Journal of Intelligent and Robotic Systems **31:** 355–377, 2001.

A PC-Based Simulator/Controller/Monitor Software for a Generic 6-DOF Manipulator

TAREK M. SOBH, ABDELSHAKOUR A. ABUZNEID and RAUL MIHALI
Department of Computer Science and Engineering, University of Bridgeport, 169 University Avenue, Bridgeport, CT 06601, U.S.A.

(Received: 26 October 1999; in final form: 21 March 2000)

Abstract. General form application is a very important issue in industrial design. Prototyping a design helps in determining system parameters, ranges and in structuring better systems. Robotics is one of the industrial design fields in which prototyping is crucial for improved functionality. Developing an environment that enables optimal and flexible design using reconfigurable links, joints, actuators and sensors is essential for using robots in the education and industrial fields. We propose a PC-based software package to control, monitor, and simulate a generic 6-DOF (six degrees of freedom) robot including a spherical wrist. This package may be used as a black box for the design implementations or as a white (detailed) box for learning about the basics of robotics and simulation technology.

Key words: automation, manipulator, prototyping, robot control, robot simulation.

1. Introduction

To design a complete and efficient robotics system there is a need for performing a sequence of cascaded tasks. The design task starts from determining the application of the robot, the performance requirements, and then determining the robot configuration and parameters suitable for that application. The physical design starts from gathering the parts and assembling the robot. Developing the required software (controller, simulator and monitor) elements is the next task. The next stage includes manipulator testing which determines the performance of the robot and the efficiency of the design. Our aim is to build a complete PC-based software package for control, monitoring and simulation of a 6-DOF manipulator, including a spherical wrist. The design will be independent of any existing specific robot parameters. The package will be an integration of several packages.

Figure 1 shows how such a PC-based robot could be controlled, possibly using different schemes.

The idea for this work came from a project we have done in a robotics class at University of Bridgeport. The project was to design a fully integrated package to control, monitor, and simulate an SIR-1 robot. The SIR-1 robot is a 6-DOF robot with a gripper. While working on the project, we continuously looked for the existence of similar prototyping packages on the market. We did a wide range search

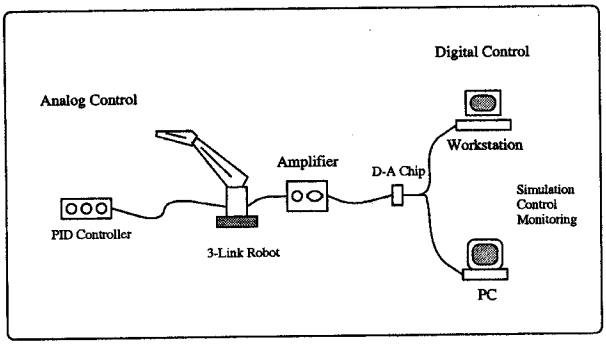

Figure 1. Typical robot control setup.

and an exhaustive market survey of what was available. We searched a variety of papers, books, book chapters, and web sites. We have also talked to a number of companies that manufacture manipulators. We found out that a reasonable progress had been made in the field, however, most of the prototyping was done for special or specific manipulators, mainly with numerical solutions. Unfortunately, PC-based controller/monitor/simulator packages for generic manipulators are not something common and sufficiently debated. Please visit the following URL's for more information:

www.bridgeport.edu/sobhdir/introb/node36.html

www.bridgeport.edu/sobhdir/introb/rep.html

www.bridgeport.edu/sobhdir/proj/wachter/

www.bridgeport.edu/sobhdir/proj/proto/paper.html

2. Background

The final design of the software package will be a collection of smaller packages. Each of these packages will be independent of any specific set of robot parameters. This can be done by making all calculations symbolically. Needless to say that will make the mathematics more difficult. By using mathematical application packages available nowadays such as Maple, Mathematica, Matlab and others, the job will be easier but not trivial. The next few sections give a theoretical background.

2.1. FORWARD KINEMATICS

The standard Denavit–Hartenberg approach is being taken, Figure 2 showing a physical six-link robot manipulator.

The D-H parameters for our prototype robot are shown in Table I. The parameters for the last 3 links are constants with the exception of θ's, the joint variables, and d_6, the offset parameter, which represents the offset distance between O_3 and the center of the wrist O.

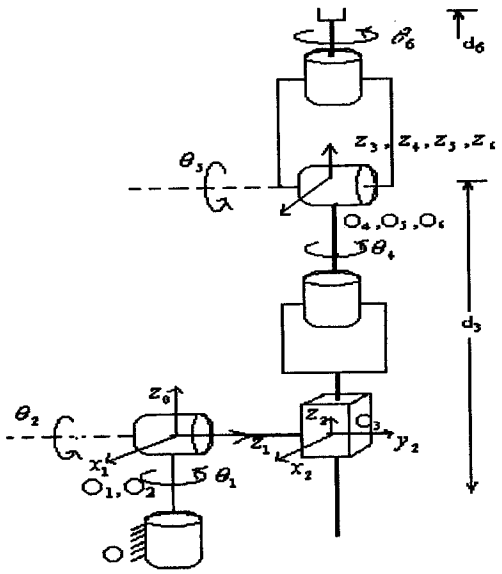

Figure 2. A physical six-link robot manipulator.

Table I. Symbolic DH parameters for the robot.

Link	a_i	α_i	d_i	θ_i
1	a_1	α_1	d_1	θ_1
2	a_2	α_2	d_2	θ_2
3	a_3	α_3	d_3	θ_3
4	0	-90	0	θ_4
5	0	$+90$	0	θ_5
6	0	0	d_6	θ_6

The corresponding transformation matrix is

$$A_0^6 = A_1 A_2 A_3 A_4 A_5 A_6, \tag{1}$$

where

$$A_i = \mathrm{Rot}_{z,\theta_i}, \mathrm{Trans}_{z,d_i}, \mathrm{Trans}_{x,a_i}, \mathrm{Rot}_{x,\alpha_i}, \tag{2}$$

$$A_i = \begin{bmatrix} c(\theta_i) & -s(\theta_i)c(\alpha_i) & s(\theta_i)s(\alpha_i) & a_i c(\theta_i) \\ s(\theta_i) & c(\theta_i)c(\alpha_i) & -c(\theta_i)s(\alpha_i) & a_i s(\theta_i) \\ 0 & s(\alpha_i) & c(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3}$$

2.2. INVERSE KINEMATICS

Inverse kinematics solves for the joint angles, given the desired position and orientation in a Cartesian space. This is a more difficult problem than forward kinematics. The complexity of inverse kinematics can be described as follows. Given a 4×4 homogeneous transformation which sets the required position and orientation,

$$H = \begin{bmatrix} R & d \\ 0 & 1 \end{bmatrix}, \tag{4}$$

where d and R are the given position and orientation of the tool frame relative to the origin. The homogeneous transformation matrix results in 12 nonlinear equations in 16-unknown variables $(a_1, a_2, a_3, \alpha_1, \alpha_2, \alpha_3, \theta_1, \ldots, \theta_6, d_1, d_2, d_3, d_6)$

$$T_{ij}(q_1, \ldots, q_6) = H_{ij}, \tag{5}$$

where $i = 1, 2, 3$, $j = 1, 2, 3, 4$.

