

# Inverse, forward and other dynamic computations computationally optimized with sparse matrix factorizations

Francesco Nori<sup>1</sup>

**Abstract**— We propose an algorithm to compute the dynamics of articulated rigid-bodies with different sensor distributions. Prior to the on-line computations, the proposed algorithm performs an off-line optimisation step to simplify the computational complexity of the underlying solution. This optimisation step consists in formulating the dynamic computations as a system of linear equations. The computational complexity of computing the associated solution is reduced by performing a permuted  $LU$ -factorisation with off-line optimised permutations. We apply our algorithm to solve classical dynamic problems: inverse and forward dynamics. The computational complexity of the proposed solution is compared to ‘gold standard’ algorithms: recursive Newton-Euler and articulated body algorithm. It is shown that our algorithm reduces the number of floating point operations with respect to previous approaches. We also evaluate the numerical complexity of our algorithm by performing tests on dynamic computations for which no gold standard is available.

## I. PREVIOUS WORKS

Real-time control of complex robots (such as humanoids) asks for efficient ways of computing position, velocity, acceleration and applied wrenches on all the bodies composing the robot articulated chain. Within this paper, we will refer to these quantities with the term *dynamic variables* even though they include positions and velocities which are strictly speaking kinematic. Efficiently computing the dynamic variables of a (possibly free-floating) robot is nowadays a solved theoretical problem. Several state of the art algorithms [1] have a computational complexity which scales linearly in the number of rigid links composing the robot. All these efficient algorithms can be obtained by exploiting the problem sparsity which derives from the iterative propagation of forces, torques and accelerations across articulated structures. Classical problems have been extensively optimized from the computational complexity point of view. As discussed in [1, Table 10.1] the recursive Newton-Euler algorithm as implemented in [2, Algorithm 5.7] is the most computationally efficient solution of the inverse dynamics problem. Similarly, the articulated body algorithm as implemented in [3, method 3] is the most computationally efficient solution of the forward dynamics problem. Yet another relevant problem is the hybrid dynamics solution, which can be efficiently solved with the articulated-body hybrid dynamics [1, Section 9.2].

In this paper we consider the computationally efficient solution of dynamic problems which do not fall within

the scope of the inverse, forward and hybrid dynamics even though they are of practical interest. The paper is organized as follows. Section II present the notation. Section III presents a specific formulation of the Newton-Euler equations and Section III-A discuss the matrix form of these constraints. Section IV presents the inverse dynamics problem and its solution with the recursive Newton-Euler algorithm (Section IV-A). Section V presents the forward dynamics problem and its solution with the articulated body algorithm (Section V-A). An alternative but computationally comparable solution of the inverse and forward dynamics is proposed in Section IV-D and Section V-D respectively. Finally, Section VI extends this solution to dynamic problems which do not fall within the scope of the inverse, forward and hybrid dynamics.

## II. NOTATION

Let  $\mathcal{A}$  and  $\mathcal{B}$  be two arbitrary sets with cardinality  $|\mathcal{A}| = m$  and  $|\mathcal{B}| = n$ . For any element in  $a \in \mathcal{A}$  let’s associate a vector  $d_a$ . Similarly, for any pair of elements  $a, b$  with  $a \in \mathcal{A}$  and  $b \in \mathcal{B}$ , let’s associate a unique matrix  $D_{a,b}$ . Let  $p = (a_1, \dots, a_m) \in S_{\mathcal{A}}$  and  $q = (b_1, \dots, b_n) \in S_{\mathcal{B}}$  be two permutations of the elements in  $\mathcal{A}$  and  $\mathcal{B}$  respectively<sup>1</sup>. We define the permuted vector  $\mathbf{d}_p$  and the permuted matrix  $\mathbf{D}_{p,q}$  induced by  $p$  and  $q$  as follows:

$$\mathbf{d}_p = \begin{bmatrix} d_{a_1} \\ \vdots \\ d_{a_m} \end{bmatrix}, \quad \mathbf{D}_{p,q} = \begin{bmatrix} D_{a_1,b_1} & \dots & D_{a_1,b_n} \\ \vdots & \ddots & \vdots \\ D_{a_m,b_1} & \dots & D_{a_m,b_n} \end{bmatrix}. \quad (1)$$

As for dynamic quantities, in this paper we follow the same notation used in [1] but we avoid using the bold symbols for matrices, since the bold symbols are reserved for distinguishing  $D_{a,b}$ ,  $a \in \mathcal{A}$ ,  $b \in \mathcal{B}$  from  $\mathbf{D}_{p,q}$ ,  $p \in S_{\mathcal{A}}$ ,  $q \in S_{\mathcal{B}}$ ;  $v$  denotes the spatial velocity (a six dimensional vector including angular velocities in the first three components and the rest as linear velocities),  $a$  denotes the spatial acceleration (again angular and then linear),  $f$  denotes the spatial force (couples in the first three components and forces in the remaining),  ${}^B X_A$  is the motion vector transformation from  $A$  to  $B$  coordinates and  ${}^B X_A^*$  is the analogous transformation for a force vector. Given a vector  $v$  and its coordinates  ${}^A v$  in  $A$  we denote with  ${}^A \dot{v}$  its temporal derivative in the  $A$  coordinates. The Euclidean cross operator  $\times$  (on  $\mathbb{R}^3$ ) is defined as the usual vector product, while for a spatial

<sup>1</sup>Francesco Nori is with iCub Facility, Istituto Italiano di Tecnologia, Italy. francesco.nori@iit.it. This paper was supported by the FP7 EU projects CoDyCo (No. 600716 ICT 2011.2.1 Cognitive Systems and Robotics), and An.Dy funded by the European Union’s Horizon 2020 Research and Innovation Programme under Grant Agreement No. 731540.

<sup>1</sup>Equivalently, let  $p \in S_{\mathcal{A}}$  and  $q \in S_{\mathcal{B}}$ , i.e. let  $p$  and  $q$  be two elements of the symmetric group on  $\mathcal{A}$  and  $\mathcal{B}$  respectively.

velocity  $v$  the cross spatial cross operator and its dual  $\times^*$  are defined as follows:

$$v \times = \begin{bmatrix} \omega \\ \dot{p} \end{bmatrix} \times = \begin{bmatrix} \omega \times & 0 \\ \dot{p} \times & \omega \times \end{bmatrix}, v \times^* = \begin{bmatrix} \omega \\ \dot{p} \end{bmatrix} \times^* = \begin{bmatrix} \omega \times & \dot{p} \times \\ 0 & \omega \times \end{bmatrix}$$

In the following sections, we restrict the analysis to robots described by kinematic trees (i.e. robots with no kinematic loops) composed of  $N_B$  rigid links numbered from 0 to  $N_B$ , 0 being the selected fixed base (world reference frame) and 1 being the selected floating base (reference rigid body in the articulated chain). Links numbering is obtained by defining a suitable spanning tree where each rigid link is associated to a unique node in the tree. Numbers can be always selected in a topological order so that each node  $i$  has a higher number than its unique parent  $\lambda_i$  and a smaller number than all the nodes in the set of its children  $\mu_i$ . Links  $i$  and  $\lambda_i$  are coupled with the joint  $i$  whose joint motion constraints are modelled with  $S_i \in \mathbb{R}^{6 \times 1}$  (therefore without loss of generality we are assuming that all joints have a single degree of freedom). For each rigid link  $i$ , the system is modelled also by supplying the spatial inertia tensor<sup>2</sup>  $I_i$  and the motion-vector transform from the reference frame of the rigid link  $i$  to the reference frame of the rigid link  $j$ , denoted  ${}^j X_i$ . For each link  $i$  the considered kinematic variables are:

- $v_i$ : the link spatial velocity,
- $q_i$ : the joint  $i$  position,
- $\dot{q}_i$ : the joint  $i$  velocity,

Similarly, the dynamic variables associated to the link  $i$  are:

- $a_i$ : the link spatial accelerations,
- $\ddot{q}_i$ : the joint  $i$  acceleration,
- $\tau_i$ : the joint  $i$  torque,
- $f_i$ : the spatial force transmitted to body  $i$  from  $\lambda_i$ ,
- $f_i^x$ : external forces acting on body  $i$ .

