# Linear-Time Dynamics of Characters with Stiff Joints

Fernando Hernández[1] Carlos Garre[1] Rubén Casillas[2] Miguel A. Otaduy[1]

[1]URJC Madrid, Spain
[2]NextLimit Technologies, Spain

**ABSTRACT**

*Characters, like other articulated objects and structures, are typically simulated using articulated dynamics algorithms. There are efficient linear-time algorithms for the simulation of open-chain articulated bodies, but complexity grows notably under additional constraints such as joint limits, loops or contact, or if the bodies undergo stiff joint forces. This paper presents a linear-time algorithm for the simulation of open-chain articulated bodies with joint limits and stiff joint forces. This novel algorithm uses implicit integration to simulate stiff forces in a stable manner, and avoids drift by formulating joint constraints implicitly. One additional interesting feature of the algorithm is that its practical implementation entails only small modifications to a popular algorithm.*

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Virtual Reality—

## 1. Introduction

Articulated models, consisting of bodies connected by joints, are popular simulation models for human and non-human characters [KM98, KRFC09, Fea87], mechanical structures [WTF06], or even molecules [RGL05], to cite just a few examples. This popularity has attracted a strong attention in the computer graphics and robotics fields for the design of efficient computational models and algorithms.

The skeleton of characters, in particular, can be described as an open kinematic chain (i.e., with no loops, but with branches). There are multiple algorithms capable of solving articulated dynamics in time linear in the number of bodies and joints, e.g., using reduced-coordinate formulations [Fea87], Lagrange-multiplier formulations [Bar96], or handling very general joints [SG10]. A discussion of the advantages and disadvantages of the various models is given in Section 2. The complexity of simulating articulated dynamics grows notably, however, if the model cannot be described as an open kinematic chain. This may happen if the model incorporates other constraints, such as loops, joint limits, or contact, or if pairs of bodies interact through stiff forces, e.g., joint stiffness.

This work presents a linear-time algorithm for simulating dynamics of open-chain articulated bodies with joint limits and stiff joint forces, following a maximal-coordinate formulation based on Lagrange multipliers. To simulate stiff joint forces in a stable manner and with large time-steps, we propose the use of implicit integration algorithms with implicit joint constraints, as described in Section 3. Effectively, the use of implicit integration creates loops in the mathematical formulation of the problem, preventing the direct application of typical linear-time algorithms based on maximal coordinates.

**Figure 1:** *Interactive simulation of a hand with articulated dynamics and soft finger contact. The hand's skeleton follows the motion tracked with a Cyberglove device, and we use our novel algorithm to simulate its dynamics.*

To solve the implicit constrained problem, we have designed an efficient algorithm, described in Section 4, that builds on the factorization algorithm by Baraff [Bar96]. Notably, this novel algorithm implies small modifications to Baraff's, making it very easy to integrate in existing implementations of articulated body dynamics.

We demonstrate the application of our algorithm to the simulation of articulated models such as an interactive hand controlled using a glove device, or an articulated character. Nevertheless, the method is general and could be applied to other types of articulated models.

## 2. Related Work

An articulated body is a multibody system formed by a set of bodies (typically rigid, but not necessarily) connected by joint constraints. A coordinate-set that describes the (unconstrained) degrees of freedom of the bodies alone is typically referred to as *maximal coordinates*. Effectively, joints reduce the number of degrees of freedom of the bodies alone, and a coordinate-set that parameterizes the actual (constrained) degrees of freedom is typically referred to as *generalized coordinates*. There are two major approaches to solving articulated body dynamics. *Reduced-coordinate formulations* work directly with generalized coordinates. In this approach, the difficulty lies in finding a proper set of generalized coordinates. *Lagrange-multiplier formulations*, on the other hand, work with maximal coordinates, and formulate the dynamics problem as a constrained optimization augmented with new unknowns (i.e., the Lagrange multipliers, which correspond to joint forces).

For open kinematic chains, which is the type of articulated body we are interested in, finding a parameterization based on genealized coordinates is relatively easy, and well-known reduced-coordinate formulations exist [Fea87]. These methods are relatively simple under explicit integration, as they compute the accelerations of the generalized coordinates following a recursive procedure on the tree structure of the articulated body. Then, the accelerations are used to integrate velocities and positions. Under implicit integration, however, the accelerations depend implicitly on new coordinate values, and the typical recursive procedures are no longer valid.

