Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2006)
M.-P. Cani, J. O'Brien (Editors)

Oriented Strands - dynamics of stiff multi-body system

Sunil Hadap

PDI/DreamWorks

Abstract

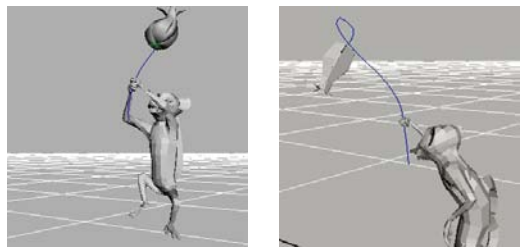
The simulation of strand like primitives modeled as dynamics of serial branched multi-body chain, albeit a potential reduced coordinate formulation, gives rise to stiff and highly non-linear differential equations. We introduce a recursive, linear time and fully implicit method to solve the stiff dynamical problem arising from such a multi-body system. We augment the merits of the proposed scheme by means of analytical constraints and an elaborate collision response model. We finally discuss a versatile simulation system based on the strand primitive for character dynamics and visual effects. We demonstrate dynamics of ears, braid, long/curly hair and foliage.

1. Introduction and Related Work

The simulation of ears, tails, braids, long wavy/curly hair, foliage, jewelry is peculiar in nature. The flexible shape is characterized by a thin and long geometry, which typically has a non-straight rest configuration. The dynamics predominantly includes the bend and the torsional components, and very rarely the length-wise stretch component. Even though being one-dimensional in nature, the intricate rendering aspects of these primitives, along with potentially highly anisotropic physical properties, demand a consistent/stable curvilinear coordinate system all along the geometry. In this paper we would like to present a versatile dynamic primitive that spans the stated characteristics and applications. We name the system as Oriented Strands, to clearly convey the picularity to the user.

Cosserat Models discussed in [Rub00] and first introduced to computer graphics community by [Pai02] give an elaborate continuum theory behind the dynamics of thin deformable objects such as strand and shells. The discrete approximation of the strand model come strikingly close to the strand-as-serial-multi-body-chain model, first proposed by [HMT01, Had03]. Since then the paradigm is successfully used for hair simulation by [CJY02,CCK05]. We too model strand as serial chain of rigid segments connected by spherical joints. Previously, the hair was typically modeled using mass-spring-hinge system, as individual hair strands [RCT91, AUK92] or as wisps [PCP01]. However, these models are not effective in representing consistent coordinates and the twist dynamics. An exhaustive overview of various hair simulation techniques is given in [Had03].

Figure 1: *Lanterns - consistent coordinates, twist dynamics*



[Fea87] developed one of the first multi-body reduced coordinate formulations that has a linear complexity. [Mir96, Kok04] further developed efficient and comprehensive impulse and constraint formulations to it. [RGL05] extended the formulation to achieve interesting sub-linear complexity, and also gives a thorough overview of the other reduced coordinate formulations. [Bar96, Sha01] gives maximal coordinate formulations which also are known to have linear complexity using sparse-matrix solution methods. The typical multi-body system resulting from the strand model, gives rise to "stiff" and highly non-linear differential equations. The numerical difficulties stem from small rotational inertia along the axis due to thin geometry, large bend and torsional stiffness-to-mass-ratio and intricate non-

straight rest shape. The non-linearity is due to velocity terms corresponding to Coriolis forces and the specific choice of the non-linear elastic model in our implementation to limit unrealistic high deformations. These difficulties call for an implicit integration scheme. Even though the reduced coordinate formulation is efficient for multi-body systems with large number of segments and relatively small number of DOFs, it is difficult to realize an implicit integration scheme, as pointed out by [Had03]. Instead, [CCK05] use a traditional maximal coordinate formulation with (soft) constraints [Bar96, Sha01], followed by an implicit integration. Complex collision response models with static and dynamic friction can be integrated into the maximal coordinate framework, with relative ease, using impact and contact velocity constraints [CCK05].

Nevertheless, the reduced coordinate formulation has certain advantages. The generalized coordinates directly facilitate the parameterization for bending and torsional stiffness dynamics. Further, they have the exact same form as the parameterization used in articulated character animation. Thus the rest shape of the multi-body system can be conveniently defined in terms of some animated local, e.g. a hairstyle can be defined in terms of the frame associated with the head joint. Even the dynamics is often expressed in terms of successive local coordinates starting from the base link. One can thus interpret the dynamic motion of the strand as overall rigid-body transformation of the strand at the base link, followed by secondary dynamics from the rest shape expressed in terms of successive local frames. This paradigm gives a tremendous advantage in terms of overall simulation workflow. Typically the base link is constrained to an animation path. Using the paradigm, it is trivial to kick-start the simulation from an arbitrary starting position and orientation of the base link. Moreover, certain concepts such as posable dynamics, local dynamics, time-scale and zero-gravity rest shape make the strand simulation system versatile. As discussed subsequently, they are often trivial to realize in the paradigm of reduced coordinate formulation. Ultimately, the choice of reduced coordinate formulation proved very rewarding for us.

The specific contributions of the paper are as follows. In Section 2 and Section 3 we develop a linear time, implicit and fully recursive scheme for reduced coordinate formulation of general branched open-chain multi-body system, using Differential Algebraic Equations (DAE). In Section 4, we discuss how to realize external bilateral and unilateral constraints on the formulated multi-body dynamics. We also discuss the numerical issues associated with the solution of Linear Complementarity Problem (LCP) arising from the formulation. In Section 5, we develop an elaborate collision response model with inelastic impact and static/dynamic friction, using unilateral constraints. Finally, in Section 6, we introduce the Oriented Strands system, implemented as dynamics of serial multi-body chain. We develop some novel concepts and give important implementation details that makes the dynamic strand primitive versatile. In Section 7, we present some illustrative examples of dynamics of ears, braid, hair and foliage.

