Calculation of achievable robot joint accelerations based on a new robot forward dynamics algorithm

Vladimir Kvrgic, Jelena Vidakovic

***Abstract*— An algorithm that calculates the feasible robot joints' accelerations based on a new forward dynamics algorithm while considering the actuators' force/torque saturations and achieves a realistic simulation of robot movements is given in this paper. While the most used forward dynamics algorithm in the literature, Walker and Orin's Method 1, calculates robot forward dynamics by executing Recursive Newton-Euler Algorithm (RNEA) n + 1 times, where n is the number of degrees-of-freedom (DoFs), algorithm used here solves forward dynamics using the modified RNEA (mRNEA) only once. Owing to that, this algorithm is very efficient. Furthermore, the computational complexity of the algorithm is even more significant when used for robot simulation as it does not require calculating joint torques as inputs for forward dynamics, unlike other methods. Another benefit of the proposed method is the ease of development and implementation for a specific robot. The proposed mRNEA and its application within the forward dynamics algorithm are demonstrated using a serial 4-DoF spatial disorientation trainer as an example.**

***Index Terms*—Robot, Forward dynamics, Joint accelerations, Simulation system, Recursive Newton–Euler algorithm**

I. Introduction

A robot simulation verifies the feasibility of programmed movements, and if necessary modifies them. It also calculates the values of forces and moments acting on robot links and joints that is essential in robot design. For this, robot simulation has to solve robot forward and inverse dynamics problems. Forward dynamics (FD) solves the motion from the forces, while inverse dynamics (ID) solves the forces from the motion [1]. ID is used within dynamic model-based control methods, and for FD calculations. FD is used mainly in simulation purposes.

FD calculates the joint accelerations $\ddot{\mathbf{q}}(t_k)$ at a time instant t_k, the joint velocities $\dot{\mathbf{q}}(t_{k+1})$ of the next interpolation cycle time (Δt) and joint positions $\mathbf{q}(t_{k+1})$ at the end of the next Δt. FD accounts for the joint torques $\mathbf{u}(t_k)$; the inertial, gravitational, and Coriolis forces of the robot links; forces and moments acting on the end effector; and the friction forces and moments of the joints.

When a robot is considered as a continuous nonlinear system, after obtaining $\ddot{\mathbf{q}}(t_k)$, the velocity $\dot{\mathbf{q}}(t_{k+1})$ and position $\mathbf{q}(t_{k+1})$, $t_{k+1} = t_k + \Delta t$ are computed using a numerical integration method, i.e., Runge–Kutta, with an integration step Δt [2]. On the other hand, ID determines the joint torques $\mathbf{u}(t_k)$ at time instant t_k which are required to generate the motion specified by the joint accelerations, and consequently, the velocities and positions. This is accomplished by using the current velocities, current positions, the forces and moments acting on the end effector, and the friction forces and moments of the joints.

One of the challenges in robot simulation is to derive algorithms that are computationally efficient and are also easy to apply to a specific robot.

The ID of a manipulator with n DoFs can be solved by well-known equations of motion which represent its joint space dynamic model

$$\mathbf{H}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) + \mathbf{J}(\mathbf{q})^{\mathrm{T}}\mathbf{k}_e = \mathbf{u}, \tag{1}$$

where $\ddot{\mathbf{q}}$, $\dot{\mathbf{q}}$, and $\mathbf{q}$ are $n \times 1$ vectors of the joint accelerations, velocities, and positions, respectively; $\mathbf{H}(\mathbf{q})$ is $n \times n$ generalized robot mass (inertia) matrix; $\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})$ is an $n \times n$ matrix specifying the centrifugal and Coriolis effects; $\mathbf{g}(\mathbf{q})$ is an $n \times 1$ vector of gravity terms; $\mathbf{k}_e$ is a 6×1 vector of the external forces and moments on link n; $\mathbf{J}(\mathbf{q})$ is a $6 \times n$ Jacobian matrix; and $\mathbf{u}$ is an $n \times 1$ vector of the input joint torques/forces. The diagonal terms of the mass matrix are related to the inertias of the corresponding DoF, and the off-diagonal terms express the inertial couplings between the DoFs [3].

From Eq. (1), it can be seen that for time instant t_k, the joint torques/forces are linear functions of the joint accelerations $\ddot{\mathbf{q}}(t_k)$ when $\mathbf{q}(t_k)$ and $\dot{\mathbf{q}}(t_k)$ are given. These equations can be obtained explicitly with the Lagrange formulation (LF) which contains the matrix $\mathbf{H}(\mathbf{q})$ and vectors $\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}}$, $\mathbf{g}(\mathbf{q})$, and $\mathbf{J}(\mathbf{q})^{\mathrm{T}}\mathbf{k}_e$. Consequently, the joint accelerations $\ddot{\mathbf{q}}(t_k)$ can be computed by solving the following system of n linear equations

$$\mathbf{H}(\mathbf{q})\ddot{\mathbf{q}} = \mathbf{u} - \mathbf{u}'(\mathbf{q},\dot{\mathbf{q}},\mathbf{k}_e), \tag{2}$$

$$\mathbf{u}'(\mathbf{q},\dot{\mathbf{q}},\mathbf{k}_e) = \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) + \mathbf{J}(\mathbf{q})^{\mathrm{T}}\mathbf{k}_e. \tag{3}$$

The LF method for derivation of robot equations of motion provides a compact analytical form containing the mass matrix $\mathbf{H}(\mathbf{q})$, and a bias vector $\mathbf{u}'$ that denotes joint torque contributions that do not correlate with the joint accelerations

Vladimir Kvrgic is with the Institute Mihajlo Pupin, University of Belgrade, 15 Volgina, 11060 Belgrade, Serbia, vladimir.kvrgic@pupin.rs.

Jelena Vidakovic is with the Lola Institute, Kneza Viseslava 70a, 11030 Belgrade, Serbia, jelena.vidakovic@li.rs).

[4–10]. Therefore, when the LF is used, the joint accelerations $\ddot{\mathbf{q}}(t_k)$ (within one interpolation cycle) can be computed by solving the system of n linear equations, where n is number of manipulator DoFs. Although it is not complex to solve FD using LF, this method is typically used for manipulators when $n \leq 3$ because of the very high computational complexity of the LF: O(n^4).

Robot dynamic models should be derived in recursive form in order to be computational efficient [5]. In contrast to the LF, the computational complexity of the RNEA is O(n). Reference [11] developed a recursive LF; however, the computational complexity of the recursive LF is O(n^3).

