# A DATA AND SOFTWARE MODEL FOR ROBOT MANIPULATOR CONTROL

**L. Peñalver * J. C. Fernández ** S. Terrasa ***
J. Tornero **** V. Hernández *****

* *Dept. de Informática de Sistemas y Computadores*
*Universidad Politécnica de Valencia, 46071-Valencia (Spain)*
*Phone: +34-96-3877572; Fax: +34-96-3877579*
*e-mail: {lourdes,sterrasa}@disca.upv.es*
** *Departamento de Informática*
*Universidad Jaume I, 12071-Castellón (Spain)*
*Phone: +34-964-728265; Fax: +34-964-728435*
*e-mail: jfernand@inf.uji.es*
*** *Dept. de Sistemas Informáticos y Computación*
*Universidad Politécnica de Valencia, 46071-Valencia (Spain)*
*Phone: +34-96-3877356; Fax: +34-96-3877359*
*e-mail: vhernand@dsic.upv.es*
**** *Dept. de Ingeniería de Sistemas y Automática*
*Universidad Politécnica de Valencia, 46071-Valencia (Spain)*
*Phone: +34-96-3879578; Fax: +34-96-3879579*
*e-mail: jtornero@isa.upv.es*

Abstract: This paper considers software design, the choice of algorithms and implementation characteristics in a robot system to solve the dynamic model and evaluate and develop tools to implement control algorithms. OMT methodology has been used to design the software because it permits several levels of abstractions. Lagrange-Euler formulation has been chosen and has been reformulated to adapt it to the control algorithms. Finally in the implementation aspects the objective has been to reduce computing time using parallel algorithms and analysing the properties of the equations eliminating the null terms. This study has been facilitated by the different levels of abstraction defined. All these aspects have been used to solve the identification of the inertial parameters for a Puma robot by energy equation and adaptive control algorithms. *Copyright ©2000 IFAC*

Keywords: Objected oriented technique, Software architecture, Data structures, Robot manipulators control, Lagrange-Euler formulation.

## 1. INTRODUCTION

To evaluate the behaviour of a robot manipulator several algorithms are available in the literature. In this work a set of programs to solve the dynamic model, evaluate and develop several tools -mathematical and algorithmical- to implement some control algorithms are presented. Given the complexity of the problem three problems have been considered: Software model, algorithms and implementation. Software model works with different abstraction levels allowing the problem to be defined for each level without knowing details for the lower levels. The OMT -Object Oriented Technique- methodology has been used. From this

methodology the definitions and procedures can be translated to a programming language C++ easily. This proposal can be integrated into a more general environment known as software architecture. In the literature all architectures define several levels of specifications. But the majority of them have been developed to solve a particular problem for mobile robots, (Albus *et al.*, 1987), (Anderson, 1993). The model presented by Nilsson and Nielsen (1992), Nilsson (1993) have been used in the solution presented here.

To decide the algorithms used in the implementation, it is necessary to choose the mathematical model for the dynamics used to simulate the robot behaviour. From the various algorithms in the literature, the Lagrange-Euler formulation has been chosen because it offers the possibility of working with different level of abstractions due to its formulation and it is an open chain formulation which facilitates its use in control algorithms, i.e. adaptive control (Craig, 1988), (Spong and Vidyasagar, 1989), (Johansson, 1990). The dynamic equations have been reformulated to solve this kind of control problems (Peñalver *et al.*, 1996), (Peñalver *et al.*, 1998), (Peñalver, 1998). A new algorithm has been developed to solve these problems both mathematical and algorithmically, unlike (Ha *et al.*, 1989), (Bennis and Khalil, 1990), (Gautier *et al.*, 1990), (Khalil and Creusot, 1997), (Kawasaki and Shimizu, 1998) that only solve problems symbolically. These results have been used to obtain a reduced model of the robot's dynamics, to solve the identification parameters. The main problem of the Lagrange-Euler formulation is its high computational cost, but there are several studies, (Zomaya, 1992), (Fernández, 1999), where this is reduced. Fernández (1999) considers two approaches: Reducing the number of operations by exploiting the intrinsic characteristics -symmetry, repeat rows, null elements etc.-, and using parallel computing because the Lagrange-Euler formulation and the presented reformulation have a high degree of parallelism.

