To appear in the ACM SIGGRAPH 2003 conference proceedings

Efficient Synthesis of Physically Valid Human Motion

Anthony C. Fang Nancy S. Pollard

Department of Computer Science
Brown University*

Abstract

Optimization is a promising way to generate new animations from a minimal amount of input data. Physically based optimization techniques, however, are difficult to scale to complex animated characters, in part because evaluating and differentiating physical quantities becomes prohibitively slow. Traditional approaches often require optimizing or constraining parameters involving joint torques; obtaining first derivatives for these parameters is generally an $O(D^2)$ process, where D is the number of degrees of freedom of the character. In this paper, we describe a set of objective functions and constraints that lead to linear time analytical first derivatives. The surprising finding is that this set includes constraints on physical validity, such as ground contact constraints. Considering only constraints and objective functions that lead to linear time first derivatives results in fast per-iteration computation times and an optimization problem that appears to scale well to more complex characters. We show that qualities such as squash-and-stretch that are expected from physically based optimization result from our approach. Our animation system is particularly useful for synthesizing highly dynamic motions, and we show examples of swinging and leaping motions for characters having from 7 to 22 degrees of freedom.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation G.1.6 [Numerical Analysis]: Optimization—Constrained Optimization

Keywords: animation, physically based animation

1 Introduction

One appealing vision in animation is that the animator should be able to create and edit motion by defining and adjusting a small number of keyframes and constraints—and that the resulting motion should remain optimal in some way. An optimization approach to animation has proven useful for editing human motion capture data, refining a "sketched" version of an animation, and for creating entirely new motions for simple characters or short segments.

Several challenges remain, however, to achieving fast, flexible, and realistic optimization of human motion. One challenge is incorporating physics into an interactive animation system. Despite impressive results obtained from physically based optimization, constraints and objective functions that require computing physical quantities such as momentum, force, and torque are typically viewed as slow, cumbersome, and difficult to control, especially for complex humanlike characters. As a result, physical validity is often sacrificed for performance. Physical validity is important, however, in situations such as those shown in Figure 1. Kinematic optimization alone is unlikely to capture the coordination of different parts of the body that is required to perform this task, such as the preparatory back swing, the tuck, or the motion of the legs to drive the character upward that is shown in the bottom row of the figure.

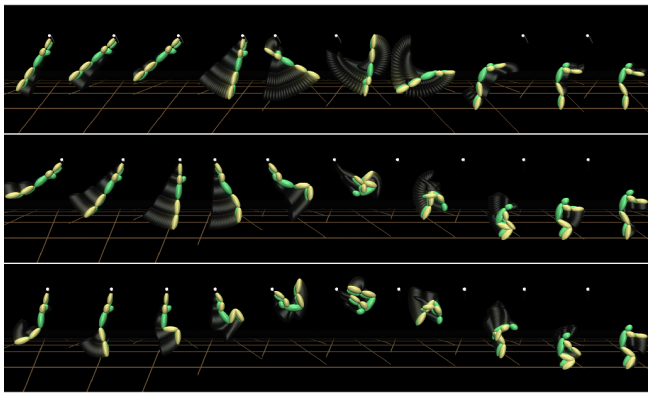

Figure 1: A single-flip dismount from a high-bar. (Top) Initial guess. (Middle) Flight duration is 0.6 seconds; Flip posture is tight and maximum height is below high-bar. (Bottom) Flight duration increased to 0.8 seconds. Flip posture is relaxed, and maximum height exceeds high-bar.

This paper presents an approach to physically based optimization that is efficient and appears to scale well to more complex characters. We use a standard problem formulation—iteratively adjust character motion to meet animator constraints and minimize an objective function. Our approach is based on restricting the definition of this optimization problem to constraints and objective functions that can be differentiated in time linear in the degrees of freedom of the character. The motivation for this approach is that solution techniques for nonlinear constrained optimization problems (e.g. SQP) typically require either analytical or numerical derivatives. Obtaining these derivatives is a computational bottleneck, and complex derivatives can lead to poor optimization performance and problems with local minima. Kinematic optimization [Gleicher 1997], which has been shown to be successful for complex characters, depends on constraints and objective functions for which first derivatives can be computed in linear time. *We have found that constraints on physics that can be derived from the aggregate force and torque applied to the character can also be differentiated in linear time.* This set includes most common constraints required for physically correct animation, such as conserving linear and angular momentum during flight, ensuring that ground contact forces can be explained by

*Providence, RI 02912, [acf|nsp]@cs.brown.edu

foot placement, constraining torque applied about an axis (e.g. the high bar in Figure 1), and limiting the coefficient of friction at any contact with the environment.

Linear time derivatives for physics constraints do not result from direct differentiation of the equations of motion in either the Newton-Euler or the Lagrangian formulation; in either case, symbolic differentiation would result in a quadratic time algorithm. In this paper, we describe how the Newton-Euler equations of motion can be rewritten to allow first derivatives of aggregate forces and torques to be computed in linear time. We note that it is not possible to compute derivatives for torques at all of the characters joints in linear time. Intuitively, quadratic time is required because motion at any joint affects torque at all joints. As a result, typical objective functions such as minimizing the sum of squared joint torques are excluded from our restricted problem setup. Our results suggest, however, that physics constraints and a kinematic measure of smooth motion such as minimizing the sum of squared joint accelerations are sufficient to capture dynamic effects such as squash-and-stretch and tucking for faster rotation, as shown in Figure 1.

While animator constraints such as key poses or an objective based on proximity to a reference motion can easily be incorporated into the system, *no motion capture data is used in our examples*, and user-supplied constraints are minimal (e.g., see Figure 7). The characteristics of the final motions fall out of the requirements of physical validity, a simple kinematic optimization function, and timing values selected for each phase of the motion.

2 Background

Constrained optimization techniques were introduced to the graphics community by Witkin and Kass [1988], who created a variety of animations involving a jumping Luxo lamp from simple descriptions including start pose, end pose, and a physically based objective function. Optimization approaches with physically based objective functions have proven difficult to extend to complex articulated characters, however, and much research has been focused on this problem. Cohen and his colleagues [Cohen 1992] [Liu et al. 1994] introduced techniques to give the user more control, including an ability to focus on windows in time, and employed a hierarchical wavelet description to allow incremental changes to affect the motion at different time scales. In his dissertation, Liu [1996] also describes how symbolic differentiation of the equations of motion can be made efficient (although still quadratic time) by cleverly aggregating terms. Grzeszczuk, Terzopoulos, and Hinton [1998] developed a neural network approximation of dynamics so that gradient search could be performed on this neural network, resulting in faster convergence to a solution.

The mix of animator control and physics present in Witkin and Kass [1988] has been expanded upon in interactive techniques developed to control physical simulations of rigid bodies [Popović et al. 2000], and a number of researchers have shown that the freefall portion of a dive can be efficiently optimized for a simplified character [Liu and Cohen 1994][Crawford 1998][Albro et al. 2000], as can motions such as weight lifting and pushups [Lo and Metaxas 1999]. Optimal control techniques, introduced to the graphics community by Brotman and Netravali [1988], have been used with success by Pandy and Anderson [2000] for simulating human lower body motions such as optimal height jumping and walking. Running times were far from interactive, but show that optimization techniques can produce realistic motion for systems of human-level complexity.

Preexisting motion data can simplify the optimization process. Full scale human motion can be optimized when closely spaced keyframes are available [Liu and Cohen 1995] or when only transitions between existing motion segments are required [Rose et al. 1996]. Popović and Witkin [1999] have shown that significant changes to motion capture data can be made by optimizing with a physically based objective function when the character is reduced to the degrees of freedom most important for the task. When physics does not dominate the motion, kinematic techniques can give the animator interactive control for motion editing (e.g., [Gleicher 1997] [Lee and Shin 1999] [Arikan and Forsyth 2002]).

