Implementation of Adaptive Control Algorithms in Robot Manipulators Using Parallel Computing*

Juan C. Fernández[1], Vicente Hernández[2], and Lourdes Peñalver[3]

[1] Dept. de Ingeniería y Ciencia de los Computadores, Universidad Jaume I, 12071-Castellón (Spain), Phone: +34-964-728265; Fax: +34-964-728435, jfernand@icc.uji.es

[2] Dept. de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, 46071-Valencia (Spain), Tel: +34 96 3877356, Fax: +34 963877359, vhernand@dsic.upv.es

[3] Dept. de Informática de Sistemas y Computadores, Universidad Politécnica de Valencia, 46071-Valencia (Spain), Phone: +34-96-3877572; Fax: +34-96-3877579, lourdes@disca.upv.es

Abstract. The dynamics equation of robot manipulators is non linear and coupled. An inverse dynamic control algorithm that requires a full knowledge of the dynamics of the system is one way to solve the control movement. Adaptive control is used to identify the unknown parameters (inertial parameters, mass, etc). The adaptive control algorithms are based on the linear relationship of inertial parameters in the dynamic equation. A formulation to generalize this relationship is applied to the Johansson adaptive algorithm. The objective of this paper is to present the implementation of this relationship using parallel computing and apply it to an on-line identification problem in real-time.

1 Introduction

The dynamic equation of robot manipulator torque in open chain is determined by highly coupled and non linear differential equation systems. It is necessary to use approximate or cancelling techniques to apply some control algorithms, such as inverse dynamic control, over the full system. To apply these control techniques it is necessary to know the dynamics of the system. This knowledge allows the existing relations among the different links to be established. The links to establish the kinematic relations of the system are defined from Denavit-Hartenberg parameters. Full knowledge of the dynamics, inertial parameters, mass and inertial moments for each arm, is usually unavailable. These parameters should be estimated using least square or adaptive control techniques. Using adaptive control it is possible to solve both movement control and parameter identification problems. Some of these algorithms can be found in [1,

* This work is supported by the CICYT Project TIC2000-1151-C07-06.

H. Kosch, L. Böszörményi, H. Hellwagner (Eds.): Euro-Par 2003, LNCS 2790, pp. 491–498, 2003.

5,7,4]. One of the problems in applying this kind of algorithm is that of obtaining the relationship

$$\tau = Y_\tau(q, \dot{q}, \ddot{q})\theta, \tag{1}$$

where $Y_\tau(q, \dot{q}, \ddot{q})$, known as regressor, is an $n \times r$ matrix, n being the number of links and r the number of parameters; and θ is the $r \times 1$ parameter vector. Considering all different parameters, r will be $10n$.

The linear relationship for the adaptive Johansson algorithm using several algebraic properties and definitions given in [6] is employed. This formulation is a computable and general solution for any robot manipulator in open chain using Denavit-Hartenberg parameters and is an extension of the Lagrange-Euler formulation. The main problem of the Lagrange-Euler formulation is its high computational cost, but there are several studies, [8,2], where this is reduced. In this paper the parallel algorithm to obtain the linear relationship for the adaptive Johansson algorithm using the Lagrange-Euler formulation is presented.

The structure of the paper is the following: In section two the dynamic model is presented. Section three describes the Johansson adaptive control algorithm. Section four presents the parallel algorithm to obtain the linear relationship for the adaptive Johansson algorithm. The results for a Puma robot are described in section five. And finally the conclusions of this paper are presented in the last section.

2 The Dynamic Model

The dynamic equation of rigid manipulators with n arms in matrix form is

$$\tau = D(q)\ddot{q} + h(q, \dot{q}) + c(q), \tag{2}$$

where τ is the $n \times 1$ vector of nominal driving torques, q is the $n \times 1$ vector of nominal generalized coordinates, $\dot{q}$ and $\ddot{q}$ are the $n \times 1$ vectors of the first and second derivatives of the vector q respectively, D is the inertia matrix, $h(q, \dot{q})$ is the vector of centrifugal and Coriolis forces and $c(q)$ is the vector of gravitational forces.

