2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Congress Center Hamburg
Sept 28 - Oct 2, 2015. Hamburg, Germany

A recursive Newton-Euler algorithm for robots with elastic joints and its application to control

Gabriele Buondonno Alessandro De Luca

Abstract— **We consider the problem of computing the inverse dynamics of a serial robot manipulator with N elastic joints in a recursive numerical way. The solution algorithm is a generalized version of the standard Newton-Euler approach, running still with linear complexity $O(N)$ but requiring to set up recursions that involve higher order derivatives of motion and force variables. Mimicking the case of rigid robots, we use this algorithm and a numerical factorization of the link inertia matrix (which needs to be inverted in the elastic joint case) for implementing on-line a feedback linearization control law for trajectory tracking purposes. The complete method has a complexity that grows as $O(N^3)$. The developed tools are generic, easy to use, and do not require symbolic Lagrangian modeling and customization, thus being of particular interest when the number N of elastic joints becomes large.**

I. Introduction

The presence of elasticity at the joint level has been traditionally considered an undesirable feature in industrial manipulators, mostly a side effect of using transmission belts or long shafts to relocate actuation, or resorting to reduction elements with a high reduction ratio, such as the Harmonic Drive™ [1], [2]. However, the recent deployment of lightweight manipulators [3]–[5], intended for physical human-robot interaction, has turned joint elasticity into a beneficial advantage as mechanical absorbing layer for safety. In fact, in the event of a collision with a human being, an elastic joint would comply with the human's body before special-purpose collision detection and reaction software would take action (see, e.g., [6]). Moreover, the design of intentionally compliant joints, such as when using serial elastic actuation (SEA) [7], is a way to achieve more natural, human-like, and energy efficient robot motion. For these reasons, there is a renewed interest in robots displaying non-negligible compliance at the joints.

Dynamic modeling of robots with elastic joints requires the consideration of a double set of variables in order to describe the distinct, dynamically related positions of the driving motors and of the driven links [8], [9]. Symbolic methods based on Lagrangian formulation have been developed for the systematic derivation of the dynamic model and for computing feedforward torque commands based on inverse dynamics [10], [11]. In turn, control algorithms aimed at high performance for the stable tracking of desired trajectories require extra measurements and a more complex design [12], [13]. In particular, under a very reasonable simplifying assumption on the angular kinetic energy of the driving motors, it has been shown in [14] that robots with elastic joints can be transformed into exactly linear and decoupled systems by means a nonlinear state feedback. This feedback linearization property is the equivalent of the so-called computed torque method for fully rigid robots. A fundamental difference, however, is that the resulting linear system turns out to be of fourth order (i.e., chains of four input-output integrators at each joint, rather than only two), with more critical stabilization issues. See Sec. II for a summary of these results.

This paper addresses two very basic problems in dynamics and control for the considered class of robots. The first is the extension to robots with joint elasticity of the classical Recursive Newton-Euler Algorithm (RNEA), originally proposed in [15] for rigid robots. This allows solving the inverse dynamics problem in a numerical way, which is efficient and exact (i.e., without resorting to approximate differentiation of quantities). Inspired by the works [16], [17] on computing the command torque derivative in rigid robots, we present in Sec. III our complete Elastic Joints Newton-Euler Algorithm, or EJNEA in short, which runs with an asymptotic complexity $O(N)$, being N the number of elastic joints. As a result, the feedforward command associated to a desired robot trajectory, defined in terms of link motion, can be obtained without the need to derive an explicit Lagrangian dynamic model in symbolic form, which may become a particularly cumbersome task for large N (say, for $N \geq 7$).

The second problem is the implementation of a feedback linearization control law for robots with elastic joints that should avoid, just as in the rigid case, the need of evaluating at run time the complex expressions of the robot dynamic model, which was previously obtained in symbolic form. In Sec. IV, the EJNEA will serve again for the purpose of accurate and exponentially stable tracking of sufficiently smooth trajectories in robots with elastic joints, once it is complemented with a suitable numerical factorization of the link inertia matrix. The resulting method runs in $O(N^3)$.

A. Basics

Consider first a robot consisting of an open kinematic chain of N rigid links connected through N *rigid* joints. Under standard assumptions [18], the Lagrangian dynamic model takes the form

$$(\boldsymbol{M}(\boldsymbol{q}) + \boldsymbol{B})\, \ddot{\boldsymbol{q}} + \boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{\tau}, \tag{1}$$

where $\boldsymbol{q} \in \mathbb{R}^N$ is the vector of link positions, $\boldsymbol{\tau} \in \mathbb{R}^N$ is the vector of motor torques, $\boldsymbol{M}(\boldsymbol{q})$ is the positive definite inertia

The authors are with the Dipartimento di Ingegneria Informatica, Automatica e Gestionale, Sapienza Università di Roma, Via Ariosto 25, 00185 Rome, Italy ({buondonno, deluca}@diag.uniroma1.it). This work is supported by the European Commission, within the FP7 ICT-287513 SAPHARI project (www.saphari.eu).

matrix of the robot links, $\boldsymbol{n}(\boldsymbol{q},\dot{\boldsymbol{q}}) = \boldsymbol{c}(\boldsymbol{q},\dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q})$ contains centrifugal, Coriolis and gravity terms, and $\boldsymbol{B} > 0$ is the constant, diagonal matrix with the drive inertia moments, i.e., the inertias of the motor rotors' around their own spinning axes, reflected through the reduction ratios.

Let the desired link motion be specified by a trajectory $\boldsymbol{q}_d(t) \in \bar{C}^2$, i.e., the class of twice differentiable time functions (with acceleration that may be discontinuous). Then, the torques needed to execute the desired motion (inverse dynamics problem) can be computed either by evaluating the symbolic model expressions or by resorting to the recursive Newton-Euler numerical algorithm

$$\boldsymbol{\tau}_{r,d} = \text{RNEA}(\boldsymbol{q}_d,\dot{\boldsymbol{q}}_d,\ddot{\boldsymbol{q}}_d) = (\boldsymbol{M}(\boldsymbol{q}_d)+\boldsymbol{B})\,\ddot{\boldsymbol{q}}_d + \boldsymbol{n}(\boldsymbol{q}_d,\dot{\boldsymbol{q}}_d). \tag{2}$$

On the other hand, if we wish to obtain exact linearization (and input-output decoupling) of the closed-loop system, we need to apply to (1) the nonlinear feedback law

$$\boldsymbol{\tau}_r = \text{RNEA}(\boldsymbol{q},\dot{\boldsymbol{q}},\boldsymbol{a}) = (\boldsymbol{M}(\boldsymbol{q})+\boldsymbol{B})\,\boldsymbol{a} + \boldsymbol{n}(\boldsymbol{q},\dot{\boldsymbol{q}}), \tag{3}$$

yielding $\ddot{\boldsymbol{q}} = \boldsymbol{a}$. The control design is completed by choosing $\boldsymbol{a} = \ddot{\boldsymbol{q}}_d + \boldsymbol{K}_D(\dot{\boldsymbol{q}}_\text{d} - \dot{\boldsymbol{q}}) + \boldsymbol{K}_P(\boldsymbol{q}_\text{d} - \boldsymbol{q})$, with diagonal, positive definite PD gain matrices. Thus, in the rigid case, the feedback linearizing control can be computed in an efficient numerical way using directly the RNEA, without any relevant modification. Note that no inversion of the robot (link plus motor) inertia matrix is necessary in the rigid case.

II. Robots with Elastic Joints

In the presence of joint elasticity, the link positions are distinct from the motor positions. Therefore, a robot with N elastic joints will require $2N$ generalized coordinates [8]

$$\boldsymbol{\Theta} = \begin{pmatrix} \boldsymbol{q} \\ \boldsymbol{\theta}_m \end{pmatrix} \in \mathbb{R}^{2N},$$

where $\boldsymbol{\theta}_m \in \mathbb{R}^N$ is the vector of motor positions, *after* the reduction gears. Under the usual, so-called Spong modeling assumptions [14], the dynamic model for robots with elastic joints takes the form

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{n}(\boldsymbol{q},\dot{\boldsymbol{q}}) + \boldsymbol{K}(\boldsymbol{q} - \boldsymbol{\theta}_m) = \boldsymbol{0} \tag{4a}$$

$$\boldsymbol{B}\ddot{\boldsymbol{\theta}}_m + \boldsymbol{K}(\boldsymbol{\theta}_m - \boldsymbol{q}) = \boldsymbol{\tau}, \tag{4b}$$

where $\boldsymbol{K} > 0$ is the constant, diagonal joint stiffness matrix. Equations (4a) and (4b) are referred to as *link equation* and *motor equation*, respectively. The state of the system is $4N$-dimensional and is given, e.g., by $(\boldsymbol{q},\boldsymbol{\theta}_m,\dot{\boldsymbol{q}},\dot{\boldsymbol{\theta}}_m)$. Note also that $\boldsymbol{M}(\boldsymbol{q})$ contains the link masses, the link inertias, the motor masses, and all the inertial components of the motors, with the exception of the drive inertia moments, which are already included in $\boldsymbol{B}$.

