A R C H I V E O F M E C H A N I C A L E N G I N E E R I N G

VOL. LXII 2015 Number 2

10.1515/meceng-2015-0016

Key words: robot dynamics, recursive algorithms, redundant manipulator

ŁUKASZ WOLIŃSKI [*], *PAWEŁ MALCZYK* [*]

DYNAMIC MODELING AND ANALYSIS OF A LIGHTWEIGHT ROBOTIC MANIPULATOR IN JOINT SPACE

The primary importance of the paper is the application of the efficient formulation for the simulation of open-loop lightweight robotic manipulator. The framework employed in the paper makes use of the spatial operator algebra and the associated equations are expressed in joint space. This compact representation of the manipulator dynamics makes it possible to solve the robot forward and inverse dynamics problems in a recursive and fast manner. In the current form, the presented algorithm can be applied for the dynamics simulation of an open-loop chain system possessing any number of joints. Specifically, the formulation has been successfully applied for the analysis of the 7DOF KUKA LWR robot. Results from a number of test cases for the robot demonstrate the verification of the calculations.

1. Introduction

Computational efficiency of multibody system dynamic simulations has been receiving increasing attention from researchers since seventies, mainly from the robotics field [1], [2], [3]. Further developments in this field can be seen in the multitude of efficient $O(n)$ (n – number of bodies) recursive sequential algorithms for analysis of tree-like topology rigid body dynamics. The mentioned works are not limited to [4], [5], [6], [7], [8], [9], [10], [11], [12], [13]. Some researchers developed recursive order '$O(n)$' formulations dynamics simulations of multibody systems with closed loops [14], [15], [16], [17]. The recursive sequential algorithms gave basis for further development of efficient low order formulations that are readily exploited up to date. To meet requirements for high-fidelity performance and accurate dynamics

[*] *Division of Theory of Machines and Robots, Institute of Aeronautics and Applied Mechanics, Faculty of Power and Aeronautical Engineering, Warsaw University of Technology, Nowowiejska 24, 00-665 Warsaw, Poland; E-mail: lwolinski@meil.pw.edu.pl, pmalczyk@meil.pw.edu.pl*

simulations of complex systems, it has become a practice to apply efficient, low order algorithms designed both for sequential and parallel computations. The parallel strategies enabled to decrease the turnaround time associated with computer simulations and even achieve results in real-time. The first attempts to exploit parallel strategies can be found in [18], [19], [20], [21], [22], [23]. More recent ideas regarding parallel algorithms for multi-body dynamics simulations can be found in [24], [25], [26], [27], [28], [29], [30], [31]. The latest comparative study on efficient sequential and parallel multi-body dynamics algorithms can be found in [32] and in recent books [33], [34].

In this paper, we discuss an approach based on the spatial operator algebra [35] that allows to obtain the equations of motion of a system in a compact matrix form. Manipulator is treated as a multibody system with rigid bodies and the equations are formulated in joint coordinates. Currently, for simplicity, neither actuator dynamics, nor joint elasticity are considered but they can be readily included in the formulation at the price of greater complexity of the system. The theoretical basis of the spatial operator algebra is concisely recalled in the paper. The primary objective of this work is the application of the efficient recursive formulae for the analysis of the KUKA lightweight robotic (LWR) manipulator. Such a procedure is usually a first step in the development of various robot controllers.

This paper is divided into six sections. Following the introduction and the extensive literature review in section 1, section 2 presents the recursive $O(n)$ algorithm in the spatial operator algebra framework. Section 3 is devoted to algorithmic steps within the formulation. Section 4 presents the numerical results associated with the simulation of the LWR robot as well as efficiency measurements. Section 5 is devoted to discussion. Summary and conclusions are presented in last section.

2. Algorithm formulation

Equations of motion for serial chains

Matrix-form equation of motion of a manipulator consisting of n rigid links can be written in a form [36]:

$$\mathbf{D}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) + \mathbf{Q}(\mathbf{q},\dot{\mathbf{q}}) = \mathbf{T}(\mathbf{q},\dot{\mathbf{q}}), \tag{1}$$

where $\mathbf{q} \in \mathbb{R}^{n\times 1}$ is a vector of joint coordinates, $\mathbf{D}(\mathbf{q}) \in \mathbb{R}^{n\times n}$ is the inertia matrix of the manipulator, $\mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) \in \mathbb{R}^{n\times n}$ is the Coriolis/centrifugal matrix, $\mathbf{G}(\mathbf{q}) \in \mathbb{R}^{n\times 1}$ is the gravity vector, $\mathbf{Q}(\mathbf{q},\dot{\mathbf{q}}) \in \mathbb{R}^{n\times 1}$ is a vector of external

forces acting on links and vector $\mathbf{T}(\mathbf{q}, \dot{\mathbf{q}}) \in \mathbb{R}^{n\times 1}$ contains driving torques at joints.

Figure 1 shows a pair of bodies connected with the revolute joint. Based on [37] we use the following notation: q_j is the relative rotational coordinate between links i and j, while $j = i + 1$; $\mathbf{s}_{ij}$ and $\mathbf{s}_{ji}$ are vectors locating joint attachment points in links i and j; $\mathbf{h}''_{ij}$ is the unit vector along rotational axis of the joint defined in the π''_i frame. Centroidal frames on links i and j are π'_i and π'_j, while π''_i and π''_j are joint frames on links i and j; π_0 is the global reference frame. $\mathbf{C}_{ij}$ and $\mathbf{C}_{ji}$ are transformation matrices: from the frame π''_i to π'_i and from π''_j to π'_j. $\mathbf{A}''_{ij}$ and $\mathbf{A}_i$ are transformation matrices: from the frame π''_j to π''_i and from π'_i to π_0. Vectors with the prime symbol are expressed in centroidal frames, with the double prime symbol – in joint frames, without any symbol – in global reference frame.

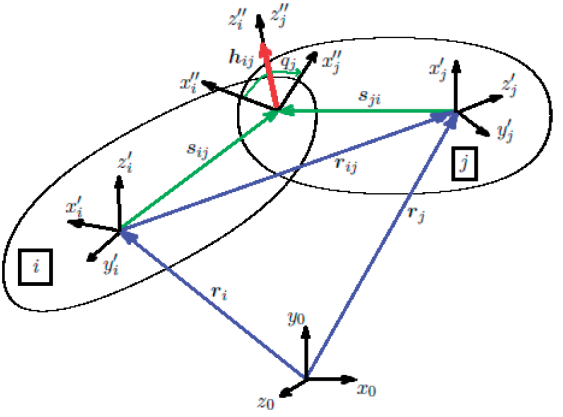

Fig. 1. Pair of bodies connected by a revolute joint

Position of the center of mass of the link j expressed in the global reference frame π_0 is given by:

$$\mathbf{r}_j = \mathbf{r}_i + \mathbf{r}_{ij}, \tag{2}$$

where:

$$\mathbf{r}_{ij} = \mathbf{s}_{ij} - \mathbf{s}_{ji}, \tag{3}$$
$$\mathbf{s}_{ij} = \mathbf{A}_i \mathbf{s}'_{ij}, \tag{4}$$
$$\mathbf{s}_{ji} = \mathbf{A}_j \mathbf{s}'_{ji}, \tag{5}$$

and the term:

$$\mathbf{A}_j = \mathbf{A}_i \mathbf{C}_{ij} \mathbf{A}''_{ij} \mathbf{C}^T_{ji} \tag{6}$$

is the transformation matrix from the frame π'_j to π_0.

Rotational axis of the j-th joint is given by vector:

$$\mathbf{h}_{ij} = \mathbf{A}_i \mathbf{C}_{ij} \mathbf{h}''_{ij}. \tag{7}$$

Velocity of the j-th link expressed in matrix form is:

$$\mathbf{V}_j = \begin{bmatrix} \dot{\mathbf{r}}_j \\ \boldsymbol{\omega}_j \end{bmatrix} = \mathbf{B}_{ij} \mathbf{V}_i + \mathbf{H}_j \dot{q}_j, \tag{8}$$

where $\dot{\mathbf{r}}_j$ is the linear velocity of the center of mass and $\boldsymbol{\omega}_j$ is the angular velocity, while $\mathbf{B}_{ij}$ and $\mathbf{H}_j$ are shift and joint space matrices, respectively and are defined in the Appendix.

