Efficient Simulation of Inextensible Cloth

Rony Goldenthal[1,2] David Harmon[1] Raanan Fattal[3] Michel Bercovier[2] Eitan Grinspun[1]

[1]Columbia University [2]The Hebrew University of Jerusalem [3]University of California, Berkeley

Abstract

Many textiles do not noticeably stretch under their own weight. Unfortunately, for better performance many cloth solvers disregard this fact. We propose a method to obtain very low strain along the warp and weft direction using Constrained Lagrangian Mechanics and a novel fast projection method. The resulting algorithm acts as a velocity filter that easily integrates into existing simulation code.

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

Keywords: Physically-based Modeling, Cloth simulation, Constrained Lagrangian Mechanics, Constraints, Stretching, Inextensibility, Isometry

1 Introduction

Our eyes are very sensitive to the behavior of fabrics, to the extent that we can identify the kind of fabric simply from its shape and motion [Griffiths and Kulke 2002]. One important fact is that most fabrics do not stretch under their own weight. Unfortunately, for many popular cloth solvers, a reduction of permissible stretching is synonymous with degradation in performance: for tractable simulation times one may settle for an unrealistic 10% or more strain (compare 1% and 10%, Figure 1). Our work alleviates this problem by introducing a numerical solver that excels at timestepping quasi-inextensible surfaces (stretching below 1%).

The solver builds on a framework of Constrained Lagrangian Mechanics (CLM) [Marsden 1999]. Warp and weft, the perpendicular sets of strands that make up a textile, are prohibited from stretching by enforcing constraint equations, not by integrating spring forces. We present numerical evidence supporting the observation that a constraint-based method is inherently well-suited to operate in the quasi-inextensible regime. In contrast, for this regime spring-based methods are known to experience a range of difficulties, leading to the adoption of various strain limiting [Provot 1995] and strain rate limiting algorithms.

We are motivated by the work of Bridson *et al.* [2002], who viewed strain limiting as one of multiple *velocity filtering* passes (another being collision handling). The velocity filter paradigm enables the design of modular systems with mix-and-match flexibility.

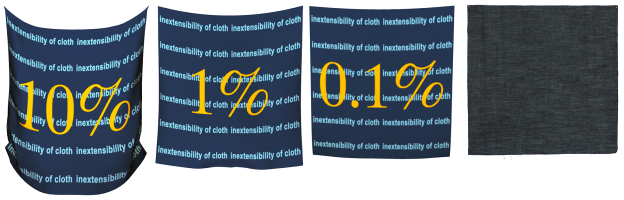

Figure 1: Importance of capturing inextensibility. *For efficiency, many simulation methods allow* 10% *or more strain, whereas many fabrics do not visibly stretch. A* $1m^2$ *patch, pinned at two corners* $1m$ *apart, is allowed to relax under gravity. We compare (*left to right*) three simulations of progressively smaller permissible strain with an actual denim patch.*

Contributions We propose a novel CLM formulation that is implicit on the constraint gradient (§4.1). We prove that the implicit method's nonlinear equations correspond to a minimization problem (§4.2): this result motivates a *fast projection method* for enforcing inextensibility (§4.3). We describe an implementation of fast projection as a simple and efficient velocity filter, as part of a framework that decouples timestepping, inextensibility, and collision passes (§4.4). Consequently, the fast projection method easily incorporates with a code's existing bending, damping, and collision models, to yield accelerated performance (§5).

Before discussing these contributions, we summarize the relevant literature (§2) and describe the basic discrete cloth model (§3).

2 Related Work

For brevity, we review work on stretch resistance; for broad surveys on cloth simulation see [House and Breen 2000; Choi and Ko 2005].

The most general approach is to treat cloth as an elastic material [Terzopoulos et al. 1987; Breen et al. 1994; Eberhardt et al. 1996; Baraff and Witkin 1998; Choi and Ko 2002]. To reduce visible stretching, elastic models typically adopt large elastic moduli or stiff springs, degrading numerical stability [Hauth et al. 2003].

To address the stiffness of the resulting differential equations, Baraff and Witkin [1998] proposed implicit integration, allowing for large, stable timesteps; adaptive timestepping was required to prevent over-stretching. Eberhardt [2000] and Boxerman *et al.* [2003] adopted implicit-explicit (IMEX) formulations, which treat only a subset of forces implicitly. Our method is closely related to the IMEX approach, in the sense that stretching forces are singled out for special treatment.

These works, and many of their sequels, improved performance by allowing some perceptible stretch of the fabric. In the quasi-inextensible regime, however, implicit methods encounter numerical limitations [Volino and Magnenat-Thalmann 2001; Boxerman 2003; Hauth et al. 2003]: the condition number of the implicit system grows with the elastic material stiffness, forcing iterative solvers to perform many iterations; additionally, timestepping algorithms such as Backward Euler and BDF2 introduce undesirable numerical damping when the system is stiff [Boxerman 2003].

Given a stiff differential equation, an alternative to implicit integration is to reduce the stiff component and reformulate it as a constraint [Hairer et al. 2002]. In the smooth setting, the penalty-force and constraint-based approaches are equivalent in the limit of an infinitely stiff penalty term [Bercovier and Pat 1984]. In the discrete setting, the constraint-based approach may be implemented with various iterative or global algorithms, as surveyed below:

Iterative enforcement Provot [1995] corrected edge lengths by iteratively displacing the incident vertices on stretched springs. While simple to implement, this approach suffers from poor convergence since each displacement may stretch other incident springs. Therefore, Provot's method is used in cases where tight tolerances are not required, *e.g.*, [Desbrun et al. 1999; Meyer et al. 2001; Fuhrmann et al. 2003]. Bridson *et al.* [2002; 2003] used Provot's approach in conjunction with strain rate limiting, bounding the rate of change of spring length per timestep to 10% of the current length. Müller *et al.* [2006] used a non-linear Gauss-Seidel approach to enforce inextensibility on each constraint separately. Bridson *et al.* observed that iterative strain limiting algorithms behave essentially as Jacobi or Gauss-Seidel solvers. In this light, it is not surprising that for finely-discretized quasi-inextensible fabrics, iterative constraint enforcement requires a prohibitive number of iterations (see §5).

