ENGINEERING TRANSACTIONS • *Enging. Trans.* • 39, 3-4, 325-349, 1991
Polish Academy of Sciences • Institute of Fundamental Technological Research

NEW COMPUTATIONAL ALGORITHMS FOR A DISCRETE KALMAN FILTER IN ROBOT DYNAMICS

K. KOZŁOWSKI (POZNAŃ)

The equivalence between the standard Newton-Euler formulation of the equation of motion for an n-link manipulator and the inverse dynamics equations has been proved in this paper. Solution to the two-point boundary-value problem leads to the forward dynamics equations which are similar to the equations of Kalman filtering and Bryson-Frazier fixed time-interval smoothing. The extensive numerical studies conducted by the author on the new inverse and forward dynamics algorithms derived from the two-point boundary value problem establish the same level of confidence as exists for current methods. In order to obtain the algorithms with the smallest coefficients of the polynomial of order $0(n)$, the categorization procedure has been implemented in this work. Software packages for both robot dynamics algorithms have been developed in Pascal and run on an IBM compatible computer. The results obtained by Rodriguez have been extended by the present author to an arbitrary manipulator with both rotational and translational joints.

1. Introduction

In standard classical kinematical and dynamical considerations the equations of motion for an n-link manipulator can be obtained as recursive Newton - Euler equations. Another approach to finding the inverse dynamics equations is to formulate the system dynamics and kinematics as a two-point boundary-value problem.

This paper extends the results presented by RODRIGUEZ in papers [17, 18, 19, 20] by proving the equivalence between the standard Newton - Euler formulation of the equations of motion for an n-link manipulator with a fixed base, and the inverse dynamics equations introduced by Rodriguez. We notice that the bias forces (which represent centripetal, Coriolis and gravity forces) given by Rodriguez are calculated according to two different expressions in papers [17] and [18]. We believe that these expressions are not calculated correctly due to wrong differentiation. In the sequel we remove the mistakes and give the corrected proof of the bias forces.

In ref. [19] RODRIGUEZ has presented the corrected version of the bias spatial forces by making use of results given in references [17] and [18]. To make the derivation clear we have decided to prove in this paper the state equation (in which the bias forces appear). A hint which has helped us to establish the correct expression for the bias forces has come from the application of the work presented by Rodriguez and co-author [20] to the standard classical and dynamical equations normally given in the form of recursive Newton - Euler equations [1, 6]. We have to notice that the spatial bias forces and spatial bias accelerations (which results from the differentiation of a vector in two different coordinate systems) are different in references [17] and [20] (in [20] they are calculated correctly). Briefly, these vectors differ because in work [17] RODRIGUEZ has considered the spatial forces on both sides of the i-th joint and recognized them as different vectors. The same applies to the spatial velocities and accelerations. Below we present the equivalence between these two sets of vectors which, later on, results as the equivalence theorem between the recursive Newton - Euler equations of motion and the inverse dynamics given in [17]. From the physical point of view these two sets of equations have to be the same. One aim of this paper is to prove this statement in a rigorous manner.

In his original work [18] RODRIGUEZ has considered a simple example in which only planar motion is allowed. Apart from that, Rodriguez has calculated the number of arithmetic operations per link which are required by the forward dynamics algorithm for a planar chain. Next, this number of operations has been compared with the number of arithmetic operations required for assembly of n-by-n inertia matrix. The initial comparison presented by Rodriguez has inspired the present author to explore the potential of the approach. The author examines the case where the spatial bias forces, spatial bias accelerations, as well as the link-to-link transformations have been included in the number of operations (Rodriguez in his example has assumed that there are no bias forces and accelerations). Next, the discussion of the planar robot is extended to an arbitrary robot with rotational and translational links. In calculating the number of operations we have assumed the modified Denavit-Hartenberg notation [6] between two successive links. Although the results are applicable to an arbitrary manipulator, additionally for inverse dynamics we have assumed that the link angle α_i (twist angle) may take values of 0°, 90°, and −90°. It must be noted that our assumptions are satisfied by most of the available manipulators.

In order to minimize the number of arithmetic calculations (FLOPS), the concept of customizing the dynamic equations developed by KHOSLA [13] has been implemented. The customization procedure guarantees that every two mathematically equivalent expressions are denoted by the same variable name. In our analysis we have calculated the FLOPS for the inverse and forward dynamics algorithms. The results have been compared with those existing in robotics literature. The author has conducted extensive numerical studies on both algorithms given by Rodriguez and tried to establish some level of confidence with the existing methods for dynamic calculations. We have found these investigations quite interesting. At the same time we have tried to implement some factorization techniques [3] to obtain the fastest recursive algorithms. The investigations which we have done lead us to a better understanding of the methods proposed by Rodriguez.

The paper is organized as follows. In Sect. 2, the notation used in formulating the equations found in this paper is discussed. In the third section the main results are obtained. In the fourth section, the computational requirements for the inverse and forward dynamics algorithms are compared with other solutions presented in the robotics literature. In Sect. 5, the simulation results developed by the author are presented. The results of this work are summarized and some conclusions are given in the final section.

2. NOTATION FOR SPATIAL DYNAMICS

In this paper we have assumed that a mechanical system of n links is numbered in an increasing order which goes from the tip of the system toward the base. Joint k in the sequence connects links k and $k+1$. Fixed base is considered to be link $n+1$. At any point at the tip of the manipulator a fictitious joint 0 is attached. The ordering system is shown in Fig.1.

The robotic system depicted in Fig.1 can contain only simple revolute and/or prismatic joints. Coordinate systems can be assigned according to a modified Denavit - Hartenberg convention but, in general, the equations which will be derived do not depend on the specific coordinate transformation used.

The formulation of all equations in this paper is carried out using spatial

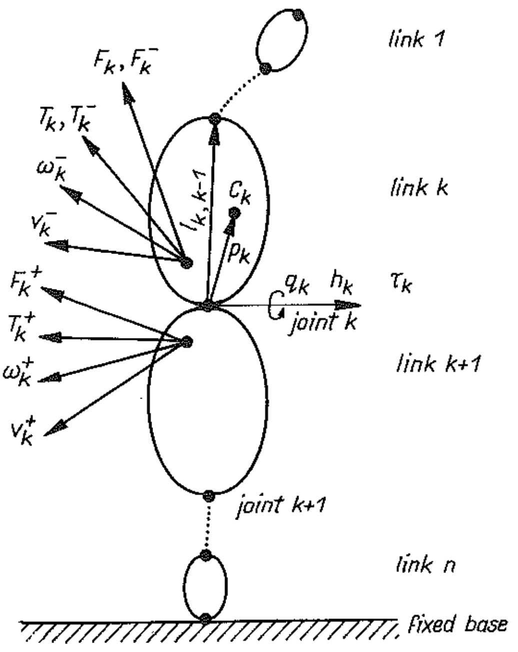

FIG. 1. N-link serial manipulator and relationship of defined quantities to link k.

notation. We have used the notaton introduced by RODRIGUEZ in [17] and [20] (Fig.1). The spatial velocity, acceleration and force are all 6×1 column vectors. A more detailed definition of these concepts is provided below.

T_k^+ and F_k^+ are 3×1 vectors representing, respectively, the constraint torque and force acting at joint k on link $k+1$. The corresponding spatial force is defined to be a 6×1 vector and is denoted by $x_k^+ = [-T_k^+ \; -F_k^+]^T$, where T is a transposition operation. Superscript $+$ indicates that the variable is calculated at a point on the link $k+1$ adjacent to joint k. This operation can be interpreted as a limit operation on the left-hand side of the point located exactly on joint k. RODRIGUEZ in [17] denotes this operation as on the "positive" side, toward the base, of joint k.