For example, to find the corresponding joint variables $(\theta_1, \theta_2, d_3, \theta_4, \theta_5\theta_6)$ for the $RRP : RRR$ manipulator shown in Figure 2 where

$$A_0^6 = \begin{bmatrix} e_{11} & e_{12} & e_{13} & d_x \\ e_{21} & e_{22} & e_{23} & d_y \\ e_{31} & e_{32} & e_{33} & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6}$$

we must solve 12 simultaneous sets of nonlinear equations. See Appendix 1. The first glance at a simple homogeneous transformation matrix eliminates the possibility of finding the solution by solving those 12 simultaneous sets of nonlinear trigonometric equations. These equations are much too difficult to solve directly in closed form and therefore we need to develop efficient techniques that solve for the particular kinematics structure of the manipulator. To solve the inverse kinematic problem, a closed form solution of the equations or a numerical solution could be used [8]. The closed form solution is preferable because in many applications where the manipulator supports or is to be supported by a sensory system, the results need to be supplied rapidly. Since the inverse kinematics can result in a range of solutions rather than a unique one, finding a closed form will make it easy to implement the fastest possible sensory tracking algorithm.

The aim of this work is to try to find a closed solution for a prototype robot which is a general 3-DOF robot having an arbitrary kinematic configuration connected to a spherical wrist. This closed form solution could be attained by different approaches. One possible approach is to decouple the inverse kinematics problem into two simpler problems, known respectively, as inverse position kinematics, and inverse orientation kinematics [1]. To put it in another way, for a six-DOF manipulator with a spherical wrist, the inverse kinematics problem may be divided into two simpler problems, first by finding the position of intersection of the wrist axes, the center, and then finding the orientation of the wrist. Let us suppose that

there are exactly six degrees of freedom and the last three joint axes intersect at a point O. We express the rotational and positional equations as

$$R_0^6(q_1, \ldots, q_6) = R_{3\times 3}, \tag{7}$$

$$d_0^6(q_1, \ldots, q_6) = d, \tag{8}$$

where d and R are the given position and orientation of the tool frame relative to the origin. The assumption of a spherical wrist means that the axes z_4, z_5 and z_6 intersect at O and hence the origins O_4 and O_5, assigned by the D-H convention, will always be at the wrist center O. The important point of this assumption for inverse kinematics is that the motion of the final three links about these axes will not change the position of O. The position of the wrist center is thus a function only of the first three joint variables. Since the origin of the tool frame O_6 is simply a translation by a distance d_6 along the z_5 axes from O_v, the vector O_{6v} relative to the frame $O_0X_0Y_0Z_0$ is

$$O_{6v} - O_v = -d_6 R_v k, \tag{9}$$

where the $_v$ symbol denotes the vectorial notation; note that R is multiplied by k because it is a translation along z axes.

Suppose P_c denotes the vector from the origin of the base frame to the wrist center. Thus, in order to have the end-effector of the robot at the point d with the orientation of the wrist center O located at the point

$$P_c = d - d_6 R k, \tag{10}$$

the orientation of the frame $O_0X_0Y_0Z_0$ with respect to the base must be given by R. If the components of the end-effector position d are denoted d_x, d_y, d_z and the components of the wrist center P_c are denoted P_x, P_y, P_z, then, in this case, the equation results in the relationship

$$\begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix} = \begin{bmatrix} d_x - d_6 r_{13} \\ d_y - d_6 r_{23} \\ d_z - d_6 r_{33} \end{bmatrix}. \tag{11}$$

Using Equation (10), we may find the values of the first three joint variables. Thus, for this class of manipulators, the determination of the inverse kinematics can be summarized in 3 steps:

Step 1: Find q_1, q_2, q_3 such that the wrist center P_c is located at $P_c = d - d_6 k$.

Step 2: Using the joint variables determined in Step 1, evaluate R_0^3.

Step 3: Find a set of *Euler angles* corresponding to the rotation matrix $R_3^6 = (R_3^6)^{-1} R$.

2.3. VELOCITY AND INVERSE VELOCITY KINEMATICS

In order to move the manipulator at constant velocity, or at any prescribed velocity, we must know the relationship between the velocity of the tool and joint velocities. To calculate the velocity, the following Jacobian matrix should be constructed:

$$J = J_1 J_2 J_3 J_4 J_5 J_6, \tag{12}$$

where

$$J_i = \begin{bmatrix} z_{i-1} \times (O_n - O_{i-1}) \\ z_{i-1} \end{bmatrix} \tag{13}$$

if i is revolute and

$$J_i = \begin{bmatrix} z_{i-1} \\ 0 \end{bmatrix} \tag{14}$$

if i is prismatic, where Z_i is the first three elements in the 3rd column of A_0^i and O_i is the first three elements in the 4th column of A_0^i. Then the forward velocity will be

$$\dot{X} = J(q)\dot{q}. \tag{15}$$

The inverse velocity problem becomes one of solving the system of linear equations. The Inverse Velocity Kinematics will then be

$$\dot{q} = J^{-1}(q)\dot{X}, \tag{16}$$

where the singularities will be discussed in the following section.

2.4. ACCELERATION AND INVERSE ACCELERATION KINEMATICS

Differentiating (15) yields the acceleration equation

$$\ddot{X} = J(q)\ddot{q} + J(\dot{q})\dot{q}. \tag{17}$$

By solving (17) for inverse acceleration, we find

$$\ddot{q} = J(q)^{-1}\ddot{X} - J(q)^{-1}J(\dot{q})\dot{q}. \tag{18}$$

2.5. SINGULARITIES

Singularities represent configurations from which certain directions of motion may be unattainable. It is possible to decouple the determination of a singular configuration for those manipulators with a spherical wrist, into two simpler problems [9]. The first is to determine the arm singularities, that is, singularities resulting from

motion of the arm, which consists of the first three or more links, while the second is to determine the wrist singularities resulting from motion of the spherical wrist. Suppose that $n = 6$, that is, the manipulator consists of a 3-DOF arm with a 3-DOF spherical wrist. In this case, the Jacobian matrix is a 6×6 matrix and a configuration is singular if and only if

$$\det J(q) = 0. \tag{19}$$

If we now partition the Jacobian matrix into 3×3 blocks as

$$J = [J_p \quad J_0] = \begin{bmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \end{bmatrix} \tag{20}$$

then, since the final three joints are always revolute,

$$J_0 = \begin{bmatrix} z_3 \times (O_6 - O_3) & z_4 \times (O_6 - O_4) & z_5 \times (O_6 - O_5) \\ z_3 & z_4 & z_5 \end{bmatrix}. \tag{21}$$

Since the wrist axes intersect at a common point O, if we choose the coordinate frames so that $O_3 = O_4 = O_5 = O_6 = O$, J_0 becomes

$$J_0 = \begin{bmatrix} 0 & 0 & 0 \\ z_3 & z_4 & z_5 \end{bmatrix}, \tag{22}$$

and the ith column J_i of J_p is

$$J_i = \begin{bmatrix} z_{i-1} \times (O - O_{i-1}) \\ z_{i-1} \end{bmatrix} \tag{23}$$

if joint i is revolute, and

$$J_i = \begin{bmatrix} z_{i-1} \\ 0 \end{bmatrix} \tag{24}$$

if joint i is prismatic. In this case, the Jacobian matrix has the block triangular form

$$J = \begin{bmatrix} J_{11} & 0 \\ J_{21} & J_{22} \end{bmatrix} \tag{25}$$

with the determinant

$$\det J = \det J_{11} \det J_{22}, \tag{26}$$

where J_{11} and J_{22} are 3×3 matrices each. J_{11} has the ith column $z_{i-1} \times (O - O_{i-1})$ if joint i is revolute, and z_{i-1} if joint i is prismatic, while

$$J_{22} = [z_3 \quad z_4 \quad z_5]. \tag{27}$$

2.6. DYNAMICS

Manipulator dynamics is concerned with the equation of motion, the way in which the manipulator moves in response to the torque applied by the actuators or external forces [7]. There are two problems related to manipulator dynamics that are important to solve:

- inverse dynamics in which the manipulator's equations of motion are solved for given motion to determine the generalized forces required for each joint (control stage) and
- direct dynamics in which the equations of motion are integrated to determine the generalized coordinate response to applied generalized forces (simulation stage).

