Adaptive Dynamics of Articulated Bodies

Stephane Redon* Nico Galoppo† Ming C. Lin‡

Department of Computer Science
University of North Carolina at Chapel Hill

Figure 1: **Adaptive dynamics of articulated characters**. In this complex scene, 200 human characters, represented by 17,800 rigid bodies and 19,000 degrees of freedom, are suddenly pushed away from the camera due to applied forces. Our adaptive dynamics algorithm allows an animator to progressively reduce the number of simulated joints in the characters as their distance to the camera increases, while automatically determining *which* joints should be animated to best approximate the characters motion. Depending on the total amount of simplification specified by the animator, a potentially significant speed-up can be achieved over typical linear-time forward dynamics algorithms.

Abstract: *Forward dynamics is central to physically-based simulation and control of articulated bodies. We present an adaptive algorithm for computing forward dynamics of articulated bodies: using novel motion error metrics, our algorithm can automatically simplify the dynamics of a multi-body system, based on the desired number of degrees of freedom and the location of external forces and active joint forces. We demonstrate this method in plausible animation of articulated bodies, including a large-scale simulation of 200 animated humanoids and multi-body dynamics systems with many degrees of freedom. The graceful simplification allows us to achieve up to two orders of magnitude performance improvement in several complex benchmarks.*

Keywords: dynamics, kinematics, level-of-detail, simulation, articulated bodies

1 Introduction

Recent advances in modeling human motion have made it possible to animate articulated figures realistically through performance capture, autonomous control, learning techniques, etc. [Bruderlin and Calvert 1996; Gillespie and Colgate 1997; Granieri et al. 1995; Faure 1999; Featherstone and Orin 2000; Ko and Badler 1993; Perlin 1995; Popovic and Witkin 1999; Yamane et al. 2004]. Despite the exciting progress in the field, simulating a large group of articulated characters or a dynamical system with many degrees of freedom remains a computational challenge. One of the central components of any control or simulation system for articulated bodies is forward dynamics. Forward dynamics computes the acceleration and the resulting motion of each link, based on the given set of external forces and active joint forces. The best known algorithms [Bae and Haug 1987; Brandl et al. 1986; Featherstone 1987; Hollerbach 1980; McMillan and Orin 1995] have a linear-time dependence on the number of degrees of freedom. For a complex scene with many articulated figures or with many degrees of freedom, however, dynamic simulation of the entire multi-body system can become extremely costly.

Barzel et al. [1996] introduced the idea of "plausible" motion, i.e. motion that could happen and look physically plausible to the viewers. For many visual applications or real-time interaction, accurately simulating all the details of the real world is not necessary. In fact, it is often sufficient to provide effective "motion texture" to make the scene appear more realistic, without committing much computational resources [Barzel et al. 1996]. For instance, the main character should be animated with the highest degree of realism possible using motion capture data or accurate full-body simulation, while the crowds secondary to the story can be simulated at much lower fidelity. Several techniques have been proposed for accelerating various types of dynamic simulation. Yet, to the best of our knowledge, there exists no known general algorithm for automatic simplification of articulated body dynamics.

Main Results: In this paper, we present a novel adaptive algorithm for automatic simplification of the forward dynamics of articulated bodies. We introduce a new hybrid-body representation that allows articulated links to be simulated as a combination of articulated and rigid bodies updated at runtime, based on the *bounded* acceleration error of simulated motion. The runtime complexity of our algorithm is $O(N_A) + f \times (O(N_A) + O(N_R^3))$, where N_A is the number of active joints, f is the frequency at which the set of active joints is updated, and N_R is the number of links being rigidified during an update of the set of active joints. Typically, f is a small constant and $N_A \ll N$. In comparison to existing methods, our algorithm offers the users complete control over the number of degrees of freedom that should be simulated, while automatically selecting the set of active joints so as to best approximate the articulated body motion, based on novel customizable motion error metrics. Depending on the amount of simplification specified by the user, our approach can significantly accelerate large-scale simulations of many articulated bodies or multi-body dynamics with many degrees of freedom.

*Stephane Redon is now at INRIA. E-mail: stephane.redon@inria.fr.
† E-mail: nico@cs.unc.edu. ‡ E-mail: lin@cs.unc.edu

Organization: The rest of the paper is organized as follows. Section 2 briefly surveys prior research on forward dynamics computation and dynamics simplification. Section 3 gives a brief overview of the divide-and-conquer algorithm for forward dynamics [Featherstone 1999a; Featherstone 1999b] that provides the basic structure for our work. We introduce a new multilevel representation for articulated bodies in Section 4, and describe our method for automatic simplification of articulated body dynamics using novel customizable motion metrics in Section 5. Section 6 presents a method to simplify the update of the simulation coefficients. We present and analyze results, and discuss limitations and potential research directions in Section 7, and conclude in Section 8.

2 Related Work

2.1 Forward Dynamics

Multi-body systems and forward dynamics have been active areas of research for decades in computer animation and robotics. We give a brief overview of related work here. For more detail, we refer the readers to a recent survey [Featherstone and Orin 2000].

To model and control the locomotion of articulated characters and robots, forward dynamics is one of the most essential steps for physically-based simulation algorithms. Some of the best known linear-time methods rely on a recursive formulation of the motion equations [Bae and Haug 1987; Brandl et al. 1986; Featherstone 1987; Hollerbach 1980; McMillan and Orin 1995]. Several authors have independently proposed to simplify the motion equations by developing new notations and formulations, including the spatial notation [Featherstone 1999a; Featherstone 1999b], the spatial operator algebra [Rodriguez et al. 1991], and Lie-Group formulations [Mueller and Maisser 2003]. Baraff presented a general, non-iterative linear-time algorithm based on Lagrange multipliers [Baraff 1996]. More recently, parallel algorithms have also been introduced to compute the forward dynamics of articulated bodies using multiple processors [Fijany et al. 1995; Featherstone 1999a; Featherstone 1999b; Anderson and Duan 2000; Yamane and Nakamura 2002].

2.2 Simulation Levels of Detail

In contrast to the vast literature on geometric simplification, there is relatively little investigation on *simulation levels of detail* (SLODs) for physically-based animation. Simulation levels of detail can be computed based on pre-recorded motion sequences, kinematics, procedural approaches, or dynamics computation.

Earlier human motion models in computer animation used procedurally generated motion, simplified dynamics and control algorithms, off-line motion mapping, or motion play-back to create SLODs [Bruderlin and Calvert 1989; Bruderlin and Calvert 1996; Girard and Maciejewski 1985; Granieri et al. 1995; Ko and Badler 1993; Perlin 1995]. Carlson and Hodgins investigated the viability of SLODs in accelerating the overall computation for a group of legged creatures [Carlson and Hodgins 1997]. Their experimental results indicated the promising potential of dynamics simplification for real-time animation. Popovic and Witkin introduced motion transformation techniques preserving the essential properties of animated character motion with drastically fewer number of degrees of freedom using space-time constraints [Popovic and Witkin 1999]. Multon et al. suggested a series of simplified walking models for moving on complex terrain [Multon et al. 1999]. Faure [1999] proposed a method to iteratively refine the computation of the forward dynamics of an articulated body by progressively correcting bilateral constraint errors. Similar to view-dependent culling for interactive rendering, Chenney et al. proposed view-dependent dynamics simplification by ignoring what is not visible to the viewer [Chenney and Forsyth 1997; Chenney et al. 1999; Chenney et al. 2001]. SLODs have also been investigated to accelerate simulations using particle systems [O'Brien et al. 2001], for hair modeling [Bertails et al. 2003; Ward et al. 2003; Ward and Lin 2003], to simulate plant motion [Perbet and Cani 2001; Beaudoin and Keyser 2004], etc.

Redon and Lin [2005] introduced an algorithm for adaptive simplification of forward *quasi-statics* of articulated bodies. This algorithm does not handle articulated bodies with non-zero velocities (*i.e.* the dynamics case), and does not allow the users to precisely control the number of degrees of freedom. In this paper, we propose perhaps the first algorithm for adaptive simplification of forward dynamics of articulated bodies. Using novel customizable motion error metrics, our algorithm can automatically simplify the dynamics of a multi-body system, based on the desired number of degrees of freedom, as well as external forces and active joint forces. Because it is based on a dynamics algorithm formulated in the configuration space, our algorithm ensures that the joint constraints are satisfied at all time.

3 Articulated-Body Dynamics

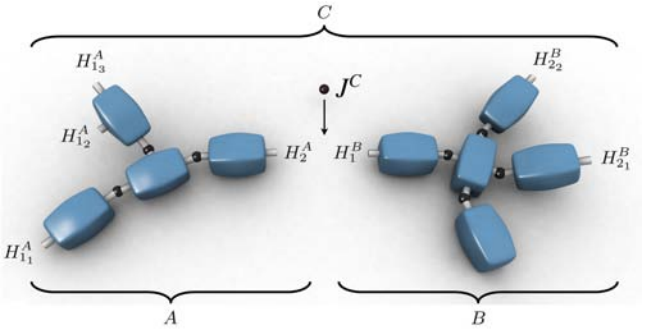

Figure 2: *An articulated body C is formed by connecting two articulated bodies A and B through J^C, the principal joint of C.*

Our algorithm is built upon the divide-and-conquer algorithm (DCA) introduced by Featherstone to compute the forward dynamics of an articulated body [Featherstone 1999a; Featherstone 1999b]. The algorithm is applicable to articulated bodies with branches and loops, although we deal in this paper with acyclic articulated bodies.