It is worth noting that reduced-coordinate formulations have been extended to support modeling of constrains for character animation [KM98], parallel solutions [Fea99a, Fea99b], or adaptive simulation [RGL05]. But, most importantly, the work of Hadap [Had06] constitutes a reduced-coordinate formulation that is very close in spirit to our work. It supports both joint constraints and stiff joint forces, and it solves them all using implicit integration and with linear cost.

Baraff [Bar96], instead, advocates for Lagrange multiplier formulations over reduced-coordinate ones. Even though reduced-coordinate formulations may often be superior in terms of performance (after all, the effective system they solve is smaller), there are interesting arguments to develop Lagrange-multiplier formulations. As argued by Baraff, Lagrange-multiplier formulations are simpler to formulate in the general case, allow handling arbitrary constraints in a unified manner (including non-holonomic constraints such as friction, contact, or joint limits), and are far superior from the perspective of code modularity, as the implementation of the different bodies in the system is completely disjoint. We should point out, however, that recent work by Si and Guenter uses symbolic differentiation to handle general scleronomic joints in a reduced-coordinate formulation [SG10]. Adding to Baraff's discussion, we also argue that Lagrange-multiplier approaches are superior to reduced-coordinate ones under implicit integration, precisely because of modularity and the possibility to handle different types of bodies and constraints in a unified manner.

Baraff presented a linear-time algorithm to solve articulated body dynamics of open kinematic chains using the Lagrange-multipliers formulation, only for explicit integration. Previous approaches in computer graphics solved Lagrange-multipliers problems by formulating joint-space inertias (i.e., $\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T$), but the resulting matrix is generally not sparse for systems with branches. Baraff also designed *anticipation* algorithms to handle additional constraints such as contact, loops or joint limits, but then the solution is far more inefficient, as each additional constraint incurs a cost linear in the number of bodies and joints simply in a matrix-setup step. Weinstein et al. [WTF06] also propose a maximal-coordinate formulation for articulated bodies, handling joints and contact all in a unified manner. As opposed to Baraff's direct solver, they propose a Gauss-Seidel-type iterative algorithm.

In this work, we present an efficient linear-time algorithm that can handle joint limits and contacts, modeling them using stiff penalty forces instead of constraints. Hairer et al. [HLW02] present an interesting connection between stiff forces and constraints in the context of numerical integration. To ensure stable simulation under stiff forces, our algorithm is based on implicit integration, and we demonstrate that its implementation involves easy-to-implement modifications to Baraff's original algorithm. Implicit integration has been successfully used in computer graphics for the simulation of systems with stiff forces using large time steps [BW98]. It is often used in the simulation of unconstrained deformations [MG04], but there are also solutions that support deformation constraints [GHF*07] or contact constraints [OTSG09]. Choe et al. [CCK05] use a maximal coordinate formulation with stiff joint forces to model hair. They substitute all joints with soft constraints, and integrate the dynamics using implicit integration. Since their method is intended for hair, they do not handle branching, and then the resulting system is tri-diagonal and can be trivially solved with linear cost. Implicit integration has also been used for the simulation of articulated characters surrounded by a layer of deformable tissue [GOT*07], but the solution approximates the interaction between the various components.

Recently, computer graphics researchers have pointed out limitations of classic implicit integration algorithms such as backward Euler in terms of energy preservation. *Symplectic* integrators, instead, can provide both stability and energy preservation [KYT*06], but their computational cost is often higher, as they require full solves of the non-linear implicit equations, and the convergence of non-linear solvers such as Newton imposes restrictions on the time-step under highly non-linear situations.

## 3. Articulated Dynamics Model

In this section, we describe the formulation of articulated body dynamics using implicit integration. First, we recall the explicit approach based on acceleration constraints, and then we introduce our implicit approach based on position constraints. We conclude with the description of the general formulation for a multibody system, highlighting the differences between implicit and explicit approaches.

### 3.1. Acceleration Constraints

Let us start considering a simple articulated body consisting of two bodies $a$ and $b$ connected through a joint. The dynam-

ics of the two bodies can be described by the Newton-Euler equations

$$\mathbf{M}_a\dot{\mathbf{v}}_a + \mathbf{J}_a^T\lambda = \mathbf{F}_a(\mathbf{q}_a,\mathbf{v}_a,\mathbf{q}_b,\mathbf{v}_b) \quad \text{and} \quad (1)$$
$$\mathbf{M}_b\dot{\mathbf{v}}_b + \mathbf{J}_b^T\lambda = \mathbf{F}_b(\mathbf{q}_a,\mathbf{v}_a,\mathbf{q}_b,\mathbf{v}_b).$$

Here, $\mathbf{q}$ includes both the position and orientation of a rigid body, $\mathbf{v}$ includes its linear and angular velocity, $\mathbf{M}$ is a $6\times 6$ mass matrix, $\mathbf{J}$ is the Jacobian of the constraints, and $\mathbf{F}$ the vector of forces and torques. These may include gravitational and inertial forces, joint elasticity and damping.

