SIGGRAPH 2003 COURSE# 29:

“Clothing Simulation and Animation”

Course Notes

Organizers:

Hyeong-Seok Ko
Seoul National University

David Breen
California Institute of Technology

Lecturers:

Michael Hauth
University of Tübingen

Ronald Fedkiw
Stanford University

Rob House
Sony Pictures Imageworks
About the Course

A character appearing with fashionable clothing can add another dimension of richness to 3D animation. Looking at the degree of detail in these days' character animation, it is easily inferable that realistic, stable, fast clothing animation technique will be in an ever increasing demand. This course is targeted to intermediate level animation researchers. It provides an introduction to state-of-the-art techniques for simulating and animating clothing.

The course begins by an introduction which is organized as a “tutorial within course” so that even beginners can get a comprehensive idea how cloth simulation is done. It may serve as a summary of the whole course, and also guide attendees what specific items should be learned in the subsequent lectures.

The three most fundamental issues for simulating cloth - the physical model, numerical techniques, and collision handling - are presented during the next 2.5 hours. In the first session, the course describes low-level models of cloth, its representation by a set of particles and the simulation of its movement through the evaluation of a set of differential equations. The second session of the course will describe how to solve the differential equations and how to handle the collisions. Since the equations tend to be stiff, serious instability problems are possible. Numerical techniques that can avoid the instabilities will be presented. Collision detection and response are important issues when simulating clothing. We present methods for detecting cloth/body and cloth/cloth collisions and incorporating the appropriate response to the collisions into simulation.

In Physical Model of Cloth I (0.75 hour), Dr. Breen will present early work on cloth modeling, supplemented with a rich collection of still images and animations that spans the last 15 years of cloth modeling research. In Physical Model of Cloth II (0.5 hour) Prof. Ko will present the later updates on the interacting particles model of cloth, including the post-buckling instability and immediate buckling assumption. Numerical Techniques for Cloth Simulation (0.75 hour) will be lectured by Michael Hauth. Collision Detection/Handling will be lectured by Prof. Fedkiw, who is the author of the paper "Robust Treatment of Collisions, Contact and Friction for Cloth Animation" in SIGGRAPH 2002.

Finally, the course closes by showing how animators use the commercial or in-house tools to model and animate the costumes of the characters. Rob House from Sony Pictures Imageworks reveals how garment construction, clothing animation, rendering of the surface details are done in the
feature film Stuart Little. Towards the end, animation researchers will hear vivid voices from animators: “what needs to be done further?”

**Prerequisites**

Rudimentary knowledge of computer graphics, computer animation, geometric modeling, linear algebra, and numerical computing

**Course Schedule**

Course Title: Clothing Simulation and Animation  
Course Number: 29

Component 1: Overview and Modeling  
1:45 Introduction – Ko  
2:15 Physical Model of Cloth I – Breen  
3:00 Physical Model of Cloth II – Ko  
3:15 Break

Component 2: Simulation and Animation  
3:30 Physical Model of Cloth II (continued) – Ko  
3:45 Numerical Techniques for Cloth Simulation – Hauth  
4:30 Collision Detection/Handling – Fedkiw  
5:00 Clothing Animation at Sony Pictures Imageworks – House
Course Speakers

Prof. Hyeong-Seok Ko
Director of Graphics & Media Lab
Seoul National University
ko@graphics.snu.ac.kr
Hyeong-Seok Ko is an associate professor in the School of Electrical Engineering at Seoul National University. Currently, he is the Director of Graphics and Media Lab at the same university, and is actively developing the techniques for realizing Digital Actors, the actors created by computer graphics technology which are so real that people cannot tell if they are animated or captured from the real world. Specific areas of current research include cloth animation, hairstyle modeling and animation, facial animation, physically-based motion retargeting. He has published two SIGGRAPH papers and numerous journal articles in the above topics. Before joining Seoul National University, he has held an assistant professor position at the University of Iowa. Ko received B.A. and M.S. degrees in Computer Science from Seoul National University in 1981 and 1985. He received a Ph.D. degree in Computer and Information Science from University of Pennsylvania. He was the conference co-chair of Computer Animation 2001 conference, and the program co-chair of Computer Animation 2003 conference, which was held at Rutgers University on May 8-9, 2003. He will serve as the program co-chair of Pacific Graphics 2004, which will be held in Seoul in 2004.

Dr. David E. Breen
Senior Research Scientist
Center for Advanced Computing Research
California Institute of Technology
david@acr.caltech.edu
David Breen is a Senior Research Scientist at the Center for Advanced Computing Research at the California Institute of Technology. Prior to this, he was the Assistant Director of Caltech's Computer Graphics Laboratory. He has held research positions at the European Computer-Industry Research Centre, the Fraunhofer Institute for Computer Graphics, and the Rensselaer Design Research Center (formerly the RPI Center for Interactive Computer Graphics). His research interests include level set models for computer graphics, volume segmentation, large data visualization, and geometric modeling. Breen received a B.A. degree in Physics from Colgate University in 1982. He received M.S. and Ph.D. degrees in Computer and Systems Engineering from Rensselaer Polytechnic Institute in 1985 and 1993. He has previously published in the SIGGRAPH Proceedings, has chaired both a SIGGRAPH panel and course, and presented at two
SIGGRAPH tutorials. Breen is currently the program co-chair of the ACM Symposium on Computer Animation. He is also the co-editor of the book Cloth Modeling and Animation (AK Peters).

Michael Hauth
Research Staff
Computer Graphics Laboratory (GRIS)
University of Tübingen
hauth@gris.uni-tuebingen.de
Michael Hauth is a member of the research staff of the Computer Graphics Laboratory (GRIS) at the University of Tübingen. He studied computer sciences, physics and mathematics at the University of Tübingen. He received a Diploma (M.Sc.) in computer sciences (1999) and a Diploma (M.Sc.) in mathematics (2000) from the University of Tübingen. In 2000/2001 he held a visiting post as research assistant at the Numerical Analysis Group of the University of Geneva.

Prof. Ronald Fedkiw
Computer Science Dept.
Stanford University
fedkiw@cs.stanford.edu
Fedkiw received his Ph.D. in Mathematics from UCLA in 1996 and did postdoctoral studies both at UCLA in Mathematics and at Caltech in Aeronautics before joining the Stanford Computer Science Department. He was awarded a Packard Foundation Fellowship, a Presidential Early Career Award for Scientists and Engineers (PECASE), an Office of Naval Research Young Investigator Program Award (ONR YIP), a Robert N. Noyce Family Faculty Scholarship, two distinguished teaching awards, etc. Currently he is on the editorial board of the Journal of Scientific Computing and the IEEE Transactions on Visualization and Computer Graphics, and participates in the reviewing process of a number of journals and funding agencies. He has published approximately 35 research papers in computational physics, computer graphics and vision, as well as a new book on level set methods. For the past two years, he has been a consultant with Industrial Light + Magic.

Robert House
Senior Technical Director
Sony Imageworks, Culver City, California
rhouse@imageworks.com
Rob has worked on 14 feature films, including "Stuart Little", "Hollowman" and "Stuart Little 2", for which he managed construction and simulation of dynamic cloth wardrobe.
Contents

A. Introduction (Slides)

B. David Baraff and Andrew Witkin, "Large Steps in Cloth Simulation". (excerpted from the Proceedings of SIGGRAPH 1998)

C. Physical Model of Cloth I (Slides)

D. David Breen, Donald House and Michael Wozny, Predicting the Drape of Woven Cloth Using Interacting Particles’. (excerpted from the Proceedings of SIGGRAPH 1998)

E. Cloth/Clothing Modeling and Animation Bibliography, Compiled by David Breen

F. Physical Model of Cloth II (Slides)


H. Numerical Techniques for Cloth Simulation (Slides)

I. Numerical Techniques for Cloth Simulation (written by Hauth)

Welcome to SIGGRAPH 2003 Course #29: “Clothing Simulation and Animation”

SIGGRAPH 2003 Course #29:
Clothing Simulation and Animation

Organizers: Hyeong-Seok Ko
David Breen
Lecturers: Michael Hauth
Ronald Fedkiw
Rob House
The Technique is in High Demand from

- Character animation
- Game industry
- Fashion industry
- Textile industry

The Demand will Continuously Grow.

The Technique is full of Challenges

- Realistic cloth
- Interactive cloth
- Stable cloth
- Complex cloth
- Collision detection/handling
Organizers and Lecturers

in the order of appearance...

Hyeong-Seok Ko
Seoul National Univ.

David Breen
Caltech

Michael Hauth
Univ. of Tubingen

Ronald Fedkiw
Stanford Univ.

Rob House
Sony Pictures Imageworks

Goal of the Course

- Learn how cloth simulation works
- Learn cloth simulation fundamentals
  - Physical model of cloth
  - Numerical techniques
  - Collision detection
- Learn clothing animation process in film industry
## Course Schedule

**1st Session**
1. Introduction (Ko, 30min.)
2. Physical Model of Cloth I (Breen, 45min.)
3. Physical Model of Cloth II (Ko, 30min.)

**2nd Session**
1. Numerical Techniques for Cloth Simulation (Hauth, 45min.)
2. Collision Detection/Handling (Fedkiw, 30min.)
3. Clothing Animation at Sony Pictures Imageworks (House, 30min.)

## Target Audience of the Course

- People who have some experiences on animation research, and want to initiate clothing animation
- Visual artists who want to understand the underlying process of clothing animation
- Not targeted to world experts of the subject
Introduction

How Cloth Simulation Works?

Hyeong-Seok Ko
Seoul National Univ.
Graphics & Media Lab

Goal of this 30 minutes

- Serves as a tutorial within the course
  - What is the big picture of the whole process?
  - "How cloth simulation works?"
  - Understands the structure of this course

- Guides what specific items should be learned in the subsequent sessions
How does it Work?

- Finding Governing Equation
  - What is the physical model behind cloth?
    \[ \ddot{x} = M^{-1} \left( -\frac{\partial E}{\partial x} + F \right) \]

---

“Physical Model of Cloth I & II”

David Breen

H.-S. Ko


How does it Work?

- Finding Governing Equation
- Solving the Equation $\ddot{x} = M^{-1}(-\frac{\partial E}{\partial x} + F)$
  - Uses numerical techniques
  - It is not that simple job

![Image of a cloth simulation]

How does it Work?

- Finding Governing Equation
- Solving the Equation

Michael Hauth

“Numerical Techniques for Cloth Simulation”
How does it Work?

- Finding Governing Equation
- Solving the Equation
- Collision Detection/Handling
  - This topic has been studied for a while
  - But cloth simulation is full of challenging cases

How does it Work?

- Finding Governing Equation
- Solving the Equation
- “Collision Detection/Handling”

Ronald Fedkiw
How does it Work?

- Finding Governing Equation
- Solving the Equation
- Collision Detection/Handling
- Creating Clothed Characters
  - Constructing garments
  - Rendering surface details
  - A lot more...

---

Rob House

“Clothing Animation at Sony Pictures Imageworks”
How Cloth Simulation Works?

Let’s take a closer look!

How to Represent Cloth?

- As a set of particles
- (with interconnecting springs)
Particle-based Cloth Simulation

- Repeat the following:
  1. Find the new position of particles
  2. Draw the surface from the particles

History of particle-based cloth modeling

“Physical Model of Cloth I”

Physically-based Cloth Simulation

- We assume particles move according to some governing equation.

\[
\ddot{x} = M^{-1} \left(-\frac{\partial E}{\partial x} + F\right)
\]

\(x\): vector, the geometric state
\(M\): diagonal matrix, mass distribution of the cloth
\(E\): a scalar function of \(x\), cloth’s internal energy
\(F\): a function of \(x\) and \(x'\), other forces acting on cloth
Notations

\[(m_{1i}, x_i) \quad (m_{2i}, x_i) \quad (m_{3i}, x_i)\]

Newton’s Equation \((ma=F)\)

\[M_i \ddot{x}_i = \sum \text{forces}\]

\[M_i = \begin{bmatrix} m_i & 0 & 0 \\ 0 & m_i & 0 \\ 0 & 0 & m_i \end{bmatrix}\]

- Bottom-up approach
- What kind of forces are acting on each particle?
- What are the mag. and dir. of the forces?
Forces resulting from Potential Energy

- Example: gravity field

\[ E = mgy \]

\[ -\frac{\partial E}{\partial \vec{x}} = -\left( \frac{\partial E}{\partial x}, \frac{\partial E}{\partial y}, \frac{\partial E}{\partial z} \right) \]

\[ = -(0, mg, 0) \]

\[ = F \]

\[ \therefore \text{Problem of finding forces reduces to finding the potential energy function.} \]

In Cloth,

- Potential energy is related to deformations.
  - Stretch / Shear / Bending
- What are the potential energy functions for each type of deformation?
  - \( E_{\text{stretch}}, E_{\text{shear}}, E_{\text{bending}} \)
  - Details in “Physical Model of Cloth I”
The Potential Function

\[ E = E_{\text{stretch}} + E_{\text{shear}} + E_{\text{bending}} \]

So, governing eq looks like

\[ M_i \ddot{x}_i = -\left[ \frac{\partial E}{\partial x} \right]_{x=x_i} = -\left[ \frac{\partial E_{\text{stretch}}}{\partial x} + \frac{\partial E_{\text{shear}}}{\partial x} + \frac{\partial E_{\text{bending}}}{\partial x} \right]_{x=x_i} \]
There are also other kinds of forces

\[ M_i \ddot{x}_i = -\left[ \frac{\partial E}{\partial x} \right]_{x=x_i} + F_i \]

Putting all particles into a vector...

\[ x = \begin{bmatrix} x_1 \\
\vdots \\
 x_N \end{bmatrix} \]
Putting all particles into a vector...

\[ M_i \ddot{x}_i = -\left[ \frac{\partial E}{\partial x} \right]_{x=x_i} + F_i \]

\[ M\ddot{x} = -\frac{\partial E}{\partial x} + F \]

\[ \ddot{x} = M^{-1}(-\frac{\partial E}{\partial x} + F) \]

Steps for Finding Governing Eqs

- Determine the potential energy functions for each type of deformations.
- Obtain the total internal force by summing space-derivatives of the potential functions.
- Identify extra external forces.
Now, numerically solve

\[
\ddot{x} = M^{-1}( - \frac{\partial E}{\partial x} + F )
\]

\[
\ddot{x}(t) = M^{-1}( - \frac{\partial E}{\partial x} + F )(t)
\]

If symbolic deriv. of E is not available...

\[
\ddot{x} = M^{-1}( - \frac{\partial E}{\partial x} + F )
\]

\[
\ddot{x}(t) = M^{-1}( - \frac{\partial E}{\partial x} + F )(t)
\]
If symbolic deriv. of E is not available...

\[
\ddot{x} = M^{-1} \left( - \frac{\partial E}{\partial x} + F \right)
\]

\[
\ddot{x}(t) = M^{-1} \left( - \frac{\partial E}{\partial x} + F \right)(t)
\]

\[
\ddot{x}(t) = M^{-1} \left\{ - \frac{E(x+\Delta x,t) - E(x,t)}{\Delta x} + F(x,t) \right\}
\]

**Numerical Integration**

\[
\ddot{x}(t) = M^{-1} \left( - \frac{\partial E}{\partial x} + F \right)(t)
\]

```plaintext
while(1) {
    compute the force part at t;
    compute accel, vel, pos;
    t = t + \Delta t;
}
```
**We want to use large $\Delta t$**

- Animators love large $\Delta t$.
- Large $\Delta t$ can cause inaccuracy.
- But that’s OK if result is visually pleasing.
- Large $\Delta t$ can cause more serious problem.
- Stability issues:
  - We want both stability and large $\Delta t$.

Numerical Techniques for Cloth Simulation (Hauth)

---

**Post-buckling Instability**

- Why unstable when wrinkles are formed?
- A physical instability
- Not a numerical instability
- Numerical techniques cannot solve the problem
- “Use immediate buckling assumption”
- Improvements in realism, stability, speed

Physical Model of Cloth II (Ko)
Collisions in Cloth Simulation

- Undetected collision can cause a lot of trouble.
- Cloth simulation is abundant of challenging cases for collision detection.
- An interesting challenge: simulate garments formed by multiple layers of cloth.
- We have to generate cloth movement due to collisions.

“Collision Detection/Handling” (Fedkiw)

How Stuart Little was Clothed?

- A lot more is needed
  - Garment construction
  - Rendering the surface details of cloth
- The process used by Sony Pictures Imageworks is revealed.
- We can hear vivid voices from animators
  - “What needs to be done further?”

“Clothing Animation at Sony Pictures Imageworks” (House)
Parts Covered and Not Covered

- Finding Governing Equation
- Solving the Equation
- Collision Detection/Handling
- Garment Construction
- Other Fashioning Features
- Real-time Simulation

Parts Covered and Not Covered

- Particle Model
- Continuum Model
Large Steps in Cloth Simulation

David Baraff  Andrew Witkin

Robotics Institute
Carnegie Mellon University

Abstract

The bottle-neck in most cloth simulation systems is that time steps must be small to avoid numerical instability. This paper describes a cloth simulation system that can stably take large time steps. The simulation system couples a new technique for enforcing constraints on individual cloth particles with an implicit integration method. The simulator models cloth as a triangular mesh, with internal cloth forces derived using a simple continuum formulation that supports modeling operations such as local anisotropic stretch or compression; a unified treatment of damping forces is included as well. The implicit integration method generates a large, unbanded sparse linear system at each time step which is solved using a modified conjugate gradient method that simultaneously enforces particles’ constraints. The constraints are always maintained exactly, independent of the number of conjugate gradient iterations, which is typically small. The resulting simulation system is significantly faster than previous accounts of cloth simulation systems in the literature.

