EUROGRAPHICS 2017 / L. Barthe and B. Benes
(Guest Editors)

Volume 36 (2017), Number 2

Geometric Stiffness for Real-time Constrained Multibody Dynamics

Sheldon Andrews[1,2] and Marek Teichmann[2] and Paul G. Kry[1]

[1]McGill University
[2]CM Labs Simulations

Figure 1: *A heavy vehicle attached to a nearly inextensible cable is dropped from the end of a crane. The simulation involves very large mass ratios, a heterogeneous collection of joints, and remains stable at a time step of* $1/60$ *s. Cable dynamics are well preserved by our adaptive damping method.*

Abstract

This paper focuses on the stable and efficient simulation of articulated rigid body systems for real-time applications. Specifically, we focus on the use of geometric stiffness, which can dramatically increase simulation stability. We examine several numerical problems with the inclusion of geometric stiffness in the equations of motion, as proposed by previous work, and address these issues by introducing a novel method for efficiently building the linear system. This offers improved tractability and numerical efficiency. Furthermore, geometric stiffness tends to significantly dissipate kinetic energy. We propose an adaptive damping scheme, inspired by the geometric stiffness, that uses a stability criterion based on the numerical integrator to determine the amount of non-constitutive damping required to stabilize the simulation. With this approach, not only is the dynamical behavior better preserved, but the simulation remains stable for mass ratios of 1,000,000-to-1 at time steps up to 0.1 s. We present a number of challenging scenarios to demonstrate that our method improves efficiency, and that it increases stability by orders of magnitude compared to previous work.

Categories and Subject Descriptors (according to ACM CCS): I.6.8 [Computer Graphics]: Simulation and Modeling/Types of Simulation—Animation

1. Introduction

Physics simulation is a cornerstone of many real-time applications such as video games, character animation tools, operator training, and robotics control. Maintaining stable behavior and real-time frame rates is a priority. Typically, this requires tuning physical parameters such as mass, stiffness, and damping in order to produce stable behavior across a broad range of simulation states. A consequence is that accuracy is often sacrificed (e.g., by introducing non-constitutive damping or using unrealistic mass ratios) and the tuning process requires a large amount of manual effort. Furthermore, users and application designers in these domains are often not physics simulation experts, and there is strong motivation for automatic methods that reduce the knowledge required to improve stability.

The constraint stabilization technique proposed by Tournier et al. [TNGF15] uses the geometric stiffness to reduce transverse vi-

brations that occur when simulating dynamical systems with stiff constraints. This allows for time steps and mass ratios that are several orders of magnitude above the range supported by standard single-step methods. However, applying this technique in a typical rigid body simulation introduces several numerical issues that affect efficiency and can compromise real-time frame rates. Furthermore, the intended physical behavior of the simulation can be adversely affected by dissipative forces.

In this paper, we propose modifications to the stable constrained dynamics approach of Tournier et al. [TNGF15], making it numerically appealing for the standard linear solvers used by many physics engines. Furthermore, we reinterpret the geometric stiffness as a *transient spring*, which briefly influences the simulation over a single time step, and a stability criterion is used to determine how much non-constitutive damping is required to stabilize the simulation, if any. The contributions of our work are as follows:

- an efficient algorithm that uses low rank updates to construct the linear system for the dynamical equations of motion, which includes the geometric stiffness matrix;
- a diagonal approximation of the geometric stiffness matrix that attempts to preserve mechanical work properties;
- an adaptive damping scheme that is derived from stability analysis and uses the approximate diagonal matrix;
- reinterpretation of the geometric stiffness as a transient spring and applying its effects as a constraint damping term;
- derivation of the geometric stiffness matrix for a library of joints commonly used in articulated rigid body simulations, as well as their low rank decompositions.

The remainder of this section outlines the notation used in this paper and provides more details about the motivation for this work. Section 2 gives a summary of related work and background material on stable physics simulation. In Section 3, an efficient method is presented for computing the geometric stiffness matrix of constraints commonly used in articulated rigid body simulation. Section 4 contains a stability analysis wherein the geometric stiffness is reinterpreted as a spring, giving an undamped harmonic oscillator. This results in an automatic method to compute damping coefficients that stabilize transverse oscillations. Simulation results and comparisons of our method with previous work are presented in Section 5. The paper concludes in Section 6. Finally, the geometric stiffness matrices for numerous articulated constraints and their low rank decompositions are provided in the appendices.

1.1. Notation

Table 1 provides an overview of variables and notation used throughout this paper. The diagram in Figure 2 shows the coordinate frames and offsets used to specify the configuration of a joint. The attachment basis and offsets are fixed to the bodies. The offset relative to the center of mass of body i is specified by s_i, which is represented in a global coordinate frame but specified by the local vector s_i' such that $s_i = R_i s_i'$. Similarly, the orthogonal basis $\{u_i, v_i, n_i\}$ are vectors in the global coordinate frame, but specified by the local basis $\{u_i', v_i', n_i'\}$ and transformed by R_i. This dependence on the body orientation is relevant for deriving the geometric stiffness matrices in Appendix A. Unless otherwise stated, all vector quantities are represented in a global coordinate frame.

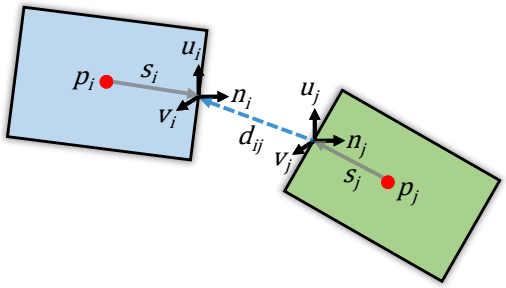

Figure 2: *Body i (light blue) and body j (light green) are coupled by a joint with attachment offsets s_i, s_j, coordinate frames $\{u_i, v_i, n_i\}$ and $\{u_j, v_j, n_j\}$, and displacement vector d_{ij}.*

p_i	position of body i
θ_i	orientation of body i
$\dot{p}_i$	linear velocity of body i
ω_i	angular velocity of body i
x_i	spatial configuration of body i, such that $x_i = \begin{pmatrix} p_i^T & \theta_i^T \end{pmatrix}^T$
$\mathbf{x}$	configuration of all bodies as $\begin{pmatrix} x_0^T & x_1^T & \dots & x_n^T \end{pmatrix}^T$
$\mathbf{v}$	velocity of all bodies as $\begin{pmatrix} \dot{p}_0^T & \omega_0^T & \dots & \dot{p}_n^T & \omega_n^T \end{pmatrix}^T$
R_i	rotation matrix corresponding to the orientation θ_i
s_i'	joint attachment offset in local frame
s_i	joint attachment offset in the global frame, $s_i = R_i s_i'$
ϕ	a position level constraint equation
J	a constraint Jacobian matrix, such that $J = \frac{d\phi}{d\mathbf{x}}$
$\mathbf{J}$	Jacobian matrix for all constraints
$\tilde{\mathbf{K}}$	geometric stiffness matrix
$\hat{\cdot}$	skew-symmetric cross product matrix of a vector
$\square_+$	an implicit term or quantity

Table 1: *Variables and notation used throughout this paper.*

1.2. Geometric Stiffness

The geometric stiffness is a tensor encoding variations in the constraint force directions, and has the form

$$\tilde{\mathbf{K}} = \frac{\partial \mathbf{J}^T}{\partial \mathbf{x}} \lambda. \tag{1}$$

Here, $\mathbf{J}$ is the matrix of constraint Jacobians, λ are the constraint forces, and $\mathbf{x}$ are the positions and orientations of bodies in the simulation. Tournier et al. [TNGF15] introduce this term as an implicit stiffness in the velocity-level discretization of the constrained dynamical equations

$$\begin{pmatrix} \tilde{\mathbf{M}} & -\mathbf{J}^T \\ \mathbf{J} & \frac{1}{h^2}\mathbf{C} \end{pmatrix} \begin{pmatrix} \mathbf{v}_+ \\ h\lambda_+ \end{pmatrix} = \begin{pmatrix} \mathbf{p} + h\mathbf{f} \\ -\frac{1}{h}\phi \end{pmatrix}, \tag{2}$$

where $\tilde{\mathbf{M}} = \mathbf{M} - h^2\tilde{\mathbf{K}}$. In this formulation, $\mathbf{p} = \mathbf{M}\mathbf{v}$ is the current momentum, $\mathbf{f}$ contains the external forces, the diagonal regularization matrix $\mathbf{C}$ makes the constraints compliant, and h is the time step size. Forming the Schur complement of the upper left block gives the reduced system

$$\left(\frac{1}{h^2}\mathbf{C} + \mathbf{J}\tilde{\mathbf{M}}^{-1}\mathbf{J}^T\right) h\lambda_+ = -\frac{1}{h}\phi - \mathbf{J}\tilde{\mathbf{M}}^{-1}(\mathbf{p} + h\mathbf{f}). \tag{3}$$

This form is used by many open source and commercial rigid body physics engines, and typically requires solving a mixed linear complementarity problem (MLCP), since the system may include both bilateral and unilateral constraints. Inclusion of the geometric stiffness in Eq. 2 and Eq. 3 may seem trivial, but it leads to numerical difficulties when solving the linear systems. These challenges relate to symmetry, positive definiteness, and efficiency.

