Linearizations for Mechanical Systems in Generalized Coordinates

Elliot R. Johnson and Todd D. Murphey

***Abstract*— We describe an algorithm for calculating the linearization of the dynamics for arbitrary constrained mechanical systems in generalized coordinates without using symbolic equations. Linearizations of dynamics are useful tools for controllability and stability analysis and can be used to generate locally stabilizing controllers for linear and non-linear systems. However, the computational expense for finding linearizations of complex mechanical systems is often cited as a limiting factor that prevents their use. Recent work has introduced new methods of calculating the dynamics of arbitrary mechanical systems in generalized coordinates without deriving large, system-specific equations of motion. This paper extends that approach to calculate the linearizations of the dynamics without using the symbolic equations of motion. Using these ideas, it becomes practical to both simulate, analyze, and control more complex mechanical systems without sacrificing the benefits of generalized coordinates. Furthermore, this method addresses systems with closed kinematic chains, constraints, and external non-conservative forcing.**

The technique is applied to an example system with a closed kinematic chain and the resulting linearization agrees with results found by symbolically differentiating the full equations of motion.

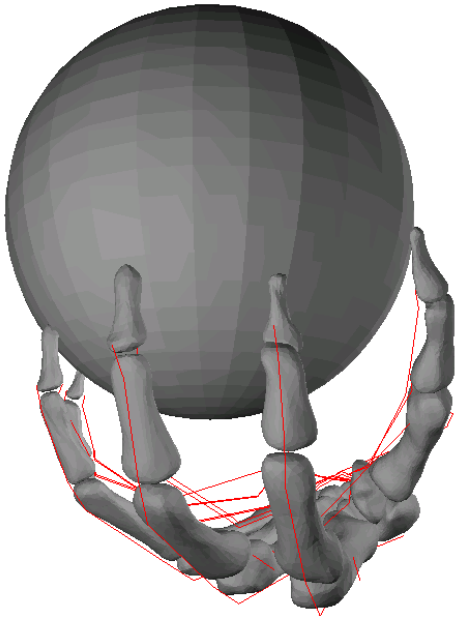
I. INTRODUCTION

The linearization of a dynamic system approximates the relationship in the neighborhood of a configuration between the dynamics and the system state/inputs as a linear system. Linearizations of non-linear systems can be used to analyze local stability and controllability properties [4]. Similarly, the linearization of a system at each point along a trajectory can be considered a time-varying linear system. Optimal control techniques like LQR theory [1] are then used with such linearizations to generate locally-stabilizing linear feedback control laws with very little manual intervention.

Linearizations are typically constructed directly from derivatives of the dynamics function. This is straightforward when the symbolic equations of motion are known and the derivatives can be found manually or using symbolic algebra software packages. As systems grow to have many degrees of freedom or closed kinematic chains, however, the symbolic equations of motion become prohibitively large, requiring large amounts of memory and taking long times to evaluate. In these cases, the linearizations are difficult to work with because the derivatives of the dynamics are typically even larger (due to new terms introduced by the chain and product rules) and it becomes impractical to use the linearization [14] in which case linearizations are either avoided or approximated by sampling methods.

E.R. Johnson elliot.r.johnson@u.northwestern.edu
T.D. Murphey t-murphey@u.northwestern.edu
Department of Mechanical Engineering, Northwestern University, 2145 Sheridan Road, Evanston, IL, USA 60208

This material is based upon work supported by the National Science Foundation under award IIS-0917837. 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.



1: The graph-based approach to calculating linearizations scales to complex mechanical systems like this dynamic model of a human hand holding an object. The linearization at this configuration shows that the system is locally controllable. (*The STL model was derived from http://www-static.cc.gatech.edu/projects/large_models/hand.html*)

Others have looked at automatic linearization, typically in the case of automatic differentiation [6], [3]. These software packages simply take algebraic expressions and apply chain rule and product rule to the expressions, so the topology of the mechanical system is never used to simplify the linearization process. In [9], the authors linearize systems based on their geometric properties. This work is similar to the present work, but does not explicitly take advantage of the mechanical topology.

