Efficient geometric sensitivity analysis of moving-base multibody dynamics

Bos, M.P.

Award date:
2019

EINDHOVEN UNIVERSITY OF TECHNOLOGY

Efficient Geometric Sensitivity Analysis of Moving-Base Multibody Dynamics

M.P. Bos (0774887)
Department of Mechanical Engineering
Dynamics and Control group
Eindhoven University of Technology
DC 2019.060

Project Coaching
dr. A. Saccon
dr. S. Traversaro (IIT, Genoa, Italy)

Project Supervision
Prof. dr. H. Nijmeijer

Committee
Prof. dr. H. Nijmeijer
dr. A. Saccon
dr. ir. R. Duits
dr. S. Traversaro (IIT, Genoa, Italy)

June 3, 2019

Abstract

Model-predictive control and trajectory optimization are powerful numerical tools to generate reference trajectories and stabilize the motion of complex robot systems, such as robot manipulators, drones, humanoids, or quadrupeds. At their core, these tools require the computation of the sensitivity of the system's dynamics with respect to the system's state and input about an equilibrium point or a nominal trajectory. This approximation of the dynamics is also known as linearization. For fixed-base systems such as industrial robot manipulators, several methods exist to compute this linearization, including finite difference approximation, Lagrangian derivation, automatic differentiation, and recursive analytical derivation. For moving-base systems such as drones, humanoids, or quadrupeds, existing methods for fixed-base systems can be applied to obtain the linearization but they then suffer from singularity issues caused by the chosen representation of the orientation of the moving base. In this thesis, instead, we propose a geometric linearization method for complex moving-base systems that is mathematically elegant, computationally efficient, and singularity-free. Computational efficiency is obtained by proposing new recursive algorithms for computing the derivatives of the inverse dynamics and the inverse of the mass matrix of the robotic system. Numerical validation of the approach by means of a comparison with the linearization obtained via geometric finite difference is presented.

Keywords: sensitivity analysis, dynamics linearization, moving-base system, differential geometry, singularity-free, analytical derivatives, multibody dynamics, recursive algorithms, forward dynamics, inverse dynamics

Acknowledgments

Over the past year I have learned more than I could ever imagine, in a topic that I never thought I could understand. With the help of certain people, I did manage to understand, and thanks to them I can happily look back on a great time during the most challenging project I have ever worked on.

The person who I want to thank the most is Alessandro Saccon, who taught me mathematics up to levels I could had never imagined me understanding. Besides technical knowledge, under his supervision I learned proper writing, reporting, and most importantly, the feeling of true responsibility by letting me make my own choices, which does come with uncertainty and pressure. Alessandro's critical feedback, combined with honest compliments and appreciation, have absolutely been a great motivation for me to always go for the best.

Another person I could not have done without is Silvio Traversaro, who not only helped me through the toughest parts of my internship, but also celebrated every success with me. The fact that I can always count on him made me feel safe, even though I was working on an unfamiliar topic, in an unfamiliar culture, surrounded by unfamiliar people. Silvio's never-ending enthusiasm has definitely motivated me to keep working on the same topic during my graduation project, where his input and feedback have continued to be a great help.

Also I would like to thank Daniele Pucci and Henk Nijmeijer. Daniele found time to teach me the fundamentals of Lie group theory in private lessons, even though he is the head of a department of 20-25 highly skilled roboticists. These lessons have been a great help in clearing up the chaos of differential geometry. Henk adviced me from the start of my master's and gave me the final insightful comments to polish up my thesis.

My thanks also go to all the people at the Dynamic Interaction Control lab of Istituto Italiano di Tecnologia (IIT). The culture and colleagues at IIT made me feel more than welcome and it has been a great pleasure to share thoughts and knowledge about rotational dynamics and control. Also I want to thank the fellow students at the Robotics Lab of the Eindhoven University of Technology for sharing the process of graduation and having interesting chats about a wide variety of topics in robotics which definitely helped me on my own project. Of course I can not forget my friends at Simon Stevin and UniPartners Eindhoven, with whom I have shared the best moments of my student life, and will probably share many more great moments.

Special thanks go to my parents, Herman and Lidia, for providing me with endless motivation and freedom. I know that I can always count on their support, no matter what my choices are, and for that I am truly grateful. I would like to thank my grandparents, Paul and Anny (or Opa en Oma), for encouraging me to pursue my passion for science and technology since I was a little kid. Finally, I would like to thank Jeanine, for her never-ending support.

Nomenclature

Algorithms

The algorithms used in this report are abbreviated as follows.

ABA	Articulated Body Algorithm (for fixed-base systems)
ABAmb	Articulated Body Algorithm for moving-base systems
CRBA	Composite Rigid Body Algorithm (for fixed-base systems)
CRBAmb	Composite Rigid Body Algorithm for moving-base systems
EIDAmb	Extended Inverse Dynamics Algorithm for moving-base systems
GBWAmb	Generalized Bias Wrench Algorithm for moving-base systems
IMMA	Inverse Mass Matrix Algorithm (for fixed-base systems)
IMMAmb	Inverse Mass Matrix Algorithm for moving-base systems
RNEA	Recursive Newton Euler Algorithm (for fixed-base systems)
RNEAmb	Recursive Newton Euler Algorithm for moving-base systems

Mathematical definitions

M, N	Smooth manifolds.
x	Point on a manifold.
T_xM, T_x^*M	Tangent and cotangent spaces of M at x.
TM, T^*M	Tangent and cotangent bundles of M at x.
$f : M \to N$	(Smooth) mapping for M to N.
$\mathrm{D}\, f : TM \to TN$	Tangent map of f.
G	Lie group.
$g \in G$	Element of Lie group.
e	Group identity.
$\cdot_G$	Operation associated to the Lie group G.
$\mathfrak{g}$	Lie algebra of G.
$[\cdot, \cdot]_\mathfrak{g}$	Lie brackets on $\mathfrak{g}$.
$L_g x$, $R_g x$	Left and right translations of $x \in G$ by $g \in G$.
gx , xg	Shorthand notation for $L_g x$, $R_g x$.
gv , vg	Shorthand notation for $\mathrm{D}\, L_g(x) \cdot v$, $\mathrm{D}\, R_g(x) \cdot v$ with $v \in T_xG$.
Ad	Adjoint representation of a Lie group to its algebra.
ad	Adjoint representation of a Lie algebra onto itself.
$S_1 \times S_2$	Cartesian product of sets S_1 and S_2.
$G_1 \times G_2$	Direct product of the Lie groups G_1 and G_2.
$\mathfrak{g}_1 \oplus \mathfrak{g}_2$	Direct sum of the Lie algebras $\mathfrak{g}_1$ and $\mathfrak{g}_2$.
$\exp : \mathfrak{g} \to G$	Exponential map of G.
$\log : G \to \mathfrak{g}$	Logarithm map (inverse of exp in a neighbourhood of e).
SO(3)	Special Orthogonal group of dimension 3.
SE(3)	Special Euclidean group of dimension 3.
$\mathfrak{so}(3)$	Lie algebra of SO(3).
$\mathfrak{se}(3)$	Lie algebra of SE(3).
$\mathbb{R}^3_\times$	Lie algebra given by $\mathbb{R}^3$ with the cross product as Lie bracket.
$\mathbb{R}^6_\times$	Lie algebra given by $\mathbb{R}^6$ with the 6D cross product as Lie bracket.
$\wedge (wedge)$	Lie algebra isomorphism from $\mathbb{R}^6_\times$ to $\mathfrak{se}(3)$ (or from $\mathbb{R}^3_\times$ to $\mathfrak{so}(3)$).
$\vee (vee)$	Lie algebra isomorphism from $\mathfrak{se}(3)$ to $\mathbb{R}^6_\times$ (or from $\mathfrak{so}(3)$ to $\mathbb{R}^3_\times$).
$I_n \in \mathbb{R}^{n\times n}$	Identity matrix of dimension n.
$\partial x/\partial y$	Partial derivative of x with respect to scalar or vector y.
$\tilde{\partial} x/\partial y$	Left-trivialized partial derivative of x with respect to Lie group element y.

Multibody dynamics notation

For variables related to multibody dynamics notation, we refer to Section 2.7, where they are ordered by category.

Contents

Chapter 1

Introduction

1.1 Motivation

Model-predictive control [1, 2, 3] and trajectory optimization [4, 5] are used more and more in the control of complex robot systems, such as humanoids and quadrupeds. These methods require the computation of the sensitivity of the control vector field with respect the state and input variables of the system about an equilibrium or nominal trajectory. This sensitivity is also known as dynamics linearization.

Sensitivity analysis of the control vector field about an equilibrium or a nominal trajectory estimates the effect of variations in variables on the associated perturbed solutions [6, Section 3.3]. These variables can be the initial conditions, input or system parameters such as mass or length of bodies. When working with a system that evolves on a vector space

$$\dot{x}(t) = f(x, u, t), \tag{1.1}$$

where $x(t) \in \mathbb{R}^n$ is the system state, $u(t) \in \mathbb{R}^m$ the system input, $t \in \mathbb{R}$ time, and $f : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R} \to \mathbb{R}^n$ the control vector field, the sensitivity of the control vector field f with respect to the system state x and input u about a nominal trajectory $\eta(x(t), u(t))$ has taken the form of the linear system

$$\dot{z}(t) = A(\eta, t)z(t) + B(\eta, t)w(t), \tag{1.2}$$

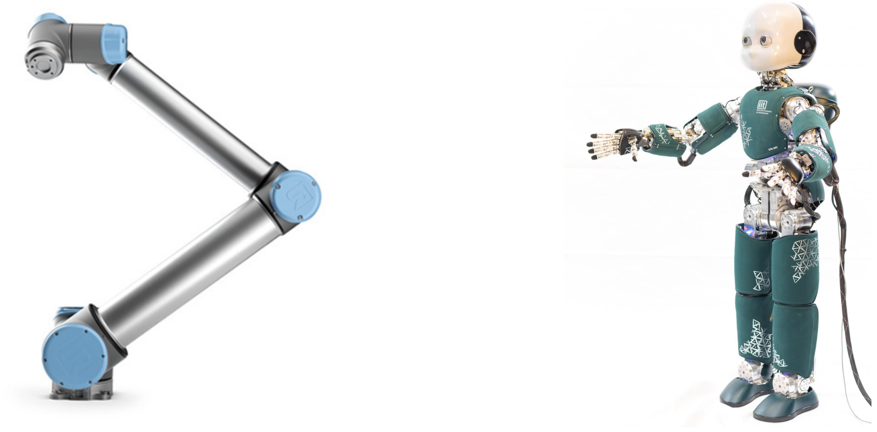
where $z(t) \in \mathbb{R}^n$ is the perturbation vector, $w(t) \in \mathbb{R}^m$ the perturbed input vector, $A(\eta, t) \in \mathbb{R}^{n\times n}$ the state matrix and $B(\eta, t) \in \mathbb{R}^{n\times m}$ the input matrix. The state matrix and input matrices are found by computing the partial derivative of function f with respect to the state x and input u along a nominal state-input curve ($A = \partial f/\partial x, B = \partial f/\partial u$).

For fixed-base systems, such as industrial robot manipulators, whose configuration spaces are vector spaces, computationally efficient algorithms exist to compute the state and input matrices of (1.2) [7]. For moving-base systems (also known as floating-base systems [8]) such as drones, humanoids, and quadrupeds, the configuration space is naturally a Lie group $\mathrm{SE}(3) \times \mathbb{R}^n$. By means of local parametrization of the orientation (using, e.g., Euler or Tait-Bryan angles), the configuration space of a moving-base system can be artificially considered as that of a fixed-base system. Therefore, algorithms for computing the sensitivity analysis for fixed-based systems can be applied to moving-based systems, although this gives rise to (parameterization) singularity issues [9, 10]. In this thesis, we explore how to obtain the sensitivity of a moving-base system treating the configuration space for what it is, namely, the Lie group $\mathrm{SE}(3) \times \mathbb{R}^n$, without resorting to any local parametrization of the rotation part.

An example of a fixed-base robot is the UR10 robot arm[1] by Universal Robots. An example of

[1]Picture retrieved from `https://www.universal-robots.com/nl/producten/ur10-robot/`

of a moving-base robot is the iCub[2], developed by the Istituto Italiano di Tecnologia. Both systems are depicted in Figure 1.1.



(a) Universal Robot's UR10. (b) iCub by Istituto Italiano di Tecnologia.

Figure 1.1: A fixed-base (left) and a moving-base (right) robot.

When a system evolves on a Lie group G, computing the sensitivity needs a different approach [11, 12], as the system state $x \in G$ does not evolve on a vector space. In [11], the authors have provided a definition of sensitivity analysis for systems evolving on a generic Lie group and demonstrated it numerically on the rotational dynamics of a rigid body on $T\mathrm{SO}(3)$. In this approach, the linearized system $\dot{z} = Az + Bw$ evolves on the Lie algebra of the Lie group. This theory can applied to full rigid body dynamics (both translational and rotational) defined on $T\mathrm{SE}(3)$ [13, Chapter 4], and even to moving-base dynamics defined on $T(\mathrm{SE}(3) \times \mathbb{R}^n)$ [14, 15, 16], making it possible to compute the singularity-free sensitivity for moving-base multibody dynamics.

Moving-base robots are typically controlled at a frequency of kilohertz, and the amount of computations increases rapidly as the number of degrees of freedom increases. As theories such as model-predictive control and trajectory optimization recompute multiple time-steps ahead during each time-step, the computational time needed to compute the sensitivity is a limiting factor. A numerically efficient method to compute the sensitivity is required to allow on-line applications.

1.2 Literature review

In this section we present an overview of related literature to our research. We investigate sensitivity analysis on Lie groups [12] and left-trivialized linearization on Lie groups [11, 17], [18, Section II.G]. After that, recursive algorithms [8, 19, 20] are discussed, as they are the foundation of efficient methods to compute rigid body dynamics. Lastly, four known methods to compute or approximate the sensitivity of multibody systems are discussed: finite differences [5], Lagrangian derivation [21], automatic differentiation [22], and recursive analytical derivation [7].

Sensitivity on Lie groups and left-trivialized linearization. The theory of left-trivialized linearization is presented in [11, 17]. It allows to compute the sensitivity for control systems on Lie groups, by using the fact that a Lie algebra can be modelled as a vector space. An infinitesimal perturbation is imposed on the input and state, so that the sensitivity can either be mathematically computed or numerically approximated. In [11], the authors apply this method to linearize

[2]Picture retrieved from `https://www.eurekalert.org/multimedia/pub/158868.php?from=380282`

rotational dynamics on $T\mathrm{SO}(3)$, demonstrated in the application of optimal control. The presented example on $T\mathrm{SO}(3)$ needs addition of translational dynamics of the moving base as well as added dynamics of the joints, before it applies to moving-base multibody dynamics. The resulting linearization is singularity-free. In [18], a compact introduction to Lie groups in robotics is given, which too shows the concept of left-trivialized linearization in Section II.G. Sensitivity analysis for multibody systems evolving on Lie groups in general is also discussed in [12], however the authors do not show explicit application to rotational or full rigid body dynamics.

Recursive algorithms. Recursive algorithms are highly efficient methods to compute robot dynamics. The equations of motion of robotic systems often include sparse matrices, induced by their branch-like structure. Recursive algorithms exploit the branch-induced sparsity in the equations of motion matrices of robotic systems by omitting unnecessary computations. Three well-known algorithms for robotic systems are the Recursive Newton Euler Algorithm (RNEA) [19], the Composite Rigid Body Algorithm (CRBA) [20] and the Articulated Body Algorithm [8, Chapter 7]. The RNEA computes the inverse dynamics: the necessary joint torques to achieve desired joint accelerations. The CRBA computes the mass matrix. The ABA computes the forward dynamics: the resulting joint accelerations with given joint torque. In [8] an overview of these three algorithms is presented, as well as how to obtain the equations of motion for multibody systems using these three algorithms. Furthermore, [8, Chapter 9] presents dedicated algorithms applied to moving-base systems in a singularity-free way: the Recursive Newton Euler Algorithm for moving-base systems (RNEAmb), the Composite Rigid Body Algorithm for moving-base systems (CRBAmb), and the Articulated Body Algorithm for moving-base systems (ABAmb).

Sensitivity analysis using finite differences. Finite differences is a relatively simple method to approximate the sensitivity. It evaluates the dynamics several times: once unperturbed, and multiple times with an added perturbation for each degree of freedom and for each input variable. For systems with a large number of degrees of freedom (e.g. humanoids or quadrupeds), this method becomes time-consuming. Furthermore, the finite difference method is prone to numerical rounding errors. Despite the time-consuming computations, there is literature available showing successful usage of finite differences on real-time applications. In [23], a method to compute the sensitivity of moving-base systems is presented. The moving-base is modelled in a singularity-free way, although little details describe how this is done. The authors state that their humanoid robot can track squatting trajectories up to 1 Hz without recomputing the sensitivity. The computational time of this method is not shown. In [5], a method is shown to apply model predictive control to humanoids, using the sensitivity computed by finite differences. The authors claim that in their application, almost all CPU time is spent computing the sensitivity. Implementation of this strategy required careful implementation and parallel processing. Moving bases are not mentioned in this paper.

Sensitivity analysis using Lagrangian derivation. Lagrangian derivation is a method to analytically derive the Lagrangian equation of motion, described in [21]. It computes the sensitivity directly from the equations of motion. Therefore, it does not utilize the sparsity in the matrices, resulting in unnecessary computations. The authors mention moving bases, but do not model them in a singularity-free way.

Sensitivity analysis using automatic differentiation. Automatic differentiation applies the chain rule to all operations and functions performed by a computer program in an automatic way. Since the derivatives of basic operations (i.e. addition, subtraction, multiplication) and functions (sin, cos, exp) are known, the chain rule can compute the partial derivatives of certain functions. In [22, 24], this method is applied to multibody systems. A tool called *RobCoGen* is used to automatically generate robot-specific rigid body dynamics code. Automatic differentiation is applied to this code, to compute the the derivatives. Moving bases are mentioned, but not modelled singularity-free. In [25] the author warns that automatic differentiation may give wrong results if it is based on pure syntactical analysis and is implemented without knowledge of the problem structure.

Sensitivity analysis using recursive analytical derivation. Recursive analytical derivation uses recursive algorithms (i.e. RNEA, ABA) and derives the equations in these algorithms step by step, by applying the chain rule. This method requires algebraic differentiation of spatial algebra. In [26] an algorithm which focuses on underactuated systems is presented. This algorithm is a hybrid algorithm: it computes partly forward dynamics (for passive joints) and partly inverse dynamics (for active joints). The configuration space is assumed to be a vector space, therefore the moving-base can not be modelled singularity-free. In [27, Chapter 6] an analytical algorithm computing the sensitivity of the inverse dynamics is presented. Moving-bases are not discussed. The authors of [7] present an analytical algorithm that computes the sensitivity of the inverse dynamics, based on the RNEA. The derivatives of the forward dynamics can be computed by deriving the ABA, although the authors state that it may also be found by using a relation between the inverse and forward dynamics, which result in lower computation times. The computation times needed to derive the sensitivity of both the inverse and forward dynamics are found to be much lower compared to the finite difference method. In [28] the same authors present an algorithm to directly compute the inverse of the mass matrix, without first computing the mass matrix itself. As it is the inverse that is needed, and not the mass matrix itself, this also leads to lower computation times. This method is also found to have lower computational times than the standard approach of using Cholesky decomposition.

1.3 Research objectives and contribution

The goal of this project is to *develop a numerically efficient and accurate method to compute the singularity-free geometric linearization of moving-base multibody systems.*

Requirements. There are four requirements related to this project:

- *(Si-F)* Singularity-free modelling of the moving base.
- *(Se-MB)* Computing the sensitivity of multibody dynamics.
- *(Ef)* Highly efficient computations so that the sensitivity can be computed on-line. We aim to eliminate as many unnecessary computations as possible.
- *(Ac)* Accurate computations. This includes both exact computations (no approximations as in finite differences) and correct computations (no wrong computations as may happen in automatic differentiation).

As far as we are concerned, there is no research yet which combines all these requirements, as is visible in Table 1.1.

Table 1.1: Overview of related theories.

Theory	**Sing-Free**	**Sens-MB**	**Ef**	**Ac**
Left-trivialized linearization [11, 17, 18]	✓			✓
Recursive algorithms [8, 19, 20]	✓		✓	✓
Finite differences [5, 23]	✓	✓		
Lagrangian derivation [21]		✓		✓
Automatic differentiation [22, 24]		✓	✓	
Analytical derivation [7, 26, 27]		✓	✓	✓

The goal can be divided in three objectives:

1. Derive and present the mathematical formulas for the singularity-free geometric linearization of moving-base multibody systems.
2. Derive numerically efficient and accurate algorithms to compute the singularity-free geometric linearization of moving-base multibody systems.
3. Verify the correctness of the derived algorithms.

1.4 Methodology

As discussed in the previous section, we need to combine several algorithms and mathematical tools in order to obtain an efficient and accurate method to compute the singularity-free geometric linearization of moving-base multibody systems. Out of the four existing methods to compute the sensitivity for multibody systems, we choose to continue working on recursive analytical derivation. Finite differences and Lagrangian derivation are simply too slow as proven in [7]. Automatic differentiation can lead to unnecessary computations as suggested in [25, 29], since it does not exploit the full knowledge of sparsity, and it is hard to verify whether intermediate steps are correct. Recursive analytical derivation does offer the desired numerically efficiency and accuracy, and is therefore our preferred method for computing the derivatives. As [7] states that the sensitivity of the forward dynamics is computed faster through computing the sensitivity of the inverse dynamics due to the computational time forward dynamics being higher than that of the inverse dynamics, our methodology is based deriving the same algorithms, but now in a singularity-free manner. Therefore our approach will be based on combining the work of [7] with the theory of left-trivialized linearization presented in [11, 17]. Below we discuss the three objectives in which our goal is divided.

1. Derive and present the mathematical formulas for the singularity-free geometric linearization of moving-base multibody systems.
To derive the mathematical formulas for singularity-free geometric linearization of moving-base multibody systems, we can combine the theory of left-trivialized linearization [11, 17] with equations of motion for moving-base systems [30, Chapter 9], [14, 15, 16]. As we aim to write the sensitivity of the forward dynamics in terms of the inverse dynamics, the relation between both must be found. For fixed-base systems, it is described in [7]. For (underactuated) moving-base systems, however, the relation can not be found easily, as there is no proper definition of inverse dynamics for underactuated systems. By extending the physical systems with virtual inputs, one can create a virtual fully actuated system [31], so that the relation between the forward and inverse dynamics can be found.

2. Derive numerically efficient and accurate algorithms to compute the singularity-free geometric linearization of moving-base multibody systems.

Once the mathematical formulas for the linearization have been derived, we aim to compute them in an efficient and accurate way. Therefore, algorithms need to be written to compute the derivatives of the inverse dynamics and the inverse of the mass matrix. An efficient and accurate algorithm to compute the sensitivity of the inverse dynamics of fixed-base systems is given in [7]. In [8], an algorithm to compute the inverse dynamics for moving-base systems is given. By combining both, and applying left-trivialized linearization, we can write an algorithm that computes the sensitivity of the inverse dynamics of moving-base systems. An efficient and accurate algorithm to compute the inverse of the mass matrix for fixed-base systems is given in [28]. In [8], an algorithm to compute the mass matrix for moving-base systems is given. By combining both, we can write an algorithm that computes the inverse of the mass matrix for moving-base systems.

3. Verify the correctness of the derived algorithms.
Once an efficient and accurate method to compute the linearization is found, we aim to verify it to be correct. There are three parts that require verification: the inverse of the mass matrix, the sensitivity of the inverse dynamics, and the sensitivity of the forward dynamics. Our strategy is to use other methods (which are less numerically efficient) to compute the same quantities. The inverse of the mass matrix for moving-base systems can easily be computed in a way that is known to be correct, by first computed the mass matrix according to the CRBAmb, and then inverting it. Our algorithm can be verified by comparing both inverted matrices. As finite differences are a simple method to compute the linearization, we can easily compare the results from our algorithms to the results of finite differences, taking into account small numerical errors in the finite difference method. The computational time for the verification is not an objective, therefore the lack of computational speed of the finite difference method is no problem. The sensitivities of the inverse dynamics can be found by running the RNEAmb once unperturbed, and once perturbed for each system state. Our algorithm for the inverse dynamics can be verified by comparing both sensitivities. By perturbing the ABAmb, our method for the forward dynamics can be verified in a similar manner.

1.5 Report outline

This report is structured as follows. In Chapter 2 we present the preliminaries related to this thesis. This includes the representations of the pose a moving base, Lie group theory and the Lie group SE(3), sensitivity analysis on vector spaces and Lie groups, multibody system definitions and notation, equations of motion and recursive algorithms.

In Chapter 3 we present the theoretical aspects of left-trivialized linearization of moving-base multibody systems. Firstly the Lie group on which moving-base systems are defined is discussed. Secondly the mathematical formulas for left-trivialized linearization of the forward dynamics are shown. Lastly, we extend the system with (non-physical) inputs to derive the relation between the left-trivialized derivatives of the forward dynamics and those of the extended inverse dynamics. We use this relation to express the derivatives of the forward dynamics in terms of the derivatives of the extended inverse dynamics.

In Chapter 4 we present the algorithmic aspects of left-trivialized linearization of moving-base multibody systems. Firstly we propose a new algorithm: the Extended Inverse Dynamics Algorithm for moving-base systems (EIDAmb), which computes the extended inverse dynamics. Secondly four new algorithms are proposed, which compute the derivatives of the extended inverse dynamics with respect to the four system state variables. Thirdly we propose a new version of the Inverse Mass Matrix Algorithm (IMMA), which computes the inverse of the mass matrix for fixed-base systems. Lastly, we propose a new algorithm: the Inverse Mass Matrix Algorithm for moving-base systems (IMMAmb), which is an extended version of the IMMA.

In Chapter 5 we present numerical verification methods and results, to prove that the proposed algorithms in Chapter 4 provide correct results. Firstly we present a designed moving-base multi-body test system, which we use to test our algorithms on. Secondly, we define the left-trivialized finite differences to verify the correctness of our obtained extended inverse dynamics derivatives. Thirdly we verify our version of the IMMA using the Composite Rigid Body Algorithm for fixed-base systems (CRBA), and fourthly we verify the correctness of the IMMAmb using the Composite Rigid Body Algorithm for moving-base systems (CRBAmb). Lastly we verify the correctness of the derivatives of the forward dynamics obtained through our proposed algorithms in Chapter 4 using the left-trivialized finite differences of the forward dynamics, which we obtain through the Articulated Body Algorithm for moving-base systems (ABAmb).

In Chapter 6 we discuss the conclusions of this thesis and present recommendations for future research.

Chapter 2

Preliminaries

This chapter aims to present an overview of all background information related to efficient geometric sensitivity analysis of moving-base multibody dynamics necessary to understand this thesis, separated in nine topics.

The first section describes commonly used representations of the pose of a moving base, and the advantages and disadvantages of those methods. The second section presents a method for sensitivity analysis of dynamical systems, of which the configuration space is a vector space. The third section presents a compact overview of the basics of Lie group theory and references to more information of this topic. The fourth section presents details on the 3D Special Euclidean group, which is the most important Lie group for this thesis. The fifth section presents a method for sensitivity analysis of dynamical systems of which the configuration space is a Lie group. The sixth section presents definitions regarding multibody systems. The seventh section presents the multibody dynamics notation used in this thesis. The eight section presents the equations of motion for both fixed-base as well as moving-base systems. The ninth section presents an overview on efficient recursive algorithms that are widely used in robotics.

2.1 Representations of the pose of a moving base

The physical attitude and position of a moving base can be modelled by multiple mathematical representations. As a result, its configuration can either be a vector space, or a Lie group.

One example of modelling a moving base is the 3D Special Euclidean group SE(3), which consists of the rotation matrix to represent the attitude, and a 3D vector in Cartesian coordinates, which represents the position. Although the position could be expressed in polar coordinates or spherical coordinates, this is unusual and Cartesian coordinates are largely used in robotic literature.

The attitude however, knows many variants in literature. Examples are Euler angles, Euler axis and angle, unit quaternions, and the rotation matrix [32].

Euler angles use three parameters to describe a physical rotation: three sequential rotations around pre-determined axes. When choosing the right set of pre-determined axes, any physical representation can be obtained with these three sequential rotations. Commonly these are chosen to be the axes of a Cartesian coordinate system. Typically, *proper Euler angles* rotate twice around one axis (e.g. x-y-x sequence), while *Tait-Bryan angles* (or *roll-pitch-yaw*) rotate once around all three axes (e.g. x-y-z sequence). Both proper Euler angles and Tait-Bryan angles are commonly considered to be Euler angles. Euler angles suffer from *singularity* issues, also known as *gimbal lock*: the rotation around the second axis can be so that the first and third axis align. In that case, it is impossible to transform the time derivative of the Euler angles to the angular velocity vector. Therefore, time-invariant continuous control laws using the three parameters of Euler angles can not be *globally defined.* Furthermore, control laws based on Euler angles may suffer from a phenomena called *unwinding* if not carefully designed. As a physical rotation may have multiple mathematical representations in Euler angles (an increment of 360 degrees in any of the sequential rotations results in the initial rotation), therefore we say that Euler angles are not a *unique* representation of a physical rotation. A closed-loop trajectory may not directly go towards its desired attitude equilibrium. Even if the desired equilibrium is close to the initial attitude, the rigid body can make large rotational movements. An example of this is depicted in Figure 2.1, where a planar rotational system shows the phenomenon of unwinding, due to the non-uniqueness of the chosen representation. Here θ is the current position and θ_d is the desired position. The system should rotate $1/4[rad]$ clockwise, however here it rotates $7/4[rad]$ counterclockwise.

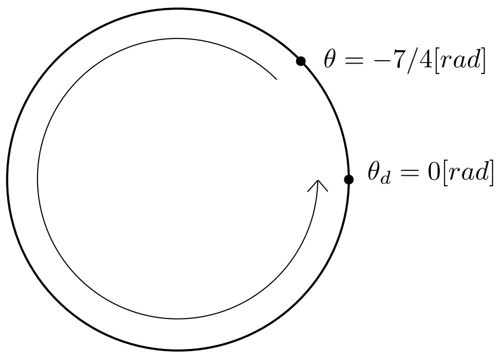


Figure 2.1: Phenomenon of unwinding on a planar rotational system, due to the non-uniqueness of the chosen representation.

Euler axis and angle uses four parameters to describe a physical attitude: three represent a unit vector, and the fourth represents a rotation around that unit vector. The transformation of time derivative of Euler axis and angle to the angular velocity vector is globally defined, meaning that the singularity does not exist for this representation. Although globally defined, unwinding may still occur, as Euler axis and angle are not unique. Again an increment of 360 degrees in the angle results in the initial angle. Furthermore, by taking the opposite direction of the axis, the same physical attitude can be achieved.

Unit quaternions are based on the Euler axis and angle and also uses four parameters. These parameters are scaled by an extension of Euler's formula. The resulting parameters are again globally defined, and due to the scaling, the increment of 360 degrees no longer exists. However, unit quaternions are still not unique, as there are exactly two sets on unit quaternions for each physical attitude, which origins from the axis being able to face two directions. Therefore, time-invariant continuous controllers based on unit quaternions may show the phenomena of unwinding if not carefully designed.

The rotation matrix uses a three-by-three matrix to represent a physical attitude, and therefore involves nine parameters. However, these parameters are under some constraints imposed by the *3D Special Orthogonal group* SO(3). This is a subgroup of the *3D Orthogonal group* O(3), which has three-by-three matrices as elements and matrix multiplication as group operation. The 3D Orthogonal group has the properties that the determinant of an element is equal to either plus or mines one, an element multiplied by its inverse is equal to the identity matrix and an element's transpose is equal to its inverse. The elements having determinant plus one belong to the 3D Special Orthogonal group, and represent physical attitudes. The rotation matrix is both globally defined and unique. The time rate of change of any element can be mapped into the angular velocity vector, and each physical attitude has exactly one mathematical representation. Time-invariant continuous control laws based on the rotation matrix do not suffer from unwinding as the representation is unique. It does suffer from having multiple equilibrium points, namely four, of which three (undesired) equilibrium points are unstable, and only the desired equilibrium is stable. Due to the unstable equilibrium points, global attitude stabilization is impossible. In Figure 2.2, this phenomenon is demonstrated. A planar rotational system is depicted, where position A is the desired (stable) equilibrium, and as a result of a time-invariant continuous control law, there must be a position B that becomes an undesired (unstable) equilibrium.

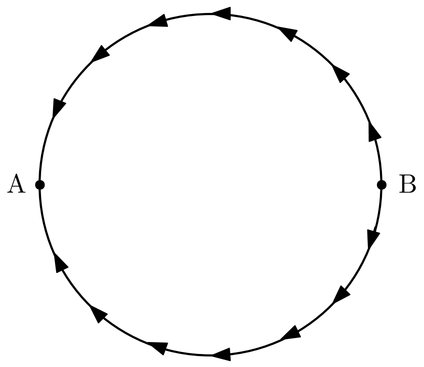


Figure 2.2: The impossibility of global attitude stabilization using a time-invariant continuous control law on a planar rotational system.

Table 2.1 presents an overview of different attitude representations and their properties.

Table 2.1: Summary of attitude representations.

Representation	**Number of parameters**	**Globally defined**	**Unique**
Euler angles	3	No	No
Euler axis and angle	4	Yes	No
Unit quaternions	4	Yes	No
Rotation matrix	9	Yes	Yes

2.2 Sensitivity analysis of dynamical systems on vector spaces

Sensitivity analysis is the study of how the outputs of a nonlinear mathematical function depend on its inputs. For dynamical systems, the mathematical function is the system's dynamics

$$\dot{x} = f(x, u, \lambda, t), \tag{2.1}$$

where x is the (system) state vector, u the input vector and λ design parameters (if any). The state vector is typically a combination of the position and velocity vectors. Sensitivity analysis of nonlinear dynamical systems studies how the time-derivative of the state vector depends on the state vector $\partial f/\partial x$, input vector $\partial f/\partial u$, and design parameters $\partial f/\partial \lambda$. For single-input-single-output systems, these quantities are scalars, while for multi-input-multi-output systems, these quantities are matrices, are therefore called the sensitivity matrices.

As presented in the introduction, the linearization of systems independent of design parameters, $\dot{x} = f(x, u, t)$, about a nominal trajectory $\eta(x(t), u(t))$ is written in state space representation as

$$\dot{z}(t) = A(\eta, t)z(t) + B(\eta, t)w(t), \tag{1.2, revisited}$$

where $z(t)$ is the perturbation vector, $w(t)$ the perturbed input vector, $A(\eta, t)$ the state matrix and $B(\eta, t)$ the input matrix. Since $x(t)$ and $u(t)$ are defined on vector spaces, the perturbation vector

$z(t)$ and perturbed input vector $w(t)$ are typically chosen equal to $x(t)$ and $u(t)$ respectively. The state and input matrices (also called sensitivity matrices) are given by

$$A(\eta, t) = \frac{\partial f(x, u, t)}{\partial x} \tag{2.2}$$

and

$$B(\eta, t) = \frac{\partial f(x, u, t)}{\partial u}. \tag{2.3}$$

It should be noted that sensitivity analysis is performed about a certain trajectory which generally depends on time. Therefore the sensitivity matrices $A(\eta, t)$ and $B(\eta, t)$ generally depend on time and need to be re-evaluated continuously to ensure that the linearized system offers a good approximation to the nonlinear system.

For more information on sensitivity analysis, linearization and perturbation theory, we refer to (respectively) [6, Section 3.3, Section 4.3, and Chapter 10].

2.3 Lie group theory

We aim to describe the pose of a moving base as an element of the Lie group SE(3). Therefore good understanding of Lie group theory is necessary. As Lie groups are a sub-class of groups, we shall first define what a group is. For the interested reader, we suggest the books [33, 34] on general Lie group theory. For roboticists and mechanical engineers, we also advice the book [35] and the papers [18, 32, 36], which discuss Lie groups in the context of geometric mechanics and robotics.

2.3.1 Groups and Lie groups

A group consists of two parts: a set and a binary operation. Applying the operation on two elements of a group results in a third element belonging to the group. Four requirements, called the group axioms, must hold for any group: closure, associativity, the identity element, and the inverse element. We define a group G, which has a set S and a group operation $\cdot$, so that the group is written as $G = (S, \cdot)$. The group axioms are defined as:

- Closure: for each $g_1, g_2 \in S$, $g_1 \cdot g_2 = g_3 \in S$.
- Associativity: for each $g_1, g_2, g_3 \in S$, $(g_1 \cdot g_2) \cdot g_3 = g_1 \cdot (g_2 \cdot g_3)$.
- Identity element: there exists an identity element $e \in S$, such that $e \cdot g = g \cdot e = g$ for any element $g \in S$. There exists only one identity element in a group.
- Inverse element: for each $g_1 \in S$ there exists an inverse element $g_2 \in S$, such that $g_1 \cdot g_2 = g_2 \cdot g_1 = e$.

A Lie group is smooth (differentiable) manifold, satisfying the conditions of a group, as well as the additional condition that the group operations are differentiable. The most simple example of a (Lie) group is the set $\mathbb{R}$, equipped with the operation of addition. We write $G = (\mathbb{R}, +)$. The four group axioms hold:

- Closure: for each $a, b \in \mathbb{R}$, $a + b = c \in \mathbb{R}$.
- Associativity: for each $a, b, c \in \mathbb{R}$, $(a + b) + c = a + (b + c)$.
- Identity element: there exists an identity element $0 \in \mathbb{R}$, such that $0 + a = a + 0 = a$ for any element $a \in \mathbb{R}$. There exists only one identity element in G, which is 0.

- Inverse element: for each $a \in \mathbb{R}$ there exists an inverse element $b(= -a) \in \mathbb{R}$, such that $a + b = b + a = 0$.

Another example of a (Lie) group is the 3D Special Orthogonal group SO(3), which is the group describing 3D rotations. This group consists of the set SO(3), equipped with the operation of matrix multiplication. For clarity, we first describe the requirements of the set SO(3):

- The set SO(3) is a set of three-by-three matrices.
- Each element $\mathbf{R} \in \mathrm{SO}(3)$ has a determinant equal to 1.
- For each element $\mathbf{R} \in \mathrm{SO}(3)$, its transpose is equal to its inverse. Therefore we write $\mathbf{R}\mathbf{R}^T = I_3$.

The group SO(3) is written as $G = (\mathrm{SO}(3), \cdot)$, where $\cdot$ represents matrix multiplication. Again the four group axioms hold:

- Closure: for each $\mathbf{R}_1, \mathbf{R}_2 \in \mathrm{SO}(3)$, $\mathbf{R}_1 \cdot \mathbf{R}_2 = \mathbf{R}_3 \in \mathrm{SO}(3)$.
- Associativity: for each $\mathbf{R}_1, \mathbf{R}_2, \mathbf{R}_3 \in \mathrm{SO}(3)$, $(\mathbf{R}_1 \cdot \mathbf{R}_2) \cdot \mathbf{R}_3 = \mathbf{R}_1 \cdot (\mathbf{R}_2 \cdot \mathbf{R}_3)$.
- Identity element: there exists an identity element $I_3 \in \mathrm{SO}(3)$, such that $I_3 \cdot \mathbf{R} = \mathbf{R} \cdot I_3 = \mathbf{R}$ for any element $\mathbf{R} \in \mathrm{SO}(3)$. There exists only one identity element in SO(3), which is I_3.
- Inverse element: for each $\mathbf{R}_1 \in \mathrm{SO}(3)$ there exists an inverse element $\mathbf{R}_2(= \mathbf{R}_1^{-1}) \in \mathrm{SO}(3)$, such that $\mathbf{R}_1 \cdot \mathbf{R}_2 = \mathbf{R}_2 \cdot \mathbf{R}_1 = I_3$.

2.3.2 Tangent spaces, left- and right-translations, and Lie algebra

Every point x on a manifold M that is smooth (note that the manifold of a Lie group is differentiable by definition) has an associated tangent space T_xM. The tangent space T_xM is a vector space that touches the manifold M at point x. Figure 2.3[1] shows a visualization for the tangent space T_xM of a point x on a sphere M. For the example of the set SO(3), the tangent space $T_{\mathbf{R}}\mathrm{SO}(3)$ contains the element $\dot{\mathbf{R}} \in T_{\mathbf{R}}\mathrm{SO}(3)$, for the specific $\mathbf{R} \in \mathrm{SO}(3)$.

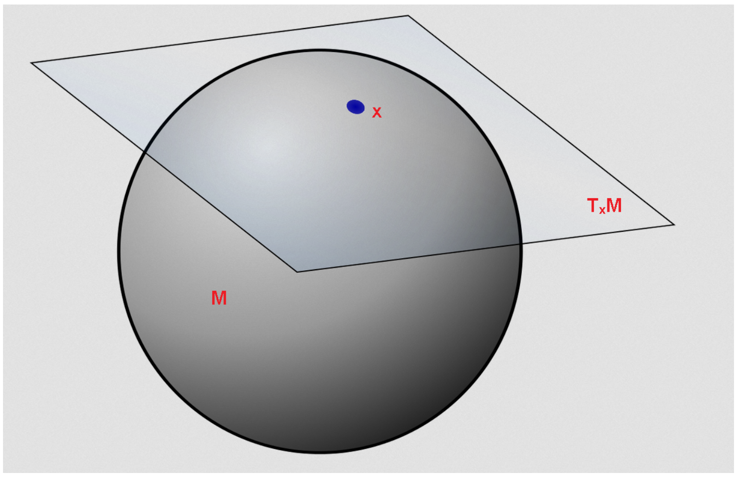


Figure 2.3: The tangent space T_xM of a point x on a sphere M.

All tangent spaces of the manifold M together form the tangent bundle TM. For the example of the set SO(3), the tangent bundle $T\mathrm{SO}(3)$ contains all elements $\dot{\mathbf{R}} \in T\mathrm{SO}(3)$, for all $\mathbf{R} \in \mathrm{SO}(3)$.

[1]Picture retrieved from `https://en.wikipedia.org/wiki/Tangent_space`

The left-translation is a mapping $L_g : G \to G$ defined as

$$L_g h := g \cdot h, \tag{2.4}$$

where G is a Lie group, $g, h \in G$ and $\cdot$ is the group operation. Likewise, the right-translation $R_g : G \to G$ is defined as

$$R_g h := h \cdot g. \tag{2.5}$$

We write $L_g h$ and $R_g h$ in shortened notation as gh and hg respectively. Using left and right translations, one can map a velocity $\dot{h} \in T_h G$ to a different tangent space,

$$T_h L_g \dot{h} = g \cdot \dot{h} \in T_{gh} G \tag{2.6}$$

and

$$T_h R_g \dot{h} = \dot{h} \cdot g \in T_{hg} G, \tag{2.7}$$

which we write in shortened notation as $g\dot{h}$ and $\dot{h}g$ respectively. The velocity can be mapped to the tangent space at the identity of the group by using a specific left-translation

$$\begin{aligned} T_g L_{g^{-1}} \dot{g} &= g^{-1}\dot{g} \in T_{g^{-1}g} G \\ &= g^{-1}\dot{g} \in T_e G. \end{aligned} \tag{2.8}$$

This left-translation of an element and its velocity with the inverse of the element is called *left-trivialization.* The tangent space at the identity element of the Lie group is called the *Lie algebra*, $\mathfrak{g} := T_e G$. Like Lie groups, Lie algebra have a binary operation called the Lie bracket $[\cdot,\cdot]_{\mathfrak{g}} : \mathfrak{g} \times \mathfrak{g} \to \mathfrak{g}$. Applying the Lie bracket on two elements of the Lie algebra results in a third element belonging to the Lie algebra. The Lie algebra is a vector space with an operation satisfying the following axioms, where all $x \in \mathfrak{g}$:

- Bilinearity: $[ax_1 + bx_2, x_3]_{\mathfrak{g}} = a[x_1, x_3]_{\mathfrak{g}} + b[x_2, x_3]_{\mathfrak{g}}$, where $a, b \in \mathbb{R}$ are scalars.
- Anticommutativity (also known as skew-symmetry): $[x_1, x_2]_{\mathfrak{g}} = -[x_2, x_1]_{\mathfrak{g}}$.
- The Jacobi identity: $[x_1, [x_2, x_3]_{\mathfrak{g}}]_{\mathfrak{g}} + [x_3, [x_1, x_2]_{\mathfrak{g}}]_{\mathfrak{g}} + [x_2, [x_3, x_1]_{\mathfrak{g}}]_{\mathfrak{g}} = 0$.

As a Lie algebra is a vector space, its elements can be identified with vectors in $\mathbb{R}^m$ through a diffeomorphism, where m is the number of degrees of freedom of the group G.

For the example of SO(3), with an element $\mathbf{R} \in \mathrm{SO}(3)$ and velocity $\dot{\mathbf{R}} \in T_{\mathbf{R}}\mathrm{SO}(3)$, the velocity is mapped into the Lie algebra by

$$L_{\mathbf{R}^{-1}} \dot{\mathbf{R}} = \mathbf{R}^{-1}\dot{\mathbf{R}} = \mathbf{R}^T\dot{\mathbf{R}} = \boldsymbol{\Omega} \in \mathfrak{so}(3), \tag{2.9}$$

where

$$\boldsymbol{\Omega} := \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \in \mathfrak{so}(3). \tag{2.10}$$

The element $\boldsymbol{\Omega}$ of the Lie algebra is mapped into a vector $\boldsymbol{\omega}$ in $\mathbb{R}^3$, as rotations on SO(3) have three degrees of freedom, by the vee-operator

$$\boldsymbol{\omega} = \boldsymbol{\Omega}^\vee = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix}. \tag{2.11}$$

Likewise, the wedge-operator maps $\boldsymbol{\omega} \in \mathbb{R}^3$ to $\boldsymbol{\Omega} \in \mathfrak{so}(3)$:

$$\boldsymbol{\Omega} = \boldsymbol{\omega}^\wedge. \tag{2.12}$$

This shows that, *as a set*, $\mathbb{R}^3$ is diffeomorphic to $\mathfrak{so}(3)$. As a Lie algebra $\mathfrak{so}(3)$ has Lie brackets $[\boldsymbol{\Omega}_1, \boldsymbol{\Omega}_2]_{\mathfrak{so}(3)} = \boldsymbol{\Omega}_1\boldsymbol{\Omega}_2 - \boldsymbol{\Omega}_2\boldsymbol{\Omega}_1$, which results in an element of the Lie algebra $\mathfrak{so}(3)$. This Lie bracket can be identified by the cross-product as Lie bracket $[\boldsymbol{\omega}_1, \boldsymbol{\omega}_2]_{\mathbb{R}^3_\times} = \boldsymbol{\omega}_1 \times \boldsymbol{\omega}_2$ for the set $\mathbb{R}^3$. We can therefore identify $\mathfrak{so}(3)$, *as a Lie algebra*, with $\mathbb{R}^3_\times$, which is the set $\mathbb{R}^3$ with cross-product as Lie bracket.

2.3.3 Adjoint representations

There exist two adjoint representations: the adjoint representation of a group onto its algebra, Ad, and the adjoint representation of an algebra onto itself, ad. The adjoint representation of a group G (with element $g \in G$) onto its algebra $\mathfrak{g}$ (with element $x \in \mathfrak{g}$) is $\mathrm{Ad}_g : G \times \mathfrak{g} \to \mathfrak{g}$, defined as

$$\mathrm{Ad}_g\, x := gxg^{-1}. \tag{2.13}$$

It is the result of a left-translation by g, and a right-translation by g^{-1} on x. The adjoint representation of Lie group SO(3) onto $\mathfrak{so}(3)$ is $\mathrm{Ad}_{\mathbf{R}} : \mathrm{SO}(3) \times \mathfrak{so}(3) \to \mathfrak{so}(3)$, defined as

$$\mathrm{Ad}_{\mathbf{R}}\, \boldsymbol{\Omega} := \mathbf{R}\boldsymbol{\Omega}\mathbf{R}^{-1}, \tag{2.14}$$

and the adjoint representation of SO(3) onto $\mathbb{R}^3_\times$ is $\mathrm{Ad}_{\mathbf{R}} : \mathrm{SO}(3) \times \mathbb{R}^3_\times \to \mathbb{R}^3_\times$, defined as

$$\mathrm{Ad}_{\mathbf{R}}\, \boldsymbol{\omega} := \mathbf{R}\boldsymbol{\omega}, \tag{2.15}$$

therefore we also write $\mathrm{Ad}_{\mathbf{R}} = \mathbf{R}$, when applied to the Lie algebra $\mathbb{R}^3_\times$. It should be noted that this result is specific to the case of SO(3) when the Lie algebra is represented by $\mathbb{R}^3_\times$ and Ad_g is not generally equal to g.

The adjoint representation of an algebra $\mathfrak{g}$ (with elements $x_1, x_2 \in \mathfrak{g}$) onto itself is $\mathrm{ad}_x : \mathfrak{g} \times \mathfrak{g} \to \mathfrak{g}$, defined as

$$\mathrm{ad}_{x_1}\, x_2 := [x_1, x_2]_{\mathfrak{g}} = x_1x_2 - x_2x_1. \tag{2.16}$$

It can be found by time-differentiating the Ad_g function. The adjoint representation of Lie algebra $\mathfrak{so}(3)$ onto itself is $\mathrm{ad}_{\boldsymbol{\Omega}} : \mathfrak{so}(3) \times \mathfrak{so}(3) \to \mathfrak{so}(3)$, defined as

$$\mathrm{ad}_{\boldsymbol{\Omega}_1}\, \boldsymbol{\Omega}_2 := [\boldsymbol{\Omega}_1, \boldsymbol{\Omega}_2]_{\mathfrak{so}(3)} = \boldsymbol{\Omega}_1\boldsymbol{\Omega}_2 - \boldsymbol{\Omega}_2\boldsymbol{\Omega}_1, \tag{2.17}$$

and the adjoint representation of $\mathbb{R}^3_\times$ onto itself is $\mathrm{ad}_{\boldsymbol{\omega}} : \mathbb{R}^3_\times \times \mathbb{R}^3_\times \to \mathbb{R}^3_\times$, defined as

$$\mathrm{ad}_{\boldsymbol{\omega}_1}\, \boldsymbol{\omega}_2 := [\boldsymbol{\omega}_1, \boldsymbol{\omega}_2]_{\mathbb{R}^3_\times} = \boldsymbol{\omega}_1 \times \boldsymbol{\omega}_2, \tag{2.18}$$

therefore we also write

$$\mathrm{ad}_{\boldsymbol{\omega}} = \boldsymbol{\omega}\times = \boldsymbol{\omega}^\wedge. \tag{2.19}$$

It should be noted that this result is specific to the Lie algebra $\mathbb{R}^3_\times$ and ad_x is not generally equal to $x\times$. For more details on adjoint representations in the context of geometric robotics, we advice [35, Sections 4.2-4.3].

2.3.4 Exponential and logarithmic maps

Each Lie group has an exponential and a logarithmic map. The exponential map converts elements of the Lie algebra into elements of the Lie group. The logarithmic map is the inverse operation of the exponential map in a neighbourhood of the identity. For the case of $\mathfrak{so}(3)$ to SO(3), the exponential map is the matrix exponential (not to confuse with exponential of individual entries of the matrix) $\exp_{\mathbf{\Omega}} : \mathfrak{so}(3) \to \mathrm{SO}(3)$ defined as

$$\exp_{\mathbf{\Omega}}(\mathbf{\Omega}) := \exp(\mathbf{\Omega}) = \mathbf{R}, \tag{2.20}$$

where $\exp_{\mathbf{\Omega}}$ is the exponential map of the Lie group, and exp is the matrix exponential. The logarithmic map $\log_{\mathbf{\Omega}} : \mathrm{SO}(3) \to \mathfrak{so}(3)$ is the inverse of the exponential map, defined as

$$\log_{\mathbf{\Omega}}(\mathbf{R}) := \log(\mathbf{R}) = \mathbf{\Omega}, \tag{2.21}$$

where $\log_{\mathbf{\Omega}}$ is the logarithmic map of the Lie group, and log is the matrix logarithm. Since the Lie algebra $\mathbb{R}^3_\times$ is a more compact way of describing the Lie algebra $\mathfrak{so}(3)$, we also define the exponential map for $\mathbb{R}^3_\times$ to SO(3) is $\exp_{\boldsymbol{\omega}} : \mathbb{R}^3_\times \to \mathrm{SO}(3)$ defined as

$$\exp_{\boldsymbol{\omega}}(\boldsymbol{\omega}) := \exp(\boldsymbol{\omega}^\wedge) = \mathbf{R}, \tag{2.22}$$

and the logarithmic map $\log_{\boldsymbol{\omega}} : \mathrm{SO}(3) \to \mathbb{R}^3_\times$ defined as

$$\log_{\boldsymbol{\omega}}(\mathbf{R}) := \log(\mathbf{R})^\vee = \boldsymbol{\omega}. \tag{2.23}$$

For more details on exponential and logarithmic maps in the context of geometric robotics, we advice [35, Section 4.4].

2.4 The 3D Special Euclidean group SE(3)

The dynamics of a single rigid body can be defined on the 3D Special Euclidean group, SE(3). In Section 2.1, we marked the advantages of using SO(3) to represent rotational dynamics. As SE(3) is a combination of rotational dynamics on SO(3) and translations dynamics on $\mathbb{R}^3$, the 3D Special Euclidean group is our preferred representation of the moving base. This section presents the details of the Lie group SE(3).

2.4.1 The Lie group SE(3)

The translation and rotation of a frame are abstract concepts that can be quantified only when expressed relative to another frame. In this example, we take the translation and rotation of the moving-base frame 0, relative to the inertial frame A. The rotation of 0 with respect to A is ${}^A\mathbf{R}_0 \in \mathrm{SO}(3)$. The translation of the origin of 0 with respect to A is ${}^A\mathbf{o}_0 \in \mathbb{R}^3$. We combine both in the 4x4 transformation matrix

$${}^A\mathbf{H}_0 := \begin{bmatrix} {}^A\mathbf{R}_0 & {}^A\mathbf{o}_0 \\ 0_{1\times3} & 1 \end{bmatrix} \in \mathrm{SE}(3). \tag{2.24}$$

The 3D Special Euclidean group is written as $G = (\mathrm{SE}(3), \cdot)$, which is the set SE(3) combined with the operation of matrix multiplication. The four group axioms hold:

- Closure: for each $\mathbf{H}_1, \mathbf{H}_2 \in \mathrm{SE}(3)$, $\mathbf{H}_1 \cdot \mathbf{H}_2 = \mathbf{H}_3 \in \mathrm{SE}(3)$.
- Associativity: for each $\mathbf{H}_1, \mathbf{H}_2, \mathbf{H}_3 \in \mathrm{SE}(3)$, $(\mathbf{H}_1 \cdot \mathbf{H}_2) \cdot \mathbf{H}_3 = \mathbf{H}_1 \cdot (\mathbf{H}_2 \cdot \mathbf{H}_3)$.

- Identity element: there exists an identity element $I_4 \in \mathrm{SE}(3)$, such that $I_4 \cdot \mathbf{H} = \mathbf{H} \cdot I_4 = \mathbf{H}$ for any element $\mathbf{H} \in \mathrm{SE}(3)$. There exists only one identity element in the SE(3), which is I_4.
- Inverse element: for each $\mathbf{H}_1 \in \mathrm{SE}(3)$ there exists an inverse element $\mathbf{H}_2 (= \mathbf{H}_1^{-1}) \in \mathrm{SE}(3)$, such that $\mathbf{H}_1 \cdot \mathbf{H}_2 = \mathbf{H}_2 \cdot \mathbf{H}_1 = I_4$.

