Workshop on Virtual Reality Interaction and Physical Simulation VRIPHYS (2017)
F. Jaillet and F. Zara (Editors)

Unified Simulation of Rigid and Flexible Bodies Using Position Based Dynamics

M. Frâncu and F. Moldoveanu

University Politehnica Bucharest, Romania

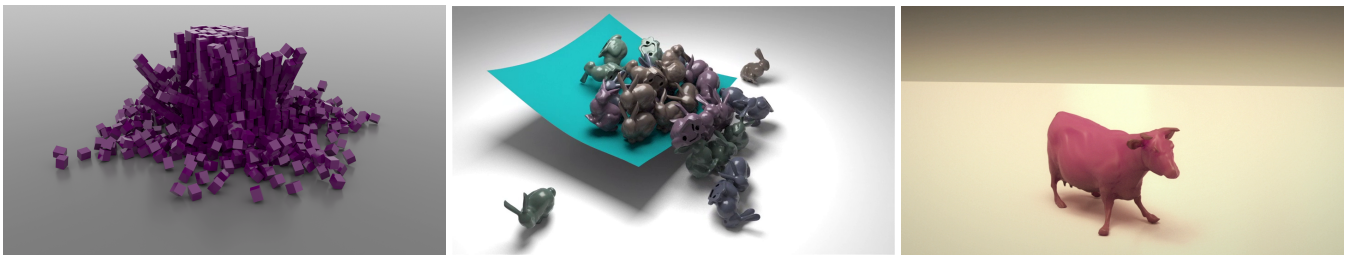

Figure 1: *From left to right: rigid boxes falling on ground, bunnies falling on a piece of cloth, a flexible cow falling on ground.*

Abstract

In this paper we present a new position based approach for simulating rigid and flexible bodies with two-way coupling. This is achieved by expressing all the dynamics as constraints and running them in the same solver. Our main contribution is an accurate contact and Coulomb friction model based on a fixed point iteration of a cone complementarity problem. We formulate the problem as a nonlinear convex minimization at position level and solve it using a new accelerated form of projected Jacobi. We add elasticity to the constraints by means of regularization and show how to add more damping in a credible manner. We also use this viscoelastic model to build an accurate position-based finite element solver for soft bodies. The novelty of this solver is that it is no longer an approximation and it is based directly on the elasticity theory of continuous media.

Categories and Subject Descriptors (according to ACM CCS): Mathematics of Computing [G.1.0]: Numerical Analysis—Numerical algorithms Computer Graphics [I.3.5]: Computational Geometry and Object Modeling—Physically based Modeling

1. Introduction

For the last decade position based dynamics (PBD) has been successfully applied to the simulation of deformable bodies [BMOT13]. This was possible due to the inherent nonlinearity of the method in terms of satisfying constraints and the full implicit formulation: not only the magnitude of the constraint forces are considered implicit, but also their directions [Gol10]. These implicit constraint directions ensure the unconditional stability of the system, especially for materials with fast changing constraint gradients and transverse oscillations, e.g. cloth or threads [TNGF15].

The main drawback of PBD is that it has no rigorous mathematical model for contact and friction and thus it is almost never used for rigid body simulations (with the exception of [DCB14]). In our literature research we have not found any clear proof for the convergence of a PBD-like method with unilateral constraints and friction. Because of this some authors choose to treat contacts as bilateral constraints [Gol10] or approximate friction at the end of the step [MHHR07] (e.g. based on penetration depth [Jak01]) without giving a sound recipe for mixing friction with the position corrections. We address this issue in this paper and then demonstrate our result with a working position based rigid body simulator.

The novelty of our simulator is that it not only offers accurate treatment of contact and friction for rigid bodies, but it can also simulate deformable bodies in a physically correct manner using the finite element method (FEM) formalism. We base our approach on the constraint regularization technique developed in [SLM06] and we express it at position level.

DOI: 10.2312/vriphys.20171083

1.1. Related work

There has been a wealth of work published on the subject of rigid body simulation with contact and friction - for a survey see [BETC14]. We note the advances made in the 90s by Baraff, Stewart and Anitescu. Given the drawbacks of penalty forces, Baraff introduced the acceleration based linear complementarity problem (LCP) method. This method had its problems too (related to impacts and the Painlevé paradox) that were later solved by a velocity based approach that allows discontinuities in the velocities, i.e. impulses. The new velocity time stepping (VTS) schemes [AH04] became very popular in computer graphics, games and real time simulators. We take a similar approach in this paper, but based on more recent work geared towards convex optimization [TA11, MHNT15, ACLM11].

Traditionally in computer graphics deformable bodies have been simulated using implicit integrators due to their unconditional stability properties. These have been applied not only to mass-spring systems, but also to simulations using the finite element method (FEM) [SB12]. Recently, the popular Backward Euler integrator has been recast as an optimization problem [BML*14] helping us to gain new insights.

While initially constraint based methods were not considered for simulating deformable bodies, this changed with the advent of PBD [MHHR07] and constraint regularization [SLM06]. PBD was originally introduced by Jakobsen for games based on molecular dynamics methods and a nonlinear version of the Stewart-Trinkle solver for rigid bodies [Jak01]. Goldenthal later showed that PBD stems from the fully implicit integration of a constrained system [Gol10]. Even though in theory constraints do not allow deformation for all the degrees of freedom (locking), in practice they proved quite successful for simulating a wide range of objects (e.g. cloth, hair, soft bodies - see [BMOT13] for a survey). We think this is due to the fact that iterative solvers are not exact and thus make the constraints softer.

The idea of a unified solver is not new and our simulator bears maybe most similarity to Autodesk Maya's Nucleus. Our results are also along the line of more recent PBD work [**?**, MMCK14, BKCW14, DCB14] and Projective Dynamics [BML*14]. In addition, a great job of emphasizing the role of nonlinearity for achieving stability was done in [KTS*14] and [TNGF15].

1.2. Contributions

We aim in this paper is to show that PBD is a physically sound method. This is done in Section 2. Also, PBD can be used for both rigid and deformable bodies with constraints, contact and friction in a single unified solver. The advantages of this formulation include better constraint satisfaction, unconditional stability and out of the box two way coupling of rigid and elastic materials. Our new viscoelastic model permits us to incorporate soft constraints, damping and FEM into PBD (more about applications in Section 3).

Another goal we had in mind was to keep the computational overhead to a minimum compared to existing methods. This is why we chose our mathematical formulation to be expressible as a matrix-free solver. We present a novel projected gradient descent algorithm for nonlinear optimization in Section 4. The algorithm is based on both the Jacobi and the Nesterov methods so it can be parallelized. In Section 5 we continue to give some more details on how to implement this solver (or a Gauss-Seidel one) for specific examples like the frictional contact constraint or the FEM tetrahedron constraint. In the end we give some code implementation notes and take a closer look at our results.

2. Mathematical model

2.1. Equations of motion

We start with the equations of motion for a general system of bodies and, at first, we will also introduce bilateral constraints between the bodies: general nonlinear functions equated to zero, describing for example a bead on a wire or joints articulating rigid bodies. The resulting equations can also be derived from Hamilton's principle and the principle of virtual work by using a Lagrangian augmented by a special constraint potential: $-\gamma^T\Psi(\mathbf{q})$ [ST96]. They form a special type of *differential algebraic equations* (DAE) [Lac07]:

$$\mathbf{M}\dot{\mathbf{v}} = \mathbf{f}_{tot} + \nabla\Psi(\mathbf{q})\gamma, \tag{1}$$

$$\dot{\mathbf{q}} = \zeta(\mathbf{q}, \mathbf{v}), \tag{2}$$

$$\Psi(\mathbf{q}) = \mathbf{0}, \tag{3}$$

where $\mathbf{v} \in \mathbb{R}^n$ is the generalized velocity vector, where n is the number of degrees of freedom of the system, $\mathbf{q} \in \mathbb{R}^{n'}$ is the generalized position vector (where $n' \geq n$ is the optimal number of parameters describing position and orientation), ζ is a general kinematic mapping between velocities and position derivatives, $\mathbf{M}$ is the mass matrix [BETC14], $\Psi(\mathbf{q})$ is a vector-valued bilateral constraint function, $\nabla\Psi(\mathbf{q})$ is its gradient (i.e. the constraint directions), $\gamma \in \mathbb{R}^m$ is a Lagrange multipliers vector enforcing the bilateral constraints in (3) (m is the number of constraints), and $\mathbf{f}_{tot}$ is the total generalized force acting on the system (external and Coriolis).