Recent work by the authors [10] has introduced tools for simulating mechanical systems in generalized coordinates without explicitly deriving the equations of motion for the system. These avoid large equations and improve scalability. Those techniques use a formal graph-based description of a system that is used to numerically calculate all the quantities (i.e, the positions, velocities, and their derivatives for every coordinate frame in the system) needed to calculate the

system dynamics at a particular configuration. The key idea to this approach is to work with the general Euler-Lagrange equations of motion and evaluate the individual terms numerically rather than plugging in symbolic equations to find system-specific equations. A software implementation of this approach is available (See `trep` at `http://trep.sourceforge.net`).

This paper is the straightforward continuation of that idea that makes linearizations practical for large, constrained systems. We take derivatives of the general Euler-Lagrange equations to find an exact equation that can be evaluated numerically. This avoids dealing with large symbolic equations and leverages the performance benefits of the graph-based framework.

Together, these techniques have been used to simulate the dynamics and find trajectory linearizations for the model of the human hand shown in Fig. 1. This system has 20 degrees of freedom and 23 non-collocated inputs. It is under-actuated and the inputs often conflict.

The paper is organized as follows. Section II gives an overview of the graph-based approach including how the position and velocities of coordinate frames are calculated (Sec. II-A) as well as how the Euler-Lagrange equations are evaluated numerically (Sec. II-B). Section III extends the ideas to calculate linearizations. Linearizations of systems with constraints are discussed in Sec. III-B. Finally, the algorithm is applied to a forced, constrained double pendulum example in Sec. IV.

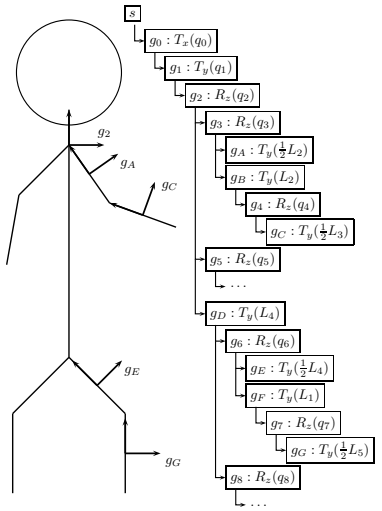
II. RECURSIVE DYNAMICS

Recursive approaches to calculating dynamics [13] [7] take advantage of special representations of mechanical systems that allow the values needed for simulation to be calculated quickly and avoid redundant calculations.

The work in this paper is based on the methods presented in [10]. Systems are represented as graphs where each node is a coordinate frame in the mechanical system and the nodes are connected by simple rigid body transformations (typically translations along and rotation about the X, Y, and Z axes though any rigid body screw motion can be used). Transformations are either constant or parameterized by real-valued variables. The set of all variables establishes the generalized coordinates for the system. Figure 2 is an example of a tree that represents a two-dimensional human form [10].

The graph description can include closed kinematic chains, but in practice the graph is converted to an acyclic directed graph (i.e, a tree) and augmented with holonomic constraints to close the kinematic chains.

The tree leads to fast and transparent algorithms to numerically[1] calculate the positions and velocities of each frame in the system. The subsequent dynamics calculations are equally direct and arise naturally by keeping the equations in general forms. We emphasize that this is only a brief overview of the simulation technique meant to emphasize the process that is being extended to find linearizations. For a detailed discussion, see [10].



2: A planar human is represented with a tree structure [10].

