Robotics: Science and Systems 2021
Held Virtually, July 12–16, 2021

Generalized Comprehensive Motion Theory for High-Order Differential Dynamics

Vincent Samy, Ko Ayusawa and Eiichi Yoshida

***Abstract*—We address the problem of calculating complex Jacobian matrices that can arise from optimization problems. An example is the inverse optimal control in human motion analysis which has a cost function that depends on the second order time-derivative of torque $\ddot{\tau}$. Thus, its gradient decomposed to, among other, the Jacobian $\delta\ddot{\tau}/\delta q$. We propose a new concept called N-order Comprehensive Motion Transformation Matrix (N-CMTM) to provide an exact analytical solution of several Jacobians. The computational complexity of the basic Jacobian and its N-order time-derivatives computed from the N-CMTM is experimentally shown to be linear to the number of joints N_j. The N-CMTM is based on well-known spatial algebra which makes it available for any type of robots. Moreover, it can be used along classical algorithms. The computational complexity of the construction of the N-CMTM itself is experimentally shown to be N^2.**

I. Introduction

Robotics researchers have been benefiting from basic computation such as Forward Kinematics (FK), Forward Dynamics (FD), Inverse Dynamics (ID) and Inverse Kinematics (IK) for various applications. Not only they are well-defined but they also have been improved in terms of performances. The work of Luh et al. [16] was one of the first to present a resolution scheme to compute inverse dynamics. They then showed its efficiency in real-time [17]. Featherstone [10] presented a revised version based on spatial algebra which can handle the translation and rotation of rigid-body coordinate simultaneously, and is less sensitive to joint type. Walker and Orin [32] presented the first Composite Rigid Body Algorithm (CRBA), and later Featherstone [10] proposed a recursive propagation method called Articulated-Body Algorithm (ABA) which is also based on spatial algebra. Park et al. [24] proposed a generalized method for ID algorithm using Lie group and Lie algebra and Sohl and Bobrow [30] presented a first order differentiation of it. In parallel, methods that exploit the sparse-matrix formulation of the FD has been developed as in [23]. Nowadays, FD is the central part of dynamic simulators [29] [22] while FK and ID are widely used for almost any robotic branch [18] such as motion planning [6] [33], control [26], mechanical design [14], etc. They are also fundamental tools for human/humanoid motion analysis as shown in [27] [7] [25] [20].

Nevertheless, classical algorithms only consider acceleration, torque and force. Going beyond, Guarino Lo Bianco [13] presented an ID algorithm that computes first derivative of generalized forces using the joint jerks with Denavit-Hartenberg parameters. Buondonno and De Luca [4] later extended it to the second derivative along the joint snap (fourth derivative of position). Flash and Hogan [11] presented a trajectory analysis of human planar motion that minimizes the jerk. Use of higher-order differential information has also been studied for flying robotics. For instance, Mellinger and Kumar [19] used the snap for a quad rotor indoor trajectory generation, and Eager et al. [8] demonstrated the interest of high-order derivatives for roller coaster. Furthermore, Carpentier and Mansard [5] provided an analytical derivative solution for spatial algebra of the ID and FD up to the first-order derivative. Finally, it is also important to note that recent pseudo-symbolic techniques such as Automatic Differentiation (AD) has the potential to compute time-derivatives [3].

In an optimization scheme where a minimization of the acceleration/jerk or torque is desired, their gradient, and eventually the Hessian, is needed. This need leads to the computation of high-order Jacobians (i.e. Jacobians of high order time derivative of physical quantity). The study on human motion analysis based on inverse optimal control by Lin et al. [15] is an example of the need for such a solution. Although numerical computation is commonly employed for those objectives, analytical solution would provide better precision and performance. Human motion imitation [31] is another example of such optimization problem. Recently, Fu et al. [12] have provided a high-order Jacobian analytical solution based on Lie algebra but is limited to fixed manipulators and to the basic Jacobian.

In this paper we present a new mathematical representation that provides an analytical solution of any high-order Jacobian matrix without having to manually perform derivation. The new representation is a generalization of the prior work [2][1], that demonstrated the capability of the Comprehensive Motion Transformation Matrix (CMTM) by performing a complex dynamic motion optimization. We call this new concept of N-CMTM where N stands for the derivative order.

Section II presents the motivation behind N-CMTM and proposes its first glance. After providing a set of notation and convention used in this paper in Section III, we introduce N-CMTM in Section IV. Section V investigates kinematics and dynamics under the N-CMTM to derive comprehensive equations and Jacobians in Section VI. The theory is numerically validated against well-known baseline methods in Section VII.

II. Motivation

Let us consider the following optimization scheme

$$\begin{aligned} \min_{x} \quad & c(Y(x)) \\ \text{subject to} \quad & g(Y(x)) \leq 0 \end{aligned} \tag{1}$$

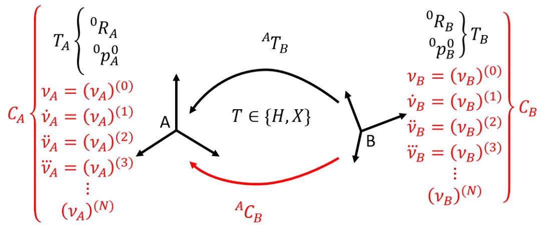

T_A
${}^0R_A$
${}^0p_A^0$
${}^AT_B$
${}^0R_B$
${}^0p_B^0$
T_B
C_A
$\nu_A = (\nu_A)^{(0)}$
$\dot{\nu}_A = (\nu_A)^{(1)}$
$\ddot{\nu}_A = (\nu_A)^{(2)}$
$\dddot{\nu}_A = (\nu_A)^{(3)}$
$(\nu_A)^{(N)}$
A
$T \in \{H, X\}$
B
$\nu_B = (\nu_B)^{(0)}$
$\dot{\nu}_B = (\nu_B)^{(1)}$
$\ddot{\nu}_B = (\nu_B)^{(2)}$
$\dddot{\nu}_B = (\nu_B)^{(3)}$
$(\nu_B)^{(N)}$
C_B
${}^AC_B$

Fig. 1: Transformation matrices from frame B to a frame A. The N-CMTM not only embeds the position and the rotation but also the spatial velocity, spatial acceleration and all the derivatives above.

where the cost function c and the constraint function g depend of physical quantities Y (such as velocities ν, forces f, or more complex quantities like $\ddot{\tau}$ the second order time-derivative of torque). Often $x = \left(q^{\mathsf{T}}, \dot{q}^{\mathsf{T}}, \ddot{q}^{\mathsf{T}}\right)^{\mathsf{T}}$ *resp.* the generalized coordinates, velocity and acceleration vectors also depend on other quantities like torques. This type of optimization is common in many robotics field like motion planning [28], control [9], etc. Optimization with high-order derivative of physical quantities can be found in human motion analysis and Inverse Optimal Control (IOC) [15]. To solve this problem we need the gradient of the functions c and g, namely

$$\frac{\partial h(Y)}{\partial x} = \frac{\partial h(Y)}{\partial Y}\frac{\partial Y}{\partial x} \quad \text{with } h = \{c, g\}. \tag{2}$$

The first part $\partial h(Y)/\partial Y$ depends of the definition of the function h which is chosen to facilitate its gradient calculation. The second part $\partial Y/\partial \chi$ has various meanings. For example, when $Y = \nu$ then $\partial \nu/\partial q$ is the basic Jacobian matrix, but it also has other meanings. A non-exhaustive list can be:

$$\frac{\partial \ddot{\nu}(q, \dot{q}, \ddot{q}, \dddot{q}, \ddddot{q})}{\partial \dot{q}} \quad \Bigg| \quad \frac{\partial \dot{f}(q, \dot{q}, \ddot{q}, \dddot{q})}{\partial \ddot{q}} \quad \Bigg| \quad \frac{\partial \ddot{\tau}(q, \dot{q}, \ddot{q}, \dddot{q}, \ddddot{q})}{\partial q} \tag{3}$$

These examples are much harder to solve without resorting to numerical differentiation. Unfortunately, numerical differentiation is computationally heavy, and the higher the derivative becomes the harder it is to get a good approximation. This is where N-CMTM can be introduced as a powerful tool through an **exact analytical solution**.

