$131n - 48$ (738)
Angeles et al. [17]	Kane	$105n - 109$ (521)	$90n - 105$ (435)
Balafoutis and Patel [23]	N–E	$93n - 69$ (489)	$81n - 66$ (420)
This work	G–A	$96n - 101$ (475)	$84n - 100$ (404)

4.2. Algorithm for the generalized inertia matrix

In the following section we show a recursive algorithm $O(n^2)$ for the calculation of the generalized inertia matrix, based on the formulation obtained in Section 3.

Step 1: Calculation of the composed mass, of the compound inertial tensor, and other time-invariant quantities that can be calculate off-line. ${}^{i}\mathbf{H}_i$ is ith body inertial tensor referred to ith local reference system.

```
Set:
$M_n = m_n$
For $i = n$ to 2, do:
    $M_{i-1} = m_{i-1} + M_i$   0M, 0A
    ${}^{i-1}\vec{\gamma}_{i-1} = m_{i-1}{}^{i-1}\vec{r}_{i-1,i-1} + M_i{}^{i-1}\vec{s}_{i-1,i}$   0M, 0A
    ${}^{i-1}\mathbf{H}_{i-1} = {}^{i-1}\mathbf{I}_{i-1} - m_{i-1}{}^{i-1}\tilde{r}_{i-1,i-1}{}^{i-1}\tilde{r}_{i-1,i-1}$   0M, 0A
    ${}^{i-1}\mathbf{E}_{i-1} = {}^{i-1}\mathbf{H}_{i-1} - M_i{}^{i-1}\tilde{s}_{i-1,i}{}^{i-1}\tilde{s}_{i-1,i}$   0M, 0A
```

Step 2: Calculation of the ${}^{j}\vec{s}_{i,j}$ vectors:

```
For $i = 1$ to $n-1$,
    for $j = i+1$ to $n$, do
        ${}^{j}\vec{s}_{i,i+1} = {}^{j}\mathbf{R}_{j-1}{}^{j-1}\vec{s}_{i,i+1}$   8M, 4A
For $i = n-2$ to 1,
    for $j = n$ to $i+2$, do
        ${}^{j}\vec{s}_{i,j} = {}^{j}\vec{s}_{i,i+1} + {}^{j}\vec{s}_{i+1,j}$   0M, 3A
```

Step 3: Recursive calculation of the $\partial{}^{j}\dot{\vec{\omega}}_j/\partial\ddot{q}_k$ vectors.

```
For $i = 1$ to $n$, set
    $\partial{}^{j}\dot{\vec{\omega}}_j/\partial\ddot{q}_j = {}^{j}\vec{z}_j$   0M, 0A
For $i = 2$ to $n$, set
    $\partial{}^{j}\dot{\vec{\omega}}_j/\partial\ddot{q}_{j-1} = \left[{}^{j}R_{j-1}^{(1,3)} \quad {}^{j}R_{j-1}^{(2,3)} \quad {}^{j}R_{j-1}^{(3,3)}\right]^{\mathrm{T}}$   0M, 0A
For $i = 1$ to $n-1$,
    for $j = i+2$ to $n$, do
        $\partial{}^{j}\dot{\vec{\omega}}_j/\partial\ddot{q}_i = {}^{i}\mathbf{R}_{i-1}(\partial{}^{j-1}\dot{\vec{\omega}}_{j-1}/\partial\ddot{q}_i)$   8M, 4A
```

Step 4: Calculation of the ${}^{i}\vec{\phi}_i$ and ${}^{i-1}\vec{\phi}_i$ vectors, (the $m_i{}^{i}\vec{r}_{i,i}$ product is calculated off-line).

```
For $i = n,\ n-1$ to 1, do
    ${}^{i}\vec{\phi}_{i+1} = {}^{i}\mathbf{R}_{i+1}{}^{i+1}\vec{\phi}_{i+1}$   8M, 4A
    ${}^{i}\vec{\phi}_i = {}^{i}\vec{\gamma}_i + {}^{i}\vec{\phi}_{i+1}$   0M, 3A
```

