Lie Group Formulation of Articulated Rigid Body Dynamics

Junggon Kim

12/10/2012, Ver 2.01

Abstract

It has been usual in most old-style text books for dynamics to treat the formulas describing linear(or translational) and angular(or rotational) motion of a rigid body separately. For example, the famous Newton's 2nd law, f = ma, for the translational motion of a rigid body has its partner, so-called the Euler's equation which describes the rotational motion of the body. Separating translation and rotation, however, causes a huge complexity in deriving the equations of motion of articulated rigid body systems such as robots.

In Section 1, an elegant single equation of motion of a rigid body moving in 3D space is derived using a Lie group formulation. In Section 2, the recursive Newton-Euler algorithm (inverse dynamics), Articulated-Body algorithm (forward dynamics) and a generalized recursive algorithm (hybrid dynamics) for open chains or tree-structured articulated body systems are rewritten with the geometric formulation for rigid body. In Section 3, dynamics of constrained systems such as a closed loop mechanism will be described. Finally, in Section 4, analytic derivatives of the dynamics algorithms, which would be useful for optimization and sensitivity analysis, are presented.[1]

1 Dynamics of a Rigid Body

This section describes the equations of motion of a single rigid body in a geometric manner.

1.1 Rigid Body Motion

To describe the motion of a rigid body, we need to represent both the position and orientation of the body. Let $\{B\}$ be a coordinate frame attached to the rigid body and $\{A\}$ be an arbitrary coordinate frame, and all coordinate frames will be right-handed Cartesian from now on. We can define a 3×3 matrix

$$R = [x_{ab}, y_{ab}, z_{ab}] \tag{1}$$

where $x_{ab}, y_{ab}, z_{ab} \in \Re^3$ are the coordinates of the coordinate axes of $\{B\}$ with respect to $\{A\}$. A matrix of this form is called a *rotation matrix* as it can be used to describe the orientation(or rotation) of a rigid body, relative to a reference frame. Since the columns

[1]GEAR (Geometric Engine for Articulated Rigid-body simulation) is a C++ implementation of the algorithms presented in this article. (http://www.cs.cmu.edu/~junggon/tools/gear.html)

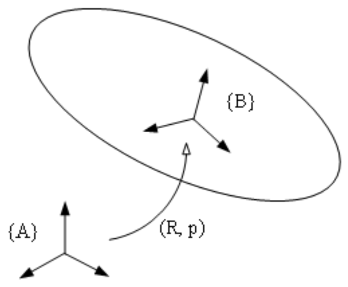

Figure 1: Coordinates frames for a rigid body

of a rotation matrix are mutually orthonormal and the coordinate frame is right-handed, the rotation matrix has two properties,

$$RR^{\mathrm{T}} = R^{\mathrm{T}}R = I, \quad \det R = 1, \tag{2}$$

and it is denoted by $SO(3)$[2].

Let $p \in \Re^3$ be the position vector of the origin of $\{B\}$ from the origin of $\{A\}$, and $R \in SO(3)$ be the rotation matrix of $\{B\}$ relative to $\{A\}$. The configuration space of the rigid body motion can be represented with the pair (R, p), which is denoted as $SE(3)$. A 4×4 matrix,

$$T = \begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix} \tag{3}$$

is called the *homogeneous representation* of $T = (R, p) \in SE(3)$, and its inverse can be obtained with

$$T^{-1} = \begin{bmatrix} R^{\mathrm{T}} & -R^{\mathrm{T}}p \\ 0 & 1 \end{bmatrix}. \tag{4}$$

From now on, a simple declaration, $T \in SE(3) : \{A\} \to \{B\}$, will be used to notify that $T \in SE(3)$ represents the orientation and position of a coordinate frame $\{B\}$ with respect to another coordinate frame $\{A\}$.

The Lie algebra of $SE(3)$, denoted as $se(3)$, is identified as a 6-dimensional vector space $(w, v) \in \Re^6$ where $w \in so(3)$, the Lie algebra of $SO(3)$. $\xi = (w, v) \in se(3)$ can also be represented as a 4×4 matrix,

$$\xi = \begin{bmatrix} [w] & v \\ 0 & 0 \end{bmatrix} \tag{5}$$

where $[w] = \begin{bmatrix} 0 & -w_3 & w_2 \\ w_3 & 0 & -w_1 \\ -w_2 & w_1 & 0 \end{bmatrix} \in \Re^{3\times 3}$ is a skew-symmetric matrix.

The adjoint action of $T \in SE(3)$ on $\xi \in se(3)$, $\mathrm{Ad} : SE(3) \times se(3) \to se(3)$, is defined as

$$\mathrm{Ad}_T\, \xi = T\, \xi\, T^{-1}. \tag{6}$$

[2] The notation *SO* abbreviates *special orthogonal* and 'special' refers to the fact that $\det R = +1$ rather than ± 1. See [2] for more details.

From (3), (4) and (5), Ad_T can be regarded as a linear transformation, $\mathrm{Ad}_T : se(3) \rightarrow se(3)$, which is defined by a 6×6 matrix

$$\mathrm{Ad}_T = \begin{bmatrix} R & 0 \\ [p]R & R \end{bmatrix} \tag{7}$$

where $T = (R, p) \in SE(3)$. The coadjoint action of T on $\xi^* \in dse(3)$ which is the dual of ξ, $\mathrm{Ad}^*_T : dse(3) \rightarrow dse(3)$, is defined by a 6×6 matrix

$$\mathrm{Ad}^*_T = \mathrm{Ad}^{\mathrm{T}}_T. \tag{8}$$

1.2 Generalized Velocity and Force

Let $T(t) = (R(t), p(t)) \in SE(3)$ be a motion trajectory of a coordinate frame attached to a rigid body with respect to an inertial frame. The *generalized velocity* of the rigid body is defined as

$$V = T^{-1}\dot{T} = \begin{bmatrix} [w] & v \\ 0 & 0 \end{bmatrix} \tag{9}$$

where $[w] = R^{\mathrm{T}}\dot{R}$ and $v = R^{\mathrm{T}}\dot{p}$. The physical meaning of $w \in \Re^3$ is the rotational(or angular) velocity of the coordinate frame attached to the body relative to the inertial frame, but expressed in the body coordinate frame. Similarly, $v \in \Re^3$ represents the velocity of the origin of the coordinate frame relative to the inertial frame, and still expressed in the body frame. The generalized velocity is an element of $se(3)$, and can be simply regarded as a 6-dimensional vector, i.e.,

$$V = \begin{pmatrix} w \\ v \end{pmatrix}. \tag{10}$$

As the generalized velocity is an instance of $se(3)$, it follows the adjoint transformation rule defined in (7). Let $\{A\}, \{B\}$ be two different coordinate frames attached to the same rigid body, and $T_a, T_b \in SE(3)$ represent the orientation and position of the two frames with respect to an inertial frame. Then, from (6) and (9), the generalized velocities of $\{A\}$ and $\{B\}$ have the following relation:

$$V_b = \mathrm{Ad}_{T_{ba}} V_a \tag{11}$$

where $T_{ba} \in SE(3) : \{B\} \rightarrow \{A\}$.

With a coordinate frame attached to a rigid body, the *generalized force* acting on the body can be defined as

$$F = \begin{pmatrix} m \\ f \end{pmatrix} \tag{12}$$

where $m \in \Re^3$ and $f \in \Re^3$ represent a moment and force acting on the body respectively, viewed in the body frame. The generalized force is known as the member of $dse(3)$, and has the following transformation rule,

$$F_b = \mathrm{Ad}^*_{T_{ab}} F_a \tag{13}$$

where F_a and F_b denote a generalized force viewed from different body frames $\{A\}$ and $\{B\}$, and $T_{ab} \in SE(3) : \{A\} \rightarrow \{B\}$.

1.3 Generalized Inertia and Momentum

The kinetic energy of a rigid body is given by the following volume integral

$$\mathrm{e} = \int_{\mathrm{vol}} \frac{1}{2} \|v\|^2 \, d\mathrm{m} \tag{14}$$

which means the sum of the kinetic energies of all the mass particles constituting the body. By introducing a coordinate frame attached to the body, (14) can be restructured as the following simple quadratic form,

$$\mathrm{e} = \frac{1}{2} V^{\mathrm{T}} \mathcal{I} V \tag{15}$$

where $V \in \Re^6$ is the generalized velocity of the body and $\mathcal{I} \in \Re^{6\times 6}$, which is known as *generalized inertia*, represents the mass and mass distribution with respect to the body frame.

To obtain an explicit form of the generalized inertia of a rigid body, let $r \in \Re^3$ be the position of a body point relative to the body frame and $(R, p) \in SE(3)$ represents orientation and position of the body frame with respect to an inertial frame respectively. Using $||v||^2 = ||\dot{p} + \dot{R}r||^2$, $R^{\mathrm{T}}\dot{R} = [w]$ and $R^{\mathrm{T}}\dot{p} = v$, (14) can be rewritten as

$$\begin{aligned}
\mathrm{e} &= \frac{1}{2} \int_{\mathrm{vol}} \left(||\dot{p}||^2 + 2\dot{p}^{\mathrm{T}}\dot{R}r + ||\dot{R}r||^2 \right) d\mathrm{m} && (16) \\
&= \frac{1}{2} \left\{ \dot{p}^{\mathrm{T}}\dot{p} \int_{\mathrm{vol}} d\mathrm{m} - 2\dot{p}^{\mathrm{T}} R \left(\int_{\mathrm{vol}} [r] d\mathrm{m} \right) w + w^{\mathrm{T}} \left(\int_{\mathrm{vol}} [r]^{\mathrm{T}}[r] d\mathrm{m} \right) w \right\} && (17) \\
&= \frac{1}{2} \left\{ \mathrm{m} v^{\mathrm{T}} v - 2v^{\mathrm{T}} \left(\int_{\mathrm{vol}} [r] d\mathrm{m} \right) w + w^{\mathrm{T}} \left(\int_{\mathrm{vol}} [r]^{\mathrm{T}}[r] d\mathrm{m} \right) w \right\} && (18) \\
&= \frac{1}{2} V^{\mathrm{T}} \mathcal{I} V && (19)
\end{aligned}$$

where $V = (w, v)$ is the generalized velocity of the body and the generalized inertia, $\mathcal{I}$, has the following explicit matrix form:

$$\mathcal{I} = \begin{bmatrix} \int_{\mathrm{vol}} [r]^{\mathrm{T}}[r] d\mathrm{m} & \int_{\mathrm{vol}} [r] d\mathrm{m} \\ \int_{\mathrm{vol}} [r]^{\mathrm{T}} d\mathrm{m} & \mathrm{m}\mathbf{1} \end{bmatrix}. \tag{20}$$

The generalized inertia is symmetric positive definite, and its upper diagonal term, $I = \int_{\mathrm{vol}} [r]^{\mathrm{T}}[r] d\mathrm{m}$, is the definition of the well-known 3×3 inertia matrix of the rigid body with respect to the body frame. If the origin of the body frame is located on the center of mass, then the generalized inertia becomes a block diagonal matrix because $\int_{\mathrm{vol}} [r] d\mathrm{m} = 0$. In addition, if the orientation of the body frame also coincides with the principle axes of the body, then the generalized inertia becomes a diagonal matrix.

