

Analytical Second-Order Partial Derivatives of Rigid-Body Inverse Dynamics

Shubham Singh¹, Ryan P. Russell¹ and Patrick M. Wensing²

Abstract—Optimization-based robot control strategies often rely on first-order dynamics approximation methods, as in iLQR. Using second-order approximations of the dynamics is expensive due to the costly second-order partial derivatives of the dynamics with respect to the state and control. Current approaches for calculating these derivatives typically use automatic differentiation (AD) and chain-rule accumulation or finite-difference. In this paper, for the first time, we present analytical expressions for the second-order partial derivatives of inverse dynamics for open-chain rigid-body systems with floating base and multi-DoF joints. A new extension of spatial vector algebra is proposed that enables the analysis. A recursive algorithm with complexity of $\mathcal{O}(Nd^2)$ is also provided where N is the number of bodies and d is the depth of the kinematic tree. A comparison with AD in CasADi shows speedups of 1.5-3 \times for serial kinematic trees with $N > 5$, and a C++ implementation shows runtimes of $\approx 51\mu s$ for a quadruped.

I. INTRODUCTION

In recent years, optimization-based methods have become popular for robot motion generation and control. Although full second-order (SO) optimization methods offer superior convergence properties, most work has focused on using only the first-order (FO) dynamics approximation, such as in iLQR [1], [2]. Differential Dynamic Programming (DDP) [3] is a use-case for full SO optimization that has gained wide interest for robotics applications [4]–[7]. Variants of DDP using multiple shooting [8] and parallelization [9] have been developed for computational and numerical improvements.

The state-of-the-art for including SO dynamics derivatives in trajectory optimization is the work by Lee et al. [10], where they use forward chain-rule expressions (i.e., recursive derivative expressions) to calculate the SO partial derivatives of joint torques with respect to joint configuration and joint rates for revolute and prismatic joint models. Nganga and Wensing [11] presented a method for getting the SO directional derivatives of inverse dynamics as needed in the backward pass of DDP by employing reverse mode Automatic Differentiation (AD) and a modified version of the Recursive Newton Euler Algorithm (RNEA) [12]. Although this strategy avoids the need for the full SO partial derivatives of RNEA, it lacks the opportunity for parallel computation across the trajectory. Full SO partial derivatives calculation, on the other hand, can be easily parallelized to accelerate the backward pass of DDP. Other trajectory optimization schemes aside from DDP may also benefit from the full SO partials.

AD tools depend on forward or reverse chain-rule accumulation and can suffer from high memory requirements [13]. Finite-

difference methods, on the other hand, can be parallelized, but suffer from low accuracy. Such inaccurate Jacobian and Hessian approximations during optimization often lead to ill-conditioning, and poor convergence [10].

Although significant work has been done for computing FO partial derivatives of inverse/forward dynamics [14]–[17], the literature still lacks analytical SO partial derivatives of rigid-body dynamics. This is mainly due to the tensor nature of SO derivatives, and the lack of established tools for working with dynamics tensors. The main contribution of this paper is to extend the FO derivatives of inverse dynamics (ID) presented in Ref. [14] to SO derivatives w.r.t the joint configuration (\mathbf{q}), velocity vector ($\dot{\mathbf{q}}$), and joint acceleration ($\ddot{\mathbf{q}}$) for multi-Degree-of-Freedom (DoF) joints modeled with Lie groups. Our method departs from the chain-rule approach used in [10], and thus gains additional efficiency in the associated algorithm. For this purpose, we contribute an extension of Featherstone’s spatial vector algebra tools [12] for tensor use. The SO derivatives algorithm herein could also be parallelized (e.g., for use with GPUs), building on the FO case in Ref. [18].

In the following sections, Spatial Vector Algebra (SVA) is reviewed for dynamics analysis, followed by an extension of SVA for use with second-order derivative tensors. Then, the algorithm is developed for the SO partial derivatives of ID. The resulting expressions and algorithm are very complicated, but are provided in the form of open-source algorithm, with full derivation in Ref. [19] while keeping the current paper self-contained. The performance of this new SO algorithm is compared to AD using CasADi [20] in MATLAB.

II. RIGID-BODY DYNAMICS BACKGROUND

Rigid-Body Dynamics: For a rigid-body system with n -dimensional configuration manifold \mathcal{Q} , the state variables are the configuration $\mathbf{q} \in \mathcal{Q}$ and the generalized velocity vector $\dot{\mathbf{q}} \in \mathbb{R}^n$, while the control variable is the generalized force/torque vector $\boldsymbol{\tau} \in \mathbb{R}^n$. With this convention, $\dot{\mathbf{q}}$ uniquely specifies the time rate of change for \mathbf{q} without strictly being its time derivative. The Inverse Dynamics (ID), is given by

$$\boldsymbol{\tau} = \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) \quad (1)$$

$$= \text{ID}(\text{model}, \mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}) \quad (2)$$

where $\mathbf{M} \in \mathbb{R}^{n \times n}$ is the mass matrix, $\mathbf{C} \in \mathbb{R}^{n \times n}$ is the Coriolis matrix, and $\mathbf{g} \in \mathbb{R}^n$ is the vector of generalized gravitational forces. An efficient $\mathcal{O}(N)$ algorithm for ID is the RNEA [12], [21].

Notation: Spatial vectors are 6D vectors that combine the linear and angular aspects of a rigid-body motion or net force [12]. Cartesian vectors are denoted with lower-case letters with a bar ($\bar{\mathbf{v}}$), spatial vectors with lower-case bold letters (e.g., \mathbf{a}), matrices with capitalized bold letters (e.g., \mathbf{A}), and tensors with capitalized calligraphic letters (e.g., \mathcal{A}). Motion vectors,

¹ Aerospace Engineering, The University of Texas at Austin, TX-78751, USA. singh281@utexas.edu, ryan.russell@utexas.edu

² Aerospace & Mechanical Engineering, University of Notre Dame, IN-46556, USA. pwensing@nd.edu

This work was supported in part by the National Science Foundation grants CMMI-1835013 and CMMI-1835186.

such as velocity and acceleration, belong to a 6D vector space denoted M^6 . Force-like vectors, such as force and momentum, belong to another 6D vector space F^6 . Spatial vectors are usually expressed in either the ground coordinate frame or a body coordinate (local) frame. For example, the spatial velocity ${}^k\mathbf{v}_k \in M^6$ of a body k expressed in the body frame is ${}^k\mathbf{v}_k = [{}^k\bar{\omega}_k^T \quad {}^k\bar{v}_k^T]^T$ where ${}^k\bar{\omega}_k \in \mathbb{R}^3$ is the angular velocity expressed in a coordinate frame fixed to the body, while ${}^k\bar{v}_k \in \mathbb{R}^3$ is the linear velocity of the origin of the body frame. When the frame used to express a spatial vector is omitted, the ground frame is assumed.

A spatial cross product between motion vectors (\mathbf{v}, \mathbf{u}) , written as $(\mathbf{v} \times) \mathbf{u}$, is given by Eq. 3. This operation gives the time rate of change of \mathbf{u} , when \mathbf{u} is moving with a spatial velocity \mathbf{v} . For a Cartesian vector $\bar{\omega}$, $\bar{\omega} \times$ is the 3D cross-product matrix. A spatial cross product between a motion and a force vector is written as $(\mathbf{v} \times^*) \mathbf{f}$, as defined by Eq. 4.

$$\mathbf{v} \times = \begin{bmatrix} \bar{\omega} \times & \mathbf{0} \\ \bar{v} \times & \bar{\omega} \times \end{bmatrix} \quad (3) \quad \mathbf{v} \times^* = \begin{bmatrix} \bar{\omega} \times & \bar{v} \times \\ \mathbf{0} & \bar{\omega} \times \end{bmatrix} \quad (4)$$

An operator $\bar{\times}^*$ is defined by swapping the order of the cross product, such that $(\mathbf{f} \bar{\times}^*) \mathbf{v} = (\mathbf{v} \times^*) \mathbf{f}$ [22]. Further introduction to SVA is provided in Ref. [12].

Connectivity: An open-chain kinematic tree (Fig. 1) is considered with N links connected by joints, each with up to 6 DoF. Body i 's parent toward the root of the tree is denoted as $\lambda(i)$, and we define $i \preceq j$ if body i is in the path from body j to the root. Joint i is defined as the connection between body i and its predecessor.

We consider joints whose configurations form a sub-group of the Lie group $SE(3)$. For a prismatic joint, the configuration and rate are represented by $\mathbf{q}_i, \dot{\mathbf{q}}_i \in \mathbb{R}$, while for a revolute joint, $\mathbf{q}_i \in SO(2)$, and $\dot{\mathbf{q}}_i \in \mathbb{R}$ gives the rotational rate of the joint. For a spherical joint, $\mathbf{q}_i \in SO(3)$, and $\dot{\mathbf{q}}_i = {}^i\bar{\omega}_{i/\lambda(i)} \in \mathbb{R}^3$ gives relative angular velocity between neighboring bodies. For a 6-DoF free motion joint, $\mathbf{q}_i \in SE(3)$, and $\dot{\mathbf{q}}_i = {}^i\mathbf{v}_{i/\lambda(i)} \in \mathbb{R}^6$.

The spatial velocities of the neighbouring bodies in the tree are then related by the recursive expression $\mathbf{v}_i = \mathbf{v}_{\lambda(i)} + \mathbf{S}_i \dot{\mathbf{q}}_i$, where $\mathbf{S}_i \in \mathbb{R}^{6 \times n_i}$ is the joint motion subspace matrix for joint i [12] with n_i its number of DoFs. The velocity \mathbf{v}_i can also be written in an analytical form as the sum of joint velocities over predecessors as $\mathbf{v}_i = \sum_{l \preceq i} \mathbf{S}_l \dot{\mathbf{q}}_l$. The derivative of the joint motion subspace matrix in local coordinates (often denoted $\dot{\mathbf{S}}_i$ [12]) is assumed to be zero. The quantity $\dot{\mathbf{S}}_i = \mathbf{v}_i \times \mathbf{S}_i$ signifies the rate of change of \mathbf{S}_i due to the local coordinate system moving.

