Robot inverse kinematics and dynamics algorithms for windows

Article · January 2004

CITATIONS
3

READS
3,111

2 authors, including:

Subir K Saha
Indian Institute of Technology Delhi
129 PUBLICATIONS **1,248** CITATIONS

SEE PROFILE

Published in "Recent Trends in Manufacturing" (Proc. of the Conf. on Adv. and Recent Trends in Manufac., Kalyani Govt. Eng. College, WB, India),Elite Publishing House, New Delhi, Dec., 2003, pp. 229-237.

Robot Inverse Kinematics and Dynamics Algorithms for Windows

P. Marothiya and S. K. Saha

Dept. of Mech. Eng., IIT Delhi, Hauz Khas, New Delhi 110 016

pankajmarothiya@rediffmail.com; saha@mech.iitd.ernet.in

Abstract

Inverse kinematics and dynamics algorithms for serial robots are presented in this paper. Industrial robots controlled by the joint actuators require the values of joint torques and forces. These values depend on the joint trajectories necessary to produce a desired end effector motion. Inverse kinematics converts the Cartesian coordinates of the end effector to corresponding joint coordinates. This can be used in further analysis, for example, to find out the joint torques and forces to move the end effector. This paper presents a C++ inverse kinematics algorithm for wrist-partitioned robots like PUMA 560. Such problem normally has eight solutions. A user can specify one, or one of them is chosen as default for further analysis. Using the joint angles obtained from the inverse kinematics, their derivatives, i.e., joint velocities and accelerations, are obtained, which are then fed to the C++ RIDIM (Recursive Inverse Dynamics for Industrial Manipulators) algorithm developed at IIT Delhi. Thus, a complete inverse kinematics and dynamics algorithm is developed that is interfaced with a VC++ program to make user-friendly algorithms for the Windows users. Such an indigenous tool is very useful for teaching of robotics. Industry users can also use such tool to develop their robot control algorithms.

1. Introduction

Robots find wide usage in industries. For example, they are used in assembly line operations, in hazardous environments, etc. Mostly, the usage could be attributed to some or the other form of `pick and place' operation or `continuous path' operation like welding. In the following paragraphs, firstly, a brief discussion about a robot manipulator is given. After that, one broad view of the steps to accomplish a given task would be given. Once the broad discussion is over, then, further discussion of the theoretical aspects,

concentrated to the scope of this paper would be mentioned. After that, algorithms would be discussed.

Robot Manipulators: Robot manipulators are mechanical systems controlled by electric/hydraulic/pneumatic actuators and an electronic control system with the help of suitable software. The manipulator consists of various links connected through powered joints to give motion to one link with respect to other link. For example, a revolute joint connecting two links provides relative link rotation. The last link of the robot carrying a welding torch or a device to pick an object is termed as the 'End Effecter' (EE), and is the operation point of the manipulator. For manipulators to work in a certain environment, its task is divided into a number of small operations, and further into individual movement of the EE from one configuration, i.e., position and orientation, into another. This traversing from initial to final configuration is done following a path or trajectory, and the operation of making the robot EE to follow a particular trajectory is termed as 'trajectory planning.' Motion of the manipulator is governed by the amount of motion imparted through its actuators. Actuators' input is only in terms of current signals (for electric motors), and the robot moves as the signal is given. Thus, to control the EE what needs to be done is to identify signals to be given to each actuator. The 'correct' set of signals is obtained by considering the kinematics and dynamics of the robot following a suitable path, and the load to be manipulated.

Manipulator Control: In order to control a robot for successfully achieving a task, following aspects need to be considered.

Manipulator Architecture**:** The architecture of the robot, viz., its geometrical aspects; all dimensions, what links are there, should be known. They are defined in terms of Denavit and Hartenberg (DH) parameters (Denavit and Hartenberg, 1955; Saha, 1999).

