Hermínio Duarte-Ramos
Amadeu Leão Rodrigues
Coordenadores

ENGENHARIA ELECTROTÉCNICA LUSO-ESPANHOLA

Volume 4

Instalações Eléctricas
Instrumentação e Controlo
Automação e Robótica
Posters

EDINOVA
Lisboa
1999

RobLib: A Library of Models for Industrial Robots

Carla S. M. Cordeiro[1], J.A. Tenreiro Machado[2]

[1] **Modern University**
Dept. of Automation and Control Eng.
Porto, Portugal

[2] **Polytechnic Inst. of Porto**
Dept. of Electrical Engineering
Porto, Portugal

Abstract

This paper describes a software package developed in the Windows environment. This software consists of a library for industrial robots models and allows the calculation of the direct and the inverse kinematics for several manipulators. The program allows trajectory planning algorithms, both in the joint and cartesian spaces, using linear or cubic splines interpolation procedures. The kinematic equations for each robot were optimised in what concerns the number of required operations in order to optimize the computational load.

Key Words: Robots, Kinematics, Trajectory Planning, Software.

1. Introduction

This paper introduces the *RobLib* package and describes his functional aspects. Given the popularity of Microsoft Windows it was decided to implement the software in this environment using the Microsoft Visual Basic language.
Many programs are available for modelling robots but most of them focus on simulation aspects [1,9]. *RobLib* consists in a library of models and algorithms for standard industrial manipulators. It contains several robots with different mechanical structures and up to six joints (rotational and prismatic). For each robot the user can calculate the direct and inverse kinematics for the positions, the velocities and the accelerations using Homogeneous Transformation Matrices (HTMs), Euler Parameters or Euler Angles. The trajectory planning algorithms can be executed using linear or cubic splines interpolation procedures, either in the operational or in the joint spaces.
In this line of thought, the paper is organised as follows. Section 2 gives an overview of the software capabilities including some application examples. Based on this description, section 3 draws the main conclusions and presents the perspectives towards future software enhancements.

2. Overview of the Software

The *RobLib* package was designed to take full advantage of the Windows environment. All the commands and data are entered through pull down menus, buttons and dialog boxes. The program has a modular structure; therefore, improvements and modifications can be made without affecting the other modules. Moreover, new modules can be easily added to expand the software package.
The main window contains a menu bar that includes the following entries: File, Robot and Help.
The Robot menu (Fig.1a) allows the user to choose the robot to be studied. The robot is classified by the series of joints from the base to the hand (*e.g.* Portique 80 - PPPRRR). After selecting a robot, *RobLib* presents a new window (fig. 1b) whose menu has two major choices: Kinematics and Trajectory Planning that are described in the next two sub-sections.

Fig. 1a - *RobLib* main window

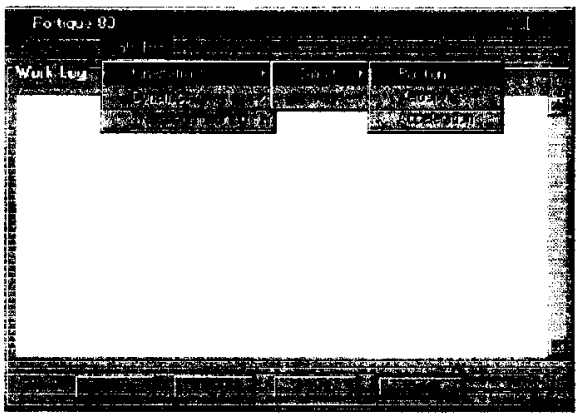


Fig. 1b - Window for the Portique 80 robot.

