A Recursive Multibody Dynamics and Sensitivity Algorithm for Branched Kinematic Chains

Garett A. Sohl

James E. Bobrow

Department of Mechanical Engineering,
University of California, Irvine
Irvine, CA 92697

In this work an efficient dynamics algorithm is developed, which is applicable to a wide range of multibody systems, including underactuated systems, branched or tree-topology systems, robots, and walking machines. The dynamics algorithm is differentiated with respect to the input parameters in order to form sensitivity equations. The algorithm makes use of techniques and notation from the theory of Lie groups and Lie algebras, which is reviewed briefly. One of the strengths of our formulation is the ability to easily differentiate the dynamics algorithm with respect to parameters of interest. We demonstrate one important use of our dynamics and sensitivity algorithms by using them to solve difficult optimal control problems for underactuated systems. The algorithms in this paper have been implemented in a software package named Cstorm (Computer simulation tool for the optimization of robot manipulators), which runs from within Matlab and Simulink. It can be downloaded from the website http://www.eng.uci.edu/~bobrow/
[DOI: 10.1115/1.1376121]

1 Introduction

The dynamics of multibody systems serve as a basis for many models of mechanical devices ranging from simple planar mechanisms to complex biological systems. Although various formulations of the equations of motion for articulated multibody systems have been developed, as we attempt to model, simulate, design, and control systems with ever increasing complexity, the need for more concise, parametric equations becomes an important consideration. If the kinematic and dynamic parameters of the system are accessible, then sensitivity equations can be developed with respect to these parameters. These sensitivities will become increasingly important for complex problems like human motion optimization [1] and animation [2]. In this paper a multibody dynamics algorithm is developed that can be used to solve these complex motion simulation problems.

Early work on dynamics formulations for serial chains for aerospace and robotics applications was conducted in the mid-1960's by Hooker and Margulies [3] and by Uicker [4]. Stepanenko et al. [5] were the first to formulate the recursive Newton-Euler equations for spatial open chains. In their formulation, the kinematics of the links are represented in fixed-frame coordinates. Their approach was reformulated by Orin [6] to provide a cleaner notation and improved efficiency. Luh et al. [7] represented the joint velocities, accelerations, and forces in the local link frames and showed that while the Lagrangian dynamics formulation increases in complexity as the cube of the number of links, the recursive formulation is $O(n)$. Hollerbach [8] provided a recursive form of the Lagrangian formulation, which was also $O(n)$. Silver [9] showed the equivalence of the Lagrangian and Newton-Euler algorithms.

Not only are the above-mentioned formulations of multibody dynamics important, but so are their sensitivity to the motion variables. For instance, Balafoutis et al. [10] developed an algorithm that creates linearized dynamic models for robot systems moving along nominal trajectories. The linearized model allows for linear control system design techniques to be applied to robot systems. Murray and Neuman [11] differentiated the Newton-Euler dynamics equations with respect to kinematic link parameters, link mass properties, and gravity in order to conduct trajectory sensitivity studies, as well as model linearization. Similarly, Martin and Bobrow [12] used derivatives of the dynamics for fully actuated robots to solve for locally optimal motions. While our results are similar in some respects to these papers, none of them deals with underactuated systems with some active and some passive joints as we do. With underactuated systems, the equations are considerably more complex since forward and inverse dynamics problems must be solved simultaneously.

In Featherstone [13], the recursive Newton-Euler equations are expressed in *spatial notation*. In spatial notation, linear and angular quantities are combined into six-dimensional velocity and force vectors. This notation greatly simplifies the analysis of rigid body dynamics by allowing the algorithms to expressed concisely. Featherstone [13] also introduced the concept of the *articulated body inertia* and produced an $O(n)$ dynamics formulation of the forward dynamics which allows joint accelerations to be computed without an explicit inversion of the mass matrix. Rodriguez and Jain [14] applied Kalman filtering techniques to formulate the forward and inverse dynamics using spatial vector notation similar to Featherstone. Building on this work in [15,16], they develop the *Spatial Operator Algebra*, which is then applied to create recursive dynamics formulations. More recently, Featherstone [17] has extended the articulated body inertia algorithm in order to achieve $O(\log(n))$ complexity on $O(n)$ parallel processors.

In Park et al. [18], the *product of exponential* formulation [19,20] is used to derive a geometric version of the recursive dynamics similar to Featherstone's, but based on the geometric concepts of Lie groups. In Ploen and Park [21], the formulation was extended to handle forward dynamics for both recursive and Lagrangian forms of the dynamics. Chen et al. [22,23] used a similar geometric formulation to automatically generate the kinematic and dynamic equations of motion for reconfigurable modular robots. The algorithm developed in this paper builds on [18] to handle the forward dynamics and sensitivity equations for underactuated, tree topology systems. The algorithm developed solves either forward or inverse kinematics. It was inspired by the work of Jain and Rodriguez [16]. We approach the problem from a very different perspective, however. Rather than applying the machinery of Kalman filter and the spatial operator algebra, our algorithm is based upon the geometric concepts of Lie groups. One

Contributed by the Dynamic Systems and Control Division for publication in the JOURNAL OF DYNAMIC SYSTEMS, MEASUREMENT, AND CONTROL. Manuscript received by the Dynamic Systems and Control Division July 10, 2000. Associate Editor: Y. Hurmuzlu.

benefit of this formulation is the ability to differentiate the equations of motion with respect to kinematic and dynamic parameters and compute exact sensitivity information.

2 Background

2.1 The Special Orthogonal and Euclidean Groups, SO(3) and SE(3). Transformations on orientations in Euclidean space are accomplished by the use of the rotation matrix Θ, which is an element of the special Orthogonal group, SO(3). The rotation group SO(3) is an example of a *Lie group*. A Lie group is a differentiable manifold G such that the following two properties are satisfied for $A,B \in G$ [24]:

1 For the mapping $f(A,B)=AB$ we require $f(A,B) \in G$ and f to be continuously differentiable.

2 The mapping $A \rightarrow A^{-1}$ must exist and be continuously differentiable.

To show that property 1 is satisfied, for $A,B \in \mathrm{SO}(3)$, note that the product AB satisfies the two properties of a rotation matrix: $\Theta\Theta^T=\Theta^T\Theta=I$, and $\det(\Theta)=+1$. Since matrix multiplication forms products of the matrix entries which are continuously differentiable, the mapping f is differentiable as well. Additionally, since $\Theta^{-1}=\Theta^T$ for rotation matrices, property 2 is trivially satisfied.

The rigid body transformation, cast in 4×4 homogeneous form, is the Lie group referred to as the special Euclidean group or SE(3). Given the rotation $\Theta \in \mathrm{SO}(3)$ and translation $b \in \mathbf{R}^3$, the inverse of a matrix composed of $(\Theta,b) \in \mathrm{SE}(3)$ can be written as:

$$\begin{bmatrix} \Theta & b \\ 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} \Theta^T & -\Theta^T b \\ 0 & 1 \end{bmatrix} \tag{1}$$

which satisfies property 2.

2.2 Lie Algebras so(3) and se(3). An important concept associated with each Lie group is the notion of a *Lie algebra*. The tangent space at the identity element of a Lie group is called the Lie algebra for that group. The Lie algebra, along with a bilinear map called the *Lie bracket*, forms a vector space. The Lie bracket, $[\cdot,\cdot]$, satisfies

1 *Skew-symmetry*: $[x,y]=-[y,x]$;
2 *Jacobi identity*: $[x,[y,z]]+[z,[x,y]]+[y,[z,x]]=0$;

for every x,y,z in the associated Lie algebra.

The Lie algebra associated with the Lie group SO(3) can be determined by evaluating the tangent vectors to a smooth curve $\Theta(t)$ on SO(3) where $\Theta(0)=I$. Differentiating both sides of $\Theta(t)\Theta(t)^T=I$ with respect to t and evaluating at $t=0$ results in $\dot{\Theta}(0)+\dot{\Theta}(0)^T=0$. Therefore the Lie algebra of SO(3), denoted by so(3), consists of the set of skew symmetric matrices on $\mathbf{R}^{3\times3}$ of the form

$$[\omega]=\begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}, \tag{2}$$

where $\omega=(\omega_x,\omega_y,\omega_z) \in \mathbf{R}^3$ and $[\cdot]$ is an operation which changes the three dimensional vector into the associated skew symmetric matrix.