Symmetry and positive definiteness. Efficiently solving the dense linear system in Eq. 3 involves using a Cholesky factorization. This requires $\tilde{\mathbf{M}}$ to be symmetric and positive definite; this is similarly a requirement for the system in Eq. 2. However, the inclusion of $\tilde{\mathbf{K}}$ can result in a non-symmetric matrix. Tournier et al. [TNGF15] suggest that a symmetric approximation of the matrix may be formed with little effect on the stabilizing properties. However, the resulting matrix may still fail to meet the positive definite requirement of the numerical method. Teran et al. [TSIF05] describe a technique for enforcing positive definiteness by clamping eigenvalues of a diagonalized elastic deformation gradient to be non-negative. This leads to robust simulations but can be costly, making it unsuitable for real-time or performance focused applications.

Efficiency. Solving the reduced system in Eq. 3 requires forming the inverse of $\tilde{\mathbf{M}}$. For a block diagonal mass matrix, which is typically the case, this is done efficiently by inverting the linear mass and performing a fast $3x3$ inversion of the inertia for each body. However, with the inclusion of geometric stiffness, inverting the full mass matrix is no longer straightforward due to nonzero terms that appear outside the block diagonal.

To overcome these limitations, we examine alternate formulations of the geometric stiffness and its inclusion in the multibody equations. Specifically, we demonstrate that the inverse of $\tilde{\mathbf{M}}$ may be obtained efficiently using low rank updates in Section 3. Furthermore, in Section 4 the geometric stiffness matrix is diagonalized as part of a stability analysis. This allows the reduced system in Eq. 3 to be solved without the previously described problems.

2. Related work

Simulation of constrained rigid body systems is an important problem in computer animation, and fast or approximate methods have been a research topic since the early days of computer graphics. In early work, Hahn [Hah88] presented an iterative technique for dealing with contact constraints, while Barzel and Barr [BB88] solve constraint forces for a variety of constraints. Subsequent work focused on efficient solutions for constrained rigid body systems [SZ90, Bar96].

It can be desirable to formulate and solve articulated systems in a minimal set of coordinates, such as joint coordinates [Fea14]. Nevertheless, while a few physics engines implement joint coordinate solvers, the more common case is that of constrained rigid body solvers, perhaps because of its simplicity and modularity. Time stepping a constrained full body formulation results in constraint violations, and thus all solvers include either a Baumgarte feedback term [Bau72], a post step [AP98, CP03], or fast iterative manifold projection [GHF*07]. Unfortunately, when time steps, velocities, constraint forces, or mass ratios are large, simulations can exhibit large constraint errors and become unstable.

Baraff and Witkin [BW98] use implicit integration in cloth simulation to address the stability issues with large time steps, and they even suggest that a backward Euler step with a linearized version of the system can be sufficient for good behavior. Goldenthal et al. [GHF*07], in contrast, take a step and project approach, which they show to be equivalent to having implicit constraint forces, making it possible to simulate effectively inextensible cloth systems with an iterative projection at each time step.

As an alternative to iterative methods, Tournier et al. [TNGF15] present a formulation that requires only one linear solve, while providing a good approximation to the full non-linear solve in comparison to the traditional constrained multi-body formulation. They achieve this by including the geometric stiffness, which informs how the forces in the system change due to motion, such as the rotation of links in a chain. The simple formulation relies on using forces from the previous time step, and can work very well when these forces change smoothly over time. One is likely to notice similarities to sequential quadratic programming (SQP) [NW06] where the second derivative of the constraint Lagrangian is typically used to solve for the optimal step. However, Tournier et al. emphasize that their focus is a single linear solve with compliant constraints. This is our focus too, with additional objectives: efficiency of implementation, stability at a wider range of mass ratios, and an adaptive approach to avoid excessive numerical damping.

With respect to adaptive simulation, Servin et al. [SLNB11] present a stability criterion for simulation of cable systems. Links are adaptively lengthened based on the tension in the cable in order to reduce the harmonic frequency, bringing it within a stable range. Our stability analysis is similar, but applied to a broader class of systems. Furthermore, the adaptive damping scheme that we describe in this paper leaves the topology of mechanical structure unchanged. Hewlett et al. [HKC*17] observed that the geometric stiffness introduces large mechanical dissipation and propose a control parameter to modulate its effects by monitoring energy drift in the system. Similarly, our approach uses an adaptive scheme to mitigate non-constitutive energy dissipation, but the criterion is based on the stability properties of a numerical integrator and is applied to individual degrees of freedom (DOF).

3. Inverse by low rank updates

In this section, we present an efficient method for constructing the inverse of $\tilde{\mathbf{M}}$ using low rank updates. The unmodified mass matrix may be efficiently inverted due to the nearly diagonal form of $\mathbf{M}$. However, inverting the augmented mass matrix requires more computational effort (e.g., an LU decomposition) due to off-diagonal elements in the geometric stiffness matrix.

We observe that for many articulated rigid body constraints the geometric stiffness matrix is sparse and low rank. These properties can be exploited to build the augmented mass matrix more efficiently. The crux of this approach is based on the Sherman-Morrison formula [SM50].

3.1. Decomposition for a simple joint

Consider a simple articulated joint, the ball-and-socket. The geometric stiffness matrix for this joint is

$$\tilde{\mathbf{K}}_{\text{bs}} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & \hat{\lambda}\hat{s}_i & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\hat{\lambda}\hat{s}_j \end{pmatrix} \in \mathbb{R}^{12\times 12}, \tag{4}$$

where $\lambda \in \mathbb{R}^3$ are the Lagrange multipliers enforcing the constraint. Obviously, this matrix is sparse and has a simple structure. To find the rank decomposition of $\tilde{\mathbf{K}}_{\text{bs}}$, we examine one of the non-zero blocks in Eq. 4 and notice that s is in the right null space of the matrix $\hat{\lambda}\hat{s}$ (the body subscript is dropped for brevity). Starting with the unit vector $\vec{s} = \frac{s}{\|s\|} \in \mathbb{R}^3$, we construct an orthonormal basis, where

$$\vec{s} \perp \vec{t}, \quad \vec{r} \perp \vec{t}, \quad \vec{r} \perp \vec{s},$$
$$\|\vec{s}\| = \|\vec{t}\| = \|\vec{r}\| = 1.$$

The column space of $\hat{\lambda}\hat{s}$ spans the subspace formed by vectors $\vec{r}$ and $\vec{t}$. The vector of Lagrange multipliers can be projected onto this basis, giving the coefficients

$$\lambda_r = \lambda^T \vec{r}$$
$$\lambda_s = \lambda^T \vec{s}$$
$$\lambda_t = \lambda^T \vec{t}.$$

Notice also that $\vec{r}$, $\vec{s}$, and $\vec{t}$ are in the row space of matrix $\hat{\lambda}\hat{s}$. Its decomposition is therefore

$$\hat{\lambda}\hat{s} = \|s\| \left(\lambda_r \vec{s}\vec{r}^T - \lambda_s \vec{r}\vec{r}^T + \lambda_t \vec{s}\vec{t}^T - \lambda_s \vec{t}\vec{t}^T \right), \tag{5}$$

which is a rank-2 matrix. This decomposition can be applied to both non-zero 3×3 blocks in Eq. 4, and finally used to reconstruct the full matrix $\tilde{\mathbf{K}}_{\text{bs}}$.

Knowing the rank decomposition of the matrix $\tilde{\mathbf{K}}_{\text{bs}}$, and having already computed the inverse mass matrix $\mathbf{M}^{-1}$ by efficient methods, the inverse of $\tilde{\mathbf{M}}$ is computed by sequentially applying the Sherman-Morrison formula.

3.2. Sherman-Morrison formula

Given a rank-1 matrix ab^T, the Sherman-Morrison formula may be used to update the inverse mass $\mathbf{M}^{-1}$ matrix by

$$(\mathbf{M} + ab^T)^{-1} = \mathbf{M}^{-1} - \frac{\mathbf{M}^{-1} a b^T \mathbf{M}^{-1}}{1 + b^T \mathbf{M}^{-1} a}.$$

The augmented mass matrix $\tilde{\mathbf{M}}^{-1}$ may therefore be obtained by sequentially applying rank-1 updates using the geometric stiffness decomposition of all joints in the system.