Keywords—Cloth, simulation, constraints, implicit integration, physically-based modeling.

1 Introduction

Physically-based cloth animation has been a problem of interest to the graphics community for more than a decade. Early work by Terzopoulos et al. [17] and Terzopoulos and Fleischer [15, 16] on deformable models correctly characterized cloth simulation as a problem in deformable surfaces, and applied techniques from the mechanical engineering and finite element communities to the problem. Since then, other research groups (notably Carignan et al. [4] and Volino et al. [20, 21]; Breen et al. [3]; and Eberhardt et al. [5]) have taken up the challenge of cloth.

Although specific details vary (underlying representations, numerical solution methods, collision detection and constraint methods, etc.), there is a deep commonality amongst all the approaches: physically-based cloth simulation is formulated as a time-varying partial differential equation which, after discretization, is numerically solved as an ordinary differential equation

\[
\dot{x} = M^{-1} \left( -\frac{\partial E}{\partial x} + F \right). \tag{1}
\]

In this equation the vector \(x\) and diagonal matrix \(M\) represent the geometric state and mass distribution of the cloth, \(E\)—a scalar function of \(x\)—yields the cloth’s internal energy, and \(F\) (a function of \(x\) and \(\dot{x}\)) describes other forces (air-drag, contact and constraint forces, internal damping, etc.) acting on the cloth.

In this paper, we describe a cloth simulation system that is much faster than previously reported simulation systems. Our system’s faster performance begins with the choice of an implicit numerical integration method to solve equation (1). The reader should note that the use of implicit integration methods in cloth simulation is far from novel: initial work by Terzopoulos et al. [15, 16, 17] applied such methods to the problem.\(^1\) Since this time though, research on cloth simulation has generally relied on explicit numerical integration (such as Euler’s method or Runge-Kutta methods) to advance the simulation, or, in the case of of energy minimization, analogous methods such as steepest-descent [3, 10].

This is unfortunate. Cloth strongly resists stretching motions while being comparatively permissive in allowing bending or shearing motions. This results in a “stiff” underlying differential equation of motion [12]. Explicit methods are ill-suited to solving stiff equations because they require many small steps to stably advance the simulation forward in time.\(^2\) In practice, the computational cost of an explicit method greatly limits the realizable resolution of the cloth. For some applications, the required spatial resolution—that is, the dimension \(n\) of the state vector \(x\)—can be quite low: a resolution of only a few hundred particles (or nodal points, depending on your formulation/terminology) can be sufficient when it comes to modeling flags or tablecloths. To animate clothing, which is our main concern, requires much higher spatial resolution to adequately represent realistic (or even semi-realistic) wrinkling and folding configurations.

In this paper, we demonstrate that implicit methods for cloth overcome the performance limits inherent in explicit simulation methods. We describe a simulation system that uses a triangular mesh for cloth surfaces, eliminating topological restrictions of rectangular meshes, and a simple but versatile formulation of the internal cloth energy forces. (Unlike previous metric-tensor-based formulations [15, 16, 17, 4] which model some deformation energies as quartic functions of positions, we model deformation energies only as quadratic functions with suitably large scaling. Quadratic energy models mesh well with implicit integration’s numerical properties.)

We also introduce a simple, unified treatment of damping forces, a subject which has been largely ignored thus far. A key step in our simulation process is the solution of an \(O(n) \times O(n)\) sparse linear system, which arises from the implicit integration method. In this respect, our implementation differs greatly from the implementation by Terzopoulos et al. [15, 17], which for large simulations

\(^1\)Additional use of implicit methods in animation and dynamics work includes Kass and Miller [8], Terzopoulos and Qin [18], and Tu [19].

\(^2\)Even worse, the number of time steps per frame tends to increase along with the problem size, for an explicit method. Cloth simulations of size \(n\)—meaning \(x \in \mathbb{R}^{O(n)}\)—generally require \(O(n)\) explicit steps per unit simulated time. Because the cost of an explicit step is also \(O(n)\) (setting aside complications such as collision detection for now) explicit methods for cloth require time \(O(n^2)\)—or worse.
used an “alternating-direction” implicit (ADI) method [12]. An ADI method generates a series of tightly banded (and thus quickly solved) linear systems rather than one large sparse system. (The price, however, is that some of the forces in the system—notably between diagonally-adjacent and non-adjacent nodes involved in self-collisions—are treated explicitly, not implicitly.) The speed (and ease) with which our sparse linear systems can be robustly solved—even for systems involving 25,000 variables or more—has convinced us that there is no benefit to be gained from using an ADI method instead (even if ADI methods could be applied to irregular triangular meshes). Thus, regardless of simulation size, we treat all forces as part of the implicit formulation. Even for extremely stiff systems, numerical stability has not been an issue for our simulator.

1.1 Specific Contributions

Much of the performance of our system stems from the development of an implicit integration formulation that handles contact and geometric constraints in a direct fashion. Specifically, our simulator enforces constraints without introducing additional penalty terms in the energy function $E$ or adding Lagrange-multiplier forces into the force $F$. (This sort of direct constraint treatment is trivial if equation (1) is integrated using explicit techniques, but is problematic for implicit methods.) Our formulation for directly imposing and maintaining constraints is harmonious with the use of an extremely fast iterative solution algorithm—a modified version of the conjugate gradient (CG) method—to solve the $O(n) \times O(n)$ linear system generated by the implicit integrator. Iterative methods do not in general solve linear systems exactly—they are run until the solution error drops below some tolerance threshold. A property of our approach, however, is that the constraints are maintained exactly, regardless of the number of iterations taken by the linear solver. Additionally, we introduce a simple method, tailored to cloth simulation, for dynamically adapting the size of time steps over the course of a simulation.

The combination of implicit integration and direct constraint satisfaction is very powerful, because this approach almost always allows us to take large steps forward. In general, most of our simulations require on average from two to three time steps per frame of 30 Hz animation, even for (relatively) fast moving cloth. The large step sizes complement the fact that the CG solver requires relatively few iterations to converge. For example, in simulating a 6,000 node system, the solver takes only 50–100 iterations to solve the 18,000 $\times$ 18,000 linear system formed at each step. Additionally, the running time of our simulator is remarkably insensitive to the cloth’s material properties (quite the opposite behavior of explicit methods). All of the above advantages translate directly into a fast running time. For example, we demonstrate results similar to those in Breen et al. [3] and Eberhardt et al. [5] (draping of a 2,600 node cloth) with a running time just over 2 seconds per frame on an SGI Octane R10000 195 Mhz processor. Similarly, we show garments (shirts, pants, skirts) exhibiting complex wrinkling and folding behavior on both key-framed and motion-captured characters. Representative running times include a long skirt with 4,530 nodes (8,844 triangles) on a dancing character at a cost of 10 seconds per frame, and a shirt with 6,450 nodes (12,654 triangles) with a cost varying between 8 to 14 seconds per frame, depending on the underlying character’s motion.

1.2 Previous Work

Terzopoulos et al. [15, 17] discretized cloth as a rectangular mesh. Energy functions were derived using a continuum formulation. This work recognized the need for damping forces; however, only a simple viscous drag force $-k\dot{x}$ was used. The linear systems resulting from the use of implicit integration techniques were solved, for small systems, by direct methods such as Choleski factorization, or using iterative techniques such as Gauss-Seidel relaxation or conjugate gradients. (For a square system of $n$ nodes, the resulting linear system has bandwidth $\sqrt{n}$. In this case, banded Choleski factorization [6] requires time $O(n^2)$.) As previously discussed, Terzopoulos et al. made use of an ADI method for larger cloth simulations.

Following Terzopoulos et al.’s treatment of deformable surfaces, work by Carignan et al. [4] described a cloth simulation system using rectangular discretization and the same formulation as Terzopoulos et al. Explicit integration was used. Carignan et al. recognized the need for damping functions which do not penalize rigid-body motions of the cloth (as simple viscous damping does) and they added a force which damps cloth stretch and shear (but not bend). Later work by the same group includes Volino et al. [20], which focuses mainly on collision detection/response and uses a triangular mesh; no mention is made of damping forces. The system uses the midpoint method (an explicit method) to advance the simulation. Thus far, the accumulated work by this group (see Volino et al. [21] for an overview) gives the only published results we know of for simulated garments on moving characters. Reported resolutions of the garments are approximately two thousand triangles per garment (roughly 1,000 nodal points) [21] with running times of several minutes per frame for each garment on an SGI R4400 150 Mhz processor.

Breen et al. [3] depart completely from continuum formulations of the energy function, and describe what they call a “particle-based” approach to the problem. By making use of real-world cloth material properties (the Kawabata measuring system) they produced highly realistic static images of draped rectangular cloth meshes with reported resolutions of up to 51 $\times$ 51 nodes. The focus of this work is on static poses for cloth, as opposed to animation; thus, their simulation process is best described as energy minimization, although methods analogous to explicit methods are used. Speed was of secondary concern in this work. Refinements by Eberhardt et al. [5]—notably, the use of higher-order explicit integration methods and Maple-optimized code, as well as a dynamic, not static treatment of the problem—obtain similarly realistic results, while dropping the computational cost to approximately 20–30 minutes per frame on an SGI R8000 processor. No mention is made of damping terms. Provot [13] focuses on improving the performance of explicit methods by a post-step modification of nodal positions. He iteratively adjusts nodal positions to eliminate unwanted stretch; the convergence properties of this method are unclear. A more comprehensive discussion on cloth research can be found in the survey paper by Ng and Grimsdale [9].

2 Simulation Overview

In this section, we give a brief overview of our simulator’s architecture and introduce some notation. The next section derives the linear system used to step the simulator forward implicitly while section 4 describes the specifics of the internal forces and their derivatives that form the linear system. Section 5 describes how constraints are maintained (once established), with a discussion in section 6 on collision detection and constraint initialization. Section 7 describes our adaptive step-size control, and we conclude in section 8 with some simulation results.

2.1 Notation and Geometry

Our simulator models cloth as a triangular mesh of particles. Given a mesh of $n$ particles, the position in world-space of the $i$th particle is $x_i \in \mathbb{R}^3$. The geometric state of all the particles is simply $x \in \mathbb{R}^{3n}$. 

SIGGRAPH '98
The same component notation applies to forces: a force \( f \in \mathbb{R}^{3n} \) acting on the cloth exerts a force \( f_i \) on the \( i \)th particle. Real-world cloth is cut from flat sheets of material and tends to resist deformations away from this initial flat state (creases and pleats not withstanding). We capture the rest state of cloth by assigning each particle an unchanging coordinate \((u_i, v_i)\) in the plane.\(^3\) Section 4 makes use of these planar coordinates.

Collisions between cloth and solid objects are handled by preventing cloth particles from interpenetrating solid objects. Our current implementation models solid objects as triangularly faced polyhedra. Each face has an associated thickness and an orientation; particles found to be sufficiently near a face, and on the wrong side, are deemed to have collided with that face, and become subject to a contact constraint. (If relative velocities are extremely high, this simple test may miss some collisions. In this case, analytically checking for intersection between previous and current positions can guarantee that no collisions are missed.) For cloth/cloth collisions, we detect the intersection between previous and current positions can guarantee.

We capture the rest state of cloth by assigning each particle an unchanging coordinate \((u_i, v_i)\) in the plane.\(^3\) Section 4 makes use of these planar coordinates.

2.2 Energy and Forces

The most critical forces in the system are the internal cloth forces which impart much of the cloth’s characteristic behavior. Breen et al. [3] describes the use of the Kawabata system of measurement for realistic determination of the in-plane shearing and out-of-plane bending forces in cloth. We call these two forces the shear and bend forces. We formulate the shear force on a per triangle basis, while the bend force is formulated on a per edge basis—between pairs of adjacent triangles.

The strongest internal force—which we call the stretch force—resists in-plane stretching or compression, and is also formulated per triangle. Under normal conditions, cloth does not stretch appreciably under its own weight. This requires the stretch force to have a high coefficient of stiffness, and in fact, it is the stretch force that is most responsible for the stiffness of equation (1). A common practice in explicitly integrated cloth systems is to improve running time by decreasing the strength of the stretch force; however, this leads to “rubbery” or “bouncy” cloth. Our system uses a very stiff stretch force to combat this problem, without any detrimental effects on the run-time performance. While the shear and bend force stiffness coefficients depend on the material being simulated, the stretch coefficient is essentially the same (large) value for all simulations. (Of course, if stretchy cloth is specifically called for, the stretch coefficient can be made smaller.)

Complementing the above three internal forces are three damping forces. In section 5, we formulate damping forces that subdue any oscillations having to do with, respectively, stretching, shearing, and bending motions of the cloth. The damping forces do not dissipate energy due to other modes of motion. Additional forces include air-drag, gravity, and user-generated generated mouse-forces (for interactive simulations). Cloth/cloth contacts generate strong repulsive linear-spring forces between cloth particles.

Combining all forces into a net force vector \( f \), the acceleration \( \ddot{x}_i \) of the \( i \)th particle is simply \( \ddot{x}_i = f_i / m_i \), where \( m_i \) is the \( i \)th particle’s mass. The mass \( m_i \) is determined by summing one third the mass of all triangles containing the \( i \)th particle. (A triangle’s mass is the product of the cloth’s density and the triangle’s fixed area in the \( uv \) coordinate system.) Defining the diagonal mass matrix \( M \in \mathbb{R}^{3n \times 3n} \) by \( \text{diag}(M) = (m_1, m_1, m_2, m_2, \ldots, m_n, m_n, m_n) \), we can write simply that

\[
\ddot{x} = M^{-1} f(x, \dot{x}).
\]

2.3 Sparse Matrices

The use of an implicit integration method, described in the next section, generates large unbounded sparse linear systems. We solve these systems through a modified conjugate gradient (CG) iterative method, described in section 5. CG methods exploit sparsity quite easily, since they are based solely on matrix-vector multiplies, and require only rudimentary sparse storage techniques. The sparsity of the matrix generated by the implicit integrator is best represented in block-fashion: for a system with \( n \) particles, we deal with an \( n \times n \) matrix, whose non-zero entries are represented as dense \( 3 \times 3 \) matrices of scalars. The matrix is represented as an array of \( n \) rows; each row is a linked list of the non-zero elements of that row, to accommodate possible run-time changes in the sparsity pattern, due to cloth/cloth contact. The (dense) vectors that are multiplied against this matrix are stored simply as \( n \) element arrays of three-component vectors. The overall implementation of sparsity is completely straightforward.

2.4 Constraints

An individual particle’s position and velocity can be completely controlled in either one, two, or three dimensions. Particles can thus be attached to a fixed or moving point in space, or constrained to a fixed or moving surface or curve. Constraints are either user-defined (the time period that a constraint is active is user-controlled) or automatically generated, in the case of contact constraints between cloth and solids. During cloth/solid contacts, the particle may be attached to the surface, depending on the magnitudes of the frictional forces required; otherwise, the particle is constrained to remain on the surface, with sliding allowed. The mechanism for releasing a contact constraint, or switching between sliding or not sliding, is described in section 5.

The constraint techniques we use on individual particles work just as well for collections of particles; thus, we could handle cloth/cloth intersections using the technique described in section 5, but the cost is potentially large. For that reason, we have chosen to deal with cloth/cloth contacts using penalty forces: whenever a particle is near a cloth triangle or is detected to have passed through a cloth triangle, we add a stiff spring with damping to pull the particle back to the correct side of the triangle. The implicit solver easily tolerates these stiff forces.