Let $\{A\}$ and $\{B\}$ be coordinate frames attached to a rigid body, $\mathcal{I}_a$ and $\mathcal{I}_b$ be the generalized inertias of the body corresponding to the two frames. Using (11), (15), and the fact that the kinetic energy of the body should remain under change of coordinate frame, the following transformation rule between the generalized inertias can be obtained:

$$\mathcal{I}_b = \mathrm{Ad}^*_{T_{ab}} \mathcal{I}_a \mathrm{Ad}_{T_{ab}} \tag{21}$$

where $T_{ab} \in SE(3) : \{A\} \to \{B\}$. If the mass, m, and the inertia matrix in a center of mass frame[3], $I_c \in \Re^{3\times 3}$, are given, one can get the generalized inertia in an arbitrary body frame from (21), rather than using (20), as

$$\mathcal{I} = \begin{bmatrix} RI_cR^{\mathrm{T}} + \mathrm{m}[p]^{\mathrm{T}}[p] & \mathrm{m}[p] \\ \mathrm{m}[p]^{\mathrm{T}} & \mathrm{m}\mathbf{1} \end{bmatrix} \tag{22}$$

where $(R, p) \in SE(3)$ represents the orientation and position of the center of mass frame with respect to the body frame.

The *generalized momentum* of a rigid body is defined as

$$L = \mathcal{I}V \tag{23}$$

where $\mathcal{I}$ and V are the generalized inertia and velocity of the body expressed in a coordinate frame attached to the body.

L_a and L_b be the generalized momentum of a rigid body expressed in different body frames, $\{A\}$ and $\{B\}$, respectively. Using (11) and (21) one can derive the following transformation rule for generalized momentums:

$$L_b = \mathrm{Ad}^*_{T_{ab}} L_a \tag{24}$$

which is same to that of generalized forces, and indeed, the generalized momentum is also known as $dse(3)$.

1.4 Time Derivatives of $se(3)$ and $dse(3)$

Recall that the time derivative of a 3-dimensional vector $x = \sum_{i=1}^{3} x_i \hat{e}_i$, expressed in a moving coordinate frame $\{\hat{e}_i\}$, can be obtained as

$$\frac{d}{dt}x = \sum_{i=1}^{3} \left\{ \left(\frac{d}{dt}x_i\right)\hat{e}_i + x_i\left(\frac{d}{dt}\hat{e}_i\right) \right\} = \dot{x} + w \times x \tag{25}$$

where $\dot{x} = \sum_{i=1}^{3}\left(\frac{d}{dt}x_i\right)\hat{e}_i = \left(\frac{dx_1}{dt}, \frac{dx_2}{dt}, \frac{dx_3}{dt}\right)$ is the component-wise time derivative of x and w is the angular velocity of the moving frame.

The time derivatives of $se(3)$ and $dse(3)$ which are expressed in a moving frame have more generalized form. A simple approach to obtain the derivatives is to transform $se(3)$(or $dse(3)$) to a stationary coordinate frame, differentiate it there, and transform the derivative back to the original moving coordinate frame with the (correct) assumption that the derivative of $se(3)$(or $dse(3)$) can be transformed with the rule of $se(3)$(or $dse(3)$).[4]Note that differentiating a vector in a stationary coordinate frame with respect to time is just getting the component-wise derivative of it.

Lemma 1. *Let $X \in se(3)$ be expressed in a moving frame attached to a rigid body. Then the time derivative of X can be obtained by*

$$\frac{d}{dt}X = \dot{X} + \mathrm{ad}_V X \tag{26}$$

[3] A coordinate frame whose origin is located on the center of mass of the body.

[4] In [1] the time derivative of a spatial velocity which is similar to the generalized velocity in this article was obtained with this approach.

where $\dot{X}$ is the component-wise time derivative of X and $\mathrm{ad}_V : se(3) \to se(3)$ *is a linear transformation defined as*

$$\mathrm{ad}_V = \begin{bmatrix} [w] & 0 \\ [v] & [w] \end{bmatrix} \tag{27}$$

where $V = (w, v) \in se(3)$ is the generalized velocity of the body.

Proof. Let $T = (R, p) \in SE(3)$ denote the orientation and position of the moving frame with respect to an inertial frame which is stationary in the space. By transforming X to the inertial frame, differentiating it there, and then transforming the result back to the original body frame, one can get

$$\frac{d}{dt}X = \mathrm{Ad}_{T^{-1}} \frac{d'}{dt}\left(\mathrm{Ad}_T X\right) = \dot{X} + \mathrm{Ad}_{T^{-1}} \frac{d'}{dt}\left(\mathrm{Ad}_T\right) X$$

where $\frac{d'}{dt}$ represents the component-wise differentiation and $\dot{X} = \frac{d'}{dt}X$. Using (7) and $V = (w, v) = (R^{\mathrm{T}}\dot{R}, R^{\mathrm{T}}\dot{p})$, one can show $\mathrm{Ad}_{T^{-1}}\frac{d'}{dt}\left(\mathrm{Ad}_T\right) = \mathrm{ad}_V$ as follows:

$$\begin{aligned} \mathrm{Ad}_{T^{-1}} \frac{d'}{dt}\mathrm{Ad}_T &= \begin{bmatrix} R^{\mathrm{T}} & 0 \\ -R^{\mathrm{T}}[p] & R^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} \dot{R} & 0 \\ [\dot{p}]R + [p]\dot{R} & \dot{R} \end{bmatrix} \\ &= \begin{bmatrix} [w] & 0 \\ [v] & [w] \end{bmatrix} \end{aligned}$$

□

Lemma 2. *Let $Y \in dse(3)$ be expressed in a moving frame attached to a rigid body. Then the time derivative of Y can be obtained by*

$$\frac{d}{dt}Y = \dot{Y} - \mathrm{ad}_V^* Y \tag{28}$$

where $\dot{Y}$ is the component-wise time derivative of Y and $\mathrm{ad}_V^* : dse(3) \to dse(3)$ *is a linear transformation defined as*

$$\mathrm{ad}_V^* = \mathrm{ad}_V^{\mathrm{T}} = \begin{bmatrix} [w] & 0 \\ [v] & [w] \end{bmatrix}^{\mathrm{T}} \tag{29}$$

where $V = (w, v) \in se(3)$ is the generalized velocity of the body.

Proof. Similarly to the proof of Lemma 1, the derivative of $Y \in dse(3)$ can be obtained by

$$\frac{d}{dt}Y = \mathrm{Ad}_T^* \frac{d'}{dt}\left(\mathrm{Ad}_{T^{-1}}^* Y\right) = \dot{Y} + \mathrm{Ad}_T^* \frac{d'}{dt}\left(\mathrm{Ad}_{T^{-1}}^*\right) Y.$$

Using (8) and $V = (w, v) = (R^{\mathrm{T}}\dot{R}, R^{\mathrm{T}}\dot{p})$, one can show $\mathrm{Ad}_T^* \frac{d'}{dt}\mathrm{Ad}_{T^{-1}}^* = -\mathrm{ad}_V^*$ as follows:

$$\begin{aligned} \mathrm{Ad}_T^* \frac{d'}{dt}\mathrm{Ad}_{T^{-1}}^* &= \begin{bmatrix} R^{\mathrm{T}} & -R^{\mathrm{T}}[p] \\ 0 & R^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} \dot{R} & [\dot{p}]R + [p]\dot{R} \\ 0 & \dot{R} \end{bmatrix} \\ &= \begin{bmatrix} [w] & [v] \\ 0 & [w] \end{bmatrix} = -\mathrm{ad}_V^{\mathrm{T}} = -\mathrm{ad}_V^* \end{aligned}$$

□

1.5 Geometric Dynamics of a Rigid Body

Equations of motion of a rigid body can be written as

$$F = \frac{d}{dt} L \tag{30}$$

where F represents the net sum of the generalized forces acting on the rigid body and L is the generalized momentum of the body. Using (23) and (28), the equations of motion of the rigid body can be written as

$$F = \mathcal{I}\dot{V} - \mathrm{ad}^*_V \mathcal{I} V \tag{31}$$

where $\dot{V}$ is the component-wise time derivative of the generalized velocity of the body. Note that $\dot{L} = \dot{\mathcal{I}} V + \mathcal{I}\dot{V} = \mathcal{I}\dot{V}$ because the components of $\mathcal{I}$ doesn't vary.

The dynamics equation of a rigid body is coordinate invariant, i.e., the structure of the equation still remains under change of coordinate frame. Let $\{A\}$ and $\{B\}$ be coordinate frames attached to a body, V_a and V_b be the generalized velocities, F_a and F_b be the generalized forces, and $\mathcal{I}_a$ and $\mathcal{I}_b$ be the generalized inertias corresponding to $\{A\}$ and $\{B\}$ respectively. Using the transformation rules, (11), (13), and (21), one can easily transform the dynamics equations with respect to $\{A\}$, $F_a = \mathcal{I}_a \dot{V}_a - \mathrm{ad}^*_{V_a} \mathcal{I}_a V_a$, to the equations in $\{B\}$, $F_b = \mathcal{I}_b \dot{V}_b - \mathrm{ad}^*_{V_b} \mathcal{I}_b V_b$, and this shows the coordinate invariance of (31) under change of coordinate frame.

2 Dynamics of Open Chain Systems

Let $q \in \Re^n$ denote the set of coordinates of all joints in a system, and for open chain systems, n is equal to the degree-of-freedom of the system. The dynamics equations of the system can be written as

$$M\ddot{q} + b = \tau \tag{32}$$

where $M(q) \in \Re^{n\times n}$ is a symmetric mass matrix of the system, $b(q,\dot{q}) \in \Re^n$ represents Coriolis, centrifugal, and gravity terms, and $\tau \in \Re^n$ denotes torque(or force) vector corresponding to the system coordinates q.

We assume the current system state $(q, \dot{q})$ is fully known. Calculating the joint torques (or forces for translational coordinates) τ with prescribe accelerations $\ddot{q}$ is called *inverse dynamics*. It is typically used to obtain the required joint torques which make the system move along a prescribed joint trajectory. On the other hand, calculating the resulting joint accelerations $\ddot{q}$ from the given joint torques τ is called *forward dynamics*, and this is usually used to simulate the system motion in time by integrating the computed acceleration to obtain the system state at the next time step.

In general, the command input to a joint coordinate can be either torque or acceleration during the simulation and the command type does not have to be same for all coordinates. Let's rewrite the equations of motion as

$$M \begin{pmatrix} \ddot{q}_u \\ \ddot{q}_v \end{pmatrix} + b = \begin{pmatrix} \tau_u \\ \tau_v \end{pmatrix} \tag{33}$$

where the subscript 'u' denotes the joint coordinates with prescribed acceleration and the subscript 'v' represents the coordinates with given or known torque. We can obtain

Inverse dynamics	$\ddot{q} \rightarrow \tau$
Forward dynamics	$\tau \rightarrow \ddot{q}$
Hybrid dynamics	$(\ddot{q}_u, \tau_v) \rightarrow (\tau_u, \ddot{q}_v)$

Table 1: Input and output of dynamics algorithms

$(\tau_u, \ddot{q}_v)$ from the prescribed $(\ddot{q}_u, \tau_v)$ by solving the equations of motion, which is called *hybrid dynamics*.[5] One possible solution for hybrid dynamics is to rearrange (33) and solve it with a direct matrix inversion. For example, the resulting accelerations of the torque-specified joint coordinates can be obtained as

$$\ddot{q}_v = M_{vv}^{-1}(\tau_v - b_v - M_{uv}\ddot{q}_u) \tag{34}$$

where $M = \left[\begin{smallmatrix} M_{uu} & M_{uv} \\ M_{uv} & M_{vv} \end{smallmatrix}\right]$, $b = \left(\begin{smallmatrix} b_u \\ b_v \end{smallmatrix}\right)$, and $q = \left(\begin{smallmatrix} q_u \\ q_v \end{smallmatrix}\right)$. The method, however, is not efficient for a complex system because it requires building the mass matrix and inverting the submatrix corresponding to the unprescribed joints, which leads to an $O(n^2)+O(n_b^3)$ algorithm where n and n_v denote the number of all coordinates and the number of unprescribed coordinates respectively.

