Projective Dynamics: Fusing Constraint Projections for Fast Simulation

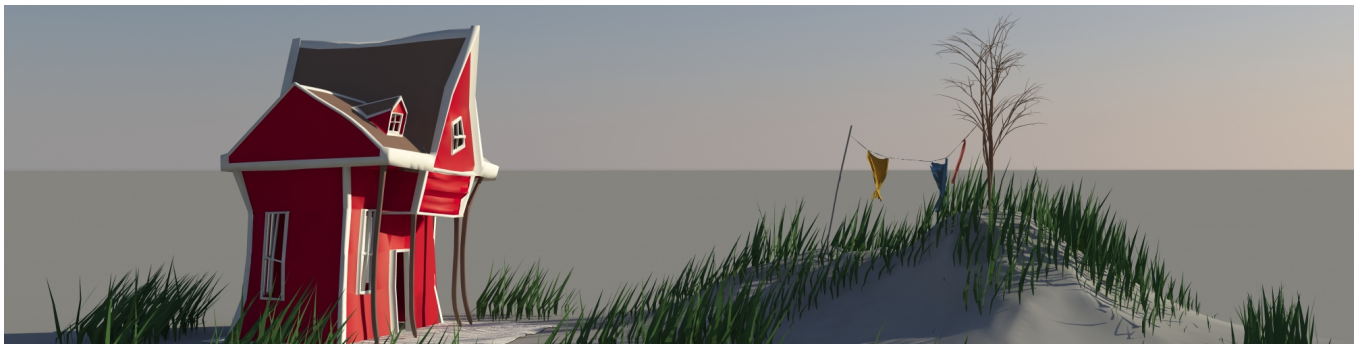

Figure 1: *We propose a new "projection-based" implicit Euler integrator that supports a large variety of geometric constraints in a single physical simulation framework. In this example, all the elements including building, grass, tree, and clothes (49k DoFs, 43k constraints), are simulated at 3.1ms/iteration using 10 iterations per frame (see also accompanying video).*

Abstract

We present a new method for implicit time integration of physical systems. Our approach builds a bridge between Finite Element methods and Position Based Dynamics, leading to a simple, efficient, robust, yet accurate solver that supports many different types of constraints. We propose specially designed energy potentials that can be solved efficiently using an alternating optimization approach. Inspired by continuum mechanics, we derive a set of continuum-based potentials that can be efficiently incorporated within our solver. We demonstrate the generality and robustness of our approach in many different applications ranging from the simulation of solids, cloths, and shells, to example-based simulation. Comparisons to Newton-based and Position Based Dynamics solvers highlight the benefits of our formulation.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics—Animation; I.6.8 [Simulation and Modeling]: Types of Simulation—Animation

Keywords: physics-based animation, implicit Euler method, position based dynamics, continuum mechanics.

1 Introduction

Physically-based simulation of deformable material has become an indispensable tool in many areas of computer graphics. Virtual worlds and more recently character animations incorporate sophisticated simulations to greatly enhance visual experience, e.g., by simulating muscles, fat, hair, clothing, or vegetation. These models are often based on finite element discretizations of continuum-mechanics formulations, allowing highly *accurate* simulation of complex non-linear materials.

Besides realism and accuracy, a number of other criteria are also important in computer graphics applications. By *generality* we mean the ability to simulate a large spectrum of behaviors, such as different types of geometries (solids, shells, rods), different material properties, or even art-directable extensions to classic physically-based simulation. *Robustness* refers to the capability to adequately handle difficult configurations, such as large deformations, degenerate geometries, and large time steps. Robustness is especially important in real-time applications where there is no "second chance" to re-run a simulation, such as in computer games or medical training simulators. The *simplicity* of a solver is often important for its practical relevance. Building on simple, easily understandable concepts – and the resulting lightweight codebases – eases the maintenance of simulators and makes them adaptable to specific application needs. *Performance* is a critical enabling criterion for realtime applications. However, performance is no less important in offline simulations, where the turnaround time for testing new scenes and simulation parameters should be minimized.

Current continuum mechanics approaches often have unfavorable trade-offs between these criteria for certain computer graphics applications, which led to the development of alternative methods, such as Position Based Dynamics (PBD). Due to its generality, simplicity, robustness, and efficiency, PBD is now implemented in a wide range of high-end products such as PhysX, Havok Cloth, Maya nCloth, and Bullet. While predominantly used in realtime applications, PBD is also often used in offline simulation. However, the desirable qualities of PBD come at the cost of limited accuracy, because PBD is not rigorously derived from continuum mechanical principles.

We propose a new implicit integration solver that bridges the gap between continuum mechanics and PBD. The key idea is to introduce energy potentials with a specific structure. More precisely, our potentials consist of a convex quadratic distance measure from a *constraint*. The constraints are general nonlinear functions that express the desired state of an element, for example that the volume of a tetrahedron must remain within given bounds. The distance measure quantifies how much individual constraints are violated in a given deformed configuration. While our solver can handle arbitrary geometric constraints, we propose a specific set of constraints derived from continuous deformation energies. These continuum-based constraints are very practical because they considerably simplify parameter tuning especially when dealing with meshes of different resolutions and non-uniform tessellation.

The main advantage of our constraint-based potentials is that their structure enables an efficient local/global optimization (block coordinate descent). Specifically, the local step consists of projecting every element onto the constraint manifold, i.e., solving a small nonlinear problem per element. The global step combines the results of individuals projections, finding a compromise between all of the individual constraints, while also taking into account global

effects such as inertia and external forces.

The local/global approach allows us to formulate an implicit integration solver that is guaranteed to weakly decrease the energy in every iteration without requiring any specific precautions. This contrasts with classical Newton's method which requires line search strategies and safeguards against singular or indefinite Hessians to guarantee robustness. Furthermore, with a fixed set of constraints, we can pre-factor the linear system of the global step, which greatly reduces computation time. The local steps consists of small independent optimization problems, which can be all executed in parallel.

To our knowledge, our method is the first to apply local/global optimization to simulate general dynamical systems. We demonstrate that this solution provides a robust and efficient approach to implicit integration, often significantly outperforming the classical Newton method. The connection between PBD and our solver reveals new interesting insights on how PBD relates to traditional approaches based on finite element methods and Newtonian mechanics.

2 Related Work

Since the pioneering work of Terzopulous and colleagues [1987], models derived from continuum mechanics play an important role in physics-based animation. The basic principle is that the resistance of an elastic object to deformations is quantified using an elastic potential energy – a scalar function whose variational derivative leads to the elastic force [Sifakis and Barbic 2012]. Unfortunately, the elastic forces are usually non-linear even for basic material models, which complicates time integration of the resulting equations of motion.

The simplest time integration schemes used in computer graphics are explicit and very fragile to large time steps [Press et al. 2007]. Implicit Euler methods significantly improve robustness [Baraff and Witkin 1998], but at the cost of solving a system of non-linear equations at every step. As shown in [Martin et al. 2011], this can be equivalently formulated as a non-convex optimization problem that operates directly on elastic potentials instead of forces. One of the main shortcomings of implicit Euler integration is artificial numerical damping. This motivated the development of symplectic integrators [Kharevych et al. 2006; Stern and Desbrun 2006] and mixed implicit-explicit methods (IMEX) [Bridson et al. 2003; Stern and Grinspun 2009], featuring better energy conservation properties. Another approach is *energy budgeting* [Su et al. 2013] which enforces energy conservation explicitly. However, implicit Euler integration continues to be one of the popular choices in applications of physics-based animation where robustness is an important criteria and numerical damping is not a major concern. Our solver is derived from the variational form of implicit Euler integration [Martin et al. 2011] as it gives an intuitive way of thinking about time integration in our framework – simply by adding another constraint to the system. This further allows us to draw connections between PBD and the implicit Euler integration scheme and results in a robust and efficient approach that is stable under large time steps.

Regardless of the particular flavor and formulation of implicit integration, Newton's method remains the computational workhorse for solving the system of non-linear equations. However, its robust implementation requires precautions such as conservative line search procedures and safeguards against indefinite Hessians [Boyd and Vandenberghe 2004]. From a performance standpoint, a serious drawback of Newton's method is the fact that the Hessian matrix and the gradient change at every iteration. Quasi-Newton methods therefore employ approximate Hessians, trading faster linear system solves for suboptimal descent directions (and therefore slower convergence) [Desbrun et al. 1999; Hahn et al. 2012]. A similar strategy, explored in the context of co-rotated elasticity, is to use carefully scheduled updates of sparse Cholesky factorization [Hecht et al. 2012]. Recently, Liu and colleagues [2013] presented a method for efficient implicit time integration of mass-spring systems by introducing auxiliary variables that enable alternating local/global optimization. This approach, also known as block coordinate descent, has been previously used with great success in geometry processing [Sorkine and Alexa 2007; Bouaziz et al. 2012]. We also employ local/global alternation in our approach, but contrary to [Liu et al. 2013] we aim at simulating *general* dynamical systems.