3 Implicit Integration

Given the known position \( x(t_0) \) and velocity \( \dot{x}(t_0) \) of the system at time \( t_0 \), our goal is to determine a new position \( x(t_0 + h) \) and velocity \( \dot{x}(t_0 + h) \) at time \( t_0 + h \). To compute the new state and velocity using an implicit technique, we must first transform equation (2) into a first-order differential equation. This is accomplished simply by defining the system’s velocity \( v = \dot{x} = \dot{x} \) and then writing

\[
\frac{d}{dt} \begin{pmatrix} x \\ \dot{x} \end{pmatrix} = \begin{pmatrix} 0 \\ \frac{d}{dt} \end{pmatrix} \begin{pmatrix} x \\ v \end{pmatrix} = \begin{pmatrix} v \\ M^{-1} f(x, \dot{x}) \end{pmatrix}.
\]

To simplify notation, we will define \( \Delta x = x(t_0 + h) - x(t_0) \) and \( \Delta \dot{x} = \dot{x}(t_0 + h) - \dot{x}(t_0) \).

\[
\Delta x = \frac{d}{dt} x(t_0 + h) - x(t_0) = \dot{x}(t_0 + h) - \dot{x}(t_0) = \Delta \dot{x}.
\]
The explicit forward Euler method applied to equation (3) approximates $\Delta x$ and $\Delta v$ as

$$
\begin{pmatrix}
\Delta x \\
\Delta v 
\end{pmatrix} = h \begin{pmatrix}
v_0 \\
M^{-1}f_0 
\end{pmatrix}
$$

where the force $f_0$ is defined by $f_0 = f(x_0, v_0)$. As previously discussed, the step size $h$ must be quite small to ensure stability when using this method. The implicit backward Euler method appears similar at first: $\Delta x$ and $\Delta v$ are approximated by

$$
\begin{pmatrix}
\Delta x \\
\Delta v 
\end{pmatrix} = h \begin{pmatrix}
v_0 + \Delta v \\
M^{-1}f(x_0 + \Delta x, v_0 + \Delta v) 
\end{pmatrix}.
$$

The difference in the two methods is that the forward method's step is based solely on conditions at time $t_0$ while the backward method's step is written in terms of conditions at the terminus of the step itself.\footnote{The method is called “backward” Euler because starting from the output state $(x_0 + \Delta x, v_0 + \Delta v)$ and using a forward Euler step to run the system backward in time (i.e. taking the step $-h(v(t_0 + h), f(x(t_0 + h), v(t_0 + h)))$ brings you back to $(x_0, v_0)$. What is the value in this? Forward Euler takes no notice of wildly changing derivatives, and proceeds forward quite blindly. Backward Euler, however, forces one to find an output state whose derivative at least points back to where you came from, imparting, essentially, an additional layer of consistency (or sanity-checking, if you will).}

The forward method requires only an evaluation of the function $f$ but the backward method requires that we solve for values of $\Delta x$ and $\Delta v$ that satisfy equation (4). Equation (4) is a nonlinear equation: rather than solve this equation exactly (which would require iteration) we apply a Taylor series expansion to $f$ and make the first-order approximation

$$
f(x_0 + \Delta x, v_0 + \Delta v) = f_0 + \partial f / \partial x \Delta x + \partial f / \partial v \Delta v.
$$

In this equation, the derivative $\partial f / \partial x$ is evaluated for the state $(x_0, v_0)$ and similarly for $\partial f / \partial v$. Substituting this approximation into equation (4) yields the linear system

$$
\begin{pmatrix}
\Delta x \\
\Delta v 
\end{pmatrix} = h \begin{pmatrix}
v_0 + \Delta v \\
M^{-1}f_0 + \partial f / \partial x \Delta x + \partial f / \partial v \Delta v 
\end{pmatrix}.
$$

Taking the bottom row of equation (5) and substituting $\Delta x = h(v_0 + \Delta v)$ yields

$$
\Delta v = hM^{-1} \begin{pmatrix}
f_0 + \partial f / \partial x (v_0 + \Delta v) + \partial f / \partial v \Delta v 
\end{pmatrix}.
$$

Letting $I$ denote the identity matrix, and regrouping, we obtain

$$
\left( I - hM^{-1} \partial f / \partial x - h^2M^{-1} \partial f / \partial v \right) \Delta v = hM^{-1} \begin{pmatrix}
f_0 + h \partial f / \partial v \end{pmatrix}.
$$

which we then solve for $\Delta v$. Given $\Delta v$, we trivially compute $\Delta x = h(v_0 + \Delta v)$.

Thus, the backward Euler step consists of evaluating $f_0$, $\partial f / \partial x$, and $\partial f / \partial v$; forming the system in equation (6); solving the system for $\Delta v$; and then updating $x$ and $v$. We use the sparse data structures described in section 2.3 to store the linear system. The sparsity pattern of equation (6) is described in the next section, while solution techniques are deferred to section 5.

4 Forces

Cloth’s material behavior is customarily described in terms of a scalar potential energy function $E(x)$; the force $f$ arising from this energy is $f = -\partial E / \partial x$. Equation (6) requires both the vector $f$ and the matrix $\partial f / \partial x$. Expressing the energy $E$ as a single monolithic function—encompassing all aspects of the cloth’s internal behavior—and then taking derivatives is impractical, from a bookkeeping point of view. A better approach is to decompose $E$ into a sum of sparse energy functions; that is, to write $E(x) = \sum E_i(x)$ where each $E_i$ depends on as few elements of $x$—as few particles—as possible.

However, even decomposing $E$ into sparse energy functions is not enough. Energy functions are an undesirable starting point because sensible damping functions cannot be derived from energy functions. Instead, we define internal behavior by formulating a vector condition $C(x)$ which we want to be zero, and then defining the associated energy as $\alpha C(x)^T C(x)$ where $\alpha$ is a stiffness constant.

In section 4.5, we show how sensible damping functions can be constructed based on this formulation. An added bonus is that starting from this vector-based energy description tends to result in a simpler, more compact, and more easily coded formulation for $\partial f / \partial x$ than proceeding from an energy function in which the structure of $C$ has been lost.

4.1 Forces and Force Derivatives

Given a condition $C(x)$ which we want to be zero, we associate an energy function $E_C$ with $C$ by writing $E_C(x) = \alpha C(x)^T C(x)$ where $\alpha$ is a stiffness constant of our choice. Assuming that $C$ depends only on a few particles, $C$ gives rise to a sparse force vector $f$. Recall from section 2.1 that we view the vector $f$ in block form; each element $f_i$ is a vector in $\mathbb{R}^3$. For each particle $i$ that $C$ depends on,

$$
f_i = -\partial E_C / \partial x_i = -\alpha \partial C / \partial x_i C(x);
$$

all the other elements of $f$ are zero.

Similarly, the derivative of $f$ is also sparse. Defining the derivative matrix $K = \partial f / \partial x$, the nonzero entries of $K$ are $K_{ij}$ for all pairs of particles $i$ and $j$ that $C$ depends on. Again, we treat $K$ in block fashion: $K \in \mathbb{R}^{3n \times 3n}$, so an element $K_{ij}$ is a $3 \times 3$ matrix. From equation (7), we have

$$
K_{ij} = \frac{\partial f_i}{\partial x_j} = -\alpha \partial C / \partial x_i \partial C / \partial x_j + \partial^2 C / \partial x_i \partial x_j C(x).
$$

Additionally, since $K_{ii}$ is a second derivative—that is, $K_{ii} = \partial^2 f / \partial x_i^2 = \partial^2 E / \partial x_i^2$—we have $K_{ii} = K_{ii}^T$ so $K$ is symmetric. Note that since $C$ does not depend on $v$, the matrix $\partial f / \partial v$ is zero.

We can now easily describe the internal forces acting on the cloth, by just writing condition functions. Forces and their derivatives are easily derived using equations (7) and (8).

4.2 Stretch Forces

Recall that every cloth particle has a changing position $x_i$ in world space, and a fixed plane coordinate $(u_i, v_i)$. Even though our cloth is modeled as a discrete set of points, grouped into triangles, it will be convenient to pretend momentarily that we have a single continuous function $w(u, v)$ that maps from plane coordinates to world space.

Stretch can be measured at any point in the cloth surface by examining the derivatives $w_u = \partial w / \partial u$ and $w_v = \partial w / \partial v$ at that point. The magnitude of $w_u$ describes the stretch or compression in the $u$ direction; the material is unstretched wherever $\|w_u\| = 1$. Stretch in the
The bend energy depends upon the four particles where $\Delta u_1 = u_1 - u_i$ and similarly for $\Delta u_2$. We approximate $w(u,v)$ as a linear function over each triangle; this is equivalent to saying that $w_u$ and $w_v$ are constant over each triangle. This lets us write $\Delta x_1 = w_u \Delta u_1 + w_v \Delta v_1$ and $\Delta x_2 = w_u \Delta u_2 + w_v \Delta v_2$. Solving for $w_u$ and $w_v$ yields

$$w_u = \frac{(\Delta x_1 \Delta x_2)}{(\Delta u_1 \Delta v_1 + \Delta u_2 \Delta v_2)}.$$  

Note that $x_1$ and $x_2$ vary during the simulation but the matrix in the above equation does not.

We can treat $w_u$ and $w_v$ as functions of $x$, realizing that they depend only on $x_u$, $x_v$, and $x_i$ and using equation (9) to obtain derivatives. The condition we use for the stretch energy is

$$C(x) = a \left( \frac{\|w_u(x)\| - b_u}{\|w_v(x)\| - b_v} \right),$$  

where $a$ is the triangle’s area in $uv$ coordinates. Usually, we set $b_u = b_v = 1$, though we need not always do so. In particular, if we want to slightly lengthen a garment (for example, a sleeve) in the $u$ direction, we can increase $b_u$, which causes $w_u$ to seek a larger value, and tends to induce wrinkles along the $u$ direction. Likewise, we might decrease $b_v$ near the end of a sleeve, inducing a tight cuff, as on a sweatshirt. We have found the ability to control shrink/stretch directions to be a unit vectors parallel to the triangle’s normal direction). When the simulation is fast enough to interact with, we add user-controlled “mouse” forces. These forces and their gradients are easily derived.

### 4.3 Shear and Bend Forces

Cloth likewise resists shearing in the plane. We can measure the extent to which cloth has sheared in a triangle by considering the inner product $\mathbf{w}_u \cdot \mathbf{w}_v$. In its rest state, this product is zero. Since the stretch term prevents the magnitudes of $w_u$ and $w_v$ from changing overly much, we need not normalize. By the small angle approximation, the product $\mathbf{w}_u \cdot \mathbf{w}_v$ is a reasonable approximation to the shear angle. The condition for shearing is simply

$$C(x) = a \left( w_u(x)^T w_v(x) \right),$$

with $a$ the triangle’s area in the $uv$ plane.

We measure bend between pairs of adjacent triangles. The condition we write for the bend energy depends upon the four particles defining the two adjoining triangles. If we let $\mathbf{n}_1$ and $\mathbf{n}_2$ denote the unit normals of the two triangles and let $\mathbf{e}$ be a unit vector parallel to the common edge, the angle $\theta$ between the two faces is defined by the relations $\sin \theta = (\mathbf{n}_1 \times \mathbf{n}_2) \cdot \mathbf{e}$ and $\cos \theta = \mathbf{n}_1 \cdot \mathbf{n}_2$. We define a condition for bending by writing simply $C(x) = \theta$ which results in a force that counters bending. The assumption that the stretch energy will keep the cloth from stretching much allows us to treat $\mathbf{n}_1$, $\mathbf{n}_2$, and $\mathbf{e}$ as having a constant length at each step of the simulation. This makes differentiating $\theta$ with respect to $x$ a manageable task.

Rectangular meshes make it simple to treat bending anisotropically. The $uv$ coordinates associated with particles make this possible for triangular meshes as well. Given material for which bending in the $u$ and $v$ directions are weighted by stiffnesses $k_u$ and $k_v$, we can emulate this anisotropy as follows. Let the edge between the triangles be between particles $i$ and $j$, and define $\Delta u = u_i - u_j$ and $\Delta v = v_i - v_j$. The stiffness weighting for this edge should simply be

$$k_u (\Delta u)^2 + k_v (\Delta v)^2.$$

### 4.4 Additional Forces

To the above forces we also add easily implemented forces such as gravity and air-drag (which is formulated on a per-triangle basis, and opposes velocities along the triangle’s normal direction). When the simulation is fast enough to interact with, we add user-controlled “mouse” forces. These forces and their gradients are easily derived.

### 4.5 Damping

The energies we have just described are functions of position only. Robust dynamic cloth simulation, however, is critically dependent on well-chosen damping forces that are a function of both position and velocity. For example, the strong stretch force must be accompanied by a suitably strong damping force if we are to prevent anomalous in-plane oscillations from arising between connected particles. However, this strong damping force must confine itself solely to damping in-plane stretching/compressing motions: stretch damping should not arise due to motions that are not causing stretch or compression. Terzopoulos et al.’s [16, 17] treatment of cloth used a simple viscous damping function which dissipated kinetic energy, independent of the type of motion. Carignan et al. [4] improved upon this somewhat, borrowing a formulation due to Platt and Barr [11]; however, their damping function—a linear function of velocity—does not match the quartic energy functions of their continuum formulation. In this section we describe a general treatment for damping that is independent of the specific energy function being damped.

It is tempting to formulate a damping function for an energy function $E(x)$ by measuring the velocity of the energy, $\dot{E} = \frac{dE}{dt}$. This is an easy trap to fall into, but it gives nonsensical results. At an equilibrium point of $E$, the gradient $\partial E/\partial x$ vanishes. Since $\dot{E} = (\partial E/\partial x)^T \dot{x}$, we find that $\dot{E}$ is zero when $E$ is at its minimum, regardless of the system’s velocity $\dot{x} = \mathbf{v}$. In general, $\dot{E}$ is always too small near the system’s rest state. Clearly, basing the damping force on $\dot{E}$ is not what we want to do.

We believe that the damping function should be defined not in terms of the energy $E$, but in terms of the condition $C(x)$ we have been using to define energies. The force $\mathbf{f}$ arising from the energy acts only in the direction $\partial C(x)/\partial x$, and so should the damping force. Additionally, the damping force should depend on the component of the system’s velocity in the $\partial C(x)/\partial x$ direction; in other words, the damping strength should depend on $(\partial C(x)/\partial x)^T \dot{x} = C(x)$. Putting this together, we propose that the damping force $\mathbf{d}$ associated with a condition $C$ have the form

$$\mathbf{d} = -k \frac{\partial C(x)}{\partial x} \cdot \mathbf{C}(x).$$  

This neatly parallels the fact that $\mathbf{f} = -k \frac{\partial C(x)}{\partial x} \cdot \mathbf{C}(x)$.
Given the condition functions \( C \) we have defined in this section for stretch, bend and shear forces, as well as those for damping forces by applying equation (11). As before, \( d_i \) is nonzero only for those particles that \( C \) depends on, and \( \partial d_i / \partial \mathbf{x} \) has the same sparsity pattern as \( \partial f / \partial \mathbf{x} \). Differentiating equation (11), we obtain

\[
\frac{\partial d_i}{\partial \mathbf{x}_j} = -k_d \left( \frac{\partial C(x)}{\partial \mathbf{x}_i} \frac{\partial C(x)^T}{\partial \mathbf{x}_j} + \frac{\partial^2 C(x)}{\partial \mathbf{x}_i \partial \mathbf{x}_j} \right). \tag{12}
\]

Note that \( \partial d_i / \partial \mathbf{x} \) is not a second derivative of some function as was the case in equation (8) so we cannot expect \( \partial d_i / \partial \mathbf{x} \) to be symmetrical. In equation (12), it is the term \((\partial C(x)/\partial \mathbf{x}_i)(\partial C(x)/\partial \mathbf{x}_j)^T\) which breaks the symmetry. Anticipating section 5.2, we find it expedient simply to leave this term out, thereby restoring symmetry. This simplification is clearly not physically justifiable, but we have not observed any ill effects from this omission. (Omitting all of equation (12), however, causes serious problems.)

Finally, equation (6) requires the derivative \( \partial d_i / \partial \mathbf{v} \). Since \( C(x) = (\partial C(x)/\partial \mathbf{x})^T \mathbf{v} \), we have

\[
\frac{\partial C(x)}{\partial \mathbf{v}} = \frac{\partial}{\partial \mathbf{v}} \left( \frac{\partial C(x)^T}{\partial \mathbf{x}} \mathbf{v} \right) = \frac{\partial C(x)}{\partial \mathbf{x}}.
\]

Using this fact, we can write

\[
\frac{\partial d_i}{\partial \mathbf{v}_j} = -k_d \frac{\partial C(x)}{\partial \mathbf{x}_i} \frac{\partial C(x)^T}{\partial \mathbf{x}_j} = -k_d \frac{\partial C(x)}{\partial \mathbf{x}_i} \frac{\partial C(x)^T}{\partial \mathbf{x}_j}.
\]

In this case, the result is symmetrical without dropping any terms.

### 5 Constraints

In this section, we describe how constraints are imposed on individual cloth particles. The constraints we discuss in this section are either automatically determined by the user (such as geometric attachment constraints on a particle) or are contact constraints (generated by the system) between a solid object and a particle. The techniques we describe in this section could be used for multi-particle constraints; however, constraints that share particle would need to be merged. Thus, a set of four-particle constraints (such as vertex/triangle or edge/edge contacts in the cloth) might merge to form a single constraint on arbitrarily many particles, which would be expensive to maintain. Because of this, we handle cloth/cloth contacts with strong springs (easily dealt with, given the simulator’s underlying implicit integration base) and “position alteration,” a technique described in section 6.

At any given step of the simulation, a cloth particle is either completely unconstrained (though subject to forces), or the particle may be constrained in either one, two or three dimensions. Given the differential nature of our formulation, it is the particle’s acceleration, or equivalently, the change in the particle’s velocity, that is constrained. If the particle is constrained in all three dimensions, then we are explicitly setting the particle’s velocity (at the next step). If the constraint is in two or one dimensions, we are constraining the particle’s velocity along either two or one mutually orthogonal axes. Before describing our constraint method, we discuss several other possible enforcement mechanisms and explain why we chose not to use them.

### Reduced Coordinates

An obvious and quite exact method for constraining a particle is to reduce the number of coordinates describing the particle’s position and velocity. A completely constrained particle would have no coordinates, while a particle with one dimension of constraint would have two coordinates. This is possible—but it complicates the system immensely. If we change the number of coordinates per particle, we alter the size of the derivative matrices in equation (6), as well as the sparsity pattern (this happens when a particle changes from having no coordinates to some coordinates, or vice versa). Given the transient nature of contact constraints between cloth and solids, this is most unappealing. The computation of the derivative matrices’ entries is also greatly complicated, because we must now introduce extra Jacobian matrices that relate a particle’s reduced coordinates to its motion in world-space. Finally, correct constraint-release behavior between cloth and solid objects is difficult to achieve using a reduced coordinate formulation. Considering all of this, we immediately rejected this method of constraints.

### Penalty Methods

We could constrain particles through the use of strong energy functions—essentially, stiff springs that attempt to prevent illegal particle motions. Since our entire formulation is geared to handle stiffness, the usual objections to enforcing constraints with springs—very stiff equations—do not carry as much weight. We tried this for a time, and found it to be a not unreasonable constraint enforcement mechanism. However, penalty methods do not enforce constraints exactly, and they do add some additional stiffness to the system. Since the mechanism we describe enforces constraints exactly, and adds no extra stiffness, we turned away from penalty methods except in the case of cloth/cloth interactions.

