Cloth Dynamics Simulation: A Masters Thesis

Richard W. DeVaul
G7 Masters Student
Visualization Science Program
College of Architecture, Texas A&M University

September 3, 1997
Revision: 1.20 α

Abstract

This thesis describes a new technique for simulating the dynamic behavior of cloth for animation applications. This technique combines a forward dynamics extension of the Breen–House draping model with a new iterative technique for constrained dynamics simuiation. The result is a cloth model that is empirically based, fast, accurate, and stable.

The cloth dynamics model described here is capable of reproducing the behavior of virtually any woven material under light loads (typical of clothing) where thread stretching is not a significant factor. There are no *a priori* restrictions on the type or number of external forces acting on the cloth. This makes the model suitable for a wide range of applications, including clothing for animated figures, flags and sails, drapes, *etc.*.

Fundamental to the implementation of the model is a new algorithm for lagrange-multiplier constrained dynamics simulation. This algorithm is an iterative technique by which cyclic constraint systems may be solved in linear time with respect to the number of constraints.

Contents

List of Figures

Chapter 1

Introduction

1.1 The Material

Cloth is a material with unique properties, with uses ranging from clothing to high-tech composite materials. Cloth is also very old, dating back at least 8,000 years. These reasons alone make cloth worth studying, but few people outside the narrow fields of textile engineering and fabric science have scientifically investigated its properties, and most of this work has been done only in the last sixty years [4].

More recently, cloth has come to the attention of computer graphics researchers who are interested in simulating its look and behavior for computer animation applications. This research dates back a decade, but the problem has become more important recently because of the expanded use of three dimensional computer animation in the movie effects industry. Animators are producing increasingly realistic virtual actors, but currently lack the tools to dress them in realistic virtual clothing. This lack of tools is largely the result of the lack of a suitable cloth dynamics model.

1.2 The Model

To be useful, a cloth dynamics model for three dimensional computer animation applications must have the following properties:

1. High performance. The model must be implemented on existing desktop hardware with acceptable performance for animation production applications.

2. Ease of use. The model must be usable by non-fabric scientists.

3. Visual accuracy. The model should produce simulated cloth behavior with the look of real cloth.

4. Flexibility. The model must be capable of reproducing the behavior of a wide range of fabrics, and be able to reproduce the distinctive characteristics of each.

5. Stability. The model should lend itself to robust, numerically stable implementations.

There are a number of special problems associated with cloth dynamics that make this set of goals difficult to achieve.

One of the difficulties of cloth dynamics modeling arises because cloth is unique and common at the same time. Cloth is a material with unique properties — nothing else moves or drapes like it — and at the same time is a common part of our environment. Cloth is so distinctive and animation viewers are so familiar with it that even small fidelity problems in simulated cloth stand out.

Conventional flexible-object models fail to convincingly reproduce cloth behavior. In particular, continuum models, which are based on an underlying assumption of smooth elastic properties, have a rubbery look when applied to cloth modeling [6, 5]. Most previous attempts at developing a workable cloth dynamics model have relied on continuum techniques.

The essential problem of modeling cloth behavior is that cloth is not a substance like rubber or plastic; cloth is more accurately thought of as a mechanism, and its distinctive behavior results from its underlying *mechanical* structure. It is the mechanical interaction of the yarns and threads which give cloth its unique properties. This interaction occurs at a scale approaching that of a typical simulation element [6, 5].

The challenge of developing a cloth dynamics model, therefore, is to create a model which is a close analog to the structure of real cloth. This approach addresses the accuracy and flexibility problems by allowing a direct correspondence of model parameters to empirically measured cloth properties. Such a model has already been developed to predict the static drape of cloth: the Breen–House interacting particle model [6]. This thesis proposes a forward-dynamics extension of the Breen–House model as a solution to the problem of developing a cloth dynamics model suitable for animation applications.

Chapter 2

Previous Work

There are two bodies of research on cloth modeling: textile engineering and cloth modeling for computer graphics. In general, there has been relatively little overlap between the two. Textile engineers are concerned with cloth primarily as an engineering material, whereas computer graphics researchers are concerned primarily with aesthetics. The first group tends to view cloth at the micro-mechanical level, whereas the second is more concerned with the material's macro-scale behavior and overall "look" [4]. The focus of research in this thesis is the modeling of cloth for computer graphics applications, although in principle these results could also be used in engineering applications.

Rather than attempt a comprehensive review of every cloth model published, this review focuses on important representatives of the various research avenues explored since the beginning of research in this area in the mid eighties. For a more comprehensive review of cloth modeling techniques the reader is referred to [12].

2.1 Early Work — Weil

Papers on cloth modeling for computer graphics go back over a decade. The first important computer graphics model was developed by Weil in 1986 [15]. Weil describes a purely geometric method for mimicking the drape of fabric suspended at constraint points. There are two phases in the simulation process.

In the first phase, the cloth surface is geometrically approximated with catenary curves. A catenary is the curve an ideal thread naturally follows

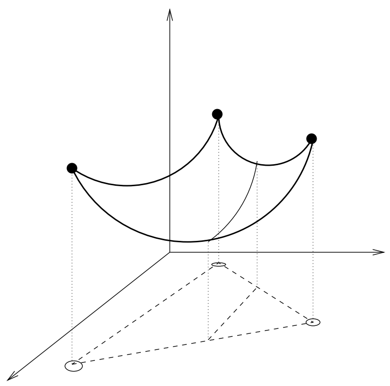

Figure 2.1: A triangle of catenary curves, subdivided once

when suspended by two points, which in two dimensions is given by

$$y = \frac{a}{2}\left(e^{x/a} + e^{-x/a}\right) = a\cosh(x/a). \tag{2.1}$$

The first step is to construct catenary curves between pairs of constraint points. In the case of crossing curves, the upper curve is eliminated. This results in triangles of constraint points connected by catenary curves. Then the triangles are further subdivided using catenaries (always choosing the highest of the three possible) until a desired resolution is reached (Figure 2.1.) In the second phase, an iterative relaxation process is used to smooth the surface interpolated from the original catenary intersection points. The higher the number of iterations, the smoother the surface.

Although this is a static drape model, it is possible to produce motion by applying the full approximation and relaxation process once, and then successively moving the constraint points slightly and re-applying the relaxation phase.

The surfaces produced by this technique have a draped, wrinkled look similar to cloth. However, other than the choice of catenary curves as a geometric representation this model makes no reference to the physical properties of cloth and can not produce true dynamics.

2.2 Continuum Techniques

Continuum models employ continuum mechanics to model cloth as an elastic sheet. Simulations are run using finite element or finite difference techniques. The first work in this area is a masters thesis by Feynman in 1986 [9]. A more

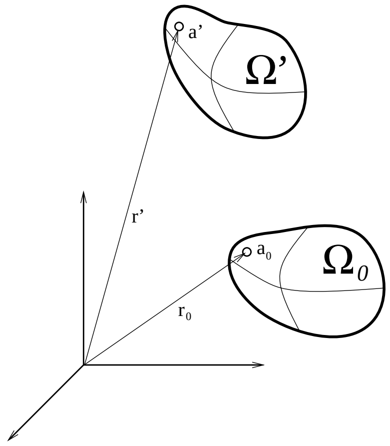

Figure 2.2: Deformation of an object resulting in a change in energy

general system for elastic modeling with cloth applications was developed by Terzopoulos *et al.* in 1987 [13].

2.2.1 Terzopoulos *et al.*

In [13] Terzopoulos *et al.* describe a general technique for modeling the dynamics of elastic objects using Lagrange's equations of motion. Referring to Figure 2.2, the behavoir of flexible body Ω is given by

$$\frac{\partial}{\partial t}\left(\mu \frac{\partial r}{\partial t}\right) + \gamma \frac{\partial r}{\partial t} + \frac{\delta \varepsilon(r)}{\delta r} = f(r,t), \qquad (2.2)$$

where $r(a,t)$ is the position of point a at time t, $\mu(a)$ is the mass density of the body at a, $\gamma(a)$ is the damping density at a, $f(r,t)$ represents the net externally applied forces, and $\varepsilon(r)$ is a *functional* which measures the net instantaneous potential energy of the elastic deformation of the body.

The first term in Equation 2.2 represents the inertial forces, the second term represents damping, and the third term is the elastic force due to deformation. In order to work with Equation 2.2 it is necessary to discretize the elastic force term using finite element or finite difference approximation methods.

Central to this technique is the assumption that object Ω can be accurately modeled using continuum mechanics. This is true for objects made of plastic, rubber, and a wide variety of other materials in which structure is visible only at a microscopic or molecular scale. Cloth is structured at a scale visible to the naked eye, which makes continuum techniques less appropriate. Cloth modeled using continuum techniques tends to look more like rubber or plastic, without the complex buckling and folding behavior of real cloth [5, 6, 10].

2.2.2 The Thalmanns *et al.*

The Thalmann group [7] has expanded on the work of Terzopoulos *et al.*to develop an elastic deformation model specifically for cloth modeling. The focus of this research has been on building a practical animation tool. In [7] the dynamic model is formulated as

$$\rho(a)\frac{d^2r}{dt^2} + \frac{\delta}{\delta r}\iint_\Omega \|E\|^2 da_1\, da_2 + \frac{\delta}{\delta v}\iint_\Omega \|\dot{E}\|^2 da_1\, da_2 + \frac{\delta}{\delta r}\iint_\Omega \|B - B_0\|^2 da_1\, da_2 = \sum F_{ex}, \tag{2.3}$$

where $E = G - G^0$ is the Lagrangian strain tensor and $\dot{E}$ is the first time derivative of E and is defined as

$$\dot{E}_{ij}(r(a)) = \frac{d}{dt}E_{ij} = \frac{1}{2}\dot{G}_{ij} = \frac{\partial r}{\partial a_i}\cdot\frac{\partial v}{\partial a_j} + \frac{\partial r}{\partial a_j}\cdot\frac{\partial v}{\partial a_i}. \tag{2.4}$$

Equations 2.3 and 2.4 are a reformulation of the Terzopoulos dynamics model (Equation 2.2), the only substantive difference being in the way damping is handled. In Equation 2.2 all motion, including rigid body motion, is subject to damping. For example, a rigid body accelerating in a vacuum without deformation would incorrectly dissipate energy, resulting in drag. In Equation 2.3 only surface deformations are damped.

