A GENERALIZED NEWTON-EULER ALGORITHM FOR DYNAMIC SIMULATION OF ROBOT-MANIPULATORS WITH REVOLUTE JOINTS

Bozhidar GRIGOROV
Technical University of Sofia, Bulgaria

Abstract. This article describes a procedure used to compose a set of second order differential equations in order to simulate dynamics of a multi-body manipulator's arm with revolution joints. The procedure is based on d'Alambert's principle and Newton-Euler equations, and considers each of the manipulator's links as in equilibrium under a multitude of inertial, gravitational and driving forces in the process of its spatial motion. The method is straightforward and is based on the repeatability of mathematical operations. The derived set of differential equation are easy to program and could be integrated using standard routines. They also can be used to solve forward, as well as reverse dynamic problems. The algorithm is verified by comparative studies using well known simulation packages as well as experimental data. It can be very successfully applied on the stage of design when not all dynamic parameters of the new device are sufficiently known.

Keywords: dynamic simulation, algorithm, Newton-Euler, robots, revolute joints

1. Introduction

Robot manipulators are complex mechanisms, and their dynamic modeling is always a delicate matter, usually serving two main purposes. First: to investigate the behavior of the mechanical system under the influence of a multitude of different forces (including the driving ones) applied to the links, and second: to help develop control algorithms for desired motion of the robot arm through space. There are two basic problems associated with dynamics of robotic mechanical systems, which are solved in order to serve the above mentioned purposes – the forward and the inverse problems.

Simply stated, the solving the forward problem means to derive time-depended functions for the joint variables given the driving force vector, gravitational and other forces acting upon the links. In the inverse problem, a time-history of either the Cartesian or the joint coordinates is given, and based on these histories, architecture and inertial parameters of the system, torque or force requirements at the different actuated joints are computed. The latter is essential for the computed-torque control of robotic manipulators, while the former is required for the simulation as well as for the real-time feedback control.

The study of the multiple rigid bodies systems dynamics is a classical task and has long-standing history. During a period of more than thirty years considerable research has been done, leading to an abundance of theoretical approaches based on different methods - D'Alambert's principle and Newton-Euler equations, Euler-Lagrange equations, etc. The application of computers has been a major break-through as it allows numerical solutions of highly non-linear problems and thus much more precise formulation of the dynamic simulation tasks for large number of articulated bodies.

Among the pioneers in this field Uicker [1, 2] and then Kahn [3] produced a method based on the Euler-Lagrange equations of rigid bodies mechanical systems, which method is used to simulate the dynamical behavior of such systems. Hollerbach [4] developed efficient Lagrangian formulation based on the recurrence relations for the velocities, accelerations, and generalized forces. Later, Armstrong [5] elaborated an recursive O(N) algorithm for mechanisms including such with spherical joints. Further on Li [6, 7] proposed a new Lagrangian formulation of dynamics for manipulators which results in well structured form equations of motion, making it possible to realize the computation of the robot manipulators dynamics in real-time on a micro/mini-computer.

Early researchers developing inverse dynamics algorithms for robotics used a Newton-Euler formulation of the problem. Further on Luh, Walker, and Paul [8] developed a very efficient recursive algorithm (RNEA) that is applicable to systems with serial kinematic chains. Fiesette at al. [9] offered a fully symbolic generation of multi-body models to deal with forward and reverse problems for open and closed loop kinematic chains and large number of bodies.

One of the most popular and cited constraint based dynamics algorithms in use today is known as the Featherstone algorithm. Originally developed for use in robotics simulations, the algorithm has become quite popular in physical simulations as it is capable of simulating the motion of articulated bodies in O(N) time, when the computation required scales linearly with the number of links in the articulated body. Featherstone [10, 11] realized that the efficiency of the simulation procedures was directly related to the efficiency of computing the joint space inertia (mass) matrix. He used efficient transformations and link coordinates to reduce the computation of the inertia matrix by about 30 %.

Today, the problems of robots dynamics are subject of many books and monographs utilizing different approaches while solving different problems. Looking through the multitude of publications one can easily recognizes several principles, observations and directions of research:

- Formulations based on Newton-Euler or Euler-Lagrange principles are equal when applied to the multibody dynamic simulation problems. It was first done by Silver [12] who showed how to derive the Euler-Lagrange equation out of Newton-Euler equations;
- The reverse dynamic problem is solved primarily using recursive Newton-Euler formulation [11, 13, 14].
- The forwards problem is generally dealt by integration of a set of second order differential equations derived by means usually of Euler-Lagrange equations.
- Considerable effort is made in order to optimize and reduce the number of computations (additions and multiplications) in order to work out optimum algorithms, which lead to a variety of approaches and interpretations. This is closely connected with the necessities of real-time robots control.