2. Differential Algebraic Equations

Typically, unconstrained dynamical problems such as cloth [BW98] and general deformable models are formulated as the following *explicit* Ordinary Differential Equation (ODE) of degree two.

$$\ddot{\mathbf{q}} = \mathbf{M}^{-1}\mathbf{Q}(t, \mathbf{q}, \dot{\mathbf{q}}) \tag{1}$$

Constrained dynamical problems such as dynamics of multi-body systems [Sha01,Bar96] are formulated as the following *semi-explicit* ODE of degree two.

$$\begin{aligned} \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} &= \mathbf{Q}(t, \mathbf{q}, \dot{\mathbf{q}}) - \Phi_{\mathbf{q}}{}^{T}\lambda \\ \Phi(t, \mathbf{q}) &= 0 \end{aligned} \tag{2}$$

where, $\mathbf{M}$ is generalized mass matrix. The force function $\mathbf{Q}$ and the constraint function Φ are typically non-linear and "stiff". In order to integrate the state vector $[\dot{\mathbf{q}}^T, \mathbf{q}^T]_t^T$, in a traditional way, one can try and solve for the derivatives of the state vector $[\ddot{\mathbf{q}}^T, \dot{\mathbf{q}}^T]_{t+1}^T$, which often turns out to be complex. Fortunately, the direct computation of derivatives is not the only way, neither it is the most efficient way, of solving the differential equations. Differential Algebraic Equations solvers are remarkable, they advance the solution $[\dot{\mathbf{q}}^T, \mathbf{q}^T]_t^T \rightarrow [\dot{\mathbf{q}}^T, \mathbf{q}^T]_{t+1}^T$, as they estimate the derivatives $[\ddot{\mathbf{q}}^T, \dot{\mathbf{q}}^T]_{t+1}^T$ at the same time.

As far as we can track, Differential Algebraic Equations (DAE) are new to computer graphics. In this section we would like to give a gentle introduction to DAE. For thorough discussion and associated theory we would like to refer to [BCP96]. DAE solvers work on the system of differential equations in its most implicit form. Consider the following DAE of degree one.

$$\mathbf{F}(\mathbf{y}, \dot{\mathbf{y}}, t) = \mathbf{0} \tag{3}$$

The implicit function $\mathbf{F}$ in differential variables $\mathbf{y}$ and free variable t may be non-linear and "stiff". Let the set $\{\mathbf{y}, \dot{\mathbf{y}}\}_t$ be the solution of the DAE, i.e. it satisfies equation 3 at time t. Then the DAE solvers use an extrapolation method, e.g. Backward Difference Formula (BDF) of an appropriate order, to extrapolate the solution to $\mathbf{y}_{t+1}^1$ and while making a numerical estimate of the derivative $\dot{\mathbf{y}}_{t+1}^1$. The estimates $\mathbf{y}_{t+1}^1, \dot{\mathbf{y}}_{t+1}^1$ typically would not satisfy equation 3. The solver then successively corrects the solution and associated derivative estimate by number of Newton-Raphson iterations. Let the residue associated with the estimate of k^{th} iteration be

$$\mathbf{rs}_{t+1}^k = \mathbf{F}(\mathbf{y}_{t+1}^k, \dot{\mathbf{y}}_{t+1}^k, t+1) \tag{4}$$

The Newton-Rapson iteration takes the following form

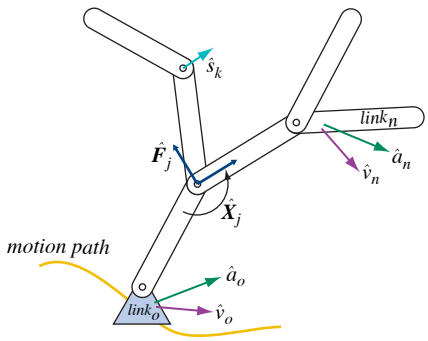


Figure 2: *Strand as multi-body system*

$$\begin{aligned} \mathbf{y}_{t+1}^{k+1} &= \mathbf{y}_{t+1}^{k} - \frac{\partial \mathbf{F}}{\partial \mathbf{y}}^{-1} \mathbf{rs}_{t+1}^{k} \\ \dot{\mathbf{y}}_{t+1}^{k+1} &= (\mathbf{y}_{t+1}^{k+1} - \mathbf{y}_t)/\triangle t \end{aligned} \tag{5}$$

Thus, in order to use DAE integrator such as DASPK [BCP96], one has to provide the integrator with a residual function **rs** that computes the residue from the estimates of the solution the integrator provides. One may also provide the Jacobian of the residue function $\partial\mathbf{F}/\partial\mathbf{y}$, or optionally the integrator can compute the Jacobian numerically. The highlight of the solution method is – the residue function **rs** and the Jacobian $\partial\mathbf{F}/\partial\mathbf{y}$ are often simple to evaluate. In the next section, we formulate a fully recursive method to evaluate the residual function for solution of a "sitff" multi-body system.

3. Recursive Residual Formulation

To describe the dynamics of the multi-body system, we use Spatial Algebra and associated notation developed by [Fea87]. Consider a serial branched multi-body system (MBS) having n links connected by n single DOF joints as shown in Figure 2. There are no loops in the kinematic chain.

The base link is denoted by $link_0$ and is typically constrained to a motion path. The other links are numbered in a breadth first manner. The velocity $\hat{\mathbf{v}}_j$, the acceleration $\hat{\mathbf{a}}_j$ and the inertia matrix $\hat{\mathbf{I}}_j$ of $link_j$ are defined in the frame $\hat{\mathbf{F}}_j$, which is rigidly attached to the link's principal inertia axis. The joint variable q_j defines the spatial transformation $\hat{\mathbf{X}}_j$ that transforms the spatial quantities defined in the parent's frame $\hat{\mathbf{F}}_i$ to the frame $\hat{\mathbf{F}}_j$ of $link_j$. The derivatives of joint variables $\dot{q}_j$ and $\ddot{q}_j$ relate the velocity and acceleration of the parent to the velocity and acceleration of $link_i$ via the spatial joint axis $\hat{\mathbf{s}}_i$.

$$\begin{aligned} \hat{\mathbf{v}}_j &= \hat{\mathbf{X}}_j\, \hat{\mathbf{v}}_i + \hat{\mathbf{s}}_j\, \dot{q}_j \\ \hat{\mathbf{a}}_j &= \hat{\mathbf{X}}_j\, \hat{\mathbf{a}}_i + \hat{\mathbf{s}}_j\, \ddot{q}_j + \hat{\mathbf{v}}_j \hat{\times} \hat{\mathbf{s}}_j\, \dot{q}_j \end{aligned} \tag{6}$$

The set of joint variables $\mathbf{q}_t$ and their derivative $\dot{\mathbf{q}}_t$ forms the system state vector $\mathbf{y}_t = [\dot{\mathbf{q}}^T, \mathbf{q}^T]_t^T$, at time t. We would like to solve the forward dynamical problem – given the base velocity $\hat{\mathbf{v}}_0$ and the base acceleration $\hat{\mathbf{a}}_0$, integrate the state from $\mathbf{y}_t$ to $\mathbf{y}_{t+1}$. In what follows we will develop a recursive residual formulation based on DAE methodology discussed in the previous section. The discussion is rather a free physical interpretation, for the rigorous proof refer to [RJFdJ04]. The procedure is surprisingly simple as compared to the traditional methods such as Articulated Body Method [Fea87], where one computes the state derivative $\dot{\mathbf{y}}_{t+1}$ explicitly.