The above overview provides a general application to analyse different adaptive control algorithms over different robot manipulator objects.

The structure of the paper is the following: In the second section the software model is developed in more depth. In section three the chosen algorithms are presented. The different implementations and examples are described in the fourth section. And finally the fifth section presents the conclusions.

## 2. SOFTWARE MODEL

The software model defines different levels of abstractions that facilitate the development and implementation of the problem. Following (Nilsson, 1993), three levels have been considered: The control level related to hardware with high temporality and security restrictions. It is difficult to transfer to other platforms given their hardware dependency. This part has not been implemented but it has been considered in the definition of the data structures. The Application level involving all processes allowing a certain degree of decision to the specialised user completely independent of the machine. This level has been developed below. And finally, the user level is completed by programs with graphic tools and environments to be used by an operator. The implementation is also explained at the end of this section.

### 2.1 *Application level*

In this level two kinds of objects have been considered: physical elements objects and process oriented objects that perform some application on the former. Following the OMT methodology two levels of abstraction are considered for the two types of objects: the object class defined by general type of object, i.e class robot, and object definition for specific objects, i.e. Puma robot. Finally different data structures to identify different elements of the robot have been defined.

2.1.1. *Object class* Several classes of objects for the two types of objects have been defined. In the case of process oriented objects three classes of objects have been defined: class Control_Algorithms that defines the data and functions required to apply a control algorithm, class Dynamics that permits the implementation of different formulations for the dynamic equations, and class Kinematics that defines the algorithms to evaluate the kinematics of the system. Two classes have been defined in the physical tools oriented objects: class Robot and class Control_Cabinet. In these objects several attributes have been defined as links to other classes of objects. Figure 1 shows the relationships between the five classes.

2.1.2. *Object definition* The definitions of objects refers to specific objects. Figure 2 shows the specific objects for the class Robot that have been implemented.

From class Dynamics different derived objects in function of the formulation used can be created. As the Lagrange-Euler formulation has been chosen then the class Lagrange_Euler can be defined. This formulation is used to obtain the dynamic equations: the energy equations and the torque equations. From this class others have been derived: Linear_Energy and Linear_Torque. The class Linear_Energy has been developed to implement the total energy as a linear function of

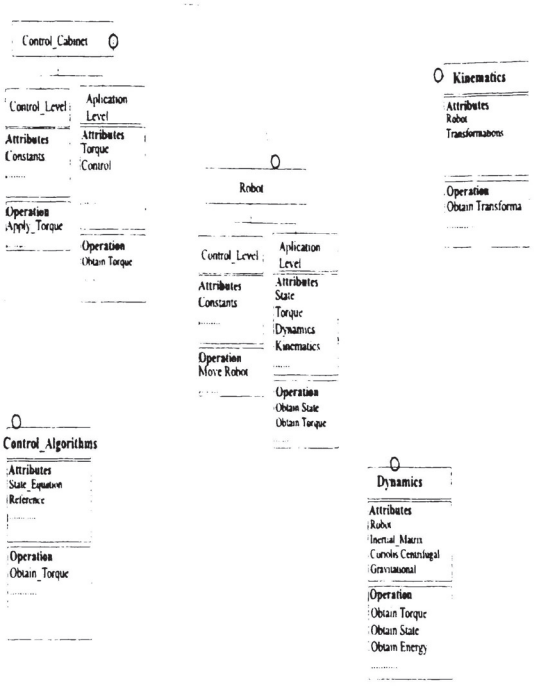


Fig. 1. Class object definition following OMT standard.

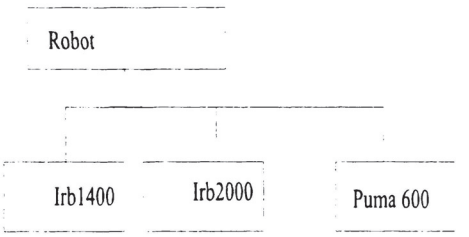


Fig. 2. Definition of class Robot.

the inertial parameters. This class is employed to solve the problem of inertial parameter identification using a reformulation from the Lagrange-Euler equation given by Peñalver *et al.* (1999). To guarantee identification two problems must be solved: a set of excited trajectories must be obtained -class Excited_Trajectories-(Gautier and Khalil, 1992) and the model must be reduced to identify only a linear set of parameters (Sheu and Walker, 1991). The class Linear_Torque permits the torque equation to be obtained as a linear relation with the inertial parameters. To do this there are several adaptive control algorithms, for example, (Johansson, 1990), class State_Equation_RJ and, (Spong and Vidyasagar, 1989), class State_Equation_ST. In figure 3 a definition of class Dynamics is shown.

