A New Computational Structure For Real-Time Dynamics

Alberto Izaguirre
Computer Science Department
New Jersey Institute of Technology
Newark, New Jersey 07102

Minoru Hashimoto
Department of Mechanical Engineering
University of Electrocommunications
1-5-1 Chofugaoka, Chofu
Tokyo, 182 Japan

Richard P. Paul
GRASP Laboratory
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, Pennsylvania 19104-6390

Vincent Hayward
Research Center for Intelligent Machines
Department of Electrical Engineering
McGill University
Montréal, Québec, Canada H3A 2A7

Abstract

We present an efficient structure for the computation of robot dynamics in real time. The fundamental characteristic of this structure is the division of the computation into a high-priority synchronous task and low-priority background tasks, possibly sharing the resources of a conventional computing unit based on commercial microprocessors. The background tasks compute the inertial and gravitational coefficients as well as the forces due to the velocities of the joints. In each control sample period, the high-priority synchronous task computes the product of the inertial coefficients by the accelerations of the joints and performs the summation of the torques due to the velocities and gravitational forces. Kircanski et al. (1986) have shown that the bandwidth of the variation of joint angles and of their velocities is an order of magnitude less than the variation of the joint accelerations. This result agrees with the experiments that we have carried out using a PUMA 260 robot.

Two main strategies contribute to reduce the computational burden associated with the evaluation of the dynamic equations. The first involves the use of efficient algorithms for the evaluation of the equations. The second is aimed at reducing the number of dynamic parameters by identifying beforehand the linear dependencies among these parameters, as well as carrying out a significance analysis of the parameters' contribution to the final joint torques.

We selected an iterative procedure for the computation of the inertial and gravitational coefficients (Featherstone 1984; Renaud 1985; Izaguirre and Paul 1986), and a recursive iteration for the computation of the velocity torques (Khalil et al. 1986). In our experiments with a PUMA 260, we obtained a set of 52 linearly independent parameters from an initial set of 78 parameters. Identification of the parameters revealed only 23 parameters to be significant.

These reductions permit the calculation of the inertias and gravitational coefficients, for the PUMA 260 without load, with 98 multiplications and 70 additions and calculation of the velocity torques with 140 multiplications and 110 additions. In the case of an arbitrary load at the end effector, calculation of the inertias and gravitational coefficients requires 190 multiplications and 150 additions and that of the velocity torques, 200 multiplications and 170 additions. Velocity torques, inertial coefficients, and gravitational coefficients can be computed in the background in 20 ms using an Intel 8087 microprocessor. The synchronous task requires only six multiplications and six additions per joint.

The actual code used to evaluate this dynamic model is entirely computer generated from experimental data, requiring no other manual intervention than performing a campaign of measurements.

The International Journal of Robotics Research,
Vol. 11, No. 4, August 1992,

1. Introduction

The inverse dynamics equations of a robot express the generalized forces or torques to be applied at the different joints of the manipulator as functions of the desired position, velocity, and acceleration of these joints. The well-known computed torque control scheme is based on calculating the generalized forces from a model of the manipulator. These forces can be used as signals to be added as feed-forward terms in a conventional feedback control loop in order to linearize and decouple the system, thus generally improving performance. The equations expressing these forces may be written in the following form:

$$F_i = (D_{ii} + I_{ai}) * \ddot{q}_i + \sum_{j=1, j\neq i}^{n} D_{ij} * \ddot{q}_i + \sum_{i,j,k} D_{ijk} * \dot{q}_i * \dot{q}_k + D_i + F_{di} * \dot{q}_i + F_{si} * \mathrm{sgn}(\dot{q}_i), \quad (1)$$

where D_{ii} is the effective inertia at the joint i, D_{ij} is the coupling inertia between joints i and j, D_{ijk} are the Coriolis and centrifugal coefficients at the joint i, D_i is the gravitational force at the joint i, I_{ai} is actuator inertia, F_{di} is the viscous friction, F_{si} is the Coulomb friction, F_i is the generalized force at the joint i, $\dot{q}_i$ is the velocity of the joint i, and $\ddot{q}_i$ is the acceleration of the joint i.

This control scheme requires the evaluation of the dynamic equations in real time—that is, within the sample period of a digital controller. For this purpose, a great deal of research has been aimed at reducing the complexity of these equations. However, the problems of the automatic generation of equations and identification of the constant parameters in the equations (functions of the moments of the links, frictions, dampings, and inertias on the motors) are problems related in a practical fashion.

In this article, we address these three problems, making appropriate references to previous works; explain our contributions; and show the results obtained from our experiments with the PUMA 260 robot.

2. Overview of the Inverse Dynamics Computation

We now review the work that has been done in the past from the point of view of implementation complexity and parameter identification for the inverse dynamic equations of robot manipulators.

2.1. *Complexity of the Inverse Dynamics Calculation*

There are two main approaches to derive the required equations: the Lagrange formalism and the Newton-Euler formalism. The Lagrangian approach, developed by Uicker (1968), has been used by several researchers, including Khan and Roth (1971), Paul (1972), and Bejcsy (1974). The principal disadvantage of this formulation is the complexity in the order $O(n^4)$ caused by redundancies in the calculation. A simplification using a forward recursion on the velocities and accelerations of the joints and a backward recursion on the generalized forces was introduced by Hollerbach (1980; 1983). This approach simplified the computation substantially, reducing the complexity to a linear function of the number of joints—i.e., $O(n)$. Unfortunately, this method cannot compute the dynamics coefficients that depend only on the joint angles. It thus encounters the same problem encountered by the Newton-Euler computation. Megahed (1984) calculated the dynamic equations based on the Lagrange equations and the dynamic coefficients. The complexity achieved is in the order of $O(n)$, requiring approximately 1000 multiplications and 700 additions for a general manipulator with six degrees of freedom.

Featherstone (1984), following a spatial notation, and Renaud (1984), following a tensorial notation, have reduced the computation of the dynamic coefficients using the Lagrangian method and the notation of the "compound link." We used the theorem of "conservation of momentum" (Izaguirre and Paul 1986), achieving equivalent results in the calculation of the inertial and gravitational coefficients. Basically, the computation consists of the calculation of the moments of the "compound link" i (the link formed by the links i through the last link) as a function of the moments of the "compound link" $i + 1$. This recursion leads to large savings in the calculations and provides a systematic method for calculating the dynamic coefficients. The complexity resulting from this approach is $O(n^3)$ if the entire dynamic model is computed and $O(n^2)$ if only the inertial and gravitational coefficients are calculated.

The second approach, the Newton-Euler method, consists of calculation of the generalized forces by using Newton's law to calculate forces and Euler's law to calculate torques. One of the first methods of calculation of the generalized forces was developed by Likins (1971). Luh et al. (1980) developed an algorithm using a forward recursion for the velocities and acclerations of the joints and a backward recursion for the calculation of the generalized forces. The complexity of the algorithm is $O(n)$, and only 800 multiplications and 600 additions are required for a general six-degree-of-freedom manipulator. Khalil et al. (1986), based on this previous work, reduced the computation by regrouping common terms. The dynamic equations can be calculated by this algorithm in 540 multiplications and 480 additions in the case of a general six-degree-of-freedom manipulator.

2.2. *Computational Structures for the Inverse Dynamic Equations*

Different approaches have been developed for the computation of dynamic equations in real time. Luh et al. (1980), based on the Newton-Euler method, computed the dynamic equations for the Stanford manipulator in 4.5 ms using floating point assembly language in a PDP-11/45. Raibert (1977) used look-up tables to compute the dynamic equations. A total of 460 multiplications and 260 additions were required to calculate the equations of a general six-degree-of-freedom manipulator, reducing the complexity of the Newton-Euler method by a factor of 2. Lin and Luh (1982) described a procedure for scheduling subtasks among a group of six microprocessors, one per joint, in order to compute the Newton-Euler equations. Their estimation indicates that 320 multiplications and 280 multiplications are required to compute the dynamics of a six-degree-of-freedom manipulator.

Orin et al. (1985) introduced a structure of control by dividing it into 10 different tasks, arriving at the conclusion that the longest task is the one that computes the inverse dynamics. Lathrop (1985) has studied the parallel computation of the inverse dynamics. A pipelined architecture reducing the Newton-Euler computation by two orders of magnitude was proposed. The latency to compute the dynamics of a six-degree-of-freedom manipulator became of the order of 15 multiplications and 43 additions. The Newton-Euler parallel implementation reduced the complexity to a logarithmic expression on the number of joints. Only 11 multiplications and 28 additions are required to compute the manipulator dynamics. Also, a systolic pipeline implementation is possible, reducing the latency to only four floating-point operations. The disadvantage of this method lies in the difficulty of implementation, because custom-designed very large scale integration (VLSI) devices are required, increasing the cost of the product. We will introduce later a solution based on microprocessors, which has the advantages of an easy implementation, maintaining at the same time the required speed of computation and the accuracy in the calculation of the equations.

2.3. *Identification of the Robot's Parameters*

The constant parameters of the dynamic equations depend on the masses, centers of gravity, and inertias of the links, as well as on the inertias and frictions of the motors. Ferreira (1984) realized that the torques could be expressed as a linear function of these parameters. He also pointed out that many parameters were linearly dependent and that it was necessary to eliminate these dependencies. However, no algorithm was suggested to achieve this. In his experiments, he identified the parameters using the torque measured in the first joint of the robot TH8, using a Kalman filter.