Trajectory Planning**:** Here the complete robot path is divided into sub-paths. For example, the pick and place operation can be divided into following tasks: a) Move the EE of the robot to the object to be picked; b) Hold the object; c) Take the gripper to the final destination; and e) Release the object. Thus, for each sub-task, there is one set of initial and final configurations. They are typically in world coordinates or in a fixed inertia frame. The desired motion from the initial to the final configuration is specified through a suitable trajectory, say, straight line, a circular arc or a polynomial curve. The important parameters

are the traverse time and the type of trajectory. Depending on the trajectory type, it could move in a jerky or a smooth way. In a way, it is similar to cam design, where cam profile may be designed for motion, acceleration, and jerk continuity. In this paper, cycloidal trajectory (Angeles, 1999) is used for smooth position, velocity and acceleration characteristics, as evident from Fig. 4.

Inverse Dynamics**:** Once the joint level trajectory, i.e., joint positions, velocities and accelerations are calculated using the above steps, the next step is to calculate what forces and torque are required by the joint actuators to move the links. This is known as Inverse Dynamics. Please note that for the dynamics of the manipulator, mass, inertia properties of the robot links and the load to be manipulated are also required as input.

Control and Actuation**:** Once, mechanical aspects are clear, what remains is how to make the actuators deliver what is required. This comes into the domain of 'Control theory.' Here, each type of actuator, its characteristics are taken into account and using these, the 'correct set' of signals is prepared using, say, PID or Fuzzy-logic based control laws.

2. Inverse Kinematics and Dynamics Algorithms

Inverse kinematics and dynamics algorithms required for robot control are introduced in this section. First, kinematics is explained, followed by dynamics.

2.1 Kinematics

Kinematics is the study of motion without regard to the forces causing the motion. Here, position and orientation of the EE is defined in terms of the position and orientation of the robot links. Each link is defined in terms of the DH parameters (Denavit and Hartenberg, 1955; Saha, 1999). Once the DH parameters are defined, we have a set of four values, viz., link length, a_i, link twist, α_i, joint length, b_i, and the joint angle, θ_i, (Fig. 1b), where $i \in [1,n]$---n being the number of degree of freedom of the robot manipulator. Also, a reference frames (F_i) is assigned to each link and the frame transformations are determined by the values of DH parameters involved. Configuration of manipulator is uniquely defined with respect to the fixed inertia frame. Note that the rotation of the frame F_{i+1} is mapped in the frame F_i by the rotation matrix $\mathbf{Q}_i$ and the translation in terms of vector $\mathbf{a}_i$,

(F_{i+1} to F_i) and $\mathbf{b_i}$ (F_i to F_{i+1}). For the 6-link manipulator, there would be six $\mathbf{Q_i}$'s and six $\mathbf{a_i}$s (and $\mathbf{b_i}$ s), and the kinematics equations are written as (Angeles, 1997)

$$\mathbf{Q_1\, Q_2\, Q_3\, Q_4\, Q_5\, Q_6 = Q} \qquad \ldots(1)$$

$$\mathbf{a_1 + Q_1\, a_2 + Q_1\, Q_2\, a_3 + Q_1\, Q_2\, Q_3\, a_4 + Q_1\, Q_2\, Q_3\, Q_4\, a_5 + Q_1\, Q_2\, Q_3\, Q_4\, Q_5\, a_6 = p} \qquad \ldots(2)$$

where $\mathbf{a_i}$, is the 3-dimenstional vector pointing from the origin of frame F_i fixed to link (i-1) to that of frame F_{I+1} fixed to link i, and **p** and **Q** are the 3-dimensitonal position vector and the 3x3 rotation matrix of the EE with respect to the inertia frame, respectively. The robot under study is of wrist-partitioned (WP) type, i.e., the last three joint axes intersect at a point. As a result, the robot's position and orientation problems can be treated separately. Now, if the left hand sides of eqs. (1) and (2) are known from a given set of DH parameters, and the problem is to find the right hand sides, it is called forward kinematics, which has unique solution. However, if the reverse question is asked, i.e., given **p** and **Q**, find the joint angles, θ_i, then it is called inverse kinematics for which the number of solutions is more than one because one has to solve a set of non-linear algebraic equations. For the WP robot, the number of solutions is eight. If the robot has only revolute joints, the positioning and orientation problems are solved as:

Positioning problem: Equation (1) is re-written as

$$\mathbf{c + Q_1\, Q_2\, Q_3\, Q_4\, a_5 + Q_1\, Q_2\, Q_3\, Q_4\, Q_5\, a_6 = p} \qquad \ldots(3)$$

where, $\mathbf{c \equiv a_1 + Q_1\, a_2 + Q_1\, Q_2\, a_3 + Q_1\, Q_2\, Q_3\, a_4}$. The interpretation of **c** is that it is the position vector of the point where the last three axes of the manipulator intersect. The expression of vector **c** after eq.(3) can be re-written as

$$\mathbf{a_2 + Q_2\, a_3 + Q_2\, Q_3\, a_4 = Q_1^T (c - a_1)} \qquad \ldots(4)$$

Since the robot is WP, it can be shown that the vector, $\mathbf{a_5 = 0}$. Thus, eq.(3) becomes

$$\mathbf{c = p - Q_1\, Q_2\, Q_3\, Q_4\, Q_5\, a_6} \text{ or } \mathbf{c = p - Q_1\, Q_2\, Q_3\, Q_4\, Q_5\, Q_6\, Q_6^T\, a_6}$$

where, $\mathbf{Q_6\, Q_6^T = 1}$, due to the orthogonal property of the orientation matrices. Putting $\mathbf{b_6 \equiv Q_6^T\, a_6}$, the final equation for the position inverse kinematics is obtained as

$$\mathbf{c = p - Qb_6} \qquad \ldots(5)$$

where the RHS is completely known. Positioning problem reduces to the solution of eq. (4) after substituting **c** from eq. (5), i.e., to find the values of first three joints, θ_1, θ_2 and θ_3. The steps are described in Angeles (1997), where Bi-quadratic equation in θ_3 is solved

using Ferrari's method (Uspensky, 1948), followed by θ_1. Finally, θ_1 and θ_3 are substituted in the kinematics equations to get θ_2. There will be a set of maximum 4 possible solutions.

Orientation Problem: In the positioning problem, θ_1, θ_2 and θ_3 are solved, hence, $\mathbf{Q_1}$ $\mathbf{Q_2}$ $\mathbf{Q_3}$ are known. These values are used to solve the orientation problem. Equation (1) is rewritten as

$$\mathbf{Q_4\, Q_5\, Q_6 = R}, \qquad \text{where } \mathbf{R \equiv Q(Q_1\, Q_2\, Q_3)^T} \qquad \ldots(6)$$

For the orientation inverse kinematics problem, the EE orientation, **Q,** is known, and from the position solutions, the orientation of F_4 is also known. Thus the end effector's axis in Frame 4 is known. Using this, and knowing the link twists, α_i, (i=4,5,6), a quadratic equation is arrived for θ_4. Corresponding to each value of θ_4, values of θ_5 and θ_6 are calculated. Thus, for each 'position' solution, a maximum of two solutions are possible for the orientation problem. Thus, there are maximum of 8 solutions are possible for a given inverse kinematics problem. A C++ program calculates all the possible solutions.