A. Inverse dynamics

Given a desired link trajectory $\boldsymbol{q}_d(t) \in \bar{C}^4$, i.e., four times differentiable (at most with discontinuous fourth derivative), it is possible to use (4) for computing the nominal torque that realizes the desired motion. We can differentiate (4a) once[1]

$$\boldsymbol{M}(\boldsymbol{q})\dddot{\boldsymbol{q}} + \dot{\boldsymbol{M}}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \dot{\boldsymbol{n}}(\boldsymbol{q},\dot{\boldsymbol{q}}) = \boldsymbol{K}(\dot{\boldsymbol{\theta}}_m - \dot{\boldsymbol{q}}), \tag{5}$$

and twice

$$\boldsymbol{M}(\boldsymbol{q})\ddddot{\boldsymbol{q}} + 2\dot{\boldsymbol{M}}(\boldsymbol{q})\dddot{\boldsymbol{q}} + \ddot{\boldsymbol{M}}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \ddot{\boldsymbol{n}}(\boldsymbol{q},\dot{\boldsymbol{q}}) = \boldsymbol{K}(\ddot{\boldsymbol{\theta}}_m - \ddot{\boldsymbol{q}}). \tag{6}$$

Defining the joint elastic torque as

$$\boldsymbol{\tau}_e = \boldsymbol{K}(\boldsymbol{\theta}_m - \boldsymbol{q}), \tag{7}$$

we have

$$\dot{\boldsymbol{\tau}}_e = \boldsymbol{K}(\dot{\boldsymbol{\theta}}_m - \dot{\boldsymbol{q}}), \qquad \ddot{\boldsymbol{\tau}}_e = \boldsymbol{K}(\ddot{\boldsymbol{\theta}}_m - \ddot{\boldsymbol{q}}). \tag{8}$$

Thus, the motor accelerations can be expressed as

$$\ddot{\boldsymbol{\theta}}_m = \ddot{\boldsymbol{q}} + \boldsymbol{K}^{-1}\ddot{\boldsymbol{\tau}}_e, \tag{9}$$

where $\ddot{\boldsymbol{\tau}}_e$ is computed as in the left-hand side of (6). Also, from (4b)

$$\boldsymbol{\tau} = \boldsymbol{B}\ddot{\boldsymbol{\theta}}_m + \boldsymbol{\tau}_e. \tag{10}$$

Combining equations and plugging everywhere $\boldsymbol{q} = \boldsymbol{q}_d(t)$, we get the final expression of the desired torque in terms of the desired link trajectory only:

$$\begin{aligned}\boldsymbol{\tau}_d = \boldsymbol{B}\boldsymbol{K}^{-1}[\,&\boldsymbol{M}(\boldsymbol{q}_d)\ddddot{\boldsymbol{q}}_d + 2\dot{\boldsymbol{M}}(\boldsymbol{q}_d)\dddot{\boldsymbol{q}}_d + \ddot{\boldsymbol{M}}(\boldsymbol{q}_d)\ddot{\boldsymbol{q}}_d \\ &+ \ddot{\boldsymbol{n}}(\boldsymbol{q}_d,\dot{\boldsymbol{q}}_d)\,] + [\,\boldsymbol{M}(\boldsymbol{q}_d) + \boldsymbol{B}\,]\ddot{\boldsymbol{q}}_d + \boldsymbol{n}(\boldsymbol{q}_d,\dot{\boldsymbol{q}}_d).\end{aligned} \tag{11}$$

Note that, when $\boldsymbol{K}$ goes to infinity, we recover the inverse dynamics torque of the rigid robot in (2).

B. Feedback linearization control

From eqs. (4) to (10), it is immediate to see that the nonlinear state feedback law

$$\begin{aligned}\boldsymbol{\tau} =& \boldsymbol{B}\boldsymbol{K}^{-1}[\,\boldsymbol{M}(\boldsymbol{q})\boldsymbol{v} + 2\dot{\boldsymbol{M}}(\boldsymbol{q})\dddot{\boldsymbol{q}} + \ddot{\boldsymbol{M}}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \ddot{\boldsymbol{n}}(\boldsymbol{q},\dot{\boldsymbol{q}})\,] \\ &+ [\,\boldsymbol{M}(\boldsymbol{q}) + \boldsymbol{B}\,]\ddot{\boldsymbol{q}} + \boldsymbol{n}(\boldsymbol{q},\dot{\boldsymbol{q}})\end{aligned} \tag{12}$$

yields $\ddddot{\boldsymbol{q}} = \boldsymbol{v}$. In order to (exponentially) stabilize the trajectory error to zero, we choose

$$\begin{aligned}\boldsymbol{v} = \ddddot{\boldsymbol{q}}_d &+ \boldsymbol{K}_3(\dddot{\boldsymbol{q}}_d - \dddot{\boldsymbol{q}}) + \boldsymbol{K}_2(\ddot{\boldsymbol{q}}_d - \ddot{\boldsymbol{q}}) \\ &+ \boldsymbol{K}_1(\dot{\boldsymbol{q}}_d - \dot{\boldsymbol{q}}) + \boldsymbol{K}_0(\boldsymbol{q}_d - \boldsymbol{q}),\end{aligned} \tag{13}$$

where $\boldsymbol{K}_0,\ldots,\boldsymbol{K}_3$ are diagonal matrices, with their diagonal elements $K_{\cdot,i}$ being such that the polynomials

$$p_i(s) = s^4 + K_{3,i}s^3 + K_{2,i}s^2 + K_{1,i}s + K_{0,i}, \quad i = 1,\ldots,n,$$

are Hurwitz.

The careful reader may have recognized a difficulty in applying the feedback linearization control law in case of elastic joints. The control law makes apparent use of the actual values of $\ddot{\boldsymbol{q}}$ and $\dddot{\boldsymbol{q}}$, which are hard or even impossible to measure directly. On the other hand, multiple on-line numerical differentiation of position measurements would

[1]With $\dot{\boldsymbol{M}}(\boldsymbol{q})$, we mean $\frac{\mathrm{d}}{\mathrm{d}t}\,[\boldsymbol{M}(\boldsymbol{q})]$. Obviously, this quantity depends on $\dot{\boldsymbol{q}}$ as well as on $\boldsymbol{q}$, but we avoid notation overloading. Similar comments apply to $\dot{\boldsymbol{n}}$ and to higher-order time derivatives.

introduce excessive noise and/or delays in a discrete-time implementation.

However, elasticity-aware designed robots have usually a richer sensory suite built in. In our case, the essential capability is that of obtaining $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$, $\boldsymbol{\tau}_e$, and $\dot{\boldsymbol{\tau}}_e$ (actually, this is an alternative state representation) from sensor data. If a direct measure of either $\boldsymbol{q}$ or $\boldsymbol{\tau}_e$ is available, together with the usual measurement of the motor position $\boldsymbol{\theta}_m$ by encoders, then the missing quantity can be computed easily from (7). Further, if all three measures $\boldsymbol{q}$, $\boldsymbol{\theta}_m$, and $\boldsymbol{\tau}_e$ were available, then computing $\boldsymbol{q}$ from (7) could be preferable to the use of its direct measure, depending on the relative accuracy of the various sensors. Finally, to obtain $\dot{\boldsymbol{\tau}}_e$ and $\dot{\boldsymbol{q}}$, one resorts to numerical differentiation methods. The full state estimation would require then only one level of differentiation, just as in the rigid case (for obtaining $\dot{\boldsymbol{q}}$ from measurements of $\boldsymbol{q}$).

