Discrete and Continuous Mechanics for Tree Representations of Mechanical Systems

Elliot R. Johnson and Todd D. Murphey
Electrical and Computer Engineering
University of Colorado at Boulder
Boulder, Colorado 80309
erjohnso@colorado.edu and murphey@colorado.edu

***Abstract*— We use a tree-based structure to represent mechanical systems comprising interconnected rigid bodies. Using this representation, we derive a simple algorithm to numerically calculate forward kinematic maps, body velocities, and their derivatives. The algorithm is computationally efficient and scales to large systems very well by using recursion to take advantage of the tree structure. Moreover, this method is less prone to modeling errors because each element of the graph is simple.**

The tree representation provides a natural framework to simulate mechanical dynamics with numeric computations rather than large symbolically-derived equations. In particular, the representation allows one to simulate systems in generalized coordinates using Lagrangian dynamics without symbolically finding the equations of motion. This method also applies to the relatively new variational integrators which numerically integrate dynamics in a way that preserve momentum and other symmetries. We show how to implement both integration schemes for an arbitrary system of interconnected rigid bodies in a computationally efficient way while avoiding symbolic equations of motion. We end with an example simulating a marionette; a mechanically complex, high degree-of-freedom system.

I. INTRODUCTION

Euler-Lagrange simulations are often preferred for robotics and controls applications because they work directly in the generalized coordinates used to analyze the system and specify desired trajectories. Moreover, they provide a certain level of automation to derive correct equations of motion. For complex mechanical systems, however, the necessary symbolic equations become unwieldy and the resulting simulations are typically slow.

The popular alternative to Euler-Lagrange simulation is the Newton-Euler force balance approach. Newton-Euler simulations have a larger configuration space than Euler-Lagrange simulations (typically $(\mathbb{R}^3 \times \mathrm{SO}(3))^n$ vs. generalized coordinates) and hide the mechanical structure in the constraint definitions. However, there are extremely fast and efficient implementations [1] [13]. Much of the performance comes from working with simple, general equations that can be efficiently implemented and evaluated rather than deriving large symbolic equations of motion. These smaller equations can be efficiently implemented and optimized by compilers.

Euler-Lagrange simulations [2] [9], on the other hand, typically rely on symbolic equations of motions that are numerically integrated. Our approach avoids such symbolic computation by representing a mechanical system as a hierarchical tree structure. The representation leads to simple recursive equations that can be used to numerically simulate an arbitrary system of rigid bodies without deriving symbolic equations. This method yields excellent performance while the simulation still takes place in generalized coordinates. Although we do not discuss it here due to space limitations, constraints (both holonomic and nonholonomic) and external generalized forces can also be included in a very straightforward fashion.

The tree representation is an intuitive description of many mechanical systems like robots or humans. They are used extensively in motion capture and computer animation and have been applied to dynamics[10][3]. The use of recursion greatly improves the simulation's efficiency[4]. In a simulation of a robotic arm, for example, the equations of motion for the hand and forearm will both implicitly include the kinematics of the upper arm. In a recursive approach, the dependence on the upper arm becomes explicit. We can then perform its calculations only once and reuse them when needed. The recursive derivation automatically makes these and similar optimizations wherever appropriate, even in non-obvious places.

The tree representation is not tied to Euler-Lagrange simulation. It provides a foundation for efficiently calculating fundamental quantities used in many methods. In particular, we can implement a variational integrator for arbitrary systems of rigid bodies.

Variational integrators are a powerful new tool for simulating dynamics [8] [7]. They are a class of integrators derived using a specific method that introduces the variational principle after the continuous-to-discrete approximation. In contrast, the traditional approach introduces an approximation after the variational principle by numerically integrating a continuous ordinary differential equation (ODE). Variational integrators preserve important symmetries, including momentum, for conservative systems. Unlike ODE methods, they are also well suited for systems with impacts and non-smooth mechanics.

II. TREE REPRESENTATIONS

A tree representation is a tree sub-graph of a graph that describes all the interconnections between rigid bodies.

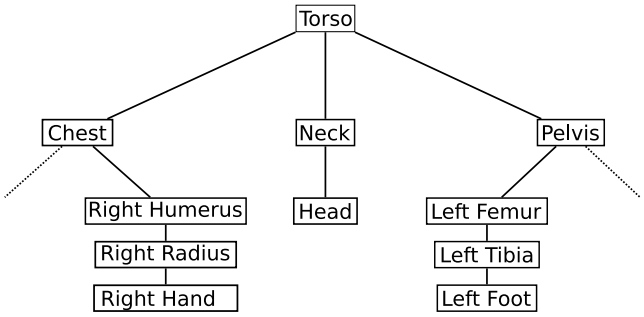


Fig. 1. A humanoid system is naturally described using the tree representation.