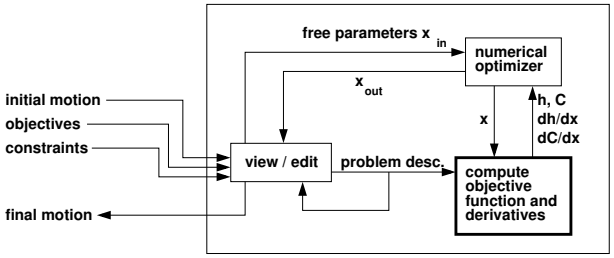

Figure 2: Optimizing motion synthesis or editing system. Parameter h is the objective function, and C are the constraint errors. This paper discusses efficient computation of the objective function, constraint errors, and their derivatives.

The idea of physically valid motion has appeared in both graphics and robotics. Dynamic filters have been developed for processing motion capture data for physical correctness [Yamane and Nakamura 2000] [Dasgupta and Nakamura 1999] [Pollard and Reitsma 2001]. Physics constraints have been used to plan biped walking motions, exploiting the idea that dynamic equilibrium can be maintained by ensuring that the zero moment point (ZMP)—the point on the ground at which ground reaction moments about horizontal axes are zero—lies within the support polygon of the feet [Vukobratović 1970] [Takanishi et al. 1985] [Nagasaka et al. 1999]. Similar ideas have also been developed in graphics by [Ko and Badler 1996], who bend the torso of a character to reduce torques at the desired ZMP, and [van de Panne 1997] who ensure that reasonable forces are available to accelerate the center of mass without creating angular acceleration. Liu and Popović [2002] show that some dynamic effects can be preserved by enforcing patterns of linear and angular momentum, which does not require computation of dynamic parameters such as contact forces and joint torques. We add to this body of work the insight that it is possible to incorporate constraints on physics as efficiently as constraints on kinematic parameters and an $O(D)$ algorithm for computing first derivatives of a broad range of physics constraints for improved performance in a optimization context.

3 Constrained Optimization

Constrained optimization has been shown to be a very powerful approach for obtaining appealing dynamic motions from a minimal amount of input information. The user adjusts the problem description in the form of keyframes, constraints, and objectives; an optimizer computes an optimal animation given this problem description; and the process repeats until the user obtains a final animation (Figure 2).

We state the optimization problem solved at each stage in the following form:

$$\min_x h(B(t)x) \text{ subject to } c(t_i) = 0,\ i = 1..m,\ t_i \in [t_s, t_f]$$

where h is the optimization function; $B(t)$ is a set of basis functions; x are the coefficients, the free parameters of the optimization; and $c(t_i)$ are the constraints. We use cubic B-splines as basis functions and follow the standard approach of enforcing constraints at a fixed

set of points in time (t_i). Enforcing physics constraints or minimizing a dynamic property such as sum squared joint torques requires an inverse dynamics computation at each time t_i.

Although the inverse dynamics computation is relatively expensive, many efficient algorithms exist, and the process is well known to require time linear in the number of degrees of freedom of the character. However, typical choices for the numerical optimizer in Figure 2 also require derivatives of the constraints and objective function. For example, the sequential quadratic programming algorithm used in [Witkin and Kass 1988] makes use of first derivatives of the constraints (the constraint Jacobian) and both first and second derivatives of the objective function (the Jacobian and the Hessian).

This paper describes how a broad range of physics constraints can be expressed based on aggregate forces and torques applied to the character, and how expressing physics constraints in this way allows us to compute the constraint Jacobian in linear time (Section 4). Objective functions are compared for efficiency in Section 5. We used an objective function that enforces smooth motion, with a linear time Jacobian computation and a constant Hessian. With this objective function and our linear time algorithm for computing the constraint Jacobian, we are able to show that physically based optimization can be performed for a 22 degree of freedom character at interactive speeds.

4 Efficient Physics Constraints

Constraints that enforce physical validity can be formulated as linear equality or inequality constraints on aggregate force. The aggregate force is a representation of all external forces and torques (excluding gravity) that would have to be applied to the character root to explain the character's motion. We classify the physics constraints for the motions in our examples into the categories of flight, bar contact, and ground contact.

Flight. One way of enforcing correct physics during flight is to ensure that the aggregate momentum of the body remains constant throughout the flight phase. Unfortunately, the constraint Jacobian that results from constraining momenta is denser than necessary as the control points that determine take-off affect all constraint equations governing the flight phase.

A more elegant solution is to restrict illegal forces during flight. During flight, no forces, with the exception of gravity, may be derived from the environment. In our system, gravity is simulated by adding an acceleration of $-G$ to the root, where G is the acceleration due to gravity (See Appendix B). When gravity is simulated in this manner, the aggregate forces computed by the dynamics equations are the sum of forces acting on the body in excess of the gravitational forces. Let the aggregate force be denoted by f_0. (In the spatial notation used here, f_0 contains both linear forces and torques.) The flight constraint is thus $\mathrm{f}_0 = 0$.

Bar contact. When the character is swinging on a high bar or monkey bars, the amount of torque that can be applied about the bar axis is constrained. Let aggregate force f_0 be represented as

$$\mathrm{f}_0 = \begin{bmatrix} \underline{\mathrm{f}}_0^a \\ \underline{\mathrm{f}}_0^b \end{bmatrix} \tag{1}$$

where $\underline{\mathrm{f}}_0^a$ is linear force and $\underline{\mathrm{f}}_0^b$ is torque about the world origin. Aggregate force is translated to a constraint point $\underline{\mathrm{c}}$ as follows:

$$\mathrm{f}_c = \begin{bmatrix} \underline{\mathrm{f}}_c^a \\ \underline{\mathrm{f}}_c^b \end{bmatrix} = \begin{bmatrix} \underline{\mathrm{f}}_0^a \\ \underline{\mathrm{f}}_0^b - \underline{\mathrm{c}}^0 \times \underline{\mathrm{f}}_0^a \end{bmatrix} \tag{2}$$

where $\underline{\mathrm{c}}^0$ is the world vector from the base of the articulation to $\underline{\mathrm{c}}$.

The bar contact constraint can then be expressed as

$$-\tau_{max} < \underline{\mathrm{s}}_{bar} \cdot \underline{\mathrm{f}}_c^b < \tau_{max} \tag{3}$$

where τ_{max} is the scalar torque limit, $\underline{\mathrm{s}}_{bar}$ is the bar axis, and $\underline{\mathrm{s}}_{bar} \cdot \underline{\mathrm{f}}_c^b$ is a projection operation that results in torque about the bar axis.

Ground contact. During ground contact, the feet can only push, not pull on the ground, contact forces should not require an unreasonable amount of friction, and the center of pressure must fall within the support polygon of the feet. These effects can be modeled with equations that constrain the linear and angular forces separately.