Once $\boldsymbol{\tau}_e$ and $\dot{\boldsymbol{\tau}}_e$ are also known, $\ddot{\boldsymbol{q}}$ and $\dddot{\boldsymbol{q}}$ can be computed from eqs. (4a) and (5) as

$$\ddot{\boldsymbol{q}} = \boldsymbol{M}^{-1}(\boldsymbol{q})\big[\boldsymbol{\tau}_e - \boldsymbol{n}(\boldsymbol{q},\dot{\boldsymbol{q}})\big] \tag{14}$$

$$\dddot{\boldsymbol{q}} = \boldsymbol{M}^{-1}(\boldsymbol{q})\left[\dot{\boldsymbol{\tau}}_e - \left(\dot{\boldsymbol{M}}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \dot{\boldsymbol{n}}(\boldsymbol{q},\dot{\boldsymbol{q}})\right)\right], \tag{15}$$

where $\ddot{\boldsymbol{q}}$ is evaluated in (15) using the value already computed by (14) and just saved.

III. The Algorithm: EJNEA

An algorithm capable of computing numerically $\boldsymbol{\tau}_d$ from a desired $\boldsymbol{q}_d$ and its first four derivatives, and which implements (11) without the need of deriving first the symbolic model, can be designed as a direct extension of the standard RNEA. In the following, we will drop for compactness the subscript d (of desired trajectory). As shown in Sec. II-A, it is necessary to compute first $\boldsymbol{\tau}_e$, then $\ddot{\boldsymbol{\tau}}_e$, and finally exploit (9) and (10) to obtain the nominal command torque $\boldsymbol{\tau}$.

Indeed, $\boldsymbol{\tau}_e$ can be computed using the standard RNEA. The only difference in this case is that the input parameters do not include the data about the drive inertia moments. The main difficulty lies in computing $\ddot{\boldsymbol{\tau}}_e$ and for this, higher-order dynamic terms need to be considered. In particular, the recursive algorithm will necessarily involve two more levels of differentiation than the standard RNEA, with the need of setting up recursions for higher-order time derivatives of motion variables. As usual, all quantities will be conveniently expressed in the *moving* reference frame of the considered link. This is very advantageous, since all dynamic parameters of the robot will be constant in these frames.

Before proceeding, we recall the usual notation in presence of multiple reference frames. Subscripts represent the link that a quantity is referring to, while superscripts (prefixed) in vectors and matrices indicate the reference frame in which a quantity is expressed. No superscript denotes by default the basis frame 0. Thus, for instance, ${}^{i}\gamma_i = {}^{i}\boldsymbol{R}_0\,{}^{0}\gamma_i = \boldsymbol{R}_i^T\,\dot{\boldsymbol{\omega}}_i$. Moreover, we are using here the *standard* Denavit-Hartenberg convention for frame assignment. The orientation of a frame i with respect to frame $i-1$, and the position of the origin of frame i with respect to frame $i-1$, expressed in frame i, are described respectively by the following rotation matrix and vector:

$${}^{i-1}\boldsymbol{R}_i = \begin{pmatrix} \mathrm{c}\theta_i & -\mathrm{c}\alpha_i\,\mathrm{s}\theta_i & \mathrm{s}\alpha_i\,\mathrm{s}\theta_i \\ \mathrm{s}\theta_i & \mathrm{c}\alpha_i\,\mathrm{c}\theta_i & -\mathrm{s}\alpha_i\,\mathrm{c}\theta_i \\ 0 & \mathrm{s}\alpha_i & \mathrm{c}\alpha_i \end{pmatrix},\ {}^{i}\boldsymbol{p}_{i,i-1} = \begin{pmatrix} a_i \\ d_i\,\mathrm{s}\alpha_i \\ d_i\,\mathrm{c}\alpha_i \end{pmatrix}.$$

With $\hat{\boldsymbol{z}}_i$, we denote the z-axis unit vector of frame i. It can be easily seen that $\hat{\boldsymbol{z}}_0 = (0\ 0\ 1)^T$, ${}^{i}\hat{\boldsymbol{z}}_i = \hat{\boldsymbol{z}}_0$, $\hat{\boldsymbol{z}}_i = {}^{0}\boldsymbol{R}_i\hat{\boldsymbol{z}}_0$, and ${}^{i}\hat{\boldsymbol{z}}_{i-1} = {}^{i}\boldsymbol{R}_{i-1}{}^{i-1}\hat{\boldsymbol{z}}_{i-1} = {}^{i}\boldsymbol{R}_{i-1}\hat{\boldsymbol{z}}_0$. Finally, we recall the well-known result

$$\dot{\boldsymbol{R}}_i = \boldsymbol{S}(\boldsymbol{\omega}_i)\boldsymbol{R}_i\,, \tag{16}$$

where $\boldsymbol{S}(\boldsymbol{\omega}_i)$ is the skew-symmetric matrix computed from $\boldsymbol{\omega}_i$ that performs the vector product operation, i.e., such that $\boldsymbol{S}(\boldsymbol{\omega}_i)\boldsymbol{v} = \boldsymbol{\omega}_i \times \boldsymbol{v}$, for every vector $\boldsymbol{v}$.

The final forward and backward recursions of EJNEA are outlined next. The algorithm is obtained by suitable differentiation of the Newton-Euler recursive equations until the needed fourth-order quantities appear.

A. Forward recursion

The following equations need to be propagated from the robot base to the tip, for $i = 1,\dots,N$:

$${}^{i}\boldsymbol{\omega}_i = {}^{i}\boldsymbol{R}_{i-1}(\,{}^{i-1}\boldsymbol{\omega}_{i-1} + \dot{\theta}_i\hat{\boldsymbol{z}}_0\,) \tag{17}$$

$${}^{i}\gamma_i = {}^{i}\boldsymbol{R}_{i-1}(\,{}^{i-1}\gamma_{i-1} + \ddot{\theta}_i\hat{\boldsymbol{z}}_0 + {}^{i-1}\boldsymbol{\omega}_{i-1}\times\dot{\theta}_i\hat{\boldsymbol{z}}_0\,) \tag{18}$$

$$\begin{aligned}{}^{i}\boldsymbol{\iota}_i = {}^{i}\boldsymbol{R}_{i-1}[\,&{}^{i-1}\boldsymbol{\iota}_{i-1} + \dddot{\theta}_i\hat{\boldsymbol{z}}_0 + {}^{i-1}\gamma_{i-1}\times\dot{\theta}_i\hat{\boldsymbol{z}}_0 \\ &+ {}^{i-1}\boldsymbol{\omega}_{i-1}\times(2\ddot{\theta}_i\hat{\boldsymbol{z}}_0 + {}^{i-1}\boldsymbol{\omega}_{i-1}\times\dot{\theta}_i\hat{\boldsymbol{z}}_0)\,]\end{aligned} \tag{19}$$

$$\begin{aligned}{}^{i}\boldsymbol{\varsigma}_i = {}^{i}\boldsymbol{R}_{i-1}\{\,&{}^{i-1}\boldsymbol{\varsigma}_{i-1} + \ddddot{\theta}_i\hat{\boldsymbol{z}}_0 + 3\,{}^{i-1}\gamma_{i-1}\times\ddot{\theta}_i\hat{\boldsymbol{z}}_0 \\ &+ 3\,{}^{i-1}\boldsymbol{\omega}_{i-1}\times(\dddot{\theta}_i\hat{\boldsymbol{z}}_0 + {}^{i-1}\boldsymbol{\omega}_{i-1}\times\ddot{\theta}_i\hat{\boldsymbol{z}}_0) \\ &+ 2\,{}^{i-1}\gamma_{i-1}\times({}^{i-1}\boldsymbol{\omega}_{i-1}\times\dot{\theta}_i\hat{\boldsymbol{z}}_0) \\ &+ {}^{i-1}\boldsymbol{\omega}_{i-1}\times[{}^{i-1}\boldsymbol{\omega}_{i-1}\times({}^{i-1}\boldsymbol{\omega}_{i-1}\times\dot{\theta}_i\hat{\boldsymbol{z}}_0) \\ &+ {}^{i-1}\gamma_{i-1}\times\dot{\theta}_i\hat{\boldsymbol{z}}_0] + {}^{i-1}\boldsymbol{\iota}_{i-1}\times\dot{\theta}_i\hat{\boldsymbol{z}}_0\}\end{aligned} \tag{20}$$