An et al. (1985) identified the parameters of a direct-driven arm by using a least-squares method to fit the measured torques along a given trajectory. The joints' position and torque along the trajectory were measured, estimating the velocity and acceleration. However, the problem of the elimination of the linearly dependent parameters was not mentioned, although it may introduce erroneous estimation of the parameters. Nevertheless, the fit seemed good. Khosla and Kanade (1986) presented an algorithm to estimate the linear dependencies in the dynamic model. Olsen and Bekey (1986) identified the constant parameters in simulation, but special cases were considered to identify different parameters. Finally, Armstrong et al. (1986) estimated the parameters of a PUMA 560, disassembling the robot to directly measure the various inertial parameters. They were also able to perform a significance analysis in order to reduce the complexity of the model.

The next section presents an identification method based on fitting the measured torques over different trajectories, removing the linear dependencies, and at the same time performing a significance analysis, which reduces the computation considerably.

3. Our Approach

We now present a new approach, based on conventional microprocessors, to compute the inverse dynamics in real time. The scheme is based on the division of the computation into a high-priority synchronous task and a background task (Izaguirre and Paul 1985, 1986). The background task updates the inertial and gravitational coefficients, as well as the generalized forces resulting from the velocities of the joints. The synchronous task computes the final generalized forces by multiplying the inertial coefficients by the acceleration of the joints, adding at the same time the gravitational and velocities forces. This computational scheme agrees with the experiments that have been done by Kircanski et al. (1986). Trajectories for the PUMA 560 robot have been calculated and the bandwidth of the position, velocity, and acceleration of the joints respectively analyzed. This work led us to conclude that the position and velocities spectra are similar, whereas the spectrum of the acceleration is about five times larger. In our experiments, we estimated the velocities and accelerations from the measurements of the joint angles.

We wanted to calculate the inertial coefficients, as well as the forces caused by the velocities, as fast as possible. To do so, we selected an efficient recursive algorithm to compute the inertial and gravitational coefficients (Izaguirre and Paul 1986) and an efficient recursive algorithm

to compute the velocity forces (Khalil et al. 1986). Identification of the parameters permits a further reduction by first eliminating the linear dependencies and dropping the parameters that are not significant.

In the next sections we will explain in detail the computation and identification of the dynamic coefficients, giving the results obtained for the PUMA 260 robot.

4. Calculation of the Dynamic Coefficients

In this section we first develop the equations for the inertial and gravity-loading coefficients for a infinitesimal mass dm located in link j. These equations are elaborated from the theorem of conservation of the momentum (Izaguirre 1985, 1986), leading to an easy and understandable procedure. The calculation for the entire link will be obtained by integrating these formulas over the mass of the link.

4.1. Calculation of D_{ij}s

The coefficient D_{ij} corresponds to the generalized force in the joint j caused by the acceleration of the joint i. We will consider only the terms for which $i < j$ and then show that $D_{ij} = D_{ji}$. We consider in this article the revolute-revolute case for the joints i and j. (All other cases are treated in detail in Izaguirre et al. [1987]). The acceleration $\ddot{q}_j$ is different from zero, but all the other accelerations and velocities are set to zero.

In the revolute-revolute case, the term D_{ij} corresponds to the torque in the joint i caused by acceleration q_j only. This torque can be calculated by derivating the angular momentum around the axis i with respect to time. We recall that the angular momentum of a point of mass dm with respect to a coordinate system is calculated by the cross product $\mathbf{r} \times \mathbf{v}\ dm$, where $\mathbf{r}$ is the position and $\mathbf{v}$ is the velocity of the mass dm in this coordinate system. The variation of the angular momentum around $\mathbf{z}$ is $dm\ \mathbf{r}_i \times (\mathbf{z}_j \times \mathbf{r}_j) \cdot \mathbf{z}_i \ddot{q}_i \Delta t$. The torque is calculated by differentiating the angular momentum with respect to time:

$$\Gamma_i = dm\, \mathbf{r}_i \times (\mathbf{z}_j \times \mathbf{r}_j) \cdot \mathbf{z}_i \ddot{q}_i. \tag{2}$$

The term D_{ij} is then given by:

$$\begin{aligned} D_{ij} &= dm\, \mathbf{r}_i \times (\mathbf{z}_j \times \mathbf{r}_j) \cdot \mathbf{z}_i \\ &= dm\, (\mathbf{z}_i \times \mathbf{r}_i) \cdot (\mathbf{z}_j \times \mathbf{r}_j). \end{aligned} \tag{3}$$

This formula reveals the symmetry between D_{ij} and D_{ji}.

4.2. Calculation of D_is

To compute the gravity coefficients D_i, we consider the revolute case (the general case being treated in Izaguirre et al. [1987]).

The torque exerted by the gravity on the revolute joint is equal to:

$$\Gamma_i = dm\, (\mathbf{r}_i \times \mathbf{g}) \cdot \mathbf{z}_i, \tag{4}$$

where $\mathbf{g}$ is the acceleration caused by gravity.

To compensate for the gravity load, a torque must be exerted in the opposite direction. Thus the coefficient D_i is given by:

$$\begin{aligned} D_i &= -dm\, (\mathbf{r}_i \times \mathbf{g}) \cdot \mathbf{z}_i \\ &= -dm\, (\mathbf{z}_i \times \mathbf{r}_i) \cdot \mathbf{g}. \end{aligned} \tag{5}$$

4.3. Integration of the Equations

In this section we will integrate the equations derived earlier over one link. This link corresponds to the "compound link" j, the link formed by the links j through the last link, n. In fact, the calculation of the coefficients D_{ij} and D_j depends only on the acceleration of the joint j, with the other links considered relatively frozen to each other. A recursive calculation of the moments of the "compound link" j as a function of the "compound link" $j+1$ leads to a large reduction in the calculation of the inertial and gravitational coefficients. This recursion will be explained later in detail.

The most difficult term to integrate is the parameter D_{ij} for the revolute-revolute case. The expression of this term for a point mass situated in the "compound link" j, $(j > i)$, is:

$$\begin{aligned} D_{ij} &= dm\, (\mathbf{z}_i \times \mathbf{r}_i) \cdot (\mathbf{z}_j \times \mathbf{r}_j) \\ &= dm\, (\mathbf{z}_i \times (\mathbf{p}_i + \mathbf{l}_j)) \cdot (\mathbf{z}_j \times \mathbf{l}_j), \end{aligned} \tag{6}$$

where $\mathbf{p}_i$ is the vector between the origins of frames i and j and $\mathbf{l}_j$ is the vector between the origin of frame j and the elementary mass dm.

The expansion of this expression leads to the following formula:

$$D_{ij} = dm\, [(\mathbf{z}_i \times \mathbf{p}_i) \cdot (\mathbf{z}_j \times \mathbf{l}_j) + (\mathbf{z}_i \times \mathbf{l}_j) \cdot (\mathbf{z}_j \times \mathbf{l}_j)]. \tag{7}$$

The integration of the first term is obtained by the substitution of the point mass by the center of gravity of the "compound link." The last term can be integrated using tensor notation. The term $(\mathbf{z}_i \times \mathbf{l}_j)$ is expressed in a tensor notation as $-\hat{\mathbf{l}}_j \mathbf{z}_i$ or as $\mathbf{z}_i \hat{\mathbf{l}}_j$, leading to the following expression:

$$dm\, (\mathbf{z}_i \times \mathbf{l}_j) \cdot (\mathbf{z}_j \times \mathbf{l}_j) = -dm\, \mathbf{z}_i \hat{\mathbf{l}}_j \hat{\mathbf{l}}_j \mathbf{z}_j, \tag{8}$$

$$-\hat{\mathbf{l}}_j \hat{\mathbf{l}}_j = -\begin{pmatrix} 0 & -l_{jz} & l_{jy} \\ l_{jz} & 0 & -l_{jx} \\ -l_{jy} & l_{jx} & 0 \end{pmatrix} \begin{pmatrix} 0 & -l_{jz} & l_{jy} \\ l_{jz} & 0 & -l_{jx} \\ -l_{jy} & l_{jx} & 0 \end{pmatrix}$$

$$= \begin{pmatrix} l_{jz}^2 + l_{jy}^2 & -l_{jx} l_{jy} & -l_{jx} l_{jz} \\ -l_{jx} l_{jy} & l_{jx}^2 + l_{jz}^2 & -l_{jy} l_{jz} \\ -l_{jx} l_{jz} & -l_{jy} l_{jz} & l_{jx}^2 + l_{jz}^2 \end{pmatrix}$$

This term, integrated over the link, corresponds to the inertial matrix of the "compound link" j in the frame j. The final expression is:

$$D_{ij} = (\mathbf{z}_i \times \mathbf{p}_i) \cdot (\mathbf{z}_j \times \mathbf{M}_j * \mathbf{D}_j) + \mathbf{z}_i \hat{\mathbf{I}}_j \mathbf{z}_j$$
$$= (\mathbf{z}_i \times \mathbf{p}_i) \cdot (\mathbf{z}_j \times \mathbf{L}_j) + \mathbf{z}_i \hat{\mathbf{I}}_j \mathbf{z}_j, \quad (9)$$

where $\mathbf{M}_j$ is the mass of the compound link j, $\mathbf{D}_j$ is the center of gravity of the "compound link" j on the origin of the frame j, $\mathbf{L}_j$ is $\mathbf{D}_j$ multiplied by $\mathbf{M}_j$ (i.e., the first moment of the "compound link" j in the frame j), and $\hat{\mathbf{I}}_j$ is the inertia matrix of the "compound link" j in the origin of the frame j.

