Towards Simulating Cloth Dynamics Using Interacting Particles

Donald H. House  Richard W. DeVaul
Visualization Laboratory, Texas A&M University

David E. Breen
European Computer-Industry Research Centre, Munich

February 7, 1996
Abstract

This paper previews a new approach being developed for modeling the dynamic behavior of cloth. This work extends the cloth-particle static draping model of Breen and House to include dynamics, and extends constrained dynamics simulation techniques developed by Witkin, Gleicher and Welch to yield performance enhancements. Fundamental to this approach is a new hierarchical approximation algorithm for constrained dynamics simulation, which it is hoped will reduce the computational time demands of the algorithm to near real-time range.
1 Introduction

A number of techniques have been developed for predicting the drape of woven cloth in its final equilibrium configuration, producing results that range from the technical to the artistic. These draping models have followed two main directions. Historically, there have been a number of continuum models that treat cloth as a more-or-less homogeneous elastic medium, with draping solutions often calculated via finite-element techniques [8,10,14,16]. More recently, a particle model [1] was developed that attempts to directly capture the underlying fine-grained mechanical structure of cloth in a set of energy equations defined at yarn crossings. Figure 1 was produced as a test-case for this model. Draping solutions from this model are calculated via an ensemble energy-minimization technique, employing modified gradient following. Both the continuum and particle approaches share the engineer’s desire that they predict actual behavior of real materials. For example Figure 2, reproduced from [5], shows a comparison of actual versus simulated drape of two different types of fabric, computed via the particle model. It is also generally demanded that these models accurately determine such classical engineering quantities as stress and strain in the material or in its constituent components [9,11].

A number of other techniques have been developed for simulating the dynamics of cloth [17]. These models have tended to differ from the draping models in several ways. First, they have been motivated by the desire to portray the behavior of cloth in motion, often for purposes of animation and entertainment. Instead of the engineer’s desire for accuracy, these models were developed with the notion that reasonable approximate behavior is fine, as long as the model can be calculated quickly. Most of them are based on using elastic meshes, producing results that sometimes appear too stretchy or rubbery to someone who has made a careful study of cloth behavior. Also, these models are not capable of the complex buckles and folds of draped cloth. Nevertheless, they do produce effective animated sequences, and have been perfected to the point where they have been applied to clothing for animated characters [7].

In this paper we preview attempts that are currently being made to bridge the gap between the models that produce accurate draping behavior and the models that seek to reproduce dynamic behavior. We begin by giving a brief synopsis of our particle-based draping model. This is followed by a description of how we have used constraint methods to turn this model into one that purports to describe both cloth dynamics and draped equilibria. Finally, we outline our experimental methodology for calculating the constrained dynamics of this new model in a way that is computationaly tractable, and may make the model of real use in engineering, design, and animation.

2 A Particle-Based Draping Model

Since our particle-based approach to modeling drape does not follow traditional continuum methods, it may be unfamiliar to readers. To understand the extensions to the model that are proposed in this paper, it will first be necessary to understand the underlying methodology. Therefore, we include the following brief summary of the structure of this model. For further information, we refer the reader to the extensive particle system bibliography found in [2], to the detailed review of cloth modeling work found in [4], and to other papers on the particle-based drape model [1,3].

We model cloth as a collection of particles that conceptually represents the crossing points of warp and weft yarns in a plain weave. Important mechanical interactions that determine the behavior of woven fabric are discretized and lumped at these crossing points. We represent the various yarn-level structural constraints with energy functions that capture simple geometric relationships
between particles. These energy functions account for the four basic mechanical interactions of yarn collision, yarn stretching, out-of-plane bending, and trellising that are shown graphically in Figure 3. The potential energy for \( i \) is given by

\[
U_i = U_{\text{repel}} + U_{\text{stretch}} + U_{\text{bend}} + U_{\text{trellis}} + U_{\text{grav}}.
\]

(1)

In this equation, \( U_{\text{repel}} \) is an artificial energy of repulsion, that effectively keeps every other particle at a minimum distance, providing some measure of yarn collision detection, helping prevent self intersection of the cloth. \( U_{\text{stretch}} \) captures energy of tensile strain between each particle and its four-connected neighbors. \( U_{\text{bend}} \) is the energy due to yarns bending out of the local plane of the cloth, and \( U_{\text{trellis}} \) is the energy due to bending around a yarn crossing in the plane. \( U_{\text{grav}} \) is the potential energy due to gravity. Repelling and stretching are functions only of interparticle distance \( r_{ij} \) (Figure 3-Ia), whereas bending and trellising are functions of various angular relationships between segments joining particles (Figure 3-IIa and 3-IIIa). \( U_{\text{grav}} \) proportional to the height of the particle. Trellising occurs when yarns are held fast at a crossing and bend to create an “S-curve” in the local plane of the cloth, and is related to shearing in a continuous sheet of material.