Before going deep into the theory, it is convenient to give a simple insight of the N-CMTM. It can be viewed as an extended homogeneous/spatial transformation matrix in the sense that it provides not only rotation/position coordinates transformation but also the velocity transformation and all of its derivatives as in Fig. 1. So it is an all-in-one transformation matrix thus can unify FK, ID and FD.

III. Notation & Background

Let A, B and C be three arbitrary coordinate frames. The translation from A to B in C coordinates is written as ${}^C r_B^A$. The rotation matrix that transforms 3D vector coordinates from B to A is written as ${}^A R_B$. A spatial vector from A to B in C coordinates is written as ${}^C \nu_B^A$. Compared to Featherstone's spatial vector, ours has a wider meaning and we have ${}^C\nu_B^A = {}^C\nu_B - {}^C\nu_A$, so it also represents the relative velocity between A and B in C coordinates. Furthermore, if A is the parent frame of B then ${}^B\nu_B^A$ is the joint velocity in the successor frame (noted v_{J} in Featherstone). Generally, we simplify the writing and call a spatial joint vector $\nu_B^A = {}^B\nu_B^A$, while a spatial link vector is denoted $\nu_B = {}^B\nu_B^0$ where 0 represents the world frame. The homogeneous matrix H and the spatial transformation X from frame B to A are given by

$$\left\{\begin{array}{lcl} {}^AH_B & = & \begin{pmatrix} {}^AR_B & {}^Ar_B^A \\ 0_{1\times3} & 1 \end{pmatrix} \\ & & \\ {}^AX_B & = & \begin{pmatrix} {}^AR_B & 0_{3\times3} \\ \left[{}^Ar_B^A\times_3\right]{}^AR_B & {}^AR_B \end{pmatrix} \end{array}\right. \tag{4}$$

and we define T to represent either ($T \in \{H, X\}$). If the transformation is from an arbitrary frame A to the world frame 0, then it is noted T_A. It should be noted that, in Featherstone's notation, the lower-left element of ${}^AX_B$ can be written as: $\left[{}^Ar_B^A\times_3\right]{}^AR_B = -{}^AR_B\left[{}^Br_A^B\times_3\right]$. The spatial transformation X admits a dual representation, noted $\bar{X}$

$$ {}^A\bar{X}_B = {}^AX_B^{-\mathsf{T}} = \begin{pmatrix} {}^AR_B & \left[{}^Ar_B^A\times_3\right]{}^AR_B \\ 0_{3\times3} & {}^AR_B \end{pmatrix}. \tag{5}$$

We write the operator $[\cdot\times_4]$ (*resp.* $[\cdot\times_6]$) that generates the differential operator of the homogeneous (*resp.* spatial) transformation matrix. More explicitly with $\nu = \left[\omega^{\mathsf{T}} v^{\mathsf{T}}\right]^{\mathsf{T}} \in \mathbb{R}^6$

$$[\nu\times_4] = \begin{pmatrix} [\omega\times_3] & v \\ 0_{1\times3} & 0 \end{pmatrix} \quad [\nu\times_6] = \begin{pmatrix} [\omega\times_3] & 0_{3\times3} \\ [v\times_3] & [\omega\times_3] \end{pmatrix} \tag{6}$$

where $\omega \in \mathbb{R}^3$ represents the angular velocity and $v \in \mathbb{R}^3$ the linear velocity. The operator $[\cdot\times_6]$ also admits a dual representation

$$[\nu\bar{\times}_6] = -[\nu\times_6]^{\mathsf{T}} = \begin{pmatrix} [\omega\times_3] & [v\times_3] \\ 0_{3\times3} & [\omega\times_3] \end{pmatrix} \tag{7}$$

The operator $[\omega\times_3]$ with $\omega \in \mathbb{R}^3$ generates the differential operator of the rotation matrix which is called skew-symmetric matrix. We note $\cdot^{\vee}$ the reverse operator of $[\cdot\times]$ so that:

$$[\nu\times]^{\vee} = \nu.$$

Let us now consider the time-derivative of a spatial transformation in the relative frame. Let A, B, e and f be four arbitrary frames. Both frame A and B admits respectively a spatial link motion ν_A and ν_B, We note $\dot{\nu}$ the time-derivative of the spatial motion vector in local coordinates and $\mathring{\nu}$ the component-wise time-derivative. As in [10] (Eq. (2.45)), the time-derivative of the spatial transformation matrix is

$$\begin{aligned} \frac{d}{dt}\left({}^AX_B\right){}^B\nu_f^e &= {}^A\mathring{\nu}_f^e - {}^AX_B{}^B\mathring{\nu}_f^e \\ &= \left({}^AX_B\left[\nu_B\times_6\right] - \left[\nu_A\times_6\right]{}^AX_B\right){}^B\nu_f^e \\ &= {}^AX_B\left[\nu_B^A\times_6\right]{}^B\nu_f^e \end{aligned} \tag{8}$$

Although we will not prove it here, this formula is also valid for the homogeneous form, and the generic equation is

$$ {}^A\dot{T}_B = {}^AT_B\left[\nu_B^A\times_k\right], \quad k \in \{4, 6\} \tag{9}$$

and finally here are some properties of the cross operator

$$\left\{ \begin{array}{rcl} \dot{A} & = & A\left[\gamma\times_k\right] \\ \left[A\gamma\times_k\right]^n & = & A\left[\gamma\times_k\right]^n A^{-1} \end{array} \right. , \; k \in \{3, 4, 6\} \tag{10}$$

with $A \in \{R, H, X\}$ and $\gamma \in \{\omega, \nu, \nu\}$. To avoid confusion, Featherstone's equivalent operator is $(A\gamma)\times = [A\gamma\times]$.

IV. N-CMTM

Ayusawa and Yoshida [2] presented CMTM that transforms motions up to the acceleration. As the CMTM comes from Lie theory, they mainly worked with the tangent space of SE(3). Here, we extend the CMTM notion to any order of derivatives and give its generalized meaning. Furthermore, for simplification, we will only consider time-derivative of physical quantities and drop infinitesimal displacement. This choice is discussed in Section VI-B4).

Let us define a spatial velocity ν and the concatenation ζ of its N time-derivatives:

$$\zeta \triangleq \left[\nu^\mathsf{T} \; \dot{\nu}^\mathsf{T} \; \cdots \; \nu^{(N)^\mathsf{T}}\right]^\mathsf{T}. \tag{11}$$

Let us consider the equation $z = ay$ and compute its N-th derivative

$$z^{(N)} = \sum_{k=0}^{N} \binom{N}{k} a^{(N-k)} y^{(k)} \tag{12}$$

where $\binom{\cdot}{\cdot}$ provides binomial coefficients, and let us define

$$\hat{g}^{(p)} = \frac{g^{(p)}}{p!}. \tag{13}$$

By substitution, $\hat{a}^{(0)} = T$, $\hat{y}^{(0)} = \left[\zeta_{\{0\}}\times_k\right] = [\nu\times_k]$ and $\hat{z}^{(0)} = \dot{T}$ into Eq. (12), this equation becomes analogeous to Eq. (9). We can then construct the matrix which is shown in Fig. 2. In short, with $M = N + 1$

$$\dot{C} = C\left[\hat{\zeta}\times_{kM}\right]. \tag{14}$$

And when $k = 6$, we can define the dual representation with

$$\dot{\bar{C}} = \bar{C}\left[\hat{\zeta}\bar{\times}_{6M}\right] \tag{15}$$

where $\left[\hat{\zeta}_{\{p\}}\bar{\times}_{6M}\right] = -\left[\hat{\zeta}_{\{p\}}\times_{6M}\right]^\mathsf{T}$ and $\bar{C}_{\{p\}} = -(C_{\{p\}})^\mathsf{T}$ for $p = 0..N$. The index $\cdot_{\{p\}}$ represents the p-th sub-vector / sub-matrix of the preceded variable. All matrices in Eq. (14) and (15) are lower block-triangular matrices and operation on it can be optimized as shown in Appendix A. $C \in \mathbb{R}^{kM\times kM}$ is called N-CMTM and depends on T so we have $k \in \{4, 6\}$ and $[\zeta\times_{kM}]$ is its differential operator which also shares the same properties of Eq. (10). In what comes next, unless specified, we adopt the operator $[\cdot\times] \triangleq [\cdot\times_k]$ where $k \in \{3, 4, 6, 4M, 6M\}$. For example, we have $[\cdot\times_6]$ when X is used.

