Autonomous Robots manuscript No.
(will be inserted by the editor)

Efficient Recursive Dynamics Algorithms for Operational-Space Control with Application to Legged Locomotion

Patrick M. Wensing · Luther R. Palmer III · David E. Orin

Received: date / Accepted: date

Abstract This paper presents new recursive dynamics algorithms that enable operational-space control of floating-base systems to be performed at faster rates. This type of control approach requires the computation of operational-space quantities and suffers from high computational order when these quantities are directly computed through the use of the mass matrix and Jacobian from the joint-space formulation. While many efforts have focused on efficient computation of the operational-space inertia matrix $\mathbf{\Lambda}$, this paper provides a recursive algorithm to compute all quantities required for floating-base control of a tree-structure mechanism. This includes the first recursive algorithm to compute the dynamically consistent pseudoinverse of the Jacobian $\bar{\boldsymbol{J}}$ for a tree-structure system. This algorithm is extended to handle arbitrary contact constraints with the ground, which are often found in legged systems, and uses effective ground contact dynamics approximations to retain computational efficiency. The usefulness of the algorithm is demonstrated through application to control of a high-speed quadruped trot in simulation. Our contact-consistent algorithm demonstrates pitch and roll stabilization for a large dog-sized quadruped running at 3.6 m/s without any contact force sensing, and is shown to outperform a simpler Raibert-style posture controller. In addition, the operational-space control approach allows the dynamic effects of the swing legs to be effectively accounted for at this high speed.

Keywords recursive dynamics algorithms · operational-space control · quadruped trot · dynamically consistent Jacobian pseudoinverse

P. M. Wensing
Department of Mechanical Engineering,
Massachusetts Institute of Technology, Cambridge, MA 02139, USA
E-mail: pwensing@mit.edu

L. R. Palmer III
Department of Computer Science & Engineering,
University of Southern Florida, Tampa, FL 33620, USA
E-mail: palmer@cse.usf.edu

D. E. Orin
Department of Electrical and Computer Engineering,
The Ohio State University, Columbus, OH 43210, USA
E-mail: orin.1@osu.edu

1 Introduction

Recursive dynamics algorithms have provided computational benefits to the solution of many difficult problems within rigid-body dynamics. This class of algorithms has been applied to solve problems in forward (Walker and Orin 1982; Featherstone 1983), inverse (Luh et al. 1980), operational-space (Rodriguez et al. 1992; Wensing et al. 2012), and centroidal (Orin et al. 2013) dynamics with computational efficiency. Through successive consideration of the dynamics of rigid-body subsystems, these algorithms are able to achieve low computational order. This low computational order often enables these algorithms to outperform competing nonrecursive algorithms in terms of their computational requirements.

This paper presents new recursive dynamics algorithms which compute the operational-space dynamics of floating-base tree-structure systems. Specifically, the paper includes the first method to recursively compute the dynamically consistent Jacobian pseudoinverse when the position and orientation of a privileged body (namely the torso, in this work) are selected as the task. Additionally, the methods are the first to provide fully-recursive dynamics algorithms for the operational-space dynamics of tree-structure systems in ground contact. The applicability of the algorithms is shown for torso control during a quadruped run for a large dog-sized model at 3.6 m/s in simulation. This speed represents a fast trot, as it is just slightly under the trot-to-gallop transition speed of 3.91 m/s that would be expected in a biological

quadruped of this size (Heglund and Taylor 1988). While the results here highlight torso control for the quadruped, the algorithms are general to provide operational-space control of any single body during periods of ground support.

The control of manipulators and legged machines in task space (also known as operational space) has enjoyed widespread application since its introduction more than 25 years ago (Khatib 1987). Application of operational-space control allows a controller to focus on the most important aspects of a motion, and offers principled methods to coordinate the contributions of many actuators in high degree-of-freedom systems. In systems with redundancy to achieve a specified task, this method also allows, in principle, for the decoupling of task and null-space dynamics. While the operational-space formalism was originally developed to describe the dynamics of a single unconstrained end-effector, it has been extended to accommodate general task spaces that depend on the motion of more than one body (Russakow et al. 1995), and to handle holonomic constraints (de Sapio and Khatib 2005).

More recent work has also demonstrated the promise of the operational-space formalism to the control of constrained and underactuated systems. Sentis and Khatib (2005) demonstrated an extension of the operational-space framework to control floating-base humanoid systems in flight. Exploiting conservation of angular momentum in flight, these methods were able to handle the coupling that each limb's motion has on the motion of the floating system as a whole. Park and Khatib (2006) built upon de Sapio and Khatib (2005) to enable operational-space control of humanoids in ground contact. Sentis et al. (2010) later extended this work for control of internal contact forces. For humanoids in double support, the large footprint provides contact force redundancy to perform any motion (Wensing et al. 2013). By incorporating model information at the dynamics level, these operational-space control methods have much more authority to modify contact forces than is capable through other position control schemes. Other model-based methods, such as those described in Mistry and Righetti (2011), address the control of contact forces as well through the use of projected inverse dynamics.

In addition to application for humanoids, these control methods for systems in contact have been applied to simulated quadruped walking on challenging terrain (Hutter et al. 2012). For legs in closed-kinematic chains with the ground, the use of operational-space control to select stance torques has great benefit over position controlled approaches, as small positional errors at the feet do not lead to large internal forces between the feet. From a practical standpoint, recent hardware implementation of whole-body operational-space control methods has provided new hardware proof of whole-body control concepts (Sentis et al. 2013). By providing model-based control at the dynamics level, these control approaches have the ability to operate more compliantly than with stiff position servos. From a broad perspective, the continued development of these methods provides great potential applicability in next-generation compliant torque-controlled robotic systems.

Over the years, many efficient algorithms have been developed to compute the operational-space dynamic equations of motion in order to support these control approaches. The largest body of work has concentrated on unconstrained manipulators, with original algorithms by Lilly (1989), Kreutz-Delgado et al. (1991), and Lilly and Orin (1993). More recent approaches have been provided by Bhalerao et al. (2013). These approaches have been extended to a more general operational space that may include multiple end-effectors (Rodriguez et al. 1992; Chang and Khatib 2001; Wensing et al. 2012). As a common theme across these approaches, novel restructuring of the efficient recursive structure of the Articulated-Body Algorithm (Featherstone 1983) enables fast computation of operational-space dynamics in each of these settings.

However, despite this large body of work, algorithms for the dynamically consistent Jacobian pseudoinverse remain largely unstudied, with no fully-recursive algorithm available in the literature. This quantity can be helpful to coordinate many actuators for task control by providing a link between individual joint torques and effective forces at a task point. In addition, recursive algorithms for operational-space dynamics under constraints have not yet emerged to support the new developments that enable operational-space control to be applied for systems with legs in contact. A recent exception is the work by Jain (2013) which has provided partially recursive algorithms for the operational-space inertia alone, in systems with internal loops.

Although operational-space dynamics algorithms for systems under constraints have not been studied to any significant degree, the dynamic simulation problem for constrained systems has received much attention. Lathrop (1986) described how constraint-propagation methods could be used to simulate tree-structure systems under contact constraints as well as internal loop-closure constraints. Parallelized constraint-propagation methods (Featherstone 1999a,b) have been developed to utilize multiple processors, when available. A comparative summary of these and other methods is provided in a recent review (Yamane and Nakamura 2009). Other algorithms have been developed for the simulation of simple closed-chain structures wherein the loop closures generated by contact can be broken through the removal of a single body (Lilly 1993; McMillan et al. 1994). The use of these algorithms as a starting point for operational-space dynamics under constraints provides an alternative approach to the one taken here. Instead, through the new application of approximated ground contact con-

straints, the algorithm here exhibits efficient computational performance while retaining computational accuracy.

The recursive algorithms developed in this paper are applied to the control of a quadruped trot. The trot is characterized by diagonal leg pairs operating in sync throughout the stride. The symmetry of this footfall pattern simplifies the mechanics of the trot in comparison to asymmetric gaits such as the gallop (Schmiedeler and Waldron 1999), but requires the efforts of the stance legs to remain coordinated during the entire stance. Many trot control approaches utilize compliant mechanisms in the legs to conserve energy during the running step (Raibert 1990; Schmiedeler and Waldron 2002; Hyon and Mita 2002; Raibert et al. 2008), among other benefits (McMahon 1985). Step-to-step control algorithms for these systems often seek to control the leg parameters, such as its length, stiffness and angle with respect to the ground at touchdown and then allow the system to operate passively during stance (Ahmadi and Buehler 1997; Palmer et al. 2003; Marhefka et al. 2003; Nichol et al. 2004). Due to the absence of continuous in-stride feedback control of the torso orientation, these footstep algorithms are very sensitive to the proper selection of leg touchdown parameters. This sensitivity places a large burden on the footstep controller and further prevents its use when ground contact is uncertain or the terrain is highly unpredictable.

Continuous in-stride stabilization of a dynamic trot presents other challenges. Multiple degrees of freedom in each leg must be coordinated to propel the torso, stabilize and reverse its vertical momentum. Further, these coordinated actions are only able to occur during the short periods of foot-ground support. This coordination is complicated by leg articulation, which causes the dynamically-coupled effects of the leg torques to be configuration dependent.

In order to address these difficulties, a dynamically consistent Jacobian pseudoinverse will be used in this work for continuous torso control. By relating joint torques of the legs in contact to corrective forces transmitted to the torso, the use of this quantity provides a crisp method to perform configuration-dependent coordination of the leg actuators. As an additional complexity, these stance torques must account for the return legs, which swing forward rapidly in preparation for their next contact phase. The algorithms used here account for the dynamic effects of these swing legs, enabling the operational-space torso controller to maintain trot stability even with crude high-level footstep control. To show the benefits of the operational-space torso control approach, the results are compared to a Raibert-style posture controller that uses roll and pitch hip torques to correct torso orientation through a decoupled control law.

The remainder of the paper is organized as follows. Section 2 briefly outlines the notation and conventions used to describe the kinematics and dynamics of tree-structure systems. Section 3 provides a review of the operational-space

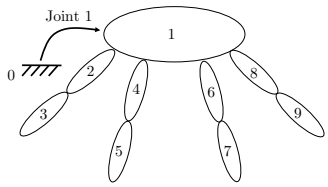

Fig. 1 An example numbering for a rigid-body system with $N_B = 9$. Joint 1 is a virtual 6-DoF joint that connects the fixed base (Body 0) to the floating base (Body 1).

equations of motion for both unconstrained (open-chain) and constrained (closed-chain) systems. While the formulae in this section can be used to perform operational-space control directly, their computational order is $O(n^3 + m^3)$ where n is the number of degrees of freedom in the system and m is the number of constraints. Section 4 provides two recursive algorithms to compute the operational-space dynamics in the open-chain and constrained closed-chain cases. Through judicious restructuring of the Articulated-Body Algorithm recursions, these algorithms have an improved order of $O(nd)$, where d is the maximum depth of the kinematic connectivity tree. Section 5 describes a quadruped trot control example, while Section 6 demonstrates the accuracy, efficiency, and effectiveness of the operational-space control algorithms proposed.

2 Conventions and Notation

This section outlines the conventions and notation that will be used to describe the connectivity and dynamics of a floating-base rigid-body system. The conventions in this paper match those used in (Featherstone and Orin 2008) and rely heavily on 6-D spatial vector algebra. The presentation here provides the foundational material required to develop the algorithms in Sections 4 and 5.

2.1 Connectivity

Any legged system can be represented by a series of N_B bodies connected by a set of joints, each with up to 6 degrees of freedom (DoF). The system's motion will be measured with respect to a fixed inertial frame, denoted as Body 0. A privileged body in the system, normally the torso or trunk for legged systems, is selected as a "floating base" and is denoted as Body 1. The remaining bodies are numbered 2 through N_B in any manner such that Body i's predecessor (towards the floating base), denoted $p(i)$, is labeled less than i. An example numbering is provided for a quadruped style topology in Figure 1. Connecting joints are labeled 1 through N_B such that joint i connects Body $p(i)$ to Body i. As shown in Figure 1, Joint 1 is defined as a 6-DoF virtual joint which connects the inertial frame to the floating base.

Given these conventions, a few additional definitions can be made to aid the development of algorithms in subsequent sections. The set $c(i)$ is defined as the set of children for Body i and $c^*(i)$ as the set of bodies in the subtree rooted at Body i, excluding i itself. For example, in Figure 1, $c(1) = \{2, 4, 6, 8\}$, $c^*(1) = \{2, \ldots, 9\}$, and $c^*(4) = \{5\}$. At any instant, denote $\mathcal{C} \subset \{1, \ldots, N_B\}$ as the set of bodies in contact with the ground and denote N_c as the number of bodies in contact. In this work, point contacts are assumed, with the contact points labeled as c_i for each $i \in \mathcal{C}$. The algorithms to be presented are general to handle other constraints, such as those introduced by line or planar contacts, as described in the following sections.

2.2 Spatial Notation

6D spatial vectors provide a compact notation to describe rigid-body dynamics and to develop rigid-body dynamics algorithms. A short introduction is provided here, while the interested reader may refer to (Featherstone 2010a,b) for further details. A coordinate frame will be attached to each body to describe motion conveniently with respect to a local basis. The kinematic relationship between neighboring bodies will be described with the general joint notation of Roberson and Schwertassek (1988). Using this notation, the spatial velocity of link i is related to its predecessor as

$$\mathbf{v}_i = \begin{bmatrix} \boldsymbol{\omega}_i \\ \boldsymbol{v}_i \end{bmatrix} = {}^i\boldsymbol{X}_{p(i)}\mathbf{v}_{p(i)} + \boldsymbol{\Phi}_i\,\dot{\boldsymbol{q}}_i\,, \tag{1}$$

where $\boldsymbol{\omega}_i$ and $\boldsymbol{v}_i$ are the angular and linear velocities of Body i, and $\dot{\boldsymbol{q}}_i$ collects joint i's joint rates[1]. The matrix ${}^i\boldsymbol{X}_{p(i)}$ above provides a transformation of spatial motion vectors from frame $p(i)$ to frame i. $\boldsymbol{\Phi}_i \in \mathbb{R}^{6\times n_i}$ is a full-column-rank matrix that describes joint i's free modes of motion, where n_i is its number of DoFs. This matrix is dependent on joint type, but takes the simplified form $\boldsymbol{\Phi}_i = [0, 0, 1, 0, 0, 0]^T$ for revolute joints following the Denavit-Hartenberg convention. By completing a basis on $\mathbb{R}^6$, joint i's constrained modes of motion are given by $\boldsymbol{\Phi}_i^c$. The total number of degrees of freedom n possessed by the system is given as $n = \sum_{i=1}^{N_B} n_i$.