We assume that the yarns in the fabric do not stretch significantly when a cloth is simply draping under its own weight. Therefore, the combined stretching and repelling energy function \( R + S \) shown in Figure 3-Ib is not empirical, and is meant only to provide collision prevention and a steep energy well that acts to tightly constrain each particle to a nominal distance \( \sigma \) from each of its 4-connected neighbors. We have had good success with the functions

\[
R(r_{ij}) = \begin{cases} 
C_0[(\sigma - r_{ij})^5 / r_{ij}] & r_{ij} \leq \sigma \\
0 & r_{ij} > \sigma,
\end{cases}
\]

(2)

and

\[
S(r_{ij}) = \begin{cases} 
0 & \sigma \leq r_{ij} \\
C_0[(r_{ij} - \sigma) / \sigma^5] & r_{ij} > \sigma,
\end{cases}
\]

(3)

where \( C_0 \) is a scale parameter.

The function \( U_{\text{repel}} \) prevents collision and self intersection, so it is calculated by summing over all particles, as given by

\[
U_{\text{repel}} = \sum_{j \neq i} R(r_{ij}).
\]

(4)

In practice, our simulation algorithm maintains a spatial enumeration, so that the summation need only be done over near neighbors. An energy well is produced by directly coupling each particle with the stretching function \( S \) only to its 4-connected neighbors, as given by

\[
U_{\text{stretch}} = \sum_{j \in N_i} S(r_{ij}),
\]

(5)

where \( N_i \) is the set of particle \( i \)'s 4-connected neighbors.

The particle energy due to gravity is simply defined as

\[
U_{\text{grav}} = m_i g h_i,
\]

(6)

where \( m_i \) and \( h_i \) are the mass and height of particle \( i \), and \( g \) is gravitational acceleration. The mass is of the small patch of cloth represented by the particle.
In contrast to stretching, we assume that bending and trellising are the significant contributors to the overall drape of cloth, when it is simply draping under its own weight.

We define a unit of the bending energy $B$ shown in Figure 3-IIb as a function of the angle formed by three particles along a weft or warp “thread line”, as shown in Figure 3-IIa. The complete bending energy is

$$U_{\text{bend},i} = \sum_{j \in M_i} B(\theta_{ij}),$$

where $M_i$ is the set of six angles $\theta_{ij}$ formed by the segments connecting particle $i$ and its eight nearest horizontal and vertical neighbors. This definition is used so that the derivative of bending energy reflects the total change in bending energy due to change in position of particle $i$. The redundancy in this formulation is taken care of later by proper scaling.

The phenomenon of trellising is diagrammed in Figure 3-IIIa and a corresponding unit of the trellising energy $T$ is shown in Figure 3-IIIb. Two segments are formed by connecting the two pairs of neighboring particles surrounding a central particle. An equilibrium crossing angle of $90^\circ$ is assumed, but one could model slippage by allowing this angle to change, over the course of a simulation, as a function of load. The trellis angle $\phi$ is then defined as the angle formed as one of the line segments moves away from this equilibrium. The complete function for our energy of trellising is

$$U_{\text{trellis},i} = \sum_{j \in K_i} T(\phi_{ij}),$$

where $K_i$ is the set of four trellising angles $\phi_{ij}$ formed around the four-connected neighbors of particle $i$. As with bending, this redundant formulation was chosen so that change in total energy with change in the particle’s position is completely accounted for locally.

The simulation of the model is implemented as a three-phase process operating over a series of small discrete time steps [12]. The first phase for a single time step calculates the dynamics of each particle as if it were falling freely under gravity in a viscous medium, and accounts for collisions between particles and surrounding geometry. The second phase performs an energy-minimization to enforce interparticle constraints. A stochastic element of the energy minimization algorithm serves both to avoid local minima and to perturb the particle grid, producing a more natural asymmetric final configuration. The third phase corrects the velocity of each particle to account for particle motion during the second phase.