We constrain the linear force using Coulomb's contact model. Coulomb's model dictates that the linear reaction force must fall within a friction cone oriented along the contact normal with angular half-width $\tan^{-1}\mu$, where μ is the coefficient of friction. The inequality constraint for the linear aggregate force is therefore

$$\cos^{-1}\left(\frac{\underline{\mathrm{N}} \cdot \underline{\mathrm{f}}_0^a}{|\underline{\mathrm{f}}_0^a|}\right) < \tan^{-1}\mu \tag{4}$$

where $\underline{\mathrm{N}}$ is the unit contact normal. Equivalently, the constraint without the inverse trigonometric functions is:

$$\frac{\underline{\mathrm{N}} \cdot \underline{\mathrm{f}}_0^a}{|\underline{\mathrm{f}}_0^a|} > \frac{1}{\sqrt{(\mu^2+1)}} \tag{5}$$

The magnitude of the normal force can be constrained as follows:

$$0 \le \underline{\mathrm{f}}_0^a \cdot \underline{\mathrm{N}} \le K_{force} \tag{6}$$

Contact torques are constrained by geometrically confining the center of pressure to the support area. In the simplified case of a rectangular support area (or a linear support area in a two-dimensional set up), the aggregate torques may be constrained directly as follows: Translate f_0 to the center $\underline{\mathrm{c}}$ of the support as in Equation 2. Let $\underline{\mathrm{T}}_x$ and $\underline{\mathrm{T}}_y$ be orthogonal vectors spanning the rectangular support, and let δx and δy be the distances from $\underline{\mathrm{c}}$ to the edge of the support along along $\underline{\mathrm{T}}_x$ and $\underline{\mathrm{T}}_y$ respectively. The torques about $\underline{\mathrm{T}}_x$ and $\underline{\mathrm{T}}_y$ may be constrained as:

$$-\delta y\,(\underline{\mathrm{N}} \cdot \underline{\mathrm{f}}_0^a) < \underline{\mathrm{f}}_c^b \cdot \underline{\mathrm{T}}_x < \delta y\,(\underline{\mathrm{N}} \cdot \underline{\mathrm{f}}_0^a) \tag{7}$$

$$-\delta x\,(\underline{\mathrm{N}} \cdot \underline{\mathrm{f}}_0^a) < \underline{\mathrm{f}}_c^b \cdot \underline{\mathrm{T}}_y < \delta x\,(\underline{\mathrm{N}} \cdot \underline{\mathrm{f}}_0^a) \tag{8}$$

The magnitude of the twist force is constrained as follows:

$$-K_{twist}\,(\underline{\mathrm{N}} \cdot \underline{\mathrm{f}}_0^a) < \underline{\mathrm{f}}_c^b \cdot \underline{\mathrm{N}} < K_{twist}\,(\underline{\mathrm{N}} \cdot \underline{\mathrm{f}}_0^a) \tag{9}$$

All physics constraints. Once all physics constraints have been expressed as constraints on aggregate force, computing derivatives on the physics constraints becomes a problem of differentiating aggregate force with respect to the free parameters of the problem. At any time t, character position q, velocity $\dot{q}$, and acceleration $\ddot{q}$ are known. The derivative of interest can be expressed in terms of q, $\dot{q}$, and $\ddot{q}$ using the chain rule:

$$\frac{\partial \mathrm{f}_0}{\partial x} = \boxed{\frac{\partial \mathrm{f}_0}{\partial q}}\frac{\partial q}{\partial x} + \frac{\partial \mathrm{f}_0}{\partial \dot{q}}\frac{\partial \dot{q}}{\partial x} + \frac{\partial \mathrm{f}_0}{\partial \ddot{q}}\frac{\partial \ddot{q}}{\partial x} \tag{10}$$

where terms $\partial q/\partial x$, $\partial \dot{q}/\partial x$, and $\partial \ddot{q}/\partial x$ are available trivially from the equations expressing joint degrees of freedom q as a function of B-spline coefficients x. The term $\partial \mathrm{f}_0/\partial q$, which we will refer to as the *force Jacobian*, is the most difficult term in this expression. *The main point of the paragraphs below is to show how the*

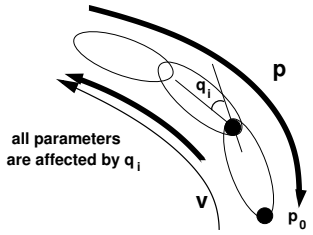

Figure 3: The effect of parameter q_i is propagated up the tree with velocities v and back down the tree with momentum terms p. Computing $\partial p_0/\partial q_i$ requires $O(D)$ time and results in an $O(D^2)$ algorithm for computing the momentum Jacobian.

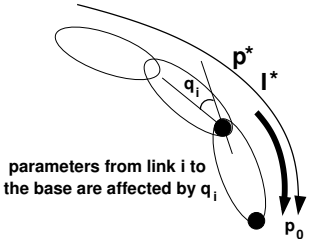

Figure 4: The effect of rewriting the recursion is to limit the effect of q_i to parameters collected at joints between i and 0. Terms required for the momentum Jacobian are accumulated in a single pass from leaf to base, and the momentum Jacobian can be computed in linear time.

force Jacobian can be computed efficiently.[1] We show that straightforward analytical computation of the force Jacobian would require time quadratic in the number of degrees of freedom of the character. However, if joint torques are not required, then this value and first derivatives for constraints based on this value can be computed in linear time. To our knowledge, our paper is the first to present a linear time algorithm for computing the force Jacobian for an articulated character or robot.

4.1 Notation

Our argument and implementation is constructed around a Newton-Euler formulation of inverse dynamics. We use spatial notation as in Featherstone [1987] for conciseness. Spatial notation involves 6-dimensional vectors, 6x6 coordinate transformations, and 6x6 inertia tensors. It combines linear and angular quantities such as force and torque or linear and angular velocity into single vectors, as shown in Equations 1 through 3. An overview of spatial notation is given in Appendix A, and terms are summarized here for reference.

D	Degrees of freedom of the articulated figure
q_i	Scalar position of link i, from motion curves (DOF i)
$\dot{q}_i$	Scalar velocity of link i, from motion curves
$\ddot{q}_i$	Scalar acceleration of link i, from motion curves
X_i^j	Spatial transform from frame i to frame j
X_i^0	Spatial transform from frame i to world frame
s_i'	Joint axis of link i (frame i)
v_i'	Local velocity of link i (frame i)
v_i	Global velocity of link i (frame i)
a_i'	Local acceleration of link i (frame i)
a_i	Global acceleration of link i (frame i)
I_i'	Spatial inertia of link i (frame i)
p_0	Aggregate momentum of articulated figure (world frame)
f_0	Aggregate force of articulated figure (world frame)

4.2 Linear Time Momentum Jacobian

Efficiently computing $\partial f_0/\partial q$, the force Jacobian, requires efficiently computing $\partial p_0/\partial q$, the momentum Jacobian, because aggregate force f_0 is the time derivative of aggregate momentum p_0. We begin with a discussion of the momentum equations and present an argument that the momentum Jacobian can be computed in linear time. Section 4.3 extends this linear time result to the force Jacobian, the quantity required to compute derivatives of physics constraints.

The usual way to compute aggregate momentum is to formulate the following recursion:

$$v_i = X_{i-1}^i v_{i-1} + s_i' \dot{q}_i \quad (11)$$

$$p_i = X_{i+1}^i p_{i+1} + I_i' v_i \quad (12)$$

where p_0 is the desired result.

Velocities v_i are propagated from base to leaf, and momentum p_i is propagated from leaf to base. Figure 3 shows this process. Parameter q_i appears in the coordinate transforms X_{i+1}^i and X_i^{i+1}, and so every v_j for $j > i$ depends on q_i, and every p_j for $j \geq 0$ depends on q_i. Unrolling the recursion to collect terms for $\partial p_0/\partial q_i$ requires $O(D)$ time. There are D terms q_i, and this approach will lead to an $O(D^2)$ computation for the momentum Jacobian. There is no clever way to simplify the calculation by aggregating terms when it is presented in this form.

We observe that rewriting the recursion solves this dilemma:

$$I_i^* = X_{i+1}^i I_{i+1}^* X_i^{i+1} + I_i' \quad (13)$$

$$p_i^* = X_{i+1}^i p_{i+1}^* + I_i^* v_i' \quad (14)$$

$$p_0 = p_0^* \quad (15)$$

The key thing to notice here is that p_i^* is expressed as a function of v_i', which is a local variable at link i. As a result, only propagation from leaf to base is required, and each parameter q_j does not affect terms computed for joints $j+1$ and beyond (Figure 4). Also note that p_i^* is in general not equal to p_i if $i \neq 0$. A term superscripted with an asterix should be treated only as an intermediary quantity, unless its subscript is zero in which case it is the desired aggregate result.