Dynamics: The spatial equation of motion [12] is given for body k as $\mathbf{f}_k = \mathbf{I}_k \mathbf{a}_k + \mathbf{v}_k \times^* \mathbf{I}_k \mathbf{v}_k$, where \mathbf{f}_k is the net spatial force on body k , \mathbf{I}_k is its spatial inertia [12], and \mathbf{a}_k is its spatial acceleration. The development in Ref. [14] presents two additional spatial motion quantities $\dot{\Psi}$, $\ddot{\Psi}$ (Eq. 5) essential to the derivation in this paper. The quantities $\dot{\Psi}_j$ and $\ddot{\Psi}_j$ represent the time-derivative of \mathbf{S}_j , and $\ddot{\Psi}_j$, respectively, due to joint j 's

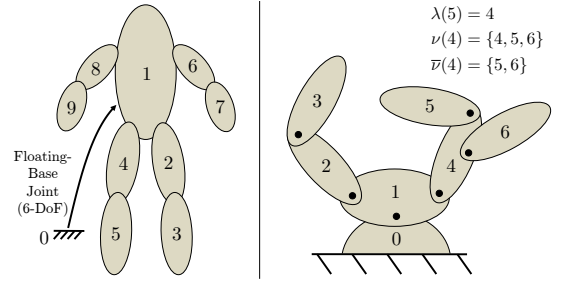


Fig. 1: Convention examples with floating- and fixed-base systems.

predecessor $\lambda(j)$ moving.

$$\begin{aligned} \dot{\Psi}_j &= \mathbf{v}_{\lambda(j)} \times \mathbf{S}_j \\ \ddot{\Psi}_j &= \mathbf{a}_{\lambda(j)} \times \mathbf{S}_j + \mathbf{v}_{\lambda(j)} \times \dot{\Psi}_j \end{aligned} \quad (5)$$

The analytical expressions for the FO partial derivatives of ID w.r.t \mathbf{q} and $\dot{\mathbf{q}}$ [14] are given below, with other similar formulations in [15], [16]. Here τ_i represents the joint torques/forces for joint i from Eq. 1. The quantity $\mathbf{f}_i^C = \sum_{k \succeq i} \mathbf{f}_k$ is the composite spatial force transmitted across joint i , and \mathbf{I}_i^C is the composite rigid-body inertia of the sub-tree rooted at body i , given as $\mathbf{I}_i^C = \sum_{k \succeq i} \mathbf{I}_k$. The quantity \mathbf{B}_k is a body-level Coriolis matrix [14], [22],

$$\mathbf{B}_k = \frac{1}{2}[(\mathbf{v}_k \times^*) \mathbf{I}_k - \mathbf{I}_k (\mathbf{v}_k \times) + (\mathbf{I}_k \mathbf{v}_k) \bar{\times}^*] \quad (6)$$

while \mathbf{B}_i^C is its composite given by $\mathbf{B}_i^C = \sum_{k \succeq i} \mathbf{B}_k$. The next sections extend Eqs. 7-8 for SO derivatives of ID.

$$\frac{\partial \tau_i}{\partial \mathbf{q}_j} = \mathbf{S}_i^T [2\mathbf{B}_i^C] \dot{\Psi}_j + \mathbf{S}_i^T \mathbf{I}_i^C \ddot{\Psi}_j, (j \preceq i) \quad (7a)$$

$$\frac{\partial \tau_j}{\partial \mathbf{q}_i} = \mathbf{S}_j^T [2\mathbf{B}_i^C \dot{\Psi}_i + \mathbf{I}_i^C \ddot{\Psi}_i + (\mathbf{f}_i^C) \bar{\times}^* \mathbf{S}_i], (j \prec i) \quad (7b)$$

$$\frac{\partial \tau_i}{\partial \dot{\mathbf{q}}_j} = \mathbf{S}_i^T [2\mathbf{B}_i^C \mathbf{S}_j + \mathbf{I}_i^C (\dot{\Psi}_j + \dot{\mathbf{S}}_j)], (j \preceq i) \quad (8a)$$

$$\frac{\partial \tau_j}{\partial \dot{\mathbf{q}}_i} = \mathbf{S}_j^T [2\mathbf{B}_i^C \mathbf{S}_i + \mathbf{I}_i^C (\dot{\Psi}_i + \dot{\mathbf{S}}_i)], (j \prec i) \quad (8b)$$

III. EXTENDING SVA FOR TENSORIAL USE

The motion space M^6 [12] is extended to a space of spatial-motion matrices $M^{6 \times n}$, where each column of such a matrix is a usual spatial motion vector. For any $\mathbf{U} \in M^{6 \times n}$ a new spatial cross-product operator $\tilde{\times}$ is considered and defined by applying the usual spatial cross-product operator (\times) to each column of \mathbf{U} . The result is a third-order tensor (Fig. 2) in $\mathbb{R}^{6 \times 6 \times n}$ where each 6×6 matrix in the 1-2 dimension is the original spatial cross-product operator on a column of \mathbf{U} .

Given two spatial motion matrices, $\mathbf{U} \in M^{6 \times n_u}$ and $\mathbf{V} \in M^{6 \times n_v}$, we can now define a cross-product operation between them as $(\mathbf{U} \tilde{\times}) \mathbf{V} \in M^{6 \times n_v \times n_u}$ via a tensor-matrix product. Such an operation, denoted as $\mathcal{Z} = \mathcal{A} \mathbf{B}$ is defined as:

$$\mathcal{Z}_{i,j,k} = \sum_{\ell} \mathcal{A}_{i,\ell,k} B_{\ell,j} \quad (9)$$

for any tensor \mathcal{A} and suitably sized matrix \mathbf{B} . Thus, the k -th page, j -th column of $\mathbf{U} \tilde{\times} \mathbf{V}$ gives the cross product of the k -th column of \mathbf{U} with the j -th column of \mathbf{V} .

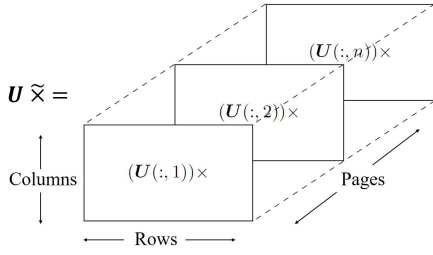


Fig. 2: $\tilde{\times}$ operates on each column of a $U \in M^{6 \times n}$ spatial matrix to create a third-order tensor. Each rectangular box is a 2D matrix

In a similar manner, consider a spatial force matrix $F \in F^{6 \times n_f}$. Defining $(V \tilde{\times}^*)$ in an analogous manner to in Fig. 2 allows taking a cross-product-like operation $V \tilde{\times}^* F$. Again, analogously, we consider a third operator $(F \tilde{\times}^*)$ that provides $V \tilde{\times}^* F = F \tilde{\times}^* V$. In each case, the tilde indicates the spatial-matrix extension of the usual spatial-vector cross products.

For later use, the product of a matrix $B \in \mathbb{R}^{n_1 \times n_2}$, and a tensor $\mathcal{A} \in \mathbb{R}^{n_2 \times n_3 \times n_4}$, likewise results in another tensor, denoted as $\mathcal{Y} = B\mathcal{A}$, and defined as:

$$\mathcal{Y}_{i,j,k} = \sum_{\ell} B_{i,\ell} \mathcal{A}_{\ell,j,k} \quad (10)$$

Two types of tensor rotations are defined for this paper:

- 1) $\mathcal{A}^{\tilde{\top}}$: Transpose along the 1-2 dimension. This operation can also be understood as the usual matrix transpose of each matrix (e.g., in Fig. 2) moving along pages of the tensor. If $\mathcal{A}^{\tilde{\top}} = \mathcal{B}$, then $\mathcal{A}_{i,j,k} = \mathcal{B}_{j,i,k}$.
- 2) $\mathcal{A}^{\tilde{R}}$: Rotation of elements along the 2-3 dimension. If $\mathcal{A}^{\tilde{R}} = \mathcal{B}$, then $\mathcal{A}_{i,j,k} = \mathcal{B}_{i,j,k}$.

Another rotation $(\tilde{R}, \tilde{\top})$ is a combination of (\tilde{R}) followed by $(\tilde{\top})$. For example, if $\mathcal{A}^{\tilde{R}, \tilde{\top}} = \mathcal{B}$, then $\mathcal{A}_{i,j,k} = \mathcal{B}_{k,i,j}$.

Properties of the operators are given in Table I. These properties naturally extend spatial vector properties [12], but with the added book-keeping required from using tensors. For example, the spatial force/vector cross-product operator \times^* satisfies $v \times^* = -v \times^\top$. Property M1 provides the matrix analogy for the spatial matrix operator $\tilde{\times}^*$.