The solution set $\{\mathbf{y}_t, \dot{\mathbf{y}}_t\}$ at time t forms the input to the DAE integrator. As highlighted in the previous section, the integrator then estimates the new state $\mathbf{y}_{t+1}^k$, and it's derivative $\dot{\mathbf{y}}_{t+1}^k$ in succession. It is our responsibility to compute the associated residue $\mathbf{rs}_{t+1}(\mathbf{y}^k, \dot{\mathbf{y}}^k)$. Given $\hat{\mathbf{v}}_0$ and $\hat{\mathbf{a}}_0$, we first compute the spatial velocities $\hat{\mathbf{v}}_j$ and spatial accelerations $\hat{\mathbf{a}}_j$ for each link $link_j$, $j = 1..n$ using forward kinematic relation, equation 6.

The residue associated with accelerations can be computed using the force balance condition. Starting with the outer most link $link_n$, the forces acting on $link_n$ are spatial body force $\hat{\mathbf{I}}_n\, \hat{\mathbf{a}}_n$, combined spatial centripetal and Coriolis force $\hat{\mathbf{v}}_n \hat{\times}\, \hat{\mathbf{I}}_n\, \hat{\mathbf{v}}_n$, external spatial force $\hat{\mathbf{f}}_n$. The force balance equation for the spatial forces is

$$\hat{\mathbf{rs}}_n \;=\; \hat{\mathbf{I}}_n\, \hat{\mathbf{a}}_n \;+\; \hat{\mathbf{v}}_n \hat{\times}\, \hat{\mathbf{I}}_n\, \hat{\mathbf{v}}_n \;-\; \hat{\mathbf{f}}_n \tag{7}$$

We still have to relate the force residue $\hat{\mathbf{rs}}_n$ which is a spatial vector to the residue in joint acceleration which is a scalar. We project the force residue on to the joint's motion subspace defined by the spatial joint axis $\hat{\mathbf{s}}_n$.

$$rs_n \;=\; \hat{\mathbf{s}}_n^S\, \hat{\mathbf{rs}}_n - Q_n \tag{8}$$

where, $\hat{\mathbf{s}}_n^S$ is spatial transpose of the joint axis and Q_n is scalar joint actuation force associated with the stiffness model. The force residue projected on the joint's motion space rs_n vanishes when the estimated state and derivative vector is the solution of DAE.

For simplicity, first consider a multi-body system with no branches. Thus, the only parent of $link_n$ would be $link_{n-1}$. In computing the force balance equation for $link_{n-1}$, along with all the obvious forces described for $link_n$, we need to add the residual force from $link_n$ that gets applied via the $joint_n$. In order to do that, we need to transform the force residue $\hat{\mathbf{rs}}_n$ into the frame of $link_{n-1}$, using inverse transformation matrix $\hat{\mathbf{X}}_n^{-1}$. The resulting force balance equation for $link_{n-1}$ is

$$\begin{aligned} \hat{\mathbf{rs}}_{n-1} &= \hat{\mathbf{I}}_{n-1}\, \hat{\mathbf{a}}_{n-1} \;+\; \hat{\mathbf{v}}_{n-1} \hat{\times}\, \hat{\mathbf{I}}_{n-1}\, \hat{\mathbf{v}}_{n-1} \;-\; \hat{\mathbf{f}}_{n-1} \\ &\quad + \hat{\mathbf{X}}_n^{-1}\, \hat{\mathbf{rs}}_n \\ rs_{n-1} &= \hat{\mathbf{s}}_{n-1}^s \hat{\mathbf{rs}}_{n-1} - Q_{n-1} \end{aligned} \tag{9}$$

We repeat this process for each $link_i$ till we reach the first link $link_1$. The residue associated with the joint velocities is trivially the difference in joint velocities $\dot{q}_i^{k*} - \dot{q}_i^k$, where $\dot{q}_i^k$ belongs to $\mathbf{y}^k$ and $\dot{q}_i^{k*}$ belongs to $\dot{\mathbf{y}}^k$.

Algorithm 1 lists the fully-recursive algorithm for computing the residue, for a general multi-body system that have branches.

It is possible to compute the analytic Jacobians for the recursive residue formulation [RJFdJ04]. Alternatively, we can let the DAE solver compute the Jacobians numerically. We particularly commend the efficient and "smart" evaluations of Jacobians in DASPK, the DAE solver we used for the implementation. The solver uses modified Newton-Rapson iteration, where the Jacobians are evaluated only when the solver observes a large change in the system's state. In practice, we found that the numerical evaluation of Jacobians is not only adequate, but also versatile. Thus, we can implement any complex stiffness model and associate general external fields to the multi-body system, as discussed in Section 6. It may not be possible to evaluate analytic Jacobians for these.

Algorithm 1 $res_{t+1}(\mathbf{y}^k, \dot{\mathbf{y}}^k, \hat{\mathbf{v}}_0, \hat{\mathbf{a}}_0)$

Require: $\mathbf{y}^k = \begin{bmatrix} \mathbf{q}^k \\ \dot{\mathbf{q}}^k \end{bmatrix}$, $\dot{\mathbf{y}}^k = \begin{bmatrix} \dot{\mathbf{q}}^{k*} \\ \ddot{\mathbf{q}}^k \end{bmatrix}$

$n \leftarrow dim(\mathbf{q})$
for $j = 1$ to n **do**
  $i \leftarrow parent(link_j)$
  $\hat{\mathbf{v}}_j \leftarrow \hat{\mathbf{X}}_j \, \hat{\mathbf{v}}_i + \hat{\mathbf{s}}_j \, \dot{q}_j$
  $\hat{\mathbf{a}}_j \leftarrow \hat{\mathbf{X}}_j \, \hat{\mathbf{a}}_i + \hat{\mathbf{s}}_j \, \ddot{q}_j + \hat{\mathbf{v}}_j \hat{\times} \hat{\mathbf{s}}_j \, \dot{q}_j$
end for