In Table 1 the input and output of inverse, forward, and hybrid dynamics are summarized. Note that hybrid dynamics is a generalization of traditional inverse and forward dynamics, i.e., they can be regarded as the extreme cases of hybrid dynamics when $q_u = q$ (inverse dynamics) and $q_v = q$ (forward dynamics).

2.1 Recursive Inverse Dynamics

A recursive Newton-Euler inverse dynamics algorithm using the geometric notations shown in Section 1 was presented in [3]. Here the algorithm is slightly modified to support multi-degree-of-freedom joints in the formulation.

Let $\{0\}$ be an inertial frame which is stationary in the space, $\{i\}$ be a coordinate frame attached to the i-th rigid body of the open chain system, and $\{\lambda(i)\}$ be a coordinate frame attached to the parent body of the i-th rigid body. Also, let $T_i \in SE(3) : \{0\} \rightarrow \{i\}$, $T_{\lambda(i)} \in SE(3) : \{0\} \rightarrow \{\lambda(i)\}$, and $T_{\lambda(i),i} \in SE(3) : \{\lambda(i)\} \rightarrow \{i\}$. From $T_i = T_{\lambda(i)}T_{\lambda(i),i}$ and (6), the generalized velocity of the i-th body can be rewritten as

$$\begin{aligned} V_i &= T_i^{-1}\dot{T}_i \qquad (35) \\ &= T_{\lambda(i),i}^{-1}T_{\lambda(i)}^{-1}\left(\dot{T}_{\lambda(i)}T_{\lambda(i),i} + T_{\lambda(i)}\dot{T}_{\lambda(i),i}\right) \qquad (36) \\ &= \mathrm{Ad}_{T_{\lambda(i),i}^{-1}}V_{\lambda(i)} + S_i\dot{q}_i \qquad (37) \end{aligned}$$

where $S_i\dot{q}_i = T_{\lambda(i),i}^{-1}\dot{T}_{\lambda(i),i} \in se(3)$ represents the relative velocity of the i-th body with respect to its parent. $S_i = S_i(q_i) \in (se(3) \times n_i)$ is called the Jacobian of the joint connecting the i-th body and its parent and $q_i \in \Re^{n_i}$ represents the coordinate vector of the joint.

As shown in (31), component-wise time derivatives of the generalized velocities of all bodies in the system are needed to build the dynamics equations for each body. Recalling

[5]We follow [1] for the terminology.

that $\dot{A}^{-1} = -A^{-1}\dot{A}A^{-1}$ for an arbitrary matrix A, and $\mathrm{ad}_{\xi_1}\xi_2 = \xi_1\xi_2 - \xi_2\xi_1$ for arbitrary $\xi_1, \xi_2 \in se(3)$, one can derive the following formula for $\dot{V}_i$, the component-wise time derivative of V_i:

$$\begin{aligned}
\dot{V}_i &= \frac{d'}{dt}\left(T^{-1}_{\lambda(i),i} V_{\lambda(i)} T_{\lambda(i),i}\right) + \dot{S}_i\dot{q}_i + S_i\ddot{q}_i && (38)\\
&= -T^{-1}_{\lambda(i),i}\dot{T}_{\lambda(i),i}T^{-1}_{\lambda(i),i}V_{\lambda(i)}T_{\lambda(i),i} + T^{-1}_{\lambda(i),i}\dot{V}_{\lambda(i)}T_{\lambda(i),i}\\
&\quad + T^{-1}_{\lambda(i),i}V_{\lambda(i)}\dot{T}_{\lambda(i),i} + \dot{S}_i\dot{q}_i + S_i\ddot{q}_i && (39)\\
&= \mathrm{Ad}_{T^{-1}_{\lambda(i),i}}\dot{V}_{\lambda(i)} + \mathrm{ad}_{\mathrm{Ad}_{T^{-1}_{\lambda(i),i}}V_{\lambda(i)}} S_i\dot{q}_i + \dot{S}_i\dot{q}_i + S_i\ddot{q}_i && (40)\\
&= \mathrm{Ad}_{T^{-1}_{\lambda(i),i}}\dot{V}_{\lambda(i)} + \mathrm{ad}_{V_i} S_i\dot{q}_i + \dot{S}_i\dot{q}_i + S_i\ddot{q}_i && (41)
\end{aligned}$$

(37) and (41) are well suited to calculate the generalized velocity and its component-wise time derivative of each body in a open chain system from the ground to the end of the system recursively, as the velocity and acceleration of the ground are known in most cases.[6] Note that $\dot{S}_i \neq 0$ as the joint Jacobian S_i is a function of q_i in general.

Let $F_i \in dse(3)$ be the generalized force transmitted to i-th body from its parent through the connecting joint, and $F_i^{\text{ext}} \in dse(3)$ be a generalized force acting on the i-th body from environment. Both F_i and F_i^{ext} are expressed in $\{i\}$. From (31), the equations of motion for the i-th body can be written as

$$F_i + F_i^{\text{ext}} - \sum_{k\in\mu(i)} \mathrm{Ad}^*_{T^{-1}_{i,k}} F_k = \mathcal{I}_i\dot{V}_i - \mathrm{ad}^*_{V_i}\mathcal{I}_i V_i \qquad (42)$$

where the left hand side of the equations represents the net force acting on the body, $\mu(i)$ is the set of child bodies of the i-th body, and $-\mathrm{Ad}^*_{T^{-1}_{i,k}}F_k$ is the generalized force, transmitted from k-th child body, expressed in $\{i\}$. It should be noted that the generalized force, F_i, for each body can be calculated by (42) from the ends to the ground recursively, as the end bodies have no child.

A recursive inverse dynamics algorithm for open chain systems is shown in Table 2, and the following is a list of symbols for the geometric inverse dynamics:

- i = index of the i-th body.
- $\lambda(i)$ = index of the parent body of the i-th body.
- $\mu(i)$ = set of indexes of the child bodies of the i-th body.
- $q_i \in \Re^{n_i}$ = coordinates of the i-th joint which connects the i-th body with its parent body.
- $\tau_i \in \Re^{n_i}$ = torque(or force) exerted by the i-th joint.
- $T_{\lambda(i),i} \in SE(3) : \{\lambda(i)\} \to \{i\}$, a function of q_i.
- $V_i \in se(3)$ = the generalized velocity of the i-th body, viewed in the body frame $\{i\}$.

[6] One can assume that $V_0 = 0$ and $\dot{V}_0 = (0, g)$ where $g \in \Re^3$ denote the gravity vector, viewed in the inertial frame, with appropriate direction and magnitude.

- $\dot{V}_i \in se(3)$ = component-wise time derivative of V_i.
- $S_i \in (se(3) \times n_i)$ = Jacobian of $T_{\lambda(i),i}$ viewed in $\{i\}$.
  $S_i = \left[T^{-1}_{\lambda(i),i} \frac{\partial T^{-1}_{\lambda(i),i}}{\partial q^1_i}, \cdots, T^{-1}_{\lambda(i),i} \frac{\partial T^{-1}_{\lambda(i),i}}{\partial q^{n_i}_i} \right]$, where $q^k_i \in \Re$ denotes the k-th coordinate of the i-th joint, i.e., $q_i = (q^1_i, \cdots, q^{n_i}_i)$.
- $\mathcal{I}_i$ = the generalized inertia of the i-th body, viewed in $\{i\}$.
- $F_i \in dse(3)$ = the generalized force transmitted to the i-th body from its parent through the connecting joint, viewed in $\{i\}$.
- $F^{\text{ext}}_i \in dse(3)$ = the generalized force acting on the i-th body from environment, viewed in $\{i\}$.

Table 2: Recursive Inverse Dynamics

```
while forward recursion do
    $T_{\lambda(i),i}$ = function of $q_i$
    $V_i = \mathrm{Ad}_{T^{-1}_{\lambda(i),i}} V_{\lambda(i)} + S_i \dot{q}_i$
    $\dot{V}_i = \mathrm{Ad}_{T^{-1}_{\lambda(i),i}} \dot{V}_{\lambda(i)} + \mathrm{ad}_{V_i} S_i \dot{q}_i + \dot{S}_i \dot{q}_i + S_i \ddot{q}_i$
end while
while backward recursion do
    $F_i = \mathcal{I}_i \dot{V}_i - \mathrm{ad}^*_{V_i} \mathcal{I}_i V_i - F^{\text{ext}}_i + \sum_{k \in \mu(i)} \mathrm{Ad}^*_{T^{-1}_{i,k}} F_k$
    $\tau_i = S^{\mathrm{T}}_i F_i$
end while
```

2.2 Recursive Forward Dynamics

Featherstone [1] found that the dynamics equations of the i-th body can be reformulated to have the following form,

$$F_i = \hat{\mathcal{I}}_i \dot{V}_i + \hat{\mathcal{B}}_i, \tag{43}$$

where $\hat{\mathcal{I}}_i$ is called as the *articulated body inertia* of the body and $\hat{\mathcal{B}}_i$ is an associated bias force. He also showed that the articulated body inertia and bias force corresponding to each body in open chain systems can be calculated recursively, and by using these new quantities, forward dynamics can be solved with an $O(n)$ algorithm. A Lie group formulation of the articulated body inertia method was reported in [4]. Here, a more general form of the geometric formulation supporting multi-degree-of-freedom joint models is presented.

Starting from (43), we'll show that the same form of dynamics equations still holds for the parent of the i-th body($\lambda(i)$-th body), and $\hat{\mathcal{I}}_{\lambda(i)}$ and $\hat{\mathcal{B}}_{\lambda(i)}$ can be calculated from $\hat{\mathcal{I}}_i$ and $\hat{\mathcal{B}}_i$, which leads a backward recursion process for them.

Let's assume that the equations of motion for i-th body can be written as (43). By substituting (41) for $\dot{V}_i$ in (43), one can get

$$F_i = \hat{\mathcal{I}}_i \left(\mathrm{Ad}_{T^{-1}_{\lambda(i),i}} \dot{V}_{\lambda(i)} + \mathrm{ad}_{V_i} S_i \dot{q}_i + S_i \ddot{q}_i + \dot{S}_i \dot{q}_i \right) + \hat{\mathcal{B}}_i, \tag{44}$$

and from $S_i^{\mathrm{T}} F_i = \tau_i$, the unknown $\ddot{q}_i$ can be written as

$$\ddot{q}_i = \left(S_i^{\mathrm{T}} \hat{\mathcal{I}}_i S_i \right)^{-1} \left\{ \tau_i - S_i^{\mathrm{T}} \hat{\mathcal{I}}_i \left(\mathrm{Ad}_{T^{-1}_{\lambda(i),i}} \dot{V}_{\lambda(i)} + \mathrm{ad}_{V_i} S_i \dot{q}_i + \dot{S}_i \dot{q}_i \right) - S_i^{\mathrm{T}} \hat{\mathcal{B}}_i \right\}. \tag{45}$$

From (42) the dynamics equations for the $\lambda(i)$-th body becomes

$$F_{\lambda(i)} = \mathcal{I}_{\lambda(i)} \dot{V}_{\lambda(i)} - \mathrm{ad}^*_{V_{\lambda(i)}} \mathcal{I}_{\lambda(i)} V_{\lambda(i)} - F^{\mathrm{ext}}_{\lambda(i)} + \sum_{k \in \mu(\lambda(i))} \mathrm{Ad}^*_{T^{-1}_{\lambda(i),k}} F_k, \tag{46}$$

and by substituting (44) for F_k in (46), one can get

$$\begin{aligned} F_{\lambda(i)} = {} & \mathcal{I}_{\lambda(i)} \dot{V}_{\lambda(i)} - \mathrm{ad}^*_{V_{\lambda(i)}} \mathcal{I}_{\lambda(i)} V_{\lambda(i)} - F^{\mathrm{ext}}_{\lambda(i)} \\ & + \sum_{k \in \mu(\lambda(i))} \mathrm{Ad}^*_{T^{-1}_{\lambda(i),k}} \left\{ \hat{\mathcal{I}}_k \left(\mathrm{Ad}_{T^{-1}_{\lambda(i),k}} \dot{V}_{\lambda(i)} + \mathrm{ad}_{V_k} S_k \dot{q}_k + S_k \ddot{q}_k + \dot{S}_k \dot{q}_k \right) + \hat{\mathcal{B}}_k \right\}. \end{aligned} \tag{47}$$