Differentiating Eq. (8) with respect to the time allows us to obtain acceleration of link j as:

$$\dot{\mathbf{V}}_j = \begin{bmatrix} \ddot{\mathbf{r}}_j \\ \dot{\boldsymbol{\omega}}_j \end{bmatrix} = \mathbf{B}_{ij} \dot{\mathbf{V}}_i + \dot{\mathbf{B}}_{ij} \mathbf{V}_i + \mathbf{H}_j \ddot{q}_j + \dot{\mathbf{H}}_j \dot{q}_j. \tag{9}$$

Matrices $\dot{\mathbf{B}}_{ij}$ and $\dot{\mathbf{H}}_j$ are defined in the Appendix.

Velocity equations (8) for all n links can be written jointly in a form:

$$\mathbf{V} = \mathbf{B}\mathbf{V} + \mathbf{H}\dot{\mathbf{q}}, \tag{10}$$

which can be expressed as:

$$\mathbf{V} = \boldsymbol{\Phi}\mathbf{H}\dot{\mathbf{q}}. \tag{11}$$

Again, the matrices $\mathbf{B}$, $\mathbf{H}$ and $\boldsymbol{\Phi}$ are defined in the Appendix.

Similarly, jointly written equations (9) for all n links have a form:

$$\begin{aligned} \dot{\mathbf{V}} &= \mathbf{B}\dot{\mathbf{V}} + \dot{\mathbf{B}}\mathbf{V} + \mathbf{H}\ddot{\mathbf{q}} + \dot{\mathbf{H}}\dot{\mathbf{q}} = \\ &= \mathbf{B}\dot{\mathbf{V}} + \dot{\mathbf{B}}\boldsymbol{\Phi}\mathbf{H}\dot{\mathbf{q}} + \mathbf{H}\ddot{\mathbf{q}} + \dot{\mathbf{H}}\dot{\mathbf{q}}, \end{aligned} \tag{12}$$

which can be transformed into:

$$\dot{\mathbf{V}} = \boldsymbol{\Phi}((\dot{\mathbf{B}}\boldsymbol{\Phi}\mathbf{H} + \dot{\mathbf{H}})\dot{\mathbf{q}} + \mathbf{H}\ddot{\mathbf{q}}), \tag{13}$$

where $\dot{\mathbf{B}}$ and $\dot{\mathbf{H}}$ are the time derivatives of $\mathbf{B}$ and $\mathbf{H}$.

The equation of motion for the link j can be written as follows:

$$\mathbf{M}_j \dot{\mathbf{V}}_j = \mathbf{F}_{ij} - \mathbf{B}^T_{jk} \mathbf{F}_{jk} + \boldsymbol{\Omega}_j \mathbf{V}_j + \mathbf{F}_{\mathbf{G}j} + \mathbf{F}_{\mathbf{Q}j}, \tag{14}$$

where: $i = j - 1$, $k = j + 1$, $\mathbf{M}_j$ is the j-th link mass and inertia matrix, $\boldsymbol{\Omega}_j$ is defined in the Appendix, $\mathbf{F}_{\mathbf{G}j}$ contains gravitational force, $\mathbf{F}_{\mathbf{Q}j}$ contains

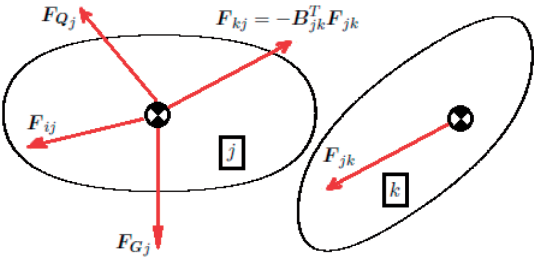

Fig. 2. Forces acting on j-th body

external force and torque acting on the center of mass of j-th link, $\mathbf{F}_{jk}$ is the force exerted by the j-th link on the k-th link. All forces are depicted on Figure 2.

Hence, the i-th link acts on the j-th link with a force:

$$\mathbf{F}_{ij} = \mathbf{M}_j \dot{\mathbf{V}}_j + \mathbf{B}_{jk}^T \mathbf{F}_{jk} - \mathbf{\Omega}_j \mathbf{V}_j - \mathbf{F}_{\mathbf{G}j} - \mathbf{F}_{\mathbf{Q}j}. \tag{15}$$

Equations of motion (15) for the whole chain can be written compactly as:

$$\mathbf{F} = \mathbf{M}\dot{\mathbf{V}} + \mathbf{B}^T\mathbf{F} - \mathbf{\Omega}\mathbf{V} - \mathbf{F}_{\mathbf{G}} - \mathbf{F}_{\mathbf{Q}}, \tag{16}$$

which could be easily transformed to:

$$\mathbf{F} = \mathbf{\Phi}^T(\mathbf{M}\dot{\mathbf{V}} - \mathbf{\Omega}\mathbf{V} - \mathbf{F}_{\mathbf{G}} - \mathbf{F}_{\mathbf{Q}}). \tag{17}$$

The power-balance equation for a single link bring us to the following relation:

$$\mathbf{H}_j^T \mathbf{F}_{ij} = \tau_j, \tag{18}$$

where τ_j is a scalar value of driving torque at joint j.

For n links the relation becomes:

$$\mathbf{H}^T\mathbf{F} = \mathbf{T}, \tag{19}$$

where $\mathbf{T}$ contains driving torques at joints.

Substituting (17) into (19):

$$\mathbf{H}^T\mathbf{\Phi}^T(\mathbf{M}\dot{\mathbf{V}} - \mathbf{\Omega}\mathbf{V} - \mathbf{F}_{\mathbf{G}} - \mathbf{F}_{\mathbf{Q}}) = \mathbf{T}. \tag{20}$$

Then, substituting (13) and (11) into (20):

$$\mathbf{H}^T\mathbf{\Phi}^T\left(\mathbf{M}\mathbf{\Phi}((\dot{\mathbf{B}}\mathbf{\Phi}\mathbf{H} + \dot{\mathbf{H}})\dot{\mathbf{q}} + \mathbf{H}\ddot{\mathbf{q}}) - \mathbf{\Omega}\mathbf{\Phi}\mathbf{H}\dot{\mathbf{q}} - \mathbf{F}_{\mathbf{G}} - \mathbf{F}_{\mathbf{Q}}\right) = \mathbf{T} \tag{21}$$

and transforming it:

$$\begin{aligned}&\mathbf{H}^T\mathbf{\Phi}^T\mathbf{M}\mathbf{\Phi}\mathbf{H}\ddot{\mathbf{q}} + \mathbf{H}^T\mathbf{\Phi}^T\mathbf{M}\mathbf{\Phi}(\dot{\mathbf{B}}\mathbf{\Phi}\mathbf{H} + \dot{\mathbf{H}} - \mathbf{\Omega}\mathbf{\Phi}\mathbf{H})\dot{\mathbf{q}} + \\ &+ \mathbf{H}^T\mathbf{\Phi}^T(-\mathbf{F_G}) + \mathbf{H}^T\mathbf{\Phi}^T(-\mathbf{F_Q}) = \mathbf{T}\end{aligned} \tag{22}$$

finally leads to:

$$\mathbf{D}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) + \mathbf{Q}(\mathbf{q},\dot{\mathbf{q}}) = \mathbf{T}(\mathbf{q},\dot{\mathbf{q}}), \tag{23}$$

where:

$$\mathbf{D}(\mathbf{q}) = \mathbf{H}^T\mathbf{\Phi}^T\mathbf{M}\mathbf{\Phi}\mathbf{H}, \tag{24}$$

$$\mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) = \mathbf{H}^T\mathbf{\Phi}^T\Big(\mathbf{M}\mathbf{\Phi}(\dot{\mathbf{B}}\mathbf{\Phi}\mathbf{H} + \dot{\mathbf{H}}) - \mathbf{\Omega}\mathbf{\Phi}\mathbf{H}\Big), \tag{25}$$

$$\mathbf{G}(\mathbf{q}) = \mathbf{H}^T\mathbf{\Phi}^T(-\mathbf{F_G}), \tag{26}$$

$$\mathbf{Q}(\mathbf{q},\dot{\mathbf{q}}) = \mathbf{H}^T\mathbf{\Phi}^T(-\mathbf{F_Q}). \tag{27}$$

One should also point out that special algorithms exist for the efficient mass matrix $\mathbf{D}(\mathbf{q})$ factorization [38] but this issue is beyond the scope of this paper. In the paper, we used fast algorithms for mass matrix inversion.

