EUROGRAPHICS 2014 / B. Lévy and J. Kautz
(Guest Editors)

*Volume 33 (2014), Number 2*

# Efficient Enforcement of Hard Articulation Constraints in the Presence of Closed Loops and Contacts

Robin Tomcin  Dominik Sibbing  Leif Kobbelt

RWTH Aachen, Computer Graphics Group

**Abstract**

*In rigid body simulation, one must distinguish between contacts (so-called unilateral constraints) and articulations (bilateral constraints). For contacts and friction, iterative solution methods have proven most useful for interactive applications, often in combination with Shock-Propagation in cases with strong interactions between contacts (such as stacks), prioritizing performance and plausibility over accuracy. For articulation constraints, direct solution methods are preferred, because one can rely on a factorization with linear time complexity for tree-like systems, even in ill-conditioned cases caused by large mass-ratios or high complexity. Despite recent advances, combining the advantages of direct and iterative solution methods wrt. performance has proven difficult and the intricacy of articulations in interactive applications is often limited by the convergence speed of the iterative solution method in the presence of closed kinematic loops (i.e. auxiliary constraints) and contacts.*

*We identify common performance bottlenecks in the dynamic simulation of unilateral and bilateral constraints and are able to present a simulation method, that scales well in the number of constraints even in ill-conditioned cases with frictional contacts, collisions and closed loops in the kinematic graph. For cases where many joints are connected to a single body, we propose a technique to increase the sparsity of the positive definite linear system. A solution to these bottlenecks is presented in this paper to make the simulation of a wider range of mechanisms possible in real-time without extensive parameter tuning.*

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

## 1. Introduction

This paper is concerned with the robust and efficient simulation of articulated rigid bodies with contacts and collisions. Our goal is to achieve real-time performance, as required by applications like training simulators and video games.

State-of-the-art physics libraries show excellent performance for scenes with vast amounts of contacts and collisions with friction. However, with respect to articulation constraints one is often limited to mechanisms with moderate complexity for performance reasons. Heavy large-scale dynamics are commonly simulated off-line and embedded kinematically to save computation time in a real-time simulation. For mechanisms with many components, such as the walking machine in Figure 6b (Strandbeest [Jan13]), however, this might not be a viable strategy if dynamic response to user interaction is desired.

Resolving bilateral constraints emerging from joint connections of articulated bodies can be achieved by solving linear systems involving a typically sparse matrix, the so-called system matrix $\mathbf{A}$ (derived in Section 3). Direct methods have proven to be superior to iterative methods in many practical applications involving the solution of large scale linear systems [BBK05] and these observations also apply to many of the cases discussed in this paper. However, we identified four common performance bottlenecks when using direct methods for articulation constraints. The major contribution of this paper is a novel solution method that addresses these bottlenecks. Moreover, it integrates seamlessly with the popular Gauss-Seidel Propagation approach for collisions and contacts, such that the dynamic real-time simulation of complex articulated bodies with collisions and contacts becomes practical without extensive parameter tuning.

**Bottleneck #1: Redundant constraints**

Redundant constraints (see Figure 1a) often appear in systems with loops in the kinematic graph and lead to a singular system matrix. A popular solution is to identify loop

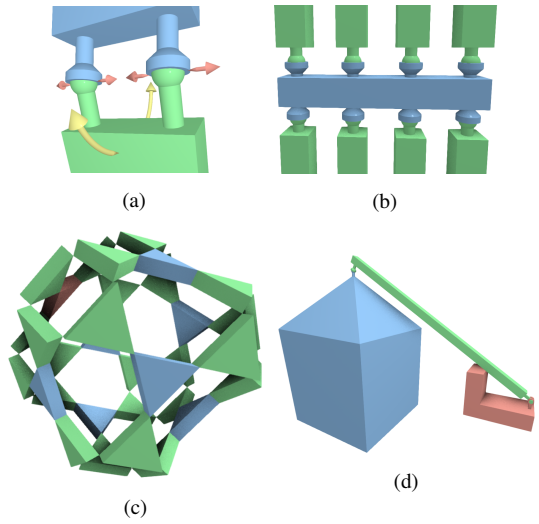



Figure 1: Illustration of cases leading to the four performance bottlenecks addressed in this paper. (a) A 1-DOF hinge joint is modeled with 2 spherical joints comprising 6 formal constraints which, by redundancy reduce to 5 actual constraints. Arrows indicate the two redundant constraints (red) and the remaining DOF (yellow). (b) Many joints connected to a single dynamic body cause a larger block of the system matrix to be dense. (c) A mechanism consisting of Cardan and revolute joints (see supplementary video). It has many loops in the kinematic graph which lead to redundant constraints and fill-in during the factorization. (d) A heavy dynamic body (blue) is connected to a red fixed body via a long green jointed rod. The convergence of an iterative contact resolution (between the red and green body) depends on the mass ratio of the blue and green body and the length of the green rod, adding a certain geometry-dependence.

closures (auxiliary constraints) and leave them out of the direct solution procedure. They are then resolved with a different strategy, e.g. iteratively, similar to contacts. However, this will cause the convergence speed to be mass and geometry dependent and in situations with ill-conditioning (e.g. large mass-ratios), the auxiliary constraints can have a spring-like behaviour or lead to poor performance. In addition, the choice of auxiliary constraints is not unique, which adds a certain bias that can lead to unexpected results.

In Section 4, we resolve this problem with a somewhat simpler regularization approach that relies on a Cholesky factorization of the resulting symmetric and positive definite (s.p.d.) systems and show, that this method yields a more predictable performance without compromising drift-free satisfaction of *all* articulation constraints.

**Bottleneck #2: Many constraints at a single body**

The system matrix contains a large dense block if many constraints are attached to a single body (see Figure 1b), leading to a slow factorization. A common remedy is the linear-time factorization of [Bar96], which relies on the solution of a larger, but always sparse formulation of the systems of equations. However, solving more general symmetric indefinite systems resulting from loops in the kinematic graph is prone to numerical instabilities and/or performance decrease. In addition, a regularization of symmetric indefinite matrices is computationally much heavier than for the positive definite case.

In Section 5, we therefore propose a technique to improve the sparsity of the s.p.d. formulation and provide a performance comparison.

**Bottleneck #3: Fill-in**

Systems with closed kinematic loops (Figure 1c) can not only lead to a singular system matrix, but also introduce additional nonzero matrix entries (so-called fill-in) during the factorization process. In Section 6 we show that well-known fill-in minimization techniques are effective in dealing with this problem, yielding good scaling for a huge class of mechanisms.

**Bottleneck #4: Coupling of contact and articulation**

For unilateral constraints like contacts and friction, direct methods such as the pivoting method introduced by Baraff [Bar94] are not as attractive to the computer graphics community for performance reasons. Recent advances in simulating huge contact groups such as stacks [GBF03] [WTF06] [Erl07] have shown that iterative methods in combination with shock-propagation scale well and plausible results can be achieved. However, Shock-Propagation shall not be the topic of this paper since we focus our attention to the interaction between contacts and articulation. In Section 7 we show that performance gains can be achieved by coupling each contact, one at a time, directly with the involved articulation constraints. This way, we eliminate the dependence of the iterative solution method on mass ratios and geometric properties of articulations (see Figure 1d).

**Our approach**

For collision and contact resolution, we use a pre-stabilization approach (see [GBF03], [WTF06], [BFS05], [Ben07] and Section 3), i.e. we iteratively compute impulses based on predictions of joint- and contact states. This eliminates the problem of constraint drift, because the desired states of constraints are targeted on a position-level.