A. POSITION AND VELOCITY

The graph description establishes a hierarchy of coordinate frames that are related to their neighbors by rigid body transformations parameterized by real-valued configuration variables. These relationships allow the position and velocity of each frame to calculated by composition. For example, the spatial configuration (i.e, position) of the k-th frame, $g_{s,k}$ is calculated from its parent's position, $g_{s,\mathrm{par}(k)}$ and the local transformation g_k:

$$g_{s,k}(q) = g_{s,\mathrm{par}(k)} g_k.$$

Similarly, a frame's body velocity $\hat{v}^b_k(q, \dot{q})$ is calculated by appropriately transforming the parent's body velocity $\hat{v}^b_{\mathrm{par}\,k}(q, \dot{q})$ and adding the local body velocity:

$$\hat{v}^b_k(q, \dot{q}) = g_k^{-1} \hat{v}^b_{\mathrm{par}(k)} g_k + g_k^{-1} \dot{g}_k.$$

The above two equations are recursive and traverse up the graph towards the stationary world frame[2] which has known position and velocity (the identity and zero elements, respectively) and terminates the recursion.

Derivatives of position and velocity are obtained directly from the above. For example, the derivative of the k-th frame's position with respect to configuration variable q_i is:

$$\frac{\partial g_{s,k}}{\partial q_i}(q) = \frac{\partial g_{s,\mathrm{par}(k)}}{\partial q_i} g_k + g_{s,\mathrm{par}(k)} \frac{\partial g_k}{\partial q_i}.$$

This equation is evaluated using recursion for the first term and the known derivative of the local transformation for the second. Since only a simple set of local transformations are allowed, all of these derivatives are determined before simulating the system and have simple forms. This process

[1] We emphasize that while these values and the derivatives are found numerically, they are exact derivatives, not numeric approximations.

[2] We have omitted the details in this discussion, but every frame in a graph description descends from the stationary world frame.

is repeated to get the higher order derivatives that are needed for simulation and calculating linearizations.

B. CALCULATING DYNAMICS

The dynamics in generalized coordinates are calculated using the Euler-Lagrange equations. The Lagrangian is written in terms of values that are provided by the graph description (e.g, the position and velocity equations in Sec. II-A). This keeps the equations general so they work with any system described by a graph and avoids large symbolic equations by working with the numerically calculated values.

The Lagrangian for the system is the sum of all of the kinetic energies in the system minus each potential energy:

$$\begin{aligned} L(q,\dot{q}) &= KE(q,\dot{q}) - PE(q) \\ &= \sum_i \tfrac{1}{2} v_i^{bT} M_i v_i^b + \sum_i V_i(q) \end{aligned} \tag{1}$$

where M_i is the inertia matrix for each mass in the system. Derivatives of the Lagrangian are found directly. For example, the first derivatives are:

$$\frac{\partial L}{\partial q}(q,\dot{q}) = \sum_i v_i^{bT} M_i \frac{\partial v_i^b}{\partial q} + \sum_i \frac{\partial V_i}{\partial q} \tag{2a}$$

$$\frac{\partial L}{\partial \dot{q}}(q,\dot{q}) = \sum_i v_i^{bT} M_i \frac{\partial v_i^b}{\partial \dot{q}} \tag{2b}$$

This is continued to find any higher order derivative required. The important point is that no other equations are substituted in for v_i^{bT} and derivatives–these are always provided numerically from the graph description equations.

Next, we must put the Euler-Lagrange equations in a form that is also calculated from the Lagrangian derivatives. This is found by expanding the time derivative in the Euler-Lagrange equation and solving for $\ddot{q}$:

$$\begin{aligned} \tfrac{\partial}{\partial t}\tfrac{\partial L}{\partial \dot{q}}(q,\dot{q}) - \tfrac{\partial L}{\partial q}(q,\dot{q}) &= f(q,\dot{q},u,t) \\ \tfrac{\partial^2 L}{\partial \dot{q}\partial \dot{q}}\ddot{q} + \tfrac{\partial^2 L}{\partial q \partial \dot{q}}\dot{q} - \tfrac{\partial L}{\partial \dot{q}} &= f \\ \ddot{q} = \left[\tfrac{\partial^2 L}{\partial \dot{q}\partial \dot{q}}\right]^{-1} &\left(f + \tfrac{\partial L}{\partial \dot{q}} - \tfrac{\partial^2 L}{\partial q \partial \dot{q}}\dot{q}\right) \end{aligned} \tag{3}$$