2.2 Dynamics

For an n-degree of freedom (dof) serial-robot (Fig. 1a), if m_i is the mass of the i^{th} link and $\mathbf{I_i}$ denotes the 3×3 inertia tensor of the i^{th} link about its mass center, C_i, then the uncoupled Newton-Euler equations (NE) governing the motion of the i^{th} link are written as

$$\mathbf{M_i\, \dot{t}_i + W_i M_i t_i = w_i} \qquad \ldots(7a)$$

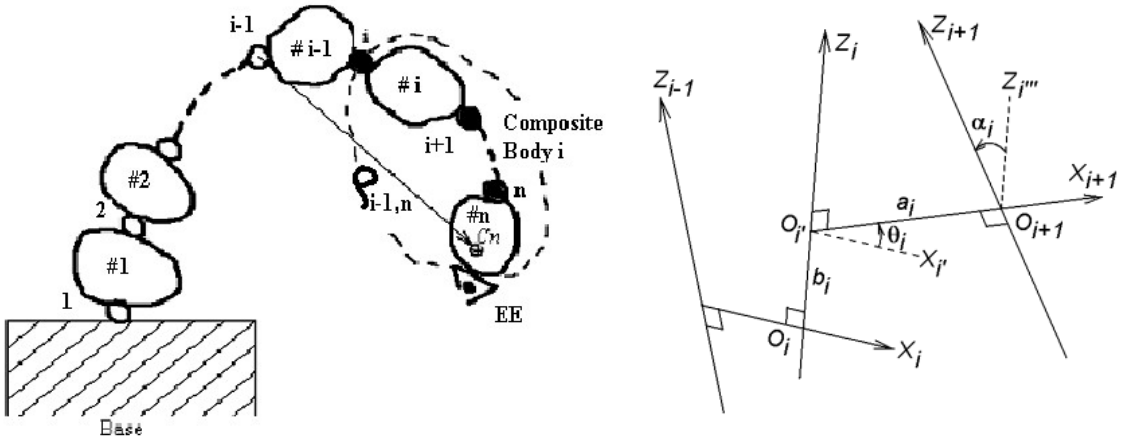
where the 6×6 mass matrix, $\mathbf{M_i}$, and the 6×6 angular velocity matrix, $\mathbf{W_i}$, for the i^{th} link are given as (Saha, 1999)

$$\mathbf{M_i} \equiv \begin{bmatrix} \mathbf{I_i} & \mathbf{O} \\ \mathbf{O} & \mathbf{m_i 1} \end{bmatrix}; \text{ and } \mathbf{W_i} \equiv \begin{bmatrix} \boldsymbol{\omega}_\mathbf{i} \times \mathbf{1} & \mathbf{O} \\ \mathbf{O} & \mathbf{O} \end{bmatrix} \qquad \ldots(7b)$$

in which $\mathbf{I_i}$ is the 3×3 inertia tensor about the mass centre of the i^{th} body, C_i, and $\boldsymbol{\omega}_\mathbf{i}\times\mathbf{1}$ is the 3×3 cross-product tensor associated with the angular velocity vector, $\boldsymbol{\omega}_\mathbf{i}$,. Note that **1** and **O** in eq. (7b) are the 3×3 identity and zero matrices, respectively. The twist and wrench vectors, $\mathbf{t_i}$ and $\mathbf{w_i}$, are then defined as

$$\mathbf{t_i} = \begin{bmatrix} \boldsymbol{\omega}_\mathbf{i} \\ \mathbf{v_i} \end{bmatrix} \quad \text{and} \quad \mathbf{w_i} = \begin{bmatrix} \mathbf{n_i} \\ \mathbf{f_i} \end{bmatrix} \qquad \ldots(8)$$

where $\boldsymbol{\omega}_i$ and $\mathbf{v}_i$ are the 3-dimensional vectors of angular velocity and the linear velocity of the mass center, C_i, of the i^{th} body, respectively.