Terzopoulos suggests creating a potential field around each model element as a means of producing collision response [13]. In cloth this can result in inappropriately springy behavior if the potential field is strong. To avoid this problem the principle of conservation of momentum is applied to collisions between model elements, resulting in perfectly inelastic collisions [7, 14].

In [14] Volino *et al.* focus on the special problems associated with collision detection and response in cloth and deformable surfaces. This includes the problem of self-intersection and collision consistency. Self-intersection calculation is made tractable through the use of a hierarchical algorithm utilizing surface curvature. Calculation times resulting from the use of this algorithm are said to be roughly proportional to the number of colliding elements and independent of the total number of elements in the surface.

Collision consistency, which is the problem of keeping all the elements of a complex surface on the appropriate side of another complex surface during collision response, is handled with heuristics and a history mechanism.

The Thalmann group has also dealt with the practical issues involved in stitching flat panels into three dimensional clothes and reducing the number and complexity of the model parameters presented to the user [7].

2.3 Non-continuum Models

Non-continuum models of cloth are based on the observation that cloth is really a mechanism rather than a substance, and derives its macro-scale dynamic properties from the micro-mechanical interaction between threads [5, 6]. Non-continuum models have arisen in response to the weakness of continuum mechanics in capturing the complex buckling and folding behavior of cloth. Non-continuum models lie at the intersection of computer graphics and fabric science research goals.

2.3.1 Breen–House Model

Breen House Wozny

In [5] and [6] Breen *et al.* describe an interacting particle model[1] for cloth drape which explicitly represents the micro-mechanical structure of cloth as an interacting particle system. Each intersection of a warp and a weft thread is represented by a particle, which interacts with adjacent particles and the environment through mechanical connections represented by energy functions. A stochastic gradient descent technique is used to relax the cloth particles towards a final equilibrium position, producing fabric drape.

This model accurately reproduces the drape of specific materials [6]. However, the model is non-dynamic[2] and in its original implementation was very slow to compute.

House DeVaul Breen

In [10] House *et al.* propose a forward-dynamics translation of the Breen–House model using a Newtonian dynamics formulation. They differentiate the energy functions of the draping model to produce position dependent

[1] For a more detailed explanation of this model, see Chapter 3.2.

[2] It is possible that pseudo-dynamic motion could be created with this model in a process similar to that proposed by Weil in [15].

forward dynamics forces, or "cloth springs." However, differentiating the stretching and repelling energy functions results in very stiff springs, which are numerically unstable.

To avoid numerical instability, they replace the stiff length-maintaining springs with Lagrange-multiplier (non-reduced coordinate formulation) fixed length constraints. The use of constraints in place of stiff springs trades the problem of numerical stability for the seemingly harder problem of efficient constraint handling. The conventional Lagrange-multiplier constraint solution algorithm is $O(n^3)$ with respect to the number of constraints[2], requiring the solution of a linear system in the form of

$$M\mathbf{x} = \mathbf{b}, \tag{2.5}$$

where M is a square matrix which may not be sparse or invertible[3]. To make a constraint-based model feasible, House *et al.* propose a hierarchical algorithm for constraint solution which reduces the order of the problem to $O(n\log(n))$.

The primary problem addressed in [10] is the efficient handling of Lagrange-multiplier constraint meshes; the bending and trellising "cloth springs" were not implemented in the test system. Despite this limitation, the performance data suggest that the use of fixed length constraints may be a feasible alternative to stiff length-maintaining springs.

2.3.2 Eberhardt Weber Strasser

A forward dynamics translation of the Breen–House model using a Lagrangian dynamics formulation is presented in [8]. The Lagrange equation for particle i is given as

$$\frac{d}{dt}\frac{\partial L}{\partial \mathbf{v}_i} = \frac{\partial L}{\partial \mathbf{x}_i}, \tag{2.6}$$

where $\mathbf{x}_i$ and $\mathbf{v}_i$ represent the position and velocity of particle i respectively. The equation for L is given as

$$L = \sum_{i=0}^{n-1} E_{kin_i} - \left(\sum_{i=0}^{n-1} E_{pot_i} + \sum_{i=0}^{n-1} E_{t_i} + \sum_{i=0}^{n-1} E_{s_i} + \sum_{i=0}^{n-1} E_{b_i} \right), \tag{2.7}$$

where E_{kin_i} is the kinetic energy of particle i, E_{pot_i} is the energy of particle i due to gravity, E_{t_i} corresponds to $U_{repel_i} + U_{stretch_i}$ in the Breen–House model, E_{s_i} is the energy of trellising, and E_{b_i} is the energy of bending. This

[3] See Chapter 4 for more information on dynamic constraint algorithms.

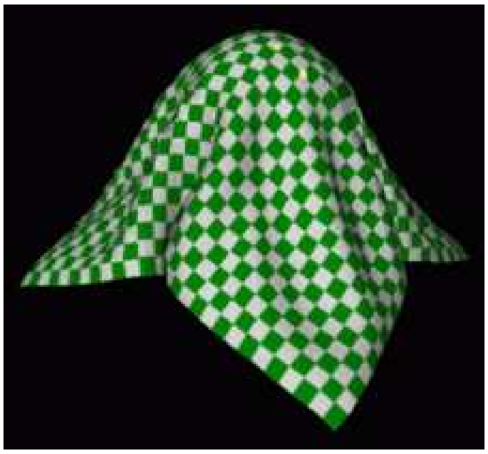

Figure 2.3: A still frame produced by the Eberhardt *et al.* system; Cloth draping over a sphere.

results in a system of ordinary differential equations, one for each coordinate of each particle. The authors experimented with using adaptive Runge-Kutta and Bulirsch-Stoer as numerical integrators.

Eberhardt *et al.* are successful in producing true dynamic behavior with their model. However, for draping simulations using a 52×52 grid the authors report numerical integrator time steps in the range of 0.1 to 6 milliseconds, and an average calculation time of around 25 minutes for 40 ms of animation. While faster than the reported times for the original Breen *et al.* iimplementation, the very small time steps taken in the integrator show the difficulties of working with the stiff springs resulting from the E_{t_i} term.

Chapter 3

Cloth Dynamics Modeling

3.1 Introduction

This chapter describes a new, empirically-based, forward dynamics cloth model, hereafter to be referred to as the CDM (cloth dynamics model). A basic understanding of physically based modeling and forward dynamics simulation is assumed; for an introduction to forward dynamics simulation and Lagrange multiplier constraint systems the reader is referred to the SIGGRAPH95 Physically Based Modeling course notes [17]. This subject is also discussed in Chapter 4.

In describing the CDM a distinction is made between the cloth model and the forward dynamics simulation in which the model operates. The distinction is important because the CDM is intended to be a modular component which can be dropped into any force-based dynamics simulation. For this reason no assumptions are made about the architecture of the simulation system beyond the foundations of force-based forward dynamics. An example implementation of the CDM and forward dynamics simulation system is discussed in Chapter **??**.

The CDM is a forward dynamics extension of the Breen–House draping model [5, 6]. This approach was first proposed by House *et al.* in [10], and is similar to that described by Eberhardt *et al.* in [8], although it differs from the Eberhardt approach in two ways:

1. The CDM uses a force-based Newtonian dynamics formulation rather than a potential-based Lagrangian formulation.

2. The CDM uses length constraints in place of the stiff springs resulting from differentiating the $U_{repel_i} + U_{stretch_i}$ terms of Equation 3.1.

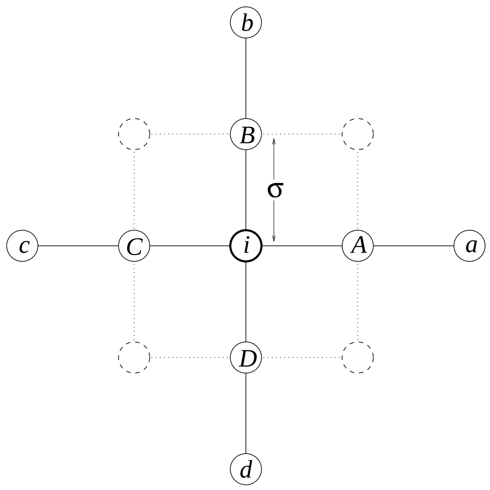

Figure 3.1: Particle i and its eight nearest thread-line neighbors

The choice of the Newtonian formulation versus Lagrangian is not important since the two are equivalent in this application. The use of fixed length constraints in the model and the development of new methods for solving such a constraint system are the primary focus of this research.

3.2 The Breen–House Model

The Breen–House draping model is the basis of the CDM. As discussed in Chapter 2.3.1, This model is based on a grid of discrete mass points representing the intersection of a warp and weft thread. The energy of each particle in this grid depends on its position. The total energy for particle i is given by

$$U_i = U_{repel_i} + U_{stretch_i} + U_{bend_i} + U_{trellis_i} + U_{grav_i}. \tag{3.1}$$

The first four functions are functions of particle i's position relative to its eight thread-line neighbors. In Figure 3.1 the four particles immediately adjacent to i are labeled A, B, C, and D, and those at one remove are labeled a, b, c, and d. Note that because the Breen–House model is a non-dynamic formulation there is no term for kinetic energy or air resistance in Equation 3.1.