In order to discretize the equations of motion we use the Implicit Euler (IE) integrator.

$$\mathbf{M}(\mathbf{v}^{l+1} - \mathbf{v}^l) = h\nabla\Psi(\mathbf{q}^{l+1})\gamma^{l+1} + h\mathbf{f}_{tot}^l, \tag{4}$$

$$\mathbf{q}^{l+1} = \mathbf{q}^l + h\mathbf{L}\mathbf{v}^{l+1}, \tag{5}$$

$$\Psi(\mathbf{q}^{l+1}) = \mathbf{0}, \tag{6}$$

where l is the current simulation frame, h is the time step (considered constant), and $\mathbf{L}(\mathbf{q}^l)$ is a linear kinematic mapping [BETC14] with $\mathbf{L}^T\mathbf{L} = \mathbf{1}$ (the identity matrix). The IE discretized equations can be brought to a minimization form:

$$\mathbf{v}^{l+1} = \underset{\Psi(\mathbf{q}^l + h\mathbf{L}\mathbf{v}) = \mathbf{0}}{\arg\min} \ \frac{1}{2}\mathbf{v}^T\mathbf{M}\mathbf{v} - \hat{\mathbf{f}}^T\mathbf{v}, \tag{7}$$

where $\hat{\mathbf{f}} = \mathbf{M}\mathbf{v}^l + h\mathbf{f}_{tot}^l$ and the new positions come from (5).

We can recast (7) in terms of correction velocities $\Delta\mathbf{v} = \mathbf{v}^{l+1} - \tilde{\mathbf{v}}$ instead of the new velocities:

$$\text{minimize } \tfrac{1}{2}\Delta\mathbf{v}^T\mathbf{M}\Delta\mathbf{v} \text{ subject to } \Psi(\tilde{\mathbf{q}} + h\mathbf{L}\Delta\mathbf{v}) = \mathbf{0}, \tag{8}$$

where $\tilde{\mathbf{v}} = \mathbf{M}^{-1}\hat{\mathbf{f}}$ and $\tilde{\mathbf{q}} = \mathbf{q}^l + h\mathbf{L}\tilde{\mathbf{v}}$. If we denote $\Delta\mathbf{q} = h\mathbf{L}\Delta\mathbf{v} = \mathbf{q}^{l+1} - \tilde{\mathbf{q}}$ then we can express (8) as:

$$\text{minimize } \tfrac{1}{2h^2}\|\Delta\mathbf{q}\|_{\mathbf{M}}^2 \text{ subject to } \Psi(\tilde{\mathbf{q}} + \Delta\mathbf{q}) = \mathbf{0}, \tag{9}$$

where $\bar{\mathbf{M}} = \mathbf{L}\mathbf{M}\mathbf{L}^T$. This last formulation is the *projection* on the constraint manifold method for solving DAE; this is the same as the IE integration of constraint forces according to [Gol10]. This derivation can also be extended to include unilateral constraints: $\Phi(\mathbf{q}^{l+1}) \geq \mathbf{0}$.

2.2. Position based dynamics

The nonlinear optimization in (9) lies at the heart of PBD. It has a quadratic objective and general nonlinear constraints and it is convex. For the bilateral constrained case Goldenthal proposed the *fast projection* method [Gol10]. This is a modified form of the *sequential quadratic programming* (SQP) strategy for solving nonlinear optimization problems [WN99]. The iterations (for equality constraints only) have the following KKT matrix form:

$$\begin{bmatrix} \bar{\mathbf{M}} - h^2\mathbf{H}_k\gamma_k & -h^2\mathbf{D}_k \\ \mathbf{D}_k^T & \mathbf{0} \end{bmatrix} \begin{pmatrix} \delta\mathbf{q}_{k+1} \\ \delta\gamma_{k+1} \end{pmatrix} = \begin{pmatrix} -\bar{\mathbf{M}}\Delta\mathbf{q}_k + h^2\mathbf{D}_k\gamma_k \\ -\Psi(\mathbf{q}_k) \end{pmatrix}, \tag{10}$$

where k is the current iteration number, $\mathbf{D} = \nabla\Psi(\mathbf{q})$ and $\mathbf{H} = \nabla^2\Psi(\mathbf{q})$ is the second derivative of the constraint function (third rank tensor). The matrix $\tilde{\mathbf{K}} = \mathbf{H}\gamma$ was recently dubbed the "geometric stiffness matrix" and plays an important role in transverse stability [TNGF15]. The main drawback of (10) is that the upper left block matrix is not easy to compute and invert, preventing us from taking a Schur complement. That is why fast projection omits the $\mathbf{H}$ term (along with simplifying the right hand side) and obtains a series of smaller $m \times m$ linear systems. Stability is guaranteed in the end by running many nonlinear iterations, similarly to [KTS*14].

The SQP angle comes in handy when adding unilateral constraints and iterations can no longer be expressed as linear systems, but rather as quadratic programs (QPs). In essence, these QPs are formed by taking the initial optimization problem (9) and linearizing the constraints around the current point $\mathbf{q}_k$.

In practice it is often better to work with the dual formulations of the QPs. This means working with Lagrange multipliers. Fast projection works with these dual variables too, as the solution to each linear system is a set of Lagrange multiplier increments $\delta\gamma_{k+1}$. The position displacements are then computed as $\delta\mathbf{q}_{k+1} = h^2\mathbf{L}\mathbf{D}_k\delta\gamma_{k+1}$ and the velocity corrections as $\delta\mathbf{v}_{k+1} = h\mathbf{D}_k\delta\gamma_{k+1}$. In general it is not true that $\gamma_{k+1} = \gamma_k + \delta\gamma_{k+1}$, but it can be used as an approximation of the constraint force magnitude. The dual QP has the form:

$$\begin{aligned} &\text{minimize}_{\delta\gamma}\ \tfrac{1}{2}\delta\gamma^T\mathbf{N}_k\delta\gamma + \delta\gamma^T\mathbf{r}_k \\ &\text{subject to } \gamma_k + \delta\gamma \in \Upsilon_k, \end{aligned} \tag{11}$$

where $\mathbf{N}_k = h^2\mathbf{D}_k^T\bar{\mathbf{M}}^{-1}\mathbf{D}_k$, the residual $\mathbf{r}_k = \mathbf{c}(\mathbf{q}_k)$ corresponding to $\mathbf{c}(\mathbf{q}) = (\Psi(\mathbf{q}), \Phi(\mathbf{q}))$ and Υ_k is the feasible set of the Lagrange multipliers (the whole real axis for bilateral constraints and only the non-negative part for unilateral constraints). $\delta\gamma_{k+1}$ is the solution of this QP and the update step is:

$$\mathbf{q}_{k+1} = \mathbf{q}_k + \delta\mathbf{q}_{k+1}, \tag{12}$$

$$\mathbf{v}_{k+1} = \mathbf{v}_k + \delta\mathbf{v}_{k+1}, \tag{13}$$

where the initial guess is $\mathbf{q}_0 = \tilde{\mathbf{q}}$, $\mathbf{v}_0 = \tilde{\mathbf{v}}$. After running a certain number of iterations or attaining a convergence criterion the new positions and velocities $(\mathbf{q}^{l+1}, \mathbf{v}^{l+1})$ are given by the last solution.

This whole process tries to solve the dual of the original projection minimization problem (9). This dual problem is hard to express in a closed form due to the nonlinear and implicit relation between the displacements and the Lagrange multipliers: $\mathbf{F}(\Delta\mathbf{q}, \gamma^{l+1}) = \bar{\mathbf{M}}\Delta\mathbf{q} - h^2\nabla\Psi(\tilde{\mathbf{q}} + \Delta\mathbf{q}) = \mathbf{0}$ [Gol10].

We can use various iterative methods to solve the dual of (9) with added unilateral constraints. We are especially interested in projected gradient descent methods like nonlinear Krylov methods or Nesterov's method [MHNT15]. In fact, PBD can be seen as either fast projection with a one-step Gauss-Seidel linear system solver or as a nonlinear Gauss-Seidel minimizer. We adopt the latter view and develop it further in Section 4. It is worth noting that neither fast projection, nor PBD are guaranteed to bring down $\mathbf{F}$ to 0 after a limited number of iterations [Gol10]. But they do a good job of projecting on the constraint manifold and minimizing the objective in (9), making them very close to IE integration.