IV. SECOND-ORDER DERIVATIVES OF ID

A. Preliminaries

The SO partial derivative of joint torque/force is also referred to as a dynamics Hessian tensor. Blocks of this rank 3 tensor are written in a form $\frac{\partial^2 \tau_i}{\partial u_j \partial u_k}$, which signifies taking partial derivative of τ_i w.r.t u_j , followed by u_k . The variables u_j and u_k can either be the joint configuration (q_j, q_k) , joint velocity (\dot{q}_j, \dot{q}_k) , or joint acceleration (\ddot{q}_j, \ddot{q}_k) . Many of these second-order partials are zero, limiting the cases to be considered. From Eq. 1, the first-order partial derivative of τ w.r.t \ddot{q} is $\partial \tau / \partial \ddot{q} = M(q)$. Taking subsequent partial derivatives results in $\partial^2 \tau / \partial \ddot{q} \partial \ddot{q} = 0$ and $\partial^2 \tau / \partial \ddot{q} \partial \dot{q} = 0$. However, the cross-derivative w.r.t \ddot{q} and q is non-trivial and equals $\partial M(q) / \partial q$. Garofalo et al. [23] present formulas for the partial derivative of $M(q)$ w.r.t q for multi-DoF Lie group joints. In this work, we re-derive that result using newly developed spatial matrix

M1)	$U \tilde{\times}^* = -(U \tilde{\times})^{\tilde{\top}}$
M2)	$-V^\top (U \tilde{\times}^*) = (U \tilde{\times} V)^{\tilde{\top}}$
M3)	$-V^\top (U \tilde{\times}^*) F = (U \tilde{\times} V)^{\tilde{\top}} F$
M4)	$(U \tilde{\times} v) = -v \times U$
M5)	$U \tilde{\times}^* F = (F \tilde{\times}^* U)^{\tilde{R}}$
M6)	$F \tilde{\times}^* U = (U \tilde{\times}^* F)^{\tilde{R}}$
M7)	$(\lambda U) \tilde{\times} = \lambda (U \tilde{\times})$
M8)	$U \tilde{\times} V = -(V \tilde{\times} U)^{\tilde{R}}$
M9)	$(v \times U) \tilde{\times} = v \times U \tilde{\times} - U \tilde{\times} v \times$
M10)	$(v \times U) \tilde{\times}^* = v \times^* U \tilde{\times}^* - U \tilde{\times}^* v \times^*$
M11)	$(U \tilde{\times}^* v) \tilde{\times}^* = U \tilde{\times}^* v \tilde{\times}^* - v \tilde{\times}^* U \tilde{\times}^*$
M12)	$(U \tilde{\times}^* F)^{\tilde{\top}} = -F^\top (U \tilde{\times})$
M13)	$V^\top (U \tilde{\times}^* F) = (V \tilde{\times} U)^{\tilde{R}, \tilde{\top}} F = (F^\top (V \tilde{\times} U)^{\tilde{R}})^{\tilde{\top}}$
M14)	$v \times^* F = F \tilde{\times}^* v$
M15)	$f \tilde{\times}^* U = U \tilde{\times}^* f$
M16)	$V^\top (U \tilde{\times}^* F)^{\tilde{R}} = [(V \tilde{\times} U)^{\tilde{R}, \tilde{\top}} F]^{\tilde{R}}$
M17)	$V^\top (U \tilde{\times}^* F)^{\tilde{R}} = -[U^\top (V \tilde{\times}^* F)^{\tilde{R}}]^{\tilde{\top}}$
M18)	$V^\top (U \tilde{\times}^* F)^{\tilde{R}} = [V^\top (U \tilde{\times}^* F)]^{\tilde{R}}$
M19)	$(B\mathcal{Y})^{\tilde{\top}} = \mathcal{Y}^{\tilde{\top}} B^\top$

TABLE I: Spatial Matrix Algebra Identities: $v \in M^6, f \in F^6, U \in M^{6 \times n}, F \in F^{6 \times m}, V \in M^{6 \times l}, B \in \mathbb{R}^{n_1 \times n_2}, \mathcal{Y} \in \mathbb{R}^{n_2 \times n_3 \times n_4}$.

operators and contribute new analytical SO partial derivatives of ID w.r.t q and \dot{q} .

For single-DoF joints, each block of the Hessian tensor $\frac{\partial^2 \tau_i}{\partial u_j \partial u_k}$ is a scalar representing a conventional SO derivative w.r.t joint angles, rates, or accelerations. In this case, the order of u_j and u_k doesn't matter. This operation becomes more nuanced when considering derivatives w.r.t configuration for a multi-DoF joint, wherein we define the operator $\frac{\partial}{\partial q_k}$ to represent a collection of Lie derivatives, as in [14]. For example, $\frac{\partial \tau_i}{\partial q_k}$ is defined as the $n_i \times n_k$ matrix where each column gives the derivative of $\tau_i \in \mathbb{R}^{n_i}$ w.r.t changes in configuration along one of the n_k free modes of joint k (see [14] for detail). When the Lie derivatives along the free modes of a joint do not commute, $\frac{\partial^2 \tau_i}{\partial q_k \partial q_k}$ can lack the usual symmetry properties. Such a case occurs, for example, with a spherical joint, since rotations do not commute. To obtain the partial derivatives of spatial quantities embedded in Eq. 7-8, some identities (App. A) are derived. These are an extension to ones defined in Ref. [14], but use the newly developed spatial matrix operators from Sec. III.

For example, identity K1 (App. A) is an extension of identity J1 in Ref. [14]. The identity J1 (Eq. 11) gives the directional derivative of the joint motion sub-space matrix S_i w.r.t the p^{th} DoF of a previous joint j in the connectivity tree:

$$\frac{\partial S_i}{\partial q_{j,p}} = s_{j,p} \times S_i \quad (j \preceq i) \quad (11)$$

where $s_{j,p}$ is the p -th column of S_j . On the other hand, K1 uses the $\tilde{\times}$ operator to extend it to the partial derivative of S_i w.r.t the full joint configuration q_j to give the tensor $\frac{\partial S_i}{\partial q_j}$ as:

$$\frac{\partial S_i}{\partial q_j} = S_j \tilde{\times} S_i \quad (j \preceq i) \quad (12)$$

The identities K4 and K9 describe the partial derivatives of $\dot{\Psi}_i$ and $\ddot{\Psi}_i$ present in Eq. 7-8. Identities K6, K10, and K12 give the partial derivatives of the composite Inertia (I_i^C), body-level Coriolis matrix (B_i^C), and net composite spatial force on a body (f_i^C). Individual partial derivatives of these quantities allow us to use the plug-and-play approach to simplify the algebra needed for SO partial derivatives of ID.

To calculate the SO partial derivatives, we take subsequent partial derivatives of the terms $\frac{\partial \tau_i}{\partial q_j}$, $\frac{\partial \tau_i}{\partial q_i}$, $\frac{\partial \tau_i}{\partial q_j}$, and $\frac{\partial \tau_i}{\partial q_i}$ w.r.t joint configuration (q), and joint velocity (\dot{q}) for joint k . We consider three cases, by changing the order of index k as:

Case A: $k \preceq j \preceq i$ **Case B:** $j \prec k \preceq i$ **Case C:** $j \preceq i \prec k$

The sections below outline the approach for calculating the SO partial derivatives. Only some cases are shown to illustrate the main idea, with a detailed summary of all the cases in App. B, and full step-by-step derivations in Ref. [19].

B. Second-order partial derivatives w.r.t q

For SO partials of ID w.r.t q , we take the partial derivatives of Eq. 7a, and 7b w.r.t q_k for cases A, B, and C mentioned above. Although, the symmetric blocks in the Hessian allow us to re-use three of those six cases. For any of the cases, the partial derivatives of Eq. 7a and 7b can be taken, as long as the accompanying conditions on the equations are met. For example, for Case B ($j \prec k \preceq i$), since $j \prec i$, only Eq. 7b can be used. A derivation for Case C ($j \preceq i \prec k$) is shown here as an example. We take the partial derivative of Eq. 7a w.r.t q_k . Applying the product rule, and using the identities K4, K9, and K13 as:

$$\frac{\partial^2 \tau_i}{\partial q_j \partial q_k} = 2S_i^\top \left(\frac{\partial B_i^C}{\partial q_k} \dot{\Psi}_j \right) + S_i^\top \left(\frac{\partial I_i^C}{\partial q_k} \ddot{\Psi}_j \right) \quad (13)$$

The quantities $\frac{\partial B_i^C}{\partial q_k}$ and $\frac{\partial I_i^C}{\partial q_k}$ are third-order tensors where the partial derivatives of matrices B_i^C and I_i^C w.r.t each DoF of joint k are stacked as matrices along the pages of the tensor. Using the identities K6 and K10 then gives:

$$\begin{aligned} \frac{\partial^2 \tau_i}{\partial q_j \partial q_k} = & 2S_i^\top \left(B_k^C [\dot{\Psi}_k] + S_k \tilde{\times}^* B_k^C - B_k^C (S_k \tilde{\times}) \right) \dot{\Psi}_j \\ & + S_i^\top \left(S_k \tilde{\times}^* I_k^C - I_k^C (S_k \tilde{\times}) \right) \ddot{\Psi}_j \end{aligned} \quad (14)$$

where the tensor $B_k^C [\dot{\Psi}_k]$ is a composite calculated for the sub-tree as $B_k^C [\dot{\Psi}_k] = \sum_{l \succeq k} B_l [\dot{\Psi}_k]$, with:

$$B_l [\dot{\Psi}_k] = \frac{1}{2} [(\dot{\Psi}_k \tilde{\times}^*) I_l - I_l (\dot{\Psi}_k \tilde{\times}) + (I_l \dot{\Psi}_k) \tilde{\times}^*] \quad (15)$$

Eq. 15 is a tensor extension of the body-level Coriolis matrix (Eq. 6) with a spatial matrix argument. Expressions for other cases are in App. B with full derivation in Ref. [19, Sec. IV].

