Projective Dynamics: Fusing Constraint Projections for Fast Simulation

Sofien Bouaziz∗
EPFL

Sebastian Martin†
VM Research

Tiantian Liu‡
University of Pennsylvania

Ladislav Kavan§
University of Pennsylvania

Mark Pauly¶
EPFL

Abstract

We present a new method for implicit time integration of physical systems. Our approach builds a bridge between nodal 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

Physics-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 nonlinear 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 physics-based simulation. Robustness refers to the capability to adequately handle difficult configurations, including 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 including 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 individual 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 insights on how PBD relates to traditional approaches based on finite element methods and Newtonian mechanics.

## 2 Related Work

Since the pioneering work of Terzopoulos 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 sympletic integrators [Hairer et al. 2002; Kharcevych et al. 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 criterion 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) as demonstrated by [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], which is limited to mass-spring systems and assumes only linear springs (Hooke’s law), we show how to generalize this concept employing projection onto constraint sets to simulate general nodal 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, in contrast to our method, 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 it is directly included in the implicit solver.
3 Continuum Mechanics View

In this section we introduce the special structure of our potentials that 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 \( \mathbf{f}_{\text{int}}(\mathbf{q}) = -\sum_i \nabla W_i(\mathbf{q}) \), where \( W_i(\mathbf{q}) \) is a scalar potential energy function. Implicit Euler time integration results in the following update rule:

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

where \( \mathbf{M} \) is the mass-matrix and \( h \) represents the simulation step size. Note that \( \mathbf{f}_{\text{int}} \) and \( \mathbf{M} \) are held constant for any given time step. Using these equations we can derive

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

This system can be converted to an optimization problem

\[
\min_{\mathbf{q}_{n+1}} \frac{1}{2h^2} ||\mathbf{M}^{1/2}(\mathbf{q}_{n+1} - \mathbf{s}_n)||_F^2 + \sum_i W_i(q_{n+1}),
\]

where \( \mathbf{s}_n = \mathbf{q}_n + h\mathbf{v}_n + h^2\mathbf{M}^{-1}\mathbf{f}_{\text{ext}} \) and ||.||_F denotes the Frobenius norm. Intuitively, this minimization problem describes the compromise between the momentum potential

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

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 Noether’s theorem, linear and angular momenta are always conserved when the elastic potential is rigid motion invariant.

The minimization of Equation 4 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 4 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 \( \mathbf{E}(\mathbf{q}) = 0 \) 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

\[
W(\mathbf{q}, \mathbf{p}) = d(\mathbf{q}, \mathbf{p}) + \delta\mathbf{E}(\mathbf{p}),
\]

Here, \( \delta\mathbf{E}(\mathbf{p}) \) is an indicator function that evaluates to zero if \( \mathbf{E}(\mathbf{p}) = 0 \) 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 6 over \( \mathbf{p} \) corresponds to a projection of \( \mathbf{q} \) onto the constraint manifold, as illustrated in Figure 2. An elastic potential analogous to \( \Psi(\mathbf{E}(\mathbf{q})) \) can therefore be defined as \( W(\mathbf{q}) = \min_{\mathbf{p}} W(\mathbf{q}, \mathbf{p}) \).

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:

\[
W(\mathbf{q}, \mathbf{p}) = \frac{\kappa}{2} ||\mathbf{A}\mathbf{q} - \mathbf{B}\mathbf{p}||^2_F + \delta\mathbf{C}(\mathbf{p}),
\]

where \( \mathbf{A} \) and \( \mathbf{B} \) are constant matrices and \( \kappa \) is a nonnegative weight. 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 \( \mathbf{C}(\mathbf{q}) = 0 \) for the set of desired configurations [Baraff and Witkin 1998], 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 7, we can reformulate the implicit integration defined in Equation 4 as the minimization
which allows massive parallelization of the local step. We will use this fact to parallelize the computation in each iteration after the projection variables have been computed.