2.4.2 Tangent bundle $T\mathrm{SE}(3)$ and Lie algebra's $\mathfrak{se}(3)$ and $\mathbb{R}^6_\times$

The tangent space of Lie group SE(3) at point ${}^A\mathbf{H}_0$ is $T_{{}^A\mathbf{H}_0}\mathrm{SE}(3)$, which contains the element ${}^A\dot{\mathbf{H}}_0$. All the tangent spaces of SE(3) together form the tangent bundle $T\mathrm{SE}(3)$, containing all elements $\dot{\mathbf{H}} \in T\mathrm{SE}(3)$, for all $\mathbf{H} \in \mathrm{SE}(3)$. As shown in Subsection 2.3.2, the tangent space $T_{\mathbf{H}}\mathrm{SE}(3)$ can be mapped to the Lie algebra $\mathfrak{se}(3)$ by applying left-trivialization

$$T_{{}^A\mathbf{H}_0} L_{{}^A\mathbf{H}_0^{-1}} {}^A\dot{\mathbf{H}}_0 = {}^A\mathbf{H}_0^{-1} {}^A\dot{\mathbf{H}}_0 = {}^0\mathbf{v}_{A,0}^\wedge \in T_{I_4}\mathrm{SE}(3) = \mathfrak{se}(3), \tag{2.25}$$

where ${}^0\mathbf{v}_{A,0}^\wedge \in \mathfrak{se}(3)$ is left-trivialized velocity, also called the left-trivialized twist. The subscript indicates that it is the velocity of frame 0 with respect to frame A, and the superscript indicates that it is expressed in frame 0. The left-trivialized twist is given by

$${}^0\mathbf{v}_{A,0}^\wedge = \begin{bmatrix} {}^0\boldsymbol{\omega}_{A,0}^\wedge & {}^0\boldsymbol{v}_{A,0} \\ 0_{1\times 3} & 0 \end{bmatrix} \in \mathfrak{se}(3), \tag{2.26}$$

where ${}^0\boldsymbol{\omega}_{A,0}^\wedge \in \mathfrak{so}(3)$ is the left-trivialized angular velocity and ${}^0\boldsymbol{v}_{A,0} \in \mathbb{R}^3$ the left-trivialized translational velocity. The Lie algebra can also be found by applying right-trivialization

$$T_{{}^A\mathbf{H}_0} R_{{}^A\mathbf{H}_0^{-1}} {}^A\dot{\mathbf{H}}_0 = {}^A\dot{\mathbf{H}}_0 {}^A\mathbf{H}_0^{-1} = {}^A\mathbf{v}_{A,0}^\wedge \in T_I\mathrm{SE}(3) = \mathfrak{se}(3), \tag{2.27}$$

where ${}^A\mathbf{v}_{A,0}^\wedge$ is the right-trivialized velocity, or right-trivialized twist, given by

$${}^A\mathbf{v}_{A,0}^\wedge = \begin{bmatrix} {}^A\boldsymbol{\omega}_{A,0}^\wedge & {}^A\boldsymbol{v}_{A,0} \\ 0_{1\times 3} & 0 \end{bmatrix} \in \mathfrak{se}(3), \tag{2.28}$$

where ${}^A\boldsymbol{\omega}_{A,0}^\wedge \in \mathfrak{so}(3)$ is the right-trivialized angular velocity and ${}^A\boldsymbol{v}_{A,0} \in \mathbb{R}^3$ the right-trivialized translational velocity. The superscript indicates that the velocity is expressed in frame A. The left-trivialized twist is related to the right-trivialized twist as

$${}^0\mathbf{v}_{A,0}^\wedge = {}^A\mathbf{H}_0^{-1} {}^A\mathbf{v}_{A,0}^\wedge {}^A\mathbf{H}_0. \tag{2.29}$$

The Lie algebra $\mathfrak{se}(3)$ consists of the set $\mathfrak{se}(3)$, with the Lie bracket

$$[\mathbf{v}_1^\wedge, \mathbf{v}_2^\wedge]_{\mathfrak{se}(3)} = \mathbf{v}_1^\wedge \mathbf{v}_2^\wedge - \mathbf{v}_2^\wedge \mathbf{v}_1^\wedge. \tag{2.30}$$

The Lie algebra $\mathfrak{se}(3)$ satisfies all three Lie algebra axioms of Subsection 2.3.2, which is a simple exercise that we leave to the reader. The Lie algebra can be identified through a diffeomorphism with the set $\mathbb{R}^6$ through the vee operator

$$({}^0\mathbf{v}_{A,0}^\wedge)^\vee = {}^0\mathbf{v}_{A,0} \in \mathbb{R}^6, \tag{2.31}$$

where ${}^0\mathbf{v}_{A,0}$ is the vector representation of the left-trivialized twist, and is given by

$${}^0\mathbf{v}_{A,0} = \begin{bmatrix} {}^0\boldsymbol{v}_{A,0} \\ {}^0\boldsymbol{\omega}_{A,0} \end{bmatrix}, \tag{2.32}$$

where ${}^0\boldsymbol{\omega}_{A,0} \in \mathbb{R}^3$ is the vector representation of the left-trivialized angular velocity. This shows that, *as a set*, $\mathbb{R}^6$ is diffeomorphic to $\mathfrak{se}(3)$. As a Lie algebra, $\mathfrak{se}(3)$ has the Lie brackets $[\mathbf{v}_1^\wedge, \mathbf{v}_2^\wedge]_{\mathfrak{se}(3)} = \mathbf{v}_1^\wedge \mathbf{v}_2^\wedge - \mathbf{v}_2^\wedge \mathbf{v}_1^\wedge$. This Lie bracket can be identified by the 6D cross-product as Lie bracket $[\mathbf{v}_1, \mathbf{v}_2]_{\mathbb{R}^6_\times} = \mathbf{v}_1 \times \mathbf{v}_2$ for the set $\mathbb{R}^6$, where the 6D cross product is defined as

$$ {}^0\mathbf{v}_{A,0}\times := \begin{bmatrix} {}^0\boldsymbol{\omega}_{A,0}^\wedge & {}^0\boldsymbol{v}_{A,0}^\wedge \\ 0_{3\times3} & {}^0\boldsymbol{\omega}_{A,0}^\wedge \end{bmatrix}. \tag{2.33} $$

We can therefore identify $\mathfrak{se}(3)$, *as a Lie algebra*, with $\mathbb{R}^6_\times$, which is the set $\mathbb{R}^6$ with 6D cross-product as Lie bracket.

2.4.3 Adjoint representations of SE(3)

There exist two adjoint representations: the adjoint representation of a group onto its algebra, Ad, and the adjoint representation of an algebra onto itself, ad. The adjoint representation of the group SE(3) onto its algebra $\mathfrak{se}(3)$ is $\mathrm{Ad}_{\mathbf{H}} : \mathrm{SE}(3) \times \mathfrak{se}(3) \to \mathfrak{se}(3)$, defined as

$$ \mathrm{Ad}_{{}^A\mathbf{H}_0}\, {}^0\mathbf{v}_{A,0}^\wedge := {}^A\mathbf{H}_0 {}^0\mathbf{v}_{A,0}^\wedge {}^A\mathbf{H}_0^{-1}. \tag{2.34} $$

and the adjoint representation of SE(3) onto $\mathbb{R}^6_\times$ is $\mathrm{Ad}_{\mathbf{H}} : \mathrm{SE}(3) \times \mathbb{R}^6_\times \to \mathbb{R}^6_\times$, defined as

$$ \mathrm{Ad}_{{}^A\mathbf{H}_0}\, {}^0\mathbf{v}_{A,0} := {}^A\mathbf{X}_0 {}^0\mathbf{v}_{A,0}, \tag{2.35} $$

where ${}^A\mathbf{X}_0 \in \mathbb{R}^{6\times6}$ is the velocity transformation matrix, defined as

$$ {}^A\mathbf{X}_0 := \begin{bmatrix} {}^A\mathbf{R}_0 & {}^A\mathbf{o}_0^\wedge {}^A\mathbf{R}_0 \\ 0_{3\times3} & {}^A\mathbf{R}_0 \end{bmatrix}. \tag{2.36} $$

We also write $\mathrm{Ad}_{{}^A\mathbf{H}_0} = {}^A\mathbf{X}_0$. It should be noted that this result is specific to the case of SE(3) when the Lie algebra is represented by $\mathbb{R}^6\times$. If the Lie algebra is represented by $\mathfrak{se}(3)$, $\mathrm{Ad}_{\mathbf{H}} \neq \mathbf{X}$.

The adjoint representation of Lie algebra $\mathfrak{se}(3)$ onto itself is $\mathrm{ad}_{\mathbf{v}^\wedge} : \mathfrak{se}(3) \times \mathfrak{se}(3) \to \mathfrak{se}(3)$, defined as

$$ \mathrm{ad}_{\mathbf{v}_1^\wedge}\, \mathbf{v}_2^\wedge := [\mathbf{v}_1^\wedge, \mathbf{v}_2^\wedge]_{\mathfrak{se}(3)} = \mathbf{v}_1^\wedge \mathbf{v}_2^\wedge - \mathbf{v}_2^\wedge \mathbf{v}_1^\wedge, \tag{2.37} $$

and the adjoint representation of $\mathbb{R}^6_\times$ onto itself is $\mathrm{ad}_{\mathbf{v}} : \mathbb{R}^6_\times \times \mathbb{R}^6_\times \to \mathbb{R}^6_\times$, defined as

$$ \mathrm{ad}_{\mathbf{v}_1}\, \mathbf{v}_2 := [\mathbf{v}_1, \mathbf{v}_2]_{\mathbb{R}^6_\times} = \mathbf{v}_1 \times \mathbf{v}_2, \tag{2.38} $$

therefore we also write $\mathrm{ad}_{\mathbf{v}} = \mathbf{v}\times$, where $\times$ denotes the 6D cross product defined in (2.33).

2.4.4 Exponential and logarithmic maps of SE(3)

The exponential and logarithmic maps, as presented in Subsection 2.3.4, of the Lie group SE(3) are defined as the matrix exponential and matrix logarithm respectively. Therefore the exponential map for the Lie group SE(3) is $\exp : \mathfrak{se}(3) \to \mathrm{SE}(3)$, defined as

$$ \exp(\mathbf{v}^\wedge) := \mathbf{H}, \tag{2.39} $$

and the logarithmic map for the Lie group SE(3) is $\log : \mathrm{SE}(3) \to \mathfrak{se}(3)$ defined as

$$ \log(\mathbf{H}) := \mathbf{v}^\wedge. \tag{2.40} $$

2.4.5 Wrenches and the dual space

A wrench ${}_B\mathbf{f}$ expressed in frame B is defined as a combination of linear force and angular torque into a 6D-vector in linear-angular order

$$ {}_B\mathbf{f} := \begin{bmatrix} {}_B\boldsymbol{f} \\ {}_B\boldsymbol{\tau} \end{bmatrix} \in \mathbb{R}^6, \tag{2.41} $$

where ${}_B\boldsymbol{f} \in \mathbb{R}^3$ is a translational force expressed in frame B, and ${}_B\boldsymbol{\tau} \in \mathbb{R}^3$ the torque expressed in frame B. A wrench is only expressed in a certain frame, as it is not a quantity that is relative to another frame. From a perspective of Lie group theory, wrenches are part of the dual space of $\mathfrak{se}(3)$, which is $\mathfrak{se}(3)^*$. In this thesis, the dual space is only important for the 6D dual cross-product, which is defined as

$$ {}^B\mathbf{v}_{A,B}\bar{\times}^* := \begin{bmatrix} {}^B\boldsymbol{\omega}^\wedge_{A,B} & 0_{3\times 3} \\ {}^B\boldsymbol{v}^\wedge_{A,B} & {}^B\boldsymbol{\omega}^\wedge_{A,B} \end{bmatrix}, \tag{2.42} $$

which is simply the negative of the transpose of (2.33). For more details about wrenches and the dual space, we refer to [30, Chapter 6] and [36, Section 1.5].
Note: what we call a twist and wrench is what in [8] is called a 'spatial velocity' and 'spatial force' respectively. Contrary to what we use, both are there defined in angular-linear order.

2.5 Sensitivity analysis of dynamical systems on Lie groups

As shown in the Section 2.3, through left-trivialization a Lie algebra associated to a Lie group G with n degrees of freedom can be identified with a vector with dimension $\mathbb{R}^n$. This is a fundamental property for sensitivity analysis of systems evolving on Lie groups, as this allows to map a state perturbation in G into a perturbation vector in $\mathbb{R}^n$. This is exploited in left-trivialized linearization of Lie groups, as presented in [11, 17].

Given a Lie group G with element g, a dynamical system evolving on G with m inputs is defined as

$$ \dot{g}(t) = f(g, u, t), \tag{2.43} $$

where $g \in G$ is the system state, $u \in \mathbb{R}^m$ the system input and $t \in \mathbb{R}$ is time. The map $f : G \times \mathbb{R}^m \times \mathbb{R} \to TG$, $(g, u, t) \mapsto f(g, u, t)$ represents the dynamical system, which maps the state g, input u, and time t to the tangent space TG of the Lie group G. The tangent space is a vector space which we want to map into a vector. It is possible to map the dynamical system $\dot{f}(g, u, t)$ to the Lie algebra $\mathfrak{g}$, so that it can be represented by a vector with the same dimension as the number of degrees of freedom in the Lie group. To do so, left-trivialization is applied. The dynamical system f can be left-trivialized by applying left-translation with the inverse of the state, g^{-1}. The resulting left-trivialized vector field λ maps the element g, input u and time t to the Lie algebra. This left-trivialized vector field $\lambda : G \times \mathbb{R}^m \times R \to \mathfrak{g}$ is defined as

$$ \lambda(g, u, t) := g^{-1}(t) f(g, u, t), \tag{2.44} $$

so that (2.43) can be rewritten to

$$ \dot{g}(t) = g(t)\lambda(g, u, t). \tag{2.45} $$

We want to linearize the system both with respect to the state g, and the input u. As the state and input are generally dependent on time, we linearize about a nominal trajectory, defined as $\eta(t) := (g(t), u(t)) \in G \times \mathbb{R}^m, t \geqslant 0$. The left-trivialized linearization of the dynamical system (2.43) about the nominal trajectory $\eta(t)$ is given by

$$ \dot{z}(t) = A(\eta, t)z(t) + B(\eta, t)w(t), \tag{2.46} $$

where $z(t) \in \mathfrak{g}$ is the perturbation vector, $w(t) \in \mathbb{R}^m$ the perturbed input vector, and where the state and input matrices are given by

$$A(\eta, t) := \mathrm{D}_1 \lambda(g, u, t) \circ \mathrm{D}\, L_g(e) - \mathrm{ad}_{\lambda(g,u,t)} \tag{2.47}$$

and

$$B(\eta, t) := \mathrm{D}_2 \lambda(g, u, t). \tag{2.48}$$

Here $z(t) \in \mathfrak{g}$ belongs to the *vector representation* of the Lie algebra, and is a representation of the system state. As we linearize about a nominal trajectory (and the state and input about which we linearize are generally not constant), the state matrix $A(\eta, t)$ and input matrix $B(\eta, t)$ are also generally not constant. In practical applications, this means that re-evaluation of the linearization is necessary. The input matrix $B(\eta, t)$ is computed through a standard derivative of the left-trivialized vector field $\lambda(g, u, t)$ with respect to the input u. The state matrix $A(\eta, t)$ is computed in two parts. The first part $\mathrm{D}_1 \lambda(g, u, t) \circ \mathrm{D}\, L_g(e)$ can be computed in a direction $z \in \mathfrak{g}$ as the limit

$$\mathrm{D}_1 \lambda(g, u, t) \circ \mathrm{D}\, L_g(e) \cdot z = \lim_{\varepsilon \to 0} \frac{1}{\varepsilon}\big[\lambda(g \exp(\varepsilon z)) - \lambda(g)\big], \tag{2.49}$$

where $\varepsilon \in \mathbb{R}$ is the perturbation scalar. As the exponential maps $\varepsilon z \in \mathfrak{g}$ from the Lie algebra to the Lie group G, $g \exp(\varepsilon z)$ represents the group operation on two group elements, $g \in G$ and $\exp(\varepsilon z) \in G$. The second part $\mathrm{ad}_{\lambda(g,u,t)}$ is a standard computation of the ad matrix representation as shown in (2.16). Further details on left-trivialized linearization can be found in [11, 17].

2.6 Multibody system definitions

This section presents definitions used in multibody systems. Definitions of bodies and joints are given, as well as how to number them. After that, frames are defined and a distinction in types of wrenches is made.

2.6.1 Bodies and joints

A multibody dynamical system is comprised of multiple bodies connected by joints. Bodies form the rigid parts of a system, such as the arms or torso for humanoid robots, while joints allow (controlled) movement in a system, such as the elbows for a humanoid robot.
Assumption 2.1: We assume that all bodies are infinitely rigid, meaning that there is no flexibility or movement possible inside a body. Therefore the only movement in a system is caused by the moving base or joints.
Assumption 2.2: We assume that all joints, except for the moving base, are *conventional* joints of the following types:

- Revolute (hinge joint): a pure rotational joint around one axis (1 DoF)
- Prismatic (sliding joint): a pure translational joint in one direction (1 DoF)
- Helical (screw joint): a combination of a rotation and a translation like a bolt and nut (1 DoF)
- Cylindrical: both a rotation and a translation around/in the same axis (2 DoF)
- Planar: a rotation and two translation on a plane (3 DoF)

It is possible to model joints with multiple degrees of freedom as multiple 1-DoF joints, which allows simplification of the recursive algorithms presented in this thesis, as they then only have to deal with 1-DoF joints. The cylindrical joint can be modelled as a combination of a revolute and a prismatic joint around/in the same axis. The planar joint can be modelled as a combination of two prismatic joints and a rotational joint, all in the same plane. Therefore, modelling just the revolute, prismatic and helical joints suffices. Joint-specific variables related to these three joint types are defined in Appendix B.2.

There are two types of joints commonly used which may result in singularity problems: the spherical joint (also known as ball and socket) and the 6-DoF joint (also known as freeflyer or free motion joint). Since both have three rotational degrees of freedom, modelling them as a vector space may result in singularity issues. To be singularity-free, they need to be modelled on Lie groups. As all joints are modelled as vector spaces in general recursive algorithms, we use algorithms dedicated to moving-base systems, where the moving-base is modelled on SE(3), separate from the joints, as opposed to [7]. Using our method, it is possible to model spherical as 1-DoF joints, however the user should be wary of singularity issues.

2.6.2 Body and joint numbering

Body numbering starts at the base (either fixed or moving), which is defined as body 0. Going outwards, the body numbers increase up to n_B, the total number of bodies, meaning that each body's number must be higher than the one of its parent. The *parent* of arbitrary body i is its inward connection and is defined as $\lambda(i)$. A body may have zero, one, or multiple outward connections which we call the body's *children*. These are defined in an array-structure $\mu(i)$. An example of body numbering is depicted in Figure 2.4, where $\mu(i) = \{j, k\}$. Note that there are multiple possibilities to number tree-based multibody systems, e.g. here 3 and 4 could be swapped.

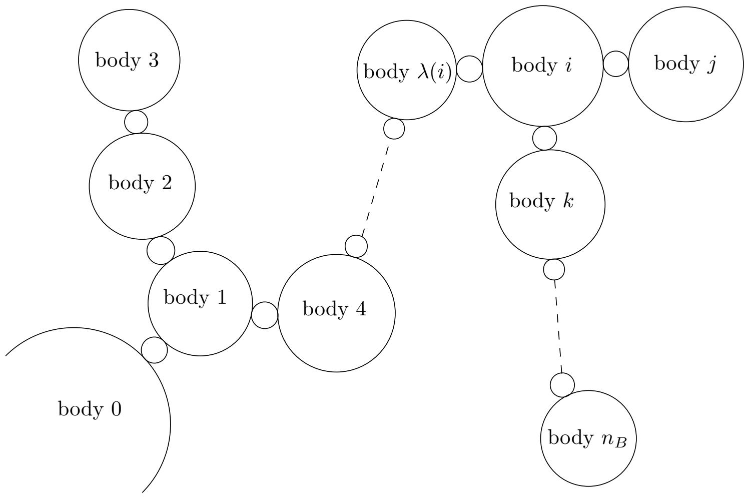


Figure 2.4: An example of body numbering.

A joint is said to connect from the *predecessor* (the body closer to the base) to the *successor* (the body further from the base). In particular, joint i has predecessor body $\lambda(i)$ and successor body i, as shown in Figure 2.5.

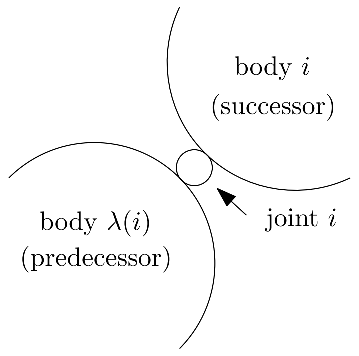


Figure 2.5: Joint numbering.

2.6.3 Frames

We define a frame as a combination of an *origin* and an *orientation frame*. The origin of frame A is defined as $\mathbf{o}_A$ and the orientation frame of frame A is defined as $[A]$. Frame A can be written as $A = (\mathbf{o}_A, [A])$ [30, 37, 38]. Several frames are used through this report, which are defined here for clarity. Frame i represents the frame attached of the i-th body of a multibody system and frame $\lambda(i)|i$ is defined as the frame attached to the i-th joint. An example is shown in Figure 2.6, where three bodies with two joints are shown. The frames are indicated in blue.

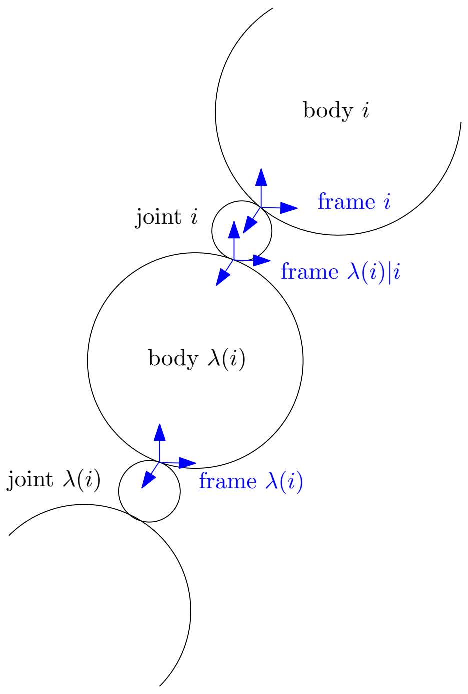


Figure 2.6: A general example of body and joint frames.

Frames i and $i|\mu(i)$ are attached to body i, and frames $\lambda(i)$ and $\lambda(i)|i$ are attached to body $\lambda(i)$. The transformation matrix from frame $\lambda(i)$ to frame $\lambda(i)|i$ is defined as ${}^{\lambda(i)|i}\mathbf{H}_{\lambda(i)}$ and is a fixed parameter of body i since both frames are fixed to body i. The transformation matrix from frame $\lambda(i)|i$ to frame i is defined as ${}^{i}\mathbf{H}_{\lambda(i)|i}$ and is a purely rotational transformation for revolute and spherical joints. If a joint i is purely rotational, it should be seen as a point having no volume, and frames $\lambda(i)|i$ and i have the same origin. For purely translational joints, ${}^{i}\mathbf{H}_{\lambda(i)|i}$ is a purely translational transformation and the frames $\lambda(i)|i$ and i have the same orientation.

Besides body-fixed frames, this report also contains frames which are not generally connected to bodies. When dealing with moving-base systems, frame A represents the inertial frame, which is fixed to the world. Frames C and D represent arbitrary frames. *Note: the letters B and E are not used to avoid confusion with other literature, where B is often the base frame and E the end-effector frame.*

2.6.4 Types of wrenches

In this thesis we distinct four types of wrenches:

- $\mathbf{f}_{\mathcal{B}i}$, the sum of all wrenches acting on body i.
- $\mathbf{f}_{\mathcal{I}i}$, the sum of all internal wrenches acting on body i.
- $\mathbf{f}_{\mathcal{E}i}$, the sum of all external wrenches acting on body i.
- $\mathbf{f}_{\mathcal{J}i}$, the wrench of body $\lambda(i)$ acting on body i, through joint i.

The sum of all wrenches acting on body i expressed in arbitrary frame C is defined as the sum of all internal and external wrenches acting on body i

$$_C\mathbf{f}_{\mathcal{B}i} = {_C\mathbf{f}_{\mathcal{I}i}} + {_C\mathbf{f}_{\mathcal{E}i}}. \tag{2.50}$$

The sum of all internal wrenches acting on body i, $\mathbf{f}_{\mathcal{I}i}$, expressed in arbitrary frame C, is defined in terms of joint wrenches as

$$_C\mathbf{f}_{\mathcal{I}i} = {_C\mathbf{f}_{\mathcal{J}i}} - \sum_{l \in \mu(i)} {_C\mathbf{f}_{\mathcal{J}l}}, \tag{2.51}$$

where $\mu(i)$ is the array containing all the children of body i, as defined in Subsection 2.6.2.

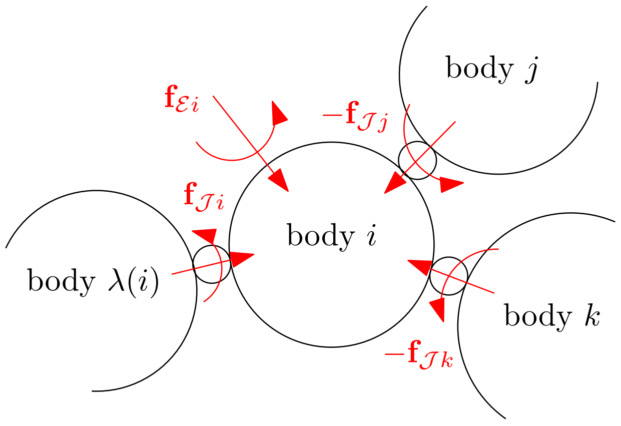

body j
$\mathbf{f}_{\mathcal{E}i}$
$-\mathbf{f}_{\mathcal{J}j}$
$\mathbf{f}_{\mathcal{J}i}$
body i
body $\lambda(i)$
body k
$-\mathbf{f}_{\mathcal{J}k}$

Figure 2.7: All wrenches acting on body i.

For the example in Figure 2.7 the children of body i are given by $\mu(i) = \{j, k\}$ and the sum of internal wrenches on body i would be

$$_C\mathbf{f}_{\mathcal{I}i} = {_C\mathbf{f}_{\mathcal{J}i}} - {_C\mathbf{f}_{\mathcal{J}j}} - {_C\mathbf{f}_{\mathcal{J}k}}. \tag{2.52}$$

2.7 Multibody dynamics notation

Many symbols used in the Eindhoven-Genoa (EG) notation have been defined in [30], although not all variables used in this thesis are included. This section presents an overview of all the variables used in this report in six tables, followed by a shortened notation. The tables show the numbers, equation of motion variables and the rigid body dynamics variables. Where necessary, explanations are found below the tables.

An important distinction must be made between two types of variables. Equation of motion variables are (as the name suggests) used to express the complete equations of motion of multibody systems. Rigid body dynamics variables are used to compute the dynamics of single bodies and joints in recursive algorithms. In Section 4.1, relations between both types of variables are shown, which we use to relate the equations of motion variables to quantities that can be computed by recursive algorithms.

2.7.1 Numbers in Eindhoven-Genoa notation

The following table shows which variables the EG notation uses to indicate the number of bodies, joints and degrees of freedom.

Table 2.2: Numbers in the Eindhoven-Genoa notation.

Row	**EG**	**Dimension**	**Explanation**
1	n_B	$\mathbb{N}$	Number of bodies (excluding the moving-base)
2	n_i	$\mathbb{N}$	Number of degrees of freedom of joint i
3	n_J	$\mathbb{N}$	Number of degrees of freedom in all joints
4	$n = 6 + n_J$	$\mathbb{N}$	Number of degrees of freedom (moving-base)

The number of degrees of freedom for a fixed-base system is given by n_J, as the only degrees of freedom in a fixed-base system are in the joints. Since in this thesis we consider only conventional joints (see Assumption 2.2) which we model as 1 DoF joints, n_i is equal to one for all joints and n_J is equal to n_B. The tables below, however, show the dimensions of variables in terms of n_i and n_J, which generally holds for all joint types.

2.7.2 Equations of motion variables in the Eindhoven-Genoa notation

The following three tables show which variables the EG notation uses to indicate the equations of motion of multibody systems. The first table shows general variables, which are valid for both fixed-base and moving-base systems. The second and third table show the fixed-base and moving-base system variables respectively, and the fourth table shows the extended moving-base system variables, which will be used in Section 3.3.

Table 2.3: General equations of motion variables in the Eindhoven-Genoa notation.

Row	**EG**	**Dimension**	**Explanation**
1	$\mathbf{s}$	$\mathbb{R}^{n_J}$	Generalized position vector or system shape
2	$\mathbf{r}$	$\mathbb{R}^{n_J}$	Generalized velocity vector
3	$\boldsymbol{\tau}$	$\mathbb{R}^{n_J}$	Joint torques or generalized forces vector

Explanations

- The generalized position vector is given by $\mathbf{s} = [\mathbf{s}_1; \mathbf{s}_2; ...; \mathbf{s}_{n_J}]$, where $\mathbf{s}_i$ relates the generalized joint position of joint i to the velocity transformation matrix ${}^i\mathbf{X}_{\lambda(i)|i}$. More details on this relation are found in Appendix B.2. The variables $\mathbf{r}$ and $\boldsymbol{\tau}$ have the same structure as $\mathbf{s}$.

Table 2.4: Fixed base equations of motion variables in the Eindhoven-Genoa notation.

Row	**EG**	**Dimension**	**Explanation**
1	$\mathbf{q}_{fb} = \mathbf{s}$	$\mathbb{R}^{n_J}$	System configuration for fixed-base systems
2	$\mathbf{M}_{fb}$	$\mathbb{R}^{n_J \times n_J}$	Generalized inertia matrix
3	$\mathbf{h}_{fb}$	$\mathbb{R}^{n_J}$	Generalized bias wrench vector for fixed-base systems
4	$\mathbf{C}_{fb}$	$\mathbb{R}^{n_J \times n_J}$	Coriolis matrix for fixed-base systems
5	$\mathbf{G}_{fb}$	$\mathbb{R}^{n_J}$	Gravitational wrench vector for fixed-base systems
6	${}^C\mathbf{J}_{A,i}$	$\mathbb{R}^{6 \times n_J}$	Jacobian of velocity of frame i w.r.t. frame A

Table 2.5: Moving base equations of motion variables in the Eindhoven-Genoa notation.

Row	**EG**	**Dimension**	**Explanation**
1	${}^A\mathbf{H}_0$	SE(3)	Transformation matrix from moving-base frame 0 to inertial frame A
2	$\mathbf{q} = ({}^A\mathbf{H}_0, \mathbf{s})$	$\mathrm{SE}(3) \times \mathbb{R}^{n_J}$	System configuration
3	${}^A\mathbf{R}_0$	SO(3)	Rotation matrix of moving-base frame 0 with respect to inertial frame A
4	${}^A\mathbf{o}_0$	$\mathbb{R}^3$	Origin of moving-base frame 0 with respect to inertial frame A
5	${}^0\mathbf{v}_{A,0}$	$\mathbb{R}^6$	(Left-trivialized) twist of moving-base frame 0 with respect to inertial frame A expressed in frame 0
6	$\boldsymbol{\nu} = ({}^0\mathbf{v}_{A,0}, \mathbf{r})$	$\mathbb{R}^6 \times \mathbb{R}^{n_J}$	Left-trivialized system velocity
7	$\mathbf{M}$	$\mathbb{R}^{n \times n}$	Mass matrix
8	$\mathbf{M}_{11}$	$\mathbb{R}^{6 \times 6}$	Inertia matrix of the whole system as a composite rigid body
9	$\mathbf{M}_{12} = {}_0\mathbf{F}$	$\mathbb{R}^{6 \times n_J}$	Wrenches required to support unit acceleration
10	$\mathbf{M}_{22}$	$\mathbb{R}^{n_J \times n_J}$	Generalized inertia matrix
11	$\mathbf{h}$	$\mathbb{R}^{n}$	Generalized bias wrench vector
12	$\mathbf{h}_1$	$\mathbb{R}^6$	Bias wrench for the whole system as a composite rigid body
13	$\mathbf{h}_2$	$\mathbb{R}^{n_J}$	Generalized bias wrench vector of the joints
14	$\mathbf{C}$	$\mathbb{R}^{n \times n}$	Coriolis matrix
15	$\mathbf{G}$	$\mathbb{R}^{n}$	Gravitational wrench vector
16	$\mathbf{S}$	$\mathbb{R}^{n \times n_J}$	Selection matrix
17	${}^C\mathbf{J}_{A,i/0}$	$\mathbb{R}^{6 \times n}$	Geometric Jacobian of velocity of frame i w.r.t. frame A with moving-base frame 0

These variables are explained in more detail in Section 2.8. For the definitions of these variables, we refer to [39, Chapters 3] and the references therein, which handle moving-base multibody equations of motion and its origin extensively.

Table 2.6: Extended moving base equations of motion variables in the Eindhoven-Genoa notation.

Row	EG	Dimension	Explanation
1	$\bar{\boldsymbol{\tau}}_b$	$\mathbb{R}^6$	Non-physical control input acting on the moving base
2	$\bar{\boldsymbol{\tau}} = [\bar{\boldsymbol{\tau}}_b; \boldsymbol{\tau}]$	$\mathbb{R}^6 \times \mathbb{R}^{n_J}$	Extended input vector

The definitions of the extended moving-base equations of motion can be found in Section 3.3.

2.7.3 Rigid body dynamics in the Eindhoven-Genoa notation

The following table shows which variables the EG notation uses to indicate rigid body dynamics, that are used in recursive algorithms. All variables are (where logical) expressed in an arbitrary frame C.

Table 2.7: Rigid body dynamics in the Eindhoven-Genoa notation.

Row	EG	Dimension	Explanation
1	${}_C\mathbb{M}_C^{\mathcal{B}i}$	$\mathbb{R}^{6\times 6}$	Inertia matrix of body i
2	${}_C\mathbb{M}_C^{\mathcal{B}i,A}$	$\mathbb{R}^{6\times 6}$	Articulated-body inertia matrix of body i
3	${}_C\mathbb{M}_C^{\mathcal{B}i,a}$	$\mathbb{R}^{6\times 6}$	Apparent articulated-body inertia matrix of body i
4	${}_C\mathbb{M}_C^{\mathcal{B}i,c}$	$\mathbb{R}^{6\times 6}$	Composite rigid body inertia matrix of body i
5	${}^C\mathbf{v}_{A,i}$	$\mathbb{R}^6$	Twist or spatial velocity of frame i w.r.t frame A
6	${}^C\dot{\mathbf{v}}_{A,i}$	$\mathbb{R}^6$	Apparent acceleration of frame i w.r.t frame A
7	${}^C\mathbf{a}_{A,i}$	$\mathbb{R}^6$	Intrinsic acceleration of frame i w.r.t frame A
8	${}^C\mathbf{a}_{grav}$	$\mathbb{R}^6$	Intrinsic gravitational acceleration
9	${}^C\mathbf{a}_i^r$	$\mathbb{R}^6$	Intrinsic acceleration relative to the moving-base acceleration, plus the gravitational acceleration of body i
10	${}^C\mathbf{a}_i^{vp}$	$\mathbb{R}^6$	Intrinsic acceleration that only accounts for the velocity product terms of body i
11	${}_C\mathbf{m}_{\mathcal{B}i}$	$\mathbb{R}^6$	Spatial momentum of body i
12	${}^C\boldsymbol{\Gamma}_{\lambda(i),i}$	$\mathbb{R}^{6\times n_i}$	Joint velocity subspace matrix of joint i
13	${}^C\dot{\boldsymbol{\Gamma}}_{\lambda(i),i}$	$\mathbb{R}^{6\times n_i}$	Joint apparent acceleration subspace matrix of joint i
14	${}^C\mathbf{A}_{\lambda(i),i}$	$\mathbb{R}^{6\times n_i}$	Joint intrinsic acceleration subspace matrix of joint i
15	${}^i\mathbf{v}_{\lambda(i),i}$	$\mathbb{R}^6$	Velocity of joint i
16	${}_C\mathbf{f}_{\mathcal{B}i}$	$\mathbb{R}^6$	Total wrench acting on body i
17	${}_C\mathbf{f}_{\mathcal{J}i}$	$\mathbb{R}^6$	Wrench transmitted from body $\lambda(i)$ to body i across joint i
18	${}_C\mathbf{f}_{\mathcal{I}i}$	$\mathbb{R}^6$	Sum of all internal wrenches acting on body i
19	${}_C\mathbf{f}_{\mathcal{E}i}$	$\mathbb{R}^6$	Sum of all external wrenches acting on body i
continued on the next page.			

Row	**EG**	**Dimension**	**Explanation**
20	${}_C\mathbf{b}_{\mathcal{B}i}$	$\mathbb{R}^6$	Bias wrench acting on body i
21	${}_C\mathbf{b}^A_{\mathcal{B}i}$	$\mathbb{R}^6$	Articulated-body bias wrench acting on body i
22	${}_C\mathbf{b}^a_{\mathcal{B}i}$	$\mathbb{R}^6$	Apparent articulated-body bias wrench acting on body i
23	${}_C\mathbf{b}^c_{\mathcal{B}i}$	$\mathbb{R}^6$	Composite rigid body bias wrench acting on body i
24	${}_C\mathbf{b}^{vp}_{\mathcal{B}0}$	$\mathbb{R}^6$	Bias wrench of moving-base with zero joint acceleration
25	${}^C\mathbf{X}_D$	$\mathbb{R}^{6\times 6}$	Velocity transformation from frame D to frame C
26	${}_C\mathbf{X}^D$	$\mathbb{R}^{6\times 6}$	Wrench transformation from frame D to frame C
27	${}_C\mathbf{U}_{\mathcal{B}i}$	$\mathbb{R}^{6\times n_i}$	Subexpression used in ABA
28	$\mathbf{D}_{\mathcal{B}i}$	$\mathbb{R}^{n_i\times n_i}$	Subexpression used in ABA
29	$\mathbf{u}_{\mathcal{B}i}$	$\mathbb{R}^{n_i}$	Subexpression used in ABA
30	${}_C\mathbf{F}_{\mathcal{B}i}$	$\mathbb{R}^6$	Required wrench to support unit acceleration of joint i
31	$\mathcal{F}_i$	$\mathbb{R}^{6\times n_J}$	Wrench set collecting the contributions of the supporting tree rooted at i
32	$\mathcal{P}_i$	$\mathbb{R}^{6\times n_J}$	Motion set which contains the contributions of all parents of joint i

Explanations

- Row 12: the joint velocity subspace matrix is given by ${}^C\mathbf{v}_{\lambda(i),i} = {}^C\mathbf{\Gamma}_{\lambda(i),i}\,\dot{\mathbf{s}}_i$.
- Row 14: the joint intrinsic acceleration subspace matrix is given by ${}^C\mathbf{a}_{\lambda(i),i} = {}^C\mathbf{A}_{\lambda(i),i}\,\dot{\mathbf{s}}_i$, where ${}^C\mathbf{A}_{\lambda(i),i} = {}^C\dot{\mathbf{\Gamma}}_{\lambda(i),i} + {}^C\mathbf{v}_{A,i} \times {}^C\mathbf{\Gamma}_{\lambda(i),i}$.
- Row 27: the subexpression is given by ${}_C\mathbf{U}_{\mathcal{B}i} = {}_C\mathbb{M}_C^{\mathcal{B}i,A}\,{}^C\mathbf{\Gamma}_{\lambda(i),i}$.
- Row 28: the subexpression is given by $\mathbf{D}_{\mathcal{B}i} = {}^C\mathbf{\Gamma}^{\mathrm{T}}_{\lambda(i),i}\,{}_C\mathbf{U}_{\mathcal{B}i}$.
- Row 29: the subexpression is given by $\mathbf{u}_{\mathcal{B}i} = \boldsymbol{\tau}_i - {}^C\mathbf{\Gamma}^{\mathrm{T}}_{\lambda(i),i}\,{}_C\mathbf{b}^A_{\mathcal{B}i}$.

2.7.4 Shortened rigid body dynamics notation

Preciseness is one of the advantages of the Eindhoven-Genoa notation, as all aspects to a variable (e.g. in which frame it is expressed, with respect to which frame it is) are present in the notation. The preciseness does come with a downside: the notation can become unnecessarily bulky and overloaded with frames. Therefore we shorten commonly used variables as follows:

$$\begin{aligned}
\mathbf{v}_i &= {}^i\mathbf{v}_{A,i} & \mathbf{f}_{\mathcal{B}i} &= {}_i\mathbf{f}_{\mathcal{B}i}\\
\mathbf{a}_i &= {}^i\mathbf{a}_{A,i} & \mathbf{f}_{\mathcal{E}i} &= {}_i\mathbf{f}_{\mathcal{E}i}\\
\mathbf{m}_{\mathcal{B}i} &= {}_i\mathbf{m}_{\mathcal{B}i} & \mathbf{f}_{\mathcal{I}i} &= {}_i\mathbf{f}_{\mathcal{I}i}\\
\mathbf{v}_{\mathcal{J}i} &= {}^i\mathbf{v}_{\lambda(i),i} & \mathbf{f}_{\mathcal{J}i} &= {}_i\mathbf{f}_{\mathcal{J}i}\\
\mathbf{c}_{\mathcal{J}i} &= {}^i\mathbf{c}_{\lambda(i),i} & \mathbb{M}_{\mathcal{B}i} &= {}_i\mathbb{M}_i^{\mathcal{B}i}\\
\mathbf{d}_{\mathcal{J}i} &= {}^i\mathbf{d}_{\lambda(i),i} & \mathbf{b}_{\mathcal{B}i} &= {}_i\mathbf{b}_{\mathcal{B}i}\\
\mathbf{\Gamma}_{\mathcal{J}i} &= {}^i\mathbf{\Gamma}_{\lambda(i),i} & \mathbf{U}_{\mathcal{B}i} &= {}_i\mathbf{U}_{\mathcal{B}i}
\end{aligned}$$

2.8 Equations of motion for fixed-base and moving-base systems

This section presents the equations of motion of both fixed-base and moving-base systems. All variables related to the equations of motion can be found in the notation section (Section 2.7) in Tables 2.3, 2.4 and 2.5.

2.8.1 Fixed-base systems

The configuration of a fixed-base system with n_J 1-DoF joints is parametrized as $\mathbf{q}_{fb} := \mathbf{s}$, where $\mathbf{s} \in \mathbb{R}^{n_J}$ is the generalized position vector, also named the system's shape. The total number of degrees of freedom is equal to n_J, since the system only has degrees of freedom in the joints. The time-derivative of the configuration is given by $\dot{\mathbf{q}}_{fb} = \dot{\mathbf{s}}$, from which we define $\mathbf{r} := \dot{\mathbf{s}}$, where $\mathbf{r} \in \mathbb{R}^{n_J}$ is the generalized velocity vector. The dynamics of a fixed-base system is written as

$$\mathbf{M}_{fb}\dot{\mathbf{r}} + \mathbf{C}_{fb}\mathbf{r} + \mathbf{G}_{fb} = \boldsymbol{\tau} + \sum_{k \in \mathcal{I}_C} {}^{C}\mathbf{J}_{A,i}^{T} \; {}_{C}\mathbf{f}_{\mathcal{E}k}, \tag{2.53}$$

where $\dot{\mathbf{r}} \in \mathbb{R}^{n_J}$ the generalized acceleration vector, $\mathbf{M}_{fb} \in \mathbb{R}^{n_J \times n_J}$ the generalized inertia matrix, $\mathbf{C}_{fb} \in \mathbb{R}^{n_J \times n_J}$ the Coriolis matrix for fixed-base systems, $\mathbf{G}_{fb} \in \mathbb{R}^{n_J}$ the gravitational wrench vector for fixed-base systems, $\boldsymbol{\tau} \in \mathbb{R}^{n_J}$ the joint torques or generalized forces vector, $\mathcal{I}_C \in \mathbb{R}$ the set of closed contacts for which i is the body on which closed contact k is acting, ${}^{C}\mathbf{J}_{A,i} \in \mathbb{R}^{6 \times n_J}$ the Jacobian of the velocity of frame i w.r.t. frame A expressed in contact frame C, and ${}_{C}\mathbf{f}_{\mathcal{E}k} \in \mathbb{R}^6$ the external wrench acting on closed contact k expressed in contact frame C. Recursive algorithms combine the Coriolis, gravitational and external wrenches into a generalized bias wrench. Therefore we rewrite the dynamics (2.53) where we use the generalized bias wrench for fixed-base systems instead of the Coriolis matrix, gravitational wrench vector and external forces and add the kinematics to obtain the equations of motion for fixed-base systems as

$$\dot{\mathbf{s}} = \mathbf{r} \tag{2.54}$$

and

$$\mathbf{M}_{fb}\dot{\mathbf{r}} + \mathbf{h}_{fb} = \boldsymbol{\tau}, \tag{2.55}$$

where $\mathbf{h}_{fb}$ is the generalized bias wrench vector for fixed-base systems, defined as

$$\mathbf{h}_{fb} := \mathbf{C}_{fb}\mathbf{r} + \mathbf{G}_{fb} - \sum_{k \in \mathcal{I}_C} {}^{C}\mathbf{J}_{A,i}^{T} \; {}_{C}\mathbf{f}_{\mathcal{E}k}. \tag{2.56}$$

2.8.2 Moving-base systems

The configuration of a moving-base system with n_J 1-DoF joints is parametrized as $\mathbf{q} := (\mathbf{H}, \mathbf{s})$, where $\mathbf{H} := {}^{A}\mathbf{H}_0 \in \mathrm{SE}(3)$ is the moving-base transformation matrix, where A is the inertial frame and 0 is the moving-base frame. The total number of degrees of freedom n is given by $n = n_J + 6$, as there are six degrees of freedom in the moving-base. The time-derivative of the configuration is given by $\dot{\mathbf{q}} = (\dot{\mathbf{H}}, \dot{\mathbf{s}})$, from which we define $\boldsymbol{\nu} := (\mathbf{v}, \mathbf{r})$ through left-trivialization (see Subsection 2.4.2), where $\mathbf{v} := {}^{0}\mathbf{v}_{A,0} \in \mathbb{R}^6$ is the moving-base twist. The kinematics of the moving base are given by $\dot{\mathbf{H}} = \mathbf{H}\mathbf{v}^{\wedge}$ and the kinematics of the joints by $\dot{\mathbf{s}} = \mathbf{r}$. The dynamics of a moving-base system is written as

$$\mathbf{M}\dot{\boldsymbol{\nu}} + \mathbf{C}\boldsymbol{\nu} + \mathbf{G} = \mathbf{S}\boldsymbol{\tau} + \sum_{k \in \mathcal{I}_C} {}^{C}\mathbf{J}_{A,i/0}^{T} \; {}_{C}\mathbf{f}_{\mathcal{E}k}, \tag{2.57}$$

where $\mathbf{M} \in \mathbb{R}^{n\times n}$ is the mass matrix, $\mathbf{C} \in \mathbb{R}^{n\times n}$ the Coriolis matrix, $\mathbf{G} \in \mathbb{R}^{n}$ the gravitational wrench vector, ${}^{C}\mathbf{J}^{T}_{A,i/0} \in \mathbb{R}^{6\times n}$ the geometric Jacobian of the velocity of frame i w.r.t. frame A expressed in contact frame C with moving-base frame 0, and

$$\mathbf{S} := [0_{6\times n_J}; I_{n_J}] \in \mathbb{R}^{n\times n_J} \tag{2.58}$$

the joint selection matrix. Recursive algorithms for moving-base systems also make use of the generalized bias wrench. Therefore we rewrite the dynamics (2.43) in terms of the generalized bias wrench and add the kinematics to obtain the equations of motion for moving-base systems as

$$\dot{\mathbf{H}} = \mathbf{H}\mathbf{v}^{\wedge}, \tag{2.59}$$

$$\dot{\mathbf{s}} = \mathbf{r}, \tag{2.60}$$

and

$$\mathbf{M}\dot{\boldsymbol{\nu}} + \mathbf{h} = \mathbf{S}\boldsymbol{\tau}, \tag{2.61}$$

where $\mathbf{h}$ is the generalized bias wrench vector, defined as

$$\mathbf{h} := \mathbf{C}\boldsymbol{\nu} + \mathbf{G} - \sum_{k\in\mathcal{I}_C} {}^{C}\mathbf{J}^{T}_{A,i/0}\ {}_{C}\mathbf{f}_{\mathcal{E}k}. \tag{2.62}$$

The dynamics (2.61) can be written in matrix form as

$$\begin{bmatrix}\mathbf{M}_{11} & \mathbf{M}_{12}\\ \mathbf{M}_{21} & \mathbf{M}_{22}\end{bmatrix}\begin{bmatrix}\dot{\mathbf{v}}\\ \dot{\mathbf{r}}\end{bmatrix} + \begin{bmatrix}\mathbf{h}_1\\ \mathbf{h}_2\end{bmatrix} = \begin{bmatrix}0_{6\times 1}\\ \boldsymbol{\tau}\end{bmatrix}, \tag{2.63}$$

where $\mathbf{M}_{11} \in \mathbb{R}^{6\times 6}$ is the 6D inertia matrix of the whole system in its current shape, $\mathbf{M}_{12} = \mathbf{M}_{21}^{\mathrm{T}} \in \mathbb{R}^{6\times n_J}$ is a matrix stacking the wrenches required to support unit acceleration [8, Section 9.3], $\mathbf{M}_{22} \in \mathbb{R}^{n_J\times n_J}$ the generalized inertia matrix, $\mathbf{h}_1 \in \mathbb{R}^{6}$ the bias wrench for the whole system as a composite rigid body and $\mathbf{h}_2 \in \mathbb{R}^{n_J}$ the generalized bias wrench vector of the joints, so that

$$\mathbf{M} = \begin{array}{cc} \begin{array}{cc} 6 & n_J \end{array} & \\ \begin{bmatrix}\mathbf{M}_{11} & \mathbf{M}_{12}\\ \mathbf{M}_{21} & \mathbf{M}_{22}\end{bmatrix} & \begin{array}{c} 6 \\ n_J \end{array} \end{array} \tag{2.64}$$

and

$$\mathbf{h} = \begin{array}{cc} 1 & \\ \begin{bmatrix}\mathbf{h}_1\\ \mathbf{h}_2\end{bmatrix} & \begin{array}{c} 6 \\ n_J \end{array} \end{array}. \tag{2.65}$$

Assumption 2.3: In this thesis we assume that there are no external wrenches, so ${}_{C}\mathbf{f}_{\mathcal{E}k} = 0\ \forall\ k$. Therefore we write the generalized bias wrench vector (2.62) without external wrenches as

$$\mathbf{h} = \mathbf{C}\boldsymbol{\nu} + \mathbf{G}. \tag{2.66}$$

2.9 Recursive rigid body dynamics algorithms

Recursive rigid body dynamics algorithms can compute the matrices and vectors of multibody equations of motion (2.55) and (2.61) in an efficient way by exploiting the sparsity in these matrices. The concept is simple: there exist multiple ways to compute a vector a of dimension n, defined as

$$a_i := \sum_{j=1}^{i} b_j \in \mathbb{R}, \tag{2.67}$$

where $b \in \mathbb{R}^n$ and $b_j \in \mathbb{R}$ is the j-th element of vector b. One can create a for loop from 1 to i which adds b_j to a_i, and does this for every a_i. When i gets relatively big, the computation of a will take many additions. This method has a number of operations of order $O(n^2)$. Another possibility is to exploit the knowledge about the structure of a and compute

$$a_i = a_{i-1} + b_i, \tag{2.68}$$

which always only consists of a single addition. This method has a number of operations of order $O(n)$. The concept of (2.68) forms the basis of recursive algorithms.

Three well-known recursive algorithms are the Recursive Newton Euler Algorithm (RNEA) [19], the Composite Rigid Body Algorithm (CRBA) [20] and the Articulated Body Algorithm [8, Chapter 7]. There are variants for both fixed-base and moving-base systems. Furthermore we briefly explore recursive analytical derivatives algorithms, which are based on recursive algorithms. The dynamics variables used here can be found in the notation section (Section 2.7) in Tables 2.3, 2.4 and 2.5.

Fixed-base recursive algorithms. The inverse dynamics is the necessary joint torques to achieve certain desired joint accelerations. It is often used in control, trajectory design and optimization, and mechanical design applications. The RNEA computes the inverse dynamics of a fixed-base system (2.55) as

$$\boldsymbol{\tau}(\mathbf{s}, \mathbf{r}, \dot{\mathbf{r}}) = \mathbf{M}_{fb}(\mathbf{s})\dot{\mathbf{r}} + \mathbf{h}_{fb}(\mathbf{s}, \mathbf{r}). \tag{2.69}$$

The RNEA is also useful for computing the generalized bias wrench vector $\mathbf{h}_{fb}(\mathbf{s}, \mathbf{r})$. If one chooses $\dot{\mathbf{r}} = 0$ as input, the result of the RNEA is simply

$$\boldsymbol{\tau}(\mathbf{s}, \mathbf{r}, 0) = \mathbf{h}_{fb}(\mathbf{s}, \mathbf{r}). \tag{2.70}$$

The RNEA can be found in Appendix E.1. The CRBA computes the mass matrix $\mathbf{M}_{fb}(\mathbf{s})$ in a recursive manner. By combining it with the RNEA to compute $\mathbf{h}_{fb}(\mathbf{s}, \mathbf{r})$, the entire dynamics can be found. This is a method to compute the forward dynamics. The CRBA can be found in Appendix E.3. The forward dynamics is used to compute the joint accelerations with given joint torques. It is often used in simulations and prediction. The ABA computes the forward dynamics of a fixed-base systems (2.55) in a recursive manner as

$$\dot{\mathbf{r}}(\mathbf{s}, \mathbf{r}, \boldsymbol{\tau}) = \mathbf{M}_{fb}^{-1}(\mathbf{s})\big(\boldsymbol{\tau} - \mathbf{h}_{fb}(\mathbf{s}, \mathbf{r})\big). \tag{2.71}$$

The ABA can be found in Appendix E.5.

Moving-base recursive algorithms. As moving-base systems have $n = n_J + 6$ degrees of freedom, but only n_J actuators, they are underactuated. Therefore there is no proper definition of inverse dynamics for moving-base systems. Even though it is not possible to achieve both a desired moving-base acceleration as well as desired joint accelerations, it is possible to achieve only desired joint accelerations without specifying the moving-base acceleration. This method is called the Recursive Newton Euler Algorithm for moving-base systems (RNEAmb). It computes the joint torques of (2.61) as

$$\mathbf{S}\boldsymbol{\tau}(\mathbf{q}, \boldsymbol{\nu}, \dot{\boldsymbol{\nu}}) = \mathbf{M}(\mathbf{s})\dot{\boldsymbol{\nu}} + \mathbf{h}(\mathbf{q}, \boldsymbol{\nu}). \tag{2.72}$$

Similar to the RNEA, the RNEAmb can also be used to compute the generalized bias wrench vector $\mathbf{h}(\mathbf{q}, \boldsymbol{\nu})$ of (2.65) by specifying $\dot{\boldsymbol{\nu}} = 0$ as input, the result of the RNEAmb is simply

$$\mathbf{S}\boldsymbol{\tau}(\mathbf{q}, \boldsymbol{\nu}, 0) = \mathbf{h}(\mathbf{q}, \boldsymbol{\nu}). \tag{2.73}$$

The RNEAmb can be found in Appendix E.2. The mass matrix for moving-base systems $\mathbf{M}(\mathbf{s})$ of (2.64) can be computed by the Composite Rigid Body Algorithm for moving-base systems (CRBAmb). Again, by combining it with the RNEAmb to compute $\mathbf{h}(\mathbf{q}, \boldsymbol{\nu})$, the entire dynamics can be found. This is a method to compute the forward dynamics. The CRBAmb can be found in Appendix E.4. The Articulated Body Algorithm for moving-base systems (ABAmb) computes the joint accelerations with given joint torques. Similar to the RNEAmb, the moving-base acceleration can not be specified. The ABAmb computes the joint accelerations of (2.61) as

$$\dot{\mathbf{r}}(\mathbf{q},\boldsymbol{\nu},\boldsymbol{\tau}) = \mathbf{M}^{-1}(\mathbf{s})\big(\boldsymbol{\tau} - \mathbf{h}(\mathbf{q},\boldsymbol{\nu})\big). \tag{2.74}$$

The ABAmb can be found in Appendix E.6.

Recursive analytical derivatives algorithms. In [7], the authors present a method, which we call recursive analytical derivatives, to efficiently compute the derivatives of the inverse dynamics of fixed-base systems. It derives the equations in the RNEA step by step, by applying the chain rule. This method is limited to systems that have its configuration space modelled as a vector space. The algorithms that compute the derivatives of the inverse dynamics are presented in Appendix E.8.

2.10 Summary

An overview of all background information related to this thesis has been presented in this chapter. The representations of the pose of a moving base have been discussed, from which we have concluded that the rotation matrix is the only representation of a rotation which is both globally defined and a unique representation. We have presented the fundamentals of three topics: sensitivity analysis on vector spaces, Lie group theory and the Lie group SE(3). The theory of left-trivialized linearization, which is a sensitivity analysis method on Lie groups, has been recalled. After that we have presented definitions and notation for multibody systems. We have used the definitions and notation to present the equation of motion for both fixed-base and moving-base systems. Lastly we have elaborated the concept of recursive rigid body algorithms, and discussed commonly used existing recursive algorithms for both fixed-base and moving-base systems.

Chapter 3

Left-Trivialized Linearization: Theoretical Aspects

In this chapter, we derive the mathematical formulas for the left-trivialized linearization of the forward dynamics of moving-base multibody systems. To do so, we first present the Lie group on which moving-base multibody dynamics are defined, including its Lie algebra, left-translation, left-trivialized tangent map and adjoint representations. Then the state and input matrices of the forward dynamics of moving-base systems are presented, which are obtained through left-trivialized linearization. Finally, we extend the system with (non-physical) inputs to derive the relation between the left-trivialized derivatives of the forward dynamics and those of the extended inverse dynamics. Using this relation, we express the forward dynamics derivatives in terms of the extended inverse dynamics derivatives and the inverse of the mass matrix.

3.1 Lie group of moving-base systems

The configuration manifold of a moving-base multibody system composed of a moving base to which several 1-DoF joints are attached is

$$Q = \mathrm{SE}(3) \times \mathbb{R}^{n_J}, \tag{3.1}$$

where $\times$ denotes the Cartesian product of sets. The moving-base pose (translation and rotation) is represented by $\mathbf{H} := {}^{A}\mathbf{H}_0 \in \mathrm{SE}(3)$ as defined in (2.24), and the joint displacements are lumped into the vector $\mathbf{s} \in \mathbb{R}^{n_J}$. The Cartesian product of these two sets combines them into a single set. We recall that the configuration was defined in Section 2.8 as $\mathbf{q} = (\mathbf{H}, \mathbf{s})$, which belongs to the manifold Q.

The state manifold of the moving-base system is

$$TQ = T\big(\mathrm{SE}(3) \times \mathbb{R}^{n_J}\big) = T\mathrm{SE}(3) \times T\mathbb{R}^{n_J}, \tag{3.2}$$

which is the tangent bundle of the configuration manifold Q. The moving-base state is represented by $(\mathbf{H}, \dot{\mathbf{H}}) \in T\mathrm{SE}(3)$, which represents the pose of the base and its time-derivative. The joint state is represented by $(\mathbf{s}, \dot{\mathbf{s}}) \in T\mathbb{R}^{n_J}$, which represents the displacements of the joints and their time-derivative. Applying left-trivialization to $\dot{\mathbf{H}}$ in $(\mathbf{H}, \dot{\mathbf{H}}) \in T\mathrm{SE}(3)$ results in $(\mathbf{H}, \mathbf{v}^{\wedge}) \in \mathrm{SE}(3) \times \mathfrak{se}(3)$, which can be further identified with $(\mathbf{H}, \mathbf{v}) \in \mathrm{SE}(3) \times \mathbb{R}^6$ by applying the vee operation on $\mathbf{v}^{\wedge}$, as in (2.31). This shows that, as a set, $\mathrm{SE}(3) \times \mathbb{R}^6$ is diffeomorphic to $T\mathrm{SE}(3)$. The manifold $T\mathbb{R}^{n_J}$ can be identified with $\mathbb{R}^{n_J} \times \mathbb{R}^{n_J}$, as they are also diffeomorphic. Therefore, we have the following identification

$$TQ \cong \mathrm{SE}(3) \times \mathbb{R}^{n_J} \times \mathbb{R}^6 \times \mathbb{R}^{n_J}, \tag{3.3}$$

where $\cong$ denotes a diffeomorphism. An element of the state manifold TQ is $(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}) \in TQ$, which is used in writing the moving-base dynamics (2.59), (2.60), and (2.61).