$\hat{\mathbf{rs}} \leftarrow \hat{\mathbf{0}} \in \Re^{6n}$
$\mathbf{rs} \leftarrow \mathbf{0} \in \Re^n$
for $i = n$ to 1 **do**
  $\hat{\mathbf{rs}}_i \leftarrow \hat{\mathbf{I}}_i \, \hat{\mathbf{a}}_i + \hat{\mathbf{v}}_i \hat{\times} \hat{\mathbf{I}}_i \, \hat{\mathbf{v}}_i - \hat{\mathbf{f}}_i$
  for all $j \leftarrow child(link_i)$ **do**
    $\hat{\mathbf{rs}}_i \leftarrow \hat{\mathbf{rs}}_i + \hat{\mathbf{X}}_j^{-1} \, \hat{\mathbf{rs}}_j$
  end for
  $rs_i \leftarrow \hat{\mathbf{s}}_i^s \hat{\mathbf{rs}}_i - Q_i$
end for

return $\begin{bmatrix} \dot{\mathbf{q}}^{k*} - \dot{\mathbf{q}}^k \\ \mathbf{rs} \end{bmatrix}$

4. Analytic Constraints

The base joint in the multi-body system in Figure 2 is typically constrained to some prescribed motion path. In practice, we would want to impose some additional constraints on the system, e.g. constraining a point on some other link to a motion path, or constraining a joint to an animated value in time. These constraints are transient in nature and often introduce cyclic dependancy in the system.

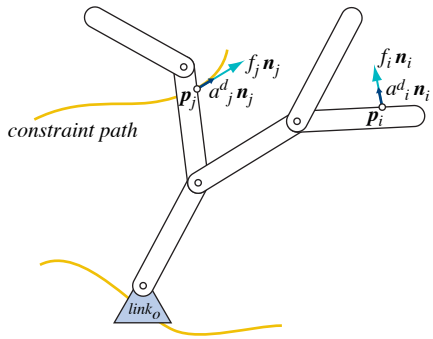


Figure 3: *Constraints*

Thus they are treated as external constraints, as opposed to defining them implicitly as part of reduced coordinate formulation.

Initially, we enthusiastically tried the DAE's natural way of defining constraints via algebraic slack variables. The general form of a DAE with algebraic constraints is

$$\mathbf{F}(\mathbf{y}, \dot{\mathbf{y}}, \mathbf{x}, t) = \mathbf{0} \tag{10}$$

where $\mathbf{x}$ is the set of algebraic variables. For each constraint, we formulated a scalar valued constraint function $\phi_i(\mathbf{y}, \dot{\mathbf{y}}, t)$ and inserted an algebraic variable associated with the residue corresponding to the constraint condition $x \equiv \phi_i(\mathbf{y}, \dot{\mathbf{y}}, t) = 0$ into the DAE. However, we soon abandoned this line of thinking for the following reasons

- The constraints are transient in nature. We either have to adjust the dimension of algebraic variables $\mathbf{x}$ according to the number of active constraints, or represent all the possible constraints and activate or deactivate them algebraically.
- We found the DAE solver's convergence rate deteriorates rapidly with each additional algebraic constraint. Further, if the constraint can not be satisfied, the integrator does not converge.
- The algebraic constraints can only represent bilateral constraints. The constraints arising from collisions are unilateral. We would have to extend our scope to Differential Variational Inequalities [PS03], which are extension of DAE that handle inequality constraints on differential and algebraic variables.

Instead, we augment the DAE based multi-body formulation inspired by methodologies proposed by [Mir96] and recently by [Kok04] on impulse dynamics and analytical constraints. We would like to give a brief overview of the methodology, along with the details on how we integrate it with the DAE framework and some interesting implementation issues.

Consider a point constraint $\mathbf{p}_j$ as depicted in Figure 3. The

trajectory of $\mathbf{p}_j$, starting with the initial conditions, can be uniquely defined by the time varying acceleration $a_j^d \mathbf{n}_j$. As discussed in the previous section, we do not directly evaluate the state derivative vectors $\dot{\mathbf{y}}_{t+1}$ in order to integrate the system $\mathbf{y}_t \rightarrow \mathbf{y}_{t+1}$. Therefore, we can not simply enforce the acceleration constraint, by directly altering the state derivatives $\dot{\mathbf{y}}_{t+1}$ as proposed by Kokkevis. We enforce the constraint by applying an external force instead. However, we don't use a penalty like force formulation. Before every DAE integration step, we analytically determine the precise nature of the force $f_j \mathbf{n}_j$, using the similar methodology as in [Kok04]. The unit constraint direction $\mathbf{n}_j$ is treated as constant and is updated for every integration step. There is a linear relationship between the magnitude of the applied force f_j and the resulting desired acceleration a_j^d

$$a_j^d = \frac{\partial a_j}{\partial f_j} f_j + a_j^0 \tag{11}$$

where, a_j^0 is the acceleration in the direction $\mathbf{n}_j$ before the force is applied. If we have another constraint point $\mathbf{p}_i$ with force having magnitude f_i in the direction $\mathbf{n}_i$, the resulting accelerations a_i^d and a_j^d will be given by the following linear system

$$\begin{bmatrix} a_i^d \\ a_j^d \end{bmatrix} = \begin{bmatrix} \partial a_i/\partial f_i & \partial a_i/\partial f_j \\ \partial a_j/\partial f_i & \partial a_j/\partial f_j \end{bmatrix} \begin{bmatrix} f_i \\ f_j \end{bmatrix} + \begin{bmatrix} a_i^0 \\ a_j^0 \end{bmatrix} \tag{12}$$

Generalizing, for m such constraints we need to determine the vector of $\mathbf{f} \in \Re^m$ unknown force magnitudes by solving the following linear system.

$$\underbrace{\mathbf{K}\mathbf{f} + \mathbf{a}^0}_{\mathbf{a}} - \mathbf{a}^d = \mathbf{0} \tag{13}$$

The Jacobian $\mathbf{K} \in \Re^{m \times m}$ can be evaluated by applying unit test force at each constraint and evaluating the changes in accelerations at every constraint. An efficient procedure to evaluate the Jacobian using the framework of Featherstone's Articulated Body Method is given in [Kok04]. The constraint forces thus determined are applied to the multi-body system over the next integration step via the external force term $\hat{\mathbf{f}}$, as discussed in the previous section.