The spatial transformation matrix ${}^i\boldsymbol{X}_{p(i)}$ in Eq. 1 can be formed from the position vector ${}^{p(i)}\boldsymbol{p}_i$ (from the origin of $p(i)$ to the origin of i) and the rotation matrix ${}^i\boldsymbol{R}_{p(i)}$ which transforms 3D vectors from $p(i)$ coordinates to i coordinates

$${}^i\boldsymbol{X}_{p(i)} = \begin{pmatrix} {}^i\boldsymbol{R}_{p(i)} & 0 \\ {}^i\boldsymbol{R}_{p(i)}\boldsymbol{S}\left({}^{p(i)}\boldsymbol{p}_i\right)^T & {}^i\boldsymbol{R}_{p(i)} \end{pmatrix}. \tag{2}$$

[1] Upright characters $\mathbf{v}$ and $\mathbf{f}$ will be used to represent a spatial velocity and force, respectively. Script characters $\boldsymbol{v}$ and $\boldsymbol{f}$ will be used to denote the linear velocity or force component of a spatial quantity (Featherstone and Orin 2008).

The quantity $\boldsymbol{S}(\boldsymbol{p})$ used here is the skew-symmetric cross product matrix for $\boldsymbol{p}$ which satisfies $\boldsymbol{S}(\boldsymbol{p})\,\boldsymbol{\omega} = \boldsymbol{p} \times \boldsymbol{\omega}$ for any $\boldsymbol{\omega} \in \mathbb{R}^3$. Similarly, the matrix ${}^i\boldsymbol{X}_{p(i)}^T$ provides a spatial transformation of spatial forces from i coordinates to $p(i)$ coordinates.

Body i's 6×6 inertia tensor, $\boldsymbol{I}_i$, maps spatial motion vectors to spatial force vectors and is represented as

$$\boldsymbol{I}_i = \begin{pmatrix} \boldsymbol{I}_i^{\mathrm{cm}} + m_i\boldsymbol{S}(\boldsymbol{c}_i)\boldsymbol{S}(\boldsymbol{c}_i)^T & m_i\boldsymbol{S}(\boldsymbol{c}_i) \\ m_i\boldsymbol{S}(\boldsymbol{c}_i)^T & m_i\mathbf{1} \end{pmatrix}. \tag{3}$$

The quantity $\boldsymbol{c}_i \in \mathbb{R}^3$ is the vector to Body i's center of mass (in i coordinates), m_i is the body's mass, and $\boldsymbol{I}_i^{\mathrm{cm}}$ is the standard 3×3 inertia tensor at the center of mass. An important feature of the spatial inertia tensor is that it allows both Newton's and Euler's rigid-body equations of motion to be incorporated compactly as

$$\mathbf{f}_i^{net} = \boldsymbol{I}_i\,\mathbf{a}_i + \mathbf{v}_i \times^* \boldsymbol{I}_i\,\mathbf{v}_i \tag{4}$$

where $\mathbf{f}_i^{net}$ represents the net spatial force (moment and linear force) on Body i. The spatial motion-force cross product operation $\times^*$ is given from the formula

$$\begin{bmatrix} \boldsymbol{\omega} \\ \boldsymbol{v} \end{bmatrix} \times^* \mathbf{f} = \begin{pmatrix} \boldsymbol{S}(\boldsymbol{\omega}) & \boldsymbol{S}(\boldsymbol{v}) \\ \mathbf{0} & \boldsymbol{S}(\boldsymbol{\omega}) \end{pmatrix} \mathbf{f}. \tag{5}$$

3 Operational-Space Dynamics

This section briefly introduces the quantities that describe operational-space dynamics (Khatib 1987) and operational-space dynamics under constraints (Park and Khatib 2006). Given a rigid-body system, consider the standard dynamic equations of motion (Featherstone and Orin 2008),

$$\boldsymbol{H}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{G}(\boldsymbol{q}) = \hat{\boldsymbol{\tau}}\,, \tag{6}$$

where $\boldsymbol{H} \in \mathbb{R}^{n\times n}$, $\boldsymbol{C} \in \mathbb{R}^n$, and $\boldsymbol{G} \in \mathbb{R}^n$ are the familiar mass matrix, velocity product term, and gravitational term, respectively. Here, the generalized force $\hat{\boldsymbol{\tau}} \in \mathbb{R}^n$ has contributions from ground contact forces $\boldsymbol{F}_c \in \mathbb{R}^{3N_c}$, actuated joint torques $\boldsymbol{\tau} \in \mathbb{R}^{n_a}$, as well a spatial force applied directly on the torso $\mathbf{f}_t \in \mathbb{R}^6$. In this case:

$$\hat{\boldsymbol{\tau}} = \boldsymbol{J}_c^T\boldsymbol{F}_c + \boldsymbol{S}^T\boldsymbol{\tau} + \boldsymbol{J}_t^T\mathbf{f}_t \tag{7}$$

where $\boldsymbol{S} \in \mathbb{R}^{n_a\times n}$ encodes the system's actuation, and $\boldsymbol{J}_c \in \mathbb{R}^{3N_c\times n}$ is the combined contact Jacobian. The torso Jacobian $\boldsymbol{J}_t = [\mathbf{1}_{6\times 6}\ \mathbf{0}_{6\times(n-6)}]$ is a simple selector matrix.

3.1 Task Dynamics

In this work, the floating-base position and orientation are selected as the desired task, with the task velocity given as $\dot{\boldsymbol{x}}_t = \begin{bmatrix}\boldsymbol{\omega}_1^T & \dot{\boldsymbol{p}}_1^T\end{bmatrix}^T$. The torso Jacobian $\boldsymbol{J}_t \in \mathbb{R}^{6\times n}$ provides the common relationship

$$\dot{\boldsymbol{x}}_t = \boldsymbol{J}_t(\boldsymbol{q})\dot{\boldsymbol{q}}\,. \tag{8}$$

The standard operational-space dynamic equations of motion (Khatib 1987) are then given as

$$\boldsymbol{\Lambda}_t(\boldsymbol{q})\,\ddot{\boldsymbol{x}}_t + \boldsymbol{\mu}_t(\boldsymbol{q},\dot{\boldsymbol{q}}) + \boldsymbol{\rho}_t(\boldsymbol{q}) = \hat{\boldsymbol{F}}_t\,, \tag{9}$$

where $\boldsymbol{\Lambda}_t(\boldsymbol{q})$ is the operational-space inertia matrix, $\boldsymbol{\mu}_t(\boldsymbol{q},\dot{\boldsymbol{q}})$ is a velocity-dependent force bias, and $\boldsymbol{\rho}_t(\boldsymbol{q})$ is a gravity-dependent force bias. $\hat{\boldsymbol{F}}_t$ is the operational force vector, analogous to $\hat{\boldsymbol{\tau}}$, that includes the effects of actuated joint torques and external forces. These quantities are given by

$$\boldsymbol{\Lambda}_t(\boldsymbol{q}) = \left(\boldsymbol{J}_t\,\boldsymbol{H}^{-1}\,\boldsymbol{J}_t^T\right)^{-1} \tag{10}$$

$$\bar{\boldsymbol{J}}_t^T(\boldsymbol{q}) = \boldsymbol{\Lambda}_t\,\boldsymbol{J}_t\,\boldsymbol{H}^{-1} \tag{11}$$

$$\boldsymbol{\mu}_t(\boldsymbol{q},\dot{\boldsymbol{q}}) = \bar{\boldsymbol{J}}_t^T\boldsymbol{C} - \boldsymbol{\Lambda}_t\,\dot{\boldsymbol{J}}_t\,\dot{\boldsymbol{q}} \tag{12}$$

$$\boldsymbol{\rho}_t(\boldsymbol{q}) = \bar{\boldsymbol{J}}_t^T\boldsymbol{G} \tag{13}$$

$$\hat{\boldsymbol{F}}_t = \bar{\boldsymbol{J}}_t^T\hat{\boldsymbol{\tau}} \tag{14}$$

$$= \bar{\boldsymbol{J}}_t^T\boldsymbol{J}_c^T\boldsymbol{F}_c + \bar{\boldsymbol{J}}_t^T\boldsymbol{S}^T\boldsymbol{\tau} + \mathbf{f}_t\,. \tag{15}$$

3.2 Task Dynamics with Contact Constraints

When portions of the system are in hard contact with the environment, the actuated joint torques and other external forces directly affect the ground reaction forces. The derivation of the task dynamics in this case is repeated from Park and Khatib (2006) to illustrate the effect of these hard contacts. Given a collection of N_c contact points, let their collective positions be given by $\boldsymbol{x}_c \in \mathbb{R}^{3N_c}$, with Jacobian $\boldsymbol{J}_c$. The assumption that each of the contact points is stationary provides

$$\begin{aligned}\dot{\boldsymbol{x}}_c &= \boldsymbol{J}_c\,\dot{\boldsymbol{q}} = \mathbf{0}\\ \ddot{\boldsymbol{x}}_c &= \boldsymbol{J}_c\,\ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}_c\,\dot{\boldsymbol{q}} = \mathbf{0}\,.\end{aligned} \tag{16}$$

In this case, the contact forces can be determined as a function of the the remainder of the forces on the system. Through examination of the operational-space equations at the contacts it can be shown that

$$\boldsymbol{F}_c = \boldsymbol{\mu}_c + \boldsymbol{\rho}_c - \bar{\boldsymbol{J}}_c^T\left(\boldsymbol{S}^T\boldsymbol{\tau} + \boldsymbol{J}_t^T\mathbf{f}_t\right). \tag{17}$$

Here $\bar{\boldsymbol{J}}_c^T$, $\boldsymbol{\mu}_c$, and $\boldsymbol{\rho}_c$ arise from the operational-space dynamics for the contact points, and can be obtained through suitable modification of Eqs. 11-13. Defining $\boldsymbol{N}_c^T = \mathbf{1} - \boldsymbol{J}_c^T\,\bar{\boldsymbol{J}}_c^T$ and substituting Eq. 17 into Eqs. 6 and 7, the constrained system dynamics are then given by

$$\begin{aligned}\boldsymbol{H}\ddot{\boldsymbol{q}} + \boldsymbol{N}_c^T\boldsymbol{C} + \boldsymbol{N}_c^T\boldsymbol{G} + \boldsymbol{J}_c^T\boldsymbol{\Lambda}_c\,\dot{\boldsymbol{J}}_c\,\dot{\boldsymbol{q}}\\ = \boldsymbol{N}_c^T\boldsymbol{S}^T\boldsymbol{\tau} + \boldsymbol{N}_c^T\boldsymbol{J}_t^T\mathbf{f}_t\,.\end{aligned} \tag{18}$$

Due to the existence of closed kinematic loops in the connectivity of the system, these equations may also be referred to as the closed-chain system dynamics.

An analogous derivation to that for Eq. 9 provides the constrained operational-space equations of motion

$$\tilde{\boldsymbol{\Lambda}}_t\,\ddot{\boldsymbol{x}}_t + \tilde{\boldsymbol{\mu}}_t + \tilde{\boldsymbol{\rho}}_t = \tilde{\bar{\boldsymbol{J}}}_t^T\boldsymbol{S}^T\boldsymbol{\tau} + \mathbf{f}_t, \tag{19}$$

where

$$\tilde{\boldsymbol{\Lambda}}_t(\boldsymbol{q}) = \left(\boldsymbol{J}_t\boldsymbol{H}^{-1}\boldsymbol{N}_c^T\boldsymbol{J}_t^T\right)^{-1} \tag{20}$$

$$= \left(\boldsymbol{J}_t\boldsymbol{N}_c\boldsymbol{H}^{-1}\boldsymbol{N}_c^T\boldsymbol{J}_t^T\right)^{-1} \tag{21}$$

$$\tilde{\bar{\boldsymbol{J}}}_t^T = \tilde{\boldsymbol{\Lambda}}_t\,\boldsymbol{J}_t\,\boldsymbol{H}^{-1}\,\boldsymbol{N}_c^T \tag{22}$$

$$= \tilde{\boldsymbol{\Lambda}}_t\,\boldsymbol{J}_t\,\boldsymbol{N}_c\,\boldsymbol{H}^{-1} \tag{23}$$

$$= \overline{\boldsymbol{J}_t\,\boldsymbol{N}_c}^T \tag{24}$$

$$\begin{aligned}\tilde{\boldsymbol{\mu}}_t(\boldsymbol{q},\dot{\boldsymbol{q}}) &= \tilde{\bar{\boldsymbol{J}}}_t^T\boldsymbol{C} + \tilde{\boldsymbol{\Lambda}}_t\,\boldsymbol{J}_t\,\boldsymbol{H}^{-1}\,\boldsymbol{J}_c^T\,\boldsymbol{\Lambda}_c\,\dot{\boldsymbol{J}}_c\,\dot{\boldsymbol{q}}\\ &= -\tilde{\boldsymbol{\Lambda}}_t\,\dot{\boldsymbol{J}}_t\,\dot{\boldsymbol{q}}\end{aligned} \tag{25}$$

$$\tilde{\boldsymbol{\rho}}_t(\boldsymbol{q}) = \tilde{\bar{\boldsymbol{J}}}_t^T\boldsymbol{G}\,. \tag{26}$$

Throughout these equations, the constraint-consistent projection identities

$$\boldsymbol{N}_c\,\boldsymbol{H}^{-1} = \boldsymbol{H}^{-1}\,\boldsymbol{N}_c^T = \boldsymbol{N}_c\,\boldsymbol{H}^{-1}\,\boldsymbol{N}_c^T$$

are used to bring them to recognized forms. While $\boldsymbol{\Lambda}_t$ in Eq. 9 is an inertia felt by a force at the torso with the contacts free to accelerate, $\tilde{\boldsymbol{\Lambda}}_t$ is the inertia felt by a force at the torso given that the contacts are pinned at their current location.