Recently, Smith et. al. defined 5 desired properties [SKV*12] for a physically accurate model that captures simultaneous impacts. While they use an interior-point quadratic programming solver (see also the staggered projections approach of [KSJP08]), we decided to rely on a more light-weight Gauss-Seidel like propagation model similar to [GBF03] [WTF06] and [BFS05] and accept the resulting symmetry violations for impacts. Plausible results for our applications can usually be achieved, as synthetic scenarios like a perfectly symmetric pool break are rare in interactive applications.

To make the following extensions better accessible and our paper more self-contained, we shortly discuss the over-

all setup in Section 3. The main contributions of this paper are related to the bottlenecks #2 and #4 (Sections 5 and 7) and some aspects are described in detail in the Appendix. In Section 7 we showcase the methodical advantages of our numerical approach such as facilitating larger simulation steps.

For the experiments presented throughout this paper, we used CholMod [CDHR08] to solve symmetric positive definite systems. Recent advances in parallelization of direct solvers for sparse matrices have shown that an iso-efficiency of $\Omega(p^{1.5})$ can be achieved for matrices arising from 3D-domains [GKG09]. From version 2.0.0 on, the CholMod Solver has GPU computing capabilities, but they were turned off in our examples for the purpose of comparison. Although for cases with many single bodies (or simple articulated bodies) the collision detection and contact propagation dominate the computation time, efficient parallel methods exist for these scenarios [Har11].

## 2. Previous work

Rigid body simulation has a long history in computer graphics, dating back to the 1980s [AG85], [Hah85] [MW88] and has its roots in mechanical engineering and robotics.
For pure articulation constraints, it became apparent that direct solution methods yield best performance. There are two contrary approaches:

Reduced coordinates eliminate constraints recursively before the simulation and describe the system configuration with kinematically independent variables. These approaches achieve linear time complexity in the number of degrees of freedom (DOF). Thus, they perform well for highly constrained systems with few DOF like a robot arm with a manipulator (see [Fea08] for more details).

Maximal coordinate approaches, on the other hand, solve a sparse system of constraint equations in each simulation step. The performance therefore depends on the number of constraints. [Bar96] and [Ben07] pointed out, that the simulation of tree-like mechanisms with linear time complexity is possible using maximal coordinates.

For assembly and disassembly of articulations, reduced coordinates require a costly re-parametrization each time a constraint is added or removed. For the same reason, collisions and contacts can not be resolved efficiently with reduced coordinates alone, necessitating hybrid solution methods. Although Weinstein et. al. pointed out in [WTF06] that reduced coordinates can complicate the simulation of systems with frequent and unpredictable contact and collision (see also [KP03]), it is often used in video games for ragdoll simulation [RGL05]. Recently, [DBDB11] simulated interacting fibers efficiently using reduced coordinates to ensure inextensibility and combined it with an iterative solution method to treat frictional contacts. For the sake of generality, we choose a maximal coordinate approach for articulation constraints in this paper, although we admit that some of the heavily constrained examples might yield better performance using reduced coordinates.

Loops in the kinematic graph pose challenges to both reduced [Fea87] [SZ90] [SV96] [CA03] and maximal coordinates. A popular approach is to partition the kinematic graph into primary and secondary (auxiliary) constraints. The primary constraints can then be solved with an established direct factorization for tree-like linkages in linear time complexity [Bar96] [Fau99] [Ben07]. Constraint violations in auxiliary constraints can be countered with constraint stabilization [Bau72]. Iterative impulse-based methods have proven more effective in the past, although a certain softness of auxiliary constraints might still appear in time-critical applications. Goldenthal [GHF*07] achieved realistic behaviour of cloth using a direct solution method to enforce inextensibility in the warp- and weft directions, while using spring forces for shearing.

We ensure that all articulation constraints behave perfectly hard with a Tikhonov regularization, which is widely used in engineering applications (e.g. Matlab Simulink [WK03]), and in the Open Dynamics Engine [Smi00]. We argue in Section 4, that the regularization error does not pose major concerns for an impulse-based position-level approach. Iterative methods commonly include similar techniques for damping, relaxation and smooth joints [Erl07].

Frictional contacts and collisions are commonly modelled as a complementarity problem. From an instantaneous-time perspective, it suffices to require only the second or first time-derivative of the constraints to be 0, which leads to a linear complementarity problem (LCP). Acceleration-level approaches like the works of Baraff [Bar89] [Bar93] [Bar95] [Bar96] required a case distinction between contacts and collisions. Later, velocity-level methods resolved this issue [ST96] [AP97] [Erl07].

The established standard approach in the computer graphics community are iterative solution methods to solve the LCP, most commonly matrix splitting methods such as Projected Gauss-Seidel and blocked Gauss-Seidel (see [AC91] [Mor99] [Jea99], [DBDB11]). These methods were originally designed for the simulation of granular material. To accelerate convergence, conjugate gradient type methods have been proposed [RA05]. However, a drift of constraints on a position-level is inevitable with these methods, necessitating spring forces or post-stabilization [CP03] [Asc97]. Direct methods for LCPs, such as Lemke's method or the simplex algorithm [Bar94] can handle numerically challanging scenarios involving large mass ratios without convergence problems, but scale poorly in the number of contacts.

The impulse-based paradigm we adopted was introduced by [Hah85] and [MC95] and gained popularity due to its rather simple implementation. Later works of [GBF03] and [WTF06] (sometimes referred to as predictor-corrector method) showed that benefits similar to the ones of velocity-based time-stepping schemes [Mor99] [ST96] [Jea99] [Erl07] with respect to correct handling of contacts and collisions can be achieved. The difference between the approaches is that the impulse-based method computes impulses based on predictions of joint states, taking a discrete- instead of an instantaneous-time perspective.

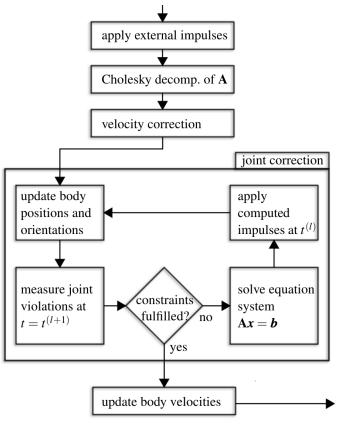



Figure 2: Illustration of a time step for articulation constraints. All impulses are applied to the bodies at $t = t^{(l)}$.

We refer to [BETC12] for a thorough distinction between and an introduction into different paradigms and solution methods. A survey of position-based methods is given in [BMOT13], but not covered in this paper, since it is mainly attractive for the simulation of cloth and deformable bodies.

## 3. Impulse-based simulation

Consider an articulated body (also referred to as multibody system) that is composed of $n_B$ rigid bodies and $n_J$ joints, each constraining the motion of its two connected bodies by $c_i$ degrees of freedom. We use maximal coordinates, introducing $n_C$ constraint equations, where $n_C = \sum_{i=1}^{n_J} c_i$ is the total number of constraints emerging from joints.

As in [Ben07], the computation of impulses is based on predictions of the joint states (see Figure 2 for a single time step from $t^{(l)}$ to $t^{(l+1)} = t^{(l)} + h$). Bodies are in free-flight between simulation steps, so the actual time integration does not need to consider constraints, making it, in principle, very easy to provide arbitrary time integration methods.