A linear time expression for the momentum Jacobian can be derived in a straightforward manner based on this form of the recursion. The results of this calculation are presented in Appendix B. Note that we are not simplifying or changing the outcome of the dynamics computation, only changing the order in which terms are computed. Aggregate momentum p_0 and the momentum Jacobian are exactly the same in both formulations.

4.3 Linear Time Force Jacobian

In a traditional inverse dynamics formulation, accelerations and forces are expressed as the time derivatives of Equations 11 and 12:

$$a_i = X_{i-1}^i a_{i-1} + s_i' \ddot{q}_i + v_i \hat{\times} s_i' \dot{q}_i \quad (16)$$

$$f_i = X_{i+1}^i f_{i+1} + I_i' a_i + v_i \hat{\times} I_i' v_i \quad (17)$$

[1] Final expressions for the force Jacobian and other terms of Equation 10 are summarized in Appendix B.

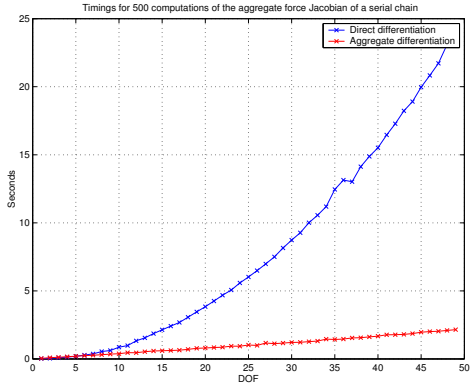

Figure 5: Timing of 500 computations of the Jacobian of the aggregate force by direct analytical differentiation and by our linear time analytical method.

where the symbol $\hat{\times}$ is the cross product operator for spatial vectors (Appendix A). As with momentum, this form results in an expression for the force Jacobian that requires $O(D^2)$ time to compute. For fast computation, we instead take the time derivative of Equation 14, which results in

$$\mathrm{f}_i^* = \mathrm{X}_{i+1}^i \mathrm{f}_{i+1}^* + \mathrm{v}_i' \hat{\times} \mathrm{p}_i^* + \mathrm{I}_i^* \mathrm{a}_i' + \dot{\mathrm{I}}_i^* \mathrm{v}_i' \tag{18}$$

This equation has the properties we are looking for. Velocity v_i' and acceleration a_i' are local to link i, and terms are propagated from leaf to base only. Note that as with aggregate momentum, f_i^* is in general different from the actual joint force f_i if $i \neq 0$.

Differentiating Equation 18 and accumulating the coefficients of derivative elements results in the simplified form as given in Appendix B. Each partial derivative of the aggregate force with respect to joint positions, velocities, and accelerations may be obtained in constant time, and hence the full Jacobian may be obtained in linear time.

4.4 Benchmarking

Figure 5 shows timing results for computation of all partial derivatives of the aggregate force by the proposed method and by direct differentiation of the Newton-Euler equations of motion. Numerically the partial derivatives are identical. The articulated model is a serial chain ranging from 3 to 50 links. As expected, the proposed method is linear in the degrees of freedom, while direct differentiation shows quadratic growth. It is also observed that despite overheads in computing aggregate intermediate terms, the linear time method shows a computational advantage with as few as 5 degrees of freedom.

4.5 The Cost of a Linear Time Algorithm

One obvious question is why has this technique of rewriting the recursion for fast computation not been explored in the robotics community? One possible reason is that there is a cost to this approach that may be higher for robotics applications than for graphics applications. In a standard Newton-Euler formulation, force parameter f_i (Equation 17) contains all of the joint force information for joint i, in particular forces in the actuated directions of motion (joint torques). In robotics, this information must be computed because it corresponds to signals sent to the motors of the robot. It must in general also be part of optimization routines, because energy consumption and joint torque limits are of particular concern when operating a robot, and none of the joints can be ignored.

In contrast, we argue that for animation of human motion, many of the effects we expect to see in physically based optimization do not depend on joint torques. We believe that physical correctness and optimization functions enforcing smooth motion are sufficient to obtain many natural characteristics of human motion.

If some torques (e.g. torques at the hip joints) are found to be important, it seems quite certain that many others (e.g. torques at the fingers) can be ignored for many motions. If a subset of K torques are required, it is straightforward to extend our approach to measure torques at these joints in $O(KD)$ time.

5 Optimization Criteria

We now return to a discussion about the objective function, $h(x)$. One traditional approach is to use the integral of the sum of squared joint torques to produce a motion that approximately minimizes energy expenditure:

$$h(x) = \int_{t=t_s}^{t_f} \left(\sum_{i=1}^{D} \tau_i^2(x,t) \right) dt \tag{19}$$

This function is expensive because computing its gradient requires $O(D^2)$ work. Adopting this function would negate our effort in constructing efficient physics constraints.

An objective function that we have found to work well is to minimize the integral of the sum of squared, weighted joint accelerations:

$$h(x) = \int_{t=t_s}^{t_f} \left(\sum_{i=1}^{D} \left(w_i \ddot{q}_i(x,t) \right)^2 \right) dt \tag{20}$$

where w_i is aggregate mass subtended at joint i with respect to the effective root. For example, the weight for the left-knee during a left-legged support is the entire body mass minus the left lower-leg. Parameters $\ddot{q}_i$ do not include translational or rotational acceleration of the character root. Note that the analytical Hessian for this objective function is constant, symmetric, positive definite, and band-diagonal.

Where a reference motion $q_R(t)$ is available, a simple objective function with low cost is to simply minimize the distance from the reference motion:

$$h(x) = \int_{t=t_s}^{t_f} \left(q(x,t) - q_R(t) \right)^2 dt \tag{21}$$

This objective function is similar to the one used in Gleicher [1997].

Other objective functions we have attempted include an integral of squared contact forces:

$$h(x) = \int_{t=t_s}^{t_f} f_c^2(t)\, dt \tag{22}$$

The Jacobian of this function is computable in linear time; our physics constraints are based upon it. Gaits generated using this function have a certain 'tip-toe' quality to them, as the function minimizes the amount of reaction force derived from the contacts.

Minimizing contact jerk (the time derivative of force) can be achieved using forward differences:

$$h(x) = \sum_{i=1}^{m-1} \left(f_c(t_i) - f_c(t_{i+1}) \right)^2 \tag{23}$$

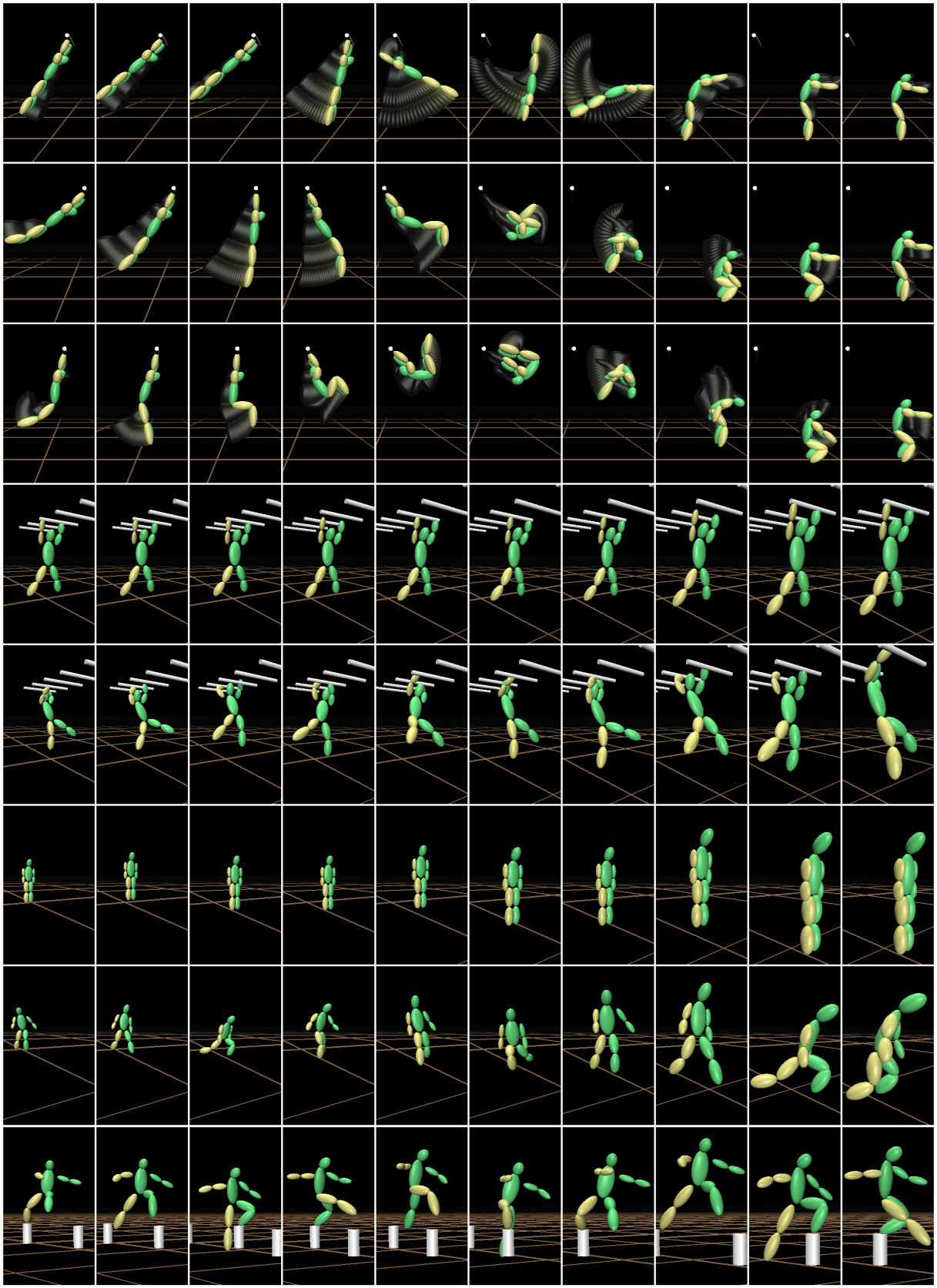

Figure 6: Samples of our results. Rows 1, 4, and 6 are initial motions. Details on the experiments are given in the text of Section 6.

Swing Setup Information	
DOF	7 (5 rotational, 2 translational)
2D/3D	2D
Number of variables	105
Implicit constraints	Hand contact during swing Feet contact during landing Swing time 0.9s, Flight time 0.6s, 0.8s
Explicit constraints	Initial COM velocity zero Fixed final pose, final joint velocity zero
Number of iterations	650
Time per iteration	0.04s
Total time	0.43min

Figure 7: All setup information for the swing example.

6 Results

Optimal Motions. Figure 6 shows a sampling of our results. The first three rows show a dismount. From top to bottom: initial motion, results with a flight time of 0.6s, and results with a flight time of 0.8s. Note the looser tuck and the higher flight trajectory in the 0.8s motion. The initial motion (shown in the top row of Figure 6) appears very unstable at landing. The character would fall over. This effect is eliminated in the optimization by enforcing the physics constraints of ground contact. Details of the optimization setup are in Figure 7. All timing information is for a 750 MHz Pentium 3 computer.

Rows 4 and 5 of Figure 6 show initial and final motion for a monkey bars example. Details are in Figure 8. Rows 6, 7, and 8 show initial and final results for a leaping character, with setup information in Figure 9. In row 7, ground penetration constraints are enforced. In row 8, they are not; the character is leaping from peg to peg. No touch-up was done on the results. In particular, the geometry of the monkey bars and the pegs was not modeled. In these examples, notice the swinging of the legs and arms, as well as body roll, pitch, and yaw. All of these effects are obtained as a result of the optimization process. In these examples, the initial motion is rigid translation of the entire character.

Our goal was to require a minimal amount of information from the animator. To set up these examples, we used 15-30 control points per degree of freedom. We found that a number of time slices (for constraint evaluation) equal to the number of control points produced good results and did not need to adjust this value for individual motions. Finer time slices would overly constrain the system, and sparser time slices allowed too much freedom for error. Each motion was set up using a constraint configuration file containing the information listed in the tables. In general, the initial motion was determined directly from constraints, with no additional user input, using linear interpolation between constrained poses. The exception was initial control points for the character root in the first example, which were set to create the overall body rotation required for the backflip. To automatically compute initial motion in a constrained pose, all joints are set at zero angle, the character is in a vertical posture, and the relevant end effector is placed at a user-specified point (e.g. hand at a specific point on the monkey bars). The vertical "zero posture" had arms up for the bar swings, legs out for the monkey bars, and arms down for the ground motions. The high bar final pose was the only pose provided as a constraint in these examples.

Timing. To empirically test the advantage of our method for fast derivative computation, we ran the peg example (bottom row of Figure 6) 5 times, each time with the identical setup except that a different technique was used to compute all required first derivatives. Figure 10 summarizes the results. The differentiation techniques tested were:

Monkey Bar Setup Information	
DOF	22 (19 rotational, 3 translational)
2D/3D	3D
Number of variables	532
Implicit constraints	Hand contact during support Support time 0.7s, Zero flight time
Explicit constraints	none
Number of iterations	1330
Time per iteration	0.11s
Total time	2.4min

Figure 8: All setup information for the monkey bar example.

Leap Setup Information	
DOF	22 (19 rotational, 3 translational)
2D/3D	3D
Number of variables	532
Implicit constraints	Foot contact during support Support time 0.35s, Flight time 0.4s
Explicit constraints	none
Number of iterations	2213
Time per iteration	0.11s
Total time	4.0min

Figure 9: All setup information for the leap example.

- **Our method.** Analytical gradient computation using our approach.
- **Direct method.** Analytical gradient obtained by direct differentiation of the equations of motion.
- **NR1.** Numerical differentiation by ordinary forward differences.
- **NR2.** Numerical differentiation by central differences.
- **NR3.** Richardson-extrapolation of order 6.

Implementation Issues. Two implementation issues were especially important for achieving the results described in this paper. First, we note that if the basis functions have local influence, the vector and matrix quantities computed during optimization are very sparse. We use the publicly-available Lancelot optimization package [Conn et al. 1992] where sparsity is accounted for by group-separability.

Second, we outline the issue of rerooting. Implementing any inverse dynamics algorithm requires selecting a character root. An ability to move the effective root to different parts of the character is very convenient. For example, when there is a single point of constraint between the character and the environment, and that point has a known and fixed trajectory, it is convenient to place the

Technique	Time per iteration	Average % error
Our method	0.11s	0
Direct method	0.62s	0
NR1	0.97s	0.10
NR2	1.92s	1.0e-04
NR3	5.73s	1.5e-06

Figure 10: Time required for one iteration of the peg example using a variety of differentiation techniques.

character root at that point. In the swing example of Figure 1, it may be convenient to root the character at the hands for the swing, at the center of mass for flight, and at the feet for landing. In a Newton-Euler inverse dynamics formulation, rerooting is typically done by changing parent / child relationships, which requires inverting joint angles and transforms at each joint and altering the flow of dynamic terms from leaves to root. Both of these changes complicate the problem description presented to the optimizer.