The equations of motion (23) describe the open-loop manipulator dynamics in joint space. The analysis of the systems with closed-loop chains (such as dual-arm manipulation) requires extension of the formalism and addition of loop-closure equations. The usual technique applied here relies on the cut joint method [39]. However, the simulation of such systems changes the class of the problem and will not be discussed here. Although the equations (23) are expressed in a compact vector-matrix form, one should note that the components of these quantities are derived recursively along the chain of the manipulator. The inherent advantage of the representation (23) lies in the convenience of manipulator model analysis and control system synthesis.

3. Algorithmic steps

To calculate the matrices $\mathbf{D}(\mathbf{q})$, $\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})$, $\mathbf{G}(\mathbf{q})$ and $\mathbf{Q}(\mathbf{q},\dot{\mathbf{q}})$ the following algorithm is used:

1. Start with initializing matrices $\mathbf{H} = \mathbf{0}_{6n\times 6}$, $\dot{\mathbf{H}} = \mathbf{0}_{6n\times 6}$, $\dot{\mathbf{B}} = \mathbf{0}_{6n\times 6n}$, $\mathbf{\Phi} = \mathbf{I}_{6n\times 6n}$, $\mathbf{M} = \mathbf{0}_{6n\times 6n}$, $\mathbf{\Omega} = \mathbf{0}_{6n\times 6n}$, $\mathbf{F_G} = \mathbf{0}_{6n\times 1}$ and $\mathbf{F_Q} = \mathbf{0}_{6n\times 1}$. Set $j = 1$.
2. Calculate $\mathbf{H}_j$ (37) and $\dot{\mathbf{H}}_j$ (39). Put them in rows $6j-5$ to $6j$ and column j of $\mathbf{H}$ and $\dot{\mathbf{H}}$. Calculate $\mathbf{M}_j$ (45) and $\mathbf{\Omega}_j$ (47). Put them in rows $6j-5$ to $6j$ and columns $6j-5$ to $6j$ of $\mathbf{M}$ and $\mathbf{\Omega}$. Calculate $\mathbf{F_G}_j$ (49) and $\mathbf{F_Q}_j$ (50). Put them in rows $6j-5$ to $6j$ of $\mathbf{F_G}$ and $\mathbf{F_Q}$.

3. Check if condition $j > 1$ is true. If not, skip this point. If yes, then set $i = j - 1$. Next, calculate $\mathbf{B}_{ij}$ (36), $\dot{\mathbf{B}}_{ij}$ (38) and $\mathbf{\Phi}_j = \mathbf{B}_{ij}\mathbf{\Phi}_i$ where $\mathbf{\Phi}_i$ is made of rows $6j-5$ to $6j$ and columns 1 to $6i$ of $\mathbf{\Phi}$. Put $\dot{\mathbf{B}}_{ij}$ in rows $6j-5$ to $6j$ and columns $6i-5$ to $6i$ of $\dot{\mathbf{B}}$. Put $\mathbf{\Phi}_j$ in rows $6j-5$ to $6j$ and columns 1 to $6i$ of $\mathbf{\Phi}$.
4. Set $j = j + 1$. If $j \leq n$, then go to step 2, else go to step 5.
5. Calculate $\mathbf{D}(\mathbf{q})$ (24), $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ (25), $\mathbf{G}(\mathbf{q})$ (26) and $\mathbf{Q}(\mathbf{q}, \dot{\mathbf{q}})$ (27).

Having completed the algorithmic steps, one can obtain a numerical form of Eq. (1). It can be used to solve the inverse or forward dynamics problem.

4. Numerical examples

4.1. Preliminaries

KUKA LWR 4+ (depicted in Fig. 3) is a 7-degree of freedom light-weight robot. It is intended both for industrial as well as research applications. Redundant anthropomorphic structure gives the robot, among other properties, the ability to avoid obstacles. Manipulator dynamic model will serve as a basis for various model based controllers and also to address manipulation and motion planning algorithms.

Fig. 3. KUKA LWR 4+ robot. Source: http://tmr.meil.pw.edu.pl

Kinematic data for the model was obtained from the official KUKA documentation. Dynamic parameters of the links are not available yet, but recently nonlinear combinations of parameters (dynamic coefficients) used in the control system by KUKA were identified in [40], and extended in [41]. For the purpose of this work (to present the algorithm) the approximation of the parameters were used. Data are shown in the Appendix.

The algorithm was implemented in the Scilab environment and used to solve inverse and forward dynamics problem for the KUKA robot.

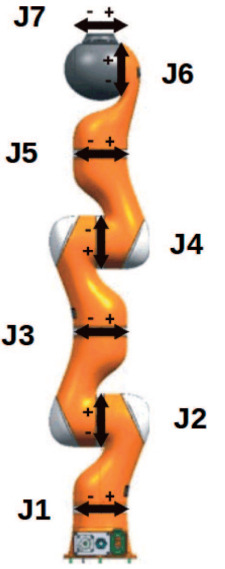

Fig. 4. Joint numbering of the KUKA LWR 4+ robot. Source: [42]

4.2. Inverse dynamics

The first numerical test case is associated with the problem of reconstructing the internal torques **T** from the motion specified at joints. In the inverse dynamics simulation all joints were set to move along the same trajectory for 14 seconds (similarly as in [40]):

$$q_{d\,j}(t) = \frac{\pi}{2}\cos\left(\frac{\pi}{7}t\right) \quad j = 1, \ldots, 7, \tag{28}$$

$$\dot{q}_{d\,j}(t) = -\frac{\pi^2}{14}\sin\left(\frac{\pi}{7}t\right) \quad j = 1, \ldots, 7, \tag{29}$$

with the initial conditions:

$$q_j(0) = \frac{\pi}{2} \quad j = 1, \ldots, 7, \tag{30}$$

$$\dot{q}_j(0) = 0 \quad j = 1, \ldots, 7. \tag{31}$$

One should note that the inverse dynamics problem is a pure algebraic task. The efficiency of joint space formulation is used here to minimize the number of arithmetic operations. The motion of all joints in the 7-DOF manipulator were simulated but for the sake of clarity only the time history of calculated torque for the axis 1 is presented in Fig. 5. It can be noted that the generated torque timeplot is smooth and symmetric, which is an expected result for this simulation. Similar results were obtained for other axes.

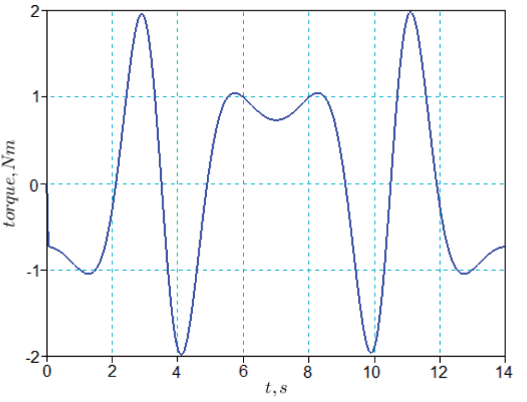

Fig. 5. Driving torque at joint 1

4.3. Forward dynamics

The forward dynamics problem deals with the task of predicting the motion of the system from known external forces and torques. To verify the algorithm, a model of the KUKA manipulator was created in ADAMS commercial software that is a solver for multibody dynamics simulation. Then, Scilab and ADAMS were set to simulate manipulator motion for 14 seconds. Driving torques at joints were specially designed to assure the smooth motion of the system. Independent joint PD controllers were used in the form:

$$\tau_j = k_j(q_{dj} - q_j) + c_j(\dot{q}_{dj} - \dot{q}_j) \quad j = 1, \ldots, 7, \tag{32}$$

where q_{dj} and $\dot{q}_{dj}$ are defined in (28) and (29), and the initial conditions are (30) and (31). Torsional stiffness **k** and torsional damping **c** that are physical representations of control gains in PD controllers associated with joints were chosen experimentally as:

$$\mathbf{k} = [10\ 1000\ 200\ 100\ 10\ 10\ 10]^T \frac{\text{Nm}}{\text{rad}}, \tag{33}$$

$$\mathbf{c} = [1000\ 10000\ 1000\ 1000\ 10\ 10\ 10]^T \frac{\text{kg} \cdot \text{m}}{\text{rad} \cdot \text{s}}. \tag{34}$$

One should point out that the torques obtained from the control law (32), together with the assumed constants (33), (34) should be similar to the time histories recorded in the inverse dynamics simulation (Fig. 5).