Guendelman et al. showed in [GBF03], that resting and sliding contacts and elastic impacts with break-away behaviour as for the famous Newton's Cradle (see also [SKV*12]) can both be handled effectively. They modify the order of the simulation step to distinguish between contacts and collisions. We refer to this technique as implicit distiction, because it eliminates the need for an ad hoc velocity threshold (as used in [BS06a] and [Mir96]) that can be hard to choose in some scenarios with external forces that involve more than just gravity. However, this comes at the cost of certain restrictions on the time integration method. In Appendix A, we show how to respect these restrictions for a Velocity-Verlet time integration, increasing the accuracy with respect to energy- and momentum conservation. Although the time step of our method is similar to [BFS05] [Ben07], we explain the basics for articulation constraints to motivate the factorization methods of the following sections.

Let $\mathbf{M} \in \mathbb{R}^{6n_B \times 6n_B}$ be the block-diagonal global body mass matrix $\mathbf{M} = \mathrm{diag}\,(m_1\,\mathbf{1}_3,\ \mathbf{I}_1,\ \ldots,\ m_{n_B}\,\mathbf{1}_3,\ \mathbf{I}_{n_B})$, where $\mathbf{1}_3 \in \mathbb{R}^{3\times 3}$ is the identity matrix, $m_i$ the body mass and $\mathbf{I}_i$ the orientation dependent inertia tensor. The linear and angular velocity of a body $i$ is labelled $\boldsymbol{v}_i$ and $\boldsymbol{\omega}_i$, respectively. Let $\mathbf{J} \in \mathbb{R}^{n_C \times 6n_B}$ denote the global sparse Jacobian that defines the configuration dependent relations between articulation constraints and bodies (see [Bar96] [BS06b] and [Erl07]). With the generalized body velocities $\boldsymbol{u} = \left(\boldsymbol{v}_1^T,\ \boldsymbol{\omega}_1^T,\ \ldots,\ \boldsymbol{v}_{n_B}^T,\ \boldsymbol{\omega}_{n_B}^T\right)^T$, the constraint forces $\boldsymbol{\lambda} \in \mathbb{R}^{n_C}$ (Lagrange multipliers) and the external forces $\boldsymbol{f}_{ext} \in \mathbb{R}^{6n_B}$ the equations of motion for some instant in time read $\mathbf{M}\dot{\boldsymbol{u}} = \mathbf{J}^T\boldsymbol{\lambda} + \boldsymbol{f}_{ext}$ . Substituting the difference quotient $\dot{\boldsymbol{u}} = \frac{\boldsymbol{u}_+ - \boldsymbol{u}_-}{\triangle t}$ yields

$$\underbrace{\boldsymbol{u}_+ - \boldsymbol{u}_-}_{\triangle \boldsymbol{u}} = \mathbf{M}^{-1}\mathbf{J}^T \underbrace{\triangle t\boldsymbol{\lambda}}_{\boldsymbol{p}} + \mathbf{M}^{-1} \underbrace{\triangle t \boldsymbol{f}_{ext}}_{\boldsymbol{p}_{ext}} \tag{1}$$

The impulse-based approach assumes finite changes in velocity in an infinitesimal time period $\triangle t$. Thus, we are searching for impulses $\boldsymbol{p} \in \mathbb{R}^{n_C}$ that yield the required jumps in body velocities $\triangle \boldsymbol{u}$ to fulfill joint constraints at the next simulation step. Bodies are assumed to be in free-flight between the simulation steps.

External impulses $\boldsymbol{p}_{ext} \in \mathbb{R}^{6n_B}$ are first applied to the bodies. Then a joint correction (also called position correction) is employed to find and apply additional joint impulses $\boldsymbol{p}$ at $t = t^{(l)}$, such that the position-level constraint violations at $t^{(l+1)}$ vanish (see Figure 2):

$$\boldsymbol{c}^{(l+1)}(\boldsymbol{p}) = \mathbf{0}\,, \quad \boldsymbol{c}^{(l+1)} \in \mathbb{R}^{n_C} \to \mathbb{R}^{n_C}$$

$\boldsymbol{c}^{(l+1)}$ is easily obtained from a preview of joint states, obtained by updating position and orientation of the related bodies temporarily.

Since an application of the standard Newton Method

$$\boldsymbol{p}_{n+1} = \boldsymbol{p}_n - \left(\nabla \boldsymbol{c}^{(l+1)}(\boldsymbol{p}_n)\right)^{-1} \boldsymbol{c}^{(l+1)}(\boldsymbol{p}_n)\,, \quad \boldsymbol{p}_0 = \mathbf{0}$$

involves evaluations of the gradient of nonlinear constraint displacements $\nabla \boldsymbol{c}^{(l+1)}(\boldsymbol{p}_n)$, we follow the work of [BS06b] and use a constant approximation

$$\nabla \boldsymbol{c}^{(l+1)}(\boldsymbol{p}_n) \approx h \nabla \dot{\boldsymbol{c}}^{(l)} \tag{2}$$

for the whole time step. The approximation is justified in light of a single step of the Euler method $\boldsymbol{c}^{(l+1)}(\boldsymbol{p}) \approx \boldsymbol{c}^{(l)} + h\dot{\boldsymbol{c}}^{(l)}(\boldsymbol{p})$ . Note, that the previous joint violations $\boldsymbol{c}^{(l)}$ do not appear in the gradient and any initial drift will be eliminated, if the iteration converges. The constraint velocities are related to body velocities by $\dot{\boldsymbol{c}} = \mathbf{J}\boldsymbol{u}$, so that with equation (1), the gradient can be expressed by

$$\mathbf{A} := \nabla \dot{\boldsymbol{c}}^{(l)} = \mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T\,.$$

$\mathbf{A}$ is called the system matrix and only depends on the current configuration and mass properties.

The Modified Newton Method is now given by

$$\boldsymbol{p}_{n+1} = \boldsymbol{p}_n - \mathbf{A}^{-1}\frac{\boldsymbol{c}^{(l+1)}(\boldsymbol{p}_n)}{h}\,, \quad \boldsymbol{p}_0 = \mathbf{0}\,,$$

so the system to be solved iteratively in each time step is

$$\mathbf{A}\triangle\boldsymbol{p}_n = -\frac{\boldsymbol{c}^{(l+1)}(\boldsymbol{p}_n)}{h}\,, \quad \triangle\boldsymbol{p}_n = \boldsymbol{p}_{n+1} - \boldsymbol{p}_n\,. \tag{3}$$

The impulses are applied to the bodies in each iteration via $\boldsymbol{u}_{n+1}^{(l)} = \boldsymbol{u}_n^{(l)} + \mathbf{M}^{-1}\mathbf{J}^T\triangle\boldsymbol{p}_n$. A costly factorization of the system matrix is performed only once per time step. In the following Sections, system (3) is rewritten for a simpler notation as

$$\mathbf{A}\boldsymbol{x} = \boldsymbol{b}\,, \quad \boldsymbol{x} := \triangle\boldsymbol{p}_n\,, \quad \boldsymbol{b} := -\frac{\boldsymbol{c}^{(l+1)}(\boldsymbol{p}_n)}{h}\,. \tag{4}$$

Although a strict quadratic converge rate is sacrificed by using a constant approximation of the gradient matrix, the method converges in practical settings after less than 10 iterations, if the step size $h$ is small enough, independent of $n_C$. If a threshold $\|\boldsymbol{c}^{(l+1)}(\boldsymbol{p}_n)\|_\infty < \varepsilon_P$ or a maximum number of iterations (we use 50 for all examples, unless mentioned otherwise) is reached, the joint correction is finished.