Step 5: In this step, the recursive calculation of the ${}^{i}\boldsymbol{\psi}_i$ tensors is performed. The similarity transformation performs the calculations in an efficient way by decomposing the ${}^{i}\mathbf{R}_{i+1}{}^{i+1}\boldsymbol{\psi}_{i+1}{}^{i+1}\mathbf{R}_i$ product.

```
Set:
${}^{n}\boldsymbol{\psi}_n = {}^{n}\mathbf{H}_n$
For $i = n-1$ to 2, do
    ${}^{i}\boldsymbol{\psi}_i = {}^{i}\mathbf{E}_i - {}^{i}\tilde{s}_{i,i+1}{}^{i}\vec{\phi}_{i+1} - \left({}^{i}\tilde{s}_{i,i+1}{}^{i}\vec{\phi}_{i+1}\right)^{\mathrm{T}} + {}^{i}\mathbf{R}_{i+1}{}^{i+1}\boldsymbol{\psi}_{i+1}{}^{i+1}\mathbf{R}_i$   31M, 40A
For $i = 1$ only ${}^{1}\boldsymbol{\psi}_1^{(3,3)}$ entry is necessary.   13M, 11A
```

Step 6: Calculation of the elements of the main diagonal of the generalized inertia matrix.

```
For $i = 1$ to $n$,
    $D_{ii} = (\partial^{i}\dot{\vec{\omega}}_i/\partial\ddot{q}_i)^{\mathrm{T}\,i}\boldsymbol{\psi}_i(\partial^{i}\dot{\vec{\omega}}_i/\partial\ddot{q}_i)$   0M, 0A
```

Step 7: Calculation of the remaining elements of the generalized inertia matrix.

```
For $j = n$ to 2,
    For $k = j-1$ to 1, do
        $D_{jk} = (\partial^{j}\dot{\vec{\omega}}_j/\partial\ddot{q}_k)^{\mathrm{T}}\left[{}^{j}\boldsymbol{\psi}_j - {}^{j}\tilde{s}_{k,j}{}^{j}\vec{\phi}_j\right](\partial^{j}\dot{\vec{\omega}}_j/\partial\ddot{q}_j)$   7M, 6A
```

Table 2 shows the computational complexity of the proposed algorithm and a comparison with the complexity of other algorithms from the best known methods.

The solution of the forward dynamic problem is completed with the calculation of the bias vector and the solution of the linear system by means of Cholesky decomposition [25]. As mentioned earlier, the algorithm to calculate the elements of the bias vector has been obtained from the algorithm to solve the inverse dynamic problem developed in Section 4.1, with the only difference between them being that the terms associated with the joint accelerations have been eliminated. Thanks to this there is a slight reduction in the number of required algebraic operations as opposed to those required by the original algorithm. Table 3 shows the total complexity

Table 2
Complexity of several methods for calculating the generalized inertia matrix

Authors	Method	Products ($n = 6$)	Sums ($n = 6$)
Walker and Orin [18]	CRBA	$12n^2 + 56n - 27$ (741)	$7n^2 + 67n - 56$ (598)
Angeles and Ma [19]	NOC	$n^3 + 17n^2 - 21n + 8$ (710)	$n^3 + 14n^2 - 16n + 5$ (629)
This work	GA	$11.5n^2 + 19.5n - 49$ (482)	$8.5n^2 + 31.5n - 69$ (426)

Table 3
Total complexity of the proposed method for solving the forward dynamic problem

	Products ($n = 6$)	Sums ($n = 6$)
Calculation of **D**	$11.5n^2 + 19.5n - 49$ (482)	$8.5n^2 + 31.5n - 69$ (426)
Bias vector calculation	$96n - 108$ (468)	$83n - 105$ (393)
Cholesky dec.	$\frac{1}{6}n^3 + \frac{1}{2}n^2 + \frac{1}{3}n$ (56)	$\frac{1}{6}n^3 + \frac{1}{2}n^2 + \frac{1}{3}n$ (56)
Linear system solving	n^2 (36)	$n^2 - n$ (30)
Total	$\frac{1}{6}n^3 + 13n^2 + \frac{695}{6}n - 157$ (1042)	$\frac{1}{6}n^3 + 10n^2 + \frac{683}{6}n - 174$ (905)

Table 4
Computational cost to solve the forward dynamic problem for $n = 6$

Authors	Method	Products	Sums
Walker and Orin [18]	CRBA	1627	1261
Featherstone [26]	ABA	1533	1415
Angeles and Ma [19]	NOC	1353	1165
This work	GA	1042	905

of the proposed method for solving the forward dynamic problem. Additionally, Table 4 shows a comparison of this method's complexity with that of other well-known methods.