2.1. Kinematics

The direct kinematics consists on determining the position and orientation of the end-effector in the cartesian space given the values of the robot joint variables. The mathematical foundations are the concept of HTMs and the Denavit-Hartenberg (DH) formalism [10-16] of assigning coordinate frames at each robot joint. *RobLib* provides information about the DH parameters for each robot as well as a schematic representation of the mechanical structure.
The results of the direct kinematics can be seen in HTM (representing the end-effector coordinates in the base reference frame) or, alternatively, using cartesian coordinates for the position and Euler Parameters or Euler Angles for the orientation [17].
Using a rectangular coordinate system for the end-effector (Fig. 2), consisting of the unit vectors **n**, **s**, **a**, can be defined a 4×4 HTM $\mathbf{T}_{1,n+1}$ as in (1), in which **p** indicates the gripper position relatively to the base.

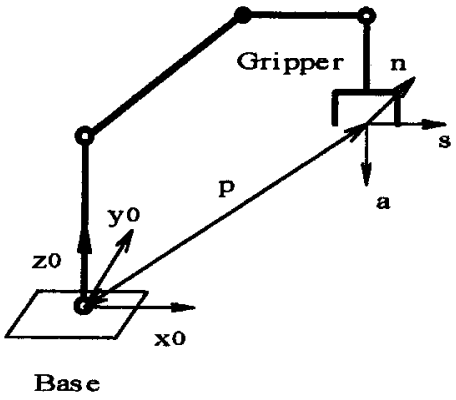


Fig. 2 - Local end-effector coordinate frame.

$$\mathbf{T}=\begin{bmatrix}\mathbf{n} & \mathbf{s} & \mathbf{a} & \mathbf{p}\\ 0 & 0 & 0 & 1\end{bmatrix}=\left[\begin{array}{ccc|c} n_x & s_x & a_x & p_x \\ n_y & s_y & a_y & p_y \\ n_z & s_z & a_z & p_z \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{1}$$

Moreover, the Euler parameters can be expressed in terms of the vectors **n**, **s** and **a** as:

$$\begin{aligned} p_E &= \frac{1}{2}\mathrm{sgn}\left[s_z - a_y\right]\sqrt{n_x - s_y - a_z + 1} \\ q_E &= \frac{1}{2}\mathrm{sgn}\left[a_x - n_z\right]\sqrt{s_y - a_z - n_x + 1} \\ r_E &= \frac{1}{2}\mathrm{sgn}\left[n_y - s_x\right]\sqrt{a_z - n_x - s_y + 1} \\ s_E &= \sqrt{1 - p_E^2 - q_E^2 - r_E^2} \end{aligned} \tag{2}$$

Fig. 3 depicts the results of the direct kinematics, in Euler Parameters, for the Portique 80 robot and a given set of inputs. The joint limits specify the range of variation for each joint. For a revolute/prismatic joint the limits are specified as the minimum and maximum angles/extensions. Those limits can not be exceeded and a warning message appears if the user tries to enter improper values.

Identical procedures are available for the velocities and accelerations. The calculation of the joint velocities requires the previous calculation of the joint positions. Furthermore, the calculation of the accelerations implies the previous calculation of the joint positions and velocities. Therefore, those options activated, consecutively, during the program session.

The inverse kinematics may lead to more than one solution [10-12]. For a given configuration the inverse kinematics implies the determination of the required joint variables. In fact, many robots exhibit multiple solutions representing different sets of joint displacements that yield the same gripper configuration (Fig. 4). The analytical solution for inverse position is obtained either through the decoupling of the kinematic equations [17] or using the trigonometric method [12].