Eq. (14) provides the skeleton of the N-CMTM construction method. Combining it with Eq. (12), $\forall p \in \{0..N-1\}$ the N-CMTM sub-matrices are

$$C_{\{p+1\}} = \hat{z}^{(p)} = \frac{1}{p+1}\sum_{k=0}^{p} C_{\{p-k\}}\left[\hat{\zeta}_{\{k\}}\times\right] \tag{16}$$

with $C_{\{0\}} = T$ (which is the 0-CMTM).

Although the N-CMTM embeds the motion information through the operator $[\cdot\times]$, it is not directly accessible. To retrieve the motion from a N-CMTM, we need to "deconstruct" it, which is strictly the reverse of the construction operation. $\forall p \in \{0..N-1\}$ we have

$$\zeta_{\{p\}} = \left[p! C^{-1}_{\{0\}}\left((p+1)C_{\{p+1\}} - \sum_{k=0}^{p-1}\frac{1}{k!}C_{\{p-k\}}\hat{\zeta}_{\{k\}}\right)\right]^\vee \tag{17}$$

V. Kinematics & Dynamics with the N-CMTM

We will now consider an open kinematic chain composed of N_J links and joints. First, we define the following:

$$\left\{ \begin{array}{rl} n_{Ji}: & \text{Number of parameters to describe joint } i \\ n_{Di}: & \text{Number of DoF of joint } i \\ p(i): & \text{Parent body of joint } i. \\ \mathcal{C}(i): & \text{Set of direct children (leave-side) of body } i. \\ \mathcal{P}(i): & \text{Set of all parents (root-side) of body } i. \end{array} \right. .$$

Let $q_i \in \mathbb{R}^{n_{Ji}}$ be the generalized coordinates of joint i. Then there exists a function q2T so that

$${}^{p(i)}T_i = \mathrm{q2T}(q_i). \tag{18}$$

The velocity $\nu_i^{p(i)}$ across joint i is related to its generalized joint velocity vector ψ_i by

$$\nu_i^{p(i)} = S_i\psi_i. \tag{19}$$

where the matrix S is called the motion subspace matrix and

$$S \triangleq \left\{ \begin{array}{lr} \left[e_3^\mathsf{T} \; 0_3^\mathsf{T}\right]^\mathsf{T} & (rotational) \\ \left[0_3^\mathsf{T} \; e_3^\mathsf{T}\right]^\mathsf{T} & (linear) \\ \left[e_3^\mathsf{T} \; he_3^\mathsf{T}\right]^\mathsf{T} & (helical) \\ \left[1_{3\times3} \; 0_{3\times3}\right]^\mathsf{T} & (spherical) \\ 1_{6\times6} & (free) \end{array} \right. \tag{20}$$

where e is the axis of the joint and h the pitch of the screw. It follows that the generalized force τ_i of joint i is

$$\tau_i = S_i^\mathsf{T} f_i^{p(i)} \tag{21}$$

where $f_i^{p(i)}$ is the force passing through joint i. From Eq. (9), we have

$$\begin{aligned} {}^{p(i)}\dot{T}_i &= {}^{p(i)}T_i\left[\nu_i^{p(i)}\times\right] \\ &= {}^{p(i)}T_i\left[S_i\psi_i\times\right] \end{aligned} \tag{22}$$

which provides us with the last necessary equation to create the mapping matrix G

$$\zeta_i^{p(i)} = G_i\Psi_i \tag{23}$$

with

$$\Psi_i \triangleq \left[\psi_i^\mathsf{T} \; \dot{\psi}_i^\mathsf{T} \; \cdots \; \psi_i^{(N)^\mathsf{T}}\right]^\mathsf{T} \tag{24}$$

and

$$G_i = \mathrm{diag}(S_i). \tag{25}$$

$$\underbrace{\begin{pmatrix} \hat{z}^{(0)} & 0 & \dots & 0 \\ \hat{z}^{(1)} & \hat{z}^{(0)} & \dots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ \hat{z}^{(n)} & \hat{z}^{(n-1)} & \dots & \hat{z}^{(0)} \end{pmatrix}}_{\dot{C}} = \underbrace{\begin{pmatrix} \hat{a}^{(0)} & 0 & \dots & 0 \\ \hat{a}^{(1)} & \hat{a}^{(0)} & \dots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ \hat{a}^{(n)} & \hat{a}^{(n-1)} & \dots & \hat{a}^{(0)} \end{pmatrix}}_{C} \underbrace{\begin{pmatrix} \hat{y}^{(0)} & 0 & \dots & 0 \\ \hat{y}^{(1)} & \hat{y}^{(0)} & \dots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ \hat{y}^{(n)} & \hat{y}^{(n-1)} & \dots & \hat{y}^{(0)} \end{pmatrix}}_{[\hat{\zeta}\times_{kM}]}$$

Fig. 2: Comprehensive Motion Transformation Matrix and its derivative. All matrices define a very specific layout corresponding to the one described in Appendix A. Considering both the layout and the sparsity can greatly improve the computation time of the N-CMTM.

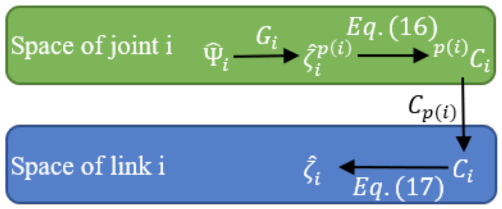


Fig. 3: Mappings from joint space to link space variables. The N-CMTM of joint an arbitrary joint i is calculated from the joint motion $\hat{\Psi}_i$. The N-CMTM of link i is calculated by multiplication of the N-CMTM of its parent link and the N-CMTM of joint i. The motion $\hat{\zeta}_i$ is then computed by deconstructing the resulting matrix.

Note that we consider that all joints are fixed here. In cases where S_i depends on q_i (*e.g.* rack and pinion joint), G_i becomes a lower block-triangular matrix made of S_i time-derivatives.

A. Forward Kinematics

The joint space of N-CMTM ${}^{p(i)}C_i$ and its link space C_i can now be built as illustrated in Fig. 3. And finally, as the N-CMTM is a transformation matrix, it also obeys to the chain rule

$$C_j = C_i {}^iC_j. \tag{26}$$

Thus the FK becomes straightforward

$$C_i = C_{p(i)} {}^{p(i)}C_i, \ \forall i \in \{1..N_J\} \tag{27}$$

where C_0 is composed of an eventual global transformation of the system as well as global motion (*e.g.* gravity).

Hereafter, an N-CMTM built from X with $N \geq 2$ is used.

B. Inverse Dynamics

As the N-CMTM is an extended transformation matrix, we can use N-CMTM instead of spacial transformation matrices in the classical Recursive Newton-Euler Algorithm (RNEA) [16].

Let $\mathrm{h_i}^{(k)}$ be the k-th order time derivative of momentum of link i, and $h_i = [\mathrm{h_i}^{(0)\mathsf{T}} \cdots \mathrm{h_i}^{(N)\mathsf{T}}]^\mathsf{T}$ are the set of the time derivatives, which has the following relationship:

$$h_i = \tilde{I}_i \zeta_i$$

where $\tilde{I}_i = \mathrm{diag}\,[I_i, \ \cdots, \ I_i]$, a block diagonal matrix composed of the inertia matrix of link i which is time invariant.