In a similar fashion, it can be shown that the Lie algebra associated with SE(3), denoted se(3), consists of 4×4 matrices of the form

$$g=\begin{bmatrix} [\omega] & v \\ 0 & 0 \end{bmatrix}, \tag{3}$$

where $v \in \mathbf{R}^3$. We can further simplify this expression by denoting

$$g=(\omega,v) \tag{4}$$

where $\omega \in \mathbf{R}^3$ and $v \in \mathbf{R}^3$. This form contains some important geometric information about the motion. To understand this information, we next describe matrix exponentials on SE(3).

2.3 Matrix Exponentials and Logarithms. The principle relation between a Lie group (SO(3) or SE(3)) and its associated Lie algebra (so(3) or se(3)) is the matrix exponential e^{At}. The closed-form expressions for the matrix exponentials for members of so(3) and se(3) are well-known [25]. For $[\omega] \in \mathrm{so}(3)$, we have

$$\exp([\omega])=I+\frac{\sin\phi}{\phi}[\omega]+\frac{1-\cos\phi}{\phi^2}[\omega]^2, \tag{5}$$

where

$$\phi=\|\omega\|$$

It is straightforward to show that $\exp([\omega]) \in \mathrm{SO}(3)$ by showing that $\exp([\omega])\exp([\omega])^T=\exp([\omega])^T\exp([\omega])=I$ and $\det(\exp([\omega]))=+1$.

The matrix exponential for $g=(\omega,v) \in \mathrm{se}(3)$ is

$$e^g=\exp\begin{bmatrix} [\omega] & v \\ 0 & 0 \end{bmatrix}=\begin{bmatrix} \exp([\omega]) & Av \\ 0 & 1 \end{bmatrix}, \tag{6}$$

where

$$A=I+\frac{1-\cos\phi}{\phi^2}[\omega]+\frac{\phi-\sin\phi}{\phi^3}[\omega]^2$$

It is easy to see that $e^g \in \mathrm{SE}(3)$ since $\exp([\omega]) \in \mathrm{SO}(3)$ and $Av \in \mathbf{R}^3$. Physically, $g=(\omega,v)$ are the *screw parameters* for the transformation: e^g is a screw motion with the axis of rotation directed along ω with the rotation angle $\|\omega\|$. The vector v determines the translational component, and for a pure rotation about ω passing through some point q, $v=q\times\omega$.

The matrix logarithm takes elements of the Lie group and returns the associated member of the Lie algebra. It also has a closed form expression for $\Theta \in \mathrm{SO}(3)$ (see [25]):

$$\log\Theta=\frac{\phi}{2\sin\phi}(\Theta-\Theta^T)=[\omega] \tag{7}$$

and for SE(3):

$$\log\begin{bmatrix} \Theta & b \\ 0 & 1 \end{bmatrix}=\begin{bmatrix} [\omega] & A^{-1}b \\ 0 & 0 \end{bmatrix} \tag{8}$$

where

$$A^{-1}=I-\frac{1}{2}[\omega]+\frac{2\sin\phi-\phi(1+\cos\phi)}{2\phi^2\sin\phi}[\omega]^2,$$

and ϕ satisfies $Tr(\phi)=1-2\cos\phi$, $|\phi|<\pi$. (Also, $\phi^2=\|\omega\|^2$.)

2.4 Spatial Velocities and Forces. We now describe the relations between the *spatial velocity* of a moving body and the Lie algebra se(3). Let $X(t)=(\Theta(t),b(t))$ be a curve on SE(3) describing the motion of a rigid body relative to an inertial frame. The tangent vector $\dot{X}(t)$ can be identified with an element of se(3) in two different ways:

$$\dot{X}X^{-1}=(\dot{\Theta}\Theta^{-1},\dot{b}-\dot{\Theta}\Theta^{-1})$$

and

$$X^{-1}\dot{X}=(\Theta^{-1}\dot{\Theta},\Theta^{-1}\dot{b})$$

which are both elements of se(3). (Observe that both $\dot{\Theta}\Theta^{-1}$ and $\Theta^{-1}\dot{\Theta}$ are skew symmetric and therefore elements of so(3).) We refer to $X^{-1}\dot{X}$ as the *body-fixed velocity* representation of $\dot{X}$ since $[\omega]\Theta^{-1}\dot{\Theta}$ and $v=\Theta^{-1}\dot{b}$ are the angular and translational velocities of the rigid body expressed in the moving frame (often re-

ferred to as a body-fixed reference frame). The pair $(\omega, v) \in se(3)$ is referred to as the spatial velocity representation of the body.

While spatial velocities are represented with elements of se(3), *spatial forces*, which consist of a moment-force pair, are regarded as inhabiting the dual space of se(3), denoted se(3)*. This distinction can be traced to the fact that forces (which behave as covectors) transform differently under a change of coordinates than velocities (which behave as tangent vectors). As a result, forces and moments can be thought of as belonging to the dual space of velocities and angular velocities. A spatial force represented in screw form, $g=(\mathcal{M},\mathcal{F}) \in se(3)^*$, is often called a *wrench*.

2.5 Adjoint Operators. Since members of se(3) have six free parameters as shown in (4), we may store them in a compact form as elements of $\mathbf{R}^6$. The adjoint operators on so(3) and se(3) will be defined on both matrix and vector forms. Since it is normally unambiguous which form is necessary when performing operations, we will not find it necessary to distinguish between the two forms. In those rare cases where it is necessary to state which form $g \in se(3)$ takes it will be pointed out in the text-otherwise the reader may assume either form.

An element of the Lie group can be used as a linear mapping on the Lie algebra. This is called the *Adjoint map* on SE(3), denoted by Ad, where $\mathrm{Ad}_G(h): se(3) \mapsto se(3)$ is given by

$$\mathrm{Ad}_G(h) = GhG^{-1} \tag{9}$$

for $G \in SE(3)$ and $h \in se(3)$. Note that h is in the 4×4 matrix form of Eq. (3). It can be easily shown that $\mathrm{Ad}_G(h)$ admits the following form for the 6-vector representation of h:

$$\mathrm{Ad}_G(h) = \begin{bmatrix} \Theta & 0 \\ [b]\Theta & \Theta \end{bmatrix} \begin{bmatrix} \omega_h \\ v_h \end{bmatrix} \tag{10}$$

Physically, the Adjoint map is a coordinate transformation on se(3). This allows us to transform spatial velocities from one reference frame to another via the SE(3) map between the two frames. For example, if $V_2=(\omega_2, v_2)$ denotes the spatial velocity of a rigid body in frame M_2 and $T_{1,2} \in SE(3)$ represents the coordinate map from frame M_2 to frame M_1 then the spatial velocity of the rigid body in frame M_1, denoted V_1, is given by $V_1 = \mathrm{Ad}_{T_{1,2}}(V_2)$.

The *dual Adjoint* operator, denoted by Ad* is a linear mapping on the dual space, se(3)*. The map $\mathrm{Ad}^*_G(h^*): se(3)^* \mapsto se(3)^*$ is given by

$$\mathrm{Ad}^*_G(h^*) = G^{-1}h^*G \tag{11}$$

for $G \in SE(3)$ and $h^* \in se(3)^*$. An equivalent form is used when h^* is represented as a 6-vector:

$$\mathrm{Ad}^*_G(h^*) = \begin{bmatrix} \Theta^T & \Theta^T[b]^T \\ 0 & \Theta^T \end{bmatrix} \begin{bmatrix} \mathcal{M} \\ \mathcal{F} \end{bmatrix} \tag{12}$$

Notice that the matrix used for the Ad* operator is the transpose of the one used for the Ad operator.