Featherstone [Featherstone 1987] defines an articulated body as a system of rigid bodies with a set of *handles*, *i.e.* locations attached to some of the rigid bodies where forces may be applied, and shows that the acceleration of these handles are affine functions of the forces applied to them, according to the following *articulated-body equations*:

$$\begin{bmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \\ \vdots \\ \mathbf{a}_m \end{bmatrix} = \begin{bmatrix} \mathbf{\Phi}_1 & \mathbf{\Phi}_{12} & \cdots & \mathbf{\Phi}_{1m} \\ \mathbf{\Phi}_{21} & \mathbf{\Phi}_2 & \cdots & \mathbf{\Phi}_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{\Phi}_{m1} & \mathbf{\Phi}_{m2} & \cdots & \mathbf{\Phi}_m \end{bmatrix} \begin{bmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \\ \vdots \\ \mathbf{f}_m \end{bmatrix} + \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \vdots \\ \mathbf{b}_m \end{bmatrix} \tag{1}$$

In this equation, $\mathbf{a}_i$ is the 6×1 spatial acceleration of handle i, $\mathbf{f}_i$ is the 6×1 spatial force applied to handle i, $\mathbf{b}_i$ is the 6×1 bias acceleration of handle i (the acceleration handle i would have if all handle forces were zero), $\mathbf{\Phi}_i$ is the 6×6 inverse articulated-body inertia of handle i, and $\mathbf{\Phi}_{ij}$ is the 6×6 cross-coupling inverse inertia between handles i and j.

In the DCA, an articulated body is recursively defined as a pair of articulated bodies connected by a joint. The sequence of assembly operations is described in a binary assembly tree, in which each

node represents a subassembly. The leaf nodes represent rigid bodies, and the root node describes the complete articulated body and its relationship to the world coordinate system. To compute the forward dynamics of the articulated bodies, four *complete* traversals of the assembly tree are performed. The first two consist of one bottom-up and one top-down pass, in which the new body positions and velocities are computed from the joint positions and accelerations, as well as the coordinate transformation matrices between the different coordinate systems in the algorithm. Then the *main pass* is a bottom-up traversal, in which the DCA calculates the articulated-body equations (1) for each node in the assembly tree from those of its children. Finally, in the top-down *back-substitution pass*, the kinematic constraint forces are propagated down the tree to compute all the joint accelerations.

3.1 Main pass

Let C be an articulated body formed by assembling an articulated body A with $m+1$ handles $H^A_{1_1}, \ldots, H^A_{1_m}, H^A_2$, and an articulated body B with $n+1$ handles $H^B_1, H^B_{2_1}, \ldots, H^B_{2_n}$. As illustrated in Figure 2, C is formed by connecting H^A_2 of A and H^B_1 of B. The other handles are non-connecting handles. The joint used to connect A and B is called the *principal joint* of C.

By isolating handle H^A_2, the articulated-body equation of A can be written:

$$\begin{bmatrix} \mathbf{a}^A_1 \\ \mathbf{a}^A_2 \end{bmatrix} = \begin{bmatrix} \mathbf{\Phi}^A_1 & \mathbf{\Phi}^A_{12} \\ \mathbf{\Phi}^A_{21} & \mathbf{\Phi}^A_2 \end{bmatrix} \begin{bmatrix} \mathbf{f}^A_1 \\ \mathbf{f}^A_2 \end{bmatrix} + \begin{bmatrix} \mathbf{b}^A_1 \\ \mathbf{b}^A_2 \end{bmatrix},$$

where $\mathbf{\Phi}^A_1 = mat(\mathbf{\Phi}^A_{1_{ij}})$ is a $6m \times 6m$ *composite* matrix of cross-coupling and inverse articulated-body inertias involving the non-connecting handles in A, $\mathbf{\Phi}^A_{12} = (\mathbf{\Phi}^A_{21})^T = col(\mathbf{\Phi}^A_{1_i 2})$ is a $6m \times 6$ composite matrix of cross-coupling inverse inertias between the non-connecting handles in A and the connecting handle H^A_2, and $\mathbf{\Phi}^A_2$ is the 6×6 inverse articulated-body inertia of handle H^A_2. The accelerations, forces, and bias acceleration are vectors of appropriate dimensions involving the corresponding handles.

The articulated-body equation of B can be similarly written:

$$\begin{bmatrix} \mathbf{a}^B_1 \\ \mathbf{a}^B_2 \end{bmatrix} = \begin{bmatrix} \mathbf{\Phi}^B_1 & \mathbf{\Phi}^B_{12} \\ \mathbf{\Phi}^B_{21} & \mathbf{\Phi}^B_2 \end{bmatrix} \begin{bmatrix} \mathbf{f}^B_1 \\ \mathbf{f}^B_2 \end{bmatrix} + \begin{bmatrix} \mathbf{b}^B_1 \\ \mathbf{b}^B_2 \end{bmatrix},$$

where the connecting handle H^B_1 is singled out. Finally, the articulated-body equation of C can be written:

$$\begin{bmatrix} \mathbf{a}^A_1 \\ \mathbf{a}^B_2 \end{bmatrix} = \begin{bmatrix} \mathbf{\Phi}^C_1 & \mathbf{\Phi}^C_{12} \\ \mathbf{\Phi}^C_{21} & \mathbf{\Phi}^C_2 \end{bmatrix} \begin{bmatrix} \mathbf{f}^A_1 \\ \mathbf{f}^B_2 \end{bmatrix} + \begin{bmatrix} \mathbf{b}^C_1 \\ \mathbf{b}^C_2 \end{bmatrix},$$

where *all* quantities are composite vectors or matrices involving the remaining $m+n$ handles $H^A_{1_1}, \ldots, H^A_{1_m}$ and $H^B_{2_1}, \ldots, H^B_{2_n}$.

Featherstone [Featherstone 1999a; Featherstone 1999b] shows that the articulated-body coefficients of C can be expressed in terms of those of A and B. Let $\mathbf{S}$ denote the $6 \times d_J$ joint's motion subspace, where d_J is the number of degrees of freedom of the joint[1], and let $\mathbf{Q}$ denote the $d_J \times 1$ vector of active joint forces (*i.e.* forces applied by joint actuators). Setting

$$\begin{array}{ll} \mathbf{V} = (\mathbf{\Phi}^A_2 + \mathbf{\Phi}^B_1)^{-1} & \boldsymbol{\beta} = \mathbf{b}^A_2 - \mathbf{b}^B_1 + \dot{\mathbf{S}}\dot{\mathbf{q}}_\circ \\ \mathbf{W} = \mathbf{V} - \mathbf{V}\mathbf{S}(\mathbf{S}^T\mathbf{V}\mathbf{S})^{-1}\mathbf{S}^T\mathbf{V} & \boldsymbol{\gamma} = \mathbf{W}\boldsymbol{\beta} + \mathbf{V}\mathbf{S}(\mathbf{S}^T\mathbf{V}\mathbf{S})^{-1}\mathbf{Q} \end{array},$$

where $\dot{\mathbf{q}}_\circ$ is the velocity of C's principal joint, the articulated-body coefficients of C are:

$$\begin{array}{ll} \mathbf{\Phi}^C_1 = \mathbf{\Phi}^A_1 - \mathbf{\Phi}^A_{12}\mathbf{W}\mathbf{\Phi}^A_{21} & \mathbf{b}^C_1 = \mathbf{b}^A_1 - \mathbf{\Phi}^A_{12}\boldsymbol{\gamma} \\ \mathbf{\Phi}^C_2 = \mathbf{\Phi}^B_2 - \mathbf{\Phi}^B_{21}\mathbf{W}\mathbf{\Phi}^B_{12} & \mathbf{b}^C_2 = \mathbf{b}^B_2 + \mathbf{\Phi}^B_{21}\boldsymbol{\gamma} \\ \mathbf{\Phi}^C_{21} = \mathbf{\Phi}^B_{21}\mathbf{W}\mathbf{\Phi}^A_{21} = (\mathbf{\Phi}^C_{12})^T & \end{array}. \quad (2)$$

[1] A typical value for a revolute joint in link coordinates is $\mathbf{S} = (0,0,1,0,0,0)^T$.

The main pass of the DCA performs these computations for each node of the assembly tree, from the bottom up. The leaf-node coefficients are:

$$\mathbf{\Phi}_i = \mathbf{\Phi}_{ij} = \mathbf{I}^{-1} \qquad \mathbf{b}_i = \mathbf{I}^{-1}(\mathbf{f}_k - \mathbf{v} \times \mathbf{I}\mathbf{v}), \quad (3)$$

where $\mathbf{I}$ and $\mathbf{v}$ are, respectively, the spatial inertia and the spatial velocity of the rigid body, and $\mathbf{f}_k$ is an acceleration-independent external force applied to the rigid body.