By substituting (45) for the unknown $\ddot{q}_k$ in (47) and arranging the equations in terms of $\dot{V}_{\lambda(i)}$, one can have the dynamics equations for the $\lambda(i)$-th body with the following desired form:

$$F_{\lambda(i)} = \hat{\mathcal{I}}_{\lambda(i)} \dot{V}_{\lambda(i)} + \hat{\mathcal{B}}_{\lambda(i)} \tag{48}$$

where

$$\hat{\mathcal{I}}_{\lambda(i)} = \mathcal{I}_{\lambda(i)} + \sum_{k \in \mu(\lambda(i))} \mathrm{Ad}^*_{T^{-1}_{\lambda(i),k}} \left\{ \hat{\mathcal{I}}_k - \hat{\mathcal{I}}_k S_k \left(S_k^{\mathrm{T}} \hat{\mathcal{I}}_k S_k \right)^{-1} S_k^{\mathrm{T}} \hat{\mathcal{I}}_k \right\} \mathrm{Ad}_{T^{-1}_{\lambda(i),k}} \tag{49}$$

$$\begin{aligned} \hat{\mathcal{B}}_{\lambda(i)} = {} & -\mathrm{ad}^*_{V_{\lambda(i)}} \mathcal{I}_{\lambda(i)} V_{\lambda(i)} - F^{\mathrm{ext}}_{\lambda(i)} \\ & + \sum_{k \in \mu(\lambda(i))} \mathrm{Ad}^*_{T^{-1}_{\lambda(i),k}} \left\{ \hat{\mathcal{B}}_k + \hat{\mathcal{I}}_k \left(\mathrm{ad}_{V_k} S_k \dot{q}_k + \dot{S}_k \dot{q}_k \right) \right. \\ & \left. + \hat{\mathcal{I}}_k S_k \left(S_k^{\mathrm{T}} \hat{\mathcal{I}}_k S_k \right)^{-1} \left(\tau_k - S_k^{\mathrm{T}} \hat{\mathcal{I}}_k (\mathrm{ad}_{V_k} S_k \dot{q}_k + \dot{S}_k \dot{q}_k) - S_k^{\mathrm{T}} \hat{\mathcal{B}}_k \right) \right\}. \end{aligned} \tag{50}$$

In summary, the $O(n)$ algorithm for forward dynamics of open chain systems consists of the following three main recursion process:

1. Forward recursion: recursively calculates V_i for each body with (37).
2. Backward recursion: recursively calculates $\hat{\mathcal{I}}_i$ and $\hat{\mathcal{B}}_i$ for each body with (49) and (50).
3. Forward recursion: recursively calculates $\ddot{q}_i$ and $\dot{V}_i$ for each body with (45) and (41).

Table 3 shows the forward dynamics algorithm for open chain systems with a few additional intermediate variables, such as η_i, Ψ_i, Π_i, and β_i, for the simplicity and efficiency of the equations.

Table 3: Recursive Forward Dynamics

```
while forward recursion do
    $T_{\lambda(i),i} = \text{function of } q_i$
    $V_i = \mathrm{Ad}_{T^{-1}_{\lambda(i),i}} V_{\lambda(i)} + S_i \dot{q}_i$
    $\eta_i = \mathrm{ad}_{V_i} S_i \dot{q}_i + \dot{S}_i \dot{q}_i$
end while
while backward recursion do
    $\hat{\mathcal{I}}_i = \mathcal{I}_i + \sum_{k\in\mu(i)} \mathrm{Ad}^*_{T^{-1}_{i,k}} \Pi_k \mathrm{Ad}_{T^{-1}_{i,k}}$
    $\hat{\mathcal{B}}_i = -\mathrm{ad}^*_{V_i} \mathcal{I}_i V_i - F_i^{\text{ext}} + \sum_{k\in\mu(i)} \mathrm{Ad}^*_{T^{-1}_{i,k}} \beta_k$
    $\Psi_i = (S_i^{\mathrm{T}} \hat{\mathcal{I}}_i S_i)^{-1}$
    $\Pi_i = \hat{\mathcal{I}}_i - \hat{\mathcal{I}}_i S_i \Psi_i S_i^{\mathrm{T}} \hat{\mathcal{I}}_i$
    $\beta_i = \hat{\mathcal{B}}_i + \hat{\mathcal{I}}_i \Big\{ \eta_i + S_i \Psi_i \Big( \tau_i - S_i^{\mathrm{T}} \big( \hat{\mathcal{I}}_i \eta_i + \hat{\mathcal{B}}_i \big) \Big) \Big\}$
end while
while forward recursion do
    $\ddot{q}_i = \Psi_i \Big\{ \tau_i - S_i^{\mathrm{T}} \hat{\mathcal{I}}_i \Big( \mathrm{Ad}_{T^{-1}_{\lambda(i),i}} \dot{V}_{\lambda(i)} + \eta_i \Big) - S_i^{\mathrm{T}} \hat{\mathcal{B}}_i \Big\}$
    $\dot{V}_i = \mathrm{Ad}_{T^{-1}_{\lambda(i),i}} \dot{V}_{\lambda(i)} + S_i \ddot{q}_i + \eta_i$
    $F_i = \hat{\mathcal{I}}_i \dot{V}_i + \hat{\mathcal{B}}_i$
end while
```

2.3 Recursive Hybrid Dynamics

A geometric recursive algorithm for hybrid dynamics of open chain systems was reported in [5], but the formulation was based on 1-dof joints. Here, more general form of the geometric hybrid dynamics is presented to support multi-degree-of-freedom joints more conveniently.

One can derive the hybrid dynamics for open chain systems with a similar way as in 2.2 for forward dynamics. Let's go back to (47). Unlike the case of forward dynamics, as $\ddot{q}_k$ in (47) corresponding to an active joint is already known, one can arrange (47) in terms of $\dot{V}_{\lambda(i)}$ by substituting (45) into (47) for $\ddot{q}_k$ of passive joints only as follows:

$$F_{\lambda(i)} = \hat{\mathcal{I}}_{\lambda(i)} \dot{V}_{\lambda(i)} + \hat{\mathcal{B}}_{\lambda(i)} \tag{51}$$

where

$$\begin{aligned}\hat{\mathcal{I}}_{\lambda(i)} &= \mathcal{I}_{\lambda(i)} + \sum_{k\in\{\mu(\lambda(i))\cap u\}} \mathrm{Ad}^*_{T^{-1}_{\lambda(i),k}} \hat{\mathcal{I}}_k \mathrm{Ad}_{T^{-1}_{\lambda(i),k}} \\ &\quad + \sum_{k\in\{\mu(\lambda(i))\cap v\}} \mathrm{Ad}^*_{T^{-1}_{\lambda(i),k}} \left\{\hat{\mathcal{I}}_k - \hat{\mathcal{I}}_k S_k \left(S_k^{\mathrm{T}}\hat{\mathcal{I}}_k S_k\right)^{-1} S_k^{\mathrm{T}}\hat{\mathcal{I}}_k\right\}\mathrm{Ad}_{T^{-1}_{\lambda(i),k}}\end{aligned} \tag{52}$$

$$\begin{aligned}\hat{\mathcal{B}}_{\lambda(i)} &= -\mathrm{ad}^*_{V_{\lambda(i)}}\mathcal{I}_{\lambda(i)}V_{\lambda(i)} - F^{\mathrm{ext}}_{\lambda(i)} \\ &\quad + \sum_{k\in\{\mu(\lambda(i))\cap u\}} \mathrm{Ad}^*_{T^{-1}_{\lambda(i),k}} \left\{\hat{\mathcal{B}}_k + \hat{\mathcal{I}}_k\left(\mathrm{ad}_{V_k}S_k\dot{q}_k + \dot{S}_k\dot{q}_k + S_k\ddot{q}_k\right)\right\} \\ &\quad + \sum_{k\in\{\mu(\lambda(i))\cap v\}} \mathrm{Ad}^*_{T^{-1}_{\lambda(i),k}} \left\{\hat{\mathcal{B}}_k + \hat{\mathcal{I}}_k\left(\mathrm{ad}_{V_k}S_k\dot{q}_k + \dot{S}_k\dot{q}_k\right)\right. \\ &\quad \left. + \hat{\mathcal{I}}_k S_k \left(S_k^{\mathrm{T}}\hat{\mathcal{I}}_k S_k\right)^{-1}\left(\tau_k - S_k^{\mathrm{T}}\hat{\mathcal{I}}_k(\mathrm{ad}_{V_k}S_k\dot{q}_k + \dot{S}_k\dot{q}_k) - S_k^{\mathrm{T}}\hat{\mathcal{B}}_k\right)\right\}.\end{aligned} \tag{53}$$

where 'u' and 'v' denote the sets of the acceleration-prescribed and torque-specified joints in the system respectively. Table 4 shows the hybrid dynamics algorithm for open chain systems.

Table 4: Recursive Hybrid Dynamics

```
while forward recursion do
    $T_{\lambda(i),i} =$ function of $q_i$
    $V_i = \mathrm{Ad}_{T^{-1}_{\lambda(i),i}} V_{\lambda(i)} + S_i\dot{q}_i$
    $\eta_i = \mathrm{ad}_{V_i} S_i\dot{q}_i + \dot{S}_i\dot{q}_i$
end while
while backward recursion do
    $\hat{\mathcal{I}}_i = \mathcal{I}_i + \sum_{k\in\mu(i)} \mathrm{Ad}^*_{T^{-1}_{i,k}}\Pi_k\mathrm{Ad}_{T^{-1}_{i,k}}$
    $\hat{\mathcal{B}}_i = -\mathrm{ad}^*_{V_i}\mathcal{I}_iV_i - F^{\mathrm{ext}}_i + \sum_{k\in\mu(i)} \mathrm{Ad}^*_{T^{-1}_{i,k}}\beta_k$
    if $i \in u$ then
        $\Pi_i = \hat{\mathcal{I}}_i$
        $\beta_i = \hat{\mathcal{B}}_i + \hat{\mathcal{I}}_i(\eta_i + S_i\ddot{q}_i)$
    else
        $\Psi_i = (S_i^{\mathrm{T}}\hat{\mathcal{I}}_iS_i)^{-1}$
        $\Pi_i = \hat{\mathcal{I}}_i - \hat{\mathcal{I}}_iS_i\Psi_iS_i^{\mathrm{T}}\hat{\mathcal{I}}_i$
        $\beta_i = \hat{\mathcal{B}}_i + \hat{\mathcal{I}}_i\Big\{\eta_i + S_i\Psi_i\Big(\tau_i - S_i^{\mathrm{T}}\big(\hat{\mathcal{I}}_i\eta_i + \hat{\mathcal{B}}_i\big)\Big)\Big\}$
    end if
end while
while forward recursion do
    if $i \in u$ then
        $\dot{V}_i = \mathrm{Ad}_{T^{-1}_{\lambda(i),i}}\dot{V}_{\lambda(i)} + S_i\ddot{q}_i + \eta_i$
        $F_i = \hat{\mathcal{I}}_i\dot{V}_i + \hat{\mathcal{B}}_i$
        $\tau_i = S_i^{\mathrm{T}}F_i$
    else
        $\ddot{q}_i = \Psi_i\Big\{\tau_i - S_i^{\mathrm{T}}\hat{\mathcal{I}}_i\Big(\mathrm{Ad}_{T^{-1}_{\lambda(i),i}}\dot{V}_{\lambda(i)} + \eta_i\Big) - S_i^{\mathrm{T}}\hat{\mathcal{B}}_i\Big\}$
        $\dot{V}_i = \mathrm{Ad}_{T^{-1}_{\lambda(i),i}}\dot{V}_{\lambda(i)} + S_i\ddot{q}_i + \eta_i$
        $F_i = \hat{\mathcal{I}}_i\dot{V}_i + \hat{\mathcal{B}}_i$
    end if
end while
```

3 Dynamics of Constrained Systems

Let $q \in \Re^n$ denote a set of joint coordinates in an unconstrained open chain system, and $A(q)\dot{q} = 0$ represents a set of m constraints enforced to the system ($A \in \Re^{m\times n}$). For a closed loop system, for example, a (virtual) unconstrained open chain system can be obtained by cutting the joint loops, and the closed loop constraints to be enforced can be written as $f(q) = 0$, which can be linearized as the form of $A\dot{q} = 0$ where $A = \frac{\partial f}{\partial q}$ is the Jacobian matrix of the constraints.