2.3. Stiffness and damping

Regularization is a technique that replaces the lower right zero block in (10) by a compliance matrix $\mathbf{C}^{-1}$ which is made up of small values (mainly on the diagonal) in order to perturb the original problem and make it easier to solve [Lac07]. In what follows we will show that regularization is also equivalent to softening the constraints, i.e. replacing the constraint forces with elastic forces.

The optimization form of the IE integrator is in general:

$$\text{minimize}_{\bar{\Delta\mathbf{q}}}\ \tfrac{1}{2h^2}\bar{\Delta\mathbf{q}}^T\mathbf{M}\bar{\Delta\mathbf{q}} + U_{int}(\mathbf{q}) + U_{ext}(\mathbf{q}), \tag{14}$$

where $\bar{\Delta\mathbf{q}} = \mathbf{q} - \mathbf{q}^l - h\mathbf{v}^l$, U_{int} and U_{ext} are the potentials of the internal and external forces respectively. In the presence of constraints $\mathbf{c}(\mathbf{q})$, we have two options to specify U_{int}: as a constraint potential energy $U_c = -\gamma^T\mathbf{c}(\mathbf{q})$ or as a penalty term

$$U_e = \tfrac{1}{2}\mathbf{c}(\mathbf{q})^T\mathbf{C}\mathbf{c}(\mathbf{q}), \tag{15}$$

where $\mathbf{C}$ is a block diagonal stiffness matrix (not to be confounded with the tangential stiffness matrix $\mathbf{K} = \mathbf{D}\mathbf{C}\mathbf{D}^T$).

Using U_c is equivalent to the position projection described by (9) and analogous to the formulation of *projective dynamics* [BML*14]. In the projection formulation the external potential is absorbed into the unconstrained position $\tilde{\mathbf{q}}$ and velocity $\tilde{\mathbf{v}}$. On the other hand, U_e is an elastic potential and corresponds to the implicit integration of elastic forces $\mathbf{f}_e = -\nabla U_e = -\mathbf{D}\mathbf{C}\mathbf{c}(\mathbf{q})$. If we make the constraint force $\mathbf{f}_c = -\nabla U_c = \mathbf{D}\gamma$ equal to $\mathbf{f}_e$ we get the new regularized (or softened) constraints:

$$\mathbf{c}(\mathbf{q}) + \mathbf{C}^{-1}\gamma \geq 0. \tag{16}$$

This amounts to using the following KKT matrix template in (10):

$$\begin{bmatrix} \bar{\mathbf{M}} & -h^2\mathbf{D} \\ \mathbf{D}^T & \mathbf{C}^{-1} \end{bmatrix}. \tag{17}$$

Although these results are mostly based on [SLM06], we are new in stressing the strong connection between implicit integration of elastic forces and regularized implicit nonlinear constraint projection (recently a similar viewpoint was developed in parallel in [MMC16]). Indeed, one can inspect the minimization in (14) and

see it remains the same whether we are using a soft constraints (16) potential or an elastic energy penalty term (15).

All we need to do in order to regularize PBD is replace the constraints with the ones in (16) and the matrix $\mathbf{N}$ with the Schur complement of (17):

$$\mathbf{N} = h^2\mathbf{D}^T\bar{\mathbf{M}}^{-1}\mathbf{D} + \mathbf{C}^{-1}. \tag{18}$$

This new formulation permits us to include stiffness into PBD. All this without having to add a non-diagonal tangential stiffness matrix to the mass matrix (like in [TNGF15]). This modification allows us to simulate both rigid and deformable bodies (e.g. mass-spring systems or finite element discretizations) and also soften contacts between them.

Another of our contributions is to add damping in a physical and credible manner to PBD. In general, we can do this by using a Rayleigh dissipation function [Lac07]:

$$\varphi = \tfrac{1}{2}\dot{\mathbf{c}}(\mathbf{q})\mathbf{R}\dot{\mathbf{c}}(\mathbf{q}), \tag{19}$$

where $\dot{\mathbf{c}}(\mathbf{q}) = \mathbf{D}^T\mathbf{v}$ and $\mathbf{R}$ is a positive definite matrix (often diagonal). This usually means adding a viscous drag force term, generated by the dissipative potential: $\mathbf{f}_d = -\nabla_{\mathbf{v}}\varphi = -\eta\dot{\mathbf{c}}(\mathbf{q})$, where η is a damping coefficient. Very important to note is that these forces act only along the constraint directions and so the damping does not look unnatural.

Using the dissipation potential in (19) yields a new regularization formula: $\mathbf{c}(\mathbf{q}) + \mathbf{C}^{-1}\mathbf{R}\dot{\mathbf{c}}(\mathbf{q}) + \mathbf{C}^{-1}\gamma \geq 0$, which in turns gives a new KKT matrix and a new Schur complement: $\mathbf{N} = h(h\mathbf{1} + \mathbf{C}^{-1}\mathbf{R})\mathbf{D}^T\bar{\mathbf{M}}^{-1}\mathbf{D} + \mathbf{C}^{-1}$. We exemplify our result in the case where the stiffness matrix is of the form $\mathbf{C} = \kappa\mathbf{1}$ and for a particular form of Rayleigh damping, i.e. $\mathbf{R} = \eta\mathbf{1} = \rho\mathbf{C}$, where $\rho = \eta/\kappa$. In the end the terms $\mathbf{N}$ and $\mathbf{r}$ in (11) get replaced by:

$$\mathbf{N} = h(h+\rho)\mathbf{D}^T\bar{\mathbf{M}}^{-1}\mathbf{D} + \mathbf{C}^{-1}, \tag{20}$$

$$\mathbf{r} = \mathbf{c}(\mathbf{q}) + \rho\mathbf{D}^T\mathbf{v}. \tag{21}$$

Note that damping can also be applied in the case of infinite stiffness $\kappa \to \infty$ (i.e. $\mathbf{C}^{-1} = \mathbf{0}$) given that the ratio ρ remains finite.

2.4. Frictional contact

We illustrate in Figure 2 the ith particle contact point of a body with a surface. One can identify a normal to the surface, $\mathbf{n}^i$, and any two tangent vectors, $\mathbf{s}^i$ and $\mathbf{t}^i$, so that together they form an orthonormal frame. When switching to generalized coordinates these normal tangent directions become the vectors $\mathbf{D}_n^i$, $\mathbf{D}_s^i$ and $\mathbf{D}_t^i$ [Ani06]. We can now present the discretized equations of motion for a constrained system of bodies with frictional contact:

$$\begin{aligned}\mathbf{M}(\mathbf{v}^{l+1} - \mathbf{v}^l) = {} & h\sum_{i\in\mathcal{G}_A}(\gamma_n^i\mathbf{D}_n^i + \gamma_s^i\mathbf{D}_s^i + \gamma_t^i\mathbf{D}_t^i) \\ & + h\sum_{i\in\mathcal{G}_B}(\gamma_B^i\nabla\Psi^i) + h\mathbf{f}_{tot}^l,\end{aligned} \tag{22}$$

$$\mathbf{q}^{l+1} = \mathbf{q}^l + h\mathbf{L}\mathbf{v}^{l+1}, \tag{23}$$

$$\Psi^i(\mathbf{q}^{l+1}) = 0, i \in \mathcal{G}_B, \tag{24}$$

$$0 \leq \Phi^i(\mathbf{q}^{l+1}) \perp \gamma_n^i \geq 0, i \in \mathcal{G}_A, \tag{25}$$

$$(\gamma_s^i, \gamma_t^i) = \underset{\sqrt{(\gamma_s^i)^2+(\gamma_t^i)^2}\leq\mu^i\gamma_n^i}{\arg\min} (\mathbf{v}^{l+1})^T(\gamma_s^i\mathbf{D}_s^i + \gamma_t^i\mathbf{D}_t^i). \tag{26}$$

In continuous form these equations form a *differential variational inequality* (DVI) [TA11]. The novelty in our approach is that we are using a full implicit Euler integrator instead of semi-implicit/symplectic Euler and we keep the non-penetration condition at position level as a nonlinear unilateral constraint. This is similar to the nonlinear scheme presented in Section 3.6 of [ST96]. The following notations were used: $\mathcal{G}_A$ is the set of active unilateral constraints, $\mathcal{G}_B$ is the set of bilateral constraints, $\Phi^i(\mathbf{q})$ is a unilateral constraint function describing contact (i.e. gap function), $\mathbf{D}_n^i$ is the gradient of the constraint function: $\nabla\Phi^i(\mathbf{q}^{l+1})$, γ_n^i is the Lagrange multiplier of the contact condition (25), i.e. normal reaction magnitude, $\mathbf{D}_s^i$ and $\mathbf{D}_t^i$ are the generalized tangent directions, γ_s^i and γ_t^i are the corresponding tangent Lagrange multipliers (i.e. friction force components), and μ^i is the friction coefficient.

