RIGID-BODY DYNAMICS ALGORITHMS FOR PHYSICALLY INTERACTIVE ROBOTS

A Thesis

Submitted to the College of Engineering
of the University of Notre Dame
in Partial Fulfillment of the Requirements
for the Degree of

Bachelors of Science
in
Mechanical Engineering

by

Sebastian F. Echeandia

Dr. Patrick M. Wensing, Advisor

Undergraduate Program in Department of Aerospace and Mechanical Engineering
Notre Dame, Indiana

April 2020

RIGID-BODY DYNAMICS ALGORITHMS FOR PHYSICALLY INTERACTIVE ROBOTS

Abstract

by

Sebastian F. Echeandia

The development of high performance physically interactive robots has motivated shifts in the design of actuators and novel control strategies. Mobile robots are changing their high-impedance actuators for less stiff actuators with low gear ratios. For these systems, model-based controllers have achieved unprecedented levels of dexterity and stability compared to other control strategies. These developments have increased the need for dynamics algorithms that can accurately compute the dynamics of the system as part of the control policy. This work presents three new algorithms that can be implemented in model-based controllers that require certain dynamics terms such as the Mass or Coriolis matrices. The first objective is to exploit a particular factorization of body-level velocity-product terms to derive an algorithm that numerically computes the Coriolis matrix. This algorithm is of the lowest possible order, outperforming other algorithms available in the literature. The second objective is to expand the mathematical framework to compute the Coriolis matrix to develop an algorithm that numerically computes the corresponding Christoffel symbols. Because of its low complexity, this algorithm is relevant for geometric control schemes that do not scale based on the use of symbolic calculations. The final objective is to improve the Articulated-Body Algorithm (ABA) to take into account al inertial effects of motor rotors by using the Gauss Principle of Least Constraints. This

algorithm provides a more accurate alternative to other standard approximations to include actuators' inertial effects. Tests using simulations of different kinematic trees show that, given their accuracy, speed, and efficiency, the three algorithms developed in this work are viable for implementation with real-time controllers.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Motivation

Many of the current research efforts in robotics are focusing on developing robots that can safely navigate human environments and interact with people in everyday places like offices and construction sites. The challenges presented in these environments, such as terrain unpredictability or human safety, have pushed researchers and engineers to design robots with compliant actuators that can deliver high torques while maintaining back-driveability. Boston Dynamics' quadrupeds evolution from Big Dog to Spot (Figure 1.1) illustrates the current trend in mobile robot design of shifting from large, high-impedance actuators to smaller, less stiff actuators with low gear ratios. This shift means that robots are designed with heavier motors and lighter gearboxes. The lower impedance of new robotic actuators allows robots to adapt to irregular terrains but also increases their energy efficiency and mechanical performance [23]. With these new actuators, robots can perform tasks that require accurate control of contact forces during scenarios like climbing stairs with unpredictable obstacles (Figure 1.2).

These high-performance tasks require optimization of contact forces to be able to achieve the desired dexterity and to maintain stability. Such optimization can be done at the planning level, but also at the control level by tracking the desired planned motion [22]. These model-based approaches to control need to consider the dynamics of the system to select joint torques and forces that will make the system follow

Figure 1.1. Boston Dynamics' quadrupeds evolution over time.

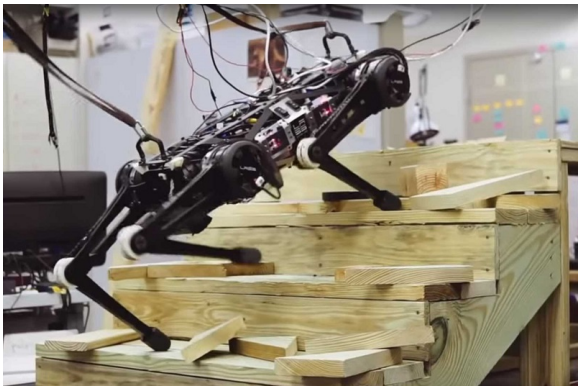


Figure 1.2. MIT Cheetah 3 climbing stairs with obstacles.

the desired behavior. Computing the dynamics of mobile robots can be a complex challenge given the large number of degrees of freedom (DoFs) of the system, which in turn increases the complexity of the equations of motion. Consequently, for these controllers to be feasible in high-DoF robots, they need to be matched with accurate and efficient rigid-body dynamics algorithms that can compute components of the equations of motions of the system fast enough for a real-time controller. Given that standard closed-loop controllers run in the order of ms, dynamics algorithms need to run in the order of μs to feed information to the control loop on time.

Because of their low computational complexity, recursive dynamics algorithms have had great success at numerically evaluating dynamics fast enough for model-based controllers. Some successful strategies include using the operational-space formulation of the dynamics of the system that enables decoupling task and null-space dynamics to achieve high dynamic performance and active force control of robot manipulator systems [15, 30]. Other popular approaches for control consider forward or inverse dynamics. In forward dynamics, the objective is to find the joint accelerations of the system, given external contact forces and internal forces and torques [21]. Inverse dynamics consists of solving for the internal forces of the system given its motion. Both of these problems can be solved using one of two main approaches: Lagrangian mechanics [12, 18] or recursive algorithms [16].

While existing dynamics algorithms are cornerstones of state-of-the-art model-based control strategies, certain challenges remain active topics of research. In particular, some applications of sensorless detection of faults acting on a robot require the computation of a residual vector using the quantity of $\mathbf{C}^\top(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}$ [2, 5], where the matrix $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ is a factorization of the Coriolis and centrifugal terms of the equations of motion. In this application, and in other robot controls laws, the $\mathbf{C}$ matrix must satisfy that $\dot{\mathbf{H}}(\mathbf{q}, \dot{\mathbf{q}}) - 2\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ is skew-symmetric, where $\mathbf{H}(\mathbf{q})$ is the joint-space inertia matrix. Given the transpose in $\mathbf{C}$, traditional Newton-Euler algorithms cannot

compute this residual vector. Similarly, research efforts in the area of gradient-based motion optimization are exploring computing components of the equations of motion, such as $\mathbf{C}$, to support optimization methods that require the calculation of partial derivatives of the dynamics [3, 13, 14, 17, 26, 27]. For moderately complex systems, these partial derivatives become prohibitively complex to be computed exactly, so they are often approximated when feeding this information into the control system.

These two examples showcase that the improvement of existing recursive algorithms and the development of new dynamics algorithms not only will allow to solve increasingly complex problems building upon forward and inverse dynamics, but will also enable the implementation of new and better control schemes. As model-based controllers evolve, efficient and accurate dynamics algorithms will continue to present a latent need. As such, the development of these algorithms is a pursuit with the potential to enable further progress in the broader field of robotics.

1.2 Objectives

The overarching objective of this research is to use numerical methods to derive new rigid-body dynamics algorithms that can be implemented in model-based controllers that require dynamic terms, such as $\mathbf{H}(\mathbf{q})$ and $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ as part of their control scheme. As such, for this project to be successful, the created algorithms must fulfill three criteria for feasibility: accuracy, speed, and efficiency. The algorithms developed in this work must equal, if not exceed, the accuracy and precision offered by other computational methods available in the literature. Similarly, the algorithms must be fast enough to run in a real-time control loop. The target runtime for each algorithm is on the order of μs for systems of a moderate number of DoFs. Finally, for the algorithms to be efficient, they must be of the lowest possible order, which means that the computational costs of the algorithm grow proportionally with the lowest possible power of the DoF of the system. These three criteria will ensure that

new algorithms are viable candidates for implementation in online controllers.

The first objective of this thesis is to exploit the link between the factorization of body-level velocity-product terms in the equations of motion of a system to the Coriolis matrix $\mathbf{C}$ to develop a recursive algorithm that can numerically compute $\mathbf{C}$ directly from the motion of the system. The developed algorithm has a computational cost of $O(Nd)$, where N is the number of bodies, and d is the depth of the kinematic tree of the mechanism. This efficiency is superior to that of other methods available in the literature that have a complexity of $O(N^2)$ [5]. This algorithm can be implemented in applications of sensorless detection, but also in passivity-based control, and adaptive control [25, 28].

The second objective is to take the mathematical framework developed for the new algorithm to compute $\mathbf{C}$ and further expand it to develop a recursive algorithm that numerically computes the underlying Christoffel symbols of the first kind of tree-structured rigid-body systems. The complexity of the developed algorithm grows as $O(Nd^2)$, which is much more efficient than the symbolic computation. Because this algorithm does not require any symbolic partial derivatives, it is an attractive alternative to approximation methods currently used in various robotics applications. In particular, this algorithm might prove useful for geometric control schemes that do not scale based on the use of symbolic calculations or in supporting trajectory optimization packages by providing analytical second-order partial derivatives of the inverse dynamics model.

Finally, the third objective is to improve the Articulated-Body Algorithm (ABA) to accurately take into account the inertial effects of motor rotors. The computational cost of the modified ABA grows as $O(N)$ so that it matches the performance of the regular ABA. ABA is a well-established forward dynamics algorithm to compute the joint accelerations; however, in its traditional form, it lacks the ability to account for the inertial effects of any component in the system other than the main bodies

of the kinematic tree. There are approximation techniques to account for inertial effects of motors, but they are only valid for actuators with large gear ratios. The Gauss principle of least constraint allows rederiving the ABA in such a way that it fundamentally accounts for rotor inertias. As more robotic systems shift towards lower gear ratios, it is essential to develop algorithms that can accurately solve the complicated dynamics of robotic systems doing high-performance tasks in uncertain environments.

1.3 Overview

Chapter 2 of this thesis provides the preliminary knowledge required for the derivation of the algorithms presented in this work. It presents a general overview of the spatial vector notation used to express the dynamics of rigid bodies more compactly. It also presents the canonical equations of motion and explores the properties that allow for the mathematical framework needed to derive new algorithms. Finally, it reviews three algorithms (Recursive Newton-Euler, Composite Rigid-Body, and Articulated-Body) that set the general recursive strategies used by the algorithms derived in this work.

Chapter 3 presents the derivation of an algorithm to compute $\mathbf{C}$ through a particular factorization of the equations of motion that allows for the recursive computation of all entries in $\mathbf{C}$. It also shows how, through some mathematical manipulation, the same factorization leads to expressions that can be used to recursively compute the Christoffel symbols without any symbolic manipulations. Chapter 4 shows how the Gauss principle of least constraint leads to the derivation of an Articulated-Body Algorithm that accurately accounts for rotor inertias. It then compares the results of the new algorithm with the traditional ABA and a modified version of the Composite Rigid-Body Algorithm that provides an approximation for the inertia effects of rotors with large gear ratios. Finally, Chapter 5 provides concluding remarks for the work

shown throughout this thesis and suggests relevant future work in light of the results.

CHAPTER 2

PRELIMINARIES: SPATIAL VECTOR NOTATION AND RIGID-BODY DYNAMICS ALGORITHMS

2.1 Introduction

This chapter presents the notation, conventions, and mathematical formalisms that are the foundation for the algorithms developed in this work. In particular, this chapter provides an overview of spatial notation and introduction to the 6D vectors used to describe rigid-body dynamics in a compact notation. The conventions to describe link connectivity within a kinematic tree are also presented, and will play an important role in the derivation of the algorithms. The chapter discusses the equations of motion for rigid-body systems and some of their relevant mathematical properties that are exploited by the algorithms in later chapters. Finally, the chapter concludes by presenting three algorithms whose overall structure and mechanics serve as a basis for the new algorithms presented in Chapter 3 and Chapter 4. Three key algorithms are summarized, namely 1) The Recursive Newton-Euler Algorithm (RNEA) that computes the forces and torques at each joint given joint velocity and acceleration; 2) The Composite Rigid-Body Algorithm (CRBA) that computes the mass matrix; and 3) the Articulated-Body Algorithm (ABA) that computes the joint acceleration given the joint configuration, velocities, forces, and torques.

2.2 Kinematic Tree Connectivity

Any rigid-body system can be described as a set of bodies connected by a set of joints with up to 6 DoFs each. This work considers kinematic trees, where N denotes the number of bodies and d the depth of the tree. Each body in the tree is assigned a number from 1 to N, and a "body" 0 is assigned to represent a fixed inertial frame. The numbering of the bodies is assigned such that for body i, its predecessor towards the root of the tree, $p(i)$, is less than i. This numbering system establishes a binary relation between bodies denoted as $\preceq$ such that $j \preceq i$ indicates that body j is in the path from body i to the root of the tree. Any pair of bodies i and j are related, expressed as $i \sim j$, if $j \preceq i$ or $i \preceq j$.

This convention is best understood through an example. Consider the branched mechanism in Figure 2.1. For $i = 6$ and $j = 3$, $3 \preceq 6$, so i and j are related. Similarly, if $i = 2$ and $j = 10$, $2 \preceq 10$, so i and j are related. When $i = 9$ and $j = 5$, however, neither of the two bodies is in the path to the root of each other; therefore, i and j are not related in this case.

Given these conventions, an additional definition will aid in the development of the algorithms in Chapter 3. For any pair of related bodies $i \sim j$, the symbol $\lceil ij \rceil$ returns the body that is closest to the leaves of the kinematic tree. That is, $\lceil ij \rceil$ is defined as

$$\lceil ij \rceil = \begin{cases} i & i \succeq j \\ j & \text{o/w.} \end{cases}.$$

Consider mechanism in Figure 2.1 again as an example. If $i = 7$ and $j = 3$, then $\lceil ij \rceil = 7$ since $i \succeq j$. Similarly, when $i = 9$ and $j = 2$, $\lceil ij \rceil$ would output 9 because $j \succeq i$. For cases when i and j are not related, say $i = 4$ and $j = 6$, $\lceil ij \rceil$ is undefined since neither of the bodies is an ancestor of the other, that is, is in the path to the root of each other.

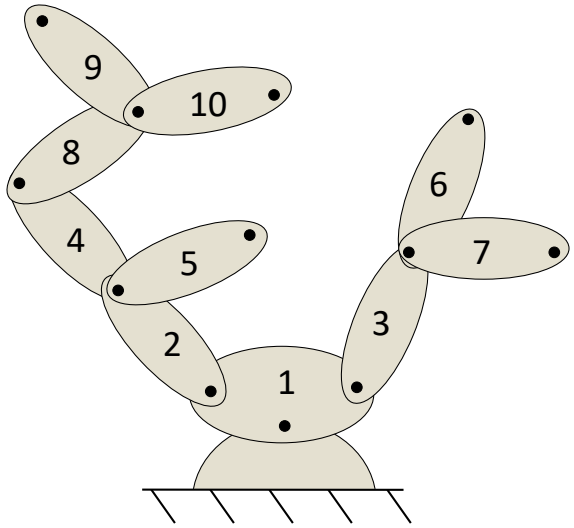


Figure 2.1. A sample branched mechanism.

2.3 Spatial Vector Notation

Spatial vectors are a convenient notation to express the equations of motion of rigid-body systems in terms of vectors that live in a 6D vector space. Spatial vectors significantly reduce the notational complexity of rigid-body dynamics expressed using traditional 3D vectors. The reduced complexity makes the properties of components of the equation of motion more apparent, which aids in the development of more advanced algorithms. This section is intended as a short introduction; the interested reader may refer to the in-depth explanation by Featherstone [8].

Consider the kinematic tree in Figure 2.2. To describe the motion of this system, each body in the tree has attached a corresponding coordinate frame as shown at joints 1, 2, and 3. These frames allow the motion of each body to be conveniently described in a local basis. With this convention, the rate of change in position and

orientation of body i is expressed in terms of a spatial velocity $\mathbf{v}_i \in \mathbb{R}^6$ as

$$\mathbf{v}_i = \begin{bmatrix} \boldsymbol{\omega}_i \\ \boldsymbol{v}_i \end{bmatrix}, \tag{2.1}$$

where $\boldsymbol{\omega}_i \in \mathbb{R}^3$ and $\boldsymbol{v}_i \in \mathbb{R}^3$ are the angular velocity of body i and linear velocity of the origin of frame i in body coordinates, respectively. Joint i connects body i with its predecessor $p(i)$ so that the velocity of body i is related to the velocity of its predecessor as [4]

$$\mathbf{v}_i = {}^i\mathbf{X}_{p(i)}\mathbf{v}_{p(i)} + \boldsymbol{\Phi}_i\dot{\mathbf{q}}_i, \tag{2.2}$$

where $\dot{\mathbf{q}}_i \in \mathbb{R}^{n_i}$ represents the joint rates of joint i and n_i indicates the number of DoF of joint i. The term $\boldsymbol{\Phi}_i \in \mathbb{R}^{6\times n_i}$ is a full-column-rank matrix that describes the free modes of motion of joint i.