Our constraint-based formulation bears some similarity with recent non-traditional approaches based on constraint projection. The idea of constraint projection is central to the Nucleus system [Stam 2009] and Position Based Dynamics [Müller et al. 2007; Bender et al. 2013]. In contrast to our solution, these methods do not treat the constraints in a global manner, but iteratively project onto them in a (non-linear) Gauss-Seidel-like fashion [Müller et al. 2007]. While the resulting algorithm is very easy to implement, this approach has a number of shortcomings: the Gauss-Seidel optimization does not converge very rapidly, the material stiffness depends on the number of iterations, and the result depends on the traversal order. In contrast, our method uses constraints to formulate elastic potentials that are rigorously combined with inertial terms as dictated by Newton's laws of motion. Our solver first computes all constraint projections separately and then finds the best compromise between them, which makes the solution independent of the order of constraints. To obtain faster convergence, constraints are expressed using differential coordinates, which often yields satisfactory results after just a few iterations. Furthermore, our solver converges to a true implicit Euler solution with our elastic energy, in contrast to Position Based Dynamics which converges to completely inelastic behavior.

Another closely related concept is shape matching [Müller et al. 2005; Rivers and James 2007] where, contrary to us, constraint projections are used to directly build elastic forces instead of potentials to simulate deformable objects. Constraint projections were also used in *strain limiting* [Provot 1995; Goldenthal et al. 2007; Thomaszewski et al. 2009; Wang et al. 2010; Narain et al. 2012] not as a standalone simulation technique but rather as a way to improve handling of stiff systems with standard time integration methods. In our approach we can also perform strain limiting but are including it directly into the implicit solver.

3 Continuum Mechanics View

In this section we introduce the special structure of our potentials which form the basis of our method. We start with the implicit time integration of FEM-discretized elastic models.

3.1 Implicit Euler Solver

Let us briefly review the variational form of implicit Euler integration [Martin et al. 2011]. We assume a mesh consisting of m vertices with positions $\mathbf{q} \in \mathbb{R}^{m\times 3}$ and velocities $\mathbf{v} \in \mathbb{R}^{m\times 3}$. The system evolves in time according to Newton's laws of motion through a discrete set of time samples $t_1, t_2, \ldots$. At time t_n, the system is defined as $\{\mathbf{q}_n, \mathbf{v}_n\}$. The sum of the external forces is defined as $\mathbf{f}_{\text{ext}}$ and the sum of internal forces as $\mathbf{f}_{\text{int}}$. We consider position dependent internal forces such that $\mathbf{f}_{\text{int}} = -\sum_i \nabla_{\mathbf{q}} W_i(\mathbf{q})$, where $W_i(\mathbf{q})$ is a scalar potential energy function. Implicit Euler time

integration results in the following update rule:

$$\mathbf{q}_{n+1} = \mathbf{q}_n + h\mathbf{v}_{n+1}$$
$$\mathbf{v}_{n+1} = \mathbf{v}_n + h\mathbf{M}^{-1}(\mathbf{f}_{\text{int}} + \mathbf{f}_{\text{ext}})$$

where $\mathbf{M}$ is the mass-matrix and h represents the simulation step size. Using these equations we can derive

$$\mathbf{M}(\mathbf{q}_{n+1} - \mathbf{q}_n - h\mathbf{v}_n) = h^2(\mathbf{f}_{\text{int}} + \mathbf{f}_{\text{ext}}).$$

This system can be converted to an optimization problem

$$\arg\min_{\mathbf{q}_{n+1}} \frac{1}{2h^2}\|\mathbf{M}^{\frac{1}{2}}(\mathbf{q}_{n+1} - \mathbf{s}_n)\|_F^2 + \sum_i W_i(\mathbf{q}_{n+1}), \quad (1)$$

where $\mathbf{s}_n = \mathbf{q}_n + h\mathbf{v}_n + h^2\mathbf{M}^{-1}\mathbf{f}_{\text{ext}}$. Intuitively, this minimization problem describes the compromise between the *momentum potential*

$$\frac{1}{2h^2}\|\mathbf{M}^{\frac{1}{2}}(\mathbf{q}_{n+1} - \mathbf{s}_n)\|_F^2, \quad (2)$$

which states that the solution should follow its momentum (plus external forces), and the elastic potential, that requires the solution to minimize the elastic deformation. The corresponding weighting terms, i.e., the mass distribution in $\mathbf{M}$, the time step h and the material stiffness of W, determine which potential has more importance in this balance. Furthermore, according to Nöther's theorem, linear and angular momenta are always conserved when the elastic potential is rigid motion invariant.

The minimization of Equation 1 is commonly performed using careful implementations of Newton's method [Martin et al. 2011]. However, this is quite costly because at each iteration a different linear system needs to be solved, as the Hessian changes from one iteration to the next. To simplify notation, we will drop below the subscript in $\mathbf{q}_{n+1}$ and just use $\mathbf{q}$.

3.2 Nonlinear Elasticity

We analyze the classical form of FEM-based nonlinear elastic energies to reveal how we can restrict the elastic potentials in Equation 1 to a structure that will allow deriving our novel solver.

Nonlinear elastic potentials. In nonlinear continuum mechanics the deformation from a rest state is measured using a discrete, elemental strain $\mathbf{E}(\mathbf{q})$, e.g., the quadratic Green's strain [Irving et al. 2004]. Numerous elastic potentials used in practice are formulated as a function of the strain using a (often nonlinear) material model $\Psi(\cdot)$, resulting in elastic potentials $W(\mathbf{q}) = \Psi(\mathbf{E}(\mathbf{q}))$. From a geometric point of view, we can observe that $\mathbf{E}(\mathbf{q}) = \mathbf{0}$ defines a constraint manifold of all possible undeformed configurations, while $\Psi(\mathbf{E}(\mathbf{q}))$ measures how far the deformed configuration is from this manifold (level sets in Figure 2). Our key observation is that these two concepts can be decoupled; the distance metric does not have to be a complicated nonlinear function because the nonlinearities are already captured by the constraint manifold.

Decoupling distance measure and constraint manifold. We introduce potential functions W that make use of an auxiliary variable $\mathbf{p}$ as

$$W(\mathbf{q}, \mathbf{p}) = d(\mathbf{q}, \mathbf{p}) + \delta_{\mathbf{E}}(\mathbf{p}). \quad (3)$$

Here, $\delta_{\mathbf{E}}(\mathbf{p})$ is an indicator function that evaluates to zero if $\mathbf{E}(\mathbf{p}) = \mathbf{0}$ and to $+\infty$ otherwise, and formalizes the requirement that $\mathbf{p}$ should lie on the constraint manifold. The function $d(\mathbf{q}, \mathbf{p})$ then measures a distance between $\mathbf{q}$ and $\mathbf{p}$. Minimizing Equation 3 over $\mathbf{p}$ corresponds to a projection of $\mathbf{q}$ onto the constraint manifold, as illustrated in Figure 2. An elastic potential analogous to $\Psi(\mathbf{E}(\mathbf{q}))$ could therefore be defined as $\tilde{W}(\mathbf{q}) = \min_{\mathbf{p}} W(\mathbf{q}, \mathbf{p})$.

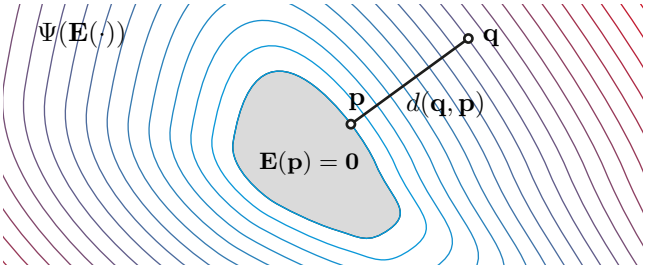

Figure 2: *The function $\Psi(\mathbf{E}(\cdot))$ defines both the constraint manifold $\mathbf{E}(\cdot) = \mathbf{0}$ as its zero level set and the elastic potential given by its isolines. By introducing a projection variable $\mathbf{p}$ in the manifold, we can decouple the manifold definition from the elastic potential, modeled as the distance function $d(\mathbf{q}, \mathbf{p})$.*

Quadratic distance measures. With this separation in mind, we can build a solver that alternates between distance minimization and projection. An important advantage of this formulation is that the distance measure can be freely chosen. The *constraint nonlinearity* (also known as geometric nonlinearity) is already taken care of by the projection on the constraint set, so the distance metric can be kept simple, trading general *material nonlinearity* against efficiency and robustness. Specifically, we consider distance metrics leading to the following potentials:

$$W(\mathbf{q}, \mathbf{p}) = \frac{w}{2}\|\mathbf{A}\mathbf{q} - \mathbf{B}\mathbf{p}\|_F^2 + \delta_{\mathbf{C}}(\mathbf{p}), \quad (4)$$

where $\mathbf{A}$ and $\mathbf{B}$ are constant matrices. The distance to the constraint set is thus modeled by a *quadratic* function in $\mathbf{q}$ and $\mathbf{p}$, which allows us to deploy an efficient solver. Moreover, we are not restricted to Green's strains but can use any constraint definition $\mathbf{C}(\mathbf{q}) = \mathbf{0}$ for the set of desired configurations, e.g. describing desired bending angles between triangles, goal volumes for tetrahedrons, or boundary conditions, as discussed below.