*Remark 1:* All these variables are expressed in body  $i$  coordinates including  $f_i^x$  which is more often expressed in absolute (i.e. body 0) coordinates.

### III. DYNAMIC CONSTRAINTS

Let's assume that all kinematic quantities, i.e. those depending on  $q$  and  $\dot{q}$  have been precomputed. In practice, these quantities are the transformation matrices  ${}^j X_i$ ,  ${}^j X_i^*$  and the linear/angular velocities  $v_{j_i}$ ,  $v_i$ . The latter can be efficiently computed with the following recursion, propagated from  $i = 1$  to  $N_B$ :

$$v_i = {}^i X_{\lambda_i} v_{\lambda_i} + S_i \dot{q}_i, \quad (2)$$

The dynamics quantities instead ( $\tau$ ,  $\ddot{q}$ ,  $f_1^x, \dots, f_{N_B}^x$ ,  $a_1, \dots, a_{N_B}$ ,  $f_1, \dots, f_{N_B}$ ) have to satisfy the Newton-Euler equations:

<sup>2</sup>The spatial inertia tensor of the rigid link  $i$  has the following form in the link reference frame:

$$I_i = \begin{bmatrix} I_{C,i} + m_i c_i \times c_i^\top & m_i c_i \times \\ m_i c_i \times^\top & m_i I_{3 \times 3} \end{bmatrix},$$

where  $I_{C,i}$  is the spatial inertia tensor with respect to the body's centre of mass,  $m_i$  is the total mass,  $c_i$  is the relative displacement between the centre of mass and the origin of the link reference frame.

$$a_i = {}^i X_{\lambda_i} a_{\lambda_i} + S_i \ddot{q}_i + c_i, \quad (3a_i)$$

$$\tau_i = S_i^\top f_i, \quad (3\tau_i)$$

$$f_i = I_i a_i + \nu_i - f_i^x + \sum_{j \in \mu_i} {}^i X_j^* f_j. \quad (3f_i)$$

where we defined  $\nu_i = v_i \times^* I_i v_i$  and  $c_i = v_i \times S_i \dot{q}_i$ .

#### A. Dynamic constraints in matrix form

Let's define the set of dynamic variables  $\mathcal{D} = \{a_i, f_i, \tau_i, f_i^x, \ddot{q}_i\}_{i=1}^{N_B}$ . We have  $|\mathcal{D}| = 5N_B$ . The elements in  $\mathcal{D}$  can be thought as generic quantities, not necessarily expressed in a specific reference frame. As described in Section II, for any  $a \in \mathcal{D}$ , let's associate a vector  $d_a$  (e.g.  $d_{a_i}$  might correspond to the coordinates of  $a_i$  expressed in a specific reference frame). Given a permutation  $q \in \mathcal{S}_{\mathcal{D}}$ ,  $d_q$  contains an ordered sequence of the elements in  $\mathcal{D}$  expressed in specific reference frame(s).

Equations (3) can be seen as a set of equations which the vector of dynamic variables  $d_p$  have to satisfy. Let's put these constraints in matrix form by defining the set of constraints  $\mathcal{C} = \{c_{(3a_i)}, c_{(3f_i)}, c_{(3\tau_i)}\}_{i=1}^{N_B}$ . We have  $|\mathcal{C}| = 3N_B$ . Given  $c \in \mathcal{C}$  let's define vectors  $b_c$  as follows:

$$b_{c_{(3a_i)}} = \begin{cases} {}^i X_0 a_0 + c_i & \text{if } \lambda_i = 0 \\ v_i \times S_i \dot{q}_i & \text{if } \lambda_i \neq 0 \end{cases},$$

$$b_{c_{(3f_i)}} = \nu_i,$$

and  $b_c = 0$  otherwise. Given  $c \in \mathcal{C}$  and  $d \in \mathcal{D}$ , let's define matrices  $D_{c,d}$  as follows (where  $j \in \mu_i$ ):

$$D_{c_{(3a_i)}, a_i} = -1_6, \quad D_{c_{(3a_i)}, a_{\lambda_i}} = {}^i X_{\lambda_i}, \quad (4a)$$

$$D_{c_{(3a_i)}, \ddot{q}_i} = S_i, \quad (4b)$$

$$D_{c_{(3f_i)}, f_i} = -1_6, \quad D_{c_{(3f_i)}, a_i} = I_i, \quad (4c)$$

$$D_{c_{(3f_i)}, f_i^x} = -1_6, \quad D_{c_{(3f_i)}, f_j} = {}^i X_j^*, \quad (4d)$$

$$D_{c_{(3\tau_i)}, \tau_i} = -1_{n_i}, \quad D_{c_{(3\tau_i)}, f_i} = S_i^\top \quad (4e)$$

and  $D_{c,d} = 0$  otherwise.

Given two permutations  $p$  and  $q$  of the elements in  $\mathcal{C}$  and  $\mathcal{D}$ , the permuted matrix  $D_{p,q}$  and permuted vectors  $d_q$ ,  $b_p$  satisfy the following linear equation:

$$D_{p,q}(q, \dot{q}) d_q + b_p(q, \dot{q}) = 0, \quad (5)$$

where we explicitly indicated the dependency on  $q$  and  $\dot{q}$ .

### IV. INVERSE DYNAMICS

The inverse dynamics problem consists in finding  $\tau_1, \dots, \tau_{N_B}$  which satisfies (3) given  $\ddot{q}_1, \dots, \ddot{q}_{N_B}$ ,  $f_1^x, \dots, f_{N_B}^x$ . In [1] the problem is formulated as the computation of the following function:

$$\tau = \text{InvD}(\text{model}, q, \dot{q}, \ddot{q}, f_1^x, \dots, f_{N_B}^x). \quad (6)$$

In the above equations we grayed out some variables that will not play a role in the following sections, and can be assumed either to be contestant (*model*) or measured ( $q, \dot{q}$ ).

### A. Inverse dynamics solved with RNEA

An efficient solution of inverse dynamics is given by the recursive Newton-Euler algorithm (RNEA), described hereafter. Equations (2) and  $(3a_i)$  are propagated from 1 to  $N_B$  with initial conditions  $v_0 = 0$  and  $a_0 = -a_g$  which corresponds to the gravitational spatial acceleration vector expressed in the body frame 0 (null in its first three components and equal to the gravitational acceleration in the last three). Equations  $(3f_i)$  and  $(3\tau_i)$  are propagated from  $N_B$  to 1.