5. Examples

In this section two examples are included. The first example is used to compare the results obtained by our method to solve the forward dynamic problem, with the CRBA and the ABA methods. In the second example, the motion of a PUMA robot is simulated by means of the integration of the mathematical model of the system, to which the necessary torques have been provided, so that the end effector of the manipulator makes a pre-established straight trajectory.

5.1. Numerical stability of the proposed method for solving the forward dynamic problem

We use the first example proposed by Ascher et al. [22]. These authors used this example to demonstrate that the ABA method is numerically more stable than the CRBA method. They proposed a system made up of two bodies with very different dimensions with the purpose of increasing the condition number of the generalized inertia matrix. Then, they solved the forward dynamics problem with single precision, for a set of positions obtained by rotation, through a complete turn of the second body. Fig. 1 shows a representation of the proposed manipulator. Body 1 is bound to a coordinate inertial system by means of a revolute joint in O_1. Body 2 is joined to the previous one by means of a revolute joint (O_2). The links are 0.02 m wide and their lengths are: $l_1 = 0.02$ m, $l_2 = 2.0$ m. Their masses are: $m_1 = 0.1$ kg and $m_2 = 10.0$ kg. The pro-

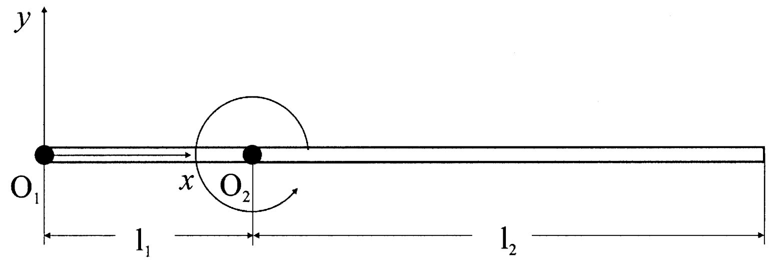

Fig. 1. Manipulator with two degrees of freedom.

cedure contemplates the rotation of body 2 around O_2 at two degree intervals until completing a turn (180 steps). In this work the CRBA and ABA methods are compared with the method developed in Section 4.2 (GA). For each of the methods a FORTRAN program is written using single precision numbers. The Gauss–Jordan elimination is used to solve the linear system for the CRBA and GA methods. The acceleration due to gravity (9.81 m/s^2) acts orthogonal to the length of the links in Fig. 1 (y-axis), and the generalized forces are null.

Fig. 2 shows, as a continuous line, the results obtained by using single precision for each of the methods. The dashed line shows the results calculated by using double precision. In the same figure is depicted the variation in the condition number of the generalized inertia matrix observed during the process. Additionally, Table 5 shows a comparison of the average and maximum differences between the solutions in single and double precision for each of the methods. The solution in double precision is denoted with the DP superscript, and the solution in single precision is denoted with the SP superscript.

As can be observed, the single precision results are closest to the double precision results in the ABA and GA methods, and the CRBA method exhibits the largest average and maximum differences. This is an indication that the operations done to find the generalized inertia matrix and