3 Johansson Adaptive Control Algorithm

The desired reference trajectory followed by the manipulator is assumed to be available as bounded functions of time in terms of joint accelerations $\ddot{q}_r$, angular velocities $\dot{q}_r$, and angular positions q_r. A stable non linear reference model is also possible if the errors of accelerations, velocities and positions are defined as

$$\begin{bmatrix} \ddot{e} \\ \dot{e} \\ e \end{bmatrix} = \begin{bmatrix} \ddot{q} - \ddot{q}_r \\ \dot{q} - \dot{q}_r \\ q - q_r \end{bmatrix}. \tag{3}$$

The control objective is to follow a given bounded reference trajectory q_r, with no position errors e or velocity errors $\dot{e}$. Let $P_{qq}, \Omega, S \in R^{n \times n}$ and $P_{\theta\theta} \in R^{r \times r}$

be positive definite matrices and define $P_{12} = P_{qq}^{-1}\Omega$. Let $Y_J \in R^{n\times r}$ and $Y_{J_0} \in R^{n\times 1}$ be defined from the relation

$$\begin{aligned} &Y_J(q_r, q, \dot{q}_r, \dot{q}, \ddot{q}_r)\theta + Y_{J_0}(q_r, q, \dot{q}_r, \dot{q}, \ddot{q}_r) \\ &= -\frac{1}{2}\dot{D}(q)(\dot{e} + P_{12}e) + D(q)(\ddot{q}_r - P_{12}\dot{e}) + h(q, \dot{q})\dot{q} + c(q). \end{aligned} \tag{4}$$

For any choice of $P_{qq} = P_{qq}^T > 0$, $P_{\theta\theta} = P_{\theta\theta}^T > 0$, $\Omega = \Omega^T > 0$, $S = S^T > 0$, the adaptive control law is given by

$$\dot{\hat{\theta}}(q_r, q, \dot{q}_r, \dot{q}, \ddot{q}_r) = -P_{\theta\theta}^{-1} Y_J^T(\dot{e} + P_{12}e), \tag{5}$$

$$\tau(q_r, q, \dot{q}_r, \dot{q}, \ddot{q}_r, \hat{\theta}) = Y_J\hat{\theta} + Y_{J_0} - (S + P_{qq}\Omega^{-1}P_{qq})(\dot{e} + P_{12}e) + P_{qq}e. \tag{6}$$

3.1 Reformulation

Using several algebraic properties and definitions given in [6] it is possible to obtain a computable version of $Y_J(q_r, q, \dot{q}_r, \dot{q}, \ddot{q}_r)$. Considering $u = \dot{e} + P_{12}e$, the expression for the derivative of the inertial matrix is given by

$$\dot{D}(q)u = Y_{\dot{D}u}(q)\theta_{\dot{D}}, \tag{7}$$

where

$$Y_{\dot{D}u}(q) = \begin{bmatrix} rtr\,(B_{\dot{D}u11}) & rtr\,(B_{\dot{D}u12}) & \cdots & rtr\,(B_{\dot{D}u1n}) \\ 0 & rtr\,(B_{\dot{D}u22}) & \ddots & rtr\,(B_{\dot{D}u2n}) \\ 0 & 0 & \ddots & \vdots \\ 0 & 0 & 0 & rtr\,(B_{\dot{D}unn}) \end{bmatrix}, \tag{8}$$

$$\theta_{\dot{D}} = \left[\nu(J_1)\ \nu(J_2) \cdots \nu(J_n)\right]^T, \tag{9}$$

with

$$B_{\dot{D}uij} = \sum_{k=1}^{j}\sum_{l=1}^{j} (U_{ij} \otimes U_{lkj} + U_{lij} \otimes U_{kj})\,\dot{q}_l\,u_k. \tag{10}$$

where J_k is the inertia tensor related to link k, U_{jk} is the effect of the movement of link k on all the points of link j and U_{kjl} is the effect of the movement of links j and l on all the points of link k. The operator rtr of a matrix is a row vector whose components are the traces of the columns in this matrix, and ν is the operator vector-column of an $m \times n$ matrix where the first m components of ν are the components of the first column of the matrix, the second m components of ν are the components of the second column of the matrix, and so on, [6].