Global enforcement In contrast to iterative constraint enforcement, House *et al.* [1996] used Lagrange multipliers with CLM to treat stretching, and presented a hierarchical treatment of the constraint forces. The Lagrange multiplier approach alleviates the difficulties associated with poor numerical conditioning and artificial damping. House *et al.* later encountered difficulties in handling collision response within the proposed framework [2000]. By building on the velocity-filter paradigm, our method handles both inextensibility and complex collisions.

House *et al.* formulated constraints as in [Witkin et al. 1990], which is subject to numerical drift that may be exacerbated by the discontinuities introduced during collision response. Drift may be attenuated using constraint-restoring springs, but the authors reported difficulty in adjusting the spring coefficients. We postulate that one reason for their difficulties with drift was consequent to the linearization of the constraint equation, which permitted higher order errors to accumulate over time. Our method does not linearize the constraint equations, and therefore it is not subject to drift.

Recently, Tsiknis [2006] proposed triangle-based strain limiting together with a global stitching step for stable constraint enforcement. Hong *et al.* [2005] used a linearized implicit formulation in order to improve stability of constrained dynamics. This allowed for larger timesteps and reduced the need for springs to maintain the cloth on the constraint manifold. Both of these approaches enforce inextensibility only for strain exceeding 10%.

In summary, when the tolerance for stretching is very small, modeling stretch response with spring-based or strain-limiting approaches is costly and even intractable; constraint-based methods present a promising alternative. The remainder of this paper discusses algorithms that excel at simulating quasi-inextensible cloth.

3 Cloth Model

Woven fabrics are not a continuous material, rather they are a complex mechanical network of interleaving yarn [Breen et al. 1994]. Since the constituent yarn is often quasi-inextensible, the material's warp and weft directions do not stretch perceptibly.

In imposing inextensibility on all edges of a triangle mesh, one quickly runs into parasitic stiffness in the bending modes, or *locking* [Zienkiewicz and Taylor 1989], since locally-convex regions of a triangle mesh are rigid under isometry. Instead, we consider warp-weft aligned quadrilateral meshes with a sparse number of triangles (*quad-dominant* meshes). A degree of freedom (DOF) counting argument suggests that constraining all edges of a quad mesh may circumvent the rigidification that occurs with triangle meshes: Given n vertices, we have $3n$ positional DOFs; their triangulation (resp. quadrangulation) introduces approximately $3n$ (resp. $2n$) edges, with corresponding inextensibility constraints. Subtracting constraints from positional DOFs leaves nearly zero DOFs for a triangulation. In the case of a quadrangulation, $O(n)$ DOFs remain, and we see that in a flat configuration they correspond to the normal direction at each vertex. Furthermore, under general mesh positions, the constraints are linearly independent, with a full-rank Jacobian treatable by a direct solver (§4).

We ask that a warp- or weft-aligned quad edge, $(\mathbf{p}_a, \mathbf{p}_b)$, maintain its undeformed length, l, by enforcing

$$C(\mathbf{p}_a,\mathbf{p}_b) = \|\mathbf{p}_b - \mathbf{p}_a\|^2/l \;-\; l = 0\ . \tag{1}$$

The solve will require the *constraint gradient*

$$\nabla_{\mathbf{p}_b} C(\mathbf{p}_a,\mathbf{p}_b) = 2(\mathbf{p}_b - \mathbf{p}_a)/l\ . \tag{2}$$

Since shearing modes excite only a mechanical interaction of warp and weft, and not a stretching of yarn, fabric does indeed shear perceptibly. Therefore, we model shear using non-stiff stretch springs applied on both diagonals of each quad.

The complete model of in-plane deformation is compatible with an existing code's quad- or triangle-based treatment of bending and collisions. With this simple formulation of inextensibility constraints in place, what is needed is an efficient method for enforcing constraints. In the following, we develop such a method.

4 Constrained Dynamics

Given a quadrilateral mesh with n vertices and m edges, the numerical integration algorithm for constrained dynamics can be developed directly from the augmented Lagrange equation [Marsden 1999],

$$L(\mathbf{x},\mathbf{v}) = \frac{1}{2}\mathbf{v}^T\mathbf{M}\mathbf{v} - V(\mathbf{x}) - \mathbf{C}(\mathbf{x})^T\boldsymbol{\lambda}\ ,$$

where $\mathbf{x}(t)$ is the time-varying $3n$-vector of vertex positions, $\mathbf{v}(t) = \dot{\mathbf{x}}(t)$ is its time derivative, $\mathbf{M}$ is the $3n\times 3n$ mass matrix, and $V(\mathbf{x})$ is the stored energy (*e.g.*, bending, shear, and gravity). $\mathbf{C}(\mathbf{x})$ is the m-vector of constraints, with the i^{th} entry corresponding to the violation of inextensibility of the i^{th} edge, as computed by (1); $\boldsymbol{\lambda}$ is the m-vector of Lagrange multipliers. The corresponding Euler-Lagrange equations are

$$\mathbf{M}\dot{\mathbf{v}} = -\nabla V(\mathbf{x}) - \nabla\mathbf{C}(\mathbf{x})^T\boldsymbol{\lambda}\ , \tag{3}$$
$$\mathbf{C}(\mathbf{x}) = \mathbf{0}\ , \tag{4}$$

where $\nabla \equiv \nabla_{\mathbf{x}}$ is the gradient with respect to position, and $-\nabla V(\mathbf{x})$ is the potential force. The term $-\nabla\mathbf{C}(\mathbf{x})^T\boldsymbol{\lambda}$ may be viewed as the constraint-maintaining force, where the factors $-\nabla\mathbf{C}(\mathbf{x})^T$ and $\boldsymbol{\lambda}$ determine the direction and scaling for the force, respectively. $\nabla\mathbf{C}(\mathbf{x})$ is a rectangular matrix whose dimensions are $m\times 3n$.

For simulation, we must discretize (3) and (4) in time using one of various schemes, each with benefits and drawbacks. One may choose differing explicit or implicit schemes for the potential and the constraint forces (similarly, potential forces are split and separately discretized in [Ascher et al. 1997]). The discrete equations replace $\mathbf{x}(t)$ and $\mathbf{v}(t)$ with $\{\mathbf{x}^0, \mathbf{x}^1, \mathbf{x}^2, \ldots\}$ and $\{\mathbf{v}^0, \mathbf{v}^1, \mathbf{v}^2, \ldots\}$, where $\mathbf{x}^n$ and $\mathbf{v}^n$ are the position and velocity of the mesh at time $t = nh$, and h is the size of the timestep.