where $f(q,\dot{q},u,t)$ represents the external, non-conservative forces acting on the system and u is a vector of the system inputs. Equation (3) is evaluated in the presented form using the numeric values that are calculated for each term (i.e, Eq. (2)). Note that the computational complexity of computing $\frac{\partial^2 L}{\partial \dot{q}\partial \dot{q}}$, $\frac{\partial L}{\partial \dot{q}}$, $\frac{\partial^2 L}{\partial q \partial \dot{q}}$ is $O(n)$ where n is the number of rigid bodies. We continue this approach to calculate linearizations.

III. LINEARIZATIONS

Linearizations approximate the behavior of a non-linear system in the neighborhood of a state as a linear system. Linear analysis tools are used with linearizations to determine properties of the non-linear system around that configuration. For example, if the linearization is controllable, the non-linear system will also be locally controllable at that configuration.

Linearizations also arise simply by virtue of being derivatives. First and second order derivatives of dynamics frequently appear in optimization problems when the derivative of a cost function is needed to implement an iterative descent method [11].

Linearizations for models based on generalized coordinates are often prohibitively expensive to find and evaluate when working with symbolic equations of motion. In this section, we demonstrate that recursive methods for simulating dynamics easily extend to calculate linearizations without using the full symbolic equations of motion.

A. CALCULATING LINEARIZATIONS

To linearize the system, we must convert the second-order Euler-Lagrange dynamics into a first order system by letting $x = \begin{bmatrix} q & \dot{q} \end{bmatrix}$ which gives $\dot{x} = \begin{bmatrix} \dot{q} & \ddot{q} \end{bmatrix}$. The linearization around x_0,u_0 is then

$$\begin{aligned} \delta\dot{x} &= \left[\begin{array}{cc} \frac{\partial \dot{q}}{\partial q} & \frac{\partial \dot{q}}{\partial \dot{q}} \\ \frac{\partial \ddot{q}}{\partial q} & \frac{\partial \ddot{q}}{\partial \dot{q}} \end{array}\right]_{\substack{x=x_0 \\ u=u_0}} \delta x + \left[\begin{array}{c} 0 \\ \frac{\partial \ddot{q}}{\partial u} \end{array}\right]_{\substack{x=x_0 \\ u=u_0}} \delta u \\ &= \left[\begin{array}{cc} 0 & I \\ \frac{\partial \ddot{q}}{\partial q} & \frac{\partial \ddot{q}}{\partial \dot{q}} \end{array}\right]_{\substack{x=x_0 \\ u=u_0}} \delta x + \left[\begin{array}{c} 0 \\ \frac{\partial \ddot{q}}{\partial u} \end{array}\right]_{\substack{x=x_0 \\ u=u_0}} \delta u \end{aligned}$$

The terms $\frac{\partial \ddot{q}}{\partial q}$ and $\frac{\partial \ddot{q}}{\partial \dot{q}}$ are found by differentiating (3). For $\frac{\partial \ddot{q}}{\partial q}$, we find

$$\begin{aligned} \tfrac{\partial^3 L}{\partial q \partial \dot{q} \partial \dot{q}}\ddot{q} + \tfrac{\partial^2 L}{\partial \dot{q}\partial \dot{q}}\tfrac{\partial \ddot{q}}{\partial q} + \tfrac{\partial^3 L}{\partial q \partial \dot{q} \partial \dot{q}}\dot{q} - \tfrac{\partial L}{\partial q} &= \tfrac{\partial f}{\partial \dot{q}} \\ \tfrac{\partial \ddot{q}}{\partial q} = \left[\tfrac{\partial^2 L}{\partial \dot{q}\partial \dot{q}}\right]^{-1} &\left(\tfrac{\partial f}{\partial q} + \tfrac{\partial L}{\partial q} - \tfrac{\partial^3 L}{\partial q \partial \dot{q} \partial \dot{q}}\ddot{q} - \tfrac{\partial^3 L}{\partial q \partial q \partial \dot{q}}\dot{q}\right) \end{aligned}$$