We represent the mechanical system as a collection of coordinate frames with a hierarchical organization. Each frame is defined as a rigid body transformation (typically depending on parameters) relative to a parent frame. Hence, all tree sub-graphs provide generalized coordinates for a given mechanical system; whatever parameters the rigid body transformations depend on are these coordinates. (We assume that we have already chosen a tree representation.) Frames can have an unlimited number of child frames. At the top of the hierarchy is the stationary world coordinate frame. Masses can be attached to each frame in the system. A simplified example of a tree representing a mechanical system is shown in Fig. 1.

The rigid body transformations are defined and represented using homogeneous coordinate transformations in $\mathrm{SE}(3)$. See [9] for an in-depth discussion.

Table I defines the notation used throughout the paper. We will typically drop the explicit dependence on q and q_i for visual clarity (e.g. $g_{s,i}$ rather than $g_{s,i}(q)$).

We make the following assumptions about the mechanical system.

1) Frames are related to their parents through six basic transformations: Translations along the parent's X, Y, and Z axes and rotations about the parent's X, Y, and Z axes.
2) Each configuration variable drives only one transformation.
3) Masses are always attached to a frame's origin with the principle axes aligned with the frame's axes.

These assumptions are not required for this approach. They are included only to simplify the resulting equations.

Assumption 1 implies that each transformation depends on only one real number. For moving joints, they depend on a single real-valued configuration variable. Fixed transformations are parameterized by constants.

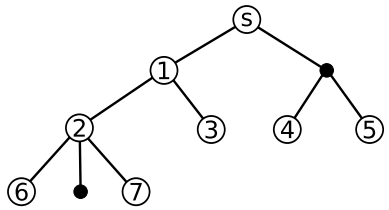
The biggest drawback of assumption 1 is forcing three-dimensional rotations to be parameterized by three Euler angles rather than a global $\mathrm{SO}(3)$ parameterization. If a global representation of $\mathrm{SO}(3)$ is required, this assumption can be removed and the equations re-derived.

Assumption 2 prevents using the same configuration variable to drive multiple transformations, which is useful for modeling systems with parallel linkages. Parallel linkages and closed chains can instead be handled by connecting open chains with constraints.

Assumption 3 guarantees that the inertia matrix, M_i, for each mass is diagonal and that the kinetic energy can be expressed as $v_i^{bT} M_i v_i^b$ [9].

TABLE I

NOTATION USED FOR THE TREE REPRESENTATION.



ⓢ Spatial Frame

● Constant Transformation

ⓘ Variable Transformation depending on q_i

$q_i(t) \in \mathbb{R}$	Configuration variable of the i-th frame.
$q(t) \in \mathbb{R}^n$	Configuration vector comprising q_0, q_1, ... , q_n.
m_i	Mass of the i-th frame.
$M_i \in \mathbb{R}^{6x6}$	Inertia Tensor of the i-th mass in body frame coordinates.
$g_{s,i}(q) \in \mathrm{SE}(3)$	Configuration of the i-th frame relative to the spatial reference frame.
$g_{j,i}(q) \in \mathrm{SE}(3)$	Configuration of the i-th frame relative to the j-th frame.
$g_i(q_i) \in \mathrm{SE}(3)$	Configuration of the i-th frame to its parent frame.
$v_i^b \in T_e SE(3)$	Body velocity (i.e., an element of the Lie algebra) of the i-th frame relative to the spatial reference frame.
$p_{s,i}(q) \in \mathbb{R}^3$	Position of the i-th frame relative to the spatial reference frame.
$\mathrm{anc}(i)$	The ancestors of the the i-th frame are the frames passed while moving up the tree from the i-th frame to the spatial frame. For example, frame 6 has ancestors $\{1, 2, s\}$.
$\mathrm{par}(i)$	Immediate parent frame of the i-th frame. For example, the parent of frame 6 is frame 2.