One widely-used family of discretizations includes SHAKE and RATTLE, which extend the (unconstrained) Verlet scheme [Hairer et al. 2002] by considering a constraint force direction, $-\nabla\mathbf{C}(\mathbf{x})^T$, evaluated at the *beginning* of the timestep.

Unfortunately, enforcing length-preserving constraints with SHAKE fails for four common geometric configurations, which we refer to as (Q1)–(Q4) and depict in Figure 2. This figure is a reproduction from [Barth et al. 1994], which discusses these drawbacks in SHAKE but does not offer a solution. In the figure, solid and hollow dots represent edge endpoints at the start and end of the timestep, as the particles would evolve if no constraints were applied. If the constraint direction, $-\nabla\mathbf{C}(\mathbf{x})^T$, is evaluated at the beginning of the timestep, $\mathbf{x}^n$, as in SHAKE, then no scaling, $\boldsymbol{\lambda}$, of the constraint direction yields a satisfied end-of-timestep constraint, $\mathbf{C}(\mathbf{x}^{n+1}) = \mathbf{0}$. Numerically, for (Q2)–(Q4) this observation manifests as a singular Jacobian in Newton's method. These four cases correspond to rapid change in edge length or orientation; in practice, they occur often.

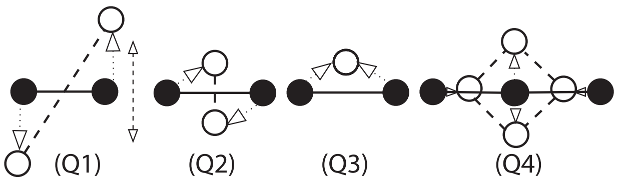

Figure 2: Failure modes of methods using an explicit constraint direction. *Reproduced from a discussion of SHAKE in [Barth et al. 1994].*

4.1 Implicit constraint direction (ICD)

Consider evaluating the constraint direction, $-\nabla\mathbf{C}(\mathbf{x})^T$, at the *end* of the timestep. We observe (and prove in Appendix A) that this resolves (Q1), (Q2) and (Q4); (Q3) remains, but is automatically remedied by decreasing the timestep. Consider the ICD timestep, which treats potential forces explicitly[1]:

$$\begin{aligned} \mathbf{v}^{n+1} &= \mathbf{v}^n - h\mathbf{M}^{-1}\left(\nabla V(\mathbf{x}^n) + \nabla\mathbf{C}(\mathbf{x}^{n+1})^T\boldsymbol{\lambda}^{n+1}\right) , \\ \mathbf{x}^{n+1} &= \mathbf{x}^n + h\mathbf{v}^{n+1} , \\ \mathbf{C}(\mathbf{x}^{n+1}) &= \mathbf{0} . \end{aligned}$$

Define $\mathbf{x}_0^{n+1} = \mathbf{x}^n + h\mathbf{v}^n - h^2\mathbf{M}^{-1}\nabla V(\mathbf{x}^n)$, *i.e.*, $\mathbf{x}_0^{n+1}$ is the position at the end of an unconstrained timestep; define $\delta\mathbf{x}^{n+1} = \mathbf{x}^{n+1} - \mathbf{x}_0^{n+1}$, *i.e.*, $\delta\mathbf{x}^{n+1}$ is the correction of the unconstrained step. Next, eliminate $\mathbf{v}^{n+1}$ by rewriting the above system as two equations, $\mathbf{F}(\delta\mathbf{x}^{n+1}, \boldsymbol{\lambda}^{n+1}) = \mathbf{0}$ and $\mathbf{C}(\mathbf{x}^{n+1}) = \mathbf{0}$, in the free variables $\delta\mathbf{x}^{n+1}$ and $\boldsymbol{\lambda}^{n+1}$, keeping in mind that $\mathbf{x}^{n+1}$ is a linear function in $\delta\mathbf{x}^{n+1}$, and defining

$$\mathbf{F}(\delta\mathbf{x}^{n+1}, \boldsymbol{\lambda}^{n+1}) = \delta\mathbf{x}^{n+1} + h^2\mathbf{M}^{-1}\nabla\mathbf{C}(\mathbf{x}^{n+1})^T\boldsymbol{\lambda}^{n+1} .$$

$\mathbf{F}(\delta\mathbf{x}^{n+1}, \boldsymbol{\lambda}^{n+1})$ and $\mathbf{C}(\mathbf{x}^{n+1})$ are the residuals of the discretization of (3) and (4), respectively. In particular, $\mathbf{F}$ measures the deviation of the trajectory away from that dictated by the governing (potential and constraint) forces; equivalently, it states that the correction of the unconstrained step is due to the constraint forces. $\mathbf{C}$ measures the deviation from the constraint manifold (in our case, the extensibility of the material). To implement ICD, we solve for the roots of $\mathbf{F}$ and $\mathbf{C}$ up to a desired tolerance using Newton's method. Solving for an ICD step is costly, because there are many unknowns ($\approx 5n$), and each Newton step requires the solution of an *indefinite* linear system, whose matrix is costly to assemble. In §4.3, we develop an approximation to ICD that addresses these drawbacks without sacrificing constraint accuracy or robustness. To arrive at this fast projection method, the following section considers ICD from an alternative, geometric viewpoint.

[1]For an implicit treatment, write $\nabla V(\mathbf{x}^{n+1})$ in place of $\nabla V(\mathbf{x}^n)$.