Walker and Orin [6,7] employed the RNEA for computing the FD and presented four methods to solve the joint accelerations. Their method 1 (WO method 1) remains the simplest and the most recommended in the literature [2,5]. According to this method, torque $\mathbf{u}'$ is computed using the RNEA. Further, each column $\mathbf{h}_i$, i = 1 to n, of matrix $\mathbf{H}$ is computed as the torque vector given by the RNEA.

In [12], a modified Recursive Newton-Euler Algorithm (mRNEA) for derivation of dynamic model of robot manipulator is presented. The mRNEA gives explicitly the mass matrix $\mathbf{H}$ and the bias vector $\mathbf{u}'$, in a similar manner as LF. Owing to that, it is easy to use in the FD computation, which executes the mRNEA only once.

The additional calculation of the input-joint-torque $\mathbf{u}$, Eqs. (2), had to be performed within a robot simulation system. With the method presented in [12], input-joint-torque calculation is performed within the FD algorithm, and therefore, the computational complexity of the simulation system is additionally reduced. As a result, the simulation system has to solve ID only once within each interpolation cycle, in comparison to simulation systems which use, for example, WO method 1, and which solve ID n + 2 times.

Proposed FD algorithm is computationally very efficient, with O(n) complexity.

The rest of the paper is organized as follows. Section II presents the proposed approach for efficient FD calculation and its implementation in a robot simulation system. Section III depicts the proposed FD algorithm for open-chain manipulators with n DoFs. A toy model for the 4-DoF spatial disorientation trainer (SDT) is also presented in Section IV. The FD algorithm, which calculates the achievable motor velocities in each interpolation cycle based on the actuator torque/force saturations, is presented in Section V. Finally, concluding remarks are given in Section VI.

II. Achievable Joint Accelerations Calculation

A path planner of the robot controller transforms the motion commands into a series of successive positions of robot joints/actuators. As they are sent to the servo controller at constant time intervals Δt, they correspond to the desired joint/actuator velocities of $\dot{q}_i(t_k)=(q_i(t_{k+1})-q_i(t_k))/\Delta t$, which can be considered constant within each Δt (up-to-date controllers have Δt between 0.01 s and 0.003 s). Thereafter, the path planer sends the desired joint velocities to the speed controllers of the actuators, whose task is to keep them constant within each Δt.

Since each joint velocity can be considered constant within each Δt, and since the current velocity $\dot{q}_i(t_k)$ and given acceleration $\ddot{q}_i(t_k)$ are known (calculated within the path planer), the joint velocity in the next interpolation cycle is $\dot{q}_i(t_{k+1})=\dot{q}_i(t_k)+\ddot{q}_i(t_k)\Delta t$, which is depicted in Fig. 1.

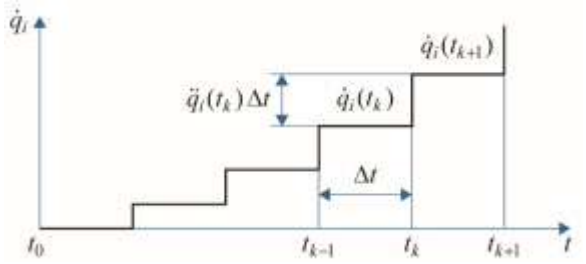

Fig. 1. Example of given joint velocity change.

Herein, a method for obtaining equation (2) explicitly with the mRNEA is proposed. Consequently, a method which calculates $\ddot{\mathbf{q}}(t_k)$ using the mRNEA only once is proposed. The presented FD algorithm utilizes the current values of $\mathbf{q}(t_k)$ and $\dot{\mathbf{q}}(t_k)$, the given values of $\mathbf{q}(t_{k+1})$ and $\dot{\mathbf{q}}(t_{k+1})$ calculated in the path interpolator, and $\mathbf{k}_e$. First, it checks if the desired positions and velocities are feasible. If they are not, it limits their values in accordance with their maximum/minimum possible values. Based on this, the algorithm calculates the desired joint accelerations $\ddot{q}_i(t_k)=(\dot{q}_i(t_{k+1})-\dot{q}_i(t_k))/\Delta t$. Next, the mRNEA calculates the joint torques/forces $\mathbf{u}$ required for the desired joint motions. In the next step, the algorithm calculates the required actuator torques $\mathbf{u}_a$, whose capabilities are examined. Unachievable torques/forces are replaced with the maximum/minimum possible, with the aim that the FD algorithm determines the achievable accelerations. Other joint accelerations keep their values obtained from the path interpolator.

Herein, within the simulation system, only attainable motor velocities and positions are sent from the path planner to the speed controller during each Δt. Consequently, joint forces and moments are calculated based on the attainable velocities and accelerations, so that their realistic values are obtained.

The FD simulation can be used in a stage of the robot design process, in which case it enables the proper design of bearings and links.

III. FD Algorithm Based on mRNEA

Herein, the FD algorithm based on mRNEA for open–chain manipulators with n DoFs is presented.

A 4 × 4 homogenous transformation matrix (HTM) that transforms point coordinates from frame j to frame i is ${}^{j}\mathbf{T}_i$, and from the base frame to frame i is $\mathbf{T}_i$. The matrix ${}^{j}\mathbf{T}_i$ contains a 3 × 3 orientation matrix ${}^{j}\mathbf{D}_i=\left[\begin{array}{ccc} {}^{j}\mathbf{x}_i & {}^{j}\mathbf{y}_i & {}^{j}\mathbf{z}_i \end{array}\right]$ and a 3 x 1 position vector ${}^{j}\mathbf{p}_i$.