In any case, deriving the set of differential equations solving the forward problem for the needs of simulation is tedious process prone to errors, and its complexity increases sharply with the number of degrees of freedom, while on many occasions the algorithms remains confusing and difficult to implement for many who are not specialists in the area. On the other hand, using two different approaches to solve the forward and reverse dynamic problems requires additional efforts and programming, which not always is quite convenient.

The aim of this article is to present a straightforward, engineering approach to the problem of dynamic simulation of articulated arms using Newton-Euler equation and D'Alambert's principle. It is based on the repeatability of mathematical actions and presumes that all the terms at the right side of the derived set of differential equations should be numerically obtained given some initial values. This solution is easy to compose, and is intended to solve the forward as well as the reverse problems of manipulator's dynamics for the purpose of simulation. The algorithm is easy to realize in any mathematical package environment or other means of programming, just following the derived equations.

2. Method

In this article an open-link kinematic chain consisting of *n* links connected by revolute joints with one degree of freedom each is considered. The links are numbered, starting from unmovable base, and local frames with the same number are attached to each link according to the classical Denavit-Hartenberg notation [15]. In this case, for every two adjacent links, a 4×4 transformation matrix ${}^{j}T_{j+1}$ is used to map positional vectors defined in {*j*+1}-th frame in respect to the frame {*j*}, or:

$$ {}^{j}T_{j+1} = \begin{bmatrix} {}^{j}R_{j+1} & \mathbf{P}_{j+1} \\ 0\,0\,0 & 1 \end{bmatrix} . \qquad (1)$$

In the above formula ${}^{j}R_{j+1}$ represents 3×3 rotational matrix describing orientation between the frames, while $\mathbf{P}_{j+1}$ is a 3×1 vector of the {*j*+1}-th frame origin described in {*j*}-th frame. Using the Denavit-Hartenberg's notation with a single DoF for each joint also presumes that all relative motions between the articulated bodies are realized along the $\hat{\mathbf{z}}$ axes of the link coordinate systems. Thus, for the joint variables using vector-columns representation should be written:

$$\theta_j = \begin{bmatrix} 0 \\ 0 \\ \theta_j \end{bmatrix} \quad \dot{\theta}_j = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_j \end{bmatrix} \quad \ddot{\theta}_j = \begin{bmatrix} 0 \\ 0 \\ \ddot{\theta}_j \end{bmatrix} . \qquad (2)$$

The links are set in motion via actuators which exert a time or position dependent vector torque $\tau_j(t) = [0\ \ 0\ \ \tau_j]^T$ at each joint.

In order to derive the generalized Newton-Euler algorithm each link *j* is regarded as in static equilibrium, at any moment of its spatial motion, under a set of all forces and moments acting on it (inertial, gravity, as well as actuator torque). Here should be added all the forces and moments "propagated" to the link under consideration from “outboard” arms, those with greater successive numbers ($j < i \le n$) as shown in Figure 1. These links can be then considered as a rigid system taking into account the fact that they are in the same state of static equilibrium, and all components of the force and moment vectors are resisted by the structure of the manipulator itself, as well as by the actuator torques at the joints.

Summing all the moments at the origin of the {*j*}-th frame (according to D'Alambert's principle) leads to the following equation:

$$-\mathbf{N}_{\Phi j} - \mathbf{N}_{j} - \sum_{i=j+1}^{n}({}^{j}\mathbf{N}_{\Phi i} + {}^{j}\mathbf{N}_{i}) + \mathbf{N}_{Gj} + \sum_{i=j+1}^{n}{}^{j}\mathbf{N}_{Gi} + {}^{j}\mathbf{N}_{Q} + \boldsymbol{\tau}_{j}(t) = 0 \quad (3)$$

where:

$\mathbf{N}_{\Phi j}$ - a moment of the inertial force acting on the link at the centre of gravity due to the linear acceleration of the link;

$\mathbf{N}_{j}$ - an inertial torque acting on the link due to the angular acceleration;

$\mathbf{N}_{Gj}$ - a moment of the gravity force (the weight of the link) acting at the mass centre;

${}^{j}\mathbf{N}_{\Phi I}$, ${}^{j}\mathbf{N}_{i}$, ${}^{j}\mathbf{N}_{Gi}$ - moments of inertial force, inertial torque, and moment of the gravity force; acting on link *i* and expressed in terms of frame {*j*} ($j < i \le n$);

${}^{j}\mathbf{N}_{Q}$ - an external moment (of torque(s) or force(s)) acting upon the end effector (*n*-th link) during the robot function and expressed in respect to frame {*j*};

$\boldsymbol{\tau}_{j}(t)$ - (usually time or position dependent) joint torque exerted by the actuator at the joint *j*.