For $\frac{\partial \ddot{q}}{\partial \dot{q}}$, we find

$$\begin{aligned} \tfrac{\partial^3 L}{\partial \dot{q} \partial \dot{q} \partial \dot{q}}\ddot{q} + \tfrac{\partial^2 L}{\partial \dot{q}\partial \dot{q}}\tfrac{\partial \ddot{q}}{\partial \dot{q}} + \tfrac{\partial^3 L}{\partial \dot{q} \partial q \partial \dot{q}}\dot{q} + \tfrac{\partial^2 L}{\partial q \partial \dot{q}}\tfrac{\partial \dot{q}}{\partial \dot{q}} - \tfrac{\partial L}{\partial \dot{q}} &= \tfrac{\partial f}{\partial \dot{q}} \\ \tfrac{\partial \ddot{q}}{\partial \dot{q}} = \left[\tfrac{\partial^2 L}{\partial \dot{q}\partial \dot{q}}\right]^{-1} &\left(\tfrac{\partial f}{\partial \dot{q}} + \tfrac{\partial L}{\partial \dot{q}} - \tfrac{\partial^3 L}{\partial \dot{q} \partial q \partial \dot{q}}\dot{q} - \tfrac{\partial^2 L}{\partial q \partial \dot{q}}\right) \end{aligned}$$

where we have taken advantage of the fact that $\frac{\partial^3 L}{\partial \dot{q}\partial \dot{q}\partial \dot{q}} = 0$ for any Lagrangian of the form (1).

Finally, we find the derivative with respect to the external inputs u:

$$\begin{aligned} \tfrac{\partial^2 L}{\partial \dot{q}\partial \dot{q}}\tfrac{\partial \ddot{q}}{\partial u} &= \tfrac{\partial f}{\partial u} \\ \tfrac{\partial \ddot{q}}{\partial \dot{q}} &= \left[\tfrac{\partial^2 L}{\partial \dot{q}\partial \dot{q}}\right]^{-1} \tfrac{\partial f}{\partial u} \end{aligned}$$

As in calculating the dynamics, the terms of the above equations are evaluated numerically and substituted to calculate $\frac{\partial \ddot{q}}{\partial q}$ and $\frac{\partial \ddot{q}}{\partial \dot{q}}$. No new structure is needed; an implementation only needs to provide functions to calculate the higher derivatives of the Lagrangian (and the higher derivatives that they, in turn, require).

B. CONSTRAINTS

The above discussion includes external, non-conservative forcing. We now demonstrate how linearizations for systems with constraints are generated. While constraints are important for most simulation applications, they are particularly important for the graph-based approach because holonomic constraints allow closed-kinematic chains to be modeled.

In continuous Euler-Lagrange mechanics, both holonomic and non-holonomic constraints[12] are defined by a function $A(q)$ such that

$$A(q)\dot{q} = 0 \tag{4}$$

is always satisfied.[3] For the remainder of this description, we will drop the dependence of $A(q)$ on q. In the graph-based approach to modeling, these functions are typically written in terms of generic frames in a system and leverage the graph calculations to generalize to arbitrary mechanical systems.