The state manifold defined above is not yet a Lie group, as a Lie group is defined by a set and an operation and we have not defined an operation yet. Taking two elements of the set $\mathrm{SE}(3) \times \mathbb{R}^{n_J} \times \mathbb{R}^6 \times \mathbb{R}^{n_J}$, $(\mathbf{H}_1, \mathbf{s}_1, \mathbf{v}_1, \mathbf{r}_1)$ and $(\mathbf{H}_2, \mathbf{s}_2, \mathbf{v}_2, \mathbf{r}_2)$, we define the group operation as

$$(\mathbf{H}_1, \mathbf{s}_1, \mathbf{v}_1, \mathbf{r}_1) \cdot (\mathbf{H}_2, \mathbf{s}_2, \mathbf{v}_2, \mathbf{r}_2) = (\mathbf{H}_1\mathbf{H}_2, \mathbf{s}_1 + \mathbf{s}_2, \mathbf{v}_1 + \mathbf{v}_2, \mathbf{r}_1 + \mathbf{r}_2). \tag{3.4}$$

There exist alternative group operations which may be chosen. An example for $TQ = T\mathrm{SE}(3) \times T\mathbb{R}^{n_J}$ would be

$$(\mathbf{H}_1, \mathbf{s}_1, \partial\mathbf{H}_1, \mathbf{r}_1) \cdot (\mathbf{H}_2, \mathbf{s}_2, \partial\mathbf{H}_2, \mathbf{r}_2) = (\mathbf{H}_1\mathbf{H}_2, \mathbf{s}_1 + \mathbf{s}_2, \mathbf{H}_1\partial\mathbf{H}_2 + \partial\mathbf{H}_1\mathbf{H}_2, \mathbf{r}_1 + \mathbf{r}_2), \tag{3.5}$$

which, together with manifold TQ forms the tangent group of $\mathrm{SE}(3) \times \mathbb{R}^{n_J}$. In [11], the authors have shown that for the case of SO(3) the group operation similar to (3.4) is much more simple in terms of complexity of the expressions obtained than (3.5).

To express a Lie group as a combination of multiple Lie groups, the group direct product $\times$ [35, Chapter 2.4] is used, which is similar to the how Cartesian product combines sets. Using the set (3.3), the operation (3.4) and the group direct product, the Lie group on which moving-base multibody systems are defined is

$$G = \mathrm{SE}(3) \times \mathbb{R}^{n_J} \times \mathbb{R}^6 \times \mathbb{R}^{n_J}. \tag{3.6}$$

In this way G is the group direct product of the Lie groups SE(3), $\mathbb{R}^{n_J}$, $\mathbb{R}^6$ and $\mathbb{R}^{n_J}$.

The Lie algebra of G in (3.6) is

$$\mathfrak{g} = \mathfrak{se}(3) \oplus \mathbb{R}^{n_J} \oplus \mathbb{R}^6 \oplus \mathbb{R}^{n_J}, \tag{3.7}$$

where $\oplus$ denotes the direct sum of two Lie algebras [35, Chapter 7.2]. The direct sum combines two Lie algebra's into a single Lie algebra, similar to the group direct product and Cartesian product. Therefore $\mathfrak{g}$ is the direct sum of the Lie algebras $\mathfrak{se}(3)$, $\mathbb{R}^{n_J}$, $\mathbb{R}^6$ and $\mathbb{R}^{n_J}$. The sub-algebra $\mathfrak{se}(3)$ can be identified with $\mathbb{R}^6_\times$, as shown in Subsection 2.4.2, where $\mathbb{R}^6_\times$ is the vector space $\mathbb{R}^6$ with the Lie bracket given by the 6D cross product which is defined in (2.33). We identify the Lie algebra with

$$\mathfrak{g} \cong \mathbb{R}^6_\times \oplus \mathbb{R}^{n_J} \oplus \mathbb{R}^6 \oplus \mathbb{R}^{n_J}, \tag{3.8}$$

which, with a slight abuse of terminology, we call the Lie algebra of Lie group G. The left-translation on the Lie group G is

$$L_{(\mathbf{H}_1, \mathbf{s}_1, \mathbf{v}_1, \mathbf{r}_1)}(\mathbf{H}_2, \mathbf{s}_2, \mathbf{v}_2, \mathbf{r}_2) = (\mathbf{H}_1\mathbf{H}_2, \mathbf{s}_1 + \mathbf{s}_2, \mathbf{v}_1 + \mathbf{v}_2, \mathbf{r}_1 + \mathbf{r}_2) \in G, \tag{3.9}$$

which has the tangent map

$$\mathrm{D}\, L_{(\mathbf{H}_1, \mathbf{s}_1, \mathbf{v}_1, \mathbf{r}_1)}(\delta\mathbf{H}_2, \delta\mathbf{s}_2, \delta\mathbf{v}_2, \delta\mathbf{r}_2) = (\mathbf{H}_1\delta\mathbf{H}_2, \delta\mathbf{s}_2, \delta\mathbf{v}_2, \delta\mathbf{r}_2) \in T(\mathrm{SE}(3) \times \mathbb{R}^{n_J} \times \mathbb{R}^6 \times \mathbb{R}^{n_J}), \tag{3.10}$$

which we call the *left-translated tangent map* of Lie group G. Using an element of the Lie algebra, $(z_H, z_s, z_\mathrm{v}, z_r) \in \mathfrak{g}$, the adjoint representation (defined in Subsection 2.3.3) of Lie group G into its Lie algebra $\mathfrak{g}$ is

$$\mathrm{Ad}_{(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r})}(z_H, z_s, z_\mathrm{v}, z_r) = (\mathbf{X}z_H, z_s, z_\mathrm{v}, z_r), \tag{3.11}$$

where $\mathbf{X}$ is the velocity transformation matrix, defined in (2.36). The adjoint representation (defined in Subsection 2.3.3) of the Lie algebra $\mathfrak{g}$ onto itself is

$$\mathrm{ad}_{(z_{H1}, z_{s1}, z_{\mathrm{v}1}, z_{r1})}(z_{H2}, z_{s2}, z_{\mathrm{v}2}, z_{r2}) = (z_{H1} \times z_{H2}, 0, 0, 0), \tag{3.12}$$

where $\times$ denotes the 6D cross product as defined in (2.33). The matrix representation of the adjoint action of the Lie algebra $\mathfrak{g}$ onto itself is

$$\mathrm{ad}_{(z_H, z_s, z_{\mathrm{v}}, z_r)} = \begin{array}{c} \\ \\ \\ \\ \\ \end{array}\begin{array}{cccc|c} 6 & n_J & 6 & n_J & \\ \hline z_H\times & 0 & 0 & 0 & 6 \\ 0 & 0 & 0 & 0 & n_J \\ 0 & 0 & 0 & 0 & 6 \\ 0 & 0 & 0 & 0 & n_J \end{array} . \tag{3.13}$$

The state $g(t)$ and input $u(t)$ of the moving-base system (2.43) are given by

$$g(t) = (\mathbf{H}(t), \mathbf{s}(t), \mathbf{v}(t), \mathbf{r}(t)) \in G, \tag{3.14}$$

and

$$u(t) = \boldsymbol{\tau}(t) \in \mathbb{R}^{n_J}, \tag{3.15}$$

so that the nominal trajectory about which we linearize is given by

$$t \mapsto \eta(t) = \big(g(t), u(t)\big) = \Big(\big(\mathbf{H}(t), \mathbf{s}(t), \mathbf{v}(t), \mathbf{r}(t)\big), \boldsymbol{\tau}(t)\Big) \in G \times \mathbb{R}^{n_J}. \tag{3.16}$$

3.2 State and input linearization matrices for moving-base systems

We first need to properly define the forward dynamics before we can derive the left-trivialized linearization of the forward dynamics. By reordering the moving-base systems dynamics (2.61), the forward dynamics function $FD : G \times \mathbb{R}^{n_J} \to \mathbb{R}^n$ is defined as

$$FD(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \boldsymbol{\tau}) := \dot{\boldsymbol{\nu}}(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \boldsymbol{\tau}) = \mathbf{M}^{-1}(\mathbf{s})\big[-\mathbf{h}(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}) + \mathbf{S}\boldsymbol{\tau}\big], \tag{3.17}$$

where $\mathbf{H} = {}^{A}\mathbf{H}_0, \mathbf{s}, \mathbf{v} = {}^{0}\mathbf{v}_{A,0}, \mathbf{r}, \boldsymbol{\tau}, \boldsymbol{\nu}, \mathbf{M}, \mathbf{h}$ and $\mathbf{S}$ are as defined in Subsection 2.8.2 and can be found in the notation section (Section 2.7) in Tables 2.3 and 2.5. The forward dynamics (3.17) can be further split up as $FD = [FD_b; FD_j]$ where

$$FD_b(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \boldsymbol{\tau}) := \dot{\mathbf{v}}(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \boldsymbol{\tau}) \tag{3.18}$$

and

$$FD_j(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \boldsymbol{\tau}) := \dot{\mathbf{r}}(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \boldsymbol{\tau}), \tag{3.19}$$

with $FD_b : G \times \mathbb{R}^{n_J} \to \mathbb{R}^6$ the forward dynamics of the moving base and $FD_j : G \times \mathbb{R}^{n_J} \to \mathbb{R}^{n_J}$ the forward dynamics of the joints.
Proposition 3.1: The left-trivialized linearization of the moving-base kinematics $\dot{\mathbf{H}} = \mathbf{H}\mathbf{v}$ and $\dot{\mathbf{s}} = \mathbf{r}$ (as in (2.59) and (2.60)) and the forward dynamics (3.17) about the trajectory $\eta(t) = (g(t), u(t))$ (as in (3.16)) is given by $\dot{z}(t) = A(\eta, t)z(t) + B(\eta, t)w(t)$ (as in (2.46)), where the perturbation vector is represented by $z = [z_H; z_s; z_{\mathrm{v}}; z_r] \in \mathfrak{g}$, the perturbed input vector as $w = \boldsymbol{\tau} \in \mathbb{R}^{n_J}$ and the state matrix A and input matrix B are given by

$$A(\eta, t) = \begin{array}{cccc|c} 6 & n_J & 6 & n_J & \\ \hline -\mathbf{v}\times & 0 & I & 0 & 6 \\ 0 & 0 & 0 & I & n_J \\ \mathrm{D}_1\, FD_b \circ \mathrm{D}\, L_H(I) & \mathrm{D}_2\, FD_b & \mathrm{D}_3\, FD_b & \mathrm{D}_4\, FD_b & 6 \\ \mathrm{D}_1\, FD_j \circ \mathrm{D}\, L_H(I) & \mathrm{D}_2\, FD_j & \mathrm{D}_3\, FD_j & \mathrm{D}_4\, FD_j & n_J \end{array} , \tag{3.20}$$

and

$$B(\eta,t) = \begin{array}{c} n_J \\ \left[\begin{array}{c} 0 \\ 0 \\ \mathrm{D}_5\, FD_b \\ \mathrm{D}_5\, FD_j \end{array}\right] \end{array} \begin{array}{c} \\ 6 \\ n_J \\ 6 \\ n_J \end{array}, \tag{3.21}$$

where FD_b and FD_j are the forward dynamics of the moving base and of the joints as in (3.18) and (3.19), and their derivatives are evaluated at $FD_b(\mathbf{H},\mathbf{s},\mathbf{v},\mathbf{r},\boldsymbol{\tau})$ and $FD_j(\mathbf{H},\mathbf{s},\mathbf{v},\mathbf{r},\boldsymbol{\tau})$. ■
Proof: Using $g = (\mathbf{H},\mathbf{s},\mathbf{v},\mathbf{r})$ and $u = \boldsymbol{\tau}$ as defined in (3.14) and (3.15), the left-trivialized vector field $\lambda(g,u,t)$ of (2.44) for the dynamics of (2.59), (2.60) and (3.17) is

$$\begin{aligned} \lambda(g,u,t) &= (\mathbf{H}^{-1}, -\mathbf{s}, -\mathbf{v}, -\mathbf{r})(\dot{\mathbf{H}}, \dot{\mathbf{s}}, \dot{\mathbf{v}}, \dot{\mathbf{r}}), \\ &= (\mathbf{H}^{-1}\dot{\mathbf{H}}, \dot{\mathbf{s}}, \dot{\mathbf{v}}, \dot{\mathbf{r}}), \\ &= (\mathbf{v}, \mathbf{r}, \dot{\mathbf{v}}, \dot{\mathbf{r}}), \end{aligned} \tag{3.22}$$

where $\dot{\mathbf{s}} = \mathbf{r}$ and $\mathbf{v} = (\mathbf{H}^{-1}\dot{\mathbf{H}})^\vee$ as defined in Section 2.8. For the Lie group $G = \mathrm{SE}(3) \times \mathbb{R}^{n_J} \times \mathbb{R}^6 \times \mathbb{R}^{n_J}$ of (3.6), the state matrix A (2.47) and input matrix B (2.48) become

$$A(\eta,t) := \mathrm{D}_1\, \lambda(g,u,t) \circ \mathrm{D}\, L_{(\mathbf{H},\mathbf{s},\mathbf{v},\mathbf{r})}(I,0,0,0) - \mathrm{ad}_{\lambda(g,u,t)} \tag{3.23}$$

and

$$B(\eta,t) := \mathrm{D}_2\, \lambda(g,u,t). \tag{3.24}$$

The matrix form of the adjoint representation in (2.16) of the Lie algebra $\lambda(g,u,t)$ onto itself, $\mathrm{ad}_{\lambda(g,u,t)}$, is given by

$$\mathrm{ad}_{\lambda(g,u,t)} = \begin{array}{c} \begin{array}{cccc} 6 & n_J & 6 & n_J \end{array} \\ \left[\begin{array}{cccc} \mathbf{v}\times & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right] \end{array} \begin{array}{c} \\ 6 \\ n_J \\ 6 \\ n_J \end{array}. \tag{3.25}$$

The left-translated tangent map $\mathrm{D}\, L_{(\mathbf{H},\mathbf{s},\mathbf{v},\mathbf{r})}(I,0,0,0)$ in the direction $z = (z_H, z_s, z_\mathrm{v}, z_r)$ is given by

$$\mathrm{D}\, L_{(\mathbf{H},\mathbf{s},\mathbf{v},\mathbf{r})}(I,0,0,0) \cdot z = (\mathrm{D}\, L_H(I) \cdot z_H, z_s, z_\mathrm{v}, z_r). \tag{3.26}$$

The left-trivialized tangent map $\mathrm{D}_1\, \lambda(g,u,t) \circ \mathrm{D}\, L_{(\mathbf{H},\mathbf{s},\mathbf{v},\mathbf{r})}(I,0,0,0)$ is found by combining the forward dynamics of the base and joints given by (3.18) and (3.19), the left-trivialized vector field $\lambda(g,u,t)$ given by (3.22) and the left-translated tangent map given by (3.26) as

$$\mathrm{D}_1\, \lambda(g,u,t) \circ \mathrm{D}\, L_{(\mathbf{H},\mathbf{s},\mathbf{v},\mathbf{r})}(I,0,0,0) = \begin{array}{c} \begin{array}{cccc} 6 & n_J & 6 & n_J \end{array} \\ \left[\begin{array}{cccc} 0 & 0 & I & 0 \\ 0 & 0 & 0 & I \\ \mathrm{D}_1\, FD_b \circ \mathrm{D}\, L_H(I) & \mathrm{D}_2\, FD_b & \mathrm{D}_3\, FD_b & \mathrm{D}_4\, FD_b \\ \mathrm{D}_1\, FD_j \circ \mathrm{D}\, L_H(I) & \mathrm{D}_2\, FD_j & \mathrm{D}_3\, FD_j & \mathrm{D}_4\, FD_j \end{array}\right] \end{array} \begin{array}{c} \\ 6 \\ n_J \\ 6 \\ n_J \end{array}, \tag{3.27}$$

which makes it clear that the left-translated tangent maps of FD_b and FD_j with respect to $\mathbf{s}, \mathbf{v}$ and $\mathbf{r}$ are nothing more than ordinary tangent vector maps. This corresponds to the expected result, since $\mathbf{s}, \mathbf{v}$ and $\mathbf{r}$ are elements of vector spaces.

Combining the definition of the state matrix (2.47), the matrix form of the adjoint representation (3.25) and the left-trivialized tangent map (3.27), the state matrix $A(\eta, t)$ is found as (3.20). The input matrix $B(\eta, t)$ is found as in (3.21), since

$$\mathrm{D}_2\, \lambda(g, u, t) = \begin{array}{c} n_J \\ \begin{bmatrix} 0 \\ 0 \\ \mathrm{D}_5\, FD_b \\ \mathrm{D}_5\, FD_j \end{bmatrix} \end{array} \begin{array}{c} \\ 6 \\ n_J \\ 6 \\ n_J \end{array} . \tag{3.28}$$

□

The following proposition presents an expression for the computation of the input matrix in (3.21).

Proposition 3.2 The input matrix $B(\eta, t)$ in (3.21) can be computed as

$$B(\eta, t) = \begin{array}{c} n_J \\ \begin{bmatrix} 0 \\ \mathbf{M}^{-1}\mathbf{S} \end{bmatrix} \end{array} \begin{array}{c} \\ n \\ n \end{array} , \tag{3.29}$$

where $\mathbf{M}$ and $\mathbf{S}$ are the mass matrix and selection matrix, as defined in Section 2.8.2. ■

Proof: The input matrix of (3.21) is the same as (2.48), namely

$$B(\eta, t) = \begin{array}{c} n_J \\ \begin{bmatrix} 0 \\ \mathrm{D}_5\, FD \end{bmatrix} \end{array} \begin{array}{c} \\ n \\ n \end{array} . \tag{3.30}$$

Finally,

$$\mathrm{D}_5\, FD = \frac{\partial \dot{\boldsymbol{\nu}}}{\partial \boldsymbol{\tau}} = \mathbf{M}^{-1}\mathbf{S}, \tag{3.31}$$

where $FD(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \boldsymbol{\tau})$ was defined in (3.17). □

3.3 Obtaining the forward dynamics linearization by using inverse dynamics

We first need to properly define the inverse dynamics before we can express the state matrix of the forward dynamics linearization in (3.20) in terms of the inverse dynamics. However, for underactuated systems, such as moving-base systems, the amount of inputs is less than the amount of degrees of freedom, meaning that there are generally not enough actuators to achieve a desired set of accelerations in all degrees of freedom. As we can not generally achieve a desired set of acceleration, it is impossible to define the inverse dynamics. We can, however, provide underactuated systems with non-physical control inputs to obtain a (non-physical) fully actuated system [31], which we call the *extended* system. We define the non-physical control input $\bar{\boldsymbol{\tau}}_b \in \mathbb{R}^6$, which can be seen as a virtual wrench applied to the base which controls all degrees of freedom of the base, making the base and thus the whole system fully actuated. The *extended input vector* $\bar{\boldsymbol{\tau}} \in \mathbb{R}^n$ is defined as

$$\bar{\boldsymbol{\tau}} = \begin{bmatrix} \bar{\boldsymbol{\tau}}_b \\ \boldsymbol{\tau} \end{bmatrix}. \tag{3.32}$$

With the extended input vector, we can define the *extended dynamics* as

$$\mathbf{M}\dot{\boldsymbol{\nu}} + \mathbf{h} = \bar{\boldsymbol{\tau}}, \tag{3.33}$$

where $\mathbf{M}$, $\boldsymbol{\nu}$ and $\mathbf{h}$ are defined as in Subsection 2.8.2 and can be found in the notation section (Section 2.7) in Tables 2.3 and 2.5. Highlighting the base and joint dynamics, the extended dynamics can be written as

$$\begin{bmatrix} \mathbf{M}_{11} & \mathbf{M}_{12} \\ \mathbf{M}_{21} & \mathbf{M}_{22} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{v}} \\ \dot{\mathbf{r}} \end{bmatrix} + \begin{bmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \end{bmatrix} = \begin{bmatrix} \bar{\boldsymbol{\tau}}_b \\ \boldsymbol{\tau} \end{bmatrix}, \tag{3.34}$$

where $\mathbf{M}_{11}$, $\mathbf{M}_{12}$, $\mathbf{M}_{21}$, $\mathbf{M}_{22}$, $\mathbf{v}$, $\mathbf{r}$, $\mathbf{h}_1$ and $\mathbf{h}_2$ are as defined in Subsection 2.8.2. More abstractly, we write the extended dynamical system of (2.43) as

$$\dot{g}(t) = \bar{f}(g, \bar{u}, t), \tag{3.35}$$

where $g(t) = (\mathbf{H}(t), \mathbf{s}(t), \mathbf{v}(t), \mathbf{r}(t))$ as in (3.14), $\bar{f} : G \times \mathbb{R}^n \times R \to TG$, $(g, \bar{u}, t) \mapsto \bar{f}(g, \bar{u}, t)$ and $\bar{u} = \bar{\boldsymbol{\tau}} \in \mathbb{R}^n$. We define the extended forward dynamics $\overline{FD} : G \times \mathbb{R}^6 \times \mathbb{R}^{n_J} \to \mathbb{R}^n$ of the extended system (3.33) as

$$\overline{FD}(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \bar{\boldsymbol{\tau}}_b, \boldsymbol{\tau}) := \dot{\boldsymbol{\nu}}(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \bar{\boldsymbol{\tau}}_b, \boldsymbol{\tau}) = \mathbf{M}^{-1}(\mathbf{s})\big[-\mathbf{h}(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}) + \bar{\boldsymbol{\tau}}(\bar{\boldsymbol{\tau}}_b, \boldsymbol{\tau})\big], \tag{3.36}$$

which can be further split up as $\overline{FD} = [\overline{FD}_b; \overline{FD}_j]$ with

$$\overline{FD}_b(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \bar{\boldsymbol{\tau}}_b, \boldsymbol{\tau}) := \dot{\mathbf{v}}(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \bar{\boldsymbol{\tau}}_b, \boldsymbol{\tau}) \tag{3.37}$$

and

$$\overline{FD}_j(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \bar{\boldsymbol{\tau}}_b, \boldsymbol{\tau}) := \dot{\mathbf{r}}(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \bar{\boldsymbol{\tau}}_b, \boldsymbol{\tau}), \tag{3.38}$$

with $\overline{FD}_b : G \times \mathbb{R}^6 \times \mathbb{R}^{n_J} \to \mathbb{R}^6$ the extended forward dynamics of the moving base and $\overline{FD}_j : G \times \mathbb{R}^6 \times \mathbb{R}^{n_J} \to \mathbb{R}^{n_J}$ the extended forward dynamics of the joints. Since the extended system (3.33) is fully actuated, we can define its inverse dynamics $\overline{ID} : G \times \mathbb{R}^6 \times \mathbb{R}^{n_J} \to \mathbb{R}^n$ as

$$\overline{ID}(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \dot{\mathbf{v}}, \dot{\mathbf{r}}) := \bar{\boldsymbol{\tau}}(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \dot{\mathbf{v}}, \dot{\mathbf{r}}) = \mathbf{M}(\mathbf{s})\, \dot{\boldsymbol{\nu}}(\dot{\mathbf{v}}, \dot{\mathbf{r}}) + \mathbf{h}(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}), \tag{3.39}$$

which can be further split up as $\overline{ID} = [\overline{ID}_b; \overline{ID}_j]$ with

$$\overline{ID}_b(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \dot{\mathbf{v}}, \dot{\mathbf{r}}) := \bar{\boldsymbol{\tau}}_b(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \dot{\mathbf{v}}, \dot{\mathbf{r}}) = \mathbf{M}_{11}(\mathbf{s})\dot{\mathbf{v}} + \mathbf{M}_{12}(\mathbf{s})\dot{\mathbf{r}} + \mathbf{h}_1(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}) \tag{3.40}$$

and

$$\overline{ID}_j(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \dot{\mathbf{v}}, \dot{\mathbf{r}}) := \boldsymbol{\tau}(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \dot{\mathbf{v}}, \dot{\mathbf{r}}) \ = \mathbf{M}_{21}(\mathbf{s})\dot{\mathbf{v}} + \mathbf{M}_{22}(\mathbf{s})\dot{\mathbf{r}} + \mathbf{h}_2(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}). \tag{3.41}$$

Here $\overline{ID}_b : G \times \mathbb{R}^6 \times \mathbb{R}^{n_J} \times \ \to \mathbb{R}^6$ is the extended inverse dynamics of the moving base and $\overline{ID}_j : G \times \mathbb{R}^6 \times \mathbb{R}^{n_J} \to \mathbb{R}^{n_J}$ the extended inverse dynamics of the joints.

The following proposition shows that the state matrix of the forward dynamics linearization in (3.20) (reported below for convenience of the reader) can be expressed in terms of the extended inverse dynamics derivatives and the inverse of the mass matrix. We recall that the state matrix was given by

$$A(\eta, t) = \begin{array}{cccc|c} 6 & n_J & 6 & n_J & \\ -\mathbf{v}\times & 0 & I & 0 & 6 \\ 0 & 0 & 0 & I & n_J \\ \mathrm{D}_1 FD_b \circ \mathrm{D}\, L_H(I) & \mathrm{D}_2 FD_b & \mathrm{D}_3 FD_b & \mathrm{D}_4 FD_b & 6 \\ \mathrm{D}_1 FD_j \circ \mathrm{D}\, L_H(I) & \mathrm{D}_2 FD_j & \mathrm{D}_3 FD_j & \mathrm{D}_4 FD_j & n_J \end{array}. \tag{3.20 revisited}$$

Proposition 3.3: The state matrix $A(\eta, t)$ of (3.20) can be written in terms of the derivatives of the extended inverse dynamics (3.39) and the inverse of the mass matrix as

$$A(\eta, t) = \begin{array}{cccc|c} 6 & n_J & 6 & n_J & \\ -\mathbf{v}\times & 0 & I & 0 & 6 \\ 0 & 0 & 0 & I & n_J \\ -\mathbf{M}^{-1}\mathrm{D}_1 \overline{ID} \circ \mathrm{D}\, L_H(I) & -\mathbf{M}^{-1}\mathrm{D}_2 \overline{ID} & -\mathbf{M}^{-1}\mathrm{D}_3 \overline{ID} & -\mathbf{M}^{-1}\mathrm{D}_4 \overline{ID} & 6 + n_J \end{array}, \tag{3.42}$$

where $\overline{ID}$ is the extended inverse dynamics as in (3.39), and its derivatives are evaluated at $\overline{ID}(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \dot{\mathbf{v}}, \dot{\mathbf{r}})$. Here, $\mathbf{M}, \mathbf{H}, \mathbf{s}, \mathbf{v}$ and $\mathbf{r}$ are as defined in Subsection 2.8.2. ■

Proof: The extended forward and extended inverse dynamics are related to each other through

$$\overline{ID} \circ \overline{FD} = id. \tag{3.43}$$

Note that this relation only holds for fully actuated systems, as there is no definition of inverse dynamics for underactuated systems as explained in the introduction of this subsection. Evaluated at an arbitrary extended joint torques vector $\bar{\boldsymbol{\tau}} \in \mathbb{R}^n$, (3.43) reads

$$\overline{ID}\big(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \overline{FD}_b(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \bar{\boldsymbol{\tau}}_b, \boldsymbol{\tau}), \overline{FD}_j(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \bar{\boldsymbol{\tau}}_b, \boldsymbol{\tau})\big) = \bar{\boldsymbol{\tau}}, \tag{3.44}$$

which in matrix form reads

$$\begin{bmatrix} \overline{ID}_b\big(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \overline{FD}_b(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \bar{\boldsymbol{\tau}}_b, \boldsymbol{\tau}), \overline{FD}_j(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \bar{\boldsymbol{\tau}}_b, \boldsymbol{\tau})\big) \\ \overline{ID}_j\big(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \overline{FD}_b(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \bar{\boldsymbol{\tau}}_b, \boldsymbol{\tau}), \overline{FD}_j(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \bar{\boldsymbol{\tau}}_b, \boldsymbol{\tau})\big) \end{bmatrix} = \begin{bmatrix} \bar{\boldsymbol{\tau}}_b \\ \boldsymbol{\tau} \end{bmatrix}. \tag{3.45}$$

For the sake of readability, we define

$$\dot{\mathbf{v}} := \overline{FD}_b(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \bar{\boldsymbol{\tau}}_b, \boldsymbol{\tau}) \tag{3.46}$$

and

$$\dot{\mathbf{r}} := \overline{FD}_j(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \bar{\boldsymbol{\tau}}_b, \boldsymbol{\tau}). \tag{3.47}$$

Differentiating (3.45) with respect to $\mathbf{H}, \mathbf{s}, \mathbf{v}$ and $\mathbf{r}$, and applying the chain rule formula on the result, we obtain

$$\begin{bmatrix} \mathrm{D}_i \overline{ID}_b \\ \mathrm{D}_i \overline{ID}_j \end{bmatrix} + \begin{bmatrix} \mathrm{D}_5 \overline{ID}_b & \mathrm{D}_6 \overline{ID}_b \\ \mathrm{D}_5 \overline{ID}_j & \mathrm{D}_6 \overline{ID}_j \end{bmatrix} \begin{bmatrix} \mathrm{D}_i \overline{FD}_b \\ \mathrm{D}_i \overline{FD}_j \end{bmatrix} = 0, \tag{3.48}$$

with $i \in \{1, 2, 3, 4\}$, as $\mathrm{D}_i \overline{ID}$ represents the left-trivialized partial derivative of the extended inverse dynamics with respect to the four system state variables. Appendix A.2 explains how (3.48) is found. In (3.48) the arguments of the inverse and forward dynamics functions are $\overline{ID}_b(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \dot{\mathbf{v}}, \dot{\mathbf{r}})$,

$\overline{ID}_j(\mathbf{H},\mathbf{s},\mathbf{v},\mathbf{r},\dot{\mathbf{v}},\dot{\mathbf{r}})$, $\overline{FD}_b(\mathbf{H},\mathbf{s},\mathbf{v},\mathbf{r},\bar{\boldsymbol{\tau}}_b,\boldsymbol{\tau})$ and $\overline{FD}_j(\mathbf{H},\mathbf{s},\mathbf{v},\mathbf{r},\bar{\boldsymbol{\tau}}_b,\boldsymbol{\tau})$, which have been left out for the sake of readability. From (3.40) and (3.41), it follows that

$$\begin{bmatrix} \mathrm{D}_5\,\overline{ID}_b & \mathrm{D}_6\,\overline{ID}_b \\ \mathrm{D}_5\,\overline{ID}_j & \mathrm{D}_6\,\overline{ID}_j \end{bmatrix} = \begin{bmatrix} \mathbf{M}_{11} & \mathbf{M}_{12} \\ \mathbf{M}_{21} & \mathbf{M}_{22} \end{bmatrix} = \mathbf{M} \tag{3.49}$$

Substituting (3.49) into (3.48) and pre-multiplying with $\mathbf{M}^{-1}$ gives

$$\begin{bmatrix} \mathrm{D}_i\,\overline{FD}_b(\mathbf{H},\mathbf{s},\mathbf{v},\mathbf{r},\bar{\boldsymbol{\tau}}_b,\boldsymbol{\tau}) \\ \mathrm{D}_i\,\overline{FD}_j(\mathbf{H},\mathbf{s},\mathbf{v},\mathbf{r},\bar{\boldsymbol{\tau}}_b,\boldsymbol{\tau}) \end{bmatrix} = -\mathbf{M}^{-1} \begin{bmatrix} \mathrm{D}_i\,\overline{ID}_b(\mathbf{H},\mathbf{s},\mathbf{v},\mathbf{r},\dot{\mathbf{v}},\dot{\mathbf{r}}) \\ \mathrm{D}_i\,\overline{ID}_j(\mathbf{H},\mathbf{s},\mathbf{v},\mathbf{r},\dot{\mathbf{v}},\dot{\mathbf{r}}) \end{bmatrix}. \tag{3.50}$$

The relation (3.50) can be applied to the 'real' underactuated dynamics (2.61) where, due to the definitions in (3.17) and (3.36), $\bar{\boldsymbol{\tau}}_b = 0_{6\times 1}$ so that

$$FD_b(\mathbf{H},\mathbf{s},\mathbf{v},\mathbf{r},\boldsymbol{\tau}) = \overline{FD}_b(\mathbf{H},\mathbf{s},\mathbf{v},\mathbf{r},0_{6\times 1},\boldsymbol{\tau}) \tag{3.51}$$

and

$$FD_j(\mathbf{H},\mathbf{s},\mathbf{v},\mathbf{r},\boldsymbol{\tau}) = \overline{FD}_j(\mathbf{H},\mathbf{s},\mathbf{v},\mathbf{r},0_{6\times 1},\boldsymbol{\tau}). \tag{3.52}$$

Substituting (3.51) and (3.52) on the left-hand side in (3.50) results in the relation between the forward dynamics of the 'real' moving-base system and extended inverse dynamics

$$\begin{aligned}\begin{bmatrix} \mathrm{D}_i\,FD_b(\mathbf{H},\mathbf{s},\mathbf{v},\mathbf{r},\boldsymbol{\tau}) \\ \mathrm{D}_i\,FD_j(\mathbf{H},\mathbf{s},\mathbf{v},\mathbf{r},\boldsymbol{\tau}) \end{bmatrix} &= -\mathbf{M}^{-1} \begin{bmatrix} \mathrm{D}_i\,\overline{ID}_b(\mathbf{H},\mathbf{s},\mathbf{v},\mathbf{r},\dot{\mathbf{v}},\dot{\mathbf{r}}) \\ \mathrm{D}_i\,\overline{ID}_j(\mathbf{H},\mathbf{s},\mathbf{v},\mathbf{r},\dot{\mathbf{v}},\dot{\mathbf{r}}) \end{bmatrix} \\ &= -\mathbf{M}^{-1}\,\mathrm{D}_i\,\overline{ID}(\mathbf{H},\mathbf{s},\mathbf{v},\mathbf{r},\dot{\mathbf{v}},\dot{\mathbf{r}}) \end{aligned} \tag{3.53}$$

Combining the original expression for $A(\eta,t)$ given by (3.20) with the relationship (3.53) relating the derivatives of the forward dynamics with those of the extended inverse dynamics, we obtain (3.42). Again note that the left-translated tangent maps of $\overline{ID}_b$ and $\overline{ID}_j$ with respect to $\mathbf{s},\mathbf{v}$ and $\mathbf{r}$ are nothing else than ordinary tangent derivatives, similar to the left-translated tangent maps of FD_b and FD_j in (3.27). □

Note that, given arbitrary inputs $\mathbf{H},\mathbf{s},\mathbf{v},\mathbf{r},\dot{\mathbf{v}}$ and $\dot{\mathbf{r}}$, it is likely that the extended inverse dynamics function $\overline{ID}$ given by (3.39) will return a non-zero base torque $\bar{\boldsymbol{\tau}}_b$. Therefore $\overline{ID}$ does not generally return $\mathbf{S}\boldsymbol{\tau}$ (equivalently $\overline{ID}_b$ and $\overline{ID}_j$ given by (3.40)-(3.41) do not generally return $0_{6\times 1}$ and $\boldsymbol{\tau}$ respectively). Only if the extended inverse dynamics $\overline{ID}$ return $\mathbf{S}\boldsymbol{\tau}$, its inputs are called *consistent*. This is formalized in the following definition.

Definition 3.1: The inputs $\mathbf{H},\mathbf{s},\mathbf{v},\mathbf{r},\dot{\mathbf{v}}$ *and* $\dot{\mathbf{r}}$ *of the extended inverse dynamics function given by* (3.39) *are called* consistent *only if the extended inverse dynamics function returns a value in the form* $\mathbf{S}\boldsymbol{\tau}$*.* ■

Therefore only if the inputs are consistent, the following holds for the extended inverse dynamics functions given by (3.39)-(3.41):

$$\overline{ID}(\mathbf{H},\mathbf{s},\mathbf{v},\mathbf{r},\dot{\mathbf{v}},\dot{\mathbf{r}}) = \mathbf{S}\boldsymbol{\tau}, \tag{3.54}$$

$$\overline{ID}_b(\mathbf{H},\mathbf{s},\mathbf{v},\mathbf{r},\dot{\mathbf{v}},\dot{\mathbf{r}}) = 0_{6\times 1}, \tag{3.55}$$

$$\overline{ID}_j(\mathbf{H},\mathbf{s},\mathbf{v},\mathbf{r},\dot{\mathbf{v}},\dot{\mathbf{r}}) = \boldsymbol{\tau}, \tag{3.56}$$

where $\mathbf{H},\mathbf{s},\mathbf{v},\mathbf{r}$ and $\mathbf{S}$ are defined as in Section 2.8.2.

When the inputs represent a *physical* system (i.e. the sensors obtain data from a physical system or the inputs are numerically obtained through simulations), the inputs $\mathbf{H},\mathbf{s},\mathbf{v},\mathbf{r},\dot{\mathbf{v}}$ and $\dot{\mathbf{r}}$ of the extended inverse dynamics (3.39)-(3.41) are consistent (neglecting sensor noise or rounding errors). For this reason, commonly used moving-base algorithms such as the RNEAmb and ABAmb assume the inputs to be consists, as they are applied to physical systems. However, in our case we apply a *mathematical* perturbation to a physical system, and the perturbed inputs are generally not consistent. Therefore the derivatives $\mathrm{D}_i\,\overline{ID}_b(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \dot{\mathbf{v}}, \dot{\mathbf{r}})$ with $i \in \{1, 2, 3, 4\}$ are generally non-zero.

Remark. The relation between the forward dynamics and inverse dynamics of (3.48) does not hold for the derivatives with respect to the joint torques vector $\boldsymbol{\tau}$. This is not necessary, since the expression for the input matrix $B(\eta, t)$ is already given by (3.29).

3.4 Summary

In this chapter we have presented and explored the Lie group and its details in Section 3.1. Based on this Lie group, the mathematical formulas of the forward dynamics linearization have been presented in Proposition 3.1. We have shown an expression for the computation of the input matrix $B(\eta, t)$ in Proposition 3.2, and we have expressed the state matrix $A(\eta, t)$ of the linearization of the forward dynamics in terms of the extended inverse dynamics derivatives and the inverse of the mass matrix in Proposition 3.3. This concludes the first objective: *Derive and present the mathematical formulas for the singularity-free geometric linearization of moving-base multibody systems*, as described in Section 1.3.

The following chapter presents numerically efficient algorithms that compute the quantities in the state and input matrices $A(\eta, t)$ given by (3.42) and $B(\eta, t)$ given by (3.29), namely the derivatives of the extended inverse dynamics and the inverse of the mass matrix.

Chapter 4

Left-Trivialized Linearization: Algorithmic Aspects

In the previous chapter we have shown the mathematical formulas of the left-trivialized linearization of moving-base dynamics in (3.42) and (3.29). However, we did not yet show how to compute the linearization in an efficient way. This chapter provides the details on efficient computation of the left-trivialized linearization through recursive analytical derivation. To compute the derivatives of the extended inverse dynamics given by (3.39), we first need to compute the extended inverse dynamics themselves. To do so, we propose a new algorithm: the Extended Inverse Dynamics Algorithm for moving-base systems (EIDAmb). We compute the derivatives of the extended inverse dynamics by using the method of recursive analytical derivation, as discussed in Section 1.2. Using this method, we propose four new algorithms that derive the EIDAmb step by step, by applying the chain rule. Each algorithm derives the EIDAmb with respect to one of the four state variables, as shown in Section 3.1. Furthermore we need to compute the inverse of the mass matrix. To do so, we could use the Composite Rigid Body Algorithm for moving-base systems (CRBAmb) and invert the obtained mass matrix. However, for fixed-base systems, the authors of [28] have shown that the inverse of the mass matrix can be computed more efficiently through their own algorithm: the Inverse Mass Matrix Algorithm for fixed-base systems (IMMA), which is based on the Articulated Body Algorithm for fixed-base systems (ABA). As we found that their algorithm contains some mistakes, we first propose a corrected version of their algorithm. By extending it, we propose a new algorithm: the Inverse Mass Matrix Algorithm for moving-base systems, based on the IMMA and the Articulated Body Algorithm for moving-base systems (ABAmb).

4.1 Extended Inverse Dynamics Algorithm

We aim to derive an algorithm that computes the extended inverse dynamics $\overline{ID}$ given by (3.39), which is repeated below for convenience of the reader. We recall that the extended inverse dynamics functions were given by

$$\overline{ID}(\mathbf{H},\mathbf{s},\mathbf{v},\mathbf{r},\dot{\mathbf{v}},\dot{\mathbf{r}}) := \bar{\boldsymbol{\tau}}(\mathbf{H},\mathbf{s},\mathbf{v},\mathbf{r},\dot{\mathbf{v}},\dot{\mathbf{r}}) = \mathbf{M}(\mathbf{s})\,\dot{\boldsymbol{\nu}}(\dot{\mathbf{v}},\dot{\mathbf{r}}) + \mathbf{h}(\mathbf{H},\mathbf{s},\mathbf{v},\mathbf{r}), \tag{3.39 revisited}$$

$$\overline{ID}_b(\mathbf{H},\mathbf{s},\mathbf{v},\mathbf{r},\dot{\mathbf{v}},\dot{\mathbf{r}}) := \bar{\boldsymbol{\tau}}_b(\mathbf{H},\mathbf{s},\mathbf{v},\mathbf{r},\dot{\mathbf{v}},\dot{\mathbf{r}}) = \mathbf{M}_{11}(\mathbf{s})\dot{\mathbf{v}} + \mathbf{M}_{12}(\mathbf{s})\dot{\mathbf{r}} + \mathbf{h}_1(\mathbf{H},\mathbf{s},\mathbf{v},\mathbf{r}) \tag{3.40 revisited}$$

and

$$\overline{ID}_j(\mathbf{H},\mathbf{s},\mathbf{v},\mathbf{r},\dot{\mathbf{v}},\dot{\mathbf{r}}) := \boldsymbol{\tau}(\mathbf{H},\mathbf{s},\mathbf{v},\mathbf{r},\dot{\mathbf{v}},\dot{\mathbf{r}}) \ = \mathbf{M}_{21}(\mathbf{s})\dot{\mathbf{v}} + \mathbf{M}_{22}(\mathbf{s})\dot{\mathbf{r}} + \mathbf{h}_2(\mathbf{H},\mathbf{s},\mathbf{v},\mathbf{r}). \tag{3.41 revisited}$$

To compute the extended inverse dynamics of the joints $\overline{ID}_j$ given by (3.41), we can make use of the Recursive Newton Euler Algorithm for moving-base systems (RNEAmb, presented in Appendix

E.2). The RNEAmb is a *hybrid* algorithm: it computes a combination of forward and inverse dynamics. It uses $\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}$ and $\dot{\mathbf{r}}$ as inputs to compute the moving-base acceleration $\dot{\mathbf{v}}$ and then the joint torques $\boldsymbol{\tau}$. Instead of requiring $\dot{\mathbf{v}}$ as an input, the RNEAmb computes it under the assumption $\bar{\boldsymbol{\tau}}_b = 0$, guaranteeing that the the variables $\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \dot{\mathbf{v}}, \dot{\mathbf{r}}$ are consistent (see Definition 3.1 for what we mean with consistency). The algorithm computes $\overline{ID}_j$ (3.41) but not $\overline{ID}_b$ (3.40), since $\overline{ID}_b$ is zero if its inputs are consistent. For our purpose of computing the left-trivialized linearization of both $\overline{ID}_b$ and $\overline{ID}_j$, which are both generally non-zero, we have designed the Extended Inverse Dynamics Algorithms for moving-base systems (EIDAmb), presented in Table 4.1 below. The EIDAmb consists of parts from three algorithms presented in [8]: the RNEAmb (which can be found in Appendix E.2, the Generalized Bias Wrench Algorithm for moving-base systems (GBWAmb, which can be found in Appendix E.7), and the Composite Rigid Body Algorithm for moving-base systems (CRBAmb, which can be found in Appendix E.4). The parts that origin from the GBWAmb are indicated with * and the parts that origin from the CRBAmb are indicated with ** in the EIDAmb in Table 4.1,

As discussed above, we need to obtain the extended inverse dynamics of the moving-base, which is defined in (3.40) as $\overline{ID}_b := \bar{\boldsymbol{\tau}}_b = \mathbf{M}_{11}\dot{\mathbf{v}} + \mathbf{M}_{12}\dot{\mathbf{r}} + \mathbf{h}_1$. The moving-base dynamics variables $\mathbf{M}_{11}, \mathbf{M}_{12}, \dot{\mathbf{v}}$ and $\mathbf{h}_1$ (detailed in Subsection 2.8.2 and found in the notation section (Section 2.7) in Tables 2.3 and 2.5) can be translated to rigid body dynamics variables used in recursive algorithms, found in the notation section in Table 2.7, as

$$\mathbf{M}_{11} = \mathbb{M}^c_{\mathcal{B}0}, \tag{4.1}$$

$$\dot{\mathbf{v}} = \mathbf{a}_0, \tag{4.2}$$

$$\mathbf{M}_{12} = {}_0\mathbf{F} \tag{4.3}$$

and

$$\mathbf{h}_1 = \mathbf{b}^{vp}_{\mathcal{B}0}, \tag{4.4}$$

where for (4.2), we made use of the relation between intrinsic and apparent acceleration, as defined in Appendix A.1. In terms of the EIDAmb variables, the extended inverse dynamics of the moving-base (3.40) can be written as

$$\overline{ID}_b(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \dot{\mathbf{v}}, \dot{\mathbf{r}}) = \bar{\boldsymbol{\tau}}_b(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \dot{\mathbf{v}}, \dot{\mathbf{r}}) = \mathbb{M}^c_{\mathcal{B}0}(\mathbf{s})\mathbf{a}_0(\dot{\mathbf{v}}) + {}_0\mathbf{F}(\mathbf{s})\dot{\mathbf{r}} + \mathbf{b}^{vp}_{\mathcal{B}0}(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \dot{\mathbf{v}}, \dot{\mathbf{r}}), \tag{4.5}$$

where $\mathbf{b}^{vp}_{\mathcal{B}0} \in \mathbb{R}^6$ is the bias wrench of body 0 with zero joint acceleration and ${}_0\mathbf{F} = [{}_0\mathbf{F}_{\mathcal{B}1}\ {}_0\mathbf{F}_{\mathcal{B}2}\ \dots\ {}_0\mathbf{F}_{\mathcal{B}n_J}] \in \mathbb{R}^{6\times n_J}$ the required wrenches to support unit accelerations, where ${}_0\mathbf{F}_{\mathcal{B}i}$ is defined as

$${}_i\mathbf{F}_{\mathcal{B}i} := \mathbb{M}^c_{\mathcal{B}i}\boldsymbol{\Gamma}_{\mathcal{J}i} \in \mathbb{R}^6, \tag{4.6}$$

$${}_0\mathbf{F}_{\mathcal{B}i} = {}_0\mathbf{X}^i{}_i\mathbf{F}_{\mathcal{B}i}. \tag{4.7}$$

In the EIDAmb, $\mathbf{a}^r_i$ is defined as

$$\mathbf{a}^r_i := {}^i\mathbf{a}_{0,i} + {}^i\mathbf{a}_{grav}, \tag{4.8}$$

where $\mathbf{a}^r_i$ is the acceleration of body i relative to the moving-base frame 0, plus the gravitational acceleration. It is the used to compute the bias wrench, which is later used to compute the acceleration of body i with respect to the inertial frame A, $\mathbf{a}_{A,i}$.

The table below presents the EIDAmb on the left, and for comparison it also presents the RNEAmb on the right. The two functions appearing in the these algorithms, 'get_gravity(model)' and 'jcalc(jtype(i),$\mathbf{s}_i$)', are explained in Appendices B.1 and B.2 respectively. We use $[x, y]$ to indicate submatrix extraction, where e.g. $X[1{:}2, 3]$ means we substract the elements in the first two rows and the third column of matrix X.

Table 4.1: EIDAmb (left) and for comparison RNEAmb (right).

Inputs	model, $\mathbf{s}, \mathbf{r}, \dot{\mathbf{r}}, {}^A\mathbf{H}_0, {}^A\mathbf{v}_{A,0}, {}^0\mathbf{a}_{A,0}$	model, $\mathbf{s}, \mathbf{r}, \dot{\mathbf{r}}, {}^A\mathbf{H}_0, {}^A\mathbf{v}_{A,0}$
Line	**EIDAmb**	**RNEAmb**
1	${}^A\mathbf{a}_{grav}$ = get_gravity(model)	${}^A\mathbf{a}_{grav}$ = get_gravity(model)
2	${}^0\mathbf{H}_A = {}^A\mathbf{H}_0^{-1}$	${}^0\mathbf{H}_A = {}^A\mathbf{H}_0^{-1}$
3	${}^0\mathbf{R}_A = {}^0\mathbf{H}_A[1{:}3, 1{:}3]$	${}^0\mathbf{R}_A = {}^0\mathbf{H}_A[1{:}3, 1{:}3]$
4	${}^0\mathbf{o}_A = {}^0\mathbf{H}_A[1{:}3, 4]$	${}^0\mathbf{o}_A = {}^0\mathbf{H}_A[1{:}3, 4]$
5	${}^0\mathbf{X}_A = \begin{bmatrix} {}^0\mathbf{R}_A & {}^0\mathbf{o}_A^\wedge {}^0\mathbf{R}_A \\ 0_{3\times3} & {}^0\mathbf{R}_A \end{bmatrix}$	${}^0\mathbf{X}_A = \begin{bmatrix} {}^0\mathbf{R}_A & {}^0\mathbf{o}_A^\wedge {}^0\mathbf{R}_A \\ 0_{3\times3} & {}^0\mathbf{R}_A \end{bmatrix}$
6	$\mathbf{v}_0 = {}^0\mathbf{X}_A {}^A\mathbf{v}_{A,0}$	$\mathbf{v}_0 = {}^0\mathbf{X}_A {}^A\mathbf{v}_{A,0}$
7	$\mathbf{a}_0^r = {}^0\mathbf{X}_A {}^A\mathbf{a}_{grav}$	$\mathbf{a}_0^r = {}^0\mathbf{X}_A {}^A\mathbf{a}_{grav}$
8*	$\mathbf{a}_0^{vp} = \mathbf{a}_0^r$	-
9	$\mathbb{M}_{\mathcal{B}0}^c = \mathbb{M}_{\mathcal{B}0}$	$\mathbb{M}_{\mathcal{B}0}^c = \mathbb{M}_{\mathcal{B}0}$
10	$\mathbf{m}_{\mathcal{B}0} = \mathbb{M}_{\mathcal{B}0}\mathbf{v}_0$	$\mathbf{m}_{\mathcal{B}0} = \mathbb{M}_{\mathcal{B}0}\mathbf{v}_0$
11	$\mathbf{b}_{\mathcal{B}0}^c = \mathbb{M}_{\mathcal{B}0}\mathbf{a}_0^r + \mathbf{v}_0 \bar{\times}^* \mathbf{m}_{\mathcal{B}0}$	$\mathbf{b}_{\mathcal{B}0}^c = \mathbb{M}_{\mathcal{B}0}\mathbf{a}_0^r + \mathbf{v}_0 \bar{\times}^* \mathbf{m}_{\mathcal{B}0}$
12*	$\mathbf{b}_{\mathcal{B}0}^{vp} = \mathbf{b}_{\mathcal{B}0}^c$	-
13	**for** $i = 1$ **to** n_B **do**	**for** $i = 1$ **to** n_B **do**
14	$[{}^i\mathbf{X}_{\lambda(i)\vert i}, \mathbf{\Gamma}_{\mathcal{J}i}]$ = jcalc(jtype(i), $\mathbf{s}_i$)	$[{}^i\mathbf{X}_{\lambda(i)\vert i}, \mathbf{\Gamma}_{\mathcal{J}i}]$ = jcalc(jtype(i), $\mathbf{s}_i$)
15	$\mathbf{v}_{\mathcal{J}i} = \mathbf{\Gamma}_{\mathcal{J}i}\mathbf{r}_i$	$\mathbf{v}_{\mathcal{J}i} = \mathbf{\Gamma}_{\mathcal{J}i}\mathbf{r}_i$
16	${}^i\mathbf{X}_{\lambda(i)} = {}^i\mathbf{X}_{\lambda(i)\vert i} {}^{\lambda(i)\vert i}\mathbf{X}_{\lambda(i)}$	${}^i\mathbf{X}_{\lambda(i)} = {}^i\mathbf{X}_{\lambda(i)\vert i} {}^{\lambda(i)\vert i}\mathbf{X}_{\lambda(i)}$
17	$\mathbf{v}_i = {}^i\mathbf{X}_{\lambda(i)}\mathbf{v}_{\lambda(i)} + \mathbf{v}_{\mathcal{J}i}$	$\mathbf{v}_i = {}^i\mathbf{X}_{\lambda(i)}\mathbf{v}_{\lambda(i)} + \mathbf{v}_{\mathcal{J}i}$
18	$\mathbf{a}_i^r = {}^i\mathbf{X}_{\lambda(i)}\mathbf{a}_{\lambda(i)}^r + \mathbf{\Gamma}_{\mathcal{J}i}\dot{\mathbf{r}}_i + \mathbf{v}_i \times \mathbf{v}_{\mathcal{J}i}$	$\mathbf{a}_i^r = {}^i\mathbf{X}_{\lambda(i)}\mathbf{a}_{\lambda(i)}^r + \mathbf{\Gamma}_{\mathcal{J}i}\dot{\mathbf{r}}_i + \mathbf{v}_i \times \mathbf{v}_{\mathcal{J}i}$
19*	$\mathbf{a}_i^{vp} = {}^i\mathbf{X}_{\lambda(i)}\mathbf{a}_{\lambda(i)}^{vp} + \mathbf{v}_i \times \mathbf{v}_{\mathcal{J}i}$	-
20	$\mathbb{M}_{\mathcal{B}i}^c = \mathbb{M}_{\mathcal{B}i}$	$\mathbb{M}_{\mathcal{B}i}^c = \mathbb{M}_{\mathcal{B}i}$
21	$\mathbf{m}_{\mathcal{B}i} = \mathbb{M}_{\mathcal{B}i}\mathbf{v}_i$	$\mathbf{m}_{\mathcal{B}i} = \mathbb{M}_{\mathcal{B}i}\mathbf{v}_i$
22	$\mathbf{b}_{\mathcal{B}i}^c = \mathbb{M}_{\mathcal{B}i}\mathbf{a}_i^r + \mathbf{v}_i \bar{\times}^* \mathbf{m}_{\mathcal{B}i}$	$\mathbf{b}_{\mathcal{B}i}^c = \mathbb{M}_{\mathcal{B}i}\mathbf{a}_i^r + \mathbf{v}_i \bar{\times}^* \mathbf{m}_{\mathcal{B}i}$
23*	$\mathbf{b}_{\mathcal{B}i}^{vp} = \mathbb{M}_{\mathcal{B}i}\mathbf{a}_i^{vp} + \mathbf{v}_i \bar{\times}^* \mathbf{m}_{\mathcal{B}i}$	-
24	**end**	**end**
25	**for** $i = n_B$ **to** 1 **do**	**for** $i = n_B$ **to** 1 **do**
26	$\mathbb{M}_{\mathcal{B}\lambda(i)}^c = \mathbb{M}_{\mathcal{B}\lambda(i)}^c + {}_{\lambda(i)}\mathbf{X}^i\ \mathbb{M}_{\mathcal{B}i}^c {}^i\mathbf{X}_{\lambda(i)}$	$\mathbb{M}_{\mathcal{B}\lambda(i)}^c = \mathbb{M}_{\mathcal{B}\lambda(i)}^c + {}_{\lambda(i)}\mathbf{X}^i\ \mathbb{M}_{\mathcal{B}i}^c {}^i\mathbf{X}_{\lambda(i)}$
27	$\mathbf{b}_{\mathcal{B}\lambda(i)}^c = \mathbf{b}_{\mathcal{B}\lambda(i)}^c + {}_{\lambda(i)}\mathbf{X}^i\ \mathbf{b}_{\mathcal{B}i}^c$	$\mathbf{b}_{\mathcal{B}\lambda(i)}^c = \mathbf{b}_{\mathcal{B}\lambda(i)}^c + {}_{\lambda(i)}\mathbf{X}^i\ \mathbf{b}_{\mathcal{B}i}^c$
28*	$\mathbf{b}_{\mathcal{B}\lambda(i)}^{vp} = \mathbf{b}_{\mathcal{B}\lambda(i)}^{vp} + {}_{\lambda(i)}\mathbf{X}^i\ \mathbf{b}_{\mathcal{B}i}^{vp}$	-
29	**end**	**end**

continued on the next page.

-	-	$\mathbf{a}_0 = -(\mathbb{M}^c_{\mathcal{B}0})^{-1}\mathbf{b}^c_{\mathcal{B}0}$
30	**for** $i = 1$ **to** n_B **do**	**for** $i = 1$ **to** n_B **do**
31	${}^i\mathbf{a}_{A,0} = {}^i\mathbf{X}_{\lambda(i)}{}^{\lambda(i)}\mathbf{a}_{A,0}$	${}^i\mathbf{a}_{A,0} = {}^i\mathbf{X}_{\lambda(i)}{}^{\lambda(i)}\mathbf{a}_{A,0}$
32	$\boldsymbol{\tau}_i = \boldsymbol{\Gamma}^T_{\mathcal{J}i}(\mathbb{M}^c_{\mathcal{B}i}{}^i\mathbf{a}_{A,0} + \mathbf{b}^c_{\mathcal{B}i})$	$\boldsymbol{\tau}_i = \boldsymbol{\Gamma}^T_{\mathcal{J}i}(\mathbb{M}^c_{\mathcal{B}i}{}^i\mathbf{a}_{A,0} + \mathbf{b}^c_{\mathcal{B}i})$
33**	${}_i\mathbf{F}_{\mathcal{B}i} = \mathbb{M}^c_{\mathcal{B}i}\boldsymbol{\Gamma}_{\mathcal{J}i}$	-
34**	$j = i$	-
35**	**while** $\lambda(j) > 0$	-
36**	${}_{\lambda(j)}\mathbf{F}_{\mathcal{B}i} = {}_{\lambda(j)}\mathbf{X}^j{}_j\mathbf{F}_{\mathcal{B}i}$	-
37**	$j = \lambda(j)$	-
38**	**end**	-
39**	${}_0\mathbf{F}_{\mathcal{B}i} = {}_0\mathbf{X}^j{}_j\mathbf{F}_{\mathcal{B}i}$	-
40	**end**	**end**
41***	$\overline{ID}_b = \mathbb{M}^c_{\mathcal{B}0}{}^0\mathbf{a}_{A,0} + {}_0\mathbf{F}\dot{\mathbf{r}} + \mathbf{b}^{vp}_{\mathcal{B}0}$	-
-	-	${}^A\mathbf{a}_{A,0} = {}^0\mathbf{X}_A\mathbf{a}_0$
Outputs	$\overline{ID}_j(=\boldsymbol{\tau}), \overline{ID}_b(=\bar{\boldsymbol{\tau}}_b)$	$\boldsymbol{\tau}, {}^A\mathbf{a}_{A,0}$

Remarks:

- Lines marked with * origin from the GBWAmb, lines marked with ** origin from the CRBAmb, and unmarked lines origin from the RNEAmb.
- The line marked with *** computes the extended inverse dynamics of the moving base in (4.5).
- Arbitrary inputs are generally not consistent (see Definition 3.1). If one wants the inputs of the extended inverse dynamics function (3.39) to be consistent, one can add the computation $\mathbf{a}_0 = -(\mathbb{M}^c_{\mathcal{B}0})^{-1}\mathbf{b}^c_{\mathcal{B}0}$ between line 29 and 30 instead of using it as an input to the algorithm. This guarantees the inputs of the extended inverse dynamics function to be consistent, at the cost of losing the freedom to choose $\mathbf{a}_0$.
- External wrenches are neglected in the algorithms presented above, as assumed in Assumption 2.3.
- The velocity of the moving base as input of the EIDAmb, ${}^A\mathbf{v}_{A,0}$ is chosen to be expressed in frame A, as it is often measured in that frame. The acceleration of the moving base as input of the EIDAmb, ${}^0\mathbf{a}_{A,0}$, is chosen to be expressed in frame 0, as it is often computed through the RNEAmb in frame 0. If one desired to express either of both quantities in a different frame, the algorithm can be easily modified.