Baraff differentiates joint constraints to formulate linear constraints on body accelerations. For the example above, his approach considers a constraint

$$\mathbf{C}(\mathbf{q}_a,\mathbf{q}_b) = 0 \;\rightarrow\; \ddot{\mathbf{C}} = \mathbf{J}_a\dot{\mathbf{v}}_a + \mathbf{J}_b\dot{\mathbf{v}}_b = 0. \quad (2)$$

Given Eq. (1) and Eq. (2) together, Baraff's approach computes accelerations that satisfy the constraints, and then integrates these accelerations to obtain the new velocities and positions. Unfortunately, acceleration constraints suffer from drift. Baraff addresses drift using stabilization forces, but this effectively means that the formulation does not satisfy hard constraints.

### 3.2. Implicit Position Constraints

We propose two main differences w.r.t. Baraff's formulation. First, we avoid drift by formulating implicit position constraints, i.e., constraints on the body positions at the end of the time step. Second, we handle stiff joint forces in a stable manner by integrating the dynamics equations with implicit methods. We model joint limits using stiff joint forces, which are handled robustly with our approach. Appendix A describes the derivative terms for joint forces.

With backward Euler implicit integration, the Newton-Euler equations with implicit position constraints can be rewritten as:

$$\mathbf{A}_a\mathbf{v}_a + \mathbf{A}_{ab}\mathbf{v}_b + \mathbf{J}_a^T\lambda = \mathbf{b}_a, \quad (3)$$
$$\mathbf{A}_{ba}\mathbf{v}_a + \mathbf{A}_b\mathbf{v}_b + \mathbf{J}_b^T\lambda = \mathbf{b}_b,$$
$$\mathbf{J}_a\mathbf{v}_a + \mathbf{J}_b\mathbf{v}_b = \mathbf{b}_\lambda,$$

with $\mathbf{A}_a = \mathbf{M}_a - \Delta t\frac{\partial \mathbf{F}_a}{\partial \mathbf{v}_a} - \Delta t^2\frac{\partial \mathbf{F}_a}{\partial \mathbf{q}_a}$, $\mathbf{b}_a = \mathbf{M}_a\mathbf{v}_{a,0} + \Delta t\mathbf{F}_a - \Delta t\frac{\partial \mathbf{F}_a}{\partial \mathbf{v}_a}\mathbf{v}_{a,0} - \Delta t\frac{\partial \mathbf{F}_a}{\partial \mathbf{v}_b}\mathbf{v}_{b,0}$, and similarly for the rest of the $\mathbf{A}$ and $\mathbf{b}$ terms. The discretized joint constraints are obtained by linearizing them at the beginning of the time step, with $\mathbf{J}_a = \frac{\partial \mathbf{C}}{\partial \mathbf{q}_a}$, $\mathbf{J}_b = \frac{\partial \mathbf{C}}{\partial \mathbf{q}_b}$, and $\mathbf{b}_\lambda = -\frac{1}{\Delta t}\mathbf{C}(\mathbf{q}_{a,0},\mathbf{q}_{b,0})$.

### 3.3. Formulation for a Multibody System

For an articulated body with an arbitrary number of bodies and joints, we group all body velocities in a vector $\mathbf{v}$, and then the implicit integration of the constrained Newton-Euler equations can be expressed as:

$$\mathbf{A}\mathbf{v} + \mathbf{J}^T\lambda = \mathbf{b}, \quad (4)$$
$$\mathbf{J}\mathbf{v} = \mathbf{b}_\lambda.$$

Following Baraff's approach, we construct a vector $\mathbf{x}$ of unknowns that groups all the body velocities and joint forces. But the ordering of the vector is not arbitrary, it must follow a *perfect elimination order* for the system at hand. For a sparse linear system of size $n$, the existence of a perfect elimination order implies that it can be factorized and solved with cost $O(n)$. A perfect elimination order exists for all sparse matrices whose incidence graph is acyclic [DER86], which is the case for articulated bodies connected as an open kinematic chain.