To accommodate constraints, the Euler-Lagrange equation is modified to include a forcing term that enforces the constraints:

$$\frac{\partial^2 L}{\partial \dot{q} \partial \dot{q}}\ddot{q} + \frac{\partial^2 L}{\partial q \partial \dot{q}}\dot{q} - \frac{\partial L}{\partial \dot{q}} = f(q, \dot{q}, u, t) + A^T\lambda$$

where λ is an unknown vector representing the magnitude of each constraint force. The above is differentiated as before to calculate $\ddot{q}$ and its derivatives:

$$\ddot{q} = \left[\frac{\partial^2 L}{\partial \dot{q} \partial \dot{q}}\right]^{-1}\left(f + A^T\lambda + \frac{\partial L}{\partial q} - \frac{\partial^2 L}{\partial q \partial \dot{q}}\dot{q}\right) \tag{5}$$

$$\frac{\partial \ddot{q}}{\partial q} = \left[\frac{\partial^2 L}{\partial \dot{q} \partial \dot{q}}\right]^{-1}\left(\frac{\partial f}{\partial q} + \frac{\partial A}{\partial q}^T\lambda + A^T\frac{\partial \lambda}{\partial q} + \frac{\partial L}{\partial \dot{q}} - \frac{\partial^3 L}{\partial q \partial \dot{q} \partial \dot{q}}\ddot{q} - \frac{\partial^3 L}{\partial q \partial q \partial \dot{q}}\dot{q}\right)$$

$$\frac{\partial \ddot{q}}{\partial \dot{q}} = \left[\frac{\partial^2 L}{\partial \dot{q} \partial \dot{q}}\right]^{-1}\left(\frac{\partial f}{\partial \dot{q}} + A^T\frac{\partial \lambda}{\partial \dot{q}} + \frac{\partial L}{\partial \dot{q}} - \frac{\partial^3 L}{\partial \dot{q} \partial q \partial \dot{q}}\dot{q} - \frac{\partial^2 L}{\partial q \partial \dot{q}}\right)$$

$$\frac{\partial \ddot{q}}{\partial u} = \left[\frac{\partial^2 L}{\partial \dot{q} \partial \dot{q}}\right]^{-1}\left(\frac{\partial f}{\partial u} + A^T\frac{\partial \lambda}{\partial u}\right)$$

To evaluate the above, we must be able to calculate λ and its derivative. λ is found by differentiating (4) and substituting (5) for $\ddot{q}$ [5]:

$$\lambda = \left(A\left[\frac{\partial^2 L}{\partial \dot{q} \partial \dot{q}}\right]^{-1}A^T\right)^{-1}\left(A\left[\frac{\partial^2 L}{\partial \dot{q} \partial \dot{q}}\right]^{-1}\left(\frac{\partial^2 L}{\partial q \partial \dot{q}}\dot{q} - \frac{\partial L}{\partial \dot{q}} - f\right) - \dot{A}\dot{q}\right)$$

We can now take the derivative as before, making use of the following identity for matrix inverses:

$$\frac{\partial}{\partial q}\left[M^{-1}\right] = -M^{-1}\frac{\partial M}{\partial q}M^{-1}$$

The derivative with respect to q is:

$$\begin{aligned}\frac{\partial \lambda}{\partial q} =& \left(A\left[\frac{\partial^2 L}{\partial \dot{q} \partial \dot{q}}\right]^{-1}A^T\right)^{-1}\left(\left(\frac{\partial A}{\partial q}\left[\frac{\partial^2 L}{\partial \dot{q} \partial \dot{q}}\right]^{-1}\right.\right. \\ &- A\left[\frac{\partial^2 L}{\partial \dot{q} \partial \dot{q}}\right]^{-1}\frac{\partial^3 L}{\partial \dot{q} \partial \dot{q} \partial q}\left[\frac{\partial^2 L}{\partial \dot{q} \partial \dot{q}}\right]^{-1}\Big)\left(\frac{\partial^2 L}{\partial q \partial \dot{q}}\dot{q} - \frac{\partial L}{\partial q} - f\right) \\ &\left.+ A\left[\frac{\partial^2 L}{\partial \dot{q} \partial \dot{q}}\right]^{-1}\left(\frac{\partial^3 L}{\partial q \partial q \partial \dot{q}}\dot{q} - \frac{\partial^2 L}{\partial q \partial \dot{q}} - \frac{\partial f}{\partial q}\right) - \frac{\partial \dot{A}}{\partial q}\dot{q}\right)\end{aligned}$$