4.2 Computing the derivatives of the extended inverse dynamics

In the previous section we have presented the EIDAmb, which computes the extended inverse dynamics given by (3.39). This section proposes four new recursive algorithms that compute the left-trivialized derivatives of the extended inverse dynamics (3.39) with respect to $\mathbf{H}, \mathbf{s}, \mathbf{v}$ and $\mathbf{r}$, which are necessary to compute the state matrix (3.42). The algorithms that compute the left-trivialized derivatives become bulky and unclear when fully written out for each variable. To improve readability, we define a shortened notation for these derivatives. The left-trivialized derivative of an arbitrary function $x(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r})$ with respect to $\mathbf{H}$ is written in shortened notation as

$$\frac{\tilde{\partial} x}{\partial \mathbf{H}} := \mathrm{D}_1\, x(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}) \circ \mathrm{D}\, L_H(I), \tag{4.9}$$

so that $\tilde{\partial} x/\partial y$ is defined as the *left-trivialized derivative of x with respect to Lie group element y.* Likewise, the partial derivatives (on vector spaces) of an arbitrary function $x(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r})$ with respect to $\mathbf{s}, \mathbf{v}$ and $\mathbf{r}$ are written in shortened notation as

$$\frac{\partial x}{\partial \mathbf{s}} := \mathrm{D}_2\, x(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}), \tag{4.10}$$

$$\frac{\partial x}{\partial \mathbf{v}} := \mathrm{D}_3\, x(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}) \tag{4.11}$$

and

$$\frac{\partial x}{\partial \mathbf{r}} := \mathrm{D}_4\, x(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}), \tag{4.12}$$

so that $\partial x/\partial y$ is defined as the *partial derivative of x with respect to scalar or vector y.*

In the left-trivialized derivatives algorithms we compute the (left-trivialized) derivatives of scalars, vectors, and matrices. The derivatives of scalars and vectors result in 1D or 2D tensors to which the standard definition of matrix multiplication can be applied. However the derivatives of matrices result in 3D tensors, for which there is no standard definition of multiplication. In the EIDAmb, there are three matrices of which we need to compute their derivatives: ${}^{i}\mathbf{X}_{\lambda(i)|i}, \mathbf{F}_{\mathcal{B}i}$ and $\mathbb{M}^{c}_{\mathcal{B}\lambda(i)}$. All three matrices depend only on the generalized position vector $\mathbf{s}$. Instead of expressing the derivatives with respect to $\mathbf{s}$ as 3D tensors, we choose to express the derivatives with respect to the scalars $\mathbf{s}_k$ for $k \in \mathbb{R}^{n_B}$, resulting in 2D tensors to which standard definition of matrix multiplication can be applied. The matrix ${}^{i}\mathbf{X}_{\lambda(i)|i}$ is only dependent on the i-th entry of $\mathbf{s}$. Therefore we only compute $\partial^{i}\mathbf{X}_{\lambda(i)|i}/\partial \mathbf{s}_i$, which saves computational time.

The extended inverse dynamics of the joints $\overline{ID}_j$ of (3.41) is equal to the joint torques $\boldsymbol{\tau}$, as defined in (3.41), meaning that the left-trivialized tangent maps of $\overline{ID}_j$ with respect to $\mathbf{H}, \mathbf{s}, \mathbf{v}$ and $\mathbf{r}$ are computed as

$$\mathrm{D}_1\, \overline{ID}_j(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \dot{\mathbf{v}}, \dot{\mathbf{r}}) \circ \mathrm{D}\, L_H(I) = \frac{\tilde{\partial} \boldsymbol{\tau}}{\partial \mathbf{H}}, \tag{4.13}$$

$$\mathrm{D}_2\, \overline{ID}_j(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \dot{\mathbf{v}}, \dot{\mathbf{r}}) = \frac{\partial \boldsymbol{\tau}}{\partial \mathbf{s}}, \tag{4.14}$$

$$\mathrm{D}_3\, \overline{ID}_j(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \dot{\mathbf{v}}, \dot{\mathbf{r}}) = \frac{\partial \boldsymbol{\tau}}{\partial \mathbf{v}} \tag{4.15}$$

and

$$\mathrm{D}_4\, \overline{ID}_j(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \dot{\mathbf{v}}, \dot{\mathbf{r}}) = \frac{\partial \boldsymbol{\tau}}{\partial \mathbf{r}}. \tag{4.16}$$

The extended inverse dynamics of the moving base $\overline{ID}_b$ of (4.5) is given by $\overline{ID}_b = \bar{\boldsymbol{\tau}}_b = \mathbb{M}^{c}_{\mathcal{B}0}\mathbf{a}_0 + {}_0\mathbf{F}\,\dot{\mathbf{r}} + \mathbf{b}^{vp}_{\mathcal{B}0}$, meaning that the left-trivialized tangent maps of the extended inverse dynamics of $\overline{ID}_b$ with respect to $\mathbf{H}, \mathbf{s}, \mathbf{v}$ and $\mathbf{r}$ are computed as

$$\mathrm{D}_1\, \overline{ID}_b(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \dot{\mathbf{v}}, \dot{\mathbf{r}}) \circ \mathrm{D}\, L_H(I) = \frac{\tilde{\partial} \mathbf{b}^{vp}_{\mathcal{B}0}}{\partial \mathbf{H}}, \tag{4.17}$$

$$\mathrm{D}_2\, \overline{ID}_b(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \dot{\mathbf{v}}, \dot{\mathbf{r}}) = \frac{\partial \mathbb{M}^{c}_{\mathcal{B}0}}{\partial \mathbf{s}}\mathbf{a}_0 + \frac{\partial {}_0\mathbf{F}}{\partial \mathbf{s}}\dot{\mathbf{r}} + \frac{\partial \mathbf{b}^{vp}_{\mathcal{B}0}}{\partial \mathbf{s}}, \tag{4.18}$$

$$\mathrm{D}_3\, \overline{ID}_b(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \dot{\mathbf{v}}, \dot{\mathbf{r}}) = \frac{\partial \mathbf{b}^{vp}_{\mathcal{B}0}}{\partial \mathbf{v}} \tag{4.19}$$

and

$$\mathrm{D}_4\,\overline{ID}_b(\mathbf{H},\mathbf{s},\mathbf{v},\mathbf{r},\dot{\mathbf{v}},\dot{\mathbf{r}}) = \frac{\partial \mathbf{b}^{vp}_{\mathcal{B}0}}{\partial \mathbf{r}}. \tag{4.20}$$

Here we used $\partial\dot{\mathbf{v}}/\partial x = 0$ and $\partial\dot{\mathbf{r}}/\partial x = 0$ for $x \in \{\mathbf{H},\mathbf{s},\mathbf{v},\mathbf{r}\}$ as $\dot{\mathbf{v}}$ and $\dot{\mathbf{r}}$ are input variables for the extended inverse dynamics, and $\partial_i\mathbf{F}_{\mathcal{B}i}/\partial\mathbf{s} = \partial\mathbb{M}^c_{\mathcal{B}i}/\partial\mathbf{s}\;\boldsymbol{\Gamma}_{\mathcal{J}i}$ where ${}_i\mathbf{F}_{\mathcal{B}i}$ is defined in (4.6) and $i \in n_B$ is the body number. Here we use the fact that $\boldsymbol{\Gamma}_{\mathcal{J}i}$ is independent of $\mathbf{s}$ for *conventional* types of joints (see Assumption 2.2 for what we mean with conventional and Appendix B.2 for the values of $\boldsymbol{\Gamma}_{\mathcal{J}i}$ for each joint type).

4.2.1 Computing the derivatives of the extended inverse dynamics w.r.t. the transformation matrix

The derivative of the extended inverse dynamics of (3.39) with respect to the transformation matrix $\mathbf{H}$ is the only left-trivialized derivative, which makes this algorithm the most complicated of all four. This section first shows the left-trivialized derivative of the velocity transformation matrix ${}^0\mathbf{X}_A$ combined with a velocity or acceleration vector. After that, we present a mathematical proposition which allows us to increase numerical efficiency by skipping unnecessary computations. Finally, the algorithm that computes the left-trivialized derivative of the EIDAmb with respect to $\mathbf{H}$ is presented.

The left-trivialized derivative of ${}^0\mathbf{X}_A$ with respect to the transformation matrix is the most complicated part of the algorithm that computes the left-trivialized derivatives of the extended inverse dynamics with respect to the transformation matrix ${}^A\mathbf{H}_0$. We choose to compute the quantities ${}^0\mathbf{X}_A{}^A\mathbf{v}_{A,0}$ and ${}^0\mathbf{X}_A{}^A\mathbf{a}_{grav}$ as a whole, as this allows us to compute the derivative of a vector instead of the derivative of a matrix. This is explained in the following proposition.

Proposition 4.1: The left-trivialized derivatives of ${}^0\mathbf{X}_A{}^A\mathbf{v}_{A,0}$ and ${}^0\mathbf{X}_A{}^A\mathbf{a}_{grav}$ with respect to the transformation matrix ${}^A\mathbf{H}_0$ are given by

$$\frac{\tilde{\partial}({}^0\mathbf{X}_A{}^A\mathbf{v}_{A,0})}{\partial{}^A\mathbf{H}_0} = \begin{bmatrix} {}^0\mathbf{R}_A{}^A\boldsymbol{\omega}^\wedge_{A,0}{}^0\mathbf{R}_A & {}^0\mathbf{R}_A{}^A\boldsymbol{v}^\wedge_{A,0}{}^0\mathbf{R}^T_A + ({}^0\mathbf{o}^\wedge_A{}^0\mathbf{R}_A{}^A\boldsymbol{\omega}_{A,0})^\wedge \\ 0_{3\times3} & {}^0\mathbf{R}_A{}^A\boldsymbol{\omega}^\wedge_{A,0}{}^0\mathbf{R}^T_A \end{bmatrix} \tag{4.21}$$

and

$$\frac{\tilde{\partial}({}^0\mathbf{X}_A{}^A\mathbf{a}_{grav})}{\partial{}^A\mathbf{H}_0} = \begin{bmatrix} {}^0\mathbf{R}_A{}^A\boldsymbol{\alpha}^\wedge_{grav}{}^0\mathbf{R}_A & {}^0\mathbf{R}_A{}^A\boldsymbol{a}^\wedge_{grav}{}^0\mathbf{R}^T_A + ({}^0\mathbf{o}^\wedge_A{}^0\mathbf{R}_A{}^A\boldsymbol{\alpha}_{grav})^\wedge \\ 0_{3\times3} & {}^0\mathbf{R}_A{}^A\boldsymbol{\alpha}^\wedge_{grav}{}^0\mathbf{R}^T_A \end{bmatrix}, \tag{4.22}$$

where ${}^A\mathbf{v}_{A,0} = [{}^A\boldsymbol{v}_{A,0}; {}^A\boldsymbol{\omega}_{A,0}]$ and ${}^A\mathbf{a}_{grav} = [{}^A\boldsymbol{a}_{grav}; {}^A\boldsymbol{\alpha}_{grav}]$. Here $\boldsymbol{v} \in \mathbb{R}^3$ represents translational velocity, $\boldsymbol{\omega} \in \mathbb{R}^3$ rotational velocity, $\boldsymbol{a} \in \mathbb{R}^3$ translational acceleration and $\boldsymbol{\alpha} \in \mathbb{R}^3$ rotational acceleration. Note that ${}^A\mathbf{v}_{A,0}$ and ${}^A\mathbf{a}_{grav}$ are independent of ${}^A\mathbf{H}_0$, as they are inputs to the algorithm.
■

Proof: We only prove (4.21) here, (4.22) can be proved in a similar manner. Let us first define $\mathbf{H} := {}^A\mathbf{H}_0$, $\mathbf{R} := {}^A\mathbf{R}_0$ and $\mathbf{o} := {}^A\mathbf{o}_0$. We can write ${}^0\mathbf{X}_A{}^A\mathbf{v}_{A,0}$ as

$$\begin{aligned} {}^0\mathbf{X}_A{}^A\mathbf{v}_{A,0} &= \begin{bmatrix} {}^0\mathbf{R}_A & {}^0\mathbf{o}^\wedge_A{}^0\mathbf{R}_A \\ 0_{3\times3} & {}^0\mathbf{R}_A \end{bmatrix} \begin{bmatrix} {}^A\boldsymbol{v}_{A,0} \\ {}^A\boldsymbol{\omega}_{A.0} \end{bmatrix} \\ &= \begin{bmatrix} {}^A\mathbf{R}^T_0 & -{}^A\mathbf{R}^{T}_0{}^A\mathbf{o}^\wedge_0 \\ 0_{3\times3} & {}^A\mathbf{R}^T_0 \end{bmatrix} \begin{bmatrix} {}^A\boldsymbol{v}_{A,0} \\ {}^A\boldsymbol{\omega}_{A.0} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{R}^T & -\mathbf{R}^T\mathbf{o}^\wedge \\ 0_{3\times3} & \mathbf{R}^T \end{bmatrix} \begin{bmatrix} {}^A\boldsymbol{v}_{A,0} \\ {}^A\boldsymbol{\omega}_{A.0} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{R}^{T\,A}\boldsymbol{v}_{A,0} - \mathbf{R}^T\mathbf{o}^{\wedge\,A}\boldsymbol{\omega}_{A.0} \\ \mathbf{R}^{T\,A}\boldsymbol{\omega}_{A.0}, \end{bmatrix} \end{aligned} \tag{4.23}$$

where we use the equality

$$ {}^{0}\mathbf{o}_{A}^{\wedge} = -{}^{0}\mathbf{R}_{A}{}^{A}\mathbf{o}_{0}^{\wedge}{}^{A}\mathbf{R}_{0}. \tag{4.24}$$

We define the vector valued functions $f_1 : \mathbb{R}^3 \times \mathrm{SO}(3) \to \mathbb{R}^3$ and $f_2 : \mathrm{SO}(3) \to \mathbb{R}^3$ as

$$ f_1(\mathbf{o}, \mathbf{R}) := \mathbf{R}^{T A}\boldsymbol{v}_{A,0} - \mathbf{R}^{T}\mathbf{o}^{\wedge A}\boldsymbol{\omega}_{A,0} \tag{4.25}$$

and

$$ f_2(\mathbf{R}) := \mathbf{R}^{T A}\boldsymbol{\omega}_{A,0}, \tag{4.26}$$

so that

$$ {}^{0}\mathbf{X}_{A}{}^{A}\mathbf{v}_{A,0} = \begin{bmatrix} f_1(\mathbf{o},\mathbf{R}) \\ f_2(\mathbf{R}) \end{bmatrix}. \tag{4.27}$$

Using the expression (4.27), we compute the left-trivialized derivative of ${}^{0}\mathbf{X}_{A}{}^{A}\mathbf{v}_{A,0}$ given by (4.23) with respect to $\mathbf{H}$ as

$$ \frac{\tilde{\partial}({}^{0}\mathbf{X}_{A}{}^{A}\mathbf{v}_{A,0})}{\partial \mathbf{H}} = \begin{bmatrix} \frac{\partial({}^{0}\mathbf{X}_{A}{}^{A}\mathbf{v}_{A,0})}{\partial \mathbf{R}\mathbf{o}} & \frac{\tilde{\partial}({}^{0}\mathbf{X}_{A}{}^{A}\mathbf{v}_{A,0})}{\partial \mathbf{R}} \end{bmatrix}, \tag{4.28}$$

where the left-hand side is a normal partial derivative which accounts for the translation, and the right-hand side is a left-trivialized partial derivative which accounts for the rotation. The left-hand side derivative is taken with respect to $\mathbf{R}\mathbf{o} \in \mathbb{R}^3$, which may seem surprising as one could expect it to be taken with respect to $\mathbf{o} \in \mathbb{R}^3$. If we had modelled the moving-base on $\mathbb{R}^3 \times \mathrm{SO}(3)$, then we would indeed need to take the derivative with respect to $\mathbf{o}$. This is due to the group operation of $\mathbb{R}^3 \times \mathrm{SO}(3)$, which is

$$ (\mathbf{o}_1, \mathbf{R}_1) \cdot (\mathbf{o}_2 \mathbf{R}_2) = (\mathbf{o}_1 + \mathbf{o}_2, \mathbf{R}_1\mathbf{R}_2), \tag{4.29}$$

where $(\mathbf{o}_1, \mathbf{R}_1), (\mathbf{o}_2, \mathbf{R}_2) \in \mathbb{R}^3 \times \mathrm{SO}(3)$. However, we model the moving-base on SE(3), which has the group operation

$$ \mathbf{H}_1 \cdot \mathbf{H}_2 = \mathbf{H}_1\mathbf{H}_2, \tag{4.30}$$

which can also be written as

$$ \begin{bmatrix} \mathbf{R}_1 & \mathbf{o}_1 \\ 0_{1\times3} & 1 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{R}_2 & \mathbf{o}_2 \\ 0_{1\times3} & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R}_1\mathbf{R}_2 & \mathbf{R}_1\mathbf{o}_2 + \mathbf{o}_1 \\ 0_{1\times3} & 1 \end{bmatrix}, \tag{4.31}$$

where $\mathbf{H}_1, \mathbf{H}_2 \in \mathrm{SE}(3)$. It is visible that the group operation of SE(3) does not simply add two translations $\mathbf{o}_1$ and $\mathbf{o}_2$, but $\mathbf{o}_2$ is pre-multiplied with $\mathbf{R}_1$. For this reason we need to take the derivative with respect to $\mathbf{R}\mathbf{o}$ in (4.28). The left elements of (4.28), $\frac{\partial({}^{0}\mathbf{X}_{A}{}^{A}\mathbf{v}_{A,0})}{\partial \mathbf{R}\mathbf{o}}$, are split us as

$$ \frac{\partial({}^{0}\mathbf{X}_{A}{}^{A}\mathbf{v}_{A,0})}{\partial \mathbf{R}\mathbf{o}} = \begin{bmatrix} \frac{\partial f_1(\mathbf{o},\mathbf{R})}{\partial \mathbf{R}\mathbf{o}} \\ \frac{\partial f_2(\mathbf{R})}{\partial \mathbf{R}\mathbf{o}} \end{bmatrix}. \tag{4.32}$$

As $f_2(\mathbf{R})$ is independent of $\mathbf{o}$, its derivatives with respect to $\mathbf{R}\mathbf{o}$ is equal to zero. The derivative of

f_1 with respect to $\mathbf{Ro}$ is computed as the normal partial derivative

$$\begin{aligned}
\frac{\partial f_1(\mathbf{o},\mathbf{R})}{\partial \mathbf{Ro}} &= \frac{\partial \mathbf{R}^{T\,A}\boldsymbol{v}_{A,0} - \mathbf{R}^T\mathbf{o}^{\wedge\,A}\boldsymbol{\omega}_{A.0}}{\partial \mathbf{Ro}} \\
&= \frac{\partial - \mathbf{R}^T\mathbf{o}^{\wedge\,A}\boldsymbol{\omega}_{A,0}}{\partial \mathbf{Ro}} \\
&= \frac{\partial \mathbf{R}^{T\,A}\boldsymbol{\omega}^{\wedge}_{A,0}\mathbf{o}}{\partial \mathbf{Ro}} \\
&= \frac{\partial \mathbf{R}^{T\,A}\boldsymbol{\omega}^{\wedge}_{A,0}\mathbf{R}^T\mathbf{Ro}}{\partial \mathbf{Ro}} \\
&= \mathbf{R}^{T\,A}\boldsymbol{\omega}^{\wedge}_{A,0}\mathbf{R}^T \\
&= {}^0\mathbf{R}_A{}^A\boldsymbol{\omega}^{\wedge}_{A,0}{}^0\mathbf{R}_A, \qquad (4.33)
\end{aligned}$$

where we use the equality

$$a^\wedge b = -b^\wedge a, \qquad (4.34)$$

with $a, b \in \mathbb{R}^3$. The right elements of (4.28), $\dfrac{\tilde{\partial}({}^0\mathbf{X}_A{}^A\mathbf{v}_{A,0})}{\partial \mathbf{R}}$, can be split up as

$$\frac{\tilde{\partial}({}^0\mathbf{X}_A{}^A\mathbf{v}_{A,0})}{\partial \mathbf{R}} = \begin{bmatrix} \mathrm{D}_2 f_1(\mathbf{o},\mathbf{R}) \circ \mathrm{D}\, L_{\mathbf{R}}(I) \\ \mathrm{D}\, f_2(\mathbf{R}) \circ \mathrm{D}\, L_{\mathbf{R}}(I) \end{bmatrix}. \qquad (4.35)$$

The upper elements of (4.35), $\mathrm{D}_2 f_1(\mathbf{o},\mathbf{R}) \circ \mathrm{D}\, L_{\mathbf{R}}(I)$ are computed in a direction $z \in \mathbb{R}^3$ as a left trivialized derivative

$$\begin{aligned}
\mathrm{D}_2 f_1(\mathbf{o},\mathbf{R}) \circ \mathrm{D}\, L_{\mathbf{R}}(I)\cdot z &= \lim_{\varepsilon\to 0}\frac{1}{\varepsilon}\big[f_1(\mathbf{o},\mathbf{R}\exp(\varepsilon z^\wedge)) - f_1(\mathbf{o},\mathbf{R})\big] \\
&= \lim_{\varepsilon\to 0}\frac{1}{\varepsilon}\big[f_1(\mathbf{o},\mathbf{R} + \varepsilon\mathbf{R}z^\wedge + o(\varepsilon^2)) - f_1(\mathbf{o},\mathbf{R})\big] \\
&= \lim_{\varepsilon\to 0}\frac{1}{\varepsilon}\big[\big(\mathbf{R} + \varepsilon\mathbf{R}z^\wedge + o(\varepsilon^2)\big)^{T\,A}\boldsymbol{v}_{A,0} - \big(\mathbf{R} + \varepsilon\mathbf{R}z^\wedge + o(\varepsilon^2)\big)^T\mathbf{o}^{\wedge\,A}\boldsymbol{\omega}_{A,0} \\
&\qquad - \big(\mathbf{R}^{T\,A}\boldsymbol{v}_{A,0} - \mathbf{R}^T\mathbf{o}^{\wedge\,A}\boldsymbol{\omega}_{A,0}\big)\big] \\
&= \lim_{\varepsilon\to 0}\frac{1}{\varepsilon}\big[\big(\mathbf{R}^T - \varepsilon z^\wedge\mathbf{R}^T + o(\varepsilon^2)\big)^A\boldsymbol{v}_{A,0} - \big(\mathbf{R}^T - \varepsilon z^\wedge\mathbf{R}^T + o(\varepsilon^2)\big)\mathbf{o}^{\wedge\,A}\boldsymbol{\omega}_{A,0} \\
&\qquad - \big(\mathbf{R}^{T\,A}\boldsymbol{v}_{A,0} - \mathbf{R}^T\mathbf{o}^{\wedge\,A}\boldsymbol{\omega}_{A,0}\big)\big] \\
&= \lim_{\varepsilon\to 0}\frac{1}{\varepsilon}\big[-\varepsilon z^\wedge\mathbf{R}^{T\,A}\boldsymbol{v}_{A,0} + \varepsilon z^\wedge\mathbf{R}^T\mathbf{o}^{\wedge\,A}\boldsymbol{\omega}_{A,0} + o(\varepsilon^2)\big] \\
&= -z^\wedge\mathbf{R}^{T\,A}\boldsymbol{v}_{A,0} + z^\wedge\mathbf{R}^T\mathbf{o}^{\wedge\,A}\boldsymbol{\omega}_{A,0} \\
&= (\mathbf{R}^{T\,A}\boldsymbol{v}_{A,0})^\wedge z - (\mathbf{R}^T\mathbf{o}^{\wedge\,A}\boldsymbol{\omega}_{A,0})^\wedge z \\
&= \mathbf{R}^{T\,A}\boldsymbol{v}^{\wedge}_{A,0}\mathbf{R}z - ({}^0\mathbf{R}_A{}^A\mathbf{o}^{\wedge}_0{}^A\boldsymbol{\omega}_{A,0})^\wedge z \\
&= {}^0\mathbf{R}_A{}^A\boldsymbol{v}^{\wedge}_{A,0}{}^0\mathbf{R}^T_A z + ({}^0\mathbf{R}_A{}^A\mathbf{R}_0{}^0\mathbf{o}^{\wedge}_A{}^0\mathbf{R}_A{}^A\boldsymbol{\omega}_{A,0})^\wedge z \\
&= {}^0\mathbf{R}_A{}^A\boldsymbol{v}^{\wedge}_{A,0}{}^0\mathbf{R}^T_A z + ({}^0\mathbf{o}^{\wedge}_A{}^0\mathbf{R}_A{}^A\boldsymbol{\omega}_{A,0})^\wedge z, \qquad (4.36)
\end{aligned}$$

where we used the exponential map of SE(3) of (2.39) and the equalities (4.24), (4.34) and

$$(\mathbf{R}\boldsymbol{\omega})^\wedge = \mathbf{R}\boldsymbol{\omega}^\wedge\mathbf{R}^T. \qquad (4.37)$$

We write (4.36) as

$$\mathrm{D}_2 f_1(\mathbf{o},\mathbf{R}) \circ \mathrm{D}\, L_{\mathbf{R}}(I) = {}^0\mathbf{R}_A{}^A\boldsymbol{v}^{\wedge}_{A,0}{}^0\mathbf{R}^T_A + ({}^0\mathbf{o}^{\wedge}_A{}^0\mathbf{R}_A{}^A\boldsymbol{\omega}_{A,0})^\wedge. \qquad (4.38)$$

The lower elements of (4.35), $\mathrm{D}\, f_2(\mathbf{R}) \circ \mathrm{D}\, L_{\mathbf{R}}(I)$ are computed in a direction $z \in \mathbb{R}^3$ as a left-trivialized derivative

$$\begin{aligned}
\mathrm{D}\, f_2(\mathbf{R}) \circ \mathrm{D}\, L_{\mathbf{R}}(I) \cdot z &= \lim_{\varepsilon \to 0} \frac{1}{\varepsilon} \big[f_2(\mathbf{R} \exp(\varepsilon z^\wedge)) - f_2(\mathbf{R}) \big] \\
&= \lim_{\varepsilon \to 0} \frac{1}{\varepsilon} \big[f_2(\mathbf{R} + \varepsilon \mathbf{R} z^\wedge + o(\varepsilon^2)) - f_2(\mathbf{R}) \big] \\
&= \lim_{\varepsilon \to 0} \frac{1}{\varepsilon} \big[\big(\mathbf{R} + \varepsilon \mathbf{R} z^\wedge + o(\varepsilon^2)\big)^{TA} \boldsymbol{\omega}_{A,0} - \mathbf{R}^{TA} \boldsymbol{\omega}_{A,0} \big] \\
&= \lim_{\varepsilon \to 0} \frac{1}{\varepsilon} \big[\mathbf{R}^{TA} \boldsymbol{\omega}_{A,0} - \varepsilon z^\wedge \mathbf{R}^{TA} \boldsymbol{\omega}_{A,0} + o(\varepsilon^2) - \mathbf{R}^{TA} \boldsymbol{\omega}_{A,0} \big] \\
&= \lim_{\varepsilon \to 0} \frac{1}{\varepsilon} \big[-\varepsilon z^\wedge \mathbf{R}^{TA} \boldsymbol{\omega}_{A,0} + o(\varepsilon^2) \big] \\
&= -z^\wedge \mathbf{R}^{TA} \boldsymbol{\omega}_{A,0} \\
&= (\mathbf{R}^{TA} \boldsymbol{\omega}_{A,0})^\wedge\, z \\
&= \mathbf{R}^{TA} \boldsymbol{\omega}_{A,0}^\wedge \mathbf{R}\, z \\
&= {}^0\mathbf{R}_A {}^A\boldsymbol{\omega}_{A,0}^\wedge {}^0\mathbf{R}_A^T\, z,
\end{aligned} \tag{4.39}$$

where we used the exponential map of SE(3) of (2.39) and the equalities (4.24) and (4.34) and (4.37). We write (4.39) as

$$\mathrm{D}\, f_2(\mathbf{R}) \circ \mathrm{D}\, L_{\mathbf{R}}(I) = {}^0\mathbf{R}_A {}^A\boldsymbol{\omega}_{A,0}^\wedge {}^0\mathbf{R}_A^T. \tag{4.40}$$

We obtain the left-trivialized derivative of ${}^0\mathbf{X}_A {}^A\mathbf{v}_{A,0}$ with respect to $\mathbf{H}$ as (4.21) by splitting it up in four submatrices, of which one is equal to zero and the others are presented in (4.33), (4.38) and (4.40). □

The following proposition shows how the left-trivialized derivative of the extended inverse dynamics of (3.39) with respect to the transformation matrix $\mathbf{H}$ can be rewritten to reduce the required amount of numerical computations.

Proposition 4.2: We can rewrite (4.17) to

$$\mathrm{D}_1\, \overline{ID}_b(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \dot{\mathbf{v}}, \dot{\mathbf{r}}) \circ \mathrm{D}\, L_H(I) = \frac{\tilde{\partial} \mathbf{b}^c_{\mathcal{B}0}}{\partial \mathbf{H}}. \tag{4.41}$$

■

Proof: In lines 7 and 8* of the EIDAmb algorithm (Table 4.1), $\mathbf{a}_0^r$ and $\mathbf{a}_0^{vp}$ are defined equal to each other. The left-trivialized derivatives of $\mathbf{a}_i^r$ and $\mathbf{a}_i^{vp}$ with respect to $\mathbf{H}$ are equal to each other, as they are both given by

$$\frac{\tilde{\partial} \mathbf{a}_i^r}{\partial \mathbf{H}} = {}^i\mathbf{X}_{\lambda(i)} \frac{\tilde{\partial} \mathbf{a}_{\lambda(i)}^r}{\partial \mathbf{H}} + \frac{\tilde{\partial} (\boldsymbol{\Gamma}_{\mathcal{J}i} \dot{\mathbf{r}}_i)}{\partial \mathbf{H}} + \frac{\tilde{\partial} \mathbf{v}_i}{\partial \mathbf{H}} \times \mathbf{v}_{\mathcal{J}i} \tag{4.42}$$

and

$$\frac{\tilde{\partial} \mathbf{a}_i^{vp}}{\partial \mathbf{H}} = {}^i\mathbf{X}_{\lambda(i)} \frac{\tilde{\partial} \mathbf{a}_{\lambda(i)}^{vp}}{\partial \mathbf{H}} + \frac{\tilde{\partial} \mathbf{v}_i}{\partial \mathbf{H}} \times \mathbf{v}_{\mathcal{J}i}. \tag{4.43}$$

The difference lies in $\tilde{\partial}(\boldsymbol{\Gamma}_{\mathcal{J}i} \dot{\mathbf{r}}_i)/\partial \mathbf{H}$, which is equal to zero as both $\boldsymbol{\Gamma}_{\mathcal{J}i}$ and $\dot{\mathbf{r}}_i$ are independent of $\mathbf{H}$. Furthermore the left-trivialized tangent spaces are independent of the terms $\mathbf{a}_i^r$ and $\mathbf{a}_i^{vp}$ (which are not equal to each other). Therefore all differences between both definitions are in this particular case equal to zero. Writing out the equation of the algorithm below step by step by applying the chain rule confirms that the left-trivialized derivatives of $\mathbf{b}^c_{\mathcal{B}0}$ and $\mathbf{b}^{vp}_{\mathcal{B}0}$ with respect to $\mathbf{H}$ are equal to each other. □

The following table presents the algorithm that computes the left-trivialized derivative of the extended inverse dynamics given by (3.39) with respect to the transformation matrix $\mathbf{H}$.

Table 4.2: Left-trivialized derivatives of the extended inverse dynamics w.r.t. the transformation matrix $\mathbf{H}$.

Inputs	*All outputs and intermediate variables of EIDAmb*	
Line	**Algorithm**	**Line in EIDAmb**
1	$\frac{\tilde{\partial}\mathbf{v}_0}{\partial\mathbf{H}} = \begin{bmatrix} {}^0\mathbf{R}_A {}^A\boldsymbol{\omega}^{\wedge}_{A,0} {}^0\mathbf{R}_A & {}^0\mathbf{R}_A {}^A\boldsymbol{v}^{\wedge}_{A,0} {}^0\mathbf{R}_A^T + ({}^0\mathbf{o}^{\wedge}_A {}^0\mathbf{R}_A {}^A\boldsymbol{\omega}_{A,0})^{\wedge} \\ 0_{3\times3} & {}^0\mathbf{R}_A {}^A\boldsymbol{\omega}^{\wedge}_{A,0} {}^0\mathbf{R}_A^T \end{bmatrix}$	6
2	$\frac{\tilde{\partial}\mathbf{a}_0^r}{\partial\mathbf{H}} = \begin{bmatrix} {}^0\mathbf{R}_A {}^A\boldsymbol{\alpha}^{\wedge}_{grav} {}^0\mathbf{R}_A & {}^0\mathbf{R}_A {}^A\boldsymbol{a}^{\wedge}_{grav} {}^0\mathbf{R}_A^T + ({}^0\mathbf{o}^{\wedge}_A {}^0\mathbf{R}_A {}^A\boldsymbol{\alpha}_{grav})^{\wedge} \\ 0_{3\times3} & {}^0\mathbf{R}_A {}^A\boldsymbol{\alpha}^{\wedge}_{grav} {}^0\mathbf{R}_A^T \end{bmatrix}$	7
3	$\frac{\tilde{\partial}\mathbf{m}_{\mathcal{B}0}}{\partial\mathbf{H}} = \mathbb{M}_{\mathcal{B}0}\frac{\tilde{\partial}\mathbf{v}_0}{\partial\mathbf{H}}$	10
4	$\frac{\tilde{\partial}\mathbf{b}^c_{\mathcal{B}0}}{\partial\mathbf{H}} = \mathbb{M}_{\mathcal{B}0}\frac{\tilde{\partial}\mathbf{a}^r_0}{\partial\mathbf{H}} + \frac{\tilde{\partial}\mathbf{v}_0}{\partial\mathbf{H}}\bar{\times}^*\mathbf{m}_{\mathcal{B}0} + \mathbf{v}_0\bar{\times}^*\frac{\tilde{\partial}\mathbf{m}_{\mathcal{B}0}}{\partial\mathbf{H}}$	11
5	**for** $i = 1$ **to** n_B **do**	13
6	$\quad\frac{\tilde{\partial}\mathbf{v}_i}{\partial\mathbf{H}} = {}^i\mathbf{X}_{\lambda(i)}\frac{\tilde{\partial}\mathbf{v}_{\lambda(i)}}{\partial\mathbf{H}}$	17
7	$\quad\frac{\tilde{\partial}\mathbf{a}^r_i}{\partial\mathbf{H}} = {}^i\mathbf{X}_{\lambda(i)}\frac{\tilde{\partial}\mathbf{a}^r_{\lambda(i)}}{\partial\mathbf{H}} + \frac{\tilde{\partial}\mathbf{v}_i}{\partial\mathbf{H}}\times\mathbf{v}_{\mathcal{J}i}$	18
8	$\quad\frac{\tilde{\partial}\mathbf{m}_{\mathcal{B}i}}{\partial\mathbf{H}} = \mathbb{M}_{\mathcal{B}i}\frac{\tilde{\partial}\mathbf{v}_i}{\partial\mathbf{H}}$	21
9	$\quad\frac{\tilde{\partial}\mathbf{b}^c_{\mathcal{B}i}}{\partial\mathbf{H}} = \mathbb{M}_{\mathcal{B}i}\frac{\tilde{\partial}\mathbf{a}^r_i}{\partial\mathbf{H}} + \frac{\tilde{\partial}\mathbf{v}_i}{\partial\mathbf{H}}\bar{\times}^*\mathbf{m}_{\mathcal{B}i} + \mathbf{v}_i\bar{\times}^*\frac{\tilde{\partial}\mathbf{m}_{\mathcal{B}i}}{\partial\mathbf{H}}$	22
10	**end**	24
11	**for** $i = n_B$ **to** 1 **do**	25
12	$\quad\frac{\tilde{\partial}\mathbf{b}^c_{\mathcal{B}\lambda(i)}}{\partial\mathbf{H}} = \frac{\tilde{\partial}\mathbf{b}^c_{\mathcal{B}\lambda(i)}}{\partial\mathbf{H}} + {}_{\lambda(i)}\mathbf{X}^i\frac{\tilde{\partial}\mathbf{b}^c_{\mathcal{B}i}}{\partial\mathbf{H}}$	27
13	**end**	29
14	**for** $i = 1$ **to** n_B **do**	30
15	$\quad\frac{\tilde{\partial}\boldsymbol{\tau}_i}{\partial\mathbf{H}} = \boldsymbol{\Gamma}^T_{\mathcal{J}i}\frac{\tilde{\partial}\mathbf{b}^c_{\mathcal{B}i}}{\partial\mathbf{H}}$	32
16	**end**	40
17	$\mathrm{D}_1\overline{ID}_b\circ\mathrm{D}\,L_H(I) = \frac{\tilde{\partial}\mathbf{b}^c_{\mathcal{B}0}}{\partial\mathbf{H}}$	41***
Outputs	$\mathrm{D}_1\overline{ID}_j\circ\mathrm{D}\,L_H(I)(=\tilde{\partial}\boldsymbol{\tau}/\partial\mathbf{H}), \mathrm{D}_1\overline{ID}_b\circ\mathrm{D}\,L_H(I)(=\tilde{\partial}\bar{\boldsymbol{\tau}}_b/\partial\mathbf{H})$	

Remarks:

- Line 1 computes the left trivialized derivative $\tilde{\partial}({}^0\mathbf{X}_A {}^A\mathbf{v}_{A,0})/\partial^A\mathbf{H}_0$ in (4.21).
- Line 2 computes the left trivialized derivative $\tilde{\partial}({}^0\mathbf{X}_A {}^A\mathbf{a}_{grav})/\partial^A\mathbf{H}_0$ in (4.22).

- Line 17 computes the derivative of the extended inverse dynamics of the moving-base with respect to the transformation matrix $\mathbf{H}$ in (4.41).

4.2.2 Computing the derivative of the extended inverse dynamics w.r.t. the generalized position vector

The table below presents the algorithm that efficiently computes the derivatives of the extended inverse dynamics (3.39) with respect to the generalized position vector $\mathbf{s}$. The right column shows which line in the EIDAmb each equation origins from. The function 'jcalcderiv(jtype(i),$\mathbf{s}_i$)' computes the derivative of the velocity transformation matrix ${}^{i}\mathbf{X}_{\lambda(i|i}$ with respect to the generalized position $\mathbf{s}_i$. Its inputs are the joint type, and the joint position $\mathbf{s}_i$. The function 'jcalcderiv' is detailed in Appendix B.3.

Table 4.3: Derivatives of the extended inverse dynamics w.r.t. the generalized position vector $\mathbf{s}$.

Inputs	*All outputs and intermediate variables of EIDAmb*	
Line	**Algorithm**	**Line in EIDAmb**
1	**for** $i = 1$ **to** n_B **do**	13
2	$\frac{\partial^i \mathbf{X}_{\lambda(i)\vert i}}{\partial \mathbf{s}_i} = \text{jcalcderiv}(\text{jtype}(i), \mathbf{s}_i)$	14
3	$\frac{\partial^i \mathbf{X}_{\lambda(i)}}{\partial \mathbf{s}_i} = \frac{\partial^i \mathbf{X}_{\lambda(i)\vert i}}{\partial \mathbf{s}_i} {}^{\lambda(i)\vert i}\mathbf{X}_{\lambda(i)}$	16
4	$\frac{\partial \mathbf{v}_i}{\partial \mathbf{s}} = {}^i\mathbf{X}_{\lambda(i)} \frac{\partial \mathbf{v}_{\lambda(i)}}{\partial \mathbf{s}}$	17
5	$\frac{\partial \mathbf{v}_i}{\partial \mathbf{s}_i} = \frac{\partial \mathbf{v}_i}{\partial \mathbf{s}_i} + \frac{\partial^i \mathbf{X}_{\lambda(i)}}{\partial \mathbf{s}_i} \mathbf{v}_{\lambda(i)}$	17
6	$\frac{\partial \mathbf{a}_i^r}{\partial \mathbf{s}} = {}^i\mathbf{X}_{\lambda(i)} \frac{\partial \mathbf{a}_{\lambda(i)}^r}{\partial \mathbf{s}} + \frac{\partial \mathbf{v}_i}{\partial \mathbf{s}} \times \mathbf{v}_{\mathcal{J}i}$	18
7	$\frac{\partial \mathbf{a}_i^r}{\partial \mathbf{s}_i} = \frac{\partial \mathbf{a}_i^r}{\partial \mathbf{s}_i} + \frac{\partial^i \mathbf{X}_{\lambda(i)}}{\partial \mathbf{s}_i} \mathbf{a}_{\lambda(i)}^r$	18
8	$\frac{\partial \mathbf{a}_i^{vp}}{\partial \mathbf{s}} = {}^i\mathbf{X}_{\lambda(i)} \frac{\partial \mathbf{a}_{\lambda(i)}^{vp}}{\partial \mathbf{s}} + \frac{\partial \mathbf{v}_i}{\partial \mathbf{s}} \times \mathbf{v}_{\mathcal{J}i}$	19*
9	$\frac{\partial \mathbf{a}_i^{vp}}{\partial \mathbf{s}_i} = \frac{\partial \mathbf{a}_i^{vp}}{\partial \mathbf{s}_i} + \frac{\partial^i \mathbf{X}_{\lambda(i)}}{\partial \mathbf{s}_i} \mathbf{a}_{\lambda(i)}^{vp}$	19*
10	$\frac{\partial \mathbf{m}_{\mathcal{B}i}}{\partial \mathbf{s}} = \mathbb{M}_{\mathcal{B}i} \frac{\partial \mathbf{v}_i}{\partial \mathbf{s}}$	21
11	$\frac{\partial \mathbf{b}_{\mathcal{B}i}^c}{\partial \mathbf{s}} = \mathbb{M}_{\mathcal{B}i} \frac{\partial \mathbf{a}_i^r}{\partial \mathbf{s}} + \frac{\partial \mathbf{v}_i}{\partial \mathbf{s}} \bar{\times}^* \mathbf{m}_{\mathcal{B}i} + \mathbf{v}_i \bar{\times}^* \frac{\partial \mathbf{m}_{\mathcal{B}i}}{\partial \mathbf{s}}$	22
12	$\frac{\partial \mathbf{b}_{\mathcal{B}i}^{vp}}{\partial \mathbf{s}} = \mathbb{M}_{\mathcal{B}i} \frac{\partial \mathbf{a}_i^{vp}}{\partial \mathbf{s}} + \frac{\partial \mathbf{v}_i}{\partial \mathbf{s}} \bar{\times}^* \mathbf{m}_{\mathcal{B}i} + \mathbf{v}_i \bar{\times}^* \frac{\partial \mathbf{m}_{\mathcal{B}i}}{\partial \mathbf{s}}$	23*
13	**end**	24
14	**for** $i = n_B$ **to** 1 **do**	25
15	**for** $k = 1$ **to** n_B **do**	26
16	$\frac{\partial \mathbb{M}_{\mathcal{B}\lambda(i)}^c}{\partial \mathbf{s}_k} = \frac{\partial \mathbb{M}_{\mathcal{B}\lambda(i)}^c}{\partial \mathbf{s}_k} + {}_{\lambda(i)}\mathbf{X}^i \frac{\partial \mathbb{M}_{\mathcal{B}i}^c}{\partial \mathbf{s}_k} {}^i\mathbf{X}_{\lambda(i)}$	26
17	**end**	26
18	$\frac{\partial \mathbb{M}_{\mathcal{B}\lambda(i)}^c}{\partial \mathbf{s}_i} = \frac{\partial \mathbb{M}_{\mathcal{B}\lambda(i)}^c}{\partial \mathbf{s}_i} + \frac{\partial {}_{\lambda(i)}\mathbf{X}^i}{\partial \mathbf{s}_i} \mathbb{M}_{\mathcal{B}i}^c {}^i\mathbf{X}_{\lambda(i)} + {}_{\lambda(i)}\mathbf{X}^i \, \mathbb{M}_{\mathcal{B}i}^c \frac{\partial^i \mathbf{X}_{\lambda(i)}}{\partial \mathbf{s}_i}$	26
19	$\frac{\partial \mathbf{b}_{\mathcal{B}\lambda(i)}^c}{\partial \mathbf{s}} = \frac{\partial \mathbf{b}_{\mathcal{B}\lambda(i)}^c}{\partial \mathbf{s}} + {}_{\lambda(i)}\mathbf{X}^i \frac{\partial \mathbf{b}_{\mathcal{B}i}^c}{\partial \mathbf{s}}$	27
20	$\frac{\partial \mathbf{b}_{\mathcal{B}\lambda(i)}^c}{\partial \mathbf{s}_i} = \frac{\partial \mathbf{b}_{\mathcal{B}\lambda(i)}^c}{\partial \mathbf{s}_i} + \frac{\partial {}_{\lambda(i)}\mathbf{X}^i}{\partial \mathbf{s}_i} \mathbf{b}_{\mathcal{B}i}^c$	27
21	$\frac{\partial \mathbf{b}_{\mathcal{B}\lambda(i)}^{vp}}{\partial \mathbf{s}} = \frac{\partial \mathbf{b}_{\mathcal{B}\lambda(i)}^{vp}}{\partial \mathbf{s}} + {}_{\lambda(i)}\mathbf{X}^i \frac{\partial \mathbf{b}_{\mathcal{B}i}^{vp}}{\partial \mathbf{s}}$	28*
	continued on the next page.	

22	$\quad\frac{\partial \mathbf{b}^{vp}_{\mathcal{B}\lambda(i)}}{\partial \mathbf{s}_i} = \frac{\partial \mathbf{b}^{vp}_{\mathcal{B}\lambda(i)}}{\partial \mathbf{s}_i} + \frac{\partial_{\lambda(i)}\mathbf{X}^i}{\partial \mathbf{s}_i}\mathbf{b}^{vp}_{\mathcal{B}i}$	28*
23	**end**	29
24	**for** $k = 1$ **to** n_B **do**	-
25	$\quad\frac{\partial \mathbb{M}^c_{\mathcal{B}0}\mathbf{a}_0}{\partial \mathbf{s}_k} = \frac{\partial \mathbb{M}^c_{\mathcal{B}0}}{\partial \mathbf{s}_k}\mathbf{a}_0$	-
26	**end**	-
27	**for** $i = 1$ **to** n_B **do**	30
28	$\quad\frac{\partial^i \mathbf{a}_{A,0}}{\partial \mathbf{s}} = {}^i\mathbf{X}_{\lambda(i)}\frac{\partial^{\lambda(i)}\mathbf{a}_{A,0}}{\partial \mathbf{s}}$	31
29	$\quad\frac{\partial^i \mathbf{a}_{A,0}}{\partial \mathbf{s}_i} = \frac{\partial^i \mathbf{a}_{A,0}}{\partial \mathbf{s}_i} + \frac{\partial^i \mathbf{X}_{\lambda(i)}}{\partial \mathbf{s}_i}{}^{\lambda(i)}\mathbf{a}_{A,0}$	31
30	$\quad$**for** $k = 1$ **to** n_B **do**	-
31	$\quad\quad\frac{\partial \mathbb{M}^c_{\mathcal{B}i}\,{}^i\mathbf{a}_{A,0}}{\partial \mathbf{s}_k} = \frac{\partial \mathbb{M}^c_{\mathcal{B}i}}{\partial \mathbf{s}_k}{}^i\mathbf{a}_{A,0}$	-
32	$\quad\quad\frac{\partial_i \mathbf{F}_{\mathcal{B}i}}{\partial \mathbf{s}_k} = \frac{\partial \mathbb{M}^c_{\mathcal{B}i}}{\partial \mathbf{s}_k}\boldsymbol{\Gamma}_{\mathcal{J}i}$	33**
33	$\quad$**end**	-
34	$\quad\frac{\partial \boldsymbol{\tau}_i}{\partial \mathbf{s}} = \boldsymbol{\Gamma}^T_{\mathcal{J}i}\left(\frac{\partial \mathbb{M}^c_{\mathcal{B}i}}{\partial \mathbf{s}}{}^i\mathbf{a}_{A,0} + \mathbb{M}^c_{\mathcal{B}i}\frac{\partial^i \mathbf{a}_{A,0}}{\partial \mathbf{s}} + \frac{\partial \mathbf{b}^c_{\mathcal{B}i}}{\partial \mathbf{s}}\right)$	33
35	$\quad j = i$	34**
36	$\quad$**while** $\lambda(j) > 0$	35**
37	$\quad\quad$**for** $k = 1$ **to** n_B **do**	36**
38	$\quad\quad\quad\frac{\partial_{\lambda(j)}\mathbf{F}_{\mathcal{B}i}}{\partial \mathbf{s}_k} = {}_{\lambda(j)}\mathbf{X}^j\frac{\partial_j \mathbf{F}_{\mathcal{B}i}}{\partial \mathbf{s}_k}$	36**
39	$\quad\quad$**end**	36**
40	$\quad\quad\frac{\partial_{\lambda(j)}\mathbf{F}_{\mathcal{B}i}}{\partial \mathbf{s}_j} = \frac{\partial_{\lambda(j)}\mathbf{F}_{\mathcal{B}i}}{\partial \mathbf{s}_j} + \frac{\partial_{\lambda(j)}\mathbf{X}^j}{\partial \mathbf{s}_j}{}_j\mathbf{F}_{\mathcal{B}i}$	36**
41	$\quad\quad j = \lambda(j)$	37**
42	$\quad$**end**	38**
43	$\quad$**for** $k = 1$ **to** n_B **do**	39**
44	$\quad\quad\frac{\partial_0 \mathbf{F}_{\mathcal{B}i}}{\partial \mathbf{s}_k} = {}_0\mathbf{X}^j\frac{\partial_j \mathbf{F}_{\mathcal{B}i}}{\partial \mathbf{s}_k}$	39**
45	$\quad$**end**	39**
46	$\quad\frac{\partial_0 \mathbf{F}_{\mathcal{B}i}}{\partial \mathbf{s}_j} = \frac{\partial_0 \mathbf{F}_{\mathcal{B}i}}{\partial \mathbf{s}_j} + \frac{\partial_0 \mathbf{X}^j}{\partial \mathbf{s}_j}{}_j\mathbf{F}_{\mathcal{B}i}$	39**
47	$\quad\frac{\partial_0 \mathbf{F}\dot{\mathbf{r}}}{\partial \mathbf{s}} = \frac{\partial_0 \mathbf{F}\dot{\mathbf{r}}}{\partial \mathbf{s}} + \frac{\partial_0 \mathbf{F}_{\mathcal{B}i}}{\partial \mathbf{s}}\dot{\mathbf{r}}$	-
48	**end**	40
49	$\mathrm{D}_2\,\overline{ID}_b = \frac{\partial \mathbb{M}^c_{\mathcal{B}0}\mathbf{a}_0}{\partial \mathbf{s}} + \frac{\partial_0 \mathbf{F}\dot{\mathbf{r}}}{\partial \mathbf{s}} + \frac{\partial \mathbf{b}^{vp}_{\mathcal{B}0}}{\partial \mathbf{s}}$	41***
Outputs	$\mathrm{D}_2\,\overline{ID}_j (= \partial \boldsymbol{\tau}/\partial \mathbf{s}), \mathrm{D}_2\,\overline{ID}_b (= \partial \bar{\boldsymbol{\tau}}_b/\partial \mathbf{s})$	

Remarks:

- Note that Proposition 4.2 does not hold for the left-trivialized derivatives of $\mathbf{b}^c_{\mathcal{B}0}$ and $\mathbf{b}^{vp}_{\mathcal{B}0}$ with respect to $\mathbf{s}$, as lines 7 and 9 add the terms $\mathbf{a}^r_{\lambda(i)}$ and $\mathbf{a}^{vp}_{\lambda(i)}$ respectively, which are generally not equal to each other.
- In the rightmost column, lines marked with * origin from the GBWAmb, lines marked with ** origin from the CRBAmb,and unmarked lines origin from the RNEAmb.
- Line 49 computes the derivative of the extended inverse dynamics of the moving-base with respect to the generalized position vector $\mathbf{s}$ in (4.18), where we use

$$\frac{\partial \mathbb{M}^c_{\mathcal{B}0}\mathbf{a}_0}{\partial \mathbf{s}} = \frac{\partial \mathbb{M}^c_{\mathcal{B}0}}{\partial \mathbf{s}}\mathbf{a}_0 \tag{4.44}$$

and

$$\frac{\partial_0 \mathbf{F}\dot{\mathbf{r}}}{\partial \mathbf{s}} = \frac{\partial_0 \mathbf{F}}{\partial \mathbf{s}}\dot{\mathbf{r}}, \tag{4.45}$$

since $\partial \mathbf{a}_0/\partial \mathbf{s} = 0$ and $\partial \dot{\mathbf{r}}/\partial \mathbf{s} = 0$.

4.2.3 Computing the derivatives of the extended inverse dynamics w.r.t. the moving-base velocity

The table below presents the algorithm that efficiently computes the derivatives of the extended inverse dynamics (3.39) the moving-base velocity $\mathbf{v}$. The right column shows which line in the EIDAmb each equation origins from. Similar to Proposition 4.2, we can rewrite (4.19) to

$$\mathrm{D}_3\, \overline{ID}_b(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \dot{\mathbf{v}}, \dot{\mathbf{r}}) = \frac{\partial \mathbf{b}^c_{\mathcal{B}0}}{\partial \mathbf{v}}, \tag{4.46}$$

as the same holds for the left-trivialized derivatives of $\mathbf{b}^c_{\mathcal{B}0}$ and $\mathbf{b}^{vp}_{\mathcal{B}0}$ with respect to $\mathbf{v}$.

Table 4.4: Derivatives of the extended inverse dynamics w.r.t. the moving-base velocity $\mathbf{v}$.

Inputs	*All outputs and intermediate variables of EIDAmb*	
Line	**Algorithm**	**Line in EIDAmb**
1	$\frac{\partial \mathbf{v}_0}{\partial \mathbf{v}} = I_6$	6
2	$\frac{\partial \mathbf{m}_{\mathcal{B}0}}{\partial \mathbf{v}} = \mathbb{M}_{\mathcal{B}0}$	10
3	$\frac{\partial \mathbf{b}^c_{\mathcal{B}0}}{\partial \mathbf{v}} = \frac{\partial \mathbf{v}_0}{\partial \mathbf{v}} \bar{\times}^* \mathbf{m}_{\mathcal{B}0} + \mathbf{v}_0 \bar{\times}^* \frac{\partial \mathbf{m}_{\mathcal{B}0}}{\partial \mathbf{v}}$	11
4	**for** $i = 1$ **to** n_B **do**	13
5	$\quad \frac{\partial \mathbf{v}_i}{\partial \mathbf{v}} = {}^i\mathbf{X}_{\lambda(i)} \frac{\partial \mathbf{v}_{\lambda(i)}}{\partial \mathbf{v}}$	17
6	$\quad \frac{\partial \mathbf{a}^r_i}{\partial \mathbf{v}} = {}^i\mathbf{X}_{\lambda(i)} \frac{\partial \mathbf{a}^r_{\lambda(i)}}{\partial \mathbf{v}} + \frac{\partial \mathbf{v}_i}{\partial \mathbf{v}} \times \mathbf{v}_{\mathcal{J}i}$	18
7	$\quad \frac{\partial \mathbf{m}_{\mathcal{B}i}}{\partial \mathbf{v}} = \mathbb{M}_{\mathcal{B}i} \frac{\partial \mathbf{v}_i}{\partial \mathbf{v}}$	21
8	$\quad \frac{\partial \mathbf{b}^c_{\mathcal{B}i}}{\partial \mathbf{v}} = \mathbb{M}_{\mathcal{B}i} \frac{\partial \mathbf{a}^r_i}{\partial \mathbf{v}} + \frac{\partial \mathbf{v}_i}{\partial \mathbf{v}} \bar{\times}^* \mathbf{m}_{\mathcal{B}i} + \mathbf{v}_i \bar{\times}^* \frac{\partial \mathbf{m}_{\mathcal{B}i}}{\partial \mathbf{v}}$	22
9	**end**	24
10	**for** $i = n_B$ **to** 1 **do**	25
11	$\quad \frac{\partial \mathbf{b}^c_{\mathcal{B}\lambda(i)}}{\partial \mathbf{v}} = \frac{\partial \mathbf{b}^c_{\mathcal{B}\lambda(i)}}{\partial \mathbf{v}} + {}_{\lambda(i)}\mathbf{X}^i \frac{\partial \mathbf{b}^c_{\mathcal{B}i}}{\partial \mathbf{v}}$	27
12	**end**	29
13	**for** $i = 1$ **to** n_B **do**	30
14	$\quad \frac{\partial \boldsymbol{\tau}_i}{\partial \mathbf{v}} = \boldsymbol{\Gamma}^T_{\mathcal{J}i} \frac{\partial \mathbf{b}^c_{\mathcal{B}i}}{\partial \mathbf{v}}$	32
15	**end**	40
16	$\mathrm{D}_3 \overline{ID}_b = \frac{\partial \mathbf{b}^c_{\mathcal{B}0}}{\partial \mathbf{v}}$	41***
Outputs	$\mathrm{D}_3 \overline{ID}_j (= \partial \boldsymbol{\tau}/\partial \mathbf{v}), \mathrm{D}_3 \overline{ID}_b (= \partial \bar{\boldsymbol{\tau}}_b/\partial \mathbf{v})$	

Remark: Line 16 computes the derivative of the extended inverse dynamics of the moving-base with respect to the moving-base velocity $\mathbf{v}$ in (4.46)

4.2.4 Computing the derivatives of the extended inverse dynamics w.r.t. the generalized velocity vector

The table below presents the algorithm that efficiently computes the derivatives of the extended inverse dynamics (3.39) the generalized velocity vector $\mathbf{r}$. The right column shows which line in the EIDAmb each equation origins from. Similar to Proposition 4.2, we can rewrite (4.20) to

$$\mathrm{D}_4 \overline{ID}_b(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \dot{\mathbf{v}}, \dot{\mathbf{r}}) = \frac{\partial \mathbf{b}^c_{\mathcal{B}0}}{\partial \mathbf{r}}, \tag{4.47}$$

as the same holds for the left-trivialized derivatives of $\mathbf{b}^c_{\mathcal{B}0}$ and $\mathbf{b}^{vp}_{\mathcal{B}0}$ with respect to $\mathbf{r}$.