3.2 Back-Substitution Pass

After the main pass, the back-substitution pass of the DCA computes the joint accelerations and the kinematic constraint forces of each node in the assembly tree, from the root to the leaves. Each node receives the values of the constraint forces $\mathbf{f}^A_1$ and $\mathbf{f}^B_2$ from its parent. The $d_J \times 1$ acceleration $\ddot{\mathbf{q}}_\circ$ of the principal joint of a node C is then:

$$\ddot{\mathbf{q}}_\circ = (\mathbf{S}^T\mathbf{V}\mathbf{S})^{-1}(\mathbf{Q} - \mathbf{S}^T\mathbf{V}(\mathbf{\Phi}^A_{21}\mathbf{f}^A_1 - \mathbf{\Phi}^B_{12}\mathbf{f}^B_2 + \boldsymbol{\beta})), \quad (4)$$

and the kinematic constraint forces are:

$$\mathbf{f}^B_1 = -\mathbf{f}^A_2 = \mathbf{W}\mathbf{\Phi}^A_{21}\mathbf{f}^A_1 - \mathbf{W}\mathbf{\Phi}^B_{12}\mathbf{f}^B_2 + \boldsymbol{\gamma} \quad (5)$$

The root node describes the relation of the complete articulated body to the world, and has only one articulated-body child B. The handle force $\mathbf{f}^B_2$ is assumed to be known (usually $\mathbf{f}^B_2 = \mathbf{0}$, since all acceleration-independent forces have already been accounted for in the bias accelerations). If the articulated body has a floating base, then $\mathbf{f}^B_1 = \mathbf{0}$ and $\mathbf{a}^B_1 = \mathbf{\Phi}^B_{12}\mathbf{f}^B_2 + \mathbf{b}^B_1$. However, if the articulated body is attached to a fixed base through a joint, the joint acceleration and forces are computed using Equations (4) and (5), but with the coefficients relative to A set to zero. Note that it is possible to simulate gravity by substituting $\mathbf{b}^B_1 - \mathbf{a}_g$ for $\mathbf{b}^B_1$, which eliminates the need to account for gravity through the action of an external force upon all leaf nodes.

4 Hybrid Bodies

In this section, we introduce *hybrid bodies*, a new multilevel representation of an articulated body, which enables us to simplify the dynamics of an articulated body by controlling how many degrees of freedom are simulated.

4.1 Definitions

Let J be a joint in the articulated body. The joint will be said to be *active* if it is simulated, *i.e.* if its acceleration, velocity and position are computed. When it is not simulated, the joint is said to be *inactive* or, alternatively, to have been *rigidified*. To simplify the dynamics of any articulated body, we select a set of active joints and simulate only those joints. We define a *hybrid body* as an articulated body whose set of active joints is a subtree of the assembly tree, with an identical root. Consequently, any node in the assembly tree is either *rigid*, when all the joints in the subassembly are inactive or the node itself is a leaf node, or *hybrid*, when the principal joint of the node is active, but some descendents of the nodes are rigid. The set of hybrid nodes is called the *active region*. The set of rigid nodes is called the *rigid region*, and the nodes which are rigid but whose parent is hybrid constitute the *rigid front*. We also classify the nodes according to applied external forces and active joint forces. An internal node is said to be in the *force update region* if the principal joint of the node is subject to an active joint force, or some descendents of the node are in the force update region, or some descendents of the node are rigid bodies to which external forces are applied. Finally, for reasons that will become clear later on, we call the union of the active region and the force update region the *update region*, and the corresponding complementary set

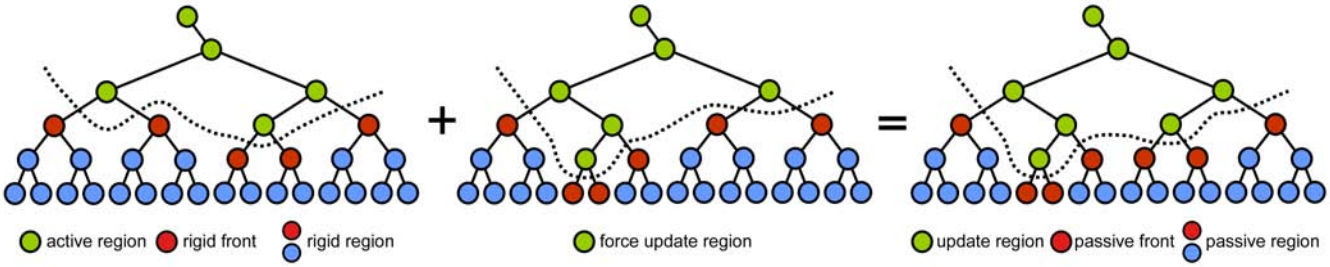

Figure 3: **Classification of the nodes of a hybrid body.** *The* active region *(left) is composed of the nodes which are simulated. The* force update region *(middle) contain the internal nodes which are directly influenced by an external force or an active joint force. The* update region *(right) is the union of the active region and the force update region.*

of nodes the *passive region*. A passive node whose parent is in the update region is said to be in the *passive front*. Figure 3 summarizes the different regions of the assembly tree of a hybrid body.

This multilevel representation of an articulated body allows us to gracefully simplify the dynamics of an articulated body, from a *fully articulated state*, where all the joints are active, to a *fully rigid state*, where all the joints are rigidified and the articulated body behaves like a single rigid body.

4.2 Hybrid-Body Dynamics

The dynamics equations of a hybrid body can be easily deduced from the articulated-body equations (2): wherever a joint is inactive, we replace the corresponding motion subspace $\mathbf{S}$ by $\mathbf{0}$. Since a node C can only be rigidified if both its children A and B have been rigidified or were actual rigid bodies, the rigid-state coefficients of a rigidified node can be recursively computed from those of its children:

$$\begin{aligned} \mathbf{\Phi}^C &= \mathbf{\Phi}^C_i = \mathbf{\Phi}^C_{ij} = \mathbf{\Phi}^B(\mathbf{\Phi}^A + \mathbf{\Phi}^B)^{-1}\mathbf{\Phi}^A \\ \mathbf{b}^C &= \mathbf{b}^C_i = \mathbf{b}^A - \mathbf{\Phi}^A(\mathbf{\Phi}^A + \mathbf{\Phi}^B)^{-1}(\mathbf{b}^A - \mathbf{b}^B) \end{aligned} \tag{6}$$

The dynamics coefficients of the hybrid nodes are computed using the regular articulated-body equations (2). Note that a rigidified node behaves exactly like a rigid body. Especially, its inverse inertia is a symmetric, positive definite matrix. Moreover, the inverse inertias and bias accelerations of all its handles are equal up to a rigid-body transformation.

4.3 Hybrid-Body Simulation

Assume that we know the set of active joints, and that this set does not change for several frames. Assume moreover that the inverse inertias of all nodes in the assembly tree have been computed, including the rigid inverse inertias of the rigid and rigidified nodes. Assume finally that the applied external forces and the active joint forces are known. The dynamics of a hybrid body can be simulated by modifying the DCA, resulting in the following *multilevel forward dynamics algorithm*:

1. **Bias accelerations computation:** as in the DCA, the bias accelerations are recursively computed from the bottom up. However, the computations take different forms depending on the region the node belongs to: the bias accelerations of an active node are computed as in the DCA using Equation (2); those of a rigid node in the force update region are computed using Equation (6); those of a node in the passive front are computed using Equation (3) with the rigid or rigidified inertia of the node. The bias accelerations of any passive node out of the passive front *do not have to be computed*.
2. **Acceleration update:** the accelerations of the *active* joints are computed using Equation (4).
3. **Velocity and position update:** the velocities and positions of the active joints are updated using the joint accelerations.
4. **Inverse inertias update:** the inverse inertias of the hybrid nodes are updated using the new joint positions. We will show in Section 6 that the inverse inertias of the rigid nodes do not need to be recomputed.

When Step 4 completes, the simulation can proceed to the next time step.

5 Adaptive dynamics

5.1 Overview

The multilevel algorithm introduced in the previous section enables us to control the complexity of the dynamics of an articulated body. Having chosen a set of active joints, we are able to simulate these joints only and potentially obtain a substantial performance improvement over a method which would simulate all of the joints. However, the actual challenge is to **predict *which* joints should be activated so as to best approximate the motion of the articulated body, *without* computing the accelerations of all the joints in the articulated body**.

In order to determine which joints should be simulated, we introduce two classes of motion metrics: acceleration-based and velocity-based. They help us formalize the simplification of the dynamics of an articulated body. Based on these motion metrics, we periodically perform the following *active region update algorithm*:

1. **Conversion to the fully articulated state:** the hybrid body is converted back to its fully articulated state. This step involves computing some articulated-body coefficients and acceleration metric coefficients. We show in Section 6 that only a limited subtree of the assembly tree has to be traversed in order to perform the conversion.
2. **Acceleration update**
   (a) **Determination of the acceleration update region:** we determine the *acceleration update region*, *i.e.* the subset of nodes of the full articulated body which matter the most according to the acceleration metric. The union of the previous active region and the acceleration update region is the *transient active region*, *i.e.* the region temporarily considered as the active region.
   (b) **Joint accelerations re-computation:** in order to ensure a physically-based simplification of the articulated body acceleration, we convert the articulated body to the corresponding *transient hybrid body*, and compute the joint accelerations in the transient active region.

3. **Velocity update**

   (a) **Determination of the new active region:** we update the joint velocities and the velocity metric values of the nodes in the transient active region. We then determine the set of nodes which are considered to be important according to the velocity metric. This set becomes the *new active region*.

   (b) **Rigidification:** if one or more nodes become inactive due to the update of the active region, we determine a set of impulses that we must apply to the transient hybrid body to perform the rigidification of these nodes.

4. **Conversion to the new hybrid state:** the articulated body is converted from the transient hybrid state to the new hybrid state, by updating a limited number of articulated-body coefficients and acceleration metric coefficients.

We provide details of the algorithm in the rest of this section.