4.2 Step and project (SAP)

Consider for a moment an alternative approach to constrained integration in two steps: (a) step forward only the potential forces to arrive at the unconstrained position, $\mathbf{x}_0^{n+1}$; (b) enforce the constraints by projecting onto the constraint manifold $\mathscr{M} = \{\mathbf{x}^{n+1} | \mathbf{C}(\mathbf{x}^{n+1}) = \mathbf{0}\}$. Methods of this form are known as manifold-projection methods [Hairer et al. 2002]. To define a specific method, we must choose a projection operator. In the method we refer to as SAP, we write the projection of the unconstrained point onto the constraint manifold as $\mathbf{x}_0^{n+1} + \delta\mathbf{x}^{n+1}$, so that the projected point extremizes the objective function

$$W(\delta\mathbf{x}^{n+1}, \boldsymbol{\lambda}^{n+1}) = \frac{1}{2h^2}(\delta\mathbf{x}^{n+1})^T\mathbf{M}(\delta\mathbf{x}^{n+1}) + \mathbf{C}(\mathbf{x}^{n+1})^T\boldsymbol{\lambda}^{n+1} ,$$

with respect to the free variables $\delta\mathbf{x}^{n+1}$ and $\boldsymbol{\lambda}^{n+1}$. Simply put, we choose the point on the constraint manifold closest to $\mathbf{x}_0^{n+1}$. To define *closest*, we need a measure of distance. Take $\mathbf{M}$ as the physical mass matrix (usually arising from a finite-basis representation of $\mathbf{x}$ and a surface mass density). Then the choice $(\delta\mathbf{x}^{n+1})^T\mathbf{M}(\delta\mathbf{x}^{n+1})$ corresponds to the L^2 norm of the mass-weighted displacement of the mesh as it moves from $\mathbf{x}_0^{n+1}$ to $\mathbf{x}^{n+1}$. Formally, it is a discretization of the smooth integral

$$\int_S \|\mathbf{x}^{n+1} - \mathbf{x}_0^{n+1}\|^2 \rho \, \mathrm{d}A ,$$

evaluated over the reference (material) domain, S. Here $\mathbf{x}^{n+1}$ and $\mathbf{x}_0^{n+1}$ are the piecewise linear immersion functions mapping each point of S into $\mathbb{R}^3$, and ρ is the (possibly nonuniform) surface mass density. We use $\|\cdot\|$ to denote the Euclidean norm in $\mathbb{R}^3$.

Theorem 1: ICD $\equiv$ SAP.

Proof: The stationary equations for $W(\delta\mathbf{x}^{n+1}, \boldsymbol{\lambda}^{n+1})$ are the ICD equations, $\mathbf{F}(\delta\mathbf{x}^{n+1}, \boldsymbol{\lambda}^{n+1}) = \mathbf{0}$ and $\mathbf{C}(\mathbf{x}^{n+1}) = \mathbf{0}$. □

Corollary In 4.1, we interpreted the roots of $\mathbf{C}$ and $\mathbf{F}$ from the ICD view. We can interpret these roots from the SAP view as follows: $\mathbf{C}(\mathbf{x}^{n+1}) = \mathbf{0}$ corresponds to finding *some* point on the constraint manifold. $\mathbf{C}(\mathbf{x}^{n+1}) = \mathbf{0}$ with $\mathbf{F}(\delta\mathbf{x}^{n+1}, \boldsymbol{\lambda}^{n+1}) = \mathbf{0}$ corresponds to finding the *closest* point on the constraint manifold.

4.3 Fast projection method

To solve SAP, one might extremize $W(\delta\mathbf{x}^{n+1}, \boldsymbol{\lambda}^{n+1})$ using Newton's method: each iteration would improve upon a guess for *the* shortest step, $\delta\mathbf{x}^{n+1}$ that projects $\mathbf{x}_0^{n+1}$ onto the constraint manifold.

Algorithm 1 Fast projection *is a velocity filter that enforces constraints. It combines the robustness of using an implicit constraint direction with the efficiency of approximate manifold projection.*

```
Input: $\tilde{\mathbf{v}}$ // candidate velocity
Input: $\tilde{\mathbf{x}}$ // known start-of-step position
1: $j \leftarrow 0$
2: $\mathbf{x}_0 \leftarrow \tilde{\mathbf{x}} + h\tilde{\mathbf{v}}$ // unconstrained timestep
3: while strain of $\mathbf{x}_j$ exceeds threshold do
4:     Solve linear system (7) for $\delta\boldsymbol{\lambda}_{j+1}$
5:     Evaluate (5) to obtain $\delta\mathbf{x}_{j+1}$
6:     $\mathbf{x}_{j+1} \leftarrow \mathbf{x}_j + \delta\mathbf{x}_{j+1}$
7:     $j \leftarrow j+1$
8: end while
Output: $\frac{1}{h}(\mathbf{x}_j - \tilde{\mathbf{x}})$ // constraint-enforcing velocity
```

Fast projection also uses a sequence of iterations, but it relaxes the requirement of SAP: starting with the unconstrained position, $\mathbf{x}_0^{n+1}$, we propose to find a *close*, but not necessarily closest, point on the constraint manifold, by taking a sequence of "smallest" steps. Fast projection starts at $\mathbf{x}_0^{n+1}$, and takes a sequence of steps, $\delta\mathbf{x}_j^{n+1}$, $j = 1, 2, \ldots$, toward the constraint manifold, with each step as short as possible.

A step of fast projection Projection onto the constraint manifold occurs at a fixed instant in time. Therefore, we omit the superscripts $(n+1)$, which refer to time, in order to emphasize the subscripts, j, which refer to a specific iteration of fast projection, *e.g.*, we write the input position, $\mathbf{x}_0^{n+1}$, as $\mathbf{x}_0$, and progressively closer approximations to the constrained position as $\mathbf{x}_1, \mathbf{x}_2, \ldots$.

Formally, the $(j+1)^{th}$ step of fast projection, $\mathbf{x}_{j+1} = \mathbf{x}_j + \delta\mathbf{x}_{j+1}$, extremizes the objective function

$$\overline{W}(\delta\mathbf{x}_{j+1}, \delta\boldsymbol{\lambda}_{j+1}) = \frac{1}{2h^2}(\delta\mathbf{x}_{j+1})^T\mathbf{M}(\delta\mathbf{x}_{j+1}) + \mathbf{C}(\mathbf{x}_{j+1})^T\delta\boldsymbol{\lambda}_{j+1} \ ,$$

with respect to the step increment, $\delta\mathbf{x}_{j+1}$, and the auxiliary variable $\delta\boldsymbol{\lambda}_{j+1}$. Expanding the constraint to first order,

$$\mathbf{C}(\mathbf{x}_{j+1}) = \mathbf{C}(\mathbf{x}_j + \delta\mathbf{x}_{j+1}) \approx \mathbf{C}(\mathbf{x}_j) + \nabla\mathbf{C}(\mathbf{x}_j)\delta\mathbf{x}_{j+1} \ ,$$

we obtain a quadratic objective function, whose stationary equations with respect to $\delta\mathbf{x}_{j+1}$ and $\delta\boldsymbol{\lambda}_{j+1}$ are

$$\delta\mathbf{x}_{j+1} = -h^2\mathbf{M}^{-1}\nabla\mathbf{C}(\mathbf{x}_j)^T\delta\boldsymbol{\lambda}_{j+1} \ , \quad (5)$$

$$\nabla\mathbf{C}(\mathbf{x}_j)\delta\mathbf{x}_{j+1} = -\mathbf{C}(\mathbf{x}_j) \ . \quad (6)$$