T_k^- and F_k^- are 3×1 vectors representing, respectively, the constraint torque and force acting on link k at joint k on the "negative" side, in an outward direction, of joint k. The $-$ superscript indicates that the corresponding variable is evaluated at a point on link k adjacent to joint k. This operation corresponds to a limit operation on the right-hand side of a point located exactly on joint k. The spatial vector will be denoted $x_k^- = [T_k^- \; F_k^-]^T$. Let us notice that Newton's third law implies $x_k^+ = x_k^-$. Next, we will denote all vectors without superscript T_k, F_k, and x_k, respectively, with the same meaning as above. We believe that the double notation is not confusing and is very useful in the proof presented in the next section.

T_{ck} and F_{ck} are, respectively, the external moment and the force acting on link k at its mass center.

ω_k^+ and v_k^+ are 3×1 vectors representing the angular and linear velocities of link $k+1$ calculated on positive side of joint k. The spatial velocity is defined as $V_k^+ = [\omega_k^+ \; v_k^+]^T$. We have assumed that the coordinate system associated with link $k+1$ is located on the negative side of joint $k+1$. The corresponding velocities are expressed in this coordinate system.

ω_k^- and v_k^- are 3×1 vectors representing, respectively, the angular and linear velocity of link k on the negative side of joint k. The corresponding spatial velocity is denoted as $V_k^- = [\omega_k^- \; v_k^-]^T$. To be consistent with the notation introduced above, we can omit the superscript to obtain the corresponding vectors ω_k, v_k, and V_k with the same meaning. Let us notice that all vectors defined in this point are specified in a coordinate frame attached to the link k.

Finally, we have come to the definitions of spatial accelerations. $\lambda_k^+ = [\dot{\omega}_k^+ \; \dot{v}_k^+]^T$ is a vector of angular and linear accelerations of link $k+1$ at the positive side of joint k. Similarly, $\lambda_k^- = [\dot{\omega}_k^- \; \dot{v}_k^-]^T$ is a vector of accelerations at the negative side of joint k. These accelerations are expressed in link $k+1$ and link k coordinates, respectively. According to the convention discussed above, vector λ_k denotes the spatial acceleration of link k.

In spatial notation, inertia matrices are expressed as 6×6 matrices. An inertia matrix for each link is defined in its own coordinate system. For link k, this matrix M_k, is defined as

$$M_k = \begin{bmatrix} I_k & m_k \hat{p}_k \\ -m_k \hat{p}_k & m_k U \end{bmatrix},$$

where I_k is the inertia matrix of link k about joint k; p_k is the position vector of the center of gravity of link k from the k-th coordinate's origin, $\hat{p}_k$ is the 3×3 matrix equivalent to the cross-product operation $p_k \times (\cdot)$; and U is the 3×3 identity. Because I_k and p_k are defined in coordinate system k, matrix M_k is constant.

For subsequent derivations, it is convenient to define the following 6×6 matrix

$$\phi_{k,m} = \begin{bmatrix} U & \hat{l}_{k,m} \\ 0 & U \end{bmatrix},$$

where $l_{k,m}$ is the vector from joint k to the joint m and $\hat{l}_{k,m}$ is the 3×3

matrix equivalent to $l_{k,m} \times (\cdot)$. This matrix is called a transition matrix which originally was defined in discrete linear systems [5, 11]. Let us notice that $l_{k,k-1}$ is a vector from the origin of a frame fixed in link k to the origin of a frame fixed in link $k-1$, and matrix $\phi_{k,k-1}$ is the position matrix between two adjacent links. In general, the orientation of a spatial kinematic vector from one coordinate system to an adjacent one may be accomplished by making use of the following spatial orientation matrix (6×6)

$$A_k = \begin{bmatrix} {}^{k+1}_{k}R & 0 \\ 0 & {}^{k+1}_{k}R \end{bmatrix},$$

where ${}^{k+1}_{k}R$ is the 3×3 rotation transformation between two coordinate systems. Some authors [1] combine these two matrices into one spatial transformation matrix. Rodriguez has defined the transition matrix in order to formulate the equations of motion using the discrete linear state space approach.

To continue with the symbol definitions presented in Fig.1, h_k is a unit vector along joint axis k (not necessarily the z_k axis), C_k denotes the mass center of link k, and q_k is the joint variable which is positive in the right-hand sense about h_k joint axis. Next, H_k^T is a 6×1 vector of the following form: $H_k^T = \begin{bmatrix} h_k \\ 0 \end{bmatrix}$ when joint k is rotational and $H_k^T = \begin{bmatrix} 0 \\ h_k \end{bmatrix}$ when joint k is translational, 0 denotes a 3×1 zero vector. Finally, τ_k is the actuated torque of joint k.

On the basis of the definition of the spatial quantities we now review the inverse and forward dynamics algorithms.

3. Development of the main algorithms

In this section we will develop a new algorithm for computing the bias spatial forces. Next we will prove the equivalence between the two inverse dynamics algorithms presented in [17] and [20], respectively. In some equations, where it is necessary, we will use the orientation matrix A_k or its inverse A_k^{-1}.

The sequence of spatial velocities satisfies the following equations

$$V_{k-1}^{+} = \phi_{k,k-1}^T V_k^{-}, \qquad k = n, \dots, 1, \tag{3.1}$$

$$V_k^{-} = A_k^{-1} V_k^{+} + H_k^T \dot{q}_k, \qquad k = n, \dots, 1. \tag{3.2}$$

Eq. (3.1) describes the spatial velocity propagation within a link k from its inner joint to its outer joint. Eq. (3.2) updates the spatial velocity in direction from the positive side to the negative side of joint k. In crossing the joint k it is necessary to use the orthogonal transformation. The initial condition for the Eq. (3.2) can be written as follows:

$$V_n^+ = 0 . \tag{3.3}$$

The accelerations satisfy the closely related recursion:

$$\begin{aligned}
\lambda_{k-1}^+ &= \phi_{k,k-1}^T \, \lambda_k^- , && (3.4)\\
\lambda_k^- &= A_k^{-1} \, (\lambda_k^+ + n_k) + H_k^T \ddot{q}_k, && (3.5)\\
\lambda_n^+ &= 0 , && (3.6)
\end{aligned}$$

where n_k is the bias acceleration

$$n_k = \begin{bmatrix} \omega_k^+ \times {}^{k+1}_{k}R h_k \, \dot{q}_k \\ v_k^+ \times {}^{k+1}_{k}R h_k \, \dot{q}_k \end{bmatrix} . \tag{3.7}$$

Let us notice that when joint k is translational,

$$n_k = \begin{bmatrix} 0 \\ \omega_k^+ \times {}^{k+1}_{k}R h_k \, \dot{q}_k \end{bmatrix} . \tag{3.7'}$$

The recursive relationships (3.4) and (3.5) can be stablished by appropriate time differentiation of (3.1) and (3.2) in an inertial frame. As before, Eq.(3.6) describes the initial condition for Eq.(3.5) and index k varies from n to 1. To perform a differentiation operation one has to remember that, by definition, appropriate vectors are expressed in local coordinate systems.