Note, that although this procedure is iterative, it is not to be confused with iterative solvers. We use a simultaneous, direct solver for articulations and the number of iterations can be bounded by some small constant that only depends on $h$ and the Tikhonov Parameter discussed in Section 4.

The velocity correction (see Figure 2) is only based on quantities at $t = t^{(l)}$ and finds joint impulses that lead to zero relative velocity for all articulation constraints when applied to the bodies. This can be achieved by solving the system once, using the relative joint velocities as right-hand side of system (4).

The velocity correction will not have much influence on the final outcome at the end of the time step, but it is applied after external impulses to improve the robustness of the following joint correction. This prevents light-weight components of an articulated body from flying off in the case of large external impulses.

## 4. Handling closed loops and redundant constraints

Redundant constraints often arise in mechanisms containing closed loops. Figure 1a shows a simple example with one redundant constraint. This leads to linearly dependent rows in the Jacobian $\mathbf{J}$ which causes the system matrix $\mathbf{A}$ to be rank-deficient (positive semi-definite).

The constraint of Figure 1a is called conflicting, if the distance of the anchor points of the lower body is different from the corresponding distance of the upper one, caused for example by an inaccurate user initialization. In this case, there is no admissible configuration to satisfy the constraints. Such systems are not considered here and the user must provide a mechanism without conflicting constraints up to the required accuracy $\varepsilon_P$ of the joint correction (cf. Section 3).

Each redundant constraint causes an eigenvalue of $\mathbf{A}$ to be zero. For this reason, the system of equations has no unique solution and cannot be solved robustly using a standard sparse Cholesky factorization. We apply a simple Tikhonov regularization approach by slightly modifying the

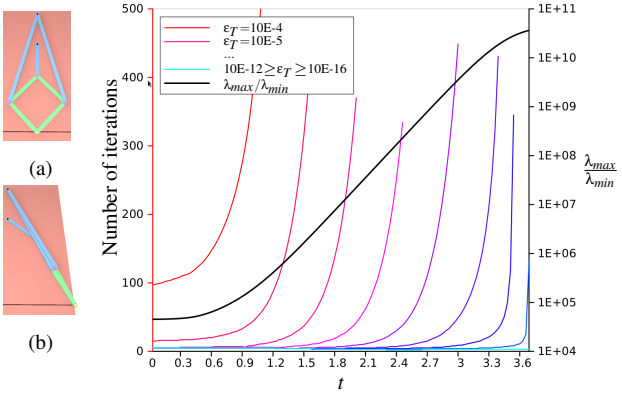



Figure 3: Simulation of a Peaucellier-Lipkin linkage with 9 redundant constraints (a). The linkage forces the yellow dot to move on the black line. The length of the green bars is 2 and constraints are fulfilled up to an accuracy of $\varepsilon_P = 10^{-9}$. An initial velocity lets it swing like a pendulum, tuned so that it nearly reaches a singular configuration ($\frac{\lambda_{max}}{\lambda_{min}} = 48245$ in (a), $\frac{\lambda_{max}}{\lambda_{min}} \approx 10^7$ in (b)). The maximum number of iterations of the joint correction is set to 500 and the step size is $h = 0.03$ s. In graph (c), the condition number $\lambda_{max}/\lambda_{min}$ (right axis) and the number of iterations (left axis) for different choices of the regularization parameter are plotted for this scenario as a function of simulation time.

system matrix and use $\mathbf{A}_\alpha = \mathbf{A} + \alpha\mathbf{1}_{n_C}$ instead of $\mathbf{A}$, where $\alpha$ is the regularization parameter. This increases all eigenvalues by $\alpha$ to ensure, that $\mathbf{A}_\alpha$ is positive definite, so the regularized system can be solved with a standard sparse Cholesky solver (we use CholMod [CDHR08]). However, this also introduces a regularization error and $\alpha$ should be chosen smaller than the smallest nonzero eigenvalue $\lambda_{min}$ of $\mathbf{A}$. We now assess the effects on the error of the solution to find the lowest possible $\alpha$ we can safely choose. If the matrix $\mathbf{A}_\alpha$ is computed with double precision, it is a slightly perturbed representation of an exact matrix $\mathbf{A}_\alpha^* = \mathbf{A}_\alpha + \mathbf{E}$, where $\mathbf{E}$ accounts for rounding errors, made during the computation of $\mathbf{A}_\alpha$. The relative error

$$h_{\mathbf{A}_\alpha} = \frac{\|\mathbf{E}\|}{\|\mathbf{A}_\alpha\|} \approx c\,\varepsilon_M \tag{5}$$

can be bounded by a small multiple of the given machine epsilon, which is $\varepsilon_M = 2^{-52} \approx 2\cdot 10^{-16}$ for double precision. The condition number $\kappa(\mathbf{A}_\alpha) = \|\mathbf{A}_\alpha\|\|\mathbf{A}_\alpha{}^{-1}\|$ is a measure for the maximum amplification of the relative error $h_{\boldsymbol{b}}$ in the right-hand side $\boldsymbol{b}$ of system (4) on the relative error in the solution $h_{\boldsymbol{x}_\alpha}$ (defined analogously to (5)).When using the spectral matrix norm for $\kappa(\mathbf{A}_\alpha)$, it is given by

$$\kappa(\mathbf{A}_\alpha) = \frac{\lambda_{max}+\alpha}{\alpha} \approx \frac{\lambda_{max}}{\alpha}\,. \tag{6}$$

$\kappa(\mathbf{A}_\alpha)$ can also be used to estimate the smallest relative error in $\mathbf{A}_\alpha$, which is able to cause it to be singular (cf. [Don87]):

$$h_{\mathbf{A}_\alpha} \leq \frac{c}{\kappa(\mathbf{A}_\alpha)} \tag{7}$$

The number $c$ is used to hide all unaccounted errors, caused by the choice of the matrix norm, errors of the computation of $\mathbf{A}_\alpha$, the scaling of $\mathbf{A}_\alpha$ and errors introduced by the algorithm to compute the solution. Usually, $c$ is a slowly growing function of the matrix dimension $n_C$ and assumed to be small. (5), (6) and (7) yield a lower bound on $\alpha$:

$$\alpha \geq c\, \lambda_{max} \varepsilon_M \,. \tag{8}$$

Since the computation of the spectral norm $\|\mathbf{A}\|_2 = \lambda_{max}$ is very costly in terms of performance, we use the matrix norm, induced from the 1-norm of vectors:

$$\|\mathbf{A}\|_\infty \overset{sym.}{=} \|\mathbf{A}\|_1 = \max_{\boldsymbol{v}\neq\mathbf{0}} \frac{\|\mathbf{A}\boldsymbol{v}\|_1}{\|\boldsymbol{v}\|_1}$$

It is computed as the maximum of the 1-norm of each column of $\mathbf{A}$. The multiplier $\varepsilon_T = c\,\varepsilon_M$ is left to choose by the user, so $\alpha$ is set to $\alpha = \varepsilon_T \|\mathbf{A}\|_1$. If $\varepsilon_T$ is chosen too small, the condition $\kappa(\mathbf{A}_\alpha)$ is very large, which causes the matrix $\mathbf{A}_\alpha$ to be too close to a singular matrix and may lead to large errors in the solution $\boldsymbol{x}_\alpha$. On the other hand, if $\varepsilon_T$ is too large, the resulting error in $\mathbf{A}_\alpha$ might cause the joint correction to converge much slower or even diverge, because the approximation of the gradient (2) is too inaccurate.