The linear acceleration of the robot link i centre of mass is

$$\dot{\mathbf{v}}_i^{\mathrm{cm}} = \left[\dot{v}_{xi}^{\mathrm{cm}} \;\; \dot{v}_{yi}^{\mathrm{cm}} \;\; \dot{v}_{zi}^{\mathrm{cm}}\right]^{\mathrm{T}} = \dot{\mathbf{v}}_i + \dot{\boldsymbol{\omega}}_i \times \mathbf{r}_i^{\mathrm{cm}} + \boldsymbol{\omega}_i \times (\boldsymbol{\omega}_i \times \mathbf{r}_i^{\mathrm{cm}}), \quad (4)$$

where $\mathbf{r}_i^{\mathrm{cm}} = \left[r_{xi}^{\mathrm{cm}} \;\; r_{yi}^{\mathrm{cm}} \;\; r_{zi}^{\mathrm{cm}}\right]^{\mathrm{T}} = \left[r_{xi} \;\; r_{yi} \;\; r_{zi}\right]^{\mathrm{T}} = \mathbf{D}_i \hat{\mathbf{r}}_i^{\mathrm{cm}}$ is the position of the link i centre of mass with respect to the coordinates of link i expressed in the base coordinates. This vector in the coordinates of link i is $\hat{\mathbf{r}}_i^{\mathrm{cm}} = \left[\hat{r}_{xi} \;\; \hat{r}_{yi} \;\; \hat{r}_{zi}\right]^{\mathrm{T}}$. A vector cross product is denoted with ×, and $\boldsymbol{\omega}_i$, $\dot{\boldsymbol{\omega}}_i$, and $\dot{\mathbf{v}}_i$ are the link angular velocity, angular acceleration, and linear acceleration, respectively, i = 1 to n. Equation (4) can be rewritten as

$$\dot{\mathbf{v}}_i^{\mathrm{cm}} = \sum_{k=1}^{i} \mathbf{b}_{ik} \ddot{q}_k + \sum_{k=1}^{i} \sum_{j=k}^{i} \mathbf{b}_{ikj} \dot{q}_k \dot{q}_j \;, \; i = 1 \text{ to } n, \quad (5)$$

where $\mathbf{b}_{ik}$ and $\mathbf{b}_{ikj}$ are 3 × 1 vectors. The total force $\mathbf{F}_i$ and total moment $\mathbf{N}_i$ exerted on link i, obtained from the NE equations, are

$$\mathbf{F}_i = \left[F_{xi} \;\; F_{yi} \;\; F_{zi}\right]^{\mathrm{T}} = m_i \left[\dot{v}_{xi}^{\mathrm{cm}} \;\; \dot{v}_{yi}^{\mathrm{cm}} \;\; \dot{v}_{zi}^{\mathrm{cm}} - g\right]^{\mathrm{T}}, \quad (6)$$

$$\mathbf{N}_i = \left[N_{xi} \;\; N_{yi} \;\; N_{zi}\right]^{\mathrm{T}} = \mathbf{I}_i^{\mathrm{cm}} \dot{\boldsymbol{\omega}}_i + \boldsymbol{\omega}_i \times (\mathbf{I}_i^{\mathrm{cm}} \boldsymbol{\omega}_i). \quad (7)$$

The mass of link i is denoted as m_i, g is Earth's acceleration and $\mathbf{I}_i^{\mathrm{cm}}$ is the 3 × 3 moment of the inertia matrix of link i about the centre of mass of that link expressed in the base coordinates. Equations (5) and (6) yield

$$\mathbf{F}_i = m_i \left(\left[0 \;\; 0 \;\; -g\right]^{\mathrm{T}} + \sum_{k=1}^{i} \mathbf{b}_{ik} \ddot{q}_k + \sum_{k=1}^{i} \sum_{j=k}^{i} \mathbf{b}_{ikj} \dot{q}_k \dot{q}_j\right), \; i = 1 \text{ to } n. \quad (8)$$

Equation (7) can be rewritten as

$$\mathbf{N}_i = \sum_{k=1}^{i} \mathbf{d}_{ik} \ddot{q}_k + \sum_{k=1}^{i} \sum_{j=k}^{i} \mathbf{d}_{ikj} \dot{q}_k \dot{q}_j \;, \; i = 1 \text{ to } n, \quad (9)$$

where $\mathbf{d}_{ik}$ and $\mathbf{d}_{ikj}$ are 3 × 1 vectors. The effects of the external forces and moments, $\mathbf{k}_{\mathrm{e}}^{\mathrm{T}} = \left[\mathbf{f}_{\mathrm{e}} \;\; \mathbf{n}_{\mathrm{e}}\right]^{\mathrm{T}}$, acting on the end effector are well-known as

$$\mathbf{f}_n = \mathbf{F}_n + \mathbf{f}_{\mathrm{e}}, \quad (10)$$

$$\mathbf{n}_n = \mathbf{N}_i + \mathbf{n}_{\mathrm{e}} + \left[p_{\mathrm{ey}} f_{\mathrm{ez}} - p_{\mathrm{ez}} f_{\mathrm{ey}} \;\; p_{\mathrm{ez}} f_{\mathrm{ex}} - p_{\mathrm{ex}} f_{\mathrm{ez}} \;\; p_{\mathrm{ex}} f_{\mathrm{ey}} - p_{\mathrm{ey}} f_{\mathrm{ex}}\right]^{\mathrm{T}}, \quad (11)$$

where $\mathbf{p}_{\mathrm{e}} = \left[p_{\mathrm{ex}} \;\; p_{\mathrm{ey}} \;\; p_{\mathrm{ez}}\right]^{\mathrm{T}} = \mathbf{D}_n \hat{\mathbf{p}}_{\mathrm{e}}$ is the position of the external force with respect to the coordinates of link n expressed in the base coordinates. This vector in the coordinates of link n is $\hat{\mathbf{p}}_{\mathrm{e}} = \left[\hat{p}_{\mathrm{ex}} \;\; \hat{p}_{\mathrm{ey}} \;\; \hat{p}_{\mathrm{ez}}\right]^{\mathrm{T}}$. Equations (8) and (10) yield

$$\mathbf{F}_i = \mathbf{e}_i + \sum_{k=1}^{i} \mathbf{e}_{ik} \ddot{q}_k, \quad (12)$$

$$\mathbf{e}_i = m_i \left(\left[0 \;\; 0 \;\; -g\right]^{\mathrm{T}} + \sum_{k=1}^{i} \sum_{j=k}^{i} \mathbf{b}_{ikj} \dot{q}_k \dot{q}_j\right), \; i = 1 \text{ to } n-1, \quad (13)$$

$$\mathbf{e}_n = m_n \left(\left[0 \;\; 0 \;\; -g\right]^{\mathrm{T}} + \sum_{k=1}^{n} \sum_{j=k}^{n} \mathbf{b}_{ikj} \dot{q}_k \dot{q}_j\right) + \mathbf{f}_{\mathrm{e}}, \; i = n, \quad (14)$$

$$\mathbf{e}_{ik} = m_i \mathbf{b}_{ik}, \; i = 1 \text{ to } n. \quad (15)$$

Similarly, Eq. (9) can be replaced with

$$\mathbf{N}_i = \mathbf{d}_i + \sum_{k=1}^{i} \mathbf{d}_{ik} \ddot{q}_k \;, \quad (16)$$

$$\mathbf{d}_i = \sum_{k=1}^{i} \sum_{j=k}^{i} \mathbf{d}_{ikj} \dot{q}_k \dot{q}_j \;, \; i = 1 \text{ to } n. \quad (17)$$