5.2 Motion Metrics

Let C be an articulated body with N_C joints, and let $\ddot{\mathbf{q}}^C = (\ddot{\mathbf{q}}_1, \ldots, \ddot{\mathbf{q}}_{N_C})^T$ denote the composite acceleration of C. The composite velocity $\dot{\mathbf{q}}^C = (\dot{\mathbf{q}}_1, \ldots, \dot{\mathbf{q}}_{N_C})^T$ is similarly defined. In order to prioritize the degrees of freedom, we introduce the following two motion metrics. The *acceleration metric value* and *velocity metric value* of an articulated body C with acceleration $\ddot{\mathbf{q}}^C$ and velocity $\dot{\mathbf{q}}^C$ are:

$$\mathscr{A}(C) = \sum_{i \in C} \ddot{\mathbf{q}}_i^T \mathbf{A}_i \ddot{\mathbf{q}}_i \qquad \mathscr{V}(C) = \sum_{i \in C} \dot{\mathbf{q}}_i^T \mathbf{V}_i \dot{\mathbf{q}}_i.$$

where $\mathbf{A}_i$ and $\mathbf{V}_i$, $i \in C$, are symmetric, positive definite (SPD) $d_{J_i} \times d_{J_i}$ weight matrices, and d_{J_i} is the number of degrees of freedom of joint i in C. The matrices $\mathbf{A}_i$ and $\mathbf{V}_i$ can be adapted to achieve a specific type of approximation, but we require them to only (potentially) depend on the position $\mathbf{q}_i$ of joint i. The simplest choice for the matrices $\mathbf{A}_i$ and $\mathbf{V}_i$ is the identity matrix.

The key to our approach is the demonstration that it is possible to compute the acceleration metric value of any node C *without having to compute the accelerations of the joints in* C:

Theorem 1 (Acceleration metric equation)
The acceleration metric value $\mathscr{A}(C)$ of an articulated body C, formed by assembling an articulated body A with $m+1$ handles and an articulated body B with $n+1$ handles, is a quadratic function of the handle forces acting upon C:

$$\mathscr{A}(C) = \begin{bmatrix} \mathbf{f}_1^A \\ \mathbf{f}_2^B \end{bmatrix}^T \begin{bmatrix} \boldsymbol{\Psi}_1^C & \boldsymbol{\Psi}_{12}^C \\ \boldsymbol{\Psi}_{21}^C & \boldsymbol{\Psi}_2^C \end{bmatrix} \begin{bmatrix} \mathbf{f}_1^A \\ \mathbf{f}_2^B \end{bmatrix} + \begin{bmatrix} \mathbf{f}_1^A \\ \mathbf{f}_2^B \end{bmatrix}^T \begin{bmatrix} \mathbf{p}_1^C \\ \mathbf{p}_2^C \end{bmatrix} + \eta^C, \tag{7}$$

where $\boldsymbol{\Psi}_1^C$ is a $6m \times 6m$ matrix, $\boldsymbol{\Psi}_2^C$ is a $6n \times 6n$ matrix, $\boldsymbol{\Psi}_{12}^C = (\boldsymbol{\Psi}_{21}^C)^T$ is a $6m \times 6n$ matrix, $\mathbf{p}_1^C$ is a $6m \times 1$ vector, $\mathbf{p}_2^C$ is a $6n \times 1$ vector, and η^C is in $\mathbb{R}$. Equation (7) is called the acceleration metric equation *of C.*

Furthermore, it can be shown by induction on the height of the assembly tree that, just like the articulated-body coefficients, the acceleration metric coefficients $\boldsymbol{\Psi}_1^C$, $\boldsymbol{\Psi}_2^C$, $\boldsymbol{\Psi}_{12}^C$, $\boldsymbol{\Psi}_{21}^C$, $\mathbf{p}_1^C$, $\mathbf{p}_2^C$ and η^C can be computed in a bottom-up fashion:

Theorem 2 (Acceleration metric coefficients)
Let C be an articulated body with children A and B, and let $\mathbf{A}_\circ$ denote the weight matrix of the principal joint of C. Setting

$$\begin{aligned} &\mathbf{U} = (\mathbf{S}^T\mathbf{V}\mathbf{S})^{-1}\mathbf{S}^T\mathbf{V} && \mathbf{R} = (\mathbf{S}^T\mathbf{V}\mathbf{S})^{-1}(\mathbf{Q} - \mathbf{S}^T\mathbf{V}\boldsymbol{\beta}) \\ &\mathbf{N} = \mathbf{U}^T\mathbf{A}_\circ\mathbf{U} && \mathbf{Y} = \mathbf{N} + \mathbf{W}(\boldsymbol{\Psi}_2^A + \boldsymbol{\Psi}_1^B)\mathbf{W} \\ &\mathbf{Z} = 2\mathbf{U}^T\mathbf{A}_\circ\mathbf{R} + \mathbf{W}(\mathbf{p}_2^A - \mathbf{p}_1^B) - 2\mathbf{W}(\boldsymbol{\Psi}_2^A + \boldsymbol{\Psi}_1^B)\boldsymbol{\gamma}, \end{aligned}$$

the coefficients of the acceleration metric $\mathscr{A}(C)$ can be written:

$$\begin{aligned} \boldsymbol{\Psi}_1^C &= \boldsymbol{\Psi}_1^A + \boldsymbol{\Phi}_{12}^A\mathbf{Y}\boldsymbol{\Phi}_{21}^A - (\boldsymbol{\Phi}_{12}^A\mathbf{W}\boldsymbol{\Psi}_{21}^A + \boldsymbol{\Psi}_{12}^A\mathbf{W}\boldsymbol{\Phi}_{21}^A) \\ \boldsymbol{\Psi}_2^C &= \boldsymbol{\Psi}_2^B + \boldsymbol{\Phi}_{21}^B\mathbf{Y}\boldsymbol{\Phi}_{12}^B - (\boldsymbol{\Phi}_{21}^B\mathbf{W}\boldsymbol{\Psi}_{12}^B + \boldsymbol{\Psi}_{21}^B\mathbf{W}\boldsymbol{\Phi}_{12}^B) \\ \boldsymbol{\Psi}_{21}^C &= -\boldsymbol{\Phi}_{21}^B\mathbf{Y}\boldsymbol{\Phi}_{21}^A + \boldsymbol{\Phi}_{21}^B\mathbf{W}\boldsymbol{\Psi}_{21}^A + \boldsymbol{\Psi}_{21}^B\mathbf{W}\boldsymbol{\Phi}_{21}^A = (\boldsymbol{\Psi}_{12}^C)^T \\ \mathbf{p}_1^C &= \mathbf{p}_1^A - \boldsymbol{\Phi}_{12}^A\mathbf{Z} - 2\boldsymbol{\Psi}_{12}^A\boldsymbol{\gamma} \\ \mathbf{p}_2^C &= \mathbf{p}_2^B + \boldsymbol{\Phi}_{21}^B\mathbf{Z} + 2\boldsymbol{\Psi}_{21}^B\boldsymbol{\gamma} \\ \eta^C &= \eta^A + \eta^B + \mathbf{R}^T\mathbf{A}_\circ\mathbf{R} + \boldsymbol{\gamma}^T(\boldsymbol{\Psi}_2^A + \boldsymbol{\Psi}_1^B)\boldsymbol{\gamma} - \boldsymbol{\gamma}^T(\mathbf{p}_2^A - \mathbf{p}_1^B). \end{aligned} \tag{8}$$

The coefficients of a leaf node are all zero, since a leaf node represents a rigid body which does not contain any joint. The coefficients of the root node are obtained by zeroing the coefficients relative to A in Equation (8).

In the next section, we show how we use Equation (7) to control the back-substitution pass and determine an *acceleration update region*, *i.e.* a subtree of C's assembly tree whose nodes matter the most according to the acceleration metric, without having to traverse the whole assembly tree.

5.3 Acceleration Update

5.3.1 Determination of the Acceleration Update Region

Let C be an articulated body. When the back-substitution pass begins, the forces $\mathbf{f}_2^B$ acting on the complete articulated body C are assumed to be known, and the acceleration metric value $\mathscr{A}(C)$ can be obtained from Equation (7). Similarly, during the back-substitution pass, the forces acting upon a node can be used to compute not only the principal joint's acceleration and forces, but also the acceleration metric value of the node, before processing its descendants.

Let ε_{max} be a user-defined threshold. If $\mathscr{A}(C) \leqslant \varepsilon_{max}$, we consider all joint accelerations in C to be sufficiently close to zero according to the acceleration metric. We stop the back-substitution pass right at the root node, implicitly setting all joint accelerations to zero, and decide that the acceleration update region is empty.

However, if the acceleration metric value $\mathscr{A}(C)$ is strictly larger than ε_{max}, we do not stop the traversal of the assembly tree right at the root node. Instead, we compute the joint acceleration and forces of C's principal joint, and recursively traverse the descendants of C using a priority queue to order the nodes according to their acceleration metric value: a node A has a higher priority than a node B if $\mathscr{A}(A) > \mathscr{A}(B)$. Thus, during the traversal, we descend first in the regions that matter the most according to the acceleration metric. We simultaneously maintain the *current acceleration error* ε_c, *i.e.* the error in the acceleration metric value of the full articulated body that would result from stopping the recursion. Thus, we set $\varepsilon_c = \mathscr{A}(C)$ at the beginning of the back-substitution pass, where C is the root node of the assembly tree. Then, whenever the acceleration $\ddot{\mathbf{q}}_\circ$ of the principal joint of a node C is computed during the recursion, we subtract the contribution $\ddot{\mathbf{q}}_\circ^T\mathbf{A}_\circ\ddot{\mathbf{q}}_\circ$ of C's principal joint from ε_c, and we push the children A and B of C into the priority queue. Since the weight matrices $\mathbf{A}_i$ are SPD, the current acceleration error can only decrease as more nodes are processed, and the back-substitution pass can be stopped as soon as the current acceleration error becomes smaller than ε_{max}. The acceleration update region is the set of N_U nodes that have been processed during this partial back-substitution pass, and the time required to determine it is $O(N_U)$. Note that, alternatively, the recursion can be

stopped when a user-defined number of nodes has been processed. This allows the user to precisely control the cost of determining an acceleration update region.