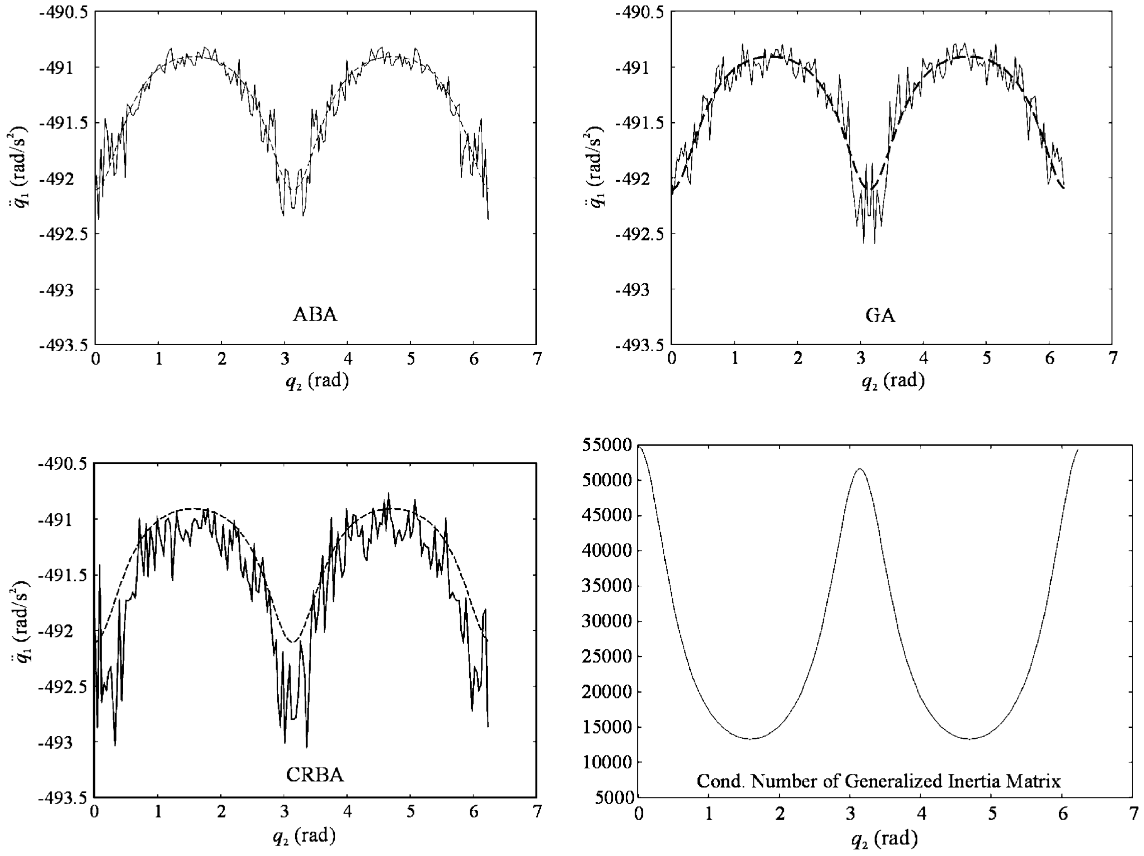

Fig. 2. Comparison of the accelerations obtained in joint 1 and variation of the condition number.

Table 5
Average and maximum differences in the acceleration of joint 1 for three methods

	ABA	CRBA	GA
Max $\Delta(\ddot{q}_2^{\mathrm{DP}} - \ddot{q}_2^{\mathrm{SP}})$ rad/s^2	0.497600	1.292200	0.5340020
Aver. $\Delta(\ddot{q}_2^{\mathrm{DP}} - \ddot{q}_2^{\mathrm{SP}})$ rad/s^2	0.013240	0.2750630	0.0087611

bias vector by the GA method developed in this work produce results that are numerically more stable than those obtained by the CRBA method.

5.2. Simulation of the motion of a PUMA robot

In this example the simulation of the motion of a PUMA robot is proposed, whose terminal element describes a straight trajectory, with constant attitude and constant velocity of 0.1 m/s. The simulation starts from the position ${}^0\vec{r}_{\mathrm{O_0,TCP}}|_{t=0\ \mathrm{s}} = [\,0.50 \quad 0.00 \quad 0.05\,]^{\mathrm{T}}$ m, with an inclination (Z, Y, Z) of the local system related to body 6 given by $[\,0° \quad 90° \quad 180°\,]$. After 5.0 s of motion the terminal element reaches the position ${}^0\vec{r}_{\mathrm{O_0,TCP}}|_{t=5\ \mathrm{s}} = [\,0.62 \quad 0.30 \quad 0.11\,]^{\mathrm{T}}$ m. In order to imple-