3.2.1 Repulsion and Stretching

Breen *et al.* [5, 6] observe that cloth does not significantly stretch under its own weight when draping. Therefore, the combined energy function $U_{repel_i} + U_{stretch_i}$ must tightly constrain adjacent particles to a specified inter-particle distance of σ.

U_{repel_i} and $U_{stretch_i}$ are based on non-empirically chosen functions $R(r_{ij})$ and $S(r_{ij})$, which are fifth degree polynomials of the form

$$R(r_{ij}) = \begin{cases} C_0 \frac{(\sigma - r_{ij})^5}{r_{ij}} & r_{ij} \leq \sigma \\ 0 & r_{ij} > \sigma \end{cases} \tag{3.2}$$

and

$$S(r_{ij}) = \begin{cases} 0 & r_{ij} \leq \sigma \\ C_0 \left(\frac{\sigma - r_{ij}}{r_{ij}}\right)^5 & r_{ij} > \sigma \end{cases}, \tag{3.3}$$

where r_{ij} is the distance between particles i and j, and C_0 is a scale factor. Together, $R(r_{ij}) + S(r_{ij})$ produce a steep energy well that maintains a distance of σ between particles i and j.

Repulsion

The repulsion function U_{repel_i} is a function of particle i's distance from every particle in the model. The primary purpose of U_{repel_i} is to maintain a minimum distance σ between particle i and particles A, B, C, and D (Figure 3.1). Because U_{repel_i} is summed over every particle in the model, it also acts as a collision response field which helps to prevent self-intersection of the cloth in a manner similar to the repulsive field used in the Terzopoulos model [13]. The equation for U_{repel_i} is

$$U_{repel_i} = \sum_{j \neq i} R(r_{ij}). \tag{3.4}$$

Stretching

The stretching function $U_{stretch_i}$ is a function of the inter-particle distance between i and particles A, B, C, and D (Figure 3.1). The purpose of $U_{stretch_i}$ is to maintain a maximum distance of separation σ between i and each of its four linked neighbors. The equation for $U_{stretch_i}$ is

$$U_{stretch_i} = \sum_{j \in N_i} S(r_{ij}), \tag{3.5}$$

where $N_i = \{A, B, C, D\}$ is the set of particle i's four connected neighbors.

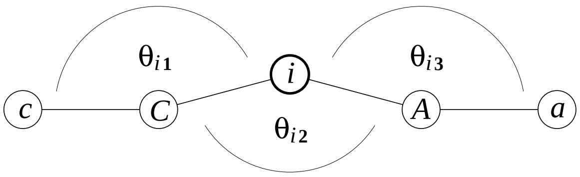

Figure 3.2: Bending energy for a single thread at particle i is a function of angles θ_{i1}, θ_{i2}, and θ_{i3}.

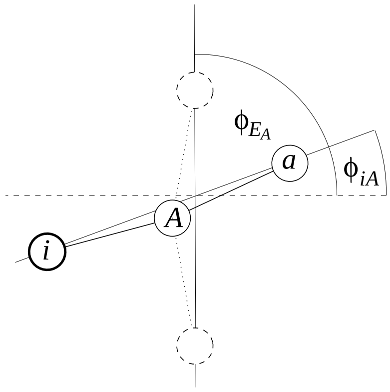

Figure 3.3: One of the four trellising angles ϕ_{ij} which contribute to the trellising energy of particle i.

3.2.2 Bending and Trellising

Unlike the stretching properties of cloth, Breen *et al.* assume that the out-of-plane bending and trellising are significant contributors to cloth's behavior.

Bending

Out-of-plane bending occurs when a thread is bent out of the plane of the cloth, as opposed to trellising, which is bending that occurs in plane. In the Breen–House model, the bending energy of particle i is formulated as

$$U_{bend_i} = \sum_{j \in M_i} B(\theta_{ij}), \tag{3.6}$$

where M_i is the set of six bending angles θ_{ij} produced by the two threads intersecting at i (Figure 3.2) and $B(\theta_{ij})$ is an empirically based bending energy function. The redundancy in this formulation allows U_{bend_i} to reflect the total change in bending energy due to the change in position of particle

i. Each U_{bend_i} is scaled appropriately to eliminate the duplication of bending energy resulting from the sharing of bending angles between adjacent particles on a thread line.

Trellising

Trellising occurs when cloth is placed under shearing stress. In the Breen–House model, the trellising energy of particle i is formulated as

$$U_{trellis_i} = \sum_{j \in K_i} T(\phi_{ij}), \tag{3.7}$$

where K_i is the set of i's four connected neighbors A, B, C, and D, and $T(\phi_{ij})$ is an empirically based trellising energy function. To calculate ϕ_{ij}, two line segments are constructed between the four connected neighbors of particle j (one of which is particle i). The angle ϕ_{ij} is the unsigned difference between the crossing angle of the two segments and an equilibrium angle, ϕ_{E_j} (Figure 3.3). Typically $\phi_{E_j} = 90°$, but slippage resulting from load can be modeled by allowing ϕ_{E_j} to change over time. As in the bending calculation, the redundancy of this formulation allows the total change in energy resulting from a change in the position of particle i to be accounted for in $U_{trellis_i}$.

3.3 The Dynamic Model

3.3.1 Newtonian Dynamics

The Acceleration of a Single Body

The state of an unconstrained rigid body i with N_i degrees of freedom[1] is described by a position vector $\mathbf{x}_i \in \mathbf{R}^{N_i}$ and the two time derivatives of this vector $\dot{\mathbf{x}}_i$ and $\ddot{\mathbf{x}}_i$. The mass properties of this body are described by M_i, which is a symmetric positive definite matrix of dimension $N_i \times N_i$. The external force acting on body i is represented by $F_i \in \mathbf{R}^{N_i}$.

The acceleration of an unconstrained rigid body i is described by Newton's second law of motion as:

$$M_i \ddot{\mathbf{x}}_i = \mathbf{F}_i, \tag{3.8}$$

where $\ddot{\mathbf{x}}_i$ is the second time derivative of i's position.

[1] Unconstrained particles have three degrees of freedom. All other unconstrained rigid bodies have six.

The Acceleration of a Dynamic System

The state of a system of unconstrained rigid bodies can be described by a position vector $\mathbf{x}$ and the two time derivatives $\dot{\mathbf{x}}$ and $\ddot{\mathbf{x}}$. The position vector has the form

$$\mathbf{x} = \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_n \end{pmatrix}, \tag{3.9}$$

where each element $\mathbf{x}_i \in \mathbf{R}^{N_i}$. The forces acting on such a system are represented by

$$\mathbf{F} = \begin{pmatrix} \mathbf{F}_1 \\ \mathbf{F}_2 \\ \vdots \\ \mathbf{F}_n \end{pmatrix}, \tag{3.10}$$

where each element $\mathbf{F}_i \in \mathbf{R}^{N_i}$. The inertial properties of the system may be described by the block diagonal matrix

$$M = \begin{pmatrix} M_1 & 0 & & \\ 0 & M_2 & 0 & \\ & 0 & \ddots & \\ & & & M_n \end{pmatrix}, \tag{3.11}$$

where each element M_i is a symmetric positive definite matrix of dimension $N_i \times N_i$. The acceleration of a system of unconstrained rigid bodies can be stated as

$$M\ddot{\mathbf{x}} = \mathbf{F}, \tag{3.12}$$

where $\ddot{\mathbf{x}}$ is the second time derivative of the position vector.

3.3.2 Extending the Breen–House model

Cloth Springs

To extend the Breen–House model into a Newtonian forward dynamics formulation, it is necessary to convert the energy functions into forward dynamics forces. Were numerical stability not an issue, translating these energy functions into forces would simply be a matter of differentiating the first

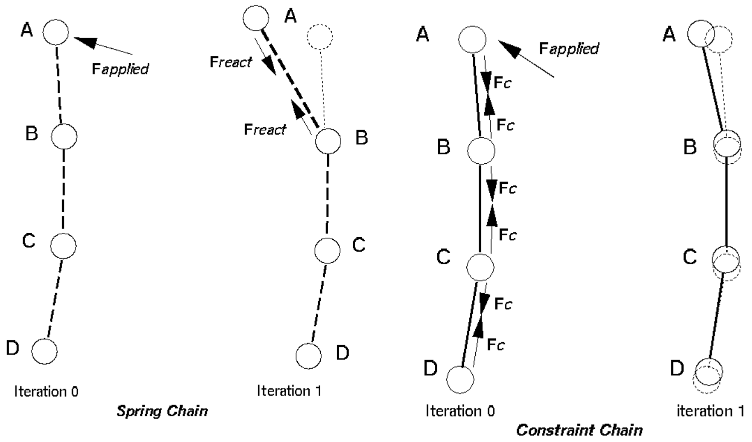

Figure 3.4: A spring chain vs. a constraint chain.

four terms[2] of Equation 3.1 with respect to the position vector:

$$\mathbf{F}_{tension_i} = \frac{\partial U_{repel_i}}{\partial \mathbf{x}_i} + \frac{\partial U_{stretch_i}}{\partial \mathbf{x}_i} \tag{3.13}$$

$$\mathbf{F}_{bend_i} = \frac{\partial U_{bend_i}}{\partial \mathbf{x}_i} \tag{3.14}$$

$$\mathbf{F}_{trellis_i} = \frac{\partial U_{trellis_i}}{\partial \mathbf{x}_i}. \tag{3.15}$$

This results in a set of non-linear spring forces which are a function of particle position, or "cloth springs." Unfortunately, the $\mathbf{F}_{tension_i}$ springs are several orders of magnitude stiffer than the other cloth springs, causing numerical integration problems.

The stiffness of the $\mathbf{F}_{tension_i}$ spring forces arises from the primary purpose of the $U_{repel_i} + U_{stretch_i}$ terms in Equation 3.1, which is to maintain the distance of separation σ between adjacent connected particles. The energy well created by the fifth degree polynomials $R(r_{ij})$ and $S(r_{ij})$ work well enough, but their choice is arbitrary. In principle, the $\mathbf{F}_{tension_i}$ springs should be arbitrarily stiff to resist any change in length.