### B. Inverse dynamics solved with matrix inversion

In this Section, we propose a way to solve the inverse dynamics problem with a matrix inversion. Remarkably, (5) represents the set of linear constraints in  $\mathbf{d}_q$ . Additionally, certain components of  $\mathbf{d}_p$  are known since  $\ddot{q}_1, \dots, \ddot{q}_{N_B}, f_1^x, \dots, f_{N_B}^x$  are given:

$$\ddot{q}_i = y_{\ddot{q}_i}, \quad (7\ddot{q}_i)$$

$$f_1^x = y_{f_1^x}. \quad (7f_1^x)$$

These constraints, will extend the set of constraints  $\mathcal{C}$ . In particular, we should define  $\mathcal{C}_{id} = \mathcal{C} \cup \{c_{(7\ddot{q}_i)}, c_{(7f_i^x)}\}_{i=1}^{N_B}$  so that  $|\mathcal{C}_{id}| = 5N_B$ . To extend the definition of  $D_{c,d}$  and  $b_c$  we define:

$$D_{c_{(7\ddot{q}_i)}, \ddot{q}_i} = 1_{n_i}, \quad b_{c_{(7\ddot{q}_i)}} = -y_{\ddot{q}_i}, \quad (8a)$$

$$D_{c_{(7f_i^x)}, f_i^x} = 1_6, \quad b_{c_{(7f_i^x)}} = -y_{f_i^x}, \quad (8b)$$

and  $D_{c,d} = 0$ ,  $b_c = 0$  otherwise. Again, given two permutations  $p_{id}$  and  $q$  of the elements in  $\mathcal{C}_{id}$  and  $\mathcal{D}$ , the permuted matrix  $\mathbf{D}_{p_{id},q}$  and permuted vectors  $\mathbf{d}_q$ ,  $\mathbf{b}_{p_{id}}$  satisfy the following linear equation:

$$\mathbf{D}_{p_{id},q}(q)\mathbf{d}_q + \mathbf{b}_{p_{id}}(q, \dot{q}) = 0, \quad (9)$$

which is representation of the inverse dynamics problem. In particular, inverting the matrix  $\mathbf{D}_{p_{id},q}$  we can compute the solution  $\mathbf{d}_q$  of the inverse dynamics. A more computationally efficient solution can be obtained as described in the following section.

### C. Inverse dynamics solved with forward substitution

In this section we prove that with suitable permutations we can compute  $\mathbf{d}_q$  which solves (9) with a computationally efficient algorithm, which is the forward substitution as defined in [4]. The idea is to build two find permutations  $p_{id}$  and  $q$  of the elements in  $\mathcal{C}_{id}$  and  $\mathcal{D}$  which lead to a lower triangular matrix  $\mathbf{D}_{p_{id},q}$  and then to solve (9) with a forward substitution. The permutations are inspired by the RNEA described in Section IV-A. Let's first consider the  $q$  permutation of the elements in  $\mathcal{D}$ . We choose:

$$q = [f_1^x, \ddot{q}_1, \dots, f_{N_B}^x, \ddot{q}_{N_B}, a_1, \dots, a_{N_B}, f_{N_B}, \tau_{N_B}, \dots, f_1, \tau_1]. \quad (10)$$

Let's also choose a permutation  $p_{id}$  of the elements in  $\mathcal{C}_{id}$ :

$$p_{id} = \left[ c_{(7f_1^x)}, c_{(7\ddot{q}_1)} \dots c_{(7f_{N_B}^x)}, c_{(7\ddot{q}_{N_B})}, c_{(3a_1)} \dots c_{(3a_{N_B})}, c_{(3f_{N_B})}, c_{(3\tau_{N_B})}, \dots, c_{(3f_1)}, c_{(3\tau_1)} \right]. \quad (11)$$

*Property 1:* The permuted matrix  $\mathbf{D}_{p_{id},q}$  induced by the permutations defined in (10) and (11) is lower triangular.

*Proof:* Since  $\mathbf{D}_{p_{id},q}$  is defined by blocks, we structure this proof by considering the blocks that constitute the matrix itself. First, we prove that the blocks in the main diagonal are diagonal matrices. These blocks are:

$$D_{c_{(7f_i^x)}, f_i^x} = 1_6, \quad D_{c_{(7\ddot{q}_i)}, \ddot{q}_i} = 1_{n_i}, \quad D_{c_{(3a_i)}, a_i} = -1_6, \\ D_{c_{(3f_i)}, f_i} = -1_6, \quad D_{c_{(3\tau_i)}, \tau_i} = -1_{n_i},$$

which are indeed diagonal. We are left with proving that the blocks in the upper triangular part of  $\mathbf{D}_{p_{id},q}$  are identically null. We consider the non-null blocks in (4) and prove that each block  $D_{c,d}$  is positioned in the lower triangular part of  $\mathbf{D}_{p_{id},q}$ . This is equivalent to prove that if  $D_{c,d} \neq 0$  then  $i_c \geq i_d$ , being  $i_c$  and  $i_d$  the position of  $c$  and  $d$  in the permutations  $p_{id}$  and  $q$ , respectively. In the definitions given in (4), we can neglect the diagonal blocks, which have been previously considered. We are left with:

$$D_{c_{(3a_i)}, a_{\lambda_i}}, \quad i_{c_{(3a_i)}} = 2N_B + i, \quad i_{a_{\lambda_i}} = 2N_B + \lambda_i \\ D_{c_{(3a_i)}, \ddot{q}_i}, \quad i_{c_{(3a_i)}} = 2N_B + i, \quad i_{\ddot{q}_i} = 2i \\ D_{c_{(3f_i)}, a_i}, \quad i_{c_{(3f_i)}} = 5N_B - 2i + 1, \quad i_{a_i} = 2N_B + i \\ D_{c_{(3f_i)}, f_i^x}, \quad i_{c_{(3f_i)}} = 5N_B - 2i + 1, \quad i_{f_i^x} = 2i - 1 \\ D_{c_{(3f_i)}, f_j}, \quad i_{c_{(3f_i)}} = 5N_B - 2i + 1, \quad i_{f_j} = 5N_B - 2j + 1 \\ D_{c_{(3\tau_i)}, f_i}, \quad i_{c_{(3\tau_i)}} = 5N_B - 2i + 2, \quad i_{f_i} = 5N_B - 2i + 1.$$

Easy computations can show that in these case  $i_c \geq i_d$  considering a numbering scheme with  $\lambda_i < i$  and  $j > i$ ,  $\forall j \in \mu_i$  as described in [1]. ■

### D. Inverse dynamics solved with LU factorization

In Section IV-C, suitable permutations led to a lower triangular structure for the matrix  $\mathbf{D}$  in (8). After obtaining the lower triangular structure, the underlying linear system can be solved with a forward substitution which is a computationally efficient algorithm to solve a linear system. In this section we are interested in computing these permutations by solving the following problem.