4 Recursive Algorithms for Floating-Base Task Control

This section provides the main contribution of the paper, recursive algorithms for floating-base task control. Control of a main element, such as the torso, is important to locomotion control for a variety of reasons. Maintaining proper pose of the torso is required to provide the legs access to desired footholds. At a high level, the task of legged locomotion is generally to employ available appendages to move the torso in a desired direction. The trot results in the following sections show that control of the torso during stance is sufficient to provide stable locomotion when combined with simple foot placement heuristics.

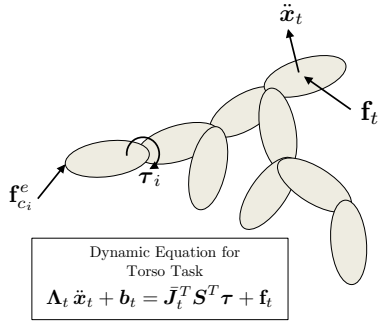

Fig. 2 Dynamic equation for the torso task with ground reaction forces measured at contacts. The bias force $\boldsymbol{b}_t$ includes the dynamic effects for each of the contact forces $\mathbf{f}^e_{c_i}$ for $i \in \mathcal{C}$. This term also contains gravitational forces and velocity-dependent Coriolis and centripetal forces. The columns of the matrix $\bar{\boldsymbol{J}}_t^T$ describe the effective torso forces created by each actuated joint torque $\boldsymbol{\tau}_i$.

4.1 Torso Control with Ground Force Sensing

In this section, it is assumed that the legged system is equipped with contact sensors at each of its foot contact points. It is assumed that these forces are instantaneously fixed, which is in contrast to the closed-chain case wherein contact forces are determined by the effects in Eq. 17. The contact sensors measure the spatial ground reaction force

$$\mathbf{f}^e_{c_i} = \begin{bmatrix} \boldsymbol{n}^{e\,T}_{c_i} & \boldsymbol{f}^{e\,T}_{c_i} \end{bmatrix}^T ,$$

where $\boldsymbol{n}^e_{c_i} \in \mathbb{R}^3$ and $\boldsymbol{f}^e_{c_i} \in \mathbb{R}^3$ represent the external moment and force applied from the earth *onto* Body i at its contact (measured in a local coordinate frame at c_i). Under the assumption of point contacts at the feet, $\boldsymbol{n}^e_{c_i} = \mathbf{0}$ in all cases. As a result, the pure forces $\boldsymbol{f}^e_{c_i}$ are concatenated to form $\boldsymbol{F}_c$. Defining

$$\boldsymbol{b}_t = \boldsymbol{\mu}_t + \boldsymbol{\rho}_t - \bar{\boldsymbol{J}}_t^T \boldsymbol{J}_c^T \boldsymbol{F}_c \tag{27}$$

provides the operational-space dynamics (Eq. 9) at the floating base as

$$\boldsymbol{\Lambda}_t \ddot{\boldsymbol{x}}_t + \boldsymbol{b}_t = \bar{\boldsymbol{J}}_t^T \boldsymbol{S}^T \boldsymbol{\tau} + \mathbf{f}_t \, . \tag{28}$$

The setting for these equations is described in Fig. 2. This section will provide a recursive algorithm to compute $\bar{\boldsymbol{J}}_t^T$, $\boldsymbol{\Lambda}_t$, and $\boldsymbol{b}_t$ from knowledge of $\boldsymbol{F}_c$, $\boldsymbol{q}$, and $\dot{\boldsymbol{q}}$. This represents the first time a recursive algorithm has been used to compute the quantity $\bar{\boldsymbol{J}}^T$ directly for a tree-structure system. The algorithm to compute $\bar{\boldsymbol{J}}_t^T$ is an $O(nd)$ algorithm, where d is the maximum depth of the connectivity tree. This represents a substantial order improvement over the $O(n^3)$ approach when Eq. 11 is used for closed-form numeric computation. For the quadruped example with $n = 18$ and $d = 4$, the recursive algorithm is an order of magnitude faster.

The approach presented here is inspired by the efficient recursions of the articulated-body algorithm (ABA) (Featherstone 1983; Featherstone and Orin 2008). The ABA proceeds with three algorithmic sweeps over the kinematic tree. The first pass of the algorithm proceeds outward from base to tips, and calculates body velocities $\mathbf{v}_i$ and associated velocity-dependent acceleration bias terms $\boldsymbol{\zeta}_i$ which satisfy

$$\mathbf{a}_i = {}^i\boldsymbol{X}_{p(i)} \, \mathbf{a}_{p(i)} + \boldsymbol{\Phi}_i \, \ddot{\boldsymbol{q}}_i + \boldsymbol{\zeta}_i \, . \tag{29}$$

The second pass of the algorithm, inward from tips to base, then computes articulated-body inertias $\boldsymbol{I}^A_i$ and bias forces $\mathbf{p}^A_i$. Considering the subtree rooted at Body i, the key concept of the ABA is that the dynamics of this subtree can be represented as (Featherstone and Orin 2008)

$$\mathbf{f}_i = \boldsymbol{I}^A_i \, \mathbf{a}_i + \mathbf{p}^A_i \, , \tag{30}$$

where $\mathbf{f}_i$ is the interaction force transmitted to Body i from its predecessor.

Intuitively, $\boldsymbol{I}^A_i$ is the inertia felt by a force acting on the subtree rooted at i and, as such, is an operational-space inertia for this subtree. To see this more rigorously, note that the subtree rooted at i is itself a rigid-body system, and thus operational-space equations of motion, in the form of Eq. 9, can be derived with the position and orientation of Body i taken as the task. The equivalence between $\boldsymbol{\Lambda}$ in this case and $\boldsymbol{I}^A_i$ can be observed as both quantities map accelerations of Body i to the forces required for motion. This correspondence is not as clear for $\mathbf{p}^A_i$ however, as this bias force includes the combined effects of external forces, joint torques, and velocity-dependent bias forces. A new inward pass developed here provides an efficient set of computations to relate the individual bias forces to each joint torque, which allows ABA-inspired recursions to be leveraged to compute the quantities $\boldsymbol{b}_t$ and $\bar{\boldsymbol{J}}_t^T$ in Eq. 28.

The third and final recursion of the ABA, once again outward across the tree, calculates body accelerations. This final pass is not used here, as the new algorithm recovers Eq. 28 after the second pass.

The new inward pass begins through consideration of Eq. 4, the dynamic force balance equation for a single rigid body, in a more detailed form

$$\mathbf{f}_i = \boldsymbol{I}_i \, \mathbf{a}_i + \boldsymbol{\beta}_i + \sum_{j \in c(i)} {}^j\boldsymbol{X}_i^T \, \mathbf{f}_j \, . \tag{31}$$

Here $\boldsymbol{\beta}_i = \mathbf{v}_i \times^* \boldsymbol{I}_i \mathbf{v}_i - \mathbf{f}^e_i$ has contributions from a velocity-dependent bias force and external force $\mathbf{f}^e_i$ defined by

$$\mathbf{f}^e_i = \begin{cases} -{}^{c_i}\boldsymbol{X}_i^T \mathbf{f}^e_{c_i}, & \text{if } i \in \mathcal{C} \\ \mathbf{0}, & \text{otherwise} \, . \end{cases}$$

It will be shown that the interaction forces $\mathbf{f}_j$, which act on each child of Body i, can be sequentially eliminated from the more general force balance equation for each body

$$\mathbf{f}_i = \boldsymbol{I}^A_i \, \mathbf{a}_i + \boldsymbol{\beta}^A_i + \sum_{j \in s(i)} {}^j\boldsymbol{X}_i^T \, \mathbf{f}_j \; - \sum_{k \in c^*(i)} \boldsymbol{B}_{ik} \, \boldsymbol{\tau}_k \, , \tag{32}$$

where initially $\boldsymbol{I}_i^A = \boldsymbol{I}_i$, $s(i) = c(i)$, $\boldsymbol{\beta}_i^A = \boldsymbol{\beta}_i$, and each $\boldsymbol{B}_{ik} = \mathbf{0}$. This provides an exact match with Eq. 31 at the start of the algorithm. At each step of the algorithm, the set $s(i)$ contains the uneliminated forces between Body i and each of its children, and $\boldsymbol{B}_{ik}$ describes the coupled force effect that torque k, in the subtree of Body i, has on Body i. Note that in the ABA algorithm for forward dynamics, the joint torques are known so that the last term in Eq. 32 is absorbed into the bias force $\boldsymbol{\beta}_i^A$. Here in order to calculate the dynamically consistent Jacobian pseudoinverse $\bar{\boldsymbol{J}}$, which gives the relationship between the torques and the spatial force on the torso, the torques are *not* known and must be carried along through the last term in the general force balance equation (Eq. 32). The torque-force coupling terms $\boldsymbol{B}_{ik}$ are new in this algorithm and are key to constructing $\bar{\boldsymbol{J}}$. The second (inward) pass of the algorithm here can be used to eliminate each of the interaction forces $\mathbf{f}_j$ in Eq. 32 similar to the ABA, and, as such, recursively provides updates for $\boldsymbol{I}_i^A$, $\boldsymbol{\beta}_i^A$, and in this case $\boldsymbol{B}_{ik}$.

The process to eliminate an interaction force can occur at any Body i with $s(i) = \{\}$. This property holds for at least one body at the beginning of the algorithm, since it holds at any leaf. At any such body, it is possible to compute $\mathbf{f}_i$ as a function of $\mathbf{a}_{p(i)}$, similar to the derivation of the ABA. This process requires finding $\ddot{\boldsymbol{q}}_i$ first as a function of $\mathbf{a}_{p(i)}$. By substituting $\mathbf{a}_i$ from Eq. 29 into Eq. 32 and multiplying both sides by $\boldsymbol{\Phi}_i^T$, the following is obtained

$$\begin{aligned}\boldsymbol{\tau}_i &= \boldsymbol{\Phi}_i^T \mathbf{f}_i \qquad (33)\\ &= \boldsymbol{\Phi}_i^T \left[\boldsymbol{I}_i^A \left({}^i\boldsymbol{X}_{p(i)}\mathbf{a}_{p(i)} + \boldsymbol{\Phi}_i\ddot{\boldsymbol{q}}_i + \boldsymbol{\zeta}_i\right) + \boldsymbol{\beta}_i^A\right] \\ &\qquad - \sum_{k\in c^*(i)} \boldsymbol{\Phi}_i^T \boldsymbol{B}_{ik}\boldsymbol{\tau}_k \qquad (34)\end{aligned}$$

which uses the fact that $\boldsymbol{\tau}_i$ is equal to the component(s) of the spatial force applied at joint i along its free mode(s) of motion. The fact that $s(i) = \{\}$ at this link allows for a solution of $\ddot{\boldsymbol{q}}_i$ that is not dependent on interaction forces. The $\ddot{\boldsymbol{q}}_i$ that satisfies Eq. 34 can be inserted into Eq. 29 to obtain $\mathbf{a}_i$ as

$$\begin{aligned}\mathbf{a}_i =& {}^i\boldsymbol{X}_{p(i)}\mathbf{a}_{p(i)} + \boldsymbol{\Phi}_i\boldsymbol{D}_i\boldsymbol{\tau}_i + \boldsymbol{\zeta}_i \\ &- \boldsymbol{K}_i\left[\boldsymbol{I}_i^A({}^i\boldsymbol{X}_{p(i)}\mathbf{a}_{p(i)} + \boldsymbol{\zeta}_i) + \boldsymbol{\beta}_i^A\right] \\ &+ \sum_{k\in c^*(i)} \boldsymbol{K}_i\boldsymbol{B}_{ik}\boldsymbol{\tau}_k\,, \qquad (35)\end{aligned}$$

where $\boldsymbol{D}_i = (\boldsymbol{\Phi}_i^T\boldsymbol{I}_i^A\boldsymbol{\Phi}_i)^{-1}$ and $\boldsymbol{K}_i = \boldsymbol{\Phi}_i\boldsymbol{D}_i\boldsymbol{\Phi}_i^T$. This result is finally combined with Eq. 32 for Body i, which results in $\mathbf{f}_i$ taking the form

$$\begin{aligned}\mathbf{f}_i = &\boldsymbol{L}_i^T\boldsymbol{I}_i^A{}^i\boldsymbol{X}_{p(i)}\,\mathbf{a}_{p(i)} + \boldsymbol{L}_i^T\left(\boldsymbol{\beta}_i^A + \boldsymbol{I}_i^A\boldsymbol{\zeta}_i\right) \\ &+ \boldsymbol{I}_i^A\boldsymbol{\Phi}_i\boldsymbol{D}_i\boldsymbol{\tau}_i - \sum_{k\in c^*(i)} \boldsymbol{L}_i^T\boldsymbol{B}_{ik}\boldsymbol{\tau}_k\,. \qquad (36)\end{aligned}$$

Each $\boldsymbol{L}_i^T$ is a force propagator across the i-th joint

$$\boldsymbol{L}_i^T = \mathbf{1}_{6\times 6} - \boldsymbol{I}_i^A\boldsymbol{K}_i\,. \qquad (37)$$

At this point, $\mathbf{f}_i$ from Eq. 36 can be substituted into the force balance equation (Eq. 32) for Body $p(i)$. Through this substitution, the interaction force $\mathbf{f}_i$ will no longer appear in the force balance for Body $p(i)$. Further, the following updates, which account for the dynamic coupling between bodies $p(i)$ and i, allow Eq. 32 to continue to hold

$$\begin{aligned}\boldsymbol{I}_{p(i)}^A &:= \boldsymbol{I}_{p(i)}^A + {}^i\boldsymbol{X}_{p(i)}^T\boldsymbol{L}_i^T\boldsymbol{I}_i^A{}^i\boldsymbol{X}_{p(i)} \\ \boldsymbol{\beta}_{p(i)}^A &:= \boldsymbol{\beta}_{p(i)}^A + {}^i\boldsymbol{X}_{p(i)}^T\boldsymbol{L}_i^T\left(\boldsymbol{\beta}_i^A + \boldsymbol{I}_i^A\boldsymbol{\zeta}_i\right) \\ \boldsymbol{B}_{p(i)i} &:= -{}^i\boldsymbol{X}_{p(i)}^T\boldsymbol{I}_i^A\boldsymbol{\Phi}_i\boldsymbol{D}_i \\ \boldsymbol{B}_{p(i)k} &:= {}^i\boldsymbol{X}_{p(i)}^T\boldsymbol{L}_i^T\boldsymbol{B}_{ik} \quad \forall k \in c^*(i) \\ s\left(p(i)\right) &:= s\left(p(i)\right)\setminus\{i\}\,.\end{aligned}$$

By processing these updates starting from the highest numbered body and counting downward, it follows when Body i is reached, $s(i)$ is necessarily empty (since each child is numbered higher than its parent). Once this backwards recursion process has completed, Eq. 32 for Body 1 (the torso) provides

$$\boldsymbol{I}_1^A\,\mathbf{a}_1 + \boldsymbol{\beta}_1^A = \sum_{k\in c^*(1)} \boldsymbol{B}_{1k}\,\boldsymbol{\tau}_k + \mathbf{f}_1 \qquad (38)$$

which closely resembles Eq. 28. At this point, $\mathbf{f}_1 = \mathbf{f}_t$ and $\boldsymbol{I}_1^A = \boldsymbol{\Lambda}_t$. Likewise, each $\boldsymbol{B}_{1k}$ provides the column(s) of $\bar{\boldsymbol{J}}_t^T$ which correspond(s) to joint k.