This, however, has little effect on the practical convergence rate of the joint correction, while $\varepsilon_T$ does not need to be finely tuned even for complex or ill-conditioned mechanisms. We use the Peaucellier-Lipkin linkage as a test case to quantify the dependence of the convergence speed on the condition number. The graph in Figure 3 shows, for example, that $\varepsilon_T$ may be chosen between $10^{-10}$ and $10^{-16}$ for a condition number of $10^9$ with practically no effect on convergence for this case. For most of the simulations described in this paper and the accompanying video, $\varepsilon_T = 10^{-10}$ does not effect convergence seriously.

As the inequality between the spectral and the 1-Matrixnorm

$$\frac{1}{\sqrt{n_C}}\|\mathbf{A}\|_1 \leq \|\mathbf{A}\|_2 \leq \sqrt{n_C}\|\mathbf{A}\|_1$$

shows, however, a rough tuning might be necessary in some extremely complex cases. For example, the tube with 150 rings (see Section 6) needed $\varepsilon_T = 10^{-12}$. We note, that a solver based on single precision can be expected to have more convergence problems in light of inequality (8).

## 5. An always sparse positive definite factorization

To solve the linear system (4), a sparse Cholesky factorization of the form $\mathbf{A} = \mathbf{L}\mathbf{D}\mathbf{L}^T$ is performed once per time step, $\mathbf{D}$ being a diagonal and $\mathbf{L}$ a lower triangular matrix with ones on its diagonal. The factorization process might add nonzeros to the factor $\mathbf{L}$ in addition to the ones present in the original matrix $\mathbf{A}$ (fill-in). The amount of fill-in depends on the order of constraints in the matrix. The elimination graph (EG) of a matrix can be used to determine fill-in by a process called vertex elimination (see [Ing06] for an introduction into the topic). For systems with a tree-like elimination graph, a depth-first order is perfect, i.e. there is no fill-in. The EG of the s.p.d. block-matrix $\mathbf{A}$ consists of a node for each joint and cliques between joints sharing a common body. The number of edges in the EG represents the number of off-diagonal nonzero entries in one half of $\mathbf{A}$.

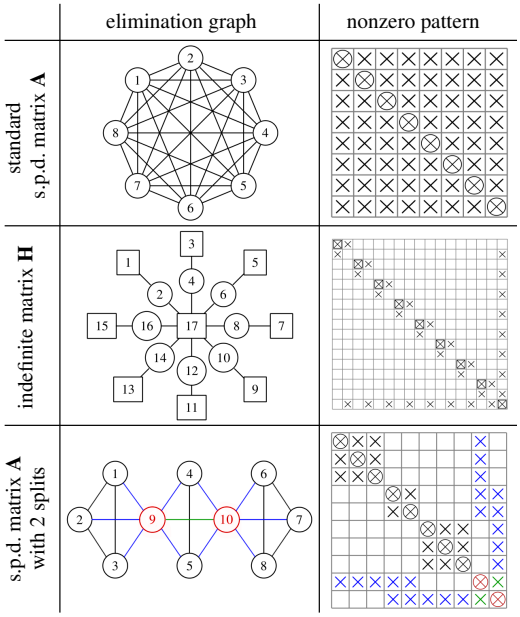



Figure 4: Illustration of the elimination graphs and nonzero patterns of three different, yet equivalent formulations for the example mechanism in Figure 1b. Circular nodes denote joints and rectangular nodes bodies. The matrices are ordered as indicated by the numbers, so that no fill-in is created during the factorizations. The indefinite formulation (middle row) leads to a larger, but always sparse matrix. However, it only works efficiently for tree-like articulations. Our method (bottom row) introduces two fixation joints (red) to reduce the number of edges of the EG in the top row and yields a sparse *and* positive definite system matrix $\mathbf{A}$.

In this section, we are concerned with cases like the one in Figure 1b, where a large clique leads to a dense matrix $\mathbf{A}$ and therefore a factorization with cubic time complexity (Figure 4, top row). A more practical example with the same performance problems is the Strandbeest in Figure 6b, since all legs are connected to the frame and a central axis.
Many authors overcome these performance problems by reformulating linear system (4) [Bar96] [Fau99] [Ben07]:

$$\underbrace{\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T}_{\mathbf{A}}\boldsymbol{x} = \boldsymbol{b} \quad \Leftrightarrow \quad \underbrace{\begin{pmatrix}\mathbf{M} & \mathbf{J}^T\\ \mathbf{J} & \mathbf{0}\end{pmatrix}}_{\mathbf{H}}\begin{pmatrix}\boldsymbol{y}\\ \boldsymbol{x}\end{pmatrix} = \begin{pmatrix}\mathbf{0}\\ -\boldsymbol{b}\end{pmatrix}$$

$\mathbf{A}$ is called the Schur complement of $\mathbf{H}$. The matrix $\mathbf{H} \in \mathbb{R}^{(6n_B+n_C)\times(6n_B+n_C)}$ is larger, but always sparse. The reason for this is that bodies are effectively included in the EG as nodes (Figure 4, middle row), so that cliques cannot form. For more general articulations containing loops, how-

ever, factorizing the indefinite matrix $\mathbf{H}$ efficiently is much more difficult, since for reasons of numerical stability, pivoting needs to be applied that might conflict with re-ordering interests of the fill-in minimization. In addition, a regularization technique for indefinite matrices that is as effective as the one described in Section 4 is unknown.

We propose an alternative approach that yields an always sparse *and* positive definite system matrix. The idea is to split the bodies into $n_P$ separate parts and connect them by $n_P - 1$ fixation joints that are comprised of 6 constraints each. Figure 4 (bottom row) illustrates a possible elimination graph that results from applying the body-splitting to the example in Figure 1b with $n_P = 3$.

The splitting is applied once before the simulation to all relevant bodies. The mass and inertia tensor of the central body are divided by $n_P$ and then $n_P - 1$ identical copies (without collision geometry or visualization) are created. They are connected at the center of gravity by the extra joints and the constraints of the original body are distributed over the parts. This breaks up the large clique of the original elimination graph into smaller ones and thus increases the sparsity of $\mathbf{A}$. The overall dynamic behaviour is not altered by this process, as long as the extra joints are simulated accurately. In Appendix B, we show that this strategy yields linear time-complexity for the factorization and solution and provide details on how to choose the number of splits to achieve good performance. We compared computation times of the indefinite and positive definite factorization and solution of two scalable examples for increasing $n_C$ in Figure 5.

## 6. Fill-in of the Cholesky factorization

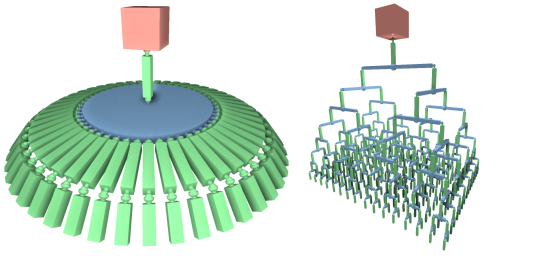
In the presence of loops in the kinematic graph, a perfect elimination order does usually not exist, meaning that fill-in is inevitable. For our single-core implementation we rely on Approximate Minimum Degree (AMD) to find an ordering that reduces fill-in. This only needs to be done once, as long as the kinematic graph of the simulated system does not change. For parallel computation, a Nested Dissection ordering is more adequate, but gives similar results with respect to the amount of fill-in [GKG09].