Algorithm 1: Projective Implicit Euler Solver

1. \( s_n = q_n + h \nu_n + h^2 M^{-1} f_{ext} \)
2. \( q_{n+1} = s_n \)
3. loop solver iteration times
   4. forall the constraints \( i \) do
      5. \( p_i = \text{ProjectOnConstraintSet}(C_i, q_{n+1}) \)
   6. end
   7. \( q_{n+1} = \text{SolveLinearSystem}(s_n, p_1, p_2, p_3, \ldots) \)
8. end
9. \( \nu_{n+1} = (q_{n+1} - q_n)/h \)

The system matrix is constant as long as the constraints are not violated. Since each constraint has its own set of auxiliary variables \( p_i \), the minimization can be performed independently for each constraint as

\[
\frac{1}{2h^2} \| M_i^T (q - s_n) \|_F^2 + \sum_i \frac{w_i}{2} \| A_i S_i q - B_i p_i \|_F^2 + \delta_{C_i}(p_i),
\]

(8)

over \( q \) and the auxiliary variables \( p_i \), where \( S_i \) is a constant selection matrix that selects the vertices involved in the \( i \)th constraint. We minimize Equation 8 using a local/global alternating minimization technique.

Local solve. First, we minimize Equation 8 over the auxiliary variables keeping the positions fixed. Since each constraint has its own set of auxiliary variables \( p_i \), the minimization can be performed independently for each constraint as

\[
\min_{p_i} \frac{w_i}{2} \| A_i S_i q - B_i p_i \|_F^2 + \delta_{C_i}(p_i),
\]

(9)

which allows massive parallelization of the local step. We will discuss specific constraint types in Section 5.

Global solve. Second, we minimize Equation 8 over the positions, keeping the auxiliary variables fixed. Since Equation 8 is quadratic in the unknowns \( q \), we can minimize it with a single linear solve. Requiring that the gradient vanishes at the critical point leads to the linear system

\[
\begin{pmatrix}
M_i + \sum_i w_i S_i^T A_i A_i S_i
\end{pmatrix} q = M_i s_n + \sum_i w_i S_i^T A_i B_i p_i.
\]

(10)

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, even for non-convex sets. 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 \( s_n \). 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 \( A \) and \( B \). If we choose \( A_i = B_i = I \), Equation 7 measures the squared Euclidean distance from \( S_i q \) to its closest point on the constraint set. With diagonal matrices, 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 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 \( A_i = B_i \) 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 \( A_i \) and \( B_i \) 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 \( A_i \) and \( B_i \) matrices can be derived from continuum formulations in order to avoid these shortcomings.

4 Position Based Dynamics View

While [Liu et al. 2013] hint at the similarity between general variational implicit Euler and PBD, in this section, we derive the exact relationship not just between implicit Euler and PBD, but also between the local/global formulation and PBD, for general constraints. 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.

4.1 Gauss-Seidel Solver

A classical PBD solver [Müller et al. 2007] performs three steps. In the first step, the positions are initialized by an explicit Euler step, ignoring internal forces. In the second step, the positions are updated by projecting the current configuration consecutively on each constraint set respecting the mass weighting. In the last step, the velocities are updated as \( \nu_{n+1} = (q_{n+1} - q_n)/h \).

We can show that the constraint resolution strategy of PBD actually
implements a Gauss-Seidel type minimization on the energy
\[ \frac{1}{2} \sum_i \| M_i^2 \nabla C(q_i) \|^2_F + \lambda \| M_i^2 \nabla C(q_i) \|^2_F, \]  
(11)

using a lumped mass matrix \( M_i \) only involving the constraint’s points. A Gauss-Seidel approach minimizes this energy by optimizing each summand sequentially, i.e., minimizing potentials of the form \( \frac{1}{2} \| M_i^2 \nabla C(q_i + \Delta q_i) \|^2_F + \delta C_i(q_i + \Delta q_i) \) where we introduce corrections \( \Delta q_i = q_i - q_i \) to simplify the derivation. Using Lagrange multipliers for the linearized constraint \( C(q_i) + \text{tr}(\nabla C(q_i)^T \Delta q_i) = 0 \), we can define the Lagrangian
\[ \frac{1}{2} \| M_i^2 \Delta q_i \|^2_F + \lambda \left( C(q_i) + \text{tr}(\nabla C(q_i)^T \Delta q_i) \right), \]  
(12)

Using the critical point condition w.r.t. \( \Delta q_i \), we find the optimal direction \( \Delta q_i = -\lambda M_i^{-\frac{1}{2}} \nabla C(q_i) \). The Lagrange multiplier \( \lambda \) can be found by requiring that the linearized constraint vanishes in this direction, i.e., \( C(q_i) - \lambda \| M_i^{-\frac{1}{2}} \nabla C(q_i) \|_F^2 = 0 \), leading to the final update
\[ \Delta q_i = -M_i^{-\frac{1}{2}} \nabla C(q_i) \frac{C(q_i)}{\| M_i^{-\frac{1}{2}} \nabla C(q_i) \|_F^2}, \]  
(13)

corresponding exactly to the mass-weighted update rule of PBD [Bender et al. 2013].