5.3.2 Joint Accelerations Re-Computation

Simply zeroing some joint accelerations and keeping the joint accelerations computed during the determination of the acceleration update region can lead to inconsistencies in the simulation, as illustrated in Figure 4. In this example, the articulated body is a chain of 5 links $l_1, \ldots, l_5$ which are initially motionless, and a force $\mathbf{f}_k$ is applied to the second link in the x direction. All joints are prismatic and can move in the x direction, and all links have the same mass m. Under the action of $\mathbf{f}_k$, only two joints J_1 and J_2 have non-zero, opposite accelerations, and only the second link begins to move, with acceleration $\mathbf{a}_2 = \mathbf{f}_k/m$ (Figure 4(a)).

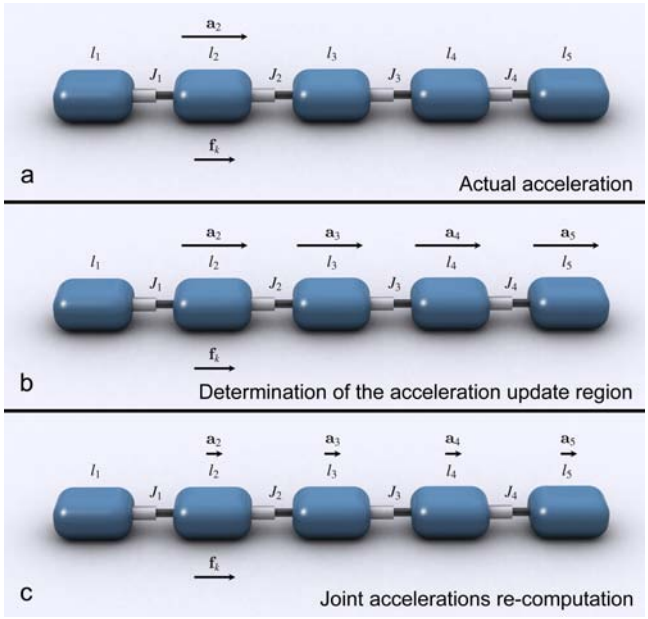

Figure 4: **Physically-based acceleration approximation.** *Once the acceleration update region has been determined, the joint accelerations are re-computed using the transient hybrid inertias to ensure a physically-based simplification of the articulated body acceleration (*cf *Section 5.3.2).*

Assume however that, during the determination of the acceleration update region, only the acceleration of joint J_1 has been computed, and the acceleration of joint J_2 has been considered to be zero (for example because the user chose to directly control the size of the acceleration update region, regardless of the resulting acceleration error). As a consequence, the links l_3, l_4 and l_5 *also* begin to move, with acceleration $\mathbf{a}_2$. The fact that these links begin to move directly results from the combined effect of the reduction of the number of degrees of freedom and the application of the force, and is not a problem *per se*. However, the articulated body now behaves as if *larger forces* than the initial one had been applied to it (Figure 4(b)).

Thus, we recompute the accelerations of the joints in the acceleration update region, as well as those of the joints which were active at the previous time step, using a partially rigidified version of the articulated body. We define the *transient active region*, *i.e.* the region temporarily considered as the active region, as the union of the previous active region and the acceleration update region, and form the corresponding *transient hybrid body*. As in Section 4, this step involves replacing the inverse inertias and bias accelerations by their rigid or hybrid counterparts, which again can be achieved without traversing the whole assembly tree (*cf* Section 6). We then compute the accelerations of the joints in the transient active region. These joint accelerations are now consistent with the reduced number of degrees of freedom in the transient hybrid body, and the forces applied to the articulated body. In the example of Figure 4, the accelerations of the links $l_2, \ldots, l_5$ become equal to $\mathbf{a}_2 = \mathbf{f}_k/(4m)$, since after the rigidification of J_2, J_3 and J_4, the action of the force $\mathbf{f}_k$ applied to l_2 has an effect on l_3, l_4 and l_5 as well (Figure 4(c)).

5.4 Velocity update

5.4.1 Determination of the New Active Region

Once the joint accelerations have been determined, we update the velocities of the joints in the transient active region, since only these joints might have a non-zero acceleration. We perform the update from the bottom up, starting from the transient rigid front, and simultaneously compute the velocity metric value $\mathscr{V}(C)$ of each processed node as follows. The velocity metric value of a node in the transient rigid front is 0, as all joints in the node have been rigidified. The velocity metric value of a node C in the transient active region is computed in constant time from the velocity $\dot{\mathbf{q}}_\circ$ of its principal joint and the velocity metric values of its children A and B: $\mathscr{V}(C) = \dot{\mathbf{q}}_\circ^T \mathbf{V}_\circ \dot{\mathbf{q}}_\circ + \mathscr{V}(A) + \mathscr{V}(B)$. When the velocities and the velocity metric values of all joints in the transient active region have been updated, we determine which nodes are important according to the velocity metric, through a procedure similar to the one used to determine the acceleration update region. The stopping criterion can be either an error threshold ε_{max} or the number of traversed nodes. The set of nodes traversed during this step becomes the new active region, and the joint velocities outside the new active region are now considered to be zero.

5.4.2 Rigidification

When the previous step completes, the velocity of each joint in the transient active region might be non-zero. However, some of these joints might not belong to the new active region, and thus should have a zero joint velocity. Similar to the acceleration update, these joint velocities cannot simply be set to zero before inactivating the joints, as this could result in inconsistencies in the motion of the articulated body. In order to cancel the velocities of the joints which are going to be inactivated, we determine and apply some *rigidification impulses* to the transient hybrid body.

Let $\dot{\mathbf{q}}^R$ denote the composite vector which contains the joint velocities that have to be cancelled, and let $\mathbf{Q}^R$ denote the composite vector of impulses that are going to be applied to the joints whose velocity must be cancelled. Both $\dot{\mathbf{q}}^R$ and $\mathbf{Q}^R$ are d_R-dimensional vectors, where d_R is the total number of degrees of freedom in the involved joints. The required instantaneous change of velocities is $\Delta\dot{\mathbf{q}}^R = -\dot{\mathbf{q}}^R$. Assuming that the effects of external forces and active joint forces (including gravity) are negligible compared to the effects of the impulses, it can be shown that there exists a SPD $d_R \times d_R$ matrix $\mathbf{K}^R$ such that $\mathbf{K}^R\mathbf{Q}^R = \Delta\dot{\mathbf{q}}^R$. The coefficients of the i-th column of $\mathbf{K}^R$ are computed by applying a unit impulse Q_i on the i-th degree of freedom in $\dot{\mathbf{q}}^R$ and reporting the accelerations induced in the d_R degrees of freedom. This is similar to the approach introduced in [Kokkevis et al. 1996]. This step involves performing d_R times the multilevel forward dynamics algorithm described in Section 4. However, since the hybrid inverse inertias of the transient hybrid body have already been computed during the recalculation of the joint accelerations, and the joint positions do not change when applying the impulses, only Steps 1 and 2 have to be performed for each of the d_R passes. Moreover, these steps can be highly optimized in this case: in Step 1, only the node where a unit impulse is applied and its ascendant nodes have non-zero bias accelerations; in Step 2, only the nodes which are going to be inactivated and their ascendant nodes have to be processed. Once the

matrix $\mathbf{K}^R$ is known, the composite impulse $\mathbf{Q}^R$ is computed and applied to the transient hybrid body by a final application of the multilevel algorithm. Because $(\mathbf{K}^R)^{-1}$ is a submatrix of the joint-space inertia matrix (*i.e.* the kinetic energy matrix) of the complete articulated body, it can be shown that the rigidification impulses cannot increase the kinetic energy of the articulated body.

Note that, when rigidification impulses are applied, the velocities of some joints in the new active region might be modified. In our tests, however, potential "popping artifacts" appeared to be fairly limited. This is essentially due to the following: (a) the joints that are rigidified typically have small velocities compared to the joints that remain active, and thus the rigidification impulses have a relatively small effect on the active joints' velocities; (b) the rigidification impulses cannot increase the kinetic energy of the articulated body; (c) some authors have noted that humans are typically less sensitive to errors in the dynamics of objects than to errors in their geometry (*e.g.* [Baraff and Witkin 1997]).

6 Coefficient Update

In this section, we describe how we update the various coefficients involved in the multilevel forward dynamics algorithm and the active region update algorithm. We show that it is possible to limit the update of the coefficients to a *subtree* of the assembly tree.

6.1 Hierarchical State Representation

Let C be an articulated body formed by assembling two articulated bodies A and B. We call the handles used to connect A and B the *principal handles* of A and B. The root node has no principal handle. As in Featherstone [Featherstone 1999a], we assign a unique reference frame to each handle in which all the quantities related to that handle are expressed. When the *quadratic coefficients*, *i.e.* the regular, rigid and hybrid inverse inertias, as well as the acceleration metric coefficients $\mathbf{\Psi}_1^C$, $\mathbf{\Psi}_2^C$, $\mathbf{\Psi}_{12}^C$ and $\mathbf{\Psi}_{21}^C$, are expressed in the handles reference frames, they only depend on the position of C's principal joint, through the coefficients $\mathbf{W}$ and $\mathbf{Y}$, as well as on their counterparts in A and B. Moreover, the weight matrices $\mathbf{A}_\circ$ and $\mathbf{V}_\circ$ of a node C only (potentially) depend on the position of C's principal joint by definition. Consequently, these coefficients only need to be updated in the active region. In order to apply a force on one of the rigid bodies composing the articulated body, however, we need to express the force in the reference frames of the handles attached to this rigid body. To do this, we maintain in each node of the assembly tree three types of coordinates transformation matrices that we use to obtain the position of a rigid body in the world frame [Redon and Lin 2005]. These three types of transforms depend on joint positions and have to be updated in the active region only[2]. Similarly, in order to express object velocities in the world frame when necessary, we maintain three corresponding types of relative velocities, which also have to be updated in the active region only. These relative velocities allow us to obtain the world frame velocities of all handles in the update region and the passive front, and where necessary in the passive region when determining the acceleration update region, without having to traverse the whole assembly tree.