We chose some heavily constrained examples in Figure 6 (actually bad cases for maximal coordinate approaches) to provide insights on how the fill-in and therefore performance in cases with loops increases. The results indicate, that the amount of fill-in increases super-linearly only for mesh-like structures like the one in Figure 6e and the large octahedron in the supplementary video.

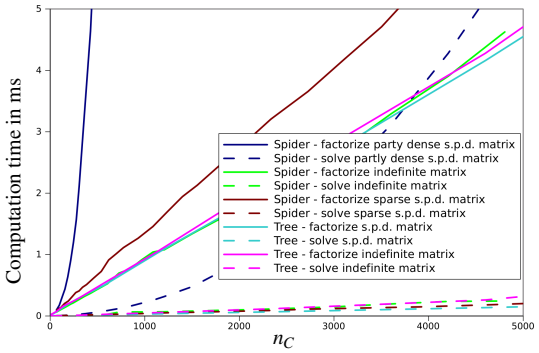
## 7. Contacts and Collisions

As pointed out in Section 3 and Appendix A, our simulation step resembles the one of [GBF03] in aspects concerning contacts and collisions. We also rely on a Gauss-Seidel-like propagation model, that resolves all detected contacts sequentially without a global view of the contact situation.

Another aspect we adopt is the implicit distinction between impacts (collisions) and contacts to prevent blocks



(a) Spider with 50 jointed legs (b) Tree structure




(c) Timings

Figure 5: Performance comparison (c) of the factorization and solution of the s.p.d. matrix $\mathbf{A}$ (using CholMod) vs. the indefinite matrix $\mathbf{H}$ (using a straight-forward implementation of the factorization and solution algorithms described in [Bar96]). The blue body of a spider-like articulated body (a) is attached to a variable number of legs with spherical joints. A standard positive definite factorization of the spider has a time complexity of $O(n_C^3)$ and the solution $O(n_C^2)$ because part of the system matrix is dense (dark blue in (c)). Linear time complexity is achieved by both the indefinite (green in (c)) and our improved s.p.d. formulation (brown in (c)), but the indefinite approach is restricted to tree-like articulations while ours can handle loops. A tree structure of variable depth (b) is also included in the graph (c).

from erroneously tumbling (instead of sliding) down an inclined plane. They divide the simulation step into a collision- and a contact-phase. Collisions are resolved based on the old body velocities $\boldsymbol{u}^{(l)}$ (or $\boldsymbol{u}_-^{(l)}$ as in Appendix A). To prevent some impacts from being resolved in the contact phase, however, the collision detection and contact generation is performed based on previews of body positions $\boldsymbol{x}^{(l+1)}$ and orientations $\boldsymbol{q}^{(l+1)}$, obtained from velocities $\boldsymbol{u}_+^{(l)}$ that already contain the external impulses (see Section 3 and Figure 2).

For collision detection, we use the free library SOLID [Ber04] which is based on an implementation of the Gilbert-Johnson-Keerthi Distance Algorithm (GJK) for convex bodies. SOLID can handle convex objects, compounds of convex objects and polygon soups and also includes a broad phase to narrow down colliding pairs of bodies efficiently.

A novel aspect of our method is that we couple each contact and collision (one at a time) with the one or two "touch-

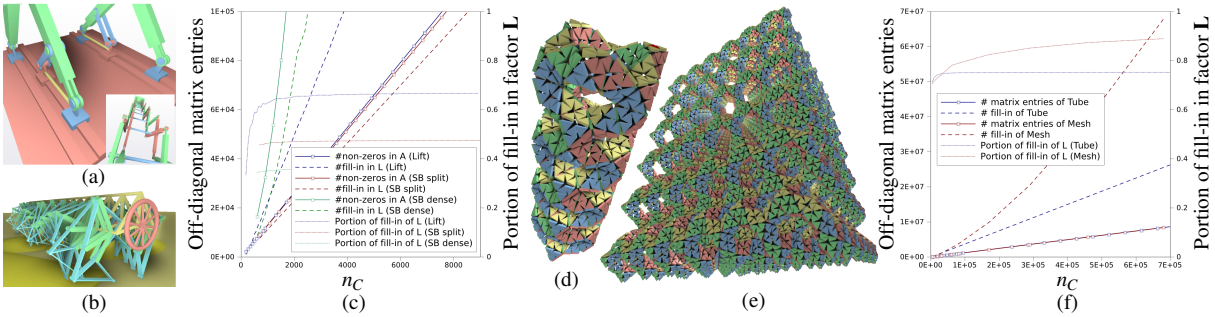



Figure 6: A scissor lift (a) with 10 segments, 500 constraints. A walking machine (b), the so-called Strandbeest (SB) with 50 legs, 4482 constraints. For different numbers of segments/legs, the number of off-diagonal nonzero entries in one half of $\mathbf{A}$ and the amount of fill-in in the factor $\mathbf{L}$ are plotted as a function of the number of constraints $n_C$ (c). The largest lift has 220 segments and the largest SB 100 legs. A complex mechanism based on the "octahedron" in Figure 1c in the shape of a tube with an intricate structure involving many loops (see supplementary video) is illustrated in (d). The depicted case consists of 8 rings (18981 constraints), while the largest example in graph (f) has 150 rings. (e) shows a cube with a mesh-like structure of these "octahedrons" (576288 constraints).

ing" multibody systems to ensure good convergence in the presence of bilateral constraints, challenging the numerics due to large mass-ratios. Figure 1d exhibits an example, where an iterative contact resolution leads to slow convergence, even if a direct solution method for the two joints is alternated with resolving the contact as in [Fau99] [Ben07].

To couple a normal contact constraint and two friction constraints directly to the articulation constraints we insert them into the linear system of equations as last rows/columns. Sparse matrix modification methods can be used to avoid redoing a costly Cholesky decomposition for each contact. However, dynamic Coulomb friction is known to lead to an asymmetric system matrix [Bar94]. This is a problem, because asymmetry necessitates an LU factorization, which is slower and requires pivoting for numerical stability.

As a remedy, one can use a less accurate, but symmetric approach to friction like the Open Dynamics Engine [Smi00], not including dynamic friction constraints into the linear system but resolving them with external impulses.

In Appendix C, we propose a different solution that is both able to handle asymmetry arising from more accurate dynamic friction models and does not require sparse matrix modification capabilities. Any sparse Cholesky Solver that is capable of applying separate forward- and back-substitutions can be used. This is possible, because we only add constraints at the end of the system, which constitutes a rather simple, special case of matrix modification without the need for complicated up- and downdate methods (see [DH05] [GGMS72]). For our demonstrations, we kept the inaccurate friction model, because it provides sufficient quality for the presented examples.

A collision that involves one or two articulated bodies is resolved by coupling them to the system matrix as described above, treating the collision in the same manner as articulation constraints. With a single solution of the extended linear system, impulses are computed and then applied to change the body's velocities, so the articulation constraints are fulfilled on a velocity-level ($\dot{\boldsymbol{c}}^{(l)} = \mathbf{0}$). This is similar to the velocity correction (see Section 3), however for the collision constraint, a separating velocity can be prescribed based on the coefficient of restitution.