(a) (b)

1, 2, …,n : Joints; #1, #2, …#n : Links

Fig. 1: (a) An n-link manipulator**;** (b) DH notations

Moreover, $\mathbf{n}_i$ and $\mathbf{f}_i$ are the 3-dimensional vectors denoting the resultant moment about C_i, and the resultant forces acting at C_i, respectively. Note that $\dot{\mathbf{t}}_i$ of eq. (7a) is the time derivative of the twist vector. Equation (7a) when written for all 'n' links, i.e., i = 1,…, n, it can be expressed in a compact form as

$$\mathbf{M}\,\dot{\mathbf{t}} + \mathbf{W}\mathbf{M}\mathbf{t} = \mathbf{w} \qquad \ldots(9)$$

where **M** and **W** are the 6n×6n generalized mass matrix and the generalized matrix of the angular velocities, respectively. They are defined as

$$\mathbf{M} \equiv diag.[\mathbf{M}_1,\ldots,\mathbf{M}_n]; \text{ and } \mathbf{W} \equiv diag.[\mathbf{W}_1,\ldots,\mathbf{W}_n] \qquad \ldots (10)$$

Moreover, the 6n-dimensional vectors of generalized twist and wrench are defined as

$$\mathbf{t} \equiv [\mathbf{t}_1^T, \mathbf{t}_2^T,\ldots, \mathbf{t}_n^T]^T; \; \mathbf{w} \equiv [\mathbf{w}_1^T, \mathbf{w}_2^T,\ldots, \mathbf{w}_n^T]^T \qquad \ldots(11)$$

It is pointed out here that the generalized twist, **t**, can be expressed as a linear transformation of the n independent joint rates, $\dot{\boldsymbol{\theta}} \equiv [\dot{\theta}_1, \ldots, \dot{\theta}_n]^T$ --- θ_i being the i^{th} joint displacement---, i.e.,

$$\mathbf{t} = \mathbf{T}\,\dot{\boldsymbol{\theta}}, \quad \text{where } \mathbf{T} \equiv \mathbf{T}_l\,\mathbf{T}_d \qquad \ldots(12)$$

T being the 6n×n Natural Orthogonal Complement (NOC) matrix (Angeles and Lee, 1988), whereas the 6n×6n and 6n×n matrices, $\mathbf{T}_l$ and $\mathbf{T}_d$, respectively, are the Decoupled

NOC (DeNOC) matrices (Saha, 1999). Pre-multiplying eq. (9) with the transpose of the NOC matrix, i.e., $\mathbf{T}^{\mathbf{T}}$, one gets n independent dynamic equations of motion of the coupled system, namely,

$$\mathbf{T}^{\mathbf{T}}\ (\mathbf{M}\,\dot{\mathbf{t}} + \mathbf{WMEt}) = \mathbf{T}^{\mathbf{T}}\ (\mathbf{w}^{\mathbf{E}} + \mathbf{w}^{\mathbf{C}}) \qquad \ldots(13)$$

where $\mathbf{w}$ is substituted by $\mathbf{w} \equiv \mathbf{w}^{\mathbf{E}} + \mathbf{w}^{\mathbf{C}}$, $\mathbf{w}^{\mathbf{E}}$ and $\mathbf{w}^{\mathbf{C}}$ being the 6n-dimensional vectors of external and constraint wrenches, respectively. The term $\mathbf{T}^{\mathbf{T}}\mathbf{w}^{\mathbf{C}}$ in eq. (13) vanishes, as the constraint wrench produces no work. Substitution of the expression of $\mathbf{T} \equiv \mathbf{T_l T_d}$ from eq. (12), and its time derivative, $\dot{\mathbf{T}} = \mathbf{T_l}\dot{\mathbf{T}}_{\mathbf{d}} + \dot{\mathbf{T}}_{\mathbf{l}}\mathbf{T_d}$, into eq. (13) results in the following form of the dynamic equations of motion:

$$\mathbf{I}\,\ddot{\theta} + \mathbf{C}\,\dot{\theta} = \tau \qquad \ldots (14)$$

which is nothing but the Euler-Lagrange equations of motion (Angeles and Lee, 1988). In eq. (14), expressions for the each element of the n×n generalized inertia matrix (GIM), $\mathbf{I}$ and, the matrix of convective inertia (MCI), $\mathbf{C}$, are written as (Saha, 1999)