For $\dot{q}$, we find:

$$\frac{\partial \lambda}{\partial \dot{q}} = \left(A\left[\frac{\partial^2 L}{\partial \dot{q} \partial \dot{q}}\right]^{-1}A^T\right)^{-1}\left(A\left[\frac{\partial^2 L}{\partial \dot{q} \partial \dot{q}}\right]^{-1}\left(\frac{\partial^3 L}{\partial \dot{q} \partial q \partial \dot{q}}\dot{q} + \frac{\partial^2 L}{\partial q \partial \dot{q}} - \frac{\partial^2 L}{\partial \dot{q} \partial q} - \frac{\partial f}{\partial \dot{q}}\right) - \dot{A}\right)$$

[3]Holonomic constraints of the form $h(q) = 0$ are differentiated with respect to time to use this representation.

Finally, we differentiate with respect to u:

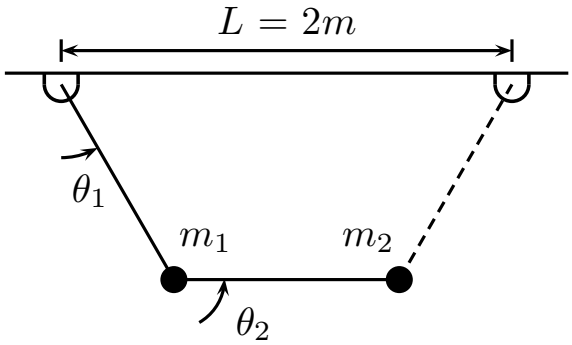
$$\frac{\partial \lambda}{\partial u} = -\left(A\left[\frac{\partial^2 L}{\partial \dot{q} \partial \dot{q}}\right]^{-1}A^T\right)^{-1}A\left[\frac{\partial^2 L}{\partial \dot{q} \partial \dot{q}}\right]^{-1}\frac{\partial f}{\partial u}$$

We now have everything needed to calculate linearizations for arbitrary mechanical systems in generalized coordinates. There was no change in the approach used to evaluate the derivative, only more terms to be evaluated. The equations still rely on the graph description and are evaluated numerically so that no large system-specific symbolic equations are generated.

IV. EXAMPLE

To demonstrate this algorithm, we now consider the example system in Fig. 3 which models a double pendulum with a constrained end point. A torque is applied at the base joint to manipulate the system.

For this example, we present the parameters of the system and numerically find the linearization of of the system in the shown configuration using the algorithm described in this paper. The results are consistent with the linearization found by symbolically differentiating the full symbolic equations of motion and



3: The example system is a double pendulum with a constraint on the bottom mass. A torque is applied to the base of the pendulum at θ_1.

Each link is $1m$ long and both disks have a mass of $1kg$. The system is subjected to gravity of $9.8\frac{m}{s^2}$. The configuration vector of the system is $q = [\theta_1, \theta_2]$.

The system is described by three local homogeneous transformations:

$$\begin{aligned} g_1 &= R(\theta_1) & g_A &= T_y(-1) \\ g_2 &= R(\theta_2) & g_B &= T_y(-1) \\ & & g_C &= T_x(2) \end{aligned}$$

where

$$R(\theta) = \begin{bmatrix}\cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1\end{bmatrix}$$

$$T_X(x) = \begin{bmatrix} 1 & 0 & x \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad T_y(y) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & y \\ 0 & 0 & 1 \end{bmatrix}$$

A holonomic constraint fixes the distance between the end of the pendulum and the right anchor to be $1m$.

$$h(q) = ||p_B - p_C||^2 - 1 = 0$$

where p_B and p_C are the x, y coordinates of the second mass and anchor, respectively. The position and any of their derivatives are provided by the tree description framework. The holonomic constraint is converted to non-holonomic constraints by differentiating with respect to time.