The choice of the simple PD controller is associated with two main reasons. Firstly, we wish to ensure reasonable torques as the inputs to the model. The application of random torques as the inputs to the model would presumably lead to chaotic motion and unreliable results. Secondly, the integrator procedure is of finite order and may generate numerical errors during the

course of simulation. The PD controller keeps the error bounded. It should be noted that the evaluation of joint torques is based only on the PD controller. There is no inverse dynamics algorithm involved in the specification of control torques.

The commercial package ADAMS defines and solves the equations of motion for a multibody system in a form of differential-algebraic equations. The differential part is connected to the dynamics of the system, whereas, the algebraic part is associated with constraint equations imposed on the motion of the bodies. To guarantee that the provided solutions of differential-algebraic equations are reliable, specialized procedures exist in ADAMS environment. In the simulation, we used Gear algorithm with differential index-2 stabilization procedure [43]. On the other hand, we exploited the BDF method implemented in Scilab [44], [45] for the integration of ordinary differential equations generated by the recursive algorithm.

The motion of all joints in the 7-DOF manipulator were simulated but for the sake of clarity, results of only one joint are presented. All numerical results are associated with the the 1-st axis of the manipulator. The readers should refer to Fig. 4 that shows the joint numbering. The differences in the numerical results between the models implemented in ADAMS and Scilab environment are captured using the following scalar measures:

$$\begin{aligned} \delta_q^i &= |q_i^{ADAMS} - q_i^{Scilab}|, \\ \delta_{\dot{q}}^i &= |\dot{q}_i^{ADAMS} - \dot{q}_i^{Scilab}|, \\ \delta_{\ddot{q}}^i &= |\ddot{q}_i^{ADAMS} - \ddot{q}_i^{Scilab}|. \end{aligned} \tag{35}$$

and presented alongside the 1-st joint position, velocity and acceleration during simulation on Figs. 6 – 11. Similar results were obtained for other axes – values never exceeded $\delta_q^i = 5 \cdot 10^{-4}$ rad, $\delta_{\dot{q}}^i = 5 \cdot 10^{-4}$ rad/s and $\delta_{\ddot{q}}^i = 0{,}014\,\mathrm{rad/s^2}$ for $i = 1, \ldots, 7$.

Differences for position and velocity are rather negligible. Larger differences (but still comparatively low) are produced for accelerations at joints. Even though different approaches were used (ADAMS – Cartesian coordinates and DAE, Scilab – joint coordinates and ODE), the results are comparable. This proves the algorithm correctness of the implementation for KUKA LWR robot.

4.4. Efficiency measurements

To compare the efficiency of various algorithms, it is necessary to have a common measure of efficiency. There are various methods to do that. One common measure of efficiency is the floating point operations count, i.e. the

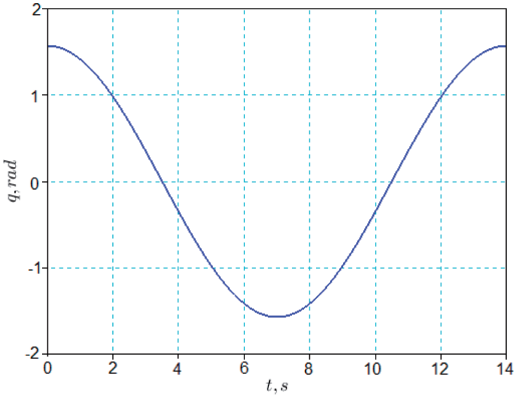

Fig. 6. Joint 1 position (q_1)

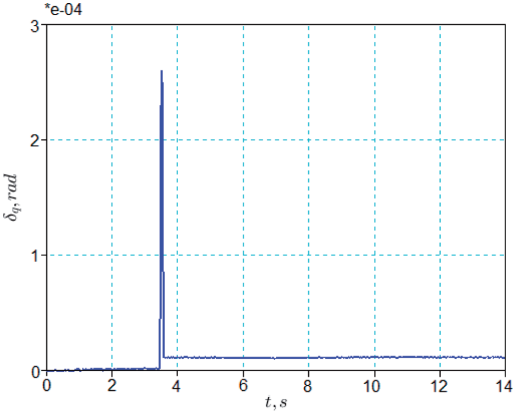

Fig. 7. Difference between ADAMS and algorithm implemented in Scilab for q_1 coordinate

number of multiplications and additions to evaluate the equations of motion for a given system. Usually this number is expressed as a function of the number of bodies or the number of degrees of freedom of a system. In the literature, there are detailed reports on operation counts associated with various recursive $O(n)$ formulations. The numerical cost is usually evaluated for unbranched kinematic chain in which all the joints are revolute, and all the bodies have the same masses, inertias, and geometric parameters.

It is well known from the literature that the recursive $O(n)$ (n – number of bodies) formulations are among the group of the most efficient approaches for forward dynamics of multibody systems. The computational efficiency of the recursive algorithms is associated with elimination of constraint forces between the interacting bodies. For an open-loop kinematic chain, the equations of motion are exclusively expressed in a minimal set of relative coordinates through the application of joint motion subspace associated with constraint equations imposed on the interacting bodies. This notion can be generalized

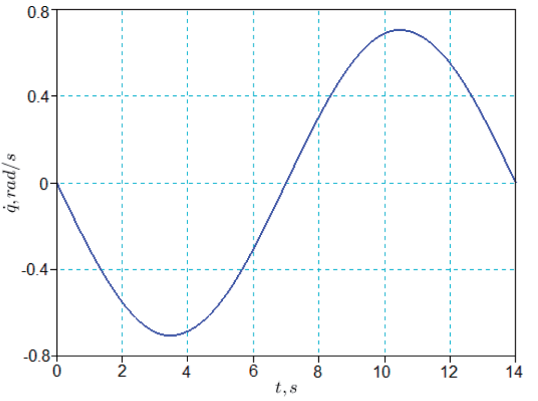

Fig. 8. Joint 1 velocity ($\dot{q}_1$)

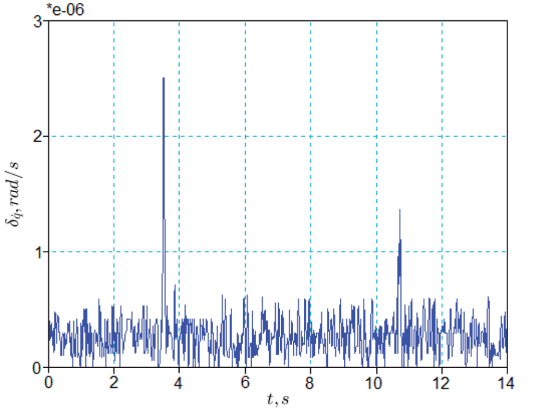

Fig. 9. Difference between ADAMS and algorithm implemented in Scilab for $\dot{q}_1$ coordinate

for the application in forward dynamics simulation of closed-loop systems as indicated in various papers in the literature.