III. Frame Positions and Derivatives

Given the above representation, the configuration of any frame with respect to the spatial reference frame is easily calculated. For the spatial reference frame, the configuration is the identity transformation, I. Otherwise, the configuration of frame k is the parent frame's configuration $g_{s,par(k)}$ transformed by the local transformation g_k. This results in a piecewise expression for $g_{s,k}(q)$[1]:

$$g_{s,k}(q) = \begin{cases} I & k = s \\ g_{s,\mathrm{par}(k)} g_k & k \neq s \end{cases} \tag{1}$$

[1]The piecewise equations presented in this paper have a specific ordering. For cases that can be simultaneously satisfied, the top-most case always takes precedence.

Eq. (1) defines a recursive algorithm to numerically calculate the configuration of any frame in the system. We can also find derivatives of frame configurations.

$$\frac{\partial}{\partial q_i} g_{s,k}(q) = \begin{cases} \frac{\partial}{\partial q_i} I & k = s \\ \frac{\partial}{\partial q_i}\left[g_{s,\mathrm{par}(k)} g_k\right] & k \neq s \end{cases} = \begin{cases} 0 & k = s \\ \frac{\partial}{\partial q_i}\left[g_{s,\mathrm{par}(k)}\right] g_k & \\ \quad + g_{s,\mathrm{par}(k)} \frac{\partial}{\partial q_i} g_k & k \neq s \end{cases} \tag{2}$$

Assumption 1 guarantees $\frac{\partial}{\partial q_i} g_k(q) = 0$ when $i \neq k$. Assumption 2 implies $\frac{\partial}{\partial q_i} g_{s,\mathrm{par}\,i}(q) = 0$ since $g_i(q)$ is the only frame that depends on q_i.

$$\frac{\partial}{\partial q_i} g_{s,k}(q) = \begin{cases} 0 & k = s \\ \frac{\partial}{\partial q_i} g_{s,\mathrm{par}(k)} g_k & i \neq k \\ g_{s,\mathrm{par}(k)} g_k' & i = k \end{cases} \tag{3}$$

where $g_k' = \frac{\partial}{\partial q_k} g_k(q_k)$. The second derivative is calculated using a similar procedure.

$$\frac{\partial^2}{\partial q_j \partial q_i} g_{s,k}(q) = \begin{cases} & k = s \\ 0 & i \notin \mathrm{anc}(k) \\ & j \notin \mathrm{anc}(k) \\ g_{s,\mathrm{par}(k)} g_k'' & i = k = j \\ \frac{\partial}{\partial q_j} g_{s,\mathrm{par}(k)} g_k' & i = k \neq j \\ \frac{\partial}{\partial q_i} g_{s,\mathrm{par}(k)} g_k' & i \neq k = j \\ \frac{\partial^2}{\partial q_j \partial q_i} g_{s,\mathrm{par}(k)} g_k & i \neq k \neq j \end{cases} \tag{4}$$

Eq. (1), (3), (4) allow us to calculate the configurations of any coordinate frames in the system along with their first and second derivatives with respect to any configuration variable. Note that these are exact derivatives and not approximations.

IV. Frame Body Velocities and Derivatives

The body velocity of a frame is the velocity of its parent frame transformed into the local coordinates plus the velocity of the frame with respect to the parent:

$$\hat{v}_k^b(q, \dot{q}) = \begin{cases} 0 & k = s \quad \text{(5a)} \\ g_k^{-1} \hat{v}_{\mathrm{par}(k)}^b g_k + g_k^{-1} \dot{g}_k & k \neq s \quad \text{(5b)} \end{cases}$$

where $\dot{g}_k = \frac{\partial}{\partial t} g_k$. Assumption 1 allows the second term in (5b) to be expressed using a twist $\hat{\xi}$ [9]:

$$\hat{v}_k^b(q, \dot{q}) = \begin{cases} 0 & k = s \\ g_k^{-1} \hat{v}_{\mathrm{par}(k)}^b g_k + \hat{\xi}_k \dot{q}_k & k \neq s \end{cases} \tag{6}$$

Note that the expression $g_k^{-1} \hat{v}_{\mathrm{par}(k)}^b g_k$ could be replaced by an $Ad_{g_k} v_{par(k)}^b$ transformation. Indeed, all the following equations can be modified to use their intrinsic counterparts. However, we focus here on as transparent an approach as possible and do not use any differential geometric formality. Derivatives of the body velocities are straightforward to find.