We replace the constraint direction $\mathbf{n}$ by a spatial vector $\hat{\mathbf{n}}$ to generalize the type of the constraint that can be represented, including the joint acceleration constraint. We further extend the constraint system to include the unilateral constraints such as collisions, friction and joint limits by posing it as a Linear Complementarity Problem (LCP).

$$\underbrace{\mathbf{K}\mathbf{f} + \mathbf{a}^0}_{\mathbf{a}} - \mathbf{a}^d \geq \mathbf{0} \quad \Leftrightarrow \quad \mathbf{f} \geq \mathbf{0} \tag{14}$$

The LCP states that forces $\mathbf{f}$, applied only in positive constraint direction, would strive to make the resulting constraint accelerations $\mathbf{a}$ equal to desired acceleration $\mathbf{a}^d$. However, force f_i will be zero if the resulting constraint acceleration a_i is greater than desired acceleration a_i^d. We will discuss the significance of the LCP formulation when we develop the collision response model in the next section.

At first, the solution of a LCP might appear as a daunting task. However, the iterative LCP methods [CPS92] are surprisingly simple and adequate for our purpose. [Ken04] gives a gentle introduction to the solution methods based on various matrix splitting techniques. Apart from the simplicity of the implementation, the iterative LCP solvers have other advantages as compared to pivoting methods. As we will discuss in the next section, we often need to apply multiple constraints on a single link. In this case, the Jacobian $\mathbf{K}$ will have linearly dependent columns. The iterative methods try to distribute the required forces evenly on the link, when multiple solutions exists in this case. Secondly, the LCP may not have a solution. The LCP problems arising from friction models are often non-convex [Bar92, PT96], particularly for high friction values. Further, the Jacobian can be singular or near singular if the limited DOFs of multi-body system does not allow motion in a constraint directions. In all these cases, we can bailout early in the solution process and still have a finite and a well distributed solution for the forces.

Algorithm 2 *sor_lcp*$(\mathbf{A}, \mathbf{x}, \mathbf{b}, \omega, \varepsilon, K_{iter})$

Require: $\mathbf{A}$ is symmetric, positive semi-definite
Ensure: $\mathbf{w} \equiv \mathbf{A}\mathbf{x} - \mathbf{b} \geq \mathbf{0}$, $\mathbf{x} \geq \mathbf{0}$, $\mathbf{x}^T\mathbf{w} = 0$

```
1:  x ← 0
2:  n ← dim(x)
3:  for k = 1 to K_iter do
4:      for i = 1 to n do
5:          δ ← 0
6:          for j = 1 to i − 1 do
7:              δ = δ + A_{i,j} x_j
8:          end for
9:          for j = i + 1 to n do
10:             δ = δ + A_{i,j} x_j
11:         end for
12:         δ = (b_i − δ)/(A_{i,i} + ε)
13:         x_i = (1 − ω)x_i + ωδ
14:         x_i = 0   if  x_i < 0
15:     end for
16: end for
```

We list an iterative LCP solver in Algorithm 2. Apart from the lines 14 and 12 it is a standard successive-over-relaxation linear system solver. Line 14 ensures the inequality condition. We add ε to the diagonal term in line 12 to make $\mathbf{A}$ positive definite, from potentially positive semi-definite, and guard against potentially zero or near zero diagonal terms in the Jacobian $\mathbf{K}$. Further, instead of any elaborate convergence check, we simply make fixed number of iterations K_{iter}, as we know that the solution may not exist. Using forces for enforcing the constraints has an advantage here. If the forces are indeterminate, they get projected into the multi-body's motion null-space, thus always giving valid

configuration, without any "pops" in the simulation. Further, as the forces are determined analytically, as compared to, say, penalty based formulation, they are small for most types of the constraints. Thus they are well within the stability zone of the integrator taking 4-8 time-steps per frame. The only exception to this is velocity impulse constraint, we will discuss this case in detail in the next section. As the constraint may not be satisfied accurately, we augment the constraint acceleration by a proportional-derivative form. To exemplify, for a positional constraint, the constraint desired acceleration and the constraint direction be

$$\begin{aligned} \mathbf{a}_i^d = \ddot{\mathbf{p}}_i^d \; + \; K_p(\mathbf{p}_i^d - \mathbf{p}_i) + \; K_d(\dot{\mathbf{p}}_i^d - \mathbf{v}_i) \\ a_i^d = \| \mathbf{a}_i^d \|, \quad \mathbf{n}_i = \mathbf{a}_i^d / a_i^d \end{aligned} \tag{15}$$

where, $\ddot{\mathbf{p}}_i^d, \dot{\mathbf{p}}_i^d, \mathbf{p}_i^d$ are acceleration, velocity and position of the constraint path, and $\mathbf{p}_i, \mathbf{v}_i$ are the current position and velocity of the constraint.

It is important to remove the effect of the constraint forces applied to multi-body system from the previous integration step, and adjust the initial constraint accelerations $\hat{\mathbf{a}}^0$ accordingly, before we determine the next set of constraint forces. We can use the same procedure that determines the Jacobian $\mathbf{K}$ by method of applying test forces for this.

5. Collision Response

We use the unilateral position constraints discussed in the previous section to develop collision response model for the multi-body system. Collision Detection is a mature subject in computer graphics. For brevity, we only enlist the requirements from the collision detection system for our purpose. Between the current configuration given by the state vector $\mathbf{y}_t$ and extrapolated configuration using derivative vector $\dot{\mathbf{y}}_t$ and next integration time step h, we find all the points on the multi-body system that would collide with the obstacles. Figure 4 shows two such collision positions – point $\mathbf{p}_i$ is already penetrated the obstacle and point $\mathbf{p}_j$ is about to collide. There may be more than one colliding point for a link. Let $\mathbf{n}_i$ be the collision normal, direction directly away from the obstacle, and $\mathbf{a}_i$ and $\mathbf{v}_i$ be collision accelerations and velocities respectively, relative to the obstacle.

