Direct Position-Based Solver for Stiff Rods

Crispin Deul[1], Tassilo Kugelstadt[2], Marcel Weiler[1] and Jan Bender[2]

[1]Graduate School CE, TU Darmstadt
[2] RWTH Aachen University

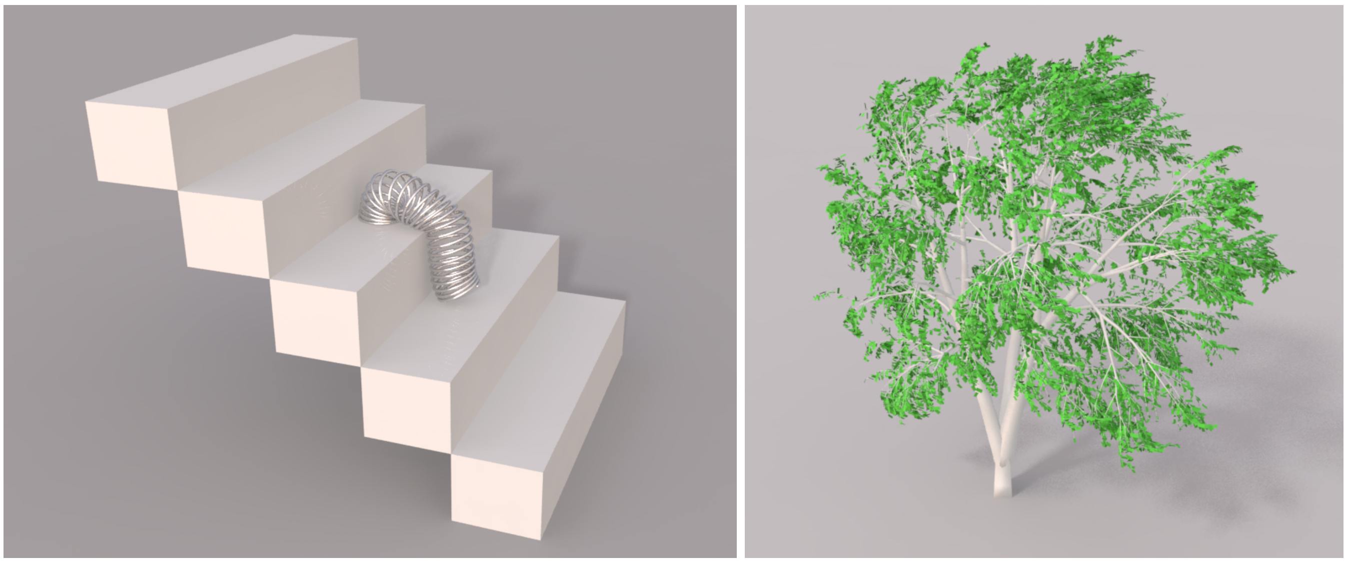

Figure 1: *Left: The Slinky model shows the performance of our solver in presence of a high number of collisions and self-collisions. Right: Our method is able to simulate tree-like structures. The white birch model represents a typical deciduous tree.*

Abstract

In this paper, we present a novel direct solver for the efficient simulation of stiff, inextensible elastic rods within the Position-Based Dynamics (PBD) framework. It is based on the XPBD algorithm, which extends PBD to simulate elastic objects with physically meaningful material parameters. XPBD approximates an implicit Euler integration and solves the system of non-linear equations using a non-linear Gauss-Seidel solver. However, this solver requires many iterations to converge for complex models and if convergence is not reached, the material becomes too soft. In contrast we use Newton iterations in combination with our direct solver to solve the non-linear equations which significantly improves convergence by solving all constraints of an acyclic structure (tree), simultaneously. Our solver only requires a few Newton iterations to achieve high stiffness and inextensibility. We model inextensible rods and trees using rigid segments connected by constraints. Bending and twisting constraints are derived from the well-established Cosserat model. The high performance of our solver is demonstrated in highly realistic simulations of rods consisting of multiple ten-thousand segments. In summary, our method allows the efficient simulation of stiff rods in the Position-Based Dynamics framework with a speedup of two orders of magnitude compared to the original XPBD approach.

CCS Concepts

•***Computing methodologies*** $\rightarrow$ ***Physical simulation;***

© 2018 The Author(s)
Computer Graphics Forum © 2018 The Eurographics Association and John Wiley & Sons Ltd. Published by John Wiley & Sons Ltd.

1. Introduction

The simulation of slender, rod-like structures has been an active research area in computer graphics over the last decades and is still of high interest. There are various applications like the simulation of ropes, threads, hair and fur of virtual characters as well as vegetation like trees or other plants. Such models are used for special effects in movies and in interactive virtual environments like games or surgical simulations. Recently, several methods for the fast and stable simulation of elastic rods have been developed. However, the stable and interactive simulation of very stiff objects like a trunk or branches of a tree, which are very common in nature and every-day life, remains a challenging problem.

Many real world materials like wood and hair are nearly inextensible and they can have a large spectrum of bending and torsion resistance. This poses high demands on the numerical integrator. Simple explicit methods require very small time steps to remain stable, which leads to high computational costs. Furthermore, numerical stability cannot be guaranteed, but this is crucial in many interactive applications. Lately, Position-based dynamics (PBD) [BMM14] has become very popular, because it is simple to implement, numerically stable and very versatile, while it can handle arbitrary position-based constraints. A major problem of PBD is that material stiffness depends on the number of iterations and on the time step size. Recently, Macklin et al. [MMC16] published an extended version of PBD, called XPBD, that solved this problem and supports physically meaningful material parameters. However, when it comes to simulations of very large and stiff structures with thousands of segments and constraints like the tree in Figure 1, the solver of XPBD requires a large number of iterations to converge. Moreover, the material becomes softer, when too few iterations are used.

In this paper we address the convergence problem of iterative solvers for rod simulations by exploiting their special acyclic structure (in the rest of the paper called tree). This structure allows for using a direct solver with linear-time complexity to solve all constraints in the tree simultaneously. Baraff [Bar96] proposed such a solver for trees of articulated rigid bodies. More recent works, e.g. of Hernandez et al. [HGCO11] or Aubry and Xian [AX15], solved the drift problem of Baraff's method (c.f. [BET14]) by formulating implicit position constraints at the end of the time step. Nevertheless, a rod modeled with one of these approaches might suffer from jitter in high tension cases [GHF*07, TNGF15]. Han and Harada [HH13] proposed a linear time algorithm for particle distance constraints with PBD. However, they only support distance constraints and no elastic potentials and only handle linear chains and no trees.

The position-based approach avoids the numerical drift problem by solving the time integration with an implicit constraint evaluation. Moreover, the discretization with an implicit constraint direction (ICD) reduces instabilities in tension situations (see Goldenthal et al. [GHF*07]). In general, the constraint formulations of XPBD lead to a non-linear system with equalities and inequalities. The system is not easy to solve and needs special treatment. Therefore, XPBD applies a non-linear Gauss-Seidel solver. This solver computes individual Newton steps for each position constraint in an iterative process. The individual solutions of the constraints influence each other through an immediate position update of the constrained generalized coordinates. As such, constraint corrections travel only locally from constraint to constraint. Thus, in case of a chain of constraints (e.g. a rod) the solver needs many iterations to converge. However, the system for a single rod does not include inequalities. Therefore, in contrast to the original XPBD approach, we solve this system with Newton's method, where the solution to the subproblems is solved with a linear-time solver. This increases convergence significantly.

Furthermore, we employ the physically accurate Cosserat rod model to achieve realistic bending and torsion resistance. We show how to discretize inextensible and unshearable rods as chains of rigid segments that are connected by a combination of zero-stretch, bending and twisting constraints.

Our method provides a significant speedup of two orders of magnitude compared to the non-linear Gauss-Seidel solver. Thus, it greatly reduces the time taken to realistically simulate complex scenes with rods consisting of several ten-thousands rigid segments and very high bending and torsion stiffness in the Position-based dynamics framework. The material stiffness of the tree in Figure 1 can be easily tuned by using Young's modulus and torsion modulus and it realistically depends on the varying thickness of the trunk and the branches by using the Cosserat model. The Slinky scene shows that collisions are naturally handled by constraints (see Section 4). Moreover, we compared our model to the analytic solution of the bending of a cantilever beam that was fixed at one end that demonstrates the accuracy of our method (see Section 5). Finally, our method is easy to implement and can be easily integrated into existing Position-based dynamics frameworks.

2. Related Work

In this section we discuss the most closely related work in strand and rod simulation as well as Position-based dynamics. Many approaches for the simulation of elastic rods or strands have been proposed in computer graphics, including mass spring models, Cosserat models, multi-body strands and position-based methods.