The detailed operation count is a good measure to predict the effort involved in the computation process. However, in reality there are other sources that can influence the computational cost associated with forward dynamics simulation. Firstly, the practical implementation issues can affect the performance of the algorithm. The turnaround time associated with the fast low order algorithm can be degraded compared to other e.g. global formulations, if the implementation is not well optimized for a given system. Secondly, the type of computer architecture used in the calculations may influence the performance of the algorithm. This issue is specifically important when parallel computations are taken into account. Communication and synchronization between different subtasks are typically some of the greatest obstacles to getting good parallel program performance, as indicated in e.g. [28], [30], [46], [47]. Thirdly, the process of forward dynamics algorithm

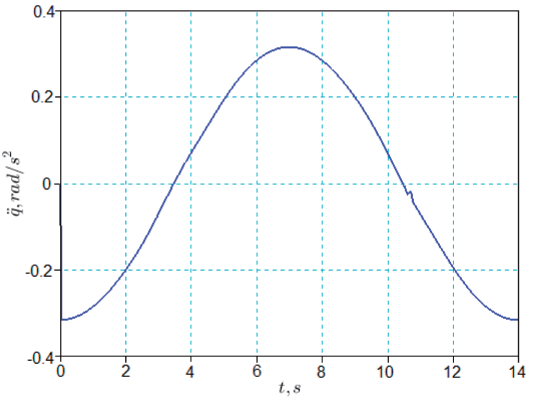

Fig. 10. Joint 1 acceleration ($\ddot{q}_1$)

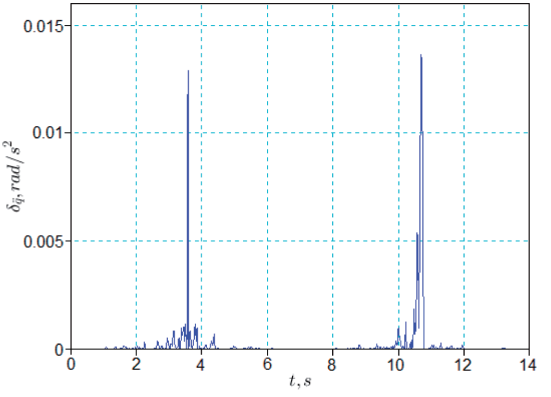

Fig. 11. Difference between ADAMS and algorithm implemented in Scilab for $\ddot{q}_1$ coordinate

benchmarking is a problem itself. The performance of recursive algorithms is easy to understand for unbranched single chains, but it is not straightforward to predict the real numerical cost for more complicated topologies such as closed-loops or coupled loops that one may frequently encounter in research and engineering practice. Moreover, there is a lack of standard benchmarks to measure the performance of multibody simulation codes as indicated in several papers from the field [48], [49], and recent international conferences (e.g. IMSD2014/ACMD2014 BEXCO Busan, Korea June 30 - July 3, 2014).

To assess the efficiency of the recursive algorithm presented in the paper, we pursue the practical avenue. The benchmarking is performed on the individual basis of KUKA 7DOF robot. In this paper we make a series of numerical experiments to compare the real performance of the simulation codes, i.e. our own implementation of the recursive algorithm, and ADAMS/Solver code. The comparison is based on the wall-clock time as well as on the number of right hand side function evaluations. The presented recursive al-

gorithm is implemented in Scilab, which is an open source software package being alternative to MATLAB. Scilab, just like MATLAB, is a high-level language intended primarily for numerical computations. The source code in Scilab is interpreted without previously compiling it into a machine language program. Therefore, it is expected that the Scilab program execution wall-clock time would be much larger than the elapsed real time associated with the executable program such as ADAMS software.

Figures 12 and 13 illustrate the elapsed wall-clock time during the program execution versus the simulation time, which is assumed to be 14 seconds for the KUKA robot model. The data in the figures are gathered for various integration procedures used for the solution process, and for different tolerances set in the integrators. It becomes apparent that Scilab based performance is degraded because of the issues raised in the previous paragraph. The ADAMS program run without updating the graphics is always more than four times faster than the analogous simulation code implemented in Scilab. It can be noted that this result is not appreciably sensitive to changing the integrator tolerances. On the other hand, the obtained results show that the physical phenomena under consideration, e.g. strong motions with relatively large accelerations, may influence the total performance of the code. In Figures 12 and 13 one may observe the fragments of the curves which are almost vertical. This means that the integration procedure is spending significant amount of time to proceed the simulation further.

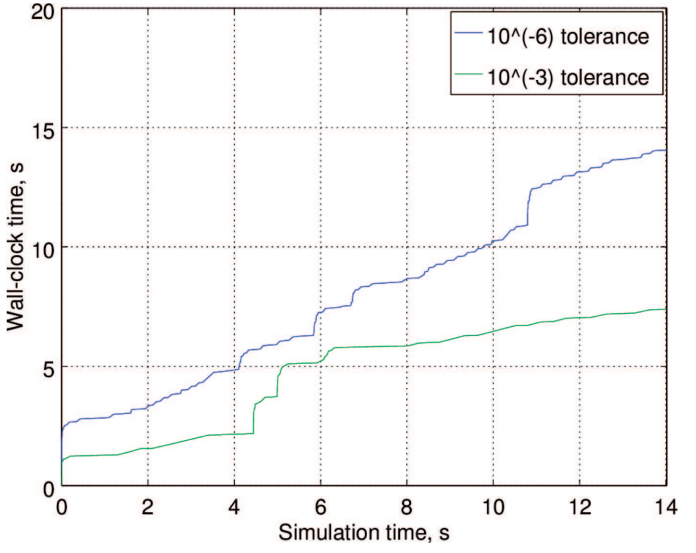

Fig. 12. Wall-clock time versus simulation time for absolute and relative tolerance of the BDF integration set to 10^{-6} and 10^{-3} (Scilab)

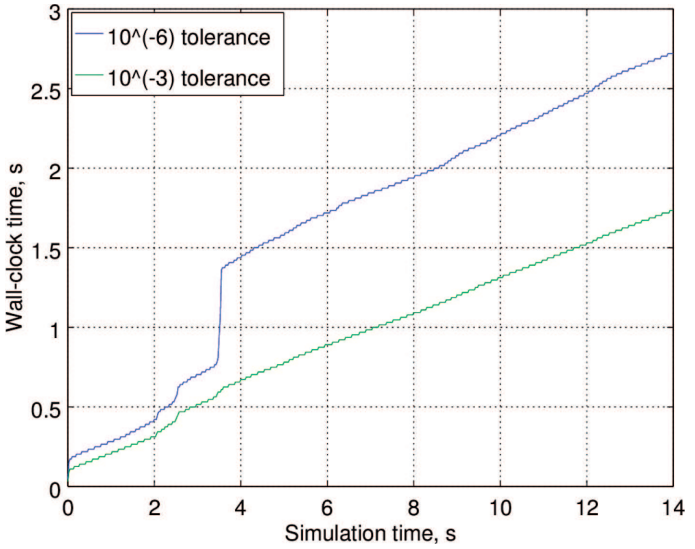

Fig. 13. Wall-clock time versus simulation time for absolute and relative tolerance of the GSTIFF SI2 integration set to 10^{-6} and 10^{-3} (ADAMS)

We also record the number of right hand side (RHS) function evaluations that allows for generating the equations of motion for a given multi-body system. Table 1 illustrates such data for various integrators available in ADAMS software as well as for BDF integrator in our implementation.

Table 1.

Number of right hand side function evaluations

Software	Integrator	Integrator tolerance	
		10^{-6}	10^{-3}
Scilab	BDF	1924	993
ADAMS	GSTIFF, SI2	4432	1745
	GSTIFF, I3	2583	1613
	WSTIFF, SI2	4852	1824
	WSTIFF, I3	3388	1489
	HHT	7431	2190
	Newmark	7495	2196
	HASTIFF, SI1	7010	1821
	HASTIFF, SI2	6272	1787

The results observed in Table 1 stay in contrast to the performance measured by elapsed real time presented in Figures 12, 13. Independently of

the tolerances set in the integrator, the number of RHS function evaluations for the recursive algorithm is always smaller than the similar measures for ADAMS software. On the basis of this result, one may carefully suspect that the implementation of the recursive algorithm for the given robot in the form of executable program can be faster than the ADAMS program for the same model and simulation conditions. However, this outcome should be interpreted wisely, as both solvers exploit various formulations for evaluation of the equations of motion. The recursive algorithm is a topological formulation that finally leads to the set of ordinary differential equations. On the other hand, ADAMS software exploits the global formulation with redundant coordinates for the system state description. This implies the necessity of the solution of differential-algebraic equations. From the above analysis it appears the performance evaluation is a difficult problem, especially when the objective and quantitative measures are taken into account.