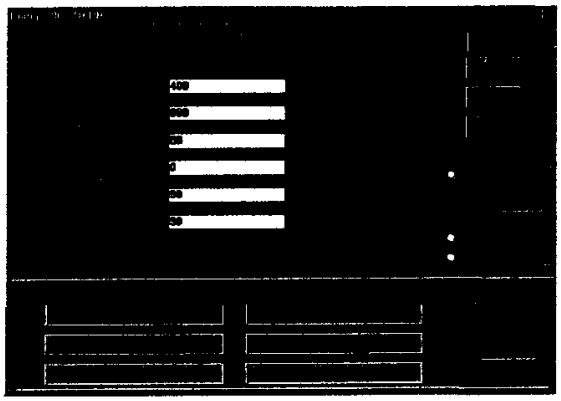
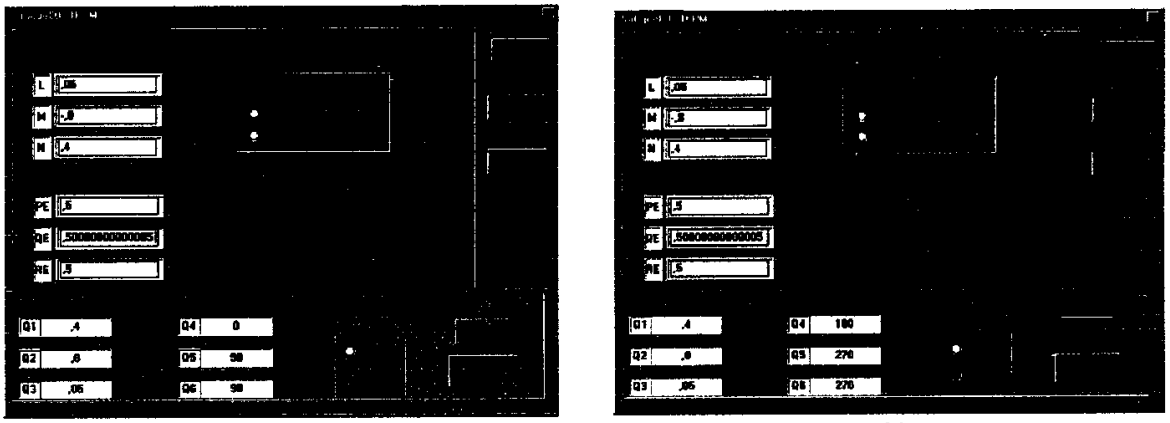


Fig.3 - Position direct kinematics, in Euler parameters, for the Portique 80 robot.



a) b)

Fig 4 - Position inverse kinematics:
a) First solution, b) Second solution

The equations for each robot were optimized in what concerns the number of required operations [18]. Table 1 presents the optimized equations for the direct kinematics of the robot Portique80 using Euler Parameters.
All the results obtained during a study session are written in a work log and can be saved in a text file.

2.2. Path Planning

The trajectory planning algorithms deal with the mathematical representation of robot paths. Given a set of points through which the end-effector must pass, the interpolation procedure approximates the desired path by a polynomial function and defines a temporal sequence of points for the robot to follow, from the initial to final locations [10].
RobLib allows the definition of points both in the cartesian and the joint spaces, while the interpolation can be made using either cubic splines or straight lines.

Position Direct Kinematics

```
Sub DKPM_PORTIQUE80 ()

C4 = OPTCOS(Q(4))
C5 = OPTCOS(Q(5))
C6 = OPTCOS(Q(6))
S4 = OPTSIN(Q(4))
S5 = OPTSIN(Q(5))
S6 = OPTSIN(Q(6))

R6 = PORTmat(7, 5)

'Portique80 DKPM Equations

D01 = S5 * R6
D02 = S4 * D01
D03 = C4 * D01
D04 = C5 * R6
D05 = S5 + C6
D06 = C4 * D05
D07 = S5 - S4
D08 = S6 * D07
D09 = C6 + S4
D10 = S5 * D09
D11 = S4 - C6
D12 = S5 * D11
D13 = S4 * C6
D14 = C4 * C5
D15 = D14 * S6
D16 = S5 * D09
D17 = S4 * C5 * S6
D18 = D14 * C6
D19 = D13 + 1
D20 = D19 - D15
D21 = C4 * S6
D22 = C5 * (1 - D13)

'RESULTS

L = Q3 - D04
M = Q2 - D02
N = Q1 + D03
PE = .5 * Sgn(D06 + D17) *
Sqr(D20 + D16)
QE = .5 * Sgn(D08 - D18) *
Sqr(D20 - D16)
RE = .5 * Sgn(D21 + D22) *
Sqr(D15 - D13 + D12 + 1)
End Sub
```