$$\frac{\partial}{\partial q_i} \hat{v}_k^b(q, \dot{q}) = \begin{cases} 0 & k = s \\ & i \notin \mathrm{anc}(k) \\ g_k^{-1\prime} \hat{v}_{\mathrm{par}(k)}^b g_k + g_k^{-1} \hat{v}_{\mathrm{par}(k)}^b g_k' & i = k \\ g_k^{-1} \frac{\partial}{\partial q_i} \hat{v}_{\mathrm{par}(k)}^b g_k & i \neq k \end{cases} \tag{7}$$

$$\frac{\partial^2}{\partial q_j \partial q_i} \hat{v}_k^b(q, \dot{q}) = \begin{cases} & k = s \\ 0 & i \notin \mathrm{anc}(k) \\ & j \notin \mathrm{anc}(k) \\ g_k^{-1\prime\prime} \hat{v}_{\mathrm{par}(k)}^b g_k + 2 g_k^{-1\prime} \hat{v}_{\mathrm{par}(k)}^b g_k' & \\ \quad + g_k^{-1} \hat{v}_{\mathrm{par}(k)}^b g_k'' & i = k = j \\ g_k^{-1\prime} \frac{\partial}{\partial q_i} \hat{v}_{\mathrm{par}(k)}^b g_k + g_k^{-1} \frac{\partial}{\partial q_i} \hat{v}_{\mathrm{par}(k)}^b g_k' & i \neq k = j \\ g_k^{-1\prime} \frac{\partial}{\partial q_j} \hat{v}_{\mathrm{par}(k)}^b g_k + g_k^{-1} \frac{\partial}{\partial q_j} \hat{v}_{\mathrm{par}(k)}^b g_k' & i = k \neq j \\ g_k^{-1} \frac{\partial^2}{\partial q_j \partial q_i} \hat{v}_{\mathrm{par}(k)}^b g_k & i \neq k \neq j \end{cases} \tag{8}$$

$$\frac{\partial^2}{\partial \dot{q}_j \partial q_i} \hat{v}_k^b(q, \dot{q}) = \begin{cases} & k = s \\ & k = j, \\ 0 & i \notin \mathrm{anc}(k) \\ & j \notin \mathrm{anc}(k) \\ g_k^{-1\prime} \frac{\partial}{\partial \dot{q}_j} \hat{v}_{\mathrm{par}(k)}^b g_k + g_k^{-1} \frac{\partial}{\partial \dot{q}_j} \hat{v}_{\mathrm{par}(k)}^b g_k' & k = i \\ g_k^{-1} \frac{\partial^2}{\partial \dot{q}_j \partial q_i} \hat{v}_{\mathrm{par}(k)}^b g_k & k \neq i \end{cases} \tag{9}$$

$$\frac{\partial}{\partial \dot{q}_i} \hat{v}_k^b(q, \dot{q}) = \begin{cases} 0 & k = s, \\ & i \notin \mathrm{anc}(k) \\ \hat{\xi}_k & i = k \\ g_k^{-1} \frac{\partial}{\partial \dot{q}_i} \hat{v}_{\mathrm{par}(k)}^b g_k & i \neq k \end{cases} \tag{10}$$

$$\frac{\partial^2}{\partial \dot{q}_j \partial \dot{q}_i} \hat{v}_k^b(q, \dot{q}) = 0 \tag{11}$$

Eq. (1) through (11) define recursive algorithms for calculating quantities that are needed to simulate a mechanical system. These equations are derived using the normal derivative. If an application requires higher derivatives, we can continue finding expressions without changing the approach. The new expressions are still numerically evaluated and can take advantage of previously computed values through recursion.

V. Primitive Transformations

Many of the equations include terms that we have not explicitly shown how to calculate (ie, g_k', g_k'', g_k^{-1}, $g_k^{-1\prime}$, $g_k^{-1\prime\prime}$, and ξ_k). These are found manually for each of the primitive transforms (Assumption 1). For the transformations used here, they take on simple forms. For example, the inverse of any primitive transformation is trivial: $g^{-1}(x) = g(-x)$.

Terms involving p_k and derivatives are trivially obtained from the related $g_{s,k}$ calculations. p_k comprises the x, y, and z coordinates of a frame in the spatial frame. These are simply extracted from the appropriate $g_{s,k}$ matrix. The derivatives of p_k (as appear in (16)) are similarly extracted from corresponding derivatives of $g_{s,k}$.