### Lagrange Multipliers

We could introduce additional constraint forces—that is, Lagrange multipliers—into our system to satisfy the constraints. This involves augmenting the linear system of equation (6) with extra variables (the multipliers) and extra equations (the constraint conditions). Unfortunately, this turns a positive definite system into an indefinite system, which means that iterative methods such as CG will need to square the system first, thereby doubling the running time and degrading the numerical conditioning of the linear system. Additionally, an iterative method will generally not enforce the constraints exactly without a large number of iterations. (A direct method for solving the augmented system would, however, avoid this problem.) Again, the constraint method we describe steps past these difficulties, so we turned away from using Lagrange multipliers.

#### 5.1 Mass Modification

The idea behind our constraint enforcement mechanism is described quite simply, although the actual implementation is somewhat more complicated, to maximize performance. A dynamic simulation usually requires knowledge of the inverse mass of objects; for example, note the appearance of \( \mathbf{M}^{-1} \), and not \( \mathbf{M} \) in equation (6). In the case of a single particle, we write \( \mathbf{\ddot{x}} = -\mathbf{M}^{-1} \mathbf{f} \) to describe a particle’s acceleration. When inverse mass is used, it becomes trivial to enforce constraints by altering the mass.

Suppose for example that we want to keep particle \( i \)’s velocity from changing. If we take \( 1/m_i \) to be zero, we give the particle an infinite mass, making it ignore all forces exerted on it. Complete control over a particle’s acceleration is thus taken care of by storing a value of zero for the particle’s inverse mass. What if we wish to constrain the particle’s acceleration in only one or two dimensions? Although we normally think of a particle’s mass as a scalar, we need not always do so. Suppose we write \( \mathbf{\ddot{x}} = \begin{pmatrix} 0 & 0 & 0 \\ 1/m_i & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \mathbf{f} \). Now \( \mathbf{\ddot{x}} \)
must lie in the xy plane; no acceleration in the z direction is possible. Note that an unconstrained particle can be considered to have the 3 × 3 inverse mass matrix \( -I \), with \( I \) the identity matrix.

Of course, we are not restricted to coordinate-aligned constraints. More generally, given a unit vector \( p \in \mathbb{R}^3 \), a particle is prevented from accelerating along \( p \) by using an inverse mass matrix \( \frac{1}{m}(I - pp^T) \); this follows from the fact that \( (I - pp^T)p = 0 \). Similarly, given two mutually orthogonal unit vectors \( p \) and \( q \), we prevent a particle from accelerating in either the \( p \) or \( q \) direction by using the inverse mass matrix \( \frac{1}{m}(I - pp^T - qq^T) \).

By allowing constrained particles to have these sorts of inverse masses, we can build constraints directly into equation (6). We will create a modified version \( W \) of \( M^{-1} \); \( W \) will be a block-diagonal matrix, with off-diagonal blocks being zero, and diagonal blocks defined as follows: let \( ndof(i) \) indicate the number of degrees of freedom particle \( i \) has, and let particle \( i \)'s prohibited directions be \( p_i \) (if \( ndof(i) = 2 \)) or \( p_i \) and \( q_i \) (if \( ndof(i) = 1 \)) with \( p_i \) and \( q_i \) mutually orthogonal unit vectors. \( W \)'s diagonal blocks are \( W_{ii} = \frac{1}{m_i}S_i \), where

\[
S_i = \begin{cases} 
1 & \text{if } ndof(i) = 3 \\
(I - p_i p_i^T) & \text{if } ndof(i) = 2 \\
(I - p_i p_i^T - q_i q_i^T) & \text{if } ndof(i) = 1 \\
0 & \text{if } ndof(i) = 0.
\end{cases} 
\] (13)

We are not limited to constraining particles to have zero accelerations in certain directions; rather, we control exactly what the change in velocity is along the constrained directions. Our idea then is to define a symmetric positive definite matrix \( A \) by

\[
A = \begin{pmatrix} 
M - \frac{\partial f}{\partial v} - h^2 \frac{\partial^2 f}{\partial x^2} 
\end{pmatrix} \Delta v = h \left( f_0 + h \frac{\partial f}{\partial x} v_0 \right) 
\] (15)

and the vector \( b \) and residual vector \( r \) as

\[
b = h \left( f_0 + h \frac{\partial f}{\partial x} v_0 \right) \quad \text{and} \quad r = A \Delta v - b.
\]

Given \( A \), \( b \), constraints on the particles, and \( z \), our modified CG method will try to find \( \Delta v \) that satisfies two conditions:

- For each particle \( i \), the component of \( r \) in the particle’s unconstrained direction(s) will be made equal to zero (assuming the method is run for sufficiently many iterations).
- For each particle \( i \), the component of \( \Delta v \), in the particle’s constrained direction(s) will be exactly \( z_i \) (no matter how many iterations are taken).

Note that these two conditions imply that unconstrained particles have \( r \) close to zero, while completely constrained particles have \( \Delta v = z \). Thus in the case when no particles are constrained, our modified CG method should produce the same result as the regular CG method.

### 5.3 The Modified Conjugate Gradient Method

The CG method (technically, the preconditioned CG method) takes a symmetric positive definite matrix \( A \), a vector \( b \) and iteratively solves \( A \Delta v = b \). The iteration stops when \( \|b - A \Delta v\| < \epsilon \|b\| \) where \( \epsilon \) is a user-defined tolerance value. The preconditioning matrix \( P \), which must be easily invertible, speeds convergence to the extent that \( P^{-1} \) approximates \( A \). We wholeheartedly refer the reader to Shewchuk [14] for information on the CG method.

We derive our modified conjugate gradient method by observing that the effect of the matrix \( W \) in equation (14) is to filter out velocity changes in the constrained directions. Our idea then is to define an invariant— for all \( i \), the component of \( \Delta v \), in the constrained direction(s) of particle \( i \) is equal to \( z_i \)—and then establish and maintain the invariant at each iteration, by defining a filtering procedure.

The role of \( \text{filter} \) is to take a vector \( a \) and perform the same filtering operation (see equation (13)) as multiplying by \( W \), but leaving out the scaling by \( 1/m \):

\[
\text{procedure filter}(a) \\
\quad \text{for } i = 1 \to n \\
\quad \quad \hat{a}_i = S_i a_i \\
\quad \text{return } \hat{a}
\]
Using filter, we define the modified CG method modified-pcg as follows:

1. procedure modified-pcg
2. \[ \Delta \mathbf{v} = z \]
3. \[ \delta_0 = \text{filter}(b)T \text{filter}(b) \]
4. \[ \mathbf{r} = \text{filter}(b - A \Delta \mathbf{v}) \]
5. \[ c = \text{filter}(P^{-1} \mathbf{r}) \]
6. \[ \delta_{\text{new}} = \mathbf{r}^T c \]
7. while \( \delta_{\text{new}} > e^2 \delta_0 \)
8. \[ q = \text{filter}(Ac) \]
9. \[ \alpha = \delta_{\text{new}} / (c^T q) \]
10. \[ \Delta \mathbf{v} = \Delta \mathbf{v} + \alpha c \]
11. \[ \mathbf{r} = \mathbf{r} - \alpha q \]
12. \[ s = P^{-1} \mathbf{r} \]
13. \[ \delta_{\text{add}} = \delta_{\text{new}} \]
14. \[ \delta_{\text{new}} = \delta_{\text{new}} + \delta_{\text{add}} \]
15. \[ c = \text{filter}(s + \delta_{\text{new}} / \delta_{\text{std}} e) \]

Line 2 of the procedure establishes our invariant. Lines 5 and 15 maintain the invariant by filtering \( e \) before adding it to \( \Delta \mathbf{v} \). The unmodified conjugate gradient method establishes a stopping criterion based on \( b^T Pb \). Since our constrained formulation ignores certain components of \( b \), our stopping criterion should as well, so we add filtering to line 3. The vector \( \mathbf{r} \) measures the solution error \( b - A \Delta \mathbf{v} \) and should not include error due to the constraints; hence we add filtering at lines 4 and 8. (Note that removing the calls to filter and changing line 2 to \( \Delta \mathbf{v} = 0 \) yields the standard preconditioned conjugate gradient method.)

We use a simple preconditioner \( P \) by making \( P \) be a diagonal matrix with \( P_{ii} = 1/A_{ii} \) so products involving \( P^{-1} \) are trivially computed. More elaborate preconditioners could be used, though we doubt there is a large speedup to be gained. Matrix-vector products with \( A \) are of course implemented in sparse matrix-vector fashion, using the data structures defined in section 2.3.

Given modified-pcg, obvious questions are “does it work?” followed by “how does it compare with the unmodified CG method?” Proofs about CG methods are difficult in general; in practice, our method always converges, which answers the first question. Prior to implementing modified-pcg, we used a penalty method and applied the standard CG method to equation (15). When we began using procedure modified-pcg, we did not notice any substantial change in the number of iterations required by the method. Empirically, we conclude that the two methods have similar convergence behavior. Result in section 8 indicate that the running time is close to \( O(n^{1.5}) \), which is what unmodified CG would be expected to deliver on this sort of problem [14].

6 Collisions

Much has been written about collision detection for cloth; we have nothing substantial to add to the subject of collision detection per se. Cloth/cloth collisions are detected by checking pairs \((p, t)\) and \((e_1, e_2)\) for intersections, where \( p \) and \( t \) are a cloth particle and a cloth triangle respectively, and \( e_1 \) and \( e_2 \) are edges of cloth triangles. Given a previous known legal state of the cloth, we postulate a linear motion for the cloth particles to the current (possibly illegal) state and check for either particle/triangle or edge/edge crossings. To avoid \( O(n^2) \) comparisons, we use a coherency-based bounding-box approach [1] to cull out the majority of pairs.

When collisions between a cloth vertex and triangle, or two cloth edges are detected, we insert a strong damped spring force to push the cloth apart. A dissipative force tangent to the contact is also applied, counteracting any sliding motion. The force is not, strictly speaking, a frictional force: rather it is proportional to the slip velocity, so it is in actuality a damping force, although it reasonably emulates dynamic friction. Applying static friction forces to cloth contacts is far more difficult, and is a problem we have not solved yet. The forces, and their derivatives with respect to position and velocity, are of course included in equation (15).

Our system detects collisions between cloth particles and solid objects by testing each individual cloth particle against the faces of each solid object. A solid object’s faces are grouped in a hierarchical bounding box tree, with the leaves of the tree being individual faces of the solid. The tree is created by a simple recursive splitting along coordinate axes. The maintenance of contacts and the application of friction forces was described in the previous section.

6.1 Constraint Initiation

Both cloth/cloth and cloth/solid collisions give rise to the same problem whenever two contacts form. For both types of collisions, our detection algorithm reports an intersection, and then takes action to remedy the situation: either by enforcing a constraint (cloth/solid collisions) or by adding a penalty force (cloth/cloth) collisions. However, since our simulator proceeds in discrete steps, collisions resulting in a reasonably substantial interpenetration depth can occur between one step and the next. Clearly, this situation needs to be remedied.

For cloth/cloth collisions, this would not appear to be a problem: the spring forces that are added work to counter the colliding velocities and then push the cloth apart. For cloth/solid collisions, however, the situation is more complicated. If we simply enforce a constraint which causes the colliding cloth particle to have a velocity consistent with the solid object’s velocity, and continue to enforce that constraint, the cloth particle will continue to remain embedded somewhere below the solid object’s surface. This is unacceptable.
One solution is to use Baumgarte stabilization [18], which schedules the particle’s acceleration so that the position and velocity error of the particle with respect to the surface decay asymptotically to zero. We experimented with this technique, but found it lacking. In particular, a fast rise to the surface was prone to noise and “jumpiness”; this could be eliminated, but at the cost of decreasing the step size. A slower rise to the surface caused visual artifacts.

We tried a simpler solution: when intersections occurred, rather than wait for a scheduled constraint or a penalty force to eliminate the intersection, we simply altered the positions of the cloth particles, effecting an instantaneous (and discontinuous) change in position. While this would be problematic when using a multi-step differential equation solver which expects continuity (such as a Runge-Kutta method), it should not interfere with a one-step solver such as the backward Euler method. Unfortunately, simply changing particle positions produced disastrous results. The stretch energy term in a cloth system is extremely strong, and altering particle positions arbitrarily introduced excessively large deformation energies in an altered particle’s neighborhood. This resulted in visibly “jumpy” behavior of the cloth in localized regions.

6.2 Position Alteration

Despite its initial failure, the ability to make arbitrary small changes in a particle’s position continued to attract our attention. The entire process of implicit integration can be considered to be a filtering process [7], and we postulated that a mechanism for filtering energy changes caused by displacing particles might make position alteration a viable technique. We considered that perhaps some sort of extra implicit step could be used as a filter, but forming and solving an additional linear system at each step seemed too expensive. Happily, we can make use of the filtering effect of implicit integration without any extra work.

Consider a particle that has collided with a solid object. The particle’s change in velocity at each step is under our control, using the constraint techniques described in section 5. Meanwhile, the particle’s position at the next step follows from equation (4):

$$\Delta x_i = h(v_{0i} + \Delta v_i)$$

(recall that $v_{0i}$ is the particle’s current velocity). The reason that changing positions after a step has been taken doesn’t work is because the particle’s neighbors receive no advance notification of the change in position: they are confronted with the alteration at the beginning of the next step. This presents an obvious solution: we simply modify the top row of equation (4) to

$$\Delta x_i = h(v_{0i} + \Delta v_i) + y_i,$$  \hspace{1cm} (17)

where $y_i$ is an arbitrary correction term of our choice, introduced solely to move a particle to a desired location during the backward Euler step. Having modified the top row of equation (4), we must follow this change through: using equation (17) and repeating the derivation of section 3 and the symmetric transform from section 5 yields the modified symmetric system

$$\frac{1}{h} \left( M - h \frac{\partial f}{\partial v} - h^2 \frac{\partial^2 f}{\partial v^2} \right) \Delta v = h \left( f_0 + h \frac{\partial f}{\partial v} v_0 + \frac{\partial f}{\partial v} \right).$$ \hspace{1cm} (18)

This modification gives us complete control over both the position and velocity of a constrained particle in just one step, without any extra computational cost. We use this technique to bring particles quickly and stably to the surface of solid objects without creating visual artifacts or limiting the allowable step size. We can also add correction terms to particles involved in cloth/cloth collisions. Without a constraint on those particles’ velocities there is no guarantee that they will go exactly where we want in one step, but the ability to induce sizeable jumps in position without excessively stiff spring forces adds greatly to the stability of the simulation.

7 Adaptive Time Stepping

The methods introduced in all of the previous sections usually allow us to take sizeable steps forward, without loss of stability. Even so, there are still times when the step size must be reduced to avoid divergence. There are a large number of methods for altering the size of a time step, for both explicit and implicit integrators, but these methods tend to concentrate on the accuracy of the simulation, and not the stability. Our goal is animation, not engineering; thus visually pleasing results, meaning a numerically stable solution, rather than overall accuracy, is the deciding voice. The trick is to recognize instability before you see it on your screen—by then it’s too late.

Stiffness, and thus any potential instability, arises almost completely from the strong stretch forces in the cloth. After each implicit step, we treat the resulting $\Delta x$ as a proposed change in the cloth’s state, and examine the stretch terms (section 4.2) for each triangle in the newly proposed state. If any triangle undergoes a drastic change in its stretch (in either the $u$ or $v$ direction) we discard the proposed state, reduce the step size, and try again. Subtlety is not required: we find that an unstable step invariably results in stretch changes that are quite large, and are thus easily detected.

Our simulation is run with a parameter that indicates the maximum allowable step size: this parameter is set by the user, and is always less than or equal to one frame. (Most of our simulations involving human motions use a step size of 0.02 seconds.) Whenever the simulator reduces the step size, after two successes with the reduced step size the simulator tries to increase the step size. If the simulator fails at the larger step size, it reduces the size again, and waits for a longer period of time before retrying to increase the step size. At its limit, the simulator will try increasing the step size every 40 steps; thus, if the user chooses too large a step, the simulator settles down to wasting only one out of every 40 steps in attempting too large a step. This method, though simple, has served us well.

8 Results

Table 1 gives a performance summary of assorted animations, shown in figures 1–6. Unaccounted overhead of the simulation (typically about 5%) includes tasks such as geometry transformations, memory allocation, etc. The clothes in figures 3–6 were modeled as discrete planar panels, and then topologically sewed. The simulator was used to relax the clothing from an initial deformed state, that got the clothes around the characters, to a well-fitting state on the characters. The $b_u$ and $b_v$ parameters (see equation (10)) were then made smaller in certain regions to produce cuffs and waistbands, or strategically increased to induce wrinkling behavior in other regions.

We also ran the simulation in figure 1 with a range of stiffnesses for the bend term. Using the stiffness parameters in figure 1 as a reference, we ran the simulation with those bend stiffnesses multiplied by 0.1, 1.0, 10, 100 and 1,000 (for a total range of 10,000 in the stiffness). The variance in the running times was under 5%. We doubt that simulators based on explicit integration methods could make a similar claim.