Now, referring to the original work done by RODRIGUEZ [17], we review the two-point boundary-value problem. The sequences of spatial forces and spatial accelerations satisfy the following set of equations

$$\begin{aligned}
x_k^- &= \phi_{k,k-1} \, x_{k-1}^+ + M_k \, \lambda_k^- + b_k , && (3.8)\\
x_k^- &= x_k^+, && (3.9)\\
\lambda_{k-1}^+ &= \phi_{k,k-1}^T \, \lambda_k^-, \qquad \lambda_n^+ = 0 , && (3.10)\\
\lambda_k^- &= \lambda_k^+ + H_k^T \, \ddot{q}_k + n_k , && (3.11)\\
\tau_k &= H_k x_k^- , && (3.12)\\
x_0^+ &= 0 , && (3.13)
\end{aligned}$$

where b_k is the bias spatial force

$$b_k = \begin{bmatrix} \omega_k^- \times I_k\omega_k^- + m_k p_k \times (\omega_k^- \times v_k^-) - T_{ck} - p_k \times F_{ck} \\ m_k\left[\omega_k^- \times v_k^- + \omega_k^- \times (\omega_k^- \times p_k)\right] - F_{ck} \end{bmatrix}. \tag{3.14}$$

The expression for the bias spatial force is true regardless of the type of the joint. Let us notice that for a planar robot $\omega_k^- \times I_k\omega_k^- = 0$. To simplify Eqs.(3.9) and (3.11), we have skipped the orientation transformation. Eq.(3.8) is based on the rigid-body equations of rotational and translational motion for link k. Eq.(3.9) reflects the equivalence of action/reaction torques and forces at joint k. Equation (3.13) states that the initial joint 0 is not under the influence of any external torques and forces. In [17] the state equation (3.8) is not correct because of the wrong expression for b_k (other equations (3.9) – (3.13) are correct). Below we give a corrected proof of Eqs.(3.8) and (3.14).

First we consider the equation of rotational motion for link k about its mass center

$$\begin{aligned} \omega_{ck} \times I_{ck}\omega_{ck} + I_{ck}\dot{\omega}_{ck} = N_k^- + N_{k-1}^+ + T_{ck} \\ +(l_{k,k-1} - p_k) \times F_{k-1}^+ - p_k \times F_k^-, \end{aligned} \tag{3.15}$$

where ω_{ck} is the angular velocity of link k (notice that $\omega_{ck} = \omega_k^-$), I_{ck} is the link inertia tensor about its mass center, N_k^- and F_k^- are, respectively, the torque and the force acting at joint k, T_{k-1}^+ and F_{k-1}^+ are the torque and the force acting at joint $k-1$, and T_{ck} and F_{ck} are the torque and the force acting at the link mass center.

The translation of the link k mass center is described by

$$F_k^- = m_k\dot{v}_{ck} - F_{k-1}^+ - F_{ck}\,, \tag{3.16}$$

where $\dot{v}_{ck}$ denotes the acceleration of the mass center. Let us notice that the velocity of the mass center can be written in the following form: $v_{ck} = v_k^- + \omega_k^- \times p_k$. Differentiation of the last equation with respect to the inertial frame gives

$$\begin{aligned} \dot{v}_{ck} &= \dot{v}_k^- + \omega_k^- \times v_k^- + \dot{\omega}_k^- \times p_k + \omega_k^- \times (\omega_k^- \times p_k) \\ &= \dot{v}_k^- + \omega_k^- \times v_{ck} + \dot{\omega}_k^- \times p_k. \end{aligned} \tag{3.17}$$

Substitution of Eq.(3.17) in Eq.(3.16) results in

$$F_k^- = m_k[\dot{v}_k^- + \omega_k^- \times v_k^- + \dot{\omega}_k^- \times p_k + \omega_k^- \times (\omega_k^- \times p_k)] - F_{k-1}^+ - F_{ck}. \tag{3.18}$$

Substitution of the right-hand side of Eq.(3.18) in Eq.(3.16) leads to

$$(3.19)\quad \begin{aligned} &I_{ck}\dot{\omega}_k^- + \omega_k^- \times I_{ck}\omega_k^- + m_k p_k \times [\dot{v}_k^- + \omega_k^- \times v_k^- + \dot{\omega}_k^- \times p_k \\ &+\omega_k^- \times (\omega_k^- \times p_k)] = N_k^- + N_{k-1}^+ + T_{ck} + l_{k,k-1} \times F_{k-1}^+ + p_k \times F_{ck}. \end{aligned}$$

The quadratic terms $p_k \times (\omega_k^- \times p_k)$ and $p_k \times (\omega_k^- \times (\omega_k^- \times p_k))$ are eliminated because the moment of inertia tensor I_k is expressed about the coordinate system origin located at joint k, instead of about center of mass I_{ck}. We use the following relations [24]:

$$\omega_k^- \times I_{ck}\omega_k^- + m_k p_k \times \left[\omega_k^- \times (\omega_k^- \times p_k)\right] = \omega_k^- \times I_k\omega_k^-$$

and

$$\begin{aligned} &I_{ck}\dot{\omega}_k^- + m_k p_k \times (\dot{\omega}_k^- \times p_k) = I_{ck}\dot{\omega}_k^- + [p_k^2\dot{\omega}_k^- - p_k(p_k\dot{\omega}_k^-)]m_k \\ &= I_{ck}\dot{\omega}_k^- + [p_k^2 E\dot{\omega}_k^- - p_k p_k\dot{\omega}_k^-]m_k = \{I_{ck} + (p_k^2 E - p_k p_k)m_k\}\dot{\omega}_k^- = I_k\dot{\omega}_k^-. \end{aligned}$$

Equation (3.19) can be rewritten as

$$(3.20)\quad \begin{aligned} I_k\dot{\omega}_k^- + \omega_k^- \times I_k\omega_k^- + m_k p_k \times \dot{v}_k^- + m_k p_k \times (\omega_k^- \times v_k^-) \\ = T_k^- + T_{k-1}^+ + l_{k,k-1} \times F_{k-1}^+ + T_{ck} + p_k \times F_{ck}. \end{aligned}$$

From the last equation we calculate T_k^- to obtain

$$(3.21)\quad \begin{aligned} T_k^- = I_k\dot{\omega}_k^- + \omega_k^- \times I_k\omega_k^- + m_k\left[p_k \times \dot{v}_k^- + p_k \times (\omega_k^- \times v_k^-)\right] \\ -T_{k-1}^+ - l_{k,k-1} \times F_{k-1}^- - T_{ck} - p_k \times F_{ck}. \end{aligned}$$

Rearranging of the formula (3.18) leads to

$$(3.22)\quad F_k^- = m_k\dot{v}_k^- + m_k\dot{\omega}_k^- + m_k[\omega_k^- \times v_k^- \times (\omega_k^- \times p_k)] - F_{k-1}^+ - F_{ck}.$$

Equations (3.21) and (3.22) combine into

$$(3.23)\quad \begin{aligned} \begin{bmatrix} T_k^- \\ F_k^- \end{bmatrix} = \begin{bmatrix} U & \hat{l}_{k,k-1} \\ 0 & U \end{bmatrix} \begin{bmatrix} -T_{k-1}^+ \\ -F_{k-1}^+ \end{bmatrix} + \begin{bmatrix} I_k & m_k\hat{p}_k \\ -m_k\hat{p}_k & m_k U \end{bmatrix} \begin{bmatrix} \dot{\omega}^- \\ \dot{v}_k^- \end{bmatrix} \\ + \begin{bmatrix} \omega_k^- \times I_k\omega_k^- + m_k p_k \times (\omega_k^- \times v_k^-) - T_{ck} - p_k \times F_{ck} \\ m_k\omega_k^- \times v_k^- + m_k\omega_k^- \times (\omega_k^- \times p_k) - F_{ck} \end{bmatrix}. \end{aligned}$$

Recalling the symbol definitions from Sect. 2, we have shown that Eq.(3.23) is a more detailed version of Eq.(3.8) with the bias forces given by [14].

We have already stated that there are several errors in the original reports written by RODRIGUEZ [17, 18]. In the second row of Eq.(4.11), in [17], the term $m_k\omega_k^- \times (\omega_k^- \times p_k)$ is missing. In a later paper [18] the first term in the second row of Eq. (3.7) is wrong. Also the translation equation of link k (Eq.(9) in [18]) is not correct, since the time differentiation is performed with respect to the inertial reference frame. As a result, the bias spatial forces are not calculated correctly. In a subsequent work [19] the bias spatial forces are given in a correct form, similar to those calculated earlier in this work; however, in the outline of the proof for the propagation of the spatial forces the references [17] and [18] are cited without apparent correction of the errors.