*Problem 1:* Given arbitrary permutations  $p$  and  $q$  of the elements in  $\mathcal{C}_{id}$  and  $\mathcal{D}$ , compute permutation matrices  $P$  and  $Q$  to obtain a triangular matrix  $P\mathbf{D}_{p,q}Q$ .

In consideration of Property 1, Problem 1 has necessarily a solution. In other terms,  $\mathbf{D}_{p,q}$  is triangularizable with permutations.

*Definition 1 (triangularizable with permutations):* A square matrix  $A$  is triangularizable with permutations if there exists a permutation of the rows and a permutation of the columns which result in a triangular matrix.

To solve Problem 1, we resort to a classical problem in matrix analysis: the sparse  $LU$  factorization with minimum filling-in [4, Section 11.1.9].

*Problem 2:* Given a square matrix  $A$  find the permutation matrices  $P$  and  $Q$ , a lower triangular matrix  $L$  and an upper triangular matrix  $U$  such that  $PAQ = LU$  and the number of filling-in (new nonzeros in  $L$  and  $U$  that are not present in  $A$ ) is minimum.

*Remark 2:* If  $A$  is triangularizable with permutations, i.e.  $A = \bar{P}\bar{L}\bar{Q}$  with  $L$  triangular, then the solution to problem 2 is given by  $P = \bar{P}^{-1}$ ,  $Q = \bar{Q}^{-1}$ ,  $L = \bar{L}$  and  $U$  equal to the identity matrix. In this case in fact, we can obtain zero filling-in which by definition is the minimum number achievable. Under this considerations, we can apply Problem 2 to solve Problem 1

*Remark 3:* Deciding if a matrix is triangularizable according to definition 1 is  $\mathcal{NP}$ -complete [5]. Similarly, solving problem 2 is also  $\mathcal{NP}$ -complete [6]<sup>3</sup>. Available numerical tools for finding their solution are not guaranteed to reach the minimum. In solving Problem 2 we will use the unsymmetric-pattern multiFrontal method, as implemented in UMFPACK [9].

Going back to Problem 1, we can take advantage of Property 1 to guarantee that  $D_{p,q}$  is always triangularizable. Therefore, remark 2 applied to  $D_{p,q}$  guarantees that solving Problem 2 will give us also a solution to Problem 1. Given the  $\mathcal{NP}$ -completeness of the underlying problem, we are not guaranteed to find a solution but numerical experiments<sup>4</sup> conducted so far with UMFPACK shows that a solution is always found if  $N_B \leq 100$ . As to this concern, one-hundred can be considered a practical upper-bound for robotic applications.

*Remark 4:* Even though Problem 2 is  $\mathcal{NP}$ -complete, in practice the problem is solved once in a *preliminary optimization* phase and its benefits can be exploited in the *runtime computations*. The preliminary optimization consists in solving Problem 2 for a worst-case sparsity pattern. The runtime computations instead consist in solving (9) for different positions  $q$  and velocities  $\dot{q}$  exploiting the optimized permutations computed previously.

A solution of Problem 2 can be used to compute permutations that allow to solve (9) efficiently. The solution uses the sparsity pattern of  $D_{p,q}(q)$ , i.e. the pattern of non-zero elements in the matrix. However, in practical applications we are interested in a sparsity pattern which somehow represents the sparsity of  $D_{p,q}(q)$  for all possible values of  $q$ . As to this concern, we define the worst case sparsity pattern.

The basic observation is that the robot structure (e.g. number of degrees of freedom, joint types, joint positions, tree structure of the robot) does not change. As a consequence, the underlying sparsity structure of (9), i.e. the non-zero elements in  $D$ , changes only for the state-dependent elements, i.e. those that depend on  $q$  and  $\dot{q}$ . Looking at (4), the only state-dependent blocks are the transformations  ${}^i X_j^*$ ,

<sup>3</sup>Even though [6] is cited several times [7], [8] as a proof of the  $\mathcal{NP}$ -completeness of Problem 2, it has to be observed that it is not clear to the authors of the present paper how to extend the results in [6] to the case of non-symmetric positive definite matrices

<sup>4</sup>Experiments are available here [https://github.com/iron76/bnt\\_time\\_varying/tree/master/experiments/computationalComplexity/RNEA](https://github.com/iron76/bnt_time_varying/tree/master/experiments/computationalComplexity/RNEA).

${}^i X_{\lambda_i}$  which depend on  $q$ . These sub-matrices do have a state-dependent sparsity structure but for the purpose of this paper, we can consider the associated worst-case sparsity structure, i.e. if an element is non-zero for at least one value of  $q$ , then it is considered as a non-zero element in the associated sparsity pattern. For classical joint types (revolute, prismatic, helical, cylindrical, planar, spherical and free-motion) the worst-case sparsity pattern can be easily computed by observing that the only  $q$  dependent elements are either sines or cosines (see [1, Table 4.1, page 79]). These functions are zero only on a countable number of configurations (and therefore on a subset whose Lebesgue measure is zero) which are easy to enumerate.

## V. FORWARD DYNAMICS

The inverse dynamics problem consists in finding  $\ddot{q}_1, \dots, \ddot{q}_{N_B}$  which satisfies (3) given  $\tau_1, \dots, \tau_{N_B}$ ,  $f_1^x, \dots, f_{N_B}^x$ . In [1] the problem is formulated as the computation of the following function:

$$\ddot{q} = \text{FwdD}(\text{model}, q, \dot{q}, \tau, f_1^x, \dots, f_{N_B}^x). \quad (12)$$

Again, in the above equations we grayed out some variables that will not play a role in the following sections, and can be assumed either to be constant (*model*) or measured ( $q, \dot{q}$ ).

### A. Forward dynamics solved with the ABA

The articulated-body algorithm [1, ABA] solves the forward dynamics problem in  $O(N_B)$  computational complexity. The algorithm consists in the following steps. First, the articulated body bias forces  $p_i^A$  and the articulated body inertias  $I_i^A$  are recursively computed iterating with  $i = N_B, \dots, 1$  the following equations:

$$p_i^A = \nu_i - f_i^x + \sum_{j \in \mu_i} {}^i X_j^* \left\{ p_j^A + I_j^a c_j + \right. \quad (12p_i)$$

$$\left. + I_j^A S_j \left( S_j^\top I_j^A S_j \right)^{-1} \left( \tau_j - S_j^\top p_j^A \right) \right\},$$

$$I_j^a = I_j^A - I_j^A S_j \left( S_j^\top I_j^A S_j \right)^{-1} S_j^\top I_j^A \quad (12I_j^a)$$

$$I_i^A = I_i + \sum_{j \in \mu_i} {}^i X_j^* I_j^a {}^j X_i \quad (12I_i^A)$$

Then the following two equations are iterated with  $i = 1, \dots, N_B$  and initial condition  $a_0 = -a_g$ .