For example, consider an update to the block of body i using $\lambda_{r_i} \vec{s}_i \vec{r}_i^T$, which is the first term in Eq. 5. We define vectors $\bar{a}, \bar{b} \in \mathbb{R}^{6n}$ with dimensionality matching an n body system. These vectors are zero except entries

$$\bar{a}_{6i+3\ldots6i+5} = \lambda_{r_i} \vec{s}_i$$
$$\bar{b}_{6i+3\ldots6i+5} = \vec{r}_i.$$

These are the 3D basis vectors padded with zeros according to the block's location in the system matrix. A low rank update of the inverse mass matrix $\mathbf{M} \in \mathbb{R}^{6n\times 6n}$ may then be performed by

$$\mathbf{M}^{-1} \leftarrow \mathbf{M}^{-1} + h^2 \|s\| \frac{\mathbf{M}^{-1} \bar{a} \bar{b}^T \mathbf{M}^{-1}}{1 + \bar{b}^T \mathbf{M}^{-1} \bar{a}}. \tag{6}$$

Note the scaling factors $-h^2$ and $\|s\|$ which is based on the inclusion of the geometric stiffness in the dynamical equations and length of the attachment offset s, respectively. Similar updates may be performed using the rank-1 matrices $\vec{s}\vec{t}^T$, $\vec{r}\vec{r}^T$, and $\vec{t}\vec{t}^T$, and for both bodies. In our implementation, the matrix-vector products of Eq. 6 are computed efficiently by exploiting the sparsity of the vectors. The process to incrementally construct the inverse augmented mass matrix is similar for other types of joints. The low rank decompositions for joints used by many articulated rigid body simulators may be found in Appendix B.

However, there is still no guarantee that the linear systems in Eq. 2 and Eq. 3 will be positive definite. In the next section, a method for diagonalizing $\tilde{\mathbf{K}}$ is proposed that results in a positive definite matrix for the linear system. The diagonalized approximation is then used as part of a stability analysis where damping is introduced only when there is an indication that forces related to the geometric stiffness may cause instability

4. Geometric stiffness inspired damping

Recall that $\tilde{\mathbf{K}}$ does not represent a material property of the physical system, but that it is convenient to interpret the geometric stiffness as a physical spring that is able to generate forces in directions transverse to the constraints. This reinterpretation is useful, since it facilitates analysis of the geometric stiffness as part of a mass-spring system of undamped harmonic oscillators.

In this section, we examine the case of a simple harmonic oscillator. We consider the stability requirements for simulating this system with a semi-implicit integration scheme, otherwise known as symplectic Euler, and then ultimately apply these requirements within the less strict approximately implicit setting. We derive a stability criterion for stable simulation and extended it to the case of a dynamical system where spring forces are generated by the geometric stiffness, with the $\tilde{\mathbf{K}}$ matrix serving as an indicator of when the system may become unstable. Thus, damping is introduced to stabilize the system using an adaptive method, which estimates a damping coefficient that is just large enough to stabilize the system.

4.1. Stability analysis of harmonic oscillator

Consider the behavior a 1D Hookean mass-spring-damper attached to a particle. The accelerations generated by the system are

$$\ddot{x} = (-kx - b\dot{x})/m, \tag{7}$$

where x is the displacement from rest length, $\dot{x}$ is the particle velocity, k is the stiffness, b is the damping, and m is the mass of the particle. Simulating this system with a semi-implicit integrator and time step h gives the velocity and position update

$$\dot{x} \leftarrow \dot{x} + h\ddot{x},$$
$$x \leftarrow x + h\dot{x}.$$

Substituting Eq. 7 and representing the updates in matrix form gives the phase space equation

$$\begin{pmatrix} x \\ \dot{x} \end{pmatrix} \leftarrow \underbrace{\begin{pmatrix} 1-h^2\frac{k}{m} & h-h\frac{b}{m} \\ -h\frac{k}{m} & 1-h\frac{b}{m} \end{pmatrix}}_{\mathbf{P}} \begin{pmatrix} x \\ \dot{x} \end{pmatrix} \tag{8}$$

with the eigenvalues of matrix $\mathbf{P}$ given by

$$-\frac{h^2k+bh-2m \pm \sqrt{h^4k^2+2bh^3k+b^2h^2-4h^2km}}{2m}.$$

If the magnitude of either eigenvalue is greater than 1, the simulation can become unstable. We note that Müller et al. [MHTG05] arrived at similar stability conditions in their work, but for an explicit integrator.

Inspection of Eq. 8 indicates that multiple parameters affect simulation stability. The mass, stiffness, and time step are typically fixed due to application or model requirements. Therefore we consider the case that only damping may be used to stabilize the system. Introducing non-constitutive damping is a common practice to stabilize physical simulations. However, manual tuning of damping coefficients is a cumbersome task since values must be selected that minimally affect the dynamics, yet leave the simulation stable for a wide range of configurations. Rather, we devise an automatic method for computing the damping based on the stability region of a semi-implicit integrator.

As is popular in computer graphics applications, we are using the single step backward Euler method with a linear approximation of the constraint Jacobian and mass matrix. The backward Euler method is well known to be very stable, with a region of stability strictly containing that of the semi-implicit integrator used in the analysis above. That is, semi-implicit Euler at best exhibits absolute stability within the negative real half of the complex plane, while backward Euler has absolute stability for all but a small region in the positive real half of the complex plane [AP98]. Our strategy is to develop a stability criterion based on an integrator with a smaller region of stability, and then apply this to the implicit integrator used in our experiments. This criterion comes in the form of a threshold based on the behavior of simple 1D undamped oscillator.

Stability threshold. A stability threshold is determined by setting $b = 0$ in the eigenvalue equations, giving an undamped oscillator. This results in the inequality

$$\frac{h^2k - \sqrt{h^4k^2-4h^2km}}{m} \leq 4, \tag{9}$$

which is true if the eigenvalue magnitudes are less than 1. Servin et al. [SLNB11] derive a similar threshold in their analysis of cable systems, although with a different approach. For an undamped oscillator with Verlet integration, they propose a stability threshold of $h^2\omega^2 \leq 4$, where $\omega^2 = \frac{k}{m}$ is the oscillation frequency of a cable. Our analysis corroborates their finding, since the square root term in Eq. 9 tends to be small.

Alpha parameter. The stability inequality used in our work is augmented with a parameter α, which is a positive scalar value that allows control over how close the system must be to the boundary before damping is required, giving

$$\frac{h^2k}{m} \leq 4\alpha. \tag{10}$$

If the inequality in Eq. 10 is violated, then the damping required for stability is found by solving for b at the boundary. This is computed directly as

$$b = \frac{h^2k - 4\alpha m}{2h}. \tag{11}$$

For a spring system with constant mass and stiffness, the value of b is also constant. However, if the spring parameters are estimated from the geometric stiffness, the damping values will change at each time step. The stability threshold also varies with the constraint forces. For the purposes of stabilizing a general class of physics simulation, the stability threshold is evaluated at each time step and damping values are computed according to Eq. 11, giving an adaptive damping scheme.

Extending this analysis directly to the matrix $\tilde{\mathbf{K}}$ is non-trivial. The geometric stiffness is representative of a complex system of transient springs. Modal analysis requires finding eigenvalues of the matrix $\mathbf{M}^{-1}\tilde{\mathbf{K}}$ which is computationally costly. Therefore, an approximation of the matrix is preferred in which the stability analysis is computationally cheap. Conveniently, if $\tilde{\mathbf{K}}$ is diagonal, the damping scheme for stabilizing a simple harmonic oscillator is applicable. The next section gives details on how we compute a diagonal approximation to simplify the stability analysis.

4.2. Diagonalizing the $\tilde{\mathbf{K}}$ matrix

Our goal is to compute a simplified matrix for use in the stability criterion. Recall the interpretation of geometric stiffness as a transient spring. In building the diagonal approximation $\mathbf{K}_\mathrm{d}$, we therefore consider the mechanical work done by $\tilde{\mathbf{K}}$. The instantaneous transfer of mechanical energy occurring over a single time step, or *instantaneous work*, done by the geometric stiffness is $0.5\,\mathbf{v}^T\tilde{\mathbf{K}}\mathbf{v}$.

We propose that our diagonal approximation can represent a spring that tends to do at least the same amount of mechanical work, by providing a stiffer system in the directions of any typical $\mathbf{v}$. That is, the damping calculated by the stability equation could fail to stabilize the simulation if a less stiff system is considered. An approximation that tends to upper bound the work is computed by assigning each diagonal element $\mathbf{K}_{\mathrm{d}i,i}$ the norm of the corresponding column vector in $\tilde{\mathbf{K}}$, such that

$$\mathbf{K}_{\mathrm{d}i,i} = \|\tilde{\mathbf{K}}_i\|. \tag{12}$$

We observe that this approximation bounds the instantaneous work in all of our experiments. Figure 3 demonstrates this observation. For rotational joints, such as the hinge, the diagonal approximation tends to match the instantaneous work closely, whereas for examples modeled using the flexible cable joint the instantaneous work is usually overestimated, but achieves our goal of upper bounding the work done by $\tilde{\mathbf{K}}$.