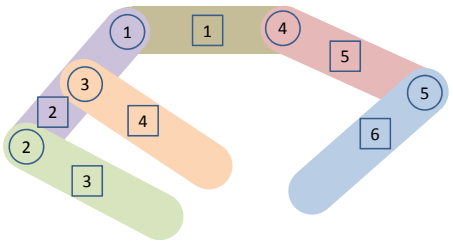


**Figure 2:** *Articulated body example with 6 bodies (squares) and 5 joints (circles).*

To establish the ordering of unknowns, we construct a tree with all bodies and joints as nodes. If one of the joints is a fixed joint, then this joint must be the root node, otherwise any body or joint may act as root. Then, the ordering of $\mathbf{x}$ is determined in a way such that a node's index must be greater than its children's indices. The ordering can be computed by performing a depth-first search or breadth-first search starting at the root node, with $n$ the index of the root node.

Given the appropriate reordering of the system, Eq. (4) can be written as $\mathbf{H}\mathbf{x} = \mathbf{z}$. For the example in Fig. 2 (reproduced from [Bar96]), with root joint 5, the matrix could be

$$\mathbf{H} = \begin{pmatrix}
\mathbf{A}_3 & \mathbf{J}_{23}^T & 0 & 0 & \mathbf{A}_{32} & 0 & 0 & 0 & 0 & 0 & 0 \\
\mathbf{J}_{23} & 0 & 0 & 0 & \mathbf{J}_{22} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \mathbf{A}_4 & \mathbf{J}_{34}^T & \mathbf{A}_{42} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \mathbf{J}_{34} & 0 & \mathbf{J}_{32} & 0 & 0 & 0 & 0 & 0 & 0 \\
\mathbf{A}_{23} & \mathbf{J}_{22}^T & \mathbf{A}_{24} & \mathbf{J}_{32}^T & \mathbf{A}_2 & \mathbf{J}_{12}^T & \mathbf{A}_{21} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \mathbf{J}_{12} & 0 & \mathbf{J}_{11} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \mathbf{A}_{12} & \mathbf{J}_{11}^T & \mathbf{A}_1 & \mathbf{J}_{41}^T & \mathbf{A}_{15} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \mathbf{J}_{41} & 0 & \mathbf{J}_{45} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \mathbf{A}_{51} & \mathbf{J}_{45}^T & \mathbf{A}_5 & \mathbf{A}_{56} & \mathbf{J}_{55}^T \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{A}_{65} & \mathbf{A}_6 & \mathbf{J}_{56}^T \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{J}_{55} & \mathbf{J}_{56} & 0
\end{pmatrix}. \quad (5)$$

In Baraff's formulation, matrix $\mathbf{H}$ contains non-zero blocks only at diagonal terms corresponding to bodies, and at off-diagonal terms corresponding to joint forces and joint constraint equations. In our formulation, matrix $\mathbf{H}$ also contains non-zero blocks for every pair of bodies that exert some force or torque on each other. Nevertheless, such terms are not arbitrary, and they correspond to joint forces (i.e., joint elasticity or joint limits) acting between adjacent bodies in the articulated body. Actually, adjacent bodies constitute grandchild-grandparent pairs in the tree. The next section describes an algorithm that exploits this property to factorize and solve the constrained dynamics equations with cost $O(n)$.

## 4. Linear Factorization Algorithm

Baraff proposed an algorithm for a factorization of the type $\mathbf{H} = \mathbf{L}\mathbf{D}\mathbf{L}^T$, with $\mathbf{D}$ a block-diagonal matrix and $\mathbf{L}$ a lower triangular matrix. Baraff's algorithm processes matrix $\mathbf{H}$ one row at a time. It exploits the property that, under explicit integration, for every row, at most one block to the right of the diagonal is non-zero. This block stores the matrix relating a node to its parent. In essence, Baraff's algorithm performs

the factorization by computing Schur complements on $2 \times 2$ blocks.

With implicit integration of joint forces, each row may contain up to two non-zero blocks to the right of the diagonal. For a row corresponding to a body, these two blocks represent the connection to the parent joint and to the grandparent body. Given two bodies $a$ and $b$ connected by a joint $j$, with $j$ the parent of $a$ (i.e., $j = \mathrm{p}(a)$), and $b$ the grandparent of $a$ (i.e., $b = \mathrm{g}(a)$), our factorization algorithm must handle matrix blocks of the form

$$\begin{pmatrix} \mathbf{A}_a & \mathbf{J}_a^T & \mathbf{A}_{ab} \\ \mathbf{J}_a & 0 & \mathbf{J}_b \\ \mathbf{A}_{ba} & \mathbf{J}_b^T & \mathbf{A}_b \end{pmatrix}. \tag{6}$$

We propose a factorization algorithm that employs parent and grandparent relationships to compute the solution to the problem $\mathbf{Hx} = \mathbf{z}$ in time linear in the number of bodies and joints. This algorithm involves small modifications to the original algorithm proposed by Baraff, hence it would be very easy to integrate in existing articulated-body implementations based on Baraff's algorithm.