Velocity Direct Kinematics

```
Sub DKVM_PORTIQUE80
()

C4 = OPTCOS(Q(4))
C5 = OPTCOS(Q(5))
S4 = OPTSIN(Q(4))
S5 = OPTSIN(Q(5))

d26 = PE * PE
d27 = QE * QE
d28 = RE * RE
d29 = d26 + d27 + d28
SE = Sqr(1 - d26 - d27 - d28)

R6 = PORTmat(7, 5)

D01 = R6 * DQ5
D02 = S5 * D01
D03 = R6 * DQ4
D04 = S5 * D03
D05 = C5 * D01
D06 = S4 * D05
D07 = C4 * D05
D08 = C4 * D04
D09 = S4 * D04
D10 = SE * DQ4
D11 = RE * DQ4
D12 = QE * DQ4
D13 = C4 * SE + S4 * PE
D14 = S4 * SE - C4 * PE
D15 = RE * C4 - S4 * QE
D16 = S4 * RE + C4 * QE
D17 = S5 * D16 + C5 * SE
D18 = S5 * D14
D19 = C5 * RE - D18
D20 = D13 * S5
D21 = D20 - C5 * QE
D22 = S5 * DQ6

'RESULTS

DL = D02 + DQ3
DM = -DQ2 - D08 - D06
DN = DQ1 - D09 + D07
DPE = .5 * (D10 + D15 *
DQ5 - D17 * DQ6)
DQE = .5 * (D13 * DQ5 -
D11 + D19 * DQ6)
DRE = .5 * (D12 + D14 *
DQ5 + D21 * DQ6)

End Sub
```

Acceleration Direct Kinematics

```
Sub DKAM_PORTIQUE80 ()

C4 = OPTCOS(Q(4))
C5 = OPTCOS(Q(5))
S4 = OPTSIN(Q(4))
S5 = OPTSIN(Q(5))

E1 = PE * PE
E2 = QE * QE
E3 = RE * RE
E4 = E1 + E2 + E3

SE = Sqr(1 - E1 - E2 - E3)
DSE = -(PE * DPE + QE * DQE + RE * DRE) / SE
R6 = PORTmat(7, 5)

D01 = S5 * R6
D02 = C5 * R6
D03 = DQ5 * DQ5
D04 = DQ4 * DQ4
D05 = D04 * DQ5
D06 = D03 * DQ4
D07 = C4 * DO2
D08 = S4 * D02
D09 = C4 * D01
D10 = S4 * D01
D11 = RE * C4 - S4 * QE
D12 = RE * S4 + QE * C4
D13 = DRE * C4 - DQE * S4
D14 = DRE * S4 + DQE * C4
D15 = DQ4 * DQ5
D16 = DQ4 * DQ6
D17 = DQ5 * DQ6
D18 = SE * C4 + PE * S4
D19 = PE * C4
D20 = SE * S4
D21 = D19 - D20
D22 = DSE * C4 + DPE * S4
D23 = DPE * C4 - DSE * S4
D24 = PE * C4 + SE * S4
D25 = DSE * S4 - DPE * C4
D26 = DPE * S4 + DSE * C4

'RESULTS
DDL = DDQ3 + D01 * DDQ5 + D02 * D03
DDM = -DDQ2 - D09 * DDQ4 - D08 * DDQ5 + D08 *
D05 + D09 * D06
DDN = DDQ1 - D10 * DDQ4 + D07 * (DDQ5 - D05)
+ D10 * D06
DDPE = .5 * (SE * DDQ4 + D11 * DDQ5 - (S5 * D12
+ C5 * SE) * DDQ6 + DSE * DQ4 + D13 * DQ5 - D12
* D15 - (S5 * D14 + C5 * DSE) * DQ6 - S5 * D11 *
D16 - (C5 * D12 - S5 * SE) * D17)
DDQE = .5 * (D18 * DDQ5 - RE * DDQ4 + (S5 * D21
+ C5 * RE) * DDQ6 - DRE * DQ4 + D22 * DQ5 + D21
* D15 + (C5 * DRE + S5 * D21) * DQ6 - S5 * D18 *
D16 + (C5 * D21 - S5 * RE) * D17)
DDRE = .5 * (QE * DDQ4 - D19 * DDQ5 + (S5 * D24
- C5 * QE) * DDQ6 + DQE * DQ4 + D25 * DQ5 + D24
* D15 + (S5 * D26 - C5 * DQE) * DQ6 + S5 * D19 *
D16 + (C5 * D24 + S5 * QE) * D17)

End Sub
```