3.3 Projective Implicit Euler Solver

Using simplified potentials as given in Equation 4, we can reformulate the implicit integration defined in Equation 1 as the minimization of

$$\frac{1}{2h^2}\|\mathbf{M}^{\frac{1}{2}}(\mathbf{q} - \mathbf{s}_n)\|_F^2 + \sum_i \frac{w_i}{2}\|\mathbf{A}_i\mathbf{S}_i\mathbf{q} - \mathbf{B}_i\mathbf{p}_i\|_F^2 + \delta_{\mathbf{C}_i}(\mathbf{p}_i) \quad (5)$$

over $\mathbf{q}$ and the auxiliary variables $\mathbf{p}_i$, where $\mathbf{S}_i$ is a constant selection matrix that selects the vertices involved in the ith constraint. We minimize Equation 5 using a local/global alternating minimization technique.

Local solve. First, we minimize Equation 5 over the auxiliary variables keeping the positions fixed by solving

$$\arg\min_{\mathbf{p}_i} \sum_i \frac{w_i}{2}\|\mathbf{A}_i\mathbf{S}_i\mathbf{q} - \mathbf{B}_i\mathbf{p}_i\|_F^2 + \delta_{\mathbf{C}_i}(\mathbf{p}_i).$$

Since each constraint has its own set of auxiliary variables $\mathbf{p}_i$, this minimization can be performed independently for each constraint, which allows massive parallelization of the local step. We will discuss specific constraint types in Section 5.

Algorithm 1: Projective Implicit Euler Solver

1 $\mathbf{s}_n = \mathbf{q}_n + h\mathbf{v}_n + h^2\mathbf{M}^{-1}\mathbf{f}_{\text{ext}}$
2 $\mathbf{q}_{n+1} = \mathbf{s}_n$
3 **loop** *solverIteration* **times**
4 **forall the** *constraints j* **do**
5 $\mathbf{p}_j = ProjectOnConstraintSet(\mathbf{C}_j, \mathbf{q}_{n+1})$
6 **end**
7 $\mathbf{q}_{n+1} = SolveLinearSystem(\mathbf{s}_n, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \ldots)$
8 **end**
9 $\mathbf{v}_{n+1} = (\mathbf{q}_{n+1} - \mathbf{q}_n)/h$

Global solve. Second, we minimize Equation 5 over the positions, keeping the auxiliary variables fixed. Since Equation 5 is quadratic in the unknowns $\mathbf{q}$, we can minimize it with a single linear solve. Requiring that the gradient vanishes at the critical point leads to the linear system

$$(\frac{\mathbf{M}}{h^2} + \sum_i w_i \mathbf{S}_i^T \mathbf{A}_i^T \mathbf{A}_i \mathbf{S}_i)\mathbf{q} = \frac{\mathbf{M}}{h^2}\mathbf{s}_n + \sum_i w_i \mathbf{S}_i^T \mathbf{A}_i^T \mathbf{B}_i \mathbf{p}_i.$$

The system matrix is constant as long as the constraints are not changing and therefore can be prefactored at initialization, allowing for very efficient global solves. The right hand side requires recomputation in each iteration after the projection variables have been updated in the local step. Note that the objective is bounded below and that both local and global steps are guaranteed to weakly decrease it. Consequently, the optimization converges, making safeguards unnecessary.

Algorithm. We summarize our optimization procedure in Algorithm 1. On line 2 we warm start the optimization using the momentum estimate $\mathbf{s}_n$. We observe that this is favorable when using only few solver iterations, leading to less damped systems than when using the last time step's solution as starting point. After solving multiple local/global iterations the velocities are updated in line 9.

Choice of $\mathbf{A}$ and $\mathbf{B}$. If we choose $\mathbf{A}_i = \mathbf{B}_i = \mathbf{I}$, Equation 4 measures the squared *Euclidean* distance from $\mathbf{S}_i\mathbf{q}$ to its closest point on the constraint set. With diagonal $\mathbf{M}$, e.g., a lumped mass matrix, the Hessian of the global solve ends up being diagonal as well, leading to a trivial linear system to solve. However, this choice corresponds to working directly with absolute positions, which results in a poor convergence rate because changes propagate slowly through the (usually locally) coupled points [Bouaziz et al. 2012].

The convergence can be greatly improved if we make use of the fact that most of the internal physical constraints are translation invariant (i.e. applying a common translation to all involved points in the constraints does not change the values of the constraints). In this case, we can choose $\mathbf{A}_i = \mathbf{B}_i$ as differential coordinate matrices (global translation in their null space). Various such matrices can be used, for example one can subtract the mean [Bouaziz et al. 2012] or simply one of the vertices involved in the constraint [Liu et al. 2013]. Note that the choice of $\mathbf{A}_i$ and $\mathbf{B}_i$ only impacts the numerical solution procedure and does not affect the conservation of momentum.

Using such differential coordinates greatly improves the convergence speed of the resulting local/global solver [Bouaziz et al. 2012]. However, without further precautions, the resulting behavior is tessellation and resolution dependent. We show in Section 5 that in certain cases the $\mathbf{A}_i$ and $\mathbf{B}_i$ matrices can be derived from continuum formulations in order to avoid these shortcomings.

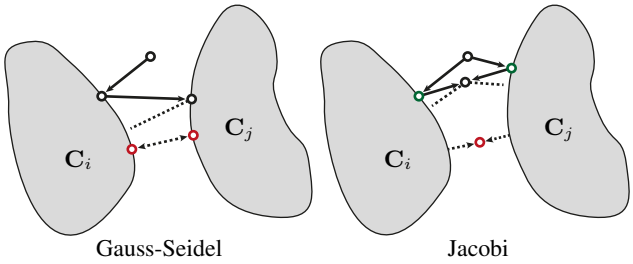

Figure 3: *Gauss-Seidel vs. Jacobi. The Gauss-Seidel algorithm used in PBD consecutively projects the current estimate on each constraint set ($\mathbf{C}_i$ and $\mathbf{C}_j$ in this case). If there is no feasible solution, i.e. the constraint sets do not overlap, the Gauss-Seidel algorithm will oscillate between the different constraints (between the two red points). On the contrary, the Jacobi algorithm projects the current estimate on each constraint set in parallel (green points) and reaches a consensus in a second step. This allows the Jacobi algorithm to converge (red point).*

4 Position Based Dynamics View

In this section we look at PBD from the point of view of the implicit Euler solver presented in the previous section. This analysis highlights the close connections of PBD to our solver, but also identifies fundamental differences that explain the higher accuracy of results obtained with our approach.

Conservation of linear momentum. In PBD, the points $\mathbf{q}$ are moved by displacements $\Delta\mathbf{q}$ to satisfy the constraints $\mathbf{C}$, i.e., for the jth constraint we have $\mathbf{C}_j(\mathbf{S}_j\mathbf{q} + \Delta\mathbf{q}_j) = \mathbf{0}$. Additionally, linear momentum needs to be conserved, i.e.

$$\sum_j \mathbf{M}_j \Delta\mathbf{q}_j = \mathbf{0},$$

using a lumped mass matrix $\mathbf{M}_j$ only involving the constraint's points. Equivalently, we can rewrite conservation of momentum as the critical point condition of

$$\frac{1}{2}\sum_j \|\mathbf{M}_j^{\frac{1}{2}} \Delta\mathbf{q}_j\|_F^2.$$

By introducing the auxiliary projection variables $\mathbf{p}_j = \mathbf{S}_j\mathbf{q} + \Delta\mathbf{q}_j$ for each constraint and requiring that the projection $\mathbf{p}_j$ has to satisfy the constraint $\mathbf{C}_j$, we can rewrite this energy as

$$\sum_j \|\mathbf{M}_j^{\frac{1}{2}}(\mathbf{S}_j\mathbf{q} - \mathbf{p}_j)\|_F^2 + \delta_{\mathbf{C}_j}(\mathbf{p}_j). \quad (6)$$

It is interesting to note that in this view the momentum term in Equation 2 can be seen as decoupled potentials where the constraint set is defined by the *momentum constraint* $\mathbf{q}_{n+1} - \mathbf{s}_n = \mathbf{0}$. Based on this formulation, let us now review how the PBD solver proceeds.