The equation of motion for an n-axes manipulator is given by

$$Q = M(q)\ddot{q} + C(q, \dot{q})\dot{q} + F(\dot{q}) + G(q), \tag{28}$$

where

q is the vector of generalized joint coordinates describing the pose of the manipulator
$\dot{q}$ is the vector of joint velocities
$\ddot{q}$ is the vector of joint accelerations
M is the symmetric joint-space inertia matrix, or a manipulator inertia tensor
C describes *Coriolis* and centripetal effects
F describes viscous and Coulomb friction and is not generally considered part of rigid-body dynamics
G is the gravity loading
Q is the vector of generalized forces associated with generalized coordinates q.

The equation may be derived via a number of techniques, including the *Lagrangian* method [**?**]. Due to the enormous computational cost of this approach it is always difficult to compute the manipulator torque for real-time control. To achieve a real-time performance, many approaches were suggested, including table lookup and approximation [3]. The most common approximation is to ignore the velocity-dependent term C, since accurate positioning and high speed motion are exclusive in a typical robot application. Practically, a *PID* controller might be a good option to achieve a real-time performance,

$$Q = \ddot{\theta}_d + k_v \dot{E} + k_p E + K_i \int E \, \mathrm{d}t, \tag{29}$$

where k_v, k_p and k_i are the derivative, proportional and integral parameters, respectively. See Figure 3 for a scheme of a PID control loop.

The advantages of using a PID controller are the following:

- Simple to implement;

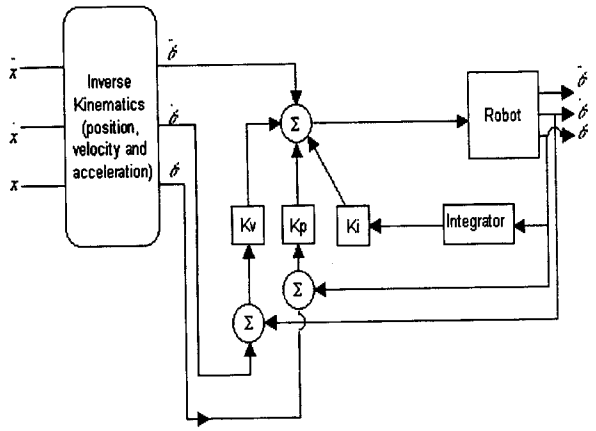

Figure 3. PID control loop.

- Suitable for a real-time control;
- The behavior of the system can be controlled by changing the feedback gains.

For a concrete decision on the controlling technique, often more details are needed, such as mechanics or motors, the case in which the controller will be adjusted accordingly.

2.7. SIMULATION

To simulate the motion of a manipulator, we may use the simulation module by manipulating (28)

$$\ddot{\theta} = M^{-1}(q)[Q - C(q, \dot{q})\dot{q} - F(\dot{q}) - G(q)]. \tag{30}$$

This represents a direct or integral or forward dynamic formulation giving a joint motion in terms of input torque. $M(q)$ is a symmetric joint-space inertia matrix and for a 6-DOF manipulator M it is a 6×6 symmetric matrix. $C(q, \dot{q})$ is the manipulator Coriolis/centripetal torque and for the 6-DOF manipulator C will be a 6×1 matrix. $F(\dot{q})$ is the joint friction torque, where

$$F_i(t) = \begin{cases} B_i + \tau_i^-, & \dot{\theta} < 0, \\ B_i + \tau_i^+, & \dot{\theta} > 0. \end{cases} \tag{31}$$

Figure 4 shows the simulation loop.

2.8. TRAJECTORY GENERATOR

The trajectory generation describes the position velocity and acceleration of each link. This includes the way how the front user interfaces to describe the desired behavior of the manipulator. This could be a very complicated problem depending

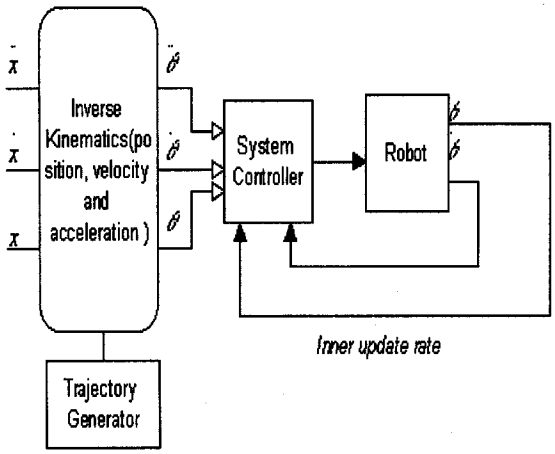

Figure 4. Simulation loop.

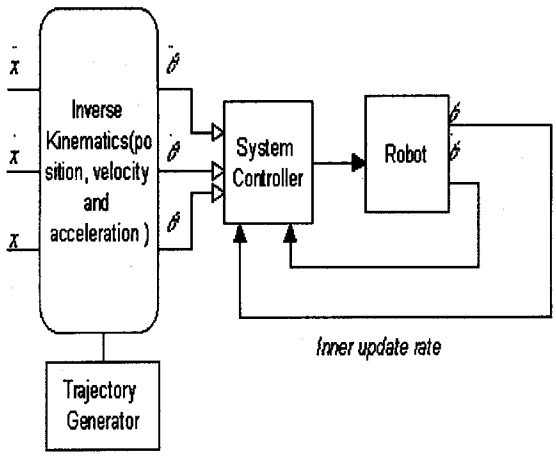

Figure 5. Trajectory generator integrated in the control loop.

on the desired accuracy of the system. In some applications we might need to specify only the goal position, whereas in other applications, we might need to specify the velocity at which the end-effector should move. Since the trajectory generation occurs at run time on a digital computer, the trajectory points are calculated at a certain rate, called the *path update rate*. One disadvantage of using a PID controller is a high update rate that is required to achieve a reasonable accuracy.

Our package role here is to calculate trajectory points that generate a smooth motion for the manipulator. The smoothness of motion is a very important issue due to physical considerations such as the required torque that causes this motion, the friction at the joints, and the frequency of update required to minimize the sampling error – cubic polynomial trajectories were used for the interpolation.

Figure 5 shows how the trajectory generator can be integrated in the control loop. It also shows two update rates, one is the inner update rate which updates the

system control with the actual joint position and velocity. The other updates the system control with the required joint values. The sampling of the two update rates can be different.