The equations of motion of the constrained system can be written as:

$$M\ddot{q} + b = \tau + A^{\mathrm{T}}\lambda \tag{54}$$

$$A\dot{q} = 0 \tag{55}$$

where (54) represents the equations of motion of the unconstrained open chain system with an additional constraint force $J^{\mathrm{T}}\lambda$ acting on the configuration space q, and the Lagrange multipliers $\lambda \in \Re^m$ give the relative magnitudes of the constraint forces.

Similarly to the previous section (§2) we will derive inverse dynamics ($\ddot{q} \to \tau$), forward dynamics ($\tau \to \ddot{q}$) and hybrid dynamics ($(\ddot{q}_u, \tau_v) \to (\tau_u, \ddot{q}_v)$) for the constrained system where $q = (q_u, q_v)$. We assume the current system state $(q, \dot{q})$ is known and consistent with the constraints, and A is full rank.

3.1 Inverse Dynamics of Constrained Systems

Inverse dynamics finds the joint torques (τ) satisfying the equations of motion when the current system state (q, $\dot{q}$) and the acceleration ($\ddot{q}$) are given. We assume the state and prescribed acceleration $(q, \dot{q}, \ddot{q})$ satisfy the constraints $A\dot{q} = 0$ and its differential $A\ddot{q} + \dot{A}\dot{q} = 0$, and focus on solving the equations of motion of the unconstrained system with the additional constraint force in (54).

The inverse dynamics solution for a constrained system may not be unique depending on the number of constraints and actuators. For example, when all joints can be actuated, any arbitrary constraint force λ and its corresponding joint torques $\tau = M\ddot{q} + b - A^{\mathrm{T}}\lambda$ becomes a solution of (54), and one particular solution is $\tau = M\ddot{q} + b$ and $\lambda = 0$ where the system acts as if it were a unconstrained open-loop system. In general, if the system is redundantly actuated or there are too many actuators, the joint torques that are required to make the system move in a prescribed way are not unique, and in this case, we often want to choose an optimal one. On the contrary, if the number of actuators is fewer than the minimum necessary to move the system, there would be no solution as expected.

Let $q = (q_a, q_p)$ where q_a and q_p denote the actuated joint coordinates and non-actuated or passive joint coordinates respectively. The passive joint torques τ_p need not to be zero but they are assumed to be determined by the system state. We want to find the active joint torques (τ_a) satisfying (54) or

$$\begin{pmatrix} \tau_{ar} \\ \tau_{pr} \end{pmatrix} = \begin{pmatrix} \tau_a + A_a^{\mathrm{T}}\lambda \\ \tau_p + A_p^{\mathrm{T}}\lambda \end{pmatrix} \tag{56}$$

where $\tau_r = M\ddot{q} + b$ is the inverse dynamics solution for the unconstrained open chain system and the subscripts 'a' and 'p' denote the components for the active and passive coordinates respectively and the subscript 'r' is for a solution of unconstrained dynamics. Since we already know τ_p, we can obtain the constraint force λ by solving the lower row of (56) or

$$A_p^{\mathrm{T}}\lambda = \tau_{pr} - \tau_p. \tag{57}$$

If $m = n_p$ where m and n_p denote the numbers of constraints and the non-actuated coordinates respectively ($A_p \in \Re^{m\times n_p}$), a unique solution $\lambda = \left(A_p^{\mathrm{T}}\right)^{-1}(\tau_{pr} - \tau_p)$ exists[7]. From the upper row of (56), the active joint torques can be obtained as

$$\tau_a = \tau_{ar} - A_a^{\mathrm{T}}\left(A_p^{\mathrm{T}}\right)^{-1}(\tau_{pr} - \tau_p). \tag{58}$$

For example, in a planar five-bar linkage system where the closed loop constrains two translation and one rotation axes ($m = 3$), if there are two active joints (i.e., three of the five joints are passive and $n_p = 3$), we can obtain unique active joints from (58) as long as A_p is full rank or the configuration is not singular.

However, if $m > n_p$ (e.g., a planar five-bar linkage with three or more active joints)[8], there are infinite number of solutions for (57) which can be written as

$$\lambda = \left(A_p^{\mathrm{T}}\right)^{\dagger}(\tau_{pr} - \tau_p) + \mathcal{N}(A_p^{\mathrm{T}}) \tag{59}$$

where $(\cdot)^{\dagger}$ and $\mathcal{N}(\cdot)$ denote the Moore-Penrose pseudoinverse and the null space of a matrix respectively, and this leads to a general solution for the active joint torques as follows.

$$\begin{aligned} \tau_a &= \tau_{ar} - A_a^{\mathrm{T}}\lambda && (60) \\ &= \tau_{ar} - A_a^{\mathrm{T}}\left\{\left(A_p^{\mathrm{T}}\right)^{\dagger}(\tau_{pr} - \tau_p) + \mathcal{N}(A_p^{\mathrm{T}})\right\} && (61) \end{aligned}$$

We can choose an optimal solution here:

- Constraint force minimization: The solution becomes $\tau_a = \tau_{ar} - A_a^{\mathrm{T}}\lambda^*$ where $\lambda^* = \left(A_p^{\mathrm{T}}\right)^{\dagger}(\tau_{pr} - \tau_p)$.

- Minimum active joint torques: We can find this by exploring the null space of A_p^{T} in (61). For example, in the case of $n > m > n_p$,

$$\begin{aligned} \tau_a &= \left(\mathbf{1} - A_a^{\mathrm{T}}N\left(A_a^{\mathrm{T}}N\right)^{\dagger}\right)\left\{\tau_{ar} - A_a^{\mathrm{T}}\left(A_p^{\mathrm{T}}\right)^{\dagger}(\tau_{pr} - \tau_p)\right\} && (62) \\ \lambda &= \left(A_p^{\mathrm{T}}\right)^{\dagger}(\tau_{pr} - \tau_p) + Ns^* && (63) \\ s^* &= \left(A_a^{\mathrm{T}}N\right)^{\dagger}\left\{\tau_{ar} - A_a^{\mathrm{T}}\left(A_p^{\mathrm{T}}\right)^{\dagger}(\tau_{pr} - \tau_p)\right\} && (64) \end{aligned}$$

[7]We assume A is full rank.

[8]We call this a redundantly actuated system.

minimizes $\|\tau_a\|^2$ where $\mathbf{1} \in \Re^{n_a \times n_a}$ denotes the identity matrix and $N \in \Re^{m\times(m-n_p)}$ represents a basis of the null space of $A_p^{\mathrm{T}} \in \Re^{n_p \times m}$. As a special case of this, if all joints are actuated ($n > m, n_p = 0$), N becomes an identity matrix and thus the minimum joint torques and the corresponding constraint forces can be written as

$$\tau = \tau_r - A^{\mathrm{T}}\lambda \tag{65}$$

$$\lambda = \left(A^{\mathrm{T}}\right)^{\dagger}\tau_r = \left(AA^{\mathrm{T}}\right)^{-1} A\tau_r, \tag{66}$$

which can also be directly deduced from (56). In the case of $n < m$, $\tau_a = 0$ becomes the minimal torque solution, which means the system forms an immobile structure due to too many constraints.

In the case of $m < n_p$ (e.g., a planar five-bar linkage with only one active joint), we cannot find active joint torques making the system move exactly as prescribed (q, $\dot{q}$, $\ddot{q}$). However, even in such a system, we can still find a solution for the active joint torques, that must be applied in order to make the active joints follow a prescribed acceleration ($\ddot{q}_a$) exactly, while letting the other passive joints move in a passive way. See hybrid dynamics of constrained systems (§3.3) for this.

3.2 Forward Dynamics of Constrained Systems

In contrast to the inverse dynamics of constrained systems whose solution depends on the system property and configuration, forward dynamics which finds the resulting acceleration ($\ddot{q}$) in response to the applied joint torques (τ) always has a unique solution.

By differentiating the constraint equations (55) to get $A\ddot{q} + \dot{A}\dot{q} = 0$ and combining this with (54), we can obtain the constraint forces as

$$\lambda = -\left(AM^{-1}A^{\mathrm{T}}\right)^{-1}\left(A\ddot{q}_r + \dot{A}\dot{q}\right) \tag{67}$$

where $\ddot{q}_r = M^{-1}(\tau - b)$ is the forward dynamics solution of the unconstrained open chain system with assuming there is no constraint [2]. Once the constraint forces are obtained from (67), we can get the joint acceleration by solving the forward dynamics of the unconstrained open chain system (54) with considering the constraint forces.

In order to improve the computational efficiency in (67), we can exploit the $O(n)$ recursive forward dynamics algorithm for open chain systems described in §2.2. The recursive algorithm can be used not only for obtaining $\ddot{q}_r$ but also for calculating $M^{-1}A^{\mathrm{T}}$ by running the algorithm m times. More specifically, we can set the joint torque with the i-th column of A^{T} call the forward dynamics algorithm (but with ignoring the term b, i.e., the term related with gravitation and joint velocities) to calculate $M^{-1}A_i^{\mathrm{T}}$, and repeat this for all columns. Note that, during the repetition, we can save much computation by reusing the intermediate quantities calculated in the dynamics algorithm.

We can additionally save computation by splitting (54) into two equations

$$M\ddot{q} + b = \tau + A^{\mathrm{T}}\lambda \quad \longrightarrow \quad \begin{cases} M\ddot{q}_r + b & = \tau \\ M\ddot{q}_c & = A^{\mathrm{T}}\lambda \end{cases} \tag{68}$$

and this reveals that the solution of the forward dynamics of a constrained system can be obtained by summing the two solutions from the unconstrained open chain system, i.e.,

$$\ddot{q} = \ddot{q}_r + \ddot{q}_c \tag{69}$$

where $\ddot{q}_r$ is the solution of the unconstrained open chain system with assuming there is no constraint, and $\ddot{q}_c$ is the solution when applied the constraint forces to the open chain system with ignoring the term b as described above. In total, we need a single call of the full version of the recursive open-chain forward dynamics algorithm ($\ddot{q} = M^{-1}(\tau - b)$) and $m+1$ calls of the simplified version of the dynamics algorithm ($\ddot{q} = M^{-1}\tau$). The complexity of the resulting algorithm becomes $O(n)+O(m^3)$ where m denotes the number of constraints and $O(m^3)$ is for inverting $AM^{-1}A^T$.