In [17] Rodriguez has proposed a very interesting interpretation of the set of Eqs.(3.8) to (3.13). According to his interpretation, the set of Eqs.(3.8) to (3.13) forms a two-point boundary-value problem in the sense that the boundary conditions are satisfied at two distinct points in space: the initial joint at the tip of the system and the terminal joint at the base. The spatial forces vanish at the tip and the base is, by definition, immobile. This two-point boundary-value problem is analogous to these encountered in optimal control and estimation theory of linear systems [5]. Thus Eqs.(3.1) – (3.14) completely describe the kinematics and dynamics of a robotic manipulator with both translational and rotational joints. The results presented by Rodriguez have been extended to both sliding and rotational joints.

Below we present a set of recursive Newton - Euler equations in terms of the spatial definitions described in Sect.2 (for this reason we use the double notation, the derivation of the set can be found in [20]).

$$(3.24)\qquad V_{n+1} = 0 ,$$
$$(3.25)\qquad V_k = \phi^T_{k+1,k} V_{k+1} + H^T_k \dot{q}_k , \qquad k = n, ..., 1 ,$$
$$(3.26)\qquad \lambda_{n+1} = 0 ,$$
$$(3.27)\qquad \lambda_k = \phi^T_{k+1,k} \lambda_{k+1} + H^T_k \ddot{q}_k + \bar{a}_k , \qquad k = n, ..., 1 ,$$
$$(3.28)\qquad x_0 = 0 ,$$
$$(3.29)\qquad x_k = \phi_{k,k-1} x_{k-1} + M_k \lambda_k + \bar{b}_k , \qquad k = 1, ..., n ,$$
$$(3.30)\qquad \tau_k = H_k x_k ,$$

where $\bar{a}_k$ is the bias acceleration given by the following expressions

$$(3.31)\qquad \bar{a}_k = \begin{bmatrix} \omega_{k+1} \times h_k \dot{q}_k \\ \omega_{k+1} \times (\omega_{k+1} \times l_{k+1,k}) \end{bmatrix} \quad \text{for rotational joint} ,$$

$$(3.32)\qquad \bar{a}_k = \begin{bmatrix} 0 \\ \omega_{k+1} \times (\omega_{k+1} \times l_{k+1,k} + 2 h_k \dot{q}_k) \end{bmatrix} \quad \text{for sliding joint} ,$$

and $\bar{b}_k$ is the bias spatial force defined as

$$\bar{b}_k = \begin{bmatrix} \omega_k \times I_k\omega_k \\ m_k\omega_k \times (\omega_k \times p_k) \end{bmatrix} \tag{3.33}$$

for both sliding and rotational joints. Equations (3.24) – (3.33) completely describe the kinematics and dynamics for an arbitrary manipulator with both types of joints.

In the next step we will prove that the set of Eqs.(3.1) – (3.14) is equivalent to the set of Eqs.(3.24) – (3.33). first we consider a robot with revolute joints only. Substituting (3.1) in (3.2) for index $k+1$, we get

$$V_k^- = \phi_{k+1}^T V_{k+1}^- + H_k^T \dot{q}_k \tag{3.34}$$

with the initial condition $V_{n+1}^- = 0$. For convenience we skip the rotational transformation in Eq.(3.2). From Eqs.(3.34) and (3.25) it is clear that $V_k^- = V_k$. Substituting Eq.(3.4) in Eq.(3.5) for index $k+1$ and omitting the rotational transformation, we get

$$\lambda_k^- = \phi_{k+1}^T \lambda_{k+1}^- + H_k^T \ddot{q}_k + n_k \tag{3.35}$$

with the initial condition $\lambda_{n+1}^- = 0$. Substituting Eq.(3.9) in Eq.(3.8) for index $k-1$ results in

$$x_k^- = \phi_{k,k-1} x_{k-1}^- + M_k \lambda_k^- + b_k \tag{3.36}$$

with the initial condition $x_0^- = 0$.

For convenience we skip the terms including the gravity force and the gravity torque acting at the mass center in the bias spatial force b_k. We will show later that it is not a restriction. We rewrite the bias accelerations and forces for the two sets of Eqs.(3.35), (3.36), and (3.27), (3.29).

$$n_k = \begin{bmatrix} \omega_{k+1}^- \times h_k\dot{q}_k \\ v_k^+ \times h_k\dot{q}_k \end{bmatrix}, \quad b_k = \begin{bmatrix} \omega_k^- \times I_k\omega_k^- + m_k p_k \times (\omega_k^- \times v_k^-) \\ m_k\omega_k^- \times v_k^- + m_k\omega_k^- \times (\omega_k^- \times p_k) \end{bmatrix}, \tag{3.37}$$

and

$$\bar{n}_k = \begin{bmatrix} \omega_{k+1} \times h_k\dot{q}_k \\ \omega_{k+1} \times (\omega_{k+1} \times l_{k+1,k}) \end{bmatrix}, \quad \bar{b}_k = \begin{bmatrix} \omega_k \times I_k\omega_k \\ m_k\omega_k \times (\omega_k \times p_k) \end{bmatrix}. \tag{3.38}$$

From the two pairs of Eqs.(3.35), (3.27), and (3.36), (3.29) it is clear that $\lambda_k^- = \lambda_k$ and $x_k^- = x_k$ when the corresponding bias spatial forces and accelerations defined by Eqs.(3.37) and (3.38) are expressed by exactly the same quantities. Note that this can be achieved when we remove the term

$\omega_k^- \times v_k^-$ from the bias spatial force b_k to the bias spatial acceleration, namely, we rewrite n_k as follows:

$$\bar{\bar{n}}_k = \begin{bmatrix} \omega_{k+1}^- \times h_k \dot{q}_k \\ v_k^+ \times h_k \dot{q}_k + \omega_k^- \times v_k^- \end{bmatrix}. \tag{3.39}$$

To complete the proof of the equivalence it is also necessary to rearrange the second row of a new bias spatial acceleration $\bar{\bar{n}}_k$ as follows:

$v_k^+ \times h_k\dot{q}_k + \omega_k^- \times v_k^- = v_k^- \times h_k\dot{q}_k - v_k^- \times \omega_k^- = v_k^- \times [h_k\dot{q}_k - \omega_k^-] = \omega_k^+ \times v_k^-$,

where we have used the equation $\omega_k^- = \omega_k^+ + h_k\dot{q}_k$ which is true for rotational joints. From Eq.(3.1) for index k we have $v_k^+ = \omega_{k+1}^- \times l_{k+1,k} + v_{k+1}^-$ and $\omega_k^+ = \omega_{k+1}^-$. Thus we can write

$$v_k^+ + h_k\dot{q}_k + \omega_k^- \times v_k^- = \omega_{k+1}^- \times (\omega_{k+1}^- \times l_{k+1}) + \omega_{k+1}^- \times v_{k+1}^-. \tag{3.40}$$

Now we prove by induction that Eq.(3.27) implies Eq.(3.35). Rearranging Eq.(3.40), we obtain

$$\omega_{k+1}^- \times (\omega_{k+1}^- \times l_{k+1,k}) = v_k^+ \times h_k\dot{q}_k + \omega_k^- \times v_k^- - \omega_{k+1}^- \times v_{k+1}^-. \tag{3.41}$$

From Eq.(3.27), for $k = n$, we get $\lambda_n = H_n^T \dot{q}_n$ because $\lambda_{n+1} = \bar{n}_n = 0$. At the same time $\lambda_k^- = H_k^T \dot{q}_k$. For the k-th induction step

$$\lambda_k = \phi_{k+1,k}^T \left\{ \phi_{k+2,k+1}^T \lambda_{k+2} + H_{k+1}^T \dot{q}_{k+1} + \begin{bmatrix} \omega_{k+1}^+ \times h_{k+1}\dot{q}_{k+1} \\ v_{k+1}^+ \times h_{k+1}\dot{q}_{k+1} + \underline{\omega_{k+1}^- \times v_{k+1}^-} \end{bmatrix} \right\}$$
$$+ H_k^T \dot{q}_k + \begin{bmatrix} \omega_k^+ \times h_k\dot{q}_k \\ v_k^+ \times h_k\dot{q}_k + \omega_k^- \times v_k^- - \underline{\omega_{k+1}^- \times v_{k+1}} \end{bmatrix}.$$

From the last equation it follows that the terms which are underlined cancel out because of the particular form of the matrix $\phi_{k+1,k}^T$. This allows to construct Eq.(3.35) for the k-th induction step, given the bias spatial accelaration defined by Eq.(3.39).