[2]The fifth term, U_{grav_i}, properly belongs to the forward dynamics "universe" rather than the cloth model; the model could be used to simulate cloth behavior in a weightless environment.

Lagrange Multiplier Fixed Length Constraints

Rather than working with very stiff springs, the length-maintaining springs can be replaced with Lagrange multiplier fixed length constraints[3]. Unlike springs which apply a force proportional to length change (*i.e.* constraint violation) fixed length constraints produce constraint-maintaining forces in direct response to applied forces and momentum. This means that unlike springs, fixed length constraints maintain a specified distance constraint exactly preventing the numerical instability problems associated with high spring constants. Figure 3.4 illustrates the difference between springs and constraints.

3.3.3 CDM Dynamics

The dynamics of the CDM may be described in Newtonian terms by

$$M\ddot{\mathbf{x}} = \mathbf{F}_E + \mathbf{F}_I + \mathbf{F}_C, \tag{3.16}$$

where M is the diagonal mass matrix, $\ddot{\mathbf{x}}$ is the second time derivative of the position vector $\mathbf{x}$, $\mathbf{F}_E$ is the sum of all forces originating outside the CDM (external forces), $\mathbf{F}_I$ is the sum of all non-constraint applied forces within the CDM (internal forces), and $\mathbf{F}_C$ is the constraint force vector.

External Forces

External forces, $\mathbf{F}_E$, include any applied force not arising from the CDM's internal dynamics or constraint forces. Gravity, collision with other objects, and aerodynamic drag are all external forces. Any number of external forces may be involved in a given simulation. A typical formulation for external forces is

$$\mathbf{F}_E = \mathbf{F}_{gravity} + \mathbf{F}_{air} + \mathbf{F}_{collide}, \tag{3.17}$$

where $\mathbf{F}_{gravity}$ is the force of gravity, $\mathbf{F}_{air}$ is the force of aerodynamic drag acting on the cloth, and $\mathbf{F}_{collide}$ is the set of impulse and contact forces resulting from interactions with other objects.

Aerodynamic forces, represented by $\mathbf{F}_{air}$ in Equation 3.17, may play a large roll in the behavior of a cloth simulation. Eberhardt *et al.* suggest the use of

$$\mathbf{F}_{air} = \frac{\rho}{2}\sum_{i=1}^{n} -\frac{\dot{\mathbf{x}}_i}{\|\dot{\mathbf{x}}_i\|}\left(c_{w_i} A_i \dot{\mathbf{x}}_i^2\right), \tag{3.18}$$

[3]Lagrange multiplier constraints are discussed in Chapter 4.

where ρ is the specific weight of air, c_{w_i} is the resistance coefficient at particle i, and A_i is the area of the cloth patch represented by particle i perpendicular to the velocity $\dot{\mathbf{x}}_i$. Note that c_{w_i} depends on the position of every particle in the cloth, but can be reasonably approximated using a small number of geometric cases [8]. See Chapter 6.1.3 for a discussion on implementing external forces.

Internal Forces

Internal forces are the set of all applied forces resulting from the internal dynamics of the CDM, not including constraint forces. A general expression for internal forces is

$$\mathbf{F}_I = f(\mathbf{x}, \dot{\mathbf{x}}, t), \tag{3.19}$$

where t represents time and $f(\mathbf{x}, \dot{\mathbf{x}}, t)$ is some function describing the system's internal dynamics. In the case of the CDM, Equation 3.19 can be restated as

$$\mathbf{F}_I = \sum_{i=1}^{n} \mathbf{F}_{bend_i} + \sum_{i=1}^{n} \mathbf{F}_{trellis_i} + f_{friction}(\mathbf{x}, \dot{\mathbf{x}}) + f_{collide}(\mathbf{x}, \dot{\mathbf{x}}). \tag{3.20}$$

The function $f_{friction}(\mathbf{x}, \dot{\mathbf{x}})$ provides internal damping, which is necessary to avoid excessively springy behavior. In real cloth this damping results from the frictional interaction of the fibers. The function $f_{collide}(\mathbf{x}, \dot{\mathbf{x}})$ produces forces which prevent the cloth from intersecting itself. See Chapter 6.1.3 for a discussion of how $f_{friction}(\mathbf{x}, \dot{\mathbf{x}})$ and $f_{collide}(\mathbf{x}, \dot{\mathbf{x}})$ may be implemented.

Fixed Length Constraints

In the CDM, fixed length constraints are used to maintain a specified inter-particle distance of σ. See Chapter 4 for an explanation of fixed length constraints and the calculation of $\mathbf{F}_C$.

Chapter 4

Dynamic Constraints

4.1 Introduction

Dynamic constraints are a means of describing a dynamic system in terms of constraints placed on the motion of its elements. This is a very natural way of representing a wide range of interesting dynamic systems, including articulated figures, mechanical linkages, systems involving contact forces and sliding, *etc.* There are two representations for dynamic constraints in common usage: reduced coordinate formulations and Lagrange multiplier formulations. This chapter describes these techniques with emphasis on Lagrange multiplier constriants.

4.1.1 Reduced Coordinate Formulations

In reduced coordinate formulations, an unconstrained system with m degrees of freedom (DOFs) is subject to a set of constraints which remove c of those DOFs, and parameterizes the remaining $n = m - c$ using a reduced set of n coordinates, which are called generalized coordinates [2]. For example, an articulated figure composed of blocks, each with six DOFs, may be described in terms of joint linkages, each with one, two, or three DOFs and a set of joint angles. This approach is used by Isaacs and Cohen in [11] to construct articulated animated figures with the DYNAMO animation system.

Finding a reduced-coordinate formulation for acyclic articulated bodies is straightforward[1] but the problem becomes hard as loops are introduced[2].

[1] In this context acyclic refers to systems in which the linkages (constraints) between objects can be described as an acyclic graph.

[2] The hierarchical nature of acyclic systems makes them easy to re-parameterize in terms of independent joint angles. Introducing loops or cycles breaks the hierarchy, making

If a reduced-coordinate formulation can be found, it is possible to solve for the accelerations of the n generalized coordinates in $O(n)$ time [2].

4.1.2 Lagrange Multiplier Formulations

Lagrange multiplier formulations maintain a set of constraints by introducing constraint forces into the original m DOF system. Lagrange multiplier formulations have a number of advantages, most notably the ability to easily describe complex, arbitrary constraint systems with any number of loops. In addition, Lagrange multiplier techniques can describe velocity constraints and other types of constraints which cannot be described using reduced coordinate formulations [2]. For a tutorial introduction to Lagrange multiplier dynamic constraints, see [17]. More in-depth treatments are to be found in [16] and [2].

The disadvantage of Lagrange multiplier formulations is principally that computing constraint forces is generally an $O(n^3)$ problem, where n is the number of constraints, since it requires solving a linear system of the form

$$A\lambda = \mathbf{B},$$

where A is a square symmetric positive definite matrix of dimension n, λ is a vector of n Lagrange multipliers, and $\mathbf{B}$ is a vector of dimension n. Very often A is sparse, in which case conjugate gradient (or other iterative convergence) methods exploiting sparseness may be used. In theory such techniques reduce the order to approximately $O(n^2)$ [17, 2], but since the sparseness of A is not guaranteed there is some question as to whether these results are practical [2].

Recently, Baraff [2] has described an $O(n)$ Lagrange multiplier solution for acyclic constraint problems. Central to this technique is an always sparse formulation, and a simple $O(n)$ technique for solving the resulting linear system. In addition, Baraff makes the observation that there is a trivial $O(n)$ solution for Lagrange-multiplier constraint chains.

4.1.3 Lagrange Multiplier Constraints in the CDM

Lagrange multiplier constraints are used in the CDM for two reasons. (1) The CDM employs constraint meshes, which are difficult to state in a reduced coordinate formulation, and (2) an iterative constraint solver developed for the CDM allows solution of the Lagrange multiplier constraint

---

simple joint-angle parameterization impossible.

system in $O(kn)$ time, where n is the number of constraints and k is the number of iterations[3]. The iterative solver is discussed in Chapter 5.

4.2 Lagrange Multiplier Dynamic Constraints

In reduced coordinate formulations, constraints are formulated by eliminating DOFs and re-parameterizing the system using a smaller number of coordinates. In Lagrange multiplier techniques, DOFs are not eliminated from the dynamics formulation. Instead, constraint forces are introduced into the system which maintain the constraints by countering applied forces and momentum. A rigid body dynamic system with Lagrange multiplier constraints can be described by the following elaboration of Equation 3.12

$$M\ddot{\mathbf{x}} = \mathbf{F}_A + \mathbf{F}_C, \tag{4.1}$$

where M is the block diagonal matrix describing the inertial properties of the system, $\mathbf{x}$ is the system position vector, $\mathbf{F}_A$ represents the applied forces in the system and $\mathbf{F}_C$ represent the constraint forces.

Calculating $\mathbf{F}_C$ is the central problem of Lagrange multiplier constraint techniques. Several different approaches to calculating Lagrange multiplier constraint forces are discussed in this chapter, but all of these techniques require that constraints be formulated as constraint functions.

4.2.1 Constraint Functions

A constraint function is an algebraic function of the system's position state which measures the degree to which a constraint is being violated. A constraint function is zero if and only if the constraint it describes is met. For example, if there is a distance constraint of σ operating between two particles, this condition could be described by the constraint function

$$c(\mathbf{x}) = \|\mathbf{x}_1 - \mathbf{x}_2\|^2 - \sigma^2, \tag{4.2}$$

where $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)$. Equation 4.2 is zero if and only if $\mathbf{x}_1$ is seperated from $\mathbf{x}_2$ by distance σ. More rigorously, a constraint function is an algebraic expression of the form

$$c(\mathbf{x}) \in \mathbf{R}, \tag{4.3}$$

where $c(\mathbf{x}) = 0$ if and only if the constraint is satisfied. It is also necessary that $dc(\mathbf{x})/dx \neq 0$ at and near all x such that $c(\mathbf{x}) = 0$.[4]

[3] Experimental evidence suggests k is independent of n. See Chapters 5 and 6.