3.3 Hybrid Dynamics of Constrained Systems

Let $q = (q_u, q_v)$ where q_u and q_v are the joint coordinates with prescribed accelerations and given or known torques respectively. The equations of motion of the constrained system can be rewritten as

$$M \begin{pmatrix} \ddot{q}_u \\ \ddot{q}_v \end{pmatrix} + b = \begin{pmatrix} \tau_u \\ \tau_v \end{pmatrix} + A^{\mathrm{T}}\lambda \tag{70}$$

$$A \begin{pmatrix} \ddot{q}_u \\ \ddot{q}_v \end{pmatrix} + \dot{A}\dot{q} = 0, \tag{71}$$

and an hybrid dynamics algorithm finds (τ_u, $\ddot{q}_v$) from given ($\ddot{q}_u$, τ_v) by solving the constrained dynamics equations.

We first show the constraint forces λ can be obtained in a similar form as in (67). From (70) we get

$$\ddot{q}_v = M_{vv}^{-1}\left(\tau_v - b_v - M_{uv}\ddot{q}_u + A_v^{\mathrm{T}}\lambda\right) \tag{72}$$

where $M = \left[\begin{smallmatrix} M_{uu} & M_{uv} \\ M_{uv} & M_{vv} \end{smallmatrix}\right]$, $b = \left(\begin{smallmatrix} b_u \\ b_v \end{smallmatrix}\right)$ and $A = \left[\begin{smallmatrix} A_u & A_v \end{smallmatrix}\right]$. Plugging this to (71) leads to

$$\left(A_v M_{vv}^{-1} A_v^{\mathrm{T}}\right)\lambda = -\left\{A_u\ddot{q}_u + A_v M_{vv}^{-1}\left(\tau_v - b_v - M_{uv}\ddot{q}_u\right) + \dot{A}\dot{q}\right\}. \tag{73}$$

If $m \leq n_v$ where m and n_v denote the numbers of constraints and the torque-specified coordinates ($A_v \in \Re^{m\times n_v}, M_{vv} \in \Re^{n_v \times n_v}$), there exists a unique solution for the constraint forces λ, and this leads to a unique solution for (τ_u, $\ddot{q}_v$).[9] In this case, the constraint forces can be obtained as follows.

$$\begin{aligned}\lambda &= -\left(A_v M_{vv}^{-1} A_v^{\mathrm{T}}\right)^{-1}\left\{A_u\ddot{q}_u + A_v M_{vv}^{-1}\left(\tau_v - b_v - M_{uv}\ddot{q}_u\right) + \dot{A}\dot{q}\right\} &(74)\\ &= -\left(A_v M_{vv}^{-1} A_v^{\mathrm{T}}\right)^{-1}\left\{A_u\ddot{q}_u + A_v\ddot{q}_{rv} + \dot{A}\dot{q}\right\} &(75)\\ &= -\left(A_v M_{vv}^{-1} A_v^{\mathrm{T}}\right)^{-1}\left(A\ddot{q}_r + \dot{A}\dot{q}\right) &(76)\end{aligned}$$

where

$$\ddot{q}_{rv} = M_{vv}^{-1}\left(\tau_v - b_v - M_{uv}\ddot{q}_u\right) \tag{77}$$

[9]Again, we assume A is full rank.

denotes the resulting acceleration of the torque-specified coordinates in the unconstrained open chain system and $\ddot{q}_r = (\ddot{q}_u, \ddot{q}_{rv})$ denotes the acceleration of the unconstrained system. Note that $\ddot{q}_u$ is the prescribed acceleration and $\ddot{q}_{rv}$ can be efficiently obtained by using the recursive algorithm described in §2.3.

Similarly to the case of forward dynamics (§3.2), $M_{vv}^{-1} A_v^{\mathrm{T}}$ can be efficiently obtained by calling a simplified version of the recursive open chain hybrid dynamics algorithm m times. More specifically, as can be easily recognized from (77), $M_{vv}^{-1} A_{vi}^{\mathrm{T}}$ can be interpreted as the resulting acceleration of the torque-specified coordinates when the term related to gravity and joint velocities is ignored and the joint command input to the unconstrained open chain system is set to ($\ddot{q}_u = 0$, $\tau_v = A_{vi}^{\mathrm{T}}$) where A_{vi}^{T} denotes the i-th column of A_v^{T}.

Once we obtain the constraint forces λ from (76), we can obtain (τ_u, $\ddot{q}_v$) by solving (70) using the recursive hybrid dynamics algorithm for open chain systems. However, splitting (70) into two equations can lead to more efficient computation as we already discussed in §3.2. The equation can be rewritten as

$$M \begin{pmatrix} \ddot{q}_u \\ \ddot{q}_v \end{pmatrix} + b = \begin{pmatrix} \tau_u \\ \tau_v \end{pmatrix} + A^{\mathrm{T}} \lambda \quad \longrightarrow \quad \begin{cases} M \begin{pmatrix} \ddot{q}_u \\ \ddot{q}_{vr} \end{pmatrix} + b & = \begin{pmatrix} \tau_{ur} \\ \tau_v \end{pmatrix} \\ M \begin{pmatrix} 0 \\ \ddot{q}_{vc} \end{pmatrix} & = \begin{pmatrix} \tau_{uc} \\ A_v^{\mathrm{T}} \lambda \end{pmatrix} \end{cases} \tag{78}$$

where $(\ddot{q}_u, \tau_v)$ and $(0, A_v^{\mathrm{T}} \lambda)$ are the inputs of the two hybrid dynamics problems for the unconstrained open chain system respectively. This separation reveals that the solution of the original hybrid dynamics problem for the constrained system can be obtained as

$$\tau_u = \tau_{ur} + \tilde{\tau}_{uc} \tag{79}$$

$$\ddot{q}_v = \ddot{q}_{vr} + \ddot{q}_{vc} \tag{80}$$

where $\tilde{\tau}_{uc} = \tau_{uc} - A_u^{\mathrm{T}} \lambda$. Note that the hybrid dynamics solution exactly matches with the forward dynamics solution in §3.2 when $q_v = q$.

If $m > n_v$, the constraint forces are not uniquely determined, which is similar to the case of inverse dynamics when there are more actuators than necessary ($m > n_p$). In this case, if $\ddot{q}_u$ was set arbitrary, there may not exist a solution for $\ddot{q}_v$ satisfying the constraint equations (71) or $A_v \ddot{q}_v = -A_u \ddot{q}_u - \dot{A} \dot{q}$. If $\ddot{q}_u$ has been set carefully so that there exist accelerations satisfying the constraints, the joint torques for the acceleration-prescribed coordinates (τ_u) has an infinite number of solutions and an optimal solution can be chosen by exploring the null space of A_v^{T}. The inverse dynamics solution (61) provides a simpler formulation where the subscripts 'a' and 'p' must be replaced with 'u' and 'v' respectively for the notation in hybrid dynamics.

4 Differentiation of the Geometric Dynamics

4.1 Basic Derivatives

It is useful to have the following derivatives for differentiating the recursive dynamics algorithms with the chain rule.

Lemma 3. *Let $p \in \Re$ be an arbitrary scalar variable and $T \in SE(3)$ be a function of p. Then*

$$\frac{\partial}{\partial p}\mathrm{Ad}_T = \mathrm{ad}_{\frac{\partial T}{\partial p}T^{-1}}\mathrm{Ad}_T. \tag{81}$$

Proof. Let ξ be an arbitrary $se(3)$. If Ad_T is regarded as a 6×6 matrix, then

$$\frac{\partial}{\partial p}\left(\mathrm{Ad}_T\xi\right) = \frac{\partial}{\partial p}\left(\mathrm{Ad}_T\right)\,\xi + \mathrm{Ad}_T\,\frac{\partial \xi}{\partial p}. \tag{82}$$

Ad_T can also be regarded as a linear mapping, $\mathrm{Ad}_T : \xi \to T\xi T^{-1}$, and in this case,

$$\begin{aligned}
\frac{\partial}{\partial p}\left(\mathrm{Ad}_T\xi\right) &= \frac{\partial}{\partial p}\left(T\xi T^{-1}\right) && (83)\\
&= \frac{\partial T}{\partial p}\xi T^{-1} + T\frac{\partial \xi}{\partial p}T^{-1} + T\xi\frac{\partial T^{-1}}{\partial p} && (84)\\
&= \frac{\partial T}{\partial p}T^{-1}T\xi T^{-1} - T\xi T^{-1}\frac{\partial T}{\partial p}T^{-1} + T\frac{\partial \xi}{\partial p}T^{-1} && (85)\\
&= \mathrm{ad}_{\frac{\partial T}{\partial p}T^{-1}}\mathrm{Ad}_T\xi + \mathrm{Ad}_T\frac{\partial \xi}{\partial p}. && (86)
\end{aligned}$$

As $\xi \in se(3)$ is arbitrary, one can see $\frac{\partial}{\partial p}\mathrm{Ad}_T = \mathrm{ad}_{\frac{\partial T}{\partial p}T^{-1}}\mathrm{Ad}_T$ from (82) and (86). □

Corollary 1. *Let $p \in \Re$ be an arbitrary scalar variable and $T_{\lambda(i),i} \in SE(3) : \{\lambda(i)\} \to \{i\}$. Then*

$$\begin{aligned}
\frac{\partial}{\partial p}\mathrm{Ad}_{T^{-1}_{\lambda(i),i}} &= -\mathrm{ad}_{\frac{\partial h_i}{\partial p}}\mathrm{Ad}_{T^{-1}_{\lambda(i),i}} && (87)\\
\frac{\partial}{\partial p}\mathrm{Ad}^*_{T^{-1}_{\lambda(i),i}} &= -\mathrm{Ad}^*_{T^{-1}_{\lambda(i),i}}\mathrm{ad}^*_{\frac{\partial h_i}{\partial p}} && (88)
\end{aligned}$$

where $\frac{\partial h_i}{\partial p} \in se(3)$ is defined as

$$\frac{\partial h_i}{\partial p} = T^{-1}_{\lambda(i),i}\frac{\partial T_{\lambda(i),i}}{\partial p}. \tag{89}$$

Corollary 2. *If $p = q_i^k$ where q_i^k denotes the k-th coordinate of the i-th joint, then*

$$\frac{\partial h_i}{\partial p} = S_i^k \tag{90}$$

where $S_i^k \in se(3)$ denotes the k-th column of the i-th joint Jacobian, $S_i \in (se(3) \times n_i)$. If $p \notin q_i = \{q_i^1, \cdots, q_i^{n_i}\}$, then

$$\frac{\partial h_i}{\partial p} = 0. \tag{91}$$

4.2 Derivatives of the Dynamics

By applying chain rule with (87) and (88), the recursive algorithm for inverse, forward, and hybrid dynamics can be differentiated with respect to an arbitrary scalar variable $p \in \Re$. Table 5 shows the derivative of the recursive inverse dynamics, and it can solve $\frac{\partial \tau}{\partial p}$ with given $\left(\frac{\partial q}{\partial p}, \frac{\partial \dot{q}}{\partial p}, \frac{\partial \ddot{q}}{\partial p}\right)$. In Table 6, the derivative of the recursive forward dynamics, which calculates $\frac{\partial \ddot{q}}{\partial p}$ with given $\frac{\partial \tau}{\partial p}$, is given. The derivative of the recursive hybrid dynamics which solves $\left(\frac{\partial \tau_u}{\partial p}, \frac{\partial \ddot{q}_v}{\partial p}\right)$ with given $\left(\frac{\partial q}{\partial p}, \frac{\partial \dot{q}}{\partial p}, \frac{\partial \ddot{q}_u}{\partial p}, \frac{\partial \tau_v}{\partial p}\right)$ is presented in Table 7.