Table 4.5: Derivatives of the extended inverse dynamics w.r.t. the generalized velocity vector $\mathbf{r}$.

Inputs	*All outputs and intermediate variables of EIDAmb*	
Line	**Algorithm**	**Line in EIDAmb**
1	**for** $i = 1$ **to** n_B **do**	13
2	$\dfrac{\partial \mathbf{v}_{\mathcal{J}i}}{\partial \mathbf{r}_i} = \mathbf{\Gamma}_{\mathcal{J}i}$	15
3	$\dfrac{\partial \mathbf{v}_i}{\partial \mathbf{r}} = {}^{i}\mathbf{X}_{\lambda(i)} \dfrac{\partial \mathbf{v}_{\lambda(i)}}{\partial \mathbf{r}}$	17
4	$\dfrac{\partial \mathbf{v}_i}{\partial \mathbf{r}_i} = \dfrac{\partial \mathbf{v}_i}{\partial \mathbf{r}_i} + \dfrac{\partial \mathbf{v}_{\mathcal{J}i}}{\partial \mathbf{r}_i}$	17
5	$\dfrac{\partial \mathbf{a}^r_i}{\partial \mathbf{r}} = {}^{i}\mathbf{X}_{\lambda(i)} \dfrac{\partial \mathbf{a}^r_{\lambda(i)}}{\partial \mathbf{r}} + \dfrac{\partial \mathbf{v}_i}{\partial \mathbf{r}} \times \mathbf{v}_{\mathcal{J}i}$	18
6	$\dfrac{\partial \mathbf{a}^r_i}{\partial \mathbf{r}_i} = \dfrac{\partial \mathbf{a}^r_i}{\partial \mathbf{r}_i} + \mathbf{v}_i \times \dfrac{\partial \mathbf{v}_{\mathcal{J}i}}{\partial \mathbf{r}_i}$	18
7	$\dfrac{\partial \mathbf{m}_{\mathcal{B}i}}{\partial \mathbf{r}} = \mathbb{M}_{\mathcal{B}i} \dfrac{\partial \mathbf{v}_i}{\partial \mathbf{r}}$	21
8	$\dfrac{\partial \mathbf{b}^c_{\mathcal{B}i}}{\partial \mathbf{r}} = \mathbb{M}_{\mathcal{B}i} \dfrac{\partial \mathbf{a}^r_i}{\partial \mathbf{r}} + \dfrac{\partial \mathbf{v}_i}{\partial \mathbf{r}} \bar{\times}^* \mathbf{m}_{\mathcal{B}i} + \mathbf{v}_i \bar{\times}^* \dfrac{\partial \mathbf{m}_{\mathcal{B}i}}{\partial \mathbf{r}}$	22
9	**end**	24
10	**for** $i = n_B$ **to** 1 **do**	25
11	$\dfrac{\partial \mathbf{b}^c_{\mathcal{B}\lambda(i)}}{\partial \mathbf{r}} = \dfrac{\partial \mathbf{b}^c_{\mathcal{B}\lambda(i)}}{\partial \mathbf{r}} + {}_{\lambda(i)}\mathbf{X}^{i} \dfrac{\partial \mathbf{b}^c_{\mathcal{B}i}}{\partial \mathbf{r}}$	27
12	**end**	29
13	**for** $i = 1$ **to** n_B **do**	30
14	$\dfrac{\partial \boldsymbol{\tau}_i}{\partial \mathbf{r}} = \mathbf{\Gamma}^T_{\mathcal{J}i} \dfrac{\partial \mathbf{b}^c_{\mathcal{B}i}}{\partial \mathbf{r}}$	32
15	**end**	40
16	$\mathrm{D}_4 \overline{ID}_b = \dfrac{\partial \mathbf{b}^c_{\mathcal{B}0}}{\partial \mathbf{r}}$	41***
Outputs	$\mathrm{D}_4 \overline{ID}_j (= \partial \boldsymbol{\tau}/\partial \mathbf{r}), \mathrm{D}_4 \overline{ID}_b (= \partial \bar{\boldsymbol{\tau}}_b/\partial \mathbf{r})$	

Remark: Line 16 computes the derivative of the extended inverse dynamics of the moving-base with respect to the generalized velocity vector $\mathbf{r}$ in (4.47)

4.3 Inverse Mass Matrix Algorithm for fixed-base systems

In [28], the author presents an algorithm to compute the inverse of the mass matrix in an efficient way. We call this algorithm the Inverse Mass Matrix Algorithm (IMMA). Instead of using the Composite Rigid Body algorithm (CRBA, presented in Appendix E.3) to compute the mass matrix, and then inverting it, the IMMA is based on the Articulated Body Algorithm (ABA, which can be found in Appendix E.5) to compute the inverse of the mass matrix directly, without first computing the mass matrix itself. We recall that the ABA computes the forward dynamics of fixed-base systems

$FD_{fb} : \mathbb{R}^{n_J} \times \mathbb{R}^{n_J} \times \mathbb{R}^{n_J} \rightarrow \mathbb{R}^{n_J}$ as

$$FD_{fb}(\mathbf{s}, \mathbf{r}, \boldsymbol{\tau}) := \dot{\mathbf{r}} = \mathbf{M}_{fb}^{-1}(\boldsymbol{\tau} - \mathbf{h}_{fb}), \tag{4.48}$$

which is obtained from the dynamics of fixed-base systems (2.55). By defining $\mathbf{h}_{fb}$ equal to zero and computing $\partial\dot{\mathbf{r}}/\partial\boldsymbol{\tau}$, the inverse of the mass matrix $\mathbf{M}_{fb}^{-1}$ is found. The bias wrench for fixed-base systems $\mathbf{h}_{fb}$ is equal to zero if the Coriolis, gravitational and external wrenches are equal to zero. The table below presents the IMMA for fixed-base systems, which computes the inverse of the mass matrix for fixed-base systems $\mathbf{M}_{fb}^{inv} = \mathbf{M}_{fb}^{-1}$, without needing to first compute the mass matrix $\mathbf{M}_{fb}$ itself.

Remark. We have discovered that the original version of the IMMA presented in [28] is incorrect. This is reported and further elaborated in Appendix C. The table below presents a corrected version of the IMMA.

Table 4.6: Inverse Mass Matrix Algorithm for fixed-base systems.

Inputs	model, $\mathbf{s}$
Line	**IMMA**
1	**for** $i = 1$ **to** n_B **do**
2	$\quad [{}^{i}\mathbf{X}_{\lambda(i)\vert i}, \mathbf{\Gamma}_{\mathcal{J}i}] = \text{jcalc}(\text{jtype}(i), \mathbf{s}_i)$
3	$\quad {}^{i}\mathbf{X}_{\lambda(i)} = {}^{i}\mathbf{X}_{\lambda(i)\vert i}\,{}^{\lambda(i)\vert i}\mathbf{X}_{\lambda(i)}$
4	$\quad \mathbb{M}^{c}_{\mathcal{B}i} = \mathbb{M}_{\mathcal{B}i}$
5	**end**
6	**for** $i = N_B$ **to** 1 **do**
7	$\quad \mathbf{U}_{\mathcal{B}i} = \mathbb{M}^{A}_{\mathcal{B}i}\mathbf{\Gamma}_{\mathcal{J}i}$
8	$\quad \mathbf{D}_{\mathcal{B}i} = \mathbf{\Gamma}^{T}_{\mathcal{J}i}\mathbf{U}_{\mathcal{B}i}$
9	$\quad \mathbf{M}^{inv}_{fb}[i,i] = \mathbf{D}^{-1}_{\mathcal{B}i}$
10	$\quad \mathbf{M}^{inv}_{fb}[i, subtree(i)] = \mathbf{M}^{inv}_{fb}[i, subtree(i)] - \mathbf{D}^{-1}_{\mathcal{B}i}\mathbf{\Gamma}^{T}_{\mathcal{J}i}\mathcal{F}_i[:, subtree(i)]$
11	$\quad$ **if** $\lambda(i) \neq 0$ **then**
12	$\quad\quad \mathcal{F}_{\lambda(i)}[:, subtree(i)] = \mathcal{F}_{\lambda(i)}[:, subtree(i)] + {}_{\lambda(i)}\mathbf{X}^{i}\big(\mathcal{F}_i[:, subtree(i)]$ $+ \mathbf{U}_{\mathcal{B}i}\mathbf{M}^{inv}_{fb}[i, subtree(i)]\big)$
13	$\quad\quad \mathbb{M}^{a}_{\mathcal{B}i} = \mathbb{M}^{A}_{\mathcal{B}i} - \mathbf{U}_{\mathcal{B}i}\mathbf{D}^{-1}_{\mathcal{B}i}\mathbf{U}^{T}_{\mathcal{B}i}$
14	$\quad\quad \mathbb{M}^{A}_{\mathcal{B}\lambda(i)} = \mathbb{M}^{A}_{\mathcal{B}\lambda(i)} + {}_{\lambda(i)}\mathbf{X}^{i}\mathbb{M}^{a}_{\mathcal{B}i}\,{}^{i}\mathbf{X}_{\lambda(i)}$
15	$\quad$ **end**
16	**end**
17	**for** $i = 1$ **to** n_B **do**
18	$\quad$ **if** $\lambda(i) \neq 0$ **then**
19	$\quad\quad \mathbf{M}^{inv}_{fb}[i,:] = \mathbf{M}^{inv}_{fb}[i,:] - \mathbf{D}^{-1}_{\mathcal{B}i}\mathbf{U}^{T}_{\mathcal{B}i}\,{}^{i}\mathbf{X}_{\lambda(i)}\mathcal{P}_{\lambda(i)}$
20	$\quad$ **end**
21	$\quad \mathcal{P}_i = \mathbf{\Gamma}_{\mathcal{J}i}\mathbf{M}^{inv}_{fb}[i,:]$
22	$\quad$ **if** $\lambda(i) \neq 0$ **then**
23	$\quad\quad \mathcal{P}_i = \mathcal{P}_i + {}^{i}\mathbf{X}_{\lambda(i)}\mathcal{P}_{\lambda(i)}$
24	$\quad$ **end**
25	**end**
26	**for** $i = 1$ **to** n_B **do**
27	$\quad$ **for** $j = i$ **to** n_B **do**
28	$\quad\quad \mathbf{M}^{inv}_{fb}[j,i] = \mathbf{M}^{inv}_{fb}[i,j]$
29	$\quad$ **end**
30	**end**
Outputs	$\mathbf{M}^{inv}_{fb}$

4.4 Inverse Mass Matrix Algorithm for moving-base systems

The IMMA (Table 4.6) can be extended to moving-base systems by using the same approach, based on the Articulated Body Algorithm for moving-base systems (ABAmb, which can be found in Appendix E.6). We call the resulting algorithm the Inverse Mass Matrix Algorithm for moving-base systems (IMMAmb). We rewrite the forward dynamics given in (3.17) for moving-base systems to

$$FD(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \boldsymbol{\tau}) := \begin{bmatrix} \dot{\mathbf{v}} \\ \dot{\mathbf{r}} \end{bmatrix} = \begin{bmatrix} \mathbf{M}_{11} & \mathbf{M}_{12} \\ \mathbf{M}_{21} & \mathbf{M}_{22} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{h}_1 \\ \boldsymbol{\tau} - \mathbf{h}_2 \end{bmatrix}. \tag{4.49}$$

The IMMAmb defines $\mathbf{h}_2$ equal to zero, so that the inverse of the mass matrix can be found as

$$\begin{bmatrix} \mathbf{M}_{11} & \mathbf{M}_{12} \\ \mathbf{M}_{21} & \mathbf{M}_{22} \end{bmatrix}^{-1} = \partial \begin{bmatrix} \dot{\mathbf{v}} \\ \dot{\mathbf{r}} \end{bmatrix} / \partial \begin{bmatrix} \mathbf{h}_1 \\ \boldsymbol{\tau} \end{bmatrix}. \tag{4.50}$$

The table below presents the IMMAmb, which computes the inverse of the mass matrix $\mathbf{M}^{inv} = \mathbf{M}^{-1}$, without needing to first compute the mass matrix $\mathbf{M}$ itself.

Table 4.7: Inverse Mass Matrix Algorithm for moving-base systems.

Inputs	model, $\mathbf{s}$
Line	**IMMAmb**
1	**for** $i = 1$ **to** n_B **do**
2	$[{}^{i}\mathbf{X}_{\lambda(i)\|i}, \boldsymbol{\Gamma}_{\mathcal{J}i}] = \text{jcalc}(\text{jtype}(i), \mathbf{s}_i)$
3	${}^{i}\mathbf{X}_{\lambda(i)} = {}^{i}\mathbf{X}_{\lambda(i)\|i}{}^{\lambda(i)\|i}\mathbf{X}_{\lambda(i)}$
4	$\mathbb{M}^{c}_{\mathcal{B}i} = \mathbb{M}_{\mathcal{B}i}$
5	**end**
6	**for** $i = N_B$ **to** 1 **do**
7	$\mathbf{U}_{\mathcal{B}i} = \mathbb{M}^{A}_{\mathcal{B}i}\boldsymbol{\Gamma}_{\mathcal{J}i}$
8	$\mathbf{D}_{\mathcal{B}i} = \boldsymbol{\Gamma}^{T}_{\mathcal{J}i}\mathbf{U}_{\mathcal{B}i}$
9	$\mathbf{M}^{inv}[i, i] = \mathbf{D}^{-1}_{\mathcal{B}i}$
10	$\mathbf{M}^{inv}[i, subtree(i)] = \mathbf{M}^{inv}[i, subtree(i)] - \mathbf{D}^{-1}_{\mathcal{B}i}\boldsymbol{\Gamma}^{T}_{\mathcal{J}i}\mathcal{F}_i[:, subtree(i)]$
11	$\mathcal{F}_{\lambda(i)}[:, subtree(i)] = \mathcal{F}_{\lambda(i)}[:, subtree(i)] + {}_{\lambda(i)}\mathbf{X}^{i}\big(\mathcal{F}_i[:, subtree(i)]$ $+ \mathbf{U}_{\mathcal{B}i}\mathbf{M}^{inv}[i, subtree(i)]\big)$
12	$\mathbb{M}^{a}_{\mathcal{B}i} = \mathbb{M}^{A}_{\mathcal{B}i} - \mathbf{U}_{\mathcal{B}i}\mathbf{D}^{-1}_{\mathcal{B}i}\mathbf{U}^{T}_{\mathcal{B}i}$
13	$\mathbb{M}^{A}_{\mathcal{B}\lambda(i)} = \mathbb{M}^{A}_{\mathcal{B}\lambda(i)} + {}_{\lambda(i)}\mathbf{X}^{i}\mathbb{M}^{a}_{\mathcal{B}i}{}^{i}\mathbf{X}_{\lambda(i)}$
14	**end**
15	$\mathcal{P}_0 = -(\mathbb{M}^{A}_{\mathcal{B}0})^{-1}\mathcal{F}_0$
16	**for** $i = 1$ **to** n_B **do**
17	$\mathbf{M}^{inv}[i, :] = \mathbf{M}^{inv}[i, :] - \mathbf{D}^{-1}_{\mathcal{B}i}\mathbf{U}^{T}_{\mathcal{B}i}{}^{i}\mathbf{X}_{\lambda(i)}\mathcal{P}_{\lambda(i)}$
18	$\mathcal{P}_i = \boldsymbol{\Gamma}_{\mathcal{J}i}\mathbf{M}^{inv}[i, :] + {}^{i}\mathbf{X}_{\lambda(i)}\mathcal{P}_{\lambda(i)}$
19	**end**
20	**for** $i = 1$ **to** n_B **do**
21	**for** $j = i$ **to** n_B **do**
22	$\mathbf{M}^{inv}[j, i] = \mathbf{M}^{inv}[i, j]$
23	**end**
24	**end**
25	$\mathbf{M}^{inv}[1{:}6, 1{:}6] = (\mathbb{M}^{A}_{\mathcal{B}0})^{-1}$
Outputs	$\mathbf{M}^{inv}$

4.5 Summary

In this chapter we have presented a numerically efficient way to compute the left-trivialized linearization of the forward dynamics of moving-base multibody systems, making use of the relation between the left-trivialized derivatives of the forward dynamics and the left-trivialized derivatives of the extended inverse dynamics, detailed in Section 3.3. We have derived the Extended Inverse Dynamics Algorithm for moving-base systems (EIDAmb) which efficiently computes the extended inverse dynamics, presented in Table 4.1. By applying recursive analytical derivatives to the EIDAmb, we have derived four algorithms that efficiently compute the left-trivialized derivatives of the extended inverse dynamics with respect to the four system state variables in Tables 4.2-4.5. We have presented a corrected version of the Inverse Mass Matrix Algorithm for fixed-base systems (IMMA) in Table 4.6 which computes the inverse mass matrix for fixed-base systems. Finally we have extended the IMMA to compute the inverse mass matrix of moving-base systems, which we call the Inverse Mass Matrix Algorithm for moving-base systems (IMMAmb) in Table 4.7. This concludes the second objective: *Derive numerically efficient and accurate algorithms to compute the singularity-free geometric linearization of moving-base multibody systems*, as described in Section 1.3.

The following chapter verifies that our newly presented algorithms computing the singularity-free linearization of the forward dynamics provide correct results, by comparing the results to the singularity-free linearization obtained by the (numerically less efficient) method of finite differences.

Chapter 5

Numerical Verification

The previous chapter presented recursive algorithms to compute the left-trivialized linearization matrices (3.42) and (3.29) of the forward dynamics (3.17) in a numerically efficient way. In total, six algorithms are required to compute this linearization: one to compute the extended inverse dynamics (3.39) in Table 4.1, four for the derivatives of the extended inverse dynamics with respect to the state parameters $\mathbf{H}, \mathbf{s}, \mathbf{v}$ and $\mathbf{r}$ in Tables 4.2-4.5, and one to compute the inverse of the mass matrix $\mathbf{M}^{inv}$ in Table 4.7. The correctness of these algorithms can be verified, since there exist other (numerically less efficient) methods to compute the same quantities. This chapter presents the used verification methods, as well as the numerical results.

5.1 Designed moving-base multibody test system

To verify the algorithms, an example of a moving-base multibody system has been designed on which the algorithms are tested. The algorithms must provide correct results for all moving-base multibody systems following the assumptions that all bodies are rigid (Assumption 2.1) and all joints are conventional and modelled as 1-DoF joints (Assumption 2.2). Three criteria for the designed system have been selected:

- The designed system must contain at least one of each type of joint defined in Appendix B.2 in each direction (x, y, z).
- The designed system must contain at least one branch, since the algorithms contain several lines of code that only apply to branched systems.
- The designed system must contain at least one branch directly at the base (so body 0 must have multiple children), since the algorithms contains a few lines of code that only apply to branches at the base.

The designed test system has therefore been chosen to have nine joints, as there are three joint types with each three directions, a branch at the base, and a branch that is not at the base. The body numbering is depicted in Figure 5.1. The joint types are given in Appendix D.1. A 3D visualization that resembles our designed model is depicted in Figure 5.2, where the yellow ball is the moving base, the red bodies have a prismatic joint, the green bodies have a revolute joint, and the blue bodies have a helical joint (for joint types, see Appendix B.2). The left image shows the system in a state where all joints positions are equal to 0, and the right image shows the system in a state where all joint positions are equal to 0.3, to visualize the movement of the joints.

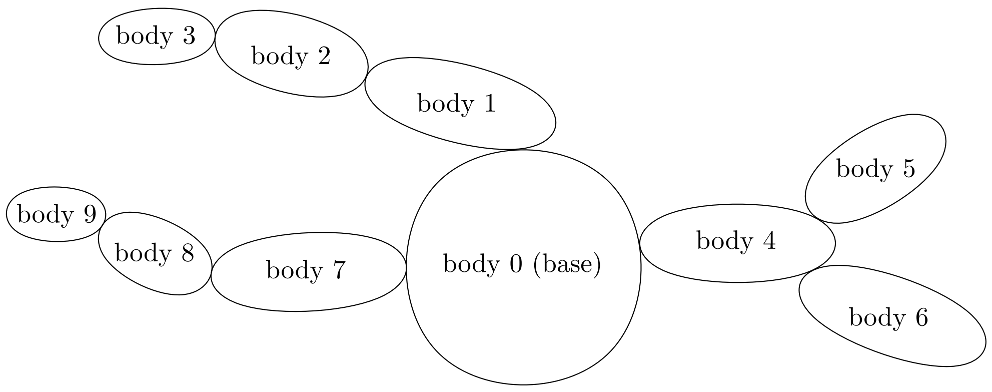
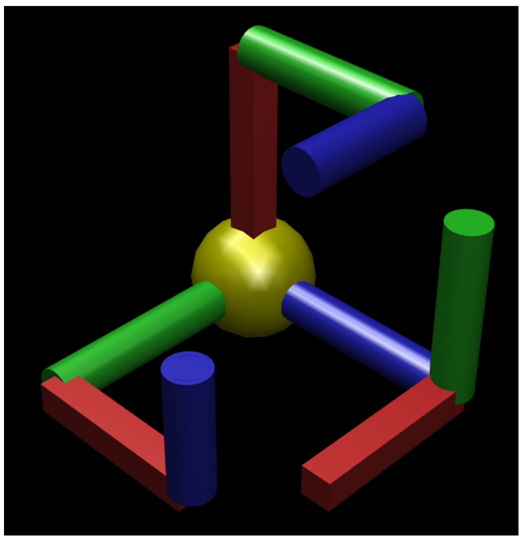
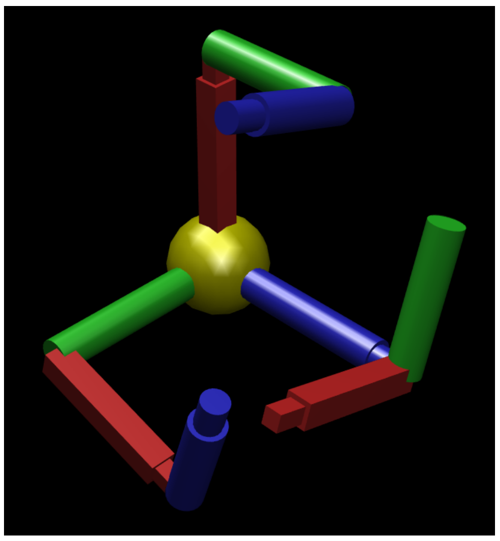


Figure 5.1: The body numbering of the designed test system.





(a) State where all joint position are equal to 0. (b) State where all joint position are equal to 0.3.

Figure 5.2: Two different states of our designed model.

To verify the correctness of our newly presented algorithms, we performed a randomized test using the design test model. This test is ran 100 times, where the inputs ${}^{A}\mathbf{H}_0, \mathbf{s}, {}^{A}\mathbf{v}_{A,0}, \mathbf{r}, \dot{\mathbf{r}}$ are randomly selected, and the input ${}^{0}\dot{\mathbf{v}}_{A,0}$ is computed through the RNEAmb (see Appendix E.2). Furthermore, the following model parameters are also randomly selected: the velocity transformation matrices ${}^{\lambda(i)|i}\mathbf{X}_{\lambda(i)}$ (which are body fixed constants), and the rigid body inertia's $\mathbb{M}_{\mathcal{B}i}$.

Besides the randomized test, we selected one set of inputs and model parameters in order to present the resulting left-trivialized derivatives and inverse mass matrices. The selected velocity transformation matrices ${}^{\lambda(i)|i}\mathbf{X}_{\lambda(i)}$ and the rigid body inertia's $\mathbb{M}_{\mathcal{B}i}$ are given in Appendix D.1 for all $i \in n_{\mathcal{B}}$. The selected set of inputs are shown below.

- ${}^{A}\mathbf{H}_0 = \begin{bmatrix} {}^{A}\mathbf{R}_0 & {}^{A}\mathbf{p}_0 \\ 0_{1\times 3} & 1 \end{bmatrix}$ as in (2.24), with
  - ${}^{A}\mathbf{R}_0 = \mathbf{R}^z(\theta_1)\,\mathbf{R}^y(\theta_2)\,\mathbf{R}^x(\theta_3)$, where $\mathbf{R}^x, \mathbf{R}^y$ and $\mathbf{R}^z$ are as defined in (B.2)-(B.4), $\theta_1 = 1, \theta_2 = 2, \theta_3 = 3$, so that ${}^{A}\mathbf{R}_0 \approx \begin{bmatrix} -0.2248 & 0.9024 & -0.3676 \\ -0.3502 & -0.4269 & -0.8337 \\ -0.9093 & -0.0587 & 0.4120 \end{bmatrix}$.

  - ${}^{A}\mathbf{p}_0 = [4; 5; 6]$.

- $\mathbf{s} = [13; 14; 15; 16; 17; 18; 19; 20; 21]$.
- ${}^{A}\mathbf{v}_{A,0} = [7; 8; 9; 10; 11; 12]$.
- $\mathbf{r} = [21; 22; 23; 24; 25; 26; 27; 28; 29]$.
- $\dot{\mathbf{r}} = [30; 31; 32; 33; 34; 35; 36; 37; 38]$.
- ${}^{0}\dot{\mathbf{v}}_{A,0}$ is found by the RNEAmb (Appendix E.2) with the above inputs, so that ${}^{0}\dot{\mathbf{v}}_{A,0} \approx [5657.9; -4306.7; -3855.9; 77.6085; 301.2763; -26.6843]$.

The following sections first present the results of the single test with the inputs and model parameters selected as above, to give insights in the resulting left-trivialized derivatives and inverse mass matrices. Afterwards, errors using the randomized test are presented.

5.2 Left-trivialized derivatives of the extended inverse dynamics algorithms

The left-trivialized derivatives algorithms presented in Section 4.2 compute the left-trivialized derivatives of the extended inverse dynamics of 3.39 in a numerically efficient way. These derivatives can also be approximated through a less numerically efficient method using finite differences. With finite differences, the extended inverse dynamics are evaluated once with unperturbed system states, and n times with one system state (where n is the number of degrees of freedom). We define the perturbation scalar $\delta \in \mathbb{R}$ as a small perturbation (e.g. $\delta = 10^{-9}$). The system is perturbed once for each degree of freedom through perturbation vectors. We distinguish two perturbation vectors: $\Delta_i^b \in \mathbb{R}^6$ for the moving base and $\Delta_k^j \in \mathbb{R}^{n_J}$ for the joints, where $i \in \{1...6\}$ and $k \in \{1...n_J\}$. These perturbation vectors are defined as zero-vectors, except the i-th or k-th index, which is equal to the perturbation scalar $\delta \in \mathbb{R}$. E.g. $\Delta_2^b := [0; \delta; 0; 0; 0; 0]$.

The perturbed extended inverse dynamics of (3.39) in the directions $\mathbf{s}, \mathbf{v}$ and $\mathbf{r}$ are given by

$$\overline{ID}(\mathbf{H}, \mathbf{s} + \Delta_k^j, \mathbf{v}, \mathbf{r}, \dot{\mathbf{v}}, \dot{\mathbf{r}}), \tag{5.1}$$

$$\overline{ID}(\mathbf{H}, \mathbf{s}, \mathbf{v} + \Delta_i^b, \mathbf{r}, \dot{\mathbf{v}}, \dot{\mathbf{r}}), \tag{5.2}$$

$$\overline{ID}(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r} + \Delta_k^j, \dot{\mathbf{v}}, \dot{\mathbf{r}}). \tag{5.3}$$

To obtain the perturbed extended inverse dynamics of (3.39) in the direction $\mathbf{H}$, we need to morph the perturbation $\Delta_i^b \in \mathbb{R}^6$ into the set $\mathfrak{se}(3)$ by the wedge operator: $(\Delta_i^b)^\wedge \in \mathfrak{se}(3)$. The exponential map $\exp : \mathfrak{se}(3) \to SE(3)$ can bring the perturbation to the set SE(3): $\exp((\Delta_i^b)^\wedge) \in \mathrm{SE}(3)$. Using the group operation of the Lie group SE(3) on $\mathbf{H}$ and $\exp((\Delta_i^b)^\wedge)$ results in the perturbed transformation matrix. We write the perturbed extended inverse dynamics of (3.39) in the direction $\mathbf{H}$ as

$$\overline{ID}(\mathbf{H}\exp((\Delta_i^b)^\wedge), \mathbf{s}, \mathbf{v}, \mathbf{r}, \dot{\mathbf{v}}, \dot{\mathbf{r}}) \tag{5.4}$$

The resulting left-trivialized derivatives are

$$[\mathrm{D}_1\,\overline{ID}(\mathbf{H},\mathbf{s},\mathbf{v},\mathbf{r},\dot{\mathbf{v}},\dot{\mathbf{r}})\circ \mathrm{D}\,L_H(I)]_i = \frac{\overline{ID}(\mathbf{H}\exp((\Delta_i^b)^\wedge),\mathbf{s},\mathbf{v},\mathbf{r},\dot{\mathbf{v}},\dot{\mathbf{r}}) - \overline{ID}(\mathbf{H},\mathbf{s},\mathbf{v},\mathbf{r},\dot{\mathbf{v}},\dot{\mathbf{r}})}{\delta}, \tag{5.5}$$

$$[\mathrm{D}_2\,\overline{ID}(\mathbf{H},\mathbf{s},\mathbf{v},\mathbf{r},\dot{\mathbf{v}},\dot{\mathbf{r}})]_k = \frac{\overline{ID}(\mathbf{H},\mathbf{s}+\Delta_k^j,\mathbf{v},\mathbf{r},\dot{\mathbf{v}},\dot{\mathbf{r}}) - \overline{ID}(\mathbf{H},\mathbf{s},\mathbf{v},\mathbf{r},\dot{\mathbf{v}},\dot{\mathbf{r}})}{\delta}, \tag{5.6}$$

$$[\mathrm{D}_3\,\overline{ID}(\mathbf{H},\mathbf{s},\mathbf{v},\mathbf{r},\dot{\mathbf{v}},\dot{\mathbf{r}})]_i = \frac{\overline{ID}(\mathbf{H},\mathbf{s},\mathbf{v}+\Delta_i^b,\mathbf{r},\dot{\mathbf{v}},\dot{\mathbf{r}}) - \overline{ID}(\mathbf{H},\mathbf{s},\mathbf{v},\mathbf{r},\dot{\mathbf{v}},\dot{\mathbf{r}})}{\delta}, \tag{5.7}$$

$$[\mathrm{D}_4\,\overline{ID}(\mathbf{H},\mathbf{s},\mathbf{v},\mathbf{r},\dot{\mathbf{v}},\dot{\mathbf{r}})]_k = \frac{\overline{ID}(\mathbf{H},\mathbf{s},\mathbf{v},\mathbf{r}+\Delta_k^j,\dot{\mathbf{v}},\dot{\mathbf{r}}) - \overline{ID}(\mathbf{H},\mathbf{s},\mathbf{v},\mathbf{r},\dot{\mathbf{v}},\dot{\mathbf{r}})}{\delta}, \tag{5.8}$$

where $i \in \{1...6\}$ and $k \in \{1...n_J\}$ and $[X]_i$ is the i-th column of matrix X. These left-trivialized derivatives can be approximated by simply evaluating the EIDAmb once unperturbed, and once for each perturbation.

The following subsections present the numerical results for both the derivatives obtained through our algorithms presented in Section 4.2 as well as the derivatives obtained through finite differences, where we use the designed test system and the selected input variables presented in Section 5.1. We choose $\delta = 10^{-6}$, which may have effect on the errors presented in this chapter, but not on the drawn conclusions that our newly presented algorithms return correct values.

5.2.1 Derivatives of extended inverse dynamics w.r.t. transformation matrix

The left-trivialized derivative of the extended inverse dynamics of (3.39) with respect to the transformation matrix $\mathbf{H}$ obtained through the finite differences method using the selected inputs is found as

$$\mathrm{D}_1\,\overline{ID}^{FinDiff} \approx \begin{pmatrix} -4.88\mathrm{E}5 & -4.01\mathrm{E}4 & 1.03\mathrm{E}6 & 7.58\mathrm{E}6 & 8.37\mathrm{E}6 & -1.1\mathrm{E}7 \\ -4.85\mathrm{E}5 & -1.58\mathrm{E}6 & 3.2\mathrm{E}5 & -8.59\mathrm{E}6 & 1.01\mathrm{E}7 & 2.18\mathrm{E}7 \\ 3.54\mathrm{E}5 & 3.89\mathrm{E}4 & -7.43\mathrm{E}5 & -7.9\mathrm{E}6 & -1.59\mathrm{E}7 & 8.68\mathrm{E}6 \\ 5.03\mathrm{E}4 & -2.26\mathrm{E}5 & -2.12\mathrm{E}5 & -3.12\mathrm{E}8 & -3.37\mathrm{E}8 & 5.17\mathrm{E}8 \\ 8.47\mathrm{E}6 & -2.64\mathrm{E}6 & -1.94\mathrm{E}7 & -2.56\mathrm{E}8 & -2.58\mathrm{E}9 & -6.65\mathrm{E}8 \\ 2.24\mathrm{E}6 & 2.45\mathrm{E}7 & 6.41\mathrm{E}6 & 1.01\mathrm{E}9 & 5.94\mathrm{E}8 & -1.85\mathrm{E}9 \\ -3.04\mathrm{E}5 & -2.05\mathrm{E}5 & 5.61\mathrm{E}5 & -2.63\mathrm{E}6 & -2.29\mathrm{E}7 & -4.27\mathrm{E}6 \\ 1.25\mathrm{E}7 & 2.47\mathrm{E}7 & -1.55\mathrm{E}7 & -3.59\mathrm{E}8 & 1.88\mathrm{E}7 & 7.87\mathrm{E}8 \\ -1.79\mathrm{E}6 & -3.71\mathrm{E}6 & 2.15\mathrm{E}6 & 6.55\mathrm{E}7 & -3.54\mathrm{E}7 & -1.58\mathrm{E}8 \\ -1.51\mathrm{E}5 & 7.34\mathrm{E}5 & 6.6\mathrm{E}5 & 3.91\mathrm{E}7 & 1.74\mathrm{E}7 & -7.43\mathrm{E}7 \\ -5.5\mathrm{E}5 & -5.63\mathrm{E}5 & 9.26\mathrm{E}5 & -5.77\mathrm{E}5 & 9.32\mathrm{E}7 & 4.47\mathrm{E}7 \\ -3.51\mathrm{E}4 & -7.28\mathrm{E}4 & 4.22\mathrm{E}4 & -2.35\mathrm{E}6 & 4.77\mathrm{E}6 & 7.23\mathrm{E}6 \\ -4.91\mathrm{E}6 & -9.57\mathrm{E}6 & 6.18\mathrm{E}6 & 2.65\mathrm{E}7 & 1.39\mathrm{E}8 & 5.06\mathrm{E}6 \\ 1.9\mathrm{E}5 & 1.11\mathrm{E}5 & -3.58\mathrm{E}5 & -4.14\mathrm{E}6 & -1.81\mathrm{E}6 & 7.67\mathrm{E}6 \\ 2.84\mathrm{E}5 & 1.95\mathrm{E}5 & -5.23\mathrm{E}5 & 2.22\mathrm{E}6 & 6.95\mathrm{E}6 & -2.15\mathrm{E}6 \end{pmatrix},$$

and the same derivative obtained through the algorithm in Table 4.2 is found as

$$\mathrm{D}_1\,\overline{ID}^{Alg} \approx \begin{pmatrix} -4.88\mathrm{E}5 & -4.01\mathrm{E}4 & 1.03\mathrm{E}6 & 7.58\mathrm{E}6 & 8.37\mathrm{E}6 & -1.1\mathrm{E}7 \\ -4.85\mathrm{E}5 & -1.58\mathrm{E}6 & 3.2\mathrm{E}5 & -8.59\mathrm{E}6 & 1.01\mathrm{E}7 & 2.18\mathrm{E}7 \\ 3.54\mathrm{E}5 & 3.89\mathrm{E}4 & -7.43\mathrm{E}5 & -7.9\mathrm{E}6 & -1.59\mathrm{E}7 & 8.68\mathrm{E}6 \\ 5.03\mathrm{E}4 & -2.26\mathrm{E}5 & -2.12\mathrm{E}5 & -3.12\mathrm{E}8 & -3.37\mathrm{E}8 & 5.17\mathrm{E}8 \\ 8.47\mathrm{E}6 & -2.64\mathrm{E}6 & -1.94\mathrm{E}7 & -2.56\mathrm{E}8 & -2.58\mathrm{E}9 & -6.65\mathrm{E}8 \\ 2.24\mathrm{E}6 & 2.45\mathrm{E}7 & 6.41\mathrm{E}6 & 1.01\mathrm{E}9 & 5.94\mathrm{E}8 & -1.85\mathrm{E}9 \\ -3.04\mathrm{E}5 & -2.05\mathrm{E}5 & 5.61\mathrm{E}5 & -2.63\mathrm{E}6 & -2.29\mathrm{E}7 & -4.27\mathrm{E}6 \\ 1.25\mathrm{E}7 & 2.47\mathrm{E}7 & -1.55\mathrm{E}7 & -3.59\mathrm{E}8 & 1.88\mathrm{E}7 & 7.87\mathrm{E}8 \\ -1.79\mathrm{E}6 & -3.71\mathrm{E}6 & 2.15\mathrm{E}6 & 6.55\mathrm{E}7 & -3.54\mathrm{E}7 & -1.58\mathrm{E}8 \\ -1.51\mathrm{E}5 & 7.34\mathrm{E}5 & 6.6\mathrm{E}5 & 3.91\mathrm{E}7 & 1.74\mathrm{E}7 & -7.43\mathrm{E}7 \\ -5.5\mathrm{E}5 & -5.63\mathrm{E}5 & 9.26\mathrm{E}5 & -5.76\mathrm{E}5 & 9.32\mathrm{E}7 & 4.47\mathrm{E}7 \\ -3.51\mathrm{E}4 & -7.28\mathrm{E}4 & 4.22\mathrm{E}4 & -2.35\mathrm{E}6 & 4.77\mathrm{E}6 & 7.23\mathrm{E}6 \\ -4.91\mathrm{E}6 & -9.57\mathrm{E}6 & 6.18\mathrm{E}6 & 2.65\mathrm{E}7 & 1.39\mathrm{E}8 & 5.06\mathrm{E}6 \\ 1.9\mathrm{E}5 & 1.11\mathrm{E}5 & -3.58\mathrm{E}5 & -4.14\mathrm{E}6 & -1.81\mathrm{E}6 & 7.67\mathrm{E}6 \\ 2.84\mathrm{E}5 & 1.95\mathrm{E}5 & -5.23\mathrm{E}5 & 2.22\mathrm{E}6 & 6.95\mathrm{E}6 & -2.15\mathrm{E}6 \end{pmatrix}.$$

We define a maximum error and an average error between both derivatives and compute them as

$$e^{max} := max \left| \frac{\mathrm{D}_1\,\overline{ID}^{Alg} - \mathrm{D}_1\,\overline{ID}^{FinDiff}}{avg|\,\mathrm{D}_1\,\overline{ID}^{Alg}|} \right| \approx 6.4985\mathrm{E}{-6} \tag{5.9}$$

and

$$e^{avg} := avg \left| \frac{\mathrm{D}_1\,\overline{ID}^{Alg} - \mathrm{D}_1\,\overline{ID}^{FinDiff}}{avg|\,\mathrm{D}_1\,\overline{ID}^{Alg}|} \right| \approx 4.9484\mathrm{E}{-7}. \tag{5.10}$$

We choose to use the absolute value of the difference between the algorithmic derivative and the finite difference derivative, which ensures that negative differences are taken into account properly. The division by the average of the algorithmic derivative is preferred over element-wise division, as some element may be equal to zero, although finite differences compute them as non-zero (e.g. 10^{-10}) which would result in an infinitely large error. To be more precise, we use the average of the absolute value of the algorithmic derivative, to ensure that positive and negative values in the derivative do not cancel out.

Using the randomized test, where we compute the derivative for the randomized inputs and system parameters 100 times, we obtain the highest maximum error and the highest average error as

$$e^{max}_{rand} \approx 8.0093\mathrm{E}{-6} \qquad \text{and} \qquad e^{avg}_{rand} \approx 7.1026\mathrm{E}{-7}. \tag{5.11}$$

These results verify that (apart from approximation errors in the finite differences) the algorithm presented in Table 4.2 provides correct results.

5.2.2 Derivatives of extended inverse dynamics w.r.t. generalized position vector

The derivative of the extended inverse dynamics of (3.39) with respect to the generalized position vector **s** obtained through the finite differences method using the selected inputs is found as

$$\mathrm{D}_2\,\overline{ID}^{FinDiff} \approx$$
$$\begin{pmatrix}
6.74\mathrm{E}4 & 1.71\mathrm{E}7 & -1.57\mathrm{E}7 & 4.65\mathrm{E}5 & -1.17\mathrm{E}6 & 2633.0 & 1.92\mathrm{E}6 & -4799.0 & 6.83\mathrm{E}5 \\
3.42\mathrm{E}5 & 2.72\mathrm{E}7 & -8.89\mathrm{E}6 & 3.61\mathrm{E}6 & -1.79\mathrm{E}6 & 1.86\mathrm{E}4 & -7.86\mathrm{E}5 & 1.89\mathrm{E}5 & -7.43\mathrm{E}5 \\
2.94\mathrm{E}5 & -4.2\mathrm{E}6 & -1.52\mathrm{E}6 & -2.5\mathrm{E}6 & 1.83\mathrm{E}6 & -1.31\mathrm{E}4 & 3.13\mathrm{E}6 & 1.55\mathrm{E}5 & 2.76\mathrm{E}5 \\
-4.65\mathrm{E}6 & -8.91\mathrm{E}8 & 5.22\mathrm{E}8 & -1.54\mathrm{E}7 & -2.86\mathrm{E}7 & -2.04\mathrm{E}6 & 1.17\mathrm{E}7 & 6.65\mathrm{E}6 & -1.87\mathrm{E}7 \\
-1.78\mathrm{E}6 & 9.74\mathrm{E}8 & -7.63\mathrm{E}8 & -2.51\mathrm{E}8 & 1.81\mathrm{E}8 & -5.1\mathrm{E}5 & 1.16\mathrm{E}7 & -2.26\mathrm{E}6 & -5.23\mathrm{E}6 \\
-1.64\mathrm{E}6 & -3.05\mathrm{E}8 & -4.27\mathrm{E}7 & -2.97\mathrm{E}8 & 1.47\mathrm{E}8 & -2.2\mathrm{E}6 & 2.46\mathrm{E}7 & -2.69\mathrm{E}6 & 3.88\mathrm{E}7 \\
6.74\mathrm{E}4 & 1.71\mathrm{E}7 & -1.57\mathrm{E}7 & 0 & 0 & 0 & 0 & 0 & 0 \\
-6.93\mathrm{E}6 & -1.54\mathrm{E}9 & 9.58\mathrm{E}8 & 0 & 0 & 0 & 0 & 0 & 0 \\
1.41\mathrm{E}6 & 2.53\mathrm{E}8 & -1.98\mathrm{E}8 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -3.84\mathrm{E}7 & -2.19\mathrm{E}7 & -2.15\mathrm{E}6 & 0 & 0 & 0 \\
0 & 0 & 0 & 5.93\mathrm{E}6 & 8.74\mathrm{E}5 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 3.89\mathrm{E}6 & 0 & 2.02\mathrm{E}4 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 2.27\mathrm{E}7 & 6.74\mathrm{E}6 & -1.64\mathrm{E}7 \\
0 & 0 & 0 & 0 & 0 & 0 & -1.01\mathrm{E}6 & 1.25\mathrm{E}5 & 3.85\mathrm{E}5 \\
0 & 0 & 0 & 0 & 0 & 0 & 2.99\mathrm{E}5 & -3.14\mathrm{E}5 & 1.05\mathrm{E}7
\end{pmatrix},$$

and the same derivative obtained through the algorithm in Table 4.3 is found as

$$\mathrm{D}_2\,\overline{ID}^{Alg} \approx$$
$$\begin{pmatrix}
6.74\mathrm{E}4 & 1.71\mathrm{E}7 & -1.57\mathrm{E}7 & 4.65\mathrm{E}5 & -1.17\mathrm{E}6 & 2633.0 & 1.92\mathrm{E}6 & -4799.0 & 6.83\mathrm{E}5 \\
3.42\mathrm{E}5 & 2.72\mathrm{E}7 & -8.89\mathrm{E}6 & 3.61\mathrm{E}6 & -1.79\mathrm{E}6 & 1.86\mathrm{E}4 & -7.86\mathrm{E}5 & 1.89\mathrm{E}5 & -7.43\mathrm{E}5 \\
2.94\mathrm{E}5 & -4.2\mathrm{E}6 & -1.52\mathrm{E}6 & -2.5\mathrm{E}6 & 1.83\mathrm{E}6 & -1.31\mathrm{E}4 & 3.13\mathrm{E}6 & 1.55\mathrm{E}5 & 2.76\mathrm{E}5 \\
-4.65\mathrm{E}6 & -8.91\mathrm{E}8 & 5.22\mathrm{E}8 & -1.54\mathrm{E}7 & -2.86\mathrm{E}7 & -2.04\mathrm{E}6 & 1.17\mathrm{E}7 & 6.65\mathrm{E}6 & -1.87\mathrm{E}7 \\
-1.78\mathrm{E}6 & 9.74\mathrm{E}8 & -7.63\mathrm{E}8 & -2.51\mathrm{E}8 & 1.81\mathrm{E}8 & -5.1\mathrm{E}5 & 1.16\mathrm{E}7 & -2.26\mathrm{E}6 & -5.23\mathrm{E}6 \\
-1.64\mathrm{E}6 & -3.05\mathrm{E}8 & -4.27\mathrm{E}7 & -2.97\mathrm{E}8 & 1.47\mathrm{E}8 & -2.2\mathrm{E}6 & 2.46\mathrm{E}7 & -2.69\mathrm{E}6 & 3.88\mathrm{E}7 \\
6.74\mathrm{E}4 & 1.71\mathrm{E}7 & -1.57\mathrm{E}7 & 0 & 0 & 0 & 0 & 0 & 0 \\
-6.93\mathrm{E}6 & -1.54\mathrm{E}9 & 9.58\mathrm{E}8 & 0 & 0 & 0 & 0 & 0 & 0 \\
1.41\mathrm{E}6 & 2.53\mathrm{E}8 & -1.98\mathrm{E}8 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -3.84\mathrm{E}7 & -2.19\mathrm{E}7 & -2.15\mathrm{E}6 & 0 & 0 & 0 \\
0 & 0 & 0 & 5.93\mathrm{E}6 & 8.74\mathrm{E}5 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 3.89\mathrm{E}6 & 0 & 2.02\mathrm{E}4 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 2.27\mathrm{E}7 & 6.74\mathrm{E}6 & -1.64\mathrm{E}7 \\
0 & 0 & 0 & 0 & 0 & 0 & -1.01\mathrm{E}6 & 1.25\mathrm{E}5 & 3.85\mathrm{E}5 \\
0 & 0 & 0 & 0 & 0 & 0 & 2.99\mathrm{E}5 & -3.14\mathrm{E}5 & 1.05\mathrm{E}7
\end{pmatrix}.$$

We define the maximum and average errors between both derivatives similar to (5.9) and (5.10) and compute them as

$$e^{max} := max \left| \frac{\mathrm{D}_2\,\overline{ID}^{Alg} - \mathrm{D}_2\,\overline{ID}^{FinDiff}}{avg|\,\mathrm{D}_2\,\overline{ID}^{Alg}|} \right| \approx 1.2275\mathrm{E}{-5} \tag{5.12}$$

and

$$e^{avg} := avg \left| \frac{\mathrm{D}_2\,\overline{ID}^{Alg} - \mathrm{D}_2\,\overline{ID}^{FinDiff}}{avg|\,\mathrm{D}_2\,\overline{ID}^{Alg}|} \right| \approx 6.6522\mathrm{E}{-7}. \tag{5.13}$$

Using the randomized test, where we compute the derivative for the randomized inputs and system parameters 100 times, we obtain the highest maximum error and the highest average error as

$$e^{max}_{rand} \approx 5.6673\mathrm{E}{-5} \qquad \text{and} \qquad e^{avg}_{rand} \approx 1.7340\mathrm{E}{-6}. \tag{5.14}$$

These results verify that (apart from approximation errors in the finite differences) the algorithm presented in Table 4.3 provides correct results.

5.2.3 Derivatives of extended inverse dynamics w.r.t. moving-base velocity

The derivative of the extended inverse dynamics of (3.39) with respect to the moving-base velocity $\mathbf{v}$ obtained through the finite differences method using the selected inputs is found as

$$\mathrm{D}_3\,\overline{ID}^{FinDiff} \approx \begin{pmatrix} -13400.0 & 63544.0 & -3867.0 & -40822.0 & -6.841\text{E}5 & -5.033\text{E}5 \\ -10099.0 & 20966.0 & 88077.0 & -1.104\text{E}6 & 1.503\text{E}6 & -1.183\text{E}6 \\ -11366.0 & -41299.0 & -7559.0 & -5.342\text{E}5 & 6.695\text{E}5 & 6.932\text{E}5 \\ -1.25\text{E}5 & 14188.0 & -44799.0 & 1.993\text{E}7 & 2.623\text{E}7 & 2.914\text{E}7 \\ -1.438\text{E}6 & -8.37\text{E}5 & -5.129\text{E}5 & 1.632\text{E}7 & -4.077\text{E}7 & 1.603\text{E}8 \\ 5.312\text{E}5 & 2.635\text{E}5 & -1.194\text{E}6 & 1.004\text{E}7 & -1.117\text{E}8 & -2.933\text{E}7 \\ -5732.0 & 34200.0 & 9380.0 & -2.898\text{E}5 & -2.153\text{E}5 & 1.208\text{E}6 \\ 3.316\text{E}5 & -9.822\text{E}5 & -1.297\text{E}6 & 4.014\text{E}7 & 3.835\text{E}7 & 1.859\text{E}7 \\ -91866.0 & 1.462\text{E}5 & 1.757\text{E}5 & -4.78\text{E}6 & -8.357\text{E}6 & -2.723\text{E}5 \\ 56011.0 & 26900.0 & -17122.0 & -2.672\text{E}6 & -3.865\text{E}6 & -2.262\text{E}6 \\ -13600.0 & 57344.0 & 26777.0 & 87433.0 & 2.568\text{E}6 & -5.46\text{E}6 \\ -76.38 & 2496.0 & 4246.0 & -8692.0 & 4.243\text{E}5 & -2.879\text{E}5 \\ -33699.0 & 3.704\text{E}5 & 5.468\text{E}5 & -9.119\text{E}6 & 1.92\text{E}6 & -1.289\text{E}7 \\ -5896.0 & -19800.0 & -9245.0 & 6.628\text{E}5 & 3.319\text{E}5 & 4.236\text{E}5 \\ -10466.0 & -28511.0 & -16300.0 & 4.026\text{E}5 & -1.789\text{E}5 & -2.037\text{E}5 \end{pmatrix},$$

and the same derivative obtained through the algorithm in Table 4.4 is found as

$$\mathrm{D}_3\,\overline{ID}^{Alg} \approx \begin{pmatrix} -13400.0 & 63544.0 & -3867.0 & -40822.0 & -6.841\text{E}5 & -5.033\text{E}5 \\ -10099.0 & 20966.0 & 88077.0 & -1.104\text{E}6 & 1.503\text{E}6 & -1.183\text{E}6 \\ -11366.0 & -41299.0 & -7559.0 & -5.342\text{E}5 & 6.695\text{E}5 & 6.932\text{E}5 \\ -1.25\text{E}5 & 14188.0 & -44799.0 & 1.993\text{E}7 & 2.623\text{E}7 & 2.914\text{E}7 \\ -1.438\text{E}6 & -8.37\text{E}5 & -5.129\text{E}5 & 1.632\text{E}7 & -4.077\text{E}7 & 1.603\text{E}8 \\ 5.312\text{E}5 & 2.635\text{E}5 & -1.194\text{E}6 & 1.004\text{E}7 & -1.117\text{E}8 & -2.933\text{E}7 \\ -5732.0 & 34200.0 & 9380.0 & -2.898\text{E}5 & -2.153\text{E}5 & 1.208\text{E}6 \\ 3.316\text{E}5 & -9.822\text{E}5 & -1.297\text{E}6 & 4.014\text{E}7 & 3.835\text{E}7 & 1.859\text{E}7 \\ -91866.0 & 1.462\text{E}5 & 1.757\text{E}5 & -4.78\text{E}6 & -8.357\text{E}6 & -2.723\text{E}5 \\ 56011.0 & 26900.0 & -17122.0 & -2.672\text{E}6 & -3.865\text{E}6 & -2.262\text{E}6 \\ -13600.0 & 57344.0 & 26777.0 & 87433.0 & 2.568\text{E}6 & -5.46\text{E}6 \\ -76.38 & 2496.0 & 4246.0 & -8692.0 & 4.243\text{E}5 & -2.879\text{E}5 \\ -33699.0 & 3.704\text{E}5 & 5.468\text{E}5 & -9.119\text{E}6 & 1.92\text{E}6 & -1.289\text{E}7 \\ -5896.0 & -19800.0 & -9245.0 & 6.628\text{E}5 & 3.319\text{E}5 & 4.236\text{E}5 \\ -10466.0 & -28511.0 & -16300.0 & 4.026\text{E}5 & -1.789\text{E}5 & -2.037\text{E}5 \end{pmatrix}.$$

We define the maximum and average errors between both derivatives similar to (5.9) and (5.10) and compute them as

$$e^{max} := max \left| \frac{\mathrm{D}_3\,\overline{ID}^{Alg} - \mathrm{D}_3\,\overline{ID}^{FinDiff}}{avg|\,\mathrm{D}_3\,\overline{ID}^{Alg}|} \right| \approx 2.6282\text{E}{-7} \tag{5.15}$$

and

$$e^{avg} := avg \left| \frac{\mathrm{D}_3\,\overline{ID}^{Alg} - \mathrm{D}_3\,\overline{ID}^{FinDiff}}{avg|\,\mathrm{D}_3\,\overline{ID}^{Alg}|} \right| \approx 2.0523\text{E}{-8}. \tag{5.16}$$

Using the randomized test, where we compute the derivative for the randomized inputs and system parameters 100 times, we obtain the highest maximum error and the highest average error as

$$e_{rand}^{max} \approx 8.7923\text{E}{-6} \qquad \text{and} \qquad e_{rand}^{avg} \approx 5.9019\text{E}{-7}. \tag{5.17}$$

These results verify that (apart from approximation errors in the finite differences) the algorithm presented in Table 4.4 provides correct results.