Let $\mathrm{f_i}^{(k)}$ be the k-th order time derivative of link force, and $f_i = [\mathrm{f_i}^{(0)\mathsf{T}} \cdots \mathrm{f_i}^{(N-1)\mathsf{T}}]^\mathsf{T}$ are the set of the derivatives. The link force in the acceleration level is computed such that $\mathrm{f_i}^{(0)} = \mathrm{h_i}^{(1)} + [\nu_i \bar{\times}]\, \mathrm{h}_i^{(0)}$. This relationship can be extended to the general form between f_i and h_i as follows:

$$f_i = h_{i\{1:N\}} + \left[\zeta_{i\{0:Q\}}\bar{\times}\right] h_{i\{0:Q\}} \tag{28}$$

where $Q = N - 1$ and $M_{\{a:b\}}$ is the sub-vector / sub-matrix of M corresponding from a-th order derivative to b-th order one. Since forces are at the momentum time-derivative level, hat notation is shifted. To overcome this, we pose $D_N = \mathrm{diag}\,[1, 2, \cdots, N]$ so we have

$$\hat{f}_i = D_N \hat{h}_{i\{1:N\}} + \left[\hat{\zeta}_{i\{0:Q\}}\bar{\times}\right] \hat{h}_{i\{0:Q\}}, \ \forall N > 0 \tag{29}$$

Under the N-CMTM, the computation of the forces across a joint i is

$$\hat{f}_i^{p(i)} = \hat{f}_i + \sum_{j \in \mathcal{C}(i)} {}^i\bar{C}_{j\{0:Q\}} \hat{f}_j^i. \tag{30}$$

The joint torque is finally computed using Eq. (21). Note that, during this process, it is also convenient to also compute the momentum passing through joint i which is provided by

$$\hat{h}_i^{p(i)} = \hat{h}_i + \sum_{j \in \mathcal{C}(i)} {}^i\bar{C}_{j\{0:N\}} \hat{h}_j^i. \tag{31}$$

From these equations, a Comprehensive Recursive Newton-Euler Algorithm (CoRNEA) can be defined (see Alg. (1))

$$\mathcal{T} = \mathrm{CoRNEA}(\mathrm{model}, \aleph) \tag{32}$$

where $\mathcal{T} = [\hat{\tau}^\mathsf{T}, \cdots, \hat{\tau}^{(Q)\mathsf{T}}]^\mathsf{T}$ is the concatenation of torque of all joints and $\aleph = [\hat{\Psi}_1^\mathsf{T}, \cdots, \hat{\Psi}_{N_J}^\mathsf{T}]^\mathsf{T}$ is the concatenation of motion of all joints.

C. Forward Dynamics

Let us now adapt N-CMTM to the Articulated-Body Algorithm (ABA) [10]. There are several variations of the forward dynamics algorithm depending on what input we are considering. Here, we consider the generalized joint angles and velocity are known and the output is the joint acceleration and its derivatives. This case is more tricky since we need to know the $i-1$ joint motion derivative before computing the next

Algorithm 1: Inverse Dynamics (CoRNEA)

```
**Input:** robot_model, $\aleph$
FK(robot_model, $\aleph$)                              // See Eq. (27)
**for** $i \leftarrow 1$ **to** $N_J$ **do**                    // From root to leaves
    $[{}^0X_i, \hat{\zeta}_i] \leftarrow$ DECONSTRUCT$(C_i)$   // See Eq. (17)
    $\hat{h}_i \leftarrow \tilde{I}_i \cdot \hat{\zeta}_i$
    $\hat{f}_i \leftarrow D_N \hat{h}_{i\{1:N\}} + \left[\hat{\zeta}_{i\{0:Q\}}\times\right] \hat{h}_{i\{0:Q\}}$
**for** $i \leftarrow N_J$ **to** $1$ **do**                    // From leaves to root
    $\hat{f}_i^{p(i)} \leftarrow \hat{f}_i^{p(i)} + \hat{f}_i$
    $\mathcal{T}_i \leftarrow G_i^{\mathsf{T}} \hat{f}_i^{p(i)})$                      // See Eq. (13)
    **if** $p(i) \neq 0$ **then**
        $\hat{f}_i^{p(p(i))} \leftarrow \hat{f}_i^{p(p(i))} + {}^{p(i)}\bar{C}_{i\{0:Q\}} \cdot \hat{f}_i^{p(i)}$
**Output:** $\mathcal{T}$
```

one. To compute the i-th derivative of joint motion, we need to know its $i-1$-th derivative first. Therefore, we need to apply the standard ABA recursively. Let us define the notation $[:k]$ which takes all elements until order k included. Mathematically, for a system composed of N_j joints, $\aleph[:n]$ is equivalent to $\forall j \in \{1..N_J\}, \forall k \in \{n+1..N\}, \aleph_j[k] = \psi_j^{(k)} = 0$, or, more explicitly

$$\left\{\begin{array}{rcl} \aleph[:n] & = & \left[\hat{\Phi}_1^{\mathsf{T}}[:n],\ \cdots,\ \hat{\Phi}_{N_j}^{\mathsf{T}}[:n]\right]^{\mathsf{T}} \\ \hat{\Phi}_j[:n] & = & \left[\hat{\phi}_j^{(0)\mathsf{T}},\ \cdots,\ \hat{\phi}_j^{(n)\mathsf{T}},\ 0,\ \cdots,\ 0\right]^{\mathsf{T}} \\ j & \in & \{1..N_j\} \end{array}\right. \tag{33}$$

so all elements of order superior or equal to n are zeros. Using the standard ABA, we get

$$\dot{\psi} = \text{ABA}(\text{model}, \tau) \tag{34}$$

then performing a CoRNEA we have

$$\mathcal{T} = \text{CoRNEA}(\text{model}, \aleph[:1]) \tag{35}$$

and since the jerk $\aleph[2]$ is 0, $\mathcal{T}[2]$ is the bias force. We then compute the next joint angle derivative

$$\ddot{\psi} = \text{ABA}(\text{model}, \dot{\tau} - \mathcal{T}[2]). \tag{36}$$

and by recursion we get all motion vectors. Alg. (2) provides the summary of the computation based on ABA and CoRENA.

Algorithm 2: Recursive Forward Dynamics

```
**Input:** model, $\psi_1, \cdots, \psi_{N_L}, \tau, \cdots, \tau^{(N)}$
$\aleph \leftarrow 0$
$\aleph[0] \leftarrow \psi_1, \cdots, \psi_{N_L}$
$\aleph[1] \leftarrow$ ABA(model, $\tau$)
**for** $i \leftarrow 2$ **to** $N$ **do**
    $\mathcal{T} \leftarrow$ CoRNEA(model, $\aleph$)
    $\aleph[i] \leftarrow$ ABA(model, $\frac{\tau^{(i)}}{i!} - \mathcal{T}[i]$)
**Output:** $\aleph$
```

Standard algorithms are not the only features essential to robotics, so we go further in the CMTM analysis and consider its involvement in comprehensive equations and Jacobians.

VI. Toward advanced algorithmic tools

In this section we present advanced kinematics and dynamics tools that the N-CMTM provides. The comprehensive equations represent the whole kinematics and dynamics by means of an unique matrix while Jacobians links the whole model parameters to one link/joint variable.

A. Comprehensive equations

To introduce the comprehensive equations we first consider the forward kinematics equation of link i

$$\hat{\zeta}_i = \hat{\zeta}_i^{p(i)} + {}^iC_{p(i)}\hat{\zeta}^{p(i)}. \tag{37}$$

Let's define the link space comprehensive motion $\mathcal{V}_L = \left[\hat{\zeta}_1^{\mathsf{T}},\ \cdots,\ \hat{\zeta}_{N_J}^{\mathsf{T}}\right]^{\mathsf{T}}$ and the joint space comprehensive motion $\mathcal{V}_J = \left[\hat{\zeta}_1^{0\mathsf{T}},\ \cdots,\ \hat{\zeta}_{N_J}^{p(N_J)\mathsf{T}}\right]^{\mathsf{T}}$ then

$$\mathcal{V}_J = \mathcal{L}\mathcal{V}_L \tag{38}$$

where

$$\mathcal{L}_{\{i,j\}} = \left\{\begin{array}{ll} 1_{6M\times 6M}, & \text{if } i=j \\ -{}^iC_j, & \text{if } j=p(i) \\ 0_{6M\times 6M}, & \text{otherwise} \end{array}\right. . \tag{39}$$

In the case of open kinematic chains, matrix $\mathcal{L}$ is always invertible and its inverse $\mathcal{L}^{-1}$ is given by

$$\mathcal{L}^{-1}_{\{i,j\}} = \left\{\begin{array}{ll} 1_{6M\times 6M}, & \text{if } i=j \\ {}^iC_j, & \text{if } j\in\mathcal{P}(i) \\ 0_{6M\times 6M}, & \text{otherwise} \end{array}\right. \tag{40}$$

where C is the N-CMTM. We set $\mathcal{G} = \text{diag}\left[G_1,\ \cdots,\ G_{N_J}\right]$ so that $\mathcal{V}_J = \mathcal{G}\aleph$, then we get the link space comprehensive motion $\mathcal{V}_L$ / momentum $\mathcal{H}_L$, and the joint space comprehensive momentum $\mathcal{H}_J$, respectively

$$\mathcal{V}_L = \left(\mathcal{L}^{-1}\mathcal{G}\right)\aleph \tag{41}$$

$$\mathcal{H}_L = \left(\mathcal{M}\mathcal{L}^{-1}\mathcal{G}\right)\aleph \tag{42}$$

$$\mathcal{H}_J = \mathcal{L}^{-\mathsf{T}}\mathcal{M}\mathcal{L}^{-1}\mathcal{G}\aleph = \left(\mathcal{I}\mathcal{G}\right)\aleph \tag{43}$$

where $\mathcal{M} = \text{diag}\left[\tilde{I}_1,\ \cdots,\ \tilde{I}_{N_J}\right]$ and $\mathcal{I}$ is the N-order joint inertia matrix. The basic Jacobian of $\mathcal{V}_L$, $\mathcal{H}_L$ and $\mathcal{H}_J$ can be derived as the coefficient matrix of $\aleph$ in Eq. (41), Eq. (42) and Eq. (43), respectively. They are the basic formulations used for deriving the Jacobians in the following subsection.