VI. Continuous Lagrangian Dynamics

The Euler-Lagrange equation provides a somewhat automated way to generate the equations of motion for a mechanical system. Given a system with generalized coordinate vector q, the dynamics are given by the Euler-Lagrange equation:

$$\frac{\partial}{\partial t}\frac{\partial L}{\partial \dot{q}}(q,\dot{q}) - \frac{\partial L}{\partial q}(q,\dot{q}) = \vec{u}(q,\dot{q},t) \tag{12}$$

where $L(q,\dot{q})$ is the system's Lagrangian and $\vec{u}(q,\dot{q},t)$ are externally applied forces expressed in the configuration coordinates. We can expand this equation using the chain rule.

$$\frac{\partial^2 L}{\partial \dot{q}\partial \dot{q}}\ddot{q} + \frac{\partial^2 L}{\partial q \partial \dot{q}}\dot{q} - \frac{\partial L}{\partial q} = \vec{u}(q,\dot{q},t) \tag{13}$$

where the dependence on q and $\dot{q}$ has been dropped. If the operator $\frac{\partial^2 L}{\partial \dot{q}\partial \dot{q}}$ is invertible[2], (13) can be solved to find $\ddot{q}$:

$$\ddot{q} = \left(\frac{\partial^2 L}{\partial \dot{q}\partial \dot{q}}\right)^{-1}\left(\vec{u}(q,\dot{q},t) + \frac{\partial L}{\partial q} - \frac{\partial^2 L}{\partial q \partial \dot{q}}\dot{q}\right) \tag{14}$$

If we can evaluate all of the terms in (14), we can integrate the equation to simulate the system over a period of time. For complex systems, we typically generate a symbolic expression for Lagrangian and find (14) with symbolic algebra software. This does not scale well. The equations tend to grow quickly with the number of bodies in the system. They are difficult to work with and are slow to evaluate.

Alternatively, we can evaluate the terms of (14) numerically using the tree representation. Not only does this avoid symbolic calculations, but it also takes advantage of the tree representation's efficiency.

A. The Continuous Lagrangian

To bridge the gap between (14) and (1)-(11), we expand the Lagrangian:

$$\begin{aligned} L(q,\dot{q}) &= \text{Kinetic Energy}(q,\dot{q}) - \text{Potential Energy}(q) \\ &= \sum_{k=0}^{\text{masses}} \tfrac{1}{2} v_k^{bT} M_k v_k^b + m_l \vec{g}\cdot p_k \end{aligned} \tag{15}$$

where $\vec{g}$ is the gravitational force vector.

We will need to find $\frac{\partial L}{\partial q_i}(q,\dot{q})$ to evaluate (14):

$$\begin{aligned} \frac{\partial L}{\partial q_i} &= \frac{\partial}{\partial q_i}\left[\sum_{k=0}^{\text{masses}} \tfrac{1}{2} v_k^{bT} M_k v_k^b - m_l \vec{g}\cdot p_k\right] \\ &= \sum_{k=0}^{\text{masses}} \tfrac{1}{2}\frac{\partial}{\partial q_i} v_k^{bT} M_k v_k^b + \tfrac{1}{2} v_k^{bT} M_k \frac{\partial}{\partial q_i} v_k^b - \\ &\qquad m_l \vec{g}\cdot\frac{\partial}{\partial q_i} p_k \\ &= \sum_{k=0}^{\text{masses}} v_k^{bT} M_k \frac{\partial}{\partial q_i} v_k^b + m_l \vec{g}\cdot\frac{\partial}{\partial q_i} p_k \end{aligned} \tag{16}$$

[2]This is the system's inertia matrix expressed in generalized coordinates.

Similar derivations yield the remaining expressions that are needed:

$$\frac{\partial^2 L}{\partial q_i \partial \dot{q}_j} = \sum_{k=0}^{\text{masses}} \frac{\partial v_k^{bT}}{\partial \dot{q}_j} M_k \frac{\partial v_k^b}{\partial q_i} + v_k^{bT} M_k \frac{\partial^2 v_k^b}{\partial q_i \partial \dot{q}_j} \tag{17}$$

$$\frac{\partial^2 L}{\partial \dot{q}_i \partial \dot{q}_j} = \sum_{k=0}^{\text{masses}} \frac{\partial v_k^{bT}}{\partial \dot{q}_j} M_k \frac{\partial v_k^b}{\partial \dot{q}_i} + v_k^{bT} M_k \frac{\partial^2 v_k^b}{\partial \dot{q}_i \partial \dot{q}_j} \tag{18}$$

Assuming we can numerically evaluate $\vec{u}(q,\dot{q},t)$, these equations allow us to calculate $\ddot{q}$ in (14) given t, q, and $\dot{q}$ without symbolically deriving the equations of motion. Constraints can also be imposed using this method with no modifications, but are not shown for space limitations.