4.1 Gauss Seidel Solver

A classical PBD solver performs three steps. In the first step, the positions are initialized by performing an explicit Euler step, ignoring internal forces, i.e. by solving Equation 2. In the second step, the positions are updated by minimizing the energy given in Equation 6 in a Gauss-Seidel fashion, i.e. the current configuration is consecutively projected on each constraint set by minimizing Equation 6

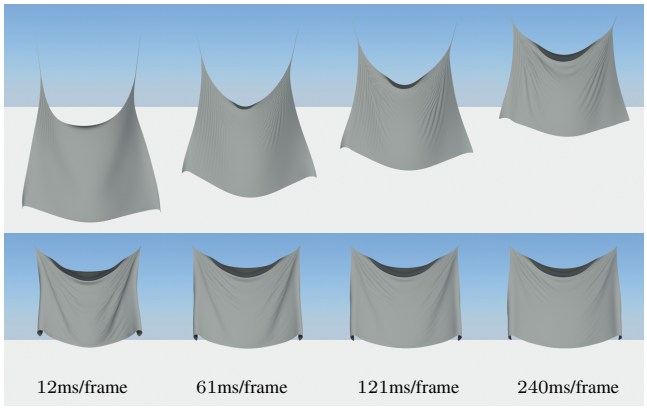

Figure 4: *For a piece of cloth with 19683 DoFs and 19360 edge constraints, PBD exhibits different material stiffness depending on the allowed time budget for a time step (top). Due to the additional momentum term and the differential coordinate formulation, our simulation behaves consistently even for different number of iterations (bottom).*

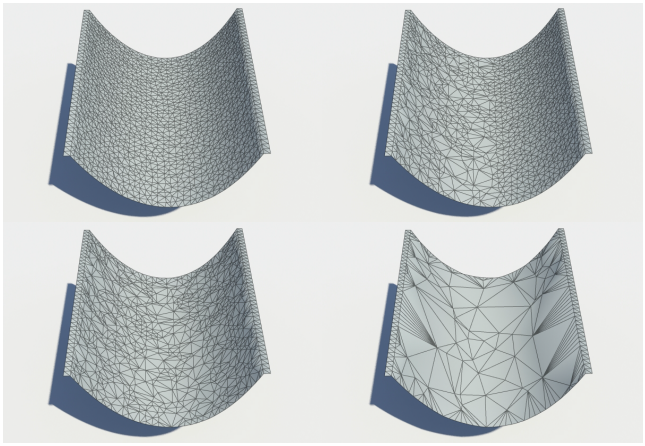

Figure 5: *For a given continuous surface, discretizing our continuum based constraints on piecewise simplicial approximations of different resolutions results in very similar qualitative behaviors.*

separately for each constraint j. In the last step, the velocities are updated as $\mathbf{v}_{n+1} = (\mathbf{q}_{n+1} - \mathbf{q}_n)/h$.

Theoretically, Gauss-Seidel has good convergence, however only for feasible constraint sets. For non-feasible sets, lacking a global view on the optimization problem, Gauss-Seidel will oscillate between the incompatible sets (see Figure 3). As an example, when simulating the compression of an elastic material with stretch constraints and boundary conditions or collisions, the constraint sets can become unfeasible and thus the solution will oscillate and not converge.

More severely, the same is true for the momentum estimation performed in the first step, which consists of first solving the constraint given in Equation 2. If added as a true constraint to the optimization, it could lead to completely incompatible constraint sets and make convergence even worse. By solving the momentum constraint first with the initial explicit Euler step, it is possible to maintain the linear momentum of the entire object, however the individual momenta of the points are washed away the longer the optimization iterates – contrary to finding a compromise between momentum and internal elasticity as suggested by the Implicit Euler solver that we propose (see Figure 4).

4.2 Jacobi Solver

In the view of Equation 6, we can solve this issue in a straightforward manner by performing two steps. First, we introduce the momentum constraint into the optimization to take into account the inertia of each point. Second, we replace the Gauss-Seidel by a Jacobi solver (see Figure 3) that is able to deal with incompatible constraints. Jacobi solvers have in general slower convergence than Gauss-Seidel solvers [Thomaszewski et al. 2009]. However, they allow the use of differential coordinate representations for faster convergence and efficient parallelization of the constraint projections that resolve this shortcoming.

As seen in the continuum mechanics view, to achieve a correct behavior we need to add back the inertia of each point by integrating the momentum constraint term defined in Equation 2 into the optimization

$$\frac{1}{2h^2}\|\mathbf{M}^{\frac{1}{2}}(\mathbf{q}-\mathbf{s}_n)\|_F^2 + \sum_j \frac{w_j}{2}\|\mathbf{M}_j^{\frac{1}{2}}(\mathbf{S}_j\mathbf{q}-\mathbf{p}_j)\|_F^2 + \delta_{\mathbf{C}_j}(\mathbf{p}_j).$$

The Jacobi solver then becomes a two-step optimization: In the local step, the current solution $\mathbf{q}$ is first projected onto the constraints independently by solving Equation 6 for all $\mathbf{p}_j$. Then, a consensus can be reached between the different solutions by solving the global step over $\mathbf{q}$.

Connection to Projective Implicit Euler. At this stage, we can see how close this Jacobi solver derived from PBD is to our projective implicit solver procedure presented in the last section – we recover this solver by choosing $\mathbf{A}_j = \mathbf{B}_j = \mathbf{M}_j^{\frac{1}{2}}$. By deriving constraints from a continuum principle in the next section we furthermore achieve better independence on mesh tessellation and convergence than with the simpler mass-based weighting used in PBD (see Figure 5).

5 Continuum-Based Constraints

Differential representations are important for our local/global solver to improve convergence. In geometry processing the gradient and the Laplace-Beltrami operators play an essential role in the design of efficient and robust models. In this section we will present a set of continuous energies based on these operators that allow the control of the differential properties of the material under deformation. We will show that their discretizations will have a form similar to Equation 4 that allow for correct behavior under mesh refinement and non-uniform discretizations. The local optimization of the discrete potentials will be discussed in Appendix A.

5.1 Strain and Its Limiting

Continuous energy. Strain energies are important for simulating materials that can stretch. We first discuss 2-manifold surfaces and then extend the results to volumes and curves. Let the undeformed surface be a differentiable 2-manifold surface S embedded in $\mathbb{R}^3$. We define the piecewise linear coordinate function of the undeformed surface by $\mathbf{g} : S \to \mathbb{R}^3$ and its deformed counterpart by $\mathbf{f} : S \to \mathbb{R}^3$. Introducing a set M of desired transformations $\mathbf{T}$, we formulate an energy measuring the change of local variation between the deformed and the undeformed surface as

$$E(\mathbf{f},\mathbf{T}) = \frac{w}{2}\int_S \|\nabla_S\mathbf{f} - \mathbf{T}\nabla_S\mathbf{g}\|_F^2 + \delta_M(\mathbf{T})\,\mathrm{d}A, \qquad (7)$$

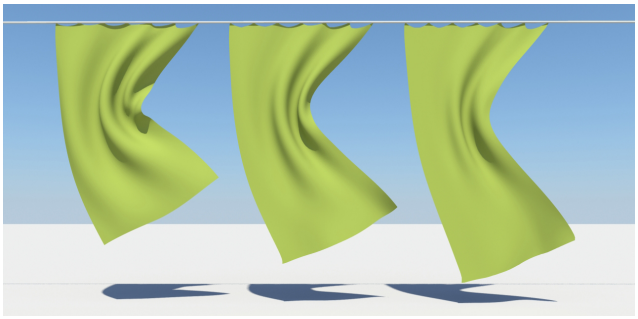

Figure 6: *Starting from the same mesh, strain limiting allows simulating material that can undergo small to moderate amount of stretching. From left to right, we use strain limits of [-10%, +10%], [-20%, +20%] and [-30%, +30%]. Notice how the cloth stretches and how the folds get absorbed when the limit increases.*

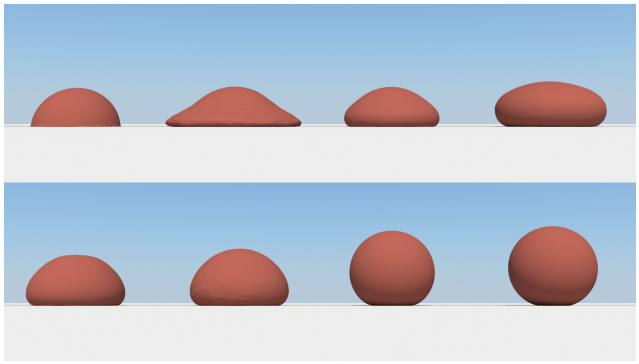

Figure 7: *Varying weight combinations of volume preservation and strain constraints allow the simulation of different types of materials for volumetric objects.*

where ∇_S is the gradient operator defined on the manifold surface S. The choice of M determines all allowed rest configurations $\mathbf{T}\nabla_S\mathbf{g}$. If M is the set of rotation matrices $SO(3)$, we are simply measuring the local deviation from a rigid motion. In this case this energy is identical to the deformation model presented by Chao et al. [2010]. If M is the set of matrices with bounded singular values $\sigma_{\min} < \sigma < \sigma_{\max}$, we can also achieve *isotropic strain limiting* similar to Wang et al. [2010]. This could be extended further to anisotropic material by using reference frames following Hernandez et al. [2013].