$$\begin{aligned}{}^{i}\boldsymbol{a}_i = {}&{}^{i}\boldsymbol{R}_{i-1}{}^{i-1}\boldsymbol{a}_{i-1} + {}^{i}\boldsymbol{\omega}_i\times({}^{i}\boldsymbol{\omega}_i\times{}^{i}\boldsymbol{p}_{i,i\text{-}1}) \\ &+ {}^{i}\gamma_i\times{}^{i}\boldsymbol{p}_{i,i\text{-}1} + \ddot{d}_i\,{}^{i}\hat{\boldsymbol{z}}_{i-1} + 2\dot{d}_i({}^{i}\boldsymbol{\omega}_i\times{}^{i}\hat{\boldsymbol{z}}_{i-1})\end{aligned} \tag{21}$$

$${}^{i}\boldsymbol{a}_{c_i} = {}^{i}\boldsymbol{a}_i + {}^{i}\boldsymbol{\omega}_i\times({}^{i}\boldsymbol{\omega}_i\times{}^{i}\boldsymbol{p}_{c_i,i}) + {}^{i}\gamma_i\times{}^{i}\boldsymbol{p}_{c_i,i} \tag{22}$$

$$\begin{aligned}{}^{i}\boldsymbol{j}_i = {}&{}^{i}\boldsymbol{R}_{i-1}{}^{i-1}\boldsymbol{j}_{i-1} + 2\,{}^{i}\gamma_i\times(\,{}^{i}\boldsymbol{\omega}_i\times{}^{i}\boldsymbol{p}_{i,i\text{-}1}) \\ &+ {}^{i}\boldsymbol{\omega}_i\times[\,{}^{i}\gamma_i\times{}^{i}\boldsymbol{p}_{i,i\text{-}1} + {}^{i}\boldsymbol{\omega}_i\times(\,{}^{i}\boldsymbol{\omega}_i\times{}^{i}\boldsymbol{p}_{i,i\text{-}1})] \\ &+ {}^{i}\boldsymbol{\iota}_i\times{}^{i}\boldsymbol{p}_{i,i\text{-}1} + \dddot{d}_i\,{}^{i}\hat{\boldsymbol{z}}_{i-1} + 3\ddot{d}_i\,{}^{i}\boldsymbol{\omega}_i\times{}^{i}\hat{\boldsymbol{z}}_{i-1} \\ &+ 3\dot{d}_i[{}^{i}\gamma_i\times{}^{i}\hat{\boldsymbol{z}}_{i-1} + {}^{i}\boldsymbol{\omega}_i\times(\,{}^{i}\boldsymbol{\omega}_i\times{}^{i}\hat{\boldsymbol{z}}_{i-1})]\end{aligned} \tag{23}$$

$$\begin{aligned}{}^{i}\boldsymbol{j}_{c_i} = {}&{}^{i}\boldsymbol{j}_i + {}^{i}\boldsymbol{\iota}_i\times{}^{i}\boldsymbol{p}_{c_i,i} + 2\,{}^{i}\gamma_i\times(\,{}^{i}\boldsymbol{\omega}_i\times{}^{i}\boldsymbol{p}_{c_i,i}) \\ &+ {}^{i}\boldsymbol{\omega}_i\times[\,{}^{i}\gamma_i\times{}^{i}\boldsymbol{p}_{c_i,i} + {}^{i}\boldsymbol{\omega}_i\times(\,{}^{i}\boldsymbol{\omega}_i\times{}^{i}\boldsymbol{p}_{c_i,i})]\end{aligned} \tag{24}$$

$$\begin{aligned}{}^{i}\boldsymbol{s}_i = {}&{}^{i}\boldsymbol{R}_{i-1}{}^{i-1}\boldsymbol{s}_{i-1} + {}^{i}\boldsymbol{\varsigma}_i\times{}^{i}\boldsymbol{p}_{i,i\text{-}1} + 3\,{}^{i}\boldsymbol{\iota}_i\times({}^{i}\boldsymbol{\omega}_i\times{}^{i}\boldsymbol{p}_{i,i\text{-}1}) \\ &+ 3\,{}^{i}\gamma_i\times[{}^{i}\gamma_i\times{}^{i}\boldsymbol{p}_{i,i\text{-}1} + {}^{i}\boldsymbol{\omega}_i\times({}^{i}\boldsymbol{\omega}_i\times{}^{i}\boldsymbol{p}_{i,i\text{-}1})] \\ &+ {}^{i}\boldsymbol{\omega}_i\times[{}^{i}\boldsymbol{\iota}_i\times{}^{i}\boldsymbol{p}_{i,i\text{-}1} + 2\,{}^{i}\gamma_i\times({}^{i}\boldsymbol{\omega}_i\times{}^{i}\boldsymbol{p}_{i,i\text{-}1})] \\ &+ {}^{i}\boldsymbol{\omega}_i\times\{{}^{i}\boldsymbol{\omega}_i\times[{}^{i}\gamma_i\times{}^{i}\boldsymbol{p}_{i,i\text{-}1} + {}^{i}\boldsymbol{\omega}_i\times({}^{i}\boldsymbol{\omega}_i\times{}^{i}\boldsymbol{p}_{i,i\text{-}1})]\} \\ &+ \ddddot{d}_i\,{}^{i}\hat{\boldsymbol{z}}_{i-1} + 4\dot{d}_i\,{}^{i}\boldsymbol{\iota}_i\times{}^{i}\hat{\boldsymbol{z}}_{i-1} + 8\dot{d}_i\,{}^{i}\gamma_i\times({}^{i}\boldsymbol{\omega}_i\times{}^{i}\hat{\boldsymbol{z}}_{i-1}) \\ &+ 4\dot{d}_i\,{}^{i}\boldsymbol{\omega}_i\times[{}^{i}\gamma_i\times{}^{i}\hat{\boldsymbol{z}}_{i-1} + {}^{i}\boldsymbol{\omega}_i\times({}^{i}\boldsymbol{\omega}_i\times{}^{i}\hat{\boldsymbol{z}}_{i-1})] \\ &+ 6\ddot{d}_i[{}^{i}\gamma_i\times{}^{i}\hat{\boldsymbol{z}}_{i-1} + {}^{i}\boldsymbol{\omega}_i\times({}^{i}\boldsymbol{\omega}_i\times{}^{i}\hat{\boldsymbol{z}}_{i-1})] \\ &+ 4\dddot{d}_i\,{}^{i}\boldsymbol{\omega}_i\times{}^{i}\hat{\boldsymbol{z}}_{i-1}\end{aligned} \tag{25}$$

$$
\begin{aligned}
{}^i\boldsymbol{s}_{c_i} = {}&{}^i\boldsymbol{s}_i + {}^i\boldsymbol{\varsigma}_i \times {}^i\boldsymbol{p}_{c_i,i} + 3\,{}^i\boldsymbol{\iota}_i \times ({}^i\boldsymbol{\omega}_i \times {}^i\boldsymbol{p}_{c_i,i}) \\
&+ 3\,{}^i\boldsymbol{\gamma}_i \times [{}^i\boldsymbol{\gamma}_i \times {}^i\boldsymbol{p}_{c_i,i} + {}^i\boldsymbol{\omega}_i \times ({}^i\boldsymbol{\omega}_i \times {}^i\boldsymbol{p}_{c_i,i})] \\
&+ {}^i\boldsymbol{\omega}_i \times [{}^i\boldsymbol{\iota}_i \times {}^i\boldsymbol{p}_{c_i,i} + 2\,{}^i\boldsymbol{\gamma}_i \times ({}^i\boldsymbol{\omega}_i \times {}^i\boldsymbol{p}_{c_i,i})] \\
&+ {}^i\boldsymbol{\omega}_i \times \{{}^i\boldsymbol{\omega}_i \times [{}^i\boldsymbol{\gamma}_i \times {}^i\boldsymbol{p}_{c_i,i} + {}^i\boldsymbol{\omega}_i \times ({}^i\boldsymbol{\omega}_i \times {}^i\boldsymbol{p}_{c_i,i})]\}.
\end{aligned} \tag{26}
$$

The following symbols have been used:

- $\boldsymbol{\omega}_i$ angular velocity of frame i;
- $\boldsymbol{\gamma}_i$ angular acceleration of frame i;
- $\boldsymbol{\iota}_i$ angular jerk of frame i;
- $\boldsymbol{\varsigma}_i$ angular snap of frame i;
- $\boldsymbol{a}_i$ acceleration of the origin of frame i;
- $\boldsymbol{a}_{c_i}$ acceleration of the center of mass (CoM) of link i;
- $\boldsymbol{j}_i$ jerk of the origin of frame i;
- $\boldsymbol{j}_{c_i}$ jerk of the CoM of link i;
- $\boldsymbol{s}_i$ snap[2] of the origin of frame i;
- $\boldsymbol{s}_{c_i}$ snap of the CoM of link i;
- $\boldsymbol{p}_{c_i,i}$ position of the CoM of the *augmented* link i (i.e., of the link plus the motor mounted on it) w.r.t. origin of frame i.