C. Cross Second Order Partial derivatives w.r.t \dot{q} and q

As explained before, the cross-SO partial derivatives of ID w.r.t \dot{q} and q results in $\frac{\partial M}{\partial q}$. The lower-triangle of the mass matrix $M(q)$ for the case $j \preceq i$ is given as [12]:

$$M_{ji} = S_j^\top I_i^C S_i \quad (16)$$

Since $M(q)$ is symmetric [12], $M_{ij} = M_{ji}^\top$. We apply the three cases A, B, and C discussed before. As an example, for Case B ($j \prec k \preceq i$), we take the partial derivative of M_{ji} w.r.t q_k and use the product rule, along with identity K13 as:

$$\frac{\partial M_{ji}}{\partial q_k} = S_j^\top \left(\frac{\partial I_i^C}{\partial q_k} S_i + I_i^C \frac{\partial S_i}{\partial q_k} \right) \quad (17)$$

Using with identities K1 and K6, and canceling terms gives:

$$\frac{\partial M_{ji}}{\partial q_k} = S_j^\top (S_k \tilde{\times}^* I_i^C) S_i \quad (18)$$

Expressions for other cases are listed in App. B, with details of derivation at Ref. [19, Sec. VII]

D. Second-order partial derivatives involving \dot{q}

For SO partial derivatives w.r.t \dot{q} , we take the partial derivatives of Eq. 8a and 8b w.r.t \dot{q}_k . A list of expressions is given in App. B, with full derivation in Ref. [19, Sec. V].

For cross-SO partial derivatives of ID, we take the partial derivative of Eqs. 8a-8b w.r.t q_k to get $\frac{\partial^2 \tau}{\partial \dot{q} \partial q}$. The three cases A, B, and C (Sec. IV-A) for Eq. 8a-8b result in six expressions, which are then also used for the symmetric term $\frac{\partial^2 \tau}{\partial q \partial \dot{q}}$ as:

$$\frac{\partial^2 \tau}{\partial q \partial \dot{q}} = \left[\frac{\partial^2 \tau}{\partial \dot{q} \partial q} \right] \tilde{R} \quad (19)$$

Here, we solve all the three cases A, B and C for both Eq. 8a and Eq. 8b. Pertaining to Case A ($k \preceq j \preceq i$), since $j \preceq i$, Eq. 8a can be safely used to get $\frac{\partial^2 \tau_i}{\partial \dot{q}_j \partial q_k}$. However, the $j \neq i$ requirement on Eq. 8b constrains the condition in Case A to $k \preceq j \prec i$. Similarly, for Case C ($j \preceq i \prec k$), taking partial derivative of Eq. 8b results in a stricter case $j \prec i \prec k$. Appendix B lists the six expressions with full derivation in Ref. [19, Sec. VI].

V. EFFICIENT IMPLEMENTATION AND ALGORITHM

For efficient implementation of the algorithm, all the cases are converted to an index order of $k \preceq j \preceq i$. This notation is explained with the help of following two examples.

Example 1: In Eq. 14, we first we switch the indices k and j , followed by j and i to get $\frac{\partial^2 \tau_j}{\partial q_k \partial q_i}$ as

$$\begin{aligned} \frac{\partial^2 \tau_j}{\partial q_k \partial q_i} = & 2S_j^\top \left(B_i^C [\dot{\Psi}_i] + S_i \tilde{\times}^* B_i^C - B_i^C (S_i \tilde{\times}) \right) \dot{\Psi}_k \\ & + S_j^\top \left(S_i \tilde{\times}^* I_i^C - I_i^C (S_i \tilde{\times}) \right) \ddot{\Psi}_k \end{aligned} \quad (20)$$

For the term $\frac{\partial^2 \tau_j}{\partial q_i \partial q_k}$, when $k \neq i$, the symmetry property of Hessian blocks can be exploited:

$$\frac{\partial^2 \tau_j}{\partial q_i \partial q_k} = \left[\frac{\partial^2 \tau_j}{\partial q_k \partial q_i} \right] \tilde{R}, (k \preceq j \prec i) \quad (21)$$

The 2-3 tensor rotation in Eq. 21 occurs due to symmetry along the 2nd and 3rd dimensions.

Example 2: In Eq. 18, switching indices k and j leads to the index order $k \prec j \preceq i$. Using property M5 leads to:

$$\frac{\partial M_{ki}}{\partial q_j} = S_k^\top ((I_i^C S_i) \tilde{\times}^* S_j) \tilde{R} \quad (22)$$

Symmetry of $M(q)$ gives us $\frac{\partial M_{ik}}{\partial q_j}$ as:

$$\frac{\partial M_{ik}}{\partial q_j} = \left[\frac{\partial M_{ki}}{\partial q_j} \right]^\top \quad (23)$$

In this case, since the symmetry is along the 1st and the 2nd dimension of $\frac{\partial M_{ki}}{\partial q_j}$, the tensor 1-2 rotation takes place.

The expressions for SO partials of ID (App. B) are first reduced to matrix and vector form to avoid tensor operations. This refactoring is due, in part, to a lack of stable tensor support in the C++ Eigen library that is often used in robotics dynamics libraries. This reduction is achieved by considering the expressions for single DoF of joints i , j , and k , one at a time. We explain this process with the help of two examples.

Example 1: Considering the case $k \prec j \preceq i$ the expression

$$\frac{\partial^2 \tau_i}{\partial \dot{q}_j \partial \dot{q}_k} = -[S_j^\top (2B_i^C[S_i]S_k)\tilde{R}]^\top$$

is studied for the p^{th} , t^{th} , and r^{th} DoFs of the joints i , j and k , respectively. The tensor term $B_i^C[S_i]$ (defined by Eq. 15) reduces to a matrix $B_i^C[s_{i,p}]$, where $s_{i,p}$ is p -th column of S_i . This term represents the value B_i^C would take if all bodies in the subtree at i moved with velocity $s_{i,p}$. The above reduction enables dropping the 3D tensor rotation (\tilde{R}). The products of $B_i^C[s_{i,p}]$ with column vectors $s_{j,t}$ and $s_{k,r}$ then provides a scalar, resulting in dropping the rotation (\tilde{R}) from above:

$$\frac{\partial^2 \tau_{i,p}}{\partial \dot{q}_{j,t} \partial \dot{q}_{k,r}} = -2s_{j,t}^\top (B_i^C[s_{i,p}]s_{k,r}) \quad (24)$$

Example 2: The term $\frac{\partial^2 \tau_i}{\partial \dot{q}_k \partial \dot{q}_j} = \left[\frac{\partial^2 \tau_i}{\partial \dot{q}_j \partial \dot{q}_k} \right]^\top$ is evaluated for the p^{th} , t^{th} , and r^{th} DoFs of the joints i , j and k , respectively. In this case, the 2-3 tensor rotation (\tilde{R}) drops out, since the resulting expression is a scalar.

$$\frac{\partial^2 \tau_{i,p}}{\partial \dot{q}_{k,r} \partial \dot{q}_{j,t}} = \frac{\partial^2 \tau_{i,p}}{\partial \dot{q}_{j,t} \partial \dot{q}_{k,r}} \quad (25)$$

Algorithm 1 (IDSVA SO) is detailed in App. B and returns all the SO partials from Sec. IV. It is implemented with all kinematic and dynamic quantities represented in the ground frame. The forward pass in Alg. 1 (Lines 2-11) solves for kinematic and dynamic quantities like a_i , B_i^C , f_i^C , $\dot{\Psi}_i$, and $\ddot{\Psi}_i$ for the entire tree. The quantity $\frac{1}{2}$ is skipped from the definition of B_i^C to make algebra simpler, and necessary adjustments are made in the algorithm. The backward pass then cycles from leaves to root of the tree and consists of three main nested loops, each one for bodies i , j , and k . These three nested loops also consist of a nested loop for each DoF of joint i , j , and k . The indices p (from 1 to n_i , Line 13), t (from 1 to n_j , Line 27), and r (from 1 to n_k , Line 39) cycle over all of the DoFs of joints i , j , and k respectively. Some of the intermediate quantities in the algorithm are defined for a DoF of a joint. For example, in Line 14, s_p is simply the p^{th} column of S_i . Similar quantities for joint j and k are defined in Line 28 and 40 respectively.

The backward pass (Lines 13-74) cycles n times in total (i.e., over all joints), with the nested loops (Lines 27-71 and 39-68)

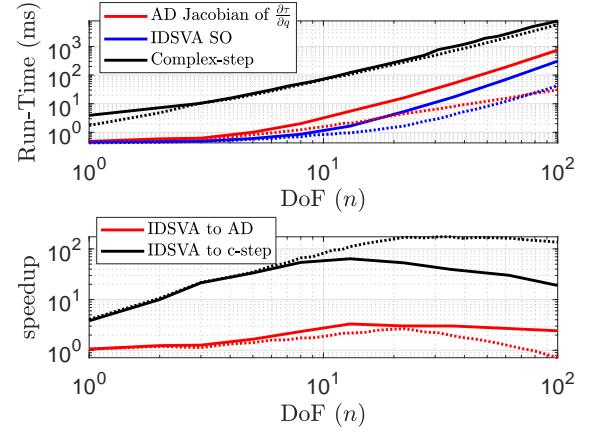


Fig. 3: a) IDSVA outperforms the AD & complex-step approach for all serial (bold)/branched (dashed) chains in the CasADi [20] virtual machine when $N < 70$ b) speedup of IDSVA over AD & complex-step for serial and branched $bf = 2$ chains.

each executing at most $6d$ times per cycle of its parent loop. Thus, the total computational complexity is $O(Nd^2)$.

While the algorithm is complex, an open-source MATLAB version of it can be found at [24], and is integrated with Featherstone's `spatial` v2 library [12]

VI. ACCURACY AND PERFORMANCE

A complex-step method [25] was used to calculate the SO partials of ID and verify the accuracy for the proposed algorithm. The complex-step approach was applied to FO derivatives of ID [14] and verified derivatives accurate to machine precision.