Table 5: Derivative of the Recursive Inverse Dynamics

```
while forward recursion do
    $\frac{\partial h_i}{\partial p} \triangleq T_{\lambda(i),i}^{-1} \frac{\partial T_{\lambda(i),i}}{\partial p}$
    $\frac{\partial V_i}{\partial p} = \mathrm{Ad}_{T_{\lambda(i),i}^{-1}} \frac{\partial V_{\lambda(i)}}{\partial p} - \mathrm{ad}_{\frac{\partial h_i}{\partial p}} \mathrm{Ad}_{T_{\lambda(i),i}^{-1}} V_{\lambda(i)} + \frac{\partial S_i}{\partial p} \dot{q}_i + S_i \frac{\partial \dot{q}_i}{\partial p}$
    $\frac{\partial \dot{V}_i}{\partial p} = \mathrm{Ad}_{T_{\lambda(i),i}^{-1}} \frac{\partial \dot{V}_{\lambda(i)}}{\partial p} - \mathrm{ad}_{\frac{\partial h_i}{\partial p}} \mathrm{Ad}_{T_{\lambda(i),i}^{-1}} \dot{V}_{\lambda(i)} + \frac{\partial S_i}{\partial p} \ddot{q}_i + S_i \frac{\partial \ddot{q}_i}{\partial p}$
            $+ \mathrm{ad}_{\frac{\partial V_i}{\partial p}} S_i \dot{q}_i + \mathrm{ad}_{V_i} \left( \frac{\partial S_i}{\partial p} \dot{q}_i + S_i \frac{\partial \dot{q}_i}{\partial p} \right) + \frac{\partial \dot{S}_i}{\partial p} \dot{q}_i + \dot{S}_i \frac{\partial \dot{q}_i}{\partial p}$
end while
while backward recursion do
    $\frac{\partial F_i}{\partial p} = \frac{\partial \mathcal{I}_i}{\partial p} \dot{V}_i + \mathcal{I}_i \frac{\partial \dot{V}_i}{\partial p} - \mathrm{ad}^*_{\frac{\partial V_i}{\partial p}} \mathcal{I}_i V_i - \mathrm{ad}^*_{V_i} \left( \frac{\partial \mathcal{I}_i}{\partial p} V_i + \mathcal{I}_i \frac{\partial V_i}{\partial p} \right) - \frac{\partial F_i^{\mathrm{ext}}}{\partial p}$
            $+ \sum_{k \in \mu(i)} \mathrm{Ad}^*_{T_{i,k}^{-1}} \left( \frac{\partial F_k}{\partial p} - \mathrm{ad}^*_{\frac{\partial h_k}{\partial p}} F_k \right)$
    $\frac{\partial \tau_i}{\partial p} = \frac{\partial S_i}{\partial p}^{\mathrm{T}} F_i + S_i^{\mathrm{T}} \frac{\partial F_i}{\partial p}$
end while
```

Table 6: Derivative of the Recursive Forward Dynamics

```
while forward recursion do
    $\frac{\partial h_i}{\partial p} \triangleq T_{\lambda(i),i}^{-1} \frac{\partial T_{\lambda(i),i}}{\partial p}$
    $\frac{\partial V_i}{\partial p} = \mathrm{Ad}_{T_{\lambda(i),i}^{-1}} \frac{\partial V_{\lambda(i)}}{\partial p} - \mathrm{ad}_{\frac{\partial h_i}{\partial p}} \mathrm{Ad}_{T_{\lambda(i),i}^{-1}} V_{\lambda(i)} + \frac{\partial S_i}{\partial p} \dot{q}_i + S_i \frac{\partial \dot{q}_i}{\partial p}$
    $\frac{\partial \eta_i}{\partial p} = \mathrm{ad}_{\frac{\partial V_i}{\partial p}} S_i \dot{q}_i + \mathrm{ad}_{V_i} \left( \frac{\partial S_i}{\partial p} \dot{q}_i + S_i \frac{\partial \dot{q}_i}{\partial p} \right) + \frac{\partial \dot{S}_i}{\partial p} \dot{q}_i + \dot{S}_i \frac{\partial \dot{q}_i}{\partial p}$
end while
while backward recursion do
    $\frac{\partial \hat{\mathcal{I}}_i}{\partial p} = \frac{\partial \mathcal{I}_i}{\partial p} + \sum_{k \in \mu(i)} \mathrm{Ad}^*_{T_{i,k}^{-1}} \left\{ \frac{\partial \Pi_k}{\partial p} - \Pi_k \mathrm{ad}_{\frac{\partial h_k}{\partial p}} - \left( \Pi_k \mathrm{ad}_{\frac{\partial h_k}{\partial p}} \right)^{\mathrm{T}} \right\} \mathrm{Ad}_{T_{i,k}^{-1}}$
    $\frac{\partial \hat{\mathcal{B}}_i}{\partial p} = -\mathrm{ad}^*_{\frac{\partial V_i}{\partial p}} \mathcal{I}_i V_i - \mathrm{ad}^*_{V_i} \left( \frac{\partial \mathcal{I}_i}{\partial p} V_i + \mathcal{I}_i \frac{\partial V_i}{\partial p} \right) - \frac{\partial F_i^{\mathrm{ext}}}{\partial p}$
            $+ \sum_{k \in \mu(i)} \mathrm{Ad}^*_{T_{i,k}^{-1}} \left( \frac{\partial \beta_k}{\partial p} - \mathrm{ad}^*_{\frac{\partial h_k}{\partial p}} \beta_k \right)$
    $\frac{\partial \Psi_i}{\partial p} = -\Psi_i \left\{ S_i^{\mathrm{T}} \frac{\partial \hat{\mathcal{I}}_i}{\partial p} S_i + \frac{\partial S_i}{\partial p}^{\mathrm{T}} \hat{\mathcal{I}}_i S_i + \left( \frac{\partial S_i}{\partial p}^{\mathrm{T}} \hat{\mathcal{I}}_i S_i \right)^{\mathrm{T}} \right\} \Psi_i$
    $\frac{\partial \Pi_i}{\partial p} = \frac{\partial \hat{\mathcal{I}}_i}{\partial p} - \Big\{ \hat{\mathcal{I}}_i S_i \frac{\partial \Psi_i}{\partial p} S_i^{\mathrm{T}} \hat{\mathcal{I}}_i + \frac{\partial \hat{\mathcal{I}}}{\partial p} S_i \Psi_i S_i^{\mathrm{T}} \hat{\mathcal{I}}_i + \left( \frac{\partial \hat{\mathcal{I}}}{\partial p} S_i \Psi_i S_i^{\mathrm{T}} \hat{\mathcal{I}}_i \right)^{\mathrm{T}}$
```

```
            $+ \hat{\mathcal{I}}_i \frac{\partial S_i}{\partial p} \Psi_i S_i^{\mathrm{T}} \hat{\mathcal{I}}_i + \left( \hat{\mathcal{I}}_i \frac{\partial S_i}{\partial p} \Psi_i S_i^{\mathrm{T}} \hat{\mathcal{I}}_i \right)^{\mathrm{T}} \Big\}$
    $\frac{\partial \beta_i}{\partial p} = \frac{\partial \hat{\mathcal{B}}_i}{\partial p} + \frac{\partial \hat{\mathcal{I}}_i}{\partial p} \Big\{ \eta_i + S_i \Psi_i \Big( \tau_i - S_i^{\mathrm{T}} \big( \hat{\mathcal{I}}_i \eta_i + \hat{\mathcal{B}}_i \big) \Big) \Big\}$
            $+ \hat{\mathcal{I}} \Big\{ \frac{\partial \eta_i}{\partial p} + \Big( \frac{\partial S_i}{\partial p} \Psi_i + S_i \frac{\partial \Psi_i}{\partial p} \Big) \Big( \tau_i - S_i^{\mathrm{T}} \big( \hat{\mathcal{I}}_i \eta_i + \hat{\mathcal{B}}_i \big) \Big)$
            $+ S_i \Psi_i \Big( \frac{\partial \tau_i}{\partial p} - \frac{\partial S_i}{\partial p}^{\mathrm{T}} \big( \hat{\mathcal{I}}_i \eta_i + \hat{\mathcal{B}}_i \big) - S_i^{\mathrm{T}} \big( \frac{\partial \hat{\mathcal{I}}_i}{\partial p} \eta_i + \hat{\mathcal{I}}_i \frac{\partial \eta_i}{\partial p} + \frac{\partial \hat{\mathcal{B}}_i}{\partial p} \big) \Big) \Big\}$
end while
while forward recursion do
    $\frac{\partial \ddot{q}_i}{\partial p} = \frac{\partial \Psi_i}{\partial p} \Big\{ \tau_i - S_i^{\mathrm{T}} \hat{\mathcal{I}}_i \big( \mathrm{Ad}_{T_{\lambda(i),i}^{-1}} \dot{V}_{\lambda(i)} + \eta_i \big) - S_i^{\mathrm{T}} \hat{\mathcal{B}}_i \Big\}$
            $+ \Psi_i \Big\{ \frac{\partial \tau_i}{\partial p} - \Big( \frac{\partial S_i}{\partial p}^{\mathrm{T}} \hat{\mathcal{I}}_i + S_i^{\mathrm{T}} \frac{\partial \hat{\mathcal{I}}_i}{\partial p} \Big) \big( \mathrm{Ad}_{T_{\lambda(i),i}^{-1}} \dot{V}_{\lambda(i)} + \eta_i \big)$
            $- S_i^{\mathrm{T}} \hat{\mathcal{I}}_i \Big( \mathrm{Ad}_{T_{\lambda(i),i}^{-1}} \frac{\partial \dot{V}_{\lambda(i)}}{\partial p} - \mathrm{ad}_{\frac{\partial h_i}{\partial p}} \mathrm{Ad}_{T_{\lambda(i),i}^{-1}} \dot{V}_{\lambda(i)} + \frac{\partial \eta_i}{\partial p} \Big)$
            $- \frac{\partial S_i}{\partial p}^{\mathrm{T}} \hat{\mathcal{B}}_i - S_i^{\mathrm{T}} \frac{\partial \hat{\mathcal{B}}_i}{\partial p} \Big\}$
    $\frac{\partial \dot{V}_i}{\partial p} = \mathrm{Ad}_{T_{\lambda(i),i}^{-1}} \frac{\partial \dot{V}_{\lambda(i)}}{\partial p} - \mathrm{ad}_{\frac{\partial h_i}{\partial p}} \mathrm{Ad}_{T_{\lambda(i),i}^{-1}} \dot{V}_{\lambda(i)} + \frac{\partial S_i}{\partial p} \ddot{q}_i + S_i \frac{\partial \ddot{q}_i}{\partial p} + \frac{\partial \eta_i}{\partial p}$
    $\frac{\partial F_i}{\partial p} = \frac{\partial \hat{\mathcal{I}}_i}{\partial p} \dot{V}_i + \hat{\mathcal{I}}_i \frac{\partial \dot{V}_i}{\partial p} + \frac{\partial \hat{\mathcal{B}}_i}{\partial p}$
end while
```