**Discussion.** 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 constraints 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 5. 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 11, we can solve these issues in a straightforward manner by performing two steps. First, 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. Second, we introduce the momentum constraint into the optimization to take into account the inertia of each point. 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 5
\[ \frac{1}{2\mu^2} \| M_i^2 (q - S_i) \|^2_F + \sum_i \frac{u_i}{2} \| M_i^2 (S_i q_i - p_i) \|^2_F + \delta C_i(p_i). \]  
(14)

The Jacobi solver then becomes a two-step optimization: In the local step, the current solution \( q_i \) is first projected onto the constraints independently by solving Equation 11 for all \( p_i \). Then, a consensus can be reached between the different solutions by solving the global step over \( q_i \).

**Connection to Projective Implicit Euler.** At this stage, we can see how close this Jacobi solver is to our projective implicit solver procedure presented in the last section – we recover this solver by choosing \( A_i = B_i = M_i^2 \). 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 7 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

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 its triangle edges of the current configuration isometrically. The integral is then transformed to a sum 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.

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

$$ W(q, T) = \frac{w}{2} A \| X_T X_q^{-1} \| F^2 + \delta_M(T), $$

where $A$ is the triangle area, $X_T = \left[ \begin{array}{c} q_1 \\ q_2 \\ q_3 \end{array} \right] \in \mathbb{R}^{2 \times 3}$ contains the triangle edges of the current configuration isometrically embedded in 2D, and similarly $X_q$ contains the triangle edges of the rest configuration. Note that this discrete potential has the same form as the one in Equation 7 where $A$ is a function of the rest state edges and the area and $B$ 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 $3 \times 3$ 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 15 we can define $M$ as the set of volumes with bounded determinants $\sigma_{\text{min}} < \sigma < \sigma_{\text{max}}$, effectively enabling us to control the amount of volume change. If $\sigma_{\text{max}} < 1$ the modeled material allows for compression and similarly if $\sigma_{\text{max}} > 1$ 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 a 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 15 defined on 3-manifold surfaces as

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

where $h(w)$ is a parametrized rest shape defined by the examples. We formulate the rest shape as $h(w) = g + \sum_i w_i (R_i g_i - g)$, where the $g_i$ define the piecewise linear coordinate functions of the examples and $R_i$ are precomputed rotation matrices defined point-wise such that it rotates $g_i$ locally to best align with the undeformed configuration $g$, similar in spirit to Koyama et al. [2012]. We can discretize this continuous energy using a piecewise linear
where \( H \) (inextensible surface) we can then rewrite the energy using auxiliary rotation matrices as

\[
E(f, R) = \frac{w}{2} \int_S \| \Delta S f - RX_h(w)X_g^{-1} \|^2 + \delta_{SO(3)}(R) \, dA, \tag{20}
\]

where \( \Delta_S \) is the Laplace-Beltrami operator defined on the manifold surface \( S \). This is because 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. Please notice how similar Equation 20 is to Equation 15 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 as well.

**Continuous energy.** Thin shells and thin plates are commonly simulated using bending energy based on dihedral angles across edges [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. 2010]. 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.

**5.4 Bending**

**Discrete potential.** If \( S \) is a 2-manifold simplicial complex Equation 20 can be discretized using a piecewise linear hat basis leading to per vertex potentials of the form

\[
W(q, R) = \frac{w}{2} A \| X_r e - RX_g e \|^2 + \delta_{SO(3)}(R), \tag{21}
\]

where \( A \) is the Voronoi area of the vertex, and \( X_r \) and \( X_g \) contain the one-ring edges of the vertex for the current configuration and for the rest configuration, respectively. The vector \( e \) stores the common cotangent weights divided by the Voronoi area [Botsch et al. 2010]. An example of the bending constraint can be found in Figure 9. 