$$\begin{aligned} \frac{\partial h}{\partial t} &= \frac{\partial h}{\partial q}\dot{q} \\ 0 &= \begin{bmatrix} \frac{\partial h}{\partial \theta_1} & \frac{\partial h}{\partial \theta_2} \\ \frac{\partial h}{\partial \theta_1} & \frac{\partial h}{\partial \theta_2} \end{bmatrix} \dot{q} \\ 0 &= A(q)\dot{q} \end{aligned}$$

The configuration and configuration velocity must be chosen so that both the original holonomic constraint $h(q) = 0$ and the derived non-holonomic constraint $A(q)\dot{q} = 0$ are satisfied. The configuration was found by inspection from Fig. 3 and is $q_0 = [0.5246, 1.0472]$. An appropriate configuration velocity was found from the null space of $A(q_0)$ and is $[1.000, 2.00116]$.

A control torque u is applied to the first joint in the system through the generalize force

$$f = \begin{bmatrix} u(t) \\ 0 \end{bmatrix}$$

.

The applied torque for the linearization calculation is $u_0 = 2Nm$.

Once the system, constraints, and forces are fully specified, the presented algorithm is used to numerically compute the linearization. For the specified configuration and input torque, the linearization was found:

$$\delta\dot{x} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -1.133 & 7.690 & -0.580 & -1.447 \\ 5.620 & -10.698 & -1.155 & 0.577 \end{bmatrix} \delta x + \begin{bmatrix} 0 \\ 0 \\ 0.500 \\ -1.000 \end{bmatrix} \delta u$$

The results were verified by symbolically differentiating the full equations of motion in Mathematica. The software implementation of our algorithm, this example, and the Mathematica notebook used to verify the solution can be found at `http://trep.sourceforge.net/examples/2010acc`.

V. CONCLUSIONS

The same techniques that provide fast and scalable simulations of arbitrary mechanical systems in generalized coordinates extend to calculate exact linearizations. This enables an important area of control theory to be used to analyze systems that were previously computationally impractical. Since this method is an extension of simulation tools, the linearizations are calculated from the same system specifications so there is no need to redefine the system in another setting. This technique also works for constrained systems so that linearizations for systems with closed kinematic chains are possible.

This theory is straightforward to extend to the second derivatives of a system's dynamics. Second derivatives arise, for example, in optimization problems [8][2] that use Newton's method to achieve quadratic convergence rates. The second derivative is found using the same procedure presented.

REFERENCES

[1] B.D.O. Anderson and J.B. Moore. *Linear Optimal Control*. Prentice Hall, Inc, 1971.

[2] T.M. Caldwell and T.D. Murphey. Second-order optimal estimation of slip state for a simple slip-steered vehicle. Bangalore, India, 2009.

[3] S. Campbell. Linearization of daes along trajectories. *Zeitschrift für Angewandte Mathematik und Physik (ZAMP)*, 46(1):70–84, 1995.

[4] C.T. Chen. *Linear System Theory and Design*. Saunders College Publishing, 1984.

[5] 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.

[6] G. Corliss, C. Faure, A. Griewank, L. Hascoet, and U. Naumann, editors. *Automatic Differentiation of Algorithms*. Springer, 2002.

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

[8] J. Hauser. A projection operator approach to optimization of trajectory functionals. Barcelona, Spain, 2002.

[9] Anil N. Hirani. *Linearization Methods For Variational Integrators and Euler-Lagrange Equations*. PhD thesis, California Institute of Technology, 2000.

[10] E. R. Johnson and T. D. Murphey. Scalable variational integrators for constrained mechanical systems in generalized coordinates. *IEEE Transactions on Robotics*, 2010.

[11] C.T. Kelley. *Iterative Methods for Optimization*. Society for Industrial Mathematics, 1987.

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

[13] 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.

[14] E. Todorov and Y. Tassa. Iterative local dynamic programming. *IEEE Adaptive Dynamic Programming and Reinforcement Learning*, 2009.