[4] Note that the gradient of Equation 4.2 is non-zero where $c(\mathbf{x}) = 0$ for values of $\sigma \neq 0$.

By the same token, A set of constraints may be described as a system of constraint equations, all of which must be satisfied (zero) for the system to be in constraint. A system of m constraints can be represented as a set of constraint functions, *i.e.*

$$C = \{c_1(\mathbf{x}), \dots, c_m(\mathbf{x})\} \tag{4.4}$$

4.2.2 Solving For Constraint Forces

Conventional Constrained Dynamics

Suppose that a system starts out with all constraints met. This means that $C = 0$ ($c_i(\mathbf{x}) = 0$ for all $i \in \{1 \dots m\}$). If the value of C stays constant over the duration of the simulation than the system will remain in constraint. In order for this condition to be met, the first and all higher time derivatives of C must be zero. Specifically,

$$\dot{C} = 0, \tag{4.5}$$

and

$$\ddot{C} = 0. \tag{4.6}$$

The first time derivative of C may be written as

$$\dot{C} = \frac{dC}{d\mathbf{x}}\dot{\mathbf{x}}, \tag{4.7}$$

and the second time derivative as

$$\ddot{C} = \frac{dC}{d\mathbf{x}}\ddot{\mathbf{x}} + \frac{d^2C}{d\mathbf{x}dt}\dot{\mathbf{x}}. \tag{4.8}$$

Taking the derivative of C with respect to the position vector yields the *Jacobian matrix* of C. Using J and $\dot{J}$ to represent the Jacobian and its first time derivative Equation 4.8 may be rewritten as

$$\ddot{C} = J\ddot{\mathbf{x}} + \dot{J}\dot{\mathbf{x}}. \tag{4.9}$$

By Equation 4.1, $\ddot{\mathbf{x}}$ may be defined as

$$\ddot{\mathbf{x}} = M^{-1}\left(\mathbf{F}_A + \mathbf{F}_C\right). \tag{4.10}$$

Combining Equations 4.6 and 4.9 results in

$$JM^{-1}\left(\mathbf{F}_A + \mathbf{F}_C\right) + \dot{J}\dot{\mathbf{x}} = 0,$$

or

$$JM^{-1}\mathbf{F}_C = -\dot{J}\dot{\mathbf{x}} - JM^{-1}\mathbf{F}_A. \tag{4.11}$$

This equation cannot be solved by itself because the right side is a vector of dimension m, where m is the number of constraints, and the unknown vector is of dimension n, where n is the dimension of the position vector $\mathbf{x}$.

In order to solve Equation 4.11 it is necessary to know more about $\mathbf{F}_C$. One important characteristic of constraint forces is that they never add or remove energy from a system. The kinetic energy of a dynamic system is

$$E_k = \frac{1}{2}M\dot{\mathbf{x}} \cdot \dot{\mathbf{x}}, \tag{4.12}$$

and its first time derivative is

$$\begin{aligned} \dot{E}_k &= M\ddot{\mathbf{x}} \cdot \dot{\mathbf{x}} \\ &= \mathbf{F}_A \cdot \dot{\mathbf{x}} + \mathbf{F}_C \cdot \dot{\mathbf{x}}. \end{aligned} \tag{4.13}$$

Equation 4.13 is the *work* done by $\mathbf{F}_A$ and $\mathbf{F}_C$. Requiring that dynamic constraints not change the kinetic energy of the system means that the constraints do no work, *i.e.*

$$\mathbf{F}_C \cdot \dot{\mathbf{x}} = 0, \tag{4.14}$$

which is known as the principal of virtual work. Combining Equation 4.14 with Equation 4.5 yields

$$\mathbf{F}_C \cdot \dot{\mathbf{x}} = 0,\ \forall\ \dot{\mathbf{x}} \mid J\dot{\mathbf{x}} = 0. \tag{4.15}$$

This requirement can be met if and only if F_C is a linear combination of the gradient vectors which compose J, or in other words

$$\mathbf{F}_C = J^\top \lambda, \tag{4.16}$$

where λ is a vector of dimension C. The scalar components of λ are called *Lagrange multipliers*. Substituting Equation 4.16 into Equation 4.11 yields

$$JM^{-1}J^\top \lambda = -\dot{J}\dot{\mathbf{x}} - JM^{-1}\mathbf{F}_A. \tag{4.17}$$

If matrix A and vector $\mathbf{B}$ are defined by

$$\begin{aligned} A &= JM^{-1}J^\top \\ \mathbf{B} &= -\dot{J}\dot{\mathbf{x}} - JM^{-1}\mathbf{F}_A, \end{aligned} \tag{4.18}$$

then λ can be defined as the solution of the linear system

$$A\lambda = \mathbf{B}, \tag{4.19}$$

where A is a square matrix of dimension C, and the dimension of λ and $\mathbf{B}$ are the same. If J is non-singular (if there are no redundant or conflicting constraints), then A is non singular and is positive definite. Assuming that A is non-singular, dense matrix methods can be used to solve for λ, from which constraint forces can be computed using Equation 4.16. As previously noted, this is generally an $O(n^3)$ process with respect to the dimension of C (the number of constraints). In [2] Baraff notes that for the special case of a constraint chain, there is a simple $O(n)$ solution for Equation 4.19.

Baraff's Formulation

In [2] Baraff describes an alternative formulation for calculating λ. Instead of Equation 4.19, Baraff proposes the matrix equation

$$\begin{pmatrix} M & -J^\top \\ -J & 0 \end{pmatrix} \begin{pmatrix} \mathbf{y} \\ \lambda \end{pmatrix} = \begin{pmatrix} 0 \\ -\mathbf{B} \end{pmatrix}. \tag{4.20}$$

Substituting the top row ($My - J^\top\lambda = 0$) into the bottom row and multiplying by -1 yields

$$Jy = J(M^{-1}J^\top\lambda) = \mathbf{B}, \tag{4.21}$$

which is Equation 4.17. Solving Equation 4.20 for λ produces both λ and y, though y is an unneeded byproduct. The advantage of this formulation is that it is always sparse, and for acyclic constraint systems it can be solved in $O(n)$ time. Unfortunately, as loops are introduced the order of the algorithm increases, and in systems involving many loops (like a constraint mesh) the order approximates that of solving Equation 4.19 by dense matrix methods.

Chapter 5

The Iterative Convergence Method

Since it is possible to solve acyclic constraint systems in linear time, it makes sense to ask whether it is might be possible to break a larger cyclic constraint system into acyclic groups, and then combine the solutions of these groups to achieve an $O(n)$ solution to the larger problem. As it turns out, there is a straightforward iterative technique by which this may be accomplished that is particularly appropriate for the cloth simulation problem that we are addressing. This chapter describes this technique by demonstrating its application in two examples, and then provides a description of the general algorithm.

5.1 A Simple Example — the Orthogonal Mesh

Consider the example of an orthogonal constraint mesh, like the one diagrammed in Figure 5.1. This figure shows a 2×2 mesh with twelve fixed-length constraints constraining nine particles. Without considering self-intersecting paths, there are thirteen loops in this system — more loops than there are constraints.

There are a number of ways of decomposing this mesh into non-redundant acyclic elements, but the simplest is to break the mesh into six straight constraint chains. These chains naturally fall into two groups: vertical (warp) chains and horizontal (weft) chains. The chains of both groups have the property of being disjoint, *i.e.* no two chains in a group share particles in common.

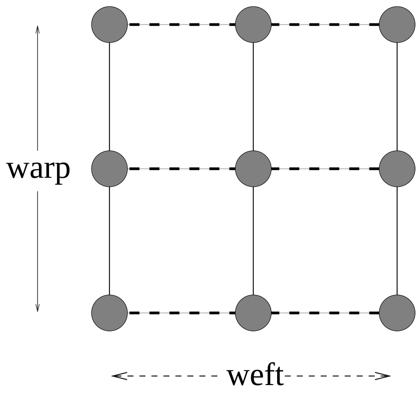

Figure 5.1: A simple orthogonal constraint mesh.

5.1.1 The Orthogonal Solution

In the case of a perfectly rectangular mesh like the one shown in Figure 5.1, every member of the warp group is orthogonal to every member of the weft group, and *vice versa*. As previous mentioned, constraint forces are linear combinations of the gradients of the constraint functions. For this reason constraint forces are always in the direction of the gradients of the respective constraint functions.

One of the most important differences between constraints and springs is that constraint forces are instantaniously transmitted througout the constraint system. Force transmission occures between constraints which act on the same object (intersect), but only to the extent that the gradients of the intersecting constraints are non-orthogonal. When intersecting constraints are orthogonal (when the inner products of the gradients are zero) constraint force transmission cannot take place. This is illustrated in Figure 5.2.

Because orthogonal constraints cannot transmit constraint forces, every chain in the mesh can be solved independently and the solutions combined to produce an overall solution to the problem. This technique is referred to as the *orthogonal solution*.