3. Package Outline

The package will handle a 6-DOF robot which includes a spherical wrist. Thus, the first three joints could be revolute or prismatic, which yields 8 different configurations of manipulators (i.e., XXX : RRR (X-denotes any type of link, R denotes a revolute link)). The user may use this package as a black box for design applications and as a white box for education and training purposes.

The first input is the manipulator configuration, see Table II. The package will ask for the D-H parameters one after another. By the manipulator configuration, the package recognizes what the variables of the manipulator are.

If the package is used as a white box, the package shows a menu with a list of tasks that the user may select.

The tasks are listed in Ttable III. The user then types the number of the task that should be performed. The package will ask for other inputs required to do the calculations for the assigned job.

The black box includes:

1. Full control loop implementation(PID and Dynamics based).
2. Full simulation loop.
3. GUI with error analysis.

The inputs and outputs can be summarized in the following sections.

3.1. FORWARD KINEMATICS

I/P : q_i where q_i will be θ_i if the joint is revolute, otherwise, it will be d_i

O/P : $x = T_{14}$,
$y = T_{24}$,

Table II.

1	RRR:RRR
2	PRR:RRR
3	RPR:RRR
4	PPR:RRR
5	RPP:RRR
6	PRP:RRR
7	RPP:RRR
8	PPP:RRR

Table III.

[1]	Forward Kinematics
[2]	Inverse Kinematics
[3]	Velocity Kinematics
[4]	Inverse Velocity Kinematics
[5]	Acceleration Kinematics
[6]	Inverse Acceleration Kinematics
[7]	Dynamics
[8]	Simulation
[8]	Simulation
[10]	Trajectory Generation

$z = T_{34}$,
$w_1 = [T_{11} \quad T_{21} \quad T_{31}]$,
$w_2 = [T_{12} \quad T_{22} \quad T_{32}]$,
$w_3 = [T_{13} \quad T_{23} \quad T_{33}]$,

where A is the transformation matrix.

3.2. INVERSE KINEMATICS

I/P : x, y, z, w_1, w_2, w_3,
O/P : q_i where $q_i = \theta_i$ if the ith joint is revolute, and $q_i = d_i$ if it is prismatic.

3.3. VELOCITY KINEMATICS

I/P : $\theta_1, \theta_2, \ldots, \theta_6$ and $\dot{\theta}_1, \dot{\theta}_2, \ldots, \dot{\theta}_6$,
O/P : $v_x, v_y, v_z, w_x, w_y, w_z$,

where $\dot{\theta}_i$ is the ith joint velocity, v is a linear tool velocity vector, and w is an angular tool velocity vector.

3.4. INVERSE VELOCITY KINEMATICS

I/P : $\theta_1, \theta_2, \ldots, \theta_6$ and $v_x, v_y, v_z, w_x, w_y, w_z$,
O/P : $\dot{\theta}_1, \dot{\theta}_2, \ldots, \dot{\theta}_6$.

3.5. ACCELERATION KINEMATICS

I/P : $\theta_1, \theta_2, \ldots, \theta_6, \dot{\theta}_1, \dot{\theta}_2, \ldots, \dot{\theta}_6$ and $\ddot{\theta}_1, \ddot{\theta}_2, \ldots \ddot{\theta}_6$,
O/P : $a_x, a_y, a_z, \dot{w}_x, \dot{w}_y, \dot{w}_z$,

where $\ddot{\theta}_i$ is the ith joint acceleration, α is the linear tool acceleration, and $\dot{w}$ is the angular tool acceleration.

3.6. INVERSE ACCELERATION KINEMATICS

I/P : $\theta_1, \theta_2, \ldots, \theta_6, \dot{\theta}_1, \dot{\theta}_2, \ldots, \dot{\theta}_6$ and $a_x, a_y, a_z, \dot{w}_x, \dot{w}_y, \dot{w}_z$,
O/P : $\ddot{\theta}_1, \ddot{\theta}_2, \ldots, \ddot{\theta}_6$.

3.7. PID CONTROLLER AND TRAJECTORY GENERATION

I/P : $x, \dot{x}, \ddot{x}, k_p, k_v, k_i$,
O/P : $\theta_i, \dot{\theta}_i, \ddot{\theta}_i$.

The trajectory generator updates the controller with a new position of the manipulator targets.

3.8. SIMULATION

I/P : $x, \dot{x}, \ddot{x}, M^{-1}(q), C(q, \dot{q}), F(\dot{q}), G(q)$,
O/P : $\ddot{\theta}_i$.

Dynamics parameters will be:

mass – mass of the link
rx – link COG with respect to the link coordinate frame
ry
rz
Ixx – elements of link inertia tensor about the link COG
Iyy
Izz
Ixy
Iyz
Ixz
Jm – armature inertia
G – reduction gear ratio/joint speed/link speed
B – viscous friction, motor referred
Tc+ – coulomb friction (positive rotation) motor referred
Tc− – coulomb friction (negative rotation) motor referred

4. Project Ideas and Progress

One target of the package is to find closed form solutions such that a direct substitution be made when parameters are entered. That requires to determine which parameters should be variables and which should be constants [2]. Variables could

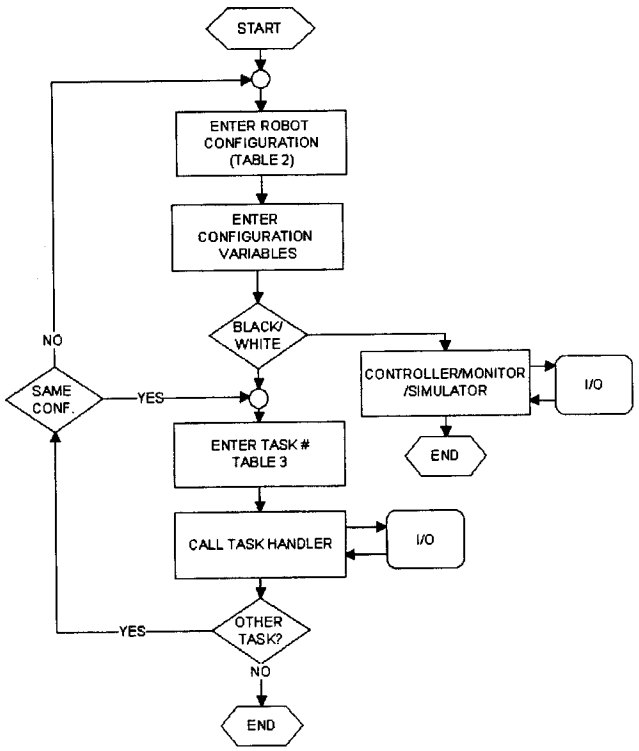

Figure 6. Task flow chart.

be robot parameter configuration variables or state variables. The former are variables that define the structure of the manipulator, so they are constants for the same robot, i.e., a's and α's. The latter describe the state of the robot (Joint Variable). Thus, θ_i may be a state variable if the ith joint is revolute otherwise, it is a configuration variable. The same thing for d_i where it will be a state variable if the joint is prismatic. When the program is run, it will ask for the configuration of the robot (one of those listed in Table II). Then the program will decide what the robot configuration variables are and ask the user to enter them one after another. According to the task, the program is asked to run, it will ask for the state variables. For example if the program is asked to calculate the Inverse Kinematics, the program will ask for the target Cartesian position and orientation to get the values of q's as an output. When the front user asks to do a task, the program calls the task handler. The task handler is a huge set of equations that are invoked when the front user enters the required input, and displays the results rapidly.