4.4. Calculation of Coefficients Using Homogeneous Transformations

In this section we explain how the inertial and gravitational coefficients can be computed once the architecture of the robot is defined by means of homogeneous transformations. These transformations describe the coordinate changes between the ith and jth frames. Let the ${}^iT_j$s be matrices representing these transformations, with the convention that the first frame is the frame number 0 and the last link corresponds to the frame number n.

The part of the homogeneous transformation corresponding to the rotation is the matrix ${}^iR_j$, and the part corresponding to the translation is the vector ${}^ip_j$.

The matrix ${}^iR_j$ can be decomposed into three vectors ${}^in_j$,${}^io_j$. and ${}^ia_j$:

$${}^iT_j = \begin{pmatrix} {}^in_{jx} & {}^io_{jx} & {}^ia_{jx} & {}^ip_{jx} \\ {}^in_{jy} & {}^io_{jy} & {}^ia_{jy} & {}^ip_{jy} \\ {}^in_{jz} & {}^io_{jz} & {}^ia_{jz} & {}^ip_{jz} \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (10)$$

Expression for the D_{ij} Coefficients

In this section we will derive the expressions of the dynamic coefficients using homogeneous transformations. For the revolute-revolute case, equation (9) becomes:

$$D_{ij} = (0\,0\,1)\,{}^iR_j I_j (0\,0\,1)^T + \left(-{}^ip_{jy}\;{}^ip_{jx}\;0\right)\,{}^iR_j \left(-{}^jL_y\;{}^jL_x\;0\right)^T, \quad (11)$$

where $i \in [0, n-1]$ and $j \in [0, n-1], n$ being the number of degrees of freedom; $({}^jL_x\;{}^jL_y\;{}^jL_z)$ are the components of the first moments of the "compound link" expressed in the frame j; and I_j is the inertia matrix of the compound link j expressed in the frame j.

Equation (11) leads to:

$$D_{ij} = \left({}^in_{jz}\;{}^io_{jz}\;{}^ia_{jz}\right) \begin{pmatrix} I_{13j} \\ I_{23j} \\ I_{33j} \end{pmatrix} + (AB) * \left(-{}^jL_y\;{}^jL_x\right)^T \quad (12)$$

with $A = \left(-{}^ip_{jy} * {}^in_{jx} + {}^ip_{jx} * {}^in_{jy}\right)$

with $B = \left(-{}^ip_{jy} * {}^io_{jx} + {}^ip_{jx} * {}^io_{jy}\right)$.

Expression of the D_i Coefficients

Equation (5) can be expressed as

$$D_i = -\left(g_x\; g_y\; g_z\right)\,{}^0R_i \left(-{}^iL_y\;{}^iL_x\;0\right)^T, \quad (13)$$

where $(g_x\; g_y\; g_z)$ are the components of the acceleration of gravity in the base frame. If the gravity is parallel to the z axis of the base frame, equation (13) assumes the following simplified form:

$$D_i = -g\left({}^0n_{iz}\;{}^0o_{iz}\right)\left(-{}^iL_y\;{}^iL_x\right)^T. \quad (14)$$

4.5. Recursion of the Terms

In the previous sections we derived the expression of the inertial and gravitational terms for the "compound link" j. In this section we will work out the recursion for the calculation of the moments of the "compound link" j as a function of those of the "compound link" $j+1$.

Recursion for the Masses

Obviously, $M_j = M_{j+1} + m_{j+1}$, where m_j is the mass of the link j and M_j is the mass of the "compound link" j.

Recursion for the First Moment

The first moment of the "compound link" j corresponds to the product of the mass M_j with the center of gravity of the "compound link" j. It can be expressed recursively by using the following expression:

$${}^j\mathbf{L}_j = {}^jR_{j+1}\left({}^{j+1}\mathcal{L}_j + {}^{j+1}\mathbf{L}_j + {}^{j+1}\mathbf{p}_{jj+1} * M_j\right), \quad (15)$$

where

- ${}^j\mathbf{L}_j$ is the first moment of the "compound link" j expressed in the frame j.
- ${}^{j+1}\mathcal{L}_j$ is the first moment of the link j expressed in the frame $j+1$. The fact that we expressed it in the frame $j+1$ is due to the selection of the Denavit-Hartenberg parameters. The moments of a link are constant with respect to the next frame, rather than with respect to the jth frame. This is why modified Denavit-Hartenberg parameters have been sometimes used (Craig 1986; Khalil et al. 1986), in which case the moments of a link are constant with respect to the relative frame. However, the two approaches lead to similar results from a

computational viewpoint; the differences are in the notation.

- ${}^{j+1}\mathbf{p}_{ij+1}$ corresponds to the vector between the origins of frames j and $j+1$. It is also constant with respect to $j+1$ if we choose the Denavit-Hartenberg parameters and with respect to j if we choose the modified Denavit-Hartenberg parameters.

Recursion for the Inertia

The inertia of the "compound link" j can be expressed as a function of "compound link" $j+1$ by transforming the inertia of the "compound link" $j+1$ from the origin of the frame $j+1$ to its center of gravity and from the center of gravity to the origin of the frame j. Also, we have to add the inertia of the link j following a similar pocedure. This can be expressed as follows:

$$\begin{aligned} {}^{j+1}I_j = {}& {}^{j+1}I_{j+1} + M_{j+1} * \left({}^{j+1}\hat{D}_{j+1}\,{}^{j+1}\hat{D}_{j+1}\right) \\ & - {}^{j+1}M_{j+1} * \left[\left({}^{j+1}\hat{D}_{j+1} + {}^{j+1}\hat{p}_{jj+1}\right) \right. \\ & \left. * \left({}^{j+1}\hat{D}_{j+1} + {}^{j+1}\hat{p}_{jj+1}\right) \right] \\ & + {}^{j+1}\mathcal{I}_j + m_j * \left({}^{j+1}\hat{d}_j + {}^{j+1}\hat{d}_j\right) \\ & - m_j * \left[\left({}^{j+1}\hat{d}_j + {}^{j+1}\hat{p}_{jj+1}\right) \left({}^{j+1}\hat{d}_j + {}^{j+1}\hat{p}_{jj+1}\right) \right], \end{aligned} \tag{16}$$

where ${}^{j+1}I_i$ is the inertia matrix of the "compound link" j expressed in the frame $j+1$, ${}^{j+1}\mathcal{I}_j$ is the constant inertia of the link j in the frame $j+1$, ${}^{j+1}\hat{p}_{jj+1}$ is the tensor corresponding to the vector p_{j-1j}, ${}^{j+1}\hat{D}_{j+1}$ is the tensor corresponding to the center of the gravity of the "compound link" $j+1$ in the frame $j+1$, and ${}^{j+1}\hat{d}_{j+1}$ is the tensor corresponding to the center of gravity of the link j in the frame $j+1$.

This leads to the following equation:

$$\begin{aligned} {}^{j}I_j = {}& {}^{j}R_{j+1} \left({}^{j+1}\mathcal{I}_j + {}^{j+1}I_{j+1} + M_j * {}^{j+1}\hat{p}_{jj+1}\,{}^{j+1}\hat{p}_{jj+1}\right)^{j} \\ & \times R_{j+1}^T - {}^{j}\hat{p}_{jj+1}\,{}^{j}\hat{L}_j - {}^{j}\hat{L}_j\,{}^{j}\hat{p}_{jj+1}, \end{aligned} \tag{17}$$

where the term ${}^{j+1}\hat{p}_{jj+1}$ is a constant if defined in the frame $j+1$.

These terms can be easily calculated using homogeneous transformation, as explained before for the cases of the inertial and gravitational coefficients.

5. Calculation of the Velocity Terms

There are two main methods to calculate the velocity terms. The first requires the calculation of the velocity coefficients and Coriolis and centrifugal terms and the multiplication of these coefficients by the velocities of the joint. The Coriolis and centrifugal terms can be efficiently calculated by using the Christoffel symbols over the inertial terms (Renaud 1984). The second method comprises the calculation of the velocity torques by using the Newton-Euler method. A simplification of this method using intermediate variables has been proposed by Khalil et al. (1986). This algorithm calculates the velocity torques by ignoring the gravitational effects.

The advantage of this last method is that the inertial and gravitational coefficients, as well as the velocity torques, can be computed independently, unlike the Christoffel symbols methods, which depends on the inertial coefficients. Also the complexity of the method is linear, rather than $O(n^3)$, for the method using the Christoffel symbols.

The forward recursion for the velocities and accelerations, considering only the influence caused by the velocities of the joints is the following:

$${}^{i}w_i = {}^{i-1}R_i^T * \left[{}^{i-1}w_{i-1} + (1-\sigma_i)(0\,0\,\dot{q}_i)^T \right], \tag{18}$$

$${}^{i}\dot{w}_i = {}^{i-1}R_i^T * \left[{}^{i-1}\dot{w}_{i-1} + (1-\sigma_i) * {}^{i-1}w_{i-1} \times (0\,0\,\dot{q}_i)^T \right], \tag{19}$$

$$\begin{aligned} {}^{i}\dot{v}_i = {}& {}^{i-1}R_i^T * {}^{i-1}\dot{v}_{i-1} + 2 * \sigma_i * {}^{i}w_i \times {}^{i-1}R_i^T (0\,0\,\dot{q}_i)^T \\ & + U_i * {}^{i}p_{i-1i}, \end{aligned} \tag{20}$$

where ${}^{i}w_i$ is the angular velocity of the frame i expressed in the frame i, ${}^{i}\dot{w}_i$ and ${}^{i}\dot{v}_i$ are the angular and linear accelerations of the frame i expressed in the frame i, and U_i is the matrix $U_i = \widehat{{}^{i}\dot{w}_i} + \widehat{{}^{i}w_i}\,\widehat{{}^{i}w_i}$.