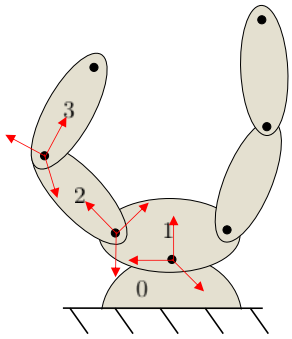


Figure 2.2. Sample kinematic tree with body coordinate frames.

The matrix ${}^i\mathbf{X}_{p(i)} \in \mathbb{R}^{6\times 6}$ is the spatial transformation matrix that transforms the basis of spatial vectors from that of frame $p(i)$ into that of frame i. It is constructed

in terms of the vector ${}^{p(i)}\boldsymbol{p}_i \in \mathbb{R}^3$ from the origin of $p(i)$ to the origin of i and the rotation matrix ${}^{i}\boldsymbol{R}_{p(i)}$ that transforms traditional 3D vectors from $p(i)$'s frame into i's. Overall the spatial transform is given by

$$ {}^{i}\mathbf{X}_{p(i)} = \begin{bmatrix} {}^{i}\boldsymbol{R}_{p(i)} & \mathbf{0} \\ -{}^{i}\boldsymbol{R}_{p(i)}\boldsymbol{S}({}^{p(i)}\boldsymbol{p}_i) & {}^{i}\boldsymbol{R}_{p(i)}, \end{bmatrix}, \tag{2.3} $$

where $\boldsymbol{S}(\boldsymbol{p})$ is the skew-symmetric 3D cross product matrix defined for any 3D vector $\boldsymbol{p} = [p_x, p_y, p_z]^T$ as

$$ \boldsymbol{S}(\boldsymbol{p}) = \begin{bmatrix} 0 & -p_z & p_y \\ p_z & 0 & -p_x \\ -p_y & p_x & 0 \end{bmatrix}. \tag{2.4} $$

Similarly, the spatial force acting on body i, $\mathbf{f}_i$, is defined in terms of the 3D torque and linear force so that

$$ \mathbf{f}_i = \begin{bmatrix} \boldsymbol{n}_i \\ \boldsymbol{f}_i \end{bmatrix}, \tag{2.5} $$

where $\boldsymbol{n}_i \in \mathbb{R}^3$ is the moment about the origin of frame i and $\boldsymbol{f}_i \in \mathbb{R}^3$ is the linear force expressed in body coordinates. The spatial force of a body is related to its motion as [10]

$$ \mathbf{f}_i = \mathbf{I}_i\mathbf{a}_i + \mathbf{v}_i \times^* \mathbf{I}_i\mathbf{v}_i, \tag{2.6} $$

where $\mathbf{a}_i \in \mathbb{R}^6$ represents the spatial acceleration of body i and $\mathbf{I}_i \in \mathbb{R}^{6\times6}$ is the spatial inertia tensor that links the spatial velocity of body i to its spatial momentum. This

tensor is defined as

$$\mathbf{I}_i = \begin{bmatrix} \bar{\boldsymbol{I}}_i & m_i \boldsymbol{S}(\boldsymbol{c}_i) \\ m_i \boldsymbol{S}(\boldsymbol{c}_i)^\top & m_i \mathbf{1}_3 \end{bmatrix}, \tag{2.7}$$

where m_i is the mass of body i, $\boldsymbol{c}_i$ is the vector to the center of mass of body i expressed in local coordinates, $\mathbf{1}_3 \in \mathbb{R}^{3\times3}$ is the identity matrix, and $\bar{\boldsymbol{I}}_i$ is the conventional 3D rotational inertial tensor about the coordinate origin with elements

$$\bar{\boldsymbol{I}}_i = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{xy} & I_{yy} & I_{yz} \\ I_{xz} & I_{yz} & I_{zz} \end{bmatrix}. \tag{2.8}$$

For any spatial velocity $\mathbf{v}$, the operator $\times^* : \mathbb{R}^6 \times \mathbb{R}^6 \to \mathbb{R}^6$ is the bi-linear cross product operator between spatial motion vectors and spatial force vectors expressed as

$$\mathbf{v} \times^* \mathbf{f} = \begin{bmatrix} \boldsymbol{S}(\boldsymbol{\omega}) & \boldsymbol{S}(\boldsymbol{v}) \\ \mathbf{0} & \boldsymbol{S}(\boldsymbol{\omega}) \end{bmatrix} \mathbf{f}. \tag{2.9}$$

Physically, $\mathbf{v} \times^* \mathbf{f}$ gives the rate of change in the force vector $\mathbf{f}$ when moving with velocity $\mathbf{v}$.

The cross product operator between spatial motion vectors and spatial motion vectors is defined in terms of $\times^*$ as $(\mathbf{v}\times) = -(\mathbf{v}\times^*)^\top$. Additionally, by swapping the order of the products of the cross product, a new unique linear operator is denoted so that $(\mathbf{f}\bar{\times}^*)\mathbf{v} = (\mathbf{v}\times^*)\mathbf{f}$. Then, the $\bar{\times}^*$ operator is expressed as

$$\mathbf{f}\bar{\times}^* = \begin{bmatrix} -\boldsymbol{S}(\boldsymbol{n}) & -\boldsymbol{S}(\boldsymbol{f}) \\ -\boldsymbol{S}(\boldsymbol{f}) & \mathbf{0} \end{bmatrix}. \tag{2.10}$$

The spatial inertia tensor is also subject to coordinate transformations using spatial transforms. The tensor ${}^{j}\mathbf{I}_i$, the spatial inertia of i expressed in body coordinates of j, is expressed as

$$ {}^{j}\mathbf{I}_i = {}^{i}\mathbf{X}_j^{\top}\, \mathbf{I}_i\, {}^{i}\mathbf{X}_j. \tag{2.11} $$

When used together, the quantities and equations presented in this chapter are sufficient to describe the dynamics of a rigid body and will be the foundation to describe the motion of a multi-body kinematic tree.

2.4 Equations of Motion

The equations of motion for a rigid-body system describe the motion of each body in the system based on kinematic quantities as well as dynamic components such as forces, torques, and gravity. In general, these equations can be derived for any particular system using Newton's second law or via Lagrangian methods. In the robotics community, the equations of motion are often expressed in two different formalisms: operational space and joint space. Operational space is often associated with the position of the end-effector, while joint space is expressed in terms of the position and orientation of joints in rigid-body systems. This thesis uses joint space as the basis for the math needed to develop new dynamics algorithms.

The equations of motion of any rigid-body system can be expressed as [1]

$$ \mathbf{H}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g} = \boldsymbol{\tau}, \tag{2.12} $$

where $\mathbf{H}(\mathbf{q}) \in \mathbb{R}^{n\times n}$ is the mass matrix containing inertial terms, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) \in \mathbb{R}^{n\times n}$ is the Coriolis matrix containing Coriolis and centrifugal force terms, $\mathbf{g} \in \mathbb{R}^n$ is the gravity force, and $\boldsymbol{\tau} \in \mathbb{R}^n$ the generalized forces in the system. For kinematic trees, $\mathbf{q}$

represents every joint variable in the mechanism contained as generalized coordinates [8]. Similarly, $\dot{\mathbf{q}}$ and $\ddot{\mathbf{q}}$ are generalized velocities and accelerations, respectively.

All terms in Eq. 2.12 are the factorization of the quantities contained in the equations of motion. $\mathbf{H}(\mathbf{q})$ and $\mathbf{g}$ are uniquely defined for a particular system. However, note that $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}$ is a purely quadratic function of the entries of $\dot{\mathbf{q}}$. This property makes $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ not uniquely defined. In fact, there are an infinite number of valid choices of $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ for a particular set of equations of motion, all of which give the same result for $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}$.

The application of the principle of energy conservation to Eq. 2.12 when $\boldsymbol{\tau} = 0$ shows that all valid factorizations of $\mathbf{C}$ must satisfy the property that $\dot{\mathbf{q}}^\top(\dot{\mathbf{H}}(\mathbf{q}, \dot{\mathbf{q}}) - 2\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}))\dot{\mathbf{q}} = 0$. Multiple control applications require the stronger property that the Coriolis matrix satisfy $\dot{\mathbf{H}}(\mathbf{q}, \dot{\mathbf{q}}) - 2\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ is skew-symmetric so that

$$\boldsymbol{\eta}^\top \left[\dot{\mathbf{H}}(\mathbf{q}, \dot{\mathbf{q}}) - 2\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\right] \boldsymbol{\eta} = 0 \quad \forall, \mathbf{q}, \dot{\mathbf{q}}, \boldsymbol{\eta} \in \mathbb{R}^n. \tag{2.13}$$

One particular choice of $\mathbf{C}$ that satisfies Eq. 2.13 is given in terms of the Christoffel symbols of the first kind. Christoffel symbols are tensor-like mathematical objects used to describe the geometry of a metric on a manifold. In the case of rigid-body dynamics, the mass matrix $\mathbf{H}$ provides a metric so that the Christoffel symbols (of the first kind) are defined as [19]

$$\Gamma_{ijk} = \frac{1}{2}\left[\frac{\partial H_{ij}}{\partial q_k} + \frac{\partial H_{ik}}{\partial q_j} - \frac{\partial H_{jk}}{\partial q_i}\right]. \tag{2.14}$$

Then, a choice for $\mathbf{C}$ in terms of the Christoffel symbols is expressed as [24]

$$\overline{\mathbf{C}}_{ij} = \sum_k \Gamma_{ijk}(\mathbf{q})\, \dot{q}_k. \tag{2.15}$$

For a system with mass matrix $\mathbf{H}(\mathbf{q})$ and $\overline{\mathbf{C}}(\mathbf{q}, \dot{\mathbf{q}})$ defined by Eq. 2.15, a matrix-

valued function $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ is called an admissible factorization if for all $\mathbf{q}, \dot{\mathbf{q}} \in \mathbb{R}^n$,

1. $\overline{\mathbf{C}}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} = \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}$; and
2. $\dot{\mathbf{H}}(\mathbf{q}, \dot{\mathbf{q}}) - 2\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ is skew symmetric.

Despite this definition of $\mathbf{C}$ being standard within the robotics community, it is difficult to use in practice. For systems that are more than a few DoFs, symbolically computing $\mathbf{H}$ and all its partial derivatives needed to compute $\overline{\mathbf{C}}$ is prohibitively computationally intensive. Therefore, novel methods to numerically compute $\overline{\mathbf{C}}$ and Γ_{ijk} in an efficient manner will improve the feasibility and runtime performance of some model-based controllers that use forms of $\mathbf{C}$ in their control laws.

2.5 Rigid-Body Dynamics Algorithms

The algorithms presented in subsequent chapters use techniques and frameworks similar to those employed by popular recursive algorithms of the lowest order to compute forward and inverse dynamics. The understanding of the mechanics of such algorithms will allow drawing parallels between the algorithms presented in this work and those extensively used in robotics applications.

2.5.1 Recursive Newton-Euler Algorithm

The RNEA is perhaps the most popular and widely used of the recursive dynamics algorithms. This algorithm exploits the recursive relation of spatial velocities and accelerations of a kinematic tree to solve the inverse dynamics problem, that is, solving for the internal forces of the system given its motion. The velocity of body i is related to the velocity of its predecessor by Eq. 2.2, its acceleration by

$$\mathbf{a}_i = {}^i\mathbf{X}_{p(i)}\mathbf{a}_{p(i)} + \mathbf{\Phi}_i\ddot{\mathbf{q}}_i + (\mathbf{v}_i\times)\mathbf{\Phi}_i\dot{\mathbf{q}}_i, \tag{2.16}$$

and its force by Eq. 2.6. The velocity, acceleration, and net force on each body in the tree can be computed using Eqs. 2.2, 2.16, and 2.6 recursively with a forward pass starting from the root towards the leaves (lines 3-7 in Algorithm 1). Then a backward pass from the leaves to the root computes the internal forces at each joint and updates the spatial force at each body to account for the reaction forces in its successor (lines 8-11). The resulting algorithm has complexity $O(N)$ to compute the torques required at each joint.

Algorithm 1 Recursive Newton-Euler Algorithm

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

1: $\mathbf{v}_0 = \mathbf{0}$
2: $\mathbf{a}_0 = -\mathbf{a}_g$
3: **for** $i = 1$ to N **do**
4: $\quad \mathbf{v}_i = {}^i\mathbf{X}_{p(i)}\mathbf{v}_{p(i)} + \mathbf{\Phi}_i\,\dot{\mathbf{q}}_i$
5: $\quad \mathbf{a}_i = {}^i\mathbf{X}_{p(i)}\mathbf{a}_{p(i)} + \mathbf{\Phi}_i\ddot{\mathbf{q}}_i + (\mathbf{v}_i\times)\mathbf{\Phi}_i\dot{\mathbf{q}}_i$
6: $\quad \mathbf{f}_i = \mathbf{I}_i\mathbf{a}_i + \mathbf{v}_i \times^* \mathbf{I}_i\mathbf{v}_i$
7: **end for**
8: **for** $j = N$ to 1 **do**
9: $\quad \boldsymbol{\tau}_j = \mathbf{\Phi}_j^\top\mathbf{f}_j$
10: $\quad \mathbf{f}_{p(j)} = \mathbf{f}_{p(j)} + {}^j\mathbf{X}_{p(j)}^\top\mathbf{f}_j$
11: **end for**
12: **return** $\boldsymbol{\tau}$, $\mathbf{f}$

2.5.2 Composite Rigid-Body Algorithm

The CRBA follows a similar strategy to that of the RNEA, but its purpose is to compute the mass matrix $\mathbf{H}$ numerically. For this, each entry of the matrix $\mathbf{H}$ is defined in terms of a composite inertial term $\mathbf{I}_i^C$. The physical interpretation of each composite inertia is the apparent inertia the subtree rooted at body i if all the joints in the tree remain fixed. So as not to obscure its physical meaning, it is defined in a

coordinate-free manner as

$$\mathbf{I}_i^C = \sum_{k \succeq i} \mathbf{I}_k. \tag{2.17}$$

Then for $i \preceq j$, $\mathbf{H}_{ij}$ is expressed as

$$\mathbf{H}_{ij} = \mathbf{\Phi}_i^\top \mathbf{I}_j^C \mathbf{\Phi}_j. \tag{2.18}$$

Note that $\mathbf{H}_{ij} = \mathbf{H}_{ji}$ due to the symmetry of $\mathbf{H}$. Physically, Eq. 2.18 indicates that $\mathbf{H}_{ij}$ gives the torque at joint i when joint j is accelerating with unit magnitude while all other joints remain fixed at rest. Then, $\mathbf{I}_j^C \mathbf{\Phi}_j$ represents the spatial force necessary to move joint j and all of its successors. Finally, $\mathbf{\Phi}_i^\top \mathbf{I}_j^C \mathbf{\Phi}_j$ represents the projection of the force onto the free modes of joint i.

The CRBA uses a similar structure to the RNEA, but in the forward pass, it only preliminarily computes the composite inertia terms for each body (lines 1-3 in Algorithm 2). Then in the backward pass, it computes $\mathbf{H}$ associated with the outermost body in the tree and updates the composite terms to account for the inertia of successors as it recursively goes down the tree to the root (lines 4-13).

2.5.3 Articulated-Body Algorithm

The Articulated-Body Algorithm was developed by Featherstone [7] as a constraint-propagation algorithm that can compute the joint accelerations from the known joint position, velocities, and applied torques. Once the joint accelerations are computed, the mechanism can be simulated using numerical integration methods. ABA is derived by considering the situation illustrated in Figure 2.3 where joint i interacts with the rest of the kinematic tree through an unknown force $\mathbf{f}_i$ computed as

$$\mathbf{f}_i = \mathbf{I}_i^A \mathbf{a}_i + \mathbf{p}_i^A, \tag{2.19}$$

Algorithm 2 Composite Rigid-Body Algorithm

Require: *model*

for $i = 1$ to N **do**
  $\mathbf{I}_i^C = \mathbf{I}_i$
end for
for $j = N$ to 1 **do**
  $\mathbf{F} = \mathbf{I}_j^C \, \mathbf{\Phi}_j$
  $i = j$
  while $i \neq 0$ **do**
    $\mathbf{H}_{ij} = \mathbf{\Phi}_i^\top \, \mathbf{F}$
    $\mathbf{H}_{ji} = \mathbf{H}_{ij}$
    $\mathbf{F} = {}^i\mathbf{X}_{p(i)}^\top \mathbf{F}$
    $i = p(i)$
  end while
  $\mathbf{I}_{p(j)}^C = \mathbf{I}_{p(j)}^C + {}^j\mathbf{X}_{p(j)}^\top \, \mathbf{I}_j^C \, {}^j\mathbf{X}_{p(j)}$
end for
return $\mathbf{H}$

where $\mathbf{I}_i^A$ is the articulated body inertia and $\mathbf{p}_i^A$ is an associated bias force. Note that the articulated inertia is the inertia felt at the base of a chain when all its joints are free to move. This meaning is in contrast to the composite inertia, which is the inertia felt if all of its joints are rigidly locked. Now, the spatial force acting at joint i and the acceleration of body i are related to the torque at joint i, $\boldsymbol{\tau}_i$, by

$$\boldsymbol{\tau}_i = \mathbf{\Phi}_i^\top \mathbf{f}_i = \mathbf{\Phi}_i^\top (\mathbf{I}_i^A (\mathbf{a}_{p(i)} + \mathbf{\Phi}_i \ddot{\mathbf{q}}_i + \dot{\mathbf{\Phi}}_i \dot{\mathbf{q}}_i) + \mathbf{p}_i^A). \tag{2.20}$$

Solving Eq. 2.20 for $\ddot{\mathbf{q}}_i$ yields

$$\ddot{\mathbf{q}}_i = \mathbf{\Phi}_i^\top \mathbf{f}_i = (\mathbf{\Phi}_i^\top \mathbf{I}_i^A \mathbf{\Phi}_i)^{-1} (\boldsymbol{\tau}_i - \mathbf{\Phi}_i^\top \mathbf{I}_i^A (\mathbf{a}_{p(i)} + \dot{\mathbf{\Phi}}_i \dot{\mathbf{q}}_i) - \mathbf{\Phi}_i^\top \mathbf{p}_i^A). \tag{2.21}$$

Equation 2.21 is a remarkable result. If the articulated body inertia and bias force are known, this equation allows the computation of the joint acceleration of each joint independently from the dynamics of the other joints. This fact is exploited by the ABA to recursively compute $\ddot{\mathbf{q}}_i$ and $\mathbf{a}_i$.