Equations (22)-(23) represent the IE integration step, corresponding to (4)-(5). Equation (25) represents the Signorini contact complementarity conditions and (26) the maximum dissipation principle that synthesizes the Coulomb friction laws [BETC14]. The latter can also be stated as the condition that the total contact force $\gamma_A^i = (\gamma_n^i, \gamma_s^i, \gamma_t^i)$ should reside inside the friction cone (see

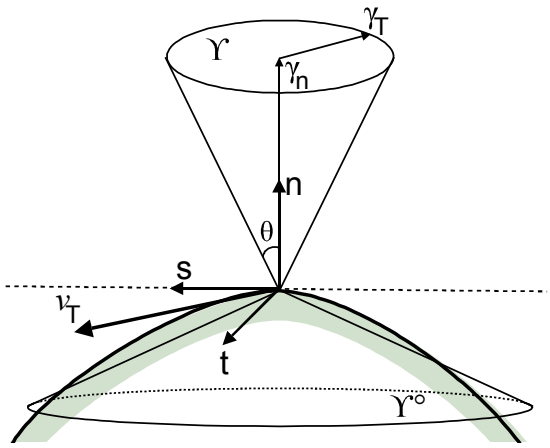

Figure 2: *Particle contact point with friction cone* Υ *given by* $\theta = \arctan\mu$ *and its polar cone* Υ° *depicted below.*

Figure 2 for an illustration):

$$\Upsilon_A^i = \left\{ \gamma_A^j | \sqrt{((\gamma_s^j)^2 + (\gamma_t^j)^2)} \leq \mu^i \gamma_n^j \right\}. \quad (27)$$

Note that the conditions (24) and (25) are strongly nonlinear which make the problem hard to tackle. Also, the problem in (22)-(26) is nonconvex due to the coupling between the friction and the normal force.

In [Ani06] the DVI is linearized and convexified so that it can be expressed in the end as a quadratic minimization problem with conic constraints. We take this formulation and extend it to the fully implicit and nonlinear case in (22)-(26):

$$\begin{aligned} &\text{minimize } W(\mathbf{v}) = \tfrac{1}{2}\mathbf{v}^T\mathbf{M}\mathbf{v} - \hat{\mathbf{f}}^T\mathbf{v} \\ &\text{subject to } \Phi^i(\mathbf{q}^{l+1}) - h\mu^i\|\mathbf{v}_T^i\| \geq 0, i \in \mathcal{G}_A, \\ &\qquad \Psi^i(\mathbf{q}^{l+1}) = 0, i \in \mathcal{G}_B, \end{aligned} \quad (28)$$

where $\|\mathbf{v}_T^i\| = \sqrt{(\mathbf{D}_s^{i,T}\mathbf{v})^2 + (\mathbf{D}_t^{i,T}\mathbf{v})^2}$ is the magnitude of the tangential relative velocity at the contact point. Our approach for solving this problem is to derive a new fixed point iteration that is equivalent to a *cone complementarity problem* (CCP) at every iteration:

$$\Upsilon_k^\circ \ni -(h\mathbf{D}_k^T\mathbf{v} + \mathbf{b}_k) \perp \gamma \in \Upsilon_k, \quad (29)$$

where $\gamma = (\gamma_A, \gamma_B)$ and $\mathbf{b}_k = (\mathbf{b}_{k,A}, \mathbf{b}_{k,B})$ - the first component corresponding to contacts $\mathbf{b}_{k,A} = (\Phi(\mathbf{q}_k) - h\mathbf{D}_{n,k}^T\mathbf{v}_k, 0, 0)$ and the second to bilateral constraints $\mathbf{b}_{k,B} = \Psi(\mathbf{q}_k) - h\nabla\Psi_k^T\mathbf{v}_k$. Υ is the direct sum of all friction and bilateral cones, Υ° is the corresponding polar cone and $\mathbf{D}$ is the concatenation of all constraint directions, i.e. $\mathbf{D}_A^i = [\mathbf{D}_n^i|\mathbf{D}_s^i|\mathbf{D}_t^i]$ and $\mathbf{D}_B^i \equiv \nabla\Psi^i$. You can consult [TA11] for more details on notation and how (29) can be derived from a linearization of (28) around $(\mathbf{q}_k, \mathbf{v}_k)$. A proof of convergence can be sketched, but is out of the scope of this paper. The CCP in (29) can be solved using another fixed point iteration based on matrix splitting (i.e. relaxation) with constraint projection, shown to converge in [TA11]. After some k iterations the solution $\mathbf{v}_{k+1}$ can be substituted for $\mathbf{v}^{l+1}$.

One drawback of the velocity based convexified smooth cone approach is that it may produce normal impulse artifacts [Ani06] but these manifest only for high slip speeds and friction coefficients [MHNT15]. Given that our approach is a fixed point iteration very similar to the one in [ACLM11] (i.e. the velocity is updated at every iteration) it may also converge to the solution of the original nonconvex problem. However, in practice, when running fewer iterations one may choose to use a different friction model in order to avoid potential artifacts. For example one can use the mixed LCP polyhedral friction cone model [ST96] or another model like box LCP friction [BETC14]. Note that both of these alternative friction models manifest anisotropy.

Notice that the minimization in (28) can be easily reformulated in terms of displacements $\delta\mathbf{q}$ instead of velocities just as we did in Section 2.1. This means that the frictional contact equations (22)-(26) can be easily incorporated into the projection formulation in (9) by using the constraints in (28). Therefore we can use the same successive minimization approach used in PBD (11) to also accommodate frictional contact by just making the distinction that Υ_k now becomes a direct sum of cones.

3. Applications

The PBD framework with the added viscoelastic and friction models is a very versatile method that allows the simulation of both rigid and soft bodies in the same solver loop. The advantages are twofold: we do not need to combine constrained dynamics with implicit integration of stiff elastic forces and we do not worry about two way coupling as it comes for granted. The types of objects we can simulate include particles, rigid bodies, mass-spring systems (e.g. cloth, threads, soft bodies), linear FEM and other PBD specific methods [BMOT13].

3.1. Particle systems

In the case of particles the number of degrees of freedom per body is 3, so for a single particle $\mathbf{q}$ becomes just the position $\mathbf{x}$ and $\mathbf{v} = \dot{\mathbf{x}}$. The position integration is also very simple:

$$\mathbf{x}^{l+1} = \mathbf{x}^l + h\dot{\mathbf{x}}^{l+1}, \quad (30)$$

and the mass matrix is a diagonal matrix $\mathbf{M} = \text{diag}(m_j\mathbf{1}_3)$, where m_j is the mass of each particle and $\mathbf{1}_3$ the identity matrix. Usually in PBD distance constraints are used:

$$\Psi(\mathbf{x}) = \|\mathbf{x}_i - \mathbf{x}_j\| - l_{ij}, \quad (31)$$

where l_{ij} is the rest length, to simulate threads and cloth [MHHR07]. We can also add contact constraints $\Phi(\mathbf{x}) \geq 0$ and accurate friction modeling as described in Section 2.4 in order to add realistic collisions and self-collisions. Particle systems with only contact and friction can be used to model very simple granular material. Cloth models can be enhanced with shearing and bending constraints and more damping along the cloth surface can be added through our viscoelastic model. Bending constraints can be either links connecting second order neighbors or can be more complex functions involving 4 vertex stencils (2 neighboring triangles), e.g. dihedral angle constraint [MHHR07]. We could also add area preserving or zero strain constraints per triangle in a more accurate continuum based approach (in a similar vein to Section 3.3).