$$\ddot{q}_i = \left( S_i^\top I_i^A S_i \right)^{-1} \left\{ \tau_i - S_i^\top \left[ I_i^A \left( {}^i X_{\lambda_i} a_{\lambda_i} + c_i \right) + p_i^A \right] \right\} \quad (13\ddot{q}_i)$$

$$a_i = {}^i X_{\lambda_i} a_{\lambda_i} + S_i \ddot{q}_i + c_i.$$

### B. Forward dynamics solved with matrix inversion

In solving the inverse dynamics, the fact that  $D_{p_{id},q}$  is lower triangular follows from the specific structure of the measurement equations (7). Changing the measurement equations would compromise the lower triangularity of  $D_{p_{id},q}$ . As a consequence, the associated linear system would not have the suitable structure to apply the forward substitution. Within this context, forward dynamics give a useful example. The measured variables for the forward

dynamic case are  $f_i^x$  and  $\tau_i$ . The latter in matrix notation can be expressed as follows:

$$\tau_i = y_{\tau_i}, \quad (14\tau_i)$$

These constraints, will extend the set of constraints  $\mathcal{C}$ . In particular, we should define  $\mathcal{C}_{fd} = \mathcal{C} \cup \{c_{(14\tau_i)}, c_{(7f_i^x)}\}_{i=1}^{N_B}$  and extend the definition of  $D_{c,d}$  and  $b_c$  as follows:

$$D_{c_{(14\tau_i)}, \tau_i} = 1_{n_i}, \quad b_{c_{(14\tau_i)}} = -y_{\tau_i},$$

and  $D_{c,d} = 0$ ,  $b_c = 0$  otherwise. Given two permutations  $p_{fd}$  and  $q$  of the elements in  $\mathcal{C}_{fd}$  and  $\mathcal{D}$ , a solution of the forward dynamics can be computed as the unique solution  $\mathbf{d}_q$  of the following liner system:

$$\mathbf{D}_{p_{fd},q}(q)\mathbf{d}_q + \mathbf{b}_{p_{fd}}(q, \dot{q}) = 0, \quad (15)$$

### C. Forward dynamics solved with forward substitution

Similarly to what observed in the inverse dynamic case, we might try to find permutations for  $p_{fd}$  of the elements in  $\mathcal{C}_{fd}$  and  $q$  of the elements in  $\mathcal{D}$  to obtain a matrix  $\mathbf{D}_{p_{fd},q}$  somehow simple to invert. Specifically, we might think that the articulated-body algorithm [1, the ABA] presented in Section (V-A), could be translated into suitable permutations that transform  $\mathbf{D}_{p_{fd},q}$  into a lower triangular matrix. Unfortunately, the ABA algorithm is instead something more than a permutation as discussed in the following.

*Property 2:* There exist:

- a permutation  $p_{fd}$  of the elements in  $\mathcal{C}_{fd}$ ;
- a permutation  $q$  of the elements in  $\mathcal{D}$ ;
- matrices  $\mathbf{W}_{q_1, q_2}^R$  defined for  $q_1, q_2 \in \mathcal{D}$ ;
- matrices  $\mathbf{W}_{p_1, p_2}^L$  defined for  $p_1, p_2 \in \mathcal{C}_{fd}$ ;

such that:

$$\mathbf{W}_{p_{fd}, p_{fd}}^L \mathbf{D}_{p_{fd}, q} \mathbf{W}_{q, q}^R$$

is lower triangular. These quantities lead to the following matrix reformulation of the ABA algorithm:

$$\mathbf{W}_{p_{fd}, p_{fd}}^L \mathbf{D}_{p_{fd}, q} \mathbf{W}_{q, q}^R \mathbf{d}_q + \mathbf{b}_{p_{fd}}(q, \dot{q}) = 0.$$

The proof of the preposition above is constructive and can be found in Appendix B.

### D. Forward dynamics solved with LU factorization

Again, part of the computational optimizations of the ABA proposed in Section V-A and revisited in Section V-C consists in row and column permutations. In this section we will suggest an algorithm for simplifying the forward dynamics computations by leveraging the intrinsic sparsity of the underlying matrices (15) and pre-computing suitable row and column permutations that will reduce the underlying computational cost. The idea consists in computing the worse case sparsity pattern for  $\mathbf{D}_{p_{fd},q}(q)$  in (15). Solving Problem 2 with this sparsity pattern will give the row-column permutations  $P$  and  $Q$  suitable for solving (15). At run-time regardless of the specific  $q$  the idea is to perform the sparse  $LU$  factorization on  $P\mathbf{D}_{p_{fd},q}(q)Q$  followed by a backward substitution on  $U$  and a forward substitution on  $L$ . Figure 1 shows the computational cost of this solution against the

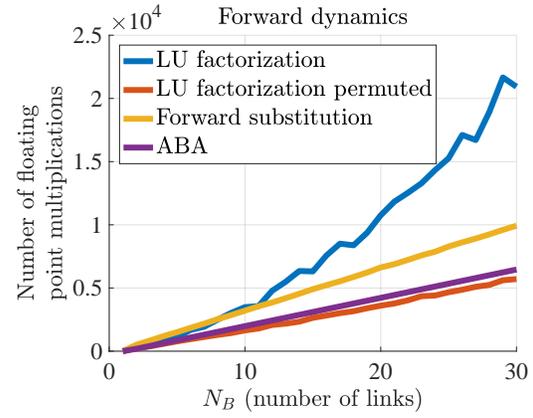


Fig. 1. Comparison of the different proposed algorithms for solving the forward dynamics of a serial chain with  $N_B$  links.

algorithms proposed in Section V-A (ABA) and Section V-C (matrix reformulation of the ABA). The computational cost of ABA is the one reported in [1, Page 202] and corresponds to the algorithmic solution proposed in [10]. The other computational costs are numerically computed with software which is available open source<sup>5</sup>.

## VI. DYNAMIC EQUATIONS AND LU-FACTORIZATION

In the previous sections, we have seen how the RNEA and the ABA are computationally efficient solutions of the inverse and forward dynamics problems. The reduction of the computational costs is obtained by suitable permutations of the matrices  $\{D_{c,d}\}_{c \in \mathcal{C}_{id}, d \in \mathcal{D}}$  and  $\{D_{c,d}\}_{c \in \mathcal{C}_{fd}, d \in \mathcal{D}}$ . In this section we consider the problem of finding similar permutations for a wider class of problems. Inspired by the matrix representations (9) and (15) of the inverse and forward dynamics respectively, we consider a generic estimation problem consisting in computing the solution  $\mathbf{d}$  of the following linear system:

$$\underbrace{\begin{bmatrix} \mathbf{D}_{p,q}(q) \\ \mathbf{D}_Y(q) \end{bmatrix}}_{\triangleq \mathbf{D}(q)} \mathbf{d} + \underbrace{\begin{bmatrix} \mathbf{b}_p(q, \dot{q}) \\ \mathbf{b}_Y(q, \dot{q}) \end{bmatrix}}_{\triangleq \mathbf{b}(q, \dot{q})} = 0, \quad (16)$$

obtained by combining (5) with a generic measurement equation  $\mathbf{D}_Y(q)\mathbf{d} + \mathbf{b}_Y(q, \dot{q}) = 0$ . In Section IV-C and Section V-C respectively, suitable permutations led to a lower triangular structure for the matrix  $\mathbf{D}$ . The underlying linear system could then be solved with a forward substitution. However, the proposed solutions rely on the specific structure of the problem and therefore extending them to solve (16) is non trivial. The algorithm proposed in Section IV-D and Section V-D seems more suitable to simplify the computational complexity of solving (16) in its generic form (i.e. beyond the inverse and forward dynamics cases).