The energy functions indicated in the curves in Figure 3 are similar in shape to those that we first used to verify the theoretical model. These initial functions were simply convenient ones that we knew would smoothly interpolate reasonable boundary conditions. In order to tie the model directly to the draping behavior of actual cloth, we developed a method for deriving the model’s energy equations from empirical mechanical data produced by the Kawabata Evaluation System [13]. This allowed us to calculate drapes such as those shown in Figure 2, that begin to accurately predict the drape characteristics of distinct fabric types. This work is detailed in [5, 6]. Our strategy was to use the Kawabata bending and shear plots, such as those for 100% cotton and 100% wool shown in Figure 4, to derive bending and trellising energy functions that were accurate for a particular type of cloth.

3 The Dynamic Model

On the surface, the conversion of the particle draping model to a dynamic model appears straightforward – one simply needs to apply the chain rule appropriately to differentiate the energy functions
with respect to each of the three coordinate directions, to produce a set of non-linear spring forces that are functions of particle displacements. However, this approach has serious computational tractability problems. Due to the very great difference between the stiffness of the yarns in the longitudinal direction compared to their stiffness when bending in or out of the plane of the cloth, the resulting differential equations are numerically very poorly formed – they are quite stiff. Compromising on longitudinal stiffness, of course, would produce a cloth model with undue stretching – making it more like a rubber sheet than cloth. This is precisely the problem that many of the earlier continuum dynamic models of cloth had.

To avoid this problem, we are conducting experiments with a model that represents the distances between adjacent particles (i.e. yarn crossings) by constraints that keep the interparticle distances fixed. Rather than using springs, as depicted in Figure 5a, to produce forces to correct constraint violations, constraint forces are applied directly and immediately, as depicted in Figure 5b, to counteract both external and internal forces tending to violate the constraints. Since constraint forces are calculated to counteract other forces, rather than to correct displacement errors, their effect is instantaneously distributed throughout the entire constrained system.

### 3.1 Constrained dynamics

In any dynamic simulation, the reaction of the state of the system to applied forces and torques is described by a set of differential equations involving the current state, and the applied forces and torques. In a particle-system simulation things are somewhat simplified, since there can be no torques. However, in a constrained particle-system, in addition to equations describing the dynamics of the system, a set of algebraic equations specifying constraint conditions is also specified. These provide conditions that must always be met by the system. For example, if we have two particles with positions \( \mathbf{x}_0 \) and \( \mathbf{x}_1 \), and the configuration of the system requires that these particles maintain a fixed distance \( \sigma \) from each other, the constraint equation

\[
C_0(\mathbf{x}_0, \mathbf{x}_1) = \left\| \mathbf{x}_1 - \mathbf{x}_0 \right\|^2 - \sigma^2,
\]

might be included. As long as \( C_0 = 0 \) the distance constraint between \( \mathbf{x}_0 \) and \( \mathbf{x}_1 \) is satisfied. A computational scheme for specifying and simulating such a constrained dynamical system is given in tutorial form by Witkin in [19]. A more in-depth discussion, with applications, is given in [18]. We briefly summarize the approach here.

A typical unconstrained dynamical particle system can be represented by the equation

\[
M\ddot{\mathbf{x}} = \mathbf{F}_E + \mathbf{F}_I.
\]  \hspace{2cm} (9)

relating accelerations in the system to its diagonal mass matrix \( M \), externally applied forces \( \mathbf{F}_E \), and internally generated forces \( \mathbf{F}_I \). The internally generated forces are given by

\[
\mathbf{F}_I = \mathcal{F}(\mathbf{x}, \dot{\mathbf{x}}),
\]  \hspace{2cm} (10)

where \( \mathcal{F} \) is a function capturing the system’s (possibly time varying) internal dynamics.