6.2 Linear Coefficients Tensors

The *linear coefficients*, *i.e.* the bias accelerations of a fully articulated body, as well as the linear and constant acceleration metric coefficients $\mathbf{p}_1^C$, $\mathbf{p}_2^C$ and η^C, also depend on external forces and active joint forces applied to the rigid bodies and joints, and the velocities of the rigid bodies. During the multilevel forward dynamics algorithm, the linear coefficients of the nodes in the passive front can be directly computed using Equation (3), because the rigid or rigidified inertia and the world frame velocity of the node are available. The coefficients in the update region are then computed from the passive front up to the root using Equations (2) and (6). Consequently, the linear coefficients of the passive nodes out of the passive front never have to be computed.

To determine the acceleration update region, however, we have to use the fully articulated version of the linear coefficients, which prevents us from using the rigid or rigidified inertias. Moreover, we do not know in advance which joint accelerations are going to be computed during the back-substitution pass, and we might thus have to compute the linear coefficients of some passive nodes which are not on the passive front. In order to avoid traversing the whole assembly tree, we introduce some *linear coefficients tensors* which allow us to obtain the linear coefficients of any passive node *in constant time*, without having to traverse the node's descendants. Especially, we demonstrate that the linear coefficients of a passive node C can be obtained directly from the velocities of the handles of C:

Theorem 3 (Linear coefficients tensors)
Let C be a node in the passive zone. The linear coefficients $\mathbf{b}_1^C$, $\mathbf{p}_2^C$, $\mathbf{p}_1^C$ and $\mathbf{p}_2^C$ of C can be obtained from the product of composite rank-three tensors $\mathbf{B}_1^C$, $\mathbf{B}_2^C$, $\mathbf{P}_1^C$ and $\mathbf{P}_2^C$, and the handles velocities:

$$\begin{array}{ll}(\mathbf{b}_1^C)_a = (\mathbf{B}_1^C)_{abc}(\mathbf{v}_1^C)_b(\mathbf{v}_1^C)_c & (\mathbf{p}_1^C)_a = (\mathbf{P}_1^C)_{abc}(\mathbf{v}_1^C)_b(\mathbf{v}_1^C)_c \\ (\mathbf{b}_2^C)_a = (\mathbf{B}_2^C)_{abc}(\mathbf{v}_2^C)_b(\mathbf{v}_2^C)_c & (\mathbf{p}_2^C)_a = (\mathbf{P}_2^C)_{abc}(\mathbf{v}_2^C)_b(\mathbf{v}_2^C)_c\end{array}.$$

These equations are formulated using the Einstein summation convention, where an expression in which an index appears more than once is summed over all possible values of the index (the ranges depend on the dimensions of the composite spatial quantities). Similarly, the constant acceleration metric coefficient η^C can be obtained from the product of a rank-four tensor $\mathbf{E}^C$ and the velocity $\mathbf{v}^C$ of the principal handle of C:

$$\eta^C = (\mathbf{E}^C)_{abcd}(\mathbf{v}^C)_a(\mathbf{v}^C)_b(\mathbf{v}^C)_c(\mathbf{v}^C)_d.$$

To convert the articulated body to its fully articulated state, the linear coefficients are computed first on the passive front using these tensors, and then in the update region using Equations (2) and (8). Then, when a passive node is processed during the determination of the acceleration update region, its linear coefficients are computed using the linear coefficients tensors. It can be shown by induction on the height of the assembly tree that the linear coefficients tensors can be computed from the bottom-up:

Theorem 4 (Linear coefficients tensors computation)
Let C be an articulated body with children A and B, and let $\mathbf{A}_\circ$ denote the weight matrix of the principal joint of C. Setting

$$\begin{aligned}(\mathbf{\Gamma})_{abc} &= (\mathbf{W})_{ak}(\mathbf{B}_2^A - \mathbf{B}_1^B)_{kbc} \\ (\mathbf{L})_{abc} &= (\mathbf{U})_{ak}(\mathbf{B}_1^B - \mathbf{B}_2^A)_{kbc} \\ (\mathbf{\Delta})_{abc} &= 2(\mathbf{U}^T\mathbf{A}_\circ)_{ak}(\mathbf{L})_{kbc} + (\mathbf{W})_{ak}(\mathbf{P}_2^A - \mathbf{P}_1^B)_{kbc} \\ &\quad - 2(\mathbf{W}(\mathbf{\Psi}_2^A + \mathbf{\Psi}_1^B))_{ak}(\mathbf{\Gamma})_{kbc},\end{aligned}$$

the linear coefficients tensors of C can be computed from those of A and B:

$$\begin{aligned}(\mathbf{B}_1^C)_{abc} &= (\mathbf{B}_1^A)_{abc} - (\mathbf{\Phi}_{12}^A)_{ak}(\mathbf{\Gamma})_{kbc} \\ (\mathbf{B}_2^C)_{abc} &= (\mathbf{B}_2^B)_{abc} + (\mathbf{\Phi}_{21}^B)_{ak}(\mathbf{\Gamma})_{kbc} \\ (\mathbf{P}_1^C)_{abc} &= (\mathbf{P}_1^A)_{abc} - (\mathbf{\Phi}_{12}^A)_{ak}(\mathbf{\Delta})_{kbc} - 2(\mathbf{\Psi}_{12}^A)_{ak}(\mathbf{\Gamma})_{kbc} \\ (\mathbf{P}_2^C)_{abc} &= (\mathbf{P}_2^B)_{abc} + (\mathbf{\Phi}_{21}^B)_{ak}(\mathbf{\Delta})_{kbc} + 2(\mathbf{\Psi}_{21}^B)_{ak}(\mathbf{\Gamma})_{kbc} \\ (\mathbf{E}^C)_{abcd} &= (\mathbf{E}^A + \mathbf{E}^B)_{abcd} + (\mathbf{L})_{kab}(\mathbf{A}_\circ)_{kl}(\mathbf{L})_{lcd} \\ &\quad + (\mathbf{\Gamma})_{kab}(\mathbf{\Psi}_2^A + \mathbf{\Psi}_1^B)_{kl}(\mathbf{\Gamma})_{lcd} - (\mathbf{\Gamma})_{kab}(\mathbf{P}_2^A - \mathbf{P}_1^B)_{kcd}.\end{aligned} \quad (9)$$

[2]Note that the position of a rigid body is expressed in the world frame only when an external force is applied to this rigid body. In particular, it is *not* necessary to compute the positions of all the rigid bodies in the world frame in order to perform the simulation. Furthermore, these world transforms are not even required to display the articulated body when the graphics library supports hierarchical representations (*e.g.* OpenGL).

The tensors $\mathbf{P}_1^C$, $\mathbf{P}_2^C$ and $\mathbf{E}^C$ of a leaf node are all zero, since a leaf node represents a rigid body which does not contain any joint, while the tensors $\mathbf{B}_1^C$ and $\mathbf{B}_2^C$ of a leaf node are determined by rewriting the spatial cross product in Equation (3) using the spatial equivalent of the Levi-Civita symbol. The coefficients of the root node are obtained by zeroing the coefficients relative to A in Equation (9).

The linear coefficients tensors only depend on the joint positions, and thus should be updated in the active region only. However, because we only use them in the passive region, we can dramatically reduce the cost of maintaining these tensors by updating them only when a node *becomes* rigid.

7 Results and Applications

7.1 Features and Benefits

The algorithms introduced in this paper have been implemented in C++ and tested on a 2.8 GHz Pentium PC with 2GBytes of RAM. Although we initially considered using a fourth-order Runge-Kutta integrator, our system proved to be stable enough with a simple explicit Euler integrator. We believe that our algorithm might actually increase the stability of the simulation for three reasons: (a) several researchers have reported that articulated bodies are more difficult to simulate and control when the number of degrees of freedom increases (*e.g.* [Ascher et al. 1997; Featherstone 2004]), while our method helps by rigidifying some subassemblies; (b) the joint accelerations are re-computed in step (2)(b) of the active region update algorithm to ensure a physically-based simplification of the joints accelerations; (c) similarly, the rigidification impulses allow for a physically-based simplification of the joint velocities.

We first present an empirical analysis of our algorithm and demonstrate how adaptive dynamics allows us to gracefully reduce the complexity of an articulated body motion and increase the efficiency of the dynamics computations. Figure 5 shows the progressive motion simplification of a pendulum with 300 degrees of freedom, under the sole influence of gravity, as the user reduces the number of active joints. The five rows represent the same time segment, and the pendulum has the same initial state in each case. For reference, Fig. 5(a) shows the motion of the fully articulated pendulum, when 300 joints are active. In this case, the average time required to perform the dynamics computations is about 5 milliseconds per time step. Fig. 5(b)-(e) show the motion simplification for the pendulum, when 100, 50, 20, and finally 1 node(s) are simulated. The resulting average computational costs per time step in these examples are respectively about 1.7, 0.7, 0.25, and 0.02 milliseconds, providing up to two orders of magnitude speed-up.