The backward recursion can be expressed by using the formula:

$${}^{i+1}\mathcal{F}_i = m_i\,{}^{i+1}\dot{v}_{i+1} + U_{i+1} * {}^{i}\mathcal{L}_i \tag{21}$$

$${}^{i+1}\mathcal{N}_i = {}^{i+1}\mathcal{I}_i\,{}^{i+1}\dot{w}_{i+1} + {}^{i+1}w_{i+1} \times {}^{i+1}\mathcal{I}_i\,{}^{i+1}w_{i+1} \tag{22}$$

$${}^{i}f_i = {}^{i}R_{i+1} \left({}^{i+1}f_{i+1} + {}^{i+1}\mathcal{F}_i\right) \tag{23}$$

$$\begin{aligned} {}^{i}n_i = {}& {}^{i}R_{i+1} \left({}^{i+1}n_{i+1} + {}^{i+1}\mathcal{N}_i + {}^{i}\mathcal{L}_{i+1} \times {}^{i+1}\dot{v}_{i+1}\right) \\ & + {}^{i}p_{ii+1} \times {}^{i}f_i \end{aligned} \tag{24}$$

where ${}^{i+1}\mathcal{F}_i$ is the force caused by the motion of link i, ${}^{i+1}\mathcal{N}_i$ is the torque caused by the motion of the link i, ${}^{i}f_i$ is the total force in the link i expressed with respect to the frame i, and ${}^{i}n_i$ is the total torque in the link i expressed with respect to the frame i. Khalil et al. (1986) picked intermediate variables that further simplified the computation; the reader is referred to this article for more detail.

6. Identification of the Dynamic Coefficients

The torques can be expressed as linear functions of the masses of the first moments (the masses of the links multiplied by the center of gravity of the links) and the second moments (the inertia matrix of the links). The easiest way to show this linear relationship is to consider the Newton-Euler method. The forces and torques

caused by the velocities and accelerations of these links are linear functions of the moments of the links—i.e., products of these moments with functions that depend on the velocities and accelerations of the links, as shown earlier. In the backward recursion, these forces and torques transform from one link to the previous one in a linear form, and the influence of the new link is taken into account by the addition of the forces and torques caused by the previous link to the force and torque caused by the actual link. Finally, because the working force or torque in a joint is a projection of the forces and torques, we show that the dynamic equations can be expressed as a linear function of the link moments.

This can be shown by using the results obtained in the calculation of the inertial and gravitational coefficients. First, it is easy to verify that the "compound link" moments are linear functions of the moments of the constituent links. Second, the coefficient D_{ij} and D_i are linear functions of the moments of the compound link. Third, as the D_{ijk} terms are calculated by the Christoffel symbols which are additions of partial derivatives of the D_{ij} terms with respect to the joint angle, we prove that the terms D_{ijk} are linear function of the link moments. Finally, as the computation of the forces or torques is made by multiplications of these coefficients by the velocities and accelerations of the links, we prove that the forces/torques are linear functions of the link moments.

The problem, however, is not trivial, as many of these terms are mutually linearly dependent. It is important to calculate these dependencies in order to seek a reduction of the number of parameters to be identified, as well as to arrive at a unique estimation. In the following paragraphs we explain the method used to eliminate these dependencies, as well as the experiments that were performed using a PUMA 260 manipulator.

6.1. Algorithm to Identify the Linear Dependencies

Although there might exist analytical methods for calculating the linear dependencies between the moments of the links, it is not easy to identify them. Instead, we have implemented a numerical procedure to find these dependencies. The algorithm is based in the calculation of the rank of a matrix.

The dynamic equations can be written as:

$$\Gamma_i = D(q, \dot{q}, \ddot{q})(\eta_1 \eta_2 \dots \eta_m)^T, \tag{25}$$

where Γ_i is the generalized force of the joint i; $(\eta_1 \eta_2 \dots \eta_m)$ is a vector of the moments of the links, the Coulomb and viscous frictions, and the inertias of the motors; $D(q, \dot{q}, \ddot{q})$ is a function of the positions, velocities, and accelerations of the joints.

For the PUMA 260, there are 10 moments for each link, six static frictions, six dampings, and six motor inertias, making a total of 78 parameters. The function D is computed numerically using the Newton-Euler method, each time taking into account the torque caused by each unit parameter—i.e., the torque caused by a parameter with value equal to 1. For the motor inertias, Coulomb frictions and viscous friction, the D parameters correspond to the acceleration, sign of the velocity, and velocity of the joint, respectively.

A set of measurements leads to:

$$\Gamma = (D_1 D_2 \dots D_m)(\eta_1 \eta_2 \dots \eta_m)^T. \tag{26}$$

If we consider n different values of the position, velocity, and acceleration of the joints, this system has a dimension equal to $(6, n, m)$ for the matrix D. If we consider 100 points, a six-joint manipulator leads to a matrix D of dimension (600×78). The system is overdetermined, and the linear dependencies correspond to the dependencies between the columns of the matrix D. Suppose now that the columns $D_1 \cdots D_i$ are linearly independent. We augment the submatrix with the column D_{i+1} to compute the new rank. If the rank is equal to $i + 1$, then the new submatrix $D_1 \dots D_{i+1}$ is linearly independent. If not, D_{i+1} is linearly dependent of the previous columns, thus

$$\alpha_1 * D_1 + \cdots + \alpha_i * D_i + \alpha_{i+1} D_{i+1} = 0, \tag{27}$$

with α_{i+1} different from zero by construction. Next, we have:

$$D_{i+1} = (-\alpha_1/\alpha_{i+1}) * D_1 - \cdots - (\alpha_i/\alpha_{i+1}) * D_i. \tag{28}$$

The expression for Γ becomes:

$$\Gamma = (D_1 \dots D_m) \left[(\eta_1 - \eta_{i+1} * \alpha_1/\alpha_{i+1}) \dots \eta_m\right]^T. \tag{29}$$

We eliminate the dependencies by dropping the corresponding parameters and modifying the remaining parameters in this last equation. The algorithm to calculate the dependencies considers a D matrix with 100 random points and successively computes the rank of the submatrices $D_1 \dots D_i$, dropping one parameter each time a new dependency is found. We implemented the algorithm by computing the rank using the singular value decomposition method. To account for numerical errors, each time a smallest singular value of the order of 10^{-10} was obtained, the matrix was considered singular.

Using the IMSL library, the dependencies for the PUMA 260 were found in 15 minutes of VAX CPU time. From a starting set of 78 independent parameters, we obtained 52 linearly independent parameters. We picked the six motor inertias as the six first parameters, because they are constant numbers. Two parameters were dependent on

the actuator inertias of links 1 and 2. The static frictions and damping were independent of the other parameters. This means that from a total of 60 moments, we obtained 34 linearly independent moments.

The identification was carried out on these new independent parameters, and the PUMA 260 robot was run over 10 different trajectories selected to excite the various contributions to the dynamics of the manipulator.

7. Experiments

The experiments were performed by running a PUMA 260 over six predetermined trajectories and four randomly generated trajectories. These trajectories were polynomials that fitted points inside the range of each link of the robot. We preserved the continuity of the trajectory, imposing zero-velocity conditions at the beginning and end of each trajectory. The experiments were performed at McGill University using the RCCL environment. The curves were time scale updated to obtain maximum torque responses and enhance the signal-to-noise-ratio. The sampling period was 28 ms.

The collected data from the measured torque and measured position has been used to calculate the velocities and accelerations of the joints, and we substituted these values into the D model of the PUMA 260. The velocities and accelerations were estimated by finite difference using the formulas $\tilde{v}_i = (\tilde{q}_{i+1} - \tilde{q}_{i-1})/(2\tau)$, and $\tilde{a}_i = (\tilde{v}_{i+1} - \tilde{v}_{i-1})/(2\tau)$, where τ is the sampling period (28 ms), to filter the accelerations and velocities values (see Appendix A). We also dropped the first 15 and last 15 samples to eliminate the effects of transients.

We used a weighted least-squares procedure, because the output torque caused by the first three links is 10 to 50 times larger than the output torque caused by the last three links. This weighting is possible because the influence of the last three links over the first three is not very significant. We calculated the average, standard deviation, maximum and minimum values of the parameters. Table 1 shows the results for the fitting of all 53 parameters.

In this table, the first six parameters correspond to the actuator inertias. The last 12 parameters correspond to Coulomb frictions (the *fsis*) and viscous frictions (the *dsis*). The rest of the parameters correspond to moments of the links of the robot. The first column in Table 1 corresponds to the parameter number in the identification. The parameters considered as significant are marked by an asterisk. The second column contains the representation of each parameter; the third through sixth columns contain the average value, standard deviation, and maximum and minimum value, respectively, of the parameter over a set of 10 different trajectories.