$$i_{ij} \equiv \mathbf{p}_{\mathbf{i}}^{\mathbf{T}}\,\tilde{\mathbf{M}}_{\mathbf{i}}\mathbf{B}_{\mathbf{ij}}\mathbf{p}_{\mathbf{j}} \qquad \ldots(15a)$$

$$c_{ij} \equiv \mathbf{p}_{\mathbf{i}}^{\mathbf{T}}(\mathbf{A}_{\mathbf{ji}}^{\mathbf{T}}\tilde{\mathbf{M}}_{\mathbf{j}}\mathbf{W}_{\mathbf{j}} + \mathbf{B}_{\mathbf{j+1,i}}^{\mathbf{T}}\tilde{\mathbf{H}}_{\mathbf{j+1,j}} + \mathbf{B}_{\mathbf{ji}}^{\mathbf{T}}\dot{\tilde{\mathbf{M}}}_{\mathbf{j}})\mathbf{p}_{\mathbf{j}} \qquad \text{if} \quad i \le j \qquad \ldots(15b)$$

$$c_{ij} \equiv \mathbf{p}_{\mathbf{i}}^{\mathbf{T}}(\tilde{\mathbf{M}}_{\mathbf{i}}\mathbf{B}_{\mathbf{ij}}\mathbf{W}_{\mathbf{j}} + \tilde{\mathbf{H}}_{\mathbf{ij}} + \dot{\tilde{\mathbf{M}}}_{\mathbf{i}}\mathbf{B}_{\mathbf{ij}})\mathbf{p}_{\mathbf{j}} \qquad \text{otherwise}$$

where, the 6×6 matrix $\mathbf{B_{ij}}$ and the 6-dimensional vector $\mathbf{p_i}$ are the block elements of the DeNOC matrices, $\mathbf{T_l}$ and $\mathbf{T_d}$, respectively, (Saha, 1999; 2001). Vector $\boldsymbol{\tau}$ contains the required joint torques and forces. Based on eq.(14), a recursive inverse dynamics algorithm is developed in Saha (1999), which is coded in C++ called as Recursive Inverse Dynamics of Industrial Manipulators (RIDIM) (Saha, 2001).

3. Algorithms: An Illustration with PUMA 560

A VC++ interface program is developed for the Windows users to combine two C++ programs, one to solve for the inverse kinematics program presented in this paper and the other one is the existing RIDIM program. The program inputs the DH parameters of the robot manipulator and stores it in a file. Then, it asks for the initial and the final configurations of the robot. This input could be fed in joint level coordinates as well as in

Cartesian coordinates. Other input required by the program are: time of moving from one initial to the final specified position, and number of steps. Based on these input, the program calculates the trajectory to be followed and stores it in another file. Then, there is an option for performing the inverse dynamics using RIDIM for the calculated trajectory, whose output (joint torques/forces) is also stored in a file. The results stored in the files are available for plotting or other analyses. What follows next is an illustration with the PUMA 560 robot shown in Fig. 3, whose DH parameters are given in Table 1.