Finally, we tried to estimate our simulator’s performance as a function of $n$, the number of cloth particles. We ran the simulation in figure 1 with cloth resolutions of 500, 899, 2,602 (shown in figure 1) and 7,359 particles. The running times were, respectively, 0.23 seconds/frame, 0.46 seconds/frame, 2.23 seconds/frame, and 10.3 seconds/frame. This is slightly better than $O(n^{1.5})$ performance, which is in line with the convergence rates of the conjugate gradient method [14] for systems such as equation (18).
<table>
<thead>
<tr>
<th>figure</th>
<th>no. vertices/no. triangles</th>
<th>time/frame (CPU sec.)</th>
<th>step size min/max (ms)</th>
<th>total frames/total steps</th>
<th>task breakdown percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2602/4944</td>
<td>2.23</td>
<td>16.5/33</td>
<td>75/80</td>
<td>25.7 50.4 18.3 1.4</td>
</tr>
<tr>
<td>2</td>
<td>2602/4944</td>
<td>3.06</td>
<td>16.5/33</td>
<td>75/80</td>
<td>17.9 63.6 15.3 0.2</td>
</tr>
<tr>
<td>3</td>
<td>6450/12654</td>
<td>7.32</td>
<td>16.5/33</td>
<td>50/52</td>
<td>18.9 37.9 30.9 2.6</td>
</tr>
<tr>
<td>4</td>
<td>6450/12654</td>
<td>14.5</td>
<td>2.5/20</td>
<td>430/748</td>
<td>16.7 29.9 46.1 2.2</td>
</tr>
<tr>
<td>5</td>
<td>8757/17352</td>
<td>38.5</td>
<td>0.625/20</td>
<td>430/1214</td>
<td>16.4 35.7 42.5 1.7</td>
</tr>
<tr>
<td>6</td>
<td>5108/101016</td>
<td>3.68</td>
<td>5/20</td>
<td>393/715</td>
<td>18.1 30.0 44.5 1.5</td>
</tr>
<tr>
<td></td>
<td>5,188/10194</td>
<td>16.7</td>
<td>5/20</td>
<td>393/701</td>
<td>11.2 26.0 57.7 1.3</td>
</tr>
<tr>
<td></td>
<td>4,330/8844</td>
<td>10.2</td>
<td>10/20</td>
<td>393/670</td>
<td>20.1 36.8 29.7 2.6</td>
</tr>
<tr>
<td></td>
<td>5,188/10194</td>
<td>16.6</td>
<td>1.25/20</td>
<td>393/753</td>
<td>13.2 30.9 50.2 1.4</td>
</tr>
</tbody>
</table>

Table 1: System performance for simulations in figures 1–6. Minimum and maximum time steps are in milliseconds of simulation time. Time/frame indicates actual CPU time for each frame, averaged over the simulation. Percentages of total running time are given for four tasks: EVAL—forming the linear system of equation (18); CG—solving equation (18); C/C—cloth/cloth collision detection; and C/S—cloth/solid collision detection.

9 Acknowledgments

This research was supported in part by an ONR Young Investigator award, an NSF CAREER award, and grants from the Intel Corporation. We thank Alias|Wavefront for supplying the models and motion capture data used in figures 5 and 6.

References


Figure 1 (top row): Cloth draping on cylinder; frames 8, 13 and 35. Figure 2 (second row): Sheet with two fixed particles; frames 10, 29 and 67. Figure 3 (third row): Shirt on twisting figure; frames 1, 24 and 46. Figure 4 (bottom row): Walking man; frames 30, 45 and 58.
Figure 5 (top row): Dancer with short skirt; frames 110, 136 and 155. Figure 6 (middle row): Dancer with long skirt; frames 185, 215 and 236. Figure 7 (bottom row): Closeups from figures 4 and 6.
Physical Model of Cloth I

Where to Begin?

• “Simulation is not reality.”
• “Simulation merely demonstrates the consequences of your assumptions.”

Alan H. Barr
California Institute of Technology
How to Simulate Cloth?

- “Cloth is not a continuous material. It is a complex mechanism”

  John Skelton
  Albany International’s Fabric Research Lab

Mechanical Complexity and Variety of Woven Fabrics

- Three Samples

<table>
<thead>
<tr>
<th>100% cotton</th>
<th>100% wool</th>
<th>Polyester/cotton blend</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="100% cotton sample" /></td>
<td><img src="image2" alt="100% wool sample" /></td>
<td><img src="image3" alt="Polyester/cotton blend sample" /></td>
</tr>
</tbody>
</table>
# Mechanical Properties of Cloth

- Are
  - Non-linear, orthotropic/anisotropic
  - Non-deterministic
  - Hysteretic
  - A nightmare

**Depend on**
- Thread type and weave
- Deformation history
- Temperature and humidity
- Phases of the moon

---

# Cloth Modeling Research

- **Two Communities**
  - Textile Science/Engineering (Peirce - 1937)
    - Low-level mechanical properties
    - CAD/CAM, industrial applications
    - Draping & macroscopic dynamics of cloth and clothing
    - Visual effects and animation
Peirce Model (1937)

- Geometric structure of yarn crossing
- 5 equations and 9 unknowns
- 4 parameters measured to solve system
- Used to predict maximum yarn packing

Peirce model extensions (1950s - 1990s)

- New solution techniques
- Non-circular, compressible threads
- Balance of forces on linearly-elastic rods
  Truly 3-dimensional
- Non-linear bending behaviors
- Hysteresis
- Threads of varying thickness
- Minimize total strain energy
Measuring Devices: The Drapemeter (1950)

\[ D = \frac{A - \frac{R_2^2}{R_1^2}}{R_2^2 - \frac{R_1^2}{R_2^2}} \]

- Quantify draping behavior
- Count number of lobes

Measuring Devices: The Kawabata Evaluation System

- Several devices for measuring the mechanical properties of cloth
- Bending, shearing, tensile, compression, etc.
- Output - graphs and constants
CG Cloth Models

- Weil (1986) - Geometric approach
- Feynman (1986) - Minimize strain energy
- Haumann (1987) - Mass-spring model
- Terzopoulos et al. (1987) - Integrate elasticity-based forces
- Magnenat-Thalmann et al. (1990 on) - Clothing animation
- Breen et al. (1991-94) - □-structure-based model
- Provot (1995-97) - Improved elastic model
- Eberhardt, Strasser et al. (1995 on) - Modeling knits
- Baraff & Witkin (1998) - Implicit integration
- Meyer et al. (2000) - Interactive “cloth”

Haumann (1987)

- Mass & Spring Model
  - Point masses
  - Edge & hinge springs
- Convert mesh into model
- Air resistance
- Collision detection
- Integrate forces acting on point masses
- Provot extensions (1995)
Terzopoulos et al. (1987)

- Based on elasticity & differential geometry
- \( f(r(a)) = \left\| G \right\|^2 + \left\| B \right\|^2 \int da, da_2 \)
- G & B - 1st & 2nd fundamental forms
  - G based on distance
  - B based on curvature

Discretize object
- Calculate forces
- Integrate forces

Terzopoulos et al. Results
N. Magnenat-Thalmann et al. (1990 - Present)

- Clothing virtual actors
- Utilize and extend Terzopoulos model
- Computational techniques
- Collision detection and response
- Design of a complete set of clothing
  - User interface
  - Data structures
- Numerous and real-time animations

Early Magnenat-Thalmann Results
Breen, House & Wozny (1991-94)

- Macroscopic behavior arises from modeling microscopic structure
- Network of interacting particles
- Each particle is based on thread-level interactions
  - Stretching, bending, trellising (shear) & gravity
- Minimize total energy, while maintaining collision constraints
- Energy based on Kawabata measurements
- Measure cloth properties and simulate it

**Cloth Particle Energy Functions**

I. Collision and Stretching

II. Bending

III. Trellising
Breen et al. Results
End of the Season

Breen et al. Results
Real vs. Virtual
Baraff and Witkin (1998)

- Rapid cloth simulation with implicit integration
- Instead of “blindly” stepping forward, solve system of equations to find a system state that points back to initial state
- Result - larger time steps, faster simulations (a few CPU-secs/frame)
- Cloth model defined for computational efficiency
Meyer, Debnunne, Desbrun & Barr -2000

• Interactive animation of cloth-like objects
• Simple mass-spring model
• Implicit Predictor-Corrector integration
• Angular momentum correction
• Enforce distance constraints between nodes

Advantages
– No need to solve system of equations
– Linear & angular momentum are conserved
– Interactive for 100 node sheet

Meyer et al. Results
Cloth Modeling Trends: More Complex Models

- Purely geometric drape (Weil)
- Continuum mechanics-based drape (Feynman)
- Mass-spring dynamic simulation (Haumann, later Provot)
- Continuum mechanics-based simulation (Terzopoulos, Magnenat-Thalmann)
- Microstructure-based drape (Breen)

Cloth Modeling Trends: More Complex Simulations

- More complex materials (Eberhardt & Strasser)
  - Wovens to knits
- More complex physical interactions (Ling)
  - Cloth in turbulent air
- Simulating complete sets of clothing (Magnenat Thalmann; Animation Industry)
- Coping with post-buckling instability (Choi & Ko)
Cloth Modeling Trends: Simplify Models for Interactivity

- Simplify Breen model
  - Eberhardt, Weber & Strasser
- Model defined for implicit integration
  - Baraff & Witkin
- Interactive mass-spring model
  - Meyer, Desbrun et al.
- Real-time clothing with mass-spring model and geometric deformations
  - Magnenat-Thalmann

Cloth/Clothing Modeling Books

D. House & D. Breen, eds., AK Peters, 2000

P. Volino & N. Magnenat-Thalmann, Springer, 2000
Predicting the Drape of Woven Cloth Using Interacting Particles

David E. Breen†1 Donald H. House‡2 Michael J. Wozny§3
†European Computer-Industry Research Centre
‡Visualization Laboratory, Texas A&M University
§National Institute of Standards and Technology

†formerly of Design Research Center, Rensselaer Polytechnic Institute
‡formerly of Department of Computer Science, Williams College

Abstract

We demonstrate a physically-based technique for predicting the drape of a wide variety of woven fabrics. The approach exploits a theoretical model that explicitly represents the microstructure of woven cloth with interacting particles, rather than utilizing a continuum approximation. By testing a cloth sample in a Kawabata fabric testing device, we obtain data that is used to tune the model’s energy functions, so that it reproduces the draping behavior of the original material. Photographs, comparing the drape of actual cloth with visualizations of simulation results, show that we are able to reliably model the unique large-scale draping characteristics of distinctly different fabric types.

Keywords: cloth, drape, physically-based modeling, particle systems, Kawabata Evaluation System.

1 Introduction

The vast number of uses for cloth are mirrored in the extraordinary variety of types of woven fabrics. These range from the most exquisite fine silks, to the coarsest of burlaps, and are woven from such diverse fibers as natural wool and synthetic polyester. Each of these unique fabrics has its own distinguishing characteristics, and is recognizable to the trained eye, perhaps most easily, by the way it drapes. It is not surprising, then, that image makers, designers, and engineers have had a keen interest in characterizing the draping properties of cloth.

In this paper we report on a new technique for reliably reproducing the characteristic drape of particular fabrics. Here, drape means the final configuration of a cloth placed over a solid object. We attempt to answer questions like “What would this shirt look like made from cotton rather than from polyester?” or “Would this dress have a more pleasing drape if made from silk rather than a light wool?” Our work on this problem began several years ago with the development of a theoretical model of woven cloth based on interacting-particle methods [8], that we used to model such complex draping configurations as those in Figure 1. More recently we have been working on a technique for using empirical data from the Kawabata Evaluation System [29] fabric measuring equipment to tune the model. With this technique we can now test a particular cloth sample, derive energy functions based on the sample’s nonlinear mechanical properties, and then use the model to reproduce the fabric’s characteristic large-scale draping behavior.

To date, most of the efforts to create a model of cloth have employed continuum mechanics, with simulations utilizing finite element or finite difference techniques. These models have provided less then satisfactory results when attempting to accurately reproduce the characteristic folds and buckles found in specific types of cloth. In the introduction to a 1978 study on textile mechanics Shanahan, Lloyd and Hearle [39] express the opinion that

“Because of the relative coarse structure of textile materials, . . . it might be more profitable . . . to use non-continuum systems directly in the problems of complex (fabric) deformation.”

Despite their reservations, they explored continuum methods for many years, but Hearle finally abandoned this approach, stating that [1]

“In dealing with 3-dimensional buckling of textile fabrics, neither the terminology nor the methodology of established (continuum) theory of bending plates and shells is of much help.”
2.1 Particle systems

Particle Systems were first used in computer graphics by Reeves in 1983 [36]. He defined a particle system model as “a cloud of primitive particles,” where each particle is generated into the system, moves, ages, and then dies. This work was later extended by Reeves and Blau [37], Fournier and Reeves [20], Sims [42], and many others to model such diverse phenomena as trees and grass, ocean spray, fireworks, waterfalls, fire, snowstorms and explosions. Reynolds [38], in his work on flocking behavior, greatly enhanced the power of the particle system as a modeling tool. He proposed the idea of coupling the particles so that they interact with each other as well as with their environment, and demonstrated that it is possible to exploit simple local rules of interaction between large numbers of simple primitives to produce complex aggregate behaviors. Miller and Pearce [32], Terzopoulos, Platt and Fleischer [46], and Tonnnesen [48] all explored coupled particle systems as a way to model liquid-like and melting materials. Miller et al. [33], Szeliski and Tonnnesen [44], and van Wijk [49] proposed particle interactions that are a function of direction, producing deformable sheets and surfaces of particles. Our own interest in coupled particle systems has lead to a variety of explorations into specialized modeling and visualization tools [6, 24, 25, 47], into computational issues [26, 27, 34], and CAD technologies [3, 4].

2.2 Cloth modeling

2.2.1 computer animation models

The first computer animation model of cloth was by Weil [50], who used a two-step geometric process to model a rectangular cloth hanging from several constraint points. Dhande et al. [17] present a hybrid drape model, which relates the parameters of a swept surface to fabric mechanical properties. Feynman [9] developed the first true physically-based cloth model. His model utilizes a set of energy equations based on the theory of elastic shells, distributed over a grid of points. Haumann and Parent [23] produced several cloth animations, including a flag waving and curtains blowing in a breeze. Terzopoulos and Fleischer [45] developed a wide range of models for computer graphics based on elasticity theory. Their finite difference and finite element simulations demonstrated 3-D cloth-like structures that bend, fold, wrinkle, interact with solid geometry, and tear. Others have extended their model to simulate complete sets of clothing [13], and how cloth responds to air flow [30]. Aono [2] also used elasticity theory to simulate ripples in cloth-like structures.

2.2.2 engineering and design models

The first cloth draping work published in the engineering community was by Shanahan et al. [39], who used the theory of sheets, shells and plates to characterize a matrix of elastic parameters for a sheet of material. Lloyd [31] later provided non-linear extensions to the matrix and used finite element methods to simulate a 3-D circular cloth being deformed by a projectile. Eischen et al. [18] modeled cloth structures using a large deformation beam and shell theory recently proposed by Simo et al. [41]. Collier et al. [15] present a finite element approach to modeling draping behavior. They also tested fabric using a drape measuring device called the Drapemeter [14], and showed that coefficients produced by their simulations compared favorably with measured values. As part of an apparel CAD system [35], Imaoka et al. [28] developed a continuum mechanics model of cloth based on the large deformation shell theory of Green and Zerna [22]. They also attempted to incorporate data from the Kawabata mechanical tester, but were unable to find a clear mapping of test data into their model.

3 A Particle-Based Model of Cloth

The fundamental principles of our particle-based model of cloth have been fully described elsewhere [8, 27], but they are briefly reviewed here, since a basic understanding is essential to the theme of this paper.

We model cloth as a collection of particles that conceptually represents the crossing points of warp and weft threads in a plain weave. Important mechanical interactions that determine the behavior of woven fabric occur at these points. Most significantly, the tension is typically so great at crossings that the threads are clamped together, providing an axis around which bending can occur in the plane of the cloth. Other more distributed interactions, such as stretching of threads and out of plane bending, can be conveniently discretized and lumped at the crossing points.

In the model, we represent the various thread-level structural constraints with energy functions that capture simple geometric relationships between particles within a local neighborhood. These
energy functions are meant to encapsulate four basic mechanical interactions: thread collision, thread stretching, out-of-plane bending, and trellising. These are shown graphically in Figure 3, and are captured in the energy equation for particle $i$,

$$U_i = U_{repel} + U_{stretch} + U_{bend} + U_{trellis} + U_{grav}.$$  \hspace{1cm} (1)

In this equation, $U_{repel}$ is an artificial energy of repulsion, that effectively keeps each particle at a minimum distance, providing some measure of thread collision detection, helping prevent self intersection of the cloth. $U_{stretch}$ captures energy of tensile strain between each particle and its four-connected neighbors. $U_{bend}$ is the energy due to threads bending out of the local plane of the cloth, and $U_{trellis}$ is the energy due to bending around a thread crossing in the plane. $U_{grav}$ 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-IIa). $U_{grav}$ is a function of the height of the particle. Trellising occurs when threads 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, but since our model treats cloth as an interwoven grid of threads, trellising is a more descriptive term.

We assume that the threads 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 effectively keep every other particle at a minimum distance, providing self intersection of the cloth.

The second phase performs an energy-minimization to enforce interparticle constraints. A stochastic element of the energy minimization algorithm serves to both 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. Even with these “sketched-in” energy functions, the simple interactions governing the particles aggregate to produce a macroscopic draping behavior that is convincingly close to that of cloth. We were able to produce visually satisfying results, such as the cloths draped over both the easy-chair and end-table in Figure 1, after just a few runs to tune constants.

An energy well is produced by directly coupling each particle with the stretching function $S$ only to its 4-connected neighbors, as given by

$$U_{stretch} = \sum_{j \in N_i} S(r_{ij}),$$  \hspace{1cm} (5)

where $N_i$ is the set of particle $i$’s four-connected neighbors.