Physically, the dual Adjoint map is a coordinate transformation on se(3)*. This allows us to transform spatial forces from one reference frame to another via the SE(3) map between the two frames. For example, if $F_2=(\mathcal{M}_2,\mathcal{F}_2) \in se(3)^*$ denotes the spatial force acting on a rigid body with respect to frame M_2 and $T_{1,2} \in SE(3)$ represents the coordinate map from frame M_2 to frame M_1 then the spatial force acting on the rigid body expressed in frame M_1, denoted F_1, is given by $F_1 = \mathrm{Ad}^*_{T_{1,2}}(F_2)$.

The Lie algebra can also be used as a linear mapping on itself. This is the *Lie bracket* operation discussed earlier. We will denote this operation by ad. It has the form:

$$\mathrm{ad}_g(h) = [g,h] = gh - hg, \tag{13}$$

for $g,h \in se(3)$ and g,h in the 4×4 matrix form of (3). The equivalent form for the 6-vector representation of h is

$$\mathrm{ad}_g(h) = \begin{bmatrix} [\omega_g] & 0 \\ [v_g] & [\omega_g] \end{bmatrix} \begin{bmatrix} \omega_h \\ v_h \end{bmatrix} \tag{14}$$

Similarly, the dual operator, ad* is given by

$$\mathrm{ad}^*_g(h^*) = [g,h^*] \tag{15}$$

or by the equivalent form:

$$\mathrm{ad}^*_g(h^*) = \begin{bmatrix} [\omega_g]^T & [v_g]^T \\ 0 & [\omega_g]^T \end{bmatrix} \begin{bmatrix} \mathcal{M} \\ \mathcal{F} \end{bmatrix} \tag{16}$$

Physically the mappings $\mathrm{ad}_g(h)$ and $\mathrm{ad}^*_g(h)$ generalize the standard cross product operation to se(3) and se(3)*.

2.6 Kinematics Using the Product of Matrix Exponentials. The kinematics of an open chain can be modeled as a sequence of homogeneous transformations between consecutive joint frames. Let the transformation which describes the motion between the frame of link i and the frame of link $i-1$ be $T_{i-1,i} \in SE(3)$. A series of sequential matrix transformations between adjacent joint frames can be combined as:

$$T_{i,j} = T_{i,i+1}T_{i+1,i+2} \ldots T_{j-1,j} \tag{17}$$

The homogeneous transformation $T_{i,i+1}$ can be written in two forms using matrix exponential notation. The first form is $T_{i,i+1=e^{S_i q_i} M_i}$, where $S_i \in se(3)$ is the joint screw written in the coordinates of link $i-1$, q_i is the current position of joint i relative to a specified zero position and M_i is the coordinate transformation between link i and link $i-1$. The second form is $T_{i,i+1} = M_i e^{S_i q_i}$, where S_i is the joint screw written in the coordinates of the body-fixed frame for link i. This representation is the more useful of the two since it allows us to write the joint screw for link i in the local frame of link i.

Note that expressing the link to link transformations in exponential form has an advantage over other representations like Denavit-Hartenberg, in that the joint motion for prismatic, revolute, and screw joints are treated in a uniform way. In addition, it is trivial to differentiate $e^{S_i q_i}$ with respect to q_i which makes it far less cumbersome to derive the sensitivity equations with our approach.

3 Hybrid Recursive Dynamics Algorithm

3.1 Newton's Law for a Single Rigid Body. The algorithms in this paper use the spatial form of Newton's law derived in [18]. If one uses the 6-vector form of the spatial force and velocity, then the Newton-Euler equations of motion for a single rigid body are

$$F = J\dot{V} - \mathrm{ad}^*_V(JV), \tag{18}$$

where ad* is defined in (16), $V=\left[\begin{smallmatrix}\omega\\ v\end{smallmatrix}\right]$, is the spatial velocity of a frame attached to the moving body whose origin is at the point of application of the applied load $F=\left[\begin{smallmatrix}M\\ f\end{smallmatrix}\right]$, and the spatial inertia J is

$$J = \begin{bmatrix} I - m[r]^2 & m[r] \\ -m[r] & m \cdot \mathbf{1} \end{bmatrix}. \tag{19}$$

In (18), the loads may be applied at an arbitrary location on the body relative to its center of mass. The mass of the body is m and the inertia about the center of mass is I. The vector from the point of application of the load to the center of mass is r. The Newton-Euler equations equivalent to (18) are

$$f = m\overline{a} \tag{20}$$

$$M = I\dot{\omega} + \omega \times I\omega + r \times m\overline{a}, \tag{21}$$

where $a = \dot{v} + \omega \times v$ is the acceleration of the point of application of the load, and $\overline{a} = a + \dot{\omega} \times r + \omega \times (\omega \times r)$ is the acceleration of

the center of mass of the body. In order to establish the equivalence between the above equations and (18), one multiplies out the components of (18) and applies the Jacobi identity $a\times(b\times c)+c\times(a\times b)+b\times(c\times a)=0$.

3.2 Extending Newton-Euler. The spatial form of the Newton-Euler equations can be applied to open kinematic chains by summing the forces on each link of the chain and transforming them to the appropriate coordinate system. For solving inverse dynamics problems the outward/inward recursions can be developed similar to Luh-Walker-Paul [7] or Featherstone [13]. This algorithm assumes that the motion (position, velocity and acceleration) for each joint is given and solves for the unknown joint torques. It consists of two recursions-one outward from the base to the tip and one inward from the tip to the base as follows:

Algorithm 3.1 (Park et al. [18]). *Recursive Inverse Dynamics.*

- Initialization

$$\text{Given: } V_0=0,\dot{V}_0,F_{n+1}$$

Outward recursion: for $i=1-n$ do

$$T_{i-1,i}=M_i e^{S_i q_i} \tag{22}$$

$$V_i=\mathrm{Ad}_{T_{i-1,i}^{-1}}(V_{i-1})+S_i\dot{q}_i \tag{23}$$

$$a_i=-\mathrm{ad}_{S_i\dot{q}_i}(\mathrm{Ad}_{T_{i-1,i}^{-1}}(V_{i-1})) \tag{24}$$

$$=-\mathrm{ad}_{S_i\dot{q}_i}(V_i) \tag{25}$$

$$\dot{V}_i=\mathrm{Ad}_{T_{i-1,i}^{-1}}(\dot{V}_{i-1})+S_i\ddot{q}_i+a_i \tag{26}$$

Inward recursion: for $i=n-1$ do

$$F_i=\mathrm{Ad}^*_{T_{i,i+1}^{-1}}(F_{i+1})+J_i\dot{V}_i-\mathrm{ad}^*_{V_i}(J_i\dot{V}_i) \tag{27}$$

$$\tau_i=S_i^T F_i \tag{28}$$

We have separated the portion of $\dot{V}_i$ due to Coriolis forces and called it a_i. This will aid us in later calculations. Note that the operation Ad transforms the spatial velocity to link i's coordinate frame, and the operation Ad* transforms the joint wrench to link i's coordinate frame.

In the following, we will call joints with known torque and unknown accelerations *passive joints* and joints with known accelerations and unknown torques *active joints*. When referring to active or passive joints, we will often use the superscripts *a* or *p*, respectively. For example, q^p refers to the set of passive joint positions, while $\dot{q}^a$ refers to the set of active joint velocities. We would like to modify Algorithm (3.1) so that it will handle both active and passive joints-allowing us to solve for the unknown torques on the active joints and the unknown accelerations on the passive joints. The first problem with trying to use this algorithm for passive joints appears in (26). Since only the active joint accelerations are known, we cannot solve for $\dot{V}_i$ during the first outward recursion if there are any passive joints in the chain. Without $\dot{V}_i$ we are also unable to use (27) to solve for the spatial force, F_i, during the inward recursion.

Using the concept of the articulated body inertia [13], we can write F_i in the following form:

$$F_i=\hat{J}_i\dot{V}_i+B_i \tag{29}$$

where $\hat{J}$ is the articulated body inertia of link i and B_i is the bias force on link *i*. Unlike (27), Eq. (29) does not require knowledge of F_{i+1} to compute F_i. This allows us to postpone the problem of computing F_i until we have found $\dot{V}_i$ and introduces the new problem of computing $\hat{J}_i$ and B_i for each joint. As it turns out, we can compute these quantities for all the joints in chain even without a complete expression for $\dot{V}_i$. This computation is done during an inward recursion from the tip to the base. The articulated body inertia and the bias force take on different forms depending upon whether the joint outward in the chain is active or passive.