<sup>5</sup>The software for these computations are available here: [https://github.com/iron76/bnt\\_time\\_varying/tree/master/experiments/computationalComplexity/ABA/serial/compare](https://github.com/iron76/bnt_time_varying/tree/master/experiments/computationalComplexity/ABA/serial/compare).

As an example, we consider a relevant case study: the inverse dynamic computation of a humanoid robot standing on two feet. The specificity of this problem is that the robot has two external wrenches applied at the feet. Differently from the inverse dynamic computations presented in Section IV-A, we assume that these external wrenches are unknown. Using the notation used in previous sections, these external wrenches can be denoted  $f_1^x$  and  $f_{N_B}^x$  by labelling with 1 and  $N_B$  the right and the left foot respectively. We therefore consider the following problem:

$$\tau = \text{InvD}(\text{model}, q, \dot{q}, \ddot{q}, f_2^x, \dots, f_{N_B-1}^x), \quad (17)$$

where we explicitly indicated that  $f_1^x$  and  $f_{N_B}^x$  are unknown even if they act on the system. The considered mechanical system is a free-floating articulated rigid body subject to constraints. Unfortunately solving (17) is ill-posed, i.e. given  $q, \dot{q}, \ddot{q}, f_2^x, \dots, f_{N_B-1}^x$  there exist multiple  $\tau, f_1^x, f_{N_B}^x$  satisfying (3). To estimate  $\tau$  we could try using additional measurements. We hereafter consider quite a common sensor distribution, nominally four load cells located on each foot<sup>6</sup>. These sensors are available in the NAO robot [11], in the QRIO robot [12] and the Atlas [13]. These sensors correspond to a three-axes force-torque sensor, i.e. they measure the net force orthogonal to the plane of the each foot and the projection of the torque on the same plane. As it is often the case, if we choose the left and right feet reference frames with the  $x$  and  $y$  axis aligned to the foot plane we have:

$$f_1^x = Y y_{lf} + H h_{lf}, \quad f_{N_B}^x = Y y_{rf} + H h_{rf}, \quad (18a)$$

$$Y \triangleq \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad H \triangleq \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad (18b)$$

with  $y_{lf}, y_{rf} \in \mathbb{R}^3$  being the measured three-axes force-torque and  $h_{lf}, h_{rf} \in \mathbb{R}^3$  being the non-directly measured components of the contact wrench  $f_1^x$ . We are now considering the following problem:

$$\tau = \text{InvD}(\text{model}, q, \dot{q}, \ddot{q}, f_2^x, \dots, f_{N_B-1}^x, y_{lf}, y_{rf}). \quad (19)$$

To understand if this problem is well-posed, it is convenient to resort to a specific form of equation (3). This formulation is detailed in [14, eq. (41)] and it combines the floating-base dynamics with the joint dynamics. We consider here a simplified version obtained by choosing the left foot as the base frame and choosing  $q$  a local parametrization of the robot pose (composed by the free-floating six-dimensional configuration and the joint  $n$ -dimensional configuration). We have:

<sup>6</sup>We first prove that these sensors are not yet sufficient to solve the inverse dynamics. Then we propose a different set of sensors which results in a well posed inverse dynamics problem.

$$M(q)\ddot{q} + h(\dot{q}, q) = \begin{bmatrix} \tau \\ 0 \end{bmatrix} + \begin{bmatrix} {}^0 J_1^\top \\ 16 \end{bmatrix} f_1^x + \sum_{j=2}^{N_B} \begin{bmatrix} {}^0 J_j^\top \\ 0 \\ X_j^* \end{bmatrix} f_j^x. \quad (20)$$

Using (18) in (20) and grouping the unobservable/observable quantities as in problem (19), we obtain:

$$Y(q, \dot{q}, \ddot{q}, f_2^x, \dots, f_{N_B-1}^x, y_{lf}, y_{rf}) = \begin{bmatrix} \tau \\ 0 \end{bmatrix} + \begin{bmatrix} {}^0 J_1^\top H h_{lf} \\ H h_{lf} \end{bmatrix} + \begin{bmatrix} {}^0 J_{N_B}^\top H h_{rf} \\ 0 \\ X_{N_B}^* H h_{rf} \end{bmatrix}. \quad (21)$$

Understanding if the inverse dynamic problem with these measurements is well-posed boils down to understanding if the following matrix is invertible:

$$\begin{bmatrix} 1_n & {}^0 J_1^\top H & {}^0 J_{N_B}^\top H \\ 0_{6 \times 6} & H & {}^0 X_{N_B}^* H \end{bmatrix}. \quad (22)$$

Given the upper triangular form of this matrix, we are left with proving the invertibility of  $[H, {}^0 X_{N_B}^* H]$ . We are interested in guaranteeing the invertibility of this matrix for any  ${}^0 X_{N_B}^*$ , i.e. regardless of the feet relative pose.

*Property 3:* Let  $H = [H_f^\top H_\mu^\top]^\top$  with  $H_f \in \mathbb{R}^{3 \times 3}$ ,  $H_\mu \in \mathbb{R}^{3 \times 3}$ . If  $H_f$  is singular, then there always exists  ${}^0 X_{N_B}^*$  which makes  $[H, {}^0 X_{N_B}^* H]$  singular.

*Proof:* Using the structure of  ${}^0 X_{N_B}^*$  and  $H$  we obtain:

$$[H \quad {}^0 X_{N_B}^* H] \stackrel{R=13}{=} \begin{bmatrix} H_f & R H_f \\ H_\mu & p \times R H_f + R H_\mu \end{bmatrix} \begin{bmatrix} 1_3 & 1_3 \\ 0_{3 \times 3} & 1_3 \end{bmatrix},$$

and the results follows by observing that the multiplicand matrix on the left is singular (the first three lines are linearly dependent) and by using the fact that the rank of a product is always less or equal the rank of the multiplied matrices. ■

*Remark 5:* Using Property 3 with definitions (18b) we can conclude that (19) is an ill-posed problem. In other terms, the inverse dynamics problem cannot be solved in the case of a humanoid robot standing on the two feet with four load cells on each foot.

Additional assumptions are needed to compute the inverse dynamics. A first realistic assumption in certain applications is to assume that the robot is standing in a very slippery surface and therefore the tangential forces along the  $x$  and  $y$  axis are negligible. In this case:

$$f_1^x = Y y_{lf} + H h_{lf}, \quad f_{N_B}^x = Y y_{rf} + H h_{rf}, \quad (23a)$$

$$Y \triangleq \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad H \triangleq \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \quad (23b)$$

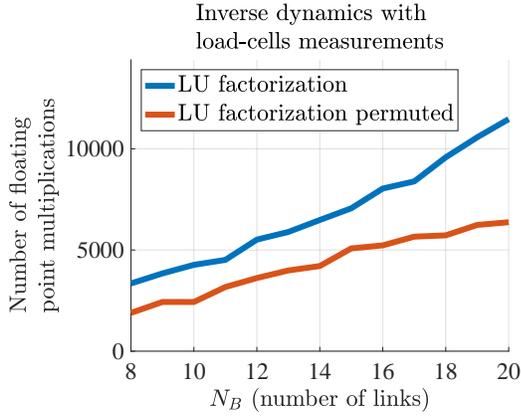


Fig. 2. Comparison of the different proposed algorithms for solving the inverse dynamics (i.e. joint torque estimation) of a serial chain with  $N_B$  links. Considered measurements are load-cells at extremal links.

with  $y_{lf}, y_{rf} \in \mathbb{R}^3$  being the measured contact forces and the measured torques on the  $x$ - $y$  plane and  $h_{lf}, h_{rf} \in \mathbb{R}$  being the torques on the  $z$  axis. In this case it was numerically observed<sup>7</sup> that the associated inverse dynamics problem is solvable. Fig. 2 shows a comparison of the number of floating point operations necessary to solve this specific inverse dynamics problem with or without a sparse  $LU$  factorization. Remarkably this case cannot be solved with classical algorithms (e.g. hybrid dynamics [1]).