Discrete potential. If S is a 2-manifold simplicial complex this energy can be discretized over triangles using a piecewise linear hat basis [Botsch et al. 2010]. The integral is then transformed to a sum of per triangle potentials of the form

$$W(\mathbf{q},\mathbf{T}) = \frac{w}{2}A\|\mathbf{X_f}\mathbf{X_g}^{-1} - \mathbf{T}\|_F^2 + \delta_M(\mathbf{T}),$$

where A is the triangle area, $\mathbf{X_f} = [\mathbf{q}_j - \mathbf{q}_i, \mathbf{q}_k - \mathbf{q}_i] \in \mathbb{R}^{2\times 2}$ contains the triangle edges of the current configuration isometrically embedded in 2D, and similarly $\mathbf{X_g}$ contains the triangle edges of the rest configuration. Note that this discrete potential has the same form as the one in Equation 4 where $\mathbf{A}$ is a function of the rest state edges and the area and $\mathbf{B}$ only depends on the rest state area. Figure 6 shows the strain limiting constraint applied to a curtain example.

Volumes and curves. This potential can be defined in a similar way for volumes: If S is a 3-manifold simplicial complex the energy can be discretized over tetrahedrons replacing the areas of the triangles by the volumes of the tetrahedrons and having 3×3 edge matrices. Note that if we perform a 1D discretization of this energy over a set of edges, we arrive at a model similar to the fast simulation of mass spring models of Liu et al. [2013] where, in addition, the edge potentials are now properly weighted by the edge length.

5.2 Area and Volume Preservation

Area and volume preservation is important for simulating incompressible materials. Using the continuous energy of Equation 7 we can define M as the set of matrices with bounded determinants $\sigma_{\min} < \det(\mathbf{T}) < \sigma_{\max}$, effectively enabling us to control the amount of volume change. If $\sigma_{\min} < 1$ the modeled material allows for compression and similarly if $\sigma_{\max} > 1$ then the material allows for expansion. Figure 7 shows the combination of volume preservation and strain constraints.

5.3 Example-Based

Example-based simulation allows modeling artistic elastic material behavior by supplying few deformation examples that the material should follow [Martin et al. 2011; Koyama et al. 2012; Jones et al. 2013]. We use an energy comparable to Equation 7 defined on 3-manifold surfaces as

$$E(\mathbf{f},\mathbf{R},\mathbf{w}) = \frac{w}{2}\int_S \|\nabla_S\mathbf{f} - \mathbf{R}\nabla_S\mathbf{h}(\mathbf{w})\|_F^2 + \delta_{SO(3)}(\mathbf{R})\,\mathrm{d}V.$$

where $\mathbf{h}(\mathbf{w})$ is a parametrized rest shape defined by the examples. We formulate the rest shape as $\mathbf{h}(\mathbf{w}) = \mathbf{g} + \sum_i w_i(\mathbf{R}_i\mathbf{g}_i - \mathbf{g})$, where the $\mathbf{g}_i$ define the piecewise linear coordinate function of the examples and $\mathbf{R}_i$ is defined point-wise such that it rotates $\mathbf{g}_i$ locally to best align with the undeformed configuration $\mathbf{g}$, similar in spirit to Koyama et al. [2012].

We can discretize this continuous energy using a piecewise linear hat basis leading to a sum of per tetrahedron potentials

$$W(\mathbf{q},\mathbf{R},\mathbf{w}) = \frac{w}{2}V\|\mathbf{X_f}\mathbf{X_g}^{-1} - \mathbf{R}\mathbf{X_h}(\mathbf{w})\mathbf{X_g}^{-1}\|_F^2 + \delta_{SO(3)}(\mathbf{R}),$$

where $\mathbf{X_h}(\mathbf{w}) = \mathbf{X_g} + \sum_i w_i(\mathbf{R}_i\mathbf{X_{g_i}} - \mathbf{X_g})$. Note that the example weights $\mathbf{w}$ can either be defined locally per element or globally, resulting in local or global coupling of the deformation, respectively. An example of three colliding cars using this constraint can be found in Figure 8.

5.4 Bending

Continuous energy. To simulate thin shells and thin plates a bending energy based on dihedral angles across edges is commonly used [Grinspun et al. 2003]. More recently, efficient models for bending of inextensible surfaces relating the Laplace-Beltrami operator to the mean curvature normal have been presented [Bergou et al. 2006; Garg et al. 2007]. These energies build upon an isotropic bending energy defined as the squared difference of mean curvature

$$E(\mathbf{f}) = \frac{w}{2}\int_S (H_\mathbf{f} - H_\mathbf{g})^2\,\mathrm{d}A,$$

where H_f and H_g are the mean curvature functions of the deformed and undeformed surface, respectively. For an isometric deformation (inextensible surface) we can then rewrite this energy using auxiliary rotation matrices as

$$E(\mathbf{f},\mathbf{R}) = \frac{w}{2}\int_S \|\Delta_S\mathbf{f} - \mathbf{R}\Delta_S\mathbf{g}\|_2^2 + \delta_{SO(3)}(\mathbf{R})\,\mathrm{d}A, \qquad (8)$$

Figure 8: *Adding the deformation examples (top) to the simulation using the example-based constraint allows the simulation of complex artistic materials. In this scene, three cars collide and react in a cartoonish manner following the prescribed examples (bottom).*

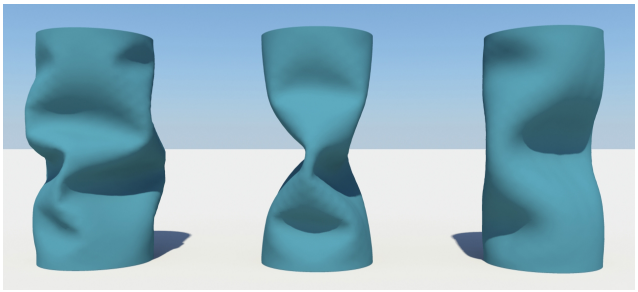

Figure 9: *Simulation of a thin shell cylinder using increasing bending weights from left to right. When the cylinder is compressed, buckling patterns at different frequencies appear.*

where Δ_S is the Laplace-Beltrami operator defined on the manifold surface S. This is due to the fact that the mean curvature vector is equal to the surface's Laplace-Beltrami operator applied to the coordinate function. For an isometric deformation the Laplace-Beltrami operator does not change and therefore can be defined on the undeformed surface. It is interesting to note that this energy is a reformulation of the energy used in [Garg et al. 2007]. Please notice how similar Equation 8 is to Equation 7 replacing the gradient by the Laplace-Beltrami. It could therefore be interesting to apply the strain limiting and example-based concepts to the bending energy.

Discrete potential. If S is a 2-manifold simplicial complex this energy can be discretized using a piecewise linear hat basis leading to per-vertex potentials defined over one-ring neighborhoods. For the ith vertex the potential is defined as

$$W(\mathbf{q},\mathbf{R}) = \frac{w}{2A_i}\| \sum_{j\in\mathcal{N}_i} w_{ij}(\mathbf{e}_\mathbf{f}^{ij} - \mathbf{R}\mathbf{e}_\mathbf{g}^{ij})\|_2^2 + \delta_{SO(3)}(\mathbf{R}),$$

where $\mathcal{N}_i$ is the set of neighborhood indices, A_i is the Voronoi area of the vertex, w_{ij} are the common cotangent weights [Botsch et al. 2010], and $\mathbf{e}_\mathbf{f}^{ij}$ and $\mathbf{e}_\mathbf{g}^{ij}$ are the edges between the vertices i and j for the current configuration and for the rest configuration, respectively. An example of the bending constraint can be found in Figure 9. As can be seen in the appendix this bending constraint allows for a very efficient local solve as it can be implemented just as a simple normalization of the mean curvature vector of the deformed configuration.

6 Discrete Constraints

The constraints derived from continuous energies presented in the previous section allow modeling a large variety of elastic bodies. For practical animation systems additional constraints are equally important and we model them directly as discrete constraints.

Positional constraints. As seen earlier, individual DoFs can be directly constrained by simply choosing $\mathbf{A}_i = \mathbf{B}_i = \mathbf{I}$ in Equation 4. *Dirichlet boundary conditions* can then be realized by defining the constraint set as the desired goal positions, in order to fix objects or create interactive handles.

Collisions. Handling collisions in an implicit manner fits naturally into our general solver and allows respecting the equilibrium of momentum and internal constraints during the collision resolution. When detecting a collision, we dynamically add new unilateral plane constraints. As for positional constraints, we again choose $\mathbf{A}_i = \mathbf{B}_i = \mathbf{I}$ in Equation 4. For a colliding point $\mathbf{q}_c$ we first find the closest surface point $\mathbf{b}$ with normal $\mathbf{n}$, defining a collision plane, such that the constraint set $\mathbf{C}$ is defined by the half space $\mathbf{n}^T(\mathbf{q}-\mathbf{b}) \geq 0$. The projection into this half space in the local step is trivial as it is either a plane projection or the identity map. Note that defining the collision constraint unilaterally allows us to overcome the commonly known sticking problems in implicit collisions handling. Similar to PBD, we handle friction and restitution by changing the velocities of colliding vertices [Müller et al. 2007] when updating velocities.