Considering $v = \ddot{q}_r - P_{12}\dot{e}$, the expression for the inertial matrix is given by

$$D(q)v = Y_{Dv}(q)\theta_D, \tag{11}$$

where

$$Y_{Dv}(q) = \begin{bmatrix} rtr\,(B_{Dv11}) & rtr\,(B_{Dv12}) & \cdots & rtr\,(B_{Dv1n}) \\ 0 & rtr\,(B_{Dv22}) & \ddots & rtr\,(B_{Dv2n}) \\ 0 & 0 & \ddots & \vdots \\ 0 & 0 & 0 & rtr\,(B_{Dvnn}) \end{bmatrix}, \theta_D = \theta_{\dot{D}}, \tag{12}$$

with

$$B_{Dvik} = \sum_{j=1}^{k} (U_{ik} \otimes U_{jk}) v_j. \tag{13}$$

The expression for the centrifugal and Coriolis forces is given by

$$h(q, \dot{q}) = Y_h(q, \dot{q}) \theta_h$$

where

$$Y_h(q, \dot{q}) = \begin{bmatrix} rtr\,(B_{h11}) & rtr\,(B_{h12}) & \cdots & rtr\,(B_{h1n}) \\ 0 & tr\,(B_{h22}) & \ddots & rtr\,(B_{h2n}) \\ 0 & 0 & \ddots & \vdots \\ 0 & 0 & 0 & rtr\,(B_{hnn}) \end{bmatrix}, \theta_h = \theta_{\dot{D}}, \tag{14}$$

with

$$B_{hij} = \sum_{k=1}^{j} \sum_{l=1}^{j} (U_{ij} \otimes U_{lkj}) \dot{q}_k \dot{q}_l. \tag{15}$$

The vector of gravitational forces can be expressed as a linear relationship with the inertial parameters

$$c(q) = Y_c(q) \theta_c, \tag{16}$$

where

$$Y_c(q) = -\begin{bmatrix} Y_{c11} & Y_{c12} & \cdots & Y_{c1n} \\ 0 & Y_{c22} & \cdots & Y_{c2n} \\ 0 & 0 & \ddots & \vdots \\ 0 & 0 & 0 & Y_{cnn} \end{bmatrix} = -\begin{bmatrix} g^T U_{11} & g^T U_{12} & \cdots & g^T U_{1n} \\ 0 & g^T U_{22} & \cdots & g^T U_{2n} \\ 0 & 0 & \ddots & \vdots \\ 0 & 0 & 0 & g^T U_{nn} \end{bmatrix}, \tag{17}$$

$$\theta_c = \begin{bmatrix} m_1 \bar{r}_1 & m_2 \bar{r}_2 & \cdots & m_n \bar{r}_n \end{bmatrix}^T, \tag{18}$$

where $\bar{r}_i$ is the position of the centre of mass of link i (m_i) with respect to the origin of coordinates of link i.

From the previous results, the computable version of Y_J is given by

$$Y_J = -\frac{1}{2} Y_{\dot{D}u} + Y_{Dv} + Y_h + Y_c. \tag{19}$$

The next section describes the parallel algorithm to obtain expressions (5), (6) and (19).

4 Parallel Algorithm

The parallel algorithm to obtain (19) is presented below, where n is the number of links and p is the number of processors. To determine the calculations of each processor, two parameters have been defined, i_k, the initial link, and f_k the final link of processor P_k. Then P_k computes the operations of the links $i_k, i_{k+1}, \cdots, f_k$. In this case:

- Processor P_1 has the values $i_1 = 1$ and $f_1 = n-p+1$ to obtain the matrices and vectors involved in the Johansson parallel algorithm.
- Processors P_k, $k = 2 : p$, have the values $i_k = f_k = n - p + k$ to obtain the matrices and vectors involved in the Johansson parallel algorithm.

To obtain (19) matrices $Y_{\dot{D}u}$, Y_{Dv}, Y_h and Y_c can be computed. These matrices are calculated using matrices U_{ij} (structure U Eq. (20)) and matrices U_{ijk} (structure DU Eq. (22)). Matrices U_{ij} mean the effect of movement of link j on all the points of link i, and matrices U_{ijk} mean the effect of movement of links j and k on all the points of link i. The structure U is given by

$$U = \begin{bmatrix} U_{11} & U_{12} & \cdots & U_{1n} \\ & U_{22} & \cdots & U_{2n} \\ & & \ddots & \vdots \\ & & & U_{nn} \end{bmatrix} = \begin{bmatrix} Q_1\ {}^0A_1 & Q_1\ {}^0A_2 & \cdots & Q_1\ {}^0A_n \\ & {}^0A_1 Q_2\ {}^1A_2 & \cdots & {}^0A_1 Q_2\ {}^1A_n \\ & & \ddots & \vdots \\ & & & {}^0A_{n-1} Q_n\ {}^{n-1}A_n \end{bmatrix}, \quad (20)$$