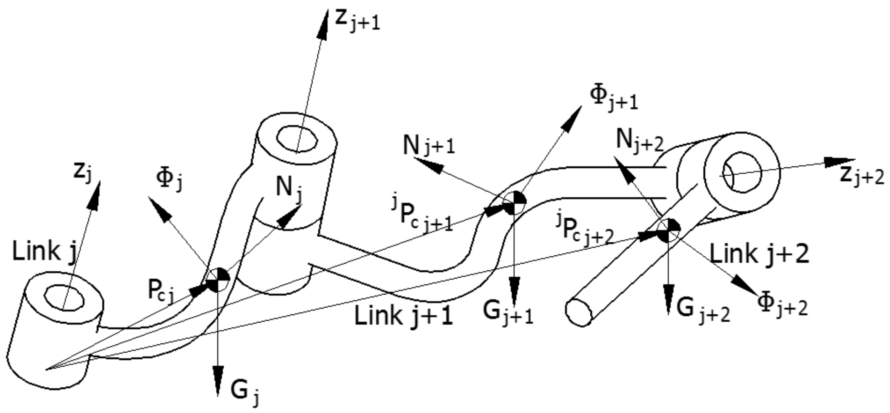

Figure 1. Local frames and forces acting on links

In order to find the inertial force and torque acting at the centre of gravity of each link, Newton-Euler equations are applied [14, 16]:

$$\boldsymbol{\Phi}_{j} = m_{j}.\dot{\mathbf{v}}_{Cj}\,;\ \mathbf{N}_{j} = I_{j}.\dot{\boldsymbol{\omega}}_{j} + \boldsymbol{\omega}_{j} \times I_{j}.\boldsymbol{\omega}_{j}\,, \quad (4)$$

which allows equation (3) to be rewritten in the form:

$$-\sum_{i=j}^{n} m_{j}\,{}^{j}\mathbf{P}_{Ci}\times{}^{j}\mathrm{R}_{i}.\dot{\mathbf{v}}_{ci} - \sum_{i=j}^{n} I_{i}.{}^{j}\mathrm{R}_{i}.\dot{\boldsymbol{\omega}}_{i} - \sum_{i=j}^{n}{}^{j}\mathrm{R}_{i}.(\boldsymbol{\omega}_{i}\times I_{i}.\boldsymbol{\omega}_{i}) + \sum_{i=j}^{n}{}^{j}\mathbf{P}_{Ci}\times{}^{j}\mathrm{R}_{0}.\mathbf{G}_{i} + {}^{j}\mathbf{N}_{Q} + \boldsymbol{\tau}_{j}(t) = 0 \quad (5)$$

In the above formulas:

$\dot{\mathbf{v}}_{Cj}$ - linear acceleration of the mass centre of link *j*;

$\boldsymbol{\omega}_{j}, \dot{\boldsymbol{\omega}}_{j}$ - angular velocity and accelerations of frame {*j*} expressed in respect to the same frame;

m_{j}, I_{j} - are the mass as well as the symmetric inertia tensor of link *j* written in respect to a frame which origin is located at the centre of mass and has the same orientation as link frame {*j*};

${}^{j}\mathbf{P}_{Ci}$ - position of the mass centre of link *i* expressed in terms of {*j*}-th frame;

${}^{j}\mathrm{R}_{i}$ - rotational matrix describing the orientation of frame {*i*} in respect to the frame {*j*};

The gravity force, acting on each link, is considered as described easily in some basic, unmovable frame {0}. Assuming the $\hat{\mathbf{z}}$ coordinate of such a frame pointing vertically up, which often is the usual case, the gravity vector-force could be designated as: $\mathbf{G}_{j} = [0 \quad 0 \quad -m_{j}{\cdot}g]^{T}$ where *g* is the earth acceleration.

Further on, expressions to calculate linear and angular velocities and accelerations of each link must be derived. Taking the advantage of manipulator structure as a chain of bodies, each one capable of motion relative to its neighbors, computing the velocities and accelerations $\boldsymbol{\omega}_{j}$, $\dot{\boldsymbol{\omega}}_{j}$, $\mathbf{v}_{j}$ and $\dot{\mathbf{v}}_{j}$ of each link is done starting from the base [16]. Or for a chain with revolute joints the following recursive dependences could be written:

$$\begin{aligned}
&\boldsymbol{\omega}_j = {}^{j}R_{j\text{-}1}.\boldsymbol{\omega}_{j\text{-}1} + \dot{\boldsymbol{\theta}}_j \\
&\dot{\boldsymbol{\omega}}_j = {}^{j}R_{j\text{-}1}.\dot{\boldsymbol{\omega}}_{j\text{-}1} + {}^{j}R_{j\text{-}1}.\boldsymbol{\omega}_{j\text{-}1} \times \dot{\boldsymbol{\theta}}_j + \ddot{\boldsymbol{\theta}}_j \\
&\mathbf{v}_j = {}^{j}R_{j\text{-}1}.(\mathbf{v}_{j\text{-}1} + \boldsymbol{\omega}_{j\text{-}1} \times \mathbf{P}_j) \qquad (6)\\
&\dot{\mathbf{v}}_j = {}^{j}R_{j\text{-}1}.[\dot{\boldsymbol{\omega}}_{j\text{-}1} \times \mathbf{P}_j + \boldsymbol{\omega}_{j\text{-}1} \times (\boldsymbol{\omega}_{j\text{-}1} \times \mathbf{P}_j) + \dot{\mathbf{v}}_{j\text{-}1}] \\
&\dot{\mathbf{v}}_{Cj} = \dot{\boldsymbol{\omega}}_j \times \mathbf{P}_{Cj} + \boldsymbol{\omega}_j \times (\boldsymbol{\omega}_j \times \mathbf{P}_{Cj}) + \dot{\mathbf{v}}_j
\end{aligned}$$