More constraints. General types of geometric constraints, as for example bending constraints using hinge angles [Bender et al. 2013], can be easily incorporated into our solver. The local solve can be performed in a general manner by minimizing Equation 4 over the auxiliary variables. For many geometric constraints closed-form solutions for this minimization can be found [Bouaziz et al. 2012]. If no closed-form solutions exist, the optimization can be solved using sequential quadratic programming [Nocedal and Wright 2006]. It is interesting to note that for the case of $\mathbf{A} = \mathbf{B} = \mathbf{M}^{\frac{1}{2}}$ one step of sequential quadratic programming is similar to the PBD update [Bender et al. 2013].

7 Results

7.1 Accuracy

In Section 5 we presented a set of new constraints derived from continuous energies. As shown in Figure 5, these new constraints allow our solver to maintain the deformation behavior under different piecewise simplicial approximations of the same underlying surface. This is an important property for computer graphics applications and interactive environments where mesh resolutions can frequently change during development and where geometric levels of detail are widely employed to increase performance. The lack of convergence of PBD approaches makes it difficult for them to handle geometric level of detail properly due to the dependence of the material behavior on the underlying meshing and of the number of iterations [Häggström 2009].

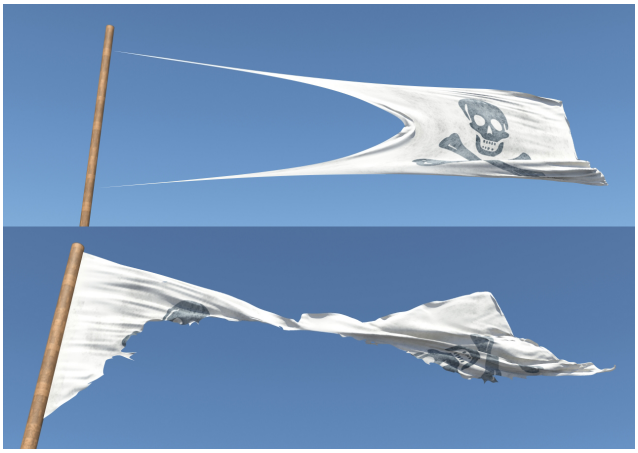

Figure 10: *Even under extreme wind forces our projective implicit solver remains stable. The solver weakly decreases the energy at each iteration making any safeguards unnecessary (top). The pirate flag is torn by the wind in real-time using dynamic updates of the constraints (bottom).*

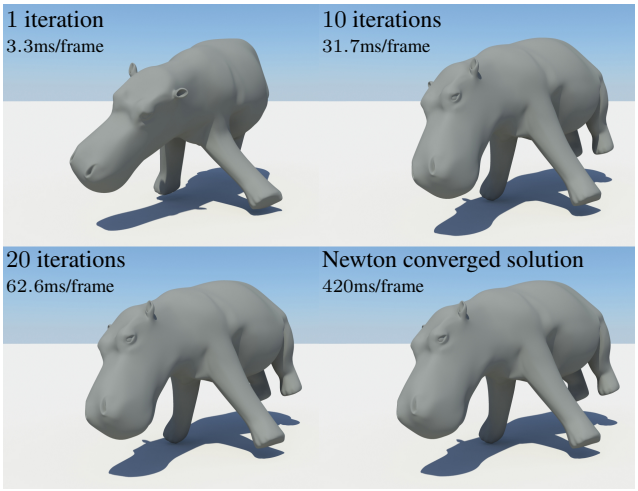

Figure 11: *This volumetric hippopotamus with 7161 DoFs and 8406 strain constraints is simulated with 1, 10, and 20 iterations of our local/global solver. It is interesting to notice that already after 10 iterations our approach looks very similar to the converged solution computed using Newton's method for a fraction of the computational cost.*

7.2 Generality

Our solver does not rely on any particular type of constraint and is able to deal with any variety of geometric constraints within the same setup, making it possible to simulate complex sceneries using a single solver and to also handle object interactions robustly in an implicit manner. In Figure 1 we show such a complex scene with different constraint types, where the objects are also coupled together. For example, the tree and the house are modeled with volumetric strain constraints whereas the washing line, the cloth, the grass and the leaves use edge strain and bending constraints.

Cloths and shells. In Figure 10 we simply use edge strain constraints to model the behavior of a pirate flag. Wind forces are added as a function of wind direction and triangle normal. When the wind forces are too strong, the pirate flag is torn. This is realized by removing edge constraints when the strain exceed a certain threshold. More complex cloths that can undergo small to moderate amounts of stretching can also be modeled using a limit on the triangle strain in combination with bending constraints (see Figure 6). By varying the weights of the strain and bending constraints other types of materials such as thin plates and thin shells can be simulated. In Figure 9 we can see an example of a thin cylinder compressed from the top and showing different buckling patterns due to different ratios of strain and bending constraint stiffnesses.

Solids and exampled-based simulation. We simulate solids by using a combination of strain and volume constraints applied on tetrahedral meshes. As shown in Figure 7 different type of materials can be modeled by varying the weights combining these constraints. Moreover, we can approximate non-linear materials by combining weak strain constraints with stronger strain limiting constraints. Then, the material is soft for small deformations while becoming stiffer when the deformation reaches the strain limit and the second constraint becomes active (see the accompanying video) – a behavior commonly modeled by nonlinear material models. Combining different quadratic potentials has been used earlier for collision handling in [Harmon et al. 2009] but also suits very well our framework to model non-linear material behavior.

Example-based simulation of volumetric meshes is also possible in our formulation. This allows an artistic control over the physical simulation. In Figure 8 three cars deform in a cartoonish manner following the input examples after colliding. Similar to [Martin et al. 2011] the car surface is embedded into a volumetric mesh, which is then deformed using our solver.

7.3 Robustness

One important advantage of our approach is numerical stability. In Figure 10 we show that even under extreme forces our solver stays robust. Similarly, our method remains reliable in situations where the mesh elements degenerate. This can be seen during the simulation of the balls compressed between two planes (see the accompanying video). The only requirement of our approach is that the mesh elements of the input model are well behaved in order to compute the discretization of the gradient and Laplace-Beltrami operators of the original manifold.

7.4 Simplicity

We illustrate the simplicity of our approach by laying out our optimization procedure in Algorithm 1. By removing line 7 and changing $\mathbf{p}_j$ to $\mathbf{q}_{n+1}$ in line 5, we are able to completely recover the structure of the original PBD algorithm [Müller et al. 2007]. Moreover notice that introducing a new constraint only requires the definition of the constraint projection used in the local solve (either exact if known or the general approximate projection scheme given in [Müller et al. 2007] if not) and the definition of a suitable quadratic distance metrics (matrices $\mathbf{A}_i$ and $\mathbf{B}_i$).

7.5 Performance

Implementation. The complete framework presented in this paper is implemented in C++. We use OpenMP to parallelize the local step and we solve the global step in parallel for the x, y and z coordinates by prefactorizing the linear system using sparse Cholesky factorization and performing three times back-substitution in parallel. Dynamic constraints are handled by rank updates and downdates of the linear system. The Eigen library (eigen.tuxfamily.org)

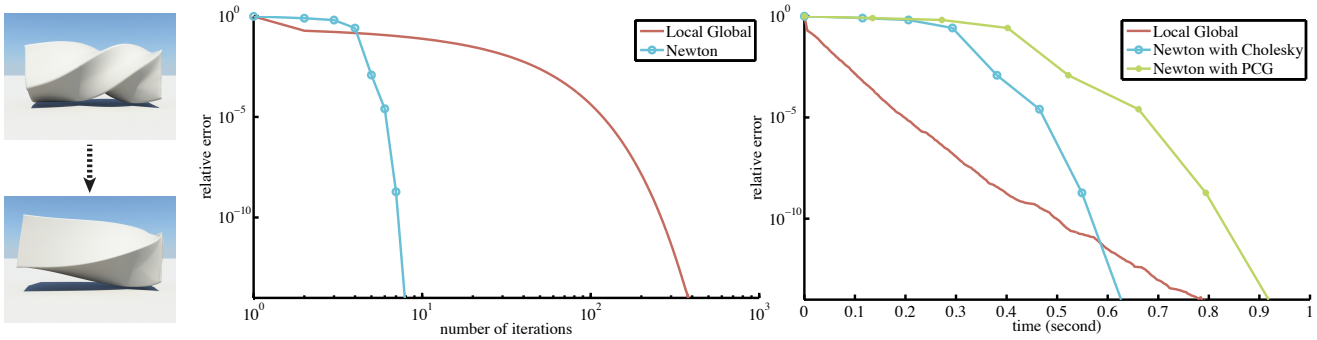

Figure 12: *By comparing the decrease of the relative error with respect to the iteration count, we observe that Newton's method converges faster than our local/global approach. However, this does not reflect the cost of each iteration as for each Newton iteration a changing linear system needs to be solved. Looking at the decrease of the relative error with respect to the computation time, we notice that our local/global approach exhibits a better performance up to a relative error of* 10^{-10} *making our approach particularly attractive for interactive applications. In these curves, the relative error is defined as the normalized error relative to the optimal solution* $(\epsilon(\mathbf{q}_i)-\epsilon(\mathbf{q}^*))/(\epsilon(\mathbf{q}_0)-\epsilon(\mathbf{q}^*))$ *and measured for a twisting bar example (left) with 4290 DoFs and 4099 tetrahedral strain constraints.*

is used for dense and sparse linear algebra. We use either the standard simplicial mass discretization [Hughes 2000] or its lumped version to compute the mass matrix without any noticeable difference.