Calculating the Orthogonal Solution

Let the function $\mathcal{C}(C, M, \mathbf{x}, \dot{\mathbf{x}}, \mathbf{F}_A)$ be defined as the constraint forces resulting from solving Equations 4.17 and 4.16 for a set of constraints C,

$$\mathcal{C}(C, M, \mathbf{x}, \dot{\mathbf{x}}, \mathbf{F}_A) = J^\top \lambda \text{ such that } JM^{-1}J^\top \lambda = -\dot{J}\dot{\mathbf{x}} - JM^{-1}\mathbf{F}_A. \quad (5.1)$$

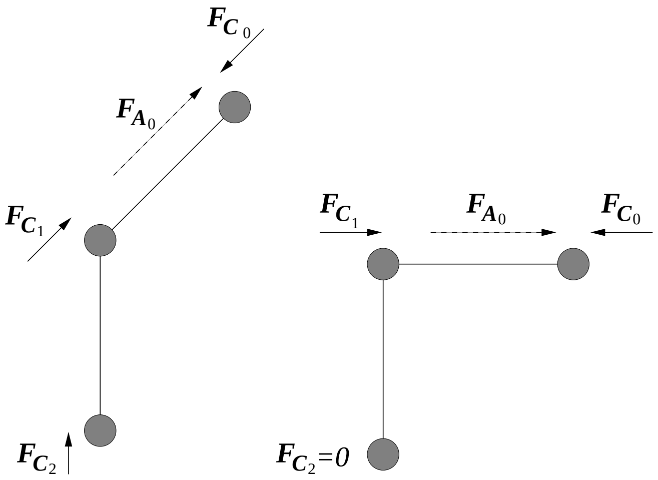

Figure 5.2: A graphic representation of constraint force transmission.

Let the applied force vectors $\mathbf{F}_{A_r} = \mathbf{F}_A$ and $\mathbf{F}_{A_f} = \mathbf{F}_A$. Solve all chains in the warp and weft groups, storing the resulting constraint forces in $\mathbf{F}_{C_r}$ and $\mathbf{F}_{C_f}$ respectively,

$$\begin{aligned}
\mathbf{F}_{C_{r_i}} &= \mathcal{C}(C_{r_i}, M, \mathbf{x}, \dot{\mathbf{x}}, \mathbf{F}_{A_r})\ \forall i\ \in \{1,2,3\} \\
\mathbf{F}_{C_{f_i}} &= \mathcal{C}(C_{f_i}, M, \mathbf{x}, \dot{\mathbf{x}}, \mathbf{F}_{A_f})\ \forall i\ \in \{1,2,3\} \\
\mathbf{F}_{C_r} &= \sum_{i=1}^{3} \mathbf{F}_{C_{r_i}} \\
\mathbf{F}_{C_f} &= \sum_{i=1}^{3} \mathbf{F}_{C_{f_i}}.
\end{aligned} \tag{5.2}$$

Sum $\mathbf{F}_{C_r}$ and $\mathbf{F}_{C_f}$ and store the result in $\mathbf{F}_C$,

$$\mathbf{F}_C = \mathbf{F}_{C_r} + \mathbf{F}_{C_f}. \tag{5.3}$$

The vector $\mathbf{F}_C$ is the orthogonal solution of the mesh.

The Order of the Orthogonal Solution

Because solving Equation 4.17 for constraint chains is an $O(n)$ process[1], the orthogonal solution is an $O(n)$ solution for the constraint mesh. A perfectly

[1] In this discussion $O(n)$ means with respect to the number of constraints unless otherwise noted.

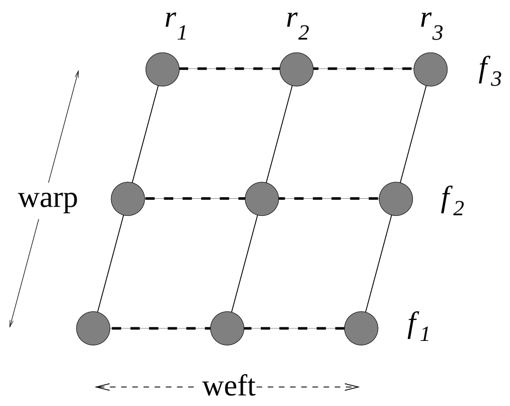

Figure 5.3: A non-orthogonal mesh.

orthogonal mesh is a special case, but the orthogonal solution forms the basis of a more general solution for a constraint mesh, which is discussed below.

5.2 The Non-orthogonal Mesh

A more complicated problem is a non-orthogonal mesh, like the one shown in Figure 5.3. Suppose that this mesh is divided into straight constraint chains, and that these chains are grouped as before into disjoint warp and weft groups. The three chains in the warp group are labeled r_1, r_2, and r_3, and the three weft chains are labeled f_1, f_2, and f_3. Because the two groups are not perfectly orthogonal, constraint forces will be transmitted from warp chains to weft chains. This means that the orthogonal solution will result in some error, because the independent solution of each chain will not take into account the constraint forces being transmitted by intersecting chains. If the mesh is very close to being orthogonal this error will be relatively small. As the mesh becomes less orthogonal, the orthogonal solution becomes less accurate. A simple iterative process, which will be referred to as the *iterative solution*, may be used to improve the accuracy of the orthogonal solution.

5.2.1 The Iterative Solution

Let ϵ be a user-defined error tolerance representing the lower-bound on correction force magnitudes.

Step One – Calculate the Orthogonal Solution

The first step is to calculate the orthogonal solution, resulting in $\mathbf{F}_{C_r}$, $\mathbf{F}_{C_f}$, and $\mathbf{F}_C = \mathbf{F}_{C_r} + \mathbf{F}_{C_f}$. At this point $\mathbf{F}_C$ is the correct solution to the constraint problem plus an unknown error vector, or

$$\mathbf{F}_C = \mathbf{F}_{C_{true}} + \mathbf{F}_{error}, \tag{5.4}$$

where $\mathbf{F}_{C_{true}}$ is the correct solution and $\mathbf{F}_{error}$ is the error vector.

Step Two — Iterative Refinement

To reduce the error of the current solution, the constraint (or correction) forces calculated during the previous step become the applied forces for the opposite group, *i.e.* let

$$\begin{aligned} \mathbf{F}_{A_r} &= \mathbf{F}_{C_f} \\ \mathbf{F}_{A_f} &= \mathbf{F}_{C_r}. \end{aligned} \tag{5.5}$$

Correction forces are calculated by letting $\dot{\mathbf{x}} = 0$ (recalculating the constraint response to momentum would be redundant) and solving Equation Group 5.2. The vector $\mathbf{F}_\Delta = \mathbf{F}_{C_r} + \mathbf{F}_{C_f}$ is a correction to the previous solution,

$$\mathbf{F}_C = \mathbf{F}_C + \mathbf{F}_\Delta. \tag{5.6}$$

The true magnitude of $\mathbf{F}_{error}$ in Equation 5.4 is unknown, but the nature of the iterative convergence process guarantees that $\|\mathbf{F}_{error}\| \leq \|\mathbf{F}_\Delta\|$. Thus, the magnitude of $\mathbf{F}_\Delta$ provides an upper-bound on the current error magnitude[2]. Therefore, if $\|\mathbf{F}_\Delta\| \leq \epsilon$ the solution is within ϵ, otherwise repeat the iterative refinement step.

The Order of the Iterative Solution

The iterative solution is based on repeatedly solving Equation Group 5.2, which (as previously demonstrated) is an $O(n)$ process. Hence, the order of the iterative solution algorithm is $O(kn)$, where k is the number of iterations. If k depends on n in any way, the true order of the algorithm could be $O(n \log n)$, $O(n^2)$, or higher. However, if k is independent of n, then the iterative solution is linear with respect to the number of constraints.

[2] I know that this assertion is true, though I do not know how to prove it. I suspect that this has already been demonstrated in the literature — any ideas for a reference?

Analytically determining the meaning of k is difficult, but experimental evidence suggests that k is independent of n, and is most strongly determined by the relative orthogonality of the mesh. See Chapter 6 for a discussion of the experimental evidence.

5.3 The General Solution

The iterative solution algorithm is a special case of a more general algorithm for calculating Lagrange-multiplier constraint forces for cyclic sets of constraints, the *general iterative convergence algorithm*. This algorithm has four basic steps:

1. Decompose the cyclic set of constraints into acyclic subsets, *i.e.*constraint chains and acyclic graphs (trees).
2. Organize disjoint acyclic subsets into groups (groups in which no two subsets share constrained elements).
3. Solve for the general orthogonal solution.
4. Iteratively refine the solution.

5.3.1 Step One — Decompose the System

Every cyclic set of n constraints C can be decomposed into $2 \leq m \leq n$ acyclic subsets S_i, such that

$$S_i \subset C, \;\; S_i \neq \emptyset, \tag{5.7}$$

and

$$S_i \cap S_j = \emptyset \; \forall \; i, j \mid i \neq j, \tag{5.8}$$

with the set of all subgroups S defined as

$$S = \{S_1, \ldots, S_m\}. \tag{5.9}$$

The set of elements constrained by S_i is referred to as E_{S_i}.

The number of ways C can be decomposed is a matter of combinatorics, but as a practical matter the decomposition should be done so as to minimize the orthogonality within subsets and maximize the orthogonality between subsets. Put another way, group together those constraints which tend to mutually transmit constraint forces, and at the same time tend not to transmit forces to other groups. Secondarily, the decomposition should minimize the number of subsets.

Choosing a good decomposition is critical to the performance of the algorithm, because the degree of orthogonality between subsets strongly determines the number of iterations in the iterative refinement step.