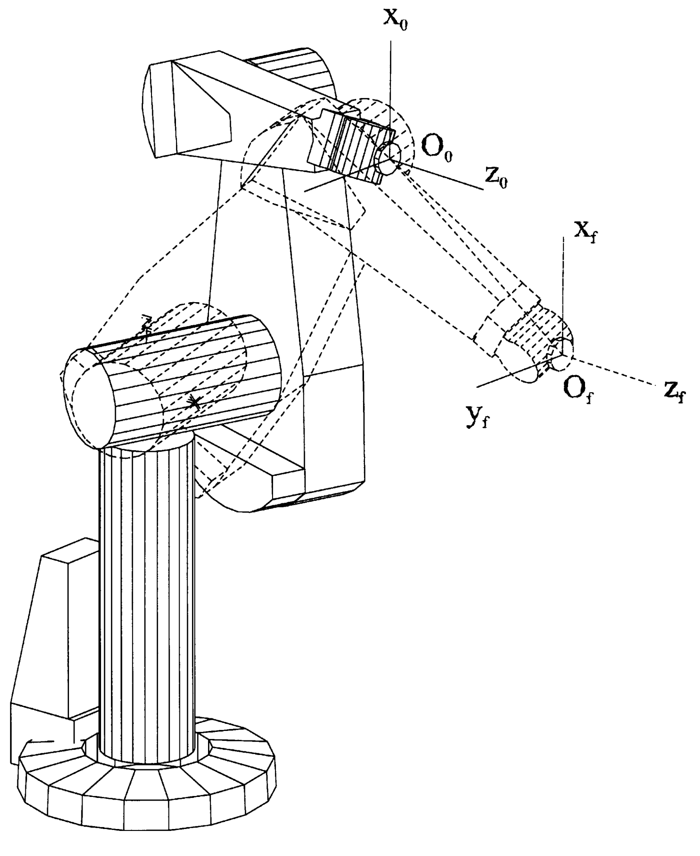

Fig. 3. Initial, final position and trajectory.

ment the simulation proposed in this example, a simulation program was written in FORTRAN based on the method explained in Section 4.2. The program written in double precision and runs on a 400 MHz Pentium II PC. The linear system was solved using Cholesky decomposition and the fourth/fifth order Runge–Kutta technique is used to integrate the differential equations. In the integration process, a tolerance of 1×10^{-6} is used and an interval of 0.1 s is employed. Fig. 3 shows the robot in the initial position; the planned trajectory; and the final position. In order to obtain the generalized forces that cause the end effector to describe the programmed trajectory, the inverse dynamic problem for every instant has been solved, using as input data the results of the resolution of the inverse kinematic problem. Fig. 4 shows a block diagram of the procedure, where the superscript "$*$" denotes the generalized coordinates, velocities, and accelerations obtained by the numerical integration of the method developed in this work. Table 6 shows the resulting average errors in angle, velocity and acceleration of the joints, compared with the results of the inverse kinematics problem.

The same simulation is made by means of programs based on the CRBA and ABA methods and in both cases the results are practically the same than with the GA method. The difference among these methods that can be detected in this case is in the processing time required for the simulation and in the number of times that the integration function is evaluated. Table 7 shows these quantities for comparison.

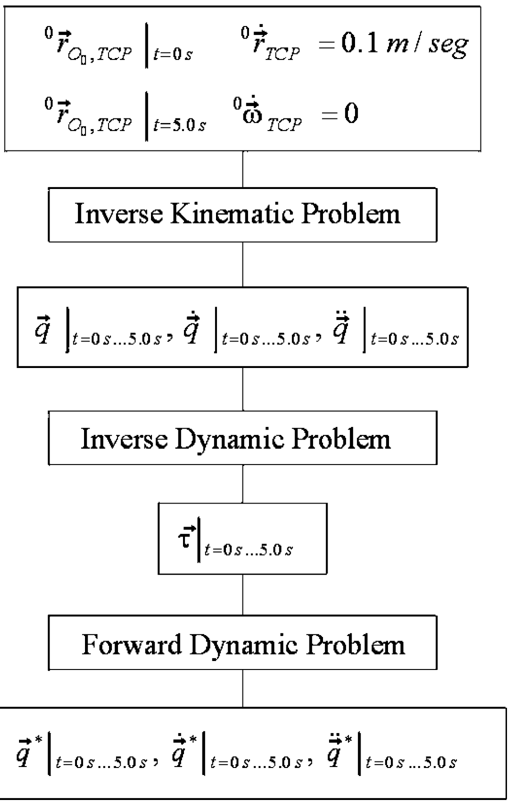
${}^{0}\vec{r}_{O_0,TCP}\big|_{t=0s}$ ${}^{0}\dot{\vec{r}}_{TCP} = 0.1\, m/seg$
${}^{0}\vec{r}_{O_0,TCP}\big|_{t=5.0s}$ ${}^{0}\dot{\vec{\omega}}_{TCP} = 0$
Inverse Kinematic Problem
$\vec{q}\big|_{t=0s...5.0s}, \dot{\vec{q}}\big|_{t=0s...5.0s}, \ddot{\vec{q}}\big|_{t=0s...5.0s}$
Inverse Dynamic Problem
$\vec{\tau}\big|_{t=0s...5.0s}$
Forward Dynamic Problem
$\vec{q}^{*}\big|_{t=0s...5.0s}, \dot{\vec{q}}^{*}\big|_{t=0s...5.0s}, \ddot{\vec{q}}^{*}\big|_{t=0s...5.0s}$