3.2. Rigid bodies

Rigid bodies add rotations to the mix. A single rigid body has 6 degrees of freedom: the generalized velocity has the same number of components $\mathbf{v} = (\dot{\mathbf{x}}, \omega)$ where ω is the angular velocity, but the generalized position has 7 components $\mathbf{q} = (\mathbf{x}, \xi)$ where $\mathbf{x}$ is the location of the center of mass and $\xi \in \mathbb{H}$ is a unit quaternion. This is because the 4 components of the quaternion $\xi = (s, \mathbf{a})$ are the minimum necessary to parametrize the orientation of a body $\mathbf{R} \in SO(3)$ in a nonsingular manner. The kinematic equation is:

$$\dot{\xi} = \tfrac{1}{2}\xi \circ (0, \omega), \quad (32)$$

where $\circ$ is the quaternion product that can also be interpreted as a linear map acting on ω. This equation can be integrated through a simple Euler method:

$$\xi^{l+1} = \xi^l + \tfrac{h}{2}\xi \circ (0, \omega^{l+1}), \quad (33)$$

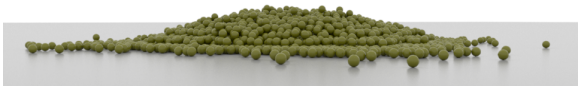

Figure 3: *Granular matter simulated using 5000 rigid spheres.*

Figure 4: *Simulation of 5000 rigid boxes falling on ground.*

followed by a renormalization of the quaternion or directly using an exponential map that keeps the result on the 4D unit sphere S^3, as described in Section 2.1 of [TA11].

When integrating the velocities note that the generalized force also includes torques and the total force acting on the system also includes centrifugal and Coriolis terms besides external interactions. For more details about handling these terms as well as the inertia tensors you can consult [BETC14].

Modeling contact and friction between two bodies can be done by adding a gap scalar function along the contact normal direction $\mathbf{n}_{ij}$: $\Phi(\mathbf{q}) = \mathbf{n}_{ij} \cdot (\mathbf{x}_i + \mathbf{R}_i\mathbf{p}_i - \mathbf{x}_j - \mathbf{R}_j\mathbf{p}_j)$, where $\mathbf{p}_i$ and $\mathbf{p}_j$ are the pair of closest points expressed in their respective local frames. The derivation of the Jacobian, i.e. the normal generalized constraint direction $\mathbf{D}_n$, can be found in many places [BETC14], and $\mathbf{D}_s$ and $\mathbf{D}_t$ can be built from it. Similarly a bilateral constraint representing a spherical joint can be represented by a 3 valued vector function: $\Psi(\mathbf{q}) = \mathbf{x}_i + \mathbf{R}_i\mathbf{p}_i - \mathbf{x}_j - \mathbf{R}_j\mathbf{p}_j = 0$ (see Figure 5). A hinge constraint has only one value, this number representing the number of unconstrained rotational degrees of freedom.

3.3. Finite element method

Following the approach in [SLM06] and adapting it to our own viscoelastic PBD method (as described in Section 2.3) we can simulate soft bodies using a linear finite element discretization, i.e. constant strain tetrahedra. The elastic energy of an element has the form $U_e = \frac{1}{2}V(\mathbf{q})\varepsilon(\mathbf{q})^T\mathbf{C}\varepsilon(\mathbf{q})$, where V is the volume of the tetrahedron, ε is the symmetric strain tensor (linearized as a 6-vector) and $\mathbf{C}$ is the stress-strain relationship matrix defined by two elastic parameters [SB12]. Comparing this energy to the general form in (15), we can identify the constraint function to be used:

$$\Psi(\mathbf{q}) = \sqrt{V(\mathbf{q})}\varepsilon(\mathbf{q}). \tag{34}$$

Remember that using the elastic potential energy $U_e = \frac{1}{2}\Psi(\mathbf{q})^T\mathbf{C}\Psi(\mathbf{q})$ in (14) is the same thing as implicitly integrating the forces arising from stresses [SB12]. But we will choose to use the completely equivalent formulation in (16) instead. In other words, our constraint tries to keep the strain of each element close to zero (as for a rigid body) but the regularization of (34) will prevent it from doing so by adding compliance to the system.

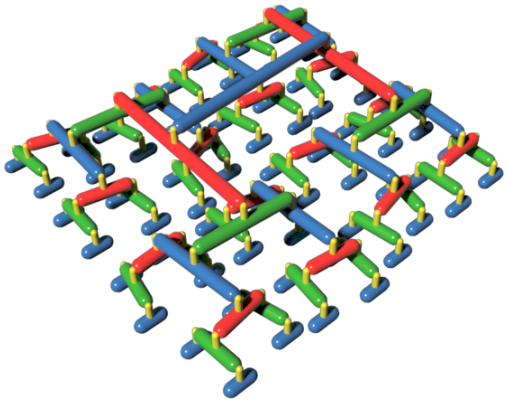

Figure 5: *Hierarchy of articulated rigid bodies.*

Each tetrahedron has 4 nodes, that we can consider as particles (lumped mass approximation), totaling 12 degrees of freedom. In the end, our nonlinear constrained dynamics approach will work just as in the case of a particle system but using the above strain constraint involving sets of 4 particles. For the strain function we can use any material model we want; just like in [SLM06] we use the nonlinear Green-Lagrange strain, as it preserves the volume under large deformations (in contrast to Cauchy strain). You can find the Jacobian for the constraint in (34) in Appendix A. We emphasize the fact that even though the the finite elements are linear, we are using a nonlinear St. Vennant-Kirchoff elasticity model.

4. Accelerated Jacobi

Most authors prefer to solve the dual problems in (11) to attacking directly the primal formulation in (9). We focus on nonlinear numerical optimization solvers and a multitude of such methods exist (e.g. SQP, see Section 2.2) [WN99]. However, we will restrict ourselves to projected gradient descent due to its simplicity and popularity among constraint based dynamics engines. Relaxation methods (Gauss-Seidel, Jacobi, successive over relaxation - SOR) are coordinate descent methods and rely on matrix splitting. Gauss-Seidel/SOR is widely used due to its robustness to solve modified LCPs [BETC14], the convexified problem [TA11] and PBD [Jak01], but it is quite hard to parallelize (through graph coloring). The most popular parallel alternative is Jacobi. More recently, accelerated projected gradient descent (APGD or Nesterov's method) has been proposed [MHNT15], though it was only applied to the linearized velocity approach. And this is also the case for conjugate gradient (CG) and minimum residual. In our experience the conjugate residuals (CR) algorithm [Saa03] has a more monotonic convergence than CG and we prefer it for low iteration counts. We argue in this paper that all these methods can be extended to their nonlinear variants (e.g. nonlinear CG). Taking inspiration from the

nonlinear forms of CR, APGD and Jacobi we propose the iterative scheme:

$$\gamma_{k+1} = \text{proj}(\gamma_k - \omega[\text{diag}(\mathbf{N})]^{-1}\mathbf{r}_k - \beta_{k+1}\delta\gamma_k), \quad (35)$$

where $\omega < 1/\rho(\mathbf{N})$, $\beta_{k+1} = \frac{\theta_k(1-\theta_k)}{\theta_k^2+\theta_{k+1}}$, and $\theta_{k+1} = \frac{1}{2}(-\theta_k^2 + \theta_k\sqrt{\theta_k^2+4})$ with $\theta_0 = 1$ (according to [MHNT15]). The actual spectral radius $\rho(\mathbf{N})$ does not need to be computed, as it can approximated from the count of incident constraints to a body.

The cone projection operator is the same as the one described in [TA11]. One can recognize in (35) a modified version of Jacobi that is accelerated using a momentum term just like in APGD in order to increase the convergence rate closer to Gauss-Seidel, while keeping the scheme order independent and parallelizable. Also, the scheme is nonlinear as the residual $\mathbf{r}_k$ is computed using updated constraint function values and gradients. You can find a pseudocode outline of the scheme in Algorithm 1.

Unconstrained step to $\tilde{\mathbf{q}}, \tilde{\mathbf{v}}$
$\mathbf{q}_0 = \tilde{\mathbf{q}}, \mathbf{v}_0 = \tilde{\mathbf{v}}$
for k = 0:maxIter-1 **do**
  Compute $\mathbf{c}(\mathbf{q}_k)$ and $\mathbf{D}_k$
  Compute the residual $\mathbf{r}_k$ - see eq. (11)
  Update Lagrange multipliers using (35)
  Compute constraint force increment $\mathbf{D}_k\delta\gamma_{k+1}$
  Update both positions and velocities using (12)-(13)
end for

Algorithm 1: Nonlinear projected gradient descent constraint solver using a Jacobi approach (instead of the accelerated form in (35) one could use a standard Jacobi update step).