Comparison with Newton. In Figure 12 we compare the performance of our local/global solver to Newton's method when solving Equation 7 for $M = SO(3)$ similar to [Chao et al. 2010]. As shown in Figure 12 the local/global approach converges slower in number of iterations. This is perfectly logical as Newton's method exhibits quadratic convergence while local/global solvers (block coordinate descent methods) have linear convergence. However, when looking at the convergence in terms of computational time, we notice that our approach is faster than Newton's method for interactive applications. For 1 Newton iteration approximatively 30 local/global iterations can be performed. This is due to the fact that at each Newton iteration the Hessian needs to be recomputed and therefore a new linear system needs to be solved.

Moreover, in Figure 11 and the accompanying video we observe that with approximatively 10 iterations the simulation looks visually similar to the converged one using Newton's method making our scheme a better choice for realtime applications where high accuracy is not the main focus. This type of behavior has already been observed in some of the previous local/global solvers used in geometry processing and simulation [Myles and Zorin 2012; Liu et al. 2013]. Note that implementing Newton's method for the continuous energies presented in Section 5 is nontrivial as one needs to differentiate SVD [McAdams et al. 2011] and new Hessian matrices have to be computed in each time step. Moreover, some safeguards need to be integrated in the optimization as the Hessian matrix may become indefinite and a line search procedure is also needed to avoid overshooting.

Comparison with Position Based Dynamics. We also compared our approach to PBD using edge strain constraints. As explained in Section 4, PBD does not include the momentum constraints making the material stiffness dependent of the number of iterations. This can be seen in the accompanying video and in Figure 4, where for different number of iterations the stiffness of the material simulated by PBD drastically changes. This is not the case in our approach where the material stiffness is much less dependent of the number of iterations (and the underlying mesh structure).

Timing. For simulation of medium sized models ($<$ 30K constraints and $<$ 30K DoFs), 5-10 iterations are usually sufficient. At 1-6ms per iteration, this enables realtime simulation on a MacBook Pro 2.7 GHz Intel Quad-core i7 with 16GB of memory. Statistics on timings and meshes can be found in the accompanying video. Moreover, the accompanying application demonstrates the performance on three different examples.

8 Limitations and Future Work

While our implicit Euler solver is efficient and robust, it exhibits implicit damping. In the near future we plan to extend our approach to symplectic integrators [Kharevych et al. 2006] which conserve the system's energy. Damping can also be observed when the optimization is terminated early. This is due to the fact that external forces may not be able to propagate fully through the mesh if the optimization is not run for enough iterations. This effect is accentuated in large meshes as more iterations are needed until convergence. As a future work, we would like to improve the speed of our solver by implementing a GPU version of our code and focus on topological changes (cutting, fracturing) that result in dynamically changing constraints. While the local steps remain simple to solve on the GPU, the global system is changing, making it even more involved to solve efficiently. This problem becomes even more accentuated if we want to extend our approach to fluid simulation similar to [Macklin and Müller 2013] where neighborhood relations always change.

Treating all constraints in a soft manner allows us to handle them efficiently in an unified way. However, in certain situations, being able to enforce hard constraints, such as for collision handling or boundary conditions, would be advantageous but also would introduce more complexity into our solver. That is, we are trading hard constraint handling versus simplicity and efficiency.

Another interesting area of further research is on enlarging our set of constraints. One direction we want to explore is modeling more complex deformation behaviors such as in anisotropic and nonlinear materials. Furthermore, it would also be attractive to integrate rigid bodies into the same simulation framework.

9 Conclusion

We introduce a new implicit constraint-based solver for real-time simulation. Our approach is based on an abstract, constraint-based description of the physical system making our method *general* in its use to simulate a large variety of different geometries and materials. To solve the constraint problem, we apply a local/global solver that is guaranteed to weakly decrease the energy making any safeguards unnecessary and giving us *robustness*. Our *simple* constraint-based formulation only requires the definition of a projection operator for a given constraints (local solve), making it very easy to implement and to introduce new models into the solver. Furthermore, the global solve only requires solving a linear system, where the system matrix is constant if the number of constraints is kept fixed, leading to *efficient* computation. Due to the independence of the local solves, the approach is also very well suited for parallelism, further boosting performance. We derive a broad set of constraints directly from continuous energies using proper discretization that make the solver robust to non-uniform meshing with different resolutions. With these qualities in mind we believe that our approach strikes the right balance between the simplicity, generality, robustness and performance of position-based simulations with the rigor and accuracy of continuum mechanics. We believe this makes our method suitable for many applications in both realtime and offline simulation in computer graphics.

References

BARAFF, D., AND WITKIN, A. 1998. Large steps in cloth simulation. In *Proc. of ACM SIGGRAPH*, 43–54.

BENDER, J., MÜLLER, M., OTADUY, M. A., AND TESCHNER, M. 2013. Position-based methods for the simulation of solid objects in computer graphics. In *EG State of the Art Reports*.

BERGOU, M., WARDETZKY, M., HARMON, D., ZORIN, D., AND GRINSPUN, E. 2006. A quadratic bending model for inextensible surfaces. In *Proc. EG Symp. Geometry Processing*, 227–230.

BOTSCH, M., KOBBELT, L., PAULY, M., ALLIEZ, P., AND LEVY, B. 2010. *Polygon Mesh Processing*. AK Peters.

BOUAZIZ, S., DEUSS, M., SCHWARTZBURG, Y., WEISE, T., AND PAULY, M. 2012. Shape-up: Shaping discrete geometry with projections. In *Comput. Graph. Forum*, vol. 31, 1657–1667.

BOYD, S. P., AND VANDENBERGHE, L. 2004. *Convex optimization*. Cambridge university press.

BRIDSON, R., MARINO, S., AND FEDKIW, R. 2003. Simulation of clothing with folds and wrinkles. In *Proc. EG Symp. Computer Animation*, 28–36.

CHAO, I., PINKALL, U., SANAN, P., AND SCHRÖDER, P. 2010. A simple geometric model for elastic deformations. *ACM Trans. Graph. 29*, 4, 38.

DESBRUN, M., SCHRÖDER, P., AND BARR, A. 1999. Interactive animation of structured deformable objects. In *Graphics Interface*, vol. 99, 10.

GARG, A., GRINSPUN, E., WARDETZKY, M., AND ZORIN, D. 2007. Cubic shells. In *Proc. EG Symp. Computer Animation*, SCA '07, 91–98.

GOLDENTHAL, R., HARMON, D., FATTAL, R., BERCOVIER, M., AND GRINSPUN, E. 2007. Efficient simulation of inextensible cloth. *ACM Trans. Graph. 26*, 3, 49.

GRINSPUN, E., HIRANI, A. N., DESBRUN, M., AND SCHRÖDER, P. 2003. Discrete shells. In *Proc. EG Symp. Computer Animation*, 62–67.

HÄGGSTRÖM, O. 2009. *Interactive Real Time Cloth Simulation with Adaptive Level of Detail*. Master's thesis.

HAHN, F., MARTIN, S., THOMASZEWSKI, B., SUMNER, R., COROS, S., AND GROSS, M. 2012. Rig-space physics. *ACM Trans. Graph. 31*, 4, 72.

HARMON, D., VOUGA, E., SMITH, B., TAMSTORF, R., AND GRINSPUN, E. 2009. Asynchronous contact mechanics. In *ACM Trans. Graph.*, vol. 28, 87.

HECHT, F., LEE, Y. J., SHEWCHUK, J. R., AND O'BRIEN, J. F. 2012. Updated sparse cholesky factors for corotational elastodynamics. *ACM Trans. Graph. 31*, 5, 123.

HERNANDEZ, F., CIRIO, G., PEREZ, A. G., AND OTADUY, M. A. 2013. Anisotropic strain limiting. In *Proc. of Congreso Español de Informática Gráfica*.

HUGHES, T. J. R. 2000. *The Finite Element Method. Linear Static and Dynamic Finite Element Analysis*. Dover Publications.

IRVING, G., TERAN, J., AND FEDKIW, R. 2004. Invertible finite elements for robust simulation of large deformation. In *Proc. EG Symp. Computer Animation*, 131–140.