B. Jacobians

Presented in Section II, we consider optimization with complex functions which need a calculation of the gradient $\partial\ddot{v}/\partial\dot{q}$, $\partial\dot{f}/\partial\ddot{q}$ or $\partial\ddot{\tau}/\partial q$. Equivalently, we want the Jacobian J_1, J_2 and J_3 that respectively maps $\ddot{v}_i = J_1\dot{\psi}_{\text{all}}$, $\dot{f}_i = J_2\ddot{\psi}_{\text{all}}$ and $\ddot{\tau}_i = J_3\psi_{\text{all}}$ for an arbitrary link/joint i. All J_1, J_2 and J_3 are non trivial to calculate but the comprehensive equation along the N-CMTM makes it feasible.

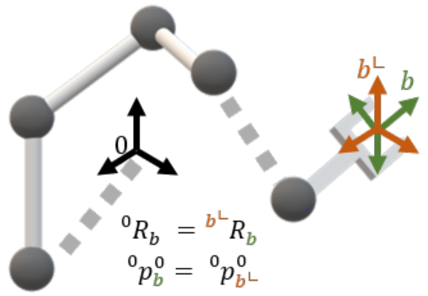

$b^{\llcorner}$
b
0
${}^{0}R_{b} = {}^{b^{\llcorner}}R_{b}$
${}^{0}p_{b}^{0} = {}^{0}p_{b^{\llcorner}}^{0}$

Fig. 4: The 3 main frames of an arbitrary link. In black, the world frame, in green, the body frame and in orange, the inertia-fixed frame.

1) Comprehensive basic Jacobian: As in [2] (Eq. (32)), the j-th column-band of comprehensive Jacobian of link i is

$$\check{\mathcal{J}}_{i,j} = U_i^{-1\,i}C_j U_j^{p(j)} G_j,\ j \in \mathcal{P}(i) \cup i \tag{44}$$

where U is a lower block-triangular matrix. If we consider the comprehensive Jacobian relative to time-derivative quantities:

$$\mathcal{J}_{i,j} = {}^{i}C_j G_j,\ j \in \mathcal{P}(i) \cup i \tag{45}$$

which also corresponds to the i-th row-band of the matrix $\mathcal{L}^{-1}\mathcal{G}$ in Eq. (41). We discuss in Section VI-B4) the whereabouts of the vanishing matrix U. Let us consider the basic Jacobian J that maps $\nu = J\psi$. By deriving N-times we get Eq. (12) with $\hat{z}^{(0)} = \nu$, $\hat{a}^{(0)} = J$ and $\hat{y}^{(0)} = \psi$. By comparison, we get for an arbitrary link i the p-th body Jacobian derivative with

$${}^{i}\hat{J}_{i,j}^{(p)} = \mathcal{J}_{i,j\,\{p\}} = {}^{i}C_{j\,\{p\}} S_j,\ j \in \mathcal{P}(i) \cup i. \tag{46}$$

Thus, the Jacobian J_1 that maps $\dddot{\nu}_i = J_1\dot{\psi}$ is $J_1 = 4!\mathcal{J}_{i,\text{all}\,\{3\}}$. In classic mechanics, the provided Jacobian is in the inertia-fixed frame $b^{\llcorner}$ (see Fig. 4) of body b which is different from the body Jacobian. The inertia-fixed frame $b^{\llcorner}$ is a rotation-fixed frame from world frame at the body origin. With the fixed rotation being the identity, we have

$${}^{0}T_b = {}^{0}T_{b^{\llcorner}}\,{}^{b^{\llcorner}}T_b \tag{47}$$

with

$$\left\{\begin{array}{lcl} {}^{0}R_{b^{\llcorner}} & = & 1_{3\times3} \\ {}^{0}r_{b^{\llcorner}}^{0} & = & {}^{0}r_b^{0} \end{array}\right. \text{ and } \left\{\begin{array}{lcl} {}^{b^{\llcorner}}R_b & = & {}^{0}R_b \\ {}^{b^{\llcorner}}r_b^{b^{\llcorner}} & = & 0_{3\times1} \end{array}\right. \tag{48}$$

The global Jacobian and the spatial (at world frame) Jacobian can be computed the same way as the body Jacobian

$$\left\{\begin{array}{lcl} {}^{b^{\llcorner}}\hat{J}_{b,j}^{(p)} & = & {}^{b^{\llcorner}}C_{j\,\{p\}} S_j \\ {}^{0}\hat{J}_{b,j}^{(p)} & = & {}^{0}C_{j\,\{p\}} S_j \end{array}\right.,\ j \in \mathcal{P}(b) \cup b. \tag{49}$$

2) Comprehensive Link force Jacobian: To compute this Jacobian, we first need the Jacobian $\mathcal{K}$ that maps $\hat{h}_i = \mathcal{K}_i\aleph$. It is directly computed from Eq. (45)

$$\hat{h}_i = \tilde{I}_i\zeta_i = \tilde{I}_i\mathcal{J}_i\aleph \tag{50}$$

so $\mathcal{K}_i = \tilde{I}_i\mathcal{J}_i$. Comparing with the link momentum given by Eq. (42), it is also the i-th row-band of matrix $\mathcal{M}\mathcal{L}^{-1}\mathcal{G}$.

The link i force Jacobian $\mathcal{N}_i$ that maps $\hat{f}_i = \mathcal{N}_i\aleph$ is obtained using Eq. (29) and (50)

$$\begin{aligned} f_i &= D_N\hat{h}_{i\{1:N\}} + \left[\hat{\zeta}_{i\{0:Q\}}\bar{\times}\right]\hat{h}_{i\{0:Q\}} \\ &= \left(D_N^{\Delta} + \left[\hat{\zeta}_{i\{0:Q\}}\bar{\times}\right]\right)\mathcal{K}_i\aleph \\ &= \mathcal{D}_i\mathcal{K}_i\aleph \end{aligned} \tag{51}$$

where D_N^{Δ} is the same as D_N with value on the upper diagonal. Thus, the Jacobian J_2 that maps $\ddot{f}_i = J_2\ddot{\psi}$ is $J_2 = 2!\mathcal{N}_{i\{2,2\}}$. Note that, due to the hat notation shift, $\mathcal{N}$ looses the properties defined in Section A.

3) Comprehensive Torque Jacobian: To compute this Jacobian, we first need the joint momentum Jacobian that maps $\hat{h}_i^{p(i)} = \mathcal{B}_i\aleph$ followed by the joint force Jacobian that maps $f_i^{p(i)} = \mathcal{Q}_i\aleph$. The former is directly computed from Eq. (43) using a method similar to the CRBA.

$$\mathcal{B}_i = \left(\mathcal{L}^{-\mathsf{T}}\mathcal{I}\mathcal{L}^{-1}\mathcal{G}\right)_{\{i,\text{all}\}}. \tag{52}$$

The comprehensive joint force Jacobian can be directly computed from the comprehensive joint momentum Jacobian:

$$\mathcal{Q}_i = \mathcal{D}_i\mathcal{B}_i. \tag{53}$$

To prove it, let us first calculate the time-derivative of Eq. (31)

$$\dot{\hat{h}}_i^{p(i)} = \dot{\hat{h}}_i + \sum_{j\in\mathcal{C}(i)} {}^{i}\bar{C}_j\left(\left[\hat{\zeta}_j^{i}\bar{\times}\right]\hat{h}_j^{i} + \dot{\hat{h}}_j^{i}\right). \tag{54}$$