Figure 6 shows the task flow chart.

To find out the final and modified shape of equations for each task, a lot of math work has to be done. The next few sections give a few examples of how we managed to do the math chores.

Closed Form Solutions for a Jacobian and an Inverse Jacobian

A Jacobian and an Inverse Jacobian are fundamental objects to calculate the velocity and inverse velocity as shown in (15) and (16). We have used Maple as a math engine to find the Jacobian. It starts from defining the variables (a_i, α_i, θ_i and d_i). After that the program starts calculating A_i's and then finding the final transformation matrix

$$A_0^6 = A_1 A_2 A_3 A_4 A_5 A_6.$$

To calculate the Jacobian, z_i's and O_i's should be involved. Maple computes them and stores them as variables. By now all the parameters required to evaluate the Jacobian are ready, we just let Maple find them out. The solved Jacobian contains a very large number of sines and cosines (inappropriate for using it directly as a code and compile it) and this is passed through simplifications, Maple can handle the task. Afterwards, Maple calculates the inverse Jacobian. To pass the result of the inverse Jacobian, it should be converted to an understandable and simplified C code. This job has to be done for each of the eight robot configurations listed in Table I. Appendix 2 shows how to use Maple to calculate the Jacobian matrix for RRP : RRR.

Closed Form Solutions for $\frac{\mathrm{d}}{\mathrm{d}t} J(q)$

The derivative of Jacobian is required for calculating the acceleration as indicated in 16. We have to derive Jacobians for t as an exterior derivative. Since the version of Maple we used does not do this job and Mathematica does, the output of Maple gets converted to be a suitable input for Mathematica. Then Mathematica differentiates the Jacobian and gets the output as a collection of sines and cosines. Although both Mathematica and Maple can pursue a further simplification, we preferred Maple's output and so a conversion from Mathematica back to Maple is required. Finally, a simplified C-code is found for the derivative of the Jacobian. This procedure is used for each configuration of the eight robot configurations. The derivative will be one part required to evaluate the acceleration and inverse acceleration kinematics as shown in (17) and (18). Appendix 3 shows the derivative for the RRP : RRR manipulator.

The two examples mentioned above show how it is tough in general to find closed form solutions, but by playing around with different tools, it may be a reality. We keep doing this until we find final closed form solutions for the 6-DOF robot, including a spherical wrist. After we do all control and simulation parts, this package is supported by a 3-D and 2-D graphical monitoring system.

Figure 7 shows the monitoring system for the SIR-1 robot as an example. The monitoring will be supported with a controller and a simulator. Figure 8 shows the interface window for the PID controller simulator.

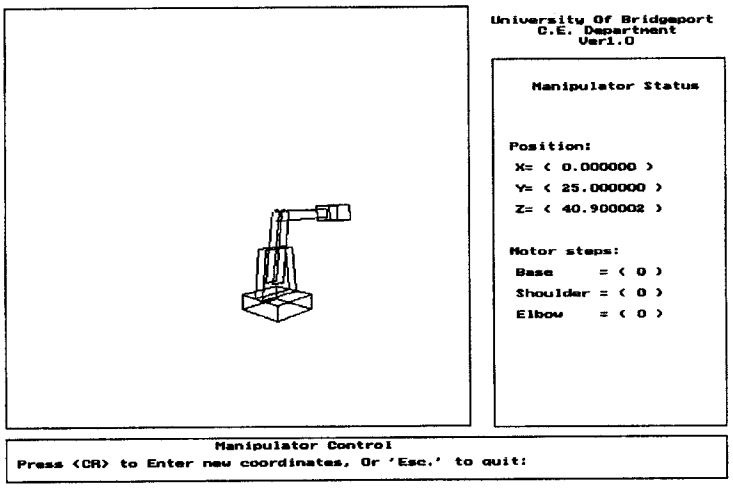

Figure 7. Monitoring menu for the SIR-1 robot.

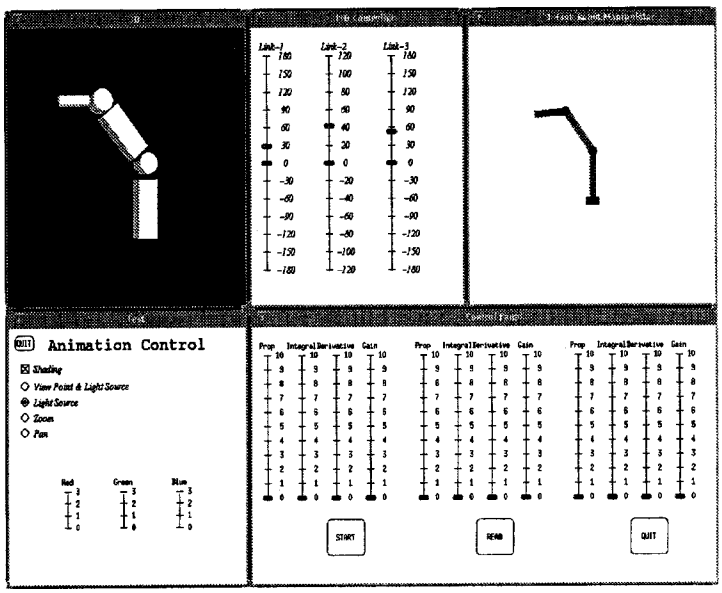

Figure 8. The interface window for the PID controller simulator.

5. Conclusion

A PC-based software package for control, monitoring, and simulation of a 6-DOF, including a spherical wrist, manipulator has been built. The design of the software package relies on symbolic calculations. The design is independent of specific robot parameters, has a generic character and can be used in various design implementations or for learning about the basics of robotics and simulation technology.

Appendix 1

Transformation Matrix for RRP : RRR with(linalg).

Only an excerpt of the material is being listed here due to space limitations, for the entire output please visit
http://www.bridgeport.edu/~abuzneid/ppp/node27.html
#SECTION00010000000000000000

Warning, new definition for norm Warning, new definition for trace

a1 := a1;

$$a1 := a1$$

a2 := a2;

$$a2 := a2$$

(.....)

theta6 := theta6;

$$\theta 6 := \theta 6$$

$A1 := matrix([[cos(theta1), -sin(theta1) * cos(alpha1), sin(theta1) * sin(alpha1), a1 * cos(theta1)], [sin(theta1), cos(theta1) * cos(alpha1), -cos(theta1) * sin(alpha1), a1 * sin(theta1)], [0, sin(alpha1), cos(alpha1), d1], [0, 0, 0, 1]]);$

$$A1 := \begin{bmatrix} \cos(\theta 1) & -\sin(\theta 1)\cos(\alpha 1) & \sin(\theta 1)\sin(\alpha 1) & a1\cos(\theta 1) \\ \sin(\theta 1) & \cos(\theta 1)\cos(\alpha 1) & -\cos(\theta 1)\sin(\alpha 1) & a1\sin(\theta 1) \\ 0 & \sin(\alpha 1) & \cos(\alpha 1) & d1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(.....)