Substituting (5) into (6), we eliminate $\delta\mathbf{x}_{j+1}$ and solve a linear system in $\delta\boldsymbol{\lambda}_{j+1}$:

$$h^2\left(\nabla\mathbf{C}(\mathbf{x}_j)\mathbf{M}^{-1}\nabla\mathbf{C}(\mathbf{x}_j)^T\right)\delta\boldsymbol{\lambda}_{j+1} = \mathbf{C}(\mathbf{x}_j) \ . \quad (7)$$

Since the linear system matrix involves $\mathbf{M}^{-1}$, the assembly of this system is most efficient for diagonal (*e.g.*, lumped) mass matrices. Finally, we compute the increment (5) to obtain $\mathbf{x}_{j+1} = \mathbf{x}_j + \delta\mathbf{x}_{j+1}$.

As with ICD/SAP, a fast projection step requires a linear solve. However, fast projection's system, (7), is smaller ($\approx 2n \times 2n$ compared to $\approx 5n \times 5n$), positive definite (compared to indefinite) and sparser. As a result it is considerably cheaper to evaluate, assemble, and solve than its ICD/SAP counterpart.

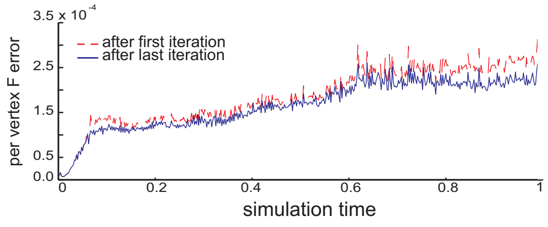

Figure 3: Effect of fast projection on the residual. *Using the ballet dancer sequence, at each timestep* (horizontal axis) *we measured the residual,* **F** *(*vertical axis*), after the first and last iterations of fast projection* (dashed-red and solid-blue curves, respectively).

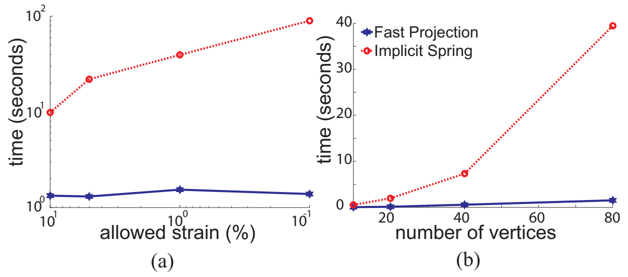

Figure 4: Performance of fast projection *vs.* implicit springs. *For a 1D chain simulated in MATLAB, we plot the computation time of one simulated second, as a function (a) of permissible strain (log-log plot for 80 vertices), and (b) of discretization resolution (linear plot for* 1% *permissible strain).*

Fast projection algorithm We repeatedly take fast projection steps until the maximal strain is below a threshold, *i.e.*, the constraint may be satisfied up to a given tolerance. This process is summarized in Algorithm 1.

Fast projection finds a manifold point, $\mathbf{x}^{n+1}$, that is close, but not closest, to the unconstrained point, $\mathbf{x}_0^{n+1}$. Referring to the Corollary, we conclude that fast projection exactly solves $\mathbf{C} = 0$ while it approximates $\mathbf{F} = 0$.

One important question is whether the fast projection's error in **F** is acceptable. Compare a sequence of fast projection iterations to ICD/SAP's sequence of Newton iterations. The first iteration of these methods is identical. At the end of this first iteration, $\mathbf{F}, \mathbf{C} \in O(h^2)$. Additional fast projection iterations seek $\mathbf{C} \to 0$, and since $C \in O(h^2)$, increments in $\mathbf{x}$ are $O(h^2)$, therefore **F** remains in $O(h^2)$. Observe that $\mathbf{F} \in O(h^2)$ is considered acceptable in many contexts, *e.g.*, [Baraff and Witkin 1998; Choi and Ko 2002] halt the Newton process after a single iteration.

To verify this claim, we measured **F** throughout the ballet dancer sequence. As recorded in Figure 3, the first iteration of the fast projection method eliminates first-order error. The remaining iterations perturb **F** only to higher-order (often decreasing the error further).

4.4 Implementation

We implement fast projection as a velocity filter, enabling easy integration into our existing cloth simulation system; refer to Algorithm 1. Step 3 requires solving a sparse symmetric positive definite linear system; we use the PARDISO [Schenk and Gärtner 2006] solver. Each row of $\nabla\mathbf{C}(\mathbf{x}_j^{n+1})$ corresponds to one edge, and is computed using (2). The right-hand side, $C(\mathbf{x}_j^{n+1})$, is given by (1).

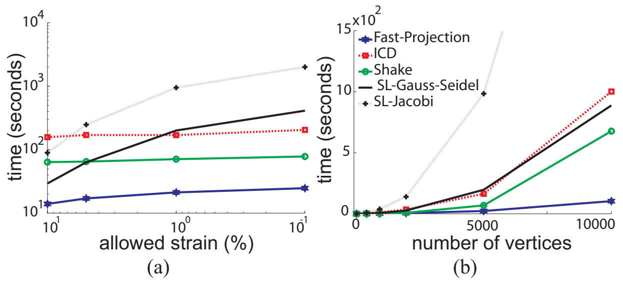

Figure 5: Performance of several constraint-enforcing methods. *For a 2D cloth, simulated in C++, we plot the computation time of one simulated second, as a function (a) of permissible strain (log-log plot for 5041 vertices), and (b) of discretization resolution (linear plot for* 1% *permissible strain).*

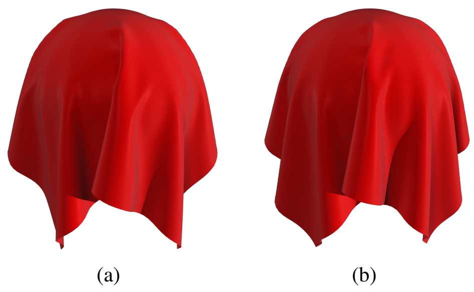

Figure 6: Qualitative visual comparison. *Snapshot of a cloth draped using (a) fast projection and (b) implicit constraint direction.*

5 Results

We describe several experiments comparing various stretch-enforcement methods. All timings are with reference to a single process on a 2.66GHz Intel Core 2 Duo.