Table 1- Optimized equations of the direct kinematics (using Euler Parameters) of the robot Portique80

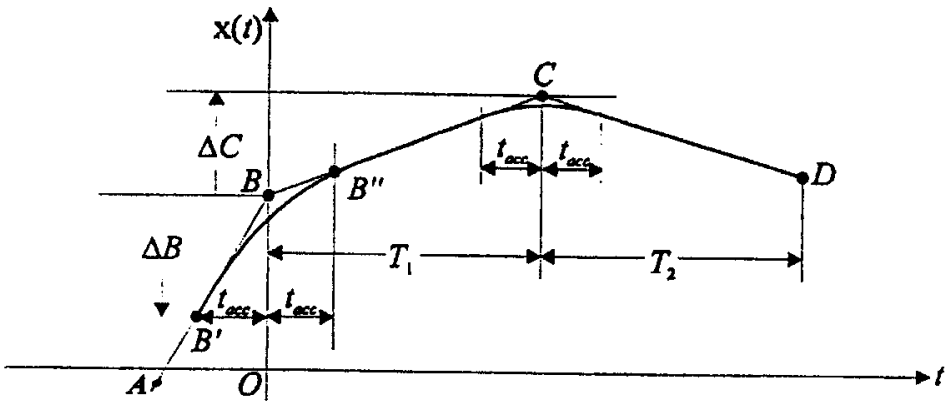


Fig. 5 – Transition between two adjacent straight line segments

If the user chooses straight lines, he can define the acceleration time allowed at the trajectory corner points, that is, he can establish the motion softness at these points.
A straight line trajectory between two points can be described by equation (3) where $p(t) \in [0,1]$ allows the choice of a velocity profile.

$$x(t) = x_0 + p(t)\,(x_f - x_0\,), \qquad 0 \le t \le \mathrm{T} \tag{3}$$

Often, there are not just two but a set of several points through which the end-effector must pass. In order to achieve a smooth motion, and avoid velocities discontinuities, the desired trajectory must be planned so that the straight lines are "bented" around the corner points that join two adjacent line segments (Fig 5). Therefore, in the interval $-t_{acc} \le t \le t_{acc}$ the trajectory is given by equations (4-8) for the position, velocity and acceleration.

$$\mathrm{x}(t) = \left[\left(\Delta C\frac{t_{acc}}{T_1} + \Delta B\right)(2-h)h^2 - 2\Delta B\right]h + B + \Delta B \tag{4}$$

$$\dot{\mathrm{x}}(t) = \left[\left(\Delta C\frac{t_{acc}}{T_1} + \Delta B\right)(1.5-h)2h^2 - \Delta B\right]\frac{1}{t_{acc}} \tag{5}$$

$$\ddot{\mathrm{x}}(t) = \left(\Delta C\frac{t_{acc}}{T_1} + \Delta B\right)(1-h)\frac{3h}{t_{acc}^2} \tag{6}$$

$$\mathrm{h} = \frac{\mathrm{t} + \mathrm{t}_{acc}}{2\mathrm{t}_{acc}}, -t_{acc} \le t \le t_{acc} \tag{7}$$

$$\Delta C = C - B \;\; ; \;\; \Delta B = B' - B \tag{8}$$

The bending of the desired trajectory is followed by a straight line with constant velocity according with the equations:

$$x(t) = \Delta C\, h + B \;;\; h = \frac{t}{T_1} \tag{9}$$

$$\dot{x}(t) = \frac{\Delta C}{T_1} \tag{10}$$

$$\ddot{x}(t) = 0 \tag{11}$$

Based on these equations, *RobLib* calculates the corresponding joint variables. For example, Fig. 6 shows the time evolution of positions, velocities and accelerations, given six points defined in joint space, using linear interpolation with t_{acc} = 2 seconds for the Portique 80 robot. The algorithm assures the velocity and the acceleration continuity at all points, including the initial and the final locations, by introducing two extra points in the trajectory.