The effective root can be relocated more easily, however, by leaving the actual root and the flow of the dynamics computation fixed and computing velocities and accelerations at the root to maintain the desired constraint. Details are given in Appendix C.

7 Discussion

This paper contributes to physically based optimization by defining and exploring a restricted class of optimization problems where physics constraints are included and first derivatives of constraints and objective functions can be computed in linear time. The fact that first derivatives can be computed in linear time instead of quadratic time suggests that our problem is simpler than previous physically based approaches and similar in complexity to very successful kinematic approaches such as minimizing distance to a reference motion. We suspect that our solution landscape will be smoother than previous physically based optimization approaches, making it feasible to handle more complex characters.

When the optimization does not converge, we can usually trace it back to the problem setup. Sometimes it is due to overconstrained equations (setup error). But often it is due to overly restrictive parameters, such as friction coefficients, joint limits, poor selection of timings, etc. At present, timings are set by the user and their values need to be reasonable (e.g., the character cannot leap too far in too short a time). Any optimization technique that makes use of local derivatives has potential problems with local minima. Our experience, however, was that as long as an expected motion sequence could be thought of as motion about some neutral position, then when the character was started in that neutral position there was no problem descending toward the expected minimum.

We were able to create a jumping Luxo and highly dynamic human motions with good success. For less dynamic activities, our system would require additional input; physics constraints plus smooth motion would not in general produce the desired results. An extreme example is "stand for 5 seconds." Given this problem definition, our system would identify a static pose near the initial guess where the projection of the center of mass is in support area. Additional information would be required to fill in the details of the standing motion.

For activities where joint torque limits are important, this torque information must be taken into account to produce good results. An extreme example of this situation is the passive swing of a multilink chain. Minimizing accelerations while maintaining physics constraints would produce a result that was valid for the body as a whole but would require non-zero torques at the joints—no whipping motion would be seen. Minimizing sum squared torques would produce the desired results. (Of course, truly passive motion can be created much more easily using forward dynamic simulation.)

More commonly, a limited set of torques or energy terms may be important. For example, the peg running motion appears very athletic because it would require high torques at the knee and hip joints. When physical parameters at certain joints are identified as important, our method can be extended to provide and differentiate these parameters for any K joints with running times of $O(KD)$, reaching the expected bound of $O(D^2)$ when all joint torques are required. An interesting research problem is to determine automatically when torques at a given joint should be considered.

Running on flat ground shows a combination of difficulties. To make this motion appear more natural, we would need to consider proper timing for the running stride, a more accurate foot model, torques at some of the joints, and perhaps also aspects of style that are not driven by physics or energy.

Complexity in the number of degrees of freedom of the character is not the only concern in physically based optimization. The number of free parameters of the optimization problem also grows linearly with total time allotted for the animation. We have not yet attempted any long motion sequences, but we note that Liu, Gortler, and Cohen [Liu et al. 1994] have shown that time complexity can be effectively managed in an optimization context, in part because the influence of any one parameter is localized in time.

It is interesting to compare our approach to that of Liu and Popović [2002]. Their paper describes the power of patterns (e.g., momentum patterns) in creating desirable animation effects, and their approach could be adapted easily to obtain linear time performance by rewriting the momentum equations as described in Section 4.2 of this paper. The idea of dynamic patterns is an exciting one. However, relying on momentum patterns without computing interaction forces between the character and the environment may result in problems with certain types of physics constraints (e.g., keeping forces within a friction cone) when the initial motion is not favorable. In the present paper, we show that it is possible to optimize motion with physics constraints in an efficient manner, so that reasonable friction conditions, for example, can be easily enforced. We believe the combination of correct physics and knowledge of natural dynamic patterns of human motion such as momentum or movement of the center of pressure in the roll of the foot on the ground could be very powerful.

Finally, we would like to emphasize that the main advantage of our approach may be as part of a more complete animation system. Our vision is that the ability to enforce physics constraints efficiently should be just one of the tools available to the animator. Details of the desired motion could be fleshed out using motion capture data, procedural techniques, keyframes, and/or objective functions appropriate to the specific task. We have shown that physics constraints can be enforced in an efficient manner. Incorporating physics constraints into traditionally kinematic animation approaches is one direction of future work.

Acknowledgments

We would like to thank Chris Atkeson for discussions and comments on the presentation of this paper. Thanks also to Jessica Hodgins and John Hughes for many helpful suggestions during the course of this project. This work was supported in part by NSF CAREER award CCR-0093072.

References

ALBRO, J. V., SOHL, G. A., AND BOBROW, J. E. 2000. On the computation of optimal high-dives. In *Proc. IEEE Intl. Conference on Robotics and Automation*.

ARIKAN, O., AND FORSYTH, D. A. 2002. Interactive motion generation from examples. *ACM Transactions on Graphics 21*, 3 (July), 483–490.

BROTMAN, L. S., AND NETRAVALI, A. N. 1988. Motion interpolation by optimal control. In *Computer Graphics (Proceedings of SIGGRAPH 88)*, vol. 22, 309–315.

COHEN, M. F. 1992. Interactive spacetime control for animation. In *Computer Graphics (Proceedings of SIGGRAPH 92)*, vol. 26, 293–302.

CONN, A. R., GOULD, N. I. M., AND TOINT, P. L. 1992. *LANCELOT: a Fortran package for large-scale nonlinear optimization (Release A)*. No. 17 in Springer Series in Computational Mathematics. Springer Verlag, Heidelberg, Berlin, New York.

CRAWFORD, L. S. 1998. *Learning Control of Complex Skills*. PhD Thesis, UC Berkeley.

DASGUPTA, A., AND NAKAMURA, Y. 1999. Making feasible walking motion of humanoid robots from human motion capture data. In *Proc. IEEE Intl. Conference on Robotics and Automation*.

FEATHERSTONE, R. 1987. *Robot Dynamics Algorithms*. Kluwer Academic Publishers, Boston, MA.

GLEICHER, M. 1997. Motion editing with spacetime constraints. In *Proceedings of the 1997 Symposium on Interactive 3D Graphics*, 139–148.

GRZESZCZUK, R., TERZOPOULOS, D., AND HINTON, G. 1998. Neuroanimator: Fast neural network emulation and control of physics-based models. In *Proceedings of SIGGRAPH 98*, Computer Graphics Proceedings, Annual Conference Series, 9–20.

KO, H., AND BADLER, N. I. 1996. Animating human locomotion with inverse dynamics. *IEEE Computer Graphics and Applications* (March), 50–59.

LEE, J., AND SHIN, S. Y. 1999. A hierarchical approach to interactive motion editing for human-like figures. In *Proceedings of SIGGRAPH 99*, Computer Graphics Proceedings, Annual Conference Series, 39–48.

LIU, Z., AND COHEN, M. 1994. Decomposition of linked figure motion: Diving. In *5th Eurographics Workshop on Animation and Simulation*.

LIU, Z., AND COHEN, M. F. 1995. Keyframe motion optimization by relaxing speed and timing. In *6th Eurographics Workshop on Animation and Simulation*.

LIU, C. K., AND POPOVIĆ, Z. 2002. Synthesis of complex dynamic character motion from simple animations. *ACM Transactions on Graphics 21*, 3 (July), 408–416.

LIU, Z., GORTLER, S. J., AND COHEN, M. F. 1994. Hierarchical spacetime control. In *Proceedings of SIGGRAPH 94*, Computer Graphics Proceedings, Annual Conference Series, 35–42.

LIU, Z. 1996. *Efficient Animation Techniques Balancing Both User Control and Physical Realism*. PhD thesis, Princeton University.

LO, J., AND METAXAS, D. 1999. Recursive dynamics and optimal control techniques for human motion planning. In *Proceedings of Computer Animation '99*, 220–234.