A system with dynamic constraints is one in which certain relationships between elements of the state must be maintained, although it may be inconvenient or inefficient to include any explicit mechanism in the dynamics of the system that assures this. Examples of such a system might be a cart constrained to ride on a track, or a bouncing ball constrained never to pass through the floor.
One way to maintain such constraints is to add a third set of forces \( \mathbf{F}_C \) to the system, whose sole purpose is to guarantee that the constraints are maintained. This yields an equation of the form

\[
M \ddot{\mathbf{x}} = \mathbf{F}_A + \mathbf{F}_I + \mathbf{F}_C.
\]  

(11)

On the surface it appears that solving Equation 11 numerically is no different from solving Equation 9. However, the presence of the constraint force term implies the existence of accelerations based on an algebraic relationship between applied forces and current state that is absent in Equation 9. It is now necessary to determine an appropriate set of constraint forces \( \mathbf{F}_C \) at each time step.

Suppose that we represent the constraints by a set of constraint functions \( \mathbf{C}(\mathbf{x}) \) with the stipulation that \( \mathbf{C} = 0 \). Assume that the initial state of the system is such that 1) all of the constraints are met, and 2) that none of the constraint equations are changing \( (\dot{\mathbf{C}} = 0) \). Then, we can assure that all constraints continue to be met if we can guarantee that constraint accelerations are held to zero over the entire course of the simulation, or

\[
\ddot{\mathbf{C}} = 0.
\]  

(12)

If \( J \) is the jacobian matrix \( J = \partial \mathbf{C}/\partial \mathbf{x} \), then by the chain rule

\[
\ddot{\mathbf{C}} = \dot{\mathbf{J}} \dot{\mathbf{x}} + \mathbf{J} \ddot{\mathbf{x}}.
\]  

(13)

But, by Equation 11

\[
\dot{\mathbf{x}} = M^{-1}(\mathbf{F}_A + \mathbf{F}_I + \mathbf{F}_C).
\]

Thus, by Equations 13 and 12 we have

\[
J M^{-1} \mathbf{F}_C = -\dot{\mathbf{J}} \dot{\mathbf{x}} - M^{-1}(\mathbf{F}_A + \mathbf{F}_I).
\]  

(14)

Since the constraints must never be violated, system velocity \( \dot{\mathbf{x}} \) must have no component in the direction of the gradient of the constraints, thus

\[
J \dot{\mathbf{x}} = 0.
\]  

(15)

Since by the principle of virtual work it is impossible that the constraint forces should do any work, we must have

\[
\mathbf{F}_C \cdot \dot{\mathbf{x}} = 0.
\]  

(16)

These two conditions can only be satisfied when the constraint forces are scalar multiples of the columns of the jacobian, i.e. each constraint force is parallel to the gradient of its corresponding constraint function. Thus we have

\[
\mathbf{F}_C = J^T \lambda,
\]  

(17)

where \( \lambda \) is a vector of lagrange multipliers. This result together with Equation 14 gives

\[
J M^{-1} J^T \lambda = -\dot{\mathbf{J}} \dot{\mathbf{x}} - J M^{-1}(\mathbf{F}_A + \mathbf{F}_I).
\]  

(18)

The matrix \( J M^{-1} J^T \) is a square matrix with number of rows equal to the number of constraints in the system. Typically, this matrix is sparse. Often it is ill conditioned, but solutions for \( \lambda \) in Equation 18 are usually easy to find using the bi-conjugate gradient approach [15]. Once the vector \( \lambda \) is determined, Equation 17 gives the constraint forces directly.
A caveat has to be introduced here about the constraint equations. Each row of the jacobian
matrix \( J \) is the gradient of one of the constraint functions, and Equation 18 is intended to keep all of
the constraints satisfied. Therefore, it is very important that constraint functions be chosen whose
gradients are non-zero when the constraints are satisfied. In more intuitive terms, the gradients of
the constraint functions determine the directions and magnitudes of the constraint forces that need
to be applied. If this gradient goes to zero when the constraint is met, it is impossible to determine
a force to keep the constraint met.

One final adjustment to Equation 18 is necessary for good practical performance. The algo-
rithm works by keeping the constraint acceleration at zero, but there is no direct control on the
constraint values themselves. Even under the best conditions, numerical drift will gradually cause
the constraints to become violated. Worse, there are certain conditions for which there may be no
solution to Equation 18. In this case, the bi-conjugate gradient approach yields a solution with
minimum mean-squared error. This behavior is desirable, but does lead to constraint violations
which the algorithm never corrects. To provide a continuous correction for small constraint viola-
tions, Witkin [19] recommends that small corrections to the \( \lambda \) vector be added proportional to the
constraint and constraint velocity vectors. This yields the final result

\[
JM^{-1}J^T \lambda = -\dot{\lambda} - JM^{-1}(F_A + F_I) - k_s C - k_d \dot{C}.
\]

(19)

Appropriate values for the scaling constants \( k_s \) and \( k_d \) must be chosen experimentally.