Mass spring models: Mass spring models are very popular in computer graphics due to their simplicity and computational performance and are widely used in cloth and strand simulation [BW98, CCK05, IMP*13]. However, mass spring models cannot naturally handle torsion effects of rods. Therefore, Selle et al. [SLF08] used additional altitude springs for the simulation of curly hair. But these springs lead to increased computational costs. Another problem is that the only material parameters of mass spring models are the stiffness constants of the springs. Therefore, tuning the material behavior is difficult. Michels et al. [MMS15] solved this issue by using special cuboidal mass spring elements for the discretization of the rod. These elements allow the mapping of real world material parameters like the Young's or torsion modulus to the spring constants. Yet their employed exponential integrator makes it mathematically unclear how hard constraints and non-smooth contacts are handled. They also report that practical time step sizes for their simulations are in the range of $10^{-3}s$, while our method runs well for step sizes larger than $10^{-2}s$ for similarly sized objects.

Cosserat rods (implicit discretization): The Cosserat model describes elastic rods which can undergo stretch, shear, bend and twist deformations. An introduction to the Cosserat theory can be found in the book of Antman [Ant05]. Pai [Pai02] introduced the Cosserat model to the computer graphics community in order to simulate sutures and catheters in virtual surgery applications. He modeled rods with an implicit discretization of the centerline by using curvature as generalized coordinate. This allowed him to find quasistatic solutions for inextensible and unshearable rods. However, it is complicated to handle interaction forces and contacts due to the implicit discretization of the centerline. Bertails et al. [BAC*06] extended this approach to the super-helix model, which also handles dynamics and was used for simulations of curly hair. Later, Bertails [Ber09] presented the linear time super-helix model, which reduced the computational complexity to linear time. The implicitly discretized Cosserat models remove all constraints by using reduced coordinates. Hence, they cannot be directly integrated into the PBD framework.

Cosserat rods (explicit discretization): A further approach is to use an explicit centerline representation. For instance Gregoire and Schömer [GS07] discretized the centerline explicitly as linear elements, which are described with position and orientation. These were used to formulate constraint energies for computing the static equilibrium of cables in virtual assembly simulations. Spillmann and Teschner [ST07] extended this approach to achieve dynamic simulations with their CoRdE model. They used the finite element method to discretize the continuous deformation energies. The equations of motion of the rod elements were derived using the Euler Lagrange equations and solved with explicit Euler integration. The explicit centerline representation allows for efficient contact and collision handling, which was used to simulate looping phenomena and knots at interactive rates [SBT07, ST08]. A major drawback of the CoRdE model is the explicit time integration, which demands small time steps to remain numerically stable. Lang et al. [LL09, LLA11] used a similar approach to simulate cables in mechanical engineering applications. They formulate the Cosserat rod model as a Lagrangian field theory and derive partial differential equations for the dynamics of the centerline and the frames. The PDEs are discretized with finite differences on a staggered grid and solved with standard solvers for stiff differential equations. In our work we also use an explicit centerline discretization, by representing the centerline as rigid segments and the frames with the rigid segment orientations. Similar to [ST07, LLA11] we use the finite difference approach to discretize the bending and torsion energy. Our big advantage over [ST07] is that we use implicit time integration and have no stability issues regarding the time step size. In contrast to [LLA11], we do not rely on a band structure in our system matrix and, thus, can simulate models with junctions, like trees, as well.

Discrete differential geometry formulation: Bergou et al. [BWR*08] introduced a discrete elastic rod model based on an explicit discretization of the centerline and generalized coordinates for the frames. For unshearable rods the director $\mathbf{d}_3$ (see Figure 2) is always aligned with the corresponding edge of the centerline. Hence, the frame can only rotate around the edge, while the angle which measures the distance to a reference frame is sufficient to describe the frame. The reference frames are constructed by parallelly transporting the reference frame from one end along the rod. This construction leads to dense energy Hessians, which makes implicit time integration expensive and therefore explicit integration was used. Later, Bergou et al. [BAV*10] extended this model to simulate discrete viscous threads by shifting the reference frame construction to the time domain. This results in banded energy Hessians, which also allows efficient implicit integration of elastic rods. However, branching structures like plants are only supported by dividing a rod into two rods and introducing a rigid body between the rods that couples the branch to the parent rod. In cases with lots of branches this increases the degrees of freedom and the computational costs. In contrast our model supports branching that is independent of the parent branch discretization.

Multi-body systems: Another approach is to simulate strands as chains of articulated rigid bodies. A reduced coordinate formulation of the multi-body chain is used by Hadap [Had06]. While this formulation handles high bending and torsion stiffness well, it is rather hard to define interacting constraints like contacts. Moreover, this approach is computationally too expensive for real-time applications. The method of Servin and Lacoursière [SL08] applies weakly relaxed kinematic constraints to simulate cables with very high mass ratio between the end segment and the cable segments. Bending and twisting constraints stabilize the system of cable segments, which is solved with a direct solver. To support large deformations of the flexible cables, the twisting constraints are formulated in terms of the angle between vectors fixed to the discretization of the centerline. As such, the method does not support anisotropic cross sections. In constrast, our model of bending and twisting resistance with the help of the Darboux vector allows us to choose different stiffness values along the main axes of the local frame. Quigley et al. [QYH*17] proposed a linear-time algorithm for the interactive simulation of trees, which are represented by articulated rigid bodies. Bending is modeled with rotational spring constraints. Each spring constraint is solved analytically while the dynamics of a whole tree is solved in a reduced coordinate fashion. This results in linear time complexity. However, the choice of constraints sacrifices the freedom to simulate arbitrarily compliant joints. Another approach to model geometric deformations of surfaces by embedding them into rigid prisms that are connected by elastic joints was presented by Botsch et al. [BPGK06]. The elastic joints could also be used to simulate stretching, bending and twisting behavior of elastic rods. However, as a geometrically motivated method which was tailored for geometric shape modeling it does not use real world material parameters and bending and torsion stiffness cannot be tuned independently.

Position-based dynamics: Due to its simplicity, performance and numerical robustness Position-based dynamics is widely used in computer graphics [BMM14]. In PBD objects are modeled as particles which are coupled by positional constraints. These constraints are solved iteratively in each time step. Originally it was used to simulate cloth and soft bodies [MHHR07], but it was also extended to simulate straight hair strands [MCK12] in real-time. Umetani et al. [USS14] simulated elastic rods with bending and twisting deformations in PBD by representing the material frames implicitly with the help of ghost particles. This requires multiple constraints to keep the frame attached to the centerline. In a constraint graph, where constraints and bodies are represented as nodes which are connected by an edge if a constraint acts on a body (see [Bar96] for details), this setup leads to loops. Therefore, the method is not well suited to be combined with a direct solver which often requires a loop-free constraint graph. Recently, Deul et al. [DCB14] presented an approach to simulate rigid bodies with orientation constraints in PBD. In our work we transfer their ideas to the discretization of rods into rigid segments that are connected by our combined zero-stretch, bending and twisting constraint. Similar orientation constraints were used by Kugelstadt and Schömer [KS16] to simulate Cosserat rods within PBD. Their constraints are modeled with a finite difference approximation of the strain measures from Cosserat theory. We use a comparable formulation in our new combined stretch, bending and twisting constraint, which increases convergence and allows us to employ a direct solver.

3. Cosserat Model

The Cosserat theory models a rod as a smooth curve $\mathbf{r}(s) : [s_0, s_1] \rightarrow \mathbb{R}^3$, called centerline. To describe the bending and twisting degrees of freedom, an orthonormal frame with basis $\{\mathbf{d}_1(s), \mathbf{d}_2(s), \mathbf{d}_3(s)\}$ is attached to each point of the centerline. In the following, the vectors $\mathbf{d}_i$ will be denoted as directors. The first and second director span the cross section of the rod, while the third director is the cross section normal, as depicted in Figure 2. The directors can

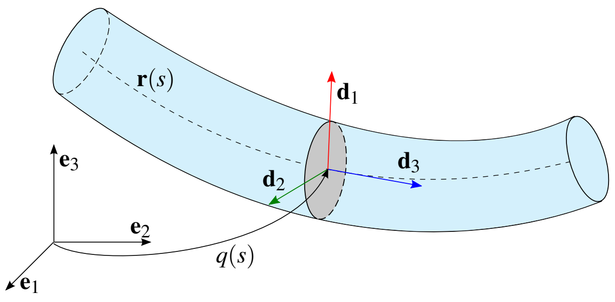

Figure 2: *Geometry of a continuous Cosserat rod.*

be parameterized with a rotation quaternion $q(s)$, which computes the directors by rotating a frame that is initially parallel to the world coordinate basis $\{\mathbf{e}_1(s), \mathbf{e}_2(s), \mathbf{e}_3(s)\}$. The frame spanned by the directors of the initial rod configuration is called material frame.

Strain measures and deformation energies: The Cosserat theory defines the strain measure $\mathbf{\Gamma}$ that determines stretch and shear as

$$\mathbf{\Gamma}(s) = \frac{\partial}{\partial s}\mathbf{r}(s) - \mathbf{d}_3(s), \tag{1}$$

where $\mathbf{r}(s)$ is a unit-speed parametrization for the rod's rest pose. It follows that the tangent of centerline $\partial_s \mathbf{r}$ has unit length and $\mathbf{\Gamma}$ vanishes for the rest configuration because $\mathbf{d}_3$ also has unit length. Stretching or compression directly corresponds to length changes of the centerline tangent, such that $\mathbf{\Gamma}$ measures stretch. Furthermore, $\mathbf{\Gamma}$ measures how far the direction of the tangent and the director $\mathbf{d}_3$ deviate, which is denoted as shear. However, we aim for unshearable and inextensible rods, i.e. $\mathbf{\Gamma} = \mathbf{0}$. Therefore, we use a discretization of the rod which guarantees unshearability and enforces inextensibility with zero-stretch constraints (see Section 3.1).