5. Discussion

The recursive method presented in the paper is not new and it is well known in the community. The mature form of the formulation was proposed by Featherstone in 1983 [8] and it is called the articulated body algorithm. The procedure was extended many times and generalized by the team from the Jet Propulsion Laboratory (Jain, Rodriguez, Kreutz-Delgado) among the others. It should be emphasized that the authors focused more on the application of the recursive method for the forward dynamics of LWR KUKA 7DOF robot. The intention in the background is related to the investigation of manipulator dynamic properties. This information is particularly useful for control and simulation purposes due the fact that KUKA LWR possesses almost open control architecture in the form of *Fast Research Interface* [50].

Detailed treatment of the recursive algorithm is given and recalled in the paper. The spatial operator algebra is employed to express the dynamics of the KUKA LWR manipulator. The minor thing associated with our implementation is the evaluation of the equations of motion. The equations are generated in coordinate frames associated with the center of mass of bodies. Originally, articulated body-like algorithms are expressed in coordinate frames other than used in the paper. The appendix involves the estimated parameters of the LWR KUKA simulation model. Intermediate rotations matrices as well as detailed geometric parameters are given. What is more important, the gathered numerical results are presented in the paper. This outcome can be useful for other authors working in the field of robotics and multibody dynamics e.g. for control or benchmarking purposes and can be easily reproduced if it is needed.

The efficiency of the recursive formulation is investigated in the form of separate subsection provided in the paper. First of all, it is noted by the authors that the performance evaluation of multibody dynamics algorithms is a difficult issue. Secondly, we carry out a series of numerical experiments to estimate the efficiency of the recursive algorithm in comparison with ADAMS software. The practical measures of the performance are taken into account such as the wall clock time measurements or the number of right hand side function evaluations. The obtained numerical results for the KUKA LWR robot are of practical matter and can be used as a baseline for future comparisons in the group of industrial-like applications.

The implicit values of the paper are coming from the current endeavors of the authors. The partial numerical results presented in the paper are useful for further developments, especially in robot parameter identification procedures. Some of the advances in the field have been approached in the recent papers [40], [41]. Nevertheless, this is still an ongoing and active area of research. Having the complete set of reliable parameters, one can apply recursive algorithm exploited in the paper to perform real-time simulation, which is usually very desirable in various model based controllers.

6. Conclusions

This paper describes the algorithmic approach for the efficient dynamics simulation of serial and tree-like systems with rigid bodies such as robotic arms. Equations of motion are derived by using spatial operator algebra and then algorithmic steps are formulated. Equations of motion are expressed in a compact matrix form that is specifically important for various applications, e.g. identification, dynamic analysis and synthesis of various types of controllers.

The joint space recursive formulation has been successfully applied for the analysis of the 7DOF KUKA LWR robot. Results from the number of test cases and the comparisons against the outcome obtained by using the commercial multibody software show the verification of the algorithm. Moreover, some efficiency results are provided and comparatively set against the outcome from the commercial multibody solver.

The parameter identification and simulation based verification of the KUKA LWR manipulator are areas of forthcoming research for the authors.

Appendix

Term $\tilde{\mathbf{x}}$ is a skew-symmetric matrix associated with the vector $\mathbf{x}$. Shift and joint space matrices:

$$\mathbf{B}_{ij} = \begin{bmatrix} \mathbf{I} & -\widetilde{\mathbf{r}}_{ij} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}_{6\times 6}, \tag{36}$$

$$\mathbf{H}_j = \begin{bmatrix} -\widetilde{\mathbf{h}}_{ij}\mathbf{s}_{ji} \\ \mathbf{h}_{ij} \end{bmatrix}_{6\times 1}. \tag{37}$$

Their time derivatives:

$$\dot{\mathbf{B}}_{ij} = \begin{bmatrix} \mathbf{0} & -(\widetilde{\boldsymbol{\omega}}_i\mathbf{r}_{ij} - \widetilde{\mathbf{h}}_{ij}\mathbf{s}_{ji}\dot{q}_j)\widetilde{\ } \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \tag{38}$$

$$\dot{\mathbf{H}}_j = \begin{bmatrix} -(\widetilde{\boldsymbol{\omega}}_i\mathbf{h}_{ij})\widetilde{\ }\mathbf{s}_{ji} - \widetilde{\mathbf{h}}_{ij}(\widetilde{\boldsymbol{\omega}}_i\mathbf{s}_{ji} + \widetilde{\mathbf{h}}_{ij}\mathbf{s}_{ji}\dot{q}_j) \\ \widetilde{\boldsymbol{\omega}}_i\mathbf{h}_{ij} \end{bmatrix} \tag{39}$$

$$\mathbf{B} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{B}_{12} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_{23} & \mathbf{0} & \dots & \mathbf{0} \\ \vdots & & \ddots & & \vdots \\ \mathbf{0} & \dots & & \mathbf{B}_{(n-1)n} & \mathbf{0} \end{bmatrix}_{6n\times 6n}, \tag{40}$$

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_1 & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{H}_2 & \dots & \mathbf{0} \\ \vdots & & \ddots & \vdots \\ \mathbf{0} & \dots & \mathbf{0} & \mathbf{H}_n \end{bmatrix}_{6n\times n}, \tag{41}$$

$$\boldsymbol{\Phi} = (\mathbf{I}-\mathbf{B})^{-1} = \tag{42}$$

$$= \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{B}_{12} & \mathbf{I} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{B}_{23}\mathbf{B}_{12} & \mathbf{B}_{23} & \mathbf{I} & \dots & \mathbf{0} \\ \vdots & & \ddots & \ddots & \vdots \\ \mathbf{B}_{(n-1)n}\dots\mathbf{B}_{12} & \dots & \dots & \mathbf{B}_{(n-1)n} & \mathbf{I} \end{bmatrix}_{6n\times 6n}$$

$$\dot{\mathbf{B}} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \dot{\mathbf{B}}_{12} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \dot{\mathbf{B}}_{23} & \mathbf{0} & \dots & \mathbf{0} \\ \vdots & & \ddots & & \vdots \\ \mathbf{0} & \dots & & \dot{\mathbf{B}}_{(n-1)n} & \mathbf{0} \end{bmatrix}_{6n\times 6n}, \tag{43}$$

$$\dot{\mathbf{H}} = \begin{bmatrix} \dot{\mathbf{H}}_1 & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \dot{\mathbf{H}}_2 & \dots & \mathbf{0} \\ \vdots & & \ddots & \vdots \\ \mathbf{0} & \dots & \mathbf{0} & \dot{\mathbf{H}}_n \end{bmatrix}_{6n\times n}. \tag{44}$$

$$\mathbf{M}_j = \begin{bmatrix} m_j\mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{J}_j \end{bmatrix}_{6\times 6}, \tag{45}$$

$$\mathbf{J}_j = \mathbf{A}_j\mathbf{J}'_j\mathbf{A}_j^T, \tag{46}$$

$$\mathbf{\Omega}_j = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -\widetilde{\omega}_j\mathbf{J}_j \end{bmatrix}_{6\times 6}, \tag{47}$$

$$\omega_j = \omega_i + \mathbf{h}_{ij}\dot{q}_j \tag{48}$$

$$\mathbf{F}_{\mathbf{G}j} = \begin{bmatrix} m_j\mathbf{g} \\ \mathbf{0} \end{bmatrix}_{6\times 1}, \tag{49}$$

$$\mathbf{F}_{\mathbf{Q}j} = \begin{bmatrix} \mathbf{P}_j \\ \mathbf{N}_j \end{bmatrix}_{6\times 1}. \tag{50}$$

Term $\mathbf{J}'_j$ denotes inertia matrix for body j with respect to its centroidal frame π'_j, $\mathbf{g}$ is gravity acceleration, $\mathbf{P}_j$ and $\mathbf{N}_j$ are force and torque acting on the center of mass of j-th link, respectively.

Parameters of the LWR 4+ simulation model

Mass and inertia parameters of the model are presented in Table (2). Moments of inertia are expressed in the centroidal frames π'. Links are numbered from the base (0) to the end effector (7).