We apply collision response in two steps – *contacts* and *impacts*. We first compute the unilateral constraints that would prevent collision points from accelerating towards the obstacle. Followed by computation of velocity impulses that would prevent collision points from moving towards the obstacle.

contacts: We decompose the collision acceleration and the collision velocity into the normal components $\mathbf{a}_{ni}, \mathbf{v}_{ni}$ and tangential components $\mathbf{a}_{ti}, \mathbf{v}_{ti}$. To prevent any acceleration towards the obstacle, we insert a unilateral constraint along the collision normal direction $\mathbf{n}_i$. The unilateral constraint corresponding to the friction acts in the tangent plane defined

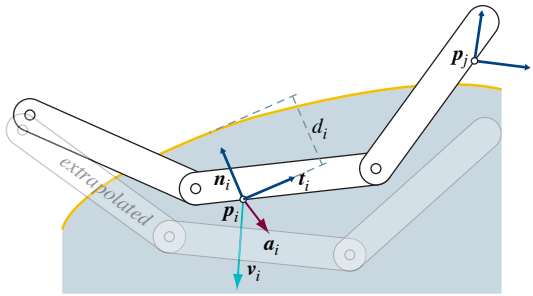


Figure 4: *Collision as unilateral constraints*

by the collision normal. We could sample the tangent plane into discrete set of tangents to formulate a complex and numerically expensive friction model based of the discrete frictional cone. Instead, taking inputs from [Kok04], we formulate a novel technique as follows. We set the unilateral constraint direction corresponding to friction as

$$\mathbf{t}_i = unit(\, \mathbf{a}_{ti} \; + \mathbf{v}_{ti}/h \,) \tag{16}$$

If both $\mathbf{a}_{ti}$ and $\mathbf{v}_{ti}$ is zero, we use previously determined tangent vector for the contact. Finally, the LCP formulation corresponding to the two unilateral constraints in the direction $\mathbf{n}_i$ and $\mathbf{t}_i$ at collision position $\mathbf{p}_i$ is

$$\begin{aligned} a_{ni} - a_{ni}^d \geq 0 \quad &\Leftrightarrow \quad f_{ni} \geq 0 \\ \mu_i f_{ni} - f_{ti} \geq 0 \quad &\Leftrightarrow \quad \lambda_i \geq 0 \\ (a_{ti} - a_{ti}^d) + \lambda_i \geq 0 \quad &\Leftrightarrow \quad f_{ti} \geq 0 \end{aligned} \tag{17}$$

We set the desired normal acceleration a_{ni}^d proportional to the penetration depth d_i if the point is penetrating, see equation 15, or zero if the collision point is outside the obstacle. The desired tangential acceleration a_{ti}^d is set to $(-\, \| \mathbf{v}_{ti} \| \, /h)$. The LCP formulation will compute required amount of normal force f_{ni} to remove the normal acceleration $\mathbf{a}_{ni}$. The tangential force f_{ti} will be at most $\mu_i f_{ni}$, and try to remove any tangential non-zero velocity component – the dynamic friction case, or if the tangential velocity is zero it will try to remove any tangential acceleration – the static friction case.

impacts: We use impulses to arrest the collision normal velocity $\mathbf{v}_{ni}$. Only those contacts are considered that have the normal velocity component $v_{ni} < 0$. For the impulse computations we can use the same acceleration constraints discussed in the previous section by setting $a_{ni}^d = -(1+\nu)v_{ni}$, where ν is collision restitution. Instead of applying potentially large forces, we alter the joint velocities $\dot{\mathbf{q}}_t$. This would invalidate the consistent solution set $\{\mathbf{y}_t, \dot{\mathbf{y}}_t\}$. We should correct $\ddot{\mathbf{q}}$ correspondingly. In reality, we found that the solver is tolerant to the error.

6. Implementation Details

Having developed the theoretical framework in the last three sections, in this section we would like to give a brief overview of the Oriented Strands system modeled as dynamics of multi-body system. It is implemented as a plug-in to Maya, as well as plug-in to our proprietary animation system. We use DASPK [BCP96] for our implementation.

Along with the robust formulation, any physically based simulation system to be successful in production environment needs to have an important aspect – to be able to art direct. In the following subsections, we develop some novel concepts towards that, along with few important implementation details. In our opinion, choice of reduced coordinate formulation and dynamics expressed in local frames makes some of these concept easier to implement.

6.1. Simulation Parameters and Workflow

The dynamic strand is composed from a hierarchy of input curves defined in a local frame, that defines the initial rest shape of the corresponding multi-body system. We provide the user with high-level control over the direct numerical simulation by means of relevant physical parameters of the dynamic strand, such as *mass per unit length, strand radius, bend stiffness/damping, torsional stiffness/damping, gravity, air drag*. The user can animate all the parameters and specify their length-wise anisotropic variation. The collision parameters *collision restitution and static/dynamic friction* are defined per obstacle surface. The strand may have additional anisotropic weights over collision parameters, along with their length-wise variation.

6.2. Stiffness Model and External Forces

In Section 3, while developing the DAE based formulation, we assumed single-DOF joints for the simplicity of discussion. However, we use three-DOF spherical joint in the implementation of Oriented Strands. The joint variable of i^{th} joint is expressed as a quaternion $\mathbf{q}_i \in \Re^4$ and the joint velocity as a vector $\mathbf{w}_i \in \Re^3$. [Had03, Fea87] gives details on how to formulate multiple-DOF joints.

We decompose the quaternion defining the relative transformation between two links into a twist component θ_t around the local y-axis and a pure bend component θ_b around a bend axis $\mathbf{b}$. We provide a nonlinear stiffness model as follows

$$\begin{aligned} \mathbf{Q}_b &= K_b(\mathbf{b})\ \mathbf{b}\ \tan((\theta_b - \theta_b^0)/2) \\ \mathbf{Q}_t &= K_t\ \mathbf{y}\ \tan((\theta_t - \theta_t^0)/2) \end{aligned} \tag{18}$$

where θ_b^0 and θ_t^0 correspond to the rest configuration. K_t is *torsional stiffness* coefficient and $K_b(\mathbf{b})$ is anisotropic *bend stiffness* coefficient. The response model is almost linear for small deformations. However, the non-linear response prevents excessive deformations and potentially joints snapping.

We support general external force fields using the plug-in architecture of Maya and that of our proprietary animation system. The user can attach any complex combination of time-varying fields such as wake, turbulence, fluid simulations and general event driven scripted force fields. The user can further specify length-wise anisotropic weights for the external force fields. The user can optionally include these fields in computing the Jacobians numerically discussed in Section 3.