5. Constraint solver

In this section we focus on computing the constraint force. This force can be applied either directly after it was computed (in a Gauss-Seidel fashion) or after traversing all of the constraints (Jacobi fashion). The Jacobi scheme in Algorithm 1 is not complete because we are not explicitly expressing the Jacobians and constraint functions involved for each type of constraint. We skip the distance constraint (31), as its description is ubiquitous in all PBD papers [MHHR07, Jak01] and we consider extending it to the Accelerated Jacobi method is trivial.

Contact only has been tackled in the past either by instantaneously considering it as a bilateral constraint or through a crude complementarity approach. Friction on the other hand has had no solid mathematical framework to rely on and we believe that our nonlinear fixed point CCP iteration is the first. You can find our pseudo-code for frictional contact between rigid bodies in Algorithm 2. Note that we identify the two bodies by the indices 1 and 2 and a contact pair is fully determined by a world normal $\mathbf{n}$ and the closest points between the two bodies $\mathbf{a}_1$ and $\mathbf{a}_2$ - each expressed in their respective frame. Contacts between rigid bodies and surface triangles (e.g. from cloth) are handled in a similar way, just that we distribute the impulse to the triangle vertices using the barycentric coordinates of the contact point. For a granular material example using rigid spheres see Figure 3. For FEM we are basically solving constraints involving each linear tetrahedral element. You can find the pseudo-code for such a constraint involving 4 particles or nodes in Algorithm 3. It is based on Sections 2.3, 3.3 and Appendix A.The output of the procedure consists of the 4 forces that will be applied to the tetrahedron vertices. Note that we solve for all the 6 Lagrange multipliers in a block matrix fashion using a direct solver.

Input: contact pair $(\mathbf{n}, \mathbf{a}_1, \mathbf{a}_2)$, β, old force γ, increment $\delta\gamma$
$\mathbf{p}_1 = \mathbf{R}_1\mathbf{a}_1, \mathbf{p}_2 = \mathbf{R}_2\mathbf{a}_2$
Compute normal residual $r_n = \mathbf{n}\cdot(\mathbf{x}_1 + \mathbf{p}_1 - \mathbf{x}_2 - \mathbf{p}_2)$ (gap)
Compute normal diagonal term d_n of matrix $\mathbf{N}$
$\gamma_n = \text{clamp}(\gamma_n - \frac{\omega}{h^2 d_n}r_n - \beta\delta\gamma_n, 0, \infty)$, $\gamma_T = 0$
$\mathbf{v}_{12} = (\mathbf{v}_1 + \omega_1 \times \mathbf{p}_1) - (\mathbf{v}_2 + \omega_2 \times \mathbf{p}_2)$ (relative velocity)
$\mathbf{v}_T = \mathbf{v}_{12} - (\mathbf{n}\cdot\mathbf{v}_{12})\mathbf{n}$ (tangential relative velocity)
if $\mathbf{v}_T \neq 0$ **then**
  Compute tangential residual $r_T = \|\mathbf{v}_T\|$ (slip speed)
  Compute tangential direction $\tau = \mathbf{v}_T / v_T$
  Compute tangential diagonal term d_T
  $(\gamma_n, \gamma_T) = \text{project}(\gamma_T - \frac{\omega}{h^2 d_T}r_T - \beta\delta\gamma_T, \gamma_n)$
end if
Output: contact force $\gamma = (\gamma_n, \gamma_T)$, i.e. $\mathbf{f} = \gamma_n\mathbf{n} + \gamma_T\tau$

Algorithm 2: Pseudo-code for computing the normal and friction forces between 2 rigid bodies in contact. Can be used with either a Jacobi or a Gauss-Seidel approach ($\omega \geq 1, \beta = 0$).

Input: tetrahedron $(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$
Compute shape matrix $\mathbf{D}_s = [\mathbf{x}_1 - \mathbf{x}_0 | \mathbf{x}_2 - \mathbf{x}_0 | \mathbf{x}_3 - \mathbf{x}_0]$
Compute deformation gradient $\mathbf{F} = \mathbf{D}_s\mathbf{D}_m^{-1}$
Compute Green strain ε from the matrix $\frac{1}{2}(\mathbf{F}^T\mathbf{F} - \mathbf{1}_3)$
Compute strain Jacobian $\mathbf{J}$ (see Appendix A)
Compute local system matrix $\mathbf{N} = h^2\mathbf{J}\mathbf{M}\mathbf{J}^T + \mathbf{C}^{-1}$
Solve $\mathbf{N}\gamma + \varepsilon = 0$
Output: internal forces $\mathbf{f} = \mathbf{J}^T\gamma = (\mathbf{f}_0, \mathbf{f}_1, \mathbf{f}_2, \mathbf{f}_3)$

Algorithm 3: Pseudo-code for computing the internal forces inside a tetrahedron. Here a block Gauss-Seidel approach is employed.

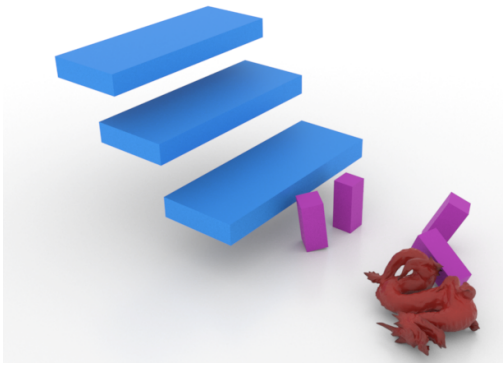

Figure 6: *A rubbery dragon falling on stairs and hitting against rigid bodies (Young's modulus $E = 0.5$ GPa and Poisson ratio $\nu = 0.2$); simulated using the FEM constraint solver in Algorithm 3.*

6. Implementation

All of the above algorithms were implemented in C++ in a unified manner such that all constraints were solved at the same time and in the same solver. For this we used a single common list of bodies that could have each a maximum of 6 degrees of freedom. This list was split into groups, each group having a different meaning (e.g. cloth, rigid body system or FEM soft body) and different types of constraint lists. Some constraint types were specific to only one group (e.g. link constraints for cloth), others were common among several groups (e.g. contact constraints) and the rest were specially designed for coupling between groups (e.g. rigid body vs. triangle).

In terms of constraint solving we used mainly two approaches: Gauss-Seidel and Accelerated Jacobi. We further optimized the latter using OpenMP parallel for loop directives. You can find the speed-up factor in comparison to Gauss-Seidel in Tables 1 and 2. Measurements were done on a dual core laptop CPU (i5-3317U) and a quad core desktop CPU (i7-3770).

	Gauss-Seidel	Accelerated Jacobi	Speedup
2000 boxes	140 ms	90 ms	1.55x
50x50 cloth	6.3 ms	2.7 ms	2.33x
100x100 cloth	27.5 ms	19 ms	1.45x

Table 1: *CPU time (for one simulation frame) comparison between Gauss-Seidel and accelerated Jacobi nonlinear constrained dynamics solvers (dual core).*

	Gauss-Seidel	Accelerated Jacobi	Speedup
2000 boxes	82.5 ms	40 ms	2.06x
100x100 cloth	15.4 ms	3.5 ms	4.4x
150x150 cloth	36.3 ms	15 ms	2.42x

Table 2: *CPU time (for one simulation frame) comparison between Gauss-Seidel and accelerated Jacobi nonlinear constrained dynamics solvers (quad core).*

Collision detection was done using both Bullet [Cou10] and our own triangle mesh tests. We implemented our own code because we needed continuous collision detection when performing tests versus cloth or for self-collisions. We accelerated these tests using OpenMP loops and a variant of dynamic AABB trees.

Most of the simulations in this paper were done in an offline manner and then exported as Alembic geometry caches to Autodesk Maya and rendered using Pixar RenderMan. Still, the simulator was written with real-time in mind and a lot of the scenarios ran at interactive rates, some even at 60 Hz. Generally we used a timestep $h = 16$ ms, gravity $g = -9.8m/s^{-2}$ and 10 to 50 iterations or more for our iterative solvers. For elastic bodies we used a Young's modulus $E = 0.5$ GPa and a Poisson ratio below 0.2. The masses of cloth and the soft bodies were raised up to around 10 kg in order interact smoothly with rigid bodies of unit mass or less.