For run-time comparison, the automatic differentiation tool CasADi [20] in MATLAB was used. AD was used to take the Jacobian of $[\frac{\partial \tau}{\partial q}, \frac{\partial \tau}{\partial \dot{q}}]^\top$, via its application to the algorithm presented in Ref. [14] for the FO derivatives. Since CasADi is not compatible with functions defined on a Lie group, systems with single DoF revolute joints were considered. Fig. 3 shows a comparison of IDSVA with the AD and complex-step approach for serial and branched chains with a branching factor bf [12]. For serial chains, IDSVA outperforms AD for all N , with speedups between 1.5 and $3\times$ for models with $N > 5$. For branched chains, the AD computational graph is highly efficient, resulting in performance gains beyond a critical N . For $bf = 2$ chains, this critical N lies at $N = 70$. The complex-step method is accurate but slow in run-time, as seen from the plot in Fig. 3.

A preliminary run-time analysis was also performed with an implementation, available at [26], extending the Pinocchio [27] open-source library. Fig. 4 gives run-time numbers (in μs) for several fixed/floating base models for the IDSVA SO (1) algorithm, implemented in C/C++ within the Pinocchio framework [27]. For reference, the run-times for RNEA [12], and the IDSVA FO algorithm given in [14] are also provided. From Fig. 4, the ratio of run-times for SO to FO derivatives increases with N due to the algorithm complexity ratio of d between the two algorithms. All computations were performed on an Intel (R) 12th Gen i5-12400 CPU with 2.5 Ghz, with

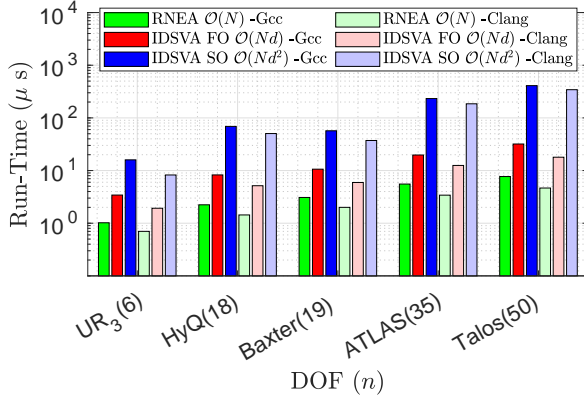


Fig. 4: Run-time comparison between Pinocchio [27] C++ implementation of RNEA [12], IDSA FO [14], and IDSA SO using GCC 9.4 (dark), Clang 10.0 (light) compilers. Fixed base- UR₃, Baxter. Floating base- HyQ, ATLAS, Talos.

turbo boost off. From Fig. 4, SO partial derivatives of a floating base 18-DoF HyQ quadruped model take 51 μ s in the C/C++ implementation.

VII. CONCLUSIONS

In this paper, SO partial derivatives of rigid-body inverse dynamics w.r.t \mathbf{q} , $\dot{\mathbf{q}}$, and $\ddot{\mathbf{q}}$ were derived using Spatial Vector Algebra (SVA) for models with multi-DoF Lie group joints. SVA was extended for spatial matrices to enable tensor operations required for the SO derivatives. This extension was done using three cross-product operators for spatial matrices. An efficient recursive algorithm was also developed to calculate the SO partial derivatives of ID by exploiting common expressions and the structure of the connectivity tree. A MATLAB run-time comparison with AD using CasADi shows a speedup between 1.5-3 \times for serial chains with $N > 5$. Future work will focus on closed-chain structures, more efficient C/C++ implementation, and further analysis considering code-generation and AD strategies in C/C++.

APPENDIX A: MULTI-DOF JOINT IDENTITIES

The following spatial vector/matrix identities are derived by taking the partial derivative w.r.t the full joint configuration (\mathbf{q}_j), or joint velocity ($\dot{\mathbf{q}}_j$) of the joint j . In each, the quantity to the left of the equals sign equals the expression to the right if $j \preceq i$, and is zero otherwise, unless otherwise stated.

$$\frac{\partial \mathbf{S}_i}{\partial \mathbf{q}_j} = \mathbf{S}_j \tilde{\times} \mathbf{S}_i \quad (\text{K1})$$

$$\frac{\partial \dot{\mathbf{S}}_i}{\partial \mathbf{q}_j} = \dot{\mathbf{S}}_j \tilde{\times} \mathbf{S}_i + \mathbf{S}_j \tilde{\times} \dot{\mathbf{S}}_i \quad (\text{K2})$$

$$\frac{\partial (\mathbf{S}_i \dot{\mathbf{q}}_i \times \mathbf{S}_i)}{\partial \mathbf{q}_j} = \mathbf{S}_j \tilde{\times} (\mathbf{S}_i \dot{\mathbf{q}}_i \times \mathbf{S}_i) \quad (\text{K3})$$

$$\frac{\partial \ddot{\mathbf{S}}_i}{\partial \mathbf{q}_j} = \ddot{\mathbf{S}}_j \tilde{\times} \mathbf{S}_i + \mathbf{S}_j \tilde{\times} \ddot{\mathbf{S}}_i \quad (\text{K4})$$

$$\frac{\partial \mathbf{I}_i}{\partial \mathbf{q}_j} = \mathbf{S}_j \tilde{\times}^* \mathbf{I}_i - \mathbf{I}_i (\mathbf{S}_j \tilde{\times}) \quad (\text{K5})$$

$$\frac{\partial \mathbf{I}_i^C}{\partial \mathbf{q}_j} = \begin{cases} \mathbf{S}_j \tilde{\times}^* \mathbf{I}_i^C - \mathbf{I}_i^C (\mathbf{S}_j \tilde{\times}), & \text{if } j \preceq i \\ \mathbf{S}_j \tilde{\times}^* \mathbf{I}_j^C - \mathbf{I}_j^C (\mathbf{S}_j \tilde{\times}), & \text{if } j \succ i \end{cases} \quad (\text{K6})$$

$$\frac{\partial \mathbf{a}_i}{\partial \mathbf{q}_j} = \ddot{\mathbf{S}}_j - \mathbf{v}_i \times \dot{\mathbf{S}}_j - \mathbf{a}_i \times \mathbf{S}_j \quad (\text{K7})$$

$$\frac{\partial (\mathbf{I}_i \mathbf{a}_i)}{\partial \mathbf{q}_j} = \mathbf{S}_j \tilde{\times}^* (\mathbf{I}_i \mathbf{a}_i) + \mathbf{I}_i \ddot{\mathbf{S}}_j - \mathbf{I}_i (\mathbf{v}_i \times \dot{\mathbf{S}}_j) \quad (\text{K8})$$

$$\frac{\partial \ddot{\mathbf{S}}_i}{\partial \mathbf{q}_j} = \ddot{\mathbf{S}}_j \tilde{\times} \mathbf{S}_i + 2\dot{\mathbf{S}}_j \tilde{\times} \dot{\mathbf{S}}_i + \mathbf{S}_j \tilde{\times} \ddot{\mathbf{S}}_i \quad (\text{K9})$$

$$\frac{\partial \mathbf{B}_i^C}{\partial \mathbf{q}_j} = \begin{cases} \mathcal{B}_i^C [\dot{\mathbf{S}}_j] + \mathbf{S}_j \tilde{\times}^* \mathbf{B}_i^C - \mathbf{B}_i^C (\mathbf{S}_j \tilde{\times}), & \text{if } j \preceq i \\ \mathcal{B}_j^C [\dot{\mathbf{S}}_j] + \mathbf{S}_j \tilde{\times}^* \mathbf{B}_j^C - \mathbf{B}_j^C (\mathbf{S}_j \tilde{\times}), & \text{if } j \succ i \end{cases} \quad (\text{K10})$$

$$\frac{\partial \mathbf{f}_i}{\partial \mathbf{q}_j} = \mathbf{I}_i \ddot{\mathbf{S}}_j + \mathbf{S}_j \tilde{\times}^* \mathbf{f}_i + 2\mathbf{B}_i \dot{\mathbf{S}}_j \quad (\text{K11})$$

$$\frac{\partial \mathbf{f}_i^C}{\partial \mathbf{q}_j} = \begin{cases} \mathbf{I}_i^C \ddot{\mathbf{S}}_j + \mathbf{S}_j \tilde{\times}^* \mathbf{f}_i^C + 2\mathbf{B}_i^C \dot{\mathbf{S}}_j, & \text{if } j \preceq i \\ \mathbf{I}_j^C \ddot{\mathbf{S}}_j + \mathbf{S}_j \tilde{\times}^* \mathbf{f}_j^C + 2\mathbf{B}_j^C \dot{\mathbf{S}}_j, & \text{if } j \succ i \end{cases} \quad (\text{K12})$$

$$\frac{\partial \mathbf{S}_i^\top}{\partial \mathbf{q}_j} = -\mathbf{S}_i^\top \mathbf{S}_j \tilde{\times}^* \quad (\text{K13})$$

$$\frac{\partial \dot{\mathbf{S}}_i}{\partial \dot{\mathbf{q}}_j} = \mathbf{S}_j \tilde{\times} \mathbf{S}_i \quad (\text{K14})$$

$$\frac{\partial \ddot{\mathbf{S}}_i}{\partial \dot{\mathbf{q}}_j} = \begin{cases} \mathbf{S}_j \tilde{\times} \mathbf{S}_i, & \text{if } j \prec i \\ 0, & \text{otherwise} \end{cases} \quad (\text{K15})$$

$$\frac{\partial \mathbf{B}_i^C}{\partial \dot{\mathbf{q}}_j} = \begin{cases} \mathcal{B}_i^C [\mathbf{S}_j], & \text{if } j \preceq i \\ \mathcal{B}_j^C [\mathbf{S}_j], & \text{if } j \succ i \end{cases} \quad (\text{K16})$$