The strain measure for bending and twisting is determined using the Darboux vector $\mathbf{\Omega}$ which is defined as

$$\frac{\partial}{\partial s}\mathbf{d}_i(s) = \mathbf{\Omega}(s) \times \mathbf{d}_i(s). \tag{2}$$

This equation shows the close relationship between the Darboux vector $\mathbf{\Omega}$ and the angular velocity $\boldsymbol{\omega}$ of a rigid body. For $\boldsymbol{\omega}$ the same equation holds, except that the spatial derivative has to be exchanged with the time derivative. Analogously to the angular velocity the Darboux vector can be expressed in terms of the rotation quaternion

$$\mathbf{\Omega}(s) = 2\bar{q}(s)\frac{\partial}{\partial s}q(s), \tag{3}$$

where $\bar{q}$ denotes the conjugate quaternion. In Equation (3) the Darboux vector is expressed w.r.t. the material frame, which allows simple interpretations of its components. The first and second component $\mathbf{\Omega}_1$ and $\mathbf{\Omega}_2$ measure local bending or curvature in $\mathbf{d}_1$ and $\mathbf{d}_2$ direction and the third component measures torsion around $\mathbf{d}_3$. In the following we will allow rods with initial bending and torsion and we denote the rest pose Darboux vector as $\mathbf{\Omega}^0$. We define the strain measure of bending and torsion as

$$\Delta\mathbf{\Omega} = \mathbf{\Omega} - \mathbf{\Omega}^0 = 2\bar{q}\frac{\partial}{\partial s}q - 2\bar{q}^0\frac{\partial}{\partial s}q^0, \tag{4}$$

which locally measures bending and torsion as the deviation of the Darboux vector from the rest-pose Darboux vector. An important property of the strain measure $\Delta\mathbf{\Omega}$ is that it is invariant to rigid body motion.

The continuous deformation energy can be defined as a quadratic form of the strain measure. We define the bending and torsion energy as

$$E_b = \frac{1}{2}\int \Delta\mathbf{\Omega}^T \boldsymbol{K}\Delta\mathbf{\Omega} ds, \qquad \mathbf{K} = \begin{pmatrix} EI_1 & 0 & 0 \\ 0 & EI_2 & 0 \\ 0 & 0 & GJ \end{pmatrix}, \tag{5}$$

where $\mathbf{K}$ is a 3×3 matrix with material parameters for bending and torsion stiffness, E is Young's modulus and G is the torsion modulus of the material. In classical mechanics the shear modulus is used to define the stiffness with respect to torsion. However, since in our work we aim for unshearable rods, we speak of the torsion modulus to emphasize that only torsion is affected by this parameter. The moments of inertia of the cross section I_1 in direction $\mathbf{d}_1$ and I_2 in direction $\mathbf{d}_2$ as well as the polar moment of inertia J are given by:

$$I_1 = \int_{\mathcal{A}} x_2^2 dx_1 dx_2, \qquad I_2 = \int_{\mathcal{A}} x_1^2 dx_1 dx_2, \qquad J = I_1 + I_2, \tag{6}$$

where $\mathcal{A}$ is the cross section area. In the special case of a circular cross section the inertia can be computed as $I_1 = I_2 = \frac{1}{4}\pi r^4$.

3.1. Discretization

We aim for the simulation of inextensible and unshearable rods. Therefore, we discretize the centerline as a chain of rigid segments that are connected by a combination of zero-stretch, and bending

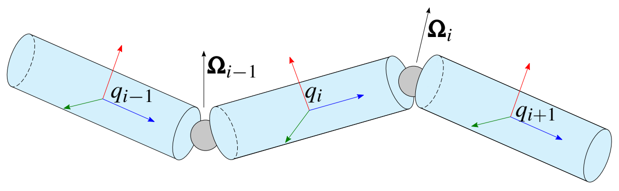

Figure 3: *The rod is discretized as a chain of rigid segments that are connected by a combined zero-stretch, bending and twisting constraint.*

and twisting constraints (see Figure 3). This guarantees that the rod cannot be stretched. Furthermore, in contrast to methods using particles and a staggered constraint distribution, we solve the combined constraints between two segments simultaneously. As a result, the convergence is increased and it is much easier to combine the constraints with a direct solver.

We use the rigid segment orientations to discretize the directors, which also guarantees that the frames are aligned with the centerline and shear deformation is not possible. Therefore, our discretization of the rod automatically satisfies the hard constraint $\boldsymbol{\Gamma} = \mathbf{0}$. Each rigid segment i is described by the position of its center of mass $\mathbf{p}_i$ and a rotation quaternion q_i.

In order to formulate a discrete bending and torsion energy, we need to discretize the Darboux vector at constraint position i. The derivative of the quaternion is discretized as finite difference approximation $\partial_s q_i = \frac{1}{l_i}(q_{i+1} - q_i)$, where l_i denotes the average length of segment i and segment $i+1$. Furthermore, we need to evaluate the quaternion, which is only known at the center of the segments, at the segment position. Therefore, we interpolate the quaternion using the arithmetic mean of the adjacent quaternions as proposed by Spillmann and Teschner [ST07], such that the discrete Darboux vector becomes

$$\boldsymbol{\Omega}_i = \frac{1}{l_i}\Im\left[(\bar{q}_{i+1} + \bar{q}_i)(q_{i+1} - q_i)\right] = \frac{2}{l_i}\Im\left[\bar{q}_i q_{i+1}\right]. \tag{7}$$

Here we have to take the imaginary/vector part of the quaternion, which is denoted as $\Im[\cdot]$. This results in the discrete bending and torsion energy

$$E_b = \frac{1}{2}\sum_i \Delta\boldsymbol{\Omega}_i^T \mathbf{K}_i \Delta\boldsymbol{\Omega}_i. \tag{8}$$

In the following we will also need the Jacobian of the Darboux vector. It can be computed by writing the quaternion product as matrix-vector multiplication

$$\bar{q}_i q_{i+1} = \begin{pmatrix} \Re[q_i] & \Im[q_i]^T \\ -\Im[q_i] & \Re[q_i]\mathbb{1}_{3\times 3} - [\Im[q_i]]^\times \end{pmatrix} \begin{pmatrix} \Re[q_{i+1}] \\ \Im[q_{i+1}] \end{pmatrix}, \tag{9}$$

where $\Re[\cdot]$ takes the real/scalar part of a quaternion, $[\cdot]^\times$ denotes the skew symmetric matrix, which can be used to write the cross product as matrix-vector product and $\mathbb{1}_{3\times 3}$ is the 3×3 identity matrix. The Darboux vector is the imaginary part of Equation (9) scaled by the inverse of the average segment length (see Equation (7)), which means the Jacobian of $\boldsymbol{\Omega}_i$ w.r.t. q_{i+1} is

$$\frac{\partial}{\partial q_{i+1}}\boldsymbol{\Omega}_i = \frac{2}{l_i}\begin{pmatrix} -\Im[q_i] & \Re[q_i]\mathbb{1}_{3\times 3} - [\Im[q_i]]^\times \end{pmatrix}. \tag{10}$$

The derivative w.r.t. q_i can be found analogously by using $\boldsymbol{\Omega}_i = \frac{2}{l_i}\Im[\bar{q}_i q_{i+1}] = -\frac{2}{l_i}\Im[\bar{q}_{i+1} q_i]$, which results in

$$\frac{\partial}{\partial q_i}\boldsymbol{\Omega}_i = -\frac{2}{l_i}\begin{pmatrix} -\Im[q_{i+1}] & \Re[q_{i+1}]\mathbb{1}_{3\times 3} - [\Im[q_{i+1}]]^\times \end{pmatrix}. \tag{11}$$

4. Constraint Solver

In this section we introduce an extension of the XPBD algorithm proposed by Macklin et al. [MMC16] in order to simulate stiff Cosserat rods. We first give a brief overview of XPBD, then present our extension of XPBD with a direct solver, and finally introduce our combined zero-stretch, bending and twisting constraint in detail.

4.1. XPBD

Our constraint solver is based on the XPBD algorithm of Macklin et al. [MMC16]. XPBD is an extension of Position-based dynamics which allows to apply physically meaningful material parameters by using compliance constraints. In this section we will summarize the core ideas of XPBD and present the system of equations that we have to solve.