To prove that Eq.(3.27) follows from Eq.(3.35), we rearrange Eq.(3.40) as follows:

$$v_k^+ + h_k\dot{q}_k = \omega_{k+1}^- + (\omega_{k+1}^- \times l_{k+1,k}) + \omega_{k+1}^- \times v_{k+1}^- - \omega_k^- \times v_k^-. \tag{3.42}$$

From Eqs.(3.35) and (3.27), for $k = n$, we have $\lambda_n^- = H_n^T \dot{q}_n$. Hence for the k-th induction step

$$\lambda_k = \phi_{k+1}^T \left\{ \phi_{k+2,k+1}^T \lambda_{k+2}^- + H_{k+1}^T \dot{q}_{k+1} \right.$$
$$\left. + \begin{bmatrix} \omega_{k+2}^- \times h_{k+1}\dot{q}_{k+1} \\ \omega_{k+2}^- \times (\omega_{k+2}^- \times l_{k+2,k+1}) + \omega_{k+2}^- \times v_{k+2}^- - \underline{\omega_{k+1}^- \times v_{k+1}^-} \end{bmatrix} \right\}$$
$$+ H_k^T \dot{q}_k + \begin{bmatrix} \omega_{k+1}^- \times h_k\dot{q}_k \\ \omega_{k+1}^- \times (\omega_{k+1}^- \times l_{k+1,k}) + \underline{\omega_{k+1}^- \times v_{k+1}^-} - \omega_k^- \times v_k^- \end{bmatrix}.$$

From the last equation it follows that the terms which are underlined cancel out due to the structure of the matrix $\phi^T_{k+1,k}$ (we have used Eq.(3.42)). To obtain Eq.(3.27) it is necessary to move the term $-\omega^-_k \times v^-_k$ through the matrix M_k to the bias spatial ~~force~~ b_k. In this way we get $\bar{b}_k$. Thus we have obtained Eq.(3.29) with the bias spatial force defined by (3.38). We have proved the equivalence of Eqs.(3.35), (3.27), and (3.36) (3.29) by appropriate switching of the term $\omega^-_k \times v^-_k$, which can be done due to the structure of the matrices M_k and $\phi^T_{k+1,k}$. Thus we have proved that $x_k = x^-_k$ and $\lambda_k = \lambda^-_k$ for a manipulator with rotational joints. In order to obtain the same result for a manipulator with translational joints, the bias spatial accelerations $\bar{\bar{n}}$ should be written in the following form:

$$\bar{\bar{n}}_k = \begin{bmatrix} 0 \\ \omega^+_k \times h_k \dot{q}_k + \omega^-_k \times v^-_k \end{bmatrix}. \tag{3.43}$$

Note that when joint k is translational, the following relationship is true

$$\omega^-_k \times v^-_k + \omega^+_k \times h_k \dot{q}_k = \omega^-_{k+1} \times (\omega^-_{k+1} \times l_{k+1,k} + 2h_k \dot{q}_k) + \omega^-_{k+1} \times v^-_{k+1}. \tag{3.44}$$

Using Eqs.(3.43) and (3.44) we can prove in a similar manner the equivalence between two pairs of Eqs.(3.35), (3.27), and (3.36), (3.29).

The results obtained in this section can be easily extended to a manipulator with arbitrary combination of rotational and sliding joints. Thus we could prove for an arbitrary manipulator that $x^-_k = x_k$ and $\lambda^-_k = \lambda_k$. We observe that the bias spatial forces b_k combine the centripetal, Coriolis, and gravity forces. We have to realize that, as a consequence of calculating the spatial accelerations from Eqs.(3.4) – (3.6), the term $\omega^-_k \times v^-_k$ has been omitted. As a result of removing this term from b_k, it is necessary to calculate the bias accelerations according to Eqs.(3.39) or (3.43). The appropriate force and torque caused by acceleration $\omega^-_k \times v^-_k$ appear in the equation of the spatial forces (3.8) because of the special structure of the matrix M_k.

Both computational schemes for the inverse dynamics algorithms presented in this section have been checked against each other for several industrial robots. The results were exactly the same.

Finally, notice that the gravity forces have the same representation in both expressions for calculating b_k and $\bar{b}_k$. In order to reduce the number of arithmetic operations for calculating the gravity forces we assume $\lambda^+_n = \lambda_{n+1} = g$, where g is 6×1 vector describing spatial gravity acceleration.

Next, we will review the forward dynamics equations based on reference [18] (the derivation of these equations can also be found in [18]). The problem is solved by applying the sweep method [5]. According to this method, the state x_k and costate λ_k are related by the equation

$$x_k = z_k + P_k \lambda_k \ , \tag{3.45}$$

where z_k and P_k can be determined by means of the recursive equations that emerge upon substitution of Eq.(3.45) in Eqs.(3.8) – (3.13). There, z_k will play the same role as the predicted state estimate plays in the Kalman filter. Similarly, P_k will play the role of the corresponding state estimation error covariance.

Below we sumarize the forward dynamics equations.

Filtering

(3.46)	initial conditions	z_0^+	=	$0, \quad P_0^+ = 0$,
(3.47)	state prediction	z_k^-	=	$\phi_{k,k-1} z_{k-1}^+ + b_k$,
(3.48)	inertia prediction	P_k^-	=	$\phi_{k,k-1} P_{k-1}^+ \phi_{k,k-1}^T + M_k$,
(3.49)	joint axis inertia	D_k	=	$H_k P_k^- H_k^T$,
(3.50)	Kalman gain	G_k	=	$P_k^- H_k^T / D_k$,
(3.51)	innovations	e_k^-	=	$\tau_k - H_k z_k^-$,
(3.52)	state update	z_k^+	=	$A_k[z_k^- + G_k e_k^-] + P_k^+ n_k$,
(3.53)	residuals	e_k^+	=	e_k^- / D_k ,
(3.54)	inertia update	P_k^+	=	$A_k[I - G_k H_k] P_k^- A_k^{-1}$.

Smoothing

(3.55)	terminal costate	λ_n^+	=	0 ,
(3.56)	costate propagation	λ_{k-1}^-	=	$\phi_{k,k-1}^T \lambda_k^-$,
(3.57)	joint acceleration	$\ddot{q}_k$	=	$e_k^+ - (A_k G_k)^T (\lambda_k^+ + n_k)$,
(3.58)	costate update	λ_k^-	=	$A^{-1}[\lambda_k^+ + n_k] + H_k^T \ddot{q}_k$.

Let us notice that the above set of equations is true for both types of joints. We have included the appropriate orientation transformation in all equations associated with the link crossing operations.

In the next section we will consider the computational complexity of the forward dynamics algorithm.