Note that, to save space, we have dropped the $\cdot_{\{a:b\}}$ notation and set $\dot{h} = h_{\{1:N\}}$. Let us now consider all bodies l so that $\mathcal{C}(l) = \emptyset$, the last leave-side bodies. We have

$$\hat{f}_l^{p(l)} = \hat{f}_l = \mathcal{D}_l\hat{h}_l = \mathcal{D}_l\hat{h}_l^{p(l)}. \tag{55}$$

Let us now assume this relation is true for an arbitrary joint j. Let $i = p(j)$, from Eq. (29), Eq. (30) and (37)

$$\begin{aligned} \hat{f}_i^{p(i)} &= \hat{f}_i + \sum_{j\in\mathcal{C}(i)} {}^{i}\bar{C}_j\hat{f}_j^{i} \\ &= \dot{\hat{h}}_i + \left[\hat{\zeta}_i\bar{\times}\right]\hat{h}_i + \sum_{j\in\mathcal{C}(i)} {}^{i}\bar{C}_j\left(\dot{\hat{h}}_j^{i} + \left[\hat{\zeta}_j\bar{\times}\right]\hat{h}_j^{i}\right) \\ &= \dot{\hat{h}}_i + \left[\hat{\zeta}_i\bar{\times}\right]\hat{h}_i + \sum_{j\in\mathcal{C}(i)} {}^{i}\bar{C}_j\left(\dot{\hat{h}}_j^{i} + \left[{}^{j}C_i\hat{\zeta}_i + \hat{\zeta}_j^{i}\bar{\times}\right]\hat{h}_j^{i}\right) \\ &= \mathcal{D}_i\hat{h}_i^{p(i)} \end{aligned} \tag{56}$$

Finally, the torque Jacobian $\mathcal{R}$ that maps $\hat{\tau} = \mathcal{R}\aleph$ is

$$\mathcal{R} = \mathcal{G}^{\mathsf{T}}\mathcal{Q}. \tag{57}$$

Thus, the Jacobian J_3 that maps $\dddot{\tau}_i = J_3\psi$ is $J_3 = 3!\mathcal{R}_{i\{3,0\}}$. See Fig. 5 for a resume of the comprehensive mappings.

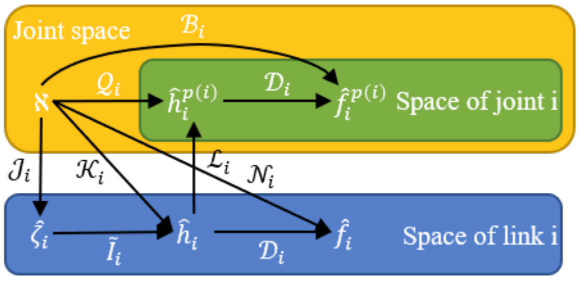


Fig. 5: Comprehensive N-CMTM mappings between joint space and link space. Each Jacobian matrix maps the concatenation of all joints motion vector $\aleph$ to one of the most common physical quantities (Link motion joint/link momentum and jint/link force).

4) Discussion: We have mostly developed instantaneous Jacobians, like $\mathcal{J}$ (Eq. (45)), over Jacobians of variation, like $\check{\mathcal{J}}$ (Eq. (44)). This is because the former is almost always enough. Let us consider a function $g(\nu)$ depending on a link velocity ν, its partial derivative with regard to the system parameters is

$$\frac{\partial g(\nu)}{\partial \chi} = \frac{\partial g(\nu)}{\partial \nu}\frac{\partial \nu}{\partial \chi} = \frac{\partial g(\nu)}{\partial \nu}\mathcal{J}. \tag{58}$$

In Section VI-B1), we said that $\check{\mathcal{J}}$ varies from $\mathcal{J}$ by matrices U which *vanished* when considering instantaneous Jacobian. This can be understood by clarifying the difference between a parameter variation like $\delta\theta$ and the instantaneous time-derivative like ψ. The relation between the two is

$$\delta\theta = \psi\delta t \tag{59}$$

The matrix U corresponds to this relationship. It enforces the constraints between parameter time-derivatives and parameter variations. Formally, $\delta\theta$ lies in the *tangent space* of q and the two are independent variables. The matrix U "corrects" this lack of dependency.

Now, when considering an optimization scheme, the next step $k+1$ is computed from an integration step ΔT which takes care of these constraints. For example

$$q_{k+1} = f_{integ}(q_k, \nu_k, \cdots, \nu_k^{(N)}, \Delta T). \tag{60}$$

This means that $\mathcal{J}$ can be used in place of $\check{\mathcal{J}}$. In other (rare) cases, *e.g.* $q_{k+1} = q_k + \alpha$ with a some step α, there is an independence between all motions, thus, Jacobians of variation (like $\check{\mathcal{J}}$) must be used. Note that, all instantaneous Jacobians here have their counterpart Jacobians of variation.

VII. Tests & scalability

We have developed a C++ library[1] to test the theory[2]. We have implemented the forward kinematics, both forward and inverse dynamics as well as all Jacobians. For the tests we used two different models:

- A manipulator robot;
- A simplified human-like model.

The human-like model is composed of 37-dof and 46 joint parameters. The root joint is a free-floating joint, the shoulders, wrists, hips and ankles are spherical joints, and the remaining joints are revolute joints. The manipulator is only composed on N_j revolute joints of different axis where N_j is chosen arbitrary. All test has been made with a compilation in Debug mode so the compiler does not influence the result.

To check the results we use several methods. The first one is the RBDyn[3] library. It is a rigid multi-body dynamic library used in various robot controllers. Since the library is limited to the computation up to the acceleration level, we use a numerical differentiation for higher order. The p-th finite difference $dv_p(t)$ is computed by

$$dv_p(t) = \frac{\nu^{(p)}(t+\Delta T) - \nu^{(p)}(t)}{\Delta T}. \tag{61}$$

We also consider a double differentiation which is given by

$$ddv_p(t) = \frac{dv_{p-1}(t+\Delta T) - dv_{p-1}(t)}{\Delta t}. \tag{62}$$

The last considered method is the Automatic Differentiation (AD) method using CppAD[4]. AD is a tool that can compute partial derivatives of a provided set of variables. Let us define the function fk so that $\dot{\nu}_j = \text{fk}(Q)$ where $\dot{\nu}_j$ is the spatial acceleration of a body j and $Q = [q_{\text{all}}^\intercal \; \psi_{\text{all}}^\intercal \; \dot{\psi}_{\text{all}}^\intercal]^\intercal$. Its first time-derivative is given by

$$\ddot{\nu}_j = \frac{d\text{fk}}{dt} = \frac{\partial \text{fk}}{\partial Q}\frac{\partial Q}{\partial t} = \frac{\partial \text{fk}}{\partial Q}\dot{Q} \tag{63}$$

and the second derivative is similar to

$$\dddot{\nu}_j = \dot{Q}^\intercal \frac{\partial^2 \text{fk}}{\partial Q^2}\dot{Q} + \frac{\partial \text{fk}}{\partial Q}\ddot{Q}. \tag{64}$$

We can use the AD to get the partial derivative parts and thus compute the time derivatives. Since it is difficult to compute the AD for high-order partial derivatives (i.e. $\partial^N \text{fk}/\partial Q^N$) due to its computational complexity, only the first and second order time-derivatives are evaluated in the case when using AD.

Finally, to evaluate high-order derivative, we also need to know the high-order joint motion. Thus, we have generated a time-dependent angular velocity

$$\psi_i = a * \sin(\omega t + b) \tag{65}$$

where a, ω and b are random 6-by-1 vectors and $*$ is a coefficient-wise multiplication. Then, we compute the motion derivative and the joint angles depending on its type. Finally, the system time step is $\Delta T = 1e^{-8}$s.

[1]https://github.com/vsamy/coma
[2]https://github.com/vsamy/cdm
[3]https://github.com/jrl-umi3218/RBDyn
[4]https://github.com/coin-or/CppAD

A. *Test of FK, ID and FD*

We first consider the human-like model and test the FK (see Eq. (27)). All numerical differentiations are computed from N-CMTM results, *i.e.* dv_p and ddv_p are computed from N-CMTM p-order results. Table (I) provides the maximum norm error of each link motion between the CMTM and each method. Remembering that, for double-precision, the epsilon machine is $1.11e^{-16}$, the N-CMTM results regarding RBDyn and AD are validated. Second-order Numerical differentiation hits fairly bad results compared to the first-order. This means that this kind of methods should not be used for high-order computation. Considering that $\Delta T = 1e^{-8}$s, the higher-order is recursively validated from the first-order numerical differentiation. Table (II) provides the maximum result error between N-CMTM and RBdyn/numerical differentiation for the ID (see Alg. (1)). An error of $1e^{-15}$ with RBDyn and $1e^{-6}$ with numerical differentiation validates the algorithms. Since the ID is validated, we tested the FD from ID results and got a maximum error of $6.88683e^{-14}$.