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

**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. We model these directly as discrete constraints.

**Positional constraints.** As seen earlier, individual DoFs can be directly constrained by simply choosing \( A_i = B_i = I \) in Equation 7. *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 \( A_i = B_i = I \) in Equation 7. For a colliding point \( q_i \), we first find the closest surface point \( b \) with normal \( n \), defining a collision plane, such that the constraint set \( C \) is defined by the half space \( n^T(q - b) \geq 0 \). 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 when updating velocities. A simple damping model can also be implemented by filtering velocities [Müller et al. 2007].

**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 7 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 (SQP) [Nocedal and Wright 2006]. As shown in Section 4.1, for the case of $\mathbf{A} = \mathbf{B} = \mathbf{M}^2$ one step of SQP is similar to the PBD update [Bender et al. 2013].

7 Results

7.1 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 7 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 exceeds 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 example-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. While we cannot model arbitrary non-linear materials, we are able to approximate some non-linear behaviour 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.

7.2 Robustness and Simplicity

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.

We illustrate the simplicity of our approach by laying out our optimization procedure in Algorithm 1. By removing line 7 and changing $p_i$ to $q_{n+1}$ 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 suitable quadratic distance metrics (matrices $A_i$ and $B_i$).
7.3 Accuracy and Performance

Comparison with Newton. In Figure 12 we compare the performance of our local/global solver to Newton’s method when solving the discretization of Equation 15 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 approximately 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 approximately 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.

Meshing independence. 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 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 [Haggström 2009].

8 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 prefactoring 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) 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.

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 multiple examples.

9 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 provide better energy behavior. 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.

We are trading hard constraints for simplicity and efficiency. Treating all constraints in a soft manner allows us to handle them in a unified and effective manner. However, in certain situations, being able to enforce hard constraints, such as for collision handling or boundary conditions, would be advantageous. Hard constraints can still be approximated by increasing the weight of the constraints. However, this can degrade the conditioning of the linear system and can result in locking artifacts.

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

10 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 real-time and offline simulation in computer graphics.

Acknowledgments. We thank James O’Brien, Adam Bargteil, Basil Fierz and Bernhard Thomaszewski for the insightful discussions and the reviewers for their valuable comments. We are grateful to luismigabril for providing the cartoon house model and to Daniel Grauer for modeling the teaser scene. We also thank Yuliy Schwartzburg for his narration of the accompanying video. This research is supported by the Swiss National Science Foundation grant 20PA21L129607, by the European Research Council under the European Unions Seventh Framework Programme (FP/2007-2013)/ERC Grant Agreement n. 257453: COSYM and by the NSF Career Award IIS-1350330.

References


A Local Solves

Strain. When minimizing over $T$ while keeping $q$ fixed in the local step

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

the optimization can be reformulated as

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

where $X_T X_g^{-1} = U \Sigma V^T$ and $T = U \Sigma^* V^T$. The optimal solution can be computed as $\Sigma^*$ being the singular values $\Sigma$ clamped between $\sigma_{\min}$ and $\sigma_{\max}$. For tetrahedrons, if $\det(X_T X_g^{-1}) < 0$, 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

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

This problem can be further transformed in

$$\min_{D} \|D\|_2^2 \text{ s.t. } \prod_i (\Sigma_{ii} + D_i) = \sigma,$$

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

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

where $C(D) = \prod_i (\Sigma_{ii} + D_i) - \sigma$.

Example-Based. We solve the optimization

$$\min_{R, w} \|X_T X_g^{-1} - RX_h(w)X_e^{-1}\|_F^2 + \delta_{SO(3)}(R),$$

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

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

$$\min_{R} \|v_f - R v_g\|_2^2 + \delta_{SO(3)}(R),$$

where $v_f = X_T e$ and $v_g = X_e e$. This corresponds in finding a rotation $R$ such that the rotated vector $v_g$ matches best the vector $v_f$. While $R$ could be found using SVD [Sorkine and Alexa 2007] this problem has an easier closed form solution where $R v_g$ can be replaced by $v_f |v_g|_2 / \|v_f\|_2$. 


This problem can be further transformed in

$$\min_{D} \|D\|_2^2 \text{ s.t. } \prod_i (\Sigma_{ii} + D_i) = \sigma,$$

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

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

where $C(D) = \prod_i (\Sigma_{ii} + D_i) - \sigma$.