The significant parameters were calculated by looking for the maximum contribution to the torque for each parameter. The parameters whose contributions for all trajectories were less than 1 percent of the maximum measured torque were not considered significant. Only 23 parameters were found significant. Another identification was made using only these 23 parameters, obtaining a better distribution (that is, a smaller standard deviation). This result can be attributed to the condition of the D matrix. In the first case, the condition number—the quotient between the smallest singular value and the biggest singular value—was 0.033/828.8. When we limited the fit to only 23 parameters, we obtained a condition number of 0.35/146.0, which is 60 times better. The fit of the parameters is shown in Table 2.

7.1. Identification Errors

The measured and fitting torques using the 23 parameter averages that resulted from the fitting of the curves are displayed in Appendix B. The results for the first three joints are almost perfect, as very slight differences are found (see Appendix B). For the last three joints, there are larger discrepancies, because the measured torques are small and the last three links are mechanically coupled. This coupling is not yet taken into account in the model.

If we compare the results of the frictions measured for the same robot by Lloyd (1984) (see Table 3), we can see that they almost agree completely.

We also compared the dynamic coefficients that Lloyd measured to calculate the gravitational coefficients. He found three parameters $\mathtt{c15} = -0.192N \cdot m$, $\mathtt{c13} = -1.1762N \cdot m$, $\mathtt{c12} = 5.509N \cdot m$. Developing our formulas (see Appendix A), we found that the coefficients have the following values: $\mathtt{c15} = -0.13743N \cdot m$, $\mathtt{c13} = -1.176644N \cdot m$, $\mathtt{c12} = 5.51792N \cdot m$.

Using these parameters, we automatically generated the program that calculates the inertias and gravitational coefficients, as well as the program that calculates the velocity torques (see Appendix A). These programs permit (1) calculation of the inertias and gravitational coefficients for the PUMA 260 without load, using 98 multiplications and 70 additions and (2) calculation of the velocity torques using 140 multiplications and 110 additions. In the case of an arbitrary load at the end effector, calculation of the inertias and gravitational coefficients requires 190 multiplications and 150 additions and that of the velocity torques, 200 multiplications and 170 additions.

7.2. Errors in Trajectory Following

To test the validity of our dynamic model, we have computed the necessary torques to follow predetermined tra-

Table 1. Statistics for the Parameters Obtained Experimentally

Parameter	Representation	Average	Standard Deviation	Maximum	Minimum
1 *	ia1	0.098889	0.050155	0.155265	0.07256
2 *	ia2	0.1436638	0.086779	0.243456	−0.0067
3 *	ia3	0.048211	0.054446	0.136380	−0.031769
4 *	ia4	0.003262	0.012358	0.032090	−0.010854
5 *	ia5	0.012713	0.018743	0.041647	−0.030187
6 *	ia6	0.002808	0.002849	0.009251	−0.002870
7 *	m6	2.721895	1.119838	4.311959	0.402778
8	x6	−0.0012975	0.002621	0.002830	−0.005907
9	y6	−0.000464	0.002	0.002262	−0.002939
10 *	z6	0.012975	0.009833	0.023582	−0.004141
11	a6	0.00340	0.006396	0.010470	−0.00916
12	b6	−0.001549	0.005895	0.008865	−0.0107
13	c6	0.000513	0.001691	0.003652	−0.00208
14	d6	−0.000107	0.001391	0.002072	−0.0030
15	e6	0.000655	0.001307	0.003036	−0.00168
16	f6	−0.000123	0.000831	0.001565	−0.0013
17	x5	0.005288	0.012684	0.035485	−0.00479
18	y5	0.0067	0.014106	0.039692	−0.01085
19	a5	−0.000094	0.008077	0.012286	−0.0172
20	c5	−0.003878	0.016054	0.019385	−0.0296
21	d5	0.000206	0.004036	0.007888	−0.00736
22	e5	0.000139	0.002689	0.005776	−0.00354
23	f5	0.002137	0.005822	0.016140	−0.00345
24	x4	0.004657	0.018356	0.046432	−0.01414
25 *	y4	−0.344338	0.229325	0.117723	−0.6942
26 *	a4	0.086604	0.092222	0.196475	−0.11892
27 *	c4	0.078255	0.081800	0.176402	−0.11721
28	d4	0.006251	0.011871	0.032354	−0.01660
29	e4	−0.000380	0.004918	0.008801	−0.0053
30	f4	0.000146	0.001974	0.003355	−0.00283
31	x3	0.015860	0.077495	0.242066	−0.05096
32	y3	0.068128	0.182303	0.254483	−0.33270
33	a3	−0.055563	0.09324	0.037403	−0.3020
34	d3	0.004356	0.020517	0.031289	−0.04308
35	e3	0.004208	0.027816	0.050051	−0.03657
36	f3	0.005254	0.021493	0.031724	−0.03638
37	x2	0.014250	0.235012	0.512454	−0.32158
38	y2	−0.007559	0.040522	0.075899	−0.0645
39	d2	−0.017301	0.049358	0.066779	−0.0837
40	e2	−0.015461	0.026013	0.025235	−0.0641
41 *	fs1	0.690330	0.074645	0.776301	0.565639
42 *	fs2	1.379400	0.229495	1.640865	0.929609
43 *	fs3	0.600034	0.127999	0.831090	0.382789
44 *	fs4	0.221244	0.038097	0.288517	0.162633
45 *	fs5	0.039534	0.037993	0.099793	−0.03222
46 *	fs6	0.097322	0.027456	0.135677	0.062352
47 *	ds1	0.640149	0.142105	0.957472	0.496185
48 *	ds2	1.054428	0.292792	1.639596	0.646664
49 *	ds3	0.419111	0.177024	0.653683	0.065445
50 *	ds4	0.087686	0.002107	0.045901	0.146658
51 *	ds5	0.099686	0.070966	0.281025	−0.00686
52 *	ds6	0.033299	0.012595	0.053150	0.019065

Table 2. Statistics for the Significant Parameters

Parameter	Representation	Average	Standard Deviation	Maximum	Minimum
1*	ia1	0.091631	0.015729	0.115515	0.069620
2*	ia2	0.136312	0.037337	0.205561	0.084086
3*	ia3	0.030843	0.022379	0.046165	−0.032295
4*	ia4	0.001781	0.005213	0.013764	−0.005222
5*	ia5	0.006759	0.012284	0.028195	−0.015815
6*	ia6	0.001262	0.002065	0.003792	−0.004269
7*	m6	2.768114	0.139741	2.971715	2.395409
8*	z6	0.014041	0.006980	0.022369	0.002290
9*	y4	−0.382190	0.025796	−0.315692	−0.423311
10*	a4	0.079781	0.022904	0.118200	0.036028
11*	c4	0.077761	0.021450	0.118039	0.033405
12*	fs1	0.787033	0.048752	0.861973	0.703532
13*	fs2	1.389280	0.221132	1.711836	1.080681
14*	fs3	0.650706	0.045556	0.751500	0.593681
15*	fs4	0.256854	0.031520	0.301434	0.199465
16*	fs5	0.036607	0.041190	0.097840	−0.03550
17*	fs6	0.106594	0.020605	0.140304	0.077563
18*	ds1	0.575662	0.057868	0.685223	0.465081
19*	ds2	0.944670	0.199482	1.362454	0.558198
20*	ds3	0.417502	0.103817	0.589464	0.292471
21*	ds4	0.066791	0.033318	0.113687	0.019180
22*	ds5	0.101721	0.030324	0.155355	0.044809
23*	ds6	0.030363	0.011697	0.056091	0.016647

jectories in two kinds of experiments: with joints 1, 2, and 3 each run independently, and with joints 1, 2, and 3 run together. We have recorded the actual trajectories produced by controlling the robot PUMA 260 in open-loop, using the computed torque, and obtaining the error with respect to the nominal trajectories. We have experimented with several trajectories, all of them giving very good results, at low and high speed. In addition, we could totally compensate for the gravitational torque, eliminating the effect of gravity on the robot.

As might be expected, in all the experiments, the errors were smaller at high speed rather than at low speed because the modeled system is essentially a double integrator. In the case of fast motions (about 180°/s), the error in a trajectory of 90° is $< 10\%$ (Figs. 6 and 7).

We also realized that the gravitational force builds up errors of large magnitude as the robot moves at low velocity. To correct this, we have compensated for the gravitational torque as a function of the joints' actual positions, thus introducing an elementary form of feedback. We can see in Figure 8 that the errors were considerably reduced.

Finally, in Figures 7 and 9, it can be verified that the model works well when three different joints move together, proving that the coupling inertias, as well as the centrifugal and Coriolis terms, are well compensated for.

8. Conclusion

The method developed in this article allows for the real-time computation of the dynamic equation of robot manipulators. This has been achieved by dividing the computation into background and synchronous tasks. This separation was motivated by the inherent properties of the dynamics of an industrial manipulator, which can be experimentally observed. In addition, in the background computation, we could separate the inertial and gravitational part from the velocity torques, thus suggesting the convenient allocation of two parallel tasks.

The reduction of parameters from 78 to 52 independent parameters and from 52 to 23 significant parameters make this even more significant. In effect, the computational load required to obtain an accurate estimate of the inertial and gravitational terms amounts to only 100 multiplications and 70 additions per update. The computation of the velocity torques estimates requires a similar computational load. When the manipulator carries an arbitrary load at the end effector, the computation for the inertial and gravitational terms becomes 200 multiplications and 150 additions; approximately the same number of operations are needed for the velocity torques.