Two small differences must be accounted for to recover $\boldsymbol{b}_t$ from $\boldsymbol{\beta}_1^A$. These differences come from two sources: the neglect of gravitational forces thus far, and the fact that $\mathbf{a}_1$ is a spatial acceleration which is not equivalent in general to the conventional acceleration vector $[\dot{\boldsymbol{\omega}}_1^T\ \ddot{\boldsymbol{p}}_1^T]^T$ (Featherstone 2001). To handle the former, a common approach is taken by biasing all accelerations within the algorithm opposite that of gravitational acceleration (Featherstone and Orin 2008). The difference between spatial and conventional velocity is handled with a common formula (Featherstone 2001). These modifications provide $\mathbf{a}_1$ in terms of $\ddot{\boldsymbol{x}}_t$ as

$$\mathbf{a}_1 = \ddot{\boldsymbol{x}}_t - {}^1\mathbf{a}_g - \begin{bmatrix}\mathbf{0}\\ \boldsymbol{\omega}_1\times\boldsymbol{v}_1\end{bmatrix}\,, \qquad (39)$$

where ${}^1\mathbf{a}_g$ is the gravitational acceleration vector expressed in torso coordinates. These substitutions provide

$$\boldsymbol{I}_1^A\ddot{\boldsymbol{x}}_t + \boldsymbol{b}_t = \sum_{k\in c^*(1)} \boldsymbol{B}_{1k}\,\boldsymbol{\tau}_k + \mathbf{f}_1\,, \qquad (40)$$

where

$$\boldsymbol{b}_t = \boldsymbol{\beta}_1^A - \boldsymbol{I}_1^A\left({}^1\mathbf{a}_g + \begin{bmatrix}\mathbf{0}\\ \boldsymbol{\omega}_1\times\boldsymbol{v}_1\end{bmatrix}\right)\,. \qquad (41)$$

This final correction recovers Eq. 28, with the complete algorithm provided in Table 1.

4.2 Torso Control without Ground Force Sensing

This section extends the previous algorithm to relax the requirement of sensing contact forces $\mathbf{f}^e_{c_i}$. As described in Section 3.2, when contact points are constrained, the system has closed kinematic chains, and the contact forces are determined by the remainder of the forces in the system. Here, the effects of these contact forces are determined recursively, by considering the *constrained* dynamics of articulated subtrees. The new closed-chain operational-space dynamics algorithm is shown to result in comparable control results to the case when contact force sensing is required.

4.2.1 Constraint Approximation

Consider a constraint point c_i, where the interface between Body i and the ground is modeled through a virtual spherical joint as shown in Fig. 3. Other types of contacts may be considered by designing the joint at c_i such that its constrained modes of motion correspond to the contact constraints. It is assumed that contact point c_i is stationary which requires

$$\mathbf{a}_{c_i} = -{}^{c_i}\mathbf{a}_g \tag{42}$$

in the algorithm due to its acceleration bias. Instead of enforcing this constraint exactly, it can approximated by

$$\mathbf{f}_{c_i} = \boldsymbol{I}_{c_i}(\mathbf{a}_{c_i} + {}^{c_i}\mathbf{a}_g)\,, \tag{43}$$

where $\boldsymbol{I}_{c_i}$ represents the inertia of the ground at contact c_i, and $\mathbf{f}_{c_i}$ is the force *exerted on the ground* by the rest of the system. By selecting $\boldsymbol{I}_{c_i}$ as a very massive inertia, the exact constraint in Eq. 42 can be approximately enforced. As a result, this approximation is called a big-ground-inertia approximation. The accuracy of this approximation with finite ground inertia is demonstrated in Section 6, while Appendix A proves that this approximation admits exact replication of the constrained dynamics in the limit as the contact inertia becomes infinite. A similar approximation method has been shown to be effective for the computation of the operational-space inertia matrix $\boldsymbol{\Lambda}$ for manipulators by Lilly and Orin (1993) wherein velocity and torque effects were not considered. Finally, since the virtual joint at c_i is unactuated, this provides

$$\boldsymbol{\Phi}^T_{c_i}\mathbf{f}_{c_i} = \mathbf{0}. \tag{44}$$

4.2.2 Algorithm Modifications

Propagation of the dynamic equations from contact c_i back to Body i can be derived similar to that described in the previous section. For each Body i in contact, this provides the modified dynamic equations of motion as

$$\begin{aligned}\mathbf{f}_i = (\boldsymbol{I}_i + {}^{c_i}\boldsymbol{X}_i^T\boldsymbol{L}_{c_i}^T\boldsymbol{I}_{c_i}{}^{c_i}\boldsymbol{X}_i)\,\mathbf{a}_i + \\ \boldsymbol{\beta}_i + {}^{c_i}\boldsymbol{X}_i^T\boldsymbol{L}_{c_i}^T\boldsymbol{I}_{c_i}{}^{c_i}\mathbf{a}_g\,,\end{aligned} \tag{45}$$

$\mathbf{v}_0 = \mathbf{0}$
${}^0\boldsymbol{X}_0 = \mathbf{1}$
for $i = 1$ **to** N_B **do**
  Compute ${}^i\boldsymbol{X}_{p(i)}(\boldsymbol{q}_i)$ and $\boldsymbol{\Phi}_i(\boldsymbol{q}_i)$
    [see (Featherstone and Orin 2008)]
  ${}^i\boldsymbol{X}_0 = {}^i\boldsymbol{X}_{p(i)}\,{}^{p(i)}\boldsymbol{X}_0$
  $\mathbf{v}_i = {}^i\boldsymbol{X}_{p(i)}\mathbf{v}_{p(i)} + \boldsymbol{\Phi}_i\dot{\boldsymbol{q}}_i$
  Compute $\boldsymbol{\zeta}_i(\boldsymbol{q}_i, \dot{\boldsymbol{q}}_i, \mathbf{v}_i)$ [see (Featherstone and Orin 2008)]
  $\boldsymbol{\beta}_i^A = \mathbf{v}_i \times^* \boldsymbol{I}_i\mathbf{v}_i - \mathbf{f}_i^e$
  $\boldsymbol{I}_i^A = \boldsymbol{I}_i$
end for

for $i = N_B$ **to** 1 **do**
  $\boldsymbol{D}_i = (\boldsymbol{\Phi}_i^T\boldsymbol{I}_i^A\boldsymbol{\Phi}_i)^{-1}$
  $\boldsymbol{K}_i = \boldsymbol{\Phi}_i\,\boldsymbol{D}_i\,\boldsymbol{\Phi}_i^T$
  $\boldsymbol{L}_i^T = \mathbf{1}_{6\times6} - \boldsymbol{I}_i^A\boldsymbol{K}_i$
  if $p(i) \neq 0$ **then**
    $\boldsymbol{I}_{p(i)}^A = \boldsymbol{I}_{p(i)}^A + {}^i\boldsymbol{X}_{p(i)}^T\,\boldsymbol{L}_i^T\,\boldsymbol{I}_i^A\,{}^i\boldsymbol{X}_{p(i)}$
    $\boldsymbol{\beta}_{p(i)}^A := \boldsymbol{\beta}_{p(i)}^A + {}^i\boldsymbol{X}_{p(i)}^T\,\boldsymbol{L}_i^T\,(\boldsymbol{\beta}_i^A + \boldsymbol{I}_i^A\boldsymbol{\zeta}_i)$
    $\boldsymbol{B}_{p(i)i} := -{}^i\boldsymbol{X}_{p(i)}^T\,\boldsymbol{I}_i^A\,\boldsymbol{\Phi}_i\,\boldsymbol{D}_i$
    for all $k \in c^*(i)$ **do**
      $\boldsymbol{B}_{p(i)k} := {}^i\boldsymbol{X}_{p(i)}^T\,\boldsymbol{L}_i^T\,\boldsymbol{B}_{ik}$
    end for k
  end
end for i

$\boldsymbol{\Lambda}_t = \boldsymbol{I}_1^A$
$\bar{\boldsymbol{J}}_t^T = \begin{bmatrix}\mathbf{1} & \boldsymbol{B}_{12} & \cdots & \boldsymbol{B}_{1N_B}\end{bmatrix}$
$\boldsymbol{b}_t = \boldsymbol{\beta}_1^A - \boldsymbol{I}_1^A\left({}^1\boldsymbol{X}_0\,{}^0\mathbf{a}_g + \begin{bmatrix}\mathbf{0}\\ \boldsymbol{\omega}_1 \times \boldsymbol{v}_1\end{bmatrix}\right)$

Table 1 Recursive Operational-Space Dynamics Algorithm for a Floating-Base Task

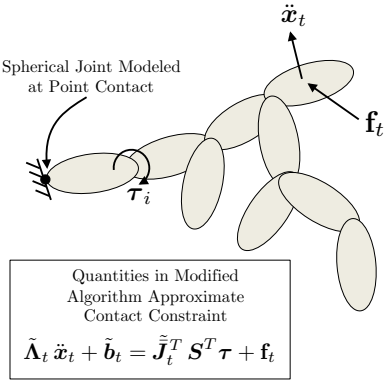

Fig. 3 Dynamic equation for the torso task with constraints modeled at contacts. The constraint-modified quantities $\tilde{\boldsymbol{b}}_t$, $\tilde{\boldsymbol{\Lambda}}_t$, and $\tilde{\bar{\boldsymbol{J}}}_t^T$ include the dynamic effects of the contact force without its explicit specification. The columns of the matrix $\tilde{\bar{\boldsymbol{J}}}_t^T$ describe the effective torso forces created by each actuated joint torque $\boldsymbol{\tau}_i$.

Additional loop following outward recursion:
for $i \in \mathcal{C}$ **do**
Compute ${}^{c_i}\boldsymbol{X}_i$
${}^{c_i}\boldsymbol{X}_0 = {}^{c_i}\boldsymbol{X}_i \, {}^{i}\boldsymbol{X}_0$
$\boldsymbol{K}_{c_i} = \boldsymbol{\Phi}_{c_i} \left(\boldsymbol{\Phi}_{c_i}^T \boldsymbol{I}_{c_i} \boldsymbol{\Phi}_{c_i} \right)^{-1} \boldsymbol{\Phi}_{c_i}^T$
$\boldsymbol{L}_{c_i}^T = \mathbf{1}_{6\times 6} - \boldsymbol{I}_{c_i} \boldsymbol{K}_{c_i}$
$\tilde{\boldsymbol{\beta}}_i^A = \mathbf{v}_i \times^* \boldsymbol{I}_i \mathbf{v}_i + {}^{c_i}\boldsymbol{X}_i^T \boldsymbol{L}_{c_i}^T \boldsymbol{I}_{c_i} \, {}^{c_i}\boldsymbol{X}_0 \, {}^0\mathbf{a}_g$
$\tilde{\boldsymbol{I}}_i^A = \boldsymbol{I}_i + {}^{c_i}\boldsymbol{X}_i^T \boldsymbol{L}_{c_i}^T \boldsymbol{I}_{c_i} \, {}^{c_i}\boldsymbol{X}_i$
end for

Modified Output:
$\tilde{\boldsymbol{\Lambda}}_t = \tilde{\boldsymbol{I}}_1^A$
$\tilde{\boldsymbol{J}}_t^T = \begin{bmatrix} \mathbf{1} & \tilde{\boldsymbol{B}}_{12} & \cdots & \tilde{\boldsymbol{B}}_{1N_B} \end{bmatrix}$
$\tilde{\boldsymbol{b}}_t = \tilde{\boldsymbol{\beta}}_1^A - \tilde{\boldsymbol{I}}_1^A \left({}^1\boldsymbol{X}_0 \, {}^0\mathbf{a}_g + \begin{bmatrix} \mathbf{0} \\ \boldsymbol{\omega}_1 \times \boldsymbol{v}_1 \end{bmatrix} \right)$