The goal is to solve Newton's equation of motion

$$\mathbf{M}\ddot{\mathbf{x}} = -\nabla U(\mathbf{x}) + \mathbf{f}_{ext}, \tag{12}$$

where $\mathbf{M}$ is the mass matrix, $\mathbf{x} = (\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_n)^T$ denotes the vector that collects all generalized coordinates of the physical system, U denotes the elastic potential and $\mathbf{f}_{ext}$ collects external forces like gravity or wind forces. In our case $\mathbf{x}$ contains all mass centers and quaternions and the block diagonal of $\mathbf{M}$ contains the masses and inertia tensors of the rigid segments. Using an implicit Euler discretization yields:

$$\mathbf{M}\left(\frac{\mathbf{x}(t+\Delta t) - 2\mathbf{x}(t) + \mathbf{x}(t-\Delta t)}{\Delta t^2}\right) = -\nabla U(\mathbf{x}(t+\Delta t)) + \mathbf{f}_{ext}. \tag{13}$$

The elastic potential $U(\mathbf{x})$ has the form

$$U(\mathbf{x}) = \frac{1}{2}\mathbf{C}(\mathbf{x})^T \boldsymbol{\alpha}^{-1}\mathbf{C}(\mathbf{x}), \tag{14}$$

where $\mathbf{C}(\mathbf{x}) = (C_0, C_1, ..., C_m)^T$ is a vector of constraint functions and $\boldsymbol{\alpha}$ is the diagonal compliance matrix describing the inverse stiffness. In case of the Cosserat bending and torsion potential we identify $\mathbf{C}$ with $\Delta\boldsymbol{\Omega}_i$ and $\boldsymbol{\alpha}$ with $\mathbf{K}_i^{-1}$ which is defined by Equations (5) and (6). This means that the compliance matrix $\boldsymbol{\alpha}$ describes the material properties in terms of Young's modulus, torsion modulus and the cross section geometry. Taking the negative gradient of the potential results in the elastic forces

$$\mathbf{f}_{el} = -\nabla U(\mathbf{x}) = -\mathbf{J}^T\boldsymbol{\alpha}^{-1}\mathbf{C} = \Delta t^2 \mathbf{J}^T\boldsymbol{\lambda}, \tag{15}$$

where $\mathbf{J} = \nabla\mathbf{C}$ denotes the constraint Jacobian. The elastic forces are decomposed into direction and magnitude by introducing the

Lagrange multiplier $\boldsymbol{\lambda} = (\lambda_0, \lambda_1, \ldots, \lambda_m)^T = -\tilde{\boldsymbol{\alpha}}^{-1}\mathbf{C}(\mathbf{x})$ with $\tilde{\boldsymbol{\alpha}} = \frac{\boldsymbol{\alpha}}{\Delta t^2}$. Plugging this into Equation (13) yields

$$\mathbf{M}(\mathbf{x}(t+\Delta t) - \tilde{\mathbf{x}}) - \mathbf{J}(\mathbf{x}(t+\Delta t))^T \boldsymbol{\lambda}(t+\Delta t) = \mathbf{0}, \quad (16)$$

$$\mathbf{C}(\mathbf{x}(t+\Delta t)) + \tilde{\boldsymbol{\alpha}}\boldsymbol{\lambda}(t+\Delta t) = \mathbf{0}, \quad (17)$$

where $\tilde{\mathbf{x}} = 2\mathbf{x}(t) - \mathbf{x}(t-\Delta t) + \Delta t^2 \mathbf{M}^{-1}\mathbf{f}_{ext}$. Note, that in contrast to many other implicit integration methods, like for example the ones of Hernandez et al. [HGCO11] or Aubry and Xian [AX15], Equation (16) contains an implicit constraint direction (ICD) $\mathbf{J}(\mathbf{x}(t+\Delta t))^T$ evaluated at the end of the time step. The inclusion of the ICD helps in reducing jitter that occurs in high tension situations when enforcing length preserving constraints like our zero-stretch constraints (see Goldenthal et al. [GHF*07] for a detailed discussion). A different approach to avoid instabilities while enforcing length preserving constraints is to introduce a geometric stiffness term to the system matrix (see Tournier et al. [TNGF15]). However, our method is based on an algorithm which already employs an ICD. As such, using the geometric stiffness term is not developed further in our paper.

The system of equations (16) and (17) is in general non-linear and can be solved with Newton iterations. Macklin et al. [MMC16] linearize these equations and propose several simplifications, resulting in the following KKT system for the i-th iteration:

$$\begin{pmatrix} \mathbf{M} & -\mathbf{J}(\mathbf{x}^i)^T \\ \mathbf{J}(\mathbf{x}^i) & \tilde{\boldsymbol{\alpha}} \end{pmatrix} \begin{pmatrix} \Delta\mathbf{x} \\ \Delta\boldsymbol{\lambda} \end{pmatrix} = -\begin{pmatrix} \mathbf{0} \\ \mathbf{C}(\mathbf{x}^i) + \tilde{\boldsymbol{\alpha}}\boldsymbol{\lambda}^i \end{pmatrix}, \quad (18)$$

where the vector $\left(\Delta\mathbf{x}^T, \Delta\boldsymbol{\lambda}^T\right)^T$ is the root update of the Newton step $\left((\mathbf{x}^{i+1})^T, (\boldsymbol{\lambda}^{i+1})^T\right)^T = \left((\mathbf{x}^i)^T, (\boldsymbol{\lambda}^i)^T\right)^T + \left(\Delta\mathbf{x}^T, \Delta\boldsymbol{\lambda}^T\right)^T$. The number of equations in (18) can be reduced by applying the Schur complement, which results in

$$\left[\mathbf{J}(\mathbf{x}^i)\mathbf{M}\mathbf{J}(\mathbf{x}^i)^T + \tilde{\boldsymbol{\alpha}}\right]\Delta\boldsymbol{\lambda} = -\mathbf{C}(\mathbf{x}^i) - \tilde{\alpha}\boldsymbol{\lambda}^i. \quad (19)$$

The updates of the positions are determined by

$$\Delta\mathbf{x} = \mathbf{M}^{-1}\mathbf{J}(\mathbf{x}^i)\Delta\lambda. \quad (20)$$

Macklin et al. solve Equation (19) with a non-linear Gauss-Seidel solver. This solver iterates over all constraints and computes solutions for each constraint in isolation. Hereby, the solution $\Delta\lambda_j$ to a single constraint equation with index j is computed as

$$\Delta\lambda_j = \frac{-C_j(\mathbf{x}^i) - \tilde{\alpha}_j\lambda_j^i}{\nabla C_j \mathbf{M}^{-1}\nabla C_j^T + \tilde{\alpha}_j}. \quad (21)$$

Immediately, after computing $\Delta\lambda_j$ position updates $\Delta\mathbf{x}$ are computed using Equation (20) and applied to the positions $\mathbf{x}^i$. As such, the non-linear system in Equation (19) is never build completely. It follows, that at no point in time there exists a subproblem in the sense of Newton's method. To sum up, the non-linear Gauss-Seidel solver computes individual Newton problems per constraint and the solutions of the constraints influence each other through the immediately updated positions. For infinitely stiff constraints with $\tilde{\alpha}_j = 0$, Equation (21) reduces to the Lagrange multiplier formula of standard Position-based dynamics [MHHR07]. This shows, that XPBD is a generalization of PBD that allows for using physically meaningful material stiffness parameters. However, the non-linear Gauss-Seidel solver is not well-suited for the simulation of stiff objects with many degrees of freedom like natural trees, because it requires many iterations until it converges.

4.2. Direct Solver

To overcome the problem of slow convergence, we solve the non-linear system for a rod and all its constraints with Newton's method. We compute solutions to the linear KKT subproblems with a sparse direct solver. For general tree structures, the matrix in Equation (19) is not necessarily sparse. However, the problem can also be solved in linear time by considering Equation (18), which is always sparse for trees. This system of equations can be solved by a modified version of Baraff's linear-time solver [Bar96], where the compliance factors must be considered. To simplify the notation we multiply the second row in Equation (18) with -1, denote the matrix as $\mathbf{H}$ and define $\mathbf{C}(\mathbf{x}^i) + \tilde{\alpha}\boldsymbol{\lambda}^i = -\mathbf{b}$, yielding

$$\begin{pmatrix} \mathbf{M} & -\mathbf{J}^T \\ -\mathbf{J} & -\tilde{\boldsymbol{\alpha}} \end{pmatrix} \begin{pmatrix} \Delta\mathbf{x} \\ \Delta\boldsymbol{\lambda} \end{pmatrix} = \mathbf{H}\begin{pmatrix} \Delta\mathbf{x} \\ \Delta\boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ -\mathbf{b} \end{pmatrix}. \quad (22)$$

This system equals the notation in [Bar96] with compliance $\tilde{\boldsymbol{\alpha}}$ added in the lower right submatrix of matrix $\mathbf{H}$. It can be solved in linear time by reordering the equations as described in [Bar96]. The reordering can be done once during initialization because it remains constant as long as the constraint topology does not change. After that the system can be solved using a sparse $\mathbf{LDL}^T$ decomposition. In typical scenarios of computer graphics simulated objects often interact with each other by collisions. However, a rod that collides at two points with the environment forms a loop. Furthermore, a rod representing a rope, e.g. a clothes line, is usually fixed at both ends to the environment and therefore also forms a loop. But, our direct solver only supports acyclic structures and no loops. Therefore, we alternate the computation of our rod solver with the computations of the non-linear Gauss-Seidel solver for loop-closing constraints and collision constraints. Thus, the position corrections of the direct solver and the position corrections due to collisions or loop closing constraints are transferred immediately from the one solver to the other. Algorithm 1 presents the alternation of direct solver and non-linear Gauss-Seidel solver, which from now on will be called KKT/GS solver, in the context of the XPBD time step. At the beginning of the time step the unconstrained positions are predicted by integrating the velocities and external forces (lines 1 and 2). Next, the initial values of the KKT/GS method are defined. Beginning from line 5 the rod, collision, or loop-closing constraints are solved iteratively. The iterations are stopped if the residual maximal error in the constraint functions is smaller than a predefined threshold η. In each iteration, first the KKT subproblems for the simulation of the rods are solved using our linear-time solver and the positions of the corresponding rigid segments as well as the Lagrange multipliers $\boldsymbol{\lambda}$ are updated (lines 6 to 9). Then, the collision and loop closing constraints are solved successively in non-linear Gauss-Seidel fashion (lines 10 to 14). After the solver iterations are finished the position at the end of the time step is set and the velocity is updated.