Here position, velocity and acceleration at joint *j*: $\boldsymbol{\theta}_j, \dot{\boldsymbol{\theta}}_j$ and $\ddot{\boldsymbol{\theta}}_j$ represent 3×1 column vectors as shown by (2).

Initiating from the frame {0} which has velocities and accelerations equal to zero, and substituting $\dot{\mathbf{v}}_j$, $\dot{\mathbf{v}}_{Cj}$, $\boldsymbol{\omega}_j$ and $\dot{\boldsymbol{\omega}}_j$ in formulas (6), for each link the equations given bellow are valid:

$$\begin{aligned}
&\boldsymbol{\omega}_j = \sum_{i=1}^{j} {}^{j}R_i.\dot{\boldsymbol{\theta}}_j \\
&\dot{\boldsymbol{\omega}}_j = \sum_{i=1}^{j} {}^{j}R_i.\ddot{\boldsymbol{\theta}}_i + \mathbf{C}_{\varepsilon j} \\
&\dot{\mathbf{v}}_j = \sum_{i=1}^{j\text{-}1} ({}^{j}R_i.\ddot{\boldsymbol{\theta}}_i \times \text{-}{}^{j}P_i) + \mathbf{C}_{Vj} \qquad (7)\\
&\dot{\mathbf{v}}_{Cj} = \sum_{i=1}^{j} [{}^{j}R_i.\ddot{\boldsymbol{\theta}}_i \times (\text{-}{}^{j}\mathbf{P}_i + \mathbf{P}_{Cj})] + \mathbf{C}_{VCj}
\end{aligned}$$

As it is clear from (7), the aim is to separate the members of (5) which do not contain angular accelerations at joins. For *j* > 1 those members are given as:

$$\begin{aligned}
&\mathbf{C}_{\varepsilon j} = {}^{j}R_{j-1}.\mathbf{C}_{\varepsilon j\text{-}1} + {}^{j}R_{j-1}.\boldsymbol{\omega}_{j-1} \times \dot{\boldsymbol{\theta}}_j \\
&\mathbf{C}_{Vj} = {}^{j}R_{j-1}.[\mathbf{C}_{\varepsilon\, j\text{-}1} \times \mathbf{P}_j + \boldsymbol{\omega}_{j-1} \times (\boldsymbol{\omega}_{j-1} \times \mathbf{P}_j) + \\
&\qquad \mathbf{C}_{Vj\text{-}1}] \qquad (8)\\
&\mathbf{C}_{VCj} = \mathbf{C}_{\varepsilon j} \times \mathbf{P}_{Cj} + \boldsymbol{\omega}_j \times (\boldsymbol{\omega}_j \times \mathbf{P}_{Cj}) + \mathbf{C}_{Vj}
\end{aligned}$$

Next step is to derive the set of second order differential equations in the form similar first to that, proposed by Vukobratovic and Potkonjak [17]:

$$M(\boldsymbol{\theta}).\ddot{\boldsymbol{\theta}} = \mathbf{G}(\boldsymbol{\theta}) - H(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) + \boldsymbol{\tau}(t), \qquad (9)$$

which can be easily integrated. Substituting (7) and (8) in (5), getting across to the left side of the equation all the members containing $\ddot{\boldsymbol{\theta}}_j$, and noticing that:

$${}^{j}R_i.\ddot{\boldsymbol{\theta}}_i \times (\text{-}{}^{j}\mathbf{P}_i + \mathbf{P}_{Cj}) = ({}^{j}\mathbf{P}_i - \mathbf{P}_{Cj}) \times {}^{j}R_i.\ddot{\boldsymbol{\theta}}_i,$$

due to the skew-symmetric property of the vector product, for the link number *j* the equation of the equilibrium will be:

$$\begin{aligned}
&\sum_{k=1}^{n}\sum_{i=l}^{n} m_i.{}^{j}\mathbf{P}_{Ci} \times {}^{j}R_i.[({}^{i}\mathbf{P}_k - \mathbf{P}_{Ci}) \times {}^{i}R_k.\ddot{\boldsymbol{\theta}}_k] + \\
&\sum_{k=1}^{n}\sum_{i=l}^{n} {}^{j}R_i.I_i.{}^{i}R_k.\ddot{\boldsymbol{\theta}}_k = \sum_{i=j}^{n} {}^{j}\mathbf{P}_{Ci} \times {}^{j}R_0.\mathbf{G}_i \text{ -} \\
&\sum_{i=j}^{n} {}^{j}R_i.(\boldsymbol{\omega}_i \times I_i.\boldsymbol{\omega}_i) \text{ -} \sum_{i=j}^{n} {}^{j}R_i.I_i.\mathbf{C}_{\varepsilon i} - \qquad (10)\\
&\sum_{i=j}^{n} m_i.{}^{j}\mathbf{P}_{Ci} \times {}^{j}R_i.C_{VCi} + {}^{j}\mathbf{N}_Q + \boldsymbol{\tau}_i(t)
\end{aligned}$$

$$l = \left\| \begin{matrix} j & \text{if } k \le j \\ k & k > j \end{matrix} \right.$$

Further on, the vector preproduct tensor is used in order to replace the vector product in the equation (8) with matrix multiplication. Utilizing the 3×3 skew-symmetric matrix form of the vectors ${}^{j}\mathbf{P}_{Ci}$ and $({}^{i}\mathbf{P}_k - \mathbf{P}_{Ci})$ or:

$$\begin{aligned}
&{}^{j}\mathbf{P}_{Ci} \times {}^{j}R_i.({}^{i}\mathbf{P}_k - \mathbf{P}_{Ci}) \times {}^{i}R_k = \\
&= {}^{j}V_{Ci}.{}^{j}R_i.{}^{i}V_k.{}^{i}R_k; \\
&{}^{j}V_{Ci} = \begin{bmatrix} 0 & -{}^{j}z_{ci} & {}^{j}y_{ci} \\ {}^{j}z_{ci} & 0 & -{}^{j}x_{ci} \\ -{}^{j}y_{ci} & {}^{j}x_{ci} & 0 \end{bmatrix}. \qquad (11)
\end{aligned}$$

${}^{i}V_k$ is computed likewise.

Taking into account (11), the state of equilibrium for link *j* is finally described by the equation (12):

$$\begin{aligned}
&\sum_{k=1}^{n}\sum_{i=l}^{n} (m_i.{}^{j}V_{Ci}.{}^{j}R_i.{}^{i}V_k.{}^{i}R_k + {}^{j}R_i.I_i.{}^{i}R_k).\ddot{\boldsymbol{\theta}}_k = \\
&\sum_{i=j}^{n} {}^{j}\mathbf{P}_{Ci} \times {}^{j}R_0.\mathbf{G}_i \text{ - } {}^{j}R_i.(\boldsymbol{\omega}_i \times I_i.\boldsymbol{\omega}_i) \text{ -} \qquad (12)\\
&{}^{j}R_i.I_i.\mathbf{C}_{\varepsilon i} - m_i.{}^{j}\mathbf{P}_{Ci} \times C_{VCi} + {}^{j}\mathbf{N}_Q + \boldsymbol{\tau}_i(t)
\end{aligned}$$

$$l = \left\| \begin{matrix} j & \text{if } k \le j \\ k & k > j \end{matrix} \right.$$

As it was said above, when consider kinematic chains with joints having single degree of freedom, any motion should be possible only along the $\hat{\mathbf{Z}}$ axis of the respective joint coordinate system, and all velocities, accelerations as well as joint torques should have the form (2). Thus, the set of second order ordinary differential equations (9) is formed by taking the dot products of (12) and the unit vector $\hat{\mathbf{z}}_j$ for each link (1 ≤ *j* ≤ *n*). These equations finally describe the motion of the system under the own weights of the links, forces applied to the end effector as well as the vector of joint driving torques

$\boldsymbol{\tau}$. Note, that one can also add to the right side of equation (12) any other conceivable, known, external forces acting upon links.

Equation in the form (6) is considered as initial-value problem and is easy to integrate utilizing Runge-Kutta or Kutta-Merson methods and standard routines, as it could be written in the form:

$$\ddot{\boldsymbol{\theta}} = \mathrm{M}^{-1}(\boldsymbol{\theta}).[\boldsymbol{\tau}(t) - \mathrm{H}(\boldsymbol{\theta},\dot{\boldsymbol{\theta}}) + \mathbf{G}(\boldsymbol{\theta})]\,. \qquad (13)$$

Equation (13) could be easily solved for $\boldsymbol{\tau}(t)$ given the time histories of positions, velocities and acceleration computed via some trajectory planning algorithm in joint or Cartesian space. This is straightforward way to obtain the necessary driving torques in joints and thus to solve the reverse problem.