As in standard Newton-Euler, the initialization of the forward recursion is zero for all quantities, with the exception of ${}^0\boldsymbol{a}_0$, which is set to $-\boldsymbol{g}$ to account for gravitational effects, being $\boldsymbol{g}$ the constant gravity acceleration vector (expressed in frame 0). The above equations are valid both for revolute and prismatic joints, with $\theta_i = q_i$ in the former case, and $d_i = q_i$ in the latter. Moreover, if joint i is prismatic, then $\dot{\theta}_i = \ddot{\theta}_i = \dddot{\theta}_i = \ddddot{\theta}_i = 0$; if it is revolute, $\dot{d}_i = \ddot{d}_i = \dddot{d}_i = \ddddot{d}_i = 0$, leading to considerable simplifications.

B. Backward recursion

The following equations need to be propagated from the robot tip to the base, for $i = N, \ldots, 1$:

$$
{}^i\boldsymbol{F}_i = m_i\,{}^i\boldsymbol{a}_{c_i} \tag{27}
$$

$$
{}^i\dot{\boldsymbol{F}}_i = m_i\,{}^i\boldsymbol{j}_{c_i} \tag{28}
$$

$$
{}^i\ddot{\boldsymbol{F}}_i = m_i\,{}^i\boldsymbol{s}_{c_i} \tag{29}
$$

$$
{}^i\boldsymbol{N}_i = {}^i\boldsymbol{I}_i\,{}^i\boldsymbol{\gamma}_i + {}^i\boldsymbol{\omega}_i \times ({}^i\boldsymbol{I}_i\,{}^i\boldsymbol{\omega}_i) \tag{30}
$$

$$
\begin{aligned}
{}^i\dot{\boldsymbol{N}}_i = {}&{}^i\boldsymbol{\omega}_i \times ({}^i\boldsymbol{I}_i\,{}^i\boldsymbol{\gamma}_i + {}^i\boldsymbol{N}_i) + {}^i\boldsymbol{I}_i({}^i\boldsymbol{\gamma}_i \times {}^i\boldsymbol{\omega}_i + {}^i\boldsymbol{\iota}_i) \\
&+ {}^i\boldsymbol{\gamma}_i \times {}^i\boldsymbol{I}_i\,{}^i\boldsymbol{\omega}_i
\end{aligned} \tag{31}
$$

$$
\begin{aligned}
{}^i\ddot{\boldsymbol{N}}_i = {}&{}^i\boldsymbol{\gamma}_i \times ({}^i\boldsymbol{I}_i\,{}^i\boldsymbol{\gamma}_i + 2\,{}^i\boldsymbol{N}_i) + {}^i\boldsymbol{\iota}_i \times {}^i\boldsymbol{I}_i\,{}^i\boldsymbol{\omega}_i \\
&+ {}^i\boldsymbol{I}_i[2({}^i\boldsymbol{\iota}_i \times {}^i\boldsymbol{\omega}_i) + {}^i\boldsymbol{\omega}_i \times ({}^i\boldsymbol{\omega}_i \times {}^i\boldsymbol{\gamma}_i) + {}^i\boldsymbol{\varsigma}_i] \\
&+ {}^i\boldsymbol{\omega}_i \times [{}^i\boldsymbol{\omega}_i \times {}^i\boldsymbol{I}_i\,{}^i\boldsymbol{\gamma}_i + 2\,{}^i\boldsymbol{I}_i({}^i\boldsymbol{\iota}_i + {}^i\boldsymbol{\gamma}_i \times {}^i\boldsymbol{\omega}_i) + {}^i\dot{\boldsymbol{N}}_i]
\end{aligned} \tag{32}
$$

$$
{}^i\boldsymbol{f}_i = {}^i\boldsymbol{R}_{i+1}\,{}^{i+1}\boldsymbol{f}_{i+1} + {}^i\boldsymbol{F}_i \tag{33}
$$

$$
{}^i\dot{\boldsymbol{f}}_i = {}^i\boldsymbol{R}_{i+1}\,{}^{i+1}\dot{\boldsymbol{f}}_{i+1} + {}^i\dot{\boldsymbol{F}}_i \tag{34}
$$

$$
{}^i\ddot{\boldsymbol{f}}_i = {}^i\boldsymbol{R}_{i+1}\,{}^{i+1}\ddot{\boldsymbol{f}}_{i+1} + {}^i\ddot{\boldsymbol{F}}_i \tag{35}
$$

$$
\begin{aligned}
{}^i\boldsymbol{n}_i = {}&{}^i\boldsymbol{R}_{i+1}\,{}^{i+1}\boldsymbol{n}_{i+1} + {}^i\boldsymbol{p}_{c_i,i} \times {}^i\boldsymbol{F}_i + {}^i\boldsymbol{p}_{i,i\text{-}1} \times {}^i\boldsymbol{f}_i \\
&+ {}^i\boldsymbol{N}_i
\end{aligned} \tag{36}
$$

$$
\begin{aligned}
{}^i\dot{\boldsymbol{n}}_i = {}&{}^i\boldsymbol{R}_{i+1}\,{}^{i+1}\dot{\boldsymbol{n}}_{i+1} + ({}^i\boldsymbol{\omega}_i \times {}^i\boldsymbol{p}_{c_i,i}) \times {}^i\boldsymbol{F}_i \\
&+ {}^i\boldsymbol{p}_{c_i,i} \times {}^i\dot{\boldsymbol{F}}_i + ({}^i\boldsymbol{\omega}_i \times {}^i\boldsymbol{p}_{i,i\text{-}1} + \dot{d}_i\,{}^i\hat{\boldsymbol{z}}_{i-1}) \times {}^i\boldsymbol{f}_i \\
&+ {}^i\boldsymbol{p}_{i,i\text{-}1} \times {}^i\dot{\boldsymbol{f}}_i + {}^i\dot{\boldsymbol{N}}_i
\end{aligned} \tag{37}
$$

[2]Snap is the compact term used for defining a fourth-order time derivative.

$$
\begin{aligned}
{}^i\ddot{\boldsymbol{n}}_i = {}&{}^i\boldsymbol{R}_{i+1}\,{}^{i+1}\ddot{\boldsymbol{n}}_{i+1} + 2\,({}^i\boldsymbol{\omega}_i \times {}^i\boldsymbol{p}_{c_i,i}) \times {}^i\dot{\boldsymbol{F}}_i \\
&+ [{}^i\boldsymbol{\gamma}_i \times {}^i\boldsymbol{p}_{c_i,i} + {}^i\boldsymbol{\omega}_i \times ({}^i\boldsymbol{\omega}_i \times {}^i\boldsymbol{p}_{c_i,i})] \times {}^i\boldsymbol{F}_i \\
&+ [{}^i\boldsymbol{\gamma}_i \times {}^i\boldsymbol{p}_{i,i\text{-}1} + {}^i\boldsymbol{\omega}_i \times ({}^i\boldsymbol{\omega}_i \times {}^i\boldsymbol{p}_{i,i\text{-}1}) + \ddot{d}_i\,{}^i\hat{\boldsymbol{z}}_{i-1} \\
&+ 2\dot{d}_i\,{}^i\boldsymbol{\omega}_i \times {}^i\hat{\boldsymbol{z}}_{i-1}] \times {}^i\boldsymbol{f}_i + {}^i\boldsymbol{p}_{c_i,i} \times {}^i\ddot{\boldsymbol{F}}_i + {}^i\ddot{\boldsymbol{N}}_i \\
&+ 2\,({}^i\boldsymbol{\omega}_i \times {}^i\boldsymbol{p}_{i,i\text{-}1} + \dot{d}_i\,{}^i\hat{\boldsymbol{z}}_{i-1}) \times {}^i\dot{\boldsymbol{f}}_i + {}^i\boldsymbol{p}_{i,i\text{-}1} \times {}^i\ddot{\boldsymbol{f}}_i
\end{aligned} \tag{38}
$$