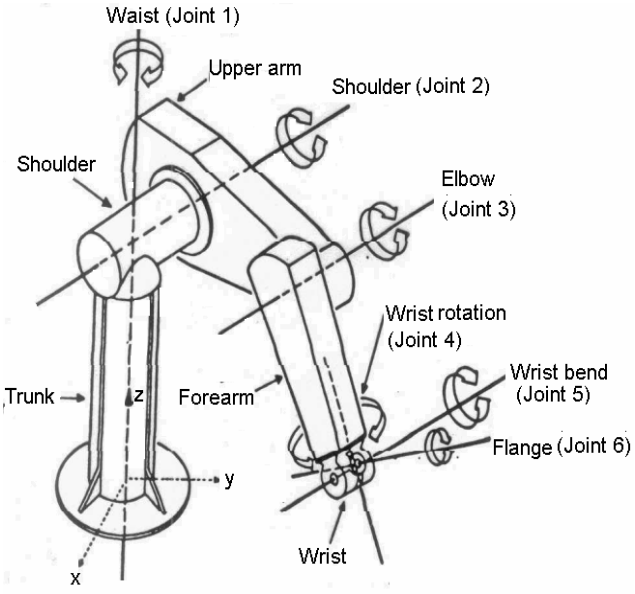


Fig. 3 PUMA 560 Robot

Table 1 DH parameters of PUMA 560

Link	a(m)	b(m)	α(deg)	θ(deg)
1	0	0	-90	θ_1
2	0.432	0.149	0	θ_2
3	0.02	0	90	θ_3
4	0	0.432	-90	θ_4
5	0	0	90	θ_5
6	0	0.056	0	θ_6

To use the inverse kinematics program the following input is use:

p= [0.67837 0.24867 0.07662]^T

Q= [-0.84086 0.25393 0.47799; 0.54115 0.37712 0.75163; 0.01060 0.89067 -0.45452]

where, **p** and **Q** are the EE position and the orientation respectively, which are taken from the output of a forward kinematics program when joint angles are, $\boldsymbol{\theta}$=[5 30 10 45 90 60]^T degrees. This means that one of the solutions from the inverse kinematics program should match with the above values. Thus, the inverse kinematics algorithm will be validated. The output of the inverse kinematics program is as follows:

The solutions are:

Sol no.: 1 5 30 10 45 90.0002 60.0001

Sol no.: 2 -149.82 -132.601 10 -83.9618 24.3137 -19.1774

Sol no.: 3 5 -47.3984 164.698 77.6014 46.3851 -12.3228

Sol no.: 4 -149.82 150 164.698 -155.036 75.9605 70.9998

Sol no.: 5 5 30 10 -135 -90.0002 -120

Sol no.: 6 -149.82 -132.601 10 96.038 -24.3137 160.822

Sol no.: 7 5 -47.3984 164.698 -102.399 -46.3851 167.677

Sol no.: 8 -149.82 150 164.698 24.9645 -75.9605 -109

Enter the solution no. to check :(press any key for default, i.e., 1) :

Now, the user is asked to select a particular solution or a default is chosen. Note that the solution number 1 (indicated above as 'Sol no.: 1') is the one that corresponds to the input values. Next, to demonstrate the RIDIM, the position, velocity and acceleration trajectory are generated based on the inverse kinematics solutions. For example, let us use the following values: Initial conditions are θ_i (0)=0 for i=1,…, 6, and θ_i (T) = 180^o for i=1, …, 6, and T= 10 sec. The trajectories are shown in Fig. 4a-c, and the inverse dynamics results for the PUMA 560 (Fig. 3) are plotted in Fig 5 a-f.

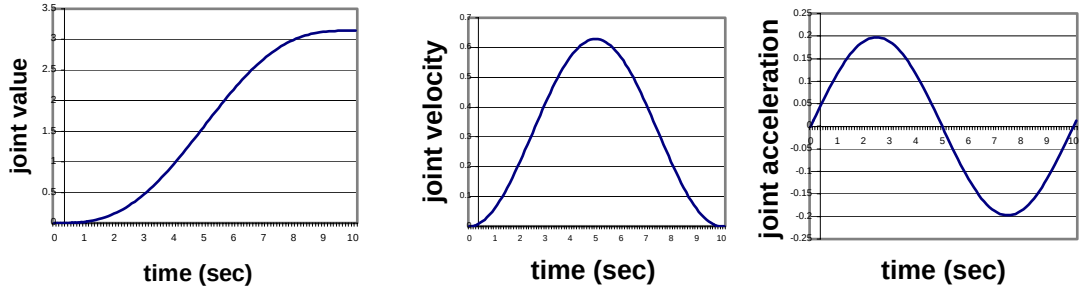


Fig. 4 Joint trajectory (a) Position, (b) velocity and (c) Acceleration.