APPENDIX B: SUMMARY

Common Terms:

$$\mathbf{A}_1 \triangleq \mathbf{S}_i \tilde{\times}^* \mathbf{B}_i^C - \mathbf{B}_i^C \mathbf{S}_i \tilde{\times}$$

$$\mathbf{A}_2 \triangleq \mathbf{S}_i \tilde{\times}^* \mathbf{I}_i^C - \mathbf{I}_i^C \mathbf{S}_i \tilde{\times}$$

SO Partial derivatives w.r.t \mathbf{q} :

$$\frac{\partial^2 \boldsymbol{\tau}_i}{\partial \mathbf{q}_j \partial \mathbf{q}_k} = - \left[\dot{\mathbf{S}}_j^\top [2\mathcal{B}_i^C [\mathbf{S}_i] \dot{\mathbf{S}}_k] \tilde{\mathbf{R}} + 2\mathbf{S}_j^\top ((\mathbf{B}_i^C \mathbf{S}_i) \tilde{\times}^* \dot{\mathbf{S}}_k) \tilde{\mathbf{R}} + \mathbf{S}_j^\top ((\mathbf{I}_i^C \mathbf{S}_i) \tilde{\times}^* \ddot{\mathbf{S}}_k) \tilde{\mathbf{R}} \right]^\top, (k \preceq j \preceq i)$$

$$\frac{\partial^2 \boldsymbol{\tau}_i}{\partial \mathbf{q}_k \partial \mathbf{q}_j} = \left[\frac{\partial^2 \boldsymbol{\tau}_i}{\partial \mathbf{q}_j \partial \mathbf{q}_k} \right]^\top, (k \prec j \preceq i)$$

$$\frac{\partial^2 \boldsymbol{\tau}_k}{\partial \mathbf{q}_i \partial \mathbf{q}_j} = \mathbf{S}_k^\top \left([2(\mathcal{B}_i^C [\dot{\mathbf{S}}_i] + \mathbf{A}_1) \dot{\mathbf{S}}_j + \mathbf{A}_2 \ddot{\mathbf{S}}_j] \tilde{\mathbf{R}} + \mathbf{S}_j \tilde{\times}^* (2\mathbf{B}_i^C \dot{\mathbf{S}}_i + \mathbf{I}_i^C \ddot{\mathbf{S}}_i + \mathbf{f}_i^C \tilde{\times}^* \mathbf{S}_i) \right), (k \prec j \preceq i)$$

$$\frac{\partial^2 \boldsymbol{\tau}_k}{\partial \mathbf{q}_j \partial \mathbf{q}_i} = \left[\frac{\partial^2 \boldsymbol{\tau}_k}{\partial \mathbf{q}_i \partial \mathbf{q}_j} \right]^\top, (k \prec j \prec i)$$

$$\frac{\partial^2 \boldsymbol{\tau}_j}{\partial \mathbf{q}_k \partial \mathbf{q}_i} = \mathbf{S}_j^\top (2(\mathcal{B}_i^C [\dot{\mathbf{S}}_i] + \mathbf{A}_1) \dot{\mathbf{S}}_k + \mathbf{A}_2 \ddot{\mathbf{S}}_k), (k \preceq j \prec i)$$

$$\frac{\partial^2 \boldsymbol{\tau}_j}{\partial \mathbf{q}_i \partial \mathbf{q}_k} = \left[\frac{\partial^2 \boldsymbol{\tau}_j}{\partial \mathbf{q}_k \partial \mathbf{q}_i} \right]^\top, (k \preceq j \prec i)$$

SO Partial derivatives w.r.t $\dot{\mathbf{q}}$:

$$\frac{\partial^2 \boldsymbol{\tau}_i}{\partial \dot{\mathbf{q}}_j \partial \dot{\mathbf{q}}_k} = -[\mathbf{S}_j^\top (2\mathcal{B}_i^C [\mathbf{S}_i] \mathbf{S}_k) \tilde{\mathbf{R}}]^\top, (k \prec j \preceq i)$$

$$\frac{\partial^2 \boldsymbol{\tau}_i}{\partial \dot{\mathbf{q}}_k \partial \dot{\mathbf{q}}_j} = \left[\frac{\partial^2 \boldsymbol{\tau}_i}{\partial \dot{\mathbf{q}}_j \partial \dot{\mathbf{q}}_k} \right]^\top, (k \prec j \preceq i)$$

$$\frac{\partial^2 \boldsymbol{\tau}_i}{\partial \dot{\mathbf{q}}_j \partial \dot{\mathbf{q}}_k} = -[\mathbf{S}_j^\top (\mathbf{A}_2 \mathbf{S}_k) \tilde{\mathbf{R}}]^\top, (k = j \preceq i)$$

$$\begin{aligned}
\frac{\partial^2 \tau_k}{\partial \dot{q}_i \partial \dot{q}_j} &= \mathbf{S}_k^\top \left[2\mathbf{B}_i^C [\mathbf{S}_i] \mathbf{S}_j \right]^{\tilde{\mathbf{R}}}, (k < j < i) \\
\frac{\partial^2 \tau_k}{\partial \dot{q}_j \partial \dot{q}_i} &= \left[\frac{\partial^2 \tau_k}{\partial \dot{q}_i \partial \dot{q}_j} \right]^{\tilde{\mathbf{R}}}, (k < j < i) \\
\frac{\partial^2 \tau_k}{\partial \dot{q}_i \partial \dot{q}_j} &= \mathbf{S}_k^\top \left[((\mathbf{I}_i^C \mathbf{S}_i) \tilde{\mathbf{x}}^* + \mathbf{S}_i \tilde{\mathbf{x}}^* \mathbf{I}_i^C) \mathbf{S}_j \right]^{\tilde{\mathbf{R}}}, (k < j = i) \\
\frac{\partial^2 \tau_j}{\partial \dot{q}_k \partial \dot{q}_i} &= \mathbf{S}_j^\top \left[2\mathbf{B}_i^C [\mathbf{S}_i] \mathbf{S}_k \right], (k \leq j < i) \\
\frac{\partial^2 \tau_j}{\partial \dot{q}_i \partial \dot{q}_k} &= \left[\frac{\partial^2 \tau_j}{\partial \dot{q}_k \partial \dot{q}_i} \right]^{\tilde{\mathbf{R}}}, (k \leq j < i)
\end{aligned}$$

Cross SO Partial w.r.t q and \dot{q} :

$$\begin{aligned}
\frac{\partial^2 \tau_i}{\partial \dot{q}_j \partial \dot{q}_k} &= -[\mathbf{S}_j^\top (2\mathbf{B}_i^C [\mathbf{S}_i] \dot{\Psi}_k)^{\tilde{\mathbf{R}}}]^{\tilde{\mathbf{T}}}, (k \leq j \leq i) \\
\frac{\partial^2 \tau_j}{\partial \dot{q}_i \partial \dot{q}_k} &= \mathbf{S}_j^\top \left[2\mathbf{B}_i^C [\mathbf{S}_i] \dot{\Psi}_k \right]^{\tilde{\mathbf{R}}}, (k \leq j < i) \\
\frac{\partial^2 \tau_i}{\partial \dot{q}_k \partial \dot{q}_j} &= [\mathbf{S}_k^\top (-2\mathbf{B}_i^C [\mathbf{S}_i] \dot{\Psi}_j + (2\mathbf{B}_i^{C^\top} \mathbf{S}_i) \tilde{\mathbf{x}}^* \mathbf{S}_j \\
&\quad + 2(\mathbf{I}_i^C \mathbf{S}_i) \tilde{\mathbf{x}}^* \dot{\Psi}_j)^{\tilde{\mathbf{R}}} + (\dot{\Psi}_k + \dot{\mathbf{S}}_k)^\top ((\mathbf{I}_i^C \mathbf{S}_i) \tilde{\mathbf{x}}^* \dot{\Psi}_j)^{\tilde{\mathbf{R}}}]^{\tilde{\mathbf{T}}}, (k < j \leq i) \\
\frac{\partial^2 \tau_k}{\partial \dot{q}_i \partial \dot{q}_j} &= \mathbf{S}_k^\top \left[(2\mathbf{B}_i^C \mathbf{S}_i + \mathbf{I}_i^C (\dot{\Psi}_i + \dot{\mathbf{S}}_i)) \tilde{\mathbf{x}}^* \mathbf{S}_j + \right. \\
&\quad \left. 2\mathbf{B}_i^C [\mathbf{S}_i] \dot{\Psi}_j \right]^{\tilde{\mathbf{R}}}, (k < j \leq i) \\
\frac{\partial^2 \tau_j}{\partial \dot{q}_k \partial \dot{q}_i} &= \mathbf{S}_j^\top (2(\mathbf{B}_i^C [\dot{\Psi}_i] + \mathbf{A}_1) \mathbf{S}_k + \mathbf{A}_2 (\dot{\Psi}_k + \dot{\mathbf{S}}_k)), \\
&\quad (k \leq j < i) \\
\frac{\partial^2 \tau_k}{\partial \dot{q}_j \partial \dot{q}_i} &= \mathbf{S}_k^\top (2(\mathbf{B}_i^C [\dot{\Psi}_i] + \mathbf{A}_1) \mathbf{S}_j + \mathbf{A}_2 (\dot{\Psi}_j + \dot{\mathbf{S}}_j)), \\
&\quad (k < j < i)
\end{aligned}$$

FO Partial of $M(q)$ w.r.t q :

$$\begin{aligned}
\frac{\partial M_{ji}}{\partial q_k} &= \frac{\partial M_{ij}}{\partial q_k} = 0, (k \leq j \leq i) \\
\frac{\partial M_{ki}}{\partial q_j} &= \mathbf{S}_k^\top ((\mathbf{I}_i^C \mathbf{S}_i) \tilde{\mathbf{x}}^* \mathbf{S}_j)^{\tilde{\mathbf{R}}}, (k < j \leq i) \\
\frac{\partial M_{ik}}{\partial q_j} &= \left[\frac{\partial M_{ki}}{\partial q_j} \right]^{\tilde{\mathbf{T}}}, (k < j \leq i) \\
\frac{\partial M_{kj}}{\partial q_i} &= \mathbf{S}_k^\top \mathbf{A}_2 \mathbf{S}_j, (k \leq j < i) \\
\frac{\partial M_{jk}}{\partial q_i} &= \left[\frac{\partial M_{kj}}{\partial q_i} \right]^{\tilde{\mathbf{T}}}, (k \leq j < i)
\end{aligned}$$

REFERENCES

- [1] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2012, pp. 4906–4913.
- [2] J. Koenemann, A. Del Prete, Y. Tassa, E. Todorov, O. Stasse, M. Bennewitz, and N. Mansard, "Whole-body model-predictive control applied to the hrp-2 humanoid," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2015, pp. 3346–3351.
- [3] D. Mayne, "A second-order gradient method for determining optimal trajectories of non-linear discrete-time systems," *Int. J. of Control*, vol. 3, no. 1, pp. 85–95, 1966.