3. Numerical example

In order to illustrate end test the algorithm, an experimental, heavy duty hydraulically driven, five DoF manipulator is considered. The 3D CAD model of the manipulator is shown in Figure 2. The manipulator is intended for moving loads along certain trajectory in space. A load of 3200 N is attached to the last link.

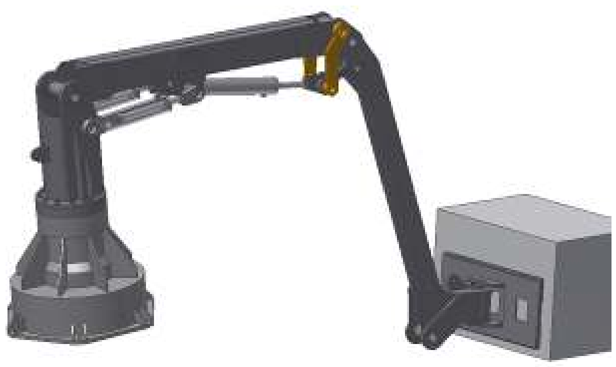
Figure 2. 3D CAD model of hydraulically driven manipulator

Solving the inverse problem. The trajectories in joint space are generated, using third order polynomials, for three trajectory points – start, middle and end, while the manipulator does not stop at the middle point. The trajectory points are specified by the coordinates of the load mass center and its orientation – rotation about the vertical axis (must be pointed out that the load is horizontal at any time). This allows solving the inverse kinematic problem for five DoF). The time history of positions, velocity and accelerations for each joint ($\boldsymbol{\theta}_j$, $\dot{\boldsymbol{\theta}}_j$, $\ddot{\boldsymbol{\theta}}_j$) are shown in Figure 3.

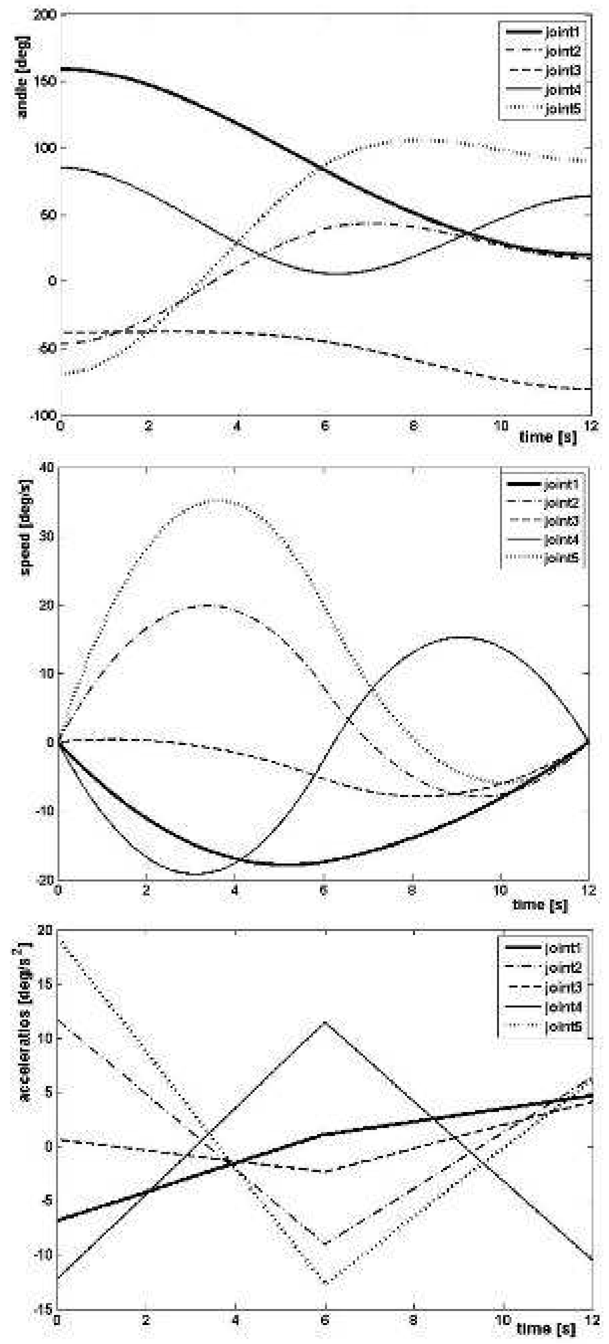

Figure 3. Time histories of positions, velocity and accelerations for five DoF manipulator

The solution of the equation (13) for $\boldsymbol{\tau}(t)$ is done by programming using the Matlab mathematical package. The result for the joints 2 to 4 (rotation about parallel horizontal axes) is shown in Figure 4.

The forward problem. To illustrate the above ratiocination, let consider revolving joints driven by hydraulic actuators.