With a diagonal stiffness matrix the stability analysis is now straightforward. In an inertial coordinate frame, it becomes a matter

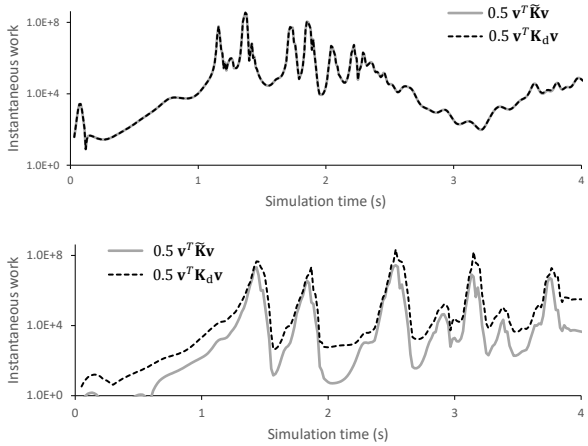

Figure 3: *Instantaneous work at each frame using the original geometric stiffness matrix and the diagonal approximation to simulate the heavy ball on cable example shown in Figure 8 using hinge joints (top) and the flexible cable joint (bottom).*

of assigning the mass and stiffness values

$$m = \mathbf{M}_{i,i}$$
$$k = \mathbf{K}_{\mathrm{d}i,i}$$

and applying Eq. 11 to compute entry $\mathbf{B}_{i,i}$ of the diagonal damping matrix. This is done for all inertial DOFs, although the damping coefficients corresponding to translational DOFs are sometimes discarded (as discussed in Section 5.4). The augmented mass matrix is then rebuilt as

$$\tilde{\mathbf{M}} = \mathbf{M} + h\mathbf{B}, \tag{13}$$

which is positive definite since $\mathbf{M}$ is positive definite and $\mathbf{B}$ contains only non-negative values along the diagonal. Here, $\mathbf{B}$ can be seen as replacing the geometric stiffness matrix, and since the damping coefficients augment the mass and inertia of bodies in the system we refer to this as the *inertial damping* formulation.

4.3. Projection to constraint space

Alternatively, damping may be applied to the constraint rows of the linear system. This stabilizes in directions that are aligned with the constraint axes, and in some cases the simulations appear more plausible when damping is applied to the the constraint equations. This can be achieved by mapping the mass and diagonalized geometric stiffness matrix to the constraint space and performing the stability analysis there.

The mass $\mathbf{M}$ and diagonal approximation $\mathbf{K}_{\mathrm{d}}$ are mapped onto each constraint row by

$$m' = (J\mathbf{M}^{-1}J^T)^{-1}$$
$$k' = (J\mathbf{K}_{\mathrm{d}}^{-1}J^T)^{-1},$$

where J is the Jacobian for a single constraint equation, and m' and k' are the effective mass and stiffness, respectively, for the constraint. Coupling of constraints through the mass and stiffness matrix is ignored here, and rather than form the effective stiffness and mass using the full constraint matrix, J is simply a row vector. This results in a single scalar value for m' and k'. Stability analysis is performed using m' and k' by applying Eq. 11 to compute a damping coefficient, b', for the constraint. However, a constraint damping term is not considered by the dynamical equations in Eq. 2. We therefore modify the formulation to support constraint damping.

4.4. Constraint space damping

The constraint equations are reorganized as a force equation for a spring-damper system

$$\lambda_+ = -\mathbf{C}^{-1}\phi_+ - \mathbf{B}'\dot{\phi}_+, \tag{14}$$

where $\mathbf{B}'$ is a diagonal matrix of non-negative damping coefficients, which are assembled from the b' computed for each constraint row. Substituting $\mathbf{J}\mathbf{v}_+ = \dot{\phi}_+$ and letting $\phi_+ = \phi + h\dot{\phi}_+$, the constraint equations become

$$\mathbf{J}\mathbf{v}_+ + \Sigma\lambda_+ = -\Gamma\frac{\phi}{h}, \tag{15}$$

where non-zero entries of the diagonal matrices Σ and Γ are computed as

$$\Gamma_{i,i} = \frac{1}{(1+h^{-1}\mathbf{C}_{i,i})\mathbf{B}'_{i,i}}, \tag{16}$$

$$\Sigma_{i,i} = \frac{h^{-2}\mathbf{C}_{i,i}}{(1+h^{-1}\mathbf{C}_{i,i})\mathbf{B}'_{i,i}}. \tag{17}$$

This is related to how constrained multibody simulators often allow the combination of compliant constraints and Baumgarte stabilization to be interpreted as an implicit stiff spring and damper.

5. Results and discussion

This section evaluates the methods proposed by this paper. We start by examining the computational gains from using the low rank matrix inversion technique described in Section 3. Challenging scenarios that demonstrate the robustness of our adaptive damping are also presented. All results were obtained using an Intel Core i7 3.3 GHz CPU and a single thread. Unless otherwise stated, the time step used for experiments is $h = 0.01$ s. Our methods have been implemented as part of the Vortex [VOR16] physics engine. The linear system in Eq. 3 is used for all simulations and solved using a standard Cholesky decomposition. For comparisons with [TNGF15], the same linear system is solved by LU decomposition.

The examples shown in Figure 8 and Figure 9 use the constraint space damping technique, whereas all other examples are stabilized by inertial damping. The accompanying video shows interactive simulations for the examples discussed in this section and indicates which of the two damping techniques is used. All videos were captured in real-time. Where a mouse spring is used to interact with the simulation, the spring stiffness is scaled proportional to the mass of the selected object.

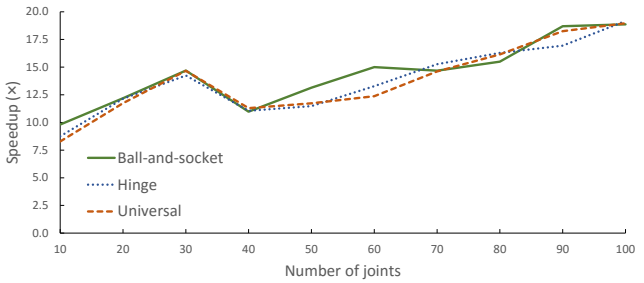

Figure 4: *The performance gain when using low rank updates to compute $\tilde{\mathbf{M}}^{-1}$. For angular joints, the gain appears to scale linearly with the number of joints.*

	Number of joints		
Method	10	50	100
Full rank	0.099 ms	6.795 ms	49.668 ms
Low rank	0.010 ms	0.517 ms	2.633 ms
Gain	9.81×	13.13×	18.87×

Table 2: *The average time to compute $\tilde{\mathbf{M}}^{-1}$ for a cable using various numbers of ball-and-socket joints .*

5.1. Performance of low rank updates

We present three examples to evaluate the computational speedup obtained by using our low rank updates.

Articulated chains. The computation time of the low rank updates and the full rank matrix inversion is compared in Figure 4. Articulated chains involving various numbers of joints and joint types are simulated, and the computation time required to compute $\tilde{\mathbf{M}}^{-1}$ is measured. As indicated by Table 2, the performance gain grows as the number of joints in the system increases. Likewise, the gain is consistent across hinge, universal, and ball-and-socket joints.

Strong robot. An example involving a heterogeneous collection of joints is shown Figure 5. The robot character is modeled using compliant constraints, including 5 ball-and-sockets, 11 universal, and 10 hinge joints. Each foot of the robot is anchored to the ground using a hinge joint, and each fingertip is attached to the end of the cable using a ball-and-socket joint. Masses range from 0.1 kg for the finger bodies to 15,000 kg for the vehicle attached to the opposite end of the cable. The vehicle is driven by a motorized prismatic joint to move it away from the robot. The DOFs in the robot are controlled using implicit PD servos with a set point matching the original pose of the character. This scenario resembles a character animation sequence that might occur in a video game, with many different types of joints and large mass ratios. Nevertheless, the simulation remains stable and the motion of the character is smooth.

Flexible cable joint. A special joint is used to model the cable for the scene depicted in Figure 5. The joint uses several dot-2 constraints, for which the geometric stiffness matrix is relatively dense (see Appendix A). Figure 6 shows that the performance gains quickly diminish as the number of flexible joints is increased, and eventually result in longer computation times. Simulations involving only the special joint indicate that our low rank method never improves performance for this type of joint. This is a limitation of our approach. Performance results from these simulations are shown in Figure 7.

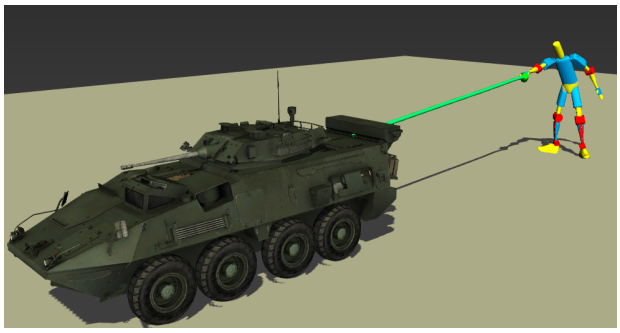

Figure 5: *A character with super strength prevents a tank from driving away. This scene involves many different joints, including hinges, ball-and-sockets, prismatic, and flexible cable joints.*

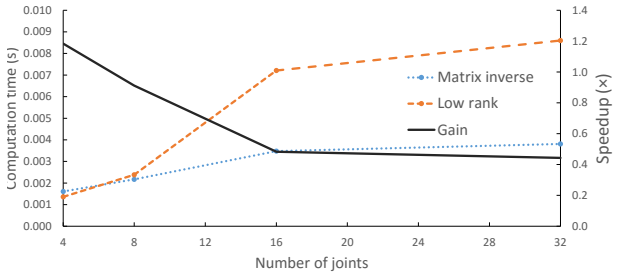

Figure 6: *Performance gains using the low rank inverse for the scene shown in Figure 5. The geometric stiffness matrix for the cable joints is not sparse, and as a result performance gains are diminished as the number of these joints is increased.*

5.2. Validation

To evaluate our adaptive dissipation method, we present experiments that monitor the energy of the system, vibrations in chains, and we explore the range of damping coefficients for systems with various mass ratios.