After resolving all detected impacts, new impacts can occur. We noticed, that joint constraints should be fulfilled before each collision detection to avoid flawed collisions and contacts. We therefore apply external impulses, a velocity correction and a joint correction before each collision detection.

To resolve a contact during the contact phase, that involves articulated bodies, the approach is very similar. However, we perform a joint correction instead of a velocity correction to resolve contacts and articulation on a position-level. This is more costly than a single solution of the linear system, but the number of required solutions can usually be bounded by a small constant (see Section 3).

We noticed popping artifacts, when the maximimum number of iterations was reached resulting in small penetrations. For this reason, we applied the contact phase twice. First, the contacts are resolved on a velocity-level only during the joint corrections. The second (position-level) contact phase is used to correct only the body locations, not their velocities to prevent bodies from popping out of the ground. Note, that the computation time is not doubled by this subdivision. Rather, the computation time is divided up since the second phase can warm-start from the results of the velocity-level contact phase and usually converges quickly.

**Comparison with ODE**

The Open Dynamics Engine is based on a velocity-level time-stepping approach [ST96]. The original implementation only featured a direct solver based on dense matrix factorizations and a pivoting method for contacts. Later, an iterative method, termed QuickStep, was added. Drift of joint-

and contact constraints is countered with spring-like external forces (Error Reduction parameter or ERP). For the direct solution method, a regularization similar to Section 4 can be used (termed Constraint Force Mixing or CFM).

The first comparison is based on the simple case example illustrated in Figure 1d, that consists of a 50 kg rod (green) and a 7.5 ton block (blue). Finding the right parameters without resorting to mass scaling turned out to be cumbersome. We had to choose a very small step size of 2 ms for reasonably plausible dynamics, while our method achieves good results for very large time steps of up to 0.25 s (except for an obvious stuttering). For the direct solver, setting the ERP greater than zero resulted in huge energy increases, jittering an popping artifacts. Joint drift could not be corrected and only be influenced by decreasing the step size even further. The best results where obtained using Quick Step with around 500 iterations, ERP around 0.1 and 2 ms step size. Our simulator needed no fine-tuning at all.

We dropped a Strandbeest with 50 legs (see Figure 6b) from a small height of 0.4 meters, about a third of its total height. We had to choose a step size smaller than 3 ms. Otherwise the spring-like behaviour of joint constraints led to an erroneous folding of some legs on impact. Loops in the mechanism do not cause any problems for the direct ODE solver with a regularization parameter of roughly CFM $= 10^{-10}$, however, the $O(n_c^3)$ dense matrix factorization led to about 3 seconds computation time per step without contacts, and between 4 and 10 seconds when touching the floor. The ODE Quickstep method is much faster, but we could not get rid of a spring-like behaviour of constraints and a small jittering due to an ERP of 0.1 to correct joint drift. The effect became more severe when increasing the weight of the yellow frame of the Strandbeest, which could only be countered with even smaller step sizes.

| | $k$ | $h$ (ms) | $t_c$ (ms) | error |
|---|---|---|---|---|
| ODE | 0 | 2 | 7464 | 0.0223 |
| | 5 | 0.75 | 19676 | 0.0258 |
| | 10 | 0.5 | 29125 | 0.0223 |
| | 20 | 0.3 | 48055 | 0.02 |
| our method | 0 | 20 | 7730 | $< 10^{-6}$ |
| | 5 | 20 | 9790 | $< 10^{-6}$ |
| | 10 | 20 | 10078 | $< 10^{-6}$ |
| | 20 | 20 | 8924 | $< 10^{-6}$ |

The table shows the total computation time $t_c$ and maximum constraint errors for 5 seconds simulation time with $k$ times its total original mass added to the frame. The performance and robustness of our simulator is not affected by an increasing weight.

## 8. Conclusion and future work

The results presented in Sections 4 to 7 show, that an extensive use of direct solution methods that includes loop closures and coupling between contacts/collisions and articulation yields benefits for simulating complex mechanisms with hard constraints. Relatively large time steps are possible, as exhibited for many of the examples in our supplementary video.

In combination with the impulse-based method, a fine-tuning of parameters can be avoided to a large extent, since the problem of joint drift does not arise, even when using regularization to handle systems with redundant constraints. Nonetheless, we believe that some of the ideas presented in this paper might be useful for other methods such as velocity-level constraint-based approaches as well.

Note, however, that the performance of our method might be worse for some contact-heavy scenarios with well-conditioned articulated bodies. A purely iterative treatment of both contacts and joints as described in [WTF06] might yield better performance for these cases.

The time integration in Appendix A led to long-term energy conservation of frictionless systems, but we noticed a small energy increase/decrease in some cases depending on the choice of the Tikhonov Parameter $\varepsilon_P$. For this reason, we did not consider using more accurate time integration methods like the Moser-Veselov integrator used by [SKV*12].

To cover cases that involve strong coupling between contacts, we plan to extend our approach to Shock-Propagation.

## 9. Acknowledgments

This research was funded by the German Research Foundation (DFG). We would like to thank David Bommes for strengthening our interest in complex mechanisms by presenting some Heureka Polyhedra (see also [Woh97]).

## References

[AC91] ALART P., CURNIER A.: A mixed formulation for frictional contact problems prone to newton like solution methods. *Comput. Methods Appl. Mech. Eng.* (1991). 3

[AG85] ARMSTRONG W., GREEN M.: The dynamics of articulated rigid bodies for purposes of animation. *The Visual Computer 1 31* (1985), 353–375. 3

[AP97] ANITESCU M., POTRA F. A.: Formulating dynamic multi-rigid-body contact problems with friction as solvable linear complementarity problems. *NONLINEAR DYNAMICS 14* (1997), 231–247. 3

[Asc97] ASCHER U. M.: Stabilization of invariants of discretized differential systems. *Numerical Algorithms 14* (1997). 3

[Bar89] BARAFF D.: Analytical methods for dynamic simulation of non-penetrating rigid bodies. In *In Proc. of ACM SIGGRAPH 1989* (1989), pp. 223–232. 3

[Bar93] BARAFF D.: Issues in computing contact forces for non-penetrating rigid bodies. *Algorithmica 10* (1993), 292–352. 3

[Bar94] BARAFF D.: Fast contact force computation for nonpenetrating rigid bodies. SIGGRAPH '94, ACM. 2, 3, 8

[Bar95] BARAFF D.: Interactive simulation of solid rigid bodies. *IEEE Comput. Graph. Appl. 15*, 3 (1995), 63–75. 3

[Bar96] BARAFF D.: Linear-time dynamics using lagrange multipliers. In *Proceedings of SIGGRAPH 1996* (1996), Computer Graphics Proceedings, Annual Conference Series. 2, 3, 4, 6, 7

[Bau72] BAUMGARTE J.: Stabilization of constraints and integrals of motion in dynamical systems. *Comput. Methods Appl. Mech. Eng. 1* (1972), 1–16. 3

[BBK05] BOTSCH M., BOMMES D., KOBBELT L.: Efficient linear system solvers for mesh processing. In *Proceedings of the 11th IMA international conference on Mathematics of Surfaces* (2005), IMA'05, Springer-Verlag, pp. 62–83. 1, 11

[Ben07] BENDER J.: *Impulse-based dynamic simulation in linear time*. Tech. rep., Universität Karlsruhe, 2007. 2, 3, 4, 6, 8

[Ber04] BERGEN G. V. D.: Freesolid - software library for interference detection. *http://www.win.tue.nl/ gino/solid/* (2004). 7