$$
\tau_{e,\,i} = \begin{cases} {}^i\boldsymbol{n}_i^T\;{}^i\hat{\boldsymbol{z}}_{i-1} & \text{if joint } i \text{ is revolute} \\ {}^i\boldsymbol{f}_i^T\;{}^i\hat{\boldsymbol{z}}_{i-1} & \text{if joint } i \text{ is prismatic} \end{cases} \tag{39}
$$

$$
\dot{\tau}_{e,\,i} = \begin{cases} ({}^i\dot{\boldsymbol{n}}_i + {}^i\boldsymbol{n}_i \times {}^i\boldsymbol{\omega}_i)^T\,{}^i\hat{\boldsymbol{z}}_{i-1} & \text{revolute} \\ ({}^i\dot{\boldsymbol{f}}_i + {}^i\boldsymbol{f}_i \times {}^i\boldsymbol{\omega}_i)^T\,{}^i\hat{\boldsymbol{z}}_{i-1} & \text{prismatic} \end{cases} \tag{40}
$$

$$
\ddot{\tau}_{e,\,i} = \begin{cases} [{}^i\ddot{\boldsymbol{n}}_i + 2({}^i\dot{\boldsymbol{n}}_i \times {}^i\boldsymbol{\omega}_i) + {}^i\boldsymbol{n}_i \times {}^i\boldsymbol{\gamma}_i \\ \quad + ({}^i\boldsymbol{n}_i \times {}^i\boldsymbol{\omega}_i) \times {}^i\boldsymbol{\omega}_i]^T\,{}^i\hat{\boldsymbol{z}}_{i-1} & \text{revolute} \\ [{}^i\ddot{\boldsymbol{f}}_i + 2({}^i\dot{\boldsymbol{f}}_i \times {}^i\boldsymbol{\omega}_i) + {}^i\boldsymbol{f}_i \times {}^i\boldsymbol{\gamma}_i \\ \quad + ({}^i\boldsymbol{f}_i \times {}^i\boldsymbol{\omega}_i) \times {}^i\boldsymbol{\omega}_i]^T\,{}^i\hat{\boldsymbol{z}}_{i-1} & \text{prismatic} \end{cases} \tag{41}
$$

$$
\ddot{\theta}_{m,\,i} = \frac{\ddot{\tau}_{e,\,i}}{K_i} + \ddot{q}_i \tag{42}
$$

$$
\tau_i = B_i\ddot{\theta}_{m,\,i} + \tau_{e,\,i}, \tag{43}
$$

where:

- m_i mass of augmented link i;
- ${}^i\boldsymbol{I}_i$ central inertia tensor of augmented link i, in frame i;
- $\boldsymbol{F}_i$ total force acting on the CoM of link i;
- $\boldsymbol{N}_i$ total torque acting on link i;
- $\boldsymbol{f}_i$ total force exerted on link i by link $i-1$;
- $\boldsymbol{n}_i$ total torque exerted on link i by link $i-1$.

It is important to remark that m_i represents the mass of the *whole* augmented link, while ${}^i\boldsymbol{I}_i$ accounts for all of the inertial properties of link i, *except* for the drive inertia moment B_i.

For the initialization step of the backward recursion, $\boldsymbol{f}_{N+1}$ and $\boldsymbol{n}_{N+1}$ are, respectively, the forces and torques exerted by the end-effector on the environment, i.e., the *opposite* of the external forces and torques acting on the end-effector. If present, these should be passed to the algorithm as an additional input, otherwise they should be set to zero. If the external forces/torques are already expressed in frame N, then ${}^N\boldsymbol{R}_{N+1}$ will be the 3×3 identity matrix.

Summarizing, given a desired $\boldsymbol{q}_d$ and its first four derivatives (at any given instant of time), the use of eqs. (17)–(26) and (27)–(43) will provide the nominal inverse dynamics torque

$$
\boldsymbol{\tau}_d = \text{EJNEA}\,(\boldsymbol{q}_d, \dot{\boldsymbol{q}}_d, \ddot{\boldsymbol{q}}_d, \dddot{\boldsymbol{q}}_d, \ddddot{\boldsymbol{q}}_d)\,, \tag{44}
$$

being the algorithm seen as a subroutine with five suitable input arguments. While (40) is not strictly needed for the computation of $\boldsymbol{\tau}_d$ in (44), it will become necessary for real-time control, as explained in Sec. IV.

IV. Application to Control

Through the use of the EJNEA subroutine, it is possible to implement also a feedback linearization control law for robots with elastic joints. In fact, the required control torque (12) can be obtained by simply calling this subroutine with

$$
\boldsymbol{\tau} = \text{EJNEA}\,(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}, \dddot{\boldsymbol{q}}, \boldsymbol{v})\,, \tag{45}
$$

where $\boldsymbol{v}$ is computed from (13), $\ddot{\boldsymbol{q}}$ from (14), and $\dddot{\boldsymbol{q}}$ from (15). In the rest of this section, we will focus on how to implement efficiently (14) and (15) in an algorithmic way.

We start by noting that the evaluation of (14) corresponds to the task of solving the direct dynamics problem for a rigid robot, in which case this is of interest mainly for performing simulations and *not* for control purposes. Several methods addressing this task are available, but since in robots with elastic joints a solution to the direct dynamics is required *also* for control purposes, real time performance will be the core aspect to consider.

The first issue is how to compute the generalized inertia matrix $\boldsymbol{M}(\boldsymbol{q})$ at the current value $\boldsymbol{q}$. In principle, this could be done by using the RNEA. If $\boldsymbol{e}_i$ is the ith column of the $N \times N$ identity matrix, then the standard Newton-Euler algorithm can be called with $\boldsymbol{g} = \boldsymbol{0}$ (removing gravity) in the form RNEA$(\boldsymbol{q}, \boldsymbol{0}, \boldsymbol{e}_i)$, returning as output the ith column $\boldsymbol{m}_i(\boldsymbol{q})$ of $\boldsymbol{M}(\boldsymbol{q})$. Repeating this computation for $i = 1, \ldots, N$ would return the full inertia matrix in $O(N^2)$ complexity. However, this is definitely not the best way to proceed. A much faster method is the Composite Rigid Body Algorithm (CRBA), which was specifically designed for this purpose and first appeared (as Method 3) in [19]. Thus, we evaluate the inertia matrix as

$$\boldsymbol{M}(\boldsymbol{q}) = \text{CRBA}(\boldsymbol{q}).$$

The second issue involves the inversion of the obtained inertia matrix in order to solve for $\ddot{\boldsymbol{q}}$ and $\dddot{\boldsymbol{q}}$. In general, inverting a (numeric) matrix is not the most efficient solution. Letting

$$\boldsymbol{A} = \boldsymbol{M}(\boldsymbol{q}) \tag{46}$$
$$\boldsymbol{b}_1 = \boldsymbol{\tau}_e - \boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \tag{47}$$
$$\boldsymbol{b}_2 = \dot{\boldsymbol{\tau}}_e - \left[\dot{\boldsymbol{M}}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \dot{\boldsymbol{n}}(\boldsymbol{q}, \dot{\boldsymbol{q}})\right], \tag{48}$$

a better approach would be to solve the two linear systems

$$\boldsymbol{A}\ddot{\boldsymbol{q}} = \boldsymbol{b}_1 \tag{49}$$
$$\boldsymbol{A}\dddot{\boldsymbol{q}} = \boldsymbol{b}_2. \tag{50}$$

This approach is in fact rather standard in direct dynamics of rigid robots. The solution can be computed using any algorithm designed for solving linear systems. In particular, since $\boldsymbol{A} = \boldsymbol{M}(\boldsymbol{q})$ is always positive definite and symmetric, specialized methods could be used, such as using the $\boldsymbol{LL}^T$ (Cholesky) or $\boldsymbol{LDL}^T$ decompositions. Obviously, in our case, we need first to solve (49), then compute $\boldsymbol{b}_2$, and finally solve (50). Note that $\boldsymbol{A}$ needs only to be computed and decomposed once (at a given configuration $\boldsymbol{q}$).