- [4] Y. Tassa, N. Mansard, and E. Todorov, "Control-limited differential dynamic programming," in *IEEE Int. Conf. on Robotics and Automation*, 2014, pp. 1168–1175.
- [5] I. Chatzinikolaïdis and Z. Li, "Trajectory optimization of contact-rich motions using implicit differential dynamic programming," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2626–2633, 2021.
- [6] C. Mastalli et al., "Crocodyl: An efficient and versatile framework for multi-contact optimal control," in *IEEE Int. Conf. on Robotics and Automation*, 2020, pp. 2536–2542.
- [7] H. Li and P. M. Wensing, "Hybrid systems differential dynamic programming for whole-body motion planning of legged robots," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5448–5455, 2020.
- [8] E. Pellegrini and R. P. Russell, "A multiple-shooting differential dynamic programming algorithm. part 1: Theory," *Acta Astronautica*, vol. 170, pp. 686–700, 2020.
- [9] B. Plancher and S. Kuindersma, "A performance analysis of parallel differential dynamic programming on a GPU," in *Int. Workshop on the Algorithmic Foundations of Robotics*, 2018, pp. 656–672.
- [10] S.-H. Lee, J. Kim, F. C. Park, M. Kim, and J. E. Bobrow, "Newton-type algorithms for dynamics-based robot movement optimization," *IEEE Transactions on Robotics*, vol. 21, no. 4, pp. 657–667, 2005.
- [11] J. N. Nganga and P. M. Wensing, "Accelerating second-order differential dynamic programming for rigid-body systems," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7659–7666, 2021.
- [12] R. Featherstone, *Rigid Body Dynamics Algorithms*. Springer, 2008.
- [13] A. Kowarz and A. Walther, "Optimal checkpointing for time-stepping procedures in ADOL-C," in *Int. Conf. on Computational Science*, 2006, pp. 541–549.
- [14] S. Singh, R. Russell, and P. M. Wensing, "Efficient analytical derivatives of rigid-body dynamics using spatial vector algebra," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 1776–1783, 2022.
- [15] A. Jain and G. Rodriguez, "Linearization of manipulator dynamics using spatial operators," *IEEE transactions on Systems, Man, and Cybernetics*, vol. 23, no. 1, pp. 239–248, 1993.
- [16] K. Ayusawa and E. Yoshida, "Comprehensive theory of differential kinematics and dynamics towards extensive motion optimization framework," *Int. J. of Robotics Research*, vol. 37, no. 13–14, pp. 1554–1572, 2018.
- [17] J. Carpentier and N. Mansard, "Analytical derivatives of rigid body dynamics algorithms," in *Robotics: Science and systems*, 2018.
- [18] B. Plancher, S. M. Neuman, T. Bourgeat, S. Kuindersma, S. Devadas, and V. J. Reddi, "Accelerating robot dynamics gradients on a cpu, gpu, and fpga," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2335–2342, 2021.
- [19] S. Singh, R. P. Russell, and P. M. Wensing, "Details of second-order partial derivatives of rigid-body inverse dynamics," 2022, arXiv:2203.00679.
- [20] J. A. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi: a software framework for nonlinear optimization and optimal control," *Mathematical Prog. Comp.*, vol. 11, no. 1, pp. 1–36, 2019.
- [21] D. E. Orin, R. McGhee, M. Vukobratović, and G. Hartoch, "Kinematic and kinetic analysis of open-chain linkages utilizing Newton-Euler methods," *Math. Biosciences*, vol. 43, no. 1–2, pp. 107–130, 1979.
- [22] S. Echeandia and P. M. Wensing, "Numerical methods to compute the coriolis matrix and christoffel symbols for rigid-body systems," *Journal of Comp. and Nonlinear Dynamics*, vol. 16, no. 9, 2021.
- [23] G. Garofalo, C. Ott, and A. Albu-Schäffer, "On the closed form computation of the dynamic matrices and their differentiations," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2013, pp. 2364–2359.
- [24] S. Singh and P. M. Wensing, https://github.com/ROAM-Lab-ND/spatial_v2_extended/blob/main/v3/derivatives/ID_SO_derivatives.m, 2022, see commit: b06fd78, 03/01/2022.
- [25] C. C. Cossette, A. Walsh, and J. R. Forbes, "The complex-step derivative approximation on matrix lie groups," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 906–913, 2020.
- [26] S. Singh, https://github.com/shubhamsingh91/pinocchio/blob/master/src/algorithm/mea_SO_derivatives.hxx, 2022.
- [27] J. Carpentier et al., "The Pinocchio C++ library: A fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives," in *IEEE/SICE Int. Symposium on System Integration*, 2019, pp. 614–619.

Algorithm 1 IDSVA SO Algorithm. Temporary variables in the algorithm are sized as $\mathbf{A}_i \in \mathbb{R}^{6 \times 6}$ for matrices, and $\mathbf{u}_i \in \mathbb{R}^{6 \times 1}$ for vectors.

Require: $\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}, model$

```

1:  $\mathbf{v}_0 = 0; \mathbf{a}_0 = -\mathbf{a}_g$ 
2: for  $j = 1$  to  $N$  do
3:    $\mathbf{v}_i = \mathbf{v}_{\lambda(i)} + \mathbf{S}_i \dot{\mathbf{q}}_i$ 
4:    $\mathbf{a}_i = \mathbf{a}_{\lambda(i)} + \mathbf{S}_i \ddot{\mathbf{q}}_i + \mathbf{v}_i \times \mathbf{S}_i \dot{\mathbf{q}}_i$ 
5:    $\dot{\mathbf{S}}_i = \mathbf{v}_i \times \mathbf{S}_i$ 
6:    $\ddot{\mathbf{S}}_i = \mathbf{v}_{\lambda(i)} \times \mathbf{S}_i$ 
7:    $\ddot{\mathbf{S}}_i = \mathbf{a}_{\lambda(i)} \times \mathbf{S}_i + \mathbf{v}_{\lambda(i)} \times \dot{\mathbf{S}}_i$ 
8:    $\mathbf{I}_i^C = \mathbf{I}_i$ 
9:    $\mathbf{B}_i^C = (\mathbf{v}_i \times^*) \mathbf{I}_i - \mathbf{I}_i (\mathbf{v}_i \times) + (\mathbf{I}_i \mathbf{v}_i) \bar{\times}^*$ 
10:   $\mathbf{f}_i^C = \mathbf{I}_i \mathbf{a}_i + (\mathbf{v}_i \times^*) \mathbf{I}_i \mathbf{v}_i$ 
11: end for
12: for  $i = N$  to  $1$  do
13:   for  $p = 1$  to  $n_i$  do
14:     $\mathbf{s}_p = \mathbf{S}_{i,p}; \dot{\psi}_p = \dot{\mathbf{S}}_{i,p}; \ddot{\psi}_p = \ddot{\mathbf{S}}_{i,p}; \dot{\mathbf{s}}_p = \dot{\mathbf{S}}_{i,p}$ 
15:     $\mathbf{B}_i^C(\mathbf{s}_p) = (\mathbf{s}_p \times^*) \mathbf{I}_i^C - \mathbf{I}_i^C (\mathbf{s}_p \times) + (\mathbf{I}_i^C \mathbf{s}_p) \bar{\times}^*$ 
16:     $\mathbf{B}_i^C(\dot{\psi}_p) = (\dot{\psi}_p \times^*) \mathbf{I}_i^C - \mathbf{I}_i^C (\dot{\psi}_p \times) + (\mathbf{I}_i^C \dot{\psi}_p) \bar{\times}^*$ 
17:     $\mathbf{A}_0 = (\mathbf{I}_i^C \mathbf{s}_p) \bar{\times}^*$ 
18:     $\mathbf{A}_1 = \mathbf{s}_p \times^* \mathbf{I}_i^C - \mathbf{I}_i^C \times \mathbf{s}_p$ 
19:     $\mathbf{A}_2 = 2\mathbf{A}_0 - \mathbf{B}_i^C(\mathbf{s}_p)$ 
20:     $\mathbf{A}_3 = \mathbf{B}_i^C(\dot{\psi}_p) + \mathbf{s}_p \times^* \mathbf{B}_i^C - \mathbf{B}_i^C \times \mathbf{s}_p$ 
21:     $\mathbf{A}_4 = (\mathbf{B}_i^{C,T} \mathbf{s}_p) \bar{\times}^*$ 
22:     $\mathbf{A}_5 = (\mathbf{B}_i^C \dot{\psi}_p + \mathbf{I}_i^C \ddot{\psi}_p + \mathbf{s}_p \times^* \mathbf{f}_i) \bar{\times}^*$ 
23:     $\mathbf{A}_6 = \mathbf{s}_p \times^* \mathbf{I}_i^C + \mathbf{A}_0$ 
24:     $\mathbf{A}_7 = (\mathbf{B}_i^C \mathbf{s}_p + \mathbf{I}_i^C (\dot{\psi}_p + \dot{\mathbf{s}}_p)) \bar{\times}^*$ 
25:     $j = i$ 
26:    while  $j > 0$  do
27:     for  $t = 1$  to  $n_j$  do
28:       $\mathbf{s}_t = \mathbf{S}_{j,t}; \dot{\psi}_t = \dot{\mathbf{S}}_{j,t}; \ddot{\psi}_t = \ddot{\mathbf{S}}_{j,t}; \dot{\mathbf{s}}_t = \dot{\mathbf{S}}_{j,t}$ 
29:       $\mathbf{u}_1 = \mathbf{A}_3^\top \mathbf{s}_t; \mathbf{u}_2 = \mathbf{A}_1^\top \mathbf{s}_t$ 
30:       $\mathbf{u}_3 = \mathbf{A}_3 \dot{\psi}_t + \mathbf{A}_1 \ddot{\psi}_t + \mathbf{A}_5 \mathbf{s}_t$ 
31:       $\mathbf{u}_4 = \mathbf{A}_6 \mathbf{s}_t; \mathbf{u}_5 = \mathbf{A}_2 \dot{\psi}_t + \mathbf{A}_4 \mathbf{s}_t$ 
32:       $\mathbf{u}_6 = \mathbf{B}_i^C(\mathbf{s}_p) \dot{\psi}_t + \mathbf{A}_7 \mathbf{s}_t$ 
33:       $\mathbf{u}_7 = \mathbf{A}_3 \mathbf{s}_t + \mathbf{A}_1 (\dot{\psi}_t + \dot{\mathbf{s}}_t)$ 
34:       $\mathbf{u}_8 = \mathbf{A}_4 \mathbf{s}_t - \mathbf{B}_i^{C,T}(\mathbf{s}_p) \dot{\psi}_t; \mathbf{u}_9 = \mathbf{A}_0 \mathbf{s}_t$ 
35:       $\mathbf{u}_{10} = \mathbf{B}_i^C(\mathbf{s}_p) \mathbf{s}_t; \mathbf{u}_{11} = \mathbf{B}_i^{C,T}(\mathbf{s}_p) \mathbf{s}_t$ 
36:       $\mathbf{u}_{12} = \mathbf{A}_1 \mathbf{s}_t$ 
37:       $k = j$ 
38:      while  $k > 0$  do
39:       for  $r = 1$  to  $n_k$  do
40:         $\mathbf{s}_r = \mathbf{S}_{k,r}; \dot{\psi}_r = \dot{\mathbf{S}}_{k,r}$ 
41:         $\ddot{\psi}_r = \ddot{\mathbf{S}}_{k,r}; \dot{\mathbf{s}}_r = \dot{\mathbf{S}}_{k,r}$ 
42:         $p_1 = \mathbf{u}_{11}^\top \dot{\psi}_r$ 
43:         $p_2 = \mathbf{u}_8^\top \dot{\psi}_r + \mathbf{u}_9^\top \ddot{\psi}_r$ 