JONES, B., POPOVIC, J., MCCANN, J., LI, W., AND BARGTEIL, A. 2013. Dynamic sprites. In *Proceedings of the Motion on Games*, 17–24.

KHAREVYCH, L., YANG, W., TONG, Y., KANSO, E., MARSDEN, J. E., SCHRÖDER, P., AND DESBRUN, M. 2006. Geometric, variational integrators for computer animation. In *Proc. EG Symp. Computer Animation*, 43–51.

KOYAMA, Y., TAKAYAMA, K., UMETANI, N., AND IGARASHI, T. 2012. Real-time example-based elastic deformation. In *Proc. EG Symp. Computer Animation*, 19–24.

LIU, T., BARGTEIL, A. W., O'BRIEN, J. F., AND KAVAN, L. 2013. Fast simulation of mass-spring systems. *ACM Trans. Graph. 32*, 6, 214.

MACKLIN, M., AND MÜLLER, M. 2013. Position based fluids. *ACM Trans. Graph. 32*, 4, 104:1–104:12.

MARTIN, S., THOMASZEWSKI, B., GRINSPUN, E., AND GROSS, M. 2011. Example-based elastic materials. In *ACM Trans. Graph.*, vol. 30, 72.

MCADAMS, A., ZHU, Y., SELLE, A., EMPEY, M., TAMSTORF, R., TERAN, J., AND SIFAKIS, E. 2011. Efficient elasticity for character skinning with contact and collisions. *ACM Trans. Graph. 30*, 4, 37:1–37:12.

MÜLLER, M., HEIDELBERGER, B., TESCHNER, M., AND GROSS, M. 2005. Meshless deformations based on shape matching. In *ACM Trans. Graph.*, vol. 24, 471–478.

MÜLLER, M., HEIDELBERGER, B., HENNIX, M., AND RATCLIFF, J. 2007. Position based dynamics. *J. Vis. Comun. Image Represent. 18*, 2, 109–118.

MYLES, A., AND ZORIN, D. 2012. Global parametrization by incremental flattening. *ACM Trans. Graph. 31*, 4, 109:1–109:11.

NARAIN, R., SAMII, A., AND O'BRIEN, J. F. 2012. Adaptive anisotropic remeshing for cloth simulation. *ACM Trans. Graph. 31*, 6, 152.

NOCEDAL, J., AND WRIGHT, S. J. 2006. *Numerical optimization*. Springer Verlag.

PRESS, W. H., TEUKOLSKY, S. A., VETTERLING, W. T., AND FLANNERY, B. P. 2007. *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press.

PROVOT, X. 1995. Deformation constraints in a mass-spring model to describe rigid cloth behavior. In *Graphics Interface 1995*, 147–154.

RIVERS, A., AND JAMES, D. 2007. FastLSM: fast lattice shape matching for robust real-time deformation. *ACM Trans. Graph. 26*, 3.

SIFAKIS, E., AND BARBIC, J. 2012. Fem simulation of 3d deformable solids: A practitioner's guide to theory, discretization and model reduction. In *ACM SIGGRAPH 2012 Courses*, SIGGRAPH '12, 20:1–20:50.

SORKINE, O., AND ALEXA, M. 2007. As-rigid-as-possible surface modeling. In *Proc. EG Symp. Geometry Processing*, 109–116.

STAM, J. 2009. Nucleus: towards a unified dynamics solver for computer graphics. In *IEEE Int. Conf. on CAD and Comput. Graph.*, 1–11.

STERN, A., AND DESBRUN, M. 2006. Discrete geometric mechanics for variational time integrators. In *ACM SIGGRAPH Courses*, 75–80.

STERN, A., AND GRINSPUN, E. 2009. Implicit-explicit variational integration of highly oscillatory problems. *Multiscale Modeling & Simulation 7*, 4, 1779–1794.

SU, J., SHETH, R., AND FEDKIW, R. 2013. Energy conservation for the simulation of deformable bodies. *IEEE Trans. Vis. Comput. Graph. 19*, 2, 189–200.

TERZOPOULOS, D., PLATT, J., BARR, A., AND FLEISCHER, K. 1987. Elastically deformable models. In *Computer Graphics (Proceedings of SIGGRAPH)*, vol. 21, 205–214.

THOMASZEWSKI, B., PABST, S., AND STRASSER, W. 2009. Continuum-based strain limiting. In *Comput. Graph. Forum*, vol. 28, 569–576.

WANG, H., O'BRIEN, J., AND RAMAMOORTHI, R. 2010. Multi-resolution isotropic strain limiting. *ACM Trans. Graph. 29*, 6, 156.

A Local Solves

Strain. When minimizing over $\mathbf{T}$ while keeping $\mathbf{q}$ fixed in the local step

$$\arg\min_{\mathbf{T}} \|\mathbf{X_f}\mathbf{X_g}^{-1} - \mathbf{T}\|_F^2 + \delta_M(\mathbf{T}),$$

the optimization can be reformulated as

$$\arg\min_{\mathbf{\Sigma}^*} \|\mathbf{\Sigma} - \mathbf{\Sigma}^*\|_F^2 \quad \text{s.t.} \quad \sigma_{min} \leq \mathbf{\Sigma}^*_{ii} \leq \sigma_{max},$$

where $\mathbf{X_f}\mathbf{X_g}^{-1} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ and $\mathbf{T} = \mathbf{U}\mathbf{\Sigma}^*\mathbf{V}^T$. The optimal solution can be computed as $\mathbf{\Sigma}^*$ being the singular values $\mathbf{\Sigma}$ clamped between σ_{min} and σ_{max}. For tethraherons, if $\det(\mathbf{X}_f\mathbf{X}_g^{-1}) < 0$, the last singular value is negated to avoid reflections.

Area and Volume. Similar to the strain constraint the local minimization of the volume constraint can be reformulated as

$$\arg\min_{\mathbf{\Sigma}^*} \|\mathbf{\Sigma} - \mathbf{\Sigma}^*\|_F^2 \quad \text{s.t.} \quad \sigma_{min} \leq \prod_i \mathbf{\Sigma}^*_{ii} \leq \sigma_{max}.$$

This problem can be further transformed in

$$\arg\min_{\mathbf{D}} \|\mathbf{D}\|_2^2 \quad \text{s.t.} \quad \prod_i (\mathbf{\Sigma}_{ii} + \mathbf{D}_i) = \sigma,$$

with $\mathbf{\Sigma}^*_{ii} = \mathbf{\Sigma}_{ii} + \mathbf{D}_i$ and where $\sigma = \sigma_{min}$ when $\prod_i \mathbf{\Sigma}^*_{ii} < \sigma_{min}$ and $\sigma = \sigma_{max}$ when $\prod_i \mathbf{\Sigma}^*_{ii} > \sigma_{max}$. This constrained minimization can be solved by iteratively solving a quadratic programming problem by linearizing the constraint leading to a simple update rule

$$\mathbf{D}^{k+1} = \frac{\nabla\mathbf{C}(\mathbf{D}^k)^T\mathbf{D}^k - \mathbf{C}(\mathbf{D}^k)}{\|\nabla\mathbf{C}(\mathbf{D}^k)\|_2^2}\nabla\mathbf{C}(\mathbf{D}^k),$$

where $\mathbf{C}(\mathbf{D}) = \prod_i(\mathbf{\Sigma}_{ii} + \mathbf{D}_i) - \sigma$.

Example-Based. We solve the optimization

$$\arg\min_{\mathbf{R},\mathbf{w}} \|\mathbf{X_f}\mathbf{X_g}^{-1} - \mathbf{R}\mathbf{X_h}(\mathbf{w})\mathbf{X_g}^{-1}\|_F^2 + \delta_{SO(3)}(\mathbf{R}),$$

using a local/global approach by minimizing over $\mathbf{R}$ and $\mathbf{w}$ iteratively. The minimization over $\mathbf{R}$ is solved using SVD following [Sorkine and Alexa 2007] and solving over $\mathbf{w}$ corresponds to solve a simple linear system.

Bending. The local solve of the bending constraint can be formulated as

$$\arg\min_{\mathbf{R}} \|\mathbf{v}_f - \mathbf{R}\mathbf{v}_g\|_2^2 + \delta_{SO(3)}(\mathbf{R}),$$

where $\mathbf{v}_f = \sum_{j\in\mathcal{N}_i} w_{ij}\mathbf{e_f}^{ij}$ and $\mathbf{v}_g = \sum_{j\in\mathcal{N}_i} w_{ij}\mathbf{e_g}^{ij}$. This corresponds in finding a rotation $\mathbf{R}$ such that the rotated vector $\mathbf{v}_g$ matches best the vector $\mathbf{v}_f$. While $\mathbf{R}$ could be found using SVD [Sorkine and Alexa 2007] this problem has an easier closed form solution where $\mathbf{R}\mathbf{v}_g$ can be replaced by $\frac{\mathbf{v}_f\|\mathbf{v}_g\|_2}{\|\mathbf{v}_f\|_2}$.