Fig. 4. Block diagram of the procedure.

Table 6
Average differences in the results of the simulation

Joint	Aver. $\lvert q_i - q_i^* \rvert$ (rad)	Aver. $\lvert \dot{q}_i - \dot{q}_i^* \rvert$ (rad/s)	Aver. $\lvert \ddot{q}_i - \ddot{q}_i^* \rvert$ (rad/s^2)
1	$6.91878269 \times 10^{-4}$	$1.28209533 \times 10^{-3}$	$2.84439095 \times 10^{-3}$
2	$1.73972101 \times 10^{-2}$	$1.66087380 \times 10^{-3}$	$9.22318761 \times 10^{-2}$
3	$8.54693990 \times 10^{-4}$	$1.66087459 \times 10^{-3}$	$3.13261587 \times 10^{-3}$
4	$1.73972067 \times 10^{-2}$	$4.05637323 \times 10^{-2}$	$9.22318581 \times 10^{-2}$
5	$1.78259578 \times 10^{-2}$	$4.17630699 \times 10^{-2}$	$9.54221869 \times 10^{-3}$
6	$6.91878269 \times 10^{-4}$	$1.21908065 \times 10^{-3}$	$2.43724278 \times 10^{-3}$

Table 7
Processing time and number of evaluations of the function for each method

	CRBA	ABM	GA
Processing time (s)	0.65	0.71	0.59
Number of function evaluations	1122	1089	1092

6. Conclusions

The G–A equations have been used for the formulation of new recursive algorithms for solving the inverse dynamics problem and computing the generalized inertia matrix. The computationally complexity of the proposed inverse dynamics algorithm is comparable with that of the most efficient algorithms that are known, as shown in Table 1. The proposed algorithm for the generalized inertia matrix is O(n^2) order and because of the number of arithmetic operations involved, it is computationally efficient, as shown in Table 2. In the algorithm, the terms of the generalized inertia matrix are obtained by means of the Hessian matrix of the Gibbs function. This approach takes full advantage of the Gibbs equations to develop a suitable novel formulation of the generalized inertia matrix for multi-body systems, particularly for robot manipulators. This approach can be extended without the need of great computational efforts to closed loop chain manipulators.

Two numerical examples have been carried out using the proposed algorithm (GA) and the best known algorithms in the literature (CRBA and ABA). The results of these simulations are shown in Section 5, where it can be observed that the GA method produces smaller errors than those of the CRBA method, but a little larger than those of the ABA method, which is, according to several authors (see [22] and [2]), the method that has the best numerical response. In addition, from Table 7, the processing time of the GA method is the shortest one, since the proposed algorithm has smaller computational complexity with respect to algorithms derived from the CRBA and ABA methods.

References

[1] W.M. Silver, On the equivalence of Lagrangian and Newton–Euler dynamics of manipulators, Int. J. Rob. Res. 1 (1982) 60–70.

[2] R. Featherstone, D.E. Orin, Robot dynamics: equations and algorithms, Proceedings of the 2000 IEEE International Conference on Robotics and Automation, San Francisco, 2000, pp. 826–834.