5.2.4 Derivatives of extended inverse dynamics w.r.t. generalized velocity vector

The derivative of the extended inverse dynamics of (3.39) with respect to the generalized velocity vector **r** obtained through the finite differences method using the selected inputs is found as

$$\mathrm{D}_4\,\overline{ID}^{FinDiff} \approx \begin{pmatrix}
-2744.0 & -1.99\text{E}6 & 1.98\text{E}5 & -7.66\text{E}4 & 1.36\text{E}4 & 1588.0 & 2.02\text{E}5 & 6466.0 & 2100.0 \\
-2.24\text{E}4 & -1.56\text{E}4 & -1.85\text{E}5 & 1.56\text{E}5 & -5.73\text{E}4 & 394.0 & 2.4\text{E}5 & -1.97\text{E}4 & 3.6\text{E}4 \\
-7100.0 & -1.77\text{E}5 & -1.85\text{E}5 & -5.82\text{E}4 & -2.66\text{E}4 & 2966.0 & 5.2\text{E}5 & -2866.0 & 2.14\text{E}4 \\
-2.27\text{E}4 & -2.35\text{E}7 & 9.35\text{E}6 & 1.33\text{E}6 & -8.71\text{E}4 & -1.03\text{E}5 & -4.25\text{E}6 & -4.81\text{E}5 & -4.98\text{E}5 \\
-1.01\text{E}6 & -6.85\text{E}7 & 1.17\text{E}7 & -3.7\text{E}6 & -2.55\text{E}6 & 2.6\text{E}5 & 2.61\text{E}6 & -2.76\text{E}5 & 5.41\text{E}5 \\
2.22\text{E}5 & 1.29\text{E}7 & -1.39\text{E}5 & -1.63\text{E}7 & 5.46\text{E}6 & 1.04\text{E}5 & -3.33\text{E}6 & 4.55\text{E}5 & -6.5\text{E}5 \\
-2744.0 & -1.99\text{E}6 & 1.98\text{E}5 & 0 & 0 & 0 & 0 & 0 & 0 \\
6.2\text{E}5 & 1.02\text{E}7 & 4.73\text{E}6 & 0 & 0 & 0 & 0 & 0 & 0 \\
-9.77\text{E}4 & -2.67\text{E}6 & -0.0149 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1.42\text{E}5 & 3.82\text{E}5 & -9.82\text{E}4 & 0 & 0 & 0 \\
0 & 0 & 0 & -3.86\text{E}5 & 0.00373 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 9.6\text{E}4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -3.17\text{E}6 & -4.5\text{E}5 & -5.06\text{E}5 \\
0 & 0 & 0 & 0 & 0 & 0 & 4.82\text{E}5 & 133.0 & 1.58\text{E}4 \\
0 & 0 & 0 & 0 & 0 & 0 & 4.61\text{E}5 & -1.47\text{E}4 & -0.00559
\end{pmatrix},$$

and the same derivative obtained through the algorithm in Table 4.3 is found as

$$\mathrm{D}_4\,\overline{ID}^{Alg} \approx \begin{pmatrix}
-2744.0 & -1.99\text{E}6 & 1.98\text{E}5 & -7.66\text{E}4 & 1.36\text{E}4 & 1588.0 & 2.02\text{E}5 & 6466.0 & 2100.0 \\
-2.24\text{E}4 & -1.56\text{E}4 & -1.85\text{E}5 & 1.56\text{E}5 & -5.73\text{E}4 & 394.0 & 2.4\text{E}5 & -1.97\text{E}4 & 3.6\text{E}4 \\
-7100.0 & -1.77\text{E}5 & -1.85\text{E}5 & -5.82\text{E}4 & -2.66\text{E}4 & 2966.0 & 5.2\text{E}5 & -2866.0 & 2.14\text{E}4 \\
-2.27\text{E}4 & -2.35\text{E}7 & 9.35\text{E}6 & 1.33\text{E}6 & -8.71\text{E}4 & -1.03\text{E}5 & -4.25\text{E}6 & -4.81\text{E}5 & -4.98\text{E}5 \\
-1.01\text{E}6 & -6.85\text{E}7 & 1.17\text{E}7 & -3.7\text{E}6 & -2.55\text{E}6 & 2.6\text{E}5 & 2.61\text{E}6 & -2.76\text{E}5 & 5.41\text{E}5 \\
2.22\text{E}5 & 1.29\text{E}7 & -1.39\text{E}5 & -1.63\text{E}7 & 5.46\text{E}6 & 1.04\text{E}5 & -3.33\text{E}6 & 4.55\text{E}5 & -6.5\text{E}5 \\
-2744.0 & -1.99\text{E}6 & 1.98\text{E}5 & 0 & 0 & 0 & 0 & 0 & 0 \\
6.2\text{E}5 & 1.02\text{E}7 & 4.73\text{E}6 & 0 & 0 & 0 & 0 & 0 & 0 \\
-9.77\text{E}4 & -2.67\text{E}6 & -2.91\text{E}{-11} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1.42\text{E}5 & 3.82\text{E}5 & -9.82\text{E}4 & 0 & 0 & 0 \\
0 & 0 & 0 & -3.86\text{E}5 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 9.6\text{E}4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -3.17\text{E}6 & -4.5\text{E}5 & -5.06\text{E}5 \\
0 & 0 & 0 & 0 & 0 & 0 & 4.82\text{E}5 & 133.0 & 1.58\text{E}4 \\
0 & 0 & 0 & 0 & 0 & 0 & 4.61\text{E}5 & -1.47\text{E}4 & -1.46\text{E}{-11}
\end{pmatrix}.$$

We define the maximum and average errors between both derivatives similar to (5.9) and (5.10) and compute them as

$$e^{max} := max \left| \frac{\mathrm{D}_4 \overline{ID}^{Alg} - \mathrm{D}_4 \overline{ID}^{FinDiff}}{avg|\, \mathrm{D}_4 \overline{ID}^{Alg}|} \right| \approx 4.2731\mathrm{E}{-7} \tag{5.18}$$

and

$$e^{avg} := avg \left| \frac{\mathrm{D}_4 \overline{ID}^{Alg} - \mathrm{D}_4 \overline{ID}^{FinDiff}}{avg|\, \mathrm{D}_4 \overline{ID}^{Alg}|} \right| \approx 3.2837\mathrm{E}{-8}. \tag{5.19}$$

Using the randomized test, where we compute the derivative for the randomized inputs and system parameters 100 times, we obtain the highest maximum error and the highest average error as

$$e^{max}_{rand} \approx 2.1186\mathrm{E}{-5} \qquad \text{and} \qquad e^{avg}_{rand} \approx 6.5810\mathrm{E}{-7}. \tag{5.20}$$

These results verify that (apart from approximation errors in the finite differences) the algorithm presented in Table 4.5 provides correct results.

5.3 Inverse Mass Matrix Algorithm for fixed-base systems

For fixed-base systems, the IMMA presented in Section 4.3 computes the inverse of the mass matrix $\mathbf{M}_{fb}$ in a numerically efficient way. The correctness of its results can be verified simply by computing the mass matrix through the CRBA (Appendix E.3) and inverting it. Both inverted mass matrices should be identical (apart from numerical rounding errors). As an example to apply these test methods on, we have chosen the designed test system and input variables presented in Section 5.1, where body 0 is now the fixed base instead of the moving base. As we claimed that the algorithm presented in [28] is incorrect, we compare both his version and our version to the inverted mass matrix found from the CRBA.

The inverse of the mass matrix $\mathbf{M}_{fb}^{-1}$ is found through the CRBA as

$$^{CRBA}\mathbf{M}_{fb}^{-1} \approx$$

$$\begin{pmatrix}
8.67\mathrm{E}{-4} & -1.48\mathrm{E}{-5} & -4.07\mathrm{E}{-4} & 0 & 0 & 0 & 0 & 0 & 0 \\
-1.48\mathrm{E}{-5} & 1.8\mathrm{E}{-6} & 4.19\mathrm{E}{-5} & 0 & 0 & 0 & 0 & 0 & 0 \\
-4.07\mathrm{E}{-4} & 4.19\mathrm{E}{-5} & 0.0012 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 7.11\mathrm{E}{-6} & 1.5\mathrm{E}{-4} & 3.13\mathrm{E}{-5} & 0 & 0 & 0 \\
0 & 0 & 0 & 1.5\mathrm{E}{-4} & 1.0 & 6.58\mathrm{E}{-4} & 0 & 0 & 0 \\
0 & 0 & 0 & 3.13\mathrm{E}{-5} & 6.58\mathrm{E}{-4} & 0.00449 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1.62\mathrm{E}{-6} & 1.85\mathrm{E}{-5} & -7.36\mathrm{E}{-5} \\
0 & 0 & 0 & 0 & 0 & 0 & 1.85\mathrm{E}{-5} & 0.00114 & -2.12\mathrm{E}{-4} \\
0 & 0 & 0 & 0 & 0 & 0 & -7.36\mathrm{E}{-5} & -2.12\mathrm{E}{-4} & 0.00818
\end{pmatrix}.$$

The inverse of the mass matrix is found through the incorrect algorithm of [28] as

$$^{incorrect}\mathbf{M}_{fb}^{-1} \approx$$

$$\begin{pmatrix}
8.67\mathrm{E}{-4} & -1.48\mathrm{E}{-5} & -3.33\mathrm{E}{-4} & 0 & 0 & 0 & 0 & 0 & 0 \\
-1.48\mathrm{E}{-5} & 1.8\mathrm{E}{-6} & 4.07\mathrm{E}{-5} & 0 & 0 & 0 & 0 & 0 & 0 \\
-3.33\mathrm{E}{-4} & 4.07\mathrm{E}{-5} & 0.00117 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 7.11\mathrm{E}{-6} & 1.5\mathrm{E}{-4} & 3.13\mathrm{E}{-5} & 0 & 0 & 0 \\
0 & 0 & 0 & 1.5\mathrm{E}{-4} & 1.0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 3.13\mathrm{E}{-5} & 0 & 0.00449 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1.62\mathrm{E}{-6} & 1.85\mathrm{E}{-5} & 1.26\mathrm{E}{-5} \\
0 & 0 & 0 & 0 & 0 & 0 & 1.85\mathrm{E}{-5} & 0.00114 & 7.75\mathrm{E}{-4} \\
0 & 0 & 0 & 0 & 0 & 0 & 1.26\mathrm{E}{-5} & 7.75\mathrm{E}{-4} & 0.00426
\end{pmatrix}.$$

We define the maximum and average errors between the inverses of the mass matrix computed through the CRBA and the incorrect version of the IMMA similar to (5.9) and (5.10) and compute them as

$$e^{max} := max \left| \frac{{}^{CRBA}\mathbf{M}_{fb}^{-1} - {}^{incorrect}\mathbf{M}_{fb}^{-1}}{avg|{}^{CRBA}\mathbf{M}_{fb}^{-1}|} \right| \approx 0.3109 \tag{5.21}$$

and

$$e^{avg} := avg \left| \frac{{}^{CRBA}\mathbf{M}_{fb}^{-1} - {}^{incorrect}\mathbf{M}_{fb}^{-1}}{avg|{}^{CRBA}\mathbf{M}_{fb}^{-1}|} \right| \approx 0.0074. \tag{5.22}$$

The inverse of the mass matrix is found through the correct algorithm of Table 4.6 as

$${}^{correct}\mathbf{M}_{fb}^{-1} \approx$$

$$\begin{pmatrix}
8.67\text{E}{-4} & -1.48\text{E}{-5} & -4.07\text{E}{-4} & 0 & 0 & 0 & 0 & 0 & 0 \\
-1.48\text{E}{-5} & 1.8\text{E}{-6} & 4.19\text{E}{-5} & 0 & 0 & 0 & 0 & 0 & 0 \\
-4.07\text{E}{-4} & 4.19\text{E}{-5} & 0.0012 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 7.11\text{E}{-6} & 1.5\text{E}{-4} & 3.13\text{E}{-5} & 0 & 0 & 0 \\
0 & 0 & 0 & 1.5\text{E}{-4} & 1.0 & 6.58\text{E}{-4} & 0 & 0 & 0 \\
0 & 0 & 0 & 3.13\text{E}{-5} & 6.58\text{E}{-4} & 0.00449 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1.62\text{E}{-6} & 1.85\text{E}{-5} & -7.36\text{E}{-5} \\
0 & 0 & 0 & 0 & 0 & 0 & 1.85\text{E}{-5} & 0.00114 & -2.12\text{E}{-4} \\
0 & 0 & 0 & 0 & 0 & 0 & -7.36\text{E}{-5} & -2.12\text{E}{-4} & 0.00818
\end{pmatrix}.$$

We define the maximum and average errors between the inverses of the mass matrix computed through the CRBA and our newly presented version of the IMMA similar to (5.9) and (5.10) and compute them as

$$e^{max} := max \left| \frac{{}^{CRBA}\mathbf{M}_{fb}^{-1} - {}^{correct}\mathbf{M}_{fb}^{-1}}{avg|{}^{CRBA}\mathbf{M}_{fb}^{-1}|} \right| \approx 1.7594\text{E}{-14} \tag{5.23}$$

and

$$e^{avg} := avg \left| \frac{{}^{CRBA}\mathbf{M}_{fb}^{-1} - {}^{correct}\mathbf{M}_{fb}^{-1}}{avg|{}^{CRBA}\mathbf{M}_{fb}^{-1}|} \right| \approx 2.2347\text{E}{-16}. \tag{5.24}$$

Using the randomized test, where we compute the inverse of the mass matrix for the randomized inputs and system parameters 100 times, we obtain the highest maximum error and the highest average error as

$$e^{max}_{rand} \approx 1.2826\text{E}{-12} \qquad \text{and} \qquad e^{avg}_{rand} \approx 3.0318\text{E}{-14}. \tag{5.25}$$

These results verify that the algorithm presented in [28] is indeed incorrect and that (apart from numerical rounding errors) our newly presented version of the IMMA presented in Table 4.6 provides correct results.

5.4 Inverse Mass Matrix Algorithm for moving-base systems

The IMMAmb presented in Section 4.4 can be verified using a method similar to the method used to verify the IMMA. The correctness of the results of the IMMAmb can be verified simply by computing the mass matrix through the CRBAmb (Appendix E.4) and inverting it. Both inverted mass matrices should be identical (apart from numerical rounding errors). As an example to apply these test methods on, we have chosen the designed test system and input variables presented in Section 5.1. Due to their sizes, the inverse of the mass matrices $\mathbf{M}^{-1}$ found through the CRBAmb and through the algorithm presented in Table 4.7 are presented in Appendix D.2. We define the maximum and average errors between the inverses of the mass matrix computed through the CRBAmb and our newly presented version of the IMMAmb similar to (5.9) and (5.10) and compute them as

$$e^{max} := max \left| \frac{\mathbf{M}^{-1}_{CRBAmb} - \mathbf{M}^{-1}_{IMMAmb}}{avg|\mathbf{M}^{-1}_{CRBAmb}|} \right| \approx 3.0712\text{E}{-14} \tag{5.26}$$

and

$$e^{avg} := avg \left| \frac{\mathbf{M}^{-1}_{CRBAmb} - \mathbf{M}^{-1}_{IMMAmb}}{avg|\mathbf{M}^{-1}_{CRBAmb}|} \right| \approx 7.4086\text{E}{-16}. \tag{5.27}$$

Using the randomized test, where we compute the inverse of the mass matrix for the randomized inputs and system parameters 100 times, we obtain the highest maximum error and the highest average error as

$$e^{max}_{rand} \approx 6.5914\text{E}{-12} \qquad \text{and} \qquad e^{avg}_{rand} \approx 9.0604\text{E}{-13}. \tag{5.28}$$

These results verify that (apart from numerical rounding errors) our newly presented version of the IMMAmb presented in Table 4.7 provides correct results.

5.5 Left-trivialized derivatives of the forward dynamics

In Section 5.2 we verified that the algorithms presented in Section 4.2 result in correct derivatives of the extended inverse dynamics. However, we state in Proposition 3.3 that the derivatives of the forward dynamics of (3.17) can be expressed in terms of the derivatives of the extended inverse dynamics of (3.39) and the inverse of the mass matrix as $\mathrm{D}_i\, FD(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \boldsymbol{\tau}) = -\mathbf{M}^{-1}\,\mathrm{D}_i\, \overline{ID}(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \dot{\mathbf{v}}, \dot{\mathbf{r}})$ where $i \in \{1, 2, 3, 4\}$. This relation can be verified by computing the derivatives of the forward dynamics through finite differences. We use the same perturbations as defined in Section 5.2, but now applied to the forward dynamics function of (3.17) to obtain the derivatives

$$[\mathrm{D}_1\, FD(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \boldsymbol{\tau}) \circ \mathrm{D}\, L_H(I)]_i = \frac{FD(\mathbf{H}\exp((\Delta^b_i)^\wedge), \mathbf{s}, \mathbf{v}, \mathbf{r}, \boldsymbol{\tau}) - FD(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \boldsymbol{\tau})}{\delta}, \tag{5.29}$$

$$[\mathrm{D}_2\, FD(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \boldsymbol{\tau})]_k = \frac{FD(\mathbf{H}, \mathbf{s} + \Delta^j_k, \mathbf{v}, \mathbf{r}, \boldsymbol{\tau}) - FD(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \boldsymbol{\tau})}{\delta}, \tag{5.30}$$

$$[\mathrm{D}_3\, FD(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \boldsymbol{\tau})]_i = \frac{FD(\mathbf{H}, \mathbf{s}, \mathbf{v} + \Delta^b_i, \mathbf{r}, \boldsymbol{\tau}) - FD(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \boldsymbol{\tau})}{\delta}, \tag{5.31}$$

$$[\mathrm{D}_4\, FD(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \boldsymbol{\tau})]_k = \frac{FD(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r} + \Delta^j_k, \boldsymbol{\tau}) - FD(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \boldsymbol{\tau})}{\delta}. \tag{5.32}$$

where $i \in \{1...6\}$ and $k \in \{1...n_J\}$ and $[X]_i$ is the i-th column of tensor X. Since the ABAmb (Appendix E.6) computes the forward dynamics, these left-trivialized derivatives can be approximated by simply evaluating the ABAmb once unperturbed, and once for each perturbation.

The following subsections present the numerical results for both the derivatives obtained through our algorithms presented in Section 4.2 as well as the derivatives obtained through finite differences, where we use the designed test system and input variables presented in Section 5.1. As the forward dynamics of (3.17) require the joint torques $\boldsymbol{\tau}$ as an input, we compute $\boldsymbol{\tau}$ to be corresponding to our selected inputs through the RNEAmb (see Appendix E.2) as

$$\boldsymbol{\tau} \approx 10^8 \cdot [-0.0951; 1.4390; 0.0297; -0.6413; 0.2144; 0.0062; 0.8426; -0.0026; 0.0317].$$

Again we choose $\delta = 10^{-6}$, which may have effect on the errors presented in this chapter, but not on the drawn conclusions that our newly presented algorithms return correct values.

5.5.1 Derivatives of forward dynamics w.r.t. transformation matrix

The derivative of the forward dynamics of (3.17) with respect to the transformation matrix $\mathbf{H}$ obtained through the finite differences method using the selected inputs is found as

$$\mathrm{D}_1\, FD^{FinDiff} \approx \begin{pmatrix} 9444.0 & 1.02\,10^4 & -1.56\,10^4 & -2.14\,10^4 & -1.4\,10^6 & -6.07\,10^5 \\ -644.0 & 293.0 & 1522.0 & -2.36\,10^4 & 1.48\,10^5 & 1.21\,10^5 \\ 2533.0 & 4000.0 & -3622.0 & -4.82\,10^4 & -1.34\,10^5 & 4.18\,10^4 \\ -157.0 & -258.0 & 219.0 & 3400.0 & 1.27\,10^4 & -1500.0 \\ -87.8 & -110.0 & 138.0 & 732.0 & 1.06\,10^4 & 3388.0 \\ -270.0 & -287.0 & 450.0 & -135.0 & 4.28\,10^4 & 2.02\,10^4 \\ -9266.0 & -9833.0 & 1.54\,10^4 & 1.39\,10^4 & 1.48\,10^6 & 6.6\,10^5 \\ 289.0 & 520.0 & -384.0 & -7700.0 & -1.69\,10^4 & 8888.0 \\ 8077.0 & 1.29\,10^4 & -1.14\,10^4 & -1.62\,10^5 & -6.05\,10^5 & 6.98\,10^4 \\ 60.3 & 129.0 & -70.9 & -1944.0 & -2599.0 & 3033.0 \\ 7.66\,10^5 & 7.96\,10^5 & -1.28\,10^6 & 5.86\,10^5 & -1.27\,10^8 & -6.06\,10^7 \\ -2.23\,10^4 & -2.38\,10^4 & 3.71\,10^4 & -4000.0 & 3.59\,10^6 & 1.68\,10^6 \\ 260.0 & 346.0 & -401.0 & -3100.0 & -2.41\,10^4 & -4588.0 \\ 2300.0 & 1400.0 & -4322.0 & 3.8\,10^4 & -5.66\,10^5 & -3.46\,10^5 \\ -8411.0 & -7700.0 & 1.46\,10^4 & -1.57\,10^4 & 8.11\,10^5 & 4.18\,10^5 \end{pmatrix}.$$

The same derivative obtained through $\mathrm{D}_1\, FD = -\mathbf{M}^{-1}\,\mathrm{D}_1\,\overline{ID}$ of (3.53), where $\mathbf{M}^{-1}$ and $\mathrm{D}_1\,\overline{ID}$ are found through the IMMAmb of Table 4.7 and the derivative algorithm of Table 4.2 respectively, is found as

$$\mathrm{D}_1\, FD^{Alg} \approx \begin{pmatrix} 9444.0 & 1.02\,10^4 & -1.56\,10^4 & -2.14\,10^4 & -1.4\,10^6 & -6.07\,10^5 \\ -644.0 & 293.0 & 1522.0 & -2.36\,10^4 & 1.48\,10^5 & 1.21\,10^5 \\ 2533.0 & 4000.0 & -3622.0 & -4.82\,10^4 & -1.34\,10^5 & 4.18\,10^4 \\ -157.0 & -258.0 & 219.0 & 3400.0 & 1.27\,10^4 & -1500.0 \\ -87.8 & -110.0 & 138.0 & 732.0 & 1.06\,10^4 & 3388.0 \\ -270.0 & -287.0 & 450.0 & -135.0 & 4.28\,10^4 & 2.02\,10^4 \\ -9266.0 & -9833.0 & 1.54\,10^4 & 1.39\,10^4 & 1.48\,10^6 & 6.6\,10^5 \\ 289.0 & 520.0 & -384.0 & -7700.0 & -1.69\,10^4 & 8888.0 \\ 8077.0 & 1.29\,10^4 & -1.14\,10^4 & -1.62\,10^5 & -6.05\,10^5 & 6.98\,10^4 \\ 60.3 & 129.0 & -70.9 & -1944.0 & -2599.0 & 3033.0 \\ 7.66\,10^5 & 7.96\,10^5 & -1.28\,10^6 & 5.85\,10^5 & -1.27\,10^8 & -6.06\,10^7 \\ -2.23\,10^4 & -2.38\,10^4 & 3.71\,10^4 & -4000.0 & 3.59\,10^6 & 1.68\,10^6 \\ 260.0 & 346.0 & -401.0 & -3100.0 & -2.41\,10^4 & -4588.0 \\ 2300.0 & 1400.0 & -4322.0 & 3.8\,10^4 & -5.66\,10^5 & -3.46\,10^5 \\ -8411.0 & -7700.0 & 1.46\,10^4 & -1.57\,10^4 & 8.11\,10^5 & 4.18\,10^5 \end{pmatrix}.$$

We define the maximum and average errors between both derivatives similar to (5.9) and (5.10) and compute them as

$$e^{max} := max\left|\frac{\mathrm{D}_1\, FD^{Alg} - \mathrm{D}_1\, FD^{FinDiff}}{avg|\,\mathrm{D}_1\, FD^{Alg}|}\right| \approx 3.8931\mathrm{E}{-5} \tag{5.33}$$

and

$$e^{avg} := avg\left|\frac{\mathrm{D}_1\, FD^{Alg} - \mathrm{D}_1\, FD^{FinDiff}}{avg|\,\mathrm{D}_1\, FD^{Alg}|}\right| \approx 1.2438\mathrm{E}{-6}. \tag{5.34}$$

Using the randomized test, where we compute the derivative for the randomized inputs and system parameters 100 times, we obtain the highest maximum error and the highest average error as

$$e_{rand}^{max} \approx 4.1023\text{E}{-5} \qquad \text{and} \qquad e_{rand}^{avg} \approx 2.3560\text{E}{-6}. \tag{5.35}$$

These results verify that (apart from approximation errors in the finite differences) the relation $\mathrm{D}_1\, FD = -\mathbf{M}^{-1}\, \mathrm{D}_1\, \overline{ID}$ of (3.53) is indeed correct, and the derivatives of the forward dynamics of (3.17) with respect to the transformation matrix $\mathbf{H}$ can be numerically efficiently computed through the derivatives of the extended inverse dynamics of Table 4.2 and the IMMAmb of Table 4.7.

5.5.2 Derivatives of forward dynamics w.r.t. generalized position vector

The derivative of the forward dynamics of (3.17) with respect to the generalized position vector $\mathbf{s}$ obtained through the finite differences method using the selected inputs is found as

$$\mathrm{D}_2\, FD^{FinDiff} \approx \begin{pmatrix}
-796.9 & -1.129\text{E}5 & 88399.0 & -90000.0 & -8811.0 & -67.1 & -4673.0 & -677.1 & 10144.0 \\
-541.9 & -75666.0 & 56999.0 & 9787.0 & 1073.0 & -1.957 & 1175.0 & 355.6 & -10400.0 \\
-1577.0 & -1.946\text{E}5 & 1.666\text{E}5 & -10399.0 & -1181.0 & 32.54 & -6414.0 & -316.2 & 5474.0 \\
82.56 & 13177.0 & -10500.0 & 799.9 & 116.5 & -2.14 & 69.81 & -14.15 & 380.8 \\
21.26 & 2618.0 & -2229.0 & 745.1 & 39.71 & 2.917 & 27.29 & 9.866 & -176.2 \\
10.02 & 1743.0 & -1294.0 & 2728.0 & 293.1 & 0.114 & 38.74 & 7.965 & -178.5 \\
342.2 & 56144.0 & -39844.0 & 94611.0 & 8650.0 & 126.9 & 1690.0 & 710.0 & -11200.0 \\
-184.7 & -28222.0 & 22999.0 & -1221.0 & -118.9 & -0.7848 & -116.0 & 12.55 & -477.7 \\
-4045.0 & -6.051\text{E}5 & 4.976\text{E}5 & -43600.0 & -3713.0 & -59.59 & -4334.0 & -202.6 & 752.0 \\
-54.34 & -9490.0 & 7444.0 & 199.1 & -65.55 & 29.53 & -154.7 & 43.24 & -840.5 \\
-9895.0 & -1.548\text{E}6 & 1.137\text{E}6 & -8.104\text{E}6 & -1.097\text{E}6 & -426.9 & -18577.0 & -5717.0 & 99866.0 \\
728.3 & 1.164\text{E}5 & -82200.0 & 2.158\text{E}5 & 23600.0 & 134.7 & -1335.0 & 605.0 & -11922.0 \\
-78.69 & -12577.0 & 10122.0 & -1587.0 & -174.7 & 0.0846 & -145.8 & -41.95 & 831.5 \\
1067.0 & 1.362\text{E}5 & -1.172\text{E}5 & -36200.0 & -2400.0 & -137.8 & 4758.0 & -593.5 & 6480.0 \\
126.5 & 14922.0 & -16800.0 & 56766.0 & 4738.0 & 104.7 & 2404.0 & 3971.0 & -1.056\text{E}5
\end{pmatrix}.$$

The same derivative obtained through $\mathrm{D}_2\, FD = -\mathbf{M}^{-1}\, \mathrm{D}_2\, \overline{ID}$ of (3.53), where $\mathbf{M}^{-1}$ and $\mathrm{D}_2\, \overline{ID}$ are found through the IMMAmb of Table 4.7 and the derivative algorithm of Table 4.3 respectively, is found as

$$\mathrm{D}_2\, FD^{Alg} \approx \begin{pmatrix}
-796.9 & -1.129\text{E}5 & 88399.0 & -90000.0 & -8811.0 & -67.1 & -4673.0 & -677.1 & 10144.0 \\
-541.9 & -75666.0 & 56999.0 & 9788.0 & 1073.0 & -1.957 & 1175.0 & 355.6 & -10400.0 \\
-1577.0 & -1.946\text{E}5 & 1.666\text{E}5 & -10399.0 & -1181.0 & 32.54 & -6414.0 & -316.2 & 5474.0 \\
82.56 & 13177.0 & -10500.0 & 799.9 & 116.5 & -2.14 & 69.81 & -14.15 & 380.8 \\
21.26 & 2618.0 & -2229.0 & 745.1 & 39.72 & 2.917 & 27.29 & 9.866 & -176.2 \\
10.02 & 1743.0 & -1294.0 & 2728.0 & 293.1 & 0.114 & 38.74 & 7.965 & -178.5 \\
342.2 & 56144.0 & -39844.0 & 94611.0 & 8650.0 & 126.9 & 1690.0 & 710.0 & -11200.0 \\
-184.7 & -28222.0 & 22999.0 & -1221.0 & -118.9 & -0.7848 & -116.0 & 12.55 & -477.7 \\
-4045.0 & -6.051\text{E}5 & 4.976\text{E}5 & -43600.0 & -3713.0 & -59.59 & -4334.0 & -202.6 & 752.0 \\
-54.34 & -9490.0 & 7444.0 & 199.1 & -65.55 & 29.53 & -154.7 & 43.24 & -840.5 \\
-9895.0 & -1.548\text{E}6 & 1.137\text{E}6 & -8.104\text{E}6 & -1.097\text{E}6 & -426.9 & -18577.0 & -5717.0 & 99866.0 \\
728.3 & 1.164\text{E}5 & -82200.0 & 2.158\text{E}5 & 23600.0 & 134.7 & -1335.0 & 605.0 & -11911.0 \\
-78.69 & -12577.0 & 10122.0 & -1587.0 & -174.7 & 0.0846 & -145.8 & -41.95 & 831.5 \\
1067.0 & 1.362\text{E}5 & -1.172\text{E}5 & -36200.0 & -2400.0 & -137.8 & 4758.0 & -593.5 & 6480.0 \\
126.5 & 14922.0 & -16800.0 & 56766.0 & 4738.0 & 104.7 & 2404.0 & 3971.0 & -1.056\text{E}5
\end{pmatrix}.$$

We define the maximum and average errors between both derivatives similar to (5.9) and (5.10) and compute them as

$$e^{max} := max \left| \frac{\mathrm{D}_2\, FD^{Alg} - \mathrm{D}_2\, FD^{FinDiff}}{avg|\,\mathrm{D}_2\, FD^{Alg}|} \right| \approx 4.8305\text{E}{-4} \tag{5.36}$$

and

$$e^{avg} := avg \left| \frac{\mathrm{D}_2\, FD^{Alg} - \mathrm{D}_2\, FD^{FinDiff}}{avg|\,\mathrm{D}_2\, FD^{Alg}|} \right| \approx 7.4819\text{E}{-6}. \tag{5.37}$$

Using the randomized test, where we compute the derivative for the randomized inputs and system parameters 100 times, we obtain the highest maximum error and the highest average error as

$$e^{max}_{rand} \approx 4.6853\text{E}{-3} \qquad \text{and} \qquad e^{avg}_{rand} \approx 1.3604\text{E}{-4}. \tag{5.38}$$

These results verify that (apart from approximation errors in the finite differences) the relation $\mathrm{D}_2\, FD = -\mathbf{M}^{-1}\,\mathrm{D}_2\,\overline{ID}$ of (3.53) is indeed correct, and the derivatives of the forward dynamics of (3.17) with respect to the generalized position vector $\mathbf{s}$ can be numerically efficiently computed through the derivatives of the extended inverse dynamics of Table 4.3 and the IMMAmb of Table 4.7.

5.5.3 Derivatives of forward dynamics w.r.t. moving-base velocity

The derivative of the forward dynamics of (3.17) with respect to the moving-base velocity $\mathbf{v}$ obtained through the finite differences method using the selected inputs is found as

$$\mathrm{D}_3\, FD^{FinDiff} \approx \begin{pmatrix}
257.5 & -974.8 & -479.6 & 942.1 & -35211.0 & 83100.0 \\
8.659 & 87.56 & -13.19 & 947.6 & 6832.0 & -8240.0 \\
97.81 & -233.8 & -189.4 & 5433.0 & 1438.0 & 10522.0 \\
-6.518 & 14.24 & 12.16 & -238.2 & -45.45 & -864.1 \\
-2.243 & 8.614 & 5.43 & -58.29 & 204.4 & -655.3 \\
-7.045 & 27.98 & 13.61 & -2.501 & 1170.0 & -2525.0 \\
-246.5 & 957.6 & 463.5 & 1014.0 & 37899.0 & -86833.0 \\
12.98 & -25.36 & -24.53 & 515.1 & 424.7 & 1252.0 \\
316.6 & -738.8 & -613.9 & 11700.0 & 2036.0 & 41466.0 \\
4.455 & -5.114 & -5.493 & 208.7 & 135.1 & 252.5 \\
19277.0 & -79566.0 & -37833.0 & -87333.0 & -3.487\text{E}6 & 7.474\text{E}6 \\
-571.1 & 2302.0 & 1136.0 & 1139.0 & 96966.0 & -2.113\text{E}5 \\
7.328 & -25.16 & -16.95 & 262.2 & -307.3 & 1552.0 \\
29.92 & -260.0 & -68.49 & -5044.0 & -19077.0 & 30999.0 \\
-43.97 & 865.4 & 432.3 & -4104.0 & 24711.0 & -49966.0
\end{pmatrix}.$$

The same derivative obtained through $\mathrm{D}_3\, FD = -\mathbf{M}^{-1}\,\mathrm{D}_3\,\overline{ID}$ of (3.53), where $\mathbf{M}^{-1}$ and $\mathrm{D}_3\,\overline{ID}$ are found through the IMMAmb of Table 4.7 and the derivative algorithm of Table 4.4 respectively, is found as

$$\mathrm{D}_3\, FD^{Alg} \approx \begin{pmatrix} 257.5 & -974.8 & -479.6 & 942.1 & -35211.0 & 83100.0 \\ 8.659 & 87.56 & -13.19 & 947.6 & 6832.0 & -8240.0 \\ 97.81 & -233.8 & -189.4 & 5433.0 & 1438.0 & 10522.0 \\ -6.518 & 14.24 & 12.16 & -238.2 & -45.45 & -864.1 \\ -2.243 & 8.614 & 5.43 & -58.29 & 204.4 & -655.3 \\ -7.045 & 27.98 & 13.61 & -2.501 & 1170.0 & -2525.0 \\ -246.5 & 957.6 & 463.5 & 1014.0 & 37899.0 & -86833.0 \\ 12.98 & -25.36 & -24.53 & 515.1 & 424.7 & 1252.0 \\ 316.6 & -738.8 & -613.9 & 11700.0 & 2036.0 & 41466.0 \\ 4.455 & -5.114 & -5.493 & 208.7 & 135.1 & 252.5 \\ 19277.0 & -79566.0 & -37833.0 & -87333.0 & -3.487\mathrm{E}6 & 7.474\mathrm{E}6 \\ -571.1 & 2302.0 & 1136.0 & 1139.0 & 96966.0 & -2.113\mathrm{E}5 \\ 7.328 & -25.16 & -16.95 & 262.2 & -307.3 & 1552.0 \\ 29.92 & -260.0 & -68.49 & -5044.0 & -19077.0 & 30999.0 \\ -43.97 & 865.4 & 432.3 & -4104.0 & 24711.0 & -49966.0 \end{pmatrix}.$$

We define the maximum and average errors between both derivatives similar to (5.9) and (5.10) and compute them as

$$e^{max} := max \left| \frac{\mathrm{D}_3\, FD^{Alg} - \mathrm{D}_3\, FD^{FinDiff}}{avg|\,\mathrm{D}_3\, FD^{Alg}|} \right| \approx 3.1249\mathrm{E}{-6} \tag{5.39}$$

and

$$e^{avg} := avg \left| \frac{\mathrm{D}_3\, FD^{Alg} - \mathrm{D}_3\, FD^{FinDiff}}{avg|\,\mathrm{D}_3\, FD^{Alg}|} \right| \approx 6.7316\mathrm{E}{-8}. \tag{5.40}$$

Using the randomized test, where we compute the derivative for the randomized inputs and system parameters 100 times, we obtain the highest maximum error and the highest average error as

$$e^{max}_{rand} \approx 1.8230\mathrm{E}{-5} \qquad \text{and} \qquad e^{avg}_{rand} \approx 1.3021\mathrm{E}{-6}. \tag{5.41}$$

These results verify that (apart from approximation errors in the finite differences) the relation $\mathrm{D}_3\, FD = -\mathbf{M}^{-1}\,\mathrm{D}_3\,\overline{ID}$ of (3.53) is indeed correct, and the derivatives of the forward dynamics of (3.17) with respect to the moving-base velocity $\mathbf{v}$ can be numerically efficiently computed through the derivatives of the extended inverse dynamics of Table 4.4 and the IMMAmb of Table 4.7.

5.5.4 Derivatives of forward dynamics w.r.t. generalized velocity vector

The derivative of the forward dynamics of (3.17) with respect to the generalized velocity vector $\mathbf{r}$ obtained through the finite differences method using the selected inputs is found as

$$\mathrm{D}_4\, FD^{FinDiff} \approx$$
$$\begin{pmatrix}
40.38 & 1239.0 & 111.0 & 5645.0 & 67.53 & -4.481 & -409.7 & -8.586 & -19.56 \\
25.5 & -14.01 & 106.0 & -642.7 & -9.524 & -2.401 & -663.7 & 18.0 & -37.02 \\
76.43 & 2954.0 & 165.7 & 712.1 & 11.33 & 0.09531 & 36.97 & -8.202 & 3.667 \\
-3.547 & -80.06 & -14.36 & -55.07 & -1.419 & 0.06651 & 28.34 & -0.8655 & 1.819 \\
-0.9339 & -11.49 & -3.378 & -42.56 & 0.1696 & 0.01848 & -3.946 & 0.3208 & -0.2986 \\
-0.5251 & -20.11 & -1.596 & -174.8 & -2.767 & -0.0687 & -1.618 & 0.1158 & 0.08181 \\
-28.27 & 1155.0 & -107.8 & -5839.0 & -58.09 & 4.431 & 285.2 & 11.89 & 11.48 \\
9.051 & 182.8 & 17.84 & 76.46 & 0.8701 & -0.1653 & -44.21 & 1.022 & -2.608 \\
210.5 & 4719.0 & 292.4 & 2679.0 & 20.69 & -2.028 & -411.0 & 1.811 & -23.02 \\
2.585 & 139.8 & 6.713 & 22.0 & 0.6916 & 0.8499 & -31.22 & 1.737 & -2.387 \\
516.0 & 10700.0 & 1762.0 & 5.242E5 & 1950.0 & 41.0 & -2957.0 & -27.55 & -269.9 \\
-36.99 & 93.54 & -170.9 & -15033.0 & -211.4 & -1.784 & -18.61 & 8.52 & 10.85 \\
3.34 & 86.15 & 13.32 & 102.1 & 1.516 & -0.1113 & 14.11 & -0.1589 & -0.4607 \\
-51.86 & -2481.0 & -95.39 & 2084.0 & 1.527 & -2.615 & -413.5 & -1.326 & -6.585 \\
-4.687 & -466.3 & -4.996 & -3463.0 & -22.92 & 1.066 & -4436.0 & 116.0 & -68.13
\end{pmatrix}.$$

The same derivative obtained through $\mathrm{D}_4\, FD = -\mathbf{M}^{-1}\, \mathrm{D}_4\, \overline{ID}$ of (3.53), where $\mathbf{M}^{-1}$ and $\mathrm{D}_4\, \overline{ID}$ are found through the IMMAmb of Table 4.7 and the derivative algorithm of Table 4.5 respectively, is found as

$$\mathrm{D}_4\, FD^{Alg} \approx$$
$$\begin{pmatrix}
40.38 & 1239.0 & 111.0 & 5645.0 & 67.53 & -4.481 & -409.7 & -8.586 & -19.56 \\
25.5 & -14.01 & 106.0 & -642.7 & -9.524 & -2.401 & -663.7 & 18.0 & -37.02 \\
76.43 & 2954.0 & 165.7 & 712.1 & 11.33 & 0.09532 & 36.97 & -8.202 & 3.667 \\
-3.547 & -80.06 & -14.36 & -55.07 & -1.419 & 0.06651 & 28.34 & -0.8655 & 1.819 \\
-0.9339 & -11.49 & -3.378 & -42.56 & 0.1696 & 0.01848 & -3.946 & 0.3208 & -0.2986 \\
-0.5251 & -20.11 & -1.596 & -174.8 & -2.767 & -0.0687 & -1.618 & 0.1158 & 0.08181 \\
-28.27 & 1155.0 & -107.8 & -5839.0 & -58.09 & 4.431 & 285.2 & 11.89 & 11.48 \\
9.051 & 182.8 & 17.84 & 76.46 & 0.8701 & -0.1653 & -44.21 & 1.022 & -2.608 \\
210.5 & 4719.0 & 292.4 & 2679.0 & 20.69 & -2.028 & -411.0 & 1.811 & -23.02 \\
2.585 & 139.8 & 6.713 & 22.0 & 0.6916 & 0.8499 & -31.22 & 1.737 & -2.387 \\
516.0 & 10700.0 & 1762.0 & 5.242E5 & 1950.0 & 41.0 & -2957.0 & -27.55 & -269.9 \\
-36.99 & 93.54 & -170.9 & -15033.0 & -211.4 & -1.784 & -18.61 & 8.52 & 10.85 \\
3.34 & 86.15 & 13.32 & 102.1 & 1.516 & -0.1113 & 14.11 & -0.1589 & -0.4607 \\
-51.86 & -2481.0 & -95.39 & 2084.0 & 1.527 & -2.615 & -413.5 & -1.326 & -6.585 \\
-4.687 & -466.3 & -4.996 & -3463.0 & -22.92 & 1.066 & -4436.0 & 116.0 & -68.13
\end{pmatrix}.$$

We define the maximum and average errors between both derivatives similar to (5.9) and (5.10) and compute them as

$$e^{max} := max \left| \frac{\mathrm{D}_4\, FD^{Alg} - \mathrm{D}_4\, FD^{FinDiff}}{avg|\, \mathrm{D}_4\, FD^{Alg}|} \right| \approx 1.9549\mathrm{E}{-6} \tag{5.42}$$

and

$$e^{avg} := avg \left| \frac{\mathrm{D}_4\, FD^{Alg} - \mathrm{D}_4\, FD^{FinDiff}}{avg|\, \mathrm{D}_4\, FD^{Alg}|} \right| \approx 2.5640\mathrm{E}{-8}. \tag{5.43}$$

Using the randomized test, where we compute the derivative for the randomized inputs and system parameters 100 times, we obtain the highest maximum error and the highest average error as

$$e_{rand}^{max} \approx 1.5693\text{E}{-4} \qquad \text{and} \qquad e_{rand}^{avg} \approx 1.3766\text{E}{-6}. \tag{5.44}$$

These results verify that (apart from approximation errors in the finite differences) the relation $\mathrm{D}_4 FD = -\mathbf{M}^{-1}\,\mathrm{D}_4 \overline{ID}$ of (3.53) is indeed correct, and the derivatives of the forward dynamics of (3.17) with respect to the generalized velocity vector $\mathbf{r}$ can be numerically efficiently computed through the derivatives of the extended inverse dynamics of Table 4.5 and the IMMAmb of Table 4.7.

5.6 Summary

In this chapter we have presented the numerical verification of our newly presented algorithms that compute singularity-free linearization of the forward dynamics. We have presented and detailed a moving-base multibody test system in Section 5.1 to verify our newly presented algorithms on. To verify that we compute the left-trivialized derivatives of the extended inverse dynamics correctly, we have compared our obtained results to the same derivatives obtained through left-trivialized finite differences. To verify that we compute the inverse mass matrix for both fixed-base and moving-base systems correctly, we have compared our obtained results through the IMMA and IMMAmb with the inverse of the mass matrix obtained through the Composite Rigid Body Algorithm for fixed-base and moving-base systems (CRBA and CRBAmb). To verify that our presented relation between the derivatives of the forward dynamics and the derivatives of the extended inverse dynamics is correct, we have compared the derivatives of the forward dynamics obtained through our derivatives algorithms and IMMAmb with the derivatives of the forward dynamics obtained through the left-trivialized finite differences of the Articulated Body Algorithm for moving-base systems (ABAmb). The numerical results of these tests verify that we correctly compute the geometric linearization of the forward dynamics. This concludes the third objective: *Verify the correctness of the derived algorithms*, and with that the goal of this thesis: *Develop a numerically efficient and accurate method to compute the singularity-free geometric linearization of moving-base multibody systems*, as described in Section 1.3.

The following chapter presents the conclusions of this thesis and discusses recommendations for further research.

Chapter 6

Conclusions and Recommendations

This chapter presents the conclusions from this thesis, followed by recommendations for future research. As stated in Section 1.3, the research goal is to *develop a numerically efficient and accurate method to compute the singularity-free geometric linearization of moving-base multibody systems.* This research goal has been divided in three objectives:

- Derive mathematical formulas for the singularity-free geometric linearization of moving-base multibody systems.
- Derive numerically efficient and accurate algorithms to compute the singularity-free geometric linearization of moving-base multibody systems.
- Verify the correctness of the derived algorithms.

Section 6.1 presents the conclusions that are drawn and the contribution of this thesis is summarized. Section 6.2 presents suggested areas for further research related to this thesis.

6.1 Conclusions

In this thesis we have derived and verified a numerically efficient and accurate singularity-free geometric linearization for moving-base multibody dynamics on $\mathrm{SE}(3) \times \mathbb{R}^{n_J}$. Below we explain steps taken to achieve this goal.

Method for computing the linearization. In Section 1.2 we have presented a literary review on all aspects of this research. Multiple methods to compute the sensitivity of multibody dynamics have been explored, which each have up- and downsides, as shown in Table 1.1. We have chosen the method of recursive analytical derivation, which uses recursive algorithms to compute the sensitivity using mathematical knowledge about the system. In current literature, recursive analytical derivation only exists applied to dynamics on vector spaces, which bring singularity issues to moving-base dynamics. Also the underactuated moving-base system is treated as fully actuated by modelling the non-actuated moving base as six separate actuated one-DoF joints. Therefore we have chosen to apply the theory of left-trivialized linearization to analytically derive singularity-free moving-base multibody dynamics.

Mathematical formulas for the linearization. In Chapter 3 we have shown the Lie group on which moving-base multibody dynamics can be modelled in a singularity-free manner. The choice of the set on which the dynamics are modelled was simple, however the choice of the group operation is important to achieve simple computations as is shown in [11]. Using left-trivialization, the time-derivatives of the state manifold have been morphed into a vector space, which allowed us to present a system state vector used in the linearized system. The state matrix has been found in terms of the left-trivialized derivatives of the forward dynamics. The input matrix has been found as the inverse of the mass matrix times the selection matrix. As [7] states that the derivatives of the forward dynamics can be found in a numerically more efficient manner through the derivatives of the inverse dynamics, we aimed to do the same. However, as we deal with underactuated systems, the inverse dynamics are not defined. Therefore we have made use of the theory in [31], which expands underactuated systems to non-physical fully actuated systems, allowing us to obtain the extended inverse dynamics. With use of the fully actuated non-physical system, the state matrix has been expressed in terms of the inverse of the mass matrix and the derivatives of the extended inverse dynamics. Chapter 3 concludes the first objective: *Derive and present the mathematical formulas for the singularity-free geometric linearization of moving-base multibody systems.*

Numerically efficient and accurate algorithms to compute the linearization. In Chapter 4 we have presented the Extended Inverse Dynamics Algorithm for moving-base systems (EIDAmb), which numerically efficiently computes the extended inverse dynamics as a recursive algorithm. The EIDAmb is based on the Recursive Newton Euler Algorithm for moving-base systems (RNEAmb) combined with the Generalized Bias Wrench Algorithm for moving-base systems (GBWAmb) and the Composite Rigid Body Algorithm for moving-base systems (CRBAmb). Using the EIDAmb, we have presented four recursive algorithms that compute the left-trivialized derivatives of the extended inverse dynamics with respect to the system states. The presented algorithms make use of the chain rule to compute the left-trivialized derivatives of each step in the EIDAmb. By using the mathematical knowledge of each step in the EIDAmb, we have left out unnecessary computations, increasing the numerical efficiency. Inspired on the Inverse Mass Matrix Algorithm (IMMA) presented in [28] that computes the inverse of the mass matrix without first computing the mass matrix, we aimed to derive an algorithm that computes the inverse of the mass matrix for moving-base systems. However, we have found that the algorithm in its current form is incorrect. By redoing the work of [28], we have found the correct version of the algorithm, which we have expanded to the Inverse Mass Matrix Algorithm for moving-base systems (IMMAmb). Chapter 4 concludes the second objective: *Derive numerically efficient and accurate algorithms to compute the singularity-free geometric linearization of moving-base multibody systems.*

Numerical verification of the presented algorithms. In Chapter 5 we have shown a moving-base multibody test system which has been used to verify the algorithms presented in Chapter 4. We have verified the correctness of the four algorithms that compute the left-trivialized derivatives of the extended inverse dynamics with respect to the state variables by comparing the results to the same derivatives computed through left-trivialized finite differences. We have also shown that the version of the IMMA presented in [28] is incorrect, and verified that our version of the IMMA is correct by comparing it to the fixed-base inverted mass matrix found through the Composite Rigid Body Algorithm (CRBA). Likewise, we have verified the correctness of the IMMAmb by comparing the results to the inverted mass matrix found through the Composite Rigid Body Algorithm for moving-base systems (CRBAmb). Finally the resulting left-trivialized derivatives of the forward dynamics expressed in terms of the left-trivialized derivatives of the extended inverse dynamics have been verified to be correct by comparing them to the same derivatives obtained through the left-trivialized finite differences of the forward dynamics, for which we have made use of the Articulated Body Algorithm for moving-base systems (ABAmb). Using this comparison, we have verified that the algorithms presented in Chapter 4 can indeed compute the left-trivialized derivatives of the forward dynamics. Chapter 5 concludes the third objective: *Verify the correctness of the derived algorithms*, and with that the goal of this thesis: *Develop a numerically efficient and accurate* *method to compute the singularity-free geometric linearization of moving-base multibody systems.*

6.2 Recommendations for future work

We see numerically efficient and accurate singularity-free geometric linearization of moving-base multibody dynamics as a stepping stone to create more efficient numerical tools such as model-predictive control and trajectory optimization. If the computational speed of the linearization method increases, more steps can be predicted ahead, a higher computational frequency can be achieved, or less CPU power is required. Having singularity-free linearization ensures that singularity issues, which can be disastrous in physical applications, are eliminated. This section presents options for future work: partly further investigation of our presented algorithms, and partly steps to get closer to real applications.

Implement algorithms in existing robot dynamics library. With this thesis we present the derived algorithms in pseudo-code and in *MATLAB*. However, direct execution of *MATLAB* code is not as computationally efficient as other languages such as *C++*. Implementing the algorithms in an already existing library brings our theory closer to real applications. We plan to implement the found algorithms in the *iDynTree* library [40] from IIT.

Compare computational time of our singularity-free algorithms and existing recursive analytical derivatives on vector spaces. The recursive analytical algorithms presented in [7] are also capable of computing the derivatives of the forward dynamics of moving-base multibody systems. However, those algorithms suffer from singularity issues in the moving-base due to its parametrization, and do not treat the moving base as a non-actuated part of the system. Using our algorithms, these singularity issues are eliminated and the system is treated as it is: underactuated. As numerical efficiency is still highly valued, we desire to compare both strategies in terms of numerical efficiency.

Test linearized system with automatic gain tuning. In [41], a method is presented to automatically tune the gains of a high-DoF robotic system. Automatic gain tuning requires the desired stiffness and damping matrices as inputs, and automatically tunes the controller gains such that the desired stiffness and damping matrices are achieved. As the stiffness and damping factors are presented in matrices, a linearized system is required to tune the controller gains such that these matrices are achieved. We can test our algorithms by using them in a simulation where the control gains are automatically tuned. If our algorithms indeed provide correct results, the system should evolve with the specified stiffness and damping.

Add holonomic constraint: foot rigidly attached to ground. A holonomic constraint specifies a position of a system. The case of a foot of a moving-base robot that is modelled to be rigidly attached to the ground is therefore a holonomic constraint, as it fixates the position of the foot. A foot is not generally fixated to the floor (e.g. walking, jumping), however this is a valid model for balancing on one foot, where one can be sure that the foot will not separate from the floor due to gravity. The algorithms presented in this thesis assume that there are no external forces acting on the system, therefore also no constraints act on the system. The Jacobian and external forces in (2.62) are state-dependent once constraints are forced, and therefore need to be taken into account in the linearization of a constrained moving-base multibody system. Adding this holonomic constraint is a step closer to real application.

Add inequality constraint: foot in contact with on ground. An inequality constraint specifies a position or velocity that has a limited range. Inequality constraints are often used to model contact and impact: the constraint prevents penetration, but allows separation. For example a foot that is in contact with the ground, which may lose contact. The position of the foot is constrained to the ground or higher. Inequality constraints allow modelling of e.g. walking and jumping. By combining unconstrained systems (e.g. flying, floating), holonomic constrained systems (e.g. balancing), and inequality constrained systems (e.g. walking, jumping), all practical uses of humanoids and quadrupeds can be modelled. Once these models are available, advanced control techniques such as model-predictive control and trajectory optimization can be applied to a variety of realistic situations.

Appendix A

Mathematical Derivations

In this appendix we discuss two mathematical derivations. The first section presents the relation between the intrinsic and the apparent acceleration, which is used in the relation (4.2). The second section presents the chain rule for multivariable functions, which is applied in the identity relation (3.45).

A.1 Apparent and intrinsic acceleration

In this appendix we discuss two definitions of 6D acceleration: apparent and intrinsic acceleration. We apply the relation between both types of acceleration in the relation (4.2) in Section 4.1. We recall the definitions as in [30, Section 5.4].

The *apparent* acceleration of a frame B with respect to a frame A expressed in a frame C is defined as the time-derivative of the corresponding velocity ${}^{C}\mathbf{v}_{A,B}$,

$$ {}^{C}\dot{\mathbf{v}}_{A,B} := \frac{d}{dt}\left({}^{C}\mathbf{v}_{A,B}\right). \tag{A.1}$$

Note that the *velocity* transformation matrix does not translate an acceleration into a different frame. Expressing an apparent acceleration into a different frame involves an extra 6D cross product term. We also define the *intrinsic* acceleration of a frame B with respect to a frame A expressed in a frame C as

$$ {}^{C}\mathbf{a}_{A,B} := {}^{C}\mathbf{X}_{A}{}^{A}\dot{\mathbf{v}}_{A,B} = {}^{C}\mathbf{X}_{B}{}^{B}\dot{\mathbf{v}}_{A,B}. \tag{A.2}$$

In the relation (4.2) we look specifically at

$$ \dot{\mathbf{v}} := {}^{0}\dot{\mathbf{v}}_{A,0} = {}^{0}\mathbf{a}_{A,0} =: \mathbf{a}_{0}, \tag{A.3}$$

which is correct according to the definition of the intrinsic acceleration given by (A.2), as

$$ {}^{A}\mathbf{a}_{A,0} = {}^{A}\mathbf{X}_{A}{}^{A}\dot{\mathbf{v}}_{A,0}. \tag{A.4}$$

For the interested reader, we advice to read [30, Section 5.4], which talks about both types of acceleration in more detail using the Eindhoven-Genoa notation, and [42], which defines both types of acceleration in a different notation. Note that in the last citation, intrinsic acceleration is called *absolute* acceleration.

A.2 Chain rule for multivariable functions

In this appendix we present the chain rule for multivariable functions, and specifically apply it to identity relation in (3.45) in Section 3.3.

The chain rule is a method to compute the derivative of function compositions. Given two functions $f(x)$ and $g(x)$ evaluated at a given x, we write the derivative of their composition as

$$\frac{\partial(f \circ g)}{\partial x} = (\frac{\partial f}{\partial x} \circ g) \cdot \frac{\partial g}{\partial x}. \tag{A.5}$$

The chain rule can be used on multivariable functions as well. For example, given $h(x, y)$ and $y = k(x)$ evaluated at a given x and y, we write the partial derivatives of their composition $h(x, k(x))$ as

$$\frac{\partial h(x, k(x))}{\partial x} = \frac{\partial h(x, y)}{\partial x} + \frac{\partial h(x, y)}{\partial y} \frac{\partial k(x)}{\partial x}. \tag{A.6}$$

Using the derivative notation, we write

$$\frac{\partial h(x, k(x))}{\partial x} = \mathrm{D}_1\, h(x, y) + \mathrm{D}_2\, h(x, y)\, \mathrm{D}_1\, k(x). \tag{A.7}$$

This is used in (3.48), where the partial derivatives of two equations are written. The first equation is

$$\overline{ID}_b\big(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \overline{FD}_b(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \bar{\boldsymbol{\tau}}_b, \boldsymbol{\tau}), \overline{FD}_j(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \bar{\boldsymbol{\tau}}_b, \boldsymbol{\tau})\big) = \bar{\boldsymbol{\tau}}_b, \tag{A.8}$$

of which we want to compute the partial derivative with respect to $\mathbf{H}, \mathbf{s}, \mathbf{v}$ and $\mathbf{r}$. It should be noted that $\partial\bar{\boldsymbol{\tau}}_b/\partial x = 0$ where $x = \mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}$, as $\bar{\boldsymbol{\tau}}$ is assigned a specified value. We write the derivative of the left-hand side with respect to the first variable, $\mathbf{H}$, as

$$\mathrm{D}_1\, \overline{ID}_b\big(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \dot{\mathbf{v}}, \dot{\mathbf{r}}\big) + \mathrm{D}_5\, \overline{ID}_b\big(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \dot{\mathbf{v}}, \dot{\mathbf{r}}\big)\, \mathrm{D}_1\, \overline{FD}_b\big(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \bar{\boldsymbol{\tau}}_b, \boldsymbol{\tau}\big) \tag{A.9}$$

$$+ \mathrm{D}_6\, \overline{ID}_b\big(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \dot{\mathbf{v}}, \dot{\mathbf{r}}\big)\, \mathrm{D}_1\, \overline{FD}_j\big(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \bar{\boldsymbol{\tau}}_b, \boldsymbol{\tau}\big), \tag{A.10}$$

where we define

$$\dot{\mathbf{v}} := \overline{FD}_b(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \bar{\boldsymbol{\tau}}_b, \boldsymbol{\tau}) \tag{A.11}$$

and

$$\dot{\mathbf{r}} := \overline{FD}_j(\mathbf{H}, \mathbf{s}, \mathbf{v}, \mathbf{r}, \bar{\boldsymbol{\tau}}_b, \boldsymbol{\tau}). \tag{A.12}$$

The same can be done for the partial derivative with respect to the second, third, and fourth arguments, hence we replace D_1 with D_i and state $i \in \{1, 2, 3, 4\}$. The partial derivatives of the bottom equation of (3.48) are found in a similar manner.

Appendix B

Algorithmic Functions

This appendix presents three functions that are used in the algorithms in this thesis. These functions, 'get_gravity', 'jcalc' and 'jcalcderiv', retrieve the gravity, compute joint variables, and compute the derivative of the joint variables respectively.

B.1 Gravity

The algorithms in this thesis obtain the gravitational acceleration ${}^{A}\mathbf{a}_{grav}$ from the function 'get_gravity(model)'. The gravitational acceleration expressed in the inertial frame A is provided in the model. By specifying it in the model, one can easily change the gravitational acceleration vector if desired. The algorithms in this thesis allow for both translational and rotational acceleration, resulting in a 6D vector.