Let assume that the pistons move with constant speed, while at a certain moment of time occurs sudden closure of the hydraulic valves. Further movement of the arms will be possible due to the compression of the working fluid as well as flexible pipes, presenting the driving torque as position dependent function. The process is a considerably complex one to be presented here in details, but it could be shown that the actual forces, velocities, elasticity and damping in the actuators can be transferred to the joint variables giving the law of the driving torques in the form:

$$\tau_j = k_j \cdot (\theta_j - \theta_{j0}) - d_j \cdot \dot{\theta}_j \,, \qquad (14)$$

where k_j is the elastic stiffness, d_j - damping factor, both transferred to the revolute joint, and θ_{j0} represents the free position (a position with zero elastic torque).

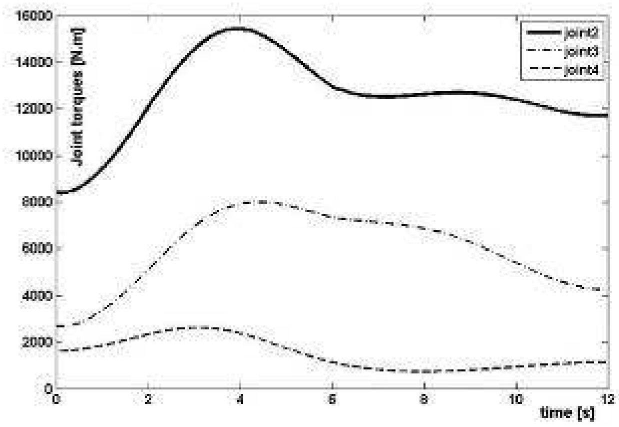

Figure 4. Computed necessary driving torques for joints 2 to 4

Substituting (14) in (12) and supplying the proper values for elasticity, damping and free position, as well as the initial values for positions and velocities for each joint, a set based on equation (12) for each link is integrated using standard routine. Results of such integration in the form of oscillations about the initial position of each joint are shown in Figure 5. The integration is done as initial value problem utilizing Matlab package with ODE23 solver.

4. Discussion and conclusion

In many cases, especially on the stage of design, often arises the need to simulate dynamic behavior of still not completely known product. On some occasions it is the need to determine the magnitude of the dynamic forces in large variety of operational conditions. Such dynamics forces with high capacity, fast moving manipulator arms could be considerable and must be taken into account when considering the structural integrity of the design. On the other hand, dynamic simulation could be required in order to create suitable control algorithms or to select the right driving system and actuators – electric drives, hydraulic actuators, etc. In general, all that tasks could be solved using some of the proposed algorithms and methods outlined in section one, but many of them are strictly specialized (mainly serving the purposes of real–time numerical control) and have to be modified in order to serve some design purposes.

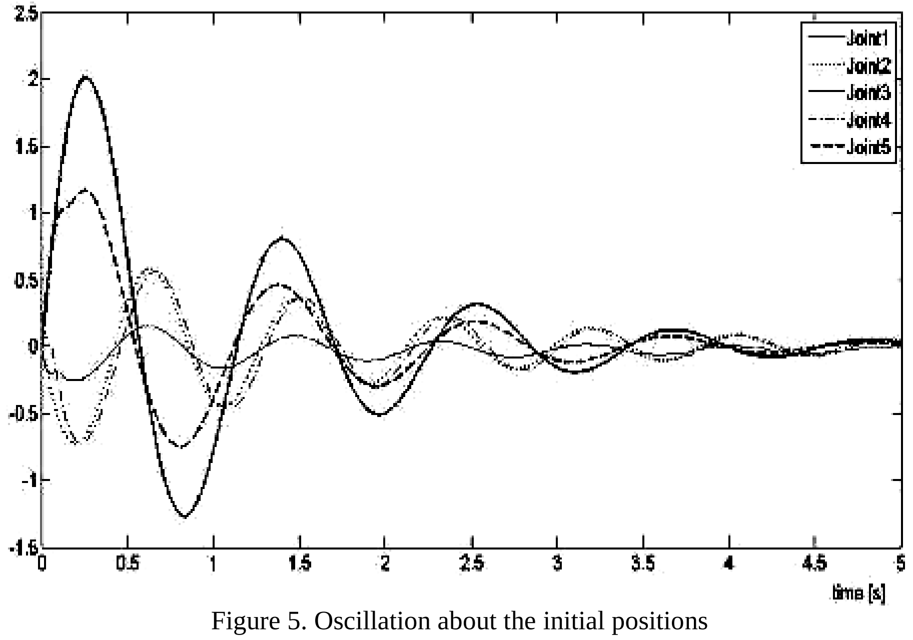

Figure 5. Oscillation about the initial positions

The algorithm proposed in this article uses an engineering approach to the problems of dynamic simulation of multibody systems forming an open-loop kinematic chain with revolute joints. Such designs form the majority of the equipment in use – industrial robots or outdoor materials handling equipment. This approach is based on Newton-Euler equations and state of equilibrium of each body during its spatial motion. It could be used to easily compose a set of second order differential equation describing the dynamic behavior, given particular time dependent or position dependent functions of driving torques in joints. Such set of differential equations should be integrated by standard routines. The advantage of the proposed method is that the set of equations can easily be written following the described rules and derived formulas, simply adding new members with the number of bodies increasing. Although such an algorithm is not very effective from order of growth point of view - close to $O(N^2)$, it could quite well be implemented for vast majority of cases when fast response is not necessary (one example is to use the results presented in Figure 4 for optimal design of the driving system).