The linear-system solver executes three loops over the nodes of the articulated body. First, a forward loop (i.e., in increasing index order, from leaves to root) computes the factoriztion itself. Then, another forward loop solves the problem $\mathbf{L}\mathbf{x}^{(1)} = \mathbf{z}$. Finally, a backward loop (i.e., in decreasing index order, from root to leaves) solves the problems $\mathbf{D}\mathbf{x}^{(2)} = \mathbf{x}^{(1)}$ and $\mathbf{L}^T\mathbf{x} = \mathbf{x}^{(2)}$. If a simulation requires the solution to multiple problems involving the same matrix $\mathbf{H}$, the factorization may be performed only once and reused in multiple solves.

The following algorithm describes our linear-time factorization for articulated bodies with implicit integration:

**procedure sparsefactor**
**for** $i = 1$ **to** $n$
  **for** $j \in \mathrm{child}(i)$
    $\mathbf{H}_{ii} = \mathbf{H}_{ii} - \mathbf{H}_{ji}^T \mathbf{H}_{jj} \mathbf{H}_{ji}$
    **for** $k \in \mathrm{child}(j)$
      $\mathbf{H}_{ii} = \mathbf{H}_{ii} - \mathbf{H}_{ki}^T \mathbf{H}_{kk} \mathbf{H}_{ki}$
    **if** $\mathrm{p}(i) \neq$ null
      $\mathbf{H}_{i,\mathrm{p}(i)} = \mathbf{H}_{i,\mathrm{p}(i)} - \mathbf{H}_{ji}^T \mathbf{H}_{jj} \mathbf{H}_{j,\mathrm{p}(i)}$
  **if** $\mathrm{p}(i) \neq$ null
    $\mathbf{H}_{i,\mathrm{p}(i)} = \mathbf{H}_{ii}^{-1} \mathbf{H}_{i,\mathrm{p}(i)}$
  **if** $\mathrm{g}(i) \neq$ null
    $\mathbf{H}_{i,\mathrm{g}(i)} = \mathbf{H}_{ii}^{-1} \mathbf{H}_{i,\mathrm{g}(i)}$

And the following algorithm describes the computation of the solution vector once the factorization has been performed:

**procedure sparsesolve**
**for** $i = 1$ **to** $n$
  $\mathbf{x}_i = \mathbf{z}_i$
  **for** $j \in \mathrm{child}(i)$
    $\mathbf{x}_i = \mathbf{x}_i - \mathbf{H}_{ij}^T \mathbf{x}_j$
    **for** $k \in \mathrm{child}(j)$
      $\mathbf{x}_i = \mathbf{x}_i - \mathbf{H}_{ik}^T \mathbf{x}_k$
**for** $i = n$ **to** $1$
  $\mathbf{x}_i = \mathbf{H}_{ii}^{-1} \mathbf{x}_i$
  **if** $\mathrm{p}(i) \neq$ null
    $\mathbf{x}_i = \mathbf{x}_i - \mathbf{H}_{i,\mathrm{p}(i)} \mathbf{x}_{\mathrm{p}(i)}$
  **if** $\mathrm{g}(i) \neq$ null
    $\mathbf{x}_i = \mathbf{x}_i - \mathbf{H}_{i,\mathrm{g}(i)} \mathbf{x}_{\mathrm{g}(i)}$

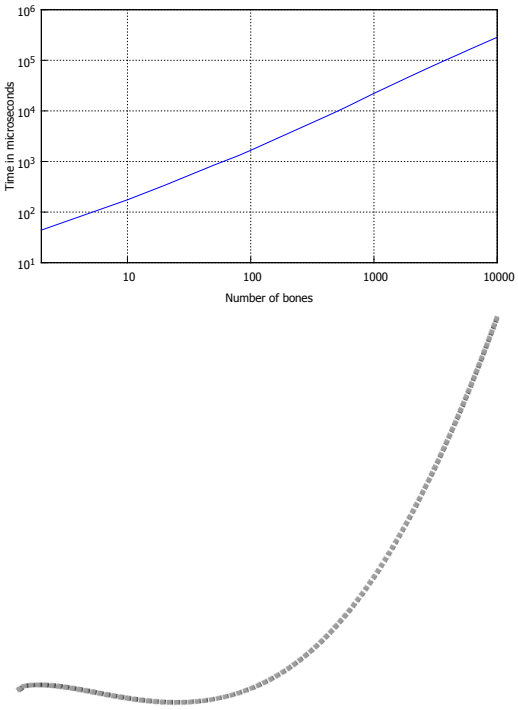


**Figure 4:** *Log-scale plot showing the running time per frame of the hanging articulated chain in the image, as a function of the number of bones.*