Table 2 Modifications to Table 1 for the Closed-Chain Operational-Space Dynamics Algorithm

where $\boldsymbol{L}_{c_i}^T$ is defined as previously. To differentiate from the non-constrained case, *constraint-modified* articulated inertias and articulated bias forces $\tilde{\boldsymbol{I}}_i^A$ and $\tilde{\boldsymbol{\beta}}_i^A$ are initialized as

$$\tilde{\boldsymbol{I}}_i^A = \boldsymbol{I}_i + {}^{c_i}\boldsymbol{X}_i^T \boldsymbol{L}_{c_i}^T \boldsymbol{I}_{c_i} \, {}^{c_i}\boldsymbol{X}_i \tag{46}$$

and

$$\tilde{\boldsymbol{\beta}}_i^A = \mathbf{v}_i \times^* \boldsymbol{I}_i \mathbf{v}_i + {}^{c_i}\boldsymbol{X}_i^T \boldsymbol{L}_{c_i}^T \boldsymbol{I}_{c_i} \, {}^{c_i}\mathbf{a}_g \tag{47}$$

at bodies in contact. With this definition, all other recursions shown in Table 1 are valid, provided that $\boldsymbol{I}_i^A$, $\boldsymbol{\beta}_i^A$, and $\boldsymbol{B}_{ik}$ are replaced with constrained counterparts $\tilde{\boldsymbol{I}}_i^A$, $\tilde{\boldsymbol{\beta}}_i^A$, and $\tilde{\boldsymbol{B}}_{ik}$, respectively. Keeping this in mind, the original algorithm in Table 1 is modified through addition of a loop immediately following the outward recursion, shown in Table 2, to correctly initialize the dynamic equations for the bodies in contact. Upon completing the inward recursion of the algorithm, the listed modifications approximate the contact constrained task dynamic equations of motion

$$\tilde{\boldsymbol{\Lambda}}_t \, \ddot{\boldsymbol{x}}_t + \tilde{\boldsymbol{b}}_t = \tilde{\boldsymbol{J}}_t^T \boldsymbol{S}^T \boldsymbol{\tau} + \mathbf{f}_t \tag{48}$$

where $\tilde{\boldsymbol{b}}_t = \tilde{\boldsymbol{\mu}}_t + \tilde{\boldsymbol{\rho}}_t$. The computational complexity of this recursive approach for constraints remains $O(nd)$. When instead forming the constrained operational-space equations numerically, for instance to form Eq. 24, the number of constraints m does factor into their $O(n^3 + m^3)$ complexity. The accuracy of the approximations that enable the recursive algorithm to maintain its efficiency is demonstrated in the results section to follow.

4.3 Applicability to the Control of Other Tasks

Both of the algorithms described thus far have been presented for torso control of a legged system. The algorithms, however, are general to provide operational-space control of any single body in the system through the application of body renumbering. That is, in the algorithms presented, the torso was selected as a privileged body, in the sense that it was selected as body number 1 (the floating base) in the kinematic connectivity tree. With this in mind, any body could be selected as Body 1 in numbering the kinematic connectivity tree. After an appropriate renumbering, the algorithms here would provide the operational-space inertia matrix, dynamically consistent Jacobian pseudoinverse, and operational-space bias forces for control of the new Body 1.

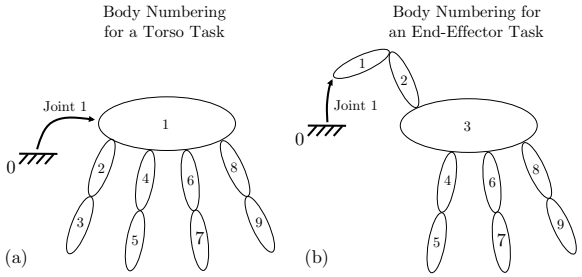

Fig. 4 Body renumbering allows the algorithms here to be applicable to operational-space tasks other than torso control. By renumbering the tree such that any body of interest is Body 1, the algorithms could be applied, for instance, to end-effector control of one limb while the other limbs are in contact with the ground.

Figure 4 shows a simple example of this renumbering process for a quadrupedal morphology, with an original body numbering for torso control given in Fig. 4(a). If one of the legs could instead operate as an end-effector, then it may be desirable to perform operational-space control of that body. By applying the body renumbering shown in Fig. 4(b), the algorithms here would be applicable for end-effector control. In particular, the closed-chain algorithm could still efficiently handle ground contacts on other limbs.

5 Floating-Base Control for a Quadruped Trot

This section details the application of the recursive algorithms presented to the control of a quadruped trot. Diagonal leg pairs are synchronized during the trot, undergoing intermittent periods of stance during which leg forces are generated to propel and stabilize the torso, followed by the flight phase during which legs swing forward in preparation for touchdown again. The articulated legs are modeled with a series spring-actuation system acting around the knee. The quadruped is shown in simulation in Fig. 5(a), with complete details of the system and its dynamic simulator in Palmer and Orin (2010). The quadruped weighs 76 kg in total and stands 60 cm high with the knees in a slightly bent configuration. The torso has a width of 35 cm and length of 1.2 m. The connectivity tree for the quadruped is shown in Fig. 5(b), where each leg has three bodies: a hip, thigh and

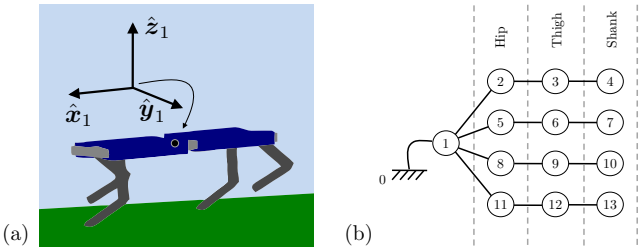

Fig. 5 (a) Quadruped model in simulation.The position and orientation of coordinate frame 1, attached to the torso, are used as the operational-space task to achieve stable trot control. (b) Connectivity tree for quadruped. Each leg is comprised of hip, thigh, and shank bodies that are each preceded by a 1-DOF abduction/adduction, hip swing, and knee joint, respectively.

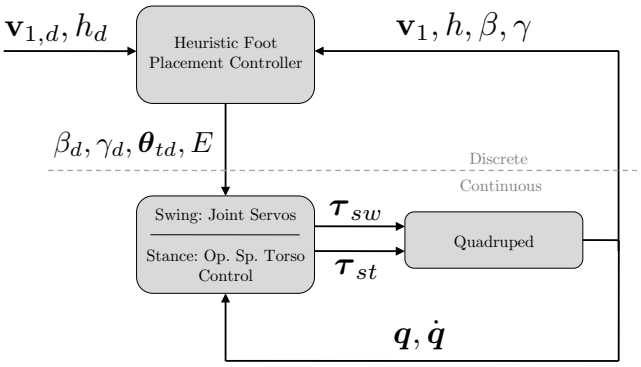

Fig. 6 Hybrid control system. At Top of Flight (TOF), a heuristic step controller selects desired leg touchdown angles ($\boldsymbol{\theta}_{td}$), and stance leg energy thrust (E) to be delivered at maximum leg compression. Desired pitch (β_d) and roll (γ_d) setpoints are also specified on a step-to-step basis. The continuous controller outputs joint torques based on servo control for swing legs ($\boldsymbol{\tau}_{sw}$) and based upon an operational-space control algorithm for legs in stance ($\boldsymbol{\tau}_{st}$). The external inputs to the system are user-prescribed torso velocity $\mathbf{v}_{1,d}$ and TOF height h_d.

shank. Each of these bodies is preceded by a single-DoF leg abduction/adduction (ab/ad) joint, hip swing joint and knee joint, respectively.

The system was simulated in the RobotBuilder environment developed by Rodenbaugh (2003). Contact forces were determined using a spring-damper penalty-based model of contact. Tangential forces were limited by static and kinetic friction coefficients of 0.75 and 0.6, respectively, where any friction violation led to slipping in simulation.

The hybrid control approach for this system, diagrammed in Fig. 6, decomposes the stabilization of a trot similar to a classic Raibert-style controller (Raibert 1986). This control system shares common characteristics with the one used previously in (Palmer and Orin 2010). Both this previous work and the work here used a high-level foot placement controller with a continuous operational-space torso controller. Previous work differed, in that it applied a fuzzy controller to select step parameters for each stride. This intelligent controller allowed for precise control of the top-of-flight (TOF) state of the quadruped.

Here instead, a heuristic footstep controller is used, which provides a much more primitive selection of the leg parameters. A decoupled Raibert-style controller (Raibert 1986) is used to select leg parameters at TOF. Forward and lateral velocities at TOF are controlled by adjusting the desired fore-aft and lateral foot touchdown locations for each leg with respect to its hip. Vertical motion control is achieved through thrust by the knee actuators during stance, which regulates the TOF height reached during the ballistic flight phase. Yaw rate is controlled through a scissoring of the fore and hind limbs before touchdown, which introduces yaw moments during stance. For each control objective (forward velocity, lateral velocity, height, and yaw rate) a simple proportional controller is used to adjust its associated control action (forward leg swing angle, lateral leg angle, thrust, or leg scissoring, respectively) away from a nominal value. This less sophisticated high-level controller places additional burden of stabilization on the low-level operational-space torso control, which additionally must regulate pitch and roll.

The operational-space torso control component differs from (Palmer and Orin 2010) in that it considers constraints at the feet through the closed-chain big-ground-inertia approximation. Additionally, this paper provides the first presentation of any of the operational-space algorithms that have been important to the past and current results.

5.1 Continuous Operational-Space Torso Control

The format of the continuous operational-space floating-base controller used here is based on a previous controller by Palmer and Orin (2007). The controller from this previous work is recast into the operational-space framework to provide connection to the new algorithms described in this paper. During stance, swing leg torques $\boldsymbol{\tau}_{sw} \in \mathbb{R}^6$ and stance leg torques $\boldsymbol{\tau}_{st} \in \mathbb{R}^6$ must be coordinated to control torso pose. Joint selector matrices $\boldsymbol{S}_{sw}$ and $\boldsymbol{S}_{st}$ are defined such that $\boldsymbol{S}_{sw}^T\boldsymbol{\tau}_{sw}$ and $\boldsymbol{S}_{st}^T\boldsymbol{\tau}_{st}$ provide the generalized forces created by the swing and stance torques, respectively. Here, swing torques are selected through PD position servos for the swing legs, leaving only the stance torques to achieve the desired torso dynamics. Given a desired torso acceleration $\ddot{\boldsymbol{x}}_d$, the desired operational-space dynamics are

$$\tilde{\boldsymbol{\Lambda}}_t \ddot{\boldsymbol{x}}_d + \tilde{\boldsymbol{b}}_t = \tilde{\bar{\boldsymbol{J}}}_t^T \boldsymbol{S}_{sw}^T \boldsymbol{\tau}_{sw} + \tilde{\bar{\boldsymbol{J}}}_t^T \boldsymbol{S}_{st}^T \boldsymbol{\tau}_{st} \tag{49}$$

where $\boldsymbol{\tau}_{st}$ remains to be selected. The ability to sense ground force may or may not be available for an experimental system. The work below assumes no such ground force sensor, but a similar approach can be taken if contact forces are known through the use of the original algorithm, and

modification of Eq. 49 to

$$\boldsymbol{\Lambda}_t \, \ddot{\boldsymbol{x}}_d + \boldsymbol{b}_t = \bar{\boldsymbol{J}}_t^T \boldsymbol{S}_{sw}^T \boldsymbol{\tau}_{sw} + \bar{\boldsymbol{J}}_t^T \boldsymbol{S}_{st}^T \boldsymbol{\tau}_{st} \, . \tag{50}$$

The desired acceleration $\ddot{\boldsymbol{x}}_d$ during stance is computed to achieve roll and pitch stability through proportional-derivative feedback but without affecting the forward, lateral, vertical and yaw motions predicted by the passive dynamics. The torso roll (γ) and pitch (β) axes correspond to the x and y axes, respectively in the torso coordinate frame (shown in Fig. 5). Their desired accelerations are set as

$$\dot{\boldsymbol{\omega}}_{1,d}^x = k_\gamma(\gamma_d - \gamma) + k_{\dot{\gamma}}(\dot{\gamma}_d - \dot{\gamma}) \tag{51}$$

$$\dot{\boldsymbol{\omega}}_{1,d}^y = k_\beta(\beta_d - \beta) + k_{\dot{\beta}}(\dot{\beta}_d - \dot{\beta}) \tag{52}$$

where k_γ, $k_{\dot{\gamma}}$, k_β, and $k_{\dot{\beta}}$ are control gains. Desired quantities γ_d, β_d, $\dot{\gamma}_d$, and $\dot{\beta}_d$ are computed from cubic spline trajectories on the roll and pitch angles. These splines are initialized at the beginning of stance to match the pitch and roll setpoints selected at TOF. Desired accelerations for the remaining components seek to replicate the passive dynamics of the system. More specifically, the torso accelerations are found through dynamics computation with $\boldsymbol{\tau}_{sw} = \boldsymbol{0}$, and a passive stance torque $\boldsymbol{\tau}_{st,p}$ which only includes torques from the passive knee springs. The resultant floating-base acceleration for this passive input is found as

$$\begin{bmatrix} \dot{\boldsymbol{\omega}}_{1,p} \\ \ddot{\boldsymbol{p}}_{1,p} \end{bmatrix} = \tilde{\boldsymbol{\Lambda}}_t^{-1} \left(\tilde{\bar{\boldsymbol{J}}}_t^T \boldsymbol{S}_{st}^T \boldsymbol{\tau}_{st,p} - \tilde{\boldsymbol{b}}_t \right) . \tag{53}$$

The desired yaw acceleration and cartesian accelerations are set as $\dot{\boldsymbol{\omega}}_{1,d}^z = \dot{\boldsymbol{\omega}}_{1,p}^z$ and $\ddot{\boldsymbol{p}}_{1,d} = \ddot{\boldsymbol{p}}_{1,p}$, which completes $\ddot{\boldsymbol{x}}_d = [\dot{\boldsymbol{\omega}}_{1,d}^T \, \ddot{\boldsymbol{p}}_{1,d}^T]^T$. It can be shown that Eq. 49 cannot be satisfied exactly since the matrix $\tilde{\bar{\boldsymbol{J}}}_t^T \boldsymbol{S}_{st}^T$ does not have full rank. This property is linked to the fact that the legs do not have redundancy to fulfill the contact constraints and the fact that external forces at the feet can produce no net moment about the line between the foot contact points. To account for the inability to track all desired accelerations, an operational-force weighting matrix is introduced

$$\boldsymbol{W} = \text{diag}([w_{\omega_x} \; w_{\omega_y} \; w_{\omega_z} \; w_{v_x} \; w_{v_y} \; w_{v_z}]) \tag{54}$$

to weight the different components of Eq. 49. An SVD of the matrix $\boldsymbol{W} \tilde{\bar{\boldsymbol{J}}}_t^T \boldsymbol{S}_{st}^T$ is finally used to compute $\boldsymbol{\tau}_{st}$:

$$\boldsymbol{\tau}_{st} = \left(\boldsymbol{W} \tilde{\bar{\boldsymbol{J}}}_t^T \boldsymbol{S}_{st}^T \right)^\dagger \boldsymbol{W} (\tilde{\boldsymbol{\Lambda}}_t \, \ddot{\boldsymbol{x}}_d + \tilde{\boldsymbol{b}}_t - \tilde{\bar{\boldsymbol{J}}}_t^T \boldsymbol{S}_{sw}^T \boldsymbol{\tau}_{sw}). \tag{55}$$

For numerical stability, all singular values $\sigma_i < \frac{1}{100}\sigma_{max}$ are discarded prior to computing the Moore-Penrose pseudoinverse $(\cdots)^\dagger$. Since poor roll and pitch tracking leads to more immediate destabilization than failure to track other desired quantities, w_{ω_x} and w_{ω_y} are generally chosen to be of a greater magnitude than other weightings.