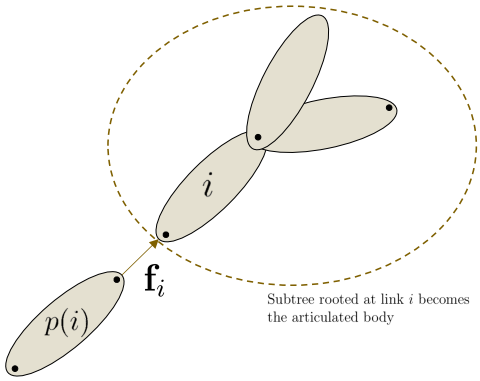


Figure 2.3. Depiction of articulated body.

The rest of the derivation of the ABA is discussed in detail in Chapter 4. For now, it is sufficient to say that its structure is similar to that of RNEA and CRBA. First, a forward sweep is carried out to calculate link velocities and initial values of articulated quantities $\mathbf{I}_i^A$ and $\mathbf{p}_i^A$ (lines 1-5 in Algorithm 3). Then a backward pass computes the articulated body inertia and bias force for each body (lines 6-15). Finally, a forward sweep computes the acceleration of each body in the tree (lines 16-20).

2.6 Conclusion

The spatial vector notation presented in this chapter not only provides notational simplicity for computing the equations of motion, but also a different viewpoint of the equations of motion that subsequent chapters exploit to develop new algorithms. The properties for an admissible factorization for $\mathbf{C}$ and one possible definition based on the Christoffel symbols are particularly important to take advantage of the structure of the open-chain rigid-body dynamics to develop the algorithms presented in Chapter 3.

The three algorithms reviewed in this chapter provide the recursive framework that will be later used in Chapter 3 and Chapter 4. While the RNEA and CRBA are

Algorithm 3 Articulated-Body Algorithm

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

1: **for** $i = 1$ to N **do**
2: $\quad \mathbf{I}_i^A = \mathbf{I}_i$
3: $\quad \mathbf{p}_i^A = \mathbf{v}_i \times^* \mathbf{I}_i \mathbf{v}_i$
4: $\quad \boldsymbol{\xi}_i = \mathbf{v}_i \times \boldsymbol{\Phi}_i \dot{\mathbf{q}}_i$
5: **end for**
6: **for** $i = N$ to 1 **do**
7: $\quad \boldsymbol{U}_i = \mathbf{I}_i^A \boldsymbol{\Phi}_i$
8: $\quad \boldsymbol{D}_i = (\boldsymbol{\Phi}_i^\top \boldsymbol{U}_i)^{-1}$
9: $\quad \mathbf{u}_i = \boldsymbol{\tau}_i - \boldsymbol{U}_i^\top \boldsymbol{\xi}_i - \boldsymbol{\Phi}_i^\top \mathbf{p}_i^A$
10: $\quad$ **if** $p(i) \neq 0$ **then**
11: $\quad\quad \mathbf{I}_{p(i)}^A = \mathbf{I}_{p(i)}^A + {}^i\mathbf{X}_{p(i)}^\top (\mathbf{I}_i^A - \boldsymbol{U}_i \boldsymbol{D}_i \boldsymbol{U}_i^\top) {}^i\mathbf{X}_{p(i)}$
12: $\quad\quad \mathbf{p}_{p(i)}^A = \mathbf{p}_{p(i)}^A + {}^i\mathbf{X}_{p(i)}^\top (\mathbf{p}_i^A + \mathbf{I}_i^A \boldsymbol{\xi}_i + \boldsymbol{U}_i \boldsymbol{D}_i \mathbf{u}_i)$
13: $\quad$ **end if**
14: $\quad i = p(i)$
15: **end for**
16: **for** $i = 1$ to N **do**
17: $\quad \mathbf{a}_i = {}^i\mathbf{X}_{p(i)} \mathbf{a}_{p(i)}$
18: $\quad \ddot{\mathbf{q}}_i = \boldsymbol{D}_i (\mathbf{u}_i - \boldsymbol{U}_i^\top \mathbf{a}_i)$
19: $\quad \mathbf{a}_i = \mathbf{a}_i + \boldsymbol{\Phi}_i \ddot{\mathbf{q}}_i + \boldsymbol{\xi}_i$
20: **end for**
21: **return** $\ddot{\mathbf{q}}$, $\mathbf{a}$

widely used for many robotics applications, they still fail to address some of the gaps presented in Chapter 1. For instance, they are unable to compute $\mathbf{C}^\top(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}$ used for sensorless contact detection or the partial derivatives employed in geometric or optimization-based control. The algorithms in the following chapters address some of these gaps by computing components of the equations of motion relevant to these problems.

CHAPTER 3

ALGORITHMS FOR THE COMPUTATION OF THE CORIOLIS MATRIX AND CHRISTOFFEL SYMBOLS

3.1 Introduction

This chapter outlines the derivation and implementation of algorithms to numerically compute the Coriolis matrix and Christoffel symbols. This chapter shows that the admissible factorization of $\mathbf{C}$ presented in Chapter 2 is related to the factorization of body-level velocity-product terms. This relationship allows for the recursive computation of $\mathbf{C}$ when implemented within a CRBA-like algorithm. A variation of the new algorithm allows numerical computation of $\overline{\mathbf{C}}$, which in turn allows for the numerical computation of the Christoffel symbols. The chapter discusses the validity of the new algorithms based on their agreement with expected results from the equations of motion. Finally, the chapter benchmarks the proposed algorithms to assess their feasibility in real-time control schemes.

3.2 Algorithm Derivation

This section provides a detailed derivation of new algorithms to recursively compute the Coriolis matrix and its corresponding Christoffel symbols. Furthermore, it presents pseudocode for both algorithms in a format ready for practical implementation. Note that the derivation of both algorithms is in coordinate-free form, but the pseudocodes are in body coordinates with the appropriate transformation matrices.

3.2.1 Numerical Methods for Computing $\mathbf{C}$

The spatial force on body i given by $\mathbf{f}_i = \mathbf{I}_i\mathbf{a}_i + (\mathbf{v}_i\times^*)\mathbf{I}_i\mathbf{v}_i$ can be factorized in terms of a new quantity $\mathbf{B}(\mathbf{v}_i, \mathbf{I}_i)$ that is bi-linear in $\mathbf{v}_i$ and $\mathbf{I}_i$. Naturally, a possible factorization is $\mathbf{B}(\mathbf{v}_i, \mathbf{I}_i) = (\mathbf{v}_i\times^*)\mathbf{I}_i$. In fact, there are an infinity of possible factorizations that satisfy 1) $\mathbf{B}(\mathbf{v}_i, \mathbf{I}_i)\mathbf{v}_i = (\mathbf{v}_i\times^*)\mathbf{I}_i\mathbf{v}_i$ and 2) $\mathbf{B}_i + \mathbf{B}_i^\top = (\mathbf{v}_i\times^*)\mathbf{I}_i - \mathbf{I}_i(\mathbf{v}_i\times)$.

One particular factorization proposed by Niemeyer and Slotine [20] that satisfies 1) and 2) is

$$\mathbf{B}(\mathbf{v}_i, \mathbf{I}_i) = \frac{1}{2}\left((\mathbf{v}_i\times^*)\mathbf{I}_i + (\mathbf{I}_i\mathbf{v}_i\bar{\times}^*) - \mathbf{I}_i(\mathbf{v}_i\times)\right). \tag{3.1}$$

Picking $\mathbf{B}$ via Eq. 3.1 leads to a $\mathbf{C}$ equal to $\overline{\mathbf{C}}$ from the Christoffel symbols.

The relationship between this new quantity and the Coriolis matrix becomes apparent under a similar structure to that of the CRBA. The structure for recursive computations is as follows. A forward sweep from root to leaves computes the velocities and accelerations of each body according to

$$\mathbf{v}_i = \sum_{j \preceq i} \mathbf{\Phi}_j\, \dot{\mathbf{q}}_j \tag{3.2}$$

$$\mathbf{a}_i = \sum_{j \preceq i} \mathbf{\Phi}_j\, \ddot{\mathbf{q}}_j + \dot{\mathbf{\Phi}}_j\, \dot{\mathbf{q}}_j, \tag{3.3}$$

where $\dot{\mathbf{\Phi}}_j = (\mathbf{v}_j\times)\mathbf{\Phi}_j + \mathring{\mathbf{\Phi}}_i$ and $\mathring{\mathbf{\Phi}}_j$ denotes the derivative due to the axes changing with respect to the local body coordinates. For revolute joints used in most applications, $\mathring{\mathbf{\Phi}}_j = \mathbf{0}$ because the axes are fixed in the local coordinates. The mathematical developments presented in this chapter are only valid for $\mathring{\mathbf{\Phi}}_j = \mathbf{0}$.

A backward sweep from leaves to root computes the inertial forces in each joint as

$$\boldsymbol{\tau}_i = \mathbf{\Phi}_i^\top \sum_{k \succeq i} \mathbf{I}_k\mathbf{a}_k + (\mathbf{v}_k\times^*)\mathbf{I}_k\mathbf{v}_k.$$

From Eqs. 2.6 and 3.1, the torque at each joint is expressed as

$$\boldsymbol{\tau}_i = \sum_{k \succeq i} \boldsymbol{\Phi}_i^\top \mathbf{I}_k \left(\sum_{j \preceq k} \boldsymbol{\Phi}_j \ddot{\mathbf{q}}_j + \dot{\boldsymbol{\Phi}}_j \dot{\mathbf{q}}_j \right) + \boldsymbol{\Phi}_i^\top \mathbf{B}_k \left(\sum_{j \preceq k} \boldsymbol{\Phi}_j \dot{\mathbf{q}}_j \right) \tag{3.4}$$

$$= \sum_{k \succeq i} \sum_{j \preceq k} \boldsymbol{\Phi}_i^\top \mathbf{I}_k \boldsymbol{\Phi}_j \ddot{\mathbf{q}}_j + \left(\boldsymbol{\Phi}_i^\top \mathbf{I}_k \dot{\boldsymbol{\Phi}}_j + \boldsymbol{\Phi}_i^\top \mathbf{B}_k \boldsymbol{\Phi}_j \right) \dot{\mathbf{q}}_j, \tag{3.5}$$

where $\mathbf{B}_k = \mathbf{B}_k(\mathbf{v}_k, \mathbf{I}_k)$.

To express $\boldsymbol{\tau}_i$ in a more useful form, it is necessary to resort to the following theorem.

Theorem 1 *Let body i with the following set*

$$S(i) = \{(k, j) \mid k \succeq i \text{ and } j \preceq k\}.$$

The set can also be represented as:

$$S(i) = \{(k, j) \mid j \sim i \text{ and } k \succeq \lceil ij \rceil\}.$$

With this in mind, $\boldsymbol{\tau}_i$ becomes

$$\boldsymbol{\tau}_i = \sum_{k \succeq i} \sum_{k \succeq \lceil ij \rceil} \boldsymbol{\Phi}_i^\top \mathbf{I}_k \boldsymbol{\Phi}_j \ddot{\mathbf{q}} + \left(\boldsymbol{\Phi}_i^\top \mathbf{I}_k \dot{\boldsymbol{\Phi}}_j + \boldsymbol{\Phi}_i^\top \mathbf{B}_k \boldsymbol{\Phi}_j \right) \dot{\mathbf{q}}_j \tag{3.6}$$

$$= \sum_{j \sim i} \boldsymbol{\Phi}_i^\top \mathbf{I}_{\lceil ij \rceil}^C \boldsymbol{\Phi}_j \ddot{\mathbf{q}}_j + \left(\boldsymbol{\Phi}_i^\top \mathbf{I}_{\lceil ij \rceil}^C \dot{\boldsymbol{\Phi}}_j + \boldsymbol{\Phi}_i^\top \mathbf{B}_{\lceil ij \rceil}^C \boldsymbol{\Phi}_j \right) \dot{\mathbf{q}}_j, \tag{3.7}$$

where the composite quantities are defined in a CRBA fashion as

$$\mathbf{I}^C_{\lceil ij \rceil} = \sum_{k \succeq \lceil ij \rceil} \mathbf{I}_k \tag{3.8}$$

$$\mathbf{B}^C_{\lceil ij \rceil} = \sum_{k \succeq \lceil ij \rceil} \mathbf{B}_k(\mathbf{v}_k, \mathbf{I}_k). \tag{3.9}$$

For the case when $i \preceq j$, $\mathbf{H}_{ij}$ is given by Eq. 2.18 and $\mathbf{C}_{ij}$ by

$$\mathbf{C}_{ij} = \mathbf{\Phi}_i^\top \left(\mathbf{I}_j^C \, \dot{\mathbf{\Phi}}_j + \mathbf{B}_j^C \, \mathbf{\Phi}_j\right) \tag{3.10}$$

so that

$$\mathbf{C}_{ji} = \mathbf{\Phi}_j^\top \, \mathbf{I}_j^C \, \dot{\mathbf{\Phi}}_i + \mathbf{\Phi}_j^\top \, \mathbf{B}_j^C \, \mathbf{\Phi}_i \tag{3.11}$$

$$(\mathbf{C}_{ji})^\top = \dot{\mathbf{\Phi}}_i^\top \, \mathbf{I}_j^C \, \mathbf{\Phi}_j + \mathbf{\Phi}_i^\top \, (\mathbf{B}_j^C)^\top \, \mathbf{\Phi}_j. \tag{3.12}$$

Given the sole dependance on composite terms, these equations are suitable for recursive computation. For this purpose, let three new quantities be express as

$$\mathbf{F}_{1,j} = \mathbf{I}_j^C \, \dot{\mathbf{\Phi}}_j + \mathbf{B}_j^C \, \mathbf{\Phi}_j \tag{3.13}$$

$$\mathbf{F}_{2,j} = \mathbf{I}_j^C \, \mathbf{\Phi}_j \tag{3.14}$$

$$\mathbf{F}_{3,j} = (\mathbf{B}_j^C)^\top \, \mathbf{\Phi}_j. \tag{3.15}$$

Then, each entry in $\mathbf{H}$ and $\mathbf{C}$ is computed by

$$\mathbf{H}_{ij} = \mathbf{\Phi}_i^\top \, \mathbf{F}_{2,j} \tag{3.16}$$

$$\mathbf{C}_{ij} = \mathbf{\Phi}_i^\top \, \mathbf{F}_{1,j} \tag{3.17}$$

$$\mathbf{C}_{ji} = \left(\dot{\mathbf{\Phi}}_i^\top \mathbf{F}_{2,j} + \mathbf{\Phi}_i^\top \mathbf{F}_{3,j}\right)^\top. \tag{3.18}$$