Both algorithms are written following the notation in the work of Baraff [Bar96]. We highlight in blue the small additions to the original algorithm, which clearly demonstrate the simplicity to achieve stable implicit integration. Note that the factorization stores the terms of $\mathbf{D}$ and $\mathbf{L}$ in place of the original matrix $\mathbf{H}$, hence the notation refers to blocks of $\mathbf{H}$ at all times. Similarly, when the algorithm terminates, the vector $\mathbf{x}$ contains the solution, i.e., the values of constraint forces $\lambda$ and body velocities $\mathbf{v}$.

## 5. Experiments and Results

We have implemented our algorithm on a quad-core 2.4 GHz PC with 3 GB of memory and a GeForce 8800 GTS. Our prototype implementation is not parallelized, hence we only make use of one of the four cores in the machine.

To verify the linear time complexity of our algorithm in practice, we have simulated articulated chains of various lengths. The plot in Fig. 4 indicates a cost of 22ms/frame for 1000 links, and 285ms/frame for 10000 links. The actual running time is just slightly superlinear due to faster memory access on smaller problems.

We have also applied our algorithm to several animation examples that illustrate its applicability. Fig. 3 shows some snapshots of a robot that is dragged from its left hand interactively using a Phantom Omni haptic device. Articulated dynamics resolve the configuration of the robot's arm as the user pulls the hand, and in the accompanying video we demonstrate the effects of turning joint torques on and off. We use spherical joints for the shoulder, the elbow and the wrist, and hinge joints for the fingers.

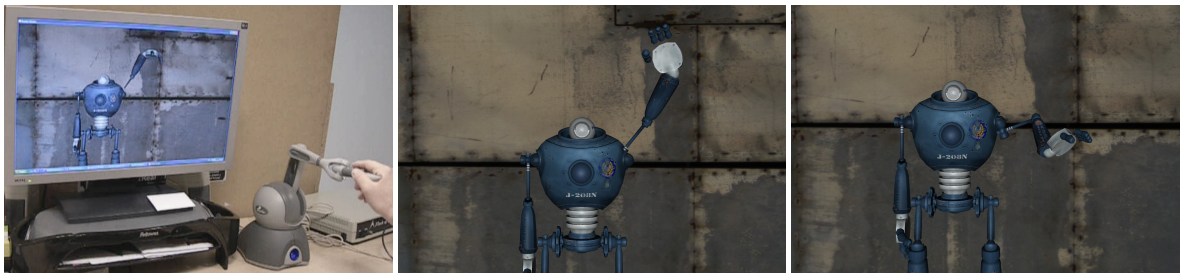

**Figure 3:** *Left: A robot is pulled from its left hand using a haptic device, and the user perceives the effect of joint stiffness. Middle and Right: Different configurations of the robot character.*

Last, Fig. 5 and Fig. 1 show an interactive simulation of a human hand. The skeleton of the hand consists of 16 bones and 15 joints, and it is connected through linear and torsional springs to a user's hand motion, tracked with a Cyberglove in real time. The skeleton model includes joint limits and joint stiffness, and it is connected to a deformable model that captures flesh deformation. The simulation of the full hand, with soft finger contact, runs at 21 fps on average, but the solution to articulated dynamics with our algorithm has a comparatively negligible cost of 0.36 ms/frame.

## 6. Discussion and Future Work

In this paper, we have described an algorithm for the simulation of open-chain articulated models with stiff forces. Our algorithm extends the maximal-coordinate formulation of Baraff [Bar96] to support implicit integration of joint forces and implicit joint constraints. Notably, this extension can be implemented in practice with small modifications to Baraff's original algorithm. There were already reduced-coordinate formulations to tackle articulated dynamics with stiff forces [Had06], but our maximal-coordinate solution is much simpler to formulate and it offers advantages in terms of code modularity and implementation complexity.