Kinetic energy. Simulation of a pendulum swinging under gravity exchanges potential energy and kinetic energy. If friction and other dissipative elements in the simulation are nominal, the total mechanical energy should be nearly constant. Figure 8 shows the kinetic energy per time step of a cable with 100 kg load and pendulum motion. The flexible cable joints have an axial stiffness of 10^{12} and bending and torsional stiffness of 10. Using the technique of Tournier et al., dissipation caused by geometric stiffness causes the peak kinetic energy to decrease by nearly 40% after 40 seconds of simulation time. When the stability criterion with constraint damping is used, the peak kinetic energy drops by only 2% after the same amount of time, indicating that the mechanical energy in the system is better preserved.

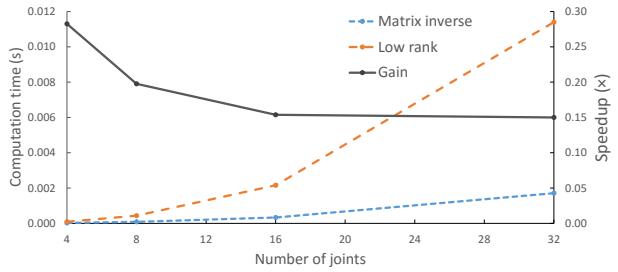

Figure 7: *Performance gain and computation time for cables simulated using various numbers of flexible cable joints. The speedup from the low rank updates is mitigated due to the non-sparse structure of the geometric stiffness matrix.*

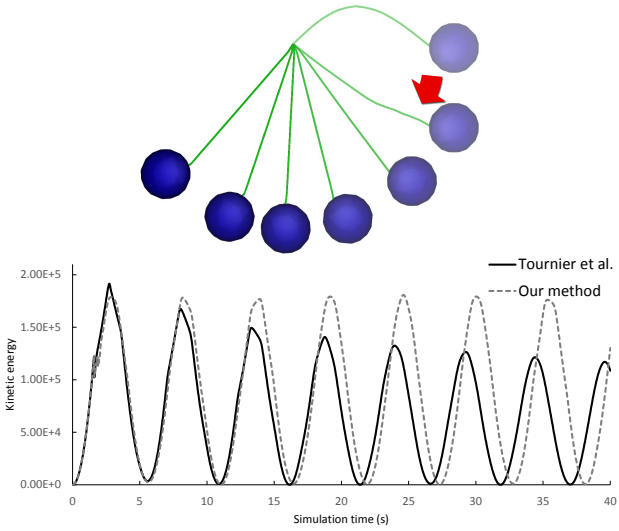

Figure 8: *A 100 kg load attached to a flexible cable is dropped and begins swinging with a pendulum motion (top). The plot (bottom) shows the kinetic energy when simulated using the technique of Tournier et al. and our method.*

Cable vibrations. In Figure 9 the vibrations of a cable are visualized when an attached heavy load (in this case 5000 kg) is dropped straight down and reaches its apex. The simulated behavior with very small time steps and no geometric stiffness or damping is provided for comparison. The vibrations seen when the adaptive damping method from Section 4 is used better represent the observed oscillations in the high fidelity simulation, and without the dynamical behavior appears to be over damped.

Damping coefficients. The plot in Figure 10 shows the damping coefficients of a selected cable link for the example shown in Figure 8. The mass is increased from 10 kg to 100 kg to 1000 kg. This clearly demonstrates that our damping algorithm adapts to constraint forces at the current simulation frame. For smaller masses, damping is only required when the load reaches its apex (around frame 160) and constraint forces are largest. As the load increases, damping is required more often to stabilize the system.

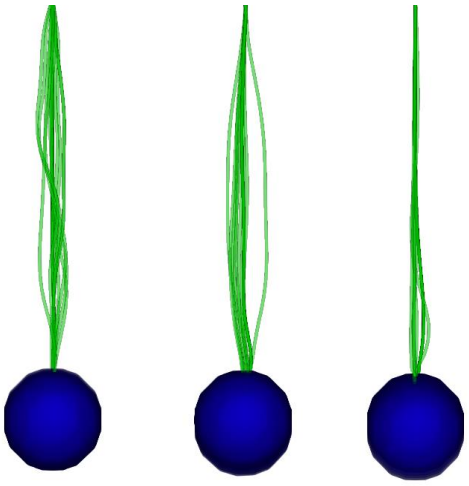

Figure 9: *Vibrations in a cable when integrated with a small time step and no damping $h = 10^{-5}$ s (left), and large time step $h = 0.01$ s with the adaptive damping (middle) and without (right). Dynamical behavior is clearly much better preserved with the adaptive scheme.*

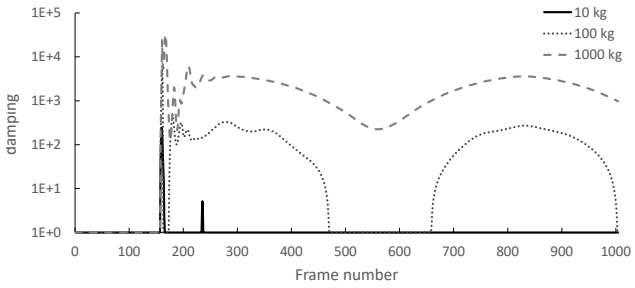

Figure 10: *For the cable simulation in Figure 8 the damping coefficients of a selected link are shown at each time step. For less massive objects (10 kg) the stabilization damping is zero for most of the simulation. As the mass increases, more damping is required as constraint forces increase.*

5.3. Other results

Finally, we present a few challenging scenarios with large mass ratios and large time steps, in which our approach is shown to be effective by producing stable and interactive simulations.

Crane with heavy load. The teaser in Figure 1 shows a 15,000 kg vehicle attached to a cable being dropped from a crane. The cable is nearly inextensible, with the axial stiffness of 10^{10}. The mass ratio in the system is similarly large, ranging from 100,000 kg for the boom to 0.2 kg for individual cable segments. Despite these challenging parameters, the simulation is stable when integrated using a time step of $1/60$ s. The adaptive damping method helps to preserve the rich cable dynamics, and is applied to the angular velocities of the simulation bodies with $\alpha = 1.0$. The articulated linkage of the crane is modeled using hinge joints, whereas the cable uses relaxed rigid joints.

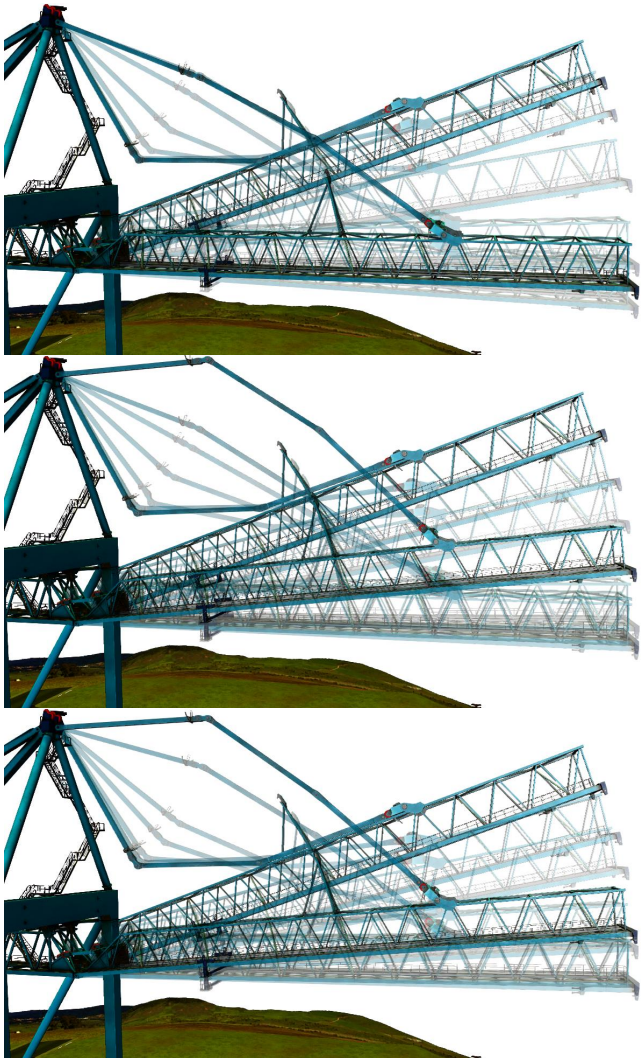

Figure 11: *An overlay of selected frames from a crane simulation. The* $100,000$ *kg boom is elevated* 20° *and then dropped. The simulation remains stable for* 0.1 *s (top),* 0.01 *s (middle) and* 0.001 *s time steps.*

Varying time steps. The damping coefficient computed by our method is dependent on the constraint forces and masses of bodies in the simulation, as we have already shown. However, Eq. 11 suggests that the damping coefficient is also dependent on the time step. Figure 11 visualizes motion of the crane boom shown in the teaser when it is dropped from a 20° angle of elevation. Despite large mass ratios and stiff constraints, the simulation remains stable at time steps of 0.1 s, 0.01 s, and 0.001 s. Also, the dynamical behavior in each case is similar, although for the largest time step value the simulation does appear to be more damped and exhibits less high frequency dynamics.