B.2 Joint calculation

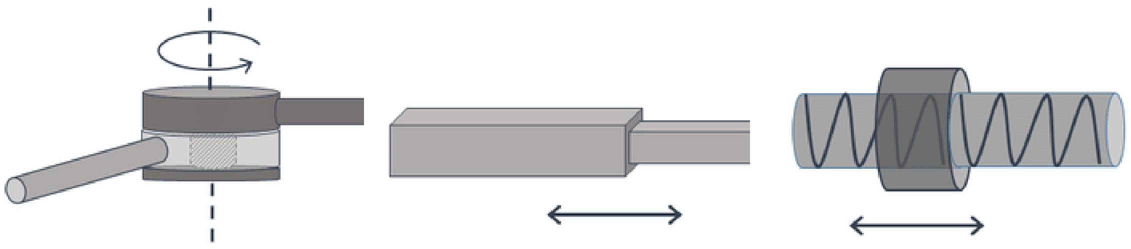
As shown in Assumption 2.2, in this thesis we assume all joints to be (modelled as joints) of the following types:

- Revolute (hinge joint) (1-DoF)
- Prismatic (sliding joint) (1-DoF)
- Helical (screw joint) (1-DoF)

These three 1-DoF joints are depicted in Figure B.1[1]. For each joint type, there is a specified velocity transformation matrix ${}^{i}\mathbf{X}_{\lambda(i)|i}$ and joint velocity subspace $\mathbf{\Gamma}_{\mathcal{J}i}$ as specified in Subsections 2.7.3 and 2.7.4. These quantities are computed by the function 'jcalc(jtype(i),$\mathbf{s}_i$)'. The input 'jtype(i)', specified by the model, is used to describe which of which type the joint is. The input $\mathbf{s}_i$ is used to specify the joints position. The velocity transformation matrix is given by

$$ {}^{i}\mathbf{X}_{\lambda(i)|i} = \begin{bmatrix} {}^{i}\mathbf{R}_{\lambda(i)|i} & {}^{i}\mathbf{o}^{\wedge}_{\lambda(i)|i}\,{}^{i}\mathbf{R}_{\lambda(i)|i} \\ 0_{3\times 3} & {}^{i}\mathbf{R}_{\lambda(i)|i} \end{bmatrix} \in \mathbb{R}^{6\times 6}, \tag{B.1} $$

[1]Pictures retrieved from `https://www.researchgate.net/figure/Revolute-R-prismatic-P-and-helical-H-joints_fig1_318418364`



(a) Revolute joint. (b) Prismatic joint. (c) Helical joint.

Figure B.1: All three 1-DoF types of joints.

where ${}^{i}\mathbf{R}_{\lambda(i)|i}$ and ${}^{i}\mathbf{o}^{\wedge}_{\lambda(i)|i}$ are specific to the joint type. We define the following rotation matrices, which are pure rotations in x, y and z direction, specified by angle $\mathbf{s}_i$

$$\mathbf{R}^x(\mathbf{s}_i) := \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\mathbf{s}_i & \sin\mathbf{s}_i \\ 0 & -\sin\mathbf{s}_i & \cos\mathbf{s}_i \end{bmatrix}, \tag{B.2}$$

$$\mathbf{R}^y(\mathbf{s}_i) := \begin{bmatrix} \cos\mathbf{s}_i & 0 & -\sin\mathbf{s}_i \\ 0 & 1 & 0 \\ \sin\mathbf{s}_i & 0 & \cos\mathbf{s}_i \end{bmatrix}, \tag{B.3}$$

$$\mathbf{R}^z(\mathbf{s}_i) := \begin{bmatrix} \cos\mathbf{s}_i & \sin\mathbf{s}_i & 0 \\ -\sin\mathbf{s}_i & \cos\mathbf{s}_i & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{B.4}$$

The joint types and their rotation matrix, origin and joint velocity subspace are shown in Table B.1, where x, y and z denote the axis around/in which the joint is moving.

Table B.1: Overview of joint types.

Joint type	**Rotation matrix** ${}^{i}\mathbf{R}_{\lambda(i)\|i}$	**Origin** ${}^{i}\mathbf{o}_{\lambda(i)\|i}$	**Joint velocity subspace** $\mathbf{\Gamma}_{\mathcal{J}i}$
Revolute	x: $\mathbf{R}^x(\mathbf{s}_i)$, y: $\mathbf{R}^y(\mathbf{s}_i)$, z: $\mathbf{R}^z(\mathbf{s}_i)$	$\begin{bmatrix}0\\0\\0\end{bmatrix}$	x: $\begin{bmatrix}0\\0\\0\\1\\0\\0\end{bmatrix}$, y: $\begin{bmatrix}0\\0\\0\\0\\1\\0\end{bmatrix}$, z: $\begin{bmatrix}0\\0\\0\\0\\0\\1\end{bmatrix}$
Prismatic	I_3	x: $\begin{bmatrix}\mathbf{s}_i\\0\\0\end{bmatrix}$, y: $\begin{bmatrix}0\\\mathbf{s}_i\\0\end{bmatrix}$, z: $\begin{bmatrix}0\\0\\\mathbf{s}_i\end{bmatrix}$	x: $\begin{bmatrix}1\\0\\0\\0\\0\\0\end{bmatrix}$, y: $\begin{bmatrix}0\\1\\0\\0\\0\\0\end{bmatrix}$, z: $\begin{bmatrix}0\\0\\1\\0\\0\\0\end{bmatrix}$
Helical (with pitch h)	x: $\mathbf{R}^x(\mathbf{s}_i)$, y: $\mathbf{R}^y(\mathbf{s}_i)$, z: $\mathbf{R}^z(\mathbf{s}_i)$	x: $\begin{bmatrix}h\,\mathbf{s}_i\\0\\0\end{bmatrix}$, y: $\begin{bmatrix}0\\h\,\mathbf{s}_i\\0\end{bmatrix}$, z: $\begin{bmatrix}0\\0\\h\,\mathbf{s}_i\end{bmatrix}$	x: $\begin{bmatrix}h\\0\\0\\1\\0\\0\end{bmatrix}$, y: $\begin{bmatrix}0\\h\\0\\0\\1\\0\end{bmatrix}$, z: $\begin{bmatrix}0\\0\\h\\0\\0\\1\end{bmatrix}$

In general, the time-derivative of the joint velocity subspace (see right column of Table B.1) $\dot{\mathbf{\Gamma}}_{\mathcal{J}i}$ is not equal to zero. For our joint types, however, it is equal to zero, as it is independent of time. In some versions of the RNEA and ABA, the time derivative of the joint subspace matrix $\dot{\mathbf{\Gamma}}_{\mathcal{J}i}$ is computed, however we omit the computation as it is equal to zero for our joint types.

B.3 Joint calculation derivatives

In the algorithm in Table 4.3, the derivative of the extended inverse dynamics (3.39) with respect to generalized position vector $\mathbf{s}$ is computed. Line 2 calls the function 'jcalcderiv(jtype(i),$\mathbf{s}_i$)' to compute the derivative of the velocity transformation matrix ${}^{i}\mathbf{X}_{\lambda(i)|i}$ with respect to $\mathbf{s}_i$. Here we make use of the definition of ${}^{i}\mathbf{X}_{\lambda(i)|i}$ given in B.2. As it only depends on the joint type and the generalized position $\mathbf{s}_i$, we can mathematically write out its derivative with respect to $\mathbf{s}_i$. In the table below we show the outcome partial derivative of ${}^{i}\mathbf{X}_{\lambda(i)|i}$ with respect to $\mathbf{s}_i$ for each joint type, where the functions c and s represent cosine and sine respectively.

Table B.2: Derivative of ${}^{i}\mathbf{X}_{\lambda(i)|i}$ w.r.t. $\mathbf{s}_i$.

Joint type	**Derivative** $\partial\, {}^{i}\mathbf{X}_{\lambda(i)\|i} / \partial \mathbf{s}_i$
Revolute around x	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -s(\mathbf{s}_i) & c(\mathbf{s}_i) & 0 & 0 & 0 \\ 0 & -c(\mathbf{s}_i) & -s(\mathbf{s}_i) & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -s(\mathbf{s}_i) & c(\mathbf{s}_i) \\ 0 & 0 & 0 & 0 & -c(\mathbf{s}_i) & -s(\mathbf{s}_i) \end{bmatrix}$
Revolute around y	$\begin{bmatrix} -s(\mathbf{s}_i) & 0 & -c(\mathbf{s}_i) & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ c(\mathbf{s}_i) & 0 & -s(\mathbf{s}_i) & 0 & 0 & 0 \\ 0 & 0 & 0 & -s(\mathbf{s}_i) & 0 & -c(\mathbf{s}_i) \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & c(\mathbf{s}_i) & 0 & -s(\mathbf{s}_i) \end{bmatrix}$
Revolute around z	$\begin{bmatrix} -s(\mathbf{s}_i) & c(\mathbf{s}_i) & 0 & 0 & 0 & 0 \\ -c(\mathbf{s}_i) & -s(\mathbf{s}_i) & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -s(\mathbf{s}_i) & c(\mathbf{s}_i) & 0 \\ 0 & 0 & 0 & -c(\mathbf{s}_i) & -s(\mathbf{s}_i) & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$
Prismatic in x	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$
Prismatic in y	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$
Prismatic in z	$\begin{bmatrix} 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$

continued on the next page.

Joint type	**Derivative** $\partial\, {}^i\mathbf{X}_{\lambda(i)\|i} / \partial \mathbf{s}_i$
Helical in x (with pitch h)	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -s(\mathbf{s}_i) & c(\mathbf{s}_i) & 0 & h\, s(\mathbf{s}_i) + h\mathbf{s}_i\, c(\mathbf{s}_i) & h\mathbf{s}_i\, s(\mathbf{s}_i) - h\, c(\mathbf{s}_i) \\ 0 & -c(\mathbf{s}_i) & -s(\mathbf{s}_i) & 0 & h\, c(\mathbf{s}_i) - h\mathbf{s}_i\, s(\mathbf{s}_i) & h\, s(\mathbf{s}_i) + h\mathbf{s}_i\, c(\mathbf{s}_i) \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -s(\mathbf{s}_i) & c(\mathbf{s}_i) \\ 0 & 0 & 0 & 0 & -c(\mathbf{s}_i) & -s(\mathbf{s}_i) \end{bmatrix}$
Helical in y (with pitch h)	$\begin{bmatrix} -s(\mathbf{s}_i) & 0 & -c(\mathbf{s}_i) & h\, s(\mathbf{s}_i) + h\mathbf{s}_i\, c(\mathbf{s}_i) & 0 & h\, c(\mathbf{s}_i) - h\mathbf{s}_i\, s(\mathbf{s}_i) \\ 0 & 0 & 0 & 0 & 0 & 0 \\ c(\mathbf{s}_i) & 0 & -s(\mathbf{s}_i) & h\mathbf{s}_i\, s(\mathbf{s}_i) - h\, c(\mathbf{s}_i) & 0 & h\, s(\mathbf{s}_i) + h\mathbf{s}_i\, c(\mathbf{s}_i) \\ 0 & 0 & 0 & -s(\mathbf{s}_i) & 0 & -c(\mathbf{s}_i) \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & c(\mathbf{s}_i) & 0 & -s(\mathbf{s}_i) \end{bmatrix}$
Helical in z (with pitch h)	$\begin{bmatrix} -s(\mathbf{s}_i) & c(\mathbf{s}_i) & 0 & h\, s(\mathbf{s}_i) + h\mathbf{s}_i\, c(\mathbf{s}_i) & h\mathbf{s}_i\, s(\mathbf{s}_i) - h\, c(\mathbf{s}_i) & 0 \\ -c(\mathbf{s}_i) & -s(\mathbf{s}_i) & 0 & h\, c(\mathbf{s}_i) - h\mathbf{s}_i\, s(\mathbf{s}_i) & h\, s(\mathbf{s}_i) + h\mathbf{s}_i\, c(\mathbf{s}_i) & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -s(\mathbf{s}_i) & c(\mathbf{s}_i) & 0 \\ 0 & 0 & 0 & -c(\mathbf{s}_i) & -s(\mathbf{s}_i) & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$

Appendix C

Correction of the IMMA

In this section we report, detail and correct the IMMA as it is presented in [28]. Table C.1 shows the original version of the IMMA in the Eindhoven-Genoa notation, where the line 34 to 38 are not presented but implied in [28].

Table C.1: Original version of the IMMA in the Eindhoven-Genoa notation.

```
Inputs   model, $\mathbf{s}$
Line     IMMA (original version)
1        for $i = 1$ to $n_B$ do
2          $[{}^{i}\mathbf{X}_{\lambda(i)|i}, \mathbf{\Gamma}_{\mathcal{J}i}] = \text{jcalc}(\text{jtype}(i), \mathbf{s}_i)$
3          ${}^{i}\mathbf{X}_{\lambda(i)} = {}^{i}\mathbf{X}_{\lambda(i)|i}\,{}^{\lambda(i)|i}\mathbf{X}_{\lambda(i)}$
4          $\mathbb{M}^{c}_{\mathcal{B}i} = \mathbb{M}_{\mathcal{B}i}$
5        end
6        for $i = N_B$ to $1$ do
7          $\mathbf{U}_{\mathcal{B}i} = \mathbb{M}^{A}_{\mathcal{B}i}\mathbf{\Gamma}_{\mathcal{J}i}$
8          $\mathbf{D}_{\mathcal{B}i} = \mathbf{\Gamma}^{T}_{\mathcal{J}i}\mathbf{U}_{\mathcal{B}i}$
9          if $i$ is not a leaf then
10           $\mathbf{M}^{inv}_{fb}[i, subtree(i)] = -\mathbf{D}^{-1}_{\mathcal{B}i}\mathbf{\Gamma}^{T}_{\mathcal{J}i}\mathcal{F}_i[:, subtree(i)]$
11           if $\lambda(i) \neq 0$ then
12             $\mathcal{F}_i[:, subtree(i)] = \mathbf{U}_{\mathcal{B}i}\,\mathbf{M}^{inv}_{fb}[i, subtree(i)]$
13             $\mathcal{F}_{\lambda(i)}[:, subtree(i)] = \mathcal{F}_{\lambda(i)}[:, subtree(i)] + {}_{\lambda(i)}\mathbf{X}^{i}\big(\mathcal{F}_i[:, subtree(i)]$
14           end
15         end
16         else
17           $\mathcal{F}_{\lambda(i)}[:, i] = \mathcal{F}_{\lambda(i)}[:, i] + {}_{\lambda(i)}\mathbf{X}^{i}\mathbf{U}_{\mathcal{B}i}\,\mathbf{M}^{inv}_{fb}[i, i]$
18         end
19         $\mathbb{M}^{inv}_{fb}[i, i] = \mathbf{D}^{-1}_{\mathcal{B}i}$
20         if $\lambda(i) \neq 0$ then
21           $\mathbb{M}^{a}_{\mathcal{B}i} = \mathbb{M}^{A}_{\mathcal{B}i} - \mathbf{U}_{\mathcal{B}i}\mathbf{D}^{-1}_{\mathcal{B}i}\mathbf{U}^{T}_{\mathcal{B}i}$
22           $\mathbb{M}^{A}_{\mathcal{B}\lambda(i)} = \mathbb{M}^{A}_{\mathcal{B}\lambda(i)} + {}_{\lambda(i)}\mathbf{X}^{i}\mathbb{M}^{a}_{\mathcal{B}i}\,{}^{i}\mathbf{X}_{\lambda(i)}$
23         end
24       end
25       for $i = 1$ to $n_B$ do
26         if $\lambda(i) \neq 0$ then
27           $\mathbf{M}^{inv}_{fb}[i, subtree(i)] = \mathbf{M}^{inv}_{fb}[i, subtree(i)] - \mathbf{D}^{-1}_{\mathcal{B}i}\mathbf{U}^{T}_{\mathcal{B}i}\,{}^{i}\mathbf{X}_{\lambda(i)}\mathcal{P}_{\lambda(i)}[:, subtree(i)]$
28         end
29         $\mathcal{P}_i = \mathbf{\Gamma}_{\mathcal{J}i}\mathbf{M}^{inv}_{fb}[i, subtree(i)]$
30         if $\lambda(i) \neq 0$ then
31           $\mathcal{P}_i[:, subtree(i)] = \mathcal{P}_i[:, subtree(i)] + {}^{i}\mathbf{X}_{\lambda(i)}\mathcal{P}_{\lambda(i)}[:, subtree(i)]$
32         end
33       end
34       for $i = 1$ to $n_B$ do
35         for $j = i$ to $n_B$ do
36           $\mathbf{M}^{inv}_{fb}[j, i] = \mathbf{M}^{inv}_{fb}[i, j]$
37         end
38       end
Outputs  $\mathbf{M}^{inv}_{fb}$
```

The code as it is presented can not be executed, since the *end* in line 15 should not be there. After removing line 15, there is the problem that $\mathbf{M}_{fb}^{inv}[i,i]$ gets defined on line 19, although it should be defined between line 8 and 9. Defining $\mathbf{M}_{fb}^{inv}[i,i]$ on line 19 returns a (falsely) diagonal inverse mass matrix. Besides these errors, the algorithm can be simplified a lot. The *if i is not a leaf* can be completely dropped, which saves us from needing to compute whether each joint is a leaf. The definition of $\mathcal{F}_i$, which is now done on three lines, can be done on a single line as $\mathcal{F}_{\lambda(i)}[:, subtree(i)] = \mathcal{F}_{\lambda(i)}[:, subtree(i)] + {}_{\lambda(i)}\mathbf{X}^i\mathbf{U}_{\mathcal{B}i}\mathbf{M}_{fb}^{inv}[i, model.subtree(i)]$. All these changes are in the backwards pass. For clarity, the optimized version of the backwards pass is written below.

Table C.2: Optimized backwards pass of the original IMMA.

Line	Backwards pass of IMMA
6	**for** $i = n_B$ **to** 1 **do**
7	$\mathbf{U}_{\mathcal{B}i} = \mathbb{M}_{\mathcal{B}i}^{A}\mathbf{\Gamma}_{\mathcal{J}i}$
8	$\mathbf{D}_{\mathcal{B}i} = \mathbf{\Gamma}_{\mathcal{J}i}^{T}\mathbf{U}_{\mathcal{B}i}$
9	$\mathbf{M}_{fb}^{inv}[i,i] = \mathbf{D}_{\mathcal{B}i}^{-1}$
10	$\mathbf{M}_{fb}^{inv}[i, subtree(i)] = \mathbf{M}_{fb}^{inv}[i, subtree(i)] - \mathbf{D}_{\mathcal{B}i}^{-1}\mathbf{\Gamma}_{\mathcal{J}i}^{T}\mathcal{F}_i[:, subtree(i)]$
11	**if** $\lambda(i) \neq 0$ **then**
12	$\mathcal{F}_{\lambda(i)}[:, subtree(i)] = \mathcal{F}_{\lambda(i)}[:, subtree(i)] + {}_{\lambda(i)}\mathbf{X}^i\mathbf{U}_{\mathcal{B}i}\mathbf{M}_{fb}^{inv}[i, subtree(i)]$
13	$\mathbb{M}_{\mathcal{B}i}^{a} = \mathbb{M}_{\mathcal{B}i}^{A} - \mathbf{U}_{\mathcal{B}i}\mathbf{D}_{\mathcal{B}i}^{-1}\mathbf{U}_{\mathcal{B}i}^{T}$
14	$\mathbb{M}_{\mathcal{B}\lambda(i)}^{A} = \mathbb{M}_{\mathcal{B}\lambda(i)}^{A} + {}_{\lambda(i)}\mathbf{X}^i\,\mathbb{M}_{\mathcal{B}i}^{a}\,{}^{i}\mathbf{X}_{\lambda(i)}$
15	**end**
16	**end**

This algorithm returns the same result as the original version in [28] (provided that the extra *end* in line 15 is removed and the $\mathbf{M}_{fb}^{inv}$ from line 19 is moved between lines 8 and 9). Instead of the original 18 lines, this algorithm only uses 11 lines of code. However, with the added changes, the resulting inverse mass matrix obtained through this algorithm is still incorrect. A term is missing in line 12 of the optimized backwards pass in Table C.2, namely ${}_{\lambda(i)}\mathbf{X}^i\mathcal{F}_i[:, subtree(i)]$. With this term added, line 12 is shown in the following table.

Table C.3: Correct line 12 of the IMMA.

Line	Line 12 of IMMA
12	$\mathcal{F}_{\lambda(i)}[:, subtree(i)] = \mathcal{F}_{\lambda(i)}[:, subtree(i)] + {}_{\lambda(i)}\mathbf{X}^i\big(\mathcal{F}_i[:, subtree(i)] + \mathbf{U}_{\mathcal{B}i}\mathbf{M}_{fb}^{inv}[i, subtree(i)]\big)$

Furthermore, when dealing with branched systems, the second forward pass of the original IMMA is incorrect. This mistake is corrected in the second forward pass below.

Table C.4: Correct second forward pass of the IMMA.

Line	**Second forward pass of IMMA**
25	**for** $i = 1$ **to** n_B **do**
26	**if** $\lambda(i) \neq 0$ **then**
27	$\mathbf{M}_{fb}^{inv}[i,:] = \mathbf{M}_{fb}^{inv}[i,:] - \mathbf{D}_{\mathcal{B}i}^{-1}\mathbf{U}_{\mathcal{B}i}^{T}\,{}^{i}\mathbf{X}_{\lambda(i)}\mathcal{P}_{\lambda(i)}$
28	**end**
29	$\mathcal{P}_i = \mathbf{\Gamma}_{\mathcal{J}i}\mathbf{M}_{fb}^{inv}[i,:]$
30	**if** $\lambda(i) \neq 0$ **then**
31	$\mathcal{P}_i = \mathcal{P}_i + {}^{i}\mathbf{X}_{\lambda(i)}\mathcal{P}_{\lambda(i)}$
32	**end**
33	**end**

The resulting correct IMMA is presented in Table 4.6 in Section 4.3, and it is verified to be correct in Section 5.3.

Appendix D

Details on Numerical Verification

In this appendix we present details on the numerical verification. The first section shows the details of the designed test system which is used in Chapter 5. The second section presents the numerical results of the inverse of the mass matrix for moving-base systems, as the resulting matrices are too big to display in the main text.

D.1 Details on the designed test system

This section shows the details on the designed test system of Section 5.1. The joint types, the velocity transformation matrices ${}^{\lambda(i)|i}\mathbf{X}_{\lambda(i)}$ (which are body fixed constants), and the rigid body inertia's $\mathbb{M}_{\mathcal{B}i}$ are presented below. The joint types of the designed test system are shown in Table D.1. The different types of joints are specified in Appendix B.2.

Table D.1: Joint types of the designed test system.

Joint	**Type**	**Direction**	**Extra details**
1	Prismatic	x	
2	Revolute	y	
3	Helical	z	pitch $h = 4$
4	Helical	x	pitch $h = 5$
5	Revolute	z	
6	Prismatic	y	
7	Revolute	x	
8	Prismatic	z	
9	Helical	y	pitch $h = 3$

The velocity transformation matrices ${}^{\lambda(i)|i}\mathbf{X}_{\lambda(i)}$ are body fixed constants. It can physically be seen as the velocity transformation matrix from joint $\lambda(i)$, resembled by frame $\lambda(i)$ to joint i, resembled by frame $\lambda(i)|i$. As both frames are attached to body $\lambda(i)$, the velocity transformation matrix between them is constant, and since it is a constant, it is modelled as a model design parameter. The velocity transformation matrices for the designed test system of Section 5.1 are chosen to be a pure rotation multiplied by a pure translation, so that the velocity transformation matrix of (2.36) is given by

$$^{\lambda(i)|i}\mathbf{X}_{\lambda(i)} = \begin{bmatrix} ^{\lambda(i)|i}\mathbf{R}_{\lambda(i)} & 0_{3\times3} \\ 0_{3\times3} & ^{\lambda(i)|i}\mathbf{R}_{\lambda(i)} \end{bmatrix} \begin{bmatrix} I_3 & ^{\lambda(i)|i}\mathbf{o}^{\wedge}_{\lambda(i)} \\ 0_{3\times3} & I_3 \end{bmatrix} \in \mathbb{R}^{6\times6}. \tag{D.1}$$

The velocity transformation matrices of the designed test system are shown in Table D.2 where $\mathbf{R}^x, \mathbf{R}^y$ and $\mathbf{R}^z$ are defined in (B.2)-(B.4).

Table D.2: Velocity transformation matrices of the designed test system.

Body	$^{\lambda(i)\|i}\mathbf{R}_{\lambda(i)}$	$^{\lambda(i)\|i}\mathbf{o}^{\wedge}_{\lambda(i)}$
1	$\mathbf{R}^x(1)$	$[2;1;7]$
2	$\mathbf{R}^z(4)$	$[1;7;6]$
3	$\mathbf{R}^y(7)$	$[8;9;3]$
4	$\mathbf{R}^x(8)$	$[7;0;7]$
5	$\mathbf{R}^y(5)$	$[6;5;8]$
6	$\mathbf{R}^x(1)$	$[3;8;0]$
7	$\mathbf{R}^x(0)$	$[4;3;2]$
8	$\mathbf{R}^z(6)$	$[7;5;1]$
9	$\mathbf{R}^y(4)$	$[3;2;2]$

The rigid body inertia's $\mathbb{M}_{\mathcal{B}i}$ are presented in Table D.3.

Table D.3: Rigid body inertia's of the designed test system.

$$\mathbb{M}_{\mathcal{B}0} = \begin{bmatrix} 120 & -23 & -68 & 0 & -14 & 6 \\ -23 & 153 & -39 & 14 & 0 & -10 \\ -68 & -39 & 76 & -6 & 10 & 0 \\ 0 & 14 & -6 & 2 & 0 & 0 \\ -14 & 0 & 10 & 0 & 2 & 0 \\ 6 & -10 & 0 & 0 & 0 & 2 \end{bmatrix} \quad \mathbb{M}_{\mathcal{B}1} = \begin{bmatrix} 129 & -71 & -138 & 0 & -24 & 12 \\ -71 & 314 & -43 & 24 & 0 & -36 \\ -138 & -43 & 241 & -12 & 36 & 0 \\ 0 & 24 & -12 & 6 & 0 & 0 \\ -24 & 0 & 36 & 0 & 6 & 0 \\ 12 & -36 & 0 & 0 & 0 & 6 \end{bmatrix}$$

$$\mathbb{M}_{\mathcal{B}2} = \begin{bmatrix} 238 & -74 & -403 & 0 & -45 & 9 \\ -74 & 959 & -42 & 45 & 0 & -81 \\ -403 & -42 & 746 & -9 & 81 & 0 \\ 0 & 45 & -9 & 9 & 0 & 0 \\ -45 & 0 & 81 & 0 & 9 & 0 \\ 9 & -81 & 0 & 0 & 0 & 9 \end{bmatrix} \quad \mathbb{M}_{\mathcal{B}3} = \begin{bmatrix} 852 & -113 & -214 & 0 & -72 & 40 \\ -113 & 725 & -357 & 72 & 0 & -24 \\ -214 & -357 & 280 & -40 & 24 & 0 \\ 0 & 72 & -40 & 8 & 0 & 0 \\ -72 & 0 & 24 & 0 & 8 & 0 \\ 40 & -24 & 0 & 0 & 0 & 8 \end{bmatrix}$$

$$\mathbb{M}_{\mathcal{B}4} = \begin{bmatrix} 45 & -41 & -36 & 0 & -6 & 6 \\ -41 & 118 & -13 & 6 & 0 & -14 \\ -36 & -13 & 117 & -6 & 14 & 0 \\ 0 & 6 & -6 & 2 & 0 & 0 \\ -6 & 0 & 14 & 0 & 2 & 0 \\ 6 & -14 & 0 & 0 & 0 & 2 \end{bmatrix} \quad \mathbb{M}_{\mathcal{B}5} = \begin{bmatrix} 122 & -34 & -34 & 0 & -8 & 7 \\ -34 & 91 & -51 & 8 & 0 & -5 \\ -34 & -51 & 75 & -7 & 5 & 0 \\ 0 & 8 & -7 & 1 & 0 & 0 \\ -8 & 0 & 5 & 0 & 1 & 0 \\ 7 & -5 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbb{M}_{\mathcal{B}6} = \begin{bmatrix} 94 & -83 & -88 & 0 & -15 & 15 \\ -83 & 230 & -42 & 15 & 0 & -30 \\ -88 & -42 & 233 & -15 & 30 & 0 \\ 0 & 15 & -15 & 5 & 0 & 0 \\ -15 & 0 & 30 & 0 & 5 & 0 \\ 15 & -30 & 0 & 0 & 0 & 5 \end{bmatrix} \quad \mathbb{M}_{\mathcal{B}7} = \begin{bmatrix} 168 & -121 & -158 & 0 & -20 & 16 \\ -121 & 361 & -77 & 20 & 0 & -32 \\ -158 & -77 & 328 & -16 & 32 & 0 \\ 0 & 20 & -16 & 4 & 0 & 0 \\ -20 & 0 & 32 & 0 & 4 & 0 \\ 16 & -32 & 0 & 0 & 0 & 4 \end{bmatrix}$$

$$\mathbb{M}_{\mathcal{B}8} = \begin{bmatrix} 569 & -503 & -246 & 0 & -28 & 56 \\ -503 & 681 & -219 & 28 & 0 & -63 \\ -246 & -219 & 1016 & -56 & 63 & 0 \\ 0 & 28 & -56 & 7 & 0 & 0 \\ -28 & 0 & 63 & 0 & 7 & 0 \\ 56 & -63 & 0 & 0 & 0 & 7 \end{bmatrix} \quad \mathbb{M}_{\mathcal{B}9} = \begin{bmatrix} 184 & -11 & -4 & 0 & -6 & 18 \\ -11 & 25 & -51 & 6 & 0 & -2 \\ -4 & -51 & 172 & -18 & 2 & 0 \\ 0 & 6 & -18 & 2 & 0 & 0 \\ -6 & 0 & 2 & 0 & 2 & 0 \\ 18 & -2 & 0 & 0 & 0 & 2 \end{bmatrix}$$

D.2 Numerical results of the inverse of the mass matrix for moving-base systems

The inverse of the mass matrix $\mathbf{M}^{-1}$ is found through the CRBAmb (Appendix E.4) and through the IMMAmb of Table 4.7 as shown below.

$$\mathbf{M}_{CRBAmb}^{-1} \approx$$
$$\begin{pmatrix}
2.1\text{E}{-3} & 6.4\text{E}{-5} & 3.7\text{E}{-4} & -1.8\text{E}{-5} & -1.7\text{E}{-6} & -1.6\text{E}{-5} & -1.9\text{E}{-3} & 2.4\text{E}{-5} & 5.0\text{E}{-4} & -3.4\text{E}{-6} & 0.015 & -6.9\text{E}{-4} & 4.4\text{E}{-5} & 5.5\text{E}{-4} & -1.0\text{E}{-3} \\
6.4\text{E}{-5} & 9.6\text{E}{-4} & -3.1\text{E}{-4} & -3.7\text{E}{-5} & 6.2\text{E}{-6} & 9.8\text{E}{-6} & 3.5\text{E}{-6} & 5.8\text{E}{-5} & 4.0\text{E}{-4} & 3.7\text{E}{-5} & -1.8\text{E}{-3} & 4.3\text{E}{-4} & 1.5\text{E}{-5} & 7.4\text{E}{-5} & 9.8\text{E}{-4} \\
3.7\text{E}{-4} & -3.1\text{E}{-4} & 1.5\text{E}{-3} & 3.8\text{E}{-6} & -1.2\text{E}{-5} & -5.2\text{E}{-6} & 1.5\text{E}{-4} & 1.0\text{E}{-5} & 9.1\text{E}{-4} & 9.3\text{E}{-6} & 1.8\text{E}{-3} & 8.8\text{E}{-5} & 6.7\text{E}{-6} & -1.1\text{E}{-3} & -5.4\text{E}{-4} \\
-1.8\text{E}{-5} & -3.7\text{E}{-5} & 3.8\text{E}{-6} & 5.6\text{E}{-6} & -6.1\text{E}{-7} & 2.2\text{E}{-7} & -1.6\text{E}{-5} & -6.5\text{E}{-6} & -6.3\text{E}{-5} & -7.0\text{E}{-6} & -1.4\text{E}{-4} & -2.0\text{E}{-5} & -4.8\text{E}{-6} & 2.2\text{E}{-5} & -3.7\text{E}{-5} \\
-1.7\text{E}{-6} & 6.2\text{E}{-6} & -1.2\text{E}{-5} & -6.1\text{E}{-7} & 5.7\text{E}{-7} & 9.1\text{E}{-8} & 5.9\text{E}{-6} & 1.2\text{E}{-7} & -1.5\text{E}{-5} & 2.1\text{E}{-6} & -1.1\text{E}{-4} & 2.7\text{E}{-5} & 3.3\text{E}{-7} & -3.6\text{E}{-6} & 1.7\text{E}{-5} \\
-1.6\text{E}{-5} & 9.8\text{E}{-6} & -5.2\text{E}{-6} & 2.2\text{E}{-7} & 9.1\text{E}{-8} & 6.9\text{E}{-7} & 1.4\text{E}{-5} & -4.5\text{E}{-8} & -6.9\text{E}{-6} & -3.3\text{E}{-7} & -4.6\text{E}{-4} & 4.9\text{E}{-5} & -4.9\text{E}{-7} & -1.5\text{E}{-6} & 1.6\text{E}{-5} \\
-1.9\text{E}{-3} & 3.5\text{E}{-6} & 1.5\text{E}{-4} & -1.6\text{E}{-5} & 5.9\text{E}{-6} & 1.4\text{E}{-5} & 3.0\text{E}{-3} & -8.7\text{E}{-7} & -3.6\text{E}{-4} & 6.7\text{E}{-5} & -0.015 & 1.2\text{E}{-3} & -1.2\text{E}{-5} & -1.2\text{E}{-3} & 1.1\text{E}{-3} \\
2.4\text{E}{-5} & 5.8\text{E}{-5} & 1.0\text{E}{-5} & -6.5\text{E}{-6} & 1.2\text{E}{-7} & -4.5\text{E}{-8} & -8.7\text{E}{-7} & 1.0\text{E}{-5} & 1.4\text{E}{-4} & 6.3\text{E}{-6} & 1.9\text{E}{-4} & 9.7\text{E}{-6} & 5.3\text{E}{-6} & -2.5\text{E}{-5} & 4.6\text{E}{-5} \\
5.0\text{E}{-4} & 4.0\text{E}{-4} & 9.1\text{E}{-4} & -6.3\text{E}{-5} & -1.5\text{E}{-5} & -6.9\text{E}{-6} & -3.6\text{E}{-4} & 1.4\text{E}{-4} & 3.1\text{E}{-3} & 3.5\text{E}{-5} & 6.9\text{E}{-3} & -5.9\text{E}{-4} & 6.0\text{E}{-5} & -5.9\text{E}{-4} & -7.9\text{E}{-5} \\
-3.4\text{E}{-6} & 3.7\text{E}{-5} & 9.3\text{E}{-6} & -7.0\text{E}{-6} & 2.1\text{E}{-6} & -3.3\text{E}{-7} & 6.7\text{E}{-5} & 6.3\text{E}{-6} & 3.5\text{E}{-5} & 2.2\text{E}{-5} & 6.3\text{E}{-5} & 1.2\text{E}{-4} & 5.2\text{E}{-6} & -8.3\text{E}{-5} & 8.4\text{E}{-5} \\
0.015 & -1.8\text{E}{-3} & 1.8\text{E}{-3} & -1.4\text{E}{-4} & -1.1\text{E}{-4} & -4.6\text{E}{-4} & -0.015 & 1.9\text{E}{-4} & 6.9\text{E}{-3} & 6.3\text{E}{-5} & 1.4 & -0.039 & 2.6\text{E}{-4} & 5.5\text{E}{-3} & -9.0\text{E}{-3} \\
-6.9\text{E}{-4} & 4.3\text{E}{-4} & 8.8\text{E}{-5} & -2.0\text{E}{-5} & 2.7\text{E}{-5} & 4.9\text{E}{-5} & 1.2\text{E}{-3} & 9.7\text{E}{-6} & -5.9\text{E}{-4} & 1.2\text{E}{-4} & -0.039 & 9.8\text{E}{-3} & 9.9\text{E}{-6} & -9.0\text{E}{-4} & 1.1\text{E}{-3} \\
4.4\text{E}{-5} & 1.5\text{E}{-5} & 6.7\text{E}{-6} & -4.8\text{E}{-6} & 3.3\text{E}{-7} & -4.9\text{E}{-7} & -1.2\text{E}{-5} & 5.3\text{E}{-6} & 6.0\text{E}{-5} & 5.2\text{E}{-6} & 2.6\text{E}{-4} & 9.9\text{E}{-6} & 6.5\text{E}{-6} & 8.1\text{E}{-6} & -7.8\text{E}{-5} \\
5.5\text{E}{-4} & 7.4\text{E}{-5} & -1.1\text{E}{-3} & 2.2\text{E}{-5} & -3.6\text{E}{-6} & -1.5\text{E}{-6} & -1.2\text{E}{-3} & -2.5\text{E}{-5} & -5.9\text{E}{-4} & -8.3\text{E}{-5} & 5.5\text{E}{-3} & -9.0\text{E}{-4} & 8.1\text{E}{-6} & 2.6\text{E}{-3} & -6.6\text{E}{-4} \\
-1.0\text{E}{-3} & 9.8\text{E}{-4} & -5.4\text{E}{-4} & -3.7\text{E}{-5} & 1.7\text{E}{-5} & 1.6\text{E}{-5} & 1.1\text{E}{-3} & 4.6\text{E}{-5} & -7.9\text{E}{-5} & 8.4\text{E}{-5} & -9.0\text{E}{-3} & 1.1\text{E}{-3} & -7.8\text{E}{-5} & -6.6\text{E}{-4} & 0.01
\end{pmatrix}$$

$$\mathbf{M}_{Alg}^{-1} \approx$$
$$\begin{pmatrix}
2.1\text{E}{-3} & 6.4\text{E}{-5} & 3.7\text{E}{-4} & -1.8\text{E}{-5} & -1.7\text{E}{-6} & -1.6\text{E}{-5} & -1.9\text{E}{-3} & 2.4\text{E}{-5} & 5.0\text{E}{-4} & -3.4\text{E}{-6} & 0.015 & -6.9\text{E}{-4} & 4.4\text{E}{-5} & 5.5\text{E}{-4} & -1.0\text{E}{-3} \\
6.4\text{E}{-5} & 9.6\text{E}{-4} & -3.1\text{E}{-4} & -3.7\text{E}{-5} & 6.2\text{E}{-6} & 9.8\text{E}{-6} & 3.5\text{E}{-6} & 5.8\text{E}{-5} & 4.0\text{E}{-4} & 3.7\text{E}{-5} & -1.8\text{E}{-3} & 4.3\text{E}{-4} & 1.5\text{E}{-5} & 7.4\text{E}{-5} & 9.8\text{E}{-4} \\
3.7\text{E}{-4} & -3.1\text{E}{-4} & 1.5\text{E}{-3} & 3.8\text{E}{-6} & -1.2\text{E}{-5} & -5.2\text{E}{-6} & 1.5\text{E}{-4} & 1.0\text{E}{-5} & 9.1\text{E}{-4} & 9.3\text{E}{-6} & 1.8\text{E}{-3} & 8.8\text{E}{-5} & 6.7\text{E}{-6} & -1.1\text{E}{-3} & -5.4\text{E}{-4} \\
-1.8\text{E}{-5} & -3.7\text{E}{-5} & 3.8\text{E}{-6} & 5.6\text{E}{-6} & -6.1\text{E}{-7} & 2.2\text{E}{-7} & -1.6\text{E}{-5} & -6.5\text{E}{-6} & -6.3\text{E}{-5} & -7.0\text{E}{-6} & -1.4\text{E}{-4} & -2.0\text{E}{-5} & -4.8\text{E}{-6} & 2.2\text{E}{-5} & -3.7\text{E}{-5} \\
-1.7\text{E}{-6} & 6.2\text{E}{-6} & -1.2\text{E}{-5} & -6.1\text{E}{-7} & 5.7\text{E}{-7} & 9.1\text{E}{-8} & 5.9\text{E}{-6} & 1.2\text{E}{-7} & -1.5\text{E}{-5} & 2.1\text{E}{-6} & -1.1\text{E}{-4} & 2.7\text{E}{-5} & 3.3\text{E}{-7} & -3.6\text{E}{-6} & 1.7\text{E}{-5} \\
-1.6\text{E}{-5} & 9.8\text{E}{-6} & -5.2\text{E}{-6} & 2.2\text{E}{-7} & 9.1\text{E}{-8} & 6.9\text{E}{-7} & 1.4\text{E}{-5} & -4.5\text{E}{-8} & -6.9\text{E}{-6} & -3.3\text{E}{-7} & -4.6\text{E}{-4} & 4.9\text{E}{-5} & -4.9\text{E}{-7} & -1.5\text{E}{-6} & 1.6\text{E}{-5} \\
-1.9\text{E}{-3} & 3.5\text{E}{-6} & 1.5\text{E}{-4} & -1.6\text{E}{-5} & 5.9\text{E}{-6} & 1.4\text{E}{-5} & 3.0\text{E}{-3} & -8.7\text{E}{-7} & -3.6\text{E}{-4} & 6.7\text{E}{-5} & -0.015 & 1.2\text{E}{-3} & -1.2\text{E}{-5} & -1.2\text{E}{-3} & 1.1\text{E}{-3} \\
2.4\text{E}{-5} & 5.8\text{E}{-5} & 1.0\text{E}{-5} & -6.5\text{E}{-6} & 1.2\text{E}{-7} & -4.5\text{E}{-8} & -8.7\text{E}{-7} & 1.0\text{E}{-5} & 1.4\text{E}{-4} & 6.3\text{E}{-6} & 1.9\text{E}{-4} & 9.7\text{E}{-6} & 5.3\text{E}{-6} & -2.5\text{E}{-5} & 4.6\text{E}{-5} \\
5.0\text{E}{-4} & 4.0\text{E}{-4} & 9.1\text{E}{-4} & -6.3\text{E}{-5} & -1.5\text{E}{-5} & -6.9\text{E}{-6} & -3.6\text{E}{-4} & 1.4\text{E}{-4} & 3.1\text{E}{-3} & 3.5\text{E}{-5} & 6.9\text{E}{-3} & -5.9\text{E}{-4} & 6.0\text{E}{-5} & -5.9\text{E}{-4} & -7.9\text{E}{-5} \\
-3.4\text{E}{-6} & 3.7\text{E}{-5} & 9.3\text{E}{-6} & -7.0\text{E}{-6} & 2.1\text{E}{-6} & -3.3\text{E}{-7} & 6.7\text{E}{-5} & 6.3\text{E}{-6} & 3.5\text{E}{-5} & 2.2\text{E}{-5} & 6.3\text{E}{-5} & 1.2\text{E}{-4} & 5.2\text{E}{-6} & -8.3\text{E}{-5} & 8.4\text{E}{-5} \\
0.015 & -1.8\text{E}{-3} & 1.8\text{E}{-3} & -1.4\text{E}{-4} & -1.1\text{E}{-4} & -4.6\text{E}{-4} & -0.015 & 1.9\text{E}{-4} & 6.9\text{E}{-3} & 6.3\text{E}{-5} & 1.4 & -0.039 & 2.6\text{E}{-4} & 5.5\text{E}{-3} & -9.0\text{E}{-3} \\
-6.9\text{E}{-4} & 4.3\text{E}{-4} & 8.8\text{E}{-5} & -2.0\text{E}{-5} & 2.7\text{E}{-5} & 4.9\text{E}{-5} & 1.2\text{E}{-3} & 9.7\text{E}{-6} & -5.9\text{E}{-4} & 1.2\text{E}{-4} & -0.039 & 9.8\text{E}{-3} & 9.9\text{E}{-6} & -9.0\text{E}{-4} & 1.1\text{E}{-3} \\
4.4\text{E}{-5} & 1.5\text{E}{-5} & 6.7\text{E}{-6} & -4.8\text{E}{-6} & 3.3\text{E}{-7} & -4.9\text{E}{-7} & -1.2\text{E}{-5} & 5.3\text{E}{-6} & 6.0\text{E}{-5} & 5.2\text{E}{-6} & 2.6\text{E}{-4} & 9.9\text{E}{-6} & 6.5\text{E}{-6} & 8.1\text{E}{-6} & -7.8\text{E}{-5} \\
5.5\text{E}{-4} & 7.4\text{E}{-5} & -1.1\text{E}{-3} & 2.2\text{E}{-5} & -3.6\text{E}{-6} & -1.5\text{E}{-6} & -1.2\text{E}{-3} & -2.5\text{E}{-5} & -5.9\text{E}{-4} & -8.3\text{E}{-5} & 5.5\text{E}{-3} & -9.0\text{E}{-4} & 8.1\text{E}{-6} & 2.6\text{E}{-3} & -6.6\text{E}{-4} \\
-1.0\text{E}{-3} & 9.8\text{E}{-4} & -5.4\text{E}{-4} & -3.7\text{E}{-5} & 1.7\text{E}{-5} & 1.6\text{E}{-5} & 1.1\text{E}{-3} & 4.6\text{E}{-5} & -7.9\text{E}{-5} & 8.4\text{E}{-5} & -9.0\text{E}{-3} & 1.1\text{E}{-3} & -7.8\text{E}{-5} & -6.6\text{E}{-4} & 0.01
\end{pmatrix}$$

Appendix E

Standard Algorithms in Eindhoven-Genoa notation

This appendix presents all algorithms relevant to this thesis, which are the RNEA, CRBA, ABA, RNEAmb, CRBAmb, ABAmb, GBWAmb, and the derivatives of the fixed-base inverse dynamics. These algorithms are presented both in Featherstone notation and Eindhoven-Genoa notation, so that they are accessible and recognizable to readers familiar to these algorithms in Featherstone notation. All algorithms have 'model', as an input, which contains the following model-specific constants:

- the number of bodies
- the gravitational constant
- each body's parent, $\lambda(i)$
- each joint's joint type
- each body's rigid part of the velocity transformation of the precedent joint w.r.t the current joint, ${}^{\lambda(i)|i}\mathbf{X}_i$
- each body's inertia, $\mathbb{M}_{\mathcal{B}i}$

For each algorithm we removed any parts containing external forces, as in Assumption 2.3 we assume there are none.

E.1 Recursive Newton Euler Algorithm (RNEA)

The following table presents the RNEA, as found on [43] without external forces, which is also presented in [8, Table 5.1] in a slightly different version.

Table E.1: The Recursive Newton Euler Algorithm for fixed-base systems (RNEA).

Line	**Featherstone**	**Eindhoven-Genoa**
Inputs	model, $\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}$	model, $\mathbf{s}, \mathbf{r}, \dot{\mathbf{r}}$
1	$\boldsymbol{a}_g$ = get_gravity(model)	${}^0\mathbf{a}_{grav}$ = get_gravity(model)
2	**for** $i = 1$ **to** N_B **do**	**for** $i = 1$ **to** n_B **do**
3	$[\boldsymbol{X}_\mathrm{J}, \boldsymbol{S}_i]$ = jcalc(jtype(i), $\boldsymbol{q}_i$)	$[{}^i\mathbf{X}_{\lambda(i)\|i}, \boldsymbol{\Gamma}_{\mathcal{J}i}]$ = jcalc(jtype(i), $\mathbf{s}_i$)
4	$\boldsymbol{v}_\mathrm{J} = \boldsymbol{S}_i \dot{\boldsymbol{q}}_i$	$\mathbf{v}_{\mathcal{J}i} = \boldsymbol{\Gamma}_{\mathcal{J}i}\mathbf{r}_i$
5	${}^i\boldsymbol{X}_{\lambda(i)} = \boldsymbol{X}_\mathrm{J}\boldsymbol{X}_\mathrm{T}(i)$	${}^i\mathbf{X}_{\lambda(i)} = {}^i\mathbf{X}_{\lambda(i)\|i}{}^{\lambda(i)\|i}\mathbf{X}_{\lambda(i)}$
6	**if** $\lambda(i) = 0$ **then**	**if** $\lambda(i) = 0$ **then**
7	$\boldsymbol{v}_i = \boldsymbol{v}_\mathrm{J}$	$\mathbf{v}_i = \mathbf{v}_{\mathcal{J}i}$
8	$\boldsymbol{a}_i = -{}^i\boldsymbol{X}_{\lambda(i)}\boldsymbol{a}_g + \boldsymbol{S}_i\ddot{\boldsymbol{q}}_i$	$\mathbf{a}_i = -{}^i\mathbf{X}_{\lambda(i)}{}^A\mathbf{a}_{grav} + \boldsymbol{\Gamma}_{\mathcal{J}i}\dot{\mathbf{r}}_i$
9	**else**	**else**
10	$\boldsymbol{v}_i = {}^i\boldsymbol{X}_{\lambda(i)}\boldsymbol{v}_{\lambda(i)} + \boldsymbol{v}_\mathrm{J}$	$\mathbf{v}_i = {}^i\mathbf{X}_{\lambda(i)}\mathbf{v}_{\lambda(i)} + \mathbf{v}_{\mathcal{J}i}$
11	$\boldsymbol{a}_i = {}^i\boldsymbol{X}_{\lambda(i)}\boldsymbol{a}_{\lambda(i)} + \boldsymbol{S}_i\ddot{\boldsymbol{q}}_i + \boldsymbol{v}_i \times \boldsymbol{v}_\mathrm{J}$	$\mathbf{a}_i = {}^i\mathbf{X}_{\lambda(i)}\mathbf{a}_{\lambda(i)} + \boldsymbol{\Gamma}_{\mathcal{J}i}\dot{\mathbf{r}}_i + \mathbf{v}_i \times \mathbf{v}_{\mathcal{J}i}$
12	**end**	**end**
13	$\boldsymbol{f}_i = \boldsymbol{I}_i\boldsymbol{a}_i + \boldsymbol{v}_i \times^* \boldsymbol{I}_i\boldsymbol{v}_i$	$\mathbf{f}_{\mathcal{J}i} = \mathbb{M}_{\mathcal{B}i}\mathbf{a}_i + \mathbf{v}_i \bar{\times}^* \mathbb{M}_{\mathcal{B}i}\mathbf{v}_i$
14	**end**	**end**
15	**for** $i = N_B$ **to** 1 **do**	**for** $i = n_B$ **to** 1 **do**
16	$\boldsymbol{\tau}_i = \boldsymbol{S}_i^\mathrm{T}\boldsymbol{f}_i$	$\boldsymbol{\tau}_i = \boldsymbol{\Gamma}_{\mathcal{J}i}^T\mathbf{f}_{\mathcal{J}i}$
17	**if** $\lambda(i) \neq 0$ **then**	**if** $\lambda(i) \neq 0$ **then**
18	$\boldsymbol{f}_{\lambda(i)} = \boldsymbol{f}_{\lambda(i)} + {}^{\lambda(i)}\boldsymbol{X}_i^*\boldsymbol{f}_i$	$\mathbf{f}_{\mathcal{J}\lambda(i)} = \mathbf{f}_{\mathcal{J}\lambda(i)} + {}_{\lambda(i)}\mathbf{X}^i\,\mathbf{f}_{\mathcal{J}i}$
19	**end**	**end**
20	**end**	**end**
Outputs	$\boldsymbol{\tau}$	$\boldsymbol{\tau}$

E.2 Recursive Newton Euler Algorithm for moving-base systems (RNEAmb)

The following table presents the RNEAmb, as found on [44] without external forces, which is also presented in [8, Table 9.6] in a slightly different version.

Table E.2: The Recursive Newton Euler Algorithm for moving-base systems (RNEAmb).

Line	**Featherstone**	**Eindhoven-Genoa**
Inputs	model, $\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}, {}^0\boldsymbol{X}_A, {}^A\boldsymbol{v}_0$	model, $\mathbf{s}, \mathbf{r}, \dot{\mathbf{r}}, {}^A\mathbf{H}_0, {}^A\mathbf{v}_{A,0}$
1	${}^A\boldsymbol{a}_g$ = get_gravity(model)	${}^A\mathbf{a}_{grav}$ = get_gravity(model)
2	-	${}^0\mathbf{H}_A = {}^A\mathbf{H}_0^{-1}$
3	-	${}^0\mathbf{R}_A = {}^0\mathbf{H}_A[1{:}3, 1{:}3]$
4	-	${}^0\mathbf{o}_A = {}^0\mathbf{H}_A[1{:}3, 4]$
5	-	${}^0\mathbf{X}_A = \begin{bmatrix} {}^0\mathbf{R}_A & {}^0\mathbf{o}_A^{\wedge}\,{}^0\mathbf{R}_A \\ 0_{3\times3} & {}^0\mathbf{R}_A \end{bmatrix}$
6	$\boldsymbol{v}_0 = {}^0\boldsymbol{X}_A{}^A\boldsymbol{v}_0$	$\mathbf{v}_0 = {}^0\mathbf{X}_A{}^A\mathbf{v}_{A,0}$
7	$\boldsymbol{a}^r_0 = {}^0\boldsymbol{X}_A{}^A\boldsymbol{a}_g$	$\mathbf{a}^r_0 = {}^0\mathbf{X}_A{}^A\mathbf{a}_{grav}$
8	$\boldsymbol{I}^c_0 = \boldsymbol{I}_0$	$\mathbb{M}^c_{\mathcal{B}0} = \mathbb{M}_{\mathcal{B}0}$
9	-	$\mathbf{m}_{\mathcal{B}0} = \mathbb{M}_{\mathcal{B}0}\mathbf{v}_0$
10	$\boldsymbol{p}^c_0 = \boldsymbol{I}_0\boldsymbol{a}^r_0 + \boldsymbol{v}_0 \times^* \boldsymbol{I}_0\boldsymbol{v}_0$	$\mathbf{b}^c_{\mathcal{B}0} = \mathbb{M}_{\mathcal{B}0}\mathbf{a}^r_0 + \mathbf{v}_0\bar{\times}^*\mathbf{m}_{\mathcal{B}0}$
11	**for** $i = 1$ **to** N_B **do**	**for** $i = 1$ **to** n_B **do**
12	$[\boldsymbol{X}_{\mathrm{J}}, \boldsymbol{S}_i]$ = jcalc(jtype(i), $\boldsymbol{q}_i$)	$[{}^i\mathbf{X}_{\lambda(i)\|i}, \mathbf{\Gamma}_{\mathcal{J}i}]$ = jcalc(jtype(i), $\mathbf{s}_i$)
13	$\boldsymbol{v}_{\mathrm{J}} = \boldsymbol{S}_i\dot{\boldsymbol{q}}_i$	$\mathbf{v}_{\mathcal{J}i} = \mathbf{\Gamma}_{\mathcal{J}i}\mathbf{r}_i$
14	${}^i\boldsymbol{X}_{\lambda(i)} = \boldsymbol{X}_{\mathrm{J}}\boldsymbol{X}_{\mathrm{T}}(i)$	${}^i\mathbf{X}_{\lambda(i)} = {}^i\mathbf{X}_{\lambda(i)\|i}{}^{\lambda(i)\|i}\mathbf{X}_{\lambda(i)}$
15	$\boldsymbol{v}_i = {}^i\mathbf{X}_{\lambda(i)}\boldsymbol{v}_{\lambda(i)} + \boldsymbol{v}_{\mathrm{J}}$	$\mathbf{v}_i = {}^i\mathbf{X}_{\lambda(i)}\mathbf{v}_{\lambda(i)} + \mathbf{v}_{\mathcal{J}i}$
16	$\boldsymbol{a}^r_i = {}^i\boldsymbol{X}_{\lambda(i)}\boldsymbol{a}^r_{\lambda(i)} + \boldsymbol{S}_i\ddot{\boldsymbol{q}}_i + \boldsymbol{v}_i \times \boldsymbol{v}_{\mathrm{J}}$	$\mathbf{a}^r_i = {}^i\mathbf{X}_{\lambda(i)}\mathbf{a}^r_{\lambda(i)} + \mathbf{\Gamma}_{\mathcal{J}i}\dot{\mathbf{r}}_i + \mathbf{v}_i \times \mathbf{v}_{\mathcal{J}i}$
17	$\boldsymbol{I}^c_i = \boldsymbol{I}_i$	$\mathbb{M}^c_{\mathcal{B}i} = \mathbb{M}_{\mathcal{B}i}$
18	-	$\mathbf{m}_{\mathcal{B}i} = \mathbb{M}_{\mathcal{B}i}\mathbf{v}_i$
19	$\boldsymbol{p}^c_i = \boldsymbol{I}_i\boldsymbol{a}^r_i + \boldsymbol{v}_i \times^* \boldsymbol{I}_i\boldsymbol{v}_i$	$\mathbf{b}^c_{\mathcal{B}i} = \mathbb{M}_{\mathcal{B}i}\mathbf{a}^r_i + \mathbf{v}_i\bar{\times}^*\mathbf{m}_{\mathcal{B}i}$
20	**end**	**end**
21	**for** $i = N_B$ **to** 1 **do**	**for** $i = n_B$ **to** 1 **do**
22	$\boldsymbol{I}^c_{\lambda(i)} = \boldsymbol{I}^c_{\lambda(i)} + {}^{\lambda(i)}\boldsymbol{X}^*_i\boldsymbol{I}^c_i\,{}^i\boldsymbol{X}_{\lambda(i)}$	$\mathbb{M}^c_{\mathcal{B}\lambda(i)} = \mathbb{M}^c_{\mathcal{B}\lambda(i)} + {}^{\lambda(i)}\mathbf{X}^*_i\,\mathbb{M}^c_{\mathcal{B}i}{}^i\mathbf{X}_{\lambda(i)}$
23	$\boldsymbol{p}^c_{\lambda(i)} = \boldsymbol{p}^c_{\lambda(i)} + {}_{\lambda(i)}\boldsymbol{X}^i\,\boldsymbol{p}^c_i$	$\mathbf{b}^c_{\mathcal{B}\lambda(i)} = \mathbf{b}^c_{\mathcal{B}\lambda(i)} + {}_{\lambda(i)}\mathbf{X}^i\,\mathbf{b}^c_{\mathcal{B}i}$
24	**end**	**end**
25	${}^0\boldsymbol{a}_0 = -(\boldsymbol{I}^c_0)^{-1}\boldsymbol{p}^c_0$	$\mathbf{a}_0 = -(\mathbb{M}^c_{\mathcal{B}0})^{-1}\mathbf{b}^c_{\mathcal{B}0}$
26	**for** $i = 1$ **to** N_B **do**	**for** $i = 1$ **to** n_B **do**
27	${}^i\boldsymbol{a}_0 = {}^i\boldsymbol{X}_{\lambda(i)}{}^{\lambda(i)}\boldsymbol{a}_0$	${}^i\mathbf{a}_{A,0} = {}^i\mathbf{X}_{\lambda(i)}{}^{\lambda(i)}\mathbf{a}_{A,0}$
28	$\boldsymbol{\tau}_i = \boldsymbol{S}_i^{\mathrm{T}}(\boldsymbol{I}^c_i{}^i\boldsymbol{a}_0 + \boldsymbol{p}^c_i)$	$\boldsymbol{\tau}_i = \mathbf{\Gamma}^T_{\mathcal{J}i}(\mathbb{M}^c_{\mathcal{B}i}{}^i\mathbf{a}_{A,0} + \mathbf{b}^c_{\mathcal{B}i})$
29	**end**	**end**
30	${}^A\boldsymbol{a}_0 = {}^0\boldsymbol{X}_A{}^0\boldsymbol{a}_0$	${}^A\mathbf{a}_{A,0} = {}^0\mathbf{X}_A\mathbf{a}_0$
Outputs	$\boldsymbol{\tau}, {}^A\boldsymbol{a}_0$	$\boldsymbol{\tau}, {}^A\mathbf{a}_{A,0}$

E.3 Composite Rigid Body Algorithm (CRBA)

The following table presents a slightly modified version of the CRBA, as found in [8, Table 6.2].