TABLE I: Max FK error between N-CMTM and each method.

order	RBDyn	$dv_p(t)$	$ddv_p(t)$	AD
velocity	$5.18703e^{-16}$	/	/	$2.28878e^{-16}$
accel...	$1.75542e^{-15}$	$1.08502e^{-7}$	/	$3.88578e^{-16}$
jerk	/	$2.83273e^{-7}$	15.6074	$1.50195e^{-15}$
snap	/	$5.10762e^{-7}$	54.8067	$2.27058e^{-15}$
crackle	/	$8.10141e^{-7}$	150.261	/

TABLE II: Max ID error between CMTM and other methods.

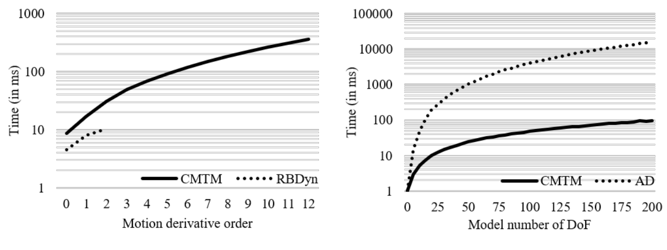
order	RBDyn	$dv_p(t)$
0 (momentum)	$5.62864e^{-16}$	/
0 (torque)	$4.33414e^{-15}$	/
1	/	$1.04706e^{-6}$
2	/	$1.2258e^{-6}$
3	/	$2.0509e^{-6}$

B. *High-Order Jacobian*

Let us now get a closer look at the Jacobian evaluation. The easiest way to check for Jacobian validity is to multiply them by the joint motion and check for the results. Since Table (I) shows that N-CMTM is correct, we compare the Jacobian results directly with itself. Table (III) provides the maximum jacobian errors. The worst error is at $1.4876e^{-12}$ and happens at the 5th order when computing the joint momentum derivative which also validate the Jacobian algorithms. The reason for larger error with order is the increase of non-zero multiplication.

TABLE III: Max Jacobian error $1e^{-16}$

order	$\mathcal{J}\aleph$	$\mathcal{K}\aleph$	$\mathcal{N}\aleph$	$\mathcal{B}\aleph$	$\mathcal{Q}\aleph/\mathcal{R}\aleph$
1	1.49629	1.49629	1.49629	23.5579	23.5579
2	15.6296	22.2720	20.1281	143.064	99.2641
3	87.0878	87.6064	76.7999	148.885	120.504
4	193.816	209.061	163.068	842.994	944.174
5	766.013	547.351	887.600	2277.34	**3241.04**



(a) (b)

Fig. 6: N-CMTM scalability computation time (in debug mode). Fig. 6a represents the computation time of the FK with regard to the N-CMTM order. For RBDyn, only the FK up to the acceleration is computed. Fig. 6b represents the computation time of the FK with regard to the model's number of joints. A 5-CMTM is compared to a two times derivative of FK using AD.

C. *N-CMTM scalability*

One of the concerns we may have about the N-CMTM is how it scales with the order of the derivatives and with the model complexity. To have some idea about the former case we considered a 100-DoF manipulator and benched N-CMTM on a simple FK (Eq. (27)) algorithm up to the 12-th order which we consider high enough. The results are presented in Fig. 6a, we also compare the N-CMTM computation time with RBDyn computation time, but of course, only up to the acceleration level for RBDyn.

The figure shows that the N-CMTM computation time grows almost linearly with regard to its order. We also remark that the N-CMTM is two to three times slower than RBDyn. The reason is twofold, as remarked by Park *et al.* in [24], this formulation is slower than optimized forward kinematic equations due to its very meaning. However, as RBDyn is a rather well optimized library and ours a first draft, the coding part is also to be taken into account. Note that N-CMTM parallel computation could close the computation time gap.

A more important scaling parameter is the model complexity. To tackle this part, we bench, also on a FK algorithm, Fig. 6b, the 5-CMTM with manipulators from 5-DoF up to 200-DoF. We also add the computation time of the second time-derivative with AD as seen in Eq. (64). Being 100 times faster at 200-DoF, the N-CMTM presents promising results. As CppAD may have lot of overhead in debug mode, a comparison on release mode has been investigated. On average, the N-CMTM is 30 times faster. Finally, As the system grows, the N-CMTM computation time grows linearly with it.

VIII. Conclusion & Future Work

We have presented N-CMTM that is an extension of classical transformation matrix. The N-CMTM not only enables the transformation of the position and orientation but also of the velocity and its derivatives. We then have adapted the RNEA and ABA algorithms to compute high-order differential quantities such as the derivatives of torque, force, and acceleration.

We have also provided comprehensive equations and various Jacobians matrices which can be used in optimization scheme. Finally, we have developed a C++ library to test the theory and to consider the scalability of the N-CMTM with regard to the order and to the number of DoF of the model. To sump up, the N-CMTM:

1) works with any order of derivatives;
2) provides easy computation of high-order Jacobians;
3) is adaptable to FK, FD and ID algorithms;
4) is easy to implement as illustrated in Fig. 3.

The future step is to develop the library in accordance to a musculoskeletal model in order to work on motion analysis which needs an implementation of close-loop wire-driven multi-body systems [21]. The main focus will be the Inverse Optimal Control [15] where high-order Jacobians are needed for optimal computation scheme. Regarding the library, although we have made some optimizations, it can still be further improved. Finally, we also want to refine the theory behind the N-CMTM and further develop the potential of it.

Acknowledgment

This work has been supported by JSPS KAKENHI Grant No. 17H00768 and No. 18H03315.

Appendix

A. Lower block-triangular matrix

Throughout the paper, many matrices get a very specific layout. They are lower block-triangular matrices defined with a sub-set of matrices. Let M be one of them. It has the following form

$$\begin{pmatrix} M_{\{0\}} & 0 & \cdots & 0 \\ M_{\{1\}} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ M_{\{N\}} & \cdots & M_{\{1\}} & M_{\{0\}} \end{pmatrix} \tag{66}$$

where $M_{\{k\}}, k \in \{0..N\}$ is a matrix.

This sparsity can be greatly exploited to improve the computation time of the multiplication operation. Let's have N so that $P = MN$. N can be either a vector or of the same layout as M. The sub-matrices $P_{\{k\}}, k \in \{0..N\}$ are computed recursively by

$$P_{\{k\}} = \sum_{p=0}^{k} M_{\{k-p\}} N_{\{p\}}, \quad k = 0..N. \tag{67}$$

Also, operations like $+$ and $-$ with N should be performed directly on the sub-matrices as well as operation $\times$ and $/$ with a scalar.

The inverse B of M exists if and only if the sub-matrix $M_{\{0\}}$ is invertible and B has the same layout of M. It can be recursively computed by

$$B_{\{i\}} = -M_{\{0\}}^{-1} \sum_{k=0}^{i-1} M_{\{i-k\}} B_{\{k\}}, \quad k = 0..N \tag{68}$$

where $B_{\{0\}} = M_{\{0\}}^{-1}$.

References

[1] K. Ayusawa, W. Suleiman, and E. Yoshida. Predictive Inverse Kinematics: optimizing Future Trajectory through Implicit Time Integration and Future Jacobian Estimation. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 566–573, 2019. doi: 10.1109/IROS40897.2019.8968110.

[2] Ko Ayusawa and Eiichi Yoshida. Comprehensive theory of differential kinematics and dynamics towards extensive motion optimization framework. *The International Journal of Robotics Research*, 37(13-14):1554–1572, 2018. doi: 10.1177/0278364918772893.

[3] Atılım Günes Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey. *The Journal of Machine Learning Research*, 18(1):5595–5637, 2017.

[4] G. Buondonno and A. De Luca. A recursive Newton-Euler algorithm for robots with elastic joints and its application to control. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5526–5532, 2015. doi: 10.1109/IROS.2015.7354160.