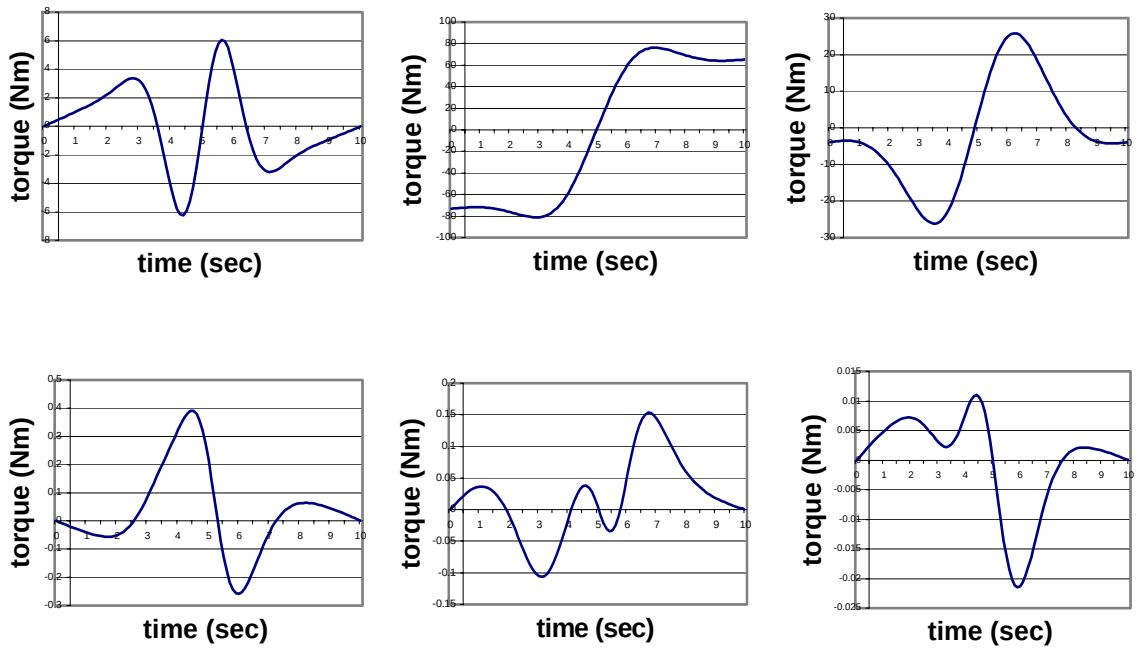


Fig. 5 Torque values at joint (a) 1; (b) 2; (c) 3; (d) 4; (e) 5; (f) 6.

4. Conclusions

Inverse kinematics and dynamics algorithms for serial robots are developed in C++, which are interfaced using a VC++ program, which made the programs user-friendly for Windows users. Such an indigenous tool is very useful for teaching of robotics, and for industries to develop robot control algorithms. The programs are available free of cost to the Indian academic institutes and the R&D Centers.

References

1. Angeles, J., 1997, *Fundamentals of Robotic Mechanical Systems*, Springer-Verlag, Berlin.
2. Angeles, J., and Lee, S.K. 1988, "The formulation of dynamical equations of holonomic mechanical systems using a natural orthogonal compliment," *ASME J. of Appl. Mech.*, **55**, pp. 243-244.
3. Denavit, J., and Hartenberg, R.S., 1955, "A kinematic notation for lower-pair mechanisms based on matrices," *ASME J. of Appl. Mech.*, **77**, pp. 215-221.
4. Saha, S. K., 1999, "Dynamics of serial multibody systems using the Decoupled Natural Orthogonal Component matrices", *ASME J. of Appl. Mech*., vol 66, pp. 986-996.
5. Saha, S.K., 2001, ``An Inverse Dynamics Software for Serial Robots, *Proc. of the 10th NaCoMM* 2001, IIT Kharagpur, Dec. 21-22, pp.131--138.
6. Uspensky, J.V., 1948, *Theory of Equations*, McGraw-Hill, New York.