In order to start the recursive computation of $\hat{J}$ and B_i, we first recognize that for the last link in a chain, Eq. (27) is the same form as (29) with

$$\hat{J}_n=J_n \tag{30}$$

$$B_n=\mathrm{Ad}^*_{T_{n,n+1}^{-1}}(F_{n+1})-\mathrm{ad}^*_{V_n}(J_nV_n) \tag{31}$$

where F_{n+1} is a known force on the last link. Using these values as a starting point, we can now develop a recursive algorithm to solve for $\hat{J}_i$ and B_i as we move in from the tip.

Let us first derive an expression for $\hat{J}_i$ and B_i for the case when joint $i+1$ is active ($\ddot{q}_{i+1}$ is known). To do this, we first substitute Eq. (26) into (29) for joint $i+1$:

$$F_{i+1}=\hat{J}_{i+1}(S_{i+1}\ddot{q}_{i+1}+\mathrm{Ad}_{T_{i,i+1}^{-1}}(\dot{V}_i)+a_{i+1})+B_{i+1} \tag{32}$$

$$a_{i+1}=-\mathrm{ad}_{S_{i+1}q_{i+1}}(\mathrm{Ad}_{T_{i,i+1}^{-1}}(V_i)) \tag{33}$$

where a_{i+1} contains the Coriolis portion of $\dot{V}_{i+1}$ as described earlier. We then substitute (32) into (27):

$$F_i=\mathrm{Ad}^*_{T_{i,i+1}^{-1}}[\hat{J}_{i+1}(S_{i+1}\ddot{q}_{i+1}+\mathrm{Ad}_{T_{i,i+1}^{-1}}(\dot{V}_i)+a_{i+1})+B_{i+1}]+J_i\dot{V}_i-\mathrm{ad}^*_{V_i}(J_iV_i) \tag{34}$$

Note that the only unknown in this equation is $\dot{V}_i$. Both $\hat{J}_{i+1}$ and B_{i+1} are known from the previous step in the recursion. We can now group terms and solve for $\hat{J}_i$ and B_i:

$$\hat{J}_i=J_i+\mathrm{Ad}^*_{T_{i,i+1}^{-1}}\hat{J}_{i+1}\mathrm{Ad}_{T_{i,i+1}^{-1}} \tag{35}$$

$$B_i=\mathrm{Ad}^*_{T_{i,i+1}^{-1}}[\hat{J}_{i+1}(S_{i+1}\ddot{q}_{i+1}+a_{i+1})+B_{i+1}]-\mathrm{ad}^*_{V_i}(J_iV_i) \tag{36}$$

The Ad and Ad* operators are used in their 6×6 matrix form in these equations.

Deriving an expression for $\hat{J}_i$ and B_i when joint $i+1$ is passive is slightly more complicated. We cannot simply use Eq. (34) since $\ddot{q}_{i+1}$ is unknown. We must find an expression for $\ddot{q}_{i+1}$ whose only unknown quantity is $\dot{V}_i$ and then substitute that expression into (34).

Since joint $i+1$ is a passive joint, the torque is known. We can relate the unknown acceleration to the known joint torque as:

$$\tau_{i+1}=S_{i+1}^T F_{i+1} \tag{37}$$

We then substitute Eq. (32) into (37):

$$\tau_{i+1}=S_{i+1}^T[\hat{J}_{i+1}(S_{i+1}\ddot{q}_{i+1}+\mathrm{Ad}_{T_{i,i+1}^{-1}}(\dot{V}_i)+a_{i+1})+B_{i+1}] \tag{38}$$

and solve for $\ddot{q}_{i+1}$ as

$$\ddot{q}_{i+1}=\frac{\tau_{i+1}-S_{i+1}^T[\hat{J}_{i+1}(\mathrm{Ad}_{T_{i,i+1}^{-1}}(\dot{V}_i)+a_{i+1})+B_{i+1}]}{S_{i+1}^T\hat{J}_{i+1}S_{i+1}} \tag{39}$$

Note that the only unknown in this equation is $\dot{V}_i$. We now substitute (39) into (34) and group terms:

$$F_i=\left[J_i+\mathrm{Ad}^*_{T_{i,i+1}^{-1}}\left(\hat{J}_{i+1}-\frac{\hat{J}_{i+1}S_{i+1}S_{i+1}^T\hat{J}_{i+1}}{S_{i+1}^T\hat{J}_{i+1}S_{i+1}}\right)\mathrm{Ad}_{T_{i,i+1}^{-1}}\right]\dot{V}_i+\mathrm{Ad}^*_{T_{i,i+1}^{-1}}\left[\left(I-\frac{\hat{J}_{i+1}S_{i+1}S_{i+1}^T}{S_{i+1}^T\hat{J}_{i+1}S_{i+1}}\right)(\hat{J}_{i+1}a_{i+1}+B_{i+1})+\frac{\hat{J}_{i+1}S_{i+1}\tau_{i+1}}{S_{i+1}^T\hat{J}_{i+1}S_{i+1}}\right]-\mathrm{ad}^*_{V_i}(J_iV_i) \tag{40}$$

Finally, we note that $\hat{J}_i$ is the coefficient of the unknown $\dot{V}_i$. B_i contains all the remaining terms.

We can now write a new inward recursion in which we solve for the articulated body inertia, $\hat{J}$, and the bias force, B_i for each link.

Algorithm 3.2. *Inward recursion for $\hat{J}_i$ and B_i.*

- Initialization of inward recursion:

$$\text{Given: } \hat{J}_{n+1}=0, z_{n+1}=0, \quad B_{n+1}=F_{n+1}$$

Inward recursion: for $i=n-1$ do

$$\hat{J}_i=\begin{cases} J_i+\mathrm{Ad}^*_{T^{-1}_{i,i+1}}\hat{J}_{i+1}\mathrm{Ad}_{T^{-1}_{i,i+1}} & (i+1)\in I^a \\ J_i+\mathrm{Ad}^*_{T^{-1}_{i,i+1}}\left[I-\dfrac{\hat{J}_{i+1}S_{i+1}S^T_{i+1}}{S^T_{i+1}\hat{J}_{i+1}S_{i+1}}\right]\hat{J}_{i+1}\mathrm{Ad}_{T^{-1}_{i,i+1}} & (i+1)\in I^p \end{cases} \tag{41}$$

$$b_i=-\mathrm{ad}^*_{V_i}(J_iV_i) \tag{42}$$

$$B_i=\begin{cases} \mathrm{Ad}^*_{T^{-1}_{i,i+1}}z_{i+1}+b_i & (i+1)\in I^a \\ \mathrm{Ad}^*_{T^{-1}_{i,i+1}}\left[z_{i+1}+\dfrac{\hat{J}_{i+1}S_{i+1}(\tau_{i+1}-S^T_{i+1}(z_{i+1}))}{S^T_{i+1}\hat{J}_{i+1}S_{i+1}}\right]+b_i & (i+1)\in I^p \end{cases} \tag{43}$$

$$z_i=\begin{cases} \hat{J}_i(S_i\ddot{q}_i+a_i)+B_i & i\in I^a \\ \hat{J}_ia_i+B_i & i\in I^p \end{cases} \tag{44}$$

The sets I^a and I^p denote the indices of the active and passive joints, respectively. The new variable z_i contains all the currently known portions of the generalized force F_i. To see this, recall that $F_i=\hat{J}_i\dot{V}_i+B_i$ and we see that z_i has the same form, but only contains the known portions of $\dot{V}_i$.