where Q_i is the constant matrix that allows us to calculate the partial derivative of ${}^iA_j$, the transformation matrix of a robot manipulator. To obtain U, matrices ${}^iA_j$ (structure A Eq. (21)) are necessary. This structure is given by

$$A = \begin{bmatrix} {}^0A_1\ {}^0A_2 & \cdots & {}^0A_n \\ & {}^1A_2 & \cdots & {}^1A_n \\ & & \ddots & \vdots \\ & & & {}^{n-1}A_n \end{bmatrix}, \quad (21)$$

where ${}^iA_j = {}^iA_{j-1}\ {}^{j-1}A_j$, $i = 0 : n-2$. The matrices of the diagonal, ${}^{i-1}A_i$, $i = 1 : n$, are obtained from the robot parameters [3]. In the parallel algorithm P_k computes ${}^iA_j$, $i = 0 : f_k - 1$, $j = i+1 : f_k$. This is the unique case where the values of parameters i_k and f_k are different from the parameters defined previously, in this case:

- Processor P_1 has the values $i_1 = 1$ and $f_1 = n-p+1$ to obtain the matrices of A.
- Processors P_k, $k = 2 : p$, have the values $i_k = 1$ and $f_k = n-p+k$ to obtain the matrices of A.

Processor P_k needs ${}^0A_i$, $i = i_k : f_k$, to obtain matrices U_{1i}, $i = i_k : f_k$. To obtain the remaining matrices of U, U_{ij}, $j = i_k : f_k$, $i = 2 : j$, each processor needs ${}^0A_{i-1}$ and ${}^{i-1}A_j$, $j = i_k : f_k$, $i = 2 : j$. All these matrices have been computed previously.

To obtain matrices U_{ijk} the following structure, DU, is defined

$$DU = \left[DU_1 \; DU_2 \cdots DU_n \right]^T. \tag{22}$$

As $U_{kjl} = U_{jkl}$, it is only necessary to compute the following block of DU_i, $i = 1 : n$

$$DU_i(i:n, i:n) = \begin{bmatrix} U_{iii} & U_{iii+1} & \cdots & U_{iin} \\ 0 & U_{ii+1i+1} & \cdots & U_{ii+1n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & U_{inn} \end{bmatrix}. \tag{23}$$

Given that P_k, $k = 1 : p$, has the required U matrices, it computes $U_{1ij} = Q_1 U_{ij}$, $j = i_k : f_k$, $i = 1 : j$.

To obtain the remaining matrices of DU, P_k, $k = 1 : p$, computes U_{ijk}, $k = i_k : f_k$, $i = 2 : k$ and $j = i : k$.

There are two situations:

- To obtain the matrices of row i, $U_{iij} = V_i Q_i \, {}^{i-1}A_j$, processor P_k needs V_i, $i = 2 : f_k$, and ${}^{i-1}A_j$, $j = i_k : f_k$, $i = 2 : j$. These matrices have been computed previously.
- To obtain the matrices of row $l > i$, $U_{ilj} = U_{il-1} Q_l \, {}^{l-1}A_j$, ${}^{l-1}A_j$ is first computed by P_k. But U_{il-1} has been computed in another processor. In order to avoid communications P_k replicates the computation of this matrix.

Then, each processor calculates the block of columns of the structures U and DU corresponding to rank $[i_k : f_k]$. P_k, $k = 1 : p$, computes $rtr(B_{\dot{D}uij})$, $rtr(B_{Dvij})$, $rtr(B_{hij})$ and Y_{cij}, $j = i_k : f_k$ and $i = 1 : j$.

As the processors have all the information to compute $B_{\dot{D}}$, B_D and B_h, no communication among them is necessary. To obtain matrix Y_c no communication is necessary because each processor has the required U matrices. With this information P_k computes expression (19).

To obtain (5), each processor P_k computes the following expression

$$\dot{\hat{\theta}}_i = \sum_{j=1}^{i} -P_{\theta\theta ii}^{-1} Y j_{ji}^T u_j, \tag{24}$$

for $i = i_k : f_k$.

And finally, the control law $\tau = Y_J \hat{\theta} - (S + P_{qq}\Omega^{-1}P_{qq})u + P_{qq}e$ must be computed. $P_{\theta\theta}$ and Ω can be considered as diagonal matrices. Each processor P_k computes τ_i, $i = 1 : f_k$, using the matrices $Y j_{ij}$ and $\hat{\theta}_i$ that it has calculated.