From these equations, the CRBA structure for computing $\mathbf{H}$ can naturally be extended to compute $\mathbf{C}$. It can be be further expanded to compute the time derivative of the mass matrix as follows. Starting from the definition of $\dot{\mathbf{H}}_{ij}$,

$$\dot{\mathbf{H}}_{ij} = \dot{\mathbf{\Phi}}_i^\top \mathbf{I}_j^C \mathbf{\Phi}_j + \mathbf{\Phi}_i^\top \left(\dot{\mathbf{I}}_j^C \mathbf{\Phi}_j + \mathbf{I}_j^C \dot{\mathbf{\Phi}}_j \right), \tag{3.19}$$

and noting that Eq. 3.1 satisfies

$$\mathbf{B}_j^C + (\mathbf{B}_j^C)^\top = \sum_{k \succeq j} (\mathbf{v}_k \times^*) \mathbf{I}_k - \mathbf{I}_k (\mathbf{v}_k \times) = \sum_{k \succeq j} \dot{\mathbf{I}}_j = \dot{\mathbf{I}}_j^C, \tag{3.20}$$

it follows that

$$\dot{\mathbf{H}}_{ij} = \dot{\mathbf{\Phi}}_i^\top \mathbf{I}_j^C \mathbf{\Phi}_j + \mathbf{\Phi}_i^\top \left[\left(\mathbf{B}_j^C + (\mathbf{B}_j^C)^\top \right) \mathbf{\Phi}_j + \mathbf{I}_j^C \dot{\mathbf{\Phi}}_j \right] \tag{3.21}$$

$$= \mathbf{\Phi}_i^\top (\mathbf{F}_1 + \mathbf{F}_3) + \dot{\mathbf{\Phi}}_i^\top \mathbf{F}_2. \tag{3.22}$$

Noting that $\mathbf{C}_{ij} + (\mathbf{C}_{ji})^\top = \left[\mathbf{C} + \mathbf{C}^\top \right]_{ij}$ and using Eqs. 3.17, 3.18, and 3.22, it follows that

$$\mathbf{C}_{ij} + (\mathbf{C}_{ji})^\top = \mathbf{\Phi}_i^\top \mathbf{F}_1 + \dot{\mathbf{\Phi}}_i^\top \mathbf{F}_2 + \mathbf{\Phi}_i^\top \mathbf{F}_3, \tag{3.23}$$

so

$$\left[\mathbf{C} + \mathbf{C}^\top \right]_{ij} = \dot{\mathbf{H}}_{ij}. \tag{3.24}$$

Therefore,

$$\dot{\mathbf{H}} = \mathbf{C} + \mathbf{C}^\top. \tag{3.25}$$

Algorithm 4 shows the recursive computation of these equations expressed in body coordinates. The forward sweep (lines 2-7) computes the velocity propagation throughout the tree as well as the initial composite terms $\mathbf{I}_i^C$ and $\mathbf{B}_i^C$. The backward sweep (lines 2-23) computes the entries of the Coriolis and mass matrices. The quan-

tities in lines 9-11 have no physical meaning, but are convenient factorizations that increase efficiency when computing $\mathbf{H}_{ij}$ and $\mathbf{C}_{ij}$. Lines 22 and 23 are the propagation of the composite terms towards the root of the tree. The while loop (lines 14-21) computes the entries for $\mathbf{H}$, $\dot{\mathbf{H}}$, and $\mathbf{C}$ associated with body i and propagates the computation down to all its predecessors.

Note that the composite terms $\mathbf{I}_i^C$ and $\mathbf{B}_i^C$ are computed with a complexity of $O(N)$. Because $\mathbf{H}$ and $\mathbf{C}$ depend only on $\mathbf{I}_i^C$ and $\mathbf{B}_i^C$, they can be computed recursively in $O(Nd)$ so that the overall complexity of Algorithm 4 is $O(Nd)$.

Algorithm 4 Coriolis Matrix Algorithm

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

1: $\mathbf{v}_0 = \mathbf{0}$
2: **for** $i = 1$ to N **do**
3: $\quad \mathbf{v}_i = {}^i\mathbf{X}_{p(i)}\,\mathbf{v}_{p(i)} + \mathbf{\Phi}_i\,\dot{\mathbf{q}}_i$
4: $\quad \dot{\mathbf{\Phi}}_i = (\mathbf{v}_i\times)\mathbf{\Phi}_i$
5: $\quad \mathbf{I}_i^C = \mathbf{I}_i$
6: $\quad \mathbf{B}_i^C = \frac{1}{2}[(\mathbf{v}_i\times^*)\mathbf{I}_i + (\mathbf{I}_i\mathbf{v}_i)\,\overline{\times}^* - \mathbf{I}_i(\mathbf{v}_i\times)]$
7: **end for**
8: **for** $j = N$ to 1 **do**
9: $\quad \mathbf{F}_1 = \mathbf{I}_j^C\,\dot{\mathbf{\Phi}}_j + \mathbf{B}_j^C\,\mathbf{\Phi}_j$
10: $\quad \mathbf{F}_2 = \mathbf{I}_j^C\,\mathbf{\Phi}_j$
11: $\quad \mathbf{F}_3 = (\mathbf{B}_j^C)^\top\mathbf{\Phi}_j$
12: $\quad \mathbf{C}_{jj} = \mathbf{\Phi}_j^\top\,\mathbf{F}_1$
13: $\quad i = j$
14: $\quad$ **while** $i > 0$ **do**
15: $\quad\quad \mathbf{F}_1 = {}^i\mathbf{X}_{p(i)}^\top\,\mathbf{F}_1;\ \mathbf{F}_2 = {}^i\mathbf{X}_{p(i)}^\top\,\mathbf{F}_2;\ \mathbf{F}_3 = {}^i\mathbf{X}_{p(i)}^\top\,\mathbf{F}_3;$
16: $\quad\quad \mathbf{C}_{ij} = \mathbf{\Phi}_i^\top\,\mathbf{F}_1$
17: $\quad\quad \mathbf{C}_{ji} = \left(\dot{\mathbf{\Phi}}_i^\top\,\mathbf{F}_2 + \mathbf{\Phi}_i^\top\mathbf{F}_3\right)^\top$
18: $\quad\quad \mathbf{H}_{ij} = (\mathbf{H}_{ji})^\top = \mathbf{\Phi}_i^\top\,\mathbf{F}_2$
19: $\quad\quad \dot{\mathbf{H}}_{ij} = (\dot{\mathbf{H}}_{ji})^\top = \dot{\mathbf{\Phi}}_i^\top\,\mathbf{F}_2 + \mathbf{\Phi}_i^\top(\mathbf{F}_1 + \mathbf{F}_3)$
20: $\quad\quad i = p(i)$
21: $\quad$ **end while**
22: $\quad \mathbf{I}_{p(j)}^C = \mathbf{I}_{p(j)}^C + {}^j\mathbf{X}_{p(j)}^\top\,\mathbf{I}_j^C\,{}^j\mathbf{X}_{p(j)}$
23: $\quad \mathbf{B}_{p(j)}^C = \mathbf{B}_{p(j)}^C + {}^j\mathbf{X}_{p(j)}^\top\,\mathbf{B}_j^C\,{}^j\mathbf{X}_{p(j)}$
24: **end for**
25: **return** $\mathbf{H}$, $\dot{\mathbf{H}}$, $\mathbf{C}$

3.2.2 Numerical Methods for Computing Christoffel Symbols

The $\mathbf{C}$ matrix resulting from Algorithm 4 is an admissible factorization because it is derived from Niemeyer and Slotine's definition Eq. 3.1. Consequently, $\mathbf{C}$ may be expressed in terms of the Christoffel symbols as

$$\mathbf{C}_{ij} = \sum_k \Gamma_{ijk}(\mathbf{q})\, \dot{q}_k. \tag{3.26}$$

Taking the derivative of Eq. 3.26 with respect to $\dot{q}_k$ and rearranging, the Christoffel symbols are expressed in terms of $\mathbf{C}_{ij}$,

$$\Gamma_{ijk} = \frac{\partial \mathbf{C}_{ij}}{\partial \dot{q}_k}. \tag{3.27}$$

Substituting Eq. 3.10 into Eq. 3.27, the Christoffel symbols become

$$\Gamma_{ijk} = \mathbf{\Phi}_i^\top \mathbf{I}^C_{\lceil ij \rceil} \frac{\partial \dot{\mathbf{\Phi}}_j}{\partial \dot{q}_k} + \mathbf{\Phi}_i^\top \frac{\partial \mathbf{B}^C_{\lceil ij \rceil}}{\partial \dot{q}_k} \mathbf{\Phi}_j. \tag{3.28}$$

Consider the first term in Eq. 3.28. Physically, this term represents that $\dot{\mathbf{\Phi}}_j$ will only change as a function of $\dot{q}_k$ if body j moves with changes in q_k. That is, body j is part of the kinematic chain from body k to the leaves (Fig. 3.1). Therefore, it is mathematically expressed as

$$\frac{\partial \dot{\mathbf{\Phi}}_j}{\partial \dot{q}_k} = \begin{cases} \mathbf{\Phi}_k \times \mathbf{\Phi}_j & \text{if } j \succeq k \\ \mathbf{0} & \text{o/w.} \end{cases}. \tag{3.29}$$

Now consider the second term in Eq. 3.27. By substituting from Eq. 3.3 and taking into account the bi-linearity of the terms in $\mathbf{B}_{\lceil ij \rceil}$,

$$\frac{\partial \mathbf{B}^C_{\lceil ij \rceil}}{\partial \dot{q}_k} = \sum_{\ell \succeq \lceil ij \rceil} \mathbf{B}\left(\frac{\partial \mathbf{v}_\ell}{\partial \dot{q}_k}, \mathbf{I}_\ell \right), \tag{3.30}$$

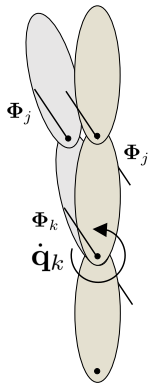


Figure 3.1. Depiction of the motion of spatial joint axes due to the joint velocities of their ancestors.

where

$$\frac{\partial \mathbf{v}_j}{\partial \dot{q}_k} = \begin{cases} \boldsymbol{\Phi}_k & \text{if } j \succeq k \\ \mathbf{0} & \text{o/w.} \end{cases} . \tag{3.31}$$

Then, Eq. 3.30 becomes

$$\frac{\partial \mathbf{B}^C_{\lceil ij \rceil}}{\partial \dot{q}_k} = \sum_{\ell \in \{\ell \succeq \lceil ij \rceil \text{ and } \ell \succeq k\}} \mathbf{B}(\boldsymbol{\Phi}_k, \mathbf{I}_\ell) = \mathbf{B}(\boldsymbol{\Phi}_k, \mathbf{I}^C_{\lceil ijk \rceil}). \tag{3.32}$$

Because of the symmetry in $\mathbf{H}$, the Christoffel symbols also exhibit symmetry in the last two indices $\Gamma_{ijk} = \Gamma_{ikj}$. For $j \succeq k$, it then follows that

$$\Gamma_{ijk} = \Gamma_{ikj} = \boldsymbol{\Phi}_i^\top \mathbf{B}(\boldsymbol{\Phi}_k, \mathbf{I}^C_{\lceil ijk \rceil}) \boldsymbol{\Phi}_j. \tag{3.33}$$

For the case when $i \preceq j \preceq k$, the possible index permutations are

$$\Gamma_{ijk} = \Gamma_{ikj} = \boldsymbol{\Phi}_i^\top \mathbf{B}(\boldsymbol{\Phi}_k, \mathbf{I}^C_k) \boldsymbol{\Phi}_j \tag{3.34}$$

$$\Gamma_{jik} = \Gamma_{jki} = \boldsymbol{\Phi}_j^\top \mathbf{B}(\boldsymbol{\Phi}_k, \mathbf{I}^C_k) \boldsymbol{\Phi}_i \tag{3.35}$$

$$\Gamma_{kij} = \Gamma_{kji} = \boldsymbol{\Phi}_k^\top \mathbf{B}(\boldsymbol{\Phi}_j, \mathbf{I}^C_k) \boldsymbol{\Phi}_i. \tag{3.36}$$

Notice that the arguments $\boldsymbol{\Phi}$ and $\mathbf{I}^C$ in $\mathbf{B}(\boldsymbol{\Phi}, \mathbf{I}^C)$ have the same index in Eqs. 3.34 and 3.35 but not in Eq. 3.36. To allow for efficient recursive computation, it is necessary that all three cases use the same $\mathbf{B}(\boldsymbol{\Phi}_k, \mathbf{I}_k^C)$ term. Therefore, expanding the last of these identities using the definition of $\mathbf{B}(\boldsymbol{\Phi}_j, \mathbf{I}_k^C)$, it becomes

$$\Gamma_{kij} = \frac{1}{2}\boldsymbol{\Phi}_k^\top \left((\boldsymbol{\Phi}_j \times^*) \mathbf{I}_k^C \boldsymbol{\Phi}_i + (\boldsymbol{\Phi}_i \times^*) \mathbf{I}_k^C \boldsymbol{\Phi}_j - \mathbf{I}_k^C (\boldsymbol{\Phi}_j \times) \boldsymbol{\Phi}_i \right). \tag{3.37}$$

Applying the properties that

$$\mathbf{v}_1^\top (\mathbf{v}_2 \times^*) \boldsymbol{F} = -\boldsymbol{F}^\top (\mathbf{v}_2 \times) \mathbf{v}_1 = \boldsymbol{F}^\top (\mathbf{v}_1 \times) \mathbf{v}_2 \tag{3.38}$$

and

$$(\mathbf{f} \,\bar{\times}^*) \mathbf{v} = (\mathbf{v} \times^*) \mathbf{f} \tag{3.39}$$

to each of the terms in Eq. 3.37, it becomes

$$\Gamma_{kij} = \frac{1}{2}\boldsymbol{\Phi}_i^\top \left[\mathbf{I}_k^C (\boldsymbol{\Phi}_k \times) - (\boldsymbol{\Phi}_k \times^*) \mathbf{I}_k^C + (\mathbf{I}_k^C \boldsymbol{\Phi}_k \,\bar{\times}^*) \right] \boldsymbol{\Phi}_j \tag{3.40}$$

$$= \boldsymbol{\Phi}_i^\top \left[(\mathbf{I}_k^C \boldsymbol{\Phi}_k \,\bar{\times}^*) - \mathbf{B}(\boldsymbol{\Phi}_k, \mathbf{I}_k^C) \right] \boldsymbol{\Phi}_j. \tag{3.41}$$

With all entries of the Christoffel symbols expressed in terms of $\mathbf{B}(\boldsymbol{\Phi}_k, \mathbf{I}_k^C)$, they can be recursively computed through Algorithm 5. The forward sweep (lines 2-4) initiates the composite inertia quantity for each body. Because the Christoffel symbols are a 3D tensor-like quantity, the backward sweep (lines 5-26) is more complex than the rest of the algorithms in this work. Lines 6 and 7 compute $\mathbf{B}$ and $\mathbf{D}$, a convenient intermediate quantity introduced for computing Γ_{kij}. The nested while loops (lines 2-20) compute all the entries of Γ. Note that there are two while loops to cover all permutations of indices j and i for a given k. Lines 21 and 22 propagate $\mathbf{B}$ and $\mathbf{D}$ down the tree as the indices change, and line 25 updates the composite inertia term

and velocity product factorizations as the algorithm sweeps towards the root of the tree. Because each of the nested while loops has a computational cost proportional to its depth, the overall efficiency of the algorithm is $O(Nd^2)$

Algorithm 5 Christoffel Symbols Algorithm

Require: $\mathbf{q}, model$

$\mathbf{v}_0 = \mathbf{0}$
for $i = 1$ to N **do**
  $\mathbf{I}_i^C = \mathbf{I}_i$
end for
for $k = N$ to 1 **do**
  $\mathbf{B} = \frac{1}{2}[(\mathbf{\Phi}_k \times^*)\mathbf{I}_k^C + (\mathbf{I}_k^C \mathbf{\Phi}_k)\bar{\times}^* - \mathbf{I}_k^C(\mathbf{\Phi}_k \times)]$
  $\mathbf{D} = (\mathbf{I}_k^C \mathbf{\Phi}_k \bar{\times}^*) - \mathbf{B}$
  $j = k$
  while $j > 0$ **do**
    $\mathbf{F}_1 = \mathbf{B}\,\mathbf{\Phi}_j$
    $\mathbf{F}_2 = \mathbf{B}^\top\,\mathbf{\Phi}_j$
    $\mathbf{F}_3 = \mathbf{D}\,\mathbf{\Phi}_j$
    $i = j$
    while $i > 0$ **do**
      $\Gamma_{ijk} = \Gamma_{ikj} = \mathbf{\Phi}_1^\top \mathbf{F}_1$
      $\Gamma_{jik} = \Gamma_{jki} = \mathbf{\Phi}_1^\top \mathbf{F}_2$
      $\Gamma_{kij} = \Gamma_{kji} = \mathbf{\Phi}_1^\top \mathbf{F}_3$
      $\mathbf{F}_1 = {}^i\mathbf{X}_{p(i)}^\top \mathbf{F}_1; \mathbf{F}_2 = {}^i\mathbf{X}_{p(i)}^\top \mathbf{F}_2; \mathbf{F}_3 = {}^i\mathbf{X}_{p(i)}^\top \mathbf{F}_3;$
      $i = p(i)$
    end while
    $\mathbf{B} = {}^j\mathbf{X}_{p(j)}^\top\,\mathbf{B}\,{}^j\mathbf{X}_{p(j)}$
    $\mathbf{D} = {}^j\mathbf{X}_{p(j)}^\top\,\mathbf{D}\,{}^j\mathbf{X}_{p(j)}$
    $j = p(j)$
  end while
  $\mathbf{I}_{p(k)}^C = \mathbf{I}_{p(k)}^C + {}^k\mathbf{X}_{p(k)}^\top\,\mathbf{I}_k^C\,{}^k\mathbf{X}_{p(k)}$
end for
return $\mathbf{\Gamma}$

3.3 Results

Algorithms 4 and 5 were implemented in C/C++ using the Rigid-Body Dynamics Library [11]. This publicly available library exploits the sparsity patterns in quantities in the equations of motion, such as the spatial inertia tensor, to minimize computational cost and runtime of dynamics operations. This section presents the results that verified the correctness and performance of the algorithms developed in this chapter.