Constraint space vs. inertial damping. For cable simulations, we observe artifacts when using the inertial damping formulation. Specifically, there is severe dissipation about the torsional axis. The supplementary video shows a comparison of stabilizing a cable simulation using the inertial and constraint space damping techniques. A heavy ball is given an initial angular velocity about the torsional axis and then dropped. The local coordinate frame of each cable segment is drawn, and as the simulation progresses the twist formed in the cable that uses constraint space stabilization is more plausible.

5.4. Discussion and limitations

The constraint space damping technique described in Section 4.3 tends to work well for the 6D rigid and flexible cable joints. A drawback of this technique is that the rank of $\mathbf{J}\mathbf{K}_\mathrm{d}\mathbf{J}^T$ must equal the rank of $\mathbf{K}_\mathrm{d}$. In other words, if stabilizing forces are in the null space of the constraint Jacobian, the projection removes them and damping cannot occur in certain directions. In this case, the constraint space may be augmented by additional constraint rows with negligible compliance values. This may be done automatically at each time step, or manually when the simulation model is being designed. Experimented with the latter scenario have shown positive results.

An interesting observation is that many articulated simulations are stabilized by damping only the rotational DOFs. Inspecting the geometric stiffness matrices in Appendix A reveals that the stabilization affects only the angular velocities of the constraint bodies. For many applications, such as physics-based character animation, this is good news since typically only hinges, universal, and ball-and-socket joints are used to model characters. However, we observed that for mechanisms using the dot-2 constraint, such as the prismatic and the flexible cable joint, applying damping to the translational degrees of freedom results in a behavior where bodies appear to move through a viscous fluid. In these cases, we simply discard the translational damping coefficient and set $\mathbf{B}_{i,i} = 0$ for these DOFs. This helps to prevent viscosity artifacts, especially for cable simulation, but has similar stabilizing properties. The accompanying video highlights this result.

6. Conclusion

This paper proposes several novel methods for improving the performance and stability of articulated multi-body simulations with stiff constraints. The geometric stiffness and augmented mass matrices are formed efficiently using low rank updates. A diagonalization scheme allows an approximation of the geometric stiffness matrix to be computed that is convenient to analyze. Using the geometric stiffness in its diagonal form, a stability criterion is used to adaptively introduce a minimal amount of damping a three fold benefit: i) the system matrix remains positive definite, permitting more efficient numerical solvers; ii) the effort of manually tuning artificial damping coefficients is greatly reduced; and iii) stable simulation systems with unprecedented stiffnesses and mass ratios is possible at real-time frame rates.

Acknowledgements

We thank the anonymous reviewers for their suggestions to improve the paper. This work was supported in part by funding from NSERC.

Appendix A: Constraint library

This section contains derivations of the geometric stiffness for a basic set of constraints that are commonly used in articulated rigid body simulation. Each derivation considers the case of a two body system where the geometric stiffness is represented by a 12×12 matrix. For succinctness, the 3×3 block structure of these matrices is exploited.

Basic partial derivatives

The instantaneous angular velocity ω of a body and the time derivative of its rotation matrix $\dot{R}$ are related by

$$\hat{\omega} = \dot{R}R^{-1}. \tag{18}$$

We can reinterpret this lemma for small angular displacements $\partial\theta$ such that

$$\partial\hat{\theta} = dRR^{-1},$$

and post-multiplying by R gives

$$\partial\hat{\theta}R = dR. \tag{19}$$

Furthermore, for a vector $n \in \mathbb{R}^3$ attached to a rotating body such that $n = Rn'$, the spatial derivative of the vector is given by

$$\frac{\partial n}{\partial \theta} = \hat{n}. \tag{20}$$

Eq. 19 and Eq. 20 are the building blocks used to derive the geometric stiffness equations for a basic constraint library.

For additional mathematical details of rigid body kinematics, please see the comprehensive work by Murray et al. [MLSS94].

Ball-and-socket joint

A ball-and-socket, or spherical, joint connecting bodies i and j constrains two points on the bodies to have the same position while allowing relative angular motion. The constraint equation can be written as

$$\phi_{\text{bs}} = p_i + s_i - p_j - s_j = 0.$$

Rearranging Eq. 18 gives $\dot{R} = \hat{\omega}R$, and so the time derivative of the constraint equation can be written as

$$\dot{\phi}_{\text{bs}} = \dot{p}_i - \hat{s}_i\omega_i - \dot{p}_j + \hat{s}_j\omega_j,$$

giving the constraint Jacobian matrix

$$J_{\text{bs}} = \begin{pmatrix} I & -\hat{s}_i & -I & \hat{s}_j \end{pmatrix}.$$

In other words,

$$\dot{\phi}_{\text{bs}} = J_{\text{bs}} \begin{pmatrix} \dot{p}_i \\ \omega_i \\ \dot{p}_j \\ \omega_j \end{pmatrix}.$$

The constraint forces acting on the two bodies are

$$J_{\text{bs}}^T\lambda_{\text{bs}} = \begin{pmatrix} \lambda_{\text{bs}} \\ \hat{s}_i\lambda_{\text{bs}} \\ -\lambda_{\text{bs}} \\ -\hat{s}_j\lambda_{\text{bs}} \end{pmatrix}, \tag{21}$$

where $\lambda_{\text{bs}} \in \mathbb{R}^3$ are the Lagrange multipliers for the ball-and-socket constraint. Differentiating Eq. 21 with respect to the spatial configuration of the bodies and applying the chain rule gives

$$\frac{\partial J_{\text{bs}}^T\lambda_{\text{bs}}}{\partial x} = J_{\text{bs}}^T\frac{\partial\lambda_{\text{bs}}}{\partial x} + \frac{\partial J_{\text{bs}}^T}{\partial x}\lambda_{\text{bs}},$$

where the first term is zero and the second term is the geometric stiffness matrix of the joint, or $\tilde{\mathbf{K}}_{\text{bs}} = \frac{\partial J_{\text{bs}}^T}{\partial x}\lambda_{\text{bs}}$. Expanding this into 3×3 blocks gives

$$\tilde{\mathbf{K}}_{\text{bs}} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & \hat{\lambda}_{\text{bs}}\hat{s}_i & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\hat{\lambda}_{\text{bs}}\hat{s}_j \end{pmatrix}. \tag{22}$$

Noting that $s_i = R_i s_i'$ and $s_j = R_j s_j'$, Eq. 22 matches the matrix provided in Tournier et al. [TNGF15].

Axial lock

An axial lock constrains the angular motion about a single axis. The constraint Jacobian matrix is the 12 component row vector

$$J_n = \begin{pmatrix} 0 & n_i^T & 0 & -n_i^T \end{pmatrix}$$

where the locked axis of rotation n_i is fixed to body i. The geometric stiffness for this constraint is

$$\tilde{\mathbf{K}}_{\text{ax},n} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & -\lambda_{\text{ax}}\hat{n}_i & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & \lambda_{\text{ax}}\hat{n}_i & 0 & 0 \end{pmatrix}. \tag{23}$$

Universal joint

The universal, or Hooke, joint has similar behavior to the spherical joint, but removes a rotational degree of freedom by keeping an axis fixed to body i perpendicular fixed to body j. The constraint equations for this joint can be written as

$$\phi_{\text{bs}} = p_i + s_i - p_j - s_j = 0$$
$$\phi_{\text{d1},u} = n_i^T u_j = 0$$

where n_i is a unit vector in coordinate frame of body i which should remain orthogonal to unit vector u_j in coordinate frame of body j. Since the derivation of ϕ_{bs} is the same as previously discussed, we instead focus on the *dot-1* constraint ϕ_{d1}. The Jacobian matrix of the dot-1 constraint is

$$J_{\text{d1},u} = \begin{pmatrix} 0 & (\hat{n}_i u_j)^T & 0 & -(\hat{n}_i u_j)^T \end{pmatrix}. \tag{24}$$