The problem with the constrained dynamics approach in the cloth simulation application is
speed. If \( L \) is the number of constraints, the underlying constrained dynamics algorithm requires
solving a sparse system of \( L \) linear equations with \( L \) unknowns. If the number of states and the
number of constraints are of the same order, then solving this linear system using the bi-conjugate
gradient approach is an \( O(L^2) \) operation\(^1\) [19]. Figure 6 shows clearly that the number of fixed
length constraints in an \( N \) by \( M \) cloth mesh is \( 2NM - N - M \), or approximately twice the number
of cloth particles. For even a relatively coarse mesh, such as the 51 by 51 mesh used in [5, 6], this
results in thousands of fixed length constraints that must be maintained. While systems on the
order of fifty constraints can be run at near-interactive speeds\(^2\) on a desktop workstation, a system
of 5,000 constraints would take proportionally 10,000 times as long to compute, the difference
between one-tenth of a second per iteration and seventeen minutes.

3.2 Grid subdivision

A classic approach to algorithm speed-up is the divide and conquer method. To improve computation
speed it often makes sense to try to divide the problem up into a set of smaller problems. For
example, if a problem has complexity \( O(N^2) \), and it is divided into two equal size parts, each part
will take about 1/4 the time of the original problem, so that the total solution will take about 1/2
the time. This, of course, assumes that there is no computational overhead in “knitting” the two
sub-part solutions back together.

---

\(^1\)The notation \( O(f(N)) \) is commonly used in algorithm analysis to denote an algorithm whose computational time
demand per component increases as the function \( f \) of the number of computational elements \( N \). This is usually an
asymptotic relation that holds strictly only as \( N \) approaches infinity. However it is generally a useful measure in
determining how an algorithm’s performance can be expected to degrade as the number of computational elements
increases.

\(^2\)Strictly speaking, interactive speed is considered to be at the TV frame-rate of thirty complete solutions per
second. However, rates as low as two or more solutions per second are still useful for many interactive tasks.
Thus, a simple approach to reducing computational overhead in the constrained cloth model would be to divide the constraint mesh into sub-meshes which are solved separately. This idea is diagrammed in Figure 7. It has the advantage of reducing the larger problem into (in this case) four smaller problems, labeled A, B, C and D in the figure, each of which is one-fourth the size of the original and requires one-sixteenth the computation time. If the computations in the system could be fully divided up in this way, the resulting computation would take one-quarter of the computation required to solve the system as a whole.

However, it is not this simple. The price that would have to be paid for subdividing the problem in this way would be that forces, which were previously instantaneously transmitted through the entire system, would now be instantaneously transmitted only through the subsystems. Figure 8 shows how this would affect the computation. A constraint force applied somewhere in quadrant A is handled correctly inside that quadrant, but is only communicated to quadrants B and C through displacement. If this type of stretching at the boundaries between subsystems were acceptable, then the problem would be solved. However, this is not the case with cloth simulation, where the resulting stretching would produce unwanted and unrealistic artifacts.

Worse, this stretching is more than a simple problem of aesthetics or accuracy. The constraint force which displaces the particles on the A-B quadrant boundary makes perfect sense from the point of view of quadrant A, but to quadrant B this is a “stealth force” which is neither an applied force nor an internal force and it does not figure into B’s constraint force calculation. The result is a displacement, which is correct from A’s point of view, but not a force transmission. On the next simulation time step, B will find that the A-B boundary constraints have been mysteriously violated.

What happens next depends on the weights \( k_s \) and \( k_d \) that were chosen for the “spring and damper” constraint correction terms added to Equation 19 to absorb numerical drift. If this term is not stiff enough correction will be inadequate, leading to increasing constraint violations and eventual linear-solver failure. If this term is stiff enough to apply a meaningful restorative force, this becomes a significant “stealth force” from quadrant A’s point of view, shifting the problem back onto quadrant A. Often this “stealth force” tug-of-war resolves itself by the kind of “buckling” at the boundary of the quadrants that is shown in Figure 9.