The code required to implement this model in a digital computer is entirely computer generated, requiring no

Table 3. Comparison of the Results Obtained for the Frictions

Parameter Number	Representation	Value Found in J. Lloyd	Value Found by Our Method
1*	fs1	0.760	0.7870
2*	fs2	1.620	1.3892
3*	fs3	0.850	0.6507
4*	fs4	0.175	0.2568
5*	fs5	0.178	0.0366
6*	fs6	0.140	0.1065
7*	ds1	0.71	0.5756
8*	ds2	0.55	0.9446
9*	ds3	0.50	0.4175
10*	ds4	0.05	0.066
11*	ds5	0.030	0.101
12*	ds6	0.065	0.030

manual intervention save for performing a campaign of measurements.

We have also experimentally evaluated the computational model by measuring the errors produced by controlling the robot in open-loop, feeding the robot actuators with currents corresponding to the calculated torques in order to follow predetermined trajectories. The results are very good for slow and fast motions, proving the validity of the model.

Several points could be taken into account in order to improve the results. First, we considered the friction constant, which is, of course, not true in a real manipulator. This might explain most of the discrepancies of our identification results. However, the range of variation is small, making the approximation reasonable, obtaining at the same time a very good fit.

As a conclusion we feel that the identification procedure outlined in this article is robust and easily applicable to real-time control. It provides an accurate model at a low computational cost available on conventional low-cost computing hardware. In addition, this computational structure should become even more applicable to new robot manipulator designs, as the research is focused on obtaining manipulators with very small coupled inertias.

Acknowledgments

This material is based on work supported by the National Science Foundation under grant no. DMC-8411879. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation. Support through a grant from the Natural Sciences and Engineering Research Council of Canada (NSERC) is also gratefully acknowledged. The authors also wish to thank Ajit Nilakantan for his help with the experiments.

Appendix A. Generated Code Example

This appendix shows an example of the input specification file, as well as the resulting output "C" code after experimental data processing, parameter identification, and model reduction. The code computes the inertial and gravitational parameters for the PUMA 260 without load. Additional examples are included in Izaguirre et al. (1987).

```
number_links 6
mass 0 0 0 0 0 m6   % initial link masses
sigma 0 0 0 0 0 0   % joint types: all revolute
alpha 90 0 -90 90 -90 0   % D-H parameters: twist angles
dpar 0 0 d3 d4 0 0   % D-H parameters: distances
apar 0 a2 0 0 0 0   % D-H parameters: offsets
adyna 0 0 0 ad4 0 0   % Link inertia
bdyna 0 0 0 0 0 0   % ...
cdyna 0 0 0 cd4 0 0
ddyna 0 0 0 0 0 0
edyna 0 0 0 0 0 0
fdyna 0 0 0 0 0 0
xgrav 0 0 0 0 0 0   % Link centers of gravity × link masses
ygrav 0 0 0 y4 0 0   % ...
zgrav 0 0 0 0 0 z6
ia1 ai2 ia3 ia4 ia5 ia6   % Motor inertia
option moment
option_update moment
variable m6 2.768114   % Some initial values
variable z6 0.01401   % ...
variable y4 -0.382190
variable ad4 0.079781
variable cd4 0.077761
variable ia1 0.091631
variable ia2 0.136312
variable ia3 0.030843
variable ia4 0.001781
variable ia5 0.006759
variable ia6 0.001262
constant a2 0.20320
constant d3 0.12624
constant d4 0.20320
stop
```

The generated "C" code is listed below.

```
#define M6 2.768114
#define Z6 0.01401
#define Y4 -0.38219
#define AD4 0.079781
#define CD4 0.077761
#define IA1 0.091631
#define IA2 0.136312
#define IA3 0.030843
#define IA4 0.001781
#define IA5 0.006759
#define IA6 0.001262
#define MC6 M6
#define MC5 MC6
#define MC4 MC5
#define MC3 MC4
#define MC2 MC3
#define MC1 MC2
#define A2 0.2032
#define D3 0.12624
```

```
#define D4 0.2032
#define KP21 A2
#define KP32 (- D3)
#define KP42 D4
#define FP42 KP42 * MC4
#define FP32 KP32 * MC3
#define FP21 KP21 * MC2
#define NP411 MC4 * (KP42*KP42)
#define NP433 MC4 * (KP42*KP42)
#define NP311 MC3 * (KP32*KP32)
#define NP333 MC3 * (KP32*KP32)
#define NP222 MC2 * (KP21*KP21)
#define NP233 MC2 * (KP21*KP21)

#include <math.h>
dyn_robot(Q,DIJ,DI)
float Q[7],DIJ[7][7],DI[7];
{
float GRAV=9.81;
float S1,S2,S3,S4,S5,S6;
float C1,C2,C3,C4,C5,C6;
float C23,S23;
float C4C4,S4S4,S4C4,C3C3,S3S3,S3C3,C2C2,S2C2;
float S2S2,C1C1,S1C1,S1S1;
float T1214,T1224,T1411,T1414,T1424,T1421;
float BS21,BS23,BS22,BA31,BS33,BS31,BS32,BA42;
float BS43,BS42,BS41,BA52,BS52,BS51,BS63;
float JS222,JS223,JS233,JA322,JA333,JS311,JS312,JS322;
float JS313,JS323,JS333,JA411,JA433,JS411,JS412,JS413;
float JS422,JS423,JA511,JA533;
float PS031,PS041,PS042,PS131,PS141,PS142,PS241;
S1 = sin(Q[1]);
C1 = cos(Q[1]);
S2 = sin(Q[2]);
C2 = cos(Q[2]);
S3 = sin(Q[3]);
C3 = cos(Q[3]);
S4 = sin(Q[4]);
C4 = cos(Q[4]);
S5 = sin(Q[5]);
C5 = cos(Q[5]);
S6 = sin(Q[6]);
C6 = cos(Q[6]);
C23 = cos(Q[2]+Q[3]);
S23 = sin(Q[2]+Q[3]);
C4C4 = C4 * C4;
S4S4 = S4 * S4;
S4C4 = S4 * C4;
C3C3 = C3 * C3;
S3S3 = S3 * S3;
S3C3 = S3 * C3;
C2C2 = C2 * C2;
S2C2 = S2 * C2;
S2S2 = S2 * S2;
C1C1 = C1 * C1;
S1C1 = S1 * C1;
S1S1 = S1 * S1;

T1214 = (C2 * A2);
T1224 = (S2 * A2);
T1411 = ((C23 * C4));
T1414 = (- (S23 * D4) + T1214);
T1424 = ((C23 * D4) + T1224);
T1421 = ((S23 * C4));

BS63 = (Z6);
BS51 = (- (S5 * BS63));
BS52 = ((C5 * BS63));
BA52 = ((Y4 + BS52) + FP42);
BS41 = ((C4 * BS51));
BS42 = ((S4 * BS51));
BS43 = (BA52);
BA42 = (BS42 + FP32);
BS32 = ((S3 * BS41) + (C3 * BS43));
BS31 = ((C3 * BS41) - (S3 * BS43));
BS33 = (- BA42);
BA31 = (BS31 + FP21);
BS22 = ((S2 * BA31) + (C2 * BS32));
BS23 = (BS33);
BS21 = ((C2 * BA31) - (S2 * BS32));

JA533 = (CD4 - NP433);
JA511 = (AD4 - NP411);
JS423 = (0 - (D4 * BS42));
JS422 = ((S4S4 * JA511) + (C4C4 * JA533) + (2.0 * (D4 * BS43)));
JS413 = (0 - (D4 * BS41));
JS412 = ((S4C4 * (JA511 - JA533)));
JS411 = ((C4C4 * JA511) + (S4S4 * JA533) + (2.0 * (D4 * BS43)));
JA433 = (- NP333);
JA411 = (JS411 - NP311);
JS333 = (JS422);
JS323 = (- (S3 * JS412) - (D3 * BS32));
JS313 = (- (C3 * JS412) - (D3 * BS31));
JS322 = ((S3S3 * JA411) + (C3C3 * JA433) + (2.0 * (D3 * BS33)));
JS312 = ((S3C3 * (JA411 - JA433)));
JS311 = ((C3C3 * JA411) + (S3S3 * JA433) + (2.0 * (D3 * BS33)));
JA333 = (JS333 - NP233);
JA322 = (JS322 - NP222);
JS233 = (JA333 + (2.0 * ((T1224 * BS22) + (T1214 * BS21))));
JS223 = ((S2 * JS313) + (C2 * JS323) - (T1224 * BS23));
JS222 = ((S2S2 * JS311) + ((2 * S2C2) * JS312) + (C2C2 * JA322) +
(2.0 * (T1214 * BS21)));

PS031 = ((C23 * D3));
PS041 = ((T1411 * D3) + (S4 * T1414));
PS042 = (- (S23 * D3));
PS131 = (- (C23 * T1224) + (S23 * T1214));
PS141 = (- (T1411 * T1424) + (T1421 * T1414));
PS142 = ((S23 * T1424) + (C23 * T1414));
PS241 = (- (C4 * D4));

DIJ[1][1] = ((JS222) + IA1);
DIJ[1][2] = (JS223);
DIJ[1][3] = ((S2 * JS313) + (C2 * JS323));
DIJ[1][4] = ((S23 * JS413) - (BS42 * PS031) + (BS41 * T1214));
DIJ[1][5] = (- (BS52 * PS041) + (BS51 * PS042));
DIJ[1][6] = (0);
DIJ[2][2] = ((JS233) + IA2);
DIJ[2][3] = (JS422 + (BS31 * A2));
DIJ[2][4] = (- JS423 - (BS42 * PS131));
DIJ[2][5] = (- (BS52 * PS141) + (BS51 * PS142));
DIJ[2][6] = (0);
DIJ[3][3] = ((JS422) + IA3);
DIJ[3][4] = (- JS423);
DIJ[3][5] = (- (BS52 * PS241));
DIJ[3][6] = (0);
DIJ[4][4] = IA4;
DIJ[4][5] = (0);
DIJ[4][6] = (0);
DIJ[5][5] = IA5;
DIJ[5][6] = (0);
DIJ[6][6] = IA6;
DI[1] = (0);
DI[2] = (GRAV * (BS21));
DI[3] = (GRAV * (- (BS32 * S2) + (BS31 * C2)));
DI[4] = (GRAV * (- (BS42 * S23)));
DI[5] = (GRAV * (- (BS52 * T1421) + (BS51 * C23)));
DI[6] = 0;
}
```

Appendix B. Experimental Results

We show here the results of the model fit, as well as trajectory following in open-loop control. Figure 1 contains one of the several trajectories used to identify the dynamic coefficients. Figures 2 and 3 shows the velocities and accelerations, respectively, estimated using central differences of positions and velocities, respectively.