Table 7: Derivative of the Recursive Hybrid Dynamics

```
while forward recursion do
    $\frac{\partial h_i}{\partial p} \triangleq T_{\lambda(i),i}^{-1} \frac{\partial T_{\lambda(i),i}}{\partial p}$
    $\frac{\partial V_i}{\partial p} = \mathrm{Ad}_{T_{\lambda(i),i}^{-1}} \frac{\partial V_{\lambda(i)}}{\partial p} - \mathrm{ad}_{\frac{\partial h_i}{\partial p}} \mathrm{Ad}_{T_{\lambda(i),i}^{-1}} V_{\lambda(i)} + \frac{\partial S_i}{\partial p} \dot{q}_i + S_i \frac{\partial \dot{q}_i}{\partial p}$
    $\frac{\partial \eta_i}{\partial p} = \mathrm{ad}_{\frac{\partial V_i}{\partial p}} S_i \dot{q}_i + \mathrm{ad}_{V_i} \Big( \frac{\partial S_i}{\partial p} \dot{q}_i + S_i \frac{\partial \dot{q}_i}{\partial p} \Big) + \frac{\partial \dot{S}_i}{\partial p} \dot{q}_i + \dot{S}_i \frac{\partial \dot{q}_i}{\partial p}$
end while
while backward recursion do
    $\frac{\partial \hat{\mathcal{I}}_i}{\partial p} = \frac{\partial \mathcal{I}_i}{\partial p} + \sum_{k \in \mu(i)} \mathrm{Ad}^*_{T_{i,k}^{-1}} \Big\{ \frac{\partial \Pi_k}{\partial p} - \Pi_k \mathrm{ad}_{\frac{\partial h_k}{\partial p}} - \Big( \Pi_k \mathrm{ad}_{\frac{\partial h_k}{\partial p}} \Big)^{\mathrm{T}} \Big\} \mathrm{Ad}_{T_{i,k}^{-1}}$
    $\frac{\partial \hat{\mathcal{B}}_i}{\partial p} = -\mathrm{ad}^*_{\frac{\partial V_i}{\partial p}} \mathcal{I}_i V_i - \mathrm{ad}^*_{V_i} \Big( \frac{\partial \mathcal{I}_i}{\partial p} V_i + \mathcal{I}_i \frac{\partial V_i}{\partial p} \Big) - \frac{\partial F_i^{\mathrm{ext}}}{\partial p}$
            $+ \sum_{k \in \mu(i)} \mathrm{Ad}^*_{T_{i,k}^{-1}} \Big( \frac{\partial \beta_k}{\partial p} - \mathrm{ad}^*_{\frac{\partial h_k}{\partial p}} \beta_k \Big)$
    if $i \in u$ then
        $\frac{\partial \Pi_i}{\partial p} = \frac{\partial \hat{\mathcal{I}}_i}{\partial p}$
        $\frac{\partial \beta_i}{\partial p} = \frac{\partial \hat{\mathcal{B}}_i}{\partial p} + \frac{\partial \hat{\mathcal{I}}_i}{\partial p} (\eta_i + S_i \ddot{q}_i) + \hat{\mathcal{I}}_i \Big( \frac{\partial \eta_i}{\partial p} + \frac{\partial S_i}{\partial p} \ddot{q}_i + S_i \frac{\partial \ddot{q}_i}{\partial p} \Big)$
    else
        $\frac{\partial \Psi_i}{\partial p} = -\Psi_i \Big\{ S_i^{\mathrm{T}} \frac{\partial \hat{\mathcal{I}}_i}{\partial p} S_i + \frac{\partial S_i}{\partial p}^{\mathrm{T}} \hat{\mathcal{I}}_i S_i + \Big( \frac{\partial S_i}{\partial p}^{\mathrm{T}} \hat{\mathcal{I}}_i S_i \Big)^{\mathrm{T}} \Big\} \Psi_i$
        $\frac{\partial \Pi_i}{\partial p} = \frac{\partial \hat{\mathcal{I}}_i}{\partial p} - \Big\{ \hat{\mathcal{I}}_i S_i \frac{\partial \Psi_i}{\partial p} S_i^{\mathrm{T}} \hat{\mathcal{I}}_i + \frac{\partial \hat{\mathcal{I}}}{\partial p} S_i \Psi_i S_i^{\mathrm{T}} \hat{\mathcal{I}}_i + \Big( \frac{\partial \hat{\mathcal{I}}}{\partial p} S_i \Psi_i S_i^{\mathrm{T}} \hat{\mathcal{I}}_i \Big)^{\mathrm{T}}$
```

```
                $+ \hat{\mathcal{I}}_i \frac{\partial S_i}{\partial p} \Psi_i S_i^{\mathrm{T}} \hat{\mathcal{I}}_i + \left( \hat{\mathcal{I}}_i \frac{\partial S_i}{\partial p} \Psi_i S_i^{\mathrm{T}} \hat{\mathcal{I}}_i \right)^{\mathrm{T}} \Big\}$
        $\frac{\partial \beta_i}{\partial p} = \frac{\partial \hat{\mathcal{B}}_i}{\partial p} + \frac{\partial \hat{\mathcal{I}}_i}{\partial p} \Big\{ \eta_i + S_i \Psi_i \Big( \tau_i - S_i^{\mathrm{T}} \big( \hat{\mathcal{I}}_i \eta_i + \hat{\mathcal{B}}_i \big) \Big) \Big\}$
                $+ \hat{\mathcal{I}} \Big\{ \frac{\partial \eta_i}{\partial p} + \Big( \frac{\partial S_i}{\partial p} \Psi_i + S_i \frac{\partial \Psi_i}{\partial p} \Big) \Big( \tau_i - S_i^{\mathrm{T}} \big( \hat{\mathcal{I}}_i \eta_i + \hat{\mathcal{B}}_i \big) \Big)$
                $+ S_i \Psi_i \Big( \frac{\partial \tau_i}{\partial p} - \frac{\partial S_i}{\partial p}^{\mathrm{T}} \big( \hat{\mathcal{I}}_i \eta_i + \hat{\mathcal{B}}_i \big) - S_i^{\mathrm{T}} \big( \frac{\partial \hat{\mathcal{I}}_i}{\partial p} \eta_i + \hat{\mathcal{I}}_i \frac{\partial \eta_i}{\partial p} + \frac{\partial \hat{\mathcal{B}}_i}{\partial p} \big) \Big) \Big\}$
    end if
end while
while forward recursion do
    if $i \in u$ then
        $\frac{\partial \dot{V}_i}{\partial p} = \mathrm{Ad}_{T^{-1}_{\lambda(i),i}} \frac{\partial \dot{V}_{\lambda(i)}}{\partial p} - \mathrm{ad}_{\frac{\partial h_i}{\partial p}} \mathrm{Ad}_{T^{-1}_{\lambda(i),i}} \dot{V}_{\lambda(i)} + \frac{\partial S_i}{\partial p} \ddot{q}_i + S_i \frac{\partial \ddot{q}_i}{\partial p} + \frac{\partial \eta_i}{\partial p}$
        $\frac{\partial F_i}{\partial p} = \frac{\partial \hat{\mathcal{I}}_i}{\partial p} \dot{V}_i + \hat{\mathcal{I}}_i \frac{\partial \dot{V}_i}{\partial p} + \frac{\partial \hat{\mathcal{B}}_i}{\partial p}$
        $\frac{\partial \tau_i}{\partial p} = \frac{\partial S_i}{\partial p}^{\mathrm{T}} F_i + S_i^{\mathrm{T}} \frac{\partial F_i}{\partial p}$
    else
        $\frac{\partial \ddot{q}_i}{\partial p} = \frac{\partial \Psi_i}{\partial p} \Big\{ \tau_i - S_i^{\mathrm{T}} \hat{\mathcal{I}}_i \big( \mathrm{Ad}_{T^{-1}_{\lambda(i),i}} \dot{V}_{\lambda(i)} + \eta_i \big) - S_i^{\mathrm{T}} \hat{\mathcal{B}}_i \Big\}$
                $+ \Psi_i \Big\{ \frac{\partial \tau_i}{\partial p} - \Big( \frac{\partial S_i}{\partial p}^{\mathrm{T}} \hat{\mathcal{I}}_i + S_i^{\mathrm{T}} \frac{\partial \hat{\mathcal{I}}_i}{\partial p} \Big) \big( \mathrm{Ad}_{T^{-1}_{\lambda(i),i}} \dot{V}_{\lambda(i)} + \eta_i \big)$
                $- S_i^{\mathrm{T}} \hat{\mathcal{I}}_i \Big( \mathrm{Ad}_{T^{-1}_{\lambda(i),i}} \frac{\partial \dot{V}_{\lambda(i)}}{\partial p} - \mathrm{ad}_{\frac{\partial h_i}{\partial p}} \mathrm{Ad}_{T^{-1}_{\lambda(i),i}} \dot{V}_{\lambda(i)} + \frac{\partial \eta_i}{\partial p} \Big)$
                $- \frac{\partial S_i}{\partial p}^{\mathrm{T}} \hat{\mathcal{B}}_i - S_i^{\mathrm{T}} \frac{\partial \hat{\mathcal{B}}_i}{\partial p} \Big\}$
        $\frac{\partial \dot{V}_i}{\partial p} = \mathrm{Ad}_{T^{-1}_{\lambda(i),i}} \frac{\partial \dot{V}_{\lambda(i)}}{\partial p} - \mathrm{ad}_{\frac{\partial h_i}{\partial p}} \mathrm{Ad}_{T^{-1}_{\lambda(i),i}} \dot{V}_{\lambda(i)} + \frac{\partial S_i}{\partial p} \ddot{q}_i + S_i \frac{\partial \ddot{q}_i}{\partial p} + \frac{\partial \eta_i}{\partial p}$
        $\frac{\partial F_i}{\partial p} = \frac{\partial \hat{\mathcal{I}}_i}{\partial p} \dot{V}_i + \hat{\mathcal{I}}_i \frac{\partial \dot{V}_i}{\partial p} + \frac{\partial \hat{\mathcal{B}}_i}{\partial p}$
    end if
end while
```

To obtain the derivatives of the dynamics, it is needed to run the associated recursive dynamics in advance, and in addition, the following quantities

$$\frac{\partial S_i}{\partial p}, \frac{\partial \dot{S}_i}{\partial p}, \frac{\partial F_i^{\mathrm{ext}}}{\partial p}, \frac{\partial \mathcal{I}_i}{\partial p}, \text{and } \frac{\partial h_i}{\partial p} \tag{92}$$

for each body should be set properly in the initialization step.

The algorithms for the derivatives of the dynamics are so general that they can be applied to differentiating the equations of motion with respect to any arbitrary scalar variable. It should be noted that, by implementing the algorithms cleverly, the calculation speed can be much faster than its naive implementation, because, in some cases, many of the quantities in (92) are zero. For example,

- if $p = q^k$ where $q^k \in \Re$ denotes the k-th coordinate of system, then
  - $\frac{\partial \mathcal{I}_i}{\partial p} = 0$
  - $\frac{\partial S_i}{\partial p} = \frac{\partial \dot{S}_i}{\partial p} = \frac{\partial h_i}{\partial p} = 0$ when $q^k \notin q_i$

- if $p = \dot{q}^k$, then
  - $\frac{\partial S_i}{\partial p} = \frac{\partial \mathcal{I}_i}{\partial p} = \frac{\partial h_i}{\partial p} = 0$
  - $\frac{\partial \dot{S}_i}{\partial p} = 0$ when $q^k \notin q_i$
- if $p = \ddot{q}^k$, then
  - $\frac{\partial \dot{S}_i}{\partial p} = \frac{\partial S_i}{\partial p} = \frac{\partial \mathcal{I}_i}{\partial p} = \frac{\partial h_i}{\partial p} = 0.$

References

[1] Roy Featherstone, *Robot Dynamics Algorithms*, Kluver Academic Publishers, 1987.

[2] Richard M. Murray, Zexiang Li and and S. Shankar Sastry, *A Mathematical Introduction to Robotic Manipulation*, CRC Press, 1994.

[3] F. C. Park, J. E. Bobrow and S. R. Ploen, “A Lie group formulation of robot dynamics,” *International Journal of Robotics Research*, vol. 14, no. 6, pp. 609-618, 1995.

[4] S. R. Ploen and F. C. Park, “Coordinate-invariant algorithms for robot dynamics,” *IEEE Transactions on Robotics and Automation*, vol. 15, no. 6, pp. 1130-1136, 1999.

[5] Garett A. Sohl and James E. Bobrow, “A recursive multibody dynamics and sensitivity algorithm for branched kinematic chains,” *Journal of Dynamic Systems, Measurement, and Control*, vol. 123, pp. 391-399, 2001.