3.3.1 Validation

Three methods compared Algorithms 4 and 5 against known algorithms, such as the RNEA, to verify their accuracy.

Method 1: From the canonical equations of motion in Eq. 2.12, it is clear that when the gravity force is 0 and $\ddot{\mathbf{q}} = 0$, the torque at each joint is equal to the product of $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ and $\dot{\mathbf{q}}$. Therefore, if Algorithm 4 is correct, for a system under those conditions, the torque $\boldsymbol{\tau}$ computed through the RNEA and the computed $\mathbf{C}$ satisfy $\mathbf{C}\dot{\mathbf{q}} - \boldsymbol{\tau} = 0$.

Method 2: After $\mathbf{C}$ is verified with Method 1, the computed $\dot{\mathbf{H}}$ is verified to satisfy $\dot{\mathbf{H}} - (\mathbf{C} + \mathbf{C}^\top) = 0$.

Method 3: Because of Eq. 3.26, with a correct $\overline{\mathbf{C}}$, the Christoffel symbols are verified if they satisfy $\sum_k \Gamma_{ijk}(\mathbf{q})\, \dot{q}_k - \overline{\mathbf{C}}_{ij} = 0$.

The three methods were tested using serial kinematic chains of 5, 10, and 15 bodies. The errors for all methods in all three scenarios were negligible, which verifies the correctness of Algorithms 4 and 5 (Table 3.1).

3.3.2 Performance

For serial kinematic chains, where the depth equals the number of bodies, the computational cost for the Coriolis matrix scales as $O(N^2)$, while that for the Christoffel

TABLE 3.1

VALIDITY CHECKS FOR SERIAL KINEMATIC CHAINS OF 5, 10, AND 15 BODIES (MAXIMUM ERROR)

	$N = 5$	$N = 10$	$N = 15$
Method 1: $\mathbf{C}\dot{\mathbf{q}} - \boldsymbol{\tau}$ (RNEA)	5.7×10^{-14}	7.3×10^{-12}	2.9×10^{-11}
Method 2: $\dot{\mathbf{H}} - (\mathbf{C} + \mathbf{C}^\top)$	3.6×10^{-15}	5.7×10^{-14}	2.3×10^{-13}
Method 3: $\sum_k [\Gamma_{1,1,k}, \dot{q}_k] - \overline{\mathbf{C}}_{1,1}$	1.1×10^{-13}	7.3×10^{-12}	1.0×10^{-12}

symbols grows as $O(N^3)$ (Fig. 3.2). Note that these trends refer to the behavior when the number of bodies is large because when N is low, the computational cost is dominated by computations that do not scale with N, such as $\mathbf{B}^C$. For a branching mechanism where $d < N$, the expected behavior is for the computational cost to grow more slowly than in the serial case where $d = N$ because $O(Nd) < O(N^2)$ and $O(Nd^2) < O(N^3)$. This behavior was confirmed by testing both algorithms on a branching mechanism with a branching factor of 2. In this case, the time to compute the Coriolis matrix grew roughly as $O(N)$ and that of the Christoffel symbols empirically scaled as $O(N^{1.25})$ (Fig. 3.3). This branching effect is present in common robotic systems such as quadrupeds and bipeds. In particular, quadrupeds have a higher branching factor than bipeds, so, as expected, the computational cost for the Coriolis matrix and Christoffel symbols is higher in bipeds than in quadrupeds (Fig. 3.4).

In terms of absolute time, Algorithms 4 and 5 proved to be fast enough for online control loops that run thousands of times per second. For rigid-body systems of 20 DoFs, the algorithms take approximately 20 μs to compute the Coriolis matrix and up

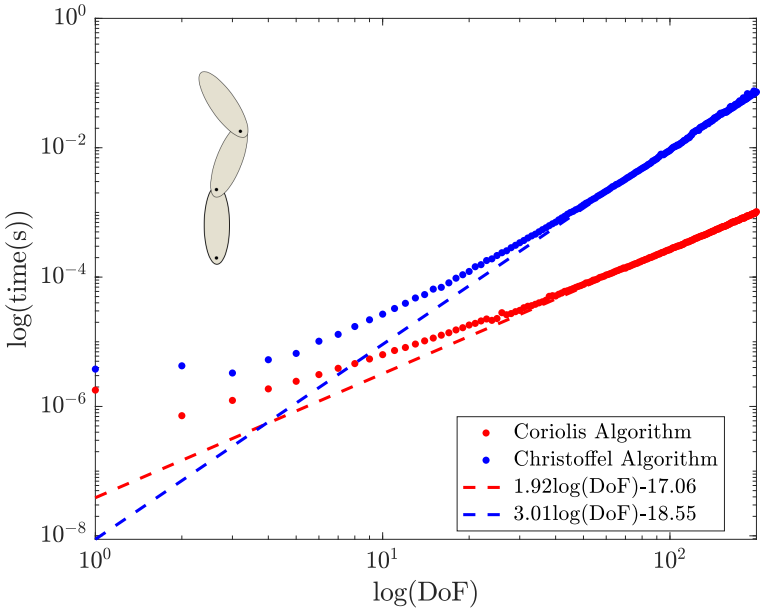


Figure 3.2. Performance for serial kinematic chains.

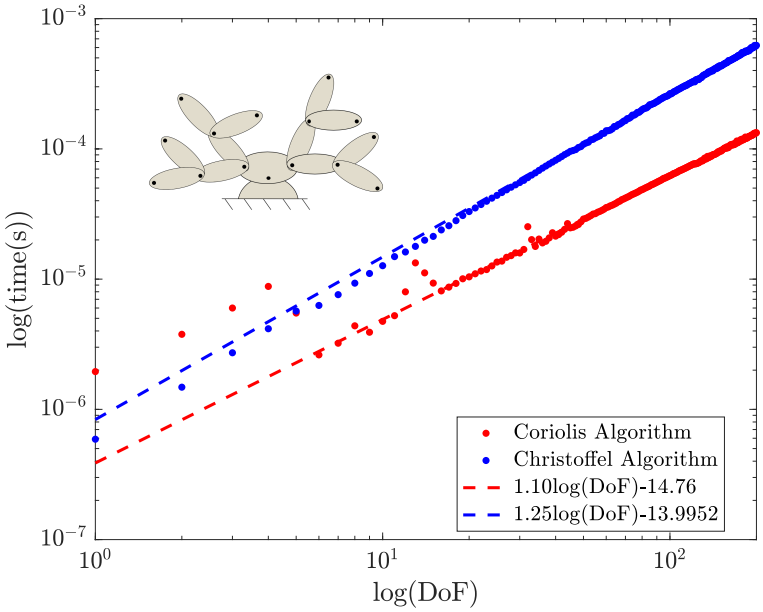


Figure 3.3. Performance for branching mechanism (branching factor of 2).

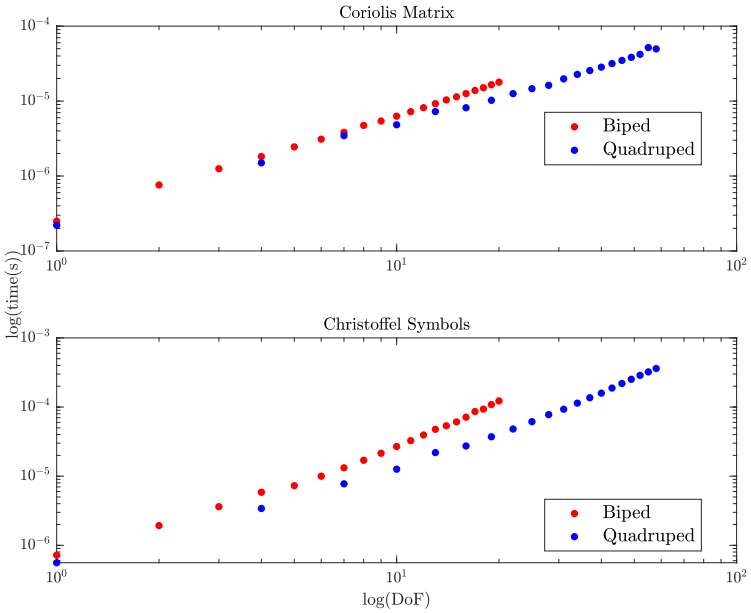


Figure 3.4. Computation cost for bipeds and quadrupeds.

to 120 μs to calculate the Christoffel symbols (Table 3.2). The longer computation times for the Christoffel symbols are associated with trees that have little to no branching. In contrast, for kinematic trees with higher branching factors, all Γ_{ijk} can be computed in as little as 33 μs.

These results are particularly promising for the Christoffel symbols algorithm. Commonly, the first step to symbolically compute Γ_{ijk} is to calculate the mass matrix symbolically with an algorithm such as CRBA or RNEA. Then it is necessary to do a large number of partial derivatives of $\mathbf{H}$ to find the Christoffel symbols. This process becomes prohibitively complex for systems with more than just a few bodies. To show the capability of Algorithm 5, it was implemented using symbolic variables in MATLAB. The algorithm calculates Γ_{ijk} in a time comparable to that of the symbolic implementation of the CRBA, which is only the first step of the alternative approach to get Γ_{ijk} (Table 3.3). This performance is even more remarkable when considering the results of Table 3.2. In the time it takes to solve for $\mathbf{H}$ symbolically, Algorithm

TABLE 3.2

COMPUTATION TIME OF CORIOLIS MATRIX AND CHRISTOFFEL SYMBOLS FOR COMMON RIGID-BODY SYSTEMS.

	$\mathbf{C}(\mu s)$	$\Gamma_{ijk}(\mu s)$
Serial Chain (20 DoF)	18	122
Branching Mechanism (20 DoF)	10	33
Biped (20 DoF)	18	122
Quadruped (19 DoF)	10	37

5 can numerically compute Γ_{ijk} more than 3,000 times.

3.4 Conclusion

This section developed computationally efficient algorithms to numerically solve for the Coriolis matrix and its associated Christoffel symbols. The algorithms exploit the bi-linearity in $\mathbf{v}$ and $\mathbf{I}$ of a suitable body-level factorization, $\mathbf{B}(\mathbf{v}, \mathbf{I})$, of velocity product terms when computing the system-level Coriolis terms. In Algorithm 4, the newfound expressions for each entry in the Coriolis matrix are computed recursively in terms of composite quantities $\mathbf{I}_i^C$ and $\mathbf{B}_i^C$. Taking the derivative for the expression of $\mathbf{C}_{ij}$ with respect to $\dot{q}_k$ allowed new expressions for the Christoffel symbols in terms of $\mathbf{B}(\mathbf{v}_i, \mathbf{I}_i)$ that are computed recursively in Algorithm 5. Both algorithms proved to be fast and of the lowest order possible. Their complexity grew as $O(Nd)$ and $O(Nd^2)$, respectively, while their computation time was on the order of μs. For a 20-DoF biped, it took the algorithms only 17 μs to compute $\mathbf{C}$ and 123 μs for Γ. The effects of branching played a substantial role in decreasing runtime, particularly

TABLE 3.3

RUNTIME OF ALGORITHMS USING SYMBOLIC VARIABLES FOR SERIAL CHAINS WITH 5,10, AND 15 BODIES.

	$N = 5$ (s)	$N = 10$ (s)	$N = 15$ (s)
H	0.06	0.18	0.36
Γ_{ijk}	0.25	1.11	2.13

for the Christoffel symbols algorithm. For a 19-DoF quadruped, the computational time for Algorithm 5 reduced to 38 μs. Given the efficiency, scalability, and speed of these algorithms, they are viable for implementation in real-time control loops as well as other dynamics applications.

CHAPTER 4

MODIFIED ARTICULATED-BODY ALGORITHM

4.1 Introduction

This chapter presents the derivation of a modified Articulated-Body Algorithm (MABA) that accurately accounts for the inertia of motor rotors. The chapter first introduces the necessary formalisms for the derivation: the Gauss Principle of Least Constraint and the model used to describe geared electric motors. The chapter shows that by reformulating the dynamics of the system as an optimization problem through the Gauss Principle of Least Constraint, it is possible to adjust the quantities of the ABA to account for rotor inertias from first principles. The chapter discusses the validity of the proposed algorithm by comparing its results with those of the ABA and other common methods to approximate the inertial effects of actuators. The chapter also discusses the performance of the modified ABA in terms of complexity and runtime.

4.2 Mathematical Formalisms and Modeling Conventions

4.2.1 Gauss Principle of Least Constraint

The Gauss Principle of Least Constraint is a formulation of mechanics used to solve for the admissible accelerations of constrained mechanical systems [31]. The principle is described in terms of a least-squares optimization problem. In particular, it states that the accelerations of a system of rigid bodies deviate from the unconstrained accelerations in a least-squares sense. Mathematically, the principle specifies

that [6]

$$\begin{aligned} \underset{\{\mathbf{a}_i\}}{\text{argmin}} \quad & \sum_i (\mathbf{a}_i - \mathbf{a}_i^{uc})^\top \mathbf{I}_i (\mathbf{a}_i - \mathbf{a}_i^{uc}) \\ \text{subject to } & \{\mathbf{a}_i\} \text{ satisfy constraints,} \end{aligned} \tag{4.1}$$

where $\mathbf{a}_i^{uc}$ is the unconstrained acceleration of body i. While this principle originally considered only massive point particles, it is also valid for rigid-body systems that are geometrically constrained by considering the inertia tensor as opposed to just the mass of each body.

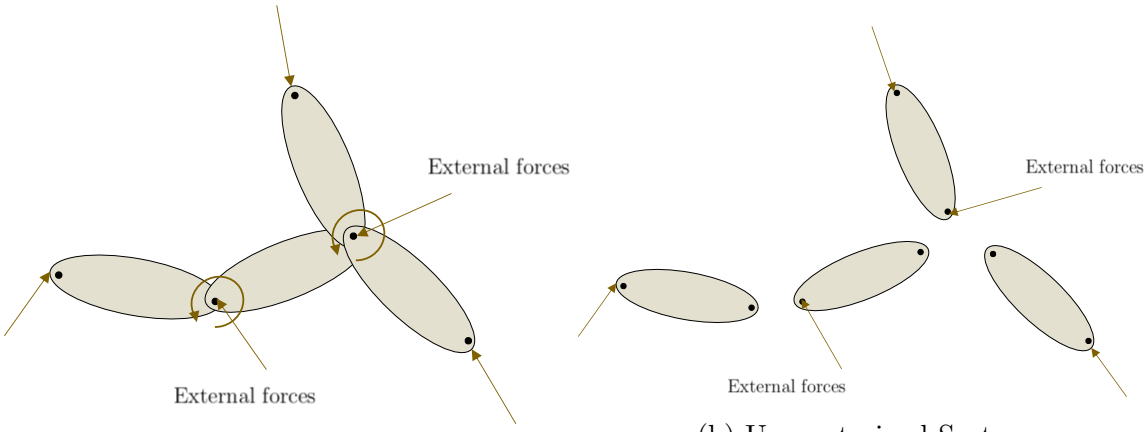
An unconstrained system (Figure 4.1b) is one in which the bodies are not geometrically constrained so that the dynamics of each body are entirely independent of the dynamics of the rest of the bodies in the system. Then, the unconstrained accelerations are the accelerations caused by external forces acting on individual particles, ignoring all forces from constrained interactions between bodies.

A constrained system (Figure 4.1a) is one in which bodies are geometrically constrained so that the dynamics of each body are coupled with those of other bodies. Because bodies are constrained, the overall dynamics of the system limit the possible values for each $\mathbf{a}_i$. The true accelerations are calculated by solving the optimization problem in the Gauss Principle of Least Constraint.

Overall, the Gauss Principle of Least Constraint provides an intuitive mathematical framework to solve the forward dynamics problem for complex systems that are common in robotics applications.