5.3.2 Step Two — Group the Subsets

The subsets resulting from Step One are grouped together according to the criteria that no two subsets in the same group G_i can share constrained elements (the subsets must be disjoint), *i.e.*

$$G_i \subset S, G_i \neq \emptyset, \tag{5.10}$$

$$G_i \cap G_j = \emptyset \;\forall\; i, j \;\; i \neq j, \tag{5.11}$$

$$E_{S_i} \cap E_{S_j} = \emptyset \;\forall\; E_{S_i}, E_{S_j} \in G_i, \tag{5.12}$$

and that the total number of groups should be kept as small as possible. This results in $2 \leq l \leq m$ groups. The applied forces acting on G_i are referred to as $\mathbf{F}_{A_i}$ and the constraint forces as $\mathbf{F}_{C_i}$. The set of all groups G is defined as

$$G = \{G_1, \ldots, G_l\}. \tag{5.13}$$

Likewise, the set of all group applied forces G_{F_A} and constraint forces G_{F_C} are defined as

$$G_{F_A} = \{\mathbf{F}_{A_1}, \ldots, \mathbf{F}_{A_l}\}, \tag{5.14}$$

$$G_{F_C} = \{\mathbf{F}_{C_1}, \ldots, \mathbf{F}_{C_l}\}. \tag{5.15}$$

Minimizing the number of groups is important because the iterative refinement step is $O(l^2)$ with respect to the number of groups. It should be noted that the minimum number of groups for a given decomposition is a function of the topology, and not the number of constraints. For example, every constraint mesh, irrespective of size, can be decomposed into constraint chains organized into two groups.

5.3.3 Step Three — the General Orthogonal Solution

The general orthogonal solution is a first approximation for solving the constraint system C. Given an applied force vector $\mathbf{F}_A$, the set of acyclic subsets S, the set of groups G, let the function $\mathcal{A}(A, M, \mathbf{x}, \dot{\mathbf{x}}, \mathbf{F})$ be defined as the constraint forces resulting from the solution of the acyclic set of constraints

A in response to applied forces $\mathbf{F}$. Calculate the constraint forces for each group G_i as

$$\mathbf{F}_{C_i} = \sum_{\forall\ S_j \in G_i} \mathcal{A}(S_j, M, \mathbf{x}, \dot{\mathbf{x}}, \mathbf{F}_A). \tag{5.16}$$

The general orthogonal solution is calculated as

$$\mathbf{F}_O = \mathbf{F}_C = \sum_{i=1}^{l} \mathbf{F}_{C_i}, \tag{5.17}$$

where $\mathbf{F}_O$ is the orthogonal solution and $\mathbf{F}_C$ is the current solution force vector for C.

5.3.4 Step Four — Iterative Refinement

There are three phases to the iterative refinement process: force collection, correction force calculation, and error evaluation.

Force Collection

In the force collection phase the G_{F_C} set of forces calculated during the previous step becomes the new applied forces in a crossover calculation which allows each group to respond to the forces produced by every other group during the previous step,

$$\mathbf{F}_{A_i} = \sum_{\forall j \neq i} \mathbf{F}_{C_j}\ \forall\ 1 \leq i \leq l. \tag{5.18}$$

Correction Force Calculation

In the correction force calculation phase, a set of correction forces G_{F_C} are calculated in response to the new set of applied forces G_{F_A} resulting from the force collection phase,

$$\mathbf{F}_{C_i} = \sum_{\forall\ S_j \in G_i} \mathcal{A}(S_j, M, \mathbf{x}, 0, \mathbf{F}_{A_i})\ \forall\ G_i \in G. \tag{5.19}$$

The correction force vector $\mathbf{F}_\Delta$ is the sum of the correction forces,

$$\mathbf{F}_\Delta = \sum_{i=1}^{l} \mathbf{F}_{C_i}. \tag{5.20}$$

This correction vector is added to the previous solution vector to produce a new solution,

$$\mathbf{F}_C = \mathbf{F}_C + \mathbf{F}_\Delta. \tag{5.21}$$

Error Evaluation

During the error evaluation phase, the magnitude of the correction vector is compared with a user-specified error tolerance ϵ. This tolerance is typically specified as,

$$\epsilon = \begin{cases} \alpha\|\mathbf{F}_O\| & \text{if } \alpha\|\mathbf{F}_O\| > \omega \\ \omega & \text{otherwise} \end{cases} , \tag{5.22}$$

where α and ω are positive real values specified by the user. If the orthogonality of the constraint groups is very low the convergence process may be slow, and in the degenerate case of all gradients being parallel $\mathbf{F}_C$ will not converge at all. For this reason it is also necessary to compare the number of iterations I against a user-specified maximum I_{max}. If $\|\mathbf{F}_\Delta\| \leq \epsilon$ or $I = I_{max}$ stop, else repeat Step Four.

5.3.5 The Order of the General Algorithm

Steps One and Two

Steps one and two of the general iterative convergence algorithm require user direction, but need only be done once for a given set of constraints C.

Step Three

Step Three is repeated once each time constraint forces are calculated. This step requires the solution of m acyclic constraint elements. Since the total number of constraints in these groups is n, and since the solution of each one of the m groups is linear with respect to the number of constraints in that group, total order of Step Three is $O(n)$ with respect to the total number of constraints in C.

Step Four

Step Four is repeated $1 \leq k \leq I_{max}$ each time constraint forces are calculated. The first phase, force collection, is $O(m^2)$ with respect to the number of groups. The second phase, correction force calculation, is identical in process to Step Three, and is therefore $O(n)$ with respect to the total number of constraints in C. Thus, the overall order of Step Four is $O(knm^2)$. As previously noted, the number of groups m is independent of the total number of constraints n.

Overall Order

The overall order of the general iterative convergence algorithm is

$$O(n) + O(knm^2) = O(knm^2) \tag{5.23}$$

where k is the number of iterations of Step Four, n is the total number of constraints in C, and m is the number of constraint groups (which is independent of n). The problem is that it is not clear what k depends on. It appears that k depends to some extent on the choice of ϵ, but it is also plausible that k could depend on n or m. Experimental evidence suggests that k is independent of both n and m, and is most strongly determined by the relative orthogonality of the acyclic constraint subsets. This proposition, and the experiments designed to test it, are discussed in the next chapter.

Chapter 6

Discussion

This chapter discusses the implementation of the forward dynamics simulation system, the CDM, and the iterative convergence constraint solver. Experiments testing various aspects of the modeling system are described, and conclusions are drawn about the usefullness of the CDM as an animation tool. Future research directions are also discussed.

6.1 Implementation

The cloth modeling system developed over the course of this research project is composed of three main components: a forward dynamics simulation system, the cloth dynamics model, and the general iterative convergence algorithm constraint solver. The implementation of each component is discussed, both in terms of general implementation issues and the specifics of the software used to test the model.

6.1.1 Research System Overview

The *research system* is the implementation used to conduct the tests and experiments described later in this chapter. The research system was written in C++ using the Gnu C++ compiler and associated development tools, Mesa (a freeware graphics library with an API similar to OpenGL), and the Linux operating system. The software also compiles and runs on the Silicon Graphics IRIX platform using CC and OpenGL. All tests reported here were conducted on a 233MHz AMD K6 based PC with 64 MB of ram.

6.1.2 Implementing The Forward Dynamics Simulation

The CDM is designed to work with virtually any Newtonian formulation forward dynamics simulation system; For that reason few implementation issues need be discussed. One issue of some importance is the choice of numerical integrator. Another issue which effects performance is the choice of an internal representation for vectors and matrices.

Numerical Integration

Cloth springs are non-linear and tend to be fairly stiff, particularly the trellising springs. We believe the use of a higher-order integrator is important to simulation stability and performance. The research system uses a fixed time-step 4th order Runge-Kutta integrator. Because cloth is the only dynamic object simulated in the research system, the use of a fixed time step allows the integration interval to be tailored to the stiffness of the cloth springs. In a more general simulation environment a variable time-step integrator may be more appropriate.

The Representation of Vectors and Matricies

The description of the CDM and forward dynamics simulation system in previous chapters have used a block vector and matrix notation, which may be easily represented as one and two dimensional arrays. However, particularly in the case of matrices, it makes sense to use a block-sparse internal representation. In the research system the position vector and its time derivatives are implemented as block sparse vectors which use a self-balancing binary search tree internally to organize the data. Matricies are represented internally by simple arrays (for band-diagonal matricies of consistant size) or arrays of block sparse vectors.

This organization has the advantage of flexibility (the number of objects in the simulation can change without re-allocating the state vector, vector operations can be done without making size comparisons, *etc.*) but accessing individual elements of the state vectors (the position vector and its first two time derivatives) requires $O(\log s)$ comparisons where s is the number of blocks in the vector. For this reason operations which are $O(n)$ in principle but require accessing individual elements of a state vector (*e.g.* solving a constraint chain) result in $O(n \log s)$ run times, where s is the size of the position vector. This most dramatically effects comparisons in run-times between small and medium sized or large simulations where $\log s$ is significantly different.

6.1.3 Implementing the Cloth Dynamics Model

Three important CDM implementation issues are: (1) choosing a formulation for external forces, (2) choosing a technique for internal damping, and (3) deciding if and how to prevent cloth self-intersection.

External Forces

The choice of external forces depends on the application, but typically this will include gravity, aerodynamic drag, and forces resulting from collision and contact between the cloth and other objects. Of these three forces, aerodynamic drag and collision/contact forces are the most complicated. The viscous drag formulation of Equation 3.18 can be adapted to reproduce the effect of wind as well as movement of the cloth in still air, and should work for virtually all animation applications.

Collision and contact forces present a more complicated problem, since they can be handled in a number of different ways. Collision detection and response is a topic of study in its own right, but whatever method is chosen the implementer must keep in mind that a single piece of simulated cloth may be composed of thousands of particles.

The research system uses the external force formulation described above, except that collision forces are constraint forces resulting from non-penetration constraints instead of applied forces. This makes no real difference to the dynamics of the cloth model, but does allow the operation of the general iterative convergence algorithm to be demonstrated for situations more complicated than a single constraint mesh.

Internal Damping

We believe that the frictional interaction between cloth fibers results in behaviorally significant damping, but are unaware of any research on this subject. Draping tests run without internal damping have resulted in behavior which looks a bit springy, though the obviousness of this effect depends greatly on the degree of external damping provided by aerodynamic drag. At the time of this writing, we believe that critically damping the cloth springs is the solution, but have not yet implemented this in the research system.