The last issue left is how to compute $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$ algorithmically, using the obtained values of $\ddot{\boldsymbol{q}}$ and $\dddot{\boldsymbol{q}}$. This can be done by using EJNEA or, even better, smaller versions of it. By EJNEA$_2^*(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$ we denote a reduced version of EJNEA, returning $\boldsymbol{\tau}_e$. Note that this corresponds to the standard RNEA. Similarly, by EJNEA$_3^*(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}, \dddot{\boldsymbol{q}})$ we denote a reduced version returning $\dot{\boldsymbol{\tau}}_e$. Then:

$$\boldsymbol{b}_1 = \boldsymbol{\tau}_e - \text{EJNEA}_2^*(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{0}) \tag{51}$$
$$\boldsymbol{b}_2 = \dot{\boldsymbol{\tau}}_e - \text{EJNEA}_3^*(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}, \boldsymbol{0}). \tag{52}$$

The computation of $\boldsymbol{b}_1$ through the use of RNEA is a standard step also in the direct dynamics of rigid robots. Note that it is possible to implement simplified versions of EJNEA$_2^*$ and EJNEA$_3^*$ including the built-in assumptions $\ddot{\boldsymbol{q}} = \boldsymbol{0}$ and $\dddot{\boldsymbol{q}} = \boldsymbol{0}$, but the performance gain would be negligible.

Summarizing, the necessary steps (with complexity) are:

- one run of CRBA to get $\boldsymbol{A} = \boldsymbol{M}(\boldsymbol{q})$ – $O(N^2)$
- one decomposition $\boldsymbol{A} = \boldsymbol{LDL}^T$ – $O(N^3)$
- one run of EJNEA$_2^*$ to get $\boldsymbol{b}_1$ – $O(N)$
- solve the linear system (49), using the $\boldsymbol{LDL}^T$ – $O(N^2)$
- one run of EJNEA$_3^*$ to get $\boldsymbol{b}_2$ – $O(N)$
- solve the linear system (50), using the $\boldsymbol{LDL}^T$ – $O(N^2)$
- one run of EJNEA to obtain the input torque $\boldsymbol{\tau}$ – $O(N)$

Therefore, the total computational cost grows asymptotically as $O(N^3)$.

For the sake of completeness, we mention also a completely different way to compute the direct dynamics, namely the Articulated Body Algorithm (ABA). For rigid robots, ABA allows to obtain $\ddot{\boldsymbol{q}}$ directly and in $O(N)$ time, without computing and then inverting $\boldsymbol{M}(\boldsymbol{q})$. An early version of ABA, which was assuming $\dot{\boldsymbol{q}} = \boldsymbol{0}$, first appeared in [20], while its full form can be found in [21, Ch. 6]. In principle, the first version of the algorithm could also work for computing $\dddot{\boldsymbol{q}}$, by simply using $\boldsymbol{b}_2$ as input in the place of $\boldsymbol{\tau}$. However, resorting to ABA for controlling robots with elastic joints is not convenient in practice. In fact, already for rigid robots, the actual computational cost of ABA is much larger than the one based on CRBA, until reaching approximately a number of dofs $N \geq 9$. For elastic joint robots, this issue is even more emphasized, since $\boldsymbol{M}(\boldsymbol{q})$ needs to be computed and decomposed only once with the proposed algorithm, while ABA would need to run twice in this case.

V. Numerical Results

To validate our method, we have performed several motion tests on elastic joint robots of increasing model complexity, and compared the inverse dynamics torques produced as output by EJNEA with those obtained by evaluating the symbolic model terms. We show here one such example, using a dynamic model of a 7R KUKA LWR with approximate dynamic parameters. Due to the lack of space, we do not report results on the application of EJNEA to the trajectory control problem discussed in Sec. IV.

The kinematic and dynamic parameters used for the numerical evaluations are listed in Tab. I. The first four rows report the actual Denavit-Hartenberg (kinematic) parameters of the considered KUKA LWR manipulator. For the dynamic parameters, the values of link masses and inertias[3] (shown rounded in the table) and the position of the link CoMs were taken from the V-Rep™ library, where such a robot model is freely available. Please note that these dynamic parameters are not meant to be the true ones; they are only realistic,

[3]For compactness, we used the inertia units [g m^2] = 10^{-3} · [kg m^2]

TABLE I

KINEMATIC AND DYNAMIC PARAMETERS

		1	2	3	4	5	6	7
α_i	rad	$\pi/2$	$-\pi/2$	$-\pi/2$	$\pi/2$	$\pi/2$	$-\pi/2$	0
a_i	cm	0	0	0	0	0	0	0
d_i	cm	0	0	40	0	39	0	0
θ_i	rad	q_1	q_2	q_3	q_4	q_5	q_6	q_7
m_i	kg	2.7	2.7	2.7	2.7	1.7	1.6	0.3
${}^i\boldsymbol{p}_{cx_i,i}$	cm	0.1340	0.1340	0.1340	0.1340	0.0993	0.0259	0
${}^i\boldsymbol{p}_{cy_i,i}$	cm	8.7777	2.6220	8.7777	2.6220	11.1650	0.5956	0
${}^i\boldsymbol{p}_{cz_i,i}$	cm	2.6220	8.7777	2.6220	8.7777	2.6958	0.5328	6.3
${}^i\boldsymbol{I}_{xx_i}$	g m^2	16.341	16.341	16.341	16.341	9.818	3.012	0.102
${}^i\boldsymbol{I}_{xy_i}$	g m^2	0.003	0.001	0.003	0.001	0.001	0	0
${}^i\boldsymbol{I}_{xz_i}$	g m^2	0.001	0.003	0.001	0.003	0.001	0	0
${}^i\boldsymbol{I}_{yy_i}$	g m^2	5.026	16.173	5.026	16.173	3.708	3.023	0.102
${}^i\boldsymbol{I}_{yz_i}$	g m^2	3.533	3.533	3.533	3.533	3.094	0.019	0
${}^i\boldsymbol{I}_{zz_i}$	g m^2	16.173	5.026	16.173	5.026	9.092	3.414	0.158
B_i	kg m^2	3.20	3.05	1.98	2.06	0.801	0.48	0.381
K_i	Nm/rad	1000	1000	1000	1000	1000	1000	1000

in order to show the performance of the algorithm for a generic 7R robot. The values for the drive inertia moments were taken from [22], while the joint stiffnesses have been chosen arbitrarily.

The desired link motion was specified by smooth rest-to-rest trajectories (polynomials of degree 7, with zero initial and final boundary conditions on velocity, acceleration, and jerk) lasting $T = 4$ s and having peak (absolute) velocities between $90°/\text{s}$ and $120°/\text{s}$. The obtained numerical results are shown in Figs. 1 to 5. During motion, the elastic joint deflections are as large as $3°$ (Fig. 3). The maximum differences between the torques that would be needed in the rigid case and those needed in the elastic case range between 2% and 5% of the peak torques for the robot with elastic joints (Fig. 5).

The algorithm was implemented in Matlab™, automatically converted to C code, and then run as a `mex` function on a standard personal computer. Remarkably, 4 seconds of inverse dynamics computations with a sampling time $\Delta t = 0.01$ s took only 13 ms of execution time, achieving an average of 33.258 μs per iteration and thus proving that the method is perfectly feasible for real-time control use.

VI. CONCLUSIONS

An efficient numerical solution has been presented for computing the inverse dynamics of robot manipulators with elastic joints, based on a differential extension of the recursive Newton-Euler algorithm for rigid robots. The solution is general and has a computational complexity that grows linearly with the number N of joints, as in the rigid case. The recursive numerical algorithm avoids even in the joint elastic case the effort of deriving symbolic models and customizing the dynamics for efficiency when N is large.

The complexity of a Lagrangian-based feedback linearization design has barred so far the use of such nonlinear control law for all but very simple robotic structures with elastic joints. Using instead our Newton-Euler algorithm, together with a numerical factorization of the link inertia matrix, we have also addressed for the first time (though still in a preliminary way) the issue of real-time implementation of feedback linearization control for robots with joint elasticity.

The extension of the presented approach to robots driven by Variable Stiffness Actuators is under preparation.