One-dimensional chain Our first experiment compares the performance of fast projection against an implicit treatment of stiff springs. We observe the scaling of computational cost as a function of (a) permissible strain and (b) mesh resolution.

The physical setup consists of a chain pinned at the top node and released to free fall under gravity. The simple 1D chain resists stretching, but not bending.

In this didactic example, timings refer to MATLAB's (sparse) direct solver. Our method shows asymptotically better performance as permissible strain vanishes (see Figure 4a). Likewise, our algorithm exhibits favorable performance as mesh resolution increases (see Figure 4b). Using 80 vertices and 1% strain, the fast projection method achieves a 25× speedup. Note that there exists considerable difficulty in setting spring coefficients *a priori* to satisfy a given strain limit. For settings more pragmatic than a simple chain, such as the following draping experiment, we are unable (despite considerable effort) to set spring coefficients that achieve a prescribed small strain. This explains why spring methods are often treated with strain-limiting procedures.

Draping cloth The next experiment compares fast projection, ICD, SHAKE, and the strain limiting approach. We evaluate how the spatial discretization and permissible strain affect performance of these four algorithms. The setup consists of draping a cloth over a polygonal model of a sphere. We measure strain before the collision reaction pass.

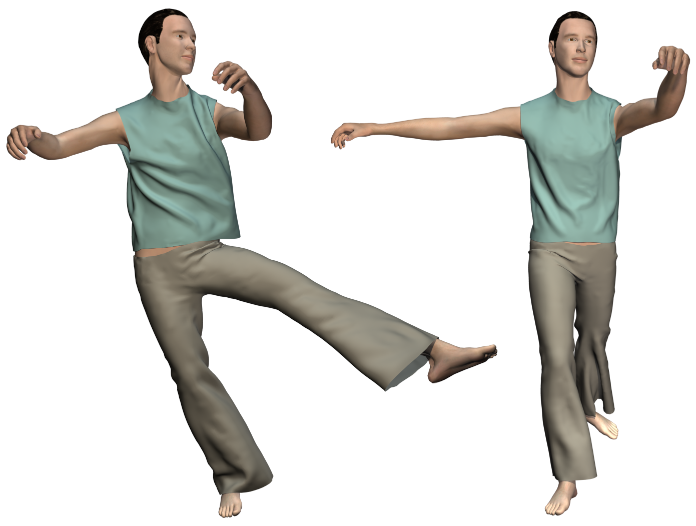

Figure 7: Inextensibility and dynamics. *Inextensibility ensures that the tight-fitting pants do not drop past the dancer's narrow waist. Using fast projection, an implicit treatment of shear and bending, and a mesh with* 10600 *vertices, the average simulation time per (30Hz) frame was* 9 *seconds.*

For the strain limiting algorithms (both Jacobi and Gauss-Siedel), we iterate until strain is in the permissible range. With Gauss-Siedel, we apply a random permutation to reduce bias resulting from the particular edge ordering. For SHAKE, we use the acceleration suggested in [Barth et al. 1994] to rebuild the matrix once per step or when it fails to converge. As a consequence, the algorithm requires extremely small timesteps to converge, but each timestep is relatively inexpensive, as matrix re-assembly and re-factoring is infrequent. ICD is able to use larger timesteps than SHAKE and still converge, however, since each timestep is substantially more expensive than a SHAKE step, the overall time is higher. Figure 5a shows a timing comparison of these methods, and Figure 5b compares performance as the stiffness is increased for a cloth mesh with approximately 5000 vertices. All CLM methods scale equally well, asymptotically better than the strain limiting approach, with the fast projection being the fastest. As we refine the resolution, and allow strain of 1% (Figure 5b), the fast projection method outperforms the other methods.

Figure 6 shows the same frame from simulations that use the fast projection and ICD methods, with qualitatively similar results. Figures 7 and 8 show still frames from more complex simulations demonstrating that fast projection is capable of producing complex, realistic simulations of cloth.

6 Discussion

Our experiments focus on measuring the performance of enforcing inextensibility using CLM compared to strain limiting and stiff springs. In addition to the direct benefit of fast projection on computation times, further benefits can be reaped from the resulting inextensibility. For example, the work of Bergou *et al.* [2006] assumes inextensibility in order to accelerate bending computation. In adopting the velocity-filtering viewpoint, we gain speed, simplicity, and software modularity—all key to a practical and maintainable implementation. However, this comes at a theoretical cost: there is no longer an efficient way to perfectly enforce both ideal inextensibility and ideal collision handling, since one filter must execute before the other, and both ideals correspond to sharp constraints. To enforce both perfectly would require combining them in a single pass, an elegant and exciting prospect from the standpoint of theory, but one which is likely to introduce considerable complexity and convergence challenges. Practically, we observe that this drawback does not cause artifacts in our simulation, for several reasons: first, we execute collision-handling last, to avoid glaring collision artifacts, yet we assert that empirically our strain remains negligible, as required. Second, unlike constraint-enforcement approaches such as [Witkin et al. 1990], the inextensibility filter does not assume that the constraint is maintained at the beginning of the timestep and errors are not accumulated during the simulation.

Conclusion Despite the fact that the most common fabrics do not visibly stretch when draped over the body, the trend in our community is to favor stretching formulations based on penalty-springs. The consequent numerical difficulties are then addressed by a combination of (a) relaxing realism by allowing 10% strain, and (b) adopting simple iterative strain and strain-rate algorithms that have poor convergence behavior. With Constrained Lagrangian Mechanics as our alternative point of departure, we demonstrate a straightforward filter, with good convergence behavior, for enforcing inextensibility. We provide one immediate and pragmatic approach to fast and realistic fabric simulation using CLM, and we hope that it will spur a renaissance of activity along this direction.