The particle energy due to gravity is simply defined as

$$U_{grav} = m_i g h_i,$$  \hspace{1cm} (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-Ib 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

$$U_{bend} = \sum_{j \in M_i} B(\theta_{ij}),$$  \hspace{1cm} (7)

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 diagramed 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 this angle could easily change over the course of a simulation to model slippage. 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

$$U_{trellis} = \sum_{j \in K_i} T(\phi_{ij}),$$  \hspace{1cm} (8)

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 [27]. 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 to both 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. Even with these “sketched-in” energy functions, the simple interactions governing the particles aggregate to produce a macroscopic draping behavior that is convincingly close to that of cloth. We were able to produce visually satisfying results, such as the cloths draped over both the easy-chair and end-table in Figure 1, after just a few runs to tune constants.


4 The Kawabata Evaluation System

Even though early experiments confirmed that we could generate reasonable looking draping behavior, there were many things about our simulated cloth that we did not know. The model was not based on physical units, so we did not know the actual size of our simulated cloth sample, and we could not query the model for any kind of mechanical information. Most importantly, we did not have a methodical means of tuning the model to simulate particular kinds of cloth.

In order to tie the model directly to the draping behavior of actual cloth, we have developed a method for deriving the model’s energy equations from empirical mechanical data produced by the Kawabata Evaluation System [29]. This system is a standard set of fabric measuring equipment that can measure the bending, shearing and tensile properties of cloth, as well as its surface roughness and compressibility. For bending, shearing and tensile properties, the equipment measures what force or moment is required to deform a fabric sample of standard size and shape, and produces plots of force or moment as a function of measured geometric deformation. Since we assume that threads do not stretch significantly when a cloth is simply draping under its own weight, we make use only of the Kawabata bending and shear plots. There is, however, no reason why Kawabata tensile data could not be used if one wished to model fabric under tensile load.

The Kawabata bending measurement is done by clamping a 20 cm × 1 cm sample of cloth along both its long edges. The sample is then bent between the clamps, as diagramed in Figure 4a, and the moment necessary to accomplish the bending is recorded. A plot of bending moment \( M \) versus curvature \( K \) is produced by assuming that the 1 cm cross-section bends with constant curvature. The shearing measurement is done by applying a shearing force along one of the long edges of a 20 cm × 5 cm cloth sample, as diagramed in Figure 4b. A plot of the force \( F \) versus shear angle \( \phi \) is produced. By cutting samples out of the original cloth in two orthogonal directions, it is possible to measure bending and shear in both the warp and weft directions.

Kawabata bending and shear plots for the 100% cotton, 100% wool, and cotton/polyester samples of Figure 2 are shown in Figure 5. Each curve plots a full deformation cycle for fabric oriented in both the warp (solid curve) and weft (dashed curve) directions. The curves are produced by applying a force (or moment) in one direction, releasing the force, reversing the direction of the force, and releasing the force once again. The plots clearly show the hysteretic behavior of cloth – the path of deformation when the cloth is stressed is different from the path when the stress is released, producing a loop in the plot.

The plots for each type of cloth are obviously quite different. Although the shear plots differ little between warp and weft, there are dramatic differences between warp and weft bending for both the 100% cotton and the polyester/cotton materials. In the 100% cotton this is due to the doubling of weft threads, and in the polyester/cotton this is due to the differing warp and weft materials. The shallower shear curve for the 100% wool indicates that it will be the most limp and easily shaped.

We have found it useful to think of the bending and shearing as being divided into three loosely defined regions, a region of initial resistance to deformation, a region of low deformation, and a region of high deformation. The mechanical behavior of cloth throughout its range of deformation is non-linear, especially during initial deformation. The mechanical properties of cloth in the low-deformation region are usually well-behaved. This is why the assumption of linear elasticity made in continuum cloth models yields a reasonable “cloth-like” behavior. The mechanical properties in the initial-resistance and high-deformation regions are not as well-behaved, and defy a simple, general mathematical description. The Kawabata plots provide information only in the initial-resistance, and low-deformation regions.

5 Derivation of the Energy Equations

The process of generating particle energy functions for woven cloth from Kawabata data has three steps. First, we determine functions that approximate the Kawabata plots. Next, we relate these approximating functions to the model’s energy functions. This is crucial, since in the case of bending, the approximating functions relate bending moment to curvature, and in the case of trellising, they relate force to shear angle, but what is needed is energy as a function of the bending and trellising angles shown in Figures 3-II and 3-III. Finally, we scale the resulting equations so that they will produce energy values in standard physical units.
5.1 Approximating the Kawabata curves

For purposes of calculating drape, we assume that the hysteresis of cloth does not play an important role. Thus, in approximating the Kawabata curves, we look only at the first of the four stages of the deformation cycles shown in Figure 5.

The most convenient way to approximate these curves is with piecewise polynomial functions. This can be done in any number of ways, but the important thing is that we obtain reasonable approximating functions $M(K)$ for bending moment, and $F(\phi)$ for shear force. We decided to interpolate the inflection points of the curves using the lowest order polynomials that were practical, and the Kawabata plots are sufficiently simple that we were able to do this with quadratic and linear segments. In each case, we first fit a function to the outer, more stable segment of the Kawabata curve, then fit additional segments to the initial segment maintaining position and slope continuity at the segment boundaries, using standard interpolation techniques [12]. In general, the slope of the Kawabata plots is difficult to determine at the origin. Therefore, although the first segment must pass through the origin, we did not enforce any slope constraint there.

5.2 The bending energy equations

Within the low deformation region within a single thread, we assume that the theory of elastic bending beams [40] is applicable, and can be used to calculate the energy of bending. The strain energy $dU$ due to bending stored in a segment $dS$ of an elastic beam is given by

$$dU = \frac{M dS}{2 \rho},$$

where $M$ is the bending moment acting on the segment and $\rho$ is its radius of curvature, which is related to curvature $K$ by $K = 1/\rho$. Within our model, each particle is separated from its 4-connected neighboring particles by the equilibrium distance $\sigma$, which is a function of the grid dimensions and particle density. Therefore each particle represents a $\sigma \times \sigma$ square of cloth. This square of cloth can be thought of as a series of elastic beams (threads) lined parallel to each other. The energy of bending in one of these threads is defined by the integral

$$U = \int_0^\sigma \frac{M(K) dS}{2 \rho}.$$

We assume that within one $\sigma \times \sigma$ patch that the moment and the curvature are constant, simplifying the energy equation for a single thread to

$$U = \frac{MK}{2 \sigma}.$$

Since $M$ is given in units of moment per unit width of sample, we can simply multiply Equation 11 by the width $\sigma$ of each patch, in order to sum up the contributions of each beam (thread) within the $\sigma \times \sigma$ patch. The energy of bending in just one direction then becomes

$$B = \frac{MK}{2 \sigma^2},$$

for each particle. This calculation is performed twice for each particle, once for bending in the warp direction, and once for bending in the weft direction. Recall that our approximation of the Kawabata bending plot provides bending moment $M$ as a function of curvature $K$, so that equation 12 yields bending energy $B$ as a function only of curvature. Thus, we need only relate curvature to the bending angles of the model.

Curvature, along a thread, at the position of a single particle can be approximated by assuming that the curvature is constant from the particle to its two neighbors. Given this assumption, a circle can be fit to the three points and the circle’s curvature can be calculated [5]. Unfortunately, this assumption becomes poor as the bending angle $\theta$ becomes small (i.e. as the threads bend in on one another). We would like the curvature $K$ to become arbitrarily large for small bending angle, in order to give reasonable high-deformation behavior. As an approximation, we fit the curve $\alpha/\theta + \beta$ to the small-angle portion of the curvature equation to yield the complete curvature equation

$$K(\theta) = \left\{ \begin{array}{ll} \frac{2}{\theta} \cos(\theta/2), & \frac{\pi}{4} < \theta \leq \pi \\ -\frac{\beta}{\theta} + \alpha + \frac{\pi}{4}, & 0 \leq \theta \leq \pi/4. \end{array} \right. \quad (13)$$

where the constants $\alpha = \frac{2}{\theta} \cos(\pi/8)$ and $\beta = \frac{\pi}{8} \sin(\pi/8)$ were chosen to maintain $C_0$ and $C_1$ continuity at $\theta = \pi/4$.

5.3 The trellising energy equations

We can calculate the energy stored in the 20 cm $\times$ 5 cm cloth sample that is sheared in a Kawabata Shear Tester from the work $W$ produced by a force $F$ acting over a displacement $dS$,

$$W = \int F dS. \quad (14)$$

If we assume that the width $l$ of the sample remains constant during shearing, then the path traveled by the point at which the shearing force is applied is a circular arc whose length is defined by $S = l\phi$, where $\phi$ is the shearing angle. If the force point is moving along a circular arc, the component of the applied shearing force in the direction of motion is $F \cos(\phi)$. Applying these results to Equation 14 yields the equation for shearing energy as a function of shear angle first derived by Cusick [16],

$$T = \int F \cos(\phi) l d\phi. \quad (15)$$

Recall that the Kawabata shearing plots provide the shearing force $F$ as a function of shearing angle $\phi$. Therefore, substituting the approximating equation for $F$ into Equation 15 and integrating yields the required energy of shearing strictly as a function of angle.

Once again we are faced with the problem of defining energy curves in the high deformation region not covered by the Kawabata data. Skelton [43] states that most woven materials cannot shear more than 45°. We approximate this constraint by introducing a singularity in the trellising energy curve at about 60°. The extra 15° permits the material to shear all the way to 45° if necessary. This singularity is introduced by fitting the function $a/(1.05 - \phi) + b$ to the slope and position at the endpoint of the Kawabata-derived energy curve (the magic number 1.05, is simply a rough approximation to $\pi/3 = 60^\circ$).

5.4 Scaling the energy equations

Up to this point no attention has been paid to the physical units of the energy equations, although we would like them to be in CGS units.

The Kawabata bending plots (see Figure 5) give moment $M$ in units of gf-cm $/$cm and curvature $K$ in cm$^{-1}$. If we take interparticle distance $\sigma$ to be in cm, then from equation 12 we see that the units for bending energy are gf-cm. Thus, scaling equation 12 by 978.80 will yield energy in ergs.

The Kawabata shearing plots (see Figure 5) give shearing force in units of gf/cm. Scaling this force by 978.80 yields dynes/cm, and multiplying the measured force by the 20 cm length of the sample.
gives force in dynes. Since the trellising energy is produced by a moment, the energy unit is dyne-cm or ergs. Since one particle represents a \( \sigma \times \sigma \) patch of cloth, we want the trellising energy per unit area. This can be computed by dividing the total energy stored in the 20 cm \( \times \) 5 cm sample by the area of the sample. Therefore the scale factor that converts energy \( T \) of equation 15 into trellising energy in ergs is 195.76\( \sigma^2 \).

6 Experimental Results

We derived energy equations for the 100% cotton, 100% wool, and polyester/cotton cloth samples shown in Figure 2 using the Kawabata data shown in Figure 5. A full derivation may be found in [10].

Given the energy equations for all three samples, we performed two sets of experiments to verify that the model was able to capture the characteristic draping behavior of each type of cloth. The first experiment was to drape real cloth over a cube and then perform the same drape in simulation, using computer visualizations of the simulation results to make visual comparisons with the actual cloth. The second experiment was to recreate the Kawabata Bending and Shear Testers in simulation, to show that the model can accurately reproduce physical measurements. Results of the first set of experiments are of most interest to the computer graphics community and are detailed below. The second set of experiments produced excellent results that are of most interest to the engineering and design community, and are detailed in [10].

In the actual draping experiments, 1 m \( \times \) 1 m sections of our three cloth samples were draped over a 0.5 m \( \times \) 0.5 m \( \times \) 0.5 m cube. The results of these drapings were photographed and are presented in the left column of Figure 8. The same scenario was recreated in simulation. Models of 1 m \( \times \) 1 m samples of 100% cotton, 100% wool and polyester/cotton, represented by a 51 \( \times \) 51 particle grid, were draped over a 0.5 m \( \times \) 0.5 m \( \times \) 0.5 m geometric model of a cube. Each simulation started with a flat cloth positioned just above and centered over the cube. The simulation was allowed to run until the cloth had draped over the cube and had settled into an equilibrium position. Equilibrium was judged manually, by examining the maximum particle movement between successive time steps. The final drapings produced by the simulations are shown in the computer graphic visualizations in the right column of Figure 8. Camera positions were chosen to accentuate the unique draping characteristics of each type of cloth.

The similarities between the actual and simulated drapings are quite evident in Figure 8. Each kind of material has a characteristic drape that is captured by the simulation. Both the actual and simulated 100% cotton develop a single large billow that comes out from the corner of the cube at a 45° angle. By contrast, the bending stiffness of the wool sample is significantly weaker than the cotton's. Therefore, it does not have the bending strength needed to support a single large billow and the corner structure collapses into two smaller folds, as seen in both the actual and simulated views. In the polyester/cotton material, the bending stiffness is significantly stronger in the warp than in the weft direction. The effect on the draping of the cloth can be clearly seen. Since bending is so much stronger in one direction than the other, the billow is literally pushed around the corner by the warp threads. This produces an asymmetric structure that wraps around the corner of the cube, as can be seen by comparing the front and side views.

7 Discussion

The draping experiments show that the model can be used to reproduce the large-scale draping behavior of specific types of cloth, but the cloth simulations do not exactly produce the drape of the actual cloth in Figure 8. This, in a sense, would be impossible, since cloth will never drape twice in exactly the same way. Instead, what we found when working with real materials is that each material does have its own “preferred” draping tendency. For example, at the corner of the cube, the cotton sample usually produced a single draping structure, the wool would form either a single fold or would collapse into more than one fold, and the polyester/cotton always produced an asymmetric fold. Of course, each material could be forced into many kinds of draping configurations, but when allowed to drape naturally, they generally produced their own characteristic structures.

Another difference evident in Figure 8 between the simulated and real drapings is in the sharpness of edges and corners. The simulated samples appear to have “soft” folds, as if being draped over a rounded cube. These differences are related to the fineness of the particle grid. A 51 \( \times \) 51 grid is capable of reproducing large-scale draping structures, but it is not sufficient for capturing the sharp bends over the edges or at the corners of a cube. We believe that utilizing a finer particle grid will remove these differences, and are currently working on an adaptive scheme that will sample a region of the cloth more or less finely based on the region’s total energy.

The computational speed of our implementation is currently its major drawback. Each simulation, starting with a flat cloth placed above the cube and falling to its final draped configuration, required about 1 CPU-week on an IBM RS/6000 workstation. The issue of speed is one that we chose to ignore for a period while developing and proving the model. There are several ways to improve speed, the most obvious being to write custom simulation code. Currently we work in an object-oriented, message-passing environment that has been excellent for rapid prototyping, but entails a heavy overhead [7, 21]. A more fundamental speed improvement could be had by improving the simulation technique. We currently use a stochastic method that follows a numerically-determined approximation to the energy gradient at each particle [27]. We have begun work to improve the efficient implementation of our model, that uses pre-calculated tables to more exactly and efficiently determine energy gradients, and are experimenting with a pure gradient-descent approach to energy minimization. Preliminary results indicate speed-ups well beyond an order of magnitude. Another approach would be to use an approximate, purely-geometric predrape, followed by the physical simulation to perfect the drape. This has already been tried with success by others [35, 51]. Finally, parallelism has been shown to be an especially efficient way of computing uncoupled particle systems [42]. The highly distributed form of our particle model, with its simple local computations and well defined neighbor interactions, should also be especially amenable to this approach [26].

To designers, one of cloth’s most important characteristics is its ability to be shaped and creased. Since we ignore hysteresis, our model, as it stands, is conservative – no energy is lost during a deformation. One consequence of this is that we cannot yet mimic self-shaping and creasing. This has not been an important issue in the kinds of free-draping studies that we have conducted, but would be of very great importance when looking at fabric under the high stresses that occur in manufacturing. It should, however, be relatively easy to extend our model in a natural way to simulate non-conservative deformation. The stretching, bending, and trellising energy functions all either explicitly or implicitly represent a “rest” value for their independent variables. It would be straightforward to represent all of these rest values explicitly, and then vary them as a function of local strain, thus mimicking the effects of slippage within the weave. This could be put on a firm physical basis, at least for the low deflection region, by adjusting “slippage” so that the full hysteresis curves from the Kawabata tester are matched.
8 Conclusion

We have presented a particle-based model capable of being tuned to reproduce the static draping behavior of specific kinds of woven cloth. There are several significant aspects to the work. It has demonstrated that a macrostructural model may be used to reproduce the macroscopic mechanical behavior of real flexible materials. It has shown that the use of such an approach can allow for the straightforward incorporation of non-linear empirical test data. The model has been verified by experiments. One generates the low-level mechanical properties of real fabrics, and the other recreates the distinctive macroscopic geometric structures of draping cloth and compares them to actual cloth drappings. This kind of evidence has not been presented in previous cloth modeling studies. With this approach, real materials may now be measured, and the measured data used to derive energy equations, allowing the draping behavior of specific materials to be confidently simulated on a computer.

Acknowledgements

Figure 1 was produced by Gene Greger and David Breen. The Kawabata measurements were provided by Ms. Janet Bulan-Brady and Mr. Herbert Barndt of the Grundy Center for Textile Evaluation at the Philadelphia College of Textiles and Science. We would like to thank Masaki Aono for translating reference [28]. Charles Gilman for assisting with the derivation of the bending equations, and David Gordon for assisting in the construction of the Drape-O-Matic Cube. This work was partially supported by the Industrial Associates Program of the Rensselaer Design Research Center, DLA contract No. DLA900-87-D-0016, and NSF grant No. CDR-8818826.