NAGASAKA, K., INOUE, H., AND INABA, M. 1999. Dynamic walking pattern generation for a humanoid robot based on optimal gradient method. In *Proc. IEEE Intl. Conference on Systems, Man, and Cybernetics*, 908–913.

PANDY, M. G., AND ANDERSON, F. C. 2000. Dynamic simulation of human movement using large-scale models of the body. In *Proc. IEEE Intl. Conference on Robotics and Automation*.

POLLARD, N. S., AND REITSMA, P. S. A. 2001. Animation of humanlike characters: Dynamic motion filtering with a physically plausible contact model. In *Yale Workshop on Adaptive and Learning Systems*.

POPOVIĆ, Z., AND WITKIN, A. P. 1999. Physically based motion transformation. In *Proceedings of SIGGRAPH 99*, Computer Graphics Proceedings, Annual Conference Series, 11–20.

POPOVIĆ, J., SEITZ, S. M., ERDMANN, M., POPOVIĆ, Z., AND WITKIN, A. P. 2000. Interactive manipulation of rigid body simulations. In *Proceedings of SIGGRAPH 00*, Computer Graphics Proceedings, Annual Conference Series, 209–218.

ROSE, C. F., GUENTER, B., BODENHEIMER, B., AND COHEN, M. F. 1996. Efficient generation of motion transitions using spacetime constraints. In *Proceedings of SIGGRAPH 96*, Computer Graphics Proceedings, Annual Conference Series, 147–154.

TAKANISHI, A., ISHIDA, M., YAMAZAKI, Y., AND KATO, I. 1985. The realization of dynamic walking by the biped walking robot WL-10RD. In *Proc. Intl. Conference on Advanced Robotics*, 459–466.

VAN DE PANNE, M. 1997. From footprints to animation. *Computer Graphics Forum 16*, 4 (Oct.), 211–223.

VUKOBRATOVIĆ, M. 1970. On the stability of biped locomotion. *IEEE Trans. Biomedical Engineering 17*, 1, 25–36.

WITKIN, A., AND KASS, M. 1988. Spacetime constraints. In *Computer Graphics (Proceedings of SIGGRAPH 88)*, vol. 22, 159–168.

YAMANE, K., AND NAKAMURA, Y. 2000. Dynamics filter – concept and implementation of on-line motion generator for human figures. In *Proc. IEEE Intl. Conference on Robotics and Automation*.

Appendix A: Overview of Spatial Notation

For rotational joints, joint axis s_i' is represented as follows:

$$\mathrm{s}_i' = \begin{bmatrix} \underline{\alpha}_i \\ \underline{\mathrm{r}}_i \times \underline{\alpha}_i \end{bmatrix} \tag{24}$$

where $\underline{\alpha}_i$ is the axis of rotation and $\underline{\mathrm{r}}_i$ is the point about which the joint rotates. Both terms are expressed in the body i local frame, and the superscript $'$ on s_i' indicates that the spatial vector is expressed in body i local frame coordinates. For prismatic (translational) joints,

$$\mathrm{s}_i' = \begin{bmatrix} 0 \\ \underline{\alpha}_i \end{bmatrix} \tag{25}$$

where $\underline{\alpha}_i$ is the axis of translation. We represent multiple degree of freedom joints as sequences of single degree of freedom joints, connected by massless and inertialess bodies.

Spatial velocity and acceleration are represented as:

$$\mathrm{v}_i' = \dot{q}_i \mathrm{s}_i', \qquad \mathrm{a}_i' = \ddot{q}_i \mathrm{s}_i' \tag{26}$$

where $\dot{q}$ and $\ddot{q}$ are the scalar velocity and acceleration of degree of freedom i—angular velocity and acceleration for rotational joints and linear velocity and acceleration for prismatic joints.

Spatial force also combines linear and angular quantities:

$$\mathrm{f}_i = \begin{bmatrix} \underline{\mathrm{f}}_i^a \\ \underline{\mathrm{f}}_i^b \end{bmatrix} \tag{27}$$

where $\underline{\mathrm{f}}_i^a$ is linear force and $\underline{\mathrm{f}}_i^b$ is torque.

Spatial transform X_i^j takes spatial quantities from frame i to frame j:

$$\mathrm{X}_i^j = \begin{bmatrix} \underline{\mathrm{R}}_i^j & 0 \\ \underline{\mathrm{r}}_i^j \times \underline{\mathrm{R}}_i^j & \underline{\mathrm{R}}_i^j \end{bmatrix} \tag{28}$$

where $\underline{\mathrm{R}}_i^j$ is the 3x3 matrix rotating vectors from frame i to frame j, and $\underline{\mathrm{r}}_i^j$ is the position of frame j expressed in frame i.

Spatial inertia represents both body mass and rotational inertia:

$$\mathrm{I}_i' = \begin{bmatrix} -m_i \underline{\mathrm{c}}_i \times & m_i \\ \underline{\mathrm{I}}_i - m_i \underline{\mathrm{c}}_i \times \underline{\mathrm{c}}_i \times & m_i \underline{\mathrm{c}}_i \times \end{bmatrix} = \begin{bmatrix} -m_i \tilde{\underline{\mathrm{c}}}_i & m_i \\ \underline{\mathrm{I}}_i - m_i \tilde{\underline{\mathrm{c}}}_i \tilde{\underline{\mathrm{c}}}_i & m_i \tilde{\underline{\mathrm{c}}}_i \end{bmatrix} \tag{29}$$

where $\underline{\mathrm{c}}_i$ is the vector to the body i center of mass in frame i, $\tilde{\underline{\mathrm{c}}}_i$ is that same vector expressed as a cross product matrix, m_i is the mass of body i, and $\underline{\mathrm{I}}_i$ is the rotational inertia of body i about its center of mass, expressed in frame i.

The spatial cross product $\hat{\times}$ is expressed in matrix form as follows:

$$\tilde{\mathrm{z}} = \begin{bmatrix} \underline{\mathrm{z}}^a \\ \underline{\mathrm{z}}^b \end{bmatrix} \hat{\times} = \begin{bmatrix} \underline{\mathrm{z}}^a \times & 0 \\ \underline{\mathrm{z}}^b \times & \underline{\mathrm{z}}^a \times \end{bmatrix} = \begin{bmatrix} \tilde{\underline{\mathrm{z}}}^a & 0 \\ \tilde{\underline{\mathrm{z}}}^b & \tilde{\underline{\mathrm{z}}}^a \end{bmatrix} \tag{30}$$

The spatial transpose is

$$\mathrm{z}^S = \begin{bmatrix} \underline{\mathrm{z}}^a \\ \underline{\mathrm{z}}^b \end{bmatrix}^S = \begin{bmatrix} (\underline{\mathrm{z}}^b)^T & (\underline{\mathrm{z}}^a)^T \end{bmatrix} \tag{31}$$

where superscript S indicates a spatial transpose and superscript T indicates an ordinary 3-vector transpose.

Appendix B: First Derivative Expressions

Conventions. For clarity, we assume a serial chain composed of of L single degree of freedom links numbered $1,2,...,L$. The subscript zero is reserved for the quantities representing the entire multibody. Where superscripted with an asterix (e.g., I_i^*) the quantity represents aggregated information accumulated from L to i.