4. Computational requirements

In this section we present the scalar operations (multiplications, additions and divisions) required to compute the inverse and forward dynamics algorithms presented in Sect.3. This work extends the results obtained in [18]. The calculation of the number of arithmetic operations is based on several assumptions. First, we assume that the link-to-link coordinate orientation transformation is the modified Denavit - Hartenberg orientation matrix. The distance between two origins of successive coordinate frames is defined to be the spatial transformation matrix. We further assume that the manipulator has both types of joints and that the twist angle α_i may take any value. For the spatial transition matrix we assume that it has, at most, 6 nonzero elements except the diagonal elements.

Next, we implement the concept of customizing the dynamic equations to reduce the computational requirements. The customizing procedure was borrowed from [13]. According to this convention, the nonzero elements of a vector or a matrix are denoted by subscribed variables, and the zero and unity elements by a 0 and 1, respectively. We propagate the nonzero elements as variables and the zero elements as zeros. This customization procedure guarantees that every two mathematically equivalent expressions are denoted by the same variable name. Using the customization procedure results in longer but faster computer programs. The direction cosine matrix can be split up into two planar rotation matrices. If we realize that every matrix has an invariant part with respect to planar rotations, we shall see that we significantly save on the number of operations.

4.1. The inverse dynamics algorithms

We now consider the number of arithmetic operations for the new inverse dynamics problem developed in Sect.3. This number of operations is given in Table 1.

The computational requirements of the general purpose implementation, given in Table 1 as left-hand columns of multiplications and additions/subtractions, respectively, incorporate the savings obtained by zero elements of the orientation matrices A_k, the sparse H_k vector, the zero initial conditions with regard to the force and torque acting at the mass center of each link, and the gravitational acceleration $\lambda_n^+ = g$ of the manipulator base.

An n degrees-of-freedom manipulator with rotational joints only (which is the worse case) requires $142n - 151$ multiplications and $109n - 134$ additions/subtractions (Table 1). For six degrees-of-freedom manipulators, the computational requirements are 701 multiplications and 520 additions/sub-

Table 1. Number of arithmetic operations per link for the inverse dynamics problem

Recursions	Multiplications		Additions/Subtractions	
b_k, n_k Eqs.(3.7), (3.14)	$74n - 83$,	$58n - 66$	$44n - 58$,	$36n - 42$
Costate propagation Eq.(3.10)	$6n - 10$,	$4n - 6$	$6n - 10$,	$4n - 6$
Costate update Eq.(3.11)	$16n - 11$,	$8n - 6$	$15n - 15$,	$11n - 11$
State propagation Eqs.(3.8), (3.12)	$30n - 31$,	$28n - 27$	$36n - 43$,	$34n - 39$
State update Eq.(3.9)	$16n - 16$,	$8n - 8$	$8n - 8$,	$4n - 4$
Total	$142n - 151$,	$106n - 113$	$109n - 134$,	$89n - 102$

tractions. The computational requirements of the Newton - Euler algorithm are documented in the literature [10, 12, 13]. For example, Khosla [13] cites 678 multiplications and 521 additions/subtractions. Note that computational load is comparable to this general-purpose implementation. For $n < 6$ the new inverse dynamics algorithm summarized in Table 1 is faster than the algorithm presented in [13].

Most of the existing manipulators have adjacent axes which are either parallel or perpendicular. For this orientation of the axes (twist angle α_i is equal to 0°, 90° or -90°), we can reduce computational load. Thus, for an n degrees-of-freedom manipulator the computational load is $106n - 113$ multiplications and $89n - 102$ additions/subtractions and is depicted in right-hand second columns in Table 1. A six degrees-of-freedom manipulator requires 523 multiplications and 432 additions/subtractions. Khosla quotes 500 multiplications and 403 additions/subtractions. Still in this case computational load of the new inverse dynamics algorithm is comparable to that of parallel/perpendicular axes.

The calculation of the two-point boundary-value problem seems to be slightly slower due to the notation introduced in Sect.2, which recognizes spatial quantities on both sides of each joint. For this reason the computational load resulting from Eqs.(3.24) – (3.33) is exactly the same as that presented by Khosla. ARMSTRONG and co-authors [2] point out that the general-purpose implementation requires 1560 calculations (multiplications and additions), for 6 degrees-of-freedom manipulator. One can incorporate nested categorization into a set of Eqs.(3.1) – (3.14) in order to obtain a smaller number of operations similar to the number of operations given by KHOSLA [13].

4.2. The forward dynamics algorithms

General solution for the forward dynamics problem consists of the two parts: filtering and smoothing. Filtering is based on Eqs.(3.46) – (3.53); smoothing calculates the joint accelerations and is based on Eqs.(3.55) – (3.58). The author believes that it is worth calculating the number of operations for the forward dynamics problem for an arbitrary n-link manipulator.

Some preliminary results obtained by the author are presented in [14, 15]. In [14] a general planar chain example has been considered which extends the results presented by RODRIGUEZ in [18]. The bias forces and the bias accelerations are not necessarily zero. The computational load for the forward dynamics algorithm for an arbitrary n-link manipulator with parallel/perpendicular axes has been considered in [15, 16]. Results obtained in [14] and [16] have been extended in the present work to an arbitrary n-link manipulator. The computational requirements of the forward dynamics problem based on the set of equations (3.46) – (3.58) are presented in Table 2.

From Table 2 it follows that for an arbitrary n-link manipulator with revolute joints only the total number of arithmetic operations is $534n - 770$ (this is the worse case). For six degrees-of-freedom manipulators the computational requirements are 1261 multiplications/divisions and 1173 additions/subtractions (total 2434 FLOPS). A very efficient forward dynamics algorithm presented in reference [4] requires $250n - 222$ multiplications and $220n - 198$ additions, which results in 2400 FLOPS for six degrees-of-freedom manipulator. Notice that for $n < 6$ the forward dynamics algorithm presented in Sect.3 is faster than the algorithm of Brandl and co-authors. This implies that the algorithm presented in Sect.3 is always more efficient than the algorithms of WALKER, ORIN [23] and FEATHERSTONE [7]. For $n > 6$ the forward dynamics algorithm is slightly slower than the algorithm of Brandl and co-authors. That is due to the spatial definitions of quantities

Table 2. Number of arithmetic operations per link for an n-link manipulator required by Bryson - Frazier smoother

Recursions	Multiplications	Additions/Subtractions	Divisions
b_k, n_k Eqs.(3.7), (3.14)	$74n - 83$	$44n - 58$	–
State propagation Eq.(3.47)	$6n - 10$	$12n - 21$	–
Inertia propagation Eq.(3.48)	$56n - 104$	$22n - 115$	–
Kálmán gain Eq.(3.50)	–	–	$5n - 10$
Innovations Eq.(3.51)	–	n	–
Residuals Eq.(3.53)	–	–	n
State update Eq.(3.52)	$46n - 56$	$39n - 47$	–
Inertia update Eq.(3.54)	$55n - 56$	$85n - 139$	–
Costate propagation Eq.(3.56)	$6n - 10$	$6n - 10$	–
Joint acceleration Eq.(3.57)	$5n - 2$	$6n - 3$	–
Costate update Eq.(3.58)	$16n - 28$	$9n - 18$	–
Total	$264n - 349$	$264n - 411$	$6n - 10$

on both sides of each joint.

It is also of interest to compare the above numbers of operations for the forward dynamics algorithms with those required to compute the multilink system inertia matrix and to invert this matrix numerically. The number of arithmetic operations required for assembly of n-by-n inertia matrix is $18n^2 + 92n - 110$ (for details see [9]). Computation of the other terms not included in the system inertia matrix requires $251n - 285$ FLOPS. Finally, the solving of the linear equations for the joint accelerations requires $\frac{1}{3}n^3 + 2\frac{1}{2}n^2 - 1\frac{5}{6}n$ arithmetic operations. The total number of multiplications and additions in this method is $\frac{1}{3}n^3 + 20\frac{1}{2}n^2 + 341\frac{1}{6}n - 395$. This method is more efficient than any of the techniques proposed by WALKER and ORIN [23]. For manipulators with 6 degrees-of-freedom the total number of arithmetic operations is 2462. Notice also that for any n the forward dynamics algorithm based on Kalman filtering and smoothing is more efficient than the above method. The totals given in Table 1 and 2 do not include the calculation required to evaluate the sines and cosines.