We are now left with the original problem of finding $\dot{V}_i$ when there are passive joints in the chain. Once that is done, we can find easily find F_i using (29). Equation (39) is the key to solving this last difficulty. If we rewrite (39) for joint i we have:

$$\ddot{q}_i=\frac{\tau_i-S^T_i[\hat{J}_i(\mathrm{Ad}_{T^{-1}_{i-1,i}}(\dot{V}_{i-1})+a_i)+B_i]}{S^T_i\hat{J}_iS_i} \tag{45}$$

If joint i is passive, given $\dot{V}_{i-1}$ we can use (45) to find $\ddot{q}_i$ and then use (26) to find $\dot{V}_i$. If joint i is active, given $\dot{V}_{i-1}$, we can compute $\dot{V}_i$ directly from (26). This means that given $\dot{V}_0$ (the acceleration of the base), we can solve for each $\dot{V}_i$ as a function of $\dot{V}_{i-1}$ as we move out from the base as follows:

Algorithm 3.3. *Outward recursion for τ^a_i and $\ddot{q}^p_i$.*

- Initialization of outward recursion

$$\text{Given: } \dot{V}_0$$

Outward recursion: for $i=1$ to n do

$$\ddot{q}_i=\frac{\tau_i-S^T_i[\hat{J}_i\mathrm{Ad}_{T^{-1}_{i-1,i}}(\dot{V}_{i-1})+z_i]}{S^T_i\hat{J}_iS_i} \quad i\in I^p \tag{46}$$

$$\tau_i=S^T_i(\hat{J}_i\dot{V}_i+B_i) \quad i\in I^a \tag{47}$$

$$\dot{V}_i=\mathrm{Ad}_{T^{-1}_{i-1,i}}(\dot{V}_{i-1})+S_i\ddot{q}_i+a_i \tag{48}$$

In this algorithm we have used the results of the previous inward recursion to obtain z_i, $\hat{J}_i$, and B_i.

3.3 Branched Chains. Branched chains are serial open chains with two or more distinct ''branches'' leading to two or more tip links. Our hybrid algorithm can be used to compute the dynamics of branched chains, but we need to present the algorithm in a slightly different form. In the previous section, the algorithm is presented in an iterative form using a simple numbering scheme ($i=1$ to n) for the joints. Link i corresponded to the ith link counting outward from the base. This numbering scheme can not be used for branched chains since more than one path outward from the base exists.

To overcome this ambiguity in link numbering we make some new definitions:

- *Parent Link*: the link inward (towards the base) from a given link.
- *Child Link(s)*: the link or links which are outward (towards the tip(s)) from a given link.

We can now use these definitions to rewrite our hybrid recursive algorithm for branched chains. The initial outward recursion is done in a *depth first* manner. This means that we move outward on a single branch until we reach the end of that branch. We then move inward along that branch until we find a separate branch which we follow to its end, and so on.

Algorithm 3.4. *Outward recursion for spatial velocity of branched chains.*

- Initialization of outward recursion

$$\text{Given: } V_0$$

Outward recursion: loop over all links in depth first manner

$$T_{P,i}=M_ie^{S_iq_i} \tag{49}$$

$$V_i=\mathrm{Ad}_{T^{-1}_{P,i}}(V_P)+S_i\dot{q}_i \tag{50}$$

$$a_i=-\mathrm{ad}_{S_i\dot{q}_i}\mathrm{Ad}_{T^{-1}_{P,i}}(V_P) \tag{51}$$

$$b_i=-\mathrm{ad}^*_{V_i}(J_iV_i) \tag{52}$$

where the index P denotes the parent link for link i. So, $T_{P,i}$ denotes the mapping from link i to its parent link and V_P denotes the spatial velocity of the parent link.

The inward recursion is done in a reversed *breadth first* manner. We start at the tip of the chain furthest (in terms of number of links) from the base. As we move inward, we compute $\hat{J}$ and B for all links which are equally far away from the base. We also allow any number external forces to be applied to a link in the branched chain. This requires the summation term in Eq. (55) which maps each external force on link i, denoted $F'_{i,j}$, into the local link frame using $\mathrm{Ad}^*_{T^{-1}_{i,j}}$. The map $T_{i,j}$ denotes the transformation from frame j, in which the external force is written, and frame i, the body-fixed frame for link i.

Algorithm 3.5. *Inward recursion for $\hat{J}_i$ and B_i for branched chains.*

- Initialization of inward recursion
- *Given: the externally applied forces on each link* $(F'_{i,*})$ *and* $\hat{J}^+_j=0$ *for each tip link*
- Inward recursion: loop over all links in reversed breadth first manner

$$\hat{J}_i=J_i+\sum_{j\in C}\mathrm{Ad}^*_{T^{-1}_{i,j}}\hat{J}^+_j\mathrm{Ad}_{T^{-1}_{i,j}} \tag{53}$$

$$B_i=b_i+\sum_{j\in C}\mathrm{Ad}^*_{T^{-1}_{i,j}}z^+_j \tag{54}$$

$$z_i = \begin{cases} \hat{J}_i a_i + B_i + \Sigma_j \mathrm{Ad}^*_{T_{i,j}^{-1}} F'_{i,j} & i \in I^p \\ \hat{J}_i (S_i \ddot{q}_i + a_i) + B_i + \Sigma_j \mathrm{Ad}^*_{T_{i,j}^{-1}} F'_{i,j} & i \in I^a \end{cases} \tag{55}$$

$$\hat{J}_i^+ = \begin{cases} \left[I - \dfrac{\hat{J}_i S_i S_i^T}{S_i^T \hat{J}_i S_i} \right] \hat{J}_i & i \in I^p \\ \hat{J}_i & i \in I^a \end{cases} \tag{56}$$

$$z_i^+ = \begin{cases} \left[z_i + \dfrac{\hat{J}_i S_i (\tau_i - S_i^T z_i)}{S_i^T \hat{J}_i S_i} \right] & i \in I^p \\ z_i & i \in I^a \end{cases} \tag{57}$$

where C denotes the set of child links for link i.

The final recursion is done in the same depth first manner as the first outward recursion:

Algorithm 3.6. *Outward recursion for τ_i^a and $\ddot{q}_i^p$ for branched chains.*

• Initialization of outward recursion

$$\text{Given: } \dot{V}_0$$

Outward recursion: loop over all joints in a depth first manner

$$\ddot{q}_i = \frac{\tau_i - S_i^T [\hat{J}_i \mathrm{Ad}_{T_{P,i}^{-1}} (\dot{V}_P) + z_i]}{S_i^T \hat{J}_i S_i} \quad i \in I^p \tag{58}$$

$$\tau_i = S_i^T (\hat{J}_i \mathrm{Ad}_{T_{P,i}^{-1}} (\dot{V}_P) + z_i) \quad i \in I^a \tag{59}$$

$$\dot{V}_i = \mathrm{Ad}_{T_{P,i}^{-1}} (\dot{V}_P) + S_i \ddot{q}_i + a_i \tag{60}$$

where, as earlier, P denotes the index of link i's parent link.

4 Sensitivity of the Dynamics Algorithm

As mentioned in the Introduction, we will compute the derivatives of the dynamics with respect to the parameters that influence the motion. Because sensitivity analysis is crucial for many aspects of the analysis of dynamical systems, this is the primary contribution of this research. The recursive dynamics algorithm can be viewed as a function of the form:

$$\begin{Bmatrix} \tau^a \\ \ddot{q}^p \end{Bmatrix} = f(q, \dot{q}, \ddot{q}^a, \tau^p), \tag{61}$$

where, if we assume that there are $n = n^a + n^p$ degrees of freedom with the number of active joints (joints with prescribed motion) being n^a, the number of passive joints (joints with prescribed torques) being n^p, then $f: \mathbf{R}^n \times \mathbf{R}^n \times \mathbf{R}^{n^a} \times \mathbf{R}^{n^p} \to \mathbf{R}^{n^a} \times \mathbf{R}^{n^p}$. The derivatives of the motion needed are $\partial f / \partial q$, $\partial f / \partial \dot{q}$, $\partial f / \partial \ddot{q}^a$, and $\partial f / \partial \tau^p$.

We construct the following algorithm to compute $\partial f / \partial x$ where $x \in \{q_j, \dot{q}_j, \ddot{q}_j^a, \tau_j^p\}$. The index j denotes the joint number with which we are taking the derivative. It is interesting that this algorithm takes on slightly different forms depending upon x and j.