5.2 Continuous Torso Control Through Hip Torque Adjustment

As a simpler alternative to operational-space control, a second controller option, based on direct proportional-derivative control of torso orientation, is used for comparison. As in traditional Raibert-style control (Raibert 1986), posture control can be carried out in stance by using torque at the hip/shoulder. For this approach, the knee torque generated by the passive knee spring is left unmodified, while the ab/ad torque (τ_a) is used to control torso roll, and the hip/shoulder swing torque (τ_s) is used to control torso pitch.

$$\tau_a = K_\gamma(\gamma_d - \gamma) + K_{\dot{\gamma}}(\dot{\gamma}_d - \dot{\gamma}) \tag{56}$$

$$\tau_s = K_\beta(\beta_d - \beta) + K_{\dot{\beta}}(\dot{\beta}_d - \dot{\beta}) \, . \tag{57}$$

Despite similarity to Eqs. 51 and 52, it is important to note that these torques are applied directly. In contrast, the desired accelerations specified in Eqs. 51 and 52 are realized through the use of the dynamically consistent pseudo-inverse which addresses any task couplings that may exist.

6 Results

This section presents both computational results for the closed-chain algorithm itself and simulation results for the quadruped trot example described in the previous section. The first subsection demonstrates the accuracy and efficiency of the closed-chain recursive dynamics algorithm. The second subsection shows the effectiveness of using operational-space control to provide continuous orientation control of the torso during the trot. The control algorithm without force sensors (closed-chain) is found to provide torso control that is comparable to the case when force sensors are required (open-chain). Additionally, the use of the operational-space control is shown to outperform the hip torque adjustment approach in terms of gait stability.

6.1 Computational Results

While a proof of the validity of the big-ground-inertia approximation is given in Appendix A, additional analysis was carried out to verify the accuracy of the resulting algorithm when double precision arithmetic was used in its computation. The recursive dynamics algorithm was run using ground masses for the big-ground-inertia approximation which varied from 10^{-4} to 10^8 times the total mass of the quadruped. The algorithm outputs $\tilde{\boldsymbol{\Lambda}}_t$, $\tilde{\bar{\boldsymbol{J}}}_t^T$, and $\tilde{\boldsymbol{b}}_t$ ($= \tilde{\boldsymbol{\mu}}_t + \tilde{\boldsymbol{\rho}}_t$) were then compared to their closed-form numeric values from Eqs. 20-26. The errors in the algorithm outputs are shown as a function of the normalized ground mass in Fig. 7. All computations were completed on the quadruped

model at mid-stance during a trot. It can be seen that up to a point, additional ground mass allows the recursive algorithm to more closely match the numeric computations.

There is a practical limit to the benefit of increasing the ground mass in the big-ground-inertia approximation, as numerical precision errors begin to degrade the accuracy of the algorithm outputs. More specifically, when the ground inertia is propagated to a leg link, for instance through Eq. 46, a larger ground mass will result in proportionally larger rounding errors during floating-point operations within the algorithm[2]. In contrast, the theoretical accuracy of the algorithm scales proportional to the inverse of the ground mass. Thus, to roughly balance the tradeoff between these effects, the ground mass can be selected as

$$M_{ground} = \frac{m_{min}}{\sqrt{\epsilon_{mach}}} \tag{58}$$

where ϵ_{mach} is the machine epsilon of the floating-point computations (Higham 2002) and m_{min} is the smallest body mass in the leg. The smallest body mass is used in this heuristic since it is most sensitive to floating-point rounding errors. The location of this heuristically constructed breakdown point is shown in Fig. 7 for $\epsilon_{mach} = 2^{-52} \approx 2.2 \times 10^{-16}$. Although the heuristic doesn't exactly provide the point of minimum error for all plots, both the bias and inertia errors attain their minima at ground masses which are within one order of magnitude of the heuristic optimum

$$\left[\frac{m_{min}}{10\sqrt{\epsilon_{mach}}}, \frac{10\, m_{min}}{\sqrt{\epsilon_{mach}}} \right] . \tag{59}$$

For other configurations, the error exhibits the same trends, although the minimum error occurs at different ground masses. However, for 500 randomly sampled quadruped configurations within the limits of the joint angles attained during a trot, the points of minimum inertia and bias errors were found to lie within this same range (Eq. 59) for every configuration.

The dynamically consistent Jacobian pseudoinverse does not break down with increased ground mass in this case. It does experience a similar breakdown when applied to systems that possess redundancy in the leg to satisfy the contact constraint. The fact that the Jacobian pseudoinverse does not break down in this example does not have any implications on the selection of the proper ground mass though, as the first quantity to break down governs proper selection.

[2] Defining $fl(x\ op\ y)$ as the floating-point value returned by a standard arithmetic operation (addition, multiplication, etc.), IEEE standard arithmetic satisfies $fl(x\ op\ y) = (x\ op\ y)(1+\delta)$ where $|\delta| \leq \epsilon_{mach}/2$ represents a normalized rounding error (Higham 2002). ϵ_{mach} is the *machine epsilon* which provides an upper bound on the maximum normalized distance from any floating-point number to a neighboring floating-point number.

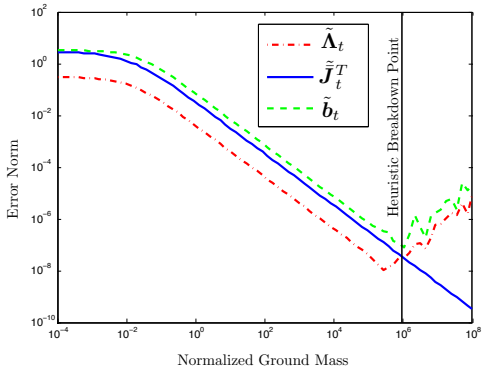

Fig. 7 Differences between the numeric computation of $\tilde{\mathbf{\Lambda}}_t$, $\tilde{\tilde{\boldsymbol{J}}}_t^T$ and $\tilde{\boldsymbol{b}}_t$ (Eqs. 20-26) and their computation with the closed-chain algorithm using the big-ground-inertia approximation. The accuracy of the approximation improves up to the point that machine precision issues begin to affect the recursive computation. The traditional Euclidean norm is used to measure the error for $\tilde{\boldsymbol{b}}_t$, while the Frobenius norm is used for $\tilde{\mathbf{\Lambda}}_t$ and $\tilde{\tilde{\boldsymbol{J}}}_t^T$.

Although the small errors in the recursive algorithm may be seen as a downside, the algorithm's computational speed vastly outperforms the closed-form numeric computations (Eqs. 20-26). This speed is important, as the operational-space control loops proposed here constitute a rather foundational system component, upon which higher-level perception/action loops would rely. Thus, this operational-space control loop would need to be closed as quickly as possible in order to leave time for other processor intensive tasks, such as vision or planning, which would exist in a physical robot. Additionally, a fast control loop would provide benefit when used with learning or adaptation algorithms in simulation where the time to evaluate a rollout is important to rapid learning. The full recursive algorithm is over 13 times faster than the numeric computation for this system on a 2.3 GHz i5 MacBook Pro. The computation of the recursive algorithm requires 13.1 μs and the closed-form numeric computation requires 173 μs on average using the Eigen matrix libraries (Eigen 2014). The computation of the control law (Eqs. 53 and 55) then takes 12.6 μs on average. This control law cost is dominated by the computation of the SVD which is necessary in this underactuated example.

The computational requirements of the recursive algorithm scale better than the closed-form numeric computations when morphologies with more bodies or contacts are considered. Figure 8 shows the required number of floating point operations (FLOPs) for each algorithm. For the recursive computations, these numbers used analysis similar to our previous work (Wensing et al. 2012). Similar analysis was also used to determine computation costs for the mass matrix $\boldsymbol{H}$, Coriolis and gravity terms $\boldsymbol{C} + \boldsymbol{G}$, as well as all required Jacobian matrices $\boldsymbol{J}_c$ and $\boldsymbol{J}_t$ needed by the numeric computations. The cost for these quantities reflects the use of common efficient algorithms summarized in Waldron

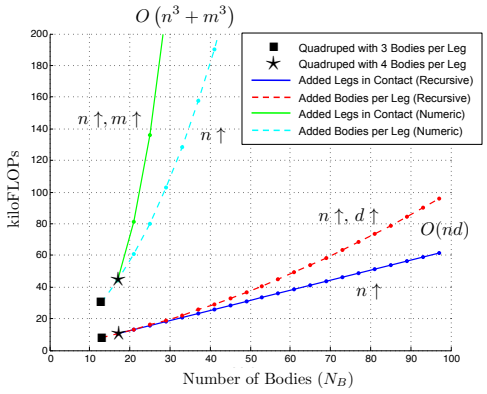

Fig. 8 Required number of Floating Point Operations (FLOPs) for the $O(nd)$ closed-chain recursive algorithm and the $O(n^3+m^3)$ numeric computation versus the number of links in the system. The bottom left points denoted by squares represent the case of a quadruped studied here with 4 legs and 3 links per leg ($N_B = 13$, $n = 18$, and $d = 4$). Starting then from a case $\star$ of a quadruped with 4 legs and 4 bodies per leg, 4 additional links are added per data point. Links are added either within the legs, which increases n and d, or to create another leg in contact, which increases n and m. Based on the timing of the quadruped studied here, 200 kiloFLOPs represents the maximum number of FLOPs that would still allow the numeric algorithm to run at approximately 1 kHz.

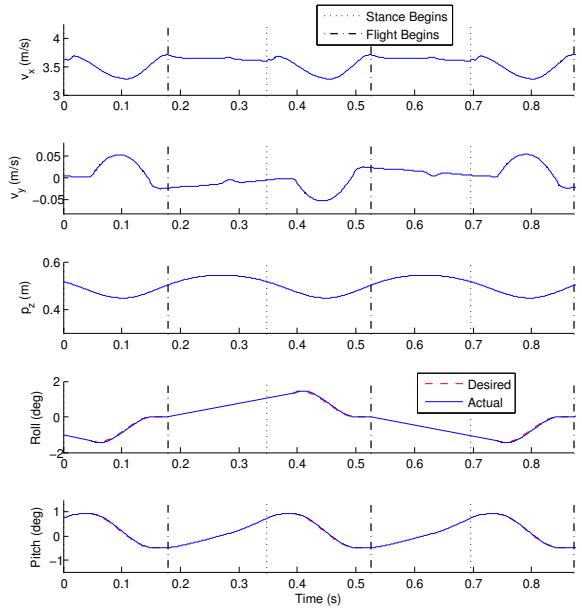

Fig. 9 Velocity, height, roll, and pitch trajectories over 2.5 steps. For this case, task weightings of $[w_{\omega_x}\ w_{\omega_y}\ w_{\omega_z}\ w_{v_x}\ w_{v_y}\ w_{v_z}]^T = [10, 10, 1, 1, 1, 1]^T$ are employed to emphasize the tracking of critical components of orientation (roll and pitch) over tracking of other torso commands.

and Schmiedeler (2008) and Featherstone and Orin (2008). Costs for all other matrix computations (Eqs. 20-26) are then included as well. The upper limit of this graph, 200 kiloFLOPs, represents a bound for the number of operations that could be completed that would still allow the numeric algorithm to run at 1 kHz. This bound was determined based on the timing and FLOP requirements for the quadruped model studied here. Well beyond the cases when the numeric algorithm breaks this FLOP bound, the recursive algorithm still provides a substantial amount of available computation time for other real-time tasks to complete on a 1 kHz schedule.

6.2 Trot Operational-Space Control Results

Both the closed-chain algorithm using the big-ground-inertia approximation as well as the open-chain algorithm for torso control lead to stable trotting at a variety of speeds. All the results here are presented for a forward speed of 3.6 m/s. This speed represents a fast trot, just slightly under the trot-to-gallop transition speed of 3.91 m/s that would be expected in a biological quadruped of this mass (Heglund and Taylor 1988).