Each processor sends the computed vector τ_i to processor P_p. This processor receives these values and it obtains the final value of τ. This is the only communication in the algorithm. The term $(S + P_{qq}\Omega^{-1}P_{qq})u + P_{qq}e$ is also computed in P_p.

5 Experimental Results

The sequential algorithm is evaluated using sequential execution time, T_s. The parallel algorithms are evaluated using parallel execution time T_p (p processors), Speed-up, $S_p = T_1/T_p$ and efficiency, $E_p = S_p/p$. The results have been obtained using the parameters of a Puma 600 robot with six links. In the parallel algorithms 2, 3 and 4 processors have been used. In each case the links have been distributed among the processors according to i_k and f_k parameters. To present the results the following notation is used: $pxax\cdots x$, where px is the number of processors used and $ax\cdots x$ is the number of links computed by each processor. For example, in $p2a31$, the first three links are calculated in P_1 and the fourth link is computed in P_2.

A Beowulf cluster with 32 nodes connected via Myrinet switch has been used. Each node is an Intel Pentium-II processor at 300MHz with 128 MBytes RAM. Communication routines in MPI and C language are used. Table (1) shows the results, in miliseconds, of computing the parallel algorithm of the adaptive Johansson control where $T_s = 9.14$, using this cluster.

Table 1. Experimental results with $n = 6$ links when a Beowulf cluster is used.

Algorithm	p	T_p	Speed-up	Efficiency
p2a33	2	8.36	1.085	54.67%
p2a42	2	7.1	1.286	64.31%
p2a51	2	5.446	1.678	83.94%
p3a222	3	5.5	1.65	55.33%
p3a321	3	4.86	1.88	62.69%
p3a411	3	4.86	1.88	62.69%
p4a3111	4	4.922	1.857	46.43%

The best efficiency is obtained using two processors, where the first processor computes links one to five and the second processor computes the last link. The shortest execution time is obtained using three processors, where the first processor calculates links one to four, the second processor computes the fifth link, and the last link is computed by processor three. Execution time increases when four processors are used because the load balance is not good. The objective is to reduce execution time even though efficiency is not good.

6 Conclusions

Given the generalized formulation for the linear relationship between variable dynamic and inertial terms it is possible to apply this formulation to the Johansson adaptive algorithm using the Lagrange-Euler formulation. Although the use of

the Lagrange-Euler formulation has a high computational cost, it is possible to reduce it in two ways:

- Eliminating the high quantity of null terms and exploiting the properties of the matrices.
- Using parallel computing to obtain the different matrices of the dynamic equation and the linear relationship of the Johansson adaptive algorithm.

Using these two techniques it is possible to obtain the linear relationship of the Johansson adaptive algorithm and apply it to an on-line identification because we have reduced the time requirements. The two techniques can be used in other adaptive algorithms, such as Slotine-Li, and in optimal control. The shortest execution time is obtained using three processors, and the best efficiency is obtained with two. The parallel algorithm can be used for a robot manipulator with more than two links. In this paper this formulation is applied to a six links Puma manipulator.

References

1. Craig, J.: Adaptive Control of Mechanical Manipulators, Addison-Wesley (1988).
2. Fernández, J.C: Simulación Dinámica y Control de Robots Industriales Utilizando Computación Paralela, Ph.D. Univ. Politécnica de Valencia (1999).
3. Fu, K.S., González, R.C., Lee, C.S.G: Robotics: Control, Sensing, Vision and Intelligence, New York, McGraw-Hill, 580 pages (1987).
4. Johansson, R.: Adaptive Control of Robot Manipulator Motion, IEEE Transactions on Robotics and Automation, 4(6), 483–490 (1990).
5. Ortega, J.M., Spong M.: Adaptive Motion Control of Rigid Robots: A Tutorial, Automatica 25(6), 877–888 (1989).
6. Peñalver, L.: Modelado Dinámico e Identificación Paramétrica para el Control de Robots Manipuladores, Ph.D. Univ. Politécnica de Valencia (1998).
7. Slotine, J.J., Li, W.: On Adaptive Control of Robot Manipulators, International Journal Robotics Research, 6(3), 49–59 (1987).
8. Zomaya, A.Y.: Modelling and Simulation of Robot Manipulators. A Parallel Processing Approach, World Scientific Series in Robotics and Automated Systems, 8, (1992).