```

```

44:    $\frac{\partial^2 \tau_{i,p}}{\partial \mathbf{q}_{j,t} \partial \mathbf{q}_{k,r}} = p_2; \frac{\partial^2 \tau_{i,p}}{\partial \mathbf{q}_{k,r} \partial \dot{\mathbf{q}}_{j,t}} = -p_1$ 
45:   if  $j \neq i$  then
46:      $\frac{\partial^2 \tau_{j,t}}{\partial \mathbf{q}_{k,r} \partial \mathbf{q}_{i,p}} = \frac{\partial^2 \tau_{j,t}}{\partial \mathbf{q}_{i,p} \partial \mathbf{q}_{k,r}} = \mathbf{u}_1^\top \dot{\psi}_r + \mathbf{u}_2^\top \ddot{\psi}_r$ 
47:      $\frac{\partial^2 \tau_{j,t}}{\partial \mathbf{q}_{i,p} \partial \dot{\mathbf{q}}_{k,r}} = \mathbf{u}_1^\top \mathbf{s}_r + \mathbf{u}_2^\top (\dot{\mathbf{s}}_r + \dot{\psi}_r)$ 
48:      $\frac{\partial^2 \tau_{j,t}}{\partial \mathbf{q}_{k,r} \partial \dot{\mathbf{q}}_{i,p}} = p_1$ 
49:      $\frac{\partial^2 \tau_{j,t}}{\partial \dot{\mathbf{q}}_{k,r} \partial \dot{\mathbf{q}}_{i,p}} = \frac{\partial^2 \tau_{j,t}}{\partial \dot{\mathbf{q}}_{i,p} \partial \dot{\mathbf{q}}_{k,r}} = \mathbf{u}_{11}^\top \mathbf{s}_r$ 
50:      $\frac{\partial \mathbf{M}_{k,r,j,t}}{\partial \mathbf{q}_{i,p}} = \frac{\partial \mathbf{M}_{j,t,k,r}}{\partial \mathbf{q}_{i,p}} = \mathbf{s}_r^\top \mathbf{u}_{12}$ 
51:   end if
52:   if  $k \neq j$  then
53:      $\frac{\partial^2 \tau_{i,p}}{\partial \mathbf{q}_{k,r} \partial \mathbf{q}_{j,t}} = p_2; \frac{\partial^2 \tau_{k,r}}{\partial \mathbf{q}_{i,p} \partial \mathbf{q}_{j,t}} = \mathbf{s}_r^\top \mathbf{u}_3$ 
54:      $\frac{\partial^2 \tau_{i,p}}{\partial \dot{\mathbf{q}}_{j,t} \partial \dot{\mathbf{q}}_{k,r}} = \frac{\partial^2 \tau_{i,p}}{\partial \dot{\mathbf{q}}_{k,r} \partial \dot{\mathbf{q}}_{j,t}} = -\mathbf{u}_{11}^\top \mathbf{s}_r$ 
55:      $\frac{\partial^2 \tau_{i,p}}{\partial \mathbf{q}_{j,t} \partial \dot{\mathbf{q}}_{k,r}} = \mathbf{s}_r^\top \mathbf{u}_5 + \mathbf{u}_9^\top (\dot{\mathbf{s}}_r + \dot{\psi}_r)$ 
56:      $\frac{\partial^2 \tau_{k,r}}{\partial \mathbf{q}_{j,t} \partial \dot{\mathbf{q}}_{i,p}} = \mathbf{s}_r^\top \mathbf{u}_6$ 
57:      $\frac{\partial \mathbf{M}_{k,r,i,p}}{\partial \mathbf{q}_{j,t}} = \frac{\partial \mathbf{M}_{i,p,k,r}}{\partial \mathbf{q}_{j,t}} = \mathbf{s}_r^\top \mathbf{u}_9$ 
58:   if  $j \neq i$  then
59:      $\frac{\partial^2 \tau_{k,r}}{\partial \mathbf{q}_{j,t} \partial \mathbf{q}_{i,p}} = \frac{\partial^2 \tau_{k,r}}{\partial \mathbf{q}_{i,p} \partial \mathbf{q}_{j,t}}$ 
60:      $\frac{\partial^2 \tau_{k,r}}{\partial \mathbf{q}_{i,p} \partial \dot{\mathbf{q}}_{j,t}} = \mathbf{s}_r^\top \mathbf{u}_7$ 
61:      $\frac{\partial^2 \tau_{k,r}}{\partial \dot{\mathbf{q}}_{j,t} \partial \dot{\mathbf{q}}_{i,p}} = \frac{\partial^2 \tau_{k,r}}{\partial \dot{\mathbf{q}}_{i,p} \partial \dot{\mathbf{q}}_{j,t}} = \mathbf{s}_r^\top \mathbf{u}_{10}$ 
62:   else
63:      $\frac{\partial^2 \tau_{k,r}}{\partial \mathbf{q}_{j,t} \partial \dot{\mathbf{q}}_{i,p}} = \mathbf{s}_r^\top \mathbf{u}_4$ 
64:   end if
65:   else
66:      $\frac{\partial^2 \tau_{i,p}}{\partial \mathbf{q}_{j,t} \partial \dot{\mathbf{q}}_{k,r}} = -\mathbf{u}_2^\top \mathbf{s}_r$ 
67:   end if
68:   end for
69:    $k = \lambda(k)$ 
70:   end while
71:   end for
72:    $j = \lambda(j)$ 
73:   end while
74:   end for
75:   if  $\lambda(i) > 0$  then
76:      $\mathbf{I}_{\lambda(i)}^C = \mathbf{I}_{\lambda(i)}^C + \mathbf{I}_i^C; \mathbf{B}_{\lambda(i)}^C = \mathbf{B}_{\lambda(i)}^C + \mathbf{B}_i^C$ 
77:      $\mathbf{f}_{\lambda(i)}^C = \mathbf{f}_{\lambda(i)}^C + \mathbf{f}_i^C$ 
78:   end if
79:   end for
80:   return  $\frac{\partial^2 \tau}{\partial \mathbf{q}^2}, \frac{\partial^2 \tau}{\partial \dot{\mathbf{q}}^2}, \frac{\partial^2 \tau}{\partial \mathbf{q} \partial \dot{\mathbf{q}}}, \frac{\partial \mathbf{M}}{\partial \mathbf{q}}$ 

```