Note that for vectors $a, b \in \mathbb{R}^3$ that $\hat{a}b = -\hat{b}a$ and $(\hat{a}b)^T = -b^T\hat{a}$. The geometric stiffness for the dot-1 constraint is

$$\tilde{\mathbf{K}}_{\text{d1},u} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & \eta & 0 & -\eta^T \\ 0 & 0 & 0 & 0 \\ 0 & -\eta & 0 & \eta^T \end{pmatrix} \tag{25}$$

where $\eta = \lambda_{\text{d1}}\hat{u}_j\hat{n}_i$ and the scalar value $\lambda_{\text{d1}} \in \mathbb{R}$ is the Lagrange multiplier of the dot-1 constraint at the previous time step. The geometric stiffness for the universal joint may be assembled as

$$\tilde{\mathbf{K}}_{\text{un}} = \tilde{\mathbf{K}}_{\text{bs}} + \tilde{\mathbf{K}}_{\text{d1},u}.$$

Hinge joint

The hinge, or revolute, joint allows a relative rotation of two bodies about a single axis. The joint behaves much like a universal, but with an additional dot-1 constraint equation

$$\begin{aligned}\phi_{\text{bs}} &= p_i + s_i - p_j - s_j = 0\\ \phi_{\text{d1},u} &= n_i^T u_j = 0\\ \phi_{\text{d1},v} &= n_i^T v_j = 0\end{aligned}$$

where v_j is a unit vector in coordinate frame of body j which is orthogonal to u_j and is constrained to be perpendicular to n_i. The geometric stiffness $\tilde{\mathbf{K}}_{\text{d1},v}$ for the dot-1 constraint between vector v_j and n_i is found by applying Eq. 25 and substituting v_j for u_j. The hinge geometric stiffness is then assembled as

$$\tilde{\mathbf{K}}_{\text{hi}} = \tilde{\mathbf{K}}_{\text{bs}} + \tilde{\mathbf{K}}_{\text{d1},u} + \tilde{\mathbf{K}}_{\text{d1},v}.$$

Prismatic joint

A prismatic joint allows a relative translation between two bodies along a single axis. The constraint equations for this joint can be written as

$$\begin{aligned}\phi_{\text{d1},u} &= n_i^T u_j = 0\\ \phi_{\text{d1},v} &= n_i^T v_j = 0\\ \phi_{\text{d1},n} &= u_i^T n_j = 0\\ \phi_{\text{d2},u} &= d_{ij}^T u_i = 0\\ \phi_{\text{d2},v} &= d_{ij}^T v_i = 0,\end{aligned}$$

where $d_{ij} = p_i + s_i - p_j - s_j$. This joint consists of three dot-1 constraints (top rows), plus equations $\phi_{\text{d2},u}$ and $\phi_{\text{d2},v}$, which are *dot-2* constraints. A single Lagrange multiplier is used for each dot-2 constraint, which together eliminate relative translation of the bodies along any axes perpendicular to n_i.

Using $\phi_{\text{d2},u}$ as an example, which constrains relative motion in the direction u_i, the Jacobian matrix is

$$J_{\text{d2},u} = \begin{pmatrix} u_i^T & u_i^T(\hat{d}_{ij} - \hat{s}_i) & -u_i^T & u_i^T \hat{s}_j \end{pmatrix} \tag{26}$$

with the constraint Jacobian for $\phi_{\text{d2},v}$ having a similar form. The geometric stiffness matrix for $J_{\text{d2},u}$ is

$$\tilde{\mathbf{K}}_{\text{d2},u} = \lambda_{\text{d2},u} \begin{pmatrix} 0 & \hat{u}_i^T & 0 & 0 \\ \hat{u}_i & (\hat{d}_{ij} - \hat{s}_i)\hat{u}_i & \hat{u}_i^T & (\hat{s}_j \hat{u}_i)^T \\ 0 & \hat{u}_i & 0 & 0 \\ 0 & \hat{s}_j \hat{u}_i & 0 & -(\hat{s}_j \hat{u}_i)^T \end{pmatrix}. \tag{27}$$

The matrix $\tilde{\mathbf{K}}_{\text{d2},v}$ may be found by applying Eq. 27 and substituting v_i for u_i. The geometric matrix for the prismatic joint is computed as

$$\tilde{\mathbf{K}}_{\text{pr}} = \tilde{\mathbf{K}}_{\text{d1},u} + \tilde{\mathbf{K}}_{\text{d1},v} + \tilde{\mathbf{K}}_{\text{d1},n} + \tilde{\mathbf{K}}_{\text{d2},u} + \tilde{\mathbf{K}}_{\text{d2},v}.$$

Rigid joint

The prismatic joint is augmented with a sixth constraint equation that restricts the translational motion of the bodies, effectively constraining the rotational and translational motion of the two bodies to be the same. This is done using an additional dot-2 constraint

$$\phi_{\text{d2},n} = d_{ij}^T n_i = 0.$$

The geometric stiffness matrix of the rigid joint is then computed as

$$\tilde{\mathbf{K}}_{\text{6D}} = \tilde{\mathbf{K}}_{\text{pr}} + \tilde{\mathbf{K}}_{\text{d2},n}.$$

Flexible cable joint

By choosing small or moderate compliances for the rigid joint, the constraint rows are relaxed and the joint becomes flexible. Compliance and damping values are then selected for each of the three rotational and three translational constrained degrees of freedom to match the behavior of an extensible cable. For example, Eq. 16 and Eq. 17 are tuned in this way for the cable results shown in Section 5.

Appendix B: Low rank decompositions

This section provides the low rank decompositions of the basic constraint library in Appendix A. All factorizations consider the case of a two body system with 12×12 geometric stiffness matrix, although decompositions for larger systems are possible.

Ball-and-socket

The low rank factorization of $\tilde{\mathbf{K}}_{\text{bs}}$ follows from the analysis in Section 3.1. The vectors $\vec{r}, \vec{s}, \vec{t}$ and the Lagrange multipliers $\lambda_r, \lambda_s, \lambda_t$ are used with subscripts to indicate the body index. Using this notation, the decomposition is as follows:

$$\begin{aligned}\bar{s}_i &= \begin{pmatrix} 0 & \vec{s}_i^T & 0 & 0 \end{pmatrix}^T\\ \bar{r}_i &= \begin{pmatrix} 0 & \vec{r}_i^T & 0 & 0 \end{pmatrix}^T\\ \bar{t}_i &= \begin{pmatrix} 0 & \vec{t}_i^T & 0 & 0 \end{pmatrix}^T\\ \bar{s}_j &= \begin{pmatrix} 0 & 0 & 0 & \vec{s}_j^T \end{pmatrix}^T\\ \bar{r}_j &= \begin{pmatrix} 0 & 0 & 0 & \vec{r}_j^T \end{pmatrix}^T\\ \bar{t}_j &= \begin{pmatrix} 0 & 0 & 0 & \vec{t}_j^T \end{pmatrix}^T\\ \tilde{\mathbf{K}}_{\text{bs}} &= \|s_i\|(\lambda_{r_i} \bar{s}_i \bar{r}_i^T - \lambda_{s_i} \vec{r}_i \vec{r}_i^T + \lambda_{t_i} \bar{s}_i \bar{t}_i^T - \lambda_{s_i} \bar{t}_i \bar{t}_i^T)\\ &\quad - \|s_j\|(\lambda_{r_j} \bar{s}_j \bar{r}_j^T - \lambda_{s_j} \vec{r}_j \vec{r}_j^T + \lambda_{t_j} \bar{s}_j \bar{t}_j^T - \lambda_{s_j} \bar{t}_j \bar{t}_j^T).\end{aligned}$$

Axial constraint

The axial constraint decomposition has the following form:

$$\begin{aligned}\bar{u}_i &= \begin{pmatrix} 0 & u_i^T & 0 & 0 \end{pmatrix}^T\\ \bar{v}_i &= \begin{pmatrix} 0 & v_i^T & 0 & 0 \end{pmatrix}^T\\ \bar{u}_i' &= \begin{pmatrix} 0 & 0 & 0 & u_i^T \end{pmatrix}^T\\ \bar{v}_i' &= \begin{pmatrix} 0 & 0 & 0 & v_i^T \end{pmatrix}^T\\ \tilde{\mathbf{K}}_{\text{ax}} &= \lambda \left(-\bar{u}_i \bar{v}_i^T + \bar{v}_i \bar{u}_i^T + \bar{u}_i' \bar{v}_i^T - \bar{v}_i' \bar{u}_i^T \right).\end{aligned}$$

Dot-1 constraint

The decomposition for the dot-1 constraint follows the Jacobian definition from Eq. 24, using vectors n_i and u_j, but the decomposition is valid for other vector pairings. The low rank decomposition of the dot-1 geometric stiffness matrix is:

$$\bar{n} = \begin{pmatrix} 0 & 0 & 0 & n_i^T \end{pmatrix}^T$$
$$\bar{u} = \begin{pmatrix} 0 & -u_j^T & 0 & 0 \end{pmatrix}^T$$
$$\bar{n}' = \begin{pmatrix} 0 & -n_i^T & 0 & n_i^T \end{pmatrix}^T$$
$$\bar{u}' = \begin{pmatrix} 0 & -u_j^T & 0 & u_j^T \end{pmatrix}^T$$
$$\tilde{\mathbf{K}}_{\mathrm{d1},u} = \lambda(\bar{u}'\bar{n}^T + \bar{n}'\bar{u}^T).$$