[BETC12] BENDER J., ERLEBEN K., TRINKLE J., COUMANS E.: Interactive simulation of rigid body dynamics in computer graphics. In *Proceedings of Eurographics* (2012). 4

[BFS05] BENDER J., FINKENZELLER D., SCHMITT A.: An impulse-based dynamic simulation system for vr applications. *Proceedings of Virtual Concept* (2005). 2, 4

[BMOT13] BENDER J., MÜLLER M., OTADUY M. A., TESCHNER M.: Position-based methods for the simulation of solid objects in computer graphics. In *EUROGRAPHICS 2013 State of the Art Reports* (2013), Eurographics Association. 4

[BS06a] BENDER J., SCHMITT A.: *Constraint-based collision and contact handling using impulses*. Tech. rep., Universität Karlsruhe, 2006. 4

[BS06b] BENDER J., SCHMITT A.: Fast dynamic simulation of multi-body systems using impulses. In *Virtual Reality Interactions and Physical Simulations* (2006). 4

[CA03] CRITCHLEY J. H., ANDERSON K. S.: A generalized recursive coordinate reduction method for multibody system dynamics. *International Journal for Multiscale Computational Engineering* (2003). 3

[CDHR08] CHEN Y., DAVIS T. A., HAGER W. W., RAJAMANICKAM S.: Algorithm 887: Cholmod, supernodal sparse cholesky factorization and update/downdate. *ACM Trans. Math. Softw. 35*, 3 (2008), 22:1–22:14. 3, 5

[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. 3

[DBDB11] DAVIET G., BERTAILS-DESCOUBES F., BOISSIEUX L.: A hybrid iterative solver for robustly capturing coulomb friction in hair dynamics. In *Proceedings of the 2011 SIGGRAPH Asia Conference* (2011), SA '11, ACM, pp. 1–12. 3

[DH05] DAVIS T. A., HAGER W. W.: Row modifications of a sparse cholesky factorization. *SIAM J. Matrix Analysis Applications 26*, 3 (2005), 621–639. 8, 13

[Don87] DONGARRA J. J.: *LINPACK Users' Guide*. Society for Industrial and Applied Mathematics, 1987. 5

[Erl07] ERLEBEN K.: Velocity-based shock propagation for multibody dynamics animation. *ACM Trans. Graph. 26*, 2 (June 2007). 2, 3, 4

[Fau99] FAURE F.: Fast iterative refinement of articulated solid dynamics. *Trans. on Vis. and Comp. Graph. 5*, 3 (1999), 268–276. 3, 6, 8

[Fea87] FEATHERSTONE R.: *Robot Dynamics Algorithm*. Kluwer, 1987. 3

[Fea08] FEATHERSTONE R.: *Rigid Body Dynamics Algorithms*. Springer, New York, 2008. 3

[GBF03] GUENDELMAN E., BRIDSON R., FEDKIW R.: Nonconvex rigid bodies with stacking. *TOG* (2003). 2, 3, 4, 7, 11

[GGMS72] GILL P. E., GOLUB G. H., MURRAY W. A., SAUNDERS M. A.: *Methods for modifying matrix factorizations*. Tech. rep., 1972. 8

[GHF*07] GOLDENTHAL R., HARMON D., FATTAL R., BERCOVIER M., GRINSPUN E.: Efficient simulation of inextensible cloth. SIGGRAPH '07, ACM. 3

[GKG09] GUPTA A., KORIC S., GEORGE T.: Sparse matrix factorization on massively parallel computers. In *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis* (2009), SC 09, ACM, pp. 1:1–1:12. 3, 7

[Hah85] HAHN J. K.: Realistic animation of rigid bodies. *Proc. of the 15th ann. conf. on Comp. graph. and interactive techniques* (1985). 3

[Har11] HARADA T.: A parallel constraint solver for a rigid body simulation. In *SIGGRAPH Asia 2011 Sketches* (2011), ACM. 3

[Ing06] INGRAM S.: Minimum degree reordering algorithms: A tutorial. Online Only, 2006. 6

[Jan13] JANSEN T.: http://www.strandbeest.com. 1

[Jea99] JEAN M.: The non-smooth contact dynamics method. *Comput. Methods Appl. Mech. Eng. 177* (1999), 235 – 257. 3

[KE04] KRYSL P., ENDRES L.: *Explicit Newmark/Verlet algorithm for Time Integration of the Rotational Dynamics of Rigid Bodies*. Tech. rep., University of California, San Diego, 2004. 11

[KP03] KRY P. G., PAI D. K.: Continuous contact simulation for smooth surfaces. *ACM Trans. Graph. 22*, 1 (2003), 106–129. 3

[KSJP08] KAUFMAN D. M., SUEDA S., JAMES D. L., PAI D. K.: Staggered projections for frictional contact in multibody systems. *ACM Trans. Graph.* (2008). 2

[MC95] MIRTICH B., CANNY J.: Impulse-based simulation of rigid bodies. In *Proceedings of the 1995 symposium on Interactive 3D graphics* (1995), ACM, pp. 181–ff. 3

[Mir96] MIRTICH B. V.: *Impulse-based Dynamic Simulation of Rigid Body Systems*. Tech. rep., University of California at Berkely, 1996. 4

[Mor99] MOREAU J. J.: Numerical aspects of the sweeping process. *Comput. Methods Appl. Mech. Eng. 177* (1999). 3

[MV91] MOSER J., VESELOV A. P.: Discrete versions of some classical integrable systems and factorization of matrix polynomials. vol. 139, Comm. Math. Phys., pp. 217–243. 11

[MW88] MOORE M., WILHELMS J.: Collision detection and response for computer animation. In *Computer Graphics* (1988), pp. 289–298. 3

[RA05] RENOUF M., ALART P.: Conjugate gradient type algorithms for frictional multi-contact problems: applications to granular materials. *Comp. Methods Appl. Mech. Eng.* (2005). 3

[RGL05] REDON S., GALOPPO N., LIN M. C.: Adaptive dynamics of articulated bodies. *ACM Trans. Graph.* (2005). 3

[SKV*12] SMITH B., KAUFMAN D. M., VOUGA E., TAMSTORF R., GRINSPUN E.: Reflections on simultaneous impact. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2012) 31*, 4 (2012), 106:1–106:12. 2, 4, 9, 11

[Smi00] SMITH R.: Open dynamics engine. *http://www.ode.org* (2000). 3, 8

[ST96] STEWART D., TRINKLE J. C.: An implicit time-stepping scheme for rigid body dynamics with coulomb friction. *Intern. Journal of Num. Meth. in Eng. 39* (1996), 2673–2691. 3, 8

[SV96] STEJSKAL V., VALASEK M.: Kinematics and dynamics of machinery. *Marcel Dekker* (1996). 3

[SZ90] SCHRÖDER P., ZELTZER D.: The virtual erector set: dynamic simulation with linear recursive constraint propagation. In *I3D* (1990), ACM, pp. 23–31. 3

[WK03] WOOD G., KENNEDY D.: Simulating mechanical systems in simulink with simmechanics. 3

[Woh97] WOHLHART K.: Kinematics and dynamics of the fulleroid. *Kluwer Academic Publishers* (1997). 9

[WTF06] WEINSTEIN R., TERAN J., FEDKIW R.: Dynamic simulation of articulated rigid bodies with contact and collision. *Trans. of Vis. and Comp. Graph.* (2006), 365–374. 2, 3, 9