[3] L. Pars, A Treatise on Analytical Dynamics, Ox Bow Press, Connecticut, 1972.
[4] E. Desloge, Relationship between Kane's Equations and the Gibbs–Appell Equations, J. Guidance Control Dyn. 10 (1) (1986).
[5] D. Levinson, Comment on relationship between Kane's equations and the Gibbs–Appell equations, J. Guidance Control Dyn. 10 (6) (1987) 593.
[6] R.L. Huston, Comment on relationship between Kane's equations and the Gibbs–Appell equations, J. Guidance Control Dyn. 11 (2) (1988) 191.
[7] I. Sharf, G.M.T. D'eleuterio, P.C. Hughes, On the dynamics of Gibbs Appell and Kane, Eur. J. Mech. A/Solids 11 (2) (1992) 145–155.
[8] F.E. Udwadia, R.E. Kalaba, The explicit Gibbs–Appell equation and generalized inverse forms, Quarterly of Applied Mathematics LVI (2), June, 1998, pp. 277–288.
[9] M. Vukobratovic, V. Potkonjak, Applied Dynamics and CAD of Manipulation Robots, Springer-Verlag, Berlin, 1985.
[10] K. Desoyer, P. Lugner, Recursive formulation for the analytical or numerical application of the Gibbs–Appell method to the dynamics of robots, Robotica 7 (1989) 343–347.
[11] A.F. Vereshchagin, Computer simulation of the dynamics of complicated mechanisms of Robotic manipulators, Eng. Cyber. 6 (1974) 65–70.
[12] I. Rudas, A. Toth, Efficient recursive algorithm for inverse dynamics, Mechatronics 3 (2) (1993) 205–214.
[13] V. Mata, S. Provenzano, J.I. Cuadrado, F. Valero, An O(n) algorithm for solving the inverse dynamic problem in robots by using the Gibbs–Appell formulation, Proceedings of Tenth World Congress on Theory of Machines and Mechanisms, 3 Oulu, Finland, 1999, pp. 1208–1215.
[14] J.Y.S. Luh, M.W. Walker, R.P. Paul, On-line computational scheme for mechanical manipulators, J. Dyn. Syst. Meas. Control 102 (1980) 69–79.
[15] J.M. Hollerbach, A recursive Lagrangian formulation of manipulator dynamics and a comparative study of dynamics formulation complexity, IEEE Trans. Syst. Man Cyber. (1980) 730–736.
[16] T. Kane, D. Levinson, The use of Kane's dynamical equations in Robotic, Int. J. Rob. Res. 2 (3) (1983) 3–21.
[17] J. Angeles, O. Ma, A. Rojas, An algorithm for the inverse dynamics of n-axis general manipulators using Kane's equations, Comp. Math. Appl. 17 (12) (1989) 1545–1561.
[18] M.W. Walker, D.E. Orin, Efficient dynamic computer simulation of Robotic mechanisms, J. Dyn. Syst. Meas. Control 104 (1982) 205–211.
[19] J. Angeles, O. Ma, Dynamic simulation of n-axis serial Robotic manipulators using a natural orthogonal complement, Int. J. Rob. Res. 7 (5) (1988) 32–47.
[20] R. Featherstone, The calculation of robot dynamics using articulated-body inertias, Int. J. Rob. Res. 2 (1) (1983) 13–30.
[21] S.K. Saha, A decomposition of the manipulator inertia matrix, IEEE Trans. Rob. Autom. 13 (2) (1997) 301–304.
[22] U.M. Ascher, D.K. Pai, B.P. Cloutier, Forward dynamics elimination methods, and formulation stiffness in robot simulation, Int. J. Rob. Res. 16 (6) (1997) 749–758.
[23] C.A. Balafoutis, R.V. Patel, Dynamic Analysis of Robot Manipulators: A Cartesian Tensor Approach, Kluwer Academic Press, Boston, 1991.
[24] J. Angeles, Fundamentals of Robotic Mechanical Systems, Springer Verlag, New York, 1997.
[25] T.J. Akai, Métodos numéricos aplicados a la ingeniería, Limusa, México, 1999.
[26] R. Featherstone, Robot dynamics algorithms, Kluwer Academic, Boston, 1987.