From robot dynamics, the force $\mathbf{f}_i$ and moment $\mathbf{n}_i$ exerted on link i by link i – 1 in the base coordinate frame, is well–known to be

$$\mathbf{f}_i = \left[f_{xi} \;\; f_{yi} \;\; f_{zi}\right]^{\mathrm{T}} = \mathbf{F}_i + \mathbf{f}_{i+1}, \quad (18)$$

$$\mathbf{n}_i = \left[n_{xi} \;\; n_{yi} \;\; n_{zi}\right]^{\mathrm{T}} = \mathbf{n}_{i+1} + \mathbf{N}_i + \mathbf{l}_i \times \mathbf{F}_i + \mathbf{p}_i^* \times \mathbf{f}_{i+1}, \quad (19)$$

where $\mathbf{l}_i = \left[l_{xi} \;\; l_{yi} \;\; l_{zi}\right]^{\mathrm{T}} = \mathbf{p}_i^* + \mathbf{r}_i^{\mathrm{cm}}$, $\mathbf{p}_{i+1}^* = \mathbf{p}_{i+1} - \mathbf{p}_i$.

The mass of the end effector can be included in the mass of link n. In accordance with Eqs. (10)–(19), $\mathbf{f}_i$ and $\mathbf{n}_i$ can be calculated as

$$\mathbf{f}_i = \sum_{k=i}^{n} \mathbf{e}_k + \sum_{k=1}^{i} \sum_{j=i}^{n} \mathbf{e}_{jk} \ddot{q}_k + \sum_{k=i+1}^{n} \sum_{j=k}^{n} \mathbf{e}_{jk} \ddot{q}_k \;, \; i = n \text{ to } 1, \quad (20)$$

$$\begin{aligned} \mathbf{n}_i &= \mathbf{n}_{i+1} + \mathbf{d}_i + \sum_{k=1}^{i} \mathbf{d}_{ik} \ddot{q}_k + \mathbf{l}_i \times \left(\mathbf{e}_i + \sum_{k=1}^{i} \mathbf{e}_{ik} \ddot{q}_k\right) \\ &+ \mathbf{p}_i^* \times \left(\sum_{k=i+1}^{n} \mathbf{e}_k + \sum_{k=1}^{i+1} \sum_{j=i+1}^{n} \mathbf{e}_{jk} \ddot{q}_k + \sum_{k=i+2}^{n} \sum_{j=k}^{n} \mathbf{e}_{jk} \ddot{q}_k\right) \\ &= \mathbf{n}_{i+1} + \mathbf{n}_{i\mathrm{c}} + \mathbf{n}_{i\mathrm{a}1} + \mathbf{n}_{i\mathrm{a}2} + \mathbf{n}_{i\mathrm{a}3}, \; i = n \text{ to } 1, \end{aligned} \quad (21)$$

where $\mathbf{n}_{i\mathrm{c}}$, $\mathbf{n}_{i\mathrm{a}1}$, $\mathbf{n}_{i\mathrm{a}2}$, and $\mathbf{n}_{i\mathrm{a}3}$ are 3 × 1 vectors, as follows:

$$\mathbf{n}_{nc}=\mathbf{d}_n+\mathbf{l}_n\times\mathbf{e}_n+\mathbf{p}_{n+1}\times\mathbf{f}_{n+1}$$

$$=\begin{bmatrix} d_{xn}+l_{yn}e_{zn}-l_{zn}e_{yn}+p_{y(n+1)}f_{z(n+1)}-p_{z(n+1)}f_{y(n+1)} \\ d_{yn}+l_{zn}e_{xn}-l_{xn}e_{zn}+p_{z(n+1)}f_{x(n+1)}-p_{x(n+1)}f_{z(n+1)} \\ d_{zn}+l_{xn}e_{yn}-l_{yn}e_{xn}+p_{x(n+1)}f_{y(n+1)}-p_{y(n+1)}f_{x(n+1)} \end{bmatrix}, \; i = n, \quad (22)$$

$$\mathbf{n}_{ic}=\mathbf{d}_i+\mathbf{l}_i\times\mathbf{e}_i+\mathbf{p}_i^*\times\mathbf{E}_{i+1}$$

$$=\begin{bmatrix} d_{xi}+l_{yi}e_{zi}-l_{zi}e_{yi}+p_{yi}^*E_{z(i+1)}-p_{zi}^*E_{y(i+1)} \\ d_{yi}+l_{zi}e_{xi}-l_{xi}e_{zi}+p_{zi}^*E_{x(i+1)}-p_{xi}^*E_{z(i+1)} \\ d_{zi}+l_{xi}e_{yi}-l_{yi}e_{xi}+p_{xi}^*E_{y(i+1)}-p_{yi}^*E_{x(i+1)} \end{bmatrix},$$

$$\mathbf{E}_{i+1}=\sum_{k=i+1}^{n}\mathbf{e}_k \;, \; i = n-1 \text{ to } 1, \quad (23)$$

$$\mathbf{n}_{ia1}=\sum_{k=1}^{i}(\mathbf{d}_{ik}+\mathbf{l}_i\times\mathbf{e}_{ik})\ddot{q}_k=\begin{bmatrix} \sum_{k=1}^{i}(d_{xik}+l_{yi}e_{zik}-l_{zi}e_{yik})\ddot{q}_k \\ \sum_{k=1}^{i}(d_{yik}+l_{zi}e_{xik}-l_{xi}e_{zik})\ddot{q}_k \\ \sum_{k=1}^{i}(d_{zik}+l_{xi}e_{yik}-l_{yi}e_{xik})\ddot{q}_k \end{bmatrix}$$

$$=[\mathbf{n}_{i1a1}\;\mathbf{n}_{i2a1}\;....\;\mathbf{n}_{iia1}][\ddot{q}_1\;\ddot{q}_2\;\cdots\;\ddot{q}_i]^{\mathrm{T}},$$