## APPENDIX

### A. The articulated body equation of motion

The idea consists in recursively computing ( $i = N_B, \dots, 1$ ) the quantities  $p_i^A$  (articulated body bias forces) and  $I_i^A$  (articulated body inertias) which satisfy the articulated body equation of motion:

$$f_i = I_i^A a_i + p_i^A. \quad (23f_i^{ab})$$

For defining these quantities, let's start by considering  $i$  such that  $\mu_i = \emptyset$ . In this case (3f<sub>i</sub>) gives  $f_i = I_i a_i + \nu_i - f_i^x$  and therefore  $I_i^A = I_i$  and  $p_i^A = \nu_i - f_i^x$ . Recursively, let's assume that  $I_j^A$  and  $p_j^A$  have been defined for every  $j \in \mu_i$  (this time non-empty) and let's find suitable expressions for  $I_i^A$  and  $p_i^A$ . This is achieved by a five step procedure.

*Step 1 - S<sub>1</sub>*. Replace (23f<sub>i</sub><sup>ab</sup>) in (3):

$$a_i = {}^i X_{\lambda_i} a_{\lambda_i} + S_i \ddot{q}_i + c_i, \quad (24a_i)$$

$$\tau_i = S_i^\top (I_i^A a_i + p_i^A) \quad (24\tau_i)$$

$$f_i = I_i a_i + \nu_i - f_i^x + \sum_{j \in \mu_i} {}^i X_j^* (I_j^A a_j + p_j^A). \quad (24f_i)$$

The latter is almost the equation we need for the recursive definition of  $I_i^A$  and  $p_i^A$  if only we could write  $a_j$  as a function of  $a_i$ . This is achieved with the following steps.

*Step 2 - S<sub>2</sub>*. Substitute (24a<sub>i</sub>) in (24τ<sub>i</sub>):

$$\tau_j = S_j^\top [I_j^A ({}^j X_{\lambda_j} a_{\lambda_j} + S_j \ddot{q}_j + c_j) + p_j^A]. \quad (25)$$

*Step 3 - S<sub>3</sub>*. Multiply the previous equation by the inverse of  $S_j^\top I_j^A S_j$  to obtain (13q̇<sub>i</sub>) for q̇<sub>j</sub>:

$$\ddot{q}_j = \begin{aligned} & (S_j^\top I_j^A S_j)^{-1} \{ \tau_j - S_j^\top [I_j^A ({}^j X_{\lambda_j} a_{\lambda_j} + c_j) + p_j^A] \} \\ & \stackrel{j \in \mu_i}{=} (S_j^\top I_j^A S_j)^{-1} \{ \tau_j - S_j^\top [I_j^A ({}^j X_i a_i + c_j) + p_j^A] \}, \end{aligned}$$

*Step 4 - S<sub>4</sub>*. Substitute the last equation in (24a<sub>i</sub>):

$$a_i = {}^i X_{\lambda_i} a_{\lambda_i} + c_i + S_i (S_i^\top I_i^A S_i)^{-1} \{ \tau_i - S_i^\top [I_i^A ({}^i X_{\lambda_i} a_{\lambda_i} + c_i) + p_i^A] \},$$

which evaluated with the substitution  $i \rightarrow j \in \mu_i$  leads to:

$$a_j = {}^j X_i a_i + c_j + S_j (S_j^\top I_j^A S_j)^{-1} \{ \tau_j - S_j^\top [I_j^A ({}^j X_i a_i + c_j) + p_j^A] \}. \quad (26)$$

*Step 5 - S<sub>5</sub>*. Substitute  $a_j$  in (24f<sub>i</sub>) with its expression in (26):

$$\begin{aligned} f_i = & \left( I_i + {}^i X_j^* I_j^A {}^j X_i \right) a_i + \nu_i - f_i^x + \\ & + \sum_{j \in \mu_i} {}^i X_j^* I_j^A S_j (S_j^\top I_j^A S_j)^{-1} \tau_j \\ & + {}^i X_j^* \left[ 1 - I_j^A S_j (S_j^\top I_j^A S_j)^{-1} S_j^\top \right] p_j^A \\ & - \left[ {}^i X_j^* I_j^A S_j (S_j^\top I_j^A S_j)^{-1} S_j^\top I_j^A {}^j X_i \right] a_i \\ & + {}^i X_j^* \left[ I_j^A - I_j^A S_j (S_j^\top I_j^A S_j)^{-1} S_j^\top I_j^A \right] c_j. \end{aligned} \quad (27)$$

and enforcing  $f_i = I_i^A a_i + p_i^A$  leads to the definitions in (12p<sub>i</sub>), (12I<sub>i</sub><sup>a</sup>) and (12I<sub>i</sub><sup>A</sup>).

### B. Proof of Property 2

We hereafter assume that the articulated body inertias  $I_i^A$  ( $i = 1, \dots, N_B$ ) have been computed. The idea is to follow the steps presented in Section A to compute the sub-blocks of the matrices  $W^L$  and  $W^R$ . The latter is defined as a matrix equivalent of *Step 1*. The former as  $W^L = W^{L,4} W^{L,3} W^{L,2} W^{L,1}$  with  $W^{L,1}$  representing *Step 2*,  $W^{L,2}$  representing *Step 3*,  $W^{L,3}$  representing *Step 4* and  $W^{L,4}$  representing *Step 5*.

*Step 1 - S<sub>1</sub>*. We compute a matrix  $W^R$  which multiplied by  $d_q$  replaces  $f_i$  with  $p_i^A = f_i - I_i^A a_i$ . As usual we define  $W^R$  by its blocks  $W_{q_1, q_2}^R$  with  $q_1, q_2 \in \mathcal{D}$ . We have:

$$W_{f_i, a_i}^R = I_i^A, \quad W_{q, q}^R = 1 \quad \forall q \in \mathcal{D}, \quad (28)$$

and  $W_{q_1, q_2}^R = 0$  otherwise.

*Remark 6:* Given two permutations  $p$  and  $q$  of the elements in  $\mathcal{C}$  and  $\mathcal{D}$ , if  $D_{p, q}$  represents (3),  $D_{p, q} W_{q, q}^R$  represents (24). Therefore, in the following  $\{c_{(3a_i)}, c_{(3\tau_i)}, c_{(3f_i)}\}_{i=1}^{N_B}$  can be read as  $\{c_{(24a_i)}, c_{(24\tau_i)}, c_{(24f_i)}\}_{i=1}^{N_B}$ .