$A6 := matrix([[cos(theta6), -sin(theta6) * cos(alpha6), sin(theta6) * sin(alpha6), a6 * cos(theta6)], [sin(theta6), cos(theta6) * cos(alpha6), -cos(theta6) * sin(alpha6), a6 * sin(theta6)], [0, sin(alpha6), cos(alpha6), d6], [0, 0, 0, 1]]);$

$$A6 := \begin{bmatrix} \cos(\theta 6) & -\sin(\theta 6) & 0 & 0 \\ \sin(\theta 6) & \cos(\theta 6) & 0 & 0 \\ 0 & 0 & 1 & d6 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$AA := evalm(A1\& * A2\& * A3\& * A4\& * A5\& * A6) :$

The next 12 nonlinear equations should be solved to find out the Inverse Kinematics of the Manipulator;

(.....)

e31 := simplify(AA[3, 1]);

$$
\begin{aligned}
e31 := {} & \cos(\theta 6)\cos(\theta 5)\cos(\theta 4)\sin(\alpha 1)\sin(\theta 2)\cos(\theta 3) \\
& + \cos(\theta 6)\cos(\theta 5)\cos(\theta 4)\sin(\theta 3)\sin(\alpha 1)\cos(\theta 2)\cos(\alpha 2) \\
& + \cos(\theta 6)\cos(\theta 5)\cos(\theta 4)\sin(\theta 3)\cos(\alpha 1)\sin(\alpha 2) \\
& - \cos(\theta 6)\cos(\theta 5)\sin(\theta 4)\sin(\alpha 1)\sin(\theta 2)\sin(\theta 3)\cos(\alpha 3) \\
& + \cos(\theta 6)\cos(\theta 5)\sin(\theta 4)\cos(\theta 3)\cos(\alpha 3)\sin(\alpha 1)\cos(\theta 2)\cos(\alpha 2) \\
& + \cos(\theta 6)\cos(\theta 5)\sin(\theta 4)\cos(\theta 3)\cos(\alpha 3)\cos(\alpha 1)\sin(\alpha 2) \\
& - \cos(\theta 6)\cos(\theta 5)\sin(\theta 4)\sin(\alpha 3)\sin(\alpha 1)\cos(\theta 2)\sin(\alpha 2) \\
& + \cos(\theta 6)\cos(\theta 5)\sin(\theta 4)\sin(\alpha 3)\cos(\alpha 1)\cos(\alpha 2) \\
& - \cos(\theta 6)\sin(\theta 5)\sin(\alpha 1)\sin(\theta 2)\sin(\theta 3)\sin(\alpha 3) \\
& + \cos(\theta 6)\sin(\theta 5)\cos(\theta 3)\sin(\alpha 3)\sin(\alpha 1)\cos(\theta 2)\cos(\alpha 2) \\
& + \cos(\theta 6)\sin(\theta 5)\cos(\theta 3)\sin(\alpha 3)\cos(\alpha 1)\sin(\alpha 2) \\
& + \cos(\theta 6)\sin(\theta 5)\cos(\alpha 3)\sin(\alpha 1)\cos(\theta 2)\sin(\alpha 2) \\
& - \cos(\theta 6)\sin(\theta 5)\cos(\alpha 3)\cos(\alpha 1)\cos(\alpha 2) \\
& - \sin(\theta 6)\sin(\theta 4)\sin(\alpha 1)\sin(\theta 2)\cos(\theta 3) \\
& - \sin(\theta 6)\sin(\theta 4)\sin(\theta 3)\sin(\alpha 1)\cos(\theta 2)\cos(\alpha 2) \\
& - \sin(\theta 6)\sin(\theta 4)\sin(\theta 3)\cos(\alpha 1)\sin(\alpha 2) \\
& - \sin(\theta 6)\cos(\theta 4)\sin(\alpha 1)\sin(\theta 2)\sin(\theta 3)\cos(\alpha 3) \\
& + \sin(\theta 6)\cos(\theta 4)\cos(\theta 3)\cos(\alpha 3)\sin(\alpha 1)\cos(\theta 2)\cos(\alpha 2) \\
& + \sin(\theta 6)\cos(\theta 4)\cos(\theta 3)\cos(\alpha 3)\cos(\alpha 1)\sin(\alpha 2) \\
& - \sin(\theta 6)\cos(\theta 4)\sin(\alpha 3)\sin(\alpha 1)\cos(\theta 2)\sin(\alpha 2) \\
& + \sin(\theta 6)\cos(\theta 4)\sin(\alpha 3)\cos(\alpha 1)\cos(\alpha 2)
\end{aligned}
$$

(.....)

Appendix 2

Only an excerpt of the material is being listed here due to space limitations, for the entire output please visit
http://www.bridgeport.edu/~abuzneid/ppp/node27.html
#SECTION00010000000000000000

This is a Jacobian for RRP : RRR
file name: JRRP.mws

a1 := a1;

$$a1 := a1$$

(.....)

theta6 := theta6;

$$\theta 6 := \theta 6$$

$A1 := matrix([[cos(theta1), -sin(theta1) * cos(alpha1), sin(theta1) * sin(alpha1), a1 * cos(theta1)], [sin(theta1), cos(theta1) * cos(alpha1), -cos(theta1) * sin(alpha1), a1 * sin(theta1)], [0, sin(alpha1), cos(alpha1), d1], [0, 0, 0, 1]]);$

$$A1 := \begin{bmatrix} \cos(\theta 1) & -\sin(\theta 1)\cos(\alpha 1) & \sin(\theta 1)\sin(\alpha 1) & a1\cos(\theta 1) \\ \sin(\theta 1) & \cos(\theta 1)\cos(\alpha 1) & -\cos(\theta 1)\sin(\alpha 1) & a1\sin(\theta 1) \\ 0 & \sin(\alpha 1) & \cos(\alpha 1) & d1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(.....)

$$J66 := J[6, 6];$$

$$\begin{aligned} J66 := {} & ((\sin(\alpha 1)\sin(\theta 2)\cos(\theta 3) + \%1\sin(\theta 3))\cos(\theta 4) + (\\ & -\sin(\alpha 1)\sin(\theta 2)\sin(\theta 3)\cos(\alpha 3) + \%1\cos(\theta 3)\cos(\alpha 3) \\ & +(-\sin(\alpha 1)\cos(\theta 2)\sin(\alpha 2) \\ & +\cos(\alpha 1)\cos(\alpha 2))\sin(\alpha 3))\sin(\theta 4))\sin(\theta 5) - (\\ & -\sin(\alpha 1)\sin(\theta 2)\sin(\theta 3)\sin(\alpha 3) + \%1\cos(\theta 3)\sin(\alpha 3) \\ & -(-\sin(\alpha 1)\cos(\theta 2)\sin(\alpha 2) + \cos(\alpha 1)\cos(\alpha 2))\cos(\alpha 3))\cos(\theta 5) \end{aligned}$$

$$\%1 := \sin(\alpha 1)\cos(\theta 2)\cos(\alpha 2) + \cos(\alpha 1)\sin(\alpha 2)$$

Then the Jacobian is converted to a C Code

C(J,optimized);

```
t1 = sin(theta1);
t2 = cos(theta2);
(.....)
t51 = (t24*t25+t39*t40)*t43-(-t45+t46-t47)*t49;
(.....)
t184 = -t141*t40+t148*t25;
J[0][0] = -t52-t54-t56-t57-t59-t61+t62-a1*t1;
(.....)
J[1][0] = t158+t160+t162+t163+t165-t166+t167+a1*t4;
(.....)
J[5][5] = t92;
```