The results shown in this paper illustrate several possible applications of our algorithm, capturing efficiently elastic properties of flesh and joints. As observed by Kry and Pai [KP06], joint compliance is a key aspect of operations such as grasping and manipulation. However, the algorithm is applicable to arbitrary open-chain articulated bodies, or even deformable objects with a skeletal structure. Hadap's work [Had06] shows good example applications, which include hair, plants, trees, etc. For deformable objects, the surface is defined through smooth interpolation of bone transformations.

There are still multiple open problems in the simulation of articulated dynamics. Perhaps the most relevant one is the design of an algorithm for linear-time dynamics under arbitrary kinematic loops and contact constraints. Efficient coupling of skeletal dynamics to a surrounding deformable flesh is also a remaining hard problem. Unlike joint stiffness, which is localized and can be efficiently handled with our algorithm, a full deformable flesh removes the branching structure of the equations and is not amenable to efficient factorization algorithms.

## Acknowledgements

This project has been supported by the Spanish Ministry of Science and Innovation (projects TIN2009-07942 and PSE-300000-2009-5). We would also like to thank Jorge Gascón, Marcos Novalbos, Laura Raya, Javier S. Zurdo, and the rest of the GMRV group at URJC Madrid.

## References

[Bar96] BARAFF D.: Linear-time dynamics using lagrange multipliers. In *Proceedings of SIGGRAPH 96* (Aug. 1996), Computer Graphics Proceedings, Annual Conference Series, pp. 137–146.

[BW98] BARAFF D., WITKIN A. P.: Large steps in cloth simulation. In *Proceedings of SIGGRAPH 98* (July 1998), Computer Graphics Proceedings, Annual Conference Series, pp. 43–54.

[CCK05] CHOE B., CHOI M. G., KO H.-S.: Simulating complex hair with robust collision handling. In *2005 ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (July 2005), pp. 153–160.

[DER86] DUFF I. S., ERISMAN A. M., REID J. K.: *Direct Methods for Sparse Matrices*. Clarendon Press, 1986.

[Fea87] FEATHERSTONE R.: *Robot Dynamics Algorithms*. Kluwer Academic Publishers, 1987.

[Fea99a] FEATHERSTONE R.: A divide-and-conquer articulated body algorithm for parallel O(n) calculation of rigid body dynamics. Part 1: Basic algorithm. *International Journal of Robotics Research 18*, 9 (1999), 867–875.

[Fea99b] FEATHERSTONE R.: A divide-and-conquer articulated body algorithm for parallel O(n) calculation of rigid body dynamics. Part 2: Trees, loops and accuracy. *International Journal of Robotics Research 18*, 9 (1999), 876–892.

[GHF*07] GOLDENTHAL R., HARMON D., FATTAL R., BERCOVIER M., GRINSPUN E.: Efficient simulation of inextensible cloth. *ACM Transactions on Graphics 26*, 3 (July 2007), 49:1–49:7.

[GOT*07] GALOPPO N., OTADUY M. A., TEKIN S., GROSS M., LIN M. C.: Soft articulated characters with fast contact handling. *Computer Graphics Forum 26*, 3 (Sept. 2007), 243–253.

[Had06] HADAP S.: Oriented strands - dynamics of stiff multibody system. In *2006 ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (Sept. 2006), pp. 91–100.

[HLW02] HAIRER E., LUBICH C., WANNER G.: *Geometric Numerical Integration*. Springer Series in Computational Mathematics, vol. 31. Springer-Verlag, 2002.

[KM98] KOKKEVIS E., METAXAS D.: Efficient dynamic constraints for animating articulated figures. *Multibody System Dynamics 2* (1998).

[KP06] KRY P. G., PAI D. K.: Interaction capture and synthesis. *ACM Transactions on Graphics 25*, 3 (July 2006), 872–880.

[KRFC09] KRY P., REVERET L., FAURE F., CANI M.-P.: Modal locomotion: Animating virtual characters with natural vibrations. *Computer Graphics Forum 28*, 2 (Apr. 2009), 289–298.

**Figure 5:** *Left: Configuration of the bones and joints that forms the human hand model. Middle: Soft finger contact with a table. Right: Finger contact, showing target configuration traked with a Cyberglove (in yellow).*

[KYT*06] KHAREVYCH L., YANG W., TONG Y., KANSO E., MARSDEN J. E., SCHRÖDER P., DESBRUN M.: Geometric, variational integrators for computer animation. In *2006 ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (Sept. 2006), pp. 43–52.