Self-Intersection

Preventing self-intersection is a difficult problem, especially when performance considerations are taken into account. The implementer must decide if a particular application will require self-intersection checking. If it does, A hierarchical curvature-based approach, like the one proposed in [14], is probably necessary to achieve adequate performance for all but the smallest simulations. Self-intersection prevention has not yet been implemented in the research system.

6.1.4 Implementing the Iterative Convergence Constraint Solver

Two important issues in the implementation of the general iterative convergence constraint algorithm are: efficiently implementing the iterative step, and choosing appropriate error tolerance values for the solver.

Efficient Iterative Refinement

The iterative refinement step of the general iterative convergence algorithm described in Chapter 5.3.4 is based on repeatedly solving $\mathcal{A}(S_j, M, \mathbf{x}, 0, \mathbf{F}_{A_i})$ as part of Equation 5.19. It should be noted that $\mathbf{x}$ never changes during this process, which means that J and $\dot{J}$ for each S_j in G_i need only be calculated once – during the orthogonal solution phase.

Superquadric ellipsoids [3] are implemented as non-penetration constraints in the research system, and the partial derivatives of the implicit form represent more than page of double precision arithmetic in C++, mostly `pow()` functions. Under these circumstances recalculating J and $\dot{J}$ each iteration of the refinement step is extremely expensive. Implementing the above approach in the research system added a small amount of complexity, but the gains in performance were significant.

Choosing Error Tolerances

The magnitude of the correction vector, graphed over iterations, appears to undergo an exponential decay[1]; Figure 6.1 shows a an example graph taken from the diagnostic output of the research system. When the solver

[1] Under conditions of low orthogonality or tight error tolerances it may take a large number of iterations for the solution to converge, and if the number of iterations becomes to large the solution may not converge at all. Under these circumstances the magnitude of successive correction vectors may actually begin to increase, probably as a result of numerical error. For cloth simulation applications this condition is rare and can be easily detected in the solver.

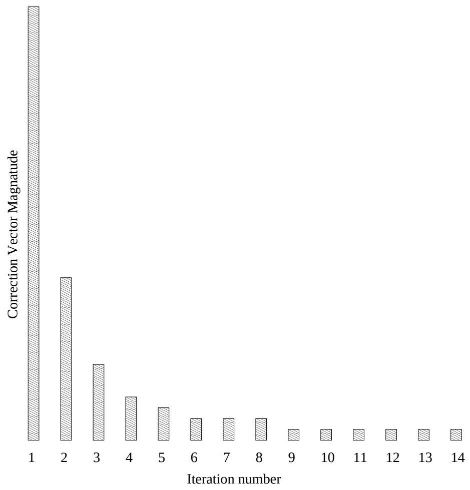

Figure 6.1: An example graph of the magnitude of $\mathbf{F}_{\triangle}$ over iterations.

is applied to a constraint mesh, the "half life" of this magnitude seems to be most strongly determined by the relative orthogonality of the mesh. For cloth applications where the trellising spring forces keep the mesh strongly orthogonal, it typically takes fewer than 20 iterations for the magnitude of the correction vector to become 1/100th of the magnitude of the orthogonal solution. This degree of accuracy ($\alpha = 1.0e^{-2}$, $\omega = 10.0$, $I_{max} = 100$) has produced good results in draping tests using the research system.

6.2 Experiments and Tests

The experiments described here fall into two categories: tests of the constraint solver and tests of the overall cloth model. The first category is particularly important, because the scalability of the model hinges on performance of the solver.

6.2.1 Testing the Constraint Solver

The Warped Mesh Test

To verify that the number of iterations of refinement required to achieve a given error tolerance does not depend on the number of constraints, an array of constraint meshes will be constructed of varying resolution and degree of orthogonality. The number of iterations required to solve each mesh will be

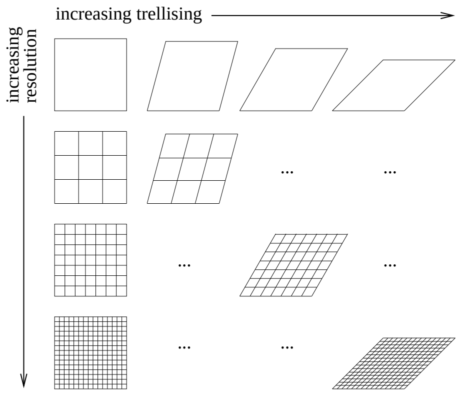

Figure 6.2: An array of meshes of increasing resolution and degree of trellising.

recorded along with the number of constraints.

The array of meshes is shown in Figure 6.2. The mesh in the upper left hand corner is composed of four particles and four constraints — two warp chains and two weft chains of length one. Each mesh to the right of the first represents fifteen degrees more trellising or shear than the previous mesh. Each mesh below the first represents twice the resolution in both warp and weft directions. Thus, the row and column of the mesh determine its resolution and degree of shear. Table **??** shows the results of this experiment.

The Sphere Draping Test

In this experiment two 1 meter square meshes are draped over a large sphere. Both meshes are the same, except that the second mesh is sampled at twice the resolution of the first. The results of this test are shown in Table **??**.

6.2.2 Testing the CDM

This set of experiments reproduces the draping tests described in [6]. One meter square pieces of cloth are draped over a large cube one-half meter on a side. The mesh is sampled at a resolution of fifty chains in both warp and weft directions. Cotton, wool, and a cotton-wool blend are simulated, and the final equilibrium positions are shown in Figure **??**. The performance data are summarized in Table **??**.

6.3 Conclusions

6.3.1 Model Accuracy

6.3.2 Solver Performance

6.3.3 Future Research

Many avenues of exploration remain. A number of general issues relating to the CDM are still to be addressed, including efficient self-intersection prevention and developing an empirical model of cloth internal damping. The CDM shows promise as an animation tool, but it is only a beginning. To be a practical tool many other problems not directly associated with modeling cloth behavior must be addressed, including the tailoring and stitching of simulated cloth elements and integrating the model with conventional animation tools. The CDM also has possible engineering applications, which would require a careful investigation and verification of the model in a fabric-science context.

A great deal remains to be done in exploring the general iterative converge algorithm. Much more can be done to investigate and verify the order of the algorithm. The concept of "relative orthogonality" must be made rigorous as it relates to the number of iterations of the solver, and the effect of constraint topology on solver behavior should be investigated. Developing an algorithm for constructing a good decomposition of an arbitrary set of constraints is an interesting research problem by itself.

Bibliography

[1] ACM SIGGRAPH. *Computer Graphics Proceedings, Annual Conference Series, 1987*, New York, July 1987.

[2] David Baraff. Linear-time dynamics using lagrange multipliers. In *Computer Graphics Proceedings, Annual Conference Series, 1996*, pages 137–146, New York, August 1996. ACM SIGGRAPH.

[3] Alan H. Barr. Superquadrics and angle-preserving transformations. *IEEE Computer Graphics and Applications*, 1, January 1981.

[4] David E. Breen. Computer graphics in textiles and apparel modeling. *IEEE Computer Graphics and Applications*, 16(5):26–27, September 1996.

[5] David E. Breen, Donald H. House, and Michael J. Wozny. A particle-based model for simulating the draping behavior of woven cloth. *Textile Research Journal*, 64(11):663–685, November 1994.

[6] David E. Breen, Donald H. House, and Michael J. Wozny. Predicting the drape of woven cloth using interacting particles. In *Computer Graphics Proceedings, Annual Conference Series, 1994*, pages 365–372, New York, August 1994. ACM SIGGRAPH.

[7] Michel Carignan, Ying Yang, Nadia Magnenat Thalmann, and Daniel Thalmann. Dressing animated synthetic actors with complex deformable clothes. In *Computer Graphics Proceedings, Annual Conference Series, 1992*, pages 92–104, New York, July 1992. ACM SIGGRAPH.

[8] Bernhard Eberhardt, Andreas Weber, and Wolfgang Strasser. A fast, flexible, particle-system model for cloth draping. *IEEE Computer Graphics and Applications*, pages 52–59, September 1996.

[9] C. R. Feynman. Modeling the appearance of cloth. MSc thesis, Department of Electrical Engineering and Computer Science, MIT, Cambridge, MA, 1986.

[10] Donald H. House, Richard W. DeVaul, and Devid E. Breen. Towards simulating cloth dynamics using interacting particles. *International Journal of Clothing Science and Technology*, 8(3):75–94, 1996.

[11] Paul M. Isaacs and Michael F. Cohen. Controlling dynamic simulation with kinematic constraints, behavior functions and inverse dynamics. In *Computer Graphics Proceedings, Annual Conference Series, 1987* [1], pages 131–140.

[12] Hing N. Ng and Richard L. Grimsdale. Computer graphics techniques for modeling cloth. *IEEE Computer Graphics and Applications*, 16(5):28–41, September 1996.

[13] Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. Elastically deformable models. In *Computer Graphics Proceedings, Annual Conference Series, 1987* [1], pages 205–214.

[14] Pascal Volino, Martin Courchesne, and Nadia Magnenat Thalmann. Versatile and efficient techniques for simulating cloth and other deformable objects. In *Computer Graphics Proceedings, Annual Conference Series, 1995*, pages 137–144, New York, August 1995. ACM SIGGRAPH.

[15] Jerry Weil. The synthesis of cloth objects. In *Computer Graphics Proceedings, Annual Conference Series, 1986*, pages 49–53, New York, August 1986. ACM SIGGRAPH.

[16] A. Witkin, M. Gleicher, and W. Welch. Interactive dynamics. In *Computer Graphics Proceedings, Annual Conference Series, 1990*, pages 11–21, New York, August 1990. ACM SIGGRAPH.

[17] Andrew Witkin, David Baraff, and Michael Kass. Physically based modeling. In *SIGGRAPH95 Course Notes*, SIGGRAPH Course Notes, page Course 34, New York, August 1995. ACM, ACM SIGGRAPH.