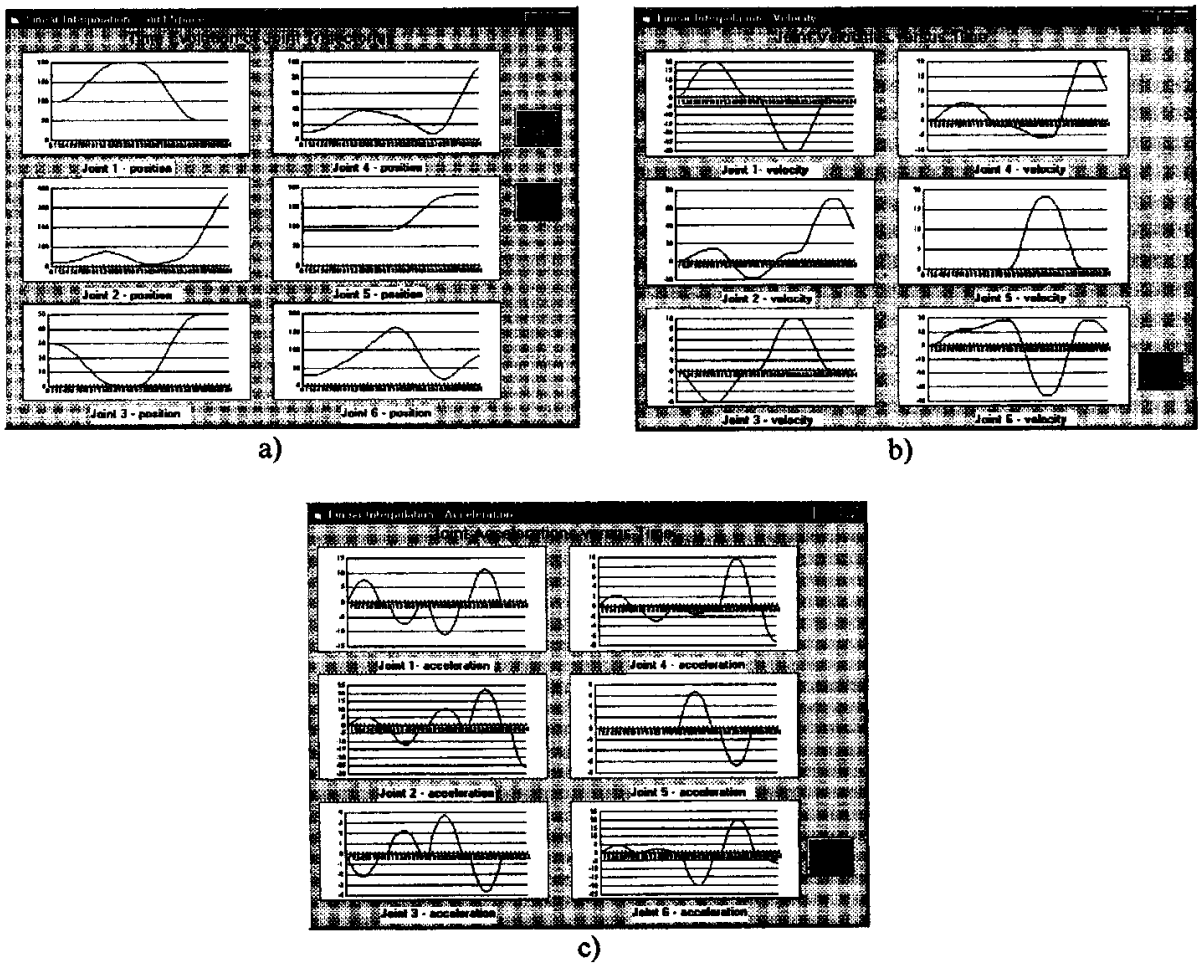


Fig. 6 - Time evolution of joint variables using a linear interpolation in the joint space with t_{acc} = 2 seconds.
a) Positions,
b) Velocities,
c) Accelerations.