Dot-2 constraint

Finally, for the dot-2 constraint we begin by forming an orthonormal basis using the vector $\vec{d}_{ij} = \frac{d_{ij}}{\|d_{ij}\|}$, such that

$$\vec{d}_{ij} \perp \vec{e}_{ij}, \quad \vec{e}_{ij} \perp \vec{f}_{ij}, \quad \vec{d}_{ij} \perp \vec{f}_{ij},$$
$$\|\vec{d}_{ij}\| = \|\vec{e}_{ij}\| = \|\vec{f}_{ij}\| = 1.$$

This basis, along with the orthonormal bases $\vec{r}_i, \vec{s}_i, \vec{t}_i$ and $\vec{r}_j, \vec{s}_j, \vec{t}_j$, is projected onto the constraint coordinate vectors v_i, and n_i giving the coefficients

$$\rho_{i,n} = \vec{r}_i^T n_i \qquad \rho_{i,v} = \vec{r}_i^T v_i \qquad \tau_{i,n} = \vec{t}_i^T n_i \qquad \tau_{i,v} = \vec{t}_i^T v_i$$
$$\rho_{j,n} = \vec{r}_j^T n_i \qquad \rho_{j,v} = \vec{r}_j^T v_i \qquad \tau_{j,n} = \vec{t}_j^T n_i \qquad \tau_{j,v} = \vec{t}_j^T v_i$$
$$\upsilon_n = \vec{f}_{ij}^T n_i \qquad \upsilon_v = \vec{f}_{ij}^T v_i \qquad \kappa_n = \vec{e}_{ij}^T n_i \qquad \kappa_v = \vec{e}_{ij}^T v_i.$$

We also define the following vectors:

$$\bar{n}_2 = \begin{pmatrix} 0 & n_i^T & 0 & 0 \end{pmatrix}^T \qquad \bar{v}_2 = \begin{pmatrix} 0 & v_i^T & 0 & 0 \end{pmatrix}^T$$
$$\bar{n}_{31} = \begin{pmatrix} -n_i^T & 0 & n_i^T & 0 \end{pmatrix}^T \qquad \bar{v}_{13} = \begin{pmatrix} v_i^T & 0 & -v_i^T & 0 \end{pmatrix}^T$$
$$\bar{n}_{42} = \begin{pmatrix} 0 & -n_i^T & 0 & n_i^T \end{pmatrix}^T \qquad \bar{v}_{24} = \begin{pmatrix} 0 & v_i^T & 0 & -v_i^T \end{pmatrix}^T$$
$$\bar{r}_i = \begin{pmatrix} 0 & \vec{r}_i^T & 0 & 0 \end{pmatrix}^T \qquad \bar{t}_i = \begin{pmatrix} 0 & \vec{t}_i^T & 0 & 0 \end{pmatrix}^T$$
$$\bar{r}_j = \begin{pmatrix} 0 & 0 & 0 & \vec{r}_j^T \end{pmatrix}^T \qquad \bar{t}_j = \begin{pmatrix} 0 & 0 & 0 & \vec{t}_j^T \end{pmatrix}^T$$
$$\bar{f}_{ij} = \begin{pmatrix} 0 & \vec{f}_{ij}^T & 0 & 0 \end{pmatrix}^T \qquad \bar{e}_{ij} = \begin{pmatrix} 0 & \vec{e}_{ij}^T & 0 & 0 \end{pmatrix}^T$$
$$\bar{c}_1 = -\rho_{j,n}\bar{t}_j + \tau_{j,n}\bar{r}_j \qquad \bar{d}_1 = -\rho_{j,v}\bar{t}_j + +\tau_{j,v}\bar{r}_j$$
$$\bar{c}_2 = \rho_{j,n}\bar{v}_2 - \rho_{j,v}\bar{n}_2 \qquad \bar{d}_2 = \tau_{j,n}\bar{v}_2 - \tau_{j,v}\bar{n}_2$$
$$\bar{c}_4 = \upsilon_n\bar{v}_2 - \upsilon_v\bar{n}_2 \qquad \bar{d}_4 = \kappa_n\bar{v}_2 - \kappa_v\bar{n}_2$$
$$\bar{c}_5 = \rho_{i,n}\bar{v}_2 - \rho_{i,v}\bar{n}_2 \qquad \bar{d}_5 = \tau_{i,n}\bar{v}_2 - \tau_{i,v}\bar{n}_2.$$

The low rank decomposition of $\tilde{\mathbf{K}}_{\mathrm{d2},u}$ becomes

$$\tilde{\mathbf{K}}_{\mathrm{d2},u} = \lambda\left(\bar{v}_{13}\bar{n}_2^T + \bar{n}_{31}\bar{v}_2^T + \bar{n}_2\bar{v}_{13}^T + \bar{v}_2\bar{n}_{31}^T\right)$$
$$+\lambda\|s_j\|\left(\bar{n}_{42}\vec{d}_1^T + \bar{v}_{24}\bar{c}_1^T + \bar{r}_j\vec{d}_2^T - \bar{t}_j\bar{c}_2^T\right)$$
$$+\lambda\|d_{ij}\|\left(\bar{f}_{ij}\vec{d}_4^T - \bar{e}_{ij}\bar{c}_4^T\right) - \lambda\|s_i\|\left(\bar{r}_i\vec{d}_5^T - \bar{t}_i\bar{c}_5^T\right).$$

This decomposition is obviously the most complex of those we provide in this paper. We note the performance gain of the low rank updates is diminished when the dot-2 constraint is used.

References

[AP98] ASCHER U. M., PETZOLD L. R.: *Computer methods for ordinary differential equations and differential-algebraic equations*, vol. 61. Siam, 1998. 3, 5

[Bar96] BARAFF D.: Linear-time dynamics using lagrange multipliers. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1996), SIGGRAPH '96, ACM, pp. 137–146. 3

[Bau72] BAUMGARTE J.: Stabilization of constraints and integrals of motion in dynamical systems. *Computer methods in applied mechanics and engineering 1*, 1 (1972), 1–16. 3

[BB88] BARZEL R., BARR A. H.: A modeling system based on dynamic constraints. In *Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1988), SIGGRAPH '88, ACM, pp. 179–188. 3

[BW98] BARAFF D., WITKIN A.: Large steps in cloth simulation. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques* (1998), ACM, pp. 43–54. 3

[CP03] CLINE M. B., PAI D. K.: Post-stabilization for rigid body simulation with contact and constraints. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on* (2003), vol. 3, IEEE, pp. 3744–3751. 3

[Fea14] FEATHERSTONE R.: *Rigid body dynamics algorithms*. Springer, 2014. 3

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

[Hah88] HAHN J. K.: Realistic animation of rigid bodies. In *Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1988), SIGGRAPH '88, ACM, pp. 299–308. 3

[HKC*17] HEWLETT J., KOVACS L., CALLEJO A., KRY P., KOVECSES J., ANGELES J.: Adaptive semi-implicit integrator for articulated mechanical systems. *ASME Journal of Computational and Nonlinear Dynamics* (2017). 3

[MHTG05] MÜLLER M., HEIDELBERGER B., TESCHNER M., GROSS M.: Meshless deformations based on shape matching. *ACM Trans. Graph. 24*, 3 (July 2005), 471–478. 5

[MLSS94] MURRAY R. M., LI Z., SASTRY S. S., SASTRY S. S.: *A mathematical introduction to robotic manipulation*. CRC press, 1994. 10

[NW06] NOCEDAL J., WRIGHT S.: *Numerical optimization*. Springer Science & Business Media, 2006. 3

[SLNB11] SERVIN M., LACOURSIERE C., NORDFELTH F., BODIN K.: Hybrid, multiresolution wires with massless frictional contacts. *IEEE transactions on visualization and computer graphics 17*, 7 (2011), 970–982. 3, 5

[SM50] SHERMAN J., MORRISON W. J.: Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. *The Annals of Mathematical Statistics 21*, 1 (1950), 124–127. 3

[SZ90] SCHRÖDER P., ZELTZER D.: The virtual erector set: Dynamic simulation with linear recursive constraint propagation. In *ACM SIGGRAPH Computer Graphics* (1990), vol. 24, ACM, pp. 23–31. 3

[TNGF15] TOURNIER M., NESME M., GILLES B., FAURE F.: Stable constrained dynamics. *ACM Transactions on Graphics (TOG) 34*, 4 (2015), 132. 1, 2, 3, 6, 10

[TSIF05] TERAN J., SIFAKIS E., IRVING G., FEDKIW R.: Robust quasistatic finite elements and flesh simulation. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2005), ACM, pp. 181–190. 3

[VOR16] Vortex. http://www.cm-labs.com/, 2016. 6