Algorithm 4.1. *Recursive gradient computation.*

• To calculate $\partial f / \partial x$: for joint j, let $x \in \{q_j, \dot{q}_j, \ddot{q}_j^a (j \in I^a), \tau_j^p (j \in I^p)\}$:

• - Initialization of Forward recursion

$$\text{Given: } V_0$$

- Forward recursion: for $i = 1 - n$ do

$$\frac{\partial V_i}{\partial x} = \begin{cases} 0 & i < j \\ \mathrm{ad}_{\mathrm{Ad}_{T_{i-1,i}^{-1}} (V_{i-1})} S_i & x = q_i \\ S_i & x = \dot{q}_i \\ 0 & x = \ddot{q}_i^a \text{ or } x = \tau_i^p \\ \mathrm{Ad}_{T_{i-1,i}^{-1}} \left(\dfrac{\partial V_{i-1}}{\partial x} \right) & i > j \end{cases}$$

$$\frac{\partial a_i}{\partial x} = \begin{cases} 0 & i < j \\ -\mathrm{ad}_{S_i} V_i - \mathrm{ad}_{S_i \dot{q}_i} \dfrac{\partial V_i}{\partial x} & x = \dot{q}_i \\ 0 & x \neq \dot{q}_i \\ -\mathrm{ad}_{S_i \dot{q}_i} \dfrac{\partial V_i}{\partial x} & i > j \end{cases}$$

$$\frac{\partial b_i}{\partial x} = \begin{cases} 0 & i < j \\ -\mathrm{ad}^*_{\partial V_i / \partial x} (J_i V_i) + \mathrm{ad}^*_{V_i} \left(J_i \left(\dfrac{\partial V_i}{\partial x} \right) \right) & i \geqslant j \end{cases}$$

- Initialization of Backward recursion

$$\text{Given: } z_{n+1}^+ = F_{n+1}, \hat{J}_{n+1}^+ = 0$$

- Backward recursion: for $i = n - 1$ do

$$h = T_{i,i+1}^{-1} \tag{62}$$

If $x = q_{i+1}$:

$$\frac{\partial \hat{J}_i}{\partial x} = \frac{\partial \mathrm{Ad}_h^*}{\partial x} \hat{J}_{i+1}^+ \mathrm{Ad}_h + \mathrm{Ad}_h^* \hat{J}_{i+1}^+ \frac{\partial \mathrm{Ad}_h}{\partial x} \tag{63}$$

$$+ \mathrm{Ad}_h^* \frac{\partial \hat{J}_{i+1}^+}{\partial x} \mathrm{Ad}_h \tag{64}$$

$$\frac{\partial B_i}{\partial x} = \mathrm{ad}^*_{\mathrm{Ad}_{M_i} - S_i} (\mathrm{Ad}_h^* (z_{i+1}^+)) \tag{65}$$

$$+ \mathrm{Ad}_h^* \left(\frac{\partial z_{i+1}^+}{\partial x} \right) + \frac{\partial b_i}{\partial x} \tag{66}$$

else:

$$\frac{\partial \hat{J}_i}{\partial x} = \mathrm{Ad}_h^* \frac{\partial \hat{J}_{i+1}^+}{\partial x} \mathrm{Ad}_h \tag{67}$$

$$\frac{\partial B_i}{\partial x} = \mathrm{Ad}_h^* \left(\frac{\partial z_{i+1}^+}{\partial x} \right) + \frac{\partial b_i}{\partial x} \tag{68}$$

end if.

$$\frac{\partial z_i}{\partial x} = \begin{cases} \dfrac{\partial \hat{J}_i}{\partial x} a_i + \hat{J}_i \left(\dfrac{\partial a_i}{\partial x} \right) + \dfrac{\partial B_i}{\partial x} & i \in I^p \\ \dfrac{\partial \hat{J}_i}{\partial x} (S_i \ddot{q}_i + a_i) + \hat{J}_i \left(S_i \dfrac{\partial \ddot{q}_i}{\partial x} + \dfrac{\partial a_i}{\partial x} \right) + \dfrac{\partial B_i}{\partial x} & i \in I^a \end{cases} \tag{69}$$

- Ω, ϕ, and $\hat{c}$ are used to simplify calculations for the passive joints:

$$\Omega_i = S_i^T \hat{J}_i S_i \tag{70}$$

$$\frac{\partial \Omega_i}{\partial x} = S_i^T \frac{\partial \hat{J}_i}{\partial x} S_i \tag{71}$$

$$\phi_i = \frac{\hat{J}_i S_i}{\Omega_i} \tag{72}$$

$$\frac{\partial \phi_i}{\partial x} = \frac{\Omega_i \frac{\partial \hat{J}_i}{\partial x} S_i - \hat{J}_i S_i \frac{\partial \Omega_i}{\partial x}}{\Omega_i^2} \tag{73}$$

$$\hat{c}_i = \frac{-S_i^T z_i + \tau_i}{\Omega} \tag{74}$$

$$\frac{\partial \hat{c}_i}{\partial x} = \frac{(S_i^T z_i + \tau_i)\frac{\partial \Omega_i}{\partial x} - \Omega\left(S_i^T \frac{\partial z_i}{\partial x} + \frac{\partial \tau_i}{\partial x}\right)}{\Omega^2} \tag{75}$$

$$\frac{\partial z_i^+}{\partial x} = \begin{cases} \frac{\partial z_i}{\partial x} - \frac{\partial \phi_i}{\partial x}(S_i^T z_i + \tau_i) - \phi_i\left(S_i^T \frac{\partial z_i}{\partial x} + \frac{\partial \tau_i}{\partial x}\right) & i \in I^p \\ \frac{\partial z_i}{\partial x} & i \in I^a \end{cases}$$

$$\frac{\partial \hat{J}_i^+}{\partial x} = \begin{cases} -\frac{\partial \phi_i}{\partial x} S_i^T \hat{J}_i + [I - \phi_i S_i^T] \frac{\partial \hat{J}_i}{\partial x} & i \in I^p \\ \frac{\partial \hat{J}_i}{\partial x} & i \in I^a \end{cases}$$

- Initialization of Forward recursion

$$\text{Given: } \dot{V}_0, \frac{\partial \dot{V}_0}{\partial x}$$

- Forward recursion: for $i = 1 - n$ do

$$\dot{V}_i^+ = \mathrm{Ad}_{T_{i,i+1}^{-1}}(\dot{V}_{i-1}) \tag{76}$$

$$\frac{\partial \dot{V}_i^+}{\partial x} = \begin{cases} \mathrm{Ad}_{T_{i,i+1}^{-1}}\left(\frac{\partial \dot{V}_{i-1}}{\partial x}\right) \\ \mathrm{Ad}_{T_{i,i+1}^{-1}}\left(\frac{\partial \dot{V}_{i-1}}{\partial x}\right) + \mathrm{ad}_{\dot{V}^+}(S_i) \quad x = q_j, \quad i = j \end{cases} \tag{77}$$

$$\dot{V}_i = \dot{V}_i^+ + S_i \ddot{q}_i + a_i \tag{78}$$

$$\frac{\partial \tau_i}{\partial x} = S_i^T\left(\hat{J}_i \frac{\partial \dot{V}_i^+}{\partial x} + \frac{\partial z_i}{\partial x} + \frac{\partial \hat{J}_i}{\partial x}\dot{V}_i^+\right) \quad i \in I^a \tag{79}$$

$$\frac{\partial \ddot{q}_i}{\partial x} = \frac{\partial \hat{c}_i}{\partial x} - \frac{\partial \phi_i}{\partial x}\dot{V}_i^+ - \phi_i \frac{\partial \dot{V}_i^+}{\partial x} \quad i \in I^p \tag{80}$$

$$\frac{\partial \dot{V}_i}{\partial x} = \frac{\partial \dot{V}_i^+}{\partial x} + S_i \frac{\partial \ddot{q}_i}{\partial x} + \frac{\partial a_i}{\partial x} \tag{81}$$

5 Example Applications: Optimal Control of Articulated Systems

In this section we present some results of our primary application of the dynamics algorithms developed in this paper-the solution of optimal control problems for articulated systems. Only relatively simple problems are discussed here. The details of the solution for these problems and more complex systems are given in [26,27].