Figure 4 shows, for the same trajectory, the measured and the estimated torque obtained by the model using 52 linearly independent parameters. The fit is very accurate for all joints. Figure 5 shows, for the same trajectory, the measured and the estimated torque obtained by the model using only 23 most significant parameters. The fit is still very accurate for the first three joints, showing small differences for the last three ones.

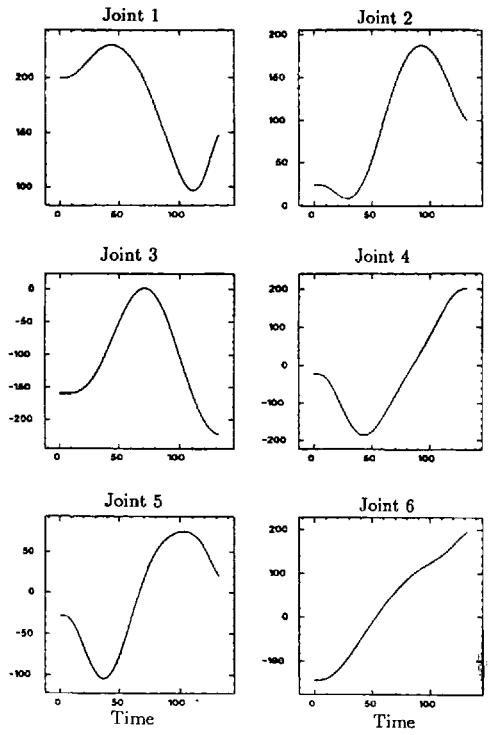

Fig. 1. Joint positions in degrees.

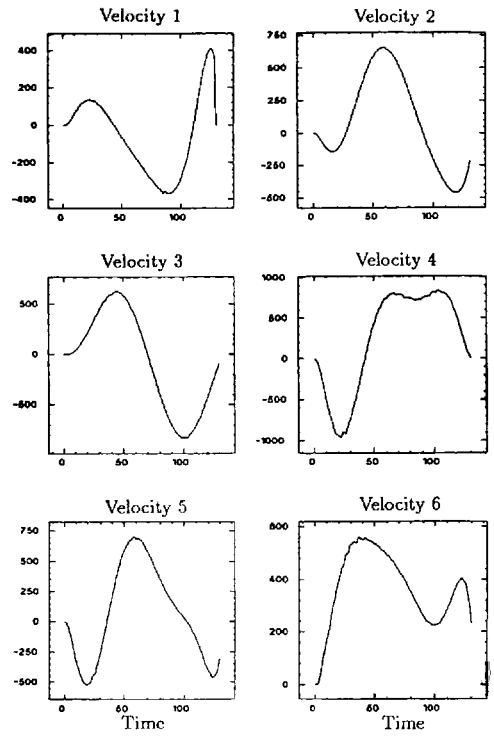

Fig. 3. Accelerations using finite differences.

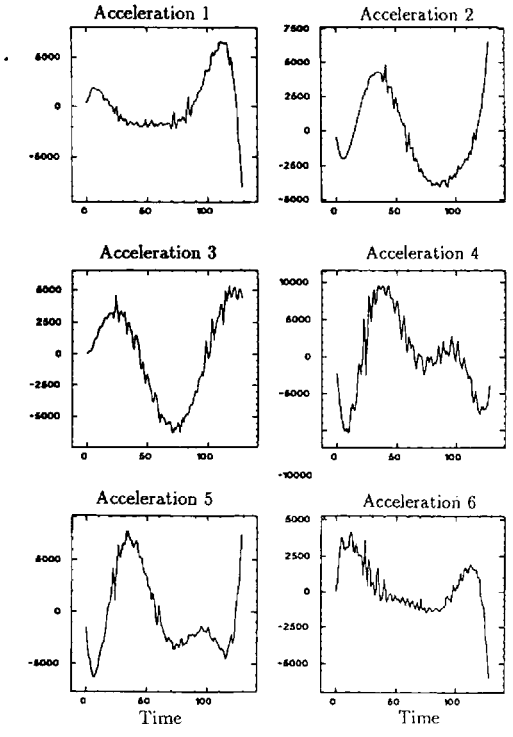

Fig. 2. Velocities using finite differences.

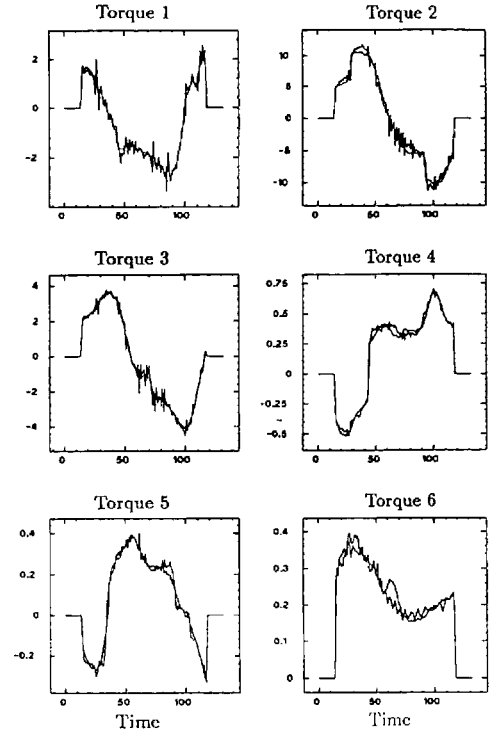

Fig. 4. Fitting of the torques for the same trajectory using 52 parameters.

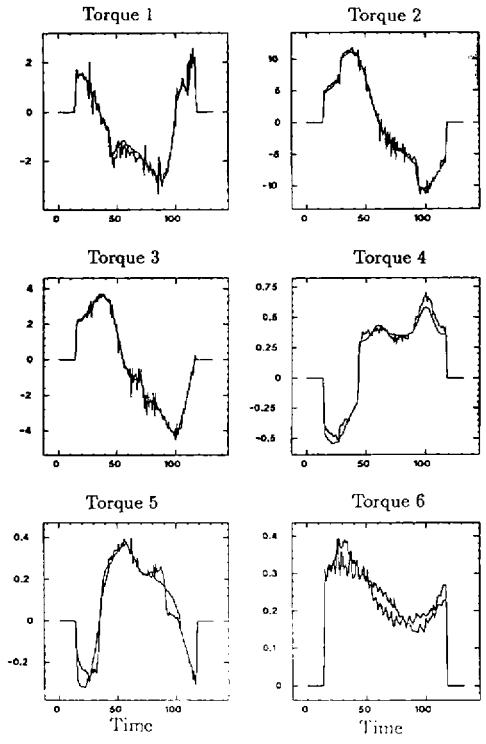

Fig. 5. Fitting of the torques for the same trajectory using 23 parameters.

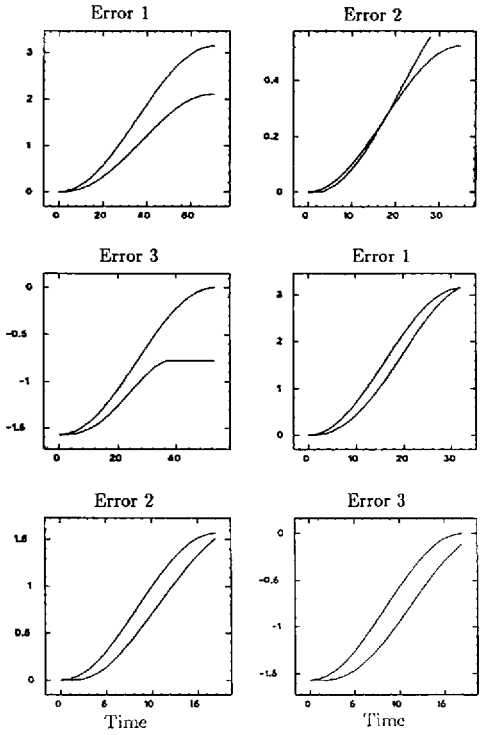

Fig. 6. Errors in trajectory following. First three frames: *Independent joints in slow motion.* Last three frames: *Independent joints in fast motion.*