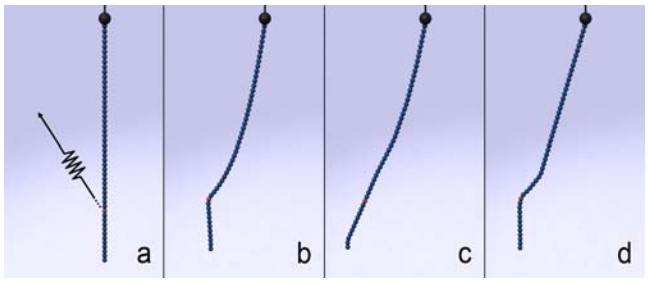

Figure 6: **Adaptive active region determination. a:** *one of the links of a* 50*-link pendulum is attached to a point in the environment through a spring.* **b:** *equilibrium state when* 50 *joints are active.* **c:** *equilibrium state when* 5 *joints are active, without adaptive determination of the active joints.* **d:** *equilibrium state when* 5 *joints are active, with adaptive determination of the active joints by our algorithm.*

Adaptive determination of the active joints is crucial to achieve a high-quality approximation of an articulated body motion, despite the potentially drastic simplification imposed by an animator. Figure 6 shows a pendulum with 50 joints in which one of the links is attached to a spring. Fig. 6(a) shows the initial state when the spring force is applied at the attachment point. Fig. 6(b) shows the final equilibrium state of the fully articulated pendulum. Fig. 6(c) shows the final equilibrium state of a hybrid body with only 5 active joints, which are selected prior to the beginning of the simulation by a breadth-first traversal of the pendulum assembly tree. Arbitrarily choosing the active joints results in a poor approximation of the equilibrium state when only few joints are active. Fig. 6(d) shows the final equilibrium state of a pendulum with only 5 active joints adaptively selected at runtime by our algorithm. Even in this extreme simplification case where 90% of nodes are rigidified, our algorithm provides a high-quality approximation of the articulated body motion.

	N_A						
	Adaptive dynamics						DCA
N_F	1	50	100	150	200	250	300
1	0.02	0.51	1.71	2.97	4.04	5.40	5.89
5	0.07	0.49	1.54	3.21	4.34	5.19	5.89
10	0.16	0.73	1.47	3.03	4.22	5.16	5.88
20	0.40	0.90	1.60	3.00	4.72	5.56	5.90
50	0.98	1.40	2.16	3.58	4.99	5.59	5.92
250	1.83	2.30	2.97	4.28	5.43	6.31	6.04

Table 1: Performance of our adaptive dynamics algorithm *vs* the DCA, when simulating the 300-link pendulum of Figure 5 (timings in milliseconds, *cf* Section 7.2).

7.2 Performance Analysis

The overall complexity of our algorithm is $O(N_A) + f \times (O(N_A) + O(N_R^3))$, where N_A is the number of active joints, f is the frequency at which the active region is updated (*e.g.* every 20 time steps), and N_R is the number of nodes that have to be rigidified. Note that, although the size N_U of the acceleration update region might be set arbitrarily, it is typically preferable to choose a number smaller than the size of the active region, so that the number of nodes that have to be rigidified is zero or a small value. Furthermore, note that the accelerations of the active joints are computed anyway during the re-computation of the accelerations. Thus, when N_A is large, N_U can be set to small values without affecting the responsiveness of the articulated body to external forces and active joint forces. Similarly, the frequency f allows the user to take advantage of the potentially high temporal coherence in the dynamics simulation (for example when small time steps are used). This again enables a trade-off between the responsiveness of the articulated body to changes (due to external forces, active joint forces, or inertial terms) and the efficiency of the overall dynamics simulation. Because of the various traversals performed in the active region update algorithm, the average cost per node is approximately three times as high as the average cost per node in the DCA or in our multilevel forward dynamics algorithm. Hence, the amortized cost of our algorithm is $O((1+3\times f)\times N_A) + f \times O(N_R^3)$.

We have thus fixed the update frequency f to $1/20$, and have tested our algorithm by varying the number N_F of forces applied at each time step to the pendulum, as well as the size N_A of the active region. The size of the acceleration update region was equal to the size of the active region in our tests. Table 1 provides the average running time of our algorithm (in milliseconds) depending on the size of the active region N_A and the number of forces N_F applied to the 300-link pendulum. The simplification of the dynamics of an articulated body enabled by our algorithm results in performance

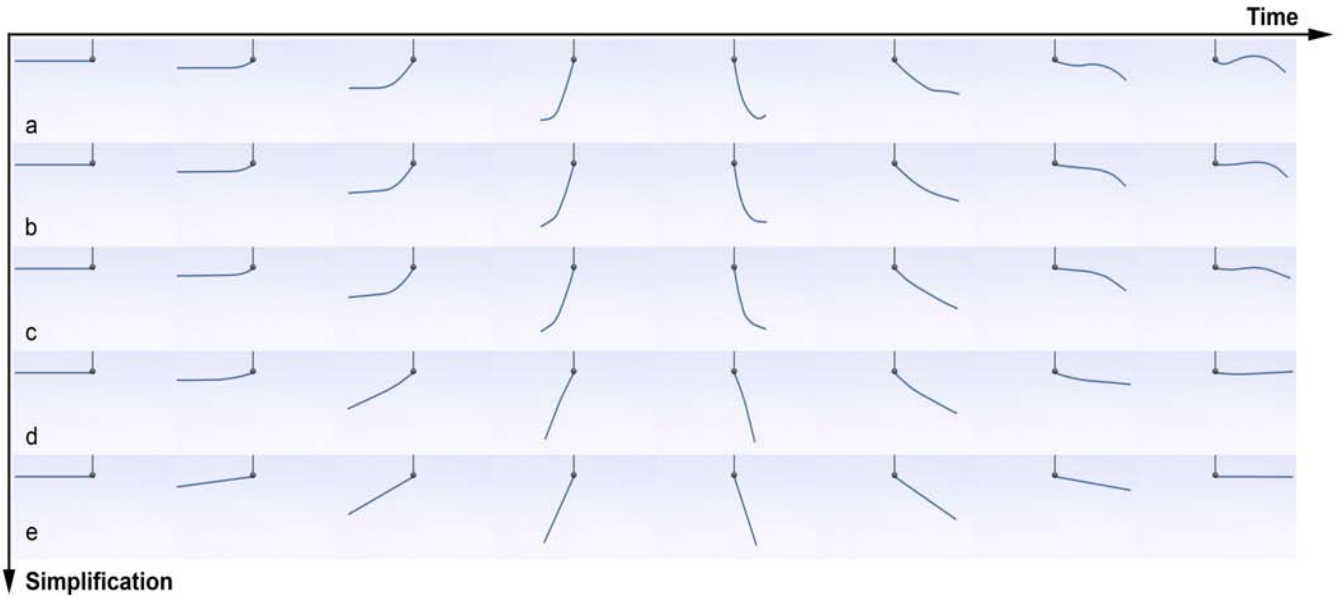

Figure 5: **Progressive simplification of the motion a pendulum (*cf* Section 7.1).**

improvements of up to two orders of magnitude in this case. The timings show the increase in efficiency when the number of active degrees of freedom and the number of applied forces decrease.

Note that a typical forward dynamics solver might be implemented using a faster algorithm than the DCA. For example, from the operation count reported in [Featherstone 1999b], the DCA is approximately three times slower than the standard articulated-body algorithm (ABA, [Featherstone 1987]) when run on a single processor. However, the ABA and other classical forward dynamics algorithms would also be linear in the number of joints in the articulated body, and thus would be outperformed by our adaptive dynamics algorithm as the simplification of the articulated body dynamics increases. Furthermore the DCA, on which our algorithm is based, is inherently parallel and rapidly outperforms the ABA when the number of processors increases. We believe that our adaptive algorithm might also be able to benefit from the use of multiple processors.

7.3 Limitations

Our algorithm has a few limitations:

Assembly tree dependency: the performance of our algorithm depends on the topology of the assembly tree. Even when only one node C could be activated to have a high-quality approximation of the articulated body motion, our method has to activate the nodes that are ascendant to C. This can reduce the quality of the approximated motion when the user allows very few active joints in the articulated body. Unless we know in advance where forces might be applied during the simulation, the best strategy is thus to build balanced assembly trees.
Potential damping: our algorithm is guaranteed not to increase the kinetic energy of the articulated body. However, the kinetic energy might be decreased when some nodes are rigidified (this clearly occurs when the single joint connecting a rigid body to a fixed base is rigidified).
Partial approximation guarantees: although some steps of our adaptive dynamics algorithm are provably error bounded (*e.g.* the determination of the acceleration update region and of the new active region), we currently do not have a proof that the other steps are error-bounded as well. Note that our algorithm can easily be turned into an approximate forward dynamics algorithm with *guaranteed* error bounds: we can ensure that the complete algorithm is error-bounded by removing the joint accelerations re-computation step and the rigidification step, and by updating the active region at every time step. However, this approach would be too conservative, as we explain in Section 5. Our adaptive dynamics algorithm provides the animator with complete control on the amount of motion simplification for articulated bodies.

7.4 Applications and Future Research

Next, we describe a few possible applications and future research directions enabled by our adaptive dynamics algorithm.