2.1.3. *Other objects* It is also necessary to define other kinds of objects to facilitate the solution to this problem. The class Polynom allows trajectories for robot simulations to be defined, the class Trajectories allows reference trajectories for robot simulations to be defined. The class Dynamic_Simulation evaluates the behaviour of

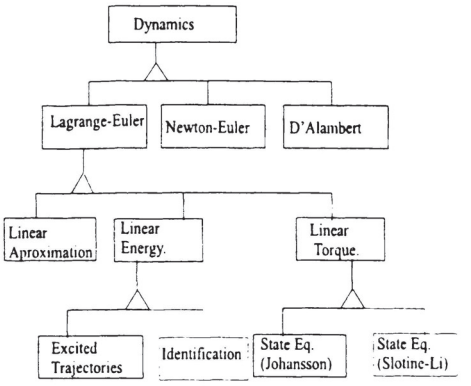


Fig. 3. Definition of class Dynamics.

the robot for a given trajectory and during a previously fixed period of time.

In figure 4 the relationship between all classes of objects is shown, only the types of objects without their derived objects.

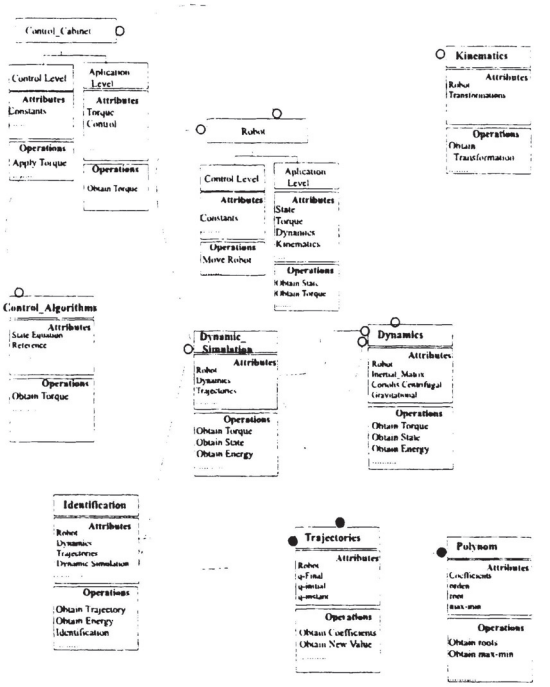

Fig. 4. Relationship between all classes of objects.

## 2.2 *User level*

The user level tries to approximate all the theory developed in the other levels (control algorithms, robot models, etc.) to the users, and allows them to simulate robot control processes and to monitor the behaviour of real robots. With this level, inexpert users can test and compare the control algorithms over some robot models without knowing exactly how these models are implemented. Thus, this level is a useful mechanism for students to understand control algorithms to compare them, and, finally, they could also test the real robot to

see the differences between the robot model and its real behaviour.

An interface as Windows tool has been implemented which allows users to work and obtain graphic results easily. This application has been developed to provide the users with a very flexible environment, that helps them simulate control algorithms and display results, graphically, at the same time as they are calculated. Furthermore, this tool can also monitor real behaviour. The application, developed for Windows 95, has two operation modes, virtual and real. In the virtual mode this tool helps the user simulate and validate theoretical results on a robot model, implemented at application level. Before starting the session, the user can select the control algorithm to be simulated and the robot model on which to simulate it. As already mentioned, during the session the user can display results at the same time as they are calculated, and can also change the data that is shown dynamically, i.e. co-ordinates, speed and acceleration $(q, \dot{q}, \ddot{q})$ of each joint. This mechanism provides the user with a flexible tool to validate results through simulation, and then test them on a real robot. In this mode the application allows the user to monitor the real behaviour of each arm of the robot. This monitoring is based on reference trekking. With this user mode it can be shown that simulated results also work on a real robot. Figure 5 shows an example of this interface.