6.3. Accurate Acceleration of Base Joint

In the reduced coordinate formulation it is critical to compute and supply the accurate velocities and accelerations of the base joint's prescribed motion path. We could have evaluated them numerically, however that would mean making repetitive evaluations of motion system at sub-frame interval, which is typically very expensive. Instead we interpolate the rigid-body transformation from four successive frames. Constructing a C_2 continuous curve that interpolates a number of prescribed rotations is a well studied problem. We use the method developed by [PK96], where we construct a piecewise cubic curve whose coefficients a_i, b_i, c_i are members of $so(3)$. The rotation is evaluated by taking the matrix exponential of this polynomial.

6.4. Time Scale and Local Dynamics

Often the dynamical simulation are encountered with very extreme and brisk animated motions. Although a robust formulations will be able to cope with the scenario, often the directors would want the motion to be selectively less violent. We introduce *time scale* β to control the amount of energy pumped in the system. It is a factor by which velocity and acceleration of the base joint get scaled and is typically between zero and one, however the user can set it more than one to accentuate the motion. The *local dynamics* γ is another similar parameter which blends out velocity and acceleration of some local dynamics reference frame.

$$\begin{aligned} \hat{\mathbf{a}}_0 &= \beta\ (\hat{\mathbf{a}}_0 - \gamma\ \hat{\mathbf{a}}_{ref}) \\ \hat{\mathbf{v}}_0 &= \beta\ (\hat{\mathbf{v}}_0 - \gamma\ \hat{\mathbf{v}}_{ref}) \end{aligned} \tag{19}$$

One scenario that is frequent is, a braid or long hair that fly away when character starts running or rides a horse. The local dynamics reference frame is simply set to the character's hip joint, and with appropriate *local dynamics* parameter one can control the amount of flyaway the user wants.

6.5. Posable Dynamics

Ears and tail, often have free secondary dynamic motion when the animator lets them "loose". However, animator would want to hit a certain pose at a certain time to make the character expressive. We tried different techniques that are based on the motion control principle. However, it did

not give desired results. For high values of K_p and K_d in the PID controller (Equation 15), the constraint follows the goal rather exactly. If we reduce K_p and K_d, due to *slew rate*, the PID controller gave a large error in achieving pose and the solution oscillated a lot before coming to rest to the animated pose. Surprisingly a very simple model worked for this specific application. We insert a spring between the dynamic strand and the desired animated pose at tip of each segment, to give a penalty based "soft" constraint. The user can animate the stiffness and damping, namely *pose strength* and *pose damping* to achieve the Posable Dynamics.

6.6. Zero-gravity Rest Shape

The rest shapes of the dynamic strands are typically modeled taking the gravity into account. Intricate hairstyle is a good example of this. Unfortunately, when we start simulating hair, the strands would sag under the gravity before they finally settles down. This would change the original art directed hair shape depending on the stiffness. Increasing the stiffness to preserve the shape would give unrealistic motion. One can go back and try to edit the rest shape so as to achieve desired shape under gravity. However, this would be very laborious and iterative process. We developed a technique to automatically compute the equivalent rest shape, without gravity, so that under gravity we would get the desired shape. The problem is a straight forward *inverse dynamics* problem in robotics. Given a set of external forces (gravity) and given the desired configuration of multi-body system, *inverse dynamics* problem finds set of joint forces required to achieve certain joint accelerations, zero in our case. We refer to [Fea87] for the details. We would like to highlight that it would be difficult to formulate this in the case of maximal coordinate formulation.

6.7. Strand-strand Interaction

Strand-strand interaction is not important in some simulation scenarios such as foliage motion, braids and ears, whereas it is critical in certain cases such hair simulation. We have implemented a modular plug-in field to Maya that computes the fluid like forces on a continuum, that can be attached to the Oriented Strands system to realize the strand-strand interactions as introduced by [HMT01]. The other interesting approaches to handle strand-strand interactions include wisp level interactions [PCP01, BKCN03], layers [LK01] and strips [CJY02].

7. Examples

In this section we would like to exemplify the versatile use of the dynamic strand primitive for character dynamics and visual effects. With each example, we would like to highlight the aspects of the formulation that is most relevant.

The example of lemurs dancing with fancy flower lanterns in (Figure 1), highlights the importance of stable and consistent coordinate frame along the strand. The tip of the strand is made heavy by using the length-wise variation of *mass per unit length* parameter, and the flower geometry is simply parented to the coordinate frame at the tip of the strand. Subtle twist dynamics adds to the realism.

Figure 5: *Moving Jungle - branched dynamics, stiff articulation, strong external force fields, scalability*

Figure 6: *Ears - posable dynamics*

The foliage simulation is a great example of branched dynamics. The individual trees are composed of hierarchy of strands, some of the segments being very stiff towards the trunk. One can follow the physically believable motion of trees under the influence of (strong) external force field such as wake and turbulence. It is also evident that the strand system is very scalable. Each tree typically has 50-100 segments and there are around 1000 trees in the "Blown by Horn" shot (video 3).

Donkey's ear exemplifies posable dynamics. Animators precisely control the subtle secondary dynamic motion

Figure 7: *Braid - "stiff" equations, local dynamics*

Figure 8: *Curly Bangs - intricate and zero-gravity rest shape*

around the animated poses, using time-varing *pose strength* parameter.

The bun followed by the long braid is composed of a single strand. The very stiff intial section gives the subtle interesting bounce and twist to the bun. The flexible section corresponds to the braid. The *local dynamics* parameter is used to control the amount of "floppyness" the braid exibits.

The simulation of curly bangs illustrate the ability of the system to handle "stiff" equations arising from the intricate rest shape. The rest shape is adjusted to account for the shape change due to gravity.

The long hair simulations highlight the effectiveness of the collision response model. The accurate computation of velocity and acceleration of the base joint results in highly realistic hair motion, where as *time scale* parameter gives control.

Figure 9: *Long Hair - accurate base acceleration, elaborate collision response, time scale*

We have not done a comprehensive performance analysis and optimization of the Oriented Strands system yet. Nevertheless, we would like to state the typical performance numbers for the hair simulations, as they represent the most of the dynamic complexities. The simulation of curly bangs uses 9 strands having an average 15 segments, each. The simulation runs at interactive rate of 2-4 Hz. The long hair simulations use 190 strands with 20-25 segments each. The simulations take less than 20 seconds per animation frame. The complexity of the strand dynamics is linear time in the total number of segments n. Whereas, the collision response is $O(m^2)$ in m number of collision points. Recently, we tried to analyze the convergence characteristics of the solver. We found that the solver uses sophisticated error control and heuristics, which result in a very wide variation in the number of non-linear iteration the solver takes. For the long hair simulations, the number varies from 2 to 213 iterations, with mean at 18.3 iterations. In the advent of multi-core and multi-cpu workstations, we would like to note that the computations of individual strands are *embarrassingly parallel*.