Kinematic parameters (lengths in *m*, angles in *rad*):

- joint 1:
  $\mathbf{C}_{01} = \mathbf{R}_x(-\frac{\pi}{2})\mathbf{R}_z(-\frac{\pi}{2})$,
  $\mathbf{C}_{10} = \mathbf{I}$,
  $\mathbf{A}''_{01} = \mathbf{R}_z(q_1)$,

Table 2.

Parameters of the LWR 4+ simulation model

link	mass, kg	Moments of inertia, kgm^2		
		J_{xx}	J_{yy}	J_{zz}
1	3	$1{,}27 \cdot 10^{-2}$	$1{,}27 \cdot 10^{-2}$	$5{,}4 \cdot 10^{-3}$
2	3	$1{,}27 \cdot 10^{-2}$	$1{,}27 \cdot 10^{-2}$	$5{,}4 \cdot 10^{-3}$
3	3	$1{,}27 \cdot 10^{-2}$	$1{,}27 \cdot 10^{-2}$	$5{,}4 \cdot 10^{-3}$
4	2,5	$1{,}017 \cdot 10^{-3}$	$4{,}5 \cdot 10^{-3}$	$4{,}5 \cdot 10^{-3}$
5	2,5	$1{,}017 \cdot 10^{-3}$	$4{,}5 \cdot 10^{-3}$	$4{,}5 \cdot 10^{-3}$
6	1	$1{,}44 \cdot 10^{-3}$	$1{,}44 \cdot 10^{-3}$	$1{,}44 \cdot 10^{-3}$
7	0,5	$3{,}66 \cdot 10^{-4}$	$3{,}66 \cdot 10^{-4}$	$2{,}25 \cdot 10^{-4}$

$\mathbf{s}'_{01} = [0 \;\; 0 \;\; 0]^T$,
$\mathbf{s}'_{10} = [0 \;\; 0 \;\; -0{,}2105]^T$,

- joint 2:
  $\mathbf{C}_{12} = \mathbf{R}_z(\frac{\pi}{2})\mathbf{R}_y(-\frac{\pi}{2})$,
  $\mathbf{C}_{21} = \mathbf{R}_z(\frac{\pi}{2})\mathbf{R}_y(-\frac{\pi}{2})$,
  $\mathbf{A}''_{12} = \mathbf{R}_z(q_2)$,
  $\mathbf{s}'_{12} = [0 \;\; 0 \;\; 0{,}1]^T$,
  $\mathbf{s}'_{21} = [0 \;\; 0 \;\; -0{,}1]^T$,
- joint 3:
  $\mathbf{C}_{23} = \mathbf{I}$,
  $\mathbf{C}_{32} = \mathbf{I}$,
  $\mathbf{A}''_{23} = \mathbf{R}_z(q_3)$,
  $\mathbf{s}'_{23} = [0 \;\; 0 \;\; 0{,}1]^T$,
  $\mathbf{s}'_{32} = [0 \;\; 0 \;\; -0{,}1]^T$,
- joint 4:
  $\mathbf{C}_{34} = \mathbf{R}_z(\frac{\pi}{2})\mathbf{R}_y(-\frac{\pi}{2})$,
  $\mathbf{C}_{43} = \mathbf{R}_z(\frac{\pi}{2})\mathbf{R}_y(-\frac{\pi}{2})$,
  $\mathbf{A}''_{34} = \mathbf{R}_z(q_4)$,
  $\mathbf{s}'_{34} = [0 \;\; 0 \;\; 0{,}1]^T$,
  $\mathbf{s}'_{43} = [0 \;\; 0 \;\; -0{,}0975]^T$,
- joint 5:
  $\mathbf{C}_{45} = \mathbf{I}$,
  $\mathbf{C}_{54} = \mathbf{I}$,
  $\mathbf{A}''_{45} = \mathbf{R}_z(q_5)$,
  $\mathbf{s}'_{45} = [0 \;\; 0 \;\; 0{,}0975]^T$,
  $\mathbf{s}'_{54} = [0 \;\; 0 \;\; -0{,}0975]^T$,
- joint 6:
  $\mathbf{C}_{56} = \mathbf{R}_z(\frac{\pi}{2})\mathbf{R}_y(-\frac{\pi}{2})$,
  $\mathbf{C}_{65} = \mathbf{R}_z(\frac{\pi}{2})\mathbf{R}_y(-\frac{\pi}{2})$,
  $\mathbf{A}''_{56} = \mathbf{R}_z(q_6)$,
  $\mathbf{s}'_{56} = [0 \;\; 0 \;\; 0{,}0975]^T$,
  $\mathbf{s}'_{65} = [0 \;\; 0 \;\; 0]^T$,
- joint 7:
  $\mathbf{C}_{67} = \mathbf{I}$,
  $\mathbf{C}_{76} = \mathbf{I}$,
  $\mathbf{A}''_{67} = \mathbf{R}_z(q_6)$,
  $\mathbf{s}'_{67} = [0 \;\; 0 \;\; 0]^T$,
  $\mathbf{s}'_{76} = [0 \;\; 0 \;\; -0{,}0309]^T$.

All vectors defining axes of rotation are: $\mathbf{h}''_{ij} = [0 \; 0 \; 1]^T$.

Acknowledgements

This work has been supported by the National Science Center under grant no. DEC-2012/07/B/ST8/03993.

Manuscript received by Editorial Board, December 31, 2014;
final version, April 15, 2015.

REFERENCES

[1] Orlandea N., Chace M., and Calahan D.: "A sparcity-oriented approach to the dynamics analysis and design of mechanical systems - parts 1 and 2," Journal of Engineering for Industry, vol. 43, pp. 773–784, 1977.

[2] Armstrong W.W.: "Recursive solution to the equations of motion of an n-link manipulator," in Proceedings of the 5th World Congress on Theory of Machines and Mechanisms, pp. 1343–1346, 1979.

[3] Hollerbach J.: "A recursive Lagrangian formulation of manipulator dynamics and a comparative study of dynamics formulation complexity," IEEE Transactions on Systems, Man, and Cybernetics, vol. SMC-10(11), pp. 730–736, 1980.

[4] Anderson K.: "An order n formulation for the motion simulation of general multi-rigid-body constrained systems," Computer and Structures, vol. 3, pp. 565–579, 1992.

[5] Baeand D.S., Haug E.J.: "A recursive formulation for constrained mechanical system dynamics: Part I: Open loop systems," Mechanics of Structures and Machines, vol. 15, pp. 359–382, 1987.

[6] Baeand D.S., Haug E.J.: "A recursive formulation for constrained mechanical system dynamics: Part II: Closed loop systems," Mechanics of Structures and Machines, vol. 15, pp. 481–506, 1988.

[7] Baraff D.: "Linear-time dynamics using lagrange multipliers," in SIG-GRAPH 96, Computer Graphics Proceedings, pp. 137–146, 1996.

[8] Featherstone R.: "The calculation of robot dynamics using articulated-body inertias," International Journal of Robotics Research, vol. 2, pp. 13–30, 1983.

[9] Jain A.: "Unified formulation of dynamics for serial rigid multibody systems," Journal of Guidance, Control, and Dynamics, vol. 14, pp. 531–542, 1991.

[10] Rodriguez G.: "Kalman filtering, smoothing and recursive robot arm forward and inverse dynamics," IEEE Journal of Robotics and Automation, vol. 6, pp. 624–639, 1987.

[11] Rosenthal D.: "An order n formulation for robotic systems," The Journal of the Astronautical Sciences, vol. 38(4), pp. 511–529, 1990.

[12] Walkerand M.W., Orin D.E.: "Efficient dynamic computer simulation of robotic mechanisms," ASME Journal of Dynamic Systems, Measurements, and Control, vol. 104, pp. 205–211, 1982.

[13] Yamaneand K., Nakamura Y.: "O(N) forward dynamics computation of open kinematic chains based on the principle of virtual work," in Pro-ceedings of IEEE International Conference on Robotics and Automation, pp. 2824–2831, 2001.

[14] Anderson K., and Critchley J.: "Improved 'order-n' performance algorithm for the simulation of constrained multi-rigid-body dynamic systems," Multi-body System Dynamics, vol. 9, pp. 185–225, 2003.

[15] Naudet J.: "Forward dynamics of multibody systems: a recursive Hamiltonian approach." Phd Thesis, Universytet Vrije, Brussels, Belgium, 2005.