The tracking performance of the closed-chain torso-control algorithm is shown in Fig. 9 and demonstrates tight control of the torso roll and pitch within a tenth of a degree during stance. During flight, the dynamics of the legs create pitch and roll disturbances, which are then eliminated in the following stance. At the beginning of stance, the feet may slip with respect to the ground. After friction slows the stance feet and firm footholds are achieved, a desired roll and pitch spline is computed, and the operational-space controller starts operation. The desired pitch spline is generated to center the pitch trajectory around $\beta = 0°$ in steady state while the roll spline is generated to return the system to $\gamma = 0°$ at liftoff. Following liftoff and until the next pair of footholds is secured, no operational-space control is performed, and no desired roll or pitch is specified. For these results, task weightings of $[w_{\omega_x}\ w_{\omega_y}\ w_{\omega_z}\ w_{v_x}\ w_{v_y}\ w_{v_z}]^T = [10, 10, 1, 1, 1, 1]^T$ were hand-selected to track the desired dynamics on the pitch and roll axes more closely than the passive translational and yaw dynamics. This higher task weight on pitch and roll is needed, since poor tracking on these axes allows the flight disturbances from leg swing to accumulate which results in rapid instability.

As mentioned previously, the torso dynamics remain underactuated when two legs are on the ground and modifications to only the stance torques are considered to influence the task dynamics. This result is due to the rank deficiency in $\tilde{\boldsymbol{J}}_t^T \boldsymbol{S}_{st}^T$. Fundamentally, this underactuation limits the degree to which the roll dynamics and lateral dynamics may be independently controlled. In order to affect the roll dynamics, a roll moment must be created on the torso, which for the most part, is created by lateral ground reaction forces. As a result, the lateral dynamics are strongly coupled to the roll dynamics. This coupling can be understood further by examining the results of the controller when different task weightings are employed. The results in Fig. 10 employ decreased task weighting for the roll and pitch mo-

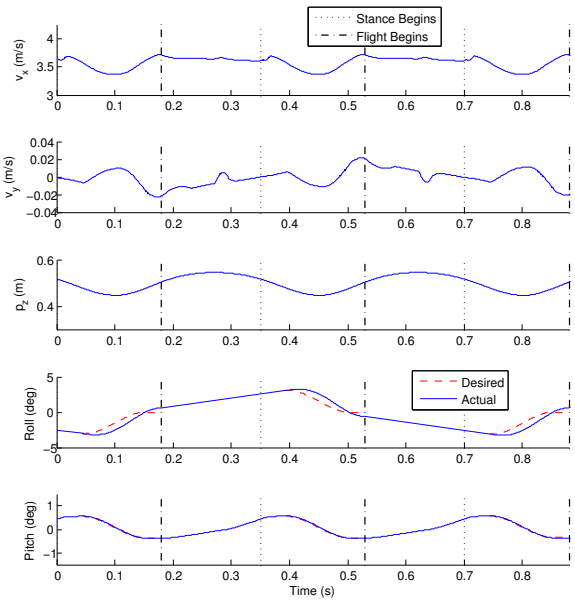

Fig. 10 Velocity, height, roll, and pitch trajectories over 2.5 steps. For this case, task weightings of $[w_{\omega_x}\ w_{\omega_y}\ w_{\omega_z}\ w_{v_x}\ w_{v_y}\ w_{v_z}]^T = [2,2,1,1,3,1]^T$ are employed to emphasize the tracking of the passive lateral velocity dynamics. Critical components of orientation (roll and pitch) are given less emphasis.

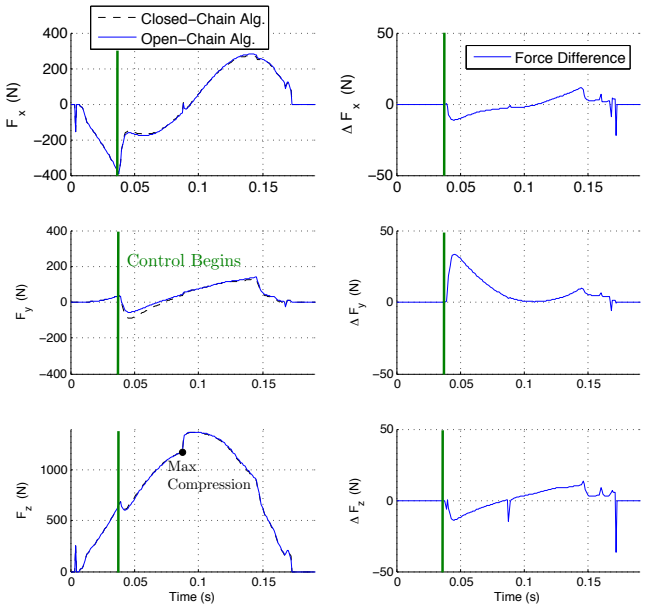

Fig. 11 Ground reaction force (GRF) comparison over one step for application of the control using the closed-chain algorithm without contact force sensing, versus the open-chain algorithm with contact force sensing. At maximum leg compression (0.085 s), force level changes are due to the impulsive injection of actuator energy at the knee.

tions along with increased task weighting for the lateral motion. By more closely following the passive lateral dynamics, this controller experiences 60% less peak-to-peak oscillation in its lateral velocity (v_y), yet experiences over 120% larger roll oscillations in comparison to the previous example. These roll oscillations are due to roll tracking errors of up to 1° which do not stabilize to zero prior to liftoff. There is little coupling between the roll and the pitch tracking, as the pitch tracking remains again within a tenth of a degree. In this case, the system converges to a steady state gait. However, if the weightings on roll and pitch motion are decreased further, the system does experience rapid roll instability.

Although the closed-chain algorithm does not require contact force sensing, the ground reaction forces (GRFs) that result are very close to those achieved by the open-chain algorithm with contact force sensing. This result is due to the use of the passive dynamics for the majority of the desired task dynamics. The use of the passive dynamics also helps to keep the GRFs within their frictional limits[3]. The measured GRFs for each of the control algorithms are shown in Fig. 11. Torso control is enabled at approximately 0.04s, when the feet have achieved a solid foothold. The GRFs in the forward x and vertical z direction are nearly indistinguishable, as both controllers replicate the passive dynamics in these directions. In the y direction however the closed-chain controller achieves a faster response of the lateral force, which provides slightly improved roll tracking.

Just as the resultant trajectories are not sensitive to the choice of an open-chain or closed-chain algorithm, they are also insensitive to the choice of big-ground mass. For the results shown, a normalized big-ground mass of 1.3×10^4 (10^6 kg) was conservatively selected for use. Within 3 orders of magnitude away from the heuristic optimum selection of ground mass (Eq. 58), the trajectories have no visual differences. This observation is due at least partially to the presence of feedback components in the controller. Further, despite non-zero contact accelerations in the control model when light ground masses are used, physical contact in the simulation model prevents any contact position drift. At the torque level, even down to a normalized big-ground mass of 5.2×10^0 (400 kg), none of the resultant control torques at mid-stance vary by more than 10% from those achieved with the heuristic optimum. However, if the contact approximation is removed entirely, such that a big-ground-inertia of $\mathbf{0}$ is used, these torques change by as much as 2000%. These results show some of the robustness of the control approach to selections of the ground mass near the heuristically constructed optimum.

The need for the more complex operational-space control is highlighted by the failure of the simpler PD hip torque adjustment strategy (Raibert 1986). The sagittal plane dynamics for this controller are shown in Fig. 12. The dynamics are shown following roughly 20 steps of a trot, during which the quadruped did not reach any limit-cycle be-

[3] Cases which require active modification of the GRFs to comply with frictional limits would be able to fall back on more complex task-space control approaches by Wensing and Orin (2013).

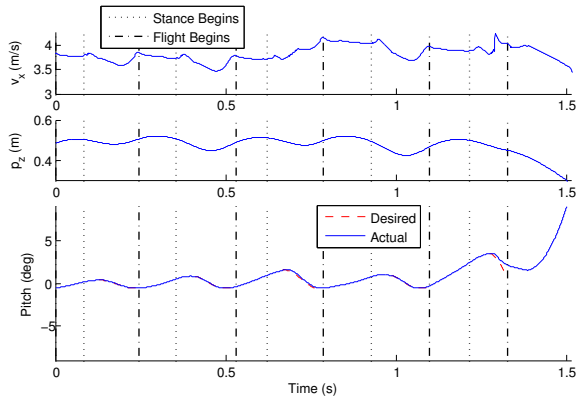

Fig. 12 Tracking of the sagittal plane dynamics with the hip torque adjustment strategy (Raibert 1986). Pitch magnitude grows from step to step and forward velocity is poorly regulated. Since the hip torque strategy does not consider the coupling between hip torque and forward velocity, these motions are not well regulated and lead to instability.

havior. Prior to hand tuning of the gains in Eqs. 56 and 57, the quadruped was only able to run for 2 steps before pitch and roll became unstable. Raibert's controller is model-free and thus may be tuned, in practice, to perform better at low speeds than the model-based approach pursed here. Still, at the high speeds considered in this work, the model-free approach was unable to be tuned to provide stable locomotion. In the steps shown leading to a fall in Fig. 12, the quadruped experiences a pitch trajectory of increasing magnitude. As hip torques are applied to correct these pitch disturbances, coupling from the hip torque to the forward motion of the quadruped causes the system to speed up well beyond the 3.6 m/s setpoint. These higher speeds require larger swing leg trajectories to place the feet forward and slow the quadruped down. Yet, these very swing leg motions contribute to increased pitch disturbances. The failure to address this pitch/forward velocity coupling inevitably leads to a fall, as the extremely pitched configuration causes the quadruped to slip irrecoverably. This example highlights the challenge to manage the many coupled task dynamics through coordinated leg behavior. Further, it showcases the power of the dynamically consistent Jacobian pseudoinverse and operational-space control to efficiently handle these challenges.

7 Summary

This paper has presented an algorithm for efficient computation of the terms of operational-space dynamics in floating-base legged systems. The algorithm includes the first method for the recursive computation of the dynamically consistent Jacobian pseudoinverse in branched kinematic trees. The low order of a second, closed-chain algorithm, which uses a big-ground-inertia approximation, is enabled through the extension of methods from operational-space dynamics for manipulators to the case of branched kinematic trees with grounded elements. This new approach allows the operational-space dynamics of constrained systems to be computed more than an order-of-magnitude faster than direct computation approaches for a quadruped trot example. The new algorithm also has a reduced order, from $O(n^3 + m^3)$ for the direct computation to $O(nd)$ where n is the total number of degrees of freedom in the system, m is the number of constraints, and d is the depth of its kinematic connectivity tree. Through understanding the constrained dynamics of the system, control algorithms based on this approach can operate without contact force sensors at constraint points.

The correctness of the algorithm has been demonstrated through control of a simulated quadruped trot at a speed of 3.6 m/s. Despite underactuation inherent in the problem, the torso controller applied during stance is able to coordinate many actuators in the legs to stabilize the most critical components of the system state (roll and pitch) while tracking the passive dynamics of the torso in other components. The use of the dynamically consistent Jacobian pseudoinverse has been shown to provide better torso stabilization in comparison to simpler PD control orientation controllers which do not account for coupled effects that leg torques have on multiple components of the task motion.

Future work has potential to exploit parallel computation in order to provide further computational benefits to the algorithms proposed. As concurrent advances continue in system actuation and inertial state sensing, the application of model-based dynamic control to physical robotic systems will be an exciting area of future work. Through the use of efficient algorithms to provide foundational control, these future systems will have computational availability to focus on important higher-level planning and control.

Acknowledgements The authors would like the thank the reviewers for their many helpful comments regarding this paper. This work was supported by a National Science Foundation Graduate Research Fellowship to Patrick Wensing, and by Grant No. CNS-0960061 from the NSF with a subaward to The Ohio State University.

A Validity of the Big-Ground-Inertia Approximation

This section provides a proof that the closed-chain algorithm with the big-ground-inertia approximation computes the correct terms $\tilde{\boldsymbol{\Lambda}}_t$, $\tilde{\boldsymbol{b}}_t$, and $\tilde{\boldsymbol{J}}_t^T$ in the limit when the contact inertias $\boldsymbol{I}_{c_j}$ become infinite. Throughout this section it is assumed that the floating base for the contact-constrained system has a full 6 degrees of motion freedom. To assist in the proof, $\tilde{\boldsymbol{\Lambda}}_M$ is defined according to its matrix definition in Eq. 20 and $\tilde{\boldsymbol{\Lambda}}_t(\epsilon)$ is defined as the output of the closed-chain algorithm with $\boldsymbol{I}_{c_j} = \epsilon^{-1}\mathbf{1}$ for each contact j. It will be shown that

$$\lim_{\epsilon \to 0} \tilde{\boldsymbol{\Lambda}}_t(\epsilon) = \tilde{\boldsymbol{\Lambda}}_M. \tag{60}$$

Proof is provided here for $\tilde{\boldsymbol{\Lambda}}_t$ since its recursive computation is the simplest algebraically, and proof that the algorithm achieves the correct limits for $\tilde{\boldsymbol{b}}_t$ and $\tilde{\boldsymbol{J}}_t^T$ follows through identical arguments.

While the algorithms thus far have been derived from unconstrained dynamic equations of motion, recursive constrained dynamics can be derived exactly (not approximately as done here for the modified algorithm) using alternative constraint-propagation methods in (Featherstone 2008). These alternative algorithms are composed in terms of inverse inertias, denoted here as $\bar{\boldsymbol{I}}_i = \boldsymbol{I}_i^{-1}$, and articulated inverse inertias $\bar{\boldsymbol{I}}_i^A$.