Newton-Euler equations of motion. The equations of motion of a serial multibody chain are compactly expressed in recursive form as follows:

$$\begin{aligned}
\mathrm{v}_i &= \mathrm{X}_{i-1}^i \mathrm{v}_{i-1} + \mathrm{s}_i' \dot{q}_i && (32)\\
\mathrm{a}_i &= \mathrm{X}_{i-1}^i \mathrm{a}_{i-1} + \mathrm{s}_i' \ddot{q}_i + \mathrm{v}_i \hat{\times} \mathrm{s}_i' \dot{q}_i && (33)\\
\mathrm{p}_i &= \mathrm{X}_{i+1}^i \mathrm{p}_{i+1} + \mathrm{I}_i' \mathrm{v}_i && (34)\\
\mathrm{f}_i &= \mathrm{X}_{i+1}^i \mathrm{f}_{i+1} + \mathrm{I}_i' \mathrm{a}_i + \mathrm{v}_i \hat{\times} \mathrm{I}_i' \mathrm{v}_i && (35)
\end{aligned}$$

where the second and fourth equations are the time derivatives of the first and third equations respectively. For a multibody rooted at its base joint, the following end condition for simulating gravity is used:

$$\mathrm{a}_0 = -G\,,\ \mathrm{v}_0 = 0\,,\ \mathrm{p}_{L+1} = 0\,,\ \mathrm{f}_{L+1} = 0 \qquad (36)$$

The Newton-Euler equations propagate quantities in two directions. To compute aggregate quantities and their derivatives efficiently, rewrite the dynamics equations as follows:

Aggregate equations — momentum. Compute:

$$\begin{aligned}
\mathrm{I}_i^* &= \mathrm{X}_{i+1}^i \mathrm{I}_{i+1}^* \mathrm{X}_i^{i+1} + \mathrm{I}_i' && (37)\\
\mathrm{p}_i^* &= \mathrm{X}_{i+1}^i \mathrm{p}_{i+1}^* + \mathrm{I}_i^* \mathrm{s}_i' \dot{q}_i && (38)
\end{aligned}$$

where body inertias and momenta propagate from the leaf to the base. I_0^* and p_0^* are the aggregate inertia and momentum of the entire body, and p_0^* is equal to p_0 computed from the previous Newton-Euler recursive equations. However, p_i^* is in general not equal to p_i where $i \neq 0$ and should only be used as an intermediary quantity in computing the aggregates.

Aggregate equations — force. Compute:

$$\begin{aligned}
\dot{\mathrm{I}}_i^* &= \mathrm{X}_{i+1}^i \dot{\mathrm{I}}_{i+1}^* \mathrm{X}_i^{i+1} + (\tilde{\mathrm{s}}_i' \mathrm{I}_i^* - \mathrm{I}_i^* \tilde{\mathrm{s}}_i') \dot{q}_i && (39)\\
\mathrm{f}_i^* &= \mathrm{X}_{i+1}^i \mathrm{f}_{i+1}^* + \mathrm{s}_i' \dot{q}_i \hat{\times} \mathrm{p}_i^* + \mathrm{I}_i^* \mathrm{s}_i' \ddot{q}_i + \dot{\mathrm{I}}_i^* \mathrm{s}_i' \dot{q}_i && (40)
\end{aligned}$$

where f_0^* is the aggregate force applied to the entire body, equal to f_0 computed previously. As before, f_i^* is in general not equal to f_i where $i \neq 0$.

First derivatives – Aggregate equations. The Jacobian may be constructed in time linear in the number of degrees of freedom as follows. All partial derivatives are expressed in frame i.[2]

$$\begin{aligned}
\frac{\partial \mathrm{I}_0^*}{\partial q_i} &= (\tilde{\mathrm{s}}_i' \mathrm{I}_i^* - \mathrm{I}_i^* \tilde{\mathrm{s}}_i') && (41)\\
\frac{\partial \mathrm{p}_0^*}{\partial \dot{q}_i} &= \mathrm{I}_i^* \mathrm{s}_i' && (42)\\
\frac{\partial \mathrm{p}_0^*}{\partial q_i} &= \frac{\partial \mathrm{I}_0^*}{\partial q_i} \mathrm{v}_{i-1}^i + \mathrm{s}_i' \hat{\times} \mathrm{p}_i^* && (43)\\
\frac{\partial \mathrm{f}_0^*}{\partial \ddot{q}_i} &= \frac{\partial \mathrm{p}_0^*}{\partial \dot{q}_i} && (44)\\
\frac{\partial \mathrm{f}_0^*}{\partial \dot{q}_i} &= \frac{\partial \mathrm{p}_0^*}{\partial q_i} + \dot{\mathrm{I}}_i^* \mathrm{s}_i' + \mathrm{v}_i \hat{\times} \frac{\partial \mathrm{p}_0^*}{\partial \dot{q}_i} && (45)\\
\frac{\partial \mathrm{f}_0^*}{\partial q_i} &= \mathrm{s}_i' \hat{\times} \mathrm{f}_i^* \\
&\quad + \frac{\partial \mathrm{I}_0^*}{\partial q_i} \mathrm{a}_{i-1}^i \\
&\quad + \mathrm{v}_{i-1}^i \hat{\times} (\mathrm{s}_i' \hat{\times} \mathrm{p}_i^* + \frac{\partial \mathrm{I}_0^*}{\partial q_i} \mathrm{v}_{i-1}^i) \\
&\quad + (\tilde{\mathrm{s}}_i' \dot{\mathrm{I}}_i^* - \dot{\mathrm{I}}_i^* \tilde{\mathrm{s}}_i') \mathrm{v}_{i-1}^i && (46)
\end{aligned}$$

where

$$\begin{aligned}
\mathrm{v}_{i-1}^i &= (\mathrm{X}_{i-1}^i \mathrm{v}_{i-1}) && (47)\\
\mathrm{a}_{i-1}^i &= (\mathrm{X}_{i-1}^i \mathrm{a}_{i-1}) && (48)
\end{aligned}$$

Appendix C: Changing the Effective Root

Suppose we wish to place the effective root of the character at the point on body i that is located at point $\underline{\mathrm{r}}'$ in body i coordinates. We wish this point to have linear velocity $\underline{\mathrm{b}}_{r,des}$ and linear acceleration $\dot{\underline{\mathrm{b}}}_{r,des}$, expressed in the world coordinate frame. The current velocity of body i in the body i frame is v_i'.

$$\mathrm{v}_i' = \begin{bmatrix} \underline{\mathrm{v}}_i^a \\ \underline{\mathrm{v}}_i^b \end{bmatrix} \qquad (49)$$

The velocity of point $\underline{\mathrm{r}}'$ on body i is computed in the body i frame as

$$\mathrm{v}_r' = \begin{bmatrix} \underline{\mathrm{v}}_i^a \\ \underline{\mathrm{v}}_i^b - \underline{\mathrm{r}}' \times \underline{\mathrm{v}}_i^a \end{bmatrix} \qquad (50)$$

and transformed to the world frame as follows:

$$\mathrm{v}_r = \mathrm{X}_i^0 \mathrm{v}_r' = \begin{bmatrix} \underline{\mathrm{v}}_r^a \\ \underline{\mathrm{v}}_r^b \end{bmatrix} \qquad (51)$$

In Equation 51, $\underline{\mathrm{v}}_r^b$ is the linear velocity of the effective root expressed in world coordinates. This velocity should be $\underline{\mathrm{b}}_{r,des}$. To obtain the correct velocity at the effective root, simply add the desired correction $(\underline{\mathrm{b}}_{r,des} - \underline{\mathrm{v}}_r^b)$ to the reference frame velocity:

$$\mathrm{v}_0 = \begin{bmatrix} 0 \\ \underline{\mathrm{b}}_{r,des} - \underline{\mathrm{v}}_r^b \end{bmatrix} \qquad (52)$$

The adjustment to a_0 is derived using similar reasoning.

When these changes are made, the actual character root can remain at the pelvis, for example, while the effective root is moved from hand to pelvis to foot or other bodies as needed. The effective root can even be set to the center of mass to obtain correct ballistic motion during flight. Derivatives of all equations with respect to parameters describing the motion can be computed in $O(D)$ time.

[2] When implementing these expressions, we found it essential to compare the numerical values of these derivatives to the identical derivatives obtained from an alternative technique such as numerical differentiation.