Algorithm 1 Time Step

```
1:  for all rigid segments do
2:      $\tilde{\mathbf{x}} \leftarrow \mathbf{x}(t) + \Delta t \mathbf{v}(t) + \Delta t^2 \mathbf{M}^{-1}\mathbf{f}_{ext}$
3:  $\mathbf{x}^0 \leftarrow \tilde{\mathbf{x}}$
4:  $\boldsymbol{\lambda}^0 \leftarrow \mathbf{0}$
5:  while $\left\| \mathbf{C}(\mathbf{x}^i) + \tilde{\boldsymbol{\alpha}}\boldsymbol{\lambda}^i \right\|_\infty > \eta$ do
6:      for all rods do
7:          solve Equation (22) using our linear-time solver
8:          update $\mathbf{x}^{i+1} \leftarrow \mathbf{x}^i + \Delta\mathbf{x}$
9:          update $\boldsymbol{\lambda}^{i+1} \leftarrow \boldsymbol{\lambda}^i + \Delta\boldsymbol{\lambda}$
10:     for all collision or loop-closing constraints do
11:         compute $\Delta\lambda$ with Equation (21)
12:         compute $\Delta\mathbf{x}$ with Equation (20)
13:         update $\mathbf{x}^{i+1} \leftarrow \mathbf{x}^i + \Delta\mathbf{x}$
14:         update $\boldsymbol{\lambda}^{i+1} \leftarrow \boldsymbol{\lambda}^i + \Delta\boldsymbol{\lambda}$
15:     $i \leftarrow i+1$
16: $\mathbf{x}(t+\Delta t) \leftarrow \mathbf{x}^i$
17: $\mathbf{v}(t+\Delta t) \leftarrow \frac{1}{\Delta t}(\mathbf{x}(t+\Delta t) - \mathbf{x}(t))$
```

4.3. Rod Constraint

In the following we define the constraint functions and their Jacobians for the zero-stretch, bending and twisting constraints, which we use to simulate elastic rods.

We combine the zero-stretch, bending and twisting constraints that couple two rigid segments into one six-dimensional constraint with the constraint function

$$\mathbf{C}(\mathbf{x}_1, q_1, \mathbf{x}_2, q_2) = \begin{pmatrix} \mathbf{R}(q_1)\mathbf{p}_1 + \mathbf{x}_1 - \mathbf{R}(q_2)\mathbf{p}_2 - \mathbf{x}_2 \\ \frac{2}{l_i}\Im\left[\bar{q}_1 q_2 - \bar{q}_1^0 q_2^0\right] \end{pmatrix}. \tag{23}$$

The first three rows connect a point $\mathbf{p}_1$ on segment 1 with a point $\mathbf{p}_2$ on segment 2. These points are given in the local coordinates of the corresponding segment and are transformed to world coordinates by applying the rotation matrix $\mathbf{R}(q)$ and translation $\mathbf{x}$ of their segment. The last three rows constrain bending and twisting (see Section 3.1). The stiffness of each constraint is given by a 6×6 compliance matrix

$$\tilde{\boldsymbol{\alpha}} = \frac{1}{\Delta t^2}\begin{pmatrix} \boldsymbol{\sigma}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{K}^{-1} \end{pmatrix}. \tag{24}$$

The diagonal matrix $\boldsymbol{\sigma}$ is the zero-stretch stiffness matrix. Furthermore, the bending and torsion stiffness $\mathbf{K}$ is the same as in Equation (5). Due to stability considerations of the solver and to avoid bad conditioning of the system matrix we weaken the inextensibility of the zero-stretch constraint slightly by setting the diagonal of $\boldsymbol{\sigma}^{-1}$ to values between 10^{-8} and 10^{-10} in our simulations.

Further, we need the constraint Jacobians w.r.t. segment 1 and segment 2, which are defined by

$$\mathbf{J}_1 = \left(\frac{\partial \mathbf{C}}{\partial \mathbf{x}_1}, \frac{\partial \mathbf{C}}{\partial q_1}\mathbf{G}(q_1)\right) = \begin{pmatrix} \mathbb{1}_{3\times 3} & -[\mathbf{R}(q_1)\mathbf{p}_1]^\times \\ \mathbf{0} & \frac{\partial \boldsymbol{\Omega}}{\partial q_1}\mathbf{G}(q_1) \end{pmatrix} \tag{25}$$

$$\mathbf{J}_2 = \left(\frac{\partial \mathbf{C}}{\partial \mathbf{x}_2}, \frac{\partial \mathbf{C}}{\partial q_2}\mathbf{G}(q_2)\right) = \begin{pmatrix} -\mathbb{1}_{3\times 3} & [\mathbf{R}(q_2)\mathbf{p}_2]^\times \\ \mathbf{0} & \frac{\partial \boldsymbol{\Omega}}{\partial q_2}\mathbf{G}(q_2) \end{pmatrix} \tag{26}$$

with

$$\mathbf{G}(q) = \frac{1}{2}\begin{pmatrix} -\Im[q]^T \\ \Re[q]\,\mathbb{1}_{3\times 3} + [\Im[q]]^\times \end{pmatrix}, \tag{27}$$

where $\mathbf{G}(q)$ describes the relationship between the quaternion velocity $\dot{q}$ and the angular velocity ω as $\dot{q} = \mathbf{G}(q)\omega$ and can also be used on the orientation level.

The mass matrix of each segment is given by

$$\mathbf{M} = \begin{pmatrix} m\mathbb{1}_{3\times 3} & & \mathbf{0} & \\ & \rho l_i I_1 & 0 & 0 \\ \mathbf{0} & 0 & \rho l_i I_2 & 0 \\ & 0 & 0 & \rho l_i J \end{pmatrix}, \tag{28}$$

where l_i is the length of segment i, ρ denotes the mass density, m is the mass of the segment and the moments of inertia I_1, I_2 and J are given by Equation (6). In our examples, we assume an isotropic rod with circular cross-section. As such, we approximate the geometry of a single segment with a cylinder and set the values for m, I_1, I_2 and J accordingly.

5. Results

We tested the performance of our method in various experiments. All experiments in this section were performed on a single core of an Intel Core i7-2600, 8GB of memory.

Cantilever beam: To validate our approach we fixed a rod of length 10m and a radius of 0.5m at one end. At the other end we applied a force F of 1000N. The rod is discretized into 50 segments and has a Young's modulus E of 1GPa. In the following we compare the deflection δ_C at the end of the rod with the analytical solution computed as $\delta_C = \left(FL^3\right)/(3EI)$, where I is the moment of the cross section (see Equation (6)). The simulated deflection is $6.7949 \cdot 10^{-3}$m whereas the analytical solution is $6.7906 \cdot 10^{-3}$m. As a result, we have a deviation of only about $4.3 \cdot 10^{-6}$m. For small deformations the elastic potential is nearly linear, which means that this result could be reached after one KKT/GS iteration as expected.

Wilberforce pendulum: We underpin the physical plausibility of our approach with the simulation of a Wilberforce pendulum (see Figure 4). The pendulum consists of a helical spring that is fixed at its upper end. A mass is attached to the lower end. At the beginning of the experiment the mass is moved out of the resting position. The accompanying video as well as Figure 5 show that the mass alternates between translational and rotational movement. This phenomenon is the result of coupled oscillations which are faithfully reproduced by our approach.