REFERENCES


Figure 6: Actual (left) vs. simulated (right) cloth drape
References


[Euler] Euler, L., Methodus inveniendi lineas curvas maximi minive proprietate gaudentes (Aditamentum I, De curvis elasticis), Lausanne, 1744; also in L. Euleri Opera Omnia, Vol. I/24, Fussli, Zurich, 1952.


Physical Model of Cloth II

- Understands interacting particles model can have a physical instability.
- Understands the nature of post-buckling instability.
- Understands cloth specific handling of post-buckling.
  - “immediate buckling assumption”
Stable but Responsive Cloth?

- Unstable Cloth
- Stable but Non-Responsive Cloth
- Responsive Cloth
- Stable but Responsive Cloth?

Stability & Responsiveness do not GET ALONG!

- Damping can Help Stability
- Damping Kills Responsiveness
What is the physical model behind cloth material?

Related Work

<table>
<thead>
<tr>
<th>Physical Model</th>
<th>Collision Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Terzopoulos and Fleischer, 1987</td>
<td>Provot, 1997</td>
</tr>
<tr>
<td>Carignan et al., 1992</td>
<td>Volino et al., 1995, 2000</td>
</tr>
<tr>
<td>Breen et al., 1994</td>
<td>Etzmuss et al., 2000</td>
</tr>
<tr>
<td>Volino et al., 1995</td>
<td>Zhang and Yuen, 2000</td>
</tr>
<tr>
<td>Eberhardt et al., 1996</td>
<td>Huh et al., 2001</td>
</tr>
<tr>
<td>Eischen et al., 1996</td>
<td>Bridson and Fedkiw, 2002</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Implicit Integration</th>
<th>Interactive Simulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baraff and Witkin, 1998</td>
<td>Provot, 1995</td>
</tr>
<tr>
<td>Volino and Thalmann, 2000</td>
<td>Desbrun, 1999</td>
</tr>
<tr>
<td>Hauth and Etzmuss, 2001</td>
<td>Meyer et al., 1999</td>
</tr>
<tr>
<td></td>
<td>Kang et al., 2000, 2002</td>
</tr>
</tbody>
</table>
Physical Model of Cloth Material

- Weak resistance to bending
- Strong resistance to tension
- What about to compression?
  - Compression stiffness ≅ Tension stiffness
  - Hookean model (Linear elastic model)
    \[ \sigma = E\varepsilon, \quad f = k(x - l) \]

Why Cloth Simulation is Unstable when Wrinkles are Formed?

An Experiment

- Two Step Process
  - Resist + Buckle

- How was the Buckling?
  - Explosive/Unstable

- Why? \( \text{strong resistance to compression} \) + \( \text{weak resistance to bending} \)

It is a Physical Instability!
It’s not a Numerical Instability
Interesting Findings

Will Cloth Behave Same Way?

- Previous physical model is suited for paper
- suited for aluminum sheet
- but not suited for cloth
- buckling of cloth is different

“Buckling”

Force : $P$

Compressed Length
Effect on Implicit Method

\[
\left( \frac{I}{\Delta t^2} + K \right) \Delta x_n = f_{n-1}
\]

- Ill-conditioned or indefinite system matrix \(\rightarrow\) Divergence!

Stability can be Improved

\[
\left( \frac{I}{\Delta t^2} + \frac{D}{\Delta t} + \alpha K \right) \Delta x_n = f_{n-1}
\]

- Reducing time step \(\rightarrow\) Longer comp time
- Reducing stiffness \(\rightarrow\) Stretchy material
- Adding damping \(\rightarrow\) Non-responsive cloth!
What We Propose:

• Find a Physical Model that just makes $K$ always positive definite!

\[
\left( \frac{I}{\Delta t^2} + \frac{D}{\Delta t} \right) K \Delta x_n = f_{n-1}
\]

How Can We Do That?

• Compression causes bending rather than shortening

Immediate Buckling Assumption
Our Observation

New Force-Displacement Curve
The Stuff We Get

- A model that does not have buckling instability
  - Large time steps / Coarse meshes
- No Fictitious Damping is Needed
  - Realistic Cloth

Results

# of particles = 3456, $\Delta t=0.011s$, CPU=0.25s
Leaping

# of particles = 5579, $\Delta t=0.011s$, CPU=0.47s

Walking

# of particles = 5608, $\Delta t=0.011s$, CPU=0.51s
Comparison

Fictitious damping
Adaptive time step size
Calculation time : 4 hours

No fictitious damping
Fixed time step size
Calculation time : 19 min

Particle Connections

• Sequential connection of elements
Particle Connections

• Sequential connection of elements
  • Global bending is not handled

Solution: Interlaced connection
Conclusion

• Post-buckling instability of previous physical model was an answer to: “Why Wrinkling is Unstable?”

• Immediate buckling assumption forms a “Realistic & Stable & Fast” Solution

Jeans

# of particles = 6624, Δt=0.011s, CPU=0.58s
Stable but Responsive Cloth

Kwang-Jin Choi  Hyeong-Seok Ko

Graphics and Media Lab
Seoul National University

Abstract

We present a semi-implicit cloth simulation technique that is very stable yet also responsive. The stability of the technique allows the use of a large fixed time step when simulating all types of fabrics and character motions. The animations generated using this technique are strikingly realistic. Wrinkles form and disappear in a quite natural way, which is the feature that most distinguishes textile fabrics from other sheet materials. Significant improvements in both the stability and realism were made possible by overcoming the post-buckling instability as well as the numerical instability. The instability caused by buckling arises from a structural instability and therefore cannot be avoided by simply employing a semi-implicit method. Addition of a damping force may help to avoid instabilities; however, it can significantly degrade the realism of the cloth motion. The method presented here uses a particle-based physical model to handle the instability in the post-buckling response without introducing any fictitious damping.

Keywords:  Deformations, Numerical Analysis, Physically Based Animation, Physically Based Modeling

CR Categories:  I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation; G.1.7 [Numerical Analysis]: Ordinary Differential Equations—Convergence and stability; Stiff equations I.6.5 [Simulation and Modeling]: Model Development

1 Introduction

A normal outfit covers more than 90 percent of the human body. Therefore, the realistic animation of cloth is imperative if we are to animate humans to a satisfactory level of detail and realism. Over the last decade a great deal of research has been dedicated to simulating cloth motion [Terzopoulos and Fleischer 1988; Carignan et al. 1992; Breen et al. 1994; Courschesnes et al. 1995; Provot 1995; Eberhardt et al. 1996; Eischen et al. 1996; Baraff and Witkin 1998; Desbrun et al. 1999; Volino and Magnenat-Thalmann 2000]. All of the methods proposed to date boil down to numerically solving an ordinary differential equation, although they differ in regard to characteristics such as stability, allowed time step size, etc.

Cloth is characterized by strong resistance to stretch and weak resistance to bending, which leads to a stiff set of equations and thus prohibits the use of large time steps. However, cloth simulation techniques must be stable and fast if they are to be of practical use. Previous studies have shown that implicit methods are well suited to solving stiff equations with a reasonable step size, and successful results have been reported in [Baraff and Witkin 1998; Volino and Magnenat-Thalmann 2000]. Another property that is crucial to the appearance of fabrics in motion is their buckling behavior. The buckling of fabrics is the process by which wrinkles form, and leads to structures such as those shown in Figure 1. The buckling of textile fabrics has a quite different nature from solid materials [Amirbayat and Hearle 1989], thus animation of a cloth would not look natural without having such property. Nevertheless it has been largely ignored though its problematic nature (instability and non-linearity) was recently pointed out by [Eischen et al. 1996] and [Yu et al. 2000]. This paper presents a stable and practical solution to this problem.

The buckling of a thin material involves a very unstable state, regardless of whether it is rigid (e.g. aluminum sheet) or flexible (e.g. fabrics). When a compressive force is applied at the extremes of a thin material, it initially resists changing shape. As this force is increased, it eventually reaches the neutral equilibrium, the point at which an infinitesimal increase or decrease in the force bifurcates the situation in two radically different directions: increasing the force leads to an unstable post-buckling response whereas the system remains at stable equilibrium if the force is decreased. (Buckling will be described in detail in Section 3.2.) Given that buckling is a ubiquitous characteristic of fabrics, creating natural looking cloth in an animation is very difficult without a stable way to model this phenomenon.

The instability of the post-buckling response arises from a structural instability [Bathe 1996], not from the stiff equations. Therefore, the buckling instability cannot be overcome by simply employing implicit methods. Some cloth simulation techniques [Baraff and Witkin 1998; Volino and Magnenat-Thalmann 2000] avoid this instability by adding damping forces\textsuperscript{1}. The damping

\textsuperscript{1}Damping is an important concept in this work. The Implicit method has an intrinsic damping effect that comes from the formulation itself, which we refer to as \textit{artificial damping}. This damping is not related to the nature of the cloth. On the other hand, we refer to damping that is deliberately added to the formulation to simulate the nature of the cloth as \textit{material intrinsic damping}. A third kind of damping is sometimes added to enhance numerical stability. We refer to this as \textit{fictitious damping}. A damping term appearing
forces help to stabilize the physical system, or equivalently, they make the system matrix better conditioned and help to maintain positive definiteness in a semi-implicit formulation. However, the damping forces can significantly degrade the realism of the simulated cloth movement. For example, [Volino and Magenmat-Thalmann 2000] and [Volino and Magenmat-Thalmann 2001] found that the damping forces can lead to systems in which wrinkles will not form on the cloth surface, wrinkles resist disappearing, or even the fabrics resist falling under their own weight.

The artificial damping in implicit methods mainly affects the in-plane deformation of the cloth because the in-plane rigidity is much higher than the bending rigidity. Therefore, although artificial damping is expected to be partially responsible for the degradation of the quality of the out-of-plane movement of the cloth, we conjecture that the degradation in the quality mentioned above arises mainly from fictitious damping. The method we propose in this paper includes artificial damping (since it is an implicit method) and material intrinsic damping, but does not include fictitious damping. The need for fictitious damping is avoided through the use of the predicted static post-buckling response as an effective way to handle the instabilities associated with post-buckling situations. Because fictitious damping is not used, our method gives significantly more realistic cloth motion. This represents a significant step forward for the simulation of textile fabrics.

For solid materials, buckling signifies a failure and thus only the mechanics prior to buckling have been studied (e.g. determining the critical load on a column). Even in the study of textiles, there has been no significant result on the buckling process that can be applied to the dynamic simulation of cloth movement. Therefore, instead of physically simulating the unstable post-buckling dynamic response, we solve the instability problem by calculating the deformation energy of the shape at the predicted static equilibrium of the post-buckling state. We treat the numerical instability caused by stiff equations using implicit time stepping. Using the physical model outlined above and implicit time stepping, we could stably integrate the equation of motion with a large fixed time step and without the need for fictitious damping forces.

2 The Physical Model

Before we describe our particle-based physical model of fabrics, we discuss the problems in recently proposed physical models.

2.1 Problems in Previous Physical Models

Cloth is not a homogeneous continuum. Therefore modeling fabrics as a continuum and employing FEM or FDM has several potential drawbacks, a number of which are described in [Amirbayat and Hearle 1989]. One drawback of such methods is that they require a very fine meshing to produce large deformations. For a simulator to be practical in a computer graphics application, however, coarse discretization (about 1 cm spacing between the nodal points) should be allowed to guarantee reasonable performance. In the analysis of almost incompressible and/or thin materials such as cloth, a continuum formulation with the elements at this scale might produce highly erratic results in the stress and strain properties.

Before we describe our particle-based physical model of fabrics, we discuss the problems in recently proposed physical models.

2.2 Using Interacting Particles

In our search for a way to overcome the drawbacks of the physical models outlined above, we found that systems of interacting particles are better suited for generating large deformations and handling the buckling problem. The method presented here draws on the idea that inspired the work of [Breen et al. 1994], who first applied the particle model to the simulation of textile fabrics. However, our particle model is much simpler than that used by Breen et al. and the treatment of compression and bending deformation is quite different from their approach. In this section, we describe the connectivity of the mass points representing the cloth surface. The associated energy functions and their derivatives are presented in the next section.

We approximate a cloth with a quadrilateral mesh of particles; thus each particle can be indexed as $P(i, j)$. Figure 2(a) shows all the connections associated with a given center particle. With the exception of particles at the boundaries, where some connections are broken, every particle has the connectivity shown in Figure 2(a). Two types of particle interaction model are employed, which are...
referred to as type 1 and type 2.

The type 1 interaction model (red lines in Figure 2) is responsible for stretch and shear resistance. For this type of interaction, the particle \( P(i, j) \) is connected to \( P(i \pm 1, j), P(i, j \pm 1) \), and \( P(i \pm 1, j \pm 1) \). Such connections are referred to as sequential connections. The type 2 interaction model (blue lines in Figure 2) is responsible for flexural and compression resistance. For type 2 interaction, the particles connected to \( P(i, j) \) are \( P(i \pm 2, j), P(i, j \pm 2), \) and \( P(i \pm 2, j \pm 2) \). Note that type 2 connections are made with every other particle, leading us to refer to them as interlaced connections. Figure 2(b) illustrates the sequential and interlaced connections in a particular direction, in which \( S_1 \sim S_2 \) are sequential connections and \( I_1 \sim I_2 \) are interlaced connections.

### 3 Energy Functions and Derivatives

In this section we first describe the type 1 interaction model. We then elaborate on the concept of buckling and its profoundly different meanings in rigid materials and fabrics, after which we describe how this distinction is reflected in our handling of the post-buckling instability. Based on fabric-specific buckling behavior, we formulate the type 2 interaction model. Finally, we add a material intrinsic damping to those models.

In the description of the energy functions and their derivatives presented below, the distinctive features of our formulation are highlighted by comparing the results of the proposed model with those presented in [Baraff and Witkin 1998] and [Volino and Magnenat-Thalmann 2000].

#### 3.1 Type 1 Interaction

The type 1 interaction is represented by a simple linear spring model, for which the energy function for particles \( i \) and \( j \) is,

\[
E = \begin{cases} 
\frac{1}{2}k_i(|x_{ij}| - L)^2 & : |x_{ij}| \geq L \\
0 & : |x_{ij}| < L 
\end{cases} 
\]  

(1)

where \( x_{ij} = x_j - x_i, L \) is the natural length, and \( k_i \) is the spring constant. Note that this energy function accounts for the tension only. The force acting on particle \( i \) due to the deformation between the two particles is,

\[
f_i = -\frac{\partial E}{\partial x_i} = \begin{cases} 
k_i(|x_{ij}| - L)\frac{x_j}{|x_{ij}|} & : |x_{ij}| \geq L \\
0 & : |x_{ij}| < L 
\end{cases} 
\]  

(2)

The Jacobian matrix of the force vector is

\[
\frac{\partial F}{\partial x_j} = \begin{cases} 
k_i x_{ij}x_j^T & = k_i(1 - \frac{L}{|x_{ij}|})(I - \frac{x_{ij}x_j^T}{x_{ij}^T x_j}) & : |x_{ij}| \geq L \\
0 & : |x_{ij}| < L
\end{cases}
\]  

(3)

The first term tells us that the stiffness in the direction of the spring interaction is constant, which is an obvious consequence of modeling the interaction using a linear spring. The second term tells us that the stiffness orthogonal to the interaction direction is proportional to \( (1 - \frac{L}{|x_{ij}|}) \). If we consider the 2-dimensional structure of the cloth, the direction orthogonal to all interactions corresponds to the out-of-plane direction. When the spring is stretched, in an implicit formulation, the second term plays an important role in stabilizing the spring because it introduces large positive eigenvalues of the system matrix in that direction.

First, we analyze the buckling of a rigid material in terms of solid mechanics. We then contrast this phenomenon with the buckling of fabrics. Consider the idealized rigid column shown in Figure 3(a), which consists of two rigid bars connected at point \( C \) by a rotational spring of stiffness \( k_\theta \). The postbuckling behavior in rigid materials is quite destructive, while the same behavior in textile fabrics naturally evolves into the shapes that are the essence of a fabric's appearance. We refer to [Amirbayat and Hearle 1989] for this fabric-specific property. Although there are clear distinctions between the buckling behavior of fabrics and rigid materials, they are not reflected in most existing cloth simulation techniques. Since our solution to the post-buckling instability is based on the distinguished feature of fabric buckling, we clarify the concept of buckling in this section.

First, we analyze the buckling of a rigid material in terms of solid mechanics. We then contrast this phenomenon with the buckling of textile fabrics. Consider the idealized rigid column shown in Figure 3(a), which consists of two rigid bars connected at point \( C \) by a rotational spring of stiffness \( k_\theta \). In this configuration, the bending resistance is condensed at point \( C \). Now, suppose an axially compressive force \( P \) is applied at point \( A \). Since the rotational spring resists bending, the two bars remain straight at equilibrium (Figure 3(b)). Now, suppose that the structure under force \( P \) is disturbed by an external force that causes a small lateral movement of point \( C \) (Figure 3(c)). The compressive force \( P \) will try to increase the
lateral displacement, while the rotational spring will try to restore the system to the original straight position. Suppose that the disturbance is removed at this moment. If $P$ is relatively small, the bars will return to the straight position (i.e. the structure is stable). However, if $P$ is relatively large the lateral displacement will become larger and larger (i.e. the structure is unstable), and the structure will eventually collapse by lateral buckling. The magnitude of the axial force at which the structure bifurcates into either the stable or unstable condition by application of an infinitesimal increase or decrease in force is known as the critical load and is denoted by $P_{cr}$. At the critical load, the deflection of the structure is mathematically arbitrary. Once the axial load exceeds $P_{cr}$, the structure collapses and the original shape cannot be recovered, regardless of the degree to which the load exceeds $P_{cr}$, and no matter how quickly the load is reduced back to a value less than or equal to $P_{cr}$.