VII. Discrete Lagrangian Dynamics

There has recently been a great deal of research in novel methods of numeric integration for mechanical systems. A result of this research is a class of integrators called variational integrators, so named because they compute state updates directly from a variational principle. Variational integrators conserve (or nearly conserve, depending on the integrator) structural quantities like momentum and energy [8]. They are also well suited for problems involving impacts and non-smooth phenomenon [5].

In discrete mechanics, we find a sequence $\{(t_0,q_0),(t_1,q_1),\ldots,(t_n,q_n)\}$ that approximates the actual trajectory of a mechanical system ($q_k \approx q(t_k)$). For simplicity, we assume a constant time-step ($t_{k+1} - t_k = \Delta t \ \forall\ k$), but in general, the time-step can be varied to use adaptive time-stepping algorithms.

The variational integrator is derived by defining the discrete Lagrangian to approximate the action integral over a short interval.

$$L_d(q_k,q_{k+1}) \approx \int_{t_k}^{t_{k+1}} L(q(\tau),\dot{q}(\tau))\mathrm{d}\tau \tag{19}$$

Using the discrete Lagrangian, the system's action integral is replaced with an action sum.

$$\begin{aligned} S(q([t_0,t_f]) &= \int_{t_0}^{t_f} L(q(\tau),\dot{q}(\tau))\mathrm{d}\tau \\ &\approx \sum_{k=0}^{n-1} L_d(q_k,q_{k+1}) \end{aligned} \tag{20}$$

Minimizing (20) with a discrete variational principle leads to an implicit difference equation known as the discrete Euler-Lagrange (DEL) equation[3]:

$$D_1 L_d(q_k,q_{k+1}) + D_2 L_d(q_{k-1},q_k) = 0 \tag{21}$$

This equation uses the previous two states to find the next state, similarly to how the continuous Euler-Lagrange equations are integrated forward. The simulation finds q_{k+1} by using a root-finding algorithm to solve (21). The time index, k, is incremented and the process repeats. Note that

[3]$D_n f(\ldots)$ is the derivative of $f(\ldots)$ with respect to its n-th argument. This is sometimes called a *slot derivative*

the derivation of (21) is exactly analogous to the variational principle used to derive (12).

We have omitted forcing and constraints for clarity, but the variational integrator framework is capable of handling both [8] [12], and will be included in the expanded version of this work.

A. Numeric Variational Integration

Implementations of variational integrators are currently derived symbolically. For complicated mechanical systems, the equations tend to scale better than the corresponding continuous differential equations, but still become too large for practical use. The tree representation allows us to instead implement variational integrators numerically.

As an example, we consider a simple variational integrator that uses a generalized midpoint approximation for the discrete Lagrangian.

$$L_d(q_k, q_{k+1}) = L\left(q_k + \alpha\left(q_{k+1} - q_k\right), \frac{q_{k+1}-q_k}{\Delta t}\right)\Delta t \quad (22)$$

where $\alpha \in [0,1]$ is an algorithm parameter and $\alpha = \frac{1}{2}$ leads to second order accuracy [12].

Continuing with our typical approach, (22) is used in (21) and expanded with the chain rule.

$$\begin{aligned} &D_1 L_d\left(q_k, q_{k+1}\right) = \\ &\frac{\partial L}{\partial q}\left(q_k + \alpha\left(q_{k+1} - q_k\right), \frac{q_{k+1}-q_k}{\Delta t}\right)(1-\alpha)\Delta t \\ &- \frac{\partial L}{\partial \dot{q}}\left(q_k + \alpha\left(q_{k+1} - q_k\right), \frac{q_{k+1}-q_k}{\Delta t}\right) \end{aligned} \quad (23)$$

$$\begin{aligned} &D_2 L_d\left(q_k, q_{k+1}\right) = \\ &\frac{\partial L}{\partial q}\left(q_k + \alpha\left(q_{k+1} - q_k\right), \frac{q_{k+1}-q_k}{\Delta t}\right)\alpha\Delta t \\ &+ \frac{\partial L}{\partial \dot{q}}\left(q_k + \alpha\left(q_{k+1} - q_k\right), \frac{q_{k+1}k-q_k}{\Delta t}\right) \end{aligned} \quad (24)$$

Eq. (23) and (24) allow us to evaluate (21) numerically using the values derived earlier for the continuous equations.