Although the standard articulated-body equations of motion are only valid when an articulated subsystem has a full 6 degrees of motion freedom, the more general equations

$$\mathbf{a}_i = \bar{\boldsymbol{I}}_i^A \, \mathbf{f}_i + \boldsymbol{b}_i^A + \sum_{k \in c^*(i)} \boldsymbol{A}_{ik} \boldsymbol{\tau}_k \tag{61}$$

apply when constraints limit motion freedom. In the case that the subsystem rooted at i has 6 degrees of motion freedom $\bar{\boldsymbol{I}}_i^A = \left(\tilde{\boldsymbol{I}}_i^A\right)^{-1}$ and Eq. 61 can be rearranged to the articulated inertia relationship used in the main text

$$\mathbf{f}_i = \tilde{\boldsymbol{I}}_i^A \mathbf{a}_i + \tilde{\boldsymbol{\beta}}_i^A + \sum_{k \in c^*(i)} \tilde{\boldsymbol{B}}_{ik} \boldsymbol{\tau}_k .$$

At each contact c_i, the constraint that $\mathbf{a}_{c_i} = -{}^{c_i}\mathbf{a}_g$, imposed approximately by Eq. 43, can be imposed exactly selecting $\bar{\boldsymbol{I}}_{c_i}^A = \mathbf{0}$ and seeding an inverse inertia recursion with the equation

$$\mathbf{a}_{c_i} = \bar{\boldsymbol{I}}_{c_i}^A \, \mathbf{f}_{c_i} - {}^{c_i}\mathbf{a}_g . \tag{62}$$

The articulated-body inertia recursions used in the main text of this paper could then be replaced with analogous recursions on the inverse inertia (Featherstone 2008) that follow

$$\bar{\boldsymbol{I}}_{p(i)}^A = \bar{\boldsymbol{I}}_{p(i)}^A - \bar{\boldsymbol{I}}_{p(i)}^A \boldsymbol{\varphi}_i \left(\boldsymbol{\varphi}_i^T (\bar{\boldsymbol{I}}_{p(i)}^A + \bar{\boldsymbol{I}}_i^A) \boldsymbol{\varphi}_i \right)^{-1} \boldsymbol{\varphi}_i^T \bar{\boldsymbol{I}}_{p(i)}^A \tag{63}$$

where $\boldsymbol{\varphi}_i = \boldsymbol{\Phi}_i^c$ (the matrix describing the constrained modes of motion for joint i). While coordinate transformations are omitted in this equation for clarity, they would be required for implementation. These recursions, while more general than the usual articulated-body recursions, are more computationally costly since the quantity

$$\left(\boldsymbol{\varphi}_i^T (\bar{\boldsymbol{I}}_{p(i)}^A + \bar{\boldsymbol{I}}_i^A) \boldsymbol{\varphi}_i \right)^{-1}$$

requires a 5×5 matrix inverse for common revolute joints. The motivation to approximate these recursions through a constraint approximation, is that it allows the use of standard articulated inertia recursions which are more computationally amenable. Recursions for $\boldsymbol{b}_i^A$ and $\boldsymbol{A}_{ik}$ can be derived analogous to those for $\tilde{\boldsymbol{\beta}}_i^A$ and $\tilde{\boldsymbol{B}}_{ik}$.

Proof of Eq. 60:

Let $\bar{\boldsymbol{I}}_{c_j} = \epsilon \mathbf{1}$ for each contact and denote $\bar{\boldsymbol{I}}_i^A(\epsilon)$ as the final value of $\bar{\boldsymbol{I}}_i^A$ obtained from the recursions of Eq. 63. In the case when $\epsilon = 0$ the following exact relationship holds

$$\tilde{\boldsymbol{\Lambda}}_M = \left(\bar{\boldsymbol{I}}_1^A(0) \right)^{-1} \tag{64}$$

since Eq. 62 imposes an exact constraint at the contact in this case. Additionally when $\epsilon \neq 0$ the inverse inertia algorithm and the regular inertia algorithm provide equivalent output. That is

$$\tilde{\boldsymbol{\Lambda}}_t(\epsilon) = \left(\bar{\boldsymbol{I}}_1^A(\epsilon) \right)^{-1} \qquad , \forall \, \epsilon > 0 \, . \tag{65}$$

Yet, the recursion equation Eq. 63 is comprised of matrix inverses, additions, and multiplies, which are all continuous with respect to their arguments. Thus, all quantities $\bar{\boldsymbol{I}}_i^A(\epsilon)$ obey

$$\lim_{\epsilon \to 0} \bar{\boldsymbol{I}}_i^A(\epsilon) = \bar{\boldsymbol{I}}_i^A(0) \, . \tag{66}$$

Combination of these relationships provides

$$\begin{aligned} \lim_{\epsilon \to 0} \tilde{\boldsymbol{\Lambda}}_t(\epsilon) &= \lim_{\epsilon \to 0} \left(\bar{\boldsymbol{I}}_1^A(\epsilon) \right)^{-1} \\ &= \left(\lim_{\epsilon \to 0} \bar{\boldsymbol{I}}_1^A(\epsilon) \right)^{-1} \\ &= \left(\bar{\boldsymbol{I}}_1^A(0) \right)^{-1} = \tilde{\boldsymbol{\Lambda}}_M \end{aligned} \tag{67}$$

where once again the continuity of the matrix inverse of $\bar{\boldsymbol{I}}_1^A(\epsilon)$ at $\epsilon = 0$ was used in going from line 1 to line 2.

References

Ahmadi, M., & Buehler, M. (1997). Stable control of a simulated one-legged running robot with hip and leg compliance. *IEEE Transactions on Robotics and Automation*, 13(1), 96–104.

Bhalerao, K., Critchley, J. H., Oetomo, D., Featherstone, R., & Khatib, O. (2013). Distributed operational space formulation. *ASME J. of Computational and Nonlinear Dynamics*, 9(2), 021012:1–10.

Chang, K.-S., & Khatib, O. (2001). Efficient recursive algorithm for the operational space inertia matrix of branching mechanisms. *Advanced Robotics*, 14(8), 703–715.

Eigen, (2014). URL http://eigen.tuxfamily.org.

Featherstone, R. (1983). The calculation of robot dynamics using articulated-body inertias. *Int. J. of Rob. Research*, 2(1), 13–30.

Featherstone, R. (1999a). A divide-and-conquer articulated-body algorithm for parallel O(log(n)) calculation of rigid-body dynamics. Part 1: Basic algorithm. *The International Journal of Robotics Research*, 18(9), 867–875.

Featherstone, R. (1999b). A divide-and-conquer articulated-body algorithm for parallel O(log(n)) calculation of rigid-body dynamics. Part 2: Trees, loops, and accuracy. *The International Journal of Robotics Research*, 18(9), 876–892.

Featherstone, R. (2001). The acceleration vector of a rigid body. *The International Journal of Robotics Research*, 20(11), 841–846.

Featherstone, R. (2008). *Rigid Body Dynamics Algorithms*. Springer.

Featherstone, R. (2010a). A beginner's guide to 6-D vectors (Part 1). *IEEE Robotics Automation Magazine*, 17(3), 83–94.

Featherstone, R. (2010b). A beginner's guide to 6-D vectors (Part 2). *IEEE Robotics Automation Magazine*, 17(4), 88–99.

Featherstone, R., & Orin, D. (2008). Chapter 2: Dynamics. In Siciliano, B., & Khatib, O. (Eds.), *Springer Handbook of Robotics*. New York: Springer.

Heglund, N. C., & Taylor, C. R. (1988). Speed, stride frequency and energy cost per stride: how do they change with body size and gait? *Journal of Experimental Biology*, 138(1), 301–318.

Higham, N. J. (2002). *Accuracy and Stability of Numerical Algorithms*, 2nd edn. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics.

Hutter, M., Hoepflinger, M., Gehring, C., Bloesch, M., Remy, C. D., & Siegwart, R. (2012). Hybrid operational space control for compliant legged systems. In Roy, N., Newman, P., & Srinivasa, S. (Eds.), *Robotics: Science and Systems VIII* (pp. 129–136). Cambridge: MIT Press.

Hyon, S.-H., & Mita, T. (2002). Development of a biologically inspired hopping robot. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, vol. 4, pp. 3984–3991.

Jain, A. (2013). Operational space inertia for robots with internal loops. In *Proc. of Multibody Dynamics*, Zagreb, Croatia, pp. 297–304.

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

Kreutz-Delgado, K., Jain, A., & Rodriguez, G. (1991). Recursive formulation of operational space control. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pp. 1750–1753.

Lathrop, R. (1986). Constrained (closed-loop) robot simulation by local constraint propagation. In *Prof. of the IEEE Int. Conf. on Robotics and Automation*, vol. 3, pp. 689–694.

Lilly, K., & Orin, D. (1993). Efficient O(N) recursive computation of the operational space inertia matrix. *IEEE Transactions on Systems, Man and Cybernetics*, 23(5), 1384–1391.

Lilly, K. W. (1989). Efficient dynamic simulation of robotic mechanisms. PhD thesis, The Ohio State University.

Lilly, K. W. (1993). *Efficient Dynamic Simulation of Robotic Mechanisms*. Norwell, MA, USA: Kluwer Academic Publishers.

Luh, J. Y. S., Walker, M. W., & Paul, R. P. C. (1980). On-line computational scheme for mechanical manipulators. *ASME Journal of Dynamic Systems, Measurement, and Control*, 102(2), 69–76.

Marhefka, D. W., Orin, D. E., Schmiedeler, J. P., & Waldron, K. J. (2003). Intelligent control of quadruped gallops. *IEEE/ASME Transactions on Mechatronics*, 8(4), 446–456.

McMahon, T. (1985). The role of compliance in mammalian running gaits. *Journal of Experimental Biology*, 115(1), 263–282.

McMillan, S., Sadayappan, P., & Orin, D. (1994). Efficient dynamic simulation of multiple manipulator systems with singular configurations. *IEEE Transactions on Systems, Man and Cybernetics*, 24(2), 306–313.

Mistry, M., & Righetti, L. (2011). Operational space control of constrained and underactuated systems. In Durrant-Whyte, H., Roy, N., & Abbeel, P. (Eds.), *Robotics: Science and Systems VII* (pp. 225–232). Cambridge: MIT Press.

Nichol, J. G., Singh, S. P., Waldron, K. J., Palmer, L. R., & Orin, D. E. (2004). System design of a quadrupedal galloping machine. *The Int. J. of Rob. Research*, 23(10-11), 1013–1027.

Orin, D. E., Goswami, A., & Lee, S.-H. (2013). Centroidal dynamics of a humanoid robot. *Autonomous Robots*, 35(2), 161–176.

Palmer, L., & Orin, D. (2007). Force redistribution in a quadruped running trot. In *IEEE Int. Conf. on Rob. and Auto.*, pp. 4343–4348.

Palmer, L., & Orin, D. (2010). Intelligent control of high-speed turning in a quadruped. *J. of Intelligent & Robotic Systems*, 58, 47–68.

Palmer, L. R., Orin, D. E., Marhefka, D. W., Schmiedeler, J. P., & Waldron, K. J. (2003). Intelligent control of an experimental articulated leg for a galloping machine. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, vol. 3, pp. 3821–3827.

Park, J., & Khatib, O. (2006). Contact consistent control framework for humanoid robots. In *Proc. of the IEEE Int. Conference on Robotics and Automation*, pp. 1963–1969.

Raibert, M., Blankespoor, K., Nelson, G., Playter, R., et al. (2008). Bigdog, the rough-terrain quadruped robot. In *Proceedings of the 17th IFAC World Congress*, pp. 10823–10825.

Raibert, M. H. (1986). *Legged robots that balance*. Cambridge, MA, USA: MIT Press.

Raibert, M. H. (1990). Trotting, pacing and bounding by a quadruped robot. *Journal of Biomechanics*, 23, 79–98.

Roberson, R. E., & Schwertassek, R. (1988). *Dynamics of Multibody Systems*. Berlin/Heidelberg/New York: Springer-Verlag.

Rodenbaugh, S. (2003). RobotBuilder: a graphical software tool for the rapid development of robotic dynamic simulations. Master's thesis, The Ohio State University, Columbus.

Rodriguez, G., Jain, A., & Kreutz-Delgado, K. (1992). Spatial operator algebra for multibody system dynamics. *Journal of the Astronautical Sciences*, 40, 27–50.

Russakow, J., Khatib, O., & Rock, S. (1995). Extended operational space formulation for serial-to-parallel chain (branching) manipulators. In *Proc. of the IEEE Int. Conf. on Rob. and Auto.*, pp. 1056–1061.

de Sapio, V., & Khatib, O. (2005). Operational space control of multibody systems with explicit holonomic constraints. In *IEEE Int. Conf. on Robotics and Automation*, pp. 2950–2956.

Schmiedeler, J. P., & Waldron, K. J. (1999). The mechanics of quadrupedal galloping and the future of legged vehicles. *The International Journal of Robotics Research*, 18(12), 1224–1234.

Schmiedeler, J. P., & Waldron, K. J. (2002). Leg stiffness and articulated leg design for dynamic locomotion. In *Proceedings of the ASME International Design Engineering Technical Conferences*, vol. 5, pp. 1105–1112.

Sentis, L., & Khatib, O. (2005). Control of free-floating humanoid robots through task prioritization. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pp. 1718–1723.

Sentis, L., Park, J., & Khatib, O. (2010). Compliant control of multicontact and center-of-mass behaviors in humanoid robots. *IEEE Trans. on Robotics*, 26(3), 483–501.

Sentis, L., Petersen, J., & Philippsen, R. (2013). Implementation and stability analysis of prioritized whole-body compliant controllers on a wheeled humanoid robot in uneven terrains. *Autonomous Robots*, 35(4), 301–319.

Waldron, K., & Schmiedeler, J. (2008). Chapter 1: Kinematics. In Siciliano, B., & Khatib, O. (Eds.), *Springer Handbook of Robotics*. New York: Springer.

Walker, M. W., & Orin, D. E. (1982). Efficient dynamic computer simulation of robotic mechanisms. *ASME Journal of Dynamic Systems, Measurement, and Control*, 104(3), 205–211.

Wensing, P., Featherstone, R., & Orin, D. E. (2012). A reduced-order recursive algorithm for the computation of the operational-space inertia matrix. In *IEEE Int. Conf. on Rob. and Automation*, pp. 4911–4917.

Wensing, P. M., & Orin, D. E. (2013). Generation of dynamic humanoid behaviors through task-space control with conic optimization. In *IEEE Int. Conf. on Rob. and Automation*, pp. 3088–3094.

Wensing, P. M., Hammam, G. B., Dariush, B., & Orin, D. E. (2013). Optimizing foot centers of pressure through force distribution in a humanoid robot. *International Journal Humanoid Robotics*, 10(3), 350027:1–21.

Yamane, K., & Nakamura, Y. (2009). Comparative study on serial and parallel forward dynamics algorithms for kinematic chains. *The International Journal of Robotics Research*, 28(5), 622–629.