[5] J. Carpentier and N. Mansard. Analytical Derivatives of Rigid Body Dynamics Algorithms. In *Proceedings of Robotics: Science and Systems*, Pittsburgh, Pennsylvania, June 2018. doi: 10.15607/RSS.2018.XIV.038.

[6] Benjamin Chrétien, Adrien Escande, and Abderrahmane Kheddar. Gpu robot motion planning using semi-infinite nonlinear programming. *IEEE Transactions on Parallel and Distributed Systems*, 27(10):2926–2939, 2016. doi: 10.1109/TPDS.2016.2521373.

[7] Michael Damsgaard, John Rasmussen, Søren Tørholm Christensen, Egidijus Surma, and Mark de Zee. Analysis of musculoskeletal systems in the anybody modeling system. *Simulation Modelling Practice and Theory*, 14(8):1100–1111, 2006. ISSN 1569-190X. doi: https://doi.org/10.1016/j.simpat.2006.09.001. URL https://www.sciencedirect.com/science/article/pii/S1569190X06000554. SIMS 2004.

[8] David Eager, Ann-Marie Pendrill, and Nina Reistad. Beyond velocity and acceleration: Jerk, snap and higher derivatives. *European Journal of Physics*, 37:065008, 11 2016. doi: 10.1088/0143-0807/37/6/065008.

[9] Adrien Escande, Nicolas Mansard, and Pierre-Brice Wieber. Hierarchical quadratic programming: Fast online humanoid-robot motion generation. *The International Journal of Robotics Research*, 33(7):1006–1028, 2014. doi: 10.1177/0278364914521306.

[10] Roy Featherstone. *Rigid Body Dynamics Algorithms*. Springer, 01 2014. ISBN 978-0-387-74314-1. doi: 10.1007/978-1-4899-7560-7.

[11] T Flash and N Hogan. The coordination of arm movements: an experimentally confirmed mathematical model. *The Journal of Neuroscience*, 5(7):1688–1703, jul 1985. doi: 10.1523/jneurosci.05-07-01688.1985.

[12] Z. Fu, E. Spyrakos-Papastavridis, Y. h. Lin, and J. S. Dai. Analytical Expressions of Serial Manipulator Jacobians and their High-Order Derivatives based on Lie Theory*. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7095–7100, 2020. doi: 10.1109/ICRA40945.2020.9197131.

[13] C. Guarino Lo Bianco. Evaluation of Generalized Force Derivatives by Means of a Recursive Newton–Euler Approach. *IEEE Transactions on Robotics*, 25(4):954–959, 2009. doi: 10.1109/TRO.2009.2024787.

[14] Yangmin Li and Q. Xu. Kinematics and inverse dynamics analysis for a general 3-prs spatial parallel mechanism. *Robotica*, 23:219 – 229, 2005.

[15] Jonathan Lin, Vincent Bonnet, Adina Panchea, Nacim Ramdani, Gentiane Venture, and Dana Kulic. Human motion segmentation using cost weights recovered from inverse optimal control. pages 1107–1113, 11 2016. doi: 10.1109/HUMANOIDS.2016.7803409.

[16] J. Luh, M. Walker, and R. Paul. Resolved-acceleration control of mechanical manipulators. *IEEE Transactions on Automatic Control*, 25(3):468–474, 1980. doi: 10.1109/TAC.1980.1102367.

[17] J. Y. S. Luh, M. W. Walker, and R. P. C. Paul. On-Line Computational Scheme for Mechanical Manipulators. *Journal of Dynamic Systems, Measurement, and Control*, 102(2):69–76, 06 1980. ISSN 0022-0434. doi: 10.1115/1.3149599.

[18] Kevin M Lynch and Frank C Park. *Modern Robotics*. Cambridge University Press, 2017.

[19] D. Mellinger and V. Kumar. Minimum snap trajectory generation and control for quadrotors. In *2011 IEEE International Conference on Robotics and Automation*, pages 2520–2525, 2011. doi: 10.1109/ICRA.2011.5980409.

[20] A. Murai, K. Kurosaki, K. Yamane, and Y. Nakamura. Musculoskeletal-see-through mirror: Computational modeling and algorithm for whole-body muscle activity visualization in real time. *Progress in Biophysics and Molecular Biology*, 103(2):310–317, 2010. ISSN 0079-6107. doi: https://doi.org/10.1016/j.pbiomolbio.2010.09.006. URL https://www.sciencedirect.com/science/article/pii/S007961071000074X. Special Issue on Biomechanical Modelling of Soft Tissue Motion.

[21] Yoshihiko Nakamura, Katsu Yamane, Yusuke Fujita, and Ichiro Suzuki. Somatosensory computation for man-machine interface from motion-capture data and musculoskeletal human model. *IEEE Transactions on Robotics*, 21(1):58–66, 2005. doi: 10.1109/TRO.2004.833798.

[22] Shin'ichiro Nakaoka, Shizuko Hattori, Fumio Kanehiro, Shuuji Kajita, and Hirohisa Hirukawa. Constraint-based dynamics simulator for humanoid robots with shock absorbing mechanisms. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3641–3647, 2007. doi: 10.1109/IROS.2007.4399415.

[23] N. Orlandea, M. A. Chace, and D. A. Calahan. A Sparsity-Oriented Approach to the Dynamic Analysis and Design of Mechanical Systems—Part 1. *Journal of Engineering for Industry*, 99(3):773–779, 08 1977. ISSN 0022-0817. doi: 10.1115/1.3439312.

[24] F.C. Park, J.E. Bobrow, and S.R. Ploen. A Lie group formulation of robot dynamics. *The International journal of robotics research*, 14(6):609–618, 1995. doi: 10.1177/027836499501400606.

[25] N.S. Pollard, J.K. Hodgins, M.J. Riley, and C.G. Atkeson. Adapting human motion for the control of a humanoid robot. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, volume 2, pages 1390–1397 vol.2, 2002. doi: 10.1109/ROBOT.2002.1014737.

[26] Ludovic Righetti, Jonas Buchli, Michael Mistry, Mrinal Kalakrishnan, and Stefan Schaal. Optimal distribution of contact forces with inverse-dynamics control. *The International Journal of Robotics Research*, 32(3):280–298, 2013. doi: 10.1177/0278364912469821. URL https://doi.org/10.1177/0278364912469821.

[27] V. Samy, K. Ayusawa, and E. Yoshida. Real-time musculoskeletal visualization of muscle tension and joint reaction forces. In *2019 IEEE/SICE International Symposium on System Integration (SII)*, pages 396–400, Jan 2019. doi: 10.1109/SII.2019.8700414.

[28] John Schulman, Yan Duan, Jonathan Ho, Alex Lee, Ibrahim Awwal, Henry Bradlow, Jia Pan, Sachin Patil, Ken Goldberg, and Pieter Abbeel. Motion planning with sequential convex optimization and convex collision checking. *The International Journal of Robotics Research*, 33(9):1251–1270, 2014. doi: 10.1177/0278364914528132.

[29] Suril V. Shah, Paramanand V. Nandihal, and Subir K. Saha. Recursive dynamics simulator (redysim): A multibody dynamics solver. *Theoretical and Applied Mechanics Letters*, 2(6):063011, 2012. ISSN 2095-0349. doi: https://doi.org/10.1063/2.1206311. URL https://www.sciencedirect.com/science/article/pii/S2095034915301999.

[30] Garett A Sohl and James E Bobrow. A recursive multibody dynamics and sensitivity algorithm for branched kinematic chains. *J. Dyn. Sys., Meas., Control*, 123(3):391–399, 2001.

[31] W. Suleiman, E. Yoshida, F. Kanehiro, J. Laumond, and A. Monin. On human motion imitation by humanoid robot. In *2008 IEEE International Conference on Robotics and Automation*, pages 2697–2704, 2008. doi: 10.1109/ROBOT.2008.4543619.

[32] M. W. Walker and D. E. Orin. Efficient Dynamic Computer Simulation of Robotic Mechanisms. *Journal of Dynamic Systems, Measurement, and Control*, 104(3):205–211, 09 1982. ISSN 0022-0434. doi: 10.1115/1.3139699.

[33] E. Yoshida, I. Belousov, C. Esteves, and J.-P. Laumond. Humanoid motion planning for dynamic tasks. In *5th IEEE-RAS International Conference on Humanoid Robots, 2005.*, pages 1–6, 2005. doi: 10.1109/ICHR.2005.1573536.