Slinky: We simulated a Slinky walking down a stairway (see Figure 1). Using this scenario we tested the performance of the rod solver in presence of a high number of self-collisions and collisions with the environment. The Slinky consists of 500 rigid segments and was simulated with a time step size of 10ms at three iterations of the alternating KKT solver for rods and non-linear Gauss-Seidel approach for collisions (see Section 4.2). We used an approach based on signed distance fields [KDB16] for collision

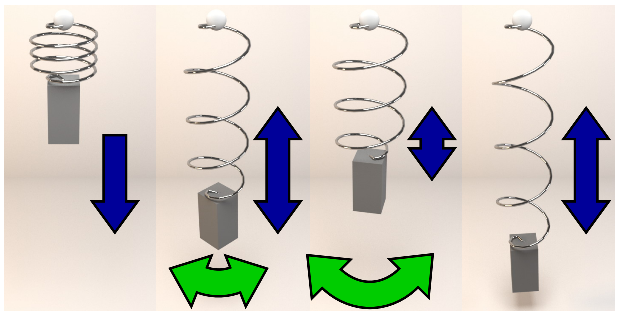

Figure 4: *First image: At the beginning of the simulation the pendulum bob of the Wilberforce pendulum is raised and then falls down. Second image: The bob moves up and down and begins to rotate around the vertical axis. Third image: After the first 12 seconds the translational up and down movement of the bob changes almost entirely to a rotation around the vertical axis. Fourth image: After about another 12 seconds the rotational movement has completely disappeared and the bob moves only upward and downward.*

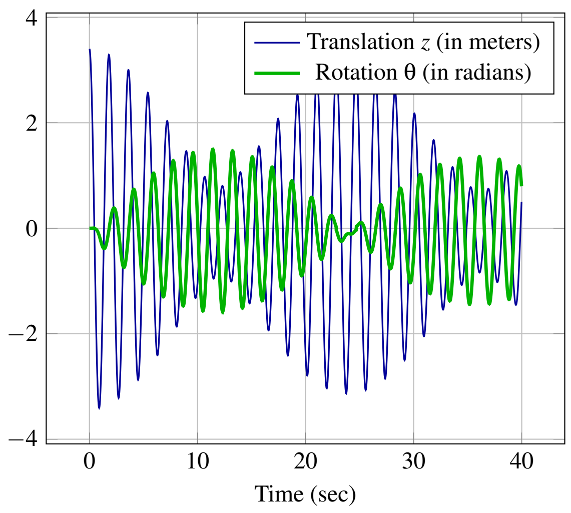

Figure 5: *Translation z and rotation θ of the pendulum bob as a function of time for our Wilberforce pendulum experiment.*

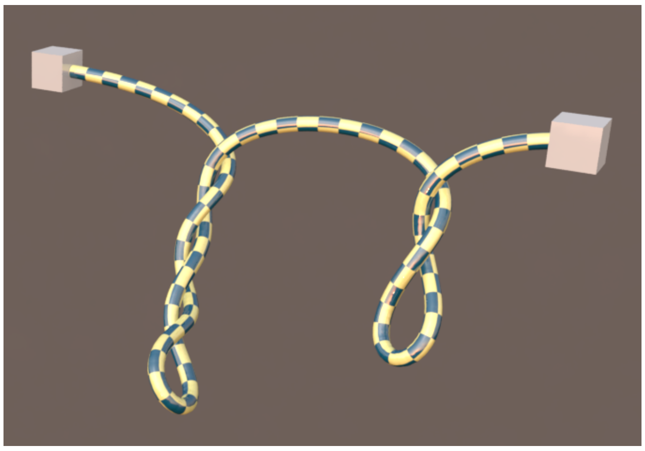

Figure 6: *An elastic rod under torsion shows the formation of plectonemes.*

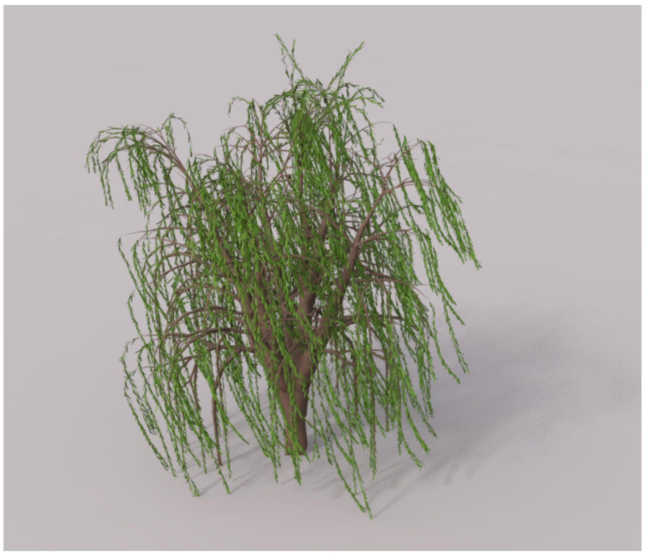

Figure 7: *The weeping willow model represents a tree with thick, as well as long, slender branches with many segments.*

detection. The result of the simulation is shown in Figure 1 and the accompanying video. The Slinky descends five stairs before stopping at the floor, while the solver handles hundreds of collisions per time step.

Knot: To demonstrate that our method is not only applicable to stiff rods, but to soft elastic rods as well, we simulated a strand under torsion. The rod is fixed at both ends. During the simulation, the left fixture is translated towards the right, while it performs ten full rotations. Figure 6 and the accompanying video show the expected formation of plectonemes. These confirm the soundness of our material model and show that self-collisions are handled well.

Tree simulation: The simulation of natural trees is one of the most demanding applications of rod simulation. Many approaches [BWR*08, Ber09, HKW09, ZB13, AX15] applied some kind of rod or beam model to tree simulation. In this paragraph we show how our approach performs in this complex simulation task. Our rod model was tested with three tree models with different topology, which were generated with Blender 2.78a's sapling tree generator. A white birch represents a typical deciduous tree (see Figure 1). The second tree is a weeping willow (see Figure 7) with long slender branches. Conifers with many small leaves are represented by a pine model (see Figure 8). Besides the trunk and the branches, we also represent each leaf of the tree with a single rigid segment

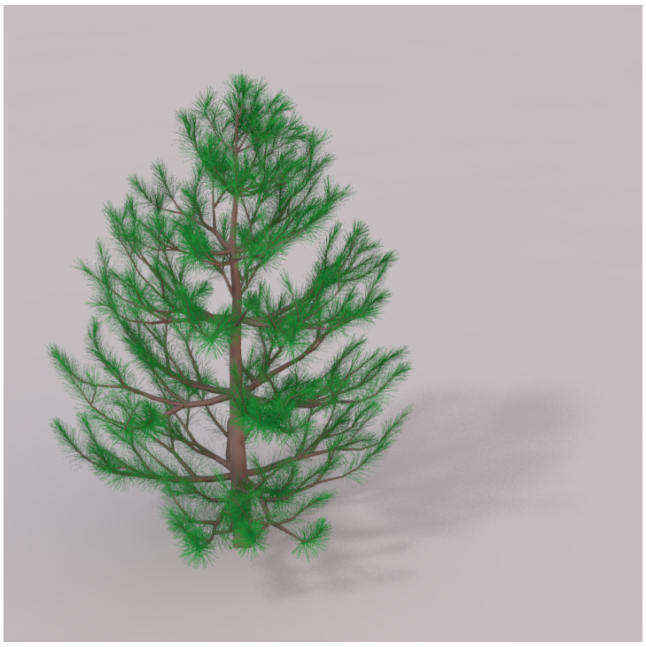

Figure 8: *An example for conifers, which contain a huge number of small leaves, is given by this pine model.*

Model	Segments	Avg. pos. corr. time
Birch	29973	658.08ms
Willow	38825	870.31ms
Pine	49548	1048.74ms

Table 1: *Number of segments and average time required for the position correction in the respective tree simulations.*

connected with the combined zero-stretch, bending and twisting constraint to the tree. As such, our model can capture the subtle movement of leaves in the wind. Wind is modeled by a procedural wind field. We first start with a subtle wind force which is successively increased to stormy conditions. Then we stop the wind to show that the model comes to rest. Each tree was simulated with a large time step size of 40ms and three iterations per time step to guarantee a upper bound for the residual error of 10^{-6}. The tree models consist of 29973, 38825, and 49548 rigid segments for the birch, the willow, and the pine, respectively. The Young's modulus is set to 12.5GPa and the torsion modulus to 6.5GPa for the trunk and the branches. For the leaves, which are more flexible, we chose a Young's modulus of 1.2GPa and a torsion modulus of 0.6GPa. The density was set to $1000 kg/m^3$. Our results show a convincing movement of the tree geometry in the wind. Furthermore, the simulation stays stable even under large wind forces. The average time taken by the position correction for the tree simulations is given in Table 1. Besides the fast simulation, a big advantage of our model compared to previous rod simulation approaches for Position-based dynamics is that we can use physically meaningful material parameters. The thickness of a branch automatically changes its stiffness. There is no need for tuning parameters for each branch by hand and simulations naturally show plausible behavior.