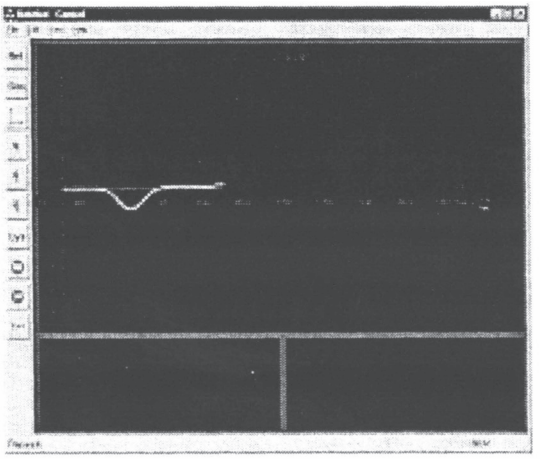

Fig. 5. Example of the interface.

## 3. ALGORITHMS

The Lagrange-Euler formulation has been chosen to solve the different problems mentioned above. The objective is to solve the problem of identification of inertial parameters off-line and on-line. In both cases all algorithms in the literature use the following property: All constant parameters of interest, such as mass, inertial momentum, etc., appear as coefficients of known functions of generalised co-ordinates. Taking each coefficient as a parameter provides a linear relationship that allows the dynamic equation to be expressed as a Hamiltonian equation

$$H(q, \dot{q}) = Y_H(q, \dot{q})\theta, \tag{1}$$

and similar expression for the torque equation

$$\tau(q, \dot{q}, \ddot{q}) = Y_\tau(q, \dot{q}, \ddot{q})\theta, \tag{2}$$

where $Y_H(q, \dot{q})$, is a $1 \times r$ matrix, $Y_\tau(q, \dot{q}, \ddot{q})$ is a $n \times r$ matrix, where $n$ is the number of links and $r$ is the number of parameters; and $\theta$ is the $r \times 1$ parameter vector. If all the different parameters are considered then $r = 10n$. The focus is to find a general mathematical and computable algorithm that allows this problem to be solved.

### 3.1 *The dynamic Model*

The description of the system can be carried out using the Denavit-Hartenberg notation. The system is composed of $n$-joints and $n + 1$ links; link 0 is the base, while link $n$ is the end-effector. Thus the kinetic energy equation can be written as:

$$K = \sum_{i=1}^{n} K_i = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{i} \sum_{k=1}^{i} [tr(U_{ji} J_i U_{ki}^T) \dot{q}_j \dot{q}_k], \tag{3}$$

and the potential energy equation as

$$P = \sum_{i=1}^{n} P_i = -\sum_{i=1}^{n} g\, {}^0A_i \left(m_i\, {}^i\bar{r}_i\right), \tag{4}$$

where ${}^0A_i$ are the homogenous transformation between the frame 0 and the frame $i$, $U_{ji}$ are the matrices associated with the effect of the link $j$ movement on the link $i$; $g$ is the gravity vector and $J_i$ are the inertial parameters tensor and $\dot{q}_j$ is the first derivative of the generalized coordinate $q_j$.

The Lagrangian function can be written as

$$H = K + P = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{i} \sum_{k=1}^{i} tr(U_{ji} J_i U_{ki}^T) \dot{q}_j \dot{q}_k - \sum_{i=1}^{n} g\, {}^0A_i (m_i\, {}^i\bar{r}_i). \tag{5}$$

Considering $B_i = \left(\sum_{j=1}^{i} \sum_{k=1}^{i} (U_{ki} \otimes U_{ji}) \dot{q}_j \dot{q}_k\right)$ then

$$H = Y_K(q, \dot{q})\ \theta_K + Y_P(q)\ \theta_P \tag{6}$$

where

$$Y_K = \frac{1}{2}\left[rowtr\left(B_1\right), \cdots, rowtr\left(B_n\right)\right]_{1\times 16} \quad (7)$$

$$\theta_K = \left[\nu_{4,4}(J_1), \cdots, \nu_{4,4}(J_n)\right]^T_{1\times 16n} \quad (8)$$

$$Y_P = \left[g^0 A_1, \ldots, g^0 A_n\right]_{1\times 16} \quad (9)$$

$$\theta_P = \left[m_1\ {}^1\bar{r}_1, \cdots, m_n\ {}^n\bar{r}_n\right]^T_{1\times 4n}, \quad (10)$$

where $\otimes$ represents the Kronecker product, $\nu_{4,4}(.)$ the operator vector column over a $4 \times 4$ matrix and the following definition is considered:

**Definition 1** Given a matrix $A \in R^{mp\times nq}$, and the partition by columns of $A$, $A = [a_1, a_2, \cdots, a_{nq}], a_i \in R^{mp\times 1}, i = 1, \cdots, nq$. The row-trace of $A$ denoted by $rowtr(\cdot)$, is the row vector defined by the following expression $rowtr(A) = [tr(a_1), tr(a_2), \cdots, tr(a_{nq})]_{1\times nq}$, where $tr(a_i)$ is the trace operator over a column vector $tr(a_i) = tr(v^{-1}_{n,m}(a_i)) = a_{i1} + a_{im+2} + a_{im+3} + \cdots + a_{im+n}$, with $m = n = 4$.

## 4. IMPLEMENTATION

The implementation characteristics are presented. The C++ language and the public domain matrix library called Newmat have been used. The classes Robot, Lagrange_Euler and Lineal_Energy are shown

```
class Robot {
public
Robot(int number_of_joints);\newline{}.....
enum DHnotion{distance, theta, length, alpha};
//Denavit-Hartenberg parameters
.....
Matrix Q[DOF+1]

//matrix relatives of type of link
//prismatic of revolution
Matrix A[DOF+1][DOF+1]; //transformation matrices
Matrix J[DOF+1][DOF+1]; //Inertial parameters matrices
GeneralizedCoordinates STATE;
.....
ObtainA();
.....
}

class Lagrange_Euler: public Dynamics{
public:
Lagrange_Euler(Robot &robot);
.....
Matrix U[DOF+1][DOF+1];
//terms relatives to the first derivative
Matrix DU[DOF+1][DOF+1][DOF+1];
//terms relatives to the second derivative
.....
void ObtianU();
void ObtainDU();
void ObtainD(); // Inertial matrix
void ObtainH(); // Coriolis and Centrifugal forces
void Obtainc(); // gravitational forces
....
void ObtianTorque();
void ObtianSTAETE();
// value of the generalised co-ordinates
}

class Lineal_Energy:public Lagrange_Euler {
public
Matrix YH
Matrix Theta
.....
void Obtain_YH();
.....
```

In this program a structure matrix appears as for example

```
Matrix A[DOF+1][DOF+1]
Matrix U[DOF+1][DOF+1]
```

but it can be written

```
Matrix **A
Matrix **U
```

to reduce memory size given that these matrices are upper triangular. The $A$ matrix has all the information about the different relationships between all the links and the $U$ matrix their first derivative. Matrix $A$ is shown below as an example of data dependency which allows the intrinsic parallelism of the whole structure to be exploited

$$A = \begin{bmatrix} {}^0A_1 & {}^0A_2 & \cdots & {}^0A_n \\ & {}^1A_2 & \cdots & {}^1A_n \\ & & \ddots & \vdots \\ & & & {}^{n-1}A_n \end{bmatrix}. \quad (11)$$

The matrices of the diagonal can be obtained in parallel. Figure 6 shows the data dependency to obtain the remaining matrices of $A$. Each

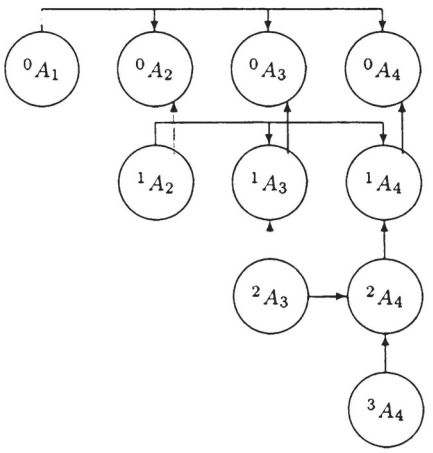


Fig. 6. Dependency data to obtain ${}^iA_j$, for $i = 0 : n-2$, $j = i+2 : n$.

processor can calculate matrices ${}^iA_j$ by columns or by rows. The procedure with the other matrices is similar. An extensive study of this is given in (Fernández, 1999).