8. Conclusion, Limitations and Future Work

The simulation system of Oriented Strands has found widespread applications in feature animation and visual effects. We would like to attribute the successful usage of Oriented Strands to the robustness coming from the implicit formulation and the comprehensive collision response model, the intuitive workflow coming from local space formulation, physically based stiffness and collision models. In addition, innovative concepts such as time scale, local dynamics, posable dynamics, zero-gravity rest shape make Oriented Strands system "art directable".

In this paper, our focus has been "stiff" dynamics of serial open-chain multi-body systems with constraints and collision response. Fortunately, the DAE based formulation can be extended to include closed-loops [RJFdJ04]. Unfortunately, the analytical constraints and collision response model discussed so far do not readily fit the framework of closed-loops. Thus, in future we would like to extend, or develop new, methodologies to include closed-loops. Intricate jewelry on animated characters is our main motivation.

Other limitations of the proposed methodology are

- The approach is computationally expensive as compared to previous methods in [Had03, CCK05]. It would not scale well to do e.g. fur dynamics.
- Although one can incorporate stretch in the strand system by animating lengths of rigid segments, the system does not handle stretch dynamically.
- Developing and implementing constraints and collision response model is not as straightforward as compared to maximal coordinate formulation [CCK05].

9. Acknowledgments

I would like to thank Dave Eberle and Deepak Tolani for their great collaboration during the development of the system, Scott Singer, Arnauld Lamorlette, Scott Peterson, Francois Antoine, Larry Cutler, Lucia Modesto, Terran Boylan, Jeffrey Jay, Daniel Dawson, Lee Graft, Damon Riesberg, David Caeiro Cebrian, Alain De Hoe and everyone who directly influenced the development and tirelessly worked towards the successful use of the system in production, my supervisors Lisa Mary-Lamb, Andrew Pearce and Sanjay Das for supporting me all along, and anyone I forgot to mention.

References

[AUK92] ANJYO K., USAMI Y., KURIHARA T.: A simple method for extracting the natural beauty of hair. In *Proceedings of the 19th annual conference on Computer graphics and interactive techniques (SIGGRAPH)* (1992), ACM SIGGRAPH.

[Bar92] BARAFF D.: *Dynamic simulation of non-penetrating rigid bodies*. PhD thesis, Department of Computer Science, Cornell University, March 1992.

[Bar96] BARAFF D.: Linear-time dynamics using lagrange multipliers. *Proceedings of SIGGRAPH 96* (August 1996), 137–146.

[BCP96] BRENAN K. E., CAMPBELL S. L., PETZOLD L. R.: *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*. Classics in Applied Mathematics. SIAM, 1996.

[BKCN03] BERTAILS F., KIM T.-Y., CANI M.-P., NEUMANN U.: Adaptive wisp tree: a multiresolution control structure for simulating dynamic clustering in hair motion. In *2003 ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (aug 2003).

[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* (2005), pp. 153–160.

[CJY02] CHANG J., JIN J., YU Y.: A practical model for mutual hair inteactions. In *Proceedings of Symposium on Computer Animation* (July 2002), ACM SIGGRAPH, San Antonio, USA.

[CPS92] COTTLE R., PANG J. S., STONE R.: *The linear complementarity problem*. Academic Press, 1992.

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

[Had03] HADAP S.: *Hair Simulation*. PhD thesis, MIRALab, CUI, University of Geneva, January 2003.

[HMT01] HADAP S., MAGNENAT-THALMANN N.: Modeling dynamic hair as a continuum. *Computer Graphics Forum 20*, 3 (2001), 329–338.

[Ken04] KENNY E.: *Stable, Robust, and Versatile Multibody Dynamics Animation*. PhD thesis, Department of Computer Science, University of Copenhagen, November 2004.

[Kok04] KOKKEVIS E.: Practical physics for articulated characters. In *Proceedings of Game Developers Conference 2004* (2004).

[LK01] LEE D.-W., KO H.-S.: Natural hairstyle modeling and animation. *Graphical Models 63*, 2 (March 2001), 67–85.

[Mir96] MIRTICH B.: *Impulse-based Dynamic Simulation of Rigid Body Systems*. PhD thesis, University of California, Berkeley, December 1996.

[Pai02] PAI D. K.: Strands: Interactive simulation of thin solids using cosserat models. *Computer Graphics Forum 21*, 3 (2002), 347–352.

[PCP01] PLANTE E., CANI M.-P., POULIN P.: A layered wisp model for simulating interactions inside long hair. In *Proceedings of Eurographics Workshop, Computer Animation and Simulation* (September 2001), EUROGRAPHICS, Manchester, UK.

[PK96] PARK F. C., KANG I. G.: Cubic interpolation on the rotation group using cayley parameters. In *Proceedings of the ASME 24th Biennial Mechanisms Conference* (Irvine, CA, 1996).

[PS03] PANG J.-S., STEWART D. E.: *Differential Variational Inequalities*. Tech. rep., http://www.cis.upenn.edu/davinci/publications, 2003.

[PT96] PANG J. S., TRINKLE J. C.: Complementarity formulations and existence of solutions of dynamic multi-rigid-body contact problems with coulomb friction. *Mathematical Programming 73* (1996), 199–226.

[RCT91] ROSENBLUM R., CARLSON W., TRIPP E.: Simulating the structure and dynamics of human hair: Modeling, rendering and animation. *The Journal of Visualization and Computer Animation 2*, 4 (October-December 1991), 141–148.

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

[RJFdJ04] RODRÍGUEZ J. I., JIMÉNEZ J. M., FUNES F. J., DE JALÓN J. G.: Recursive and residual algorithms for the efficient numerical integration of multi-body systems. *Multibody System Dynamics 11*, 4 (May 2004), 295–320.

[Rub00] RUBIN M.: *Cosserat Theories: Shells, Rods and Points*. Springer, 2000.

[Sha01] SHABANA A. A.: *Computational Dynamics*. Wiley-Interscience, 2001.