Comparison of the KKT/GS solver to the non-linear Gauss-Seidel solver: We compared the performance of the KKT/GS solver to the solution computed with the non-linear Gauss-Seidel solver of Macklin et al. [MMC16] in two scenes. In the first scene we tested the cantilever beam example from above with different discretizations. We ran the simulation for 60s to let the beam swing out of the initial state and come to rest. This part of the the simulation was performed using our direct solver at three iterations per time step. Thereafter, we ran 500000 KKT/GS iterations as well as 500000 non-linear Gauss-Seidel iterations (one GS iteration means one traversal through the whole matrix) and monitored the Chebyshev norm of the residual error in $\mathbf{b}$ (see Equation (22)). The results are depicted in Figure 9. After one iteration the error of the KKT/GS solver is about $2.2 \cdot 10^{-13}$, $2.8 \cdot 10^{-16}$, and $4.4 \cdot 10^{-18}$, and the error of the non-linear Gauss-Seidel solver is 10^{-1}, $5 \cdot 10^{-3}$, and $3.5 \cdot 10^{-4}$ for a discretization of 1000, 50, and 8 segments, respectively. In the first three iterations the error of the KKT/GS solver decreases even further to about $4.4 \cdot 10^{-16}$, $9.4 \cdot 10^{-19}$, and $5.9 \cdot 10^{-19}$. However, the non-linear Gauss-Seidel solver needs about 1000 and 50000 iterations for the beams with 8 and 50 segments, respectively, to arrive at an error value which is comparable to the error after only one KKT/GS iteration. The smallest error for the non-linear Gauss-Seidel solver with a 1000 segment beam is about $1.3 \cdot 10^{-4}$ after 500000 iterations. However, the direct solver used to compute the KKT subproblems increases the computational cost of one KKT/GS iteration compared to one non-linear Gauss-Seidel iteration. In the case of the cantilever beam, one iteration of the KKT/GS solver is 3.1 times more costly than one iteration of the non-linear Gauss-Seidel solver. To find out if the improved convergence is worth the increased cost, we compare the error after one KKT/GS step with the error after four Gauss-Seidel iterations. While the Gauss-Seidel solver still has an error of $1.4 \cdot 10^{-3}$, the error of our KKT/GS solver remains at merely $2.8 \cdot 10^{-16}$. It follows that the increased convergence of the KKT/GS solver outweighs the computational overhead by far.

The second test scene is the birch simulated in an increasing wind field. We simulated the scene in several runs. For each successive run we decreased the threshold for the Chebyshev norm of the residual error in $\mathbf{b}$. Note, that the stiffness of the simulated object decreases if the residual error in $\mathbf{b}$ is too large. Thus, a threshold that is too high will result in a perceptually wrong animation. Our own experiments have shown that an error threshold of $\eta = 10^{-6}$ is sufficient to reach a relative change in stiffness of about 3%. The best threshold value would be 10^{-9} after which the measured deviation in stiffness does not change significantly. Table 2 shows the required KKT/GS iterations and non-linear Gauss-Seidel iterations at different threshold values. The iteration count of the KKT/GS solver stays in a range of 1 to 5 iterations. The Gauss-Seidel solver on the other hand requires more than 200 iterations even for a threshold of 10^{-2}. Though, at this threshold the residual errors lead to a significant deviation of the simulation to the expected result.

For this scene one iteration of the KKT/GS solver is 5.6 times more costly than one iteration of the non-linear Gauss-Seidel solver. This factor is higher than for the cantilever beam due to the

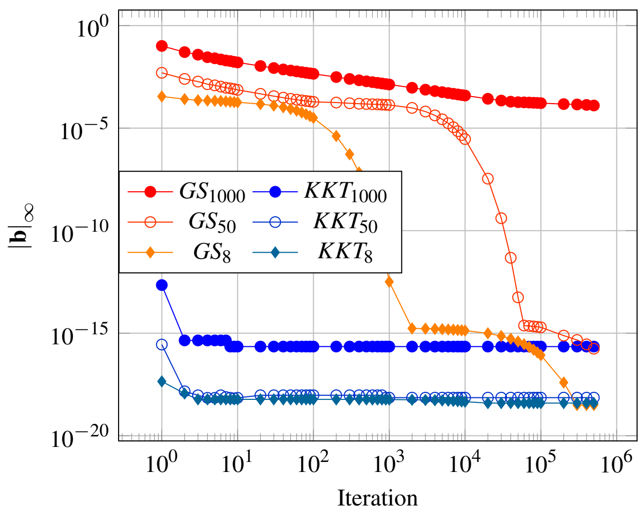

Figure 9: *Convergence of the Gauss-Seidel solver (GS) and KKT/GS (KKT) solver in the cantilever beam scene, plotted on a log-log scale. Measurements are shown for discretizations of the beam with 1000, 50, and 8 segments.*

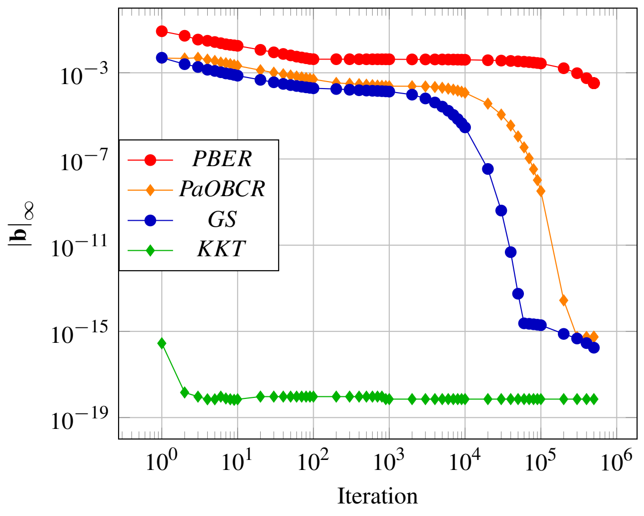

Figure 10: *Convergence of different simulation methods for the cantilever beam discretized with 50 segments, plotted on a log-log scale. The methods in comparison are Position-Based Elastic Rods (PBER), Position and Orientation Based Cosserat Rods (PaOBCR), the Gauss-Seidel (GS) solver and our KKT/GS (KKT) solver.*

increased complexity of the birch. The branching of the tree results in an increase of matrix-matrix products in our direct solver. Despite the even higher costs per iteration compared to the cantilever scene, the increased convergence leads to a speedup of two orders of magnitude for the birch scene (see Table 2 for the exact values).

	Average iterations		Pos. corr.
Error threshold η	KKT/GS	Gauss-Seidel	speedup
10^{-2}	1.49	208.73	25.5
10^{-6}	2.98	2055.23	124.8
10^{-9}	4.15	3629.69	164.6

Table 2: *Even for large error thresholds our KKT/GS solver needs significantly fewer iterations than the Gauss-Seidel solver. Not only does this compensate for the cost of the KKT/GS iterations, but it also speeds up the position correction by multiple orders of magnitude. Measurements are taken from the birch example.*

Comparison to previous methods: We compared our method to Position-based Elastic Rods (PBER) by Umetani et al. [USS14] and Position and Orientation Based Cosserat Rods (PaOBCR) by Kugelstadt and Schömer [KS16]. In order to make the comparison as fair as possible, we exchanged the PBD-stiffness parameter of both approaches with a compliant implementation for bending and twisting. Furthermore, we lumped the segment masses of our model to the particle masses of these approaches. In case of the approach of Kugelstadt and Schömer we applied the part of the inertia tensor of our segments which corresponds to a rotation around the axis orthogonal to the rod centerline to their quaternion weighting w_q. However, the approach of Umetani et al. describes no direct representation of inertia. Inertia is indirectly modeled by the weight distribution between particles and ghost particles. We distributed the lumped segment mass evenly between the two particles and one ghost particle representing a segment of the rod. For the following comparison we solved all constraints of the three different approaches with the non-linear Gauss-Seidel solver, even our own combined zero-stretch, bending and twisting constraint, to present convergence properties and mathematical effort. Furthermore, we compared these results to the performance of our KKT/GS solver approach. For this test we also employed the cantilever beam with 50 segments. As shown in Figure 10 PBER and PaOBCR converge slower than our combined zero-stretch, bending and twisting constraint. However, our combined constraint solved with the non-linear Gauss-Seidel solver is 13.9 times computationally more expensive than its cheapest competitor, PaOBCR. Even after comparing the cost of both methods after they reach the same error threshold PaOBCR ist cheaper. However, PaOBCR would loose its essential advantage of avoiding matrix inversions in a combination with a direct solver in a KKT/GS method. Furthermore, it is not clear how to combine the staggered distribution of kinematic values like position and orientation of the PaOBCR model with a direct solver for trees. Even though, PaOBCR is faster than our combined constraint solved with non-linear Gauss-Seidel, the KKT/GS method with our direct solver is by far the fastest approach. PaOBCR needs over 100000 iterations to decrease the order of magnitude in the residual error to a value that is reached by the KKT/GS method after only one iteration. Thereby PaOBCR is about 5890 times more expensive. Another advantage of our direct solver is that all constraints are solved at once, which means that instabilities arising from certain constraint orderings as reported by Umetani et al. [USS14] and Kugelstadt and Schömer [KS16] are completely avoided.