View-dependent articulated body dynamics: by appropriately customizing the weight matrices $\mathbf{A}_\circ$ and $\mathbf{V}_\circ$ in the motion metrics, it might be possible to develop a framework for rigorous view-dependent simplification of articulated body dynamics, which would determine the joints that have to be active based on the maximum amount of pixel error specified by the animator.
Perceptually-based simplification: some authors have studied human perception of animated characters [Reitsma and Pollard 2003]. It would be interesting to determine to what extent humans are sensitive to errors in general articulated body dynamics. It might then be possible to tune the motion metrics of our framework and take advantage of the results of these perceptual studies.
Adaptive collision detection and response: since our algorithm simplifies the dynamics of articulated bodies by rigidifying some subassemblies, it might be possible to speed up the collision detection and response as well by taking advantage of the temporal coherence in the acceleration structures.
Articulated body control simplification: in our examples, we only used very simple controllers (*e.g.* proportional-derivative controllers to help enforce joint limits in human characters). Since our algorithm handles active joint forces, and thus actuated joints, it might be possible to develop adaptive controllers that could be combined with our adaptive dynamics framework. For example, one could imagine that the motion of a rigidified character ankle is compensated by a modification of the character hip motion which, in turn, would force the character to "tip-toe" or walk on his heels, as humans can. Passive systems (*e.g.* hair, chains, and other articulated models) are also handled by our algorithm and are of interest as well. We envision a continuum of adaptive algorithms for the control and simulation of articulated figures, as well as hybrid algorithms using a combination of physically-based simulation, learning techniques, and data-driven modeling.

8 Conclusion

We have introduced a new approach for automatic simplification of the dynamics of an articulated body. Our algorithm allows the user to specify the number of degrees of freedom to simulate, and adaptively evolves the set of simulated joints over time to best approximate the underlying motion, based on novel customizable motion error metrics. Depending on the amount of simplification specified by an animator, this framework potentially offers significant speedups over typical linear-time forward dynamics algorithms.

We plan to perform a formal error analysis and examine ways to extend this approach to inverse dynamics. Besides the possible future research directions mentioned in Section 7.4, we would like to further investigate potential applications and extensions to other areas, such as interactive manipulation of molecular structures, rapid prototyping of complex mechanisms, "motion textures" for training environments, etc.

Acknowledgements

The authors wish to thank Dr. Roy Featherstone for his help with the DCA, James T. Pineda for proofreading the submission version of this paper, and the anonymous reviewers for their thorough and constructive comments on improving this paper. This project is supported in part by the Army Research Office, Intel Corporation, the National Science Foundation, and the Office of Naval Research.

References

ANDERSON, K. S., AND DUAN, S. 2000. Highly parallelizable low-order dynamics simulation algorithm for multi-rigid-body systems. *AIAA Journal on Guidance, Control, and Dynamics 23*, 2, 355–364.

ASCHER, U. M., PAI, D. K., AND CLOUTIER, B. P. 1997. Forward dynamics, elimination methods, and formulation stiffness in robot simulation. In *International Journal of Robotics Research, vol. 16, no. 6, pp. 749-758*.

BAE, D., AND HAUG, E. 1987. A recursive formulation for constrained mechanical systems dynamics: Part i. open-loop systems. *Mechanical Structures and Machines, Vol. 15, No. 3, pp. 359-382*.

BARAFF, D., AND WITKIN, A. 1997. Partitioned dynamics. Technical Report CMU-RI-TR-97-33, Robotics Institute, Carnegie Mellon University.

BARAFF, D. 1996. Linear-Time simulation using lagrange multipliers. In *SIGGRAPH 96 Conference Proceedings*, Addison Wesley, H. Rushmeier, Ed., Annual Conference Series, ACM SIGGRAPH, 137–146. held in New Orleans, Louisiana, 04-09 August 1996.

BARZEL, R., HUGHES, J., AND WOOD, D. 1996. Plausible motion simulation for computer graphics animation. *Proc. of Eurographics Workshop on Computer Animation and Simulation*, pp. 183–197.

BEAUDOIN, J., AND KEYSER, J. 2004. Simulation levels of detail for plant motion. *Proc. of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*.

BERTAILS, F., KIM, T.-Y., CANI, M.-P., AND NEUMANN, U. 2003. Adaptive wisp tree - a multiresolution control structure for simulating dynamic clustering in hair motion. *Proc. of ACM SIGGRAPH / Eurographics Symposium on Computer Animation*.

BRANDL, H., JOHANNI, R., AND OTTER, M. 1986. A very efficient algorithm for the simulation of robots and similar multibody systems without inversion of the mass matrix. *IFAC/IFIP/IMACS Symposium, pp. 95-100*.

BRUDERLIN, A., AND CALVERT, T. 1989. Goal-directed, dynamic animation of human walking. In *Computer Graphics (Proc. of SIGGRAPH'89)*, 233–242.

BRUDERLIN, A., AND CALVERT, T. 1996. Knowledge-driven, interactive animation of human running. In *Proc. of Graphics Interface*, 213–221.

CARLSON, D., AND HODGINS, J. 1997. Simulation levels of detail for real-time animation. In *Proc. of Graphics Interface 1997*.

CHENNEY, S., AND FORSYTH, D. 1997. View-dependent culling of dynamic systems in virtual environments. In *Proc. of ACM Symposium on Interactive 3D Graphics*.

CHENNEY, S., ICHNOWSKI, J., AND FORSYTH, D. 1999. Dynamics modeling and culling. *IEEE Computer Graphics and Applications March/April*, pp. 79–87.

CHENNEY, S., ARIKAN, ., AND FORSYTH, D. 2001. Proxy simulations for efficient dynamics. *Proc. of Eurographics 2001, Short Presentations*.

FAURE, F. 1999. Fast iterative refinement of articulated solid dynamics. *IEEE Trans. on Visualization and Computer Graphics 5*, 3, 268–276.

FEATHERSTONE, R., AND ORIN, D. E. 2000. Robot dynamics: Equations and algorithms. *IEEE Int. Conf. Robotics and Automation, pp. 826-834*.

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

FEATHERSTONE, R. 1999. A divide-and-conquer articulated body algorithm for parallel o(log(n)) calculation of rigid body dynamics. part 1: Basic algorithm. *International Journal of Robotics Research 18(9):867-875*.

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

FEATHERSTONE, R. 2004. An empirical study of the joint space inertia matrix. In *International Journal of Robotics Research, vol. 23, no. 9, pp. 859-871*.

FIJANY, A., SHARF, I., AND D'ELEUTERIO, G. 1995. Parallel o(log n) algorithms for computation of manipulator forward dynamics. *IEEE Trans. Robotics and Automation 11(3):389400*.

GILLESPIE, R. B., AND COLGATE, J. E. 1997. A survey of multibody dynamics for virtual environments. *Proc. of ASME Int. Mech. Engr. Conf. and Expo.*.

GIRARD, M., AND MACIEJEWSKI, A. 1985. Computational modeling for computer animation of legged figures. In *Computer Graphics (Proc. of SIGGRAPH)*, vol. 19, 263–270.

GRANIERI, J., CRABTREE, J., AND BADLER, N. 1995. Production and playback of human figure motion for 3d virtual environments. In *Proc. of VRAIS*, 127–135.

HOLLERBACH, J. 1980. A recursive lagrangian formulation of manipulator dynamics and a comparative study of dynamics formulation complexity. *IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-10, No. 11*.

KO, H., AND BADLER, N. 1993. Straight-line walking animation based on kinematic generalization that preserves the original characteristics. In *Proc. of Graphics Interfaces*.

KOKKEVIS, E., METAXAS, D., AND BADLER, N. 1996. User-controlled physics-based animation for articulated figures. *Proc. of Computer Animation*, 16–26.

MCMILLAN, S., AND ORIN, D. E. 1995. Efficient computation of articulated-body inertias using successive axial screws. *IEEE Trans. on Robotics and Automation, vol. 11, pp. 606-611*.

MUELLER, A., AND MAISSER, P. 2003. A lie-group formulation of kinematics and dynamics of constrained mbs and its application to analytical mechanics. *Multibody System Dynamics, vol. 9, no. 4, pp. 311-352(42)*.

MULTON, F., VALTON, B., JOUIN, B., AND COZOT, R. 1999. Motion levels of detail for real-time virtual worlds. *Proc. of ASTC-VR'99*.

O'BRIEN, D., FISHER, S., AND LIN, M. 2001. Simulation level of detail for automatic simplification of particle system dynamics. *Proc. of Computer Animation*, 210–219.

PERBET, F., AND CANI, M. 2001. Animating prairies in real-time. *Proc. of ACM Symposium on Interactive 3D graphics*.

PERLIN, K. 1995. Real time responsive animation with personality. *IEEE Transactions on Visualization and Computer Graphics 1*, 1, 5–15.

POPOVIC, Z., AND WITKIN, A. 1999. Physically based motion transformation. In *Proc. of SIGGRAPH 1999*, 11–20.

REDON, S., AND LIN, M. C. 2005. An efficient, error-bounded approximation algorithm for simulating quasi-statics of complex linkages. In *Proceedings of ACM Symposium on Solid and Physical Modeling*.

REITSMA, P. S. A., AND POLLARD, N. S. 2003. Perceptual metrics for character animation: Sensitivity to errors in ballistic motion. In *ACM Transactions on Graphics 22(3) 537-542, SIGGRAPH 2003 Proceedings*.

RODRIGUEZ, G., JAIN, A., AND KREUTZ-DELGADO, K. 1991. Spatial operator algebra for manipulator modelling and control. *Int. J. Robotics Research, vol. 10, no. 4, pp. 371-381*.

WARD, K., AND LIN, M. 2003. Proc. of pacific graphics. *Adaptive Grouping and Subdivision for Simulating Hair Dynamics*.

WARD, K., LIN, M., LEE, J., FISHER, S., AND MACRI, D., 2003. Modeling hair using level-of-detail representations. http://gamma.cs.unc.edu/HSLOD.

YAMANE, K., AND NAKAMURA, Y. 2002. Efficient parallel dynamics computation of human figures. *Proc. of IEEE Int. Conf. on Robotics and Automation*, pp. 530–537.

YAMANE, K., KUFFNER, J., AND HODGINS, J. 2004. Synthesizing animations of human manipulation tasks. *ACM Trans. on Graphics (Proc. SIGGRAPH 2004)*.