$$[\mathbf{n}_{i1a1}\;\mathbf{n}_{i2a1}\;....\;\mathbf{n}_{iia1}]=[\mathbf{d}_{i1}+\mathbf{l}_i\times\mathbf{e}_{i1}\;\mathbf{d}_{i2}+\mathbf{l}_i\times\mathbf{e}_{i2}\;....\;\mathbf{d}_{ii}+\mathbf{l}_i\times\mathbf{e}_{ii}], \quad (24)$$
$i = n$ to 1,

$$\mathbf{n}_{ia2}=\mathbf{p}_i^*\times\sum_{k=1}^{i+1}\mathbf{E}_{(i+1)k}\ddot{q}_k=\begin{bmatrix} \sum_{k=1}^{i+1}(p_{yi}^*E_{z(i+1)k}-p_{zi}^*E_{y(i+1)k})\ddot{q}_k \\ \sum_{k=1}^{i+1}(p_{zi}^*E_{x(i+1)k}-p_{xi}^*E_{z(i+1)k})\ddot{q}_k \\ \sum_{k=1}^{i+1}(p_{xi}^*E_{y(i+1)k}-p_{yi}^*E_{x(i+1)k})\ddot{q}_k \end{bmatrix}$$

$$=[\mathbf{n}_{i1a2}\;\mathbf{n}_{i2a2}\cdots\mathbf{n}_{i(i+1)a2}][\ddot{q}_1\;\ddot{q}_2\cdots\ddot{q}_{i+1}]^{\mathrm{T}},$$

$$[\mathbf{n}_{i1a2}\;\mathbf{n}_{i2a2}\;\cdots\;\mathbf{n}_{i(i+1)a2}]=[\mathbf{p}_i^*\times\mathbf{E}_{(i+1)1}\;\mathbf{p}_i^*\times\mathbf{E}_{(i+1)2}\;\cdots\;\mathbf{p}_i^*\times\mathbf{E}_{(i+1)(i+1)}],$$

$$\mathbf{E}_{(i+1)k}=\sum_{j=i+1}^{n}\mathbf{e}_{jk} \;, \; i = n-1 \text{ to } 1, \; \mathbf{n}_{na2}=\mathbf{0}, \quad (25)$$

$$\mathbf{n}_{ia3}=\mathbf{p}_i^*\times\sum_{k=i+2}^{n}\mathbf{E}_{kk}\ddot{q}_k=\begin{bmatrix} \sum_{k=i+2}^{n}(p_{yi}^*E_{zkk}-p_{zi}^*E_{ykk})\ddot{q}_k \\ \sum_{k=i+2}^{n}(p_{zi}^*E_{xkk}-p_{xi}^*E_{zkk})\ddot{q}_k \\ \sum_{k=i+2}^{n}(p_{xi}^*E_{ykk}-p_{yi}^*E_{xkk})\ddot{q}_k \end{bmatrix}$$

$$=[\mathbf{0}\;\mathbf{0}\;\ldots\;\mathbf{n}_{i(i+2)a3}\;\ldots\;\mathbf{n}_{ina3}][\ddot{q}_1\;\ddot{q}_2\;\cdots\;\ddot{q}_{i+2}\;\cdots\;\ddot{q}_n]^{\mathrm{T}},$$

$$[\mathbf{0}\;\mathbf{0}\;\ldots\;\mathbf{n}_{i(i+2)a3}\;\ldots\;\mathbf{n}_{ina3}]=[\mathbf{0}\;\mathbf{0}\;\ldots\;\mathbf{p}_i^*\times\mathbf{E}_{(i+2)(i+2)}\;\ldots\;\mathbf{p}_i^*\times\mathbf{E}_{nn}],$$

$$\mathbf{E}_{kk}=\sum_{j=k}^{n}\mathbf{e}_{jk} \;, \; i = n-2 \text{ to } 1, \; \mathbf{n}_{na3}=\mathbf{n}_{(n-1)a3}=\mathbf{0}. \quad (26)$$

In order to reduce the number of counts, the vector $\mathbf{n}_{i+1}$ can be included in the vectors $\mathbf{n}_{ic}$, $\mathbf{n}_{ia1}$, $\mathbf{n}_{ia2}$, and $\mathbf{n}_{ia3}$. In this way, Eq. (21) is transformed into

$$\mathbf{n}_i=\mathbf{n}'_{ic}+(\mathbf{n}'_{ia1}+\mathbf{n}'_{ia2}+\mathbf{n}'_{ia3})\,\ddot{\mathbf{q}} \;, \; i = n \text{ to } 1, \quad (27)$$

where $\ddot{\mathbf{q}}=[\ddot{q}_1\;\ddot{q}_2\;....\;\ddot{q}_n]^{\mathrm{T}}$ is an $n \times 1$ vector, $\mathbf{n}'_{ic}$ is a 3×1 vector, and $\mathbf{n}'_{ia1}$, $\mathbf{n}'_{ia2}$, and $\mathbf{n}'_{ia3}$ are $3 \times n$ vectors. They are given in the following equations:

$$\mathbf{n}'_{ic}=\mathbf{n}_{(i+1)c}+\mathbf{n}_{ic}, \quad (28)$$

$$\mathbf{n}'_{ia1}=[\mathbf{n}'_{i1a1}\;\mathbf{n}'_{i2a1}\;....\;\mathbf{n}'_{ina1}]$$

$$=\left[\sum_{k=i}^{n}(\mathbf{d}_{k1}+\mathbf{l}_k\times\mathbf{e}_{k1})\;\sum_{k=i}^{n}(\mathbf{d}_{k2}+\mathbf{l}_k\times\mathbf{e}_{k2})\;....\;\mathbf{d}_{nn}+\mathbf{l}_n\times\mathbf{e}_{nn}\right], \quad (29)$$

$$\mathbf{n}'_{ia2}=\left[\sum_{k=i}^{n-1}\mathbf{n}_{k1a2}\;\sum_{k=i}^{n-1}\mathbf{n}_{k2a2}\;\sum_{k=i}^{n-1}\mathbf{n}_{k3a2}\;\sum_{k=i}^{n-1}\mathbf{n}_{k4a2}\;....\;\mathbf{n}_{(n-1)(n-1)a2}\right]$$

$$=\left[\sum_{k=i}^{n-1}(\mathbf{p}_k^*\times\mathbf{E}_{(k+1)1})\;\sum_{k=i}^{n-1}(\mathbf{p}_k^*\times\mathbf{E}_{(k+1)2})\;\sum_{k=i}^{n-1}(\mathbf{p}_k^*\times\mathbf{E}_{(k+1)3})\right. \quad (30)$$

$$\left.\sum_{k=i}^{n-1}(\mathbf{p}_k^*\times\mathbf{E}_{(k+1)4})\;....\;\mathbf{n}_{(n-1)(n-1)a2}\right],$$

$$\mathbf{n}'_{ia3}=\left[\mathbf{0}\;\mathbf{0}\;\mathbf{p}_1^*\times\mathbf{E}_{33}\;(\mathbf{p}_1^*+\mathbf{p}_2^*)\times\mathbf{E}_{44}\;....\;\sum_{i=1}^{n-3}\mathbf{p}_i^*\times\mathbf{E}_{(n-1)(n-1)}\;\sum_{i=1}^{n-2}\mathbf{p}_i^*\times\mathbf{e}_{nn}\right]. \quad (31)$$