6. Conclusion

We presented a KKT/GS approach for the efficient simulation of stiff, inextensible elastic rods in the Position-based dynamics framework. It allows the simulation of hard constrains and elastic potentials with physically meaningful material parameters in a unified framework. Even though the solution of our KKT subproblems has higher computational costs than one iteration of nonlinear Gauss-Seidel, this is outweighed by the huge increase in convergence, so that large and stiff objects can be simulated with a speedup of two orders of magnitude. By using XPBD and the KKT/GS solver, our method does not have the major drawbacks of PBD, namely that the material stiffness does not depend on the number of iterations, the time step and the ordering of the constraints.

Our main drawback compared to some other methods in rod or tree simulation is that we often have to solve more than one system of linear equations per time step. This is a result of the particular time discretization of XPBD which, however, leads to increased stability. Furthermore, we chose XPBD intentionally to achieve a tight coupling of our rod simulations with the vast amount of solutions for different physical effects that have been published for the Position-based dynamics framework. In order to reduce the computational effort, we want to work on improving the performance of our direct solver. One of the most time consuming parts is the factorization of the system matrix. We want to develop an optimized solver in the future.

A minor limitation of our approach is that XPBD makes some simplifications in formulating the system for the KKT subproblems. Nevertheless, our method is numerically stable and delivers plausible results, like our comparison to the analytical solution has shown. In many interactive applications, like games, these features are more important than the highest physical accuracy. For the future we plan to investigate if we can modify our method, so that it works without the simplifications made in XPBD.

Acknowledgment

This work is supported by the German Research Foundation (DFG) under contract numbers BE 5132/3-1 and GSC 233/2, and by the 'Excellence Initiative' of the German Federal and State Governments and the Graduate School of Computational Engineering at TU Darmstadt.

References

[Ant05] ANTMAN S.: *Nonlinear Problems of Elasticity; 2nd ed.* Springer, Dordrecht, 2005. 3

[AX15] AUBRY J.-M., XIAN X.: Fast implicit simulation of flexible trees. In *Mathematical Progress in Expressive Image Synthesis II*. Springer, 2015, pp. 47–61. 2, 6, 8

[BAC*06] BERTAILS F., AUDOLY B., CANI M.-P., QUERLEUX B., LEROY F., LÉVÊQUE J.-L.: Super-helices for predicting the dynamics of natural hair. *ACM Trans. Graph. 25*, 3 (2006), 1180–1187. 3

[Bar96] BARAFF D.: Linear-time dynamics using Lagrange multipliers. In *International Conference on Computer Graphics and Interactive Techniques* (1996), ACM, pp. 137–146. 2, 4, 6

[BAV*10] BERGOU M., AUDOLY B., VOUGA E., WARDETZKY M., GRINSPUN E.: Discrete viscous threads. *ACM Transactions on Graphics 29*, 4 (2010), 116. 3

[Ber09] BERTAILS F.: Linear time super-helices. *Computer Graphics Forum 28*, 2 (2009), 417–426. 3, 8

[BET14] BENDER J., ERLEBEN K., TRINKLE J.: Interactive Simulation of Rigid Body Dynamics in Computer Graphics. *Computer Graphics Forum 33*, 1 (2014), 246–270. 2

[BMM14] BENDER J., MÜLLER M., MACKLIN M.: A Survey on Position-Based Simulation Methods in Computer Graphics. *Computer Graphics Forum 33*, 6 (2014), 228–251. 2, 3

[BPGK06] BOTSCH M., PAULY M., GROSS M. H., KOBBELT L.: Primo: coupled prisms for intuitive surface modeling. In *Symposium on Geometry Processing* (2006), no. EPFL-CONF-149310, pp. 11–20. 3

[BW98] BARAFF D., WITKIN A.: Large steps in cloth simulation. In *International Conference on Computer Graphics and Interactive Techniques* (1998), ACM, pp. 43–54. 2

[BWR*08] BERGOU M., WARDETZKY M., ROBINSON S., AUDOLY B., GRINSPUN E.: Discrete elastic rods. *ACM Trans. Graph. 27*, 3 (2008), 1–12. 3, 8

[CCK05] CHOE B., CHOI M. G., KO H.-S.: Simulating complex hair with robust collision handling. In *ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (2005), ACM, pp. 153–160. 2

[DCB14] DEUL C., CHARRIER P., BENDER J.: Position-based rigid-body dynamics. *Computer Animation and Virtual Worlds* (2014). 4

[GHF*07] GOLDENTHAL R., HARMON D., FATTAL R., BERCOVIER M., GRINSPUN E.: Efficient simulation of inextensible cloth. *ACM Transactions on Graphics 26*, 3 (2007). 2, 6

[GS07] GRÉGOIRE M., SCHÖMER E.: Interactive simulation of one-dimensional flexible parts. *Computer-Aided Design 39*, 8 (2007), 694–707. 3

[Had06] HADAP S.: Oriented strands: dynamics of stiff multi-body system. In *ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (2006), Eurographics Association, pp. 91–100. 3

[HGCO11] HERNÁNDEZ F., GARRE C., CASILLAS R., OTADUY M. A.: Linear-time dynamics of characters with stiff joints. In *Proceedings of V Ibero-American Symposium on Computer Graphics* (2011). 2, 6

[HH13] HAN D., HARADA T.: Tridiagonal matrix formulation for inextensible hair strand simulation. In *Proceedings of Virtual Reality Interactions and Physical Simulations (VRIPhys)* (2013), The Eurographics Association. 2

[HKW09] HABEL R., KUSTERNIG A., WIMMER M.: Physically guided animation of trees. *Computer Graphics Forum 28*, 2 (2009), 523–532. 8

[IMP*13] IBEN H., MEYER M., PETROVIC L., SOARES O., ANDERSON J., WITKIN A.: Artistic simulation of curly hair. In *ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (2013), ACM, pp. 63–71. 2

[KDB16] KOSCHIER D., DEUL C., BENDER J.: Hierarchical hp-adaptive signed distance fields. In *ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (2016), Eurographics Association. 7

[KS16] KUGELSTADT T., SCHÖMER E.: Position and orientation based Cosserat rods. In *ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (2016), Eurographics Association, pp. 169–178. 4, 10

[LL09] LANG H., LINN J.: *Lagrangian field theory in space-time for geometrically exact Cosserat rods*. ITWM, Fraunhofer Inst. Techno-und Wirtschaftsmathematik, 2009. 3

[LLA11] LANG H., LINN J., ARNOLD M.: Multi-body dynamics simulation of geometrically exact Cosserat rods. *Multibody System Dynamics 25*, 3 (2011), 285–312. 3

[MCK12] MÜLLER M., CHENTANEZ N., KIM T.-Y.: Fast simulation of inextensible hair and fur. In *Proceedings of Virtual Reality Interactions and Physical Simulations (VRIPhys)* (2012), Eurographics Association. 3

[MHHR07] MÜLLER M., HEIDELBERGER B., HENNIX M., RATCLIFF J.: Position based dynamics. *Journal of Visual Communication and Image Representation 18*, 2 (2007), 109–118. 3, 6

[MMC16] MACKLIN M., MÜLLER M., CHENTANEZ N.: XPBD: Position-based simulation of compliant constrained dynamics. In *Proceedings of the 9th International Conference on Motion in Games* (2016), MIG '16, ACM, pp. 49–54. 2, 5, 6, 9

[MMS15] MICHELS D. L., MUELLER J. P. T., SOBOTTKA G. A.: A physically based approach to the accurate simulation of stiff fibers and stiff fiber meshes. *Computers & Graphics 53* (2015), 136–146. 2

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

[QYH*17] QUIGLEY E., YU Y., HUANG J., LIN W., FEDKIW R.: Real-time interactive tree animation. *IEEE Transactions on Visualization and Computer Graphics* (2017). 3

[SBT07] SPILLMANN J., BECKER M., TESCHNER M.: Non-iterative computation of contact forces for deformable objects. *Journal of WSCG* (2007). 3

[SL08] SERVIN M., LACOURSIÈRE C.: Rigid body cable for virtual environments. *IEEE Transactions on Visualization and Computer Graphics 14*, 4 (2008), 783–796. 3

[SLF08] SELLE A., LENTINE M., FEDKIW R.: A mass spring model for hair simulation. *ACM Transactions on Graphics 27*, 3 (2008), 64:1–64:11. 2

[ST07] SPILLMANN J., TESCHNER M.: C o R d E: Cosserat rod elements for the dynamic simulation of one-dimensional elastic objects. In *ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (2007), Eurographics Association, pp. 63–72. 3, 5

[ST08] SPILLMANN J., TESCHNER M.: An adaptive contact model for the robust simulation of knots. *Computer Graphics Forum 27*, 2 (2008), 497–506. 3

[TNGF15] TOURNIER M., NESME M., GILLES B., FAURE F.: Stable Constrained Dynamics. *ACM Transactions on Graphics* (2015), 1–10. 2, 6

[USS14] UMETANI N., SCHMIDT R., STAM J.: Position-based Elastic Rods. In *ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (2014), pp. 21–30. 3, 10

[ZB13] ZHAO Y., BARBIČ J.: Interactive authoring of simulation-ready plants. *ACM Transactions on Graphics 32*, 4 (2013), 1–12. 8