3. Conclusions and Future Developments

A program for the calculation of the direct and the inverse kinematics and trajectory planning algorithms was presented. The *RobLib* program is implemented in Visual Basic for Microsoft Windows and provides a library of models for industrial robots.
Future versions will expand the capabilities of *RobLib* in order to include new trajectory planning interpolation procedures and to implement dynamic and control algorithms. Due to *RobLib* modular structure improvements and modifications can be made without affecting the rest of the program and, therefore, new modules can be easily added to expand the package.

References

[1] J.A. Tenreiro Machado and Alexandra M. Galhano, *WinRob: An Educational Program for Robotics*, Int. Journal Elect. Engineering. Education, vol. 34, pp. 37-47, 1997

[2] S. Derby, *In Position: Simulating Robotic Workcells on a Micro*, Computers in Mechanical Engineering, pp. 34-37, Sept. 1986.

[3] R.B. White, R.K. Read, M.C. Koch and R.J. Schilling, *A Graphics Simulator for a Robotic Arm*, IEEE Trans. on Education, vol. 32, no. 4, pp. 417-429, Nov. 1989.

[4] T. Raz, *Graphics Robot Simulator for Teaching Introductory Robotics*, IEEE Trans. on Education, vol. 32, no. 2, pp. 153-159, May 1989.

[5] R.E. Parkin, *An Interactive Robotic Simulation Package*, Simulation, vol. 56, no. 5, pp. 337-345, May 1991.

[6] A.M. Eydgahi and J.J. Sheehan, *A Computer Animation of Robotic Manipulators and Workcells*, IEEE Control Systems, vol. 11, n. 4, pp. 56-59, June 1991

[7] M.C. Leu and Y.S. Wang, *Studying Robot Kinematics and Dynamics with the Aid of MATHEMATICA*, Int. J. of Mechanical Eng. Education, vol. 19, n. 3, pp. 213-228, 1990.

[8] N. Vira and E. Tunstel, *Use of Symbolic Computation in Robotics Education*, IEEE Trans. on Education, vol. 35, no. 1, pp. 18-30, Feb. 1992.

[9] J.F. Nethery and M.W. Spong, *Robotica: A Mathematica Package for Robot Analysis*, IEEE Robotics & Automation Magazine, vol. 1, no. 1, pp. 13-20, March 1994.

[10] K. S. Fu; R. C. Gonzalez; C. S. G. Lee, *Robotics: Control, Sensing, Vision and Intelligence*, McGraw-Hill, 1987.

[11] J. Robert Schilling, *Fundamentals of Robotics: Analysis and Control*, Prentice-Hall, 1990.

[12] R. Paul, *Robot Manipulators: Mathematics, Programming and Control*, MIT Press, 1981.

[13] M. Spong; M. Vidyasagar, *Robot Dynamics and Control*, John Wiley & Sons, 1989.

[14] H. Asada, J.-J. E. Slotine, *Robot Analysis and Control*, John Wiley & Sons, 1986.

[15] J. Craig, *Introduction to Robotics: Mechanics and Control*, Addison-Wesley, 1989

[16] W. Wolovich, *Robotics: Basic Analysis and Design*, Holt, Rinehart and Winston, 1987.

[17] S. Megahead, *Principles of Robot Modelling and Simulation*, John Wiley & Sons, 1993.

[18] J. Tenreiro Machado and A. M. Galhano, *Benchmarking Computer Systems for Robot Control*, IEEE Trans. on Education, vol. 38, no. 3, pp. 205-210, Aug. 1995.