Acknowledgments We are grateful for the valuable feedback from our reviewers, and in particular for the keen eyes and diligent guidance of the primary reviewer. We thank OptiTex for providing the 3D garment geometry as well as the animated figurines. We are grateful to David Ismailov for setting up the cloth models, and Ruzz Oved and Yaniv Gorali for lighting and shading our scenes. Our work benefited from the valuable insights of Jerrold E. Marsden, Ari Stern, and Max Wardetzky, and from the generous support of the NSF (MSPA 0528402, CSR 0614770, CAREER 0643268), Autodesk, mental images, NVIDIA, and Elsevier.

References

ASCHER, U. M., RUUTH, S. J., AND SPITERI, R. J. 1997. Implicit–explicit Runge–Kutta methods for time-dependent partial differential equations. *Applied Numerical Mathematics: Transactions of IMACS 25*, 2–3, 151–167.

BARAFF, D., AND WITKIN, A. 1998. Large steps in cloth simulation. In *Proceedings of SIGGRAPH 98*, ACM Press / ACM SIGGRAPH, New York, NY, USA, 43–54.

BARTH, E., KUCZERA, K., LEIMKUHLER, B., AND SKEEL, R. 1994. Algorithms for Constrained Molecular Dynamics. March.

BERCOVIER, M., AND PAT, T. 1984. A C^0 finite element method for the analysis of inextensibile pipe lines. *Computers and Structures 18*, 6, 1019–1023.

BERGOU, M., WARDETZKY, M., HARMON, D., ZORIN, D., AND GRINSPUN, E. 2006. A quadratic bending model for inextensible surfaces. In *Fourth Eurographics Symposium on Geometry Processing*, 227–230.

BOXERMAN, E. 2003. *Speeding up cloth simulation*. Master's thesis, University of British Columbia.

BREEN, D. E., HOUSE, D. H., AND WOZNY, M. J. 1994. Predicting the drape of woven cloth using interacting particles. In *Proceedings of ACM SIGGRAPH 1994*, ACM Press/ACM SIGGRAPH, New York, NY, USA, 365–372.

BRIDSON, R., FEDKIW, R. P., AND ANDERSON, J. 2002. Robust treatment of collisions, contact, and friction for cloth animation. *ACM Transactions on Graphics 21*, 3 (July), 594–603.

BRIDSON, R., MARINO, S., AND FEDKIW, R. 2003. Simulation of clothing with folds and wrinkles. In *Symposium on Computer animation*, 28–36.

CHOI, K.-J., AND KO, H.-S. 2002. Stable but responsive cloth. *ACM Transactions on Graphics" 21*, 3, 604–611.

CHOI, K.-J., AND KO, H.-S. 2005. Research problems in clothing simulation. *Computer-Aided Design 37*, 6, 585–592.

DESBRUN, M., SCHRÖDER, P., AND BARR, A. 1999. Interactive animation of structured deformable objects. In *Graphics Interface '99*, 1–8.

EBERHARDT, B., WEBER, A., AND STRASSER, W. 1996. A fast, flexible, particle-system model for cloth draping. *IEEE Comput. Graph. Appl. 16*, 5, 52–59.

EBERHARDT, B., ETZMUSS, O., AND HAUTH, M. 2000. Implicit-explicit schemes for fast animation with particle systems 137–154.

FUHRMANN, A., GROSS, C., AND LUCKAS, V. 2003. Interactive animation of cloth including self collision detection. In *WSCG '03*, 141–148.

GRIFFITHS, P., AND KULKE, T. 2002. Clothing movement—visual sensory evaluation and its correlation to fabric properties. *Journal of sensory studies 17*, 3, 229–255.

HAIRER, E., LUBICH, C., AND WANNER, G. 2002. *Geometric Numerical Integration*. No. 31 in Springer Series in Computational Mathematics. Springer-Verlag.

HAUTH, M., ETZMUSS, O., AND STRASSER, W. 2003. Analysis of numerical methods for the simulation of deformable models. *The Visual Computer 19*, 7-8, 581–600.

HONG, M., CHOI, M.-H., JUNG, S., WELCH, S., AND TRAPP, J. 2005. Effective constrained dynamic simulation using implicit constraint enforcement. In *International Conference on Robotics and Automation*, 4520–4525.

HOUSE, D. H., AND BREEN, D. E., Eds. 2000. *Cloth modeling and animation*. A. K. Peters, Ltd., Natick, MA, USA.

HOUSE, D. H., DEVAUL, R. W., AND BREEN, D. E. 1996. Towards simulating cloth dynamics using interacting particles. *International Journal of Clothing Science and Technology 8*, 3, 75–94.

MARSDEN, J. 1999. *Introduction to Mechanics and Symmetry*. Springer.

MEYER, M., DEBUNNE, G., DESBRUN, M., AND BARR, A. H. 2001. Interactive animation of cloth-like objects in virtual reality. *The Journal of Visualization and Computer Animation 12*, 1 (Feb.), 1–12.

MÜLLER, M., HEIDELBERGER, B., HENNIX, M., AND RATCLIFF, J. 2006. Position based dynamics. In *Proceedings of Virtual Reality Interactions and Physical Simulation (VRIPHYS)*, C. Mendoza and I. Navazo, Eds., 71–80.

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

SCHENK, O., AND GÄRTNER, K. 2006. On fast factorization pivoting methods for sparse symmetric indefinite systems. *Elec. Trans. Numer. Anal 23*, 158–179.