Eq. (21) is solved for q_{k+1} using a numeric root-finding algorithm which, in general, need derivatives of the function to work. The standard Newton-Raphson method, for example, uses the gradient of the function to find the step direction [11]. We can directly calculate the derivative instead of resorting to a numeric approximation. First, we define the function to be solved:

$$f(q_{k+1}) = D_1 L_d\left(q_k, q_{k+1}\right) + D_2 L_d\left(q_{k-1}, q_k\right) \quad (25)$$

The derivative is then

$$\begin{aligned} &Df(q_{k+1}) = D_2 D_1 L_d\left(q_k, q_{k+1}\right) \\ &= \frac{\partial^2 L}{\partial q \partial q}\left(q_k + \alpha\left(q_{k+1} - q_k\right), \frac{q_{k+1}-q_k}{\Delta t}\right)(1-\alpha)\alpha\Delta t \\ &+ \frac{\partial^2 L}{\partial \dot{q} \partial q}\left(q_k + \alpha\left(q_{k+1} - q_k\right), \frac{q_{k+1}-q_k}{\Delta t}\right)(1-\alpha) \\ &- \frac{\partial^2 L}{\partial q \partial \dot{q}}\left(q_k + \alpha\left(q_{k+1} - q_k\right), \frac{q_{k+1}-q_k}{\Delta t}\right)\alpha \\ &- \frac{\partial^2 L}{\partial \dot{q} \partial \dot{q}}\left(q_k + \alpha\left(q_{k+1} - q_k\right), \frac{q_{k+1}-q_k}{\Delta t}\right)\frac{1}{\Delta t} \end{aligned} \quad (26)$$

Despite the intimidating appearance of (26), every term can be numerically evaluated using previous equations.

The integrator is initialized with the previous configuration, the current configuration, and a time-step. A root-finding algorithm finds the next configuration by solving (21) using (26). The solution is incremented so that the current configuration q_k becomes the previous q_{k-1} and the configuration q_{k+1} found by the root-finder becomes the current configuration q_k. The process is then repeated until the desired simulation time is reached. Just as in the ODE case, higher-order and adaptive time-stepping may be implemented as well without any adaptation to the basic procedure given here.

VIII. Example: Marionette

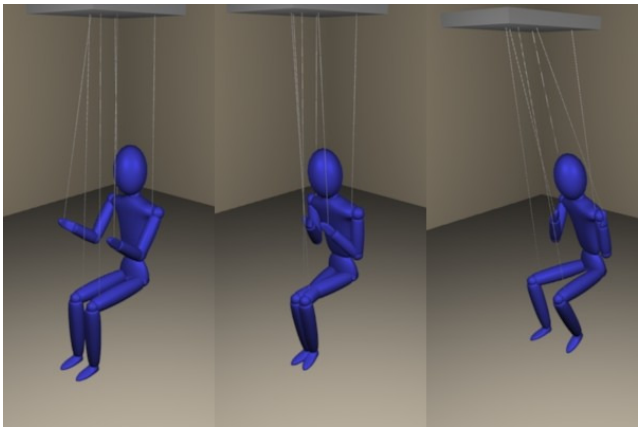


Fig. 2. The tree representation can handle the complex dynamics of a marionette.

Recently the authors have been engaged in a project for simulation and control of marionettes [6], partially motivating the current work. Marionettes have many degrees of freedom, typically 40 or more. Moreover, they have highly coupled dynamics, partially because of their limited actuation. Hence, they are excellent examples of complex mechanical devices. Figure 2 shows a marionette being simulated using a variational integrator based on the tree representation. The simulation includes dissipative forces and constraints. The constraints use kinematic configuration variables [6]. The mixed dynamic-kinematic modeling technique used for the strings are implemented without modifying either the tree representation or the constrained variational integrator, though we do not focus on this here.

The simulation takes place directly in the marionette's generalized coordinates. This is convenient for motion planning and optimal control where the desired trajectory or state is usually specified as joint angles rather than the positions and orientations of each body part.

IX. Conclusions and Future Works

The tree representation provides a strong foundation for simulating mechanical systems. It provides remarkably simple equations and obvious avenues for numerical optimization. The method of specifying the model is also less error-

prone because every component of the tree is a simple transformation. For many of the same reasons, we find that even if an error is made in the specification of the system, it is easier to find the error in this setting.