7. Results

As you can see from the pictures (Figures 1 to 6) we were able to simulate a broad range of objects, both rigid and elastic. The novelty of our approach is that we can simulate all of them in the same constraint based solver and in this way obtain accurate and stable two way coupling almost out of the box.

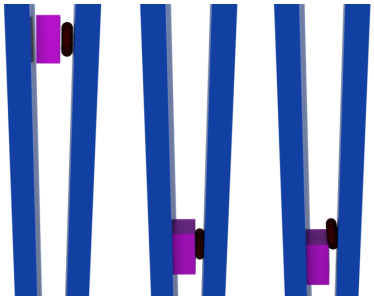

Figure 7: *Friction coupling between a rigid box and a deformable torus. From left to right: the initial position, our friction model and PBD velocity postprocessing friction ($\mu = 0.3$).*

In looking at our results we relied a lot on visual inspection. This was done especially to see if the system remained stable, which is quite hard to determine numerically. We focused more on the novelties we brought to these simulations: the coupling between rigid bodies and elastic materials (Figure 1 - middle and Figure 6), the PBD-like nonlinear approach to frictional contact and rigid bodies (Figure 1 - left), the nonlinear constraint based approach to viscoelasticity and the finite element method (Figure 1 - right) and the accelerated parallel Jacobi solver. All of these behaved robustly in our experiments. In Figure 7 you can see an experiment comparing our friction model with the simple velocity postprocessing step described in [MHHR07]. You can see in the middle that our method leads to a stable configuration, while on the right the torus slips and eventually looses contact with the box.

In order to test the accuracy of our smooth cone friction model we conducted the following experiment: a rigid box on a horizontal plane was given an initial sidewards velocity of 5 m/s. The theoretical result for the distance traveled until full stop is given by the formula $d = \frac{v_0^2}{2\mu g}$. In our case for $\mu = 0.1$ this value is 127.551m. We used for comparison the box LCP friction model [BETC14]. When the direction of the push was aligned with one of the tangential axes both models gave a numeric result around 127.17m. However, when pushed along the first bisector the box only moved for 89.78m in the box model ($\sqrt{2}$ less), while in the case of our model the result stayed roughly the same as before. This proves our model is more accurate as it is fully isotropic.

In Figure 8 you can see a comparison between the velocity based approach and our nonlinear position based method. Our conclusion is that for low iteration counts the methods are indiscernible but the balance tips in our favor for larger numbers of objects. The price for having less penetration is more violent contacts, i.e. very high velocities are needed to correct large initial penetration errors. These can be addressed through smaller time steps, larger collision tolerances or contact damping (which may introduce extra penetration for the same number of iterations). Still, position projection is a valid way of doing rigid body simulation with many gains at a marginally bigger computational cost (under 10%). Velocity time

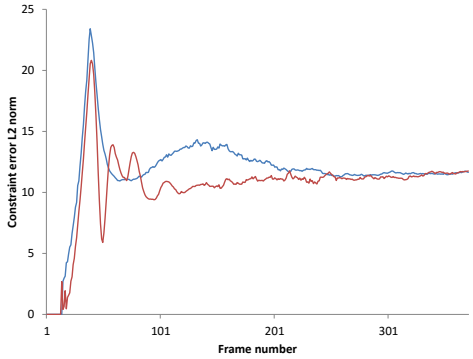

Figure 8: *VTS (blue) vs. PBD (red): plot of penetration depths (constraint error L2 norm - vertical axis) over time (under 400 frames - horizontal axis). Simulation scenario: 2000 rigid boxes falling on ground (12x15x12 cm, 10 iterations, $\mu = 0.3$, VTS stabilization factor $\gamma = 0.4$).*

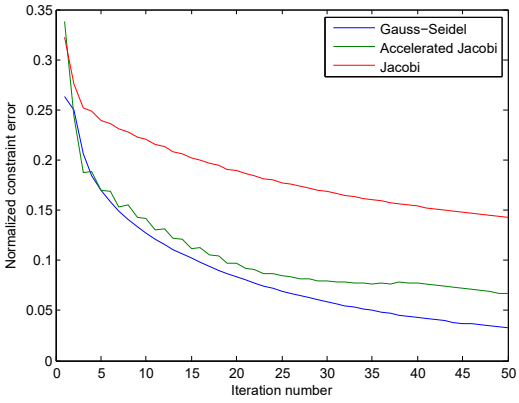

Figure 9: *Convergence plot of Gauss-Seidel (blue), Accelerated Jacobi (green) and Jacobi (red). At each iteration we measured the constraint error and plotted its evolution until the solver is halted prematurely after 50 iterations.*

stepping is still a very powerful method but from our experience it cannot be integrated in a stable manner and in the same solver with position based methods. However, velocity projection can be used with success as an initial step before position projection, enforcing better projection on the phase space constraint manifold. This works well especially for contacts, whereas for bilateral contacts is acts more as a damper.

The convergence rate of Accelerated Jacobi is depicted in Figure 9. The experiment was done in Matlab using 100 disks falling in a 2D box. We took the constraint error (penetration depth) L2 norm and normalized it against the maximum attained by each solver. Then we averaged each normalized error for each iteration number over 200 frames and obtained this plot. You can see clearly that Accelerated Jacobi performs much better than traditional Jacobi. In general, Accelerated Jacobi and Gauss-Seidel are comparable, with one out-performing the other depending on the context, but never too far from each other. We thus obtained a parallel contender to sequential Gauss-Seidel.

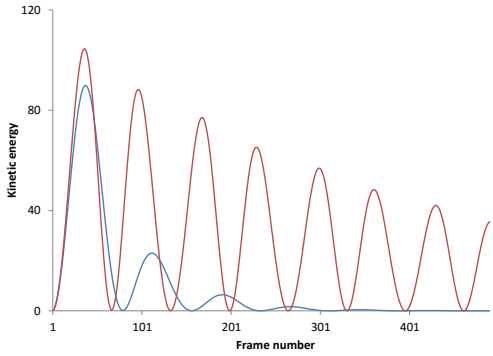

Figure 10: *Comparison of damped (blue, $\rho = 10h$) and normal (red) simulation of hanging cloth (30×30 grid).*

For testing our damping method we let a piece of cloth hang from two corners and observed the oscillations. As you can see in Figure 10 the kinetic energy of the undamped cloth keeps oscillating for a long time while the damped one goes to zero quite fast.

8. Conclusions

The main contribution of this paper is to introduce a mathematically sound formulation of position-based dynamics (PBD) relying on nonlinear convex optimization. This allows us to rigorously include contact and friction into position projection solvers for the first time. We demonstrate a working rigid body simulator using our novel CCP fixed point iteration scheme. The fully implicit and nonlinear scheme allows stable two-way coupling with deformable bodies. For soft bodies we present the a physically correct FEM solver using constraint regularization and PBD solvers. We add credible damping that only acts along constraint directions. We also derived a new accelerated form of the Jacobi solver that can compete against Gauss-Seidel and has the advantage of being parallel and unbiased.

Our position projection solver for contact and friction has some drawbacks that we hope to address in the future: violent impacts and possible convexification artifacts. Still, we preferred the smooth friction cone approach because it is more physical (i.e. isotropic) and can be expressed as a convex minimization (although the original problem is nonconvex).

9. Acknowledgements

We would like to thank prof. Dan Negrut and Hammad Mazhar from the University of Wisconsin-Madison for their support and feedback and also to Alin Dumitru and Liviu Dinu from StaticVFX for their help with rendering and more. Our department colleagues, Victor Asavei and Anca Morar, assisted this research in many ways and Sergiu Craitoiu helped with work on the deformable bodies.