4.2.2 Joint Modeling

Previous attempts to use the Gauss Principle of Least Constraint to model kinematic chains have mostly ignored the inertial effects of rotors and gears [29]. It is necessary to include these elements in the model of each joint to account for their inertial effects. This work considers joints to be as shown in Figure 4.2. The dis-



(a) Constrained System

(b) Unconstrained System

Figure 4.1. Visual comparison between constrained and unconstrained systems.

assembled joint in this figure is composed of body $p(i)$, rotor R_i, gearbox G_i, and body i so that these four elements form a kinematic loop. Notice that the model for the gearing of the joint is one body linked to the rotor, so all internal forces and friction associated with gear dynamics are ignored. Similarly, the model assumes that the gearbox is massless and without any inertia. For most robotics systems, this assumption is valid because motor rotors are often several times heavier than their gears.

A particularly important consequence of the joint model used in this work is that the term describing the free modes of motion of rotor R_i must be equal to the term describing the free modes of motion of body i because the rotor is rotating along the same axis as the body. While the proposed joint model does not account for all interactions inside the joint, it provides a more accurate representation than what dynamics algorithms in the literature commonly use. Furthermore, the mathematical methods presented in this chapter can easily expand to include other terms such as gearing inertia or internal friction to obtain an even more accurate account of joint inertial effects.

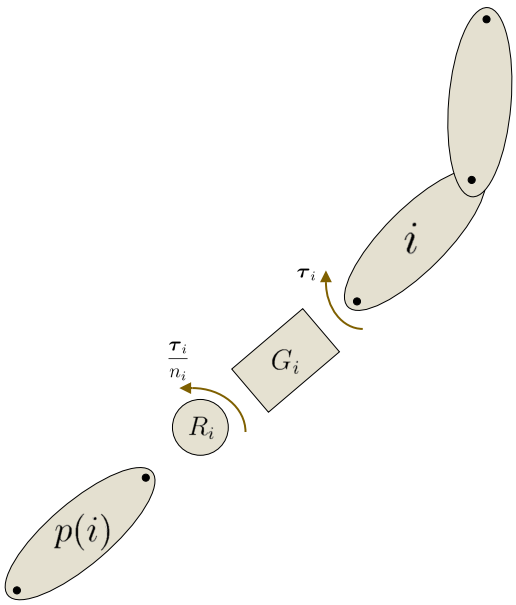


Figure 4.2. Dissasembled joint between body $p(i)$ and body i

4.3 Algorithm Derivation

The general strategy to derive an ABA that accounts for rotor inertias from first principles is as follows. First, determine expressions for the constrained and unconstrained accelerations of body i, as well as rotor R_i. Then solve the optimization problem for the Gauss Principle of Least Constraint for a single body subject to the appropriate constraints. Then find expressions for the articulated inertia and bias forces based on the solution of the optimization problem and use the new expression within the structure of the ABA to solve the forward dynamics problem. Similar to Chapter 3, the derivation in this chapter uses coordinate-free expressions, but the algorithm shows the appropriate transformation matrices.

4.3.1 Identification of Optimization Problem

Consider the unconstrained case for the joint in Figure 4.2. The only two forces present in the unconstrained system are a torque and its equal and opposite reaction.

Then, according to the ABA definition of force in terms of articulated quantities from Eq. 2.19, the unconstrained acceleration of body i is

$$\mathbf{a}_i^{uc} = (\mathbf{I}_i^A)^{-1}(\mathbf{\Phi}_i \boldsymbol{\tau}_i - \mathbf{p}_i^A), \tag{4.2}$$

where $\mathbf{p}_i^A$ is an associated bias force.

Similarly, the rotor feels the same forces, but its unconstrained acceleration is expressed a bit differently. In particular, because the torque felt by body i is the output torque of the motor multiplied by the gear ratio of the gearbox, n_i, the torque felt by the motor is $\boldsymbol{\tau}_i$ divided by n_i. Furthermore, because the inertia the rotor feels is only its own, the unconstrained acceleration of R_i is given by

$$\mathbf{a}_{R_i}^{uc} = \mathbf{I}_{R_i}^{-1}(\mathbf{\Phi}_i \frac{\boldsymbol{\tau}_i}{n_i} - \mathbf{p}_{R_i}), \tag{4.3}$$

where $\mathbf{p}_{R_i} = \mathbf{v}_{R_i} \times^* \mathbf{I}_{R_i}\mathbf{v}_{R_i}$ and $\mathbf{v}_{R_i} = \mathbf{v}_{p(i)} + \mathbf{\Phi}_i \dot{\mathbf{q}}_i n_i$ is the spatial velocity of the rotor.

Now consider the constrained case. When the joint is assembled, the admissible accelerations of each joint are limited by the accelerations of the rest of the kinematic tree. Therefore, the constrained acceleration of body i is simply given by Eq. 2.16 as $\mathbf{a}_i = \mathbf{a}_{p(i)} + \mathbf{\Phi}_i \ddot{\mathbf{q}}_i + (\mathbf{v}_i \times)\mathbf{\Phi}_i \dot{\mathbf{q}}_i$. A similar expression applies for the constrained acceleration of the rotor. Notably, the changes to Eq. 2.16 are that the joint velocity and acceleration of R_i are equal to the joint velocity and acceleration of body i multiplied by the gear ratio n_i because of the effects of the gearbox. Then, the equation for the constrained acceleration of R_i becomes

$$\mathbf{a}_{R_i} = \mathbf{a}_{p(i)} + \mathbf{\Phi}_i \ddot{\mathbf{q}}_i n_i + (\mathbf{v}_{R_i} \times)\mathbf{\Phi}_i \dot{\mathbf{q}}_i n_i. \tag{4.4}$$

From the Gauss Principle of Least Constraint, the acceleration of each body must

minimally deviate from its unconstrained acceleration at any given joint. Therefore, an appropriate reformulation of Eq. 4.1 that accounts for rotor inertias is given by

$$\begin{aligned} \underset{\ddot{\mathbf{q}}_i}{\text{argmin}} \quad & (\mathbf{a}_i - \mathbf{a}_i^{uc})^\top \mathbf{I}_i^A (\mathbf{a}_i - \mathbf{a}_i^{uc}) + (\mathbf{a}_{R_i} - \mathbf{a}_{R_i}^{uc})^\top \mathbf{I}_{R_i} (\mathbf{a}_{R_i} - \mathbf{a}_{R_i}^{uc}) \\ \text{where} \quad & \mathbf{a}_i = \mathbf{a}_{p(i)} + \mathbf{\Phi}_i \ddot{\mathbf{q}}_i + (\mathbf{v}_i \times) \mathbf{\Phi}_i \dot{\mathbf{q}}_i, \text{ and} \\ & \mathbf{a}_{R_i} = \mathbf{a}_{p(i)} + \mathbf{\Phi}_i \ddot{\mathbf{q}}_i n_i + (\mathbf{v}_{R_i} \times) \mathbf{\Phi}_i \dot{\mathbf{q}}_i n_i. \end{aligned} \tag{4.5}$$

4.3.2 Solving the Optimization Problem

Expanding the objective function of Eq. 4.5 it yields

$$z = \mathbf{a}_i^\top \mathbf{I}_i^A \mathbf{a}_i - 2\mathbf{a}_i^\top \mathbf{I}_i^A \mathbf{a}_i^{uc} + (\mathbf{a}_i^{uc})^\top \mathbf{I}_i^A \mathbf{a}_i^{uc} + \mathbf{a}_{R_i}^\top \mathbf{I}_{R_i} \mathbf{a}_{R_i} - 2\mathbf{a}_{R_i}^\top \mathbf{I}_{R_i} \mathbf{a}_{R_i}^{uc} + (\mathbf{a}_{R_i}^{uc})^\top \mathbf{I}_{R_i} \mathbf{a}_{R_i}^{uc}. \tag{4.6}$$

To solve the optimization problem presented by the Gauss Principle of Least Constraint, this expression must be differentiated with respect to $\ddot{\mathbf{q}}$, so any terms that do not depend on $\ddot{\mathbf{q}}$ will become 0 after differentiation. From the definitions of unconstrained accelerations for body and rotors in Eqs. 4.2 and 4.3, these terms do not depend on $\ddot{\mathbf{q}}$. Any terms in the objective function that depend solely on unconstrained accelerations will become irrelevant for the solution of Eq. 4.5. Thus, the relevant terms of the objective function are

$$z_1 = \mathbf{a}_i^\top \mathbf{I}_i^A \mathbf{a}_i - 2\mathbf{a}_i^\top \mathbf{I}_i^A \mathbf{a}_i^{uc}, \tag{4.7}$$

and

$$z_2 = \mathbf{a}_{R_i}^\top \mathbf{I}_{R_i} \mathbf{a}_{R_i} - 2\mathbf{a}_{R_i}^\top \mathbf{I}_{R_i} \mathbf{a}_{R_i}^{uc}. \tag{4.8}$$

Consider first the terms of body i, z_1. Substituting the constraints into the

relevant terms of the objective function, the expression becomes

$$\begin{aligned} z_1 = (\mathbf{a}_{p(i)} + \mathbf{\Phi}_i\ddot{\mathbf{q}}_i + (\mathbf{v}_i\times)\mathbf{\Phi}_i\dot{\mathbf{q}}_i)^\top \mathbf{I}_i^A(\mathbf{a}_{p(i)} + \mathbf{\Phi}_i\ddot{\mathbf{q}}_i + (\mathbf{v}_i\times)\mathbf{\Phi}_i\dot{\mathbf{q}}_i) \\ - 2(\mathbf{a}_{p(i)} + \mathbf{\Phi}_i\ddot{\mathbf{q}}_i + (\mathbf{v}_i\times)\mathbf{\Phi}_i\dot{\mathbf{q}}_i)^\top \mathbf{I}_i^A\mathbf{a}_i^{uc}. \end{aligned} \tag{4.9}$$

Further removing terms that are not dependent on $\ddot{\mathbf{q}}$, the expression simplifies to

$$z_1 = \ddot{\mathbf{q}}_i^\top \mathbf{\Phi}_i^\top \mathbf{I}_i^A \mathbf{\Phi}_i \ddot{\mathbf{q}}_i + 2\ddot{\mathbf{q}}_i^\top \mathbf{\Phi}_i^\top \mathbf{I}_i^A(\mathbf{a}_{p(i)} + (\mathbf{v}_i\times)\mathbf{\Phi}_i\dot{\mathbf{q}}_i) - 2\ddot{\mathbf{q}}_i^\top \mathbf{\Phi}_i^\top \mathbf{I}_i^A\mathbf{a}_i^{uc}. \tag{4.10}$$

Taking the gradient of Eq. 4.10 with respect to $\ddot{\mathbf{q}}_i$,

$$\frac{\partial z_1}{\partial \ddot{\mathbf{q}}_i} = 2\mathbf{\Phi}_i^\top \mathbf{I}_i^A \mathbf{\Phi}_i \ddot{\mathbf{q}}_i + 2\mathbf{\Phi}_i^\top \mathbf{I}_i^A(\mathbf{a}_{p(i)} + (\mathbf{v}_i\times)\mathbf{\Phi}_i\dot{\mathbf{q}}_i) - 2\mathbf{\Phi}_i^\top \mathbf{I}_i^A\mathbf{a}_i^{uc}. \tag{4.11}$$

Now consider the terms of rotor R_i, z_2. Again, the expression simplifies by discarding the terms that do not depend on $\ddot{\mathbf{q}}$. The expression further simplifies by defining the quantity $\mathbf{\Phi}_{R_i} = \mathbf{\Phi}_i n_i$ so that the expression for z_2 becomes

$$z_2 = \ddot{\mathbf{q}}_i^\top \mathbf{\Phi}_{R_i}^\top \mathbf{I}_{R_i} \mathbf{\Phi}_{R_i} \ddot{\mathbf{q}}_i + 2\ddot{\mathbf{q}}_i^\top \mathbf{\Phi}_{R_i}^\top \mathbf{I}_{R_i}(\mathbf{a}_{p(i)} + (\mathbf{v}_{R_i}\times)\mathbf{\Phi}_{R_i}\dot{\mathbf{q}}_i) - 2\ddot{\mathbf{q}}_i^\top \mathbf{\Phi}_{R_i}^\top \mathbf{I}_{R_i}\mathbf{a}_{R_i}^{uc}. \tag{4.12}$$

Taking the gradient,

$$\frac{\partial z_2}{\partial \ddot{\mathbf{q}}_i} = 2\mathbf{\Phi}_{R_i}^\top \mathbf{I}_{R_i} \mathbf{\Phi}_{R_i} \ddot{\mathbf{q}}_i + 2\mathbf{\Phi}_{R_i}^\top \mathbf{I}_{R_i}(\mathbf{a}_{p(i)} + (\mathbf{v}_{R_i}\times)\mathbf{\Phi}_{R_i}\dot{\mathbf{q}}_i) - 2\mathbf{\Phi}_{R_i}^\top \mathbf{I}_{R_i}\mathbf{a}_{R_i}^{uc}. \tag{4.13}$$

Now, by adding Eqs. 4.11 and 4.13 and equating to 0, the full equation to find $\ddot{\mathbf{q}}$ is

$$0 = \frac{\partial z}{\partial \ddot{\mathbf{q}}_i} = \frac{\partial z_1}{\partial \ddot{\mathbf{q}}_i} + \frac{\partial z_2}{\partial \ddot{\mathbf{q}}_i} \tag{4.14}$$

so that

$$0 = 2\boldsymbol{\Phi}_i^\top \mathbf{I}_i^A \boldsymbol{\Phi}_i \ddot{\mathbf{q}}_i + 2\boldsymbol{\Phi}_i^\top \mathbf{I}_i^A (\mathbf{a}_{p(i)} + (\mathbf{v}_i \times) \boldsymbol{\Phi}_i \dot{\mathbf{q}}_i) - 2\boldsymbol{\Phi}_i^\top \mathbf{I}_i^A \mathbf{a}_i^{uc} + 2\boldsymbol{\Phi}_{R_i}^\top \mathbf{I}_{R_i} \boldsymbol{\Phi}_{R_i} \ddot{\mathbf{q}}_i + 2\boldsymbol{\Phi}_{R_i}^\top \mathbf{I}_{R_i} (\mathbf{a}_{p(i)} + (\mathbf{v}_{R_i} \times) \boldsymbol{\Phi}_{R_i} \dot{\mathbf{q}}_i) - 2\boldsymbol{\Phi}_{R_i}^\top \mathbf{I}_{R_i} \mathbf{a}_{R_i}^{uc}. \tag{4.15}$$

For simplicity of notation, before solving for the optimal $\ddot{\mathbf{q}}$, the following new quantities are defined.

$$\boldsymbol{D}_i = \boldsymbol{\Phi}_i^\top \mathbf{I}_i^A \boldsymbol{\Phi}_i + \boldsymbol{\Phi}_{R_i}^\top \mathbf{I}_{R_i} \boldsymbol{\Phi}_{R_i}, \tag{4.16}$$

$$\boldsymbol{\xi}_i = (\mathbf{v}_i \times) \boldsymbol{\Phi}_i \dot{\mathbf{q}}_i, \tag{4.17}$$

and

$$\boldsymbol{\xi}_{R_i} = (\mathbf{v}_{R_i} \times) \boldsymbol{\Phi}_{R_i} \dot{\mathbf{q}}_i. \tag{4.18}$$

Solving Eq. 4.15 yields that the optimal $\ddot{\mathbf{q}}_i$ is expressed in terms of the new quantities as

$$\ddot{\mathbf{q}}_i = -\boldsymbol{D}_i^{-1} \boldsymbol{\Phi}_i^\top \mathbf{I}_i^A (\mathbf{a}_{p(i)} + \boldsymbol{\xi}_i - \mathbf{a}_i^{uc}) - \boldsymbol{D}_i^{-1} \boldsymbol{\Phi}_{R_i}^\top \mathbf{I}_{R_i} (\mathbf{a}_{p(i)} + \boldsymbol{\xi}_{R_i} - \mathbf{a}_{R_i}^{uc}). \tag{4.19}$$

4.3.3 Finding Recursive Relations

The $\ddot{\mathbf{q}}_i$ obtained through the Gauss Principle of Least Constraint in Eq. 4.19 has the remarkable property of being independent of the dynamics at other joints. If the articulated-body inertia and bias force of each body are known, $\ddot{\mathbf{q}}_i$ and $\mathbf{a}_i$ can be computed recursively. This subsection is dedicated to finding expressions for the articulated-body inertia and bias force given the new definition of $\ddot{\mathbf{q}}_i$.