The forces and moments exerted on link i by link $i-1$ in the coordinates of link $i-1$ are

$$\hat{\mathbf{f}}_i=\left[\hat{f}_{xi}\;\;\hat{f}_{yi}\;\;\hat{f}_{zi}\right]^{\mathrm{T}}=\mathbf{D}_{i-1}^{\mathrm{T}}\mathbf{f}_i \;\text{ and }\; \hat{\mathbf{n}}_i=\left[\hat{n}_{xi}\;\hat{n}_{yi}\;\hat{n}_{zi}\right]^{\mathrm{T}}=\mathbf{D}_{i-1}^{\mathrm{T}}\mathbf{n}_i. \quad (32)$$

The projection of $\mathbf{n}_i$ along the axis of motion of joint i is

$$u_i=\mathbf{z}_{i-1}^{\mathrm{T}}\mathbf{n}_i, \quad (33)$$

where $\mathbf{z}_{i-1}$ is a unit vector of the axis of motion, given in the first three elements of the third column of the matrix $\mathbf{T}_i$. Consequently, using Eqs. (22)–(33), the joint torques u_i are

$$u_i=\sum_{j=1}^{n}h_{ij}\ddot{q}_j+u'_i \;, \; i=1 \text{ to } n, \quad (34)$$

$$\sum_{j=1}^{n}h_{ij}\ddot{q}_j=\mathbf{z}_{i-1}^{\mathrm{T}}(\mathbf{n}'_{ia1}+\mathbf{n}'_{ia2}+\mathbf{n}'_{ia3})\,\ddot{\mathbf{q}}, \quad (35)$$

$$u'_i=\mathbf{z}_{i-1}^{\mathrm{T}}\mathbf{n}'_{ic}. \quad (36)$$

Herein, $\dot{q}_i(t_k)$ and $\ddot{q}_i(t_k)$ are used to calculate h_{ij} and u'_i.

The computations needed to solve the linear system of Eq. (34) in order to compute $\ddot{q}_i(t_k)$ can be performed using Gaussian elimination.

IV. Numerical Example of 4-DoFs Spatial Disorientation Trainer

The spatial disorientation trainer (SDT) is designed as a 4-DoFs manipulator with rotational axes, Fig 2. Herein, $\mathbf{p}_1^*=a_1\begin{bmatrix}c_1 & s_1 & 0\end{bmatrix}^{\mathrm{T}}$, $\mathbf{p}_2^*=\begin{bmatrix}0 & 0 & d_2\end{bmatrix}^{\mathrm{T}}$, $\mathbf{p}_3^*=\mathbf{p}_4^*=\mathbf{p}_5^*=\mathbf{0}$.

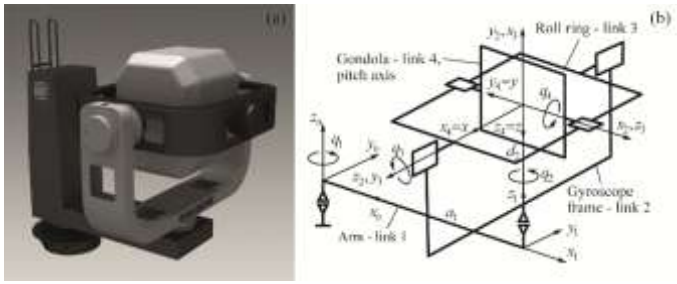

Fig. 2. (a) 3D model of the four DoFs SDT. (b) Coordinate frames of the SDT.

The vectors $\mathbf{n}'_{ic}$, $\mathbf{n}'_{ia1}$, $\mathbf{n}'_{ia2}$, and $\mathbf{n}'_{ia3}$, Eqs. (22)–(31), for the SDT are

$$\mathbf{n}_{4c}=\mathbf{d}_4+\mathbf{l}_4\times\mathbf{e}_4,\ \mathbf{n}_{3c}=\mathbf{d}_3+\mathbf{l}_3\times\mathbf{e}_3,\ \mathbf{n}_{2c}=\mathbf{d}_2+\mathbf{l}_2\times\mathbf{e}_2+\mathbf{p}_2^*\times\mathbf{E}_3,$$

$$\mathbf{n}_{1c}=\mathbf{d}_1+\mathbf{l}_1\times\mathbf{e}_1+\mathbf{p}_1^*\times\mathbf{E}_2,\ \text{where}\ \mathbf{E}_3=\mathbf{e}_3+\mathbf{e}_4,\ \mathbf{E}_2=\mathbf{E}_3+\mathbf{e}_2, \tag{37}$$

$$\begin{aligned}
\mathbf{n}_{4a1}&=\sum_{k=1}^{4}(\mathbf{d}_{4k}+\mathbf{l}_4\times\mathbf{e}_{4k})\ddot{q}_k=\sum_{k=1}^{4}\mathbf{n}_{4ka1}\ddot{q}_k,\\
\mathbf{n}_{3a1}&=\sum_{k=1}^{3}(\mathbf{d}_{3k}+\mathbf{l}_3\times\mathbf{e}_{3k})\ddot{q}_k=\sum_{k=1}^{3}\mathbf{n}_{3ka1}\ddot{q}_k,\\
\mathbf{n}_{2a1}&=\sum_{k=1}^{2}(\mathbf{d}_{2k}+\mathbf{l}_2\times\mathbf{e}_{2k})\ddot{q}_k=\sum_{k=1}^{2}\mathbf{n}_{2ka1}\ddot{q}_k,\\
\mathbf{n}_{1a1}&=(\mathbf{d}_{11}+\mathbf{l}_1\times\mathbf{e}_{11})\ddot{q}_1=\mathbf{n}_{11a1}\ddot{q}_1,
\end{aligned} \tag{38}$$