5. SIMULATION RESULTS

The inverse and forward dynamics algorithms presented in Sect.3 have been coded in the TURBO PASCAL 5.0 language and run on the IBM PC compatible computer [9, 15]. The modified Denavit-Hartenberg notation has been used. The software programs have modular structure which allows for extension of existing software packages. The programs which solve the inverse and forward dynamics problems are called INV_DYN and FORW_DYN, respectively. Sine and cosine functions have been discretized over the range from 0° to 90° with an interval 0.1° (repeated time-consuming calculations of sine and cosine functions have been omitted thanks to putting the discretization results in look-up tables). Software packages have been written in a user-friendly manner with all necessary information presented on the screen. The simulation results can be presented both in graphical and numerical form. The programs allow the user to simulate the inverse and forward dynamics for any manipulator with maximum 10 degrees of freedom. The user has to specify the joint trajectory (in the simulation process we have assumed fifth-order polynomial trajectory for each joint). Next, we have to specify dynamic parameters for each joint. The inverse dynamics algorithm has to be run first in order to calculate the joint torques or forces. These data are used as input for the forward dynamics program which results in joint accelerations. The latter are exactly the same as the accelerations generated by the trajectory generator at the very begining of the simulation process.

Several examples have been run in order to test the software packages. Varius examples can be found in reference [9]. There have been considered the inverse and forward dynamics models for such robots as DDA (Direct Drive Arm) I and II [13], Puma 560 [15] and others.

As a representative example let us consider the dynamic model of the

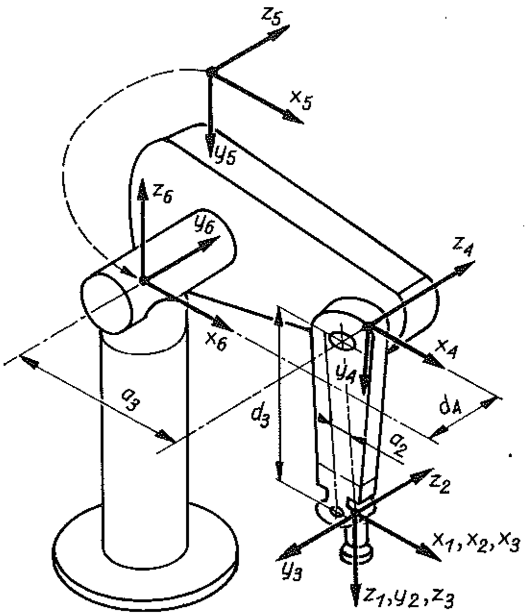

FIG. 2. The robot PUMA 560 with coordinate systems according to the modified Denavit - Hartenberg notation.

PUMA 560 robot. The PUMA manipulator has been presented in Fig.2. Notice that in Fig.2 we have used the modified Denavit - Hartenberg notation and the link numbering system starting from the tip of the manipulator toward the base, according to the convention introduced by Rodriguez.

The mass parameters of the links have been enumerated in reference [15] and we are not going to describe these parameters in detail. All joints of the manipulator were moved simultaneously in a time period of 4 second. The total movements of joints were respectively:

joint 1 in the range from $-150°$ to $150°$
joint 2 in the range from $-90°$ to $90°$,
joint 3 in the range from $-130°$ to $130°$,
joint 4 in the range from $-240°$ to $70°$,
joint 5 in the range from $-220°$ to $40°$,
joint 6 in the range from $-150°$ to $150°$,

with zero initial conditions for velocities and accelerations. For each joint 300 data points were recorded. A comlpete set of joint displacements q_i and their derivatives $\dot{q}_i$, $\ddot{q}_i$ as well as joint torques τ_i can be found in [15].

In this paper we show the torque of joint 5 (Fig.3). Joint 5 has been chosen because interactive forces and torques are the strongest between the first three links of the PUMA 560 manipulator.

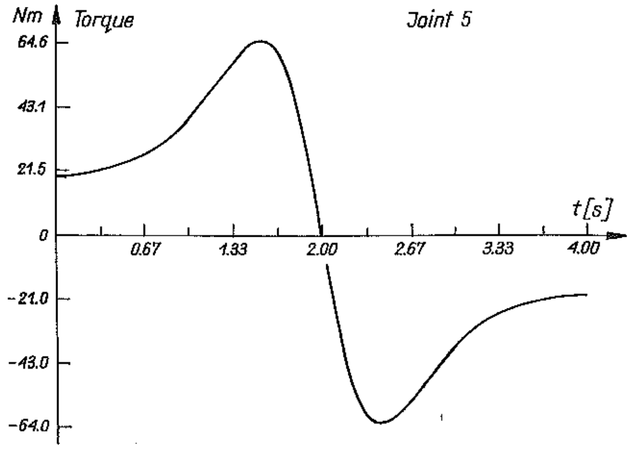

FIG. 3. Joint 5 torque for the PUMA 560 manipulator.

The results have been compared with these presented by VUSCOVIC and co-authors [15], which were obtained by making use of the standard Newton-Euler formulation. Our results are exactly the same. This substantiates the present author's belief that the algorithms presented in Sect. 3 are correct.

It is known that the model based control of a manipulator [1] requires a dynamics model (i.e. an inverse dynamics algorithm) to be calculated in real time. In the presented example the torques for the PUMA 560 manipulator were calculated with the frequency 200 Hz on the IBM PC compatible computer with 80386 processor and 80387 coprocessor. The minimum required frequency of cited in the literature is 50 Hz. One can notice that the new inverse dynamics algorithm presented in Sect. 3 satisfies these requirements. At the same time we have measured the frequency of calculating the forward dynamics algorithm presented in this paper. For our example we have obtained the frequency around 100 Hz. These results have been compared with the minimum number of arithmetic operations presented in Sect. 4 and the clock frequency of the computer. The frequency calculated from these data, i.e. the number of arithmetic operations and the clock frequency, is the same as the frequency obtained experimentally for several examples presented in [15].

We can conclude that the necessary number of arithmetic operations presented in Sect. 4 for the inverse and forward dynamics algorithms is correct.

6. Summary and conclusions

In this paper, we have reviewed the inverse and forward dynamics problems which have been originally solved by Rodriguez. We have presented a new proof for the bias spatial forces. Also we have proved the equivalence between the standard Newton - Euler formulation and the inverse dynamics algorithm derived by Rodriguez. We have presented the computational complexity of both inverse and forward dynamics algorithms based on Kalman filtering techniques and compared it with the computational complexity of the methods described in literature. Both algorithms are very efficient, in comparison with the standard methods excluding parallel algorithms for robot dynamics computation [8].

Apart from that, we have investigated the computational improvements which result from the application of fast algorithms such as factorization methods [3]. In particular, we have used the Agee and Turner factorization method, for the inertia prediction equation (3.47). However, the application of this technique has not rendered satisfying results as far as the computational complexity is concerned, due the special form of the matrix H_k. The details of this analysis can be found in [9].

The forward dynamics equations are also not well suited for the fast Kalman techniques [16]; however, we are still investigating this problem. Simulation results for both inverse and forward dynamics algorithms have been tested on several industrial robots and are reported in [15].

Much work is required to establish the computational complexity for the closed-form inertia matrix inverse algorithms which have been developed in [18]. The studies of the computational complexity of the flexible manipulator inverse and forward dynamics [21] have still to be conducted. It will be the subject of future publications of the present author.