Consider a test force on body $p(i)$ (Figure 4.3) so that

$$\mathbf{f}_{p(i)} = \mathbf{I}_{p(i)} \mathbf{a}_{p(i)} + \mathbf{p}_{p(i)} + \mathbf{I}_i^A \mathbf{a}_i + \mathbf{p}_i^A + \mathbf{I}_{R_i} \mathbf{a}_{R_i} + \mathbf{p}_{R_i}. \tag{4.20}$$

This force is also equal to the articulated terms based on Featherstone's definition in Eq. 2.19, so the equations becomes

$$\mathbf{I}^A_{p(i)}\mathbf{a}_{p(i)} + \mathbf{p}^A_{p(i)} = \mathbf{I}_{p(i)}\mathbf{a}_{p(i)} + \mathbf{p}_{p(i)} + \mathbf{I}^A_i\mathbf{a}_i + \mathbf{p}^A_i + \mathbf{I}_{R_i}\mathbf{a}_{R_i} + \mathbf{p}_{R_i}. \tag{4.21}$$

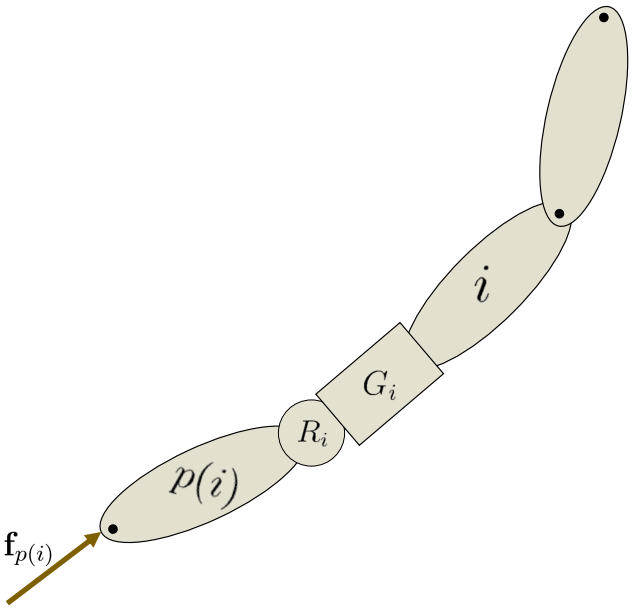


Figure 4.3. Force acting on body $p(i)$.

Expanding the terms $\mathbf{I}^A_i\mathbf{a}_i$ and $\mathbf{I}_{R_i}\mathbf{a}_{R_i}$ in terms of $\ddot{\mathbf{q}}$,

$$\mathbf{I}^A_i\mathbf{a}_i = \mathbf{I}^A_i(\mathbf{a}_{p(i)} - \mathbf{\Phi}_i\ddot{\mathbf{q}}_i + \boldsymbol{\xi}_i), \tag{4.22}$$

$$\mathbf{I}_{R_i}\mathbf{a}_{R_i} = \mathbf{I}_{R_i}(\mathbf{a}_{p(i)} - \mathbf{\Phi}_{R_i}\ddot{\mathbf{q}}_i + \boldsymbol{\xi}_{R_i}). \tag{4.23}$$

By substituting Eqs. 4.19, 4.22 and 4.23 into Eq. 4.21 the updated definitions of

articulated inertia and the articulated bias force become

$$\mathbf{I}^A_{p(i)} = \mathbf{I}_{p(i)} + \mathbf{I}^A_i + \mathbf{I}_{R_i} - \mathbf{I}^A_i \boldsymbol{\Phi}_i \boldsymbol{D}_i^{-1} \boldsymbol{\Phi}_i^\top \mathbf{I}^A_i - \mathbf{I}_{R_i} \boldsymbol{\Phi}_{R_i} \boldsymbol{D}_i^{-1} \boldsymbol{\Phi}_{R_i}^\top \mathbf{I}_{R_i} \tag{4.24}$$

and

$$\begin{aligned}\mathbf{p}^A_{p(i)} = \mathbf{p}_{p(i)} + \mathbf{p}^A_i + \mathbf{p}_{R_i} + \mathbf{I}^A_i \boldsymbol{\xi}_i + \mathbf{I}_{R_i} \boldsymbol{\xi}_{R_i} + \mathbf{I}^A_i \boldsymbol{\Phi}_i \boldsymbol{D}_i^{-1} \boldsymbol{\Phi}_i^\top \mathbf{I}^A_i \mathbf{a}^{uc}_i - \mathbf{I}^A_i \boldsymbol{\Phi}_i \boldsymbol{D}_i^{-1} \boldsymbol{\Phi}_i^\top \mathbf{I}^A_i \boldsymbol{\xi}_i \\ + \mathbf{I}_{R_i} \boldsymbol{\Phi}_{R_i} \boldsymbol{D}_i^{-1} \boldsymbol{\Phi}_{R_i}^\top \mathbf{I}_{R_i} \mathbf{a}^{uc}_{R_i} - \mathbf{I}_{R_i} \boldsymbol{\Phi}_{R_i} \boldsymbol{D}_i^{-1} \boldsymbol{\Phi}_{R_i}^\top \mathbf{I}_{R_i} \boldsymbol{\xi}_{R_i}.\end{aligned} \tag{4.25}$$

4.3.4 Modified Articulated-Body Algorithm

Using the updated $\mathbf{I}^A_{p(i)}$ and $\mathbf{p}^A_{p(i)}$, the modified ABA is shown in Algorithm 6. The structure is almost identical to that of the regular ABA. There is one sweep forward to initialize articulated terms and propagate velocities (lines 1-6), a backward pass to update the articulated terms (lines 8-20), and a final forward sweep to compute the joint rates and accelerations of each body (lines 21-24).

4.4 Results

Algorithm 6 was implemented in MATLAB using the Spatial_v2 package for rigid-body dynamics developed by Roy Featherstone [9]. This section presents the validity and performance checks that verify the correctness and efficiency of the algorithm.

Unless otherwise noted, all tests performed in this chapter consider mechanisms where all bodies are thin-walled cylinders of mass of 1 kg and length 1 m with $\dot{\mathbf{q}} = 0$ and gravity equal to 0. All rotors have a mass of 0.5 kg and a gear ratio of 5. Furthermore, the torque and joint position at each joint are set equal to the number of the joint in N·m and radians, respectively. For example, $\boldsymbol{\tau}_2 = 2$ N·m, and $\mathbf{q}_{14} = 14$ rad.

Algorithm 6 Articulated-Body Algorithm

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

for $i = 1$ to N **do**
  $\mathbf{I}_i^A = \mathbf{I}_i$
  $\boldsymbol{\Phi}_{R_i} = \boldsymbol{\Phi}_i n_i$
  $\mathbf{p}_i^A = \mathbf{v}_i \times^* \mathbf{I}_i \mathbf{v}_i$
  $\mathbf{p}_{R_i} = \mathbf{v}_{R_i} \times^* \mathbf{I}_{R_i} \mathbf{v}_{R_i}$
  $\mathbf{v}_i = {}^i\mathbf{X}_{p(i)} \mathbf{v}_{p(i)} + \boldsymbol{\Phi}_i \dot{\mathbf{q}}_i$
  $\mathbf{v}_{R_i} = {}^{R_i}\mathbf{X}_{p(i)} \mathbf{v}_{p(i)} + \boldsymbol{\Phi}_{R_i} \dot{\mathbf{q}}_i$
  $\boldsymbol{\xi}_i = (\mathbf{v}_i \times) \boldsymbol{\Phi}_i \dot{\mathbf{q}}_i$
  $\boldsymbol{\xi}_{R_i} = (\mathbf{v}_{R_i} \times) \boldsymbol{\Phi}_{R_i} \dot{\mathbf{q}}_i$
end for
for $i = N$ to 1 **do**
  $\mathbf{a}_i^{uc} = (\mathbf{I}_i^A)^{-1} (\boldsymbol{\Phi}_i \boldsymbol{\tau}_i - \mathbf{p}_i^A)$
  $\mathbf{a}_{R_i}^{uc} = \mathbf{I}_{R_i}^{-1} (\boldsymbol{\Phi}_i \frac{\boldsymbol{\tau}_i}{n_i} - \mathbf{p}_{R_i})$
  $\boldsymbol{D}_i = \boldsymbol{\Phi}_i^\top \mathbf{I}_i^A \boldsymbol{\Phi}_i + \boldsymbol{\Phi}_{R_i}^\top \mathbf{I}_{R_i} \boldsymbol{\Phi}_{R_i}$
  if $p(i) \neq 0$ **then**
    $\mathbf{I}_{p(i)}^A = \mathbf{I}_{p(i)} + {}^i\mathbf{X}_{p(i)}^\top [\mathbf{I}_i^A + \mathbf{I}_{R_i} - \mathbf{I}_i^A \boldsymbol{\Phi}_i \boldsymbol{D}_i^{-1} \boldsymbol{\Phi}_i^\top \mathbf{I}_i^A - \mathbf{I}_{R_i} \boldsymbol{\Phi}_{R_i} \boldsymbol{D}_i^{-1} \boldsymbol{\Phi}_{R_i}^\top \mathbf{I}_{R_i}]^i\mathbf{X}_{p(i)}$
    $\mathbf{p}_{p(i)}^A = \mathbf{p}_{p(i)} + {}^i\mathbf{X}_{p(i)}^\top [\mathbf{p}_i^A + \mathbf{p}_{R_i} + \mathbf{I}_i^A \boldsymbol{\xi}_i + \mathbf{I}_{R_i} \boldsymbol{\xi}_{R_i} + \mathbf{I}_i^A \boldsymbol{\Phi}_i \boldsymbol{D}_i^{-1} \boldsymbol{\Phi}_i^\top \mathbf{I}_i^A (\mathbf{a}_i^{uc} - \boldsymbol{\xi}_i) + \mathbf{I}_{R_i} \boldsymbol{\Phi}_{R_i} \boldsymbol{D}_i^{-1} \boldsymbol{\Phi}_{R_i}^\top \mathbf{I}_{R_i} (\mathbf{a}_{R_i}^{uc} - \boldsymbol{\xi}_{R_i})]^i\mathbf{X}_{p(i)}$
  end if
end for
for $i = 1$ to N **do**
  $\ddot{\mathbf{q}}_i = -\boldsymbol{D}_i^{-1} \boldsymbol{\Phi}_i^\top [\mathbf{I}_i^A ({}^i\mathbf{X}_{p(i)} \mathbf{a}_{p(i)} + \boldsymbol{\xi}_i - \mathbf{a}_i^{uc}) + n_i \mathbf{I}_{R_i} ({}^i\mathbf{X}_{p(i)} \mathbf{a}_{p(i)} + \boldsymbol{\xi}_{R_i} - \mathbf{a}_{R_i}^{uc})]$
  $\mathbf{a}_i = {}^i\mathbf{X}_{p(i)} \mathbf{a}_{p(i)} + \boldsymbol{\Phi}_i \ddot{\mathbf{q}}_i + (\mathbf{v}_i \times) \boldsymbol{\Phi}_i \dot{\mathbf{q}}_i$
end for
return $\ddot{\mathbf{q}}$, $\mathbf{a}$

TABLE 4.1

CONSISTENCY CHECK FOR A SERIAL KINEMATIC CHAINS OF 5, 10, AND 15 BODIES (MAXIMUM ERROR)

	$N = 5$ (N·m)	$N = 10$ (N·m)	$N = 15$ (N·m)
$\boldsymbol{\tau} - \boldsymbol{\tau}_{RNEA}$	4.3×10^{-14}	4.3×10^{-13}	2.4×10^{-13}

4.4.1 Validation

Method 1: Self-consistency. If the algorithm is mathematically consistent, the inverse and forward dynamics must truly be the inverse of each other. To verify this is the case, the $\ddot{\mathbf{q}}$ computed via Algorithm 6 became the input for the RNEA that computes the torque at each joint. This procedure checked the results for serial kinematic chains of 5,10 and 15 bodies. The torques from the RNEA in all of these cases match the initial input torques for the modified ABA (Table 4.1); therefore, the algorithm is consistent.

Method 2: Comparison to usual approximation methods. In the industrial setting where actuators tend to have large gear ratios, the inertial effects of rotors are approximating by modifying the mass matrix in the CRBA. The motor inertia term $\boldsymbol{\Phi}_i^\top \mathbf{I}_{R_i} \boldsymbol{\Phi}_i n_i^2$ adds to the diagonal entries of $\mathbf{H}$. Then according to the canonical equations of motion, $\ddot{\mathbf{q}}$ is computed with a new $\mathbf{H}$ as

$$\ddot{\mathbf{q}} = \mathbf{H}^{-1}(\boldsymbol{\tau} - \mathbf{C}\dot{\mathbf{q}} - \mathbf{g}). \tag{4.26}$$

When the actuators are inertialess, the modified CRBA (MCRBA), the modified ABA in Algorithm 6 and the regular ABA should produce the same results. Similarly,

TABLE 4.2

MAXIMUM DIFFERENCE IN $\ddot{\mathbf{q}}$ FOR SERIAL CHAINS WITH 5, 10, AND 15 BODIES WITH **INERTIALESS ACTUATORS**.

	$N = 5$ (rad·s^{-2})	$N = 10$ (rad·s^{-2})	$N = 15$ (rad·s^{-2})
MABA and ABA	8.4×10^{-15}	2.9×10^{-14}	1.9×10^{-13}
MABA and MCRBA	1.5×10^{-14}	6.1×10^{-14}	2.2×10^{-13}
MCRBA and ABA	6.2×10^{-15}	6.7×10^{-14}	2.1×10^{-13}

when the rotors' inertia is significant, MABA's results must be closer to those of MCRBA's approximation than the results of the regular ABA. Differences between MABA and MCRBA results are inevitable because Algorithm 6 is exact, whereas the other is an approximation.

Consider serial kinematic chains of 5, 10 and 15 bodies. As expected, for the case when $\mathbf{I}_{R_i} = 0$, all three algorithms produce the same results without variation (Table 4.2). In contrast, when the rotors' inertia is significant, the ABA produces results that are vastly different from those of the other two algorithms, but the results from MABA and MCRBA are equal to multiple decimal places. This is the case for serial chains as well as branched mechanisms (Tables 4.3 and 4.4), which speaks to the correctness of Algorithm 6.

The close agreement between MABA and MCRBA in these tests is contingent upon $\dot{\mathbf{q}} = 0$. When $\dot{\mathbf{q}} \neq 0$, there are Coriolis terms from the rotor that are completely ignored by the CRBA. If $\dot{\mathbf{q}}$ is not close to 0, the Coriolis terms become significant so that MCRBA is no longer valid. Consider serial kinematic chains of 5, 10, and 15 bodies as in previous cases, but with $\dot{\mathbf{q}} = \frac{\pi}{2}$ rad·s^{-1} for all joints. In this case,

TABLE 4.3

MAXIMUM DIFFERENCE IN $\ddot{\mathbf{q}}$ FOR SERIAL CHAINS WITH 5, 10, AND 15 BODIES.

	$N = 5$ (rad·s^{-2})	$N = 10$ (rad·s^{-2})	$N = 15$ (rad·s^{-2})
MABA and ABA	23.12	39.14	57.72
MABA and MCRBA	0.026	0.037	0.062
MCRBA and ABA	23.15	39.18	57.79

the difference in $\ddot{\mathbf{q}}$ between MABA and CRABA can be as large as hundreds of rad·s^{-2} (Table 4.5). When the discrepancies are this large, MABA provides the most accurate results. Because MABA accounts for rotor inertia from first principles, the Coriolis terms are implicitly accounted for in Algorithm 6, which makes it much more accurate than MCRBA under these circumstances.

4.4.2 Performance

The benchmarking performance of Algorithm 6 included serial kinematic chains and branching mechanisms with a branching factor of 2. As expected, the algorithm's complexity grew as $O(N)$, matching the performance of the regular ABA (Figure 4.4). Furthermore, branching had no effects on the computational time of the algorithm (Figure 4.5). Because the algorithm has no loop that runs through the ancestors of each body within the middle backward pass, the depth of each branch has no effect on complexity.