[MG04] MÜLLER M., GROSS M. H.: Interactive virtual materials. In *Graphics Interface 2004* (May 2004), pp. 239–246.

[OTSG09] OTADUY M. A., TAMSTORF R., STEINEMANN D., GROSS M.: Implicit contact handling for deformable objects. *Computer Graphics Forum 28*, 2 (Apr. 2009), 559–568.

[RGL05] REDON S., GALOPPO N., LIN M. C.: Adaptive dynamics of articulated bodies. *ACM Transactions on Graphics 24*, 3 (Aug. 2005), 936–945.

[SG10] SI W., GUENTER B.: Linear-time dynamics for multibody systems with general joint models. *Proc. of ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (2010).

[WTF06] WEINSTEIN R., TERAN J., FEDKIW R.: Dynamic simulation of articulated rigid bodies with contact and collision. *IEEE Transactions on Visualization and Computer Graphics 12*, 3 (May/June 2006), 365–374.

## Appendix A: Derivatives of Joint Torques

Let us consider two bodies $a$ and $b$ with orientations $\mathbf{R}_a$ and $\mathbf{R}_b$. Their relative rotation is $\mathbf{R} = \mathbf{R}_a\mathbf{R}_b^T$, which can be defined as a rotation $\phi$ around an axis $\mathbf{u}$.

We consider joint torques of the form $\mathbf{T}_a = -k \cdot (\phi\mathbf{u})$, which resist the relative rotation. To implement implicit integration with such joint torques, we need to compute derivatives of the form $\frac{\partial \mathbf{T}_a}{\partial \theta_a} = -k \cdot \frac{\partial \phi\mathbf{u}}{\partial \theta_a}$, etc., with $\theta_a$ the linearized rotation angles of body $a$.

The derivative of the angle-axis of rotation can be expressed as:

$$\frac{\partial \phi\mathbf{u}}{\partial \theta_a} = \frac{\phi}{2\sin\phi}(\mathbf{R}^T + \mathbf{I}). \tag{7}$$

For small rotation angles, the derivative can be computed as:

$$\lim_{\phi \to 0} \frac{\partial \phi\mathbf{u}}{\partial \theta_a} = \mathbf{I}. \tag{8}$$

**Proof:**

Let us consider a differential rotation of body $a$, $d\theta_a$. Then, the new relative rotation is $(\mathbf{I} + d\theta_a^*)\mathbf{R}$, where $d\theta_a^*$ is a skew-symmetric matrix that represents a cross product with vector $d\theta_a$. After this differential rotation, we can also express a differential change in the angle-axis of rotation, which becomes $\phi\mathbf{u} + d(\phi\mathbf{u})$.

By definition of angle-axis, the rotation of a vector in the direction of the rotation axis leaves the vector intact. Then, we can express the following identity:

$$(\mathbf{I} + d\theta_a^*)\mathbf{R}(\phi\mathbf{u} + d(\phi\mathbf{u})) \equiv \phi\mathbf{u} + d(\phi\mathbf{u}). \tag{9}$$

Operating, and applying the property of the rotation axis to $\mathbf{R}\phi\mathbf{u} \equiv \phi\mathbf{u}$, we get:

$$(\mathbf{R} - \mathbf{I})d(\phi\mathbf{u}) = \phi\mathbf{u}^* d\theta_a. \tag{10}$$

Now, we premultiply both sides by $(\mathbf{R}^T + \mathbf{I})$ to obtain:

$$(\mathbf{R} - \mathbf{R}^T)d(\phi\mathbf{u}) = (\mathbf{R}^T + \mathbf{I})\phi\mathbf{u}^* d\theta_a = \phi\mathbf{u}^*(\mathbf{R}^T + \mathbf{I})d\theta_a. \tag{11}$$

Next, we apply the matrix form of Rodrigues' rotation formula, $\mathbf{R} = \cos\phi\mathbf{I} + \sin\phi\mathbf{u}^* + (1 - \cos\phi)\mathbf{u}\mathbf{u}^T$:

$$2\sin\phi\mathbf{u}^* d(\phi\mathbf{u}) = \phi\mathbf{u}^*(\mathbf{R}^T + \mathbf{I})d\theta_a \tag{12}$$

Last, eliminating $\mathbf{u}^*$ from both sides of the equation leads to the derivative

$$\frac{\partial \phi\mathbf{u}}{\partial \theta_a} = \frac{\phi}{2\sin\phi}(\mathbf{R}^T + \mathbf{I}). \tag{13}$$