Appendix 3

Only an excerpt of the material is being listed here due to space limitations, for the entire output please visit
http://www.bridgeport.edu/~abuzneid/ppp/node27.html
#SECTION00010000000000000000

$$DJ[1,1] := -(a1*cos(theta1)*dtheta1) - a2*cos(theta1)*cos(theta2)*dtheta1 - a2*cos(alpha1)*cos(theta1)*cos(theta2)*dtheta2 - d2*dtheta1*sin(alpha1)*sin(theta1) + a2*cos(alpha1)*dtheta1*sin(theta1)*sin(theta2) + a2*dtheta2*sin(theta1)*sin(theta2) - d3*(cos(alpha2)*dtheta1*sin(alpha1)*sin(theta1) + cos(alpha1)*cos(theta2)*dtheta1*sin(alpha2)*sin(theta1) + cos(theta2)*dtheta2*sin(alpha2)*sin(theta1) + cos(theta1)*dtheta1*sin(alpha2)*sin(theta2) + cos(alpha1)*cos(theta1)*dtheta2*sin(alpha2)*sin(theta2)) - a3*cos(theta3)*(cos(theta1)*cos(theta2)*dtheta1 + cos(alpha1)*cos(theta1)*cos(theta2)*dtheta2 - cos(alpha1)*dtheta1*sin(theta1)*sin(theta2) - dtheta2*sin(theta1)*sin(theta2)) - dd3*(-(cos(alpha2)*cos(theta1)*sin(alpha1)) - cos(alpha1)*cos(theta1)*cos(theta2)*sin(alpha2)+sin(alpha2)*sin(theta1)*sin(theta2)) - a3*(-(cos(alpha1)*cos(alpha2)*cos(theta2)*dtheta1*sin(theta1)) - cos(alpha2)*cos(theta2)*dtheta2*sin(theta1) + dtheta1*sin(alpha1)*sin(alpha2)*sin(theta1) - cos(alpha2)*cos(theta1)*dtheta1*sin(theta2) - cos(alpha1)*cos(alpha2)*cos(theta1)*dtheta2*sin(theta2))*sin(theta3) - d6*(-(cos(theta5)*(cos(theta3)*sin(alpha3)*(-(cos(alpha1)*cos(alpha2)*cos(theta2)*dtheta1*sin(theta1)) - cos(alpha2)*cos(theta2)*dtheta2*sin(theta1) + dtheta1*sin(alpha1)*sin(alpha2)*sin(theta1) - cos(alpha2)*cos(theta1)*dtheta1*sin(theta2) - cos(alpha1)*cos(alpha2)*cos(theta1)*dtheta2*sin(theta2)) - cos(alpha3)*(cos(alpha2)*dtheta1*sin(alpha1)*sin(theta1) + cos(alpha1)*cos(theta2)*dtheta1*sin(alpha2)*sin(theta1) + cos(theta2)*dtheta2*sin(alpha2)*sin(theta1) + cos(theta1)*dtheta1*sin(alpha2)*sin(theta2) + cos(alpha1)*cos(theta1)*dtheta2*sin(alpha2)*sin(theta2)) - sin(alpha3)*(cos(theta1)*cos(theta2)*dtheta1 + cos(alpha1)*cos(theta1)*cos(theta2)*dtheta2 - cos(alpha1)*dtheta1*sin(theta1)*sin(theta2) - dtheta2*sin(theta1)*sin(theta2))*sin(theta3))) + cos(theta5)*dtheta5*(cos(theta4)*(cos(theta3)*(cos(theta2)*sin(theta1) + cos(alpha1)*cos(theta1)*sin(theta2)) + (cos(alpha1)*cos(alpha2)*cos(theta1)*cos(theta2) - cos(theta1)*sin(alpha1)*sin(alpha2) - cos(alpha2)*sin(theta1)*sin(theta2))*sin(theta3)) + (cos(alpha3)*cos(theta3)*(cos(alpha1)*cos(alpha2)*cos(theta1)*cos(theta2) - cos(theta1)*sin(alpha1)*sin(alpha2) - cos(alpha2)*sin(theta1)*sin(theta2)) + sin(alpha3)*(-(cos(alpha2)*cos(theta1)*sin(alpha1)) - cos(alpha1)*cos(theta1)*cos(theta2)*sin(alpha2) + sin(alpha2)*sin(theta1)*sin(theta2)) - cos(alpha3)*(cos(theta2)*sin(theta1) + cos(alpha1)*cos(theta1)*sin(theta2))*sin(theta3))*sin(theta4)) + dtheta5*(cos(theta3)*sin(alpha3)*(cos(alpha1)*cos(alpha2)*cos(theta1)*cos(theta2) - cos(theta1)*sin(alpha1)*sin(alpha2) - cos(alpha2)*sin(theta1)*sin(theta2)) - cos(alpha3)*(-(cos(alpha2)*cos(theta1)*

$sin(alpha1)) - cos(alpha1) * cos(theta1) * cos(theta2) * sin(alpha2) + sin(alpha2) * sin(theta1) * sin(theta2)) - sin(alpha3) * (cos(theta2) * sin(theta1) + cos(alpha1) * cos(theta1) * sin(theta2)) * sin(theta3)) * sin(theta5) + (cos(theta4) * dtheta4 * (cos(alpha3) * cos(theta3) * (cos(alpha1) * cos(alpha2) * cos(theta1) * cos(theta2) - cos(theta1) * sin(alpha1) * sin(alpha2) - cos(alpha2) * sin(theta1) * sin(theta2)) + sin(alpha3) * (-(cos(alpha2) * cos(theta1) * sin(alpha1)) - cos(alpha1) * cos(theta1) * cos(theta2) * sin(alpha2) + sin(alpha2) * sin(theta1) * sin(theta2)) - cos(alpha3) * (cos(theta2) * sin(theta1) + cos(alpha1) * cos(theta1) * sin(theta2)) * sin(theta3)) + cos(theta4) * (cos(theta3) * (cos(theta1) * cos(theta2) * dtheta1 + cos(alpha1) * cos(theta1) * cos(theta2) * dtheta2 - cos(alpha1) * dtheta1 * sin(theta1) * sin(theta2) - dtheta2 * sin(theta1) * sin(theta2)) + (-(cos(alpha1) * cos(alpha2) * cos(theta2) * dtheta1 * sin(theta1)) - cos(alpha2) * cos(theta2) * dtheta2 * sin(theta1) + dtheta1 * sin(alpha1) * sin(alpha2) * sin(theta1) - cos(alpha2) * cos(theta1) * dtheta1 * sin(theta2) - cos(alpha1) * cos(alpha2) * cos(theta1) * dtheta2 * sin(theta2)) * sin(theta3)) - dtheta4 * (cos(theta3) * (cos(theta2) * sin(theta1) + cos(alpha1) * cos(theta1) * sin(theta2)) + (cos(alpha1) * cos(alpha2) * cos(theta1) * cos(theta2) - cos(theta1) * sin(alpha1) * sin(alpha2) - cos(alpha2) * sin(theta1) * sin(theta2)) * sin(theta3)) * sin(theta4) + (cos(alpha3) * cos(theta3) * (-(cos(alpha1) * cos(alpha2) * cos(theta2) * dtheta1 * sin(theta1)) - cos(alpha2) * cos(theta2) * dtheta2 * sin(theta1) + dtheta1 * sin(alpha1) * sin(alpha2) * sin(theta1) - cos(alpha2) * cos(theta1) * dtheta1 * sin(theta2) - cos(alpha1) * cos(alpha2) * cos(theta1) * dtheta2 * sin(theta2)) + sin(alpha3) * (cos(alpha2) * dtheta1 * sin(alpha1) * sin(theta1) + cos(alpha1) * cos(theta2) * dtheta1 * sin(alpha2) * sin(theta1) + cos(theta2) * dtheta2 * sin(alpha2) * sin(theta1) + cos(theta1) * dtheta1 * sin(alpha2) * sin(theta2) + cos(alpha1) * cos(theta1) * dtheta2 * sin(alpha2) * sin(theta2)) - cos(alpha3) * (cos(theta1) * cos(theta2) * dtheta1 + cos(alpha1) * cos(theta1) * cos(theta2) * dtheta2 - cos(alpha1) * dtheta1 * sin(theta1) * sin(theta2) - dtheta2 * sin(theta1) * sin(theta2)) * sin(theta3)) * sin(theta4)) * sin(theta5));`