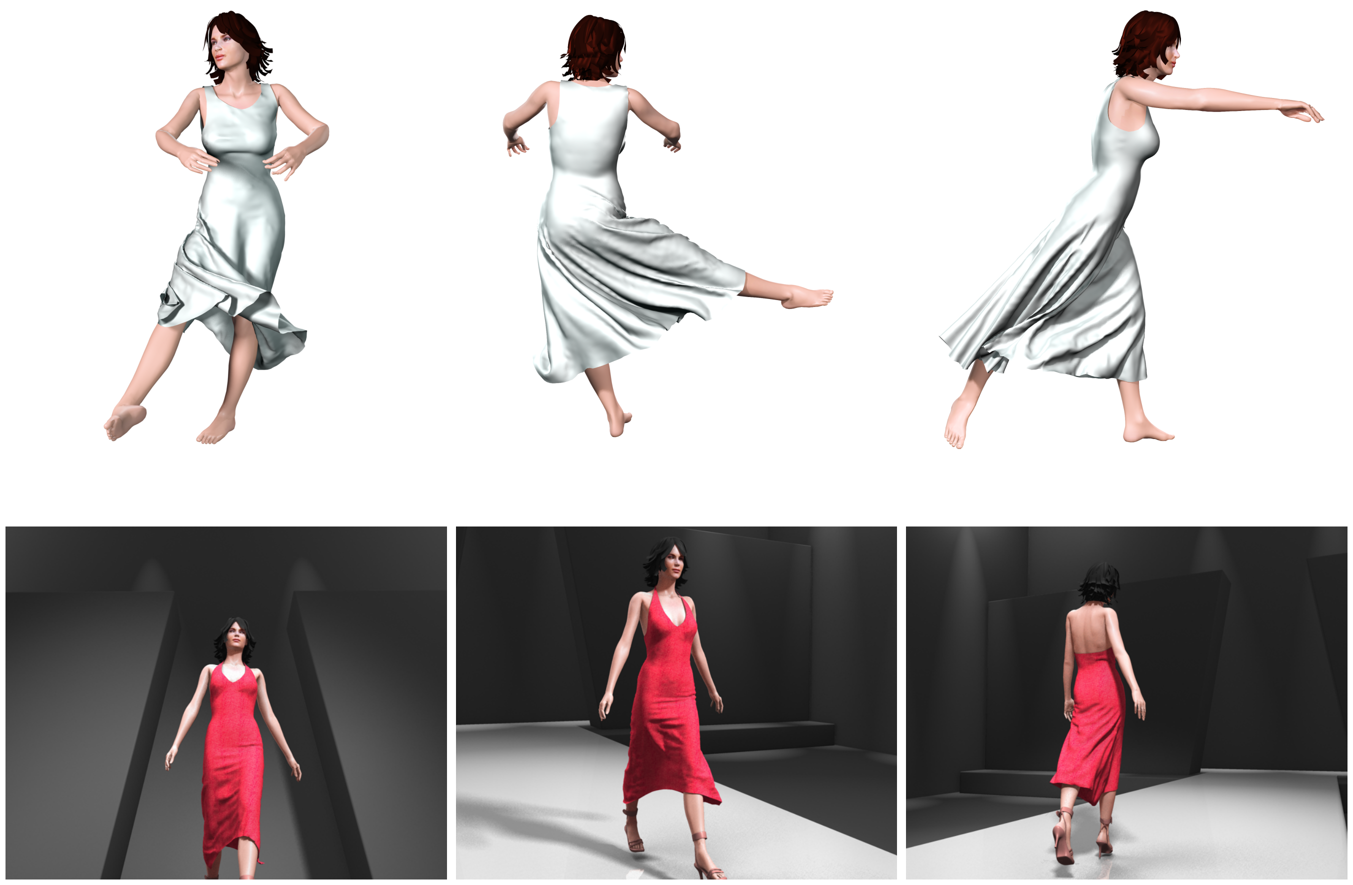

Figure 8: Enforcing inextensibility using fast projection yields lively motion with detailed wrinkles and folds. *Frames from ballet and runway sequences simulated using fast projection. The elastic term was integrated implicitly* (top) *and explicitly* (bottom)*, respectively. The cloth contains* 8325 (top) *and 10688* (bottom) *vertices, with average simulation time per (30Hz) frame of* 5.2 *and* 7.8 *seconds, respectively.*

TERZOPOULOS, D., PLATT, J., BARR, A., AND FLEISCHER, K. 1987. Elastically deformable models. In *Computer Graphics (Proceedings of ACM SIGGRAPH 87)*, ACM Press, New York, NY, USA, 205–214.

TSIKNIS, K. D. 2006. *Better cloth through unbiased strain limiting and physics-aware subdivision*. Master's thesis, The University of British Columbia.

VOLINO, P., AND MAGNENAT-THALMANN, N. 2001. Comparing efficiency of integration methods for cloth simulation. *Computer Graphics International*, 265–274.

WITKIN, A., GLEICHER, M., AND WELCH, W. 1990. Interactive dynamics. *Computer Graphics (Proceedings of ACM SIGGRAPH 90) 24*, 2, 11–21.

ZIENKIEWICZ, O. C., AND TAYLOR, R. C. 1989. *The finite element method*. McGraw Hill. 2.

Appendix A

We briefly explain why ICD and fast projection (FP) are not troubled by configurations (Q1), (Q2), and (Q4), and are resilient to (Q3). Facts about the behavior of SHAKE are taken from [Barth et al. 1994].

Q1 SHAKE's force $\nabla C(\mathbf{x}^n)^T \boldsymbol{\lambda}^{n+1}$ cannot reduce the single edge's length back to l; our force $\nabla C(\mathbf{x}_j^{n+1})^T \boldsymbol{\lambda}^{n+1}$ can reduce that edge's length back to l.

Q2 $\nabla C(\mathbf{x}_j^{n+1})$ and $\nabla C(\mathbf{x}^n)^T$ are both full-rank, yet SHAKE fails since $\nabla C(\mathbf{x}_j^{n+1})\mathbf{M}^{-1}\nabla C(\mathbf{x}^n)^T$ is singular; FP uses $\nabla C(\mathbf{x}_j^{n+1})\mathbf{M}^{-1}\nabla C(\mathbf{x}_j^{n+1})^T$, and ICD uses $\nabla C(\mathbf{x}_j^{n+1})D\nabla C(\mathbf{x}_j^{n+1})^T$, where D is a symmetric full-rank matrix; in both cases this product is not singular.

Q3 ICD and FP may fail if $\nabla C(\mathbf{x}_j^{n+1})$ is rank-deficient; for sufficiently small timestep, h, this case is always avoidable.

Q4 $\nabla C(\mathbf{x}^n)$ is rank-deficient, so SHAKE fails; ICD and FP do not use $\nabla C(\mathbf{x}^n)$.