### 3.3 A Hierarchical Approach

What is needed is an improvement over the original simple approach that retains the computational speedup resulting from subdividing the problem, but which also integrates the solutions to the subproblems in a way that ensures an accurate solution to the overall problem. One way of accomplishing this would be to somehow translate a constraint subsystem into something very much like a single constraint, which could then be plugged into a constraint force calculation at a higher level.

If we partition the vector \( C \) of \( N \) constraint functions into subvectors \( C_s \), each with \( N_s \) constraint functions, then a single group constraint function

\[
C_\Sigma = \| C_s \|^2
\]

could be defined as the squared-magnitude of each of these subvectors. \( C_\Sigma \) would be zero if and only if all of the constraints are satisfied, which is one of the necessary properties of a constraint function. Unfortunately, since

\[
\nabla C_\Sigma = 2 \| C_s \| \nabla C_s
\]
this formulation results in a zero gradient when all constraints are satisfied. We have previously noted that this is an undesirable property for a constraint function.

To solve the zero gradient problem, the grouped constraint function could be reformulated as

$$C_{\Sigma} = \sum_{i=1}^{N_s} |C_{s_i}|,$$

the sum of the absolute values of each constraint in the group, which can be defined to have the gradient

$$\nabla C_{\Sigma} = \sum_{i=1}^{N_s} \begin{cases} \nabla C_{s_i}, & C_{s_i} \geq 0 \\ -\nabla C_{s_i}, & C_{s_i} < 0 \end{cases}$$

This gradient has the property of being non-zero everywhere, though it is discontinuous at zero.

The advantage of this formulation is that a single linear equation replaces a system of $N_s$ linear equations. The disadvantage of this approach is a loss of dimensionality. Whenever we replace a group of constraints by a single constraint $C_s$, we calculate only a single Lagrange multiplier $\lambda_s$ which scales the constraint forces for the entire constraint group, in contrast to the $N_s$ Lagrange multipliers which independently scaled the gradients for each individual constraint. Under many circumstances this formulation will either fail to maintain all constraints or it will not be possible to calculate constraint forces that do not violate the principle of virtual work, as given by Equation 16. Nevertheless, the notion that constraints can be combined into groups is an attractive one and worth pursuing further, since the computation time pay-off promises to be quite large.

To address the loss of dimensionality, the grouped constraint function given by Equation 20 could be modified to allow a scale factor $\rho_i$ for each of the constituent constraints $C_{s_i}$, giving

$$C_{\Sigma} = \sum_{i=1}^{N_s} |\rho_i C_{s_i}|.$$  \hspace{1cm} (21)

The inclusion of these scale factors allows the gradient of each constraint to be scaled before it is summed. If these values are well chosen, the summed constraint gradient will be appropriate for the system of constraints, i.e. some linear scaling of this gradient will produce a correct constraint force which maintains all constraints and does not violate the principle of virtual work. But how do we find these scaling values?

The correct scaling constants $\rho_i$ to use in Equation 21 are exactly the Lagrange multipliers we would obtain by solving the original constraint force calculation of Equation 19, over only the constraints and particles involved in the subgroup. This can be verified by observing that if each constraint in the system were scaled by the corresponding Lagrange multiplier, the sum of the gradients of the scaled constraints would be the constraint force.

This formulation has the drawback that any subgroup that has no internal or applied forces, or only forces applied perpendicular to the gradients of the constraints, will yield $\lambda$ values of zero. This is only a problem for constraint force transmission because, as shown in Figure 10, no constraint forces would result from these circumstances anyway. Also, this is typically a problem for only one simulation time step, since any constraint violations occurring because of the error at one step will result in internal forces at the next time step, and the constraint system will then be able to correctly transmit forces across groups.

If constraints that are likely to transmit forces are intentionally grouped together then the constraint force transmission problem virtually disappears for reasonable geometric configurations.
For the cloth model, there is a natural grouping of constraints along a single yarn (i.e. along each grid row and column), as shown in Figure 11a. This grouping will be effective except for extremely warped configurations, as in Figure 11b, which may defeat the original geometric rational for grouping constraints. However, these configurations will be rare for the cloth model, since the spring torques due to bending and trellising shown in Figure 12 will work to maintain a good force transmission geometry. For large meshes it will be desirable to subdivide rows or columns further to decrease group size and processing time.