References

[ACLM11] ACARY V., CADOUX F., LEMARÉCHAL C., MALICK J.: A Formulation of the Linear Discrete Coulomb Friction Problem via Convex Optimization. *ZAMM-Journal of Applied Mathematics and Mechanics 91*, 2 (2011), 155–175. 2, 5

[AH04] ANIȚESCU M., HART G. D.: A Constraint-Stabilized Time-Stepping Approach for Rigid Multibody Dynamics with Joints, Contact and Friction. *International Journal for Numerical Methods in Engineering 60*, 14 (2004), 2335–2371. 2

[Ani06] ANIȚESCU M.: Optimization-Based Simulation of Nonsmooth Rigid Multibody Dynamics. *Mathematical Programming 105*, 1 (2006), 113–143. 4, 5

[BETC14] BENDER J., ERLEBEN K., TRINKLE J., COUMANS E.: Interactive Simulation of Rigid Body Dynamics in Computer Graphics. *Computer Graphics Forum 33*, 1 (2014), 246–270. 2, 4, 5, 6, 8

[BKCW14] BENDER J., KOSCHIER D., CHARRIER P., WEBER D.: Position-Based Simulation of Continuous Materials. *Computers & Graphics 44* (2014), 1–10. 2

[BML*14] BOUAZIZ S., MARTIN S., LIU T., KAVAN L., PAULY M.: Projective Dynamics: Fusing Constraint Projections for Fast Simulation. *ACM Trans. Graph. 33*, 4 (2014), 154:1–154:11. 2, 3

[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. 1, 2, 5

[Cou10] COUMANS E.: Bullet Physics Engine. *Open Source Software: http://bulletphysics.org* (2010). 8

[DCB14] DEUL C., CHARRIER P., BENDER J.: Position-Based Rigid Body Dynamics. In *Proceedings of the 27th International Conference on Computer Animation and Social Agents* (May 2014). 1, 2

[Gol10] GOLDENTHAL A. R.: *Implicit Treatment of Constraints for Cloth Simulation*. PhD thesis, 2010. 1, 2, 3

[Jak01] JAKOBSEN T.: Advanced Character Physics. In *Game Developers Conference Proceedings* (2001), pp. 383–401. 1, 2, 6, 7

[KTS*14] KAUFMAN D. M., TAMSTORF R., SMITH B., AUBRY J.-M., GRINSPUN E.: Adaptive Nonlinearity for Collisions in Complex Rod Assemblies. *ACM Trans. Graph. 33*, 4 (July 2014), 123:1–123:12. 2, 3

[Lac07] LACOURSIÈRE C.: *Ghosts and Machines: Regularized Variational Methods for Interactive Simulations of Multibodies with Dry Frictional Contacts*. PhD thesis, Umeå University, Computing Science, 2007. 2, 3, 4

[MHHR07] MÜLLER M., HEIDELBERGER B., HENNIX M., RATCLIFF J.: Position Based Dynamics. *J. Vis. Comun. Image Represent. 18*, 2 (2007), 109–118. 1, 2, 5, 7, 8

[MHNT15] MAZHAR H., HEYN T., NEGRUȚ D., TASORA A.: Using Nesterov's Method to Accelerate Multibody Dynamics with Friction and Contact. *ACM Transactions on Graphics (TOG) 34*, 3 (2015), 32. 2, 3, 5, 6, 7

[MMC16] MACKLIN M., MÜLLER M., CHENTANEZ N.: XPBD: Position-based Simulation of Compliant Constrained Dynamics. In *Proceedings of the 9th International Conference on Motion in Games* (New York, NY, USA, 2016), MIG '16, ACM, pp. 49–54. 3

[MMCK14] MACKLIN M., MÜLLER M., CHENTANEZ N., KIM T.-Y.: Unified Particle Physics for Real-Time Applications. *ACM Transactions on Graphics (TOG) 33*, 4 (2014), 104. 2

[Saa03] SAAD Y.: *Iterative Methods for Sparse Linear Systems*, 2nd ed. Society for Industrial and Applied Mathematics, 2003. 6

[SB12] SIFAKIS E., BARBIC J.: FEM Simulation of 3D Deformable Solids: A Practitioner's Guide to Theory, Discretization and Model Reduction. In *ACM SIGGRAPH 2012 Courses* (New York, NY, USA, 2012), SIGGRAPH '12, ACM, pp. 20:1–20:50. 2, 6

[SLM06] SERVIN M., LACOURSIÈRE C., MELIN N.: Interactive Simulation of Elastic Deformable Materials. In *SIGRAD 2006 Conference Proceedings* (2006), pp. 22–32. 1, 2, 3, 6

[ST96] STEWART D., TRINKLE J. C.: An Implicit Time-Stepping Scheme for Rigid Body Dynamics with Coulomb Friction. *International Journal for Numerical Methods in Engineering 39* (1996), 2673–2691. 2, 4, 5

[TA11] TASORA A., ANIȚESCU M.: A Matrix-Free Cone Complementarity Approach for Solving Large-Scale, Nonsmooth, Rigid Body Dynamics. *Computer Methods in Applied Mechanics and Engineering 200*, 5 (2011), 439–453. 2, 4, 5, 6, 7

[TNGF15] TOURNIER M., NESME M., GILLES B., FAURE F.: Stable Constrained Dynamics. *ACM Trans. Graph. 34*, 4 (July 2015), 132:1–132:10. 1, 2, 3, 4

[WN99] WRIGHT S. J., NOCEDAL J.: *Numerical Optimization*. Springer New York, 1999. 3, 6

Appendix A: Jacobians for constraint based FEM

The constraint function is $\Psi(\mathbf{x}) = \sqrt{V}\varepsilon$, where V is the tetrahedron volume and $\mathbf{x}$ is made up of the four tetrahedron vertices: $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$. There are 6 constraints corresponding to the 6 unique values making up the symmetric strain tensor. These can be split in two: a normal part and a shear part. Let Λ be a helper matrix of the same dimension as the Jacobian (6x12). It too can be split in two 3x12 matrices Λ_n and Λ_s and the normal one is:

$$\Lambda_n = \begin{bmatrix} \Lambda_n^0 & \Lambda_n^1 & \Lambda_n^2 & \Lambda_n^3 \end{bmatrix}, \tag{36}$$

where $\Lambda_n^i = \mathrm{diag}(\mathbf{y}_i)$, $\mathbf{y}_1$ to $\mathbf{y}_3$ are the rows of a matrix $\mathbf{X}$ and $\mathbf{y}_0 = -\mathbf{y}_1 - \mathbf{y}_2 - \mathbf{y}_3$. The matrix $\mathbf{X}$ is the inverse of the initial shape matrix $\mathbf{D}_s$. We are assuming linear finite elements, i.e. tetrahedra of constant strain. For shear we define a similar helper matrix:

$$\Lambda_s = \frac{1}{2}\begin{bmatrix} \Lambda_s^0 & \Lambda_s^1 & \Lambda_s^2 & \Lambda_s^3 \end{bmatrix}, \tag{37}$$

where

$$\Lambda_s^i = \begin{bmatrix} 0 & y_{iz} & y_{iy} \\ y_{iz} & 0 & y_{ix} \\ y_{iy} & y_{ix} & 0 \end{bmatrix}. \tag{38}$$

The normal and shear Jacobians of Ψ are then:

$$\mathbf{J}_\alpha = \sqrt{V}\Lambda_\alpha \mathbf{F}^T + \frac{1}{2\sqrt{(V)}}\varepsilon_\alpha \nabla V, \tag{39}$$

where $\alpha \in \{n, s\}$, $\mathbf{F} = \mathbf{D}_m\mathbf{X}$ is the deformation gradient, $\varepsilon_n = (\varepsilon_{11}, \varepsilon_{22}, \varepsilon_{33})$ and $\varepsilon_s = (\varepsilon_{23}, \varepsilon_{13}, \varepsilon_{12})$. The gradient of the volume is then a 12 component row vector consisting of the following partial derivatives:

$$\nabla V = \left(\frac{\partial V}{\partial \mathbf{x}_0}\ \frac{\partial V}{\partial \mathbf{x}_1}\ \frac{\partial V}{\partial \mathbf{x}_2}\ \frac{\partial V}{\partial \mathbf{x}_3}\right), \tag{40}$$

where

$$\frac{\partial V}{\partial \mathbf{x}_1} = \frac{1}{6}(\mathbf{x}_2 - \mathbf{x}_0) \times (\mathbf{x}_3 - \mathbf{x}_0), \tag{41}$$

$$\frac{\partial V}{\partial \mathbf{x}_2} = \frac{1}{6}(\mathbf{x}_3 - \mathbf{x}_0) \times (\mathbf{x}_1 - \mathbf{x}_0), \tag{42}$$

$$\frac{\partial V}{\partial \mathbf{x}_3} = \frac{1}{6}(\mathbf{x}_1 - \mathbf{x}_0) \times (\mathbf{x}_2 - \mathbf{x}_0), \tag{43}$$

$$\frac{\partial V}{\partial \mathbf{x}_0} = -\frac{\partial V}{\partial \mathbf{x}_1} - \frac{\partial V}{\partial \mathbf{x}_2} - \frac{\partial V}{\partial \mathbf{x}_3}. \tag{44}$$