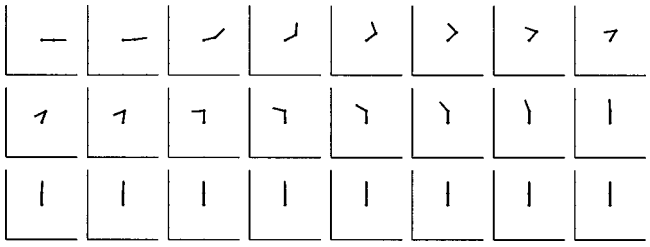

Fig. 1 Initial path for planar 2R problem

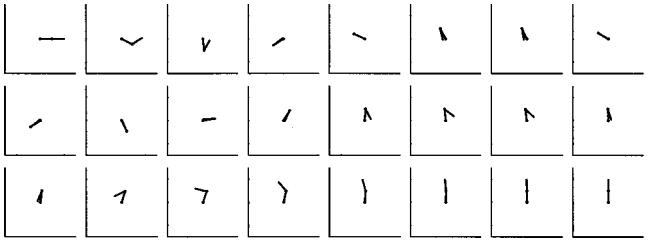

Fig. 2 Final path for planar 2R problem

Any reliable numerical algorithm for solving the following problems will need to solve the equations of motion many times with different parameterizations of the solution. In the search for an optimal solution, the algorithm will also need the gradient of the cost function. This will involve the derivatives of the dynamic equations as developed in this paper. The example solutions were obtained using the Cstorm package. In each example, we used B-spline polynomials to parameterize the motion of the active joints. The parameters of the splines were then varied in a constrained SQP optimization algorithm, implemented with Matlab's *constr* function. All of the gradients were computed analytically using the algorithms developed in this paper. Because of the nonlinear, nonconvex nature of these problems, all of the solutions are *locally* optimal. The initial and final active joint velocities were constrained to be equal to zero for these examples.

5.1 2R Planar Arm. For this example, we wanted to find the minimum effort motion which moved the two link planar system from the horizontal position to the vertical position. In this case both joints were actuated. A cost function of the form:

$$J = \frac{1}{2}\int_{t_o}^{t_f} \|\tau\|^2 dt \tag{82}$$

was used. Figure 1 shows the initial guess for the motion. The frames are spaced at equal intervals in time. We chose the initial motion in a manner that provided a smooth first guess for the motion and satisfied the boundary conditions. The initial value of the cost function was 73.57. Figure 2 shows the optimal (locally) motion we achieved using the Cstorm package. At first the robot allows gravity to take over and it swings down while folding up the second link. It then swings the first joint into the upward posture. A small pumping motion is applied to the second link in order to move it into the vertical posture. The final value of the cost function was 9.9.

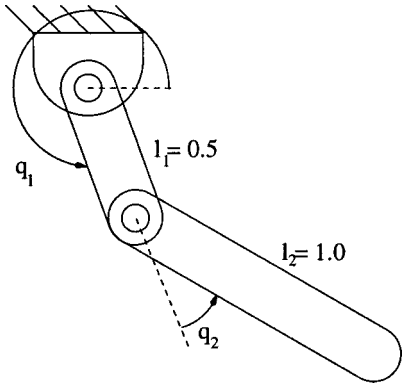

Fig. 3 Acrobot

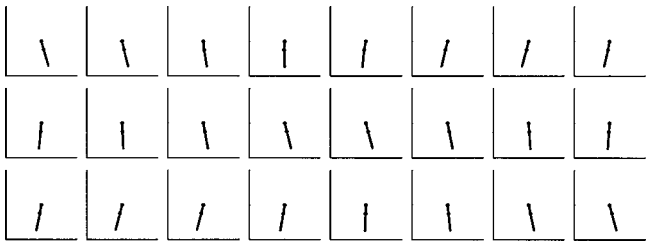

Fig. 4 Initial path for Acrobot swing-up motion

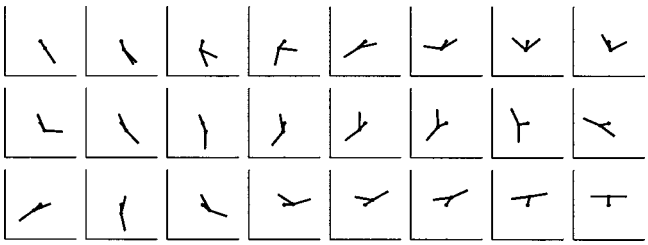

Fig. 9 Locally optimal swing-up for branched Acrobot

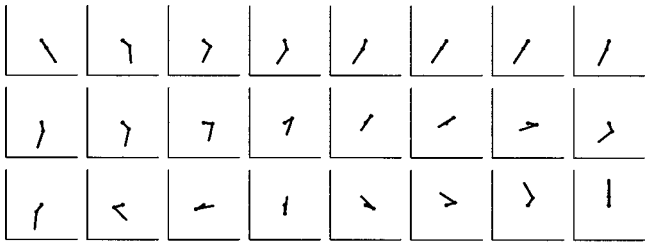

Fig. 5 Optimal swing up motion with $q_1(0) = -1.0$

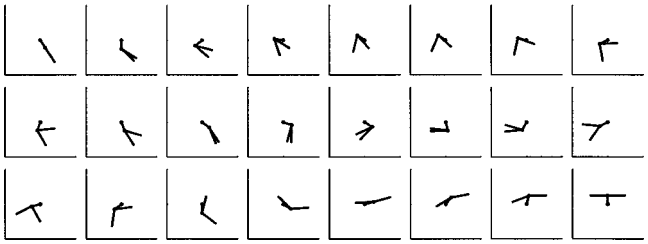

Fig. 10 Locally optimal swing-up for branched Acrobot with different initial condition

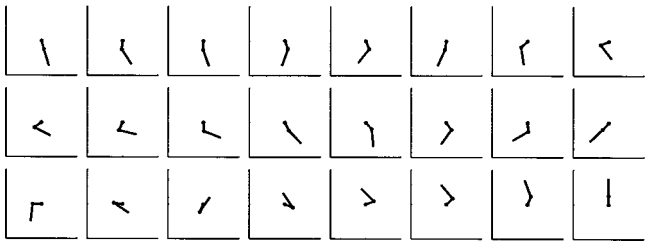

Fig. 6 Optimal swing up motion with $q_1(0) = -1.3$

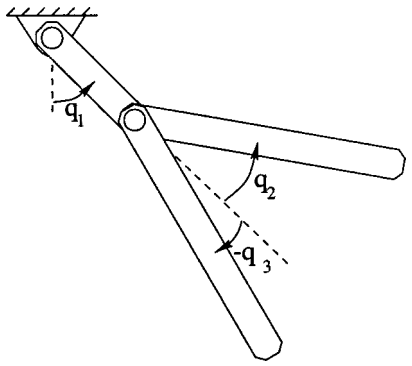

Fig. 7 Branched Acrobot

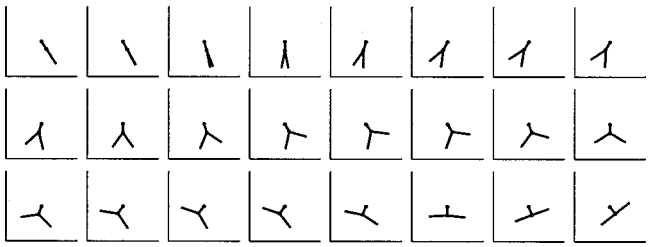

Fig. 8 Initial path for branched Acrobot swing-up problem

5.2 Acrobot. In this section we consider the swing-up motion for the Acrobot shown in Fig. 3. The Acrobot is a two-link robot with no motor at its base. Is has been widely studied by Spong [28] and others. In this example we prescribe the motion of the elbow joint in an attempt to drive the system from an initial hanging configuration to a vertically inverted configuration. Note that it is not apparent what elbow motion will drive the system to the desired final configuration. A cost function of the form

$$J = c_1\left(q_1(t_f) - \frac{\pi}{2}\right)^2 + c_2(\dot{q}_1(t_f))^2 + \frac{1}{2}\int_{t_o}^{t_f} \|\tau^a\|^2 dt \qquad (83)$$

was used to produce the desired motion. The constants c_1 and c_2 were used to drive the passive base joint to the vertical position ($q_1 = \pi/2$) with zero velocity. The integral term was used to penalize the torque on the active joint used to produce the motion.