[16] Saha K.S., and Schiehlen W.: ,"Recursive kinematics and dynamics for par-allel structured closed-loop multibody systems," Mechanics of Structures and Machines, vol. 29(2), pp. 143–175, 2001.
[17] Stejskal V., and Valasek M.: Kinematics and Dynamics of Machinery. Marcel Dekker, NY, 1996.
[18] Kasahara H., Fujii H., and Iwata M.: "Parallel processing of robot motion simulation," in Proceedings IFAC World Congress, Munich, 1987.
[19] Bae D.S., Kuhl J., and Haug E.J.: "A recursive formulation for constrained mechanical system dynamics: Part III. parallel processor implementation," Mechanics of Structures and Machines, vol. 16, pp. 249–269, 1988.
[20] Chungand S., Haug E.J.: "Real-time simulation of multibody dynamics on shared memory multiprocessors," Journal of Dynamic Systems, Measurement, and Control, vol. 115, pp. 627–637, 1993.
[21] Fijanyand A., Bejczy A.K.: "Techniques for parallel computation of mechanical manipulator dynamics. part II: Forward dynamics," Control and Dynamics Systems, vol. 40, pp. 357–410, 1991.
[22] Fijany A., Sharf I., and D'Eleuterio G.: "Parallel O(log n) algorithms for computation of manipulator for ward dynamics," IEEE Transactionson Robotics and Automation, vol. 11, pp. 389–400, 1995.
[23] Leeand C., Chang P.: "Efficient parallel algorithms for robot forward dynamics," IEEE Transactionson Systems, Man, and Cybernetics, vol. 18(2), pp. 238–251, 1988.
[24] Featherstone R.: "A divide-and-conquer articulated body algorithm for parallel O(log n) calculation of rigid body dynamics. Part 1: Basic algorithm," International Journal of Robotics Research, vol. 18, pp. 867–875, 1999.
[25] Featherstone R.: "A divide-and-conquer articulated body algorithm for parallel O(log n) calculation of rigid body dynamics. Part 2: Trees, loops, and accuracy," International Journal of Robotics Research, vol. 18, pp. 876–892, 1999.
[26] Anderson K.S., and Duan S.: "Highly parallelizable low-order dynamics simulation algorithm for multi-rigid-body systems," Journal of Guidance, Control, and Dynamics, vol. 23, pp. 355–364, 2000.
[27] Critchley J., and Anderson K.: "A parallel logarithmic order algorithm for general multibody system dynamics," Multibody System Dynamics, vol. 12, pp. 75–93, 2004.
[28] Yamane K., and Nakamura Y.: "Parallel O(log n) algorithm for dynamics simulation of humanoid robots," in Proceedings of the 6th IEEE-RAS International Conference on Humanoid Robots, (Genova, Italy), pp. 554–559, 2006.
[29] Mukherjee R., and Anderson K.: "Orthogonal complement based divide-and-conquer algorithm for constrained multibody systems," Nonlinear Dynamics, vol. 48, pp. 199–215, 2007.
[30] Malczyk P., and Frączek J.: "Cluster computing of mechanisms dynamics using recursive formulation," Multibody System Dynamics, vol. 20(2), pp. 177–196, 2008.
[31] Malczyk P., and Frączek J.: "A divide and conquer algorithm for constrained multibody system dynamics based on augmented lagrangian method with projections-based error correction," Nonlinear Dynamics, vol. 70(1), pp. 871–889, 2012.
[32] Yamane K., and Nakamura Y.: "Comparative study on serial and parallel forward dynamics algorithms for kinematic chains," International Journal of Robotics Research, vol. 28(5), pp. 622–629, 2009.
[33] Featherstone R.: Rigid Body Dynamics Algorithms. Springer, 2008.
[34] Jain A.: Robot and Multibody Dynamics: Analysis and Algorithms. Springer; 1st Edition, 2011.
[35] Rodriguez G., Jain A., and Kreutz-Delgado K.: "Spatial operator algebra for manipulator modelling and control," International Journal of Robotics Research, vol. 10(4), pp. 371–381, 1991.

[36] Craig J.: Introduction to Robotics. Mechanics & Control. Addison-Wesley Publishing Company, 1986.
[37] Bae D.S., and Haug E.J.: "A recursive for mulation for constrained mechanical system dynamics: Part I: Open loop systems," Mechanics of Structures and Machines, vol. 15, pp. 359–382, 1987.
[38] Jain A.: Robot and Multibody Dynamics: Analysis and Algorithms. Springer, 2011.
[39] Malczyk P., and Frączek J.: "Cluster computing of mechanisms dynamics using recursive formulation," Multibody System Dynamics, vol. 20(2), pp. 177–196, 2008.
[40] Gaz C., Flacco F., and Luca A.D.: "Identifying the dynamic model used by the kuka lwr: A reverse engineering approach," in Proceedings of the IEEE International Conference on Robotics and Automation 2014, (Hong Kong, China), pp. 1386–1392, May 31- June 7 2014.
[41] Jubien A., Gautier M., and Janot A.: "Dynamic identificatio of the kuka lightweight robot: comparison between actual and confidetial kuka'sparameters," in Proceedings of the IEEE/ASME International Conference on Advanced Intelligent Mechatronics 2014, (Besancon, France), pp. 483–488, July 8-11 2014.
[42] "Lightweight robot 4+ specification, version: Spez lbr 4+ v2en," Issued: 06.07.2010, 2010.
[43] "Adams/solver (c++) statements. integrator," Adams 2013.1 Help, 2013.
[44] Scilab Help (https://help.scilab.org/docs/5.5.1/en_US/ode.html), 2014.
[45] Hindmarsh A.: "Serial fortran solvers for ode initial value problems," http://computation.llnl.gov/casc/odepack/, 2014.
[46] Malczyk P., Frączek J., and Cuadrado J.: "Parallel index-3 formulation for real-time multibody dynamics simulations," in Proceedings of the First Joint International Conference on Multibody System Dynamics-IMSD 2010, ISBN978-952-214-778-3, (Lappeenranta, Finland), 25-27 May 2010.
[47] Critchley J.: "A parallel logarithmic time complexity algorithm for the simulation of general multibody system dynamics." Praca doktorska, Rensselaer Polytechnic Institute, Troy, New York, USA, 2003.
[48] Gonzales M., Dopico D., Lugris U., and Cuadrado J.: "A benchmarking system for mbs simulation software: Problem standarization and performance measurement," Multibody System Dynamics, vol. 16, pp. 179—190, 2006.
[49] IFToMM: "Library of computational benchmark problems iftomm technical committee for multibody dynamics." Available online at http://iftomm-multibody.org/benchmark/, May 2015.
[50] Schreiber G., Stemmer A., and Bischo˙ R.: "The fast research interface for the kuka lightweight robot," in Proceedings of the IEEE ICRA 2010 Workshop on ICRA 2010 Workshop on Innovative Robot Control Architectures for Demanding (Research) Applications – How to Modify and Enhance Commercial Controllers, (Anchorage), pp. 15–21, May 2010.

Analiza dynamiki manipulatora w przestrzeni współrzędnych złączowych

S t r e s z c z e n i e

W artykule przedstawiono efektywny algorytm do analizy dynamiki manipulatora przestrzennego o otwartym łańcuchu kinematycznym. Równania opisujące dynamikę układu zapisano w formie algebraiczno-macierzowej w przestrzeni współrzędnych złączowych. Wprowadzona zwarta reprezentacja równań opisujących ruch manipulatora pozwoliła na rozwiązanie zadania prostego i odwrotnego dynamiki manipulatora w rekursywny i wydajny sposób. Algorytm uogólniono na przypadki

analizy dynamiki otwartych łańcuchów kinematycznych z dowolną liczbą stopni swobody. Opracowane sformułowanie zastosowano do analizy dynamiki manipulatora KUKA LWR o siedmiu stopniach swobody. Zweryfikowano poprawność obliczeń numerycznych dla testowych przypadków ruchu manipulatora, a wyniki porównano z rezultatami otrzymanymi w pakiecie komercyjnym do obliczeń dynamiki układów wieloczłonowych.