$$\mathbf{n}'_{4a1}=\begin{bmatrix}\mathbf{n}_{41a1} & \mathbf{n}_{42a1} & \mathbf{n}_{43a1} & \mathbf{n}_{44a1}\end{bmatrix}$$

$$=\begin{bmatrix}\mathbf{d}_{41}+\mathbf{l}_4\times\mathbf{e}_{41} & \mathbf{d}_{42}+\mathbf{l}_4\times\mathbf{e}_{42} & \mathbf{d}_{43}+\mathbf{l}_4\times\mathbf{e}_{43} & \mathbf{d}_{44}+\mathbf{l}_4\times\mathbf{e}_{44}\end{bmatrix},$$

$$\mathbf{n}'_{3a1}=\mathbf{n}_{3a1}+\mathbf{n}_{4a1}=\begin{bmatrix}\mathbf{n}'_{31a1} & \mathbf{n}'_{32a1} & \mathbf{n}'_{33a1} & \mathbf{n}'_{44a1}\end{bmatrix}$$

$$=\begin{bmatrix}\sum_{k=3}^{4}(\mathbf{d}_{k1}+\mathbf{l}_k\times\mathbf{e}_{k1}) & \sum_{k=3}^{4}(\mathbf{d}_{k2}+\mathbf{l}_k\times\mathbf{e}_{k2}) & \sum_{k=3}^{4}(\mathbf{d}_{k3}+\mathbf{l}_k\times\mathbf{e}_{k3}) & \mathbf{n}'_{44a1}\end{bmatrix},$$

$$\mathbf{n}'_{2a1}=\mathbf{n}_{2a1}+\mathbf{n}'_{3a1}=\begin{bmatrix}\mathbf{n}'_{21a1} & \mathbf{n}'_{22a1} & \mathbf{n}'_{33a1} & \mathbf{n}'_{44a1}\end{bmatrix}$$

$$=\begin{bmatrix}\sum_{k=2}^{4}(\mathbf{d}_{k1}+\mathbf{l}_k\times\mathbf{e}_{k1}) & \sum_{k=2}^{4}(\mathbf{d}_{k2}+\mathbf{l}_k\times\mathbf{e}_{k2}) & \mathbf{n}'_{33a1} & \mathbf{n}'_{44a1}\end{bmatrix},$$

$$\mathbf{n}'_{1a1}=\mathbf{n}_{1a1}+\mathbf{n}'_{2a1}=\begin{bmatrix}\mathbf{n}'_{11a1} & \mathbf{n}'_{22a1} & \mathbf{n}'_{33a1} & \mathbf{n}'_{44a1}\end{bmatrix}$$

$$=\begin{bmatrix}\sum_{k=1}^{4}(\mathbf{d}_{k1}+\mathbf{l}_k\times\mathbf{e}_{k1}) & \mathbf{n}'_{22a1} & \mathbf{n}'_{33a1} & \mathbf{n}'_{44a1}\end{bmatrix}, \tag{39}$$

$$\mathbf{n}_{4a2}=\mathbf{0},\ \mathbf{p}_3^*=\mathbf{0}\Rightarrow\mathbf{n}_{3a2}=\mathbf{0},$$

$$\mathbf{n}'_{4a2}=\mathbf{n}'_{3a2}=\begin{bmatrix}\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0}\end{bmatrix},$$

$$\mathbf{n}'_{2a2}=\begin{bmatrix}\mathbf{p}_2^*\times\mathbf{E}_{31} & \mathbf{p}_2^*\times\mathbf{E}_{32} & \mathbf{p}_2^*\times\mathbf{E}_{33} & \mathbf{0}\end{bmatrix},$$

$$\mathbf{n}'_{1a2}=\begin{bmatrix}\mathbf{p}_1^*\times\mathbf{E}_{21}+\mathbf{p}_2^*\times\mathbf{E}_{31} & \mathbf{p}_1^*\times\mathbf{E}_{22}+\mathbf{p}_2^*\times\mathbf{E}_{32} & \mathbf{p}_2^*\times\mathbf{E}_{33} & \mathbf{0}\end{bmatrix}, \tag{40}$$

where
$\mathbf{E}_{31}=\mathbf{e}_{41}+\mathbf{e}_{31}$, $\mathbf{E}_{32}=\mathbf{e}_{42}+\mathbf{e}_{32}$, $\mathbf{E}_{33}=\mathbf{e}_{43}+\mathbf{e}_{33}$, $\mathbf{E}_{21}=\mathbf{e}_{31}+\mathbf{e}_{21}$, $\mathbf{E}_{22}=\mathbf{e}_{32}+\mathbf{e}_{22}$,

$$\mathbf{n}'_{2a3}=\begin{bmatrix}\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{p}_2^*\times\mathbf{E}_{44}\end{bmatrix},\ \mathbf{n}'_{1a3}=\begin{bmatrix}\mathbf{0} & \mathbf{0} & \mathbf{p}_1^*\times\mathbf{E}_{33} & (\mathbf{p}_1^*+\mathbf{p}_2^*)\times\mathbf{E}_{44}\end{bmatrix}, \tag{41}$$

where $\mathbf{E}_{44}=\mathbf{e}_{44}$.

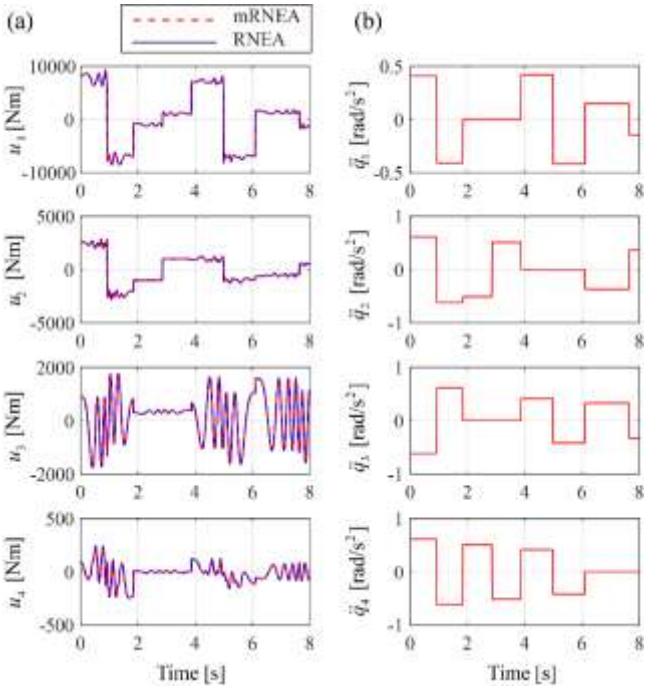

Fig. 3. (a) Link torques u_i calculated using RNEA and mRNEA. (b) Link accelerations of SDT.