(.....)

(.....)

$DJ[6, 4] := cos(alpha3) * dtheta2 * sin(alpha1) * sin(alpha2) * sin(theta2) + cos(alpha2) * cos(theta3) * dtheta2 * sin(alpha1) * sin(alpha3) * sin(theta2) + cos(theta2) * dtheta2 * sin(alpha1) * sin(alpha3) * sin(theta3);$

$DJ[6, 5] := cos(theta4) * (-(cos(alpha2) * cos(alpha3) * cos(theta3) * dtheta2 * sin(alpha1) * sin(theta2)) + dtheta2 * sin(alpha1) * sin(alpha2) * sin(alpha3) * sin(theta2) - cos(alpha3) * cos(theta2) * dtheta2 *$

$sin(alpha1) * sin(theta3)) - cos(theta4) * dtheta4 * (cos(theta3) * sin(alpha1) * sin(theta2) + (cos(alpha2) * cos(theta2) * sin(alpha1) + cos(alpha1) * sin(alpha2)) * sin(theta3)) - dtheta4 * (cos(alpha3) * cos(theta3) * (cos(alpha2) * cos(theta2) * sin(alpha1) + cos(alpha1) * sin(alpha2)) + (cos(alpha1) * cos(alpha2) - cos(theta2) * sin(alpha1) * sin(alpha2)) * sin(alpha3) - cos(alpha3) * sin(alpha1) * sin(theta2) * sin(theta3)) * sin(theta4) - (cos(theta2) * cos(theta3) * dtheta2 * sin(alpha1) - cos(alpha2) * dtheta2 * sin(alpha1) * sin(theta2) * sin(theta3)) * sin(theta4);$

$DJ[6, 6] := -(cos(theta5) * (-(cos(alpha3) * dtheta2 * sin(alpha1) * sin(alpha2) * sin(theta2)) - cos(alpha2) * cos(theta3) * dtheta2 * sin(alpha1) * sin(alpha3) * sin(theta2) - cos(theta2) * dtheta2 * sin(alpha1) * sin(alpha3) * sin(theta3))) + cos(theta5) * dtheta5 * (cos(theta4) * (cos(theta3) * sin(alpha1) * sin(theta2) + (cos(alpha2) * cos(theta2) * sin(alpha1) + cos(alpha1) * sin(alpha2)) * sin(theta3)) + (cos(alpha3) * cos(theta3) * (cos(alpha2) * cos(theta2) * sin(alpha1) + cos(alpha1) * sin(alpha2)) + (cos(alpha1) * cos(alpha2) - cos(theta2) * sin(alpha1) * sin(alpha2)) * sin(alpha3) - cos(alpha3) * sin(alpha1) * sin(theta2) * sin(theta3)) * sin(theta4)) + dtheta5 * (-(cos(alpha3) * (cos(alpha1) * cos(alpha2) - cos(theta2) * sin(alpha1) * sin(alpha2))) + cos(theta3) * (cos(alpha2) * cos(theta2) * sin(alpha1) + cos(alpha1) * sin(alpha2)) * sin(alpha3) - sin(alpha1) * sin(alpha3) * sin(theta2) * sin(theta3)) * sin(theta5) + (cos(theta4) * dtheta4 * (cos(alpha3) * cos(theta3) * (cos(alpha2) * cos(theta2) * sin(alpha1) + cos(alpha1) * sin(alpha2)) + (cos(alpha1) * cos(alpha2) - cos(theta2) * sin(alpha1) * sin(alpha2)) * sin(alpha3) - cos(alpha3) * sin(alpha1) * sin(theta2) * sin(theta3)) + cos(theta4) * (cos(theta2) * cos(theta3) * dtheta2 * sin(alpha1) - cos(alpha2) * dtheta2 * sin(alpha1) * sin(theta2) * sin(theta3)) + (-(cos(alpha2) * cos(alpha3) * cos(theta3) * dtheta2 * sin(alpha1) * sin(theta2)) + dtheta2 * sin(alpha1) * sin(alpha2) * sin(alpha3) * sin(theta2) - cos(alpha3) * cos(theta2) * dtheta2 * sin(alpha1) * sin(theta3)) * sin(theta4) - dtheta4 * (cos(theta3) * sin(alpha1) * sin(theta2) + (cos(alpha2) * cos(theta2) * sin(alpha1) + cos(alpha1) * sin(alpha2)) * sin(theta3)) * sin(theta4)) * sin(theta5);$

References

1. Spong, M. and Vidyasagar, W. M.: *Robot Dynamics and Control*, Wiley, 1989.
2. Ho, C. Y. and Sriwattanathamma, J.: *Robot Kinematics, Symbolic Automation and Numerical Synthesis*, Ablex Publishing Corporation.
3. Corke, P. I.: *Robotics Toolkit*, CSIRO, Division of Manufacturing Technology, February 1994.
4. Sobh, T. M., Dekhil, M., Henderson, T. C. and Sabbavarapu, A.: Prototyping a three-link robot manipulator, in: *ASME Press Series on Robotics and Manufacturing; Recent Trends in Research and Applications* 6, 1996, pp. 781–786.

5. Dekhil, M., Sobh, T. M., Henderson, T. C. and Mecklenburg, R.: UPE: Utah prototyping environment for robot manipulators, in: *J. Intelligent Robotic Systems* **17** (1996), 31–60.
6. Dekhil, M., Sobh, T. M. and Henderson, T.: URK: Utah robot kit – A 3-link robot manipulator prototype, in: *Proceedings of the IEEE International Conference on Robotics and Automation*, San Diego, May 1994.
7. Craig, J.: *Introduction to Robotics*, Addison-Wesley, 1989.
8. Herrera-Bendezu, L. G., Mu, E. and Cain, J. T.: Symbolic computation of robot manipulator kinematics, in: *IEEE Int. Conf. Robotics and Automation*, 1988, pp. 1335–1340.
9. Rieseler, H. and Wahl, F. M.: Fast symbolic computation of the inverse kinematics of robots, in: *IEEE Int. Conf. Robotics and Automation*, 1990, pp. 462–467.