Figure 4 shows the initial guess for the motion. Notice that this guess is very poor. The passive base joint does not even begin to swing up into the vertical position. Figure 5 shows the final motion obtained using Cstorm for the initial condition $q_1(0) = -1.0$, $\dot{q}(0) = q_2(0) = \dot{q}_2(0) = 0.0$. This produces a motion similar to those proposed by Spong [28], in which the lower link pumps energy into the system and this energy causes the first link to move into the vertical position. Figure 6 shows a slightly different motion obtained with the different initial condition of $q_1(0) = -1.3$, $\dot{q}(0) = q_2(0) = \dot{q}_2(0) = 0.0$.

5.3 Branched Acrobot. For the final example we examine the simple branched chain system shown in Fig. 7. This system is similar to the Acrobot in that it has a passive base joint, but it has two ''legs'' which can be used to pump energy into the system. The same cost function (83) was used for this system as for the acrobot since the desired final position of the unactuated base joint is the same for both cases. However, the two legs of the branched chain make the dynamics of the two systems very different. There are now two active joints, both of which contribute to the torque term in the integral. Figure 8 shows the initial guess for the swing up motion for the initial conditions of $q_1(0) = -1.0$ and all the other positions and velocities equal to zero. Figure 9 shows the local minimum that was obtained. Figure 10 shows another local minimum obtained from a different initial guess.

6 Conclusions

In this paper we presented a Lie group formulation for the recursive dynamics of underactuated tree topology systems. We note that prismatic, revolute, and screw type joints can all be written in the same form using matrix exponentials. This allows a relatively straight forward derivation of the dynamics algorithm using spatial velocities and wrenches. The exponentials also allowed us to explicitly compute the derivatives of the dynamics algorithm. One important application that needs these derivatives is dynamic motion planning. Several simple example problems were solved for fully and underactuated systems. The dynamics algorithms developed in this paper provide an essential element for the analysis and trajectory planning for many of today's complex articulated machines.

References

[1] Pandy, M. G., and Anderson, F. C., 2000, ''Dynamic simulation of human movement using large-scale models of the body,'' Proceedings of the 2000 IEEE Conference on Robotics and Automation (San Francisco, CA), IEEE, Apr., pp. 676–680.

[2] van-de Panne, M., Laszlo, J., Huang, P., and Faloutsos, P., 2000, ''Towards agile animated characters,'' Proceedings of the 2000 IEEE Conference on Robotics and Automation, San Francisco, CA, IEEE, Apr. pp. 682–687.

[3] Hooker, W. W., and Margulies, G., 1965, ''The dynamical attitude equations for an *n*-body satellite,'' J. Astronaut. Sci., **12**, No. 4, pp. 123–128.

[4] Uicker, J. J., 1965, ''On the dynamic analysis of spatial linkages using 4×4 matrices,'' Ph.D. thesis, Northwestern University.

[5] Stepanenko, Y., and Vukobratovic, M., 1976, ''Dynamics of articulated open-chain active mechanisms,'' Math. Biosci., **28**, No. 1–2, pp. 137–170.

[6] Orin, D., McGhee, R., Vukobratovic, M., and Hartoch, G., 1979, ''Kinematic and kinetic analysis of open-chain linkages utilizing newton-euler methods,'' Math. Biosci., **43**, No. 1–2, pp. 107–130.

[7] Luh, J., Walker, M., and Paul, R., 1980, ''On-line computational scheme for mechanical manipulators,'' ASME J. Dyn. Syst., Meas., Control, **102**, No. 2, pp. 69–76.

[8] Hollerbach, J. M., 1980, ''A recursive lagrangian formulation of manipulator dynamics and a comparative study of dynamics formulation complexity,'' IEEE Trans. Syst. Man Cybern., **10**, No. 11, pp. 730–736.

[9] Silver, W. M., 1982, ''On the equivalance of lagrangian and newton-euler dynamics for manipulators,'' Int. J. Robot. Res., **1**, No. 2, pp. 118–128.

[10] Balafoutis, C. A., Misra, P., and Patel, R. V., 1986, ''Recursive evaluation of linearized robot dynamic models,'' IEEE J. Rob. Autom., **RA-2**, No. 3, pp. 146–155.

[11] Murray, J. J., and Neuman, C. P., 1986, ''Linearization and sensitivity models of the newton-euler dynamic robot model,'' ASME J. Dyn. Syst., Meas., Control, **108**, No. 3, pp. 272–276.

[12] Martin, B. J., and Bobrow, J. E., 1999, ''Minimum effort motions for open-chain manipulators with task-dependent end-effector constraints,'' Int. J. Robot. Res., **18**, No. 2, pp. 213–224.

[13] Featherstone, R., 1987, *Robot dynamics algorithms*, Kluwer, Boston.

[14] Rodriguez, G., 1987, ''Kalman filtering, smoothing and recursive robot arm forward and inverse dynamics,'' IEEE J. Rob. Autom., **RA-3**, No. 6, pp. 510–521.

[15] Rodriguez, G., Jain, A., and Kreutz-Delgado, K., 1991, ''A spatial operator algebra for manipulator modeling and control,'' Int. J. Robot. Res., **5**, No. 4, pp. 510–521.

[16] Jain, A., and Rodriguez, G., 1993, ''An analysis of the kinematics and dynamics of underactuated manipulators,'' IEEE J. Rob. Autom., **9**, No. 4, pp. 411–421.

[17] Featherstone, R., 1999, ''A divide-and-conquer articulated-body algorithm for parallel $o(\log(n))$ calculation of rigid body dynamics. part 1: Basic algorithm,'' Int. J. Robot. Res., **18**, No. 9, pp. 867–875.

[18] Park, F. C., Bobrow, J. E., and Ploen, S. R., 1995, ''A lie group formulation of robot dynamics,'' Int. J. Robot. Res., **14**, No. 6, pp. 609–618.

[19] Brockett, R. W., 1983, ''Robotic manipulators and the product of exponentials formula,'' Proc. of Int. Symposium on the Mathematical Theory of Networks and Systems, Beer Sheba, Israel, pp. 120–129.

[20] Li, Z., 1989, ''Kinematics, planning and control of dextrous robot hands,'' Ph.D. thesis, University of California, Berkeley.

[21] Ploen, S. R., and Park, F. C., 1999, ''Coordinate invariant algorithms for robot dynamics,'' IEEE Trans. Rob. Autom., **15**, No. 6, pp. 1130–1135.

[22] Chen, IM, and Yang, G., 1998, ''Automatic model generation for modular reconfigurable robot dynamics,'' ASME J. Dyn. Syst., Meas., Control, **120**, No. 3, pp. 346–352.

[23] Chen, IM, Yeo, S. H., Chen, G., and Yang, G., 1999, ''Kernel for modular robot applications: automatic modeling techniques,'' Int. J. Robot. Res., **18**, No. 2, pp. 225–242.

[24] Spivak, M., 1965, *Calculus on manifolds*, The Benjamin/Cummings Publishing Co.

[25] Murray, R. M., Li, Z., and Sastry, S. S., 1994, *A mathematical introduction to robotic manipulation*, CRC Press.

[26] Sohl, G. A., and Bobrow, J. E., 1999, ''Optimal motions for underactuated manipulators,'' ASME Design Technical Conferences, Las Vegas, Nevada, ASME, Sept.

[27] Wang, C-Y. E., Timoszyk, W. K., and Bobrow, J. E., 1999, ''Weightlifting motion planning for a puma 762 robot,'' Proceedings of the 1999 IEEE Conference on Robotics and Automation, Detroit, MI, IEEE, May, pp. 480–485.

[28] Spong, M. W., 1994, ''Swing up control of the acrobot,'' Proceedings 1994 IEEE International Conference on Robotics and Automation, Los Alamitos, CA, IEEE, May, pp. 2356–2361.