In this numerical example, the consecutive link positions $q_i(t_k)$, velocities $\dot{q}_i(t_k)$, and accelerations $\ddot{q}_i(t_k)$ of the SDT are used as the input. Based on these data, and the inertial and geometric parameters of the SDT links, the torques u_i are calculated using RNEA and mRNEA. The same results were obtained using both algorithms, Fig. 3(a). Consequently, solution of the linear system of Eqs. (34) gave the values of $\ddot{q}_i(t_k)$ that coincides with the input accelerations, Fig. 3(b).

V. FD ALGORITHM RELATIVE TO MANIPULATOR ACTUATORS

Similar to Eq. (1), the motion equations for the manipulator relative to the torques/forces of the robot actuators can be written as

$$\mathbf{H}_{\mathrm{a}}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{u}'_{\mathrm{a}}(\mathbf{q},\dot{\mathbf{q}},\mathbf{k}_{\mathrm{ea}}) = \mathbf{u}_{\mathrm{a}}\,, \tag{42}$$

$$\mathbf{u}'_{\mathrm{a}}(\mathbf{q},\dot{\mathbf{q}},\mathbf{k}_{\mathrm{ea}}) = \mathbf{C}_{\mathrm{a}}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}_{\mathrm{a}}(\mathbf{q}) + \mathbf{J}(\mathbf{q})^{\mathrm{T}}\mathbf{k}_{\mathrm{ea}} + \mathbf{F}_v\dot{\mathbf{q}} + \mathbf{F}_s\,\mathrm{sign}(\dot{\mathbf{q}})\,. \tag{43}$$

Herein, $\mathbf{H}_{\mathrm{a}}(\mathbf{q})$, $\mathbf{u}'_{\mathrm{a}}(\mathbf{q},\dot{\mathbf{q}},\mathbf{k}_{\mathrm{ea}})$, and $\mathbf{u}_{\mathrm{a}}$ relate to the actuator rotors. $\mathbf{F}_{\mathrm{v}}$ denotes an $n \times n$ diagonal matrix of viscous friction coefficients f_{vi}. The static friction torques are considered as Coulomb friction torques; $\mathbf{F}_{\mathrm{s}}$ is an $n \times n$ diagonal matrix of the Coulomb friction constants. Herein, $\mathrm{sign}(\dot{\mathbf{q}})$ denotes an $n \times 1$ vector whose components are given by the sign functions of single joint velocities.

If certain absolute value of u_{ai} exceeds its limit, it is reduced to the maximum possible. These achievable values of u_{ai} are then used in the following linear system of n equations to calculate the achievable (realistic) joint accelerations $\ddot{q}_i$:

$$\sum_{j=1}^{n} h_{\mathrm{a}\,ij}\ddot{q}_j = u_{\mathrm{a}\,i} - u'_{\mathrm{a}\,i}\,,\ i = 1 \text{ to } n. \tag{44}$$

VI. CONCLUSIONS

An algorithm which calculates the achievable joint accelerations in each interpolation cycle based on the FD model and actuators' capabilities was given in this paper. Presented algorithm enables the setting of only attainable joint velocities within each interpolation cycle as determined from the joint acceleration by taking into account the achievable actuator torques. As a result, a precise simulation of the robot movements is provided. This algorithm can indicate to the operator that the programmed parameters of the movements are not achievable. Furthermore, calculation of the realistic forces and moments of the robot joints can be achieved when the simulation system is used in the design phase.

In presented simulation, the mRNEA that gives the mass matrix **H** and the bias vector $\mathbf{u}'$ of a dynamic model was used. Consequently, mRNEA allows for solving FD by calculating ID only once. It was shown that proposed FD algorithm does not need to calculate the input vector **u** of the FD algorithm, which additionally increases the computational efficiency of the presented method. Compared with the other methods given in the literature, the algorithm presented herein is one of the most efficient ones. Apart from that, it is very simple to develop and implement.

ACKNOWLEDGMENT

This research is supported by the Ministry of Education, Science and Technological Development of Serbia.

REFERENCES

[1] E. Otten, "Inverse and forward dynamics: models of multi–body systems," Philosophical Transactions of the Royal Society of London, Series B: Biological Sciences, 358(1437) (2003) 1493-1500.

[2] B. Siciliano, L. Sciavicco, L. Villani, G. Oriolo, "Dynamics," in: Robotics: Modelling, Planning and Control, Springer–Verlag London Limited, 2009, pp. 247–302.

[3] M. Hirschkorn, J. Kövecses, "The role of the mass matrix in the analysis of mechanical systems," Multibody Syst. Dyn., 30 (2013) 397–412.

[4] R. Featherstone, D. Orin, Dynamics, in: Handbook of robotics, B. Siciliano, O. Khatib (Eds.), Springer–Verlag London Limited, 2016, pp. 35–65.

[5] R. Featherstone, D. Orin, "Robot Dynamics, Equations and Algorithms," IEEE international conference on robotics and automation (ICRA), San Francisco, CA, 24–28 (2000) 826–834.

[6] M. Walker, D. Orin, "Efficient dynamic computer simulation of robotic mechanisms," Trans. ASME, J. Dynamic Systems, Measurement & Control: 104 (1982) 205–211.

[7] M. Walker, D. Orin, "Dynamics," in: Robot motion: Planning and Control, M. Brady, J. Hollerbach, T. Johnson, T. Lozano–Perez, M. Mason (Eds.), MIT Press, 1984, pp. 51–126.

[8] M. Walker, "Kinematics and Dynamics," in: Handbook of industrial robotics, S.Y. Nof (Eds.), John Wiley and Sons, Inc., 1985, pp. 80–95.

[9] L. Tsai, "Dynamics of Serial Manipulators," in: Robot analysis: the mechanics of serial and parallel manipulators, John Wiley and Sons, Inc, 1999, pp. 372–423.

[10] R. Featherstone, Rigid Body Dynamics Algorithms, Springer, 2008.

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

[12] V. Kvrgic, J. Vidakovic, "Efficient method for robot forward dynamics computation," Mechanism and Machine Theory 145 (2020) (10368) 1-25, DOI: 10.1016/j.mechmachtheory.2019.103680.