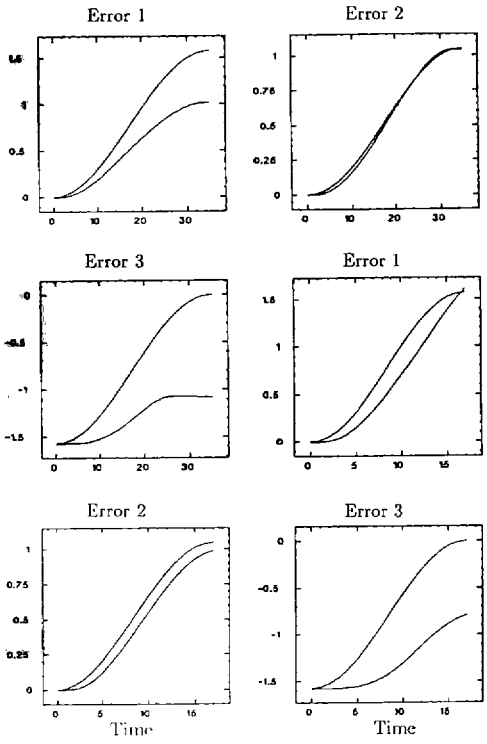

Fig. 7. Errors in trajectory following. First three frames: *First three joints together in slow motion.* Last three frames: *First three joints together in fast motion.*

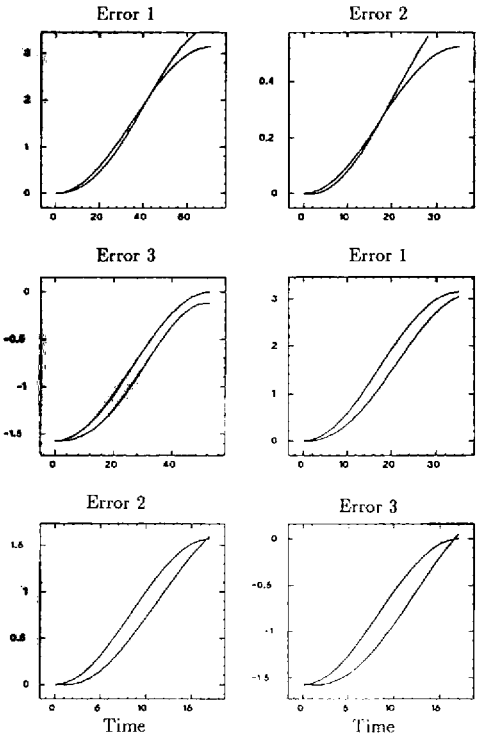

Fig. 8. Errors in trajectory following (same as Figure 6, but with on-line gravity compensation).

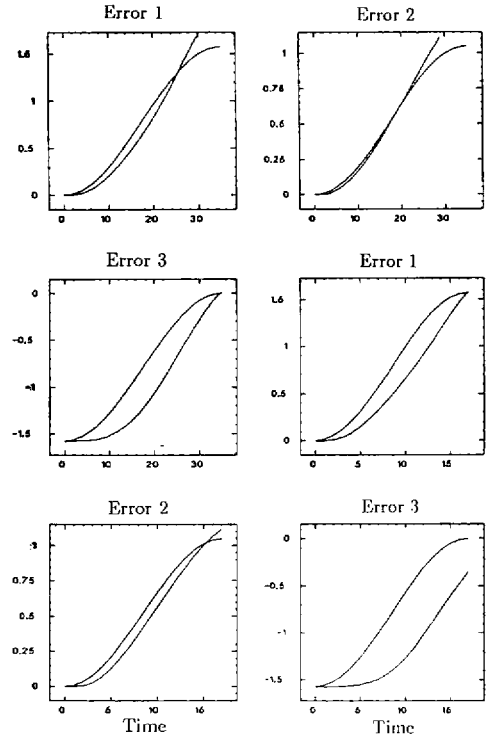

Fig. 9. Errors in trajectory following (same as Figure 7, but with on-line gravity compensation).

Finally, Figures 6 and 7 contain the errors in trajectory following in open-loop control. Figure 6 shows the trajectories for individual joint motions of the first three joints. Figure 7 shows the trajectories for the motion of all three joints moving together. Figures 8 and 9 are similar to Figures 6 and 7 after on-line gravitational compensation.

References

An, C. H., Atkeson, G. C., and Hollerbach, J. M. 1985 (Fort Lauderdale, December). Estimation of inertial parameters of rigid body links of manipulators. *Proc. of the 24th Conf. on Dec. and Control*, pp. 990–1002.

Armstrong, B., Khatib, O., and Burdick, J. 1986 (San Francisco, April). The explicit dynamic model and inertial parameters of the PUMA 560 Arm. *Proc. Conf. IEEE Robotics and Automation*, pp. 510–518.

Bejcsy, A. K. 1974. Robot arm dynamic and control NASA technical memo, JPL 33-699.

Craig, J. 1986. *Introduction to Robotics Mechanics and Control.* Reading, MA: Addison-Wesley.

Featherstone, R. 1984. Robot dynamics algorithms. Ph.D. thesis, University of Edinburg.

Ferreira, E. P. 1984. Contribution a l'identification de paramètres et à la commande dynamique adaptative de robots manipulateurs. Thèse de Doct. Ingénieur, Université Paul Sabatier, Toulouse, France.

Hollerbach, J. 1980. A recursive Lagrangian formulation and a comparative study of dynamics formulations. *IEEE Trans. Sys. Man Cybernet.* SMC-10(11):730–736.

Hollerbach, J. 1983. Dynamics. In Brady, M., Hollerbach, J. M., Johnson, T. L., Lozano-Pérez, T., and Mason, M. T. (eds.): *Robot Motion, Planning and Control.* Cambridge, MA: MIT Press.

Izaguirre, A., and Paul, R. P. (St. Louis, March) 1985. Computation of the inertia and gravitational coefficients of the dynamic equations for a robot manipulator with a load. *Proc. IEEE Conference on Robotics and Automation*, pp. 1024–1031.

Izaguirre, A., and Paul, R. P. (San Francisco, April) 1985. Automatic generation of the dynamic equations of the robot manipulators using a LISP program. *Proc. IEEE Conference on Robotics and Automation*, pp. 220–226.

Izaguirre, A., Paul, R. P., Hashimoto, M., and Hayward, V. 1987. A new computational structure for real time dynamics. Grasp Laboratory report, University of Pennsylvania.

Khan, M. E., and Roth B. 1971. The near minimum time control of open loop articulated kinematic chains. *Trans. of ASME. J. of Dyn. Sys. Engineering Control*, pp. 164–172.

Khalil, W., Kleinfinger, J. K., and Gautier, M. (San Francisco, April) 1986. Reducing the computational burden of the dynamic model robots. *Proc. IEEE Conference on Robotics and Automation*, pp. 525–530.

Kircanski, N., Kircanski, M., Timcenko O., and Vukobratovic, M. (Krakow) 1986. An approach to the development of real time robot models. *Proc. ROMANCY, IFToMM Symposium.*

Khosla, P., and Kanade, T. (San Francisco, April) 1986. An algorithm to determine the identifiable parameters in dynamic robot models. Presented at the *IEEE Conf. on Robotics and Automation.*

Lathrop, R. (St. Louis, March) 1985. Parallelism in manipulator dynamics. *Proc. IEEE Conf. Robotics and Automation*, pp. 772–778.

Likins, P. 1971. Passive and semi-active attitude stabilizations-flexible spacecraft. *ARGARD-LS*, pp. 45–71.

Lin, C., and Luh, J. Y. S. 1982. Scheduling of parallel computation for a computer-controlled mechanical manipulator. *IEEE Trans. Sys. Man Cybernet.* SMC-12(2):214–234.

Lloyd, J. E. 1986. Implementation of a robot programming environment. Master's thesis, Dept. of Electrical Engineering, McGill University.

Luh, J. Y. S., Paul, R. P., and Walker, M. 1980. On-line computational scheme for mechanical manipulators. *IEEE Trans. Auto. Control* 25(3):468–474.

Megahed, S. M. 1984. Contribution a la modélisation géometrique et dynamique des robots manipula-

teurs à structure de chaine cinématique simple ou complexe. Thèse d'État, Université Paul Sabatier, Toulouse, France.

Orin, D. E., Chao, H. H., Olson, K. W., and Schrader, W. W. (St. Louis, March) 1985. Pipeline/parallel algorithms for the Jacobian and inverse dynamics computations. *Proc. IEEE Conf. Robotics and Automation*, pp. 785–789.

Oslen, H. B., and Bekey, G. (San Francisco, April) 1986. Identification of robot dynamics. *Proc. IEEE Conf. Robotics and Automation*, pp. 1004–1010.

Paul, R. P. (November) 1972. Modeling, trajectory calculation and servoing of a computer controlled arm. AIM 177, Stanford Artificial Intelligence Laboratory, Stanford University.

Paul, R. P. 1981. Robot manipulators: Mathematics, programming, and control. Cambridge, MA: MIT Press.

Raibert, M. (New Orleans, December) 1977. Analytical equations vs. look-up table for manipulation: a unifying concept. *Proc. IEEE Conf. on Decision and Control.*

Renaud, M. 1984. An efficient iterative analytical procedure for obtaining a robot manipulator dynamic model. In Brady, M., and Paul, R. P. (eds.): *Robotic Research: The First International Symposium.* Cambridge, MA: MIT Press, pp. 749–764.

Uicker, J. 1968. Dynamic behavior of spatial linkages. *Trans. ASME Mech.* 5(68):1–15.