<sup>7</sup>[https://github.com/iron76/bnt\\_time\\_varying/tree/master/experiments/computationalComplexity/SIE](https://github.com/iron76/bnt_time_varying/tree/master/experiments/computationalComplexity/SIE)

*Step 2 - S<sub>2</sub>*. We define  $W^{L,1}$  which left multiplies  $D_{p,q}W_{q,q}^R$  to substitute  $(24a_i)$  in  $(24\tau_i)$ . The exceptions to  $W_{p_1,p_2}^{L,1} = 0$  are:

$$W_{c(3\tau_i),c(3a_i)}^{L,1} = S_i^\top I_i^A, \quad W_{p,p}^{L,1} = 1 \quad \forall p \in \mathcal{C}_{fd}, \quad (29)$$

*Step 3 - S<sub>3</sub>*. We define  $W^{L,2}$  which left multiplies  $W_{p,p}^{L,1}D_{p,q}W_{q,q}^R$  to multiply (25) by the inverse of  $S_j^\top I_j^A S_j$ . The exceptions to  $W_{p_1,p_2}^{L,2} = 0$  are:

$$W_{c(3\tau_i),c(3\tau_i)}^{L,2} = (S_j^\top I_j^A S_j)^{-1}, \quad W_{p,p}^{L,2} = 1 \quad \forall p \in \mathcal{C}_{fd}, q \neq \tau_i \quad (30)$$

*Step 4 - S<sub>4</sub>*. We define  $W^{L,3}$  which left multiplies  $W_{p,p}^{L,2}W_{p,p}^{L,1}D_{p,q}W_{q,q}^R$  to replace the occurrences of  $\ddot{q}_j$  in  $(24a_i)$  with their expression obtained in the previous step. The exceptions to  $W_{p_1,p_2}^{L,3} = 0$  are:

$$W_{c(3a_i),c(3\tau_i)}^{L,3} = -S_i, \quad W_{p,p}^{L,3} = 1 \quad \forall p \in \mathcal{C}_{fd}. \quad (31)$$

*Step 5 - S<sub>5</sub>*. We define  $W^{L,4}$  which left multiplies  $W_{p,p}^{L,3}W_{p,p}^{L,2}W_{p,p}^{L,1}D_{p,q}W_{q,q}^R$  to replace the occurrences of  $a_j$  in  $(24f_i)$  with their expression obtained in the previous step. The exceptions to  $W_{p_1,p_2}^{L,4} = 0$  are:

$$W_{c(3f_i),c(3a_j)}^{L,4} = {}^i X_j^* I_j^A, \quad W_{p,p}^{L,4} = 1 \quad \forall p \in \mathcal{C}_{fd}. \quad (32)$$

We are left with the definition of the permutations. Let's first consider the  $q$  permutation of the elements in  $\mathcal{D}$ . We choose:

$$q = [f_1^x, \tau_1, \dots, f_{N_B}^x, \tau_{N_B}, f_{N_B}, \dots, f_1, a_1, \dots, a_{N_B}, \ddot{q}_1, \dots, \ddot{q}_{N_B}]. \quad (33)$$

Let's also choose a permutation  $p_{fd}$  of the elements in  $\mathcal{C}_{fd}$ :

$$p_{fd} = \left[ c(7f_1^x), c(3\tau_{N_B}) \dots c(7f_{N_B}^x), c(3\tau_1), c(3f_{N_B}) \dots c(3f_1), c(3a_1), \dots, c(3a_{N_B}), c(14\ddot{q}_1), \dots, c(14\ddot{q}_{N_B}) \right]. \quad (34)$$

- [1] R. Featherstone, *Rigid Body Dynamics Algorithms*. Springer, 2008.
- [2] C. Balafoutis and R. Patel, *Dynamic Analysis of Robot Manipulators: A Cartesian Tensor Approach*, ser. The Springer International Series in Engineering and Computer Science. Springer US, 1991. [Online]. Available: <https://books.google.it/books?id=7BcPyUjmlpUC>
- [3] D. Orin and M. Walker, "Efficient dynamic computer simulation of robotic mechanisms," *ASME Journal of Dynamic Systems, Measurement and Control*, 1982.
- [4] G. Golub and C. Van Loan, "Matrix computations 4th ed," 2013.
- [5] G. Fertin, I. Rusu, and S. Vialette, "Obtaining a Triangular Matrix by Independent Row-Column Permutations," in *26th International Symposium on Algorithms and Computation*, Nagoya, France, Dec. 2015. [Online]. Available: <https://hal.archives-ouvertes.fr/hal-01189621>
- [6] M. Yannakakis, "Computing the minimum fill-in is np-complete," *SIAM Journal on Algebraic Discrete Methods*, vol. 2, no. 1, pp. 77–79, 1981.
- [7] J. Dongarra, V. Eijkhout, and P. Luszczek, "Recursive approach in sparse matrix lu factorization," *Sci. Program.*, vol. 9, no. 1, pp. 51–60, Jan. 2001. [Online]. Available: <http://dx.doi.org/10.1155/2001/569670>
- [8] L. Grigori, E. G. Boman, S. Donfack, and T. A. Davis, "Hypergraph-based unsymmetric nested dissection ordering for sparse lu factorization," *SIAM J. Sci. Comput.*, vol. 32, no. 6, pp. 3426–3446, Nov. 2010. [Online]. Available: <http://dx.doi.org/10.1137/080720395>
- [9] T. A. Davis, "Algorithm 832: Umfpack v4.3—an unsymmetric-pattern multifrontal method," *ACM Trans. Math. Softw.*, vol. 30, no. 2, pp. 196–199, June 2004. [Online]. Available: <http://doi.acm.org/10.1145/992200.992206>
- [10] H. Brandl, R. Johanni, and M. Otter, "A very efficient algorithm for the simulation of robots and similar multibody systems without inversion of the mass matrix." in *IFAC/IFIP/IMACS Symposium on Theory of Robots*, 1986, pp. 95–100.
- [11] D. Gouaillier, V. Hugel, P. Blazevic, C. Kilner, J. Monceaux, P. Lafourcade, B. Marnier, J. Serre, and B. Maisonnier, "Mechatronic design of nao humanoid," in *2009 IEEE International Conference on Robotics and Automation*, May 2009, pp. 769–774.
- [12] T. Ishida, "Development of a small biped entertainment robot qrio," in *Micro-Nanomechatronics and Human Science, 2004 and The Fourth Symposium Micro-Nanomechatronics for Information-Based Society, 2004.*, Oct 2004, pp. 23–28.
- [13] S. Kuindersma, R. Deits, M. Fallon, A. Valenzuela, H. Dai, F. Permenter, T. Koolen, P. Marion, and R. Tedrake, "Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot," *Autonomous Robots*, vol. 40, no. 3, pp. 429–455, 2016. [Online]. Available: <http://dx.doi.org/10.1007/s10514-015-9479-3>
- [14] S. Traversaro, D. Pucci, and F. Nori, "A unified view of the equations of motion used for control design of humanoid robots," *Submitted to Multibody System Dynamics - Springer*, 2017. [Online]. Available: <https://traversaro.github.io/preprints/changebase.pdf>