## 5. CONCLUSIONS

In this paper a software architecture has been presented for several control algorithms in robot manipulators. From the three levels of the software model, the application model has been considered. In the application model the object oriented methodology is used to develop the software. In this design two aspects are considered: the independence of implementation and the facility of using other types of algorithms different to those given here; and the symbiosis of the design with the algorithms chosen that facilitates the resolution of the problem in a compact form. This paper shows the possibility of exploiting the intrinsic parallelism of the data structure defined to reduce computational cost. Moreover a graphical tool has been developed for Windows at user lever, that allows results to be obtained dynamically. All the trials have been performed by simulation. The authors are also interested in including statistic computation to compare results, test them and obtain graphic results of these comparisons. This software will be integrate to control a real Puma 500 robot.

This design facilitates the addition of new robot models and new formulations for the dynamics and kinematics of the system, as well as new control algorithms by adapting the necessary formulation. Moreover the algorithms in the application level are independent of the machine they run.

## 6. REFERENCES


Albus, S. A., H. G. McCain and R. Lumia (1987). Nasa/nbs standard reference model for telerobot control system architecture (nasrem). Technical report.

Anderson, R. J. (1993). Smart: A modular architecture for robotics and teleoperation. In: *In IEEE International Conference on Robotics and Automation.* DSP Associates, Santa Clara California. pp. 461–421.

Bennis, F. and W. Khalil (1990). Minimum inertial parameters of robots with parallelogram closed-loops. *IEEE* pp. 1026–1031.

Craig, J. (1988). *Adaptative Control of Mechanical Manipulators.* Library of congres Catalogin-in-Publication Data. Addison-Wesley Publishing Company, Inc.

Fernández, J. C. (1999). Simulación dinámica y control de robots industriales utilizando Computación Paralela. PhD thesis. Universidad Politécnica de Valencia.

Gautier, M. and W. Khalil (1992). Exciting trajectories for the identification of base inertial parameters of robots.. *INt. J. Robot. Res.* **11**(4), 362–375.

Gautier, M., F. Bennis and W. Khalil (1990). The use of generalized links to determine the minimun inertial parameters of robots. *J. Robotic Systems* pp. 225–242.

Ha, I. J., M. S. Ko and S. K. Kwon (1989). An efficient estimation algorithm for the model parameters of robotic manipulators. *IEEE Transac. On Robotics and Automation* **5**(3), 386–394.

Johansson, R. (1990). Adaptive control of robot manipulator motion. *IEEE Transactions On Robotics and Automation* **6**(4), 483–490.

Kawasaki, H. and T. Shimizu (1998). Symbolic analysis of robot base parameter set using grobner-basis. *Journal of Robotics and Mechatronics* **10**(6), 475–481.

Khalil, W. and D. Creusot (1997). SYMORO+: A system for the symbolic modelling of robots. *Robotica* **15**, 153–161.

Nilsson, K. (1993). Objected oriented dsp programing. In: *In Proceedings of the Foruth International Conference on Signal Processing Applications & Technology.* DSP Associates, Santa Clara California. pp. 561–570.

Peñalver, L. (1998). Modelado Dinámico e Identificación Parámetrica Para el Control de Robots Manipuladores. PhD thesis. Universidad Politécnica de Valencia.

Peñalver, L., J. Tornero and V. Hernández (1998). The energy equation: New propierties to obtain the linear relation of inertial parameteres. In: *Controlo 98.*

Peñalver, L., V. Hernández, J. Tornero and J. C. Fernández (1996). Total energy lineal relation with the inertial parameters. In: *International Measurement Confederation. Measurement and Control in Robotics. 6th International Symposium* (IMEKO, Ed.). IMEKO. pp. 298–303.

Sheu, S. Y. and M. W. Walker (1991). Identifying the independent inertial parameter space of robot manipulators. *The International Journal of Robotics Research* **10**(6), 668–682.

Spong, M. W. and M. Vidyasagar (1989). *Robot Dynamics And Control.*

Zomaya, A. Y. (1992). *Modelling and Simulation of Robot Manipulators. A Parallel Processing Approach.* Vol. 8 of *World Scientific Series in Robotics and Automated Systems.* World Scientific.