Table E.3: The Composite Rigid Body Algorithm for fixed-base systems (CRBA).

Line	**Featherstone**	**Eindhoven-Genoa**
Inputs	model, $\boldsymbol{q}$	model, $\mathbf{s}$
1	$\mathbf{H} = 0$	$\mathbf{M}_{fb} = 0$
2	**for** $i = N_B$ **to** 1 **do**	**for** $i = n_B$ **to** 1 **do**
3	$[\boldsymbol{X}_\mathrm{J}, \boldsymbol{S}_i] = \mathrm{jcalc}(\mathrm{jtype}(i), \boldsymbol{q}_i)$	$[{}^i\mathbf{X}_{\lambda(i)\vert i}, \boldsymbol{\Gamma}_{\mathcal{J}i}] = \mathrm{jcalc}(\mathrm{jtype}(i), \mathbf{s}_i)$
4	${}^i\boldsymbol{X}_{\lambda(i)} = \boldsymbol{X}_\mathrm{J}\boldsymbol{X}_\mathrm{T}(i)$	${}^i\mathbf{X}_{\lambda(i)} = {}^i\mathbf{X}_{\lambda(i)\vert i}\,{}^{\lambda(i)\vert i}\mathbf{X}_{\lambda(i)}$
5	$\boldsymbol{I}_i^c = \boldsymbol{I}_i$	$\mathbb{M}^c_{\mathcal{B}i} = \mathbb{M}_{\mathcal{B}i}$
6	**if** $\lambda(i) \neq 0$ **then**	**if** $\lambda(i) \neq 0$ **then**
7	$\boldsymbol{I}^c_{\lambda(i)} = \boldsymbol{I}^c_{\lambda(i)} + {}^{\lambda(i)}\boldsymbol{X}_i^* \boldsymbol{I}_i^c\,{}^i\boldsymbol{X}_{\lambda(i)}$	$\mathbb{M}^c_{\mathcal{B}\lambda(i)} = \mathbb{M}^c_{\mathcal{B}\lambda(i)} + {}_{\lambda(i)}\mathbf{X}^i\,\mathbb{M}^c_{\mathcal{B}i}\,{}^i\mathbf{X}_{\lambda(i)}$
8	**end**	**end**
9	**end**	**end**
10	**for** $i = 1$ **to** N_B **do**	**for** $i = 1$ **to** n_B **do**
11	$\boldsymbol{F} = \boldsymbol{I}_i^c \boldsymbol{S}_i$	${}_i\mathbf{F}_{\mathcal{B}i} = \mathbb{M}^c_{\mathcal{B}i}\boldsymbol{\Gamma}_{\mathcal{J}i}$
12	$\boldsymbol{H}_{ii} = \mathbf{S}_i^\mathrm{T}\boldsymbol{F}$	$\mathbf{M}^{ii}_{fb} = \boldsymbol{\Gamma}_i^T\mathbf{F}_{\mathcal{B}i}$
13	$j = i$	$j = i$
14	**while** $\lambda(j) \neq 0$ **do**	**while** $\lambda(j) \neq 0$ **do**
15	$\boldsymbol{F} = {}^{\lambda(j)}\boldsymbol{X}_j^*\boldsymbol{F}$	${}_{\lambda(j)}\mathbf{F}_{\mathcal{B}i} = {}_{\lambda(j)}\mathbf{X}^j\,{}_j\mathbf{F}_{\mathcal{B}i}$
16	$j = \lambda(j)$	$j = \lambda(j)$
17	$\boldsymbol{H}_{ij} = \boldsymbol{F}^\mathrm{T}\boldsymbol{S}_j$	$\mathbf{M}^{ij}_{fb} = {}_j\mathbf{F}^T_{\mathcal{B}i}\boldsymbol{\Gamma}_{\mathcal{J}j}$
18	$\boldsymbol{H}_{ji} = \boldsymbol{H}_{ij}$	$\mathbf{M}^{ji}_{fb} = \mathbf{M}^{ij}_{fb}$
19	**end**	**end**
20	**end**	**end**
Outputs	$\boldsymbol{H}$	$\mathbf{M}$

E.4 Composite Rigid Body Algorithm for moving-base systems (CRBAmb)

The following table presents a slightly modified version of the CRBAmb, as found in [8, Table 9.5].

Table E.4: The Composite Rigid Body Algorithm for moving-base systems (CRBAmb).

Line	**Featherstone**	**Eindhoven-Genoa**
Inputs	model, $\boldsymbol{q}$	model, $\mathbf{s}$
1	$\mathbf{H} = 0$	$\mathbf{M} = 0$
2	**for** $i = N_B$ **to** 1 **do**	**for** $i = n_B$ **to** 1 **do**
3	$[\boldsymbol{X}_\mathrm{J}, \boldsymbol{S}_i] = \text{jcalc}(\text{jtype}(i), \boldsymbol{q}_i)$	$[{}^i\mathbf{X}_{\lambda(i)\vert i}, \boldsymbol{\Gamma}_{\mathcal{J}i}] = \text{jcalc}(\text{jtype}(i), \mathbf{s}_i)$
4	${}^i\boldsymbol{X}_{\lambda(i)} = \boldsymbol{X}_\mathrm{J}\boldsymbol{X}_\mathrm{T}(i)$	${}^i\mathbf{X}_{\lambda(i)} = {}^i\mathbf{X}_{\lambda(i)\vert i}{}^{\lambda(i)\vert i}\mathbf{X}_{\lambda(i)}$
5	$\boldsymbol{I}^c_i = \boldsymbol{I}_i$	$\mathbb{M}^c_{\mathcal{B}i} = \mathbb{M}_{\mathcal{B}i}$
6	$\boldsymbol{I}^c_{\lambda(i)} = \boldsymbol{I}^c_{\lambda(i)} + {}^{\lambda(i)}\boldsymbol{X}^*_i \boldsymbol{I}^c_i\, {}^i\boldsymbol{X}_{\lambda(i)}$	$\mathbb{M}^c_{\mathcal{B}\lambda(i)} = \mathbb{M}^c_{\mathcal{B}\lambda(i)} + {}_{\lambda(i)}\mathbf{X}^i\, \mathbb{M}^c_{\mathcal{B}i}\, {}^i\mathbf{X}_{\lambda(i)}$
7	**end**	**end**
8	**for** $i = 1$ **to** N_B **do**	**for** $i = 1$ **to** n_B **do**
9	$\boldsymbol{F} = \boldsymbol{I}^c_i\boldsymbol{S}_i$	${}_i\mathbf{F}_{\mathcal{B}i} = \mathbb{M}^c_{\mathcal{B}i}\boldsymbol{\Gamma}_{\mathcal{J}i}$
10	$\boldsymbol{H}_{ii} = \mathbf{S}^\mathrm{T}_i\boldsymbol{F}$	$\mathbf{M}_{ii} = \boldsymbol{\Gamma}^T_i\mathbf{F}_{\mathcal{B}i}$
11	$j = i$	$j = i$
12	**while** $\lambda(j) \neq 0$ **do**	**while** $\lambda(j) \neq 0$ **do**
13	$\boldsymbol{F} = {}^{\lambda(j)}\boldsymbol{F}^*_j\boldsymbol{F}$	${}_{\lambda(j)}\mathbf{F}_{\mathcal{B}i} = {}_{\lambda(j)}\mathbf{X}^j\, {}_j\mathbf{F}_{\mathcal{B}i}$
14	$j = \lambda(j)$	$j = \lambda(j)$
15	$\boldsymbol{H}_{ij} = \boldsymbol{F}^\mathrm{T}\boldsymbol{S}_j$	$\mathbf{M}_{ij} = {}_j\mathbf{F}^T_{\mathcal{B}i}\boldsymbol{\Gamma}_{\mathcal{J}j}$
16	$\boldsymbol{H}_{ji} = \boldsymbol{H}_{ij}$	$\mathbf{M}_{ji} = \mathbf{M}_{ij}$
17	**end**	**end**
18	$\boldsymbol{F}_i = {}^0\mathbf{X}^*_j\boldsymbol{F}_i$	${}_0\mathbf{F}_{\mathcal{B}i} = {}_0\mathbf{X}^j\, {}_j\mathbf{F}_{\mathcal{B}i}$
19	**end**	**end**
20	$\boldsymbol{F} = [\boldsymbol{F}_1 \ldots \boldsymbol{F}_{N_B}]$	${}_0\mathbf{F} = [{}_0\mathbf{F}_1 \ldots {}_0\mathbf{F}_{\mathcal{B}n_B}]$
21	$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{I}^c_0 & \boldsymbol{F} \\ \boldsymbol{F}^T & \boldsymbol{H} \end{bmatrix}$	$\mathbf{M} = \begin{bmatrix} \mathbb{M}^c_{\mathcal{B}0} & {}_0\mathbf{F} \\ {}_0\mathbf{F}^T & \mathbf{M} \end{bmatrix}$
Outputs	$\boldsymbol{H}$	$\mathbf{M}$

E.5 Articulated Body Algorithm (ABA)

The following table presents the ABA, as found on [45] without external forces, which is also presented in [8, Table 7.1] in a slightly different version.

Table E.5: The Articulated Body Algorithm for fixed-base systems (ABA).

Line	**Featherstone**	**Eindhoven-Genoa**
Inputs	model, $\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{\tau}$	model, $\mathbf{s}, \mathbf{r}, \boldsymbol{\tau}$
1	$\boldsymbol{a}_g =$ get_gravity(model)	${}^0\mathbf{a}_{grav} =$ get_gravity(model)
2	**for** $i = 1$ **to** N_B **do**	**for** $i = 1$ **to** n_B **do**
3	$[\boldsymbol{X}_\mathrm{J}, \boldsymbol{S}_i] = \mathrm{jcalc}(\mathrm{jtype}(i), \boldsymbol{q}_i)$	$[{}^i\mathbf{X}_{\lambda(i)\vert i}, \boldsymbol{\Gamma}_{\mathcal{J}i}] = \mathrm{jcalc}(\mathrm{jtype}(i), \mathbf{s}_i)$
4	$\boldsymbol{v}_\mathrm{J} = \boldsymbol{S}_i \dot{\boldsymbol{q}}_i$	$\mathbf{v}_{\mathcal{J}i} = \boldsymbol{\Gamma}_{\mathcal{J}i}\mathbf{r}_i$
5	${}^i\boldsymbol{X}_0 = \boldsymbol{X}_\mathrm{J}\boldsymbol{X}_\mathrm{T}(i)$	${}^i\mathbf{X}_{\lambda(i)} = {}^i\mathbf{X}_{\lambda(i)\vert i}{}^{\lambda(i)\vert i}\mathbf{X}_{\lambda(i)}$
6	$\boldsymbol{v}_i = {}^i\boldsymbol{X}_{\lambda(i)}\boldsymbol{v}_{\lambda(i)} + \boldsymbol{v}_\mathrm{J}$	$\mathbf{v}_i = {}^i\mathbf{X}_{\lambda(i)}\mathbf{v}_{\lambda(i)} + \mathbf{v}_{\mathcal{J}i}$
7	$\boldsymbol{c}_i = \boldsymbol{v}_i \times \boldsymbol{v}_\mathrm{J}$	-
8	$\boldsymbol{I}_i^A = \boldsymbol{I}_i$	$\mathbb{M}_{\mathcal{B}i}^A = \mathbb{M}_{\mathcal{B}i}$
9	-	$\mathbf{m}_{\mathcal{B}i} = \mathbb{M}_{\mathcal{B}i}\mathbf{v}_i$
10	$\boldsymbol{p}_i^A = \boldsymbol{v}_i \times^* \boldsymbol{I}_i\boldsymbol{v}_i$	$\mathbf{b}_{\mathcal{B}i}^A = \mathbf{v}_i \bar{\times}^* \mathbf{m}_{\mathcal{B}i}$
11	**end**	**end**
12	**for** $i = N_B$ **to** 1 **do**	**for** $i = n_B$ **to** 1 **do**
13	$\boldsymbol{U}_i = \boldsymbol{I}_i^A\boldsymbol{S}_i$	$\mathbf{U}_{\mathcal{B}i} = \mathbb{M}_{\mathcal{B}i}^A\boldsymbol{\Gamma}_{\mathcal{J}i}$
14	$\boldsymbol{D}_i = \boldsymbol{S}_i^\mathrm{T}\boldsymbol{U}_i$	$\mathbf{D}_{\mathcal{B}i} = \boldsymbol{\Gamma}_{\mathcal{J}i}^T\mathbf{U}_{\mathcal{B}i}$
15	$\boldsymbol{u}_i = \boldsymbol{\tau}_i - \boldsymbol{S}_i^\mathrm{T}\boldsymbol{p}_i^A$	$\mathbf{u}_{\mathcal{B}i} = \boldsymbol{\tau}_i - \boldsymbol{\Gamma}_{\mathcal{J}i}^T\mathbf{b}_{\mathcal{B}i}^A$
16	**if** $\lambda(i) \neq 0$ **then**	**if** $\lambda(i) \neq 0$ **then**
17	$\boldsymbol{I}^a = \boldsymbol{I}_i^A - \boldsymbol{U}_i\boldsymbol{D}_i^{-1}\boldsymbol{U}_i^\mathrm{T}$	$\mathbb{M}_{\mathcal{B}i}^a = \mathbb{M}_{\mathcal{B}i}^A - \mathbf{U}_{\mathcal{B}i}\mathbf{D}_{\mathcal{B}i}^{-1}\mathbf{U}_{\mathcal{B}i}^T$
18	$\boldsymbol{p}^a = \boldsymbol{p}_i^A + \boldsymbol{I}^a\boldsymbol{c}_i + \boldsymbol{U}_i\boldsymbol{D}_i^{-1}\boldsymbol{u}_i$	$\mathbf{b}_{\mathcal{B}i}^a = \mathbf{b}_{\mathcal{B}i}^A + \mathbb{M}_{\mathcal{B}i}^a(\mathbf{v}_i \times \mathbf{v}_{\mathcal{J}i}) + \mathbf{U}_{\mathcal{B}i}\mathbf{D}_{\mathcal{B}i}^{-1}\mathbf{u}_{\mathcal{B}i}$
19	$\boldsymbol{I}_{\lambda(i)}^A = \boldsymbol{I}_{\lambda(i)}^A + {}^{\lambda(i)}\boldsymbol{X}_i^*\boldsymbol{I}^a\,{}^i\boldsymbol{X}_{\lambda(i)}$	$\mathbb{M}_{\mathcal{B}\lambda(i)}^A = \mathbb{M}_{\mathcal{B}\lambda(i)}^A + {}_{\lambda(i)}\mathbf{X}^i\,\mathbb{M}_{\mathcal{B}i}^a\,{}^i\mathbf{X}_{\lambda(i)}$
20	$\boldsymbol{p}_{\lambda(i)}^A = \boldsymbol{p}_{\lambda(i)}^A + {}^{\lambda(i)}\boldsymbol{X}_i^*\boldsymbol{p}^a$	$\mathbf{b}_{\mathcal{B}\lambda(i)}^A = \mathbf{b}_{\mathcal{B}\lambda(i)}^A + {}_{\lambda(i)}\mathbf{X}^i\mathbf{b}_{\mathcal{B}i}^a$
21	**end**	**end**
22	**end**	**end**
23	$\boldsymbol{a}_0 = \boldsymbol{a}_g$	$\mathbf{a}_0 = {}^0\mathbf{a}_{grav}$
24	**for** $i = 1$ **to** N_B **do**	**for** $i = 1$ **to** n_B **do**
25	$\boldsymbol{a}' = {}^i\boldsymbol{X}_{\lambda(i)}\boldsymbol{a}_{\lambda(i)} + \boldsymbol{c}_i$	$\mathbf{a}' = {}^i\mathbf{X}_{\lambda(i)}\mathbf{a}_{\lambda(i)} + \mathbf{v}_i \times \mathbf{v}_{\mathcal{J}i}$
26	$\ddot{\boldsymbol{q}}_i = \boldsymbol{D}_i^{-1}(\boldsymbol{u}_i - \boldsymbol{U}_i^\mathrm{T}\boldsymbol{a}_i')$	$\dot{\mathbf{r}}_i = \mathbf{D}_{\mathcal{B}i}^{-1}(\mathbf{u}_{\mathcal{B}i} - \mathbf{U}_{\mathcal{B}i}^T\mathbf{a}_i')$
27	$\boldsymbol{a}_i = \boldsymbol{a}_i' + \boldsymbol{S}_i\ddot{\boldsymbol{q}}_i$	$\mathbf{a}_i = \mathbf{a}_i' + \boldsymbol{\Gamma}_{\mathcal{J}i}\dot{\mathbf{r}}_i$
28	**end**	**end**
Outputs	$\ddot{\boldsymbol{q}}$	$\dot{\mathbf{r}}$

E.6 Articulated Body Algorithm for moving-base systems (ABAmb)

The following table presents the ABAmb, as found on [46] without external forces, which is also presented in [8, Table 9.4] in a slightly different version.

Table E.6: The Articulated Body Algorithm for moving-base systems (ABAmb).

Line	**Featherstone**	**Eindhoven-Genoa**
Inputs	model, $\mathbf{q}, \dot{\mathbf{q}}, \boldsymbol{\tau}, {}^0\boldsymbol{X}_A, {}^A\boldsymbol{v}_0$	model, $\mathbf{s}, \mathbf{r}, \boldsymbol{\tau}, {}^A\mathbf{H}_0, {}^A\mathbf{v}_{A,0}$
1	${}^A\boldsymbol{a}_g$ = get_gravity(model)	${}^A\mathbf{a}_{grav}$ = get_gravity(model)
2	-	${}^0\mathbf{H}_A = {}^A\mathbf{H}_0^{-1}$
3	-	${}^0\mathbf{R}_A = {}^0\mathbf{H}_A[1{:}3, 1{:}3]$
4	-	${}^0\mathbf{o}_A = {}^0\mathbf{H}_A[1{:}3, 4]$
5	-	${}^0\mathbf{X}_A = \begin{bmatrix} {}^0\mathbf{R}_A & {}^0\mathbf{o}_A^{\wedge}\,{}^0\mathbf{R}_A \\ 0_{3\times 3} & {}^0\mathbf{R}_A \end{bmatrix}$
6	$\boldsymbol{v}_0 = {}^0\boldsymbol{X}_A{}^A\boldsymbol{v}_0$	$\mathbf{v}_0 = {}^0\mathbf{X}_A{}^A\mathbf{v}_{A,0}$
7	$\boldsymbol{I}_0^A = \boldsymbol{I}_0$	$\mathbb{M}_{\mathcal{B}0}^A = \mathbb{M}_{\mathcal{B}0}$
8	-	$\mathbf{m}_{\mathcal{B}0} = \mathbb{M}_{\mathcal{B}0}\mathbf{v}_0$
9	$\boldsymbol{p}_0^A = \boldsymbol{v}_0 \times^* \boldsymbol{I}_0\boldsymbol{v}_0$	$\mathbf{b}_{\mathcal{B}0}^A = \mathbf{v}_0 \bar{\times}^* \mathbf{m}_{\mathcal{B}0}$
10	**for** $i = 1$ **to** N_B **do**	**for** $i = 1$ **to** n_B **do**
11	$[\boldsymbol{X}_\mathrm{J}, \boldsymbol{S}_i]$ = jcalc(jtype(i), $\boldsymbol{q}_i$)	$[{}^i\mathbf{X}_{\lambda(i)\vert i}, \boldsymbol{\Gamma}_{\mathcal{J}i}]$ = jcalc(jtype(i), $\mathbf{s}_i$)
12	$\boldsymbol{v}_\mathrm{J} = \boldsymbol{S}_i\dot{\boldsymbol{q}}_i$	$\mathbf{v}_{\mathcal{J}i} = \boldsymbol{\Gamma}_{\mathcal{J}i}\mathbf{r}_i$
13	${}^i\boldsymbol{X}_{\lambda(i)} = \boldsymbol{X}_\mathrm{J}\boldsymbol{X}_\mathrm{T}(i)$	${}^i\mathbf{X}_{\lambda(i)} = {}^i\mathbf{X}_{\lambda(i)\vert i}{}^{\lambda(i)\vert i}\mathbf{X}_{\lambda(i)}$
14	$\boldsymbol{v}_i = {}^i\mathbf{X}_{\lambda(i)}\boldsymbol{v}_{\lambda(i)} + \boldsymbol{v}_\mathrm{J}$	$\mathbf{v}_i = {}^i\mathbf{X}_{\lambda(i)}\mathbf{v}_{\lambda(i)} + \mathbf{v}_{\mathcal{J}i}$
15	$\boldsymbol{c}_i = \boldsymbol{v}_i \times \boldsymbol{v}_\mathrm{J}$	-
16	$\boldsymbol{I}_i^A = \boldsymbol{I}_i$	$\mathbb{M}_{\mathcal{B}i}^A = \mathbb{M}_{\mathcal{B}i}$
17	-	$\mathbf{m}_{\mathcal{B}i} = \mathbb{M}_{\mathcal{B}i}\mathbf{v}_i$
18	$\boldsymbol{p}_i^A = \boldsymbol{v}_i \times^* \boldsymbol{I}_i\boldsymbol{v}_i$	$\mathbf{b}_{\mathcal{B}i}^A = \mathbf{v}_i \bar{\times}^* \mathbf{m}_{\mathcal{B}i}$
19	**end**	**end**
20	**for** $i = N_B$ **to** 1 **do**	**for** $i = n_B$ **to** 1 **do**
21	$\boldsymbol{U}_i = \boldsymbol{I}_i^A\boldsymbol{S}_i$	$\mathbf{U}_{\mathcal{B}i} = \mathbb{M}_{\mathcal{B}i}^A\boldsymbol{\Gamma}_{\mathcal{J}i}$
22	$\boldsymbol{D}_i = \boldsymbol{S}_i^\mathrm{T}\boldsymbol{U}_i$	$\mathbf{D}_{\mathcal{B}i} = \boldsymbol{\Gamma}_{\mathcal{J}i}^T\mathbf{U}_{\mathcal{B}i}$
23	$\boldsymbol{u}_i = \boldsymbol{\tau}_i - \boldsymbol{S}_i^\mathrm{T}\boldsymbol{p}_i^A$	$\mathbf{u}_{\mathcal{B}i} = \boldsymbol{\tau}_i - \boldsymbol{\Gamma}_{\mathcal{J}i}^T\mathbf{b}_{\mathcal{B}i}^A$
24	$\boldsymbol{I}^a = \boldsymbol{I}_i^A - \boldsymbol{U}_i\boldsymbol{D}_i^{-1}\boldsymbol{U}_i^\mathrm{T}$	$\mathbb{M}_{\mathcal{B}i}^a = \mathbb{M}_{\mathcal{B}i}^A - \mathbf{U}_{\mathcal{B}i}\mathbf{D}_{\mathcal{B}i}^{-1}\mathbf{U}_{\mathcal{B}i}^T$
25	$\boldsymbol{p}^a = \boldsymbol{p}_i^A + \boldsymbol{I}^a\boldsymbol{c}_i + \boldsymbol{U}_i\boldsymbol{D}_i^{-1}\boldsymbol{u}_i$	$\mathbf{b}_{\mathcal{B}i}^a = \mathbf{b}_{\mathcal{B}i}^A + \mathbb{M}_{\mathcal{B}i}^a(\mathbf{v}_i \times \mathbf{v}_{\mathcal{J}i}) + \mathbf{U}_{\mathcal{B}i}\mathbf{D}_{\mathcal{B}i}^{-1}\mathbf{u}_{\mathcal{B}i}$
26	$\boldsymbol{I}_{\lambda(i)}^A = \boldsymbol{I}_{\lambda(i)}^A + {}^{\lambda(i)}\boldsymbol{X}_i^*\boldsymbol{I}^a\,{}^i\boldsymbol{X}_{\lambda(i)}$	$\mathbb{M}_{\mathcal{B}\lambda(i)}^A = \mathbb{M}_{\mathcal{B}\lambda(i)}^A + {}_{\lambda(i)}\mathbf{X}^i\,\mathbb{M}_{\mathcal{B}i}^a\,{}^i\mathbf{X}_{\lambda(i)}$
27	$\boldsymbol{p}_{\lambda(i)}^A = \boldsymbol{p}_{\lambda(i)}^A + {}^{\lambda(i)}\boldsymbol{X}_i^*\boldsymbol{p}^a$	$\mathbf{b}_{\mathcal{B}\lambda(i)}^A = \mathbf{b}_{\mathcal{B}\lambda(i)}^A + {}_{\lambda(i)}\mathbf{X}^i\mathbf{b}_{\mathcal{B}i}^a$
28	**end**	**end**
29	$\boldsymbol{a}_0 = -(\boldsymbol{I}_0^A)^{-1}\boldsymbol{p}_0^A$	${}^0\mathbf{a}_{A,0} = -(\mathbb{M}_{\mathcal{B}0}^A)^{-1}\mathbf{b}_{\mathcal{B}0}^A$
30	**for** $i = 1$ **to** N_B **do**	**for** $i = 1$ **to** n_B **do**
31	$\boldsymbol{a}_i' = {}^i\boldsymbol{X}_{\lambda(i)}\boldsymbol{a}_{\lambda(i)} + \boldsymbol{c}_i$	$\mathbf{a}_i' = {}^i\mathbf{X}_{\lambda(i)}\mathbf{a}_{\lambda(i)} + \mathbf{v}_i \times \mathbf{v}_{\mathcal{J}i}$
32	$\ddot{\boldsymbol{q}}_i = \boldsymbol{D}_i^{-1}(\boldsymbol{u}_i - \boldsymbol{U}_i^\mathrm{T}\boldsymbol{a}_i')$	$\dot{\mathbf{r}}_i = \mathbf{D}_{\mathcal{B}i}^{-1}(\mathbf{u}_{\mathcal{B}i} - \mathbf{U}_{\mathcal{B}i}^T\mathbf{a}_i')$
33	$\boldsymbol{a}_i = \boldsymbol{a}_i' + \boldsymbol{S}_i\ddot{\boldsymbol{q}}_i$	$\mathbf{a}_i = \mathbf{a}_i' + \boldsymbol{\Gamma}_{\mathcal{J}i}\dot{\mathbf{r}}_i$
34	**end**	**end**
35	${}^0\boldsymbol{a}_0 = {}^0\boldsymbol{a}_0 + {}^0\boldsymbol{X}_A{}^A\boldsymbol{a}_g$	${}^0\mathbf{a}_{A,0} = {}^0\mathbf{a}_{A,0} + {}^0\mathbf{X}_A{}^A\mathbf{a}_{grav}$
Outputs	$\ddot{\boldsymbol{q}}, {}^0\boldsymbol{a}_0$	$\dot{\mathbf{r}}, {}^0\mathbf{a}_{A,0}$

E.7 Generalized Bias Wrench Algorithm for moving-base systems (GBWAmb)

The following table presents a slightly modified version of the GBWAmb without external forces, as found in [8, Table 9.5].

Table E.7: The Generalized Bias Wrench Algorithm for moving-base systems (GBWAmb).

Line	**Featherstone**	**Eindhoven-Genoa**
Inputs	model, $\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}, {}^0\boldsymbol{X}_A, {}^A\boldsymbol{v}_0, {}^0\boldsymbol{a}_0$	model, $\mathbf{s}, \mathbf{r}, \dot{\mathbf{r}}, {}^A\mathbf{H}_0, {}^A\mathbf{v}_{A,0}, {}^0\mathbf{a}_{A,0}$
1	${}^A\boldsymbol{a}_g$ = get_gravity(model)	${}^A\mathbf{a}_{grav}$ = get_gravity(model)
2	-	${}^0\mathbf{H}_A = {}^A\mathbf{H}_0^{-1}$
3	-	${}^0\mathbf{R}_A = {}^0\mathbf{H}_A[1{:}3, 1{:}3]$
4	-	${}^0\mathbf{o}_A = {}^0\mathbf{H}_A[1{:}3, 4]$
5	-	${}^0\mathbf{X}_A = \begin{bmatrix} {}^0\mathbf{R}_A & {}^0\mathbf{o}_A^\wedge {}^0\mathbf{R}_A \\ 0_{3\times3} & {}^0\mathbf{R}_A \end{bmatrix}$
6	$\boldsymbol{v}_0 = {}^0\boldsymbol{X}_A {}^A\boldsymbol{v}_0$	$\mathbf{v}_0 = {}^0\mathbf{X}_A {}^A\mathbf{v}_{A,0}$
7	-	$\mathbf{m}_{\mathcal{B}0} = \mathbb{M}_{\mathcal{B}0}\mathbf{v}_0$
8	$\boldsymbol{a}_0^{vp} = {}^0\boldsymbol{X}_A {}^A\boldsymbol{a}_g$	$\mathbf{a}_0^{vp} = {}^0\mathbf{X}_A {}^A\mathbf{a}_{grav}$
9	$\boldsymbol{f}_0 = \boldsymbol{I}_0\boldsymbol{a}_0^{vp} + \boldsymbol{v}_0 \times^* \boldsymbol{I}_0\boldsymbol{v}_0$	$\mathbf{b}_{\mathcal{B}0}^{vp} = \mathbb{M}_{\mathcal{B}0}\mathbf{a}_0^{vp} + \mathbf{v}_0 \bar{\times}^* \mathbf{m}_{\mathcal{B}0}$
10	**for** $i = 1$ **to** N_B **do**	**for** $i = 1$ **to** n_B **do**
11	$[\boldsymbol{X}_\mathrm{J}, \boldsymbol{S}_i]$ = jcalc(jtype(i), $\boldsymbol{q}_i$)	$[{}^i\mathbf{X}_{\lambda(i)\|i}, \mathbf{\Gamma}_{\mathcal{J}i}]$ = jcalc(jtype(i), $\mathbf{s}_i$)
12	$\boldsymbol{v}_\mathrm{J} = \boldsymbol{S}_i\dot{\boldsymbol{q}}_i$	$\mathbf{v}_{\mathcal{J}i} = \mathbf{\Gamma}_{\mathcal{J}i}\mathbf{r}_i$
13	${}^i\boldsymbol{X}_{\lambda(i)} = \boldsymbol{X}_\mathrm{J}\boldsymbol{X}_\mathrm{T}(i)$	${}^i\mathbf{X}_{\lambda(i)} = {}^i\mathbf{X}_{\lambda(i)\|i} {}^{\lambda(i)\|i}\mathbf{X}_{\lambda(i)}$
14	$\boldsymbol{v}_i = {}^i\mathbf{X}_{\lambda(i)}\boldsymbol{v}_{\lambda(i)} + \boldsymbol{v}_\mathrm{J}$	$\mathbf{v}_i = {}^i\mathbf{X}_{\lambda(i)}\mathbf{v}_{\lambda(i)} + \mathbf{v}_{\mathcal{J}i}$
15	-	$\mathbf{m}_{\mathcal{B}i} = \mathbb{M}_{\mathcal{B}i}\mathbf{v}_i$
16	$\boldsymbol{a}_i^{vp} = {}^i\boldsymbol{X}_{\lambda(i)}\boldsymbol{a}_{\lambda(i)}^{vp} + \boldsymbol{v}_i \times \boldsymbol{v}_\mathrm{J}$	$\mathbf{a}_i^{vp} = {}^i\mathbf{X}_{\lambda(i)}\mathbf{a}_{\lambda(i)}^{vp} + \mathbf{v}_i \times \mathbf{v}_{\mathcal{J}i}$
17	$\boldsymbol{f}_i = \boldsymbol{I}_i\boldsymbol{a}_i^{vp} + \boldsymbol{v}_i \times^* \boldsymbol{I}_i\boldsymbol{v}_i$	$\mathbf{b}_{\mathcal{B}i}^{vp} = \mathbb{M}_{\mathcal{B}i}\mathbf{a}_i^{vp} + \mathbf{v}_i \bar{\times}^* \mathbf{m}_{\mathcal{B}i}$
18	**end**	**end**
19	**for** $i = N_B$ **to** 1 **do**	**for** $i = n_B$ **to** 1 **do**
20	$\boldsymbol{C}_i = \boldsymbol{S}_i^\mathrm{T}\boldsymbol{f}_i$	$\mathbf{h}_{\mathcal{B}i} = \mathbf{\Gamma}_{\mathcal{J}i}^T\mathbf{b}_{\mathcal{B}i}^{vp}$
21	$\boldsymbol{f}_{\lambda(i)} = \boldsymbol{f}_{\lambda(i)} + {}^{\lambda(i)}\boldsymbol{X}_i^*\boldsymbol{f}_i$	$\mathbf{b}_{\mathcal{B}\lambda(i)}^{vp} = \mathbf{b}_{\mathcal{B}\lambda(i)}^{vp} + {}_{\lambda(i)}\mathbf{X}^i\, \mathbf{b}_{\mathcal{B}i}^{vp}$
22	**end**	**end**
23	-	$\mathbf{h} = [\mathbf{b}_{\mathcal{B}0}^{vp}; \mathbf{h}_{\mathcal{B}1} ... \mathbf{h}_{\mathcal{B}n_B}]$
Outputs	$\boldsymbol{f}_0, \boldsymbol{C}$	$\mathbf{h}$

E.8 Derivatives of fixed-base inverse dynamics

The following two subsections present slightly modified versions of the algorithms presented in [7], which compute the derivatives of fixed-base inverse dynamics. The inverse dynamics of fixed-base systems are found by rewriting (2.55) to

$$ID_{fb}(\mathbf{s}, \mathbf{r}, \dot{\mathbf{r}}) := \boldsymbol{\tau} = \mathbf{M}_{fb}\dot{\mathbf{r}} + \mathbf{h}_{fb}. \tag{E.1}$$

The derivatives of the fixed-base inverse dynamics with respect to $\mathbf{s}$ and $\mathbf{r}$ are expressed as $\mathrm{D}_1\, ID_{fb}(\mathbf{s}, \mathbf{r}, \dot{\mathbf{r}})$ and $\mathrm{D}_2\, ID_{fb}(\mathbf{s}, \mathbf{r}, \dot{\mathbf{r}})$ respectively and are computed by the algorithms in the following two subsections.

We aim to remove unclarities from the algorithms by rewriting them more specifically. The author of [7] writes one general algorithm for both derivatives, where many explanations and exceptions are provided. Instead, we write two algorithms specifically for the derivatives with respect to $\mathbf{s}$ and $\mathbf{r}$, so no specific cases require further clarification.

E.8.1 Derivatives of fixed-base inverse dynamics w.r.t. generalized position vector

The algorithm presented below makes use of the following equations [7]:

$$\frac{\partial\, {}^i\mathbf{X}_{\lambda(i)}}{\partial \mathbf{s}_i}\mathbf{v}_i = ({}^i\mathbf{X}_{\lambda(i)}\mathbf{v}_i) \times \mathbf{\Gamma}_{\mathcal{J}i}, \tag{E.2}$$

$$\frac{\partial\, {}^i\mathbf{X}_{\lambda(i)}}{\partial \mathbf{s}_i}\mathbf{a}_i = ({}^i\mathbf{X}_{\lambda(i)}\mathbf{a}_i) \times \mathbf{\Gamma}_{\mathcal{J}i} \tag{E.3}$$

and

$$\frac{\partial\, {}_{\lambda(i)}\mathbf{X}^i}{\partial \mathbf{s}_i}\mathbf{f}_{\mathcal{J}i} = {}_{\lambda(i)}\mathbf{X}^i(\mathbf{\Gamma}_{\mathcal{J}i}\bar{\times}^*\mathbf{f}_{\mathcal{J}i}). \tag{E.4}$$

Table E.8: Derivatives of fixed-base inverse dynamics w.r.t. generalized position vector $\mathbf{s}$.

Inputs	*All outputs and intermediate variables of RNEA*	
Line	**Featherstone**	**Eindhoven-Genoa**
1	**for** $i = 1$ **to** N_B **do**	**for** $i = 1$ **to** n_B **do**
2	**if** $\lambda(i) \neq 0$ **then**	**if** $\lambda(i) \neq 0$ **then**
3	$\frac{\partial \boldsymbol{v}_i}{\partial \boldsymbol{q}} = {}^i\boldsymbol{X}_{\lambda(i)} \frac{\partial \boldsymbol{v}_{\lambda(i)}}{\partial \boldsymbol{q}}$	$\frac{\partial \mathbf{v}_i}{\partial \mathbf{s}} = {}^i\mathbf{X}_{\lambda(i)} \frac{\partial \mathbf{v}_{\lambda(i)}}{\partial \mathbf{s}}$
4	$\frac{\partial \boldsymbol{v}_i}{\partial \boldsymbol{q}_i} = \frac{\partial \boldsymbol{v}_i}{\partial \boldsymbol{q}_i} + ({}^i\boldsymbol{X}_{\lambda(i)} \boldsymbol{v}_{\lambda(i)}) \times \boldsymbol{S}_i$	$\frac{\partial \mathbf{v}_i}{\partial \mathbf{s}_i} = \frac{\partial \mathbf{v}_i}{\partial \mathbf{s}_i} + ({}^i\mathbf{X}_{\lambda(i)} \mathbf{v}_{\lambda(i)}) \times \mathbf{\Gamma}_{\mathcal{J}i}$
5	$\frac{\partial \boldsymbol{a}_i}{\partial \boldsymbol{q}} = {}^i\boldsymbol{X}_{\lambda(i)} \frac{\partial \boldsymbol{a}_{\lambda(i)}}{\partial \boldsymbol{q}}$	$\frac{\partial \mathbf{a}_i}{\partial \mathbf{s}} = {}^i\mathbf{X}_{\lambda(i)} \frac{\partial \mathbf{a}_{\lambda(i)}}{\partial \mathbf{s}}$
6	$\frac{\partial \boldsymbol{a}_i}{\partial \boldsymbol{q}_i} = \frac{\partial \boldsymbol{a}_i}{\partial \boldsymbol{q}_i} + ({}^i\boldsymbol{X}_{\lambda(i)} \boldsymbol{a}_{\lambda(i)}) \times \boldsymbol{S}_i +$ $\frac{\partial \boldsymbol{v}_i}{\partial \boldsymbol{q}} \times \boldsymbol{v}_{\mathrm{J}}$	$\frac{\partial \mathbf{a}_i}{\partial \mathbf{s}_i} = \frac{\partial \mathbf{a}_i}{\partial \mathbf{s}_i} + ({}^i\mathbf{X}_{\lambda(i)} \mathbf{a}_{\lambda(i)}) \times \mathbf{\Gamma}_{\mathcal{J}i} +$ $\frac{\partial \mathbf{v}_i}{\partial \mathbf{s}} \times \mathbf{v}_{\mathcal{J}i}$
7	**end**	**end**
8	$\frac{\partial \boldsymbol{h}_i}{\partial \boldsymbol{q}} = \boldsymbol{I}_i \frac{\partial \boldsymbol{v}_i}{\partial \boldsymbol{q}}$	$\frac{\partial \mathbf{m}_{\mathcal{B}i}}{\partial \mathbf{s}} = \mathbb{M}_{\mathcal{B}i} \frac{\partial \mathbf{v}_i}{\partial \mathbf{s}}$
9	$\frac{\partial \boldsymbol{f}_i}{\partial \boldsymbol{q}} = \boldsymbol{I}_i \frac{\partial \boldsymbol{a}_i}{\partial \boldsymbol{q}} + \frac{\partial \boldsymbol{v}_i}{\partial \boldsymbol{q}} \times^* \boldsymbol{h}_i + \boldsymbol{v}_i \times^* \frac{\partial \boldsymbol{h}_i}{\partial \boldsymbol{q}}$	$\frac{\partial \mathbf{f}_{\mathcal{J}i}}{\partial \mathbf{s}} = \mathbb{M}_{\mathcal{B}i} \frac{\partial \mathbf{a}_i}{\partial \mathbf{s}} + \frac{\partial \mathbf{v}_i}{\partial \mathbf{s}} \bar{\times}^* \mathbf{m}_{\mathcal{B}i} + \mathbf{v}_i \bar{\times}^* \frac{\partial \mathbf{m}_{\mathcal{B}i}}{\partial \mathbf{s}}$
10	**end**	**end**
11	**for** $i = N_B$ **to** 1 **do**	**for** $i = n_B$ **to** 1 **do**
12	$\frac{\partial \boldsymbol{\tau}_i}{\partial \boldsymbol{q}} = \boldsymbol{S}_i^{\mathrm{T}} \frac{\partial \boldsymbol{f}_i}{\partial \boldsymbol{q}}$	$\frac{\partial \boldsymbol{\tau}_i}{\partial \mathbf{s}} = \mathbf{\Gamma}_{\mathcal{J}i}^T \frac{\partial \mathbf{f}_{\mathcal{J}i}}{\partial \mathbf{s}}$
13	**if** $\lambda(i) \neq 0$ **then**	**if** $\lambda(i) \neq 0$ **then**
14	$\frac{\partial \boldsymbol{f}_{\lambda(i)}}{\partial \boldsymbol{q}} = \frac{\partial \boldsymbol{f}_{\lambda(i)}}{\partial \boldsymbol{q}} + {}^{\lambda(i)}\boldsymbol{X}_i^* \frac{\partial \boldsymbol{f}_i}{\partial \boldsymbol{q}}$	$\frac{\partial \mathbf{f}_{\mathcal{J}\lambda(i)}}{\partial \mathbf{s}} = \frac{\partial \mathbf{f}_{\mathcal{J}\lambda(i)}}{\partial \mathbf{s}} + {}_{\lambda(i)}\mathbf{X}^i \frac{\partial \mathbf{f}_{\mathcal{J}i}}{\partial \mathbf{s}}$
15	$\frac{\partial \boldsymbol{f}_{\lambda(i)}}{\partial \boldsymbol{q}_i} = \frac{\partial \boldsymbol{f}_{\lambda(i)}}{\partial \boldsymbol{q}_i} + {}^{\lambda(i)}\boldsymbol{X}_i^* (\boldsymbol{S}_i \times^* \boldsymbol{f}_i)$	$\frac{\partial \mathbf{f}_{\mathcal{J}\lambda(i)}}{\partial \mathbf{s}_i} = \frac{\partial \mathbf{f}_{\mathcal{J}\lambda(i)}}{\partial \mathbf{s}_i} + {}_{\lambda(i)}\mathbf{X}^i (\mathbf{\Gamma}_{\mathcal{J}i} \bar{\times}^* \mathbf{f}_{\mathcal{J}i})$
16	**end**	**end**
17	**end**	**end**
Outputs	$\partial \boldsymbol{\tau} / \partial \boldsymbol{q}$	$\partial \boldsymbol{\tau} / \partial \mathbf{s}$

E.8.2 Derivatives of fixed-base inverse dynamics w.r.t. generalized velocity vector

The algorithm presented below makes use of the following equation [7]:

$$\frac{\partial \mathbf{v}_{\mathcal{J}i}}{\partial \mathbf{r}_i} = \mathbf{\Gamma}_{\mathcal{J}i}. \tag{E.5}$$

Table E.9: Derivatives of fixed-base inverse dynamics w.r.t. generalized velocity vector $\mathbf{r}$.

Inputs	*All outputs and intermediate variables of RNEA*	
Line	**Featherstone**	**Eindhoven-Genoa**
1	**for** $i = 1$ **to** N_B **do**	**for** $i = 1$ **to** n_B **do**
2	**if** $\lambda(i) \neq 0$ **then**	**if** $\lambda(i) \neq 0$ **then**
3	$\frac{\partial \boldsymbol{v}_i}{\partial \dot{\boldsymbol{q}}} = {}^{i}\boldsymbol{X}_{\lambda(i)} \frac{\partial \boldsymbol{v}_{\lambda(i)}}{\partial \dot{\boldsymbol{q}}}$	$\frac{\partial \mathbf{v}_i}{\partial \mathbf{r}} = {}^{i}\mathbf{X}_{\lambda(i)} \frac{\partial \mathbf{v}_{\lambda(i)}}{\partial \mathbf{r}}$
4	$\frac{\partial \boldsymbol{a}_i}{\partial \dot{\boldsymbol{q}}} = {}^{i}\boldsymbol{X}_{\lambda(i)} \frac{\partial \boldsymbol{a}_{\lambda(i)}}{\partial \dot{\boldsymbol{q}}}$	$\frac{\partial \mathbf{a}_i}{\partial \mathbf{r}} = {}^{i}\mathbf{X}_{\lambda(i)} \frac{\partial \mathbf{a}_{\lambda(i)}}{\partial \mathbf{r}}$
5	$\frac{\partial \boldsymbol{a}_i}{\partial \dot{\boldsymbol{q}}_i} = \frac{\partial \boldsymbol{a}_i}{\partial \dot{\boldsymbol{q}}_i} + \frac{\partial \boldsymbol{v}_i}{\partial \dot{\boldsymbol{q}}} \times \boldsymbol{v}_{\mathrm{J}} + \boldsymbol{v}_i \times \boldsymbol{S}_i$	$\frac{\partial \mathbf{a}_i}{\partial \mathbf{r}_i} = \frac{\partial \mathbf{a}_i}{\partial \mathbf{r}_i} + \frac{\partial \mathbf{v}_i}{\partial \mathbf{r}} \times \mathbf{v}_{\mathcal{J}i} + \mathbf{v}_i \times \mathbf{\Gamma}_{\mathcal{J}i}$
6	**end**	**end**
7	$\frac{\partial \boldsymbol{v}_i}{\partial \dot{\boldsymbol{q}}_i} = \frac{\partial \boldsymbol{v}_i}{\partial \dot{\boldsymbol{q}}_i} + \boldsymbol{S}_i$	$\frac{\partial \mathbf{v}_i}{\partial \mathbf{r}_i} = \frac{\partial \mathbf{v}_i}{\partial \mathbf{r}_i} + \mathbf{\Gamma}_{\mathcal{J}i}$
8	$\frac{\partial \boldsymbol{h}_i}{\partial \dot{\boldsymbol{q}}} = \boldsymbol{I}_i \frac{\partial \boldsymbol{v}_i}{\partial \dot{\boldsymbol{q}}}$	$\frac{\partial \mathbf{m}_{\mathcal{B}i}}{\partial \mathbf{r}} = \mathbb{M}_{\mathcal{B}i} \frac{\partial \mathbf{v}_i}{\partial \mathbf{r}}$
9	$\frac{\partial \boldsymbol{f}_i}{\partial \dot{\boldsymbol{q}}} = \boldsymbol{I}_i \frac{\partial \boldsymbol{a}_i}{\partial \dot{\boldsymbol{q}}} + \frac{\partial \boldsymbol{v}_i}{\partial \dot{\boldsymbol{q}}} \times^* \boldsymbol{h}_i + \boldsymbol{v}_i \times^* \frac{\partial \boldsymbol{h}_i}{\partial \dot{\boldsymbol{q}}}$	$\frac{\partial \mathbf{f}_{\mathcal{J}i}}{\partial \mathbf{r}} = \mathbb{M}_{\mathcal{B}i} \frac{\partial \mathbf{a}_i}{\partial \mathbf{r}} + \frac{\partial \mathbf{v}_i}{\partial \mathbf{r}} \bar{\times}^* \mathbf{m}_{\mathcal{B}i} + \mathbf{v}_i \bar{\times}^* \frac{\partial \mathbf{m}_{\mathcal{B}i}}{\partial \mathbf{r}}$
10	**end**	**end**
11	**for** $i = N_B$ **to** 1 **do**	**for** $i = n_B$ **to** 1 **do**
12	$\frac{\partial \boldsymbol{\tau}_i}{\partial \dot{\boldsymbol{q}}} = \boldsymbol{S}_i^{\mathrm{T}} \frac{\partial \boldsymbol{f}_i}{\partial \dot{\boldsymbol{q}}}$	$\frac{\partial \boldsymbol{\tau}_i}{\partial \mathbf{r}} = \mathbf{\Gamma}_{\mathcal{J}i}^{T} \frac{\partial \mathbf{f}_{\mathcal{J}i}}{\partial \mathbf{r}}$
13	**if** $\lambda(i) \neq 0$ **then**	**if** $\lambda(i) \neq 0$ **then**
14	$\frac{\partial \boldsymbol{f}_{\lambda(i)}}{\partial \dot{\boldsymbol{q}}} = \frac{\partial \boldsymbol{f}_{\lambda(i)}}{\partial \dot{\boldsymbol{q}}} + {}^{\lambda(i)}\boldsymbol{X}_i^* \frac{\partial \boldsymbol{f}_i}{\partial \dot{\boldsymbol{q}}}$	$\frac{\partial \mathbf{f}_{\mathcal{J}\lambda(i)}}{\partial \mathbf{r}} = \frac{\partial \mathbf{f}_{\mathcal{J}\lambda(i)}}{\partial \mathbf{r}} + {}_{\lambda(i)}\mathbf{X}^{i} \frac{\partial \mathbf{f}_{\mathcal{J}i}}{\partial \mathbf{r}}$
15	**end**	**end**
16	**end**	**end**
Outputs	$\partial \boldsymbol{\tau} / \partial \dot{\boldsymbol{q}}$	$\partial \boldsymbol{\tau} / \partial \mathbf{r}$

Bibliography

[1] Jonas Koenemann, Andrea Del Prete, Yuval Tassa, Emanuel Todorov, Olivier Stasse, Maren Bennewitz, and Nicolas Mansard. Whole-body model-predictive control applied to the HRP-2 humanoid. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3346–3351, 2015.

[2] Perle Geoffroy, Nicolas Mansard, Maxime Raison, Sofiane Achiche, and Emo Todorov. From inverse kinematics to optimal control. In Jadran Lenarčič and Carlo Galletti, editors, *Advances in Robot Kinematics*, pages 409–418. Springer, 2014.

[3] Farbod Farshidian, Edo Jelavic, Asutosh Satapathy, Markus Giftthaler, and Jonas Buchli. Real-time motion planning of legged robots: A model predictive control approach. *IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, pages 577–584, 2017.

[4] Michael Posa, Cecilia Cantu, and Russ Tedrake. A direct method for trajectory optimization of rigid bodies through contact. *The International Journal of Robotics Research*, 33(1):69–81, 2014.

[5] Yuval Tassa, Tom Erez, and Emanuel Todorov. Synthesis and stabilization of complex behaviors through online trajectory optimization. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4906–4913, 2012.

[6] Hassan K Khalil. *Nonlinear systems.* Prentice hall, 2002.

[7] Justin Carpentier and Nicolas Mansard. Analytical derivatives of rigid body dynamics algorithms. *Robotics: Science and Systems (RSS)*, 2018.

[8] Roy Featherstone. *Rigid body dynamics algorithms.* Springer, 2008.

[9] Marcelo H Ang and Vassilios D Tourassis. Singularities of Euler and roll-pitch-yaw representations. *IEEE Transactions on Aerospace and Electronic Systems*, pages 317–324, 1987.

[10] James Diebel. Representing attitude: Euler angles, unit quaternions, and rotation vectors. *Matrix*, 58(15-16):1–35, 2006.

[11] Alessandro Saccon, John Hauser, and A Pedro Aguiar. Optimal control on Lie groups: The projection operator approach. *IEEE Transactions on Automatic Control*, 58(9):2230–2245, 2013.

[12] Valentin Sonneville and Olivier Brüls. Sensitivity analysis for multibody systems formulated on a Lie group. *Multibody System Dynamics*, 31(1):47–67, 2014.

[13] Richard M Murray. *A mathematical introduction to robotic manipulation.* CRC press, 2017.

[14] Pål J From, Vincent Duindam, Kristin Y Pettersen, Jan T Gravdahl, and Shankar Sastry. Singularity-free dynamic equations of vehicle–manipulator systems. *Simulation Modelling Practice and Theory*, 18(6):712–731, 2010.

[15] Daniele Pucci, Silvio Traversaro, and Francesco Nori. Momentum control of an underactuated flying humanoid robot. *IEEE Robotics and Automation Letters*, 3(1):195–202, 2018.

[16] Ko Ayusawa, Gentiane Venture, and Yoshihiko Nakamura. Identification of humanoid robots dynamics using floating-base motion dynamics. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2854–2859, 2008.

[17] Alessandro Saccon, A Pedro Aguiar, and John Hauser. Lie group projection operator approach: Optimal control on T SO (3). *IEEE Decision and Control and European Control Conference (CDC-ECC)*, pages 6973–6978, 2011.

[18] Joan Solà, Jeremie Deray, and Dinesh Atchuthan. A micro Lie theory for state estimation in robotics. *arXiv preprint arXiv:1812.01537*, 2018.

[19] John YS Luh, Michael W Walker, and Richard PC Paul. On-line computational scheme for mechanical manipulators. *J. DYN. SYS. MEAS. & CONTR.*, 102(2):69–76, 1980.

[20] Michael W Walker and David E Orin. Efficient dynamic computer simulation of robotic mechanisms. *Journal of Dynamic Systems, Measurement, and Control*, 104(3):205–211, 1982.

[21] Gianluca Garofalo, Christian Ott, and Alin Albu-Schäffer. On the closed form computation of the dynamic matrices and their differentiations. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2364–2359, 2013.

[22] Markus Giftthaler, Michael Neunert, Markus Stäuble, Marco Frigerio, Claudio Semini, and Jonas Buchli. Automatic differentiation of rigid body dynamics for optimal control and estimation. *Advanced Robotics*, 31(22):1225–1237, 2017.

[23] Sean Mason, Ludovic Righetti, and Stefan Schaal. Full dynamics lqr control of a humanoid robot: An experimental study on balancing and squatting. *IEEE-RAS International Conference on Humanoid Robots*, pages 374–379, 2014.

[24] Michael Neunert, Markus Giftthaler, Marco Frigerio, Claudio Semini, and Jonas Buchli. Fast derivatives of rigid body dynamics for control, optimization and estimation. *IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*, pages 91–97, 2016.

[25] Kurt S Anderson and YuHung Hsu. Analytical fully-recursive sensitivity analysis for multibody dynamic chain systems. *Multibody System Dynamics*, 8(1):1–27, 2002.

[26] 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*, 123(3):391–399, 2001.

[27] Frank C Park, Beobkyoon Kim, Cheongjae Jang, and Jisoo Hong. Geometric algorithms for robot dynamics: A tutorial review. *Applied Mechanics Reviews*, 70(1):010803, 2018.

[28] Justin Carpentier. Analytical inverse of the joint space inertia matrix. *available online at hal.laas.fr*, 2018.

[29] Peter Eberhard. Adjoint variable method for sensitivity analysis of multibody systems interpreted as a continuous, hybrid form of automatic differentiation. *Computational Differentiation Techniques, Applications, and Tools, M. Berz, C. Bischof, G. Corliss, and A. Griewank, A., eds., SIAM, Philadelphia, PA*, pages 319–328, 1996.

[30] Silvio Traversaro and Alessandro Saccon. Multibody dynamics notation, revision 2. *available online at tue.research.nl*, 2019.

[31] Alessandro Saccon, John Hauser, and Alessandro Beghi. Trajectory exploration of a rigid motorcycle model. *IEEE Transactions on Control Systems Technology*, 20(2):424–437, 2012.

[32] Nalin A Chaturvedi, Amit K Sanyal, and N Harris McClamroch. Rigid-body attitude control. *IEEE control systems magazine*, 31(3):30–51, 2011.

[33] Veeravalli S Varadarajan. *Lie groups, Lie algebras, and their representations*, volume 102. Springer, 2013.

[34] Wulf Rossmann et al. *Lie groups: an introduction through linear groups*, volume 5. Oxford University Press, 2002.

[35] Jon M Selig. *Geometric fundamentals of robotics.* Springer, 2004.

[36] Stefano Stramigioli and Herman Bruyninckx. Geometry and screw theory for robotics. *available online at psu.edu*, 2001:75, 2001.

[37] T. De Laet, S. Bellens, R. Smits, E. Aertbeliën, H. Bruyninckx, and J. De Schutter. Geometric relations between rigid bodies (part 1): Semantics for standardization. *Robotics & Automation Magazine, IEEE*, 20(1):84–93, 2013.

[38] M.W. Spong, S. Hutchinson, and M. Vidyasagar. *Robot modeling and control*, volume 3. Wiley New York, 2006.

[39] Silvio Traversaro, Daniele Pucci, and Francesco Nori. A unified view of the equations of motion used for control design of humanoid robots. *available online at researchgate.net*, 2017.

[40] Francesco Nori, Silvio Traversaro, Jorhabib Eljaik, Francesco Romano, Andrea Del Prete, and Daniele Pucci. icub whole-body control through force regulation on rigid non-coplanar contacts. *Frontiers in Robotics and AI*, 2:6, 2015.

[41] Daniele Pucci, Gabriele Nava, and Francesco Nori. Automatic gain tuning of a momentum based balancing controller for humanoid robots. In *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pages 158–164. IEEE, 2016.

[42] Roy Featherstone. The acceleration vector of a rigid body. *The International Journal of Robotics Research*, 20(11):841–846, 2001.

[43] Roy Featherstone. RNEA. `http://royfeatherstone.org/spatial/v2/#ID`. Online; accessed 19th April 2019.

[44] Roy Featherstone. RNEAmb. `http://royfeatherstone.org/spatial/v2/#IDfb`. Online; accessed 19th April 2019.

[45] Roy Featherstone. ABA. `http://royfeatherstone.org/spatial/v2/#FDab`. Online; accessed 19th April 2019.

[46] Roy Featherstone. ABAmb. `http://royfeatherstone.org/spatial/v2/#FDfb`. Online; accessed 19th April 2019.