References

1. C.H.An, C.G.Atkeson and J.M.Hollerbach, *Model based control of a robot manipulator*, MIT Press, 1988.

2. B.Armstrong, O.Khatib and J.Burdick, *The explicit dynamic model and inertial parameters of the Puma 560 arm*, Proc. of the 1986 Intern. Conf. on Robotics and Automation, 510-518, San Francisco 1986.

3. G.J.Bierman, *Factorization methods for discrete sequential estimation*, Academic Press, 1977.

4. H.BRANDL, R.JOHANNI AND M.OTTER, *A very efficient algorithm for the simulation of robots and similar multibody systems without inversion of the mass matrix*, Proc. of the IFAC/IFIP/IMACS Intern. Symp. on the Theory of Robots, 365-370, Vienna 1986.

5. A.E.BRYSON and Y.C.HO, *Applied optimal control*, Blaisdell 1969.

6. J.J.CRAIG, *Introduction to robotics mechanics and control*, Addison-Vesley Publ. Comp., 1986.

7. R.FEATHERSTONE, *The calculation of robot dynamics using articulated-body inertias*, Int.J.Rob.Res., **2**, 13-29, 1983.

8. A.FIJANY and A.K.BEJCZY, *A class of parallel algorithms for computation of the manipulator inertia matrix*, IEEE Trans. on Robotics and Automation, **5**, 5, 600-615, 1989.

9. M.GMIĄT AND P.MAĆKOWIAK, *Application of Kalman filtering techniques in robot dynamics algorithms*, M.S.Thesis [in Polish], Poznań Technical University, 1990.

10. J.M.HOLLERBACH, *A recursive Lagrangian formulation of manipulator dynamics and a comparative study of dynamics formulation complexity*, IEEE Trans.Syst., Man and Cybernet., SMC-10, 11, 730-736, 1980.

11. R.E.KALMAN, *A new approach to linear filtering and prediction problems*, ASME Trans. J. Basic Eng., vol. D, 35-45, 1960.

12. W.KHALIL and J.F.KLEINFINGER, *Minimum operations and minimum parameters of the dynamic models of tree structure robots*, IEEE J. Rob. and Aut., **RA-3**, 6, 517-526, 1987.

13. P.K.KHOSLA, *Real-time control and identification of direct-drive manipulators*, Ph. D. Thesis, Carnegie-Mellon University, 1986.

14. K.KOZŁOWSKI, *Robot dynamics algorithms using Kalman filtering and smoothing techniques*[in Polish], in: Tech.Rep. 89-007 written by KASIŃSKI, K.KOZŁOWSKI and M.PIŃCZAK, 21-94, Poznań Technical University, 1989.

15. K.KOZŁOWSKI and W.WRÓBLEWSKI, *Efficient 0(n) computation of the inverse and forward dynamics algorithm* [in Polish], Tech. Rep.90-004, Poznań Technical University, 1990.

16. P.MAROSZ, *Robot dynamics algorithms and Kalman filtering techniques* [in Polish], M.S.Thesis, Poznań Technical University, 1989.

17. G.RODRIGUEZ, *Kalman filtering, smoothing and recursive robot arm forward and inverse dynamics*, JPL Publ., 86-48, NASA, 1986.

18. G.RODRIGUEZ, *Kalman filtering, smoothing and recursive robot arm forward and inverse dynamics*, IEEE J.Rob. and Aut., **RA-3**, 6, 624-639, 1987.

19. G.RODRIGUEZ, *Recursive dynamics of topological trees of rigid bodies via Kalman filtering and Bryson-Frazier smoothing*, 6th VPI/US Symp. Control of Large Structures, VA, 45-60, Blacksburg, June 1987.

20. G.RODRIGUEZ and K.KREUTZ, *Recursive mass matrix factorization and inversion. An operator approach to open-and closed-chain multibody dynamics*, JPL. Publ., 88-11, NASA, March 15, 1988.

21. R.RODRIGUEZ, *Spatial operator approach to flexible manipulator inverse and forward dynamics*, Proc. of the 1990 Intern. Conf. on Robotics and Automation, 845-850, Cincinnati 1990.

22. M.VUSKOVIC, TING LIANG and KASI ANANTHA, *Decoupled parallel recursive Newton - Euler algorithm for inverse dynamics*, Proc. of the 1990 IEEE Intern. Conf. on Robotics and Automation, 832-838, Cincinnati 1990.

23. M.W.WALKER and D.E.ORIN, *Efficient dynamics computer simulation of robotics mechanisms*, J.Dyn. Syst., Measur. and Contr., **104**, 205-211, 1982.

24. J.WITTERBURG, *Dynamics of systems of rigid bodies*, R.G.Teuber, Stuttgart 1977.

S T R E S Z C Z E N I E

NOWE ALGORYTMY DLA DYSKRETNEGO FILTRU KALMANA W ZASTOSOWANIU DO ROZWIĄZANIA ZAGADNIENIA ODWROTNEGO ORAZ PROSTEGO DYNAMIKI MANIPULATORÓW.

W pracy udowodniono rownoważność dwóch metod budowy modelu matematycznego dynamiki manpulatora metody rekurencyjnej wywodzącej się z równań Newtona-Eulera oraz metody sprowadzającej się do rozpatrywania zagadnień kinematyki oraz dynamiki w postaci "dwupunktowego zagadnienia brzegowego". Rozwiązanie "dwupunktowego zagadnienia brzegowego" dla manipulatora o n stopniach swobody prowadzi do równań filtru Kalmana oraz wygładzania Brysona-Fraziera. W pracy przedstawiono analizę porównawczą algorytmów rozwiązania zagadnienia odwrotnego oraz prostego z dotychczas stosowanymi algorytmami pod względem ich efektywności, tj. liczby koniecznych działań arytmetycznych. Jako wynik szczegółowy, w pracy pokazano, że zależność złożoności obliczeniowej algorytmu zagadnienia prostego dynamiki, opartego na filtracji i wygładzaniu Kalmana, od liczby ogniw jest wielomianem stopnia pierwszego. W pracy wskazano na możliwość praktycznego zastosowania poprawnych algorytmów w zagadnieniu sterowania robotem opierając się na jego modelu w czasie rzeczywistym. Wyniki badań symulacyjnych rozwiązania zagadnienia odwrotnego i prostego zilustrowano na przykładzie robota PUMA 560.

Р е з ю м е

НОВЫЕ АЛГОРИТМЫ ДЛЯ ДИСКРЕТНОГО ФИЛЬТРА КАЛМАНА В ПРИМЕНЕНИИ К РЕШЕНИЯМ ОБРАТНОЙ И ПРЯМОЙ ЗАДАЧИ ДИНАМИКИ МАНИПУЛЯТОРОВ

В работе доказана эквивалентность двух методов построения математической модели динамики манипулятора и рекуррентного метода, выводящегося из уравлений Ньютона-Эйлера и метода, сводящегося к рассматрению задач кинематики и динамики в виде "двухточечной краевой задачи". Решение "двухточечной краевой задачи" для манипулятора о n степенях свободы приводит к управнениям фильтра Кальмана и сглаживания Брайсона-Фразьера. В работе представлен сравнительный анализ алгоритмов решения обратной и прямой задач с применыаемыми до сих пор алгоритмами по отношению к их эффективности, т.е. количества необходимых арифметических действий. В работе показано, как частный результат, что зависимость расчетной сложности алгоритма прямой задачи динамики, опирающейся на фильтрации и сглаживании Кальмана, от количества звеньев является многочленом первой степени. В работе указана возможность практического применения правильных алгоритмов в задачах управления роботом, опираясь на его модель в действительном времени. Результаты моделирующих исследований решения обратной и прямой задач иллюстрированы на примере робота ПУМА 560.

ROBOTICS AND AUTOMATION LABORATORY
TECHNICAL UNIVERSITY OF POZNAŃ.

Received November 6, 1990 г.