The presented method also uses a systematic process to derive higher order derivatives of the forward kinematic maps. This is particularly well suited to variational integration. Better integrators are derived using better quadrature rules for (19), but may lead to higher order derivatives in the resulting equations. Similarly, we could use a better root-finding algorithm that needs a higher order derivatives of (21) to converge faster. The key is that the need for higher-order derivatives does not pose any difficulty in this setting.

The recursive equation definitions are also advantageous from a numerical efficiency standpoint. Once a value has been calculated, it can be cached and reused for later calculations. For example, the frame configurations, $g_{s,k}$, are used by all of the tree representation equations. Once they have been calculated, their numerical values can be reused when evaluating derivatives later. This explicitly avoids duplicating calculations that are otherwise repeated in purely symbolic equations.

In practice, this leads to significant speed improvements. Fig. 3 shows the results of simulating N-link pendulums over a ten second period. The dotted line indicates the computational time when caching is not used, which is similar to the case of a symbolically derived variational integrator. The solid line indicates the computational time when caching is used. Even with 19 links, the simulation takes less than 20 seconds.

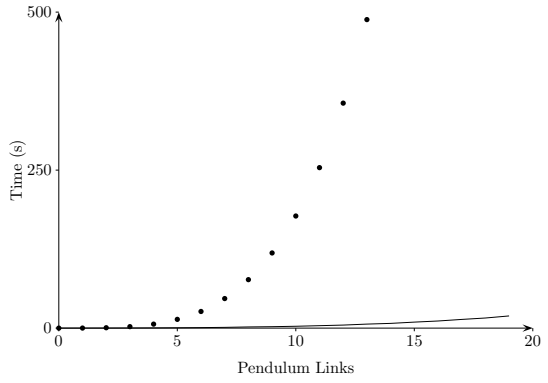


Fig. 3. Simulation times for pendulums of different links. The dotted line indicates the performance without caching.

In this paper, we have only discussed unconstrained mechanical systems of rigid bodies. These techniques can also be extended to include forcing and constraints (such as springs, damping, and non-slip constraints). The Euler-Lagrange equations will have additional terms, but the basic approach is the same. The tree representation requires no modification.

While we have made several assumptions to simplify our equations, they are not required by this representation. Many of the assumptions can be safely removed so long as the derivatives are re-derived from the base equations, (1) and (5). Most importantly, the equations could be derived to allow direct $\mathrm{SE}(3)$ parameterization of three dimensional rotations.

X. ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under CAREER award CMS-0546430. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

We would additionally like to acknowledge useful conversations with Prof. Magnus Egerstedt at the Georgia Institute of Technology.

REFERENCES

[1] D. Baraff. Non-penetrating rigid body simulation. In *State of the Art Reports*, 1993.

[2] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun. *Principles of Robot Motion*. The MIT Press, 2005.

[3] A.C. Fang and N.S. Pollard. Efficient synthesis of physically valid human motion. 2003.

[4] R. Featherstone. *Robot Dynamics Algorithms*. Kluwer Academic Publishers, 1987.

[5] R.C. Fetecau, J.E. Marsden, M. Ortiz, and M. West. Nonsmooth lagrangian mechanics and variational collision integrators. *SIAM Journal on Applied Dynamical Systems*, 2003.

[6] E.R. Johnson and T.D. Murphey. Dynamic modeling and motion planning for marionettes: Rigid bodies articulated by massless strings. In *International Conference on Robotics and Automation*, Rome, Italy, 2007.

[7] L. Kharevych, Weiwei, Y. Tong, E. Kanso, J. E. Marsden, P. Schroder, and M. Desbrun. Geometric, variational integrators for computer animation. *Eurographics/ACM SIGGRAPH Symposium on Computer Animation*, 2006.

[8] J. E. Marsen and M. West. Discrete mechanics and variational integrators. *Acta Numerica*, pages 357–514, 2001.

[9] R.M. Murray, Z. Li, and S.S. Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994.

[10] Y. Nakamura and K. Yamane. Dynamics computation of structure-varying kinematic chains and its application to human figures. *IEEE Transactions on Robotics and Automation*, 16(2), 2000.

[11] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C Second Edition*. Cambridge University Press, 1992.

[12] Matthew West. Variational integrators. *California Institute of Technology Thesis*, 2004.

[13] A. Witkin, M. Gleicher, and W. Welch. Interactive dynamics. In *Computer Graphics*, 1990.