The hierarchical constrained dynamics algorithm for the particle cloth model will work as follows:

- Group all constraints into a single group at the top of the hierarchy. If we assume a square grid of \( N \) yarn crossings (particles), then we will have \( O(N) \) constraints.

- Subdivide constraints into one group for each warp yarn and one group for each weft yarn to form the second level of the hierarchy.

- Subdivide each yarn into subgroups, and these subgroups into further subgroups until a predetermined small group size is obtained. If on each division we create a fixed number of new subgroups then the total number of subgroups will be \( O(\log N) \).

- At the lowest subgroup, use Equation 19 to calculate lagrange multipliers for each constraint in the subgroup. Since each subgroup is of a predetermined maximum size, this takes a constant amount if time per subgroup, but the number of subgroups will be \( O(N) \).

- Repeat, working up to the top subgroup, the calculation of lagrange multipliers for each of the combined constraints of Equation 21, using the lagrange multipliers determined at the next lower level as scaling constants \( \rho \). The computation time at each level is \( O(N) \).

- Calculate the constraint forces at the top of the hierarchy by multiplying the single lagrange multiplier at this level by the gradient of the single constraint at this level. There will be exactly one constraint force calculated for each original constraint, so the time is again \( O(N) \).

At each level in the hierarchy, the computational effort will be \( O(N) \), and there are \( O(\log N) \) levels, so the total computation time will be \( O(N \log N) \).

4 Discussion

We have implemented and tested the hierarchical constrained dynamics algorithm for very coarse cloth meshes of up to \( 11 \times 11 \) yarns. The implementation does not yet include the bending and trellising forces that would be exerted in real cloth, and the only forces modeled in the system dynamics are inertial, viscous and frictional. Interparticle distances are maintained by the hierarchical constraint dynamics approach outlined above. We also use constraint dynamics to solve the collision-detection and non-intersection problem between surrounding geometry and the cloth model. No effort has been made to include avoidance of cloth self-intersection. At this point, the model is more like one of a very loose chain-mail than of real cloth. Nevertheless, by the elimination of all but the bare-essentials from the model we have been able to run a large number of fast tests to allow us to perfect the simulation approach. Figure 13 shows one time step in a simulation of the cloth patch falling over some simple geometry consisting of two spheres and a cube. The
entire sequence of the cloth falling over the geometry, and then eventually sliding off to fall free, is computed and displayed in under a minute on a Silicon Graphics R4000 Indigo Elan.

The algorithm, as currently implemented, does not take full advantage of the speedups that are possible. There is no attempt to efficiently exploit sparseness. Also, in the hierarchical algorithm it is logical to subdivide the problem to a fine enough level that the biconjugate gradient solver, which is appropriate for large linear systems, is no longer needed. For the small fixed size problems to be solved in the hierarchical algorithm, it would be much more efficient to use a direct algebraic solution. However, even though none of these efficiencies have been exploited in the current implementation, we have run side-by-side comparisons with the conventional constrained dynamics algorithm and noted significant improvements in performance. Improvements are most apparent for larger problems.

Future work will involve expanding the cloth model by adding bending and trellising forces, tuned to match Kawabata test data, and reimplementing the hierarchical algorithm with fixed size direct algebraic linear system solvers. We are planning to conduct large-scale tests with varying constraint group sizes and to compare performance and accuracy with the non-hierarchical constrained dynamics system. We will be seeking to show that 1) the constrained dynamic approach will allow accurate calculations of cloth dynamics, 2) that the hierarchical approach to solving the constrained dynamics problem will allow the solution to be computed rapidly enough that it can be considered as a viable tool for interactive modeling, and 3) that the physical accuracy of the particle-based drape model can be maintained in the new fast dynamic model. When the completed model is perfected, we will begin experiments with constructing clothing and explore uses of the model in dressing animated characters.

5 Conclusion

The concept of constrained dynamics simulation has a very natural application to cloth simulation using the interacting particle dynamic model. We feel that the new hierarchical algorithm for constrained dynamics described here is particularly appropriate for simulating the type of mechanical system represented by cloth. This is because the geometry of force-transmission can be ordered along warp and weft yarns so that good guesses can be made as to how to subdivide the system into force-transmitting constraint groups. Initial tests confirm this supposition. Further validation of the approach awaits complete implementation and testing to make the necessary accuracy versus speed comparisons with more conventional approaches.