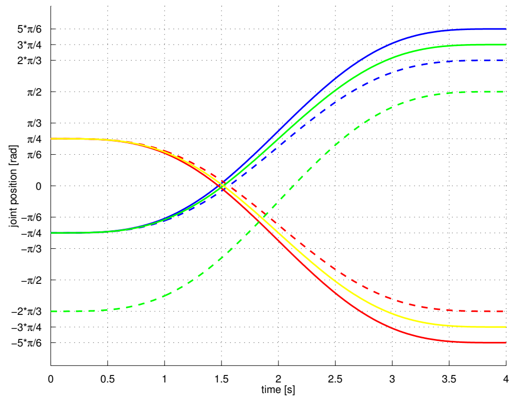

Fig. 1. Joint positions $\boldsymbol{q}_d$. Coding for this and the following figures: solid blue = joint 1, dashed blue = 2, solid red = 3, dashed red = 4, solid green = 5, dashed green = 6, and solid yellow = 7

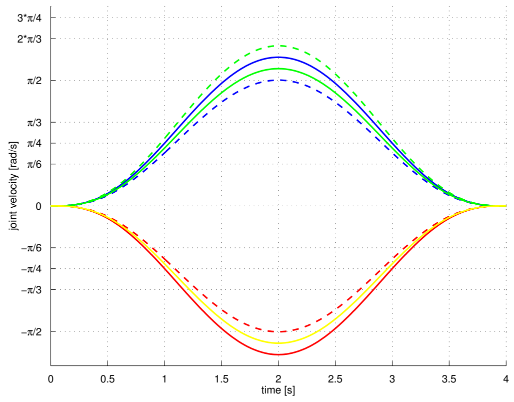

Fig. 2. Joint velocities $\dot{\boldsymbol{q}}_d$

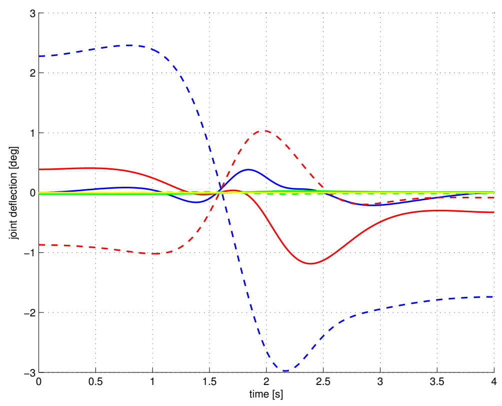

Fig. 3. Joint deflections $\boldsymbol{\phi}_d = \boldsymbol{\theta}_{m,d} - \boldsymbol{q}_d$

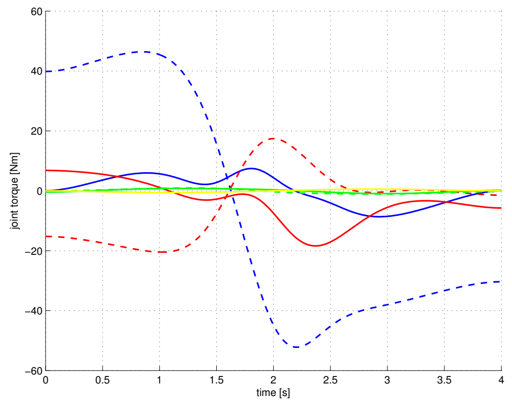

Fig. 4. Motor torques $\boldsymbol{\tau}_d$

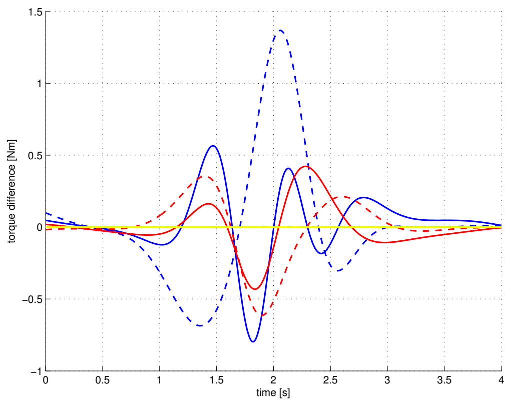

Fig. 5. Difference between the desired motor torque $\boldsymbol{\tau}_d$ in the joint elastic case and the desired motor torque $\boldsymbol{\tau}_{r,d}$ in the rigid case

Acknowledgement

We would like to thank Taigo Maria Bonanni and Martina Deturres for their early work on this subject.

References

[1] [Online]. Available: http://harmonicdrive.net/

[2] T. D. Tuttle and W. P. Seering, "A nonlinear model of a harmonic drive gear transmission," *IEEE Trans. on Robotics and Automation*, vol. 12, no. 3, pp. 368–374, 1996.

[3] G. Hirzinger, A. Albu-Schäffer, M. Hähnle, I. Schaefer, and N. Sporer, "On a new generation of torque controlled light-weight robots," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 2001, pp. 3356–3363.

[4] A. Albu-Schäffer, S. Haddadin, C. Ott, A. Stemmer, T. Wimböck, and G. Hirzinger, "The DLR lightweight robot – Lightweight design and soft robotics control concepts for robots in human environments," *Industrial Robot*, vol. 34, no. 5, pp. 376–385, 2007.

[5] R. Bischoff, J. Kurth, G. Schreiber, R. Koeppe, A. Albu-Schäffer, A. Beyer, O. Eiberger, S. Haddadin, A. Stemmer, G. Grunwald, and G. Hirzinger, "The KUKA-DLR lightweight robot arm: A new reference platform for robotics research and manufacturing," in *Proc. 41st Int. Symp. on Robotics*, 2010, pp. 1–10.

[6] A. De Luca, A. Albu-Schäffer, S. Haddadin, and G. Hirzinger, "Collision detection and safe reaction with the DLR-III lightweight robot arm," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2006, pp. 1623–1630.

[7] G. Pratt and M. Williamson, "Series elastic actuators," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 1995, pp. 399–406.

[8] A. De Luca and W. Book, "Robots with flexible elements," in *Handbook of Robotics*, B. Siciliano and O. Khatib, Eds., Springer, 2008, pp. 287–319.

[9] P. Tomei, "A simple PD controller for robots with elastic joints," *IEEE Trans. on Automatic Control*, vol. 36, no. 10, pp. 1208–1213, 1991.

[10] M. Thümmel, M. Otter, and J. Bals, "Control of robots with elastic joints based on automatic generation of inverse dynamics models," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2001, pp. 925–930.

[11] R. Höpler and M. Thümmel, "Symbolic computation of the inverse dynamics of elastic joint robots," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 2004, pp. 4314–4319.

[12] A. De Luca and P. Lucibello, "A general algorithm for dynamic feedback linearization of robots with elastic joints," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 1998, pp. 504–510.

[13] C. Ott, *Cartesian Impedance Control of Redundant and Flexible-Joint Robots*, Springer Tracts in Advanced Robotics (STAR), vol. 49, Springer, 2008.

[14] M. Spong, "Modeling and control of elastic joint robots," *ASME J. of Dynamic Systems, Measurement, and Control*, vol. 109, no. 4, pp. 310–319, 1987.

[15] J. Y. S. Luh, M. W. Walker, and R. P. C. Paul, "On-line computational scheme for mechanical manipulators," *ASME J. of Dynamic Systems, Measurement, and Control*, vol. 102, no. 2, pp. 69–76, 1980.

[16] C. Guarino Lo Bianco and E. Fantini, "A recursive Newton-Euler approach for the evaluation of generalized forces derivatives," in *Proc. 12th IEEE Int. Conf. on Methods and Models in Automation and Robotics*, 2006, pp. 739–744.

[17] C. Guarino Lo Bianco, "Evaluation of generalized force derivatives by means of a recursive newton-euler approach," *IEEE Trans. on Robotics*, vol. 25, no. 4, pp. 954–959, 2009.

[18] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modeling, Planning and Control*, 3rd ed. London: Springer, 2008.

[19] M. W. Walker and D. E. Orin, "Efficient dynamic computer simulation of robotic mechanisms," *ASME J. of Dynamic Systems, Measurement, and Control*, vol. 104, no. 3, pp. 205–211, 1982.

[20] R. Featherstone, "The calculation of robot dynamics using articulated-body inertias," *Int. J. of Robotics Research*, vol. 2, no. 1, pp. 13–30, 1983.

[21] ——, *Robot Dynamics Algorithms*. Kluwer Academic Publ., 1987.

[22] A. J. A. Jubien, M. Gautier, "Dynamic identification of the Kuka LWR robot using motor torques and joint torque sensors data," in *Proc. 19th IFAC World Congr,*, 2014, pp. 8391–8396.