The same sets of differential equations are to be used in order to solve reverse dynamic problem. Here however, all the equations are derived in closed form which leads to the direct computation of necessary joint torques. The closed form of the equations also permits a piecewise solution – each torque is computed without regard to the others.

A procedure based on the above described algorithm is easy to program and allows different structures with no limit to degrees of freedom to be examined with slight or even without any modification of the main program. The algorithm as well as the program realization is verified by using numerical simulation with Autodesk Inventor simulation package and 3D geometrical model of the prototype shown in Figure 2.

References

1. Uicker, J.J. (1965) *On the Dynamic Analysis of Spatial Linkages Using 4x4 Matrices*. Ph. D. Thesis, Northwestern University, Evanston
2. Uicker, J.J. (1967) *Dynamic Force Analysis of Spatial Linkages*. ASME Journal of Applied Mechanics, ISSN: 0021-8936Vol. 34, p 418-424
3. Kahn, M.E. (1969) *The Near-Minimum-Time Control of Open-Loop Articulated Kinematic Chains*. Stanford Artificial Intelligence Project Memo AIM-106
4. Hollerbach, J.M. (1980) *A Recursive Lagrangian Formulation of Manipulator Dynamics and a Comparative Study of Dynamics Formulation Complexity*. IEEE Systems, Man and Cybernetics, ISSN: 1094-6977, Vol. 10, issue 11
5. Armstrong, W.W. (1979) *Recursive Solution to the Equations of Motion of an N-Link Manipulator*. Proc. 5th World Congress on Theory of Machines and Mechanisms, Vol. 2, p. 1343-1346, New York, US
6. Li, C.-J., Sankar, T.S. (1992) *Fast Inverse Dynamics Computation in Real-time Robot Control*. Mechanism and Machine Theory, ISSN: 0094-114X, No. 27, p. 741-750
7. Li, C.-J., (1989) *A New Lagrangian Formulation of Dynamics of Robot Manipulators*. ASME Journal Dynamic, Systems, Measurement, and Control, ISSN: 1528-9028, Vol. 111, p. 559-567
8. Luh, J.Y.S., Walker, M.W., Paul, R.P.C. (1980) *On-Line Computational Scheme for Mechanical Manipulators*. ASME, Journal Dynamic Systems, Measurement & Control, ISSN: 1528-9028, Vol. 102, No. 2, p. 69-76
9. Fisette, P., Postiau, T., Sass, L., Samin, J.C. (2002) *Fully Symbolic Generation of Complex Multibody Models*. Mechanics of Structures and Machines, ISSN: 0890-5452, Vol. 30, p. 31-82
10. Featherstone, R. (1983) *The Calculation of Robot Dynamics using Articulated-Body Inertias*. International Journal of Robotics Research, ISSN: 0278-3649, Vol. 2, No. 1, p. 13-30
11. Featherstone, R. (1987) *Rigid Body Dynamics Algorithms*. Springer, ISBN-13: 978-0387743141
12. Silver, W.M. (1982) *On the Equivalence of Lagrangian and Newton-Euler Dynamics for Manipulators*. International Journal of Robotics Research, ISSN: 0278-3649, Vol. 2, p. 118-128
13. Angeles, J., Ma, O. (1988) *Dynamic Simulation of n-Axis Serial Robotic Manipulators Using a Natural Orthogonal Complement*. International Journal of Robotics Research, ISSN: 0278-3649, Vol. 7, No. 5, p. 32-47
14. Angeles, Jorge. (2007) *Fundamentals of Robotic Mechanical Systems. Theory, Methods, and Algorithms*. Third Edition, Springer Science+Business Media, ISBN-13: 978-0387 29412-4
15. Denavit, J., Hartenberg, R.S. (1955) *A Kinematic Notation for Lower-pair Mechanisms Based on Matrices*. ASME Journal of Applied Mechanics, ISSN 0021-8936, Vol. 22, p. 215-221
16. Craig, J.J. (2005) *Introduction to Robotics: Mechanics and Control*. 3rd Edition, Prentice Hall, ISBN-13: 978-0201543612
17. Vukobratovic, M., Potkonjak, V. (1982) *Dynamics of Manipulation Robots. Theory and Application*. Scientific Fundamentals of robotics, tome 1, Springer-Verlag, ISBN 978-6-3-540-11628-6, Berlin, Germany

Received in June 2013