Acknowledgements

Figure 1 was produced by Gene Greger and David Breen. This work was partially supported by DLA contract No. DLA900-87-D-0016, NSF grant No. CDR-8818826, and by the Visualization Laboratory at Texas A&M University.

References


**Figures**
Figure 1: Draping cloth objects
The complex draping configurations of the cloths over the chair and table were obtained using the particle cloth model. Reprinted with permission from [5]
Figure 2: Actual (left) vs. simulated (right) cotton and wool drapes
Energy functions in the particle cloth model can be tuned to reproduce the draping behavior of specific cloth types.
Figure 3: Cloth model energy functions
Relationship between particle displacements and the stretching, bending and trellising energy functions in the particle cloth model. Reprinted with permission from [5].
Figure 4: Kawabata Bending and Shear Plots
Plots from the Kawabata testing system, plotting applied load on the vertical axis versus geometric deformation on the horizontal axis. Reprinted with permission from [5].
Figure 5: Chains Modeled Numerically by Spring Forces and Constraint Forces
a) A force applied to one end of a numerically-modeled dynamic chain connected by springs is transmitted via displacements over several time steps. In the first iteration after a force is applied to A, no restoring force exists to keep distance A-B from increasing. On the next iteration, the length change produces a restoring force which is applied to particles A and B and which also results in length B-C increasing, etc.

b) In the constraint chain, a force applied to particle A instantaneously produces constraint forces that are transmitted along the entire chain, maintaining each length constraint exactly. No additional energy is added because constraint forces can do no work.

Figure 6: Interparticle Length Constraints vs. Number of Particles
There are $M(N-1)$ constraints in $M$ rows of $N$ particles. Likewise, there are $N(M-1)$ constraints in the $N$ columns, for a total of $2NM - N - M$ constraints.
Figure 7: Subdividing a Grid of Particles into Quadrants
A rectangular mesh divided into smaller rectangular quadrants designated A, B, C and D. These quadrants correspond to four separate constraint groups.

Figure 8: Simple Subdivision Isolates Force Effects to Quadrants
A force applied in quadrant A is handled appropriately within the quadrant, but results in length constraint violations at the boundaries with B and C. This stretching is illustrated in the difference in length between $L_1$, the original length of a boundary constraint, and $L_2$, the new stretched length.
Figure 9: Effect of Uniform Force Along One Edge
A force applied uniformly to the left side of the mesh is handled appropriately in quadrants A and C but is not transmitted to quadrants B and D. The resulting “tug of war” between boundary constraints is resolved by pushing the boundary particles up out of the plane of the mesh.

Figure 10: Applied Forces vs. Constraint Forces
a) A particle is constrained to lie at a fixed distance between two nails. If the nails and particle are colinear, and a force is applied perpendicular to this line, no constraint force will result. This is because the constraint force is a linear scaling of a gradient, and in this case the projection of the applied force onto either gradient is zero.

b) The particle and the nails are no longer exactly colinear. As a result, the projection of the applied force on to the two gradients is non-zero, and a constraint force can be applied to exactly counter the applied force.
Figure 11: Grouping Constraints Along Warp and Weft Directions
a) Grouping constraints along the warp or weft (row or column) directions makes sense because constraint forces are transmitted along a linked chain to the degree to which the links are parallel. Thus, a force on link A should immediately result in a force on link D.
b) If the geometric configuration becomes too warped, the original rationale for grouping constraints will be defeated. Under these circumstances it might make computational sense (but not physical sense) to regroup the constraints as A-B-2-1 and D-C-3-4.

Figure 12: Bending and Trellising Torques Maintain Reasonable Yarn Alignments
The bending and trellising springs produce torques that work to maintain a quasilinear arrangement of particles in a single row or column. These forces help maintain “sane” configurations for constraint grouping.
Figure 13: An Intermediate Step in a Test Simulation Using Hierarchical Constrained Dynamics
An 11 × 11 simulated cloth mesh falling over some simple geometric elements. Forces maintaining
interparticle distances and preventing cloth-geometry interpenetration are all calculated by the
heirarchical constraint dynamics algorithm.