In terms of runtime, Algorithm 6 performed similarly to the regular ABA (Table 4.6). The difference between the runtime of both algorithms is less than a millisecond,

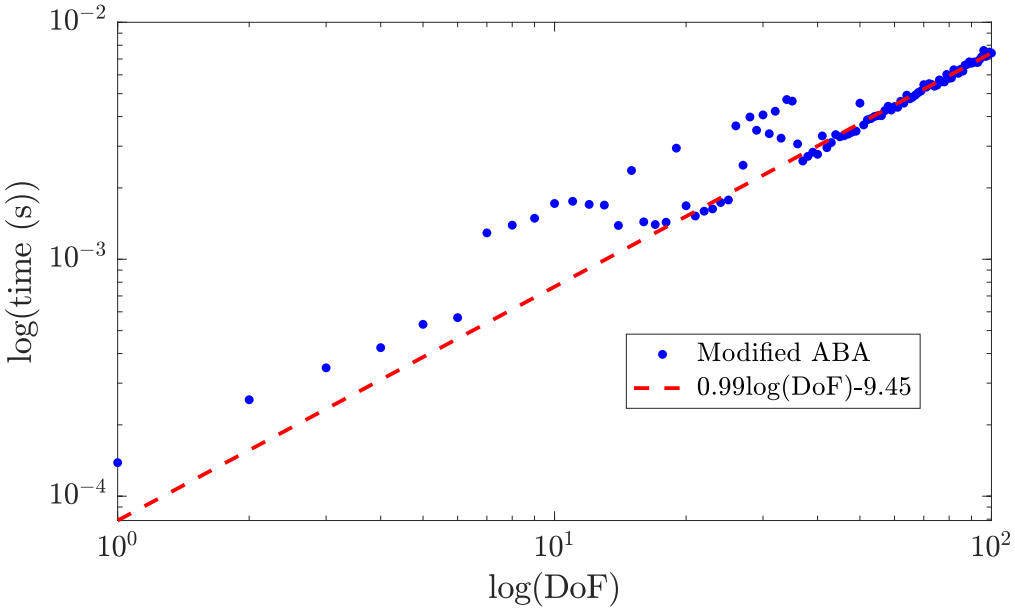


Figure 4.4. Computation time for serial kinematic chains.

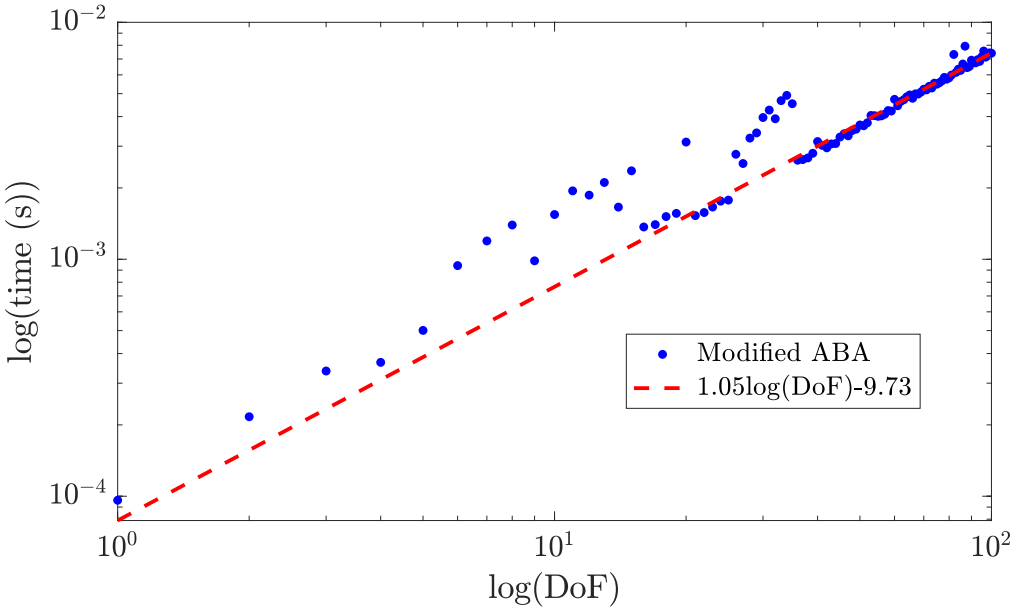


Figure 4.5. Computation time for branching mechanism (branching factor of 2).

TABLE 4.4

MAXIMUM DIFFERENCE IN $\ddot{\mathbf{q}}$ FOR BRANCHING MECHANISM WITH 5, 10, AND 15 BODIES (BRANCHING FACTOR OF 2).

	$N = 5$ (rad·s^{-2})	$N = 10$ (rad·s^{-2})	$N = 15$ (rad·s^{-2})
MABA and ABA	28.16	79.64	179.38
MABA and MCRBA	0.037	0.070	0.11
MCRBA and ABA	28.19	79.69	179.47

and it is because the expressions for the articulated terms in Algorithm 6 are much more computationally expensive than those in the ABA. Still, Algorithm 6 proved to be fast. It can solve the forward dynamics problem a thousand times per second. Note that the times in Table 4.6 come from implementation in MATLAB, so times can improve significantly if the algorithm runs using a compiled language such as C++.

4.5 Conclusion

The work presented in this chapter shows that considering rigid-body dynamics as an optimization problem through the Gauss Principle of Least Constraint allows for more accurate dynamics algorithms. In particular, the chapter showed how by adding rotor inertia terms and solving the optimization problem, it is possible to find a Modified Articulated-Body Algorithm that accounts for rotor inertias from first principles. The chapter also showed how the proposed algorithm compares to a common approximation of actuators' inertial effects and how its complexity and runtime match those of the regular Articulated-Body Algorithm.

TABLE 4.5

MAXIMUM DIFFERENCE IN $\ddot{\mathbf{q}}$ FOR SERIAL CHAINS WITH 5, 10, AND 15 BODIES WHEN $\dot{\mathbf{q}} = \frac{\pi}{2}$ RAD·S^{-1}.

	$N = 5$ (rad·s^{-2})	$N = 10$ (rad·s^{-2})	$N = 15$ (rad·s^{-2})
MABA and ABA	83.41	171.97	776.78
MABA and MCRBA	1.51	23.47	102.61
MCRBA and ABA	84.06	179.53	836.16

TABLE 4.6

RUNTIME FOR MABA AND ABA.

	MABA (ms)	ABA (ms)
Serial Chain (20 DoF)	1.4	0.6
Branching Mechanism (20 DoF)	1.5	0.6

Arguably the most impressive feature of this chapter is the proven flexibility of the Gauss Principle. Many dynamics algorithms still use approximations for the effects of actuators and gearing in the overall dynamics of the system. These effects can be accounted for exactly using the methodology described in this chapter. Notably, by adding terms to the optimization problem in Eq. 4.5, it is possible to solve for and adapt ABA to consider any other desired structural elements. Placing rigid-body dynamics in the context of the Gauss Principle allows improving dynamics algorithms

further to be more accurate and flexible in accounting for the different characteristics of modern-day actuators.

CHAPTER 5

CONCLUSIONS

5.1 Summary and Conclusions

The results presented in this work showcase that algorithms to compute the Coriolis matrix, its Christoffel symbols, and the inertial effects of motor rotors are viable for real-time implementation in model-based control applications. Three recursive algorithms, the Recursive Newton-Euler Algorithm, the Articulated-Body Algorithm, and the Composite Rigid-Body Algorithm, were reviewed. These algorithms are widely used for many robotics applications and provide the framework needed to develop new algorithms that expand the capacity to compute other dynamic terms numerically.

Then, using a suitable body-level factorization of the velocity product terms, it was possible to develop a recursive algorithm to compute each entry in the Coriolis matrix in terms of composite quantities in CRBA style. Then, taking the derivatives of the expressions in the Coriolis matrix algorithm leads to an expression for the Christoffel symbols in terms of body-level factorization terms that can be computed recursively in an algorithm. These two algorithms are of the lowest possible complexity, $O(Nd)$ and $O(Nd^2)$, respectively, and their runtimes are on the order of a few tens of μs for systems with 20 DoF. The performance of both algorithms provides a scalable, efficient, and fast alternative to other algorithms and approximation methods. These results are particularly promising for the Christoffel symbols algorithm because it is thousands of times faster than other alternatives available in the literature involving symbolic partial derivatives.

This thesis also showed how the Gauss Principle of Least Constraint makes it possible to derive an algorithm that accounts for rotor inertias from first principles by adding terms to the objective function of the optimization problem as well as relevant constraints. This algorithm uses the same structure as the Articulated-Body Algorithm, but it has expressions for the articulated terms that include the inertial effects of actuators. This algorithm is of equal complexity and similar runtime as the original Articulated-Body Algorithm, which makes it viable for implementation in real-time controllers or rapid simulation for learning-based control design strategies. This new algorithm offers a much more accurate alternative to the regular Articulated-Body Algorithm when the inertial effects of the rotors are significant.

In conclusion, this thesis used numerical methods to derive new rigid-body dynamics algorithms that are feasible for implementation in model-based controllers that require dynamics calculations as part of their control scheme. The presented algorithms proved to meet or exceed the performance of other currently available algorithms in the literature. Because of their scalability, accuracy, and speed, the algorithms developed in this work are valid candidates for real-time implementation in applications of sensorless contact detection, passivity-based control, and adaptive control.

5.2 Future Work

All the results presented in this work were obtained through computer simulations of simple rigid-body systems. While these results successfully prove, in theory, the accuracy and viability of the algorithms presented in this thesis, further empirical evidence is needed for establishing the true capabilities of these algorithms. First, implementing the Coriolis matrix algorithm in the real-time control of a robot that uses sensorless contact detection, such as MIT's Cheetah robot, will provide empirical data for how fast and accurate the algorithm is under practical circumstances.

Similarly, implementing the Christoffel symbols algorithm in geometric controllers that use approximations for the Christoffel symbols will allow measuring how this new algorithm might improve the performance of certain physical robots.

Another research avenue regarding the Modified Articulated-Body Algorithm is to expand it further to include other inertial effects. Adding gears and internal friction might significantly improve the accuracy of the algorithms, particularly in physical systems where the gearing system has comparable mass to the motor. Furthermore, the joint model used to develop the Modified Articulated-Body Algorithm assumes that the free modes of motion of the rotor and the body are the same. This assumption limits the applicability of the algorithm to systems with basic revolute joints. Forgoing this assumption allows making the algorithm fully general and applicable to any physical system. Then, a rederivation of the algorithm with more complicated expressions for the articulated terms is required.

Further work for the Modified Articulated-Body Algorithm involves physical systems. The implementation of this algorithm with a physical robot with significant inertial effects from the actuators will measure how accurate this algorithm is compared to the regular Articulated-Body Algorithm and the Modified Composite Rigid-Body Algorithm approximation. Obtaining data points from different physical systems will allow developing a metric to determine when different approximations and algorithms are no longer valid due to the inertial effects from actuators. Further research involving the characterization of inertial effects of different structural components provides opportunities to fine tune the Modified Articulated-Body Algorithm and further improve its accuracy.

BIBLIOGRAPHY

1. *Springer Handbook of Robotics*. Springer handbooks. Springer International Publishing : Imprint: Springer, Cham, 2nd edition.. edition, 2016. ISBN 9783319325521.

2. G. Bledt, P. Wensing, S. Ingersoll, and S. Kim. Contact model fusion for event-based locomotion in unstructured terrains. pages 4399–4406, May 2018. doi: 10.1109/ICRA.2018.8460904.

3. J. Bobrow, F. Park, and A. Sideris. *Recent Advances on the Algorithmic Optimization of Robot Motion*, pages 21–41. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006. ISBN 978-3-540-36119-0. doi: 10.1007/978-3-540-36119-0_2. URL `https://doi.org/10.1007/978-3-540-36119-0_2`.

4. A. Carvalho and A. Suleman. Simulation of rigid-body impact using the articulated-body algorithm. *Robotica*, 29(5):649–656, 2011. ISSN 02635747. URL `http://search.proquest.com/docview/899301478/`.

5. A. De Luca and L. Ferrajoli. A modified newton-euler method for dynamic computations in robot fault detection and control. In *2009 IEEE International Conference on Robotics and Automation*, pages 3359–3364. IEEE, 2009. ISBN 9781424427888.

6. Y. Fan, R. Kalaba, H. Natsuyama, and F. Udwadia. Reflections on the gauss principle of least constraint. *Journal of Optimization Theory and Applications*, 127(3):475–484, 2005. ISSN 0022-3239.

7. R. Featherstone. The calculation of robot dynamics using articulated-body inertias. *The International Journal of Robotics Research*, 2(1):13–30, 1983. ISSN 0278-3649.

8. R. Featherstone. *Rigid body dynamics algorithms*. Springer, New York, 2008. ISBN 9780387743158.

9. R. Featherstone, Feb 2015. URL `http://royfeatherstone.org/spatial/v2/index.html#ID`.

10. R. Featherstone and D. Orin. Robot dynamics: equations and algorithms. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, volume 1, pages 826–834 vol.1. IEEE, 2000. ISBN 0780358864.

11. M. L. Felis. Rbdl: an efficient rigid-body dynamics library using recursive algorithms. *Autonomous Robots*, pages 1–17, 2016. ISSN 1573-7527. doi: 10.1007/s10514-016-9574-0. URL http://dx.doi.org/10.1007/s10514-016-9574-0.

12. J. M. Hollerbach. A recursive lagrangian formulation of maniputator dynamics and a comparative study of dynamics formulation complexity. *IEEE Transactions on Systems, Man, and Cybernetics*, 10(11):730–736, 1980. ISSN 0018-9472.

13. A. Jain and G. Rodriguez. Recursive linearization of manipulator dynamics models. In *1990 IEEE International Conference on Systems, Man, and Cybernetics Conference Proceedings*, pages 475–480. IEEE, 1990. ISBN 0879425970.

14. A. Jain and G. Rodriguez. Linearization of manipulator dynamics using spatial operators. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(1):239–248, 1993. ISSN 0018-9472.

15. O. Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Journal on Robotics and Automation*, 3(1):43–53, 1987. ISSN 0882-4967.

16. J. Luh, M. Walker, and R. Paul. On-line computational scheme for mechanical manipulators. *Journal of Dynamic Systems, Measurement and Control, Transactions of the ASME*, 102(2):69–76, 1980. ISSN 00220434.

17. J. Murray and D. Johnson. The linearized dynamic robot model: efficient computation and practical applications. In *Proceedings of the 28th IEEE Conference on Decision and Control*, pages 1659–1664 vol.2. IEEE, 1989.

18. J. Murray and C. Neuman. Organizing customized robot dynamics algorithms for efficient numerical evaluation. *IEEE Transactions on Systems, Man, and Cybernetics*, 18(1):115–125, 1988. ISSN 0018-9472.

19. A. Mller and Z. Terze. Geometric methods and formulations in computational multibody system dynamics. *Acta Mechanica*, 227(12):3327–3350, 2016. ISSN 00015970. URL http://search.proquest.com/docview/1846648706/.

20. G. Niemeyer and J.-J. E. Slotine. Performance in adaptive manipulator control. *The International Journal of Robotics Research*, 10(2):149–161, 1991. ISSN 0278-3649.

21. E. Otten. Inverse and forward dynamics: models of multibody systems. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 358(1437): 1493–1500, 2003. ISSN 0962-8436.

22. L. Righetti, J. Buchli, M. Mistry, M. Kalakrishnan, and S. Schaal. Optimal distribution of contact forces with inverse-dynamics control. *The International Journal of Robotics Research*, 32(3):280–298, 2013. ISSN 0278-3649.

23. C. Semini, N. Tsagarakis, E. Guglielmino, M. Focchi, F. Cannella, and D. Caldwell. Design of hyq -a hydraulically and electrically actuated quadruped robot. *Proceedings of the Institution of Mechanical Engineers. Part I: Journal of Systems and Control Engineering*, 225:831–849, 08 2011. doi: 10.1177/0959651811402275.

24. B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo. *Robotics: Modelling, Planning and Control*. 2009. ISBN 9781846286414. URL `http://search.proquest.com/docview/35584129/`.

25. J.-J. E. Slotine and W. Li. On the adaptive control of robot manipulators. *The International Journal of Robotics Research*, 6(3):49–59, 1987. ISSN 0278-3649.

26. G. A. Sohl and J. E. Bobrow. A recursive multibody dynamics and sensitivity algorithm for branched kinematic chains. *Journal of Dynamic Systems, Measurement, and Control*, 123(3):391, 2001. ISSN 0022-0434.

27. W. Suleiman, E. Yoshida, J.-P. Laumond, and A. Monin. Optimizing humanoid motions using recursive dynamics and lie groups. In *2008 3rd International Conference on Information and Communication Technologies: From Theory to Applications*, pages 1–6. IEEE, 2008. ISBN 9781424417513.

28. M. Takegaki and S. Arimoto. A new feedback method for dynamic control of manipulators. *Journal of Dynamic Systems, Measurement and Control, Transactions of the ASME*, 103(2):119–125, 1981. ISSN 00220434.

29. A. Vereshchagin. Computer simulation of the dynamics of complicated mechanisms of robot-manipulators. *Eng Cybern*, 12(6):65–70, 1974.

30. P. Wensing, R. Featherstone, and D. E. Orin. A reduced-order recursive algorithm for the computation of the operational-space inertia matrix. In *2012 IEEE International Conference on Robotics and Automation*, pages 4911–4917. IEEE, 2012. ISBN 9781467314039.

31. K. Yunt. Gauss' principle and principle of least constraints for dissipative mechanical systems. *IFAC Proceedings Volumes*, 45(2):842–847, 2012. ISSN 1474-6670.