Now, let us consider the behavior when a fabric buckles. As for rigid materials, a textile fabric will buckle when subjected to an axial force greater than the critical load. When it buckles, it exhibits an unstable post-buckling response similar to that found in rigid materials. In contrast to rigid materials, however, fabrics do not break or collapse. Instead, they quickly pass the unstable state and reach a stable equilibrium (a smoothly bent shape). Moreover, the bent shape tends to return to the original straight shape when the axial load is removed [Amirbayat and Hearle 1989].

As described above, textile fabrics pass through an unstable state when they buckle. The simulation of this unstable post-buckling response requires special care if divergence problems are to be avoided. Once the material goes into the unstable post-buckling state, the deflection increases even when the load decreases. In other words, the stiffness of the material in the buckling direction is instantaneously negative. Especially in semi-implicit methods where the internal force is explicitly predicted with derivatives [Baraff and Witkin 1998; Volino and Magnenat-Thalmann 2000], this structural instability makes the system matrix extremely ill-conditioned or indefinite, and a large time step often leads to divergence. There have been a number of efforts to avoid this post-buckling instability in the analysis of cloth deformation. For example, [Eisken et al. 1996] used an adaptive arc-length control of the load-deflection curve, and [Yu et al. 2000] employed an explicit method with fictitious damping to avoid divergence.

We solve the structural instability problem by predicting the static post-buckling response. The approach developed here is based on the above observations regarding fabric-specific behavior. The concept underlying our approach is that since the fabric quickly passes the unstable post-buckling state to reach a stable equilibrium, it has little chance to get into the unstable post-buckling state at the discrete time steps of the simulation. Thus we assume that the fabric is not in the unstable post-buckling state at any time step. Then, in calculating the internal force at each time step, we can evaluate the deformation energy in the area where the cloth buckles from the locally estimated deformed shape, which corresponds to the shape at the static equilibrium after buckling. The details of this procedure are presented in the next section. Unless the time step is miniscule, the loss of accuracy resulting from the approximated buckling response will be much less than the accuracy loss (and instabilities) resulting from the plain implicit time stepping. The approximated response model relieves the burden of simulating the unstable post-buckling dynamic response. According to our simulations, the approximated response model generates very realistic cloth motion with significantly improved stability. Wrinkle formation was quite natural. The simulation could be performed with a large step size.

### 3.3 Type 2 Interaction

The type 2 interaction model is responsible for the post-buckling response created by compressive and bending forces. We predict the shape of the fabric after buckling and calculate the deformation energy from the deformed shape as described in the previous section.

The beam structure shown in Figure 4(a) approximates the region between two particles. Prior to buckling, the structure is a straight beam of length $L$.

After the structure buckles under a compressive load (Figure 4(b)), it will eventually reach a stable equilibrium structure. To predict the equilibrium shape, we use the moment equilibrium equation under the pinned ends condition[Gere 2001], which is given by

$$k_b\kappa + Py = 0,$$

where $k_b$ is the flexural rigidity, $\kappa$ is the curvature, $P$ is the compressive load, and $y$ is the deflection. Because we are modeling systems with large deflections, we cannot use the approximation $\kappa = y''$. Using the exact expression for the curvature $\kappa$, we obtained several numerical solutions corresponding to different values of $k_b$.

---

For a more general moment equilibrium equation for the analysis of fabric buckling, see [Kang et al. 2001].
and \( P \). Two of these solutions are shown in Figure 5. The results show that the shape after buckling is close to a circular arc even when the deformation is large. Therefore, we approximated the equilibrium shape as a circular arc with constant arc length. As an alternative, we could have constructed a table of the numerical solutions of the moment equation at various values of \( k_b \) for more accurate analysis. This was not undertaken because the results produced using the circular arc assumption were quite realistic. The bending deformation energy can be calculated from the estimated shape using the relation:

\[
E = \frac{1}{2} \int_0^L M \kappa dx
\]  

(5)

where \( M \) is the bending moment and \( \kappa \) is the curvature. Taking into account the linear relationship between the curvature and bending moment, and the constant curvature through the structure, the integral yields the solution:

\[
E = \frac{1}{2} k_b L \kappa^2,
\]  

(6)

where \( k_b \) is the flexural rigidity. Since the arc length is assumed to be the same as the initial straight length \( L \), the curvature \( \kappa \) can be expressed solely in terms of the distance between the two extremities.

\[
\kappa = \frac{2}{L} \sin^{-1}\left(\frac{|x_{ij}|}{L}\right),
\]  

(7)

where \( \sin(x) = \frac{\sin x}{x} \).

The force vector is derived as,

\[
f_i \equiv k_b \kappa L \frac{d\kappa}{d|x_{ij}|} x_{ij} = k_b \kappa L \left( \frac{d|x_{ij}|}{dx} \right)^{-1} x_{ij} \]

\[
= k_b \kappa^2 (\cos \frac{\kappa L}{2} - \sin(\frac{\kappa L}{2}))^{-1} x_{ij} \]

\[
\equiv f_b(|x_{ij}|) \frac{x_{ij}}{|x_{ij}|}
\]  

(8)

(9)

(10)

The blue curve in Figure 6 depicts the dependence of \( f_b \) on the distance between particles (approximated with a fifth order polynomial function). The unit of each axis in this graph is made dimensionless. The system shows the following behavior. When the compression force \( P \) is initially applied (top right corner of Figure 6), the structure remains straight until the load reaches the buckling load \( P_{cr} \). However, in real systems geometric imperfections in the structure cause the fabric to start to buckle at the onset of loading, giving an actual curve (the green curve in the graph) that exhibits finite deflection even at small magnitudes of the compression force, and asymptotically approaches \( f_b \) as compressive force increases [Gere 2001]. To model this characteristic, we used the function \( f_b^* \) in our final implementation:

\[
f_b^* = \begin{cases} 
  c_b(|x_{ij}|) - L & : f_b < c_b(|x_{ij}|) - L \\
  f_b & : \text{otherwise}
\end{cases}
\]  

(11)

where \( c_b \) is a constant of our choice, usually assigned a value comparable to \( k_b \). Although we could have used a higher order function to model the deflection at small values of the compression force, we found no significant difference in the results obtained using higher order functions and those obtained with the linear function \( c_b(|x_{ij}|) - L \) (red curve in Figure 6).

The Jacobian matrix of the force vector is derived as

\[
\frac{\partial f_i}{\partial x_j} = \frac{df_b^*}{d|x_{ij}|} x_{ij} x_j^T + f_b^* \left( I - x_{ij} x_{ij}^T x_j x_j^T \right).
\]  

(12)

In the above equation, \( \frac{df_b^*}{d|x_{ij}|} \) is always positive but \( \frac{f_b^*}{|x_{ij}|} \) is always negative, creating the possibility that the second term could turn the system matrix indefinite. To guarantee the positive definiteness of the system matrix, we dropped this term and thus the force in the orthogonal direction is not affected by implicit filtering. Although this force is not filtered, there is little possibility that the system will diverge even when a large time step is employed because the magnitude of this force is very small compared to the stretch force, and it is always finite. Note that in a model where the repulsive and contractive forces are equally strong, dropping the orthogonal term can make the system unstable under a large time step unless the bending resistance is of comparable strength.

The above discussion of the type 2 interaction model highlights the necessity of interlacing the type 2 connections in the manner shown in Figure 2(b). If only sequential connections were used, the global shape could be bent without increasing the bending energy provided that each local connection maintained the initial distance.

### 3.4 Damping

The physical model described above is quite stable; thus, there is no need for additional energy dissipative terms to stabilize the numerical procedure. However, we do need to consider the intrinsic damping property of fabrics. Without an appropriate (material intrinsic) damping term, the simulated fabric can exhibit large unrealistic in-plane oscillations. To include this type of damping, we added a simple linear damper along the direction of interaction.

The damping force exerted on particle \( i \) from the interaction with particle \( j \) is given by,

\[
f_i = -k_d (v_i - v_j)
\]  

(13)

and the Jacobian matrix is simply expressed as,

\[
\frac{\partial f_i}{\partial v_j} = k_d \mathbf{I}.
\]  

(14)

Note that the force term in Equation 13 does not add any damping to the orthogonal direction. This is important because the most interesting fabric deformation occurs in the out-of-plane direction.

The above force term does not create a filtering effect under a semi-implicit formulation when the cloth undergoes a linear rigid motion (i.e., when the velocity vectors of all the particles are identical). This is the case because for such a motion the 3×3 block-wise row sums of the matrix \( \frac{df_i}{dx} \) are 3×3 zero matrices from the above equations, and the rigid motion vector \( v_{rigid} \) is an eigenvector of
with zero eigenvalue. Therefore, there is no implicit filtering effect caused by \( \frac{\partial f}{\partial v} \). Even when the \( \frac{\partial f}{\partial v} \) terms in Equations 3 and 12 are included in the system equation, there will be no implicit filtering to the rigid motion, provided that all the interacting directions of particle \( i \) are orthogonal to \( v_i \) for all \( i \) and that the cloth is not stretched, since the \( \frac{\partial f}{\partial v} \) terms in Equations 3 and 12 also produce null vectors when multiplied by \( v_{rigid} \). Therefore, the motion to the orthogonal direction would not be filtered under a semi-implicit formulation.

In more general situations, even though the above condition is not met, our formulation has very small artifical damping in the out-of-plane direction, which potentially makes the cloth movement look more realistic.

## 4 Numerical Integration

We use semi-implicit integration with a second-order backward difference formula (BDF). The \( k \)-th order BDF is defined as,

\[
d\frac{\mathbf{x}}{dt} = \frac{1}{\Delta t} \sum_{q=1}^{k} \frac{1}{q} (\Delta t^{-1})^q d\mathbf{x} + \mathbf{f}
\]

(15)

where

\[
\Delta t^{-1} \mathbf{x} = \mathbf{x}^{n+1} - \mathbf{x}^n.
\]

For \( k = 2 \), the discretization of \( \dot{x} \) becomes,

\[
\dot{x} = \frac{1}{\Delta t} \left( \frac{3}{2} \mathbf{x}^{n+1} - 2 \mathbf{x}^n + \frac{1}{2} \mathbf{x}^{n-1} \right).
\]

(16)

Considering both performance and accuracy, we chose second order BDF for the semi-implicit formulation. The second order BDF creates less artificial damping than the first order BDF, but is equally stable.

The state equation of motion

\[
\left( \begin{array}{c}
\dot{\mathbf{x}} \\
\dot{v}
\end{array} \right) = \left( \begin{array}{c}
\mathbf{v} \\
\mathbf{M}^{-1} \mathbf{f}
\end{array} \right)
\]

is discretized with the second order BDF as

\[
\frac{1}{\Delta t} \left( \frac{3}{2} \mathbf{x}^{n+1} - 2 \mathbf{x}^n + \frac{1}{2} \mathbf{x}^{n-1} \right) = \left( \begin{array}{c}
\mathbf{v}^{n+1} \\
\mathbf{M}^{-1} \mathbf{f}^{n+1}
\end{array} \right).
\]

(18)

The nonlinear term \( \mathbf{f}^{n+1} \) in the above equation is replaced with

\[
\mathbf{f}^{n+1} = \mathbf{f}^n + \frac{\partial \mathbf{f}}{\partial \mathbf{x}} (\mathbf{x}^{n+1} - \mathbf{x}^n) + \frac{\partial \mathbf{f}}{\partial \mathbf{v}} (\mathbf{v}^{n+1} - \mathbf{v}^n).
\]

(19)

By combining Equations 18 and 19, we can obtain a linear system rearranged either for \( \Delta^{-1} \mathbf{x} \) or \( \Delta^{-1} \mathbf{v} \). If we rearrange the linear system for \( \Delta^{-1} \mathbf{x} \), the equation becomes,

\[
(I - \Delta t^2 \frac{3}{2} \mathbf{M}^{-1} \frac{\partial \mathbf{f}}{\partial \mathbf{x}} - \Delta t \frac{4}{9} \mathbf{M}^{-1} \frac{\partial \mathbf{f}}{\partial \mathbf{v}}) (\mathbf{x}^{n+1} - \mathbf{x}^n)
\]

\[
= \frac{1}{3} (\mathbf{x}^n - \mathbf{x}^{n-1}) + \Delta t \frac{4}{9} (8 \mathbf{v}^n - 2 \mathbf{v}^{n-1})
\]

\[
+ \frac{4 \Delta t^2}{9} \mathbf{M}^{-1} (\mathbf{f}^n - \frac{\partial \mathbf{f}}{\partial \mathbf{v}} \mathbf{v}^n) - 2 \Delta t \frac{4}{9} \mathbf{M}^{-1} \frac{\partial \mathbf{f}}{\partial \mathbf{v}} (\mathbf{x}^n - \mathbf{x}^{n-1}).
\]

(20)

The linear system of Equation 20 is sparse and generally unbounded. We solve this system using a preconditioned conjugate gradient method. We used a \( 3 \times 3 \) block diagonal matrix for the preconditioner, which showed an improvement of approximately 20% over the diagonal preconditioner. In addition, we assessed other preconditioners such as IC and ILU but found no performance gain though the number of iterations decreased.

## 5 Collision Handling

Collision detection and response model is not a contribution of this paper. In this section, we briefly describe how we handled collisions in our implementation.

To detect collisions we use a voxel-based collision detection algorithm similar to that proposed by [Zhang and Yuen 2000]. After voxelizing the space in which the cloth is enclosed, we register each cloth particle and solid triangle to the corresponding voxels based on their spatial coordinates. Then, we independently perform collision detection for each voxel. This voxelization method locates the possible collisions very efficiently and shows nearly linear performance.

We detect the cloth-solid collision by checking the particle-triangle pairs to determine if particles are beneath the solid surface. To avoid missing pairs near the voxel boundaries, the triangles are redundantly registered to the nearby voxels. When a collision is detected, the particle’s next displacement along the normal direction of the colliding surface is determined, and this constraint is enforced using the invariant method in the conjugate gradient iteration proposed in [Baraff and Witkin 1998]. For the tangential direction, we add a frictional force that is proportional both to the constraint force and to the velocity difference between the solid surface and the particle in contact.

To test for self-collision, we check the particle-particle pairs. If the particles are too close, we simply add a repulsive proximity force between the colliding particles. The Jacobian matrix of this force is made to have null eigenvalues in the directions orthogonal to the repelling direction as in the case of the type 2 interaction model in Section 3.3.

## 6 Results

This section reports the results from several simulations. The animations of these simulations can be found at http://graphics.snu.ac.kr/~kjchoi/cloth.htm

Table 1 summarizes the performance of our algorithm on a Pentium3-550 machine. In this table, the CPU sec/frame field cites the total CPU time required to carry out all of the steps (i.e., collision detection, linear system setup, conjugate gradient iteration, etc.) required to produce one frame of 30 Hz animation. For all the simulations, the collision detection time was less than 20% of the total CPU time. The mesh resolutions of the clothes in the animations were about \( L = \sim 2cm \).

For the simulations involving human motions (Animations 1~4), the time step was fixed to \( \Delta t = 1/90s \) throughout the animation; thus the simulator produced one frame of 30 Hz animation with three time steps. The simulations were stable despite the use of the fixed time step. All attempts to use a time step greater than 1/90s encountered collision handling failures before the stability limit was reached.

In Animation 1(a), the character is wearing a one-piece made of a thin fabric. The nature of the fabric was controlled by assigning a small value to the bending rigidity. The character walks at a normal pace without any fast movements. Nevertheless the cloth motion is more responsive than would be expected for a real cloth. To produce a more fabric-like motion, we increased the bending rigidity and intrinsic damping, and reduced the frictional force. The result was Animation 1(b). Animations 2 and 3 contain more vigorous character motions, which created very dynamic movement of the cloth and wrinkles. In Animation 4 the character is wearing jeans. The jeans fabric was modeled by assigning it a high bending rigidity and a high resistance to buckling. The animation produced the
Table 1: Performance Summary

<table>
<thead>
<tr>
<th>animation</th>
<th># particles of cloth</th>
<th># triangles of solid</th>
<th>CPU sec/frame</th>
<th>time step (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1(a)</td>
<td>5608</td>
<td>13802</td>
<td>6.13</td>
<td>0.011</td>
</tr>
<tr>
<td>1(b)</td>
<td>5579</td>
<td>13802</td>
<td>3.86</td>
<td>0.011</td>
</tr>
<tr>
<td>2</td>
<td>5579</td>
<td>15308</td>
<td>3.68</td>
<td>0.011</td>
</tr>
<tr>
<td>3 top skirt</td>
<td>3396</td>
<td>15308</td>
<td>2.98</td>
<td>0.011</td>
</tr>
<tr>
<td>4 top pants</td>
<td>3294</td>
<td>14324</td>
<td>3.25</td>
<td>0.011</td>
</tr>
<tr>
<td>5</td>
<td>2601</td>
<td>0</td>
<td>0.18</td>
<td>0.2</td>
</tr>
</tbody>
</table>

Figure 7: Snapshots from Animation 5: Stability Test ($\Delta t = 0.2s$)

buckled shapes near the knees and ankles quite well, which are frequently observed in real jeans. Several snapshots taken from each of the animations 1~4 are shown in Figure 8.

Animation 5 was designed purely to determine the maximum time step that could be used in our cloth animation technique; it was not intended as a realistic animation. The simulation modeled a square of fabric draped over a solid box, as shown in Figure 7. To exclude the collision detection problem we did not explicitly include the box: Instead, we simply constrained the movement of the sub-square region of the fabric. Additionally, we disabled both self-collision and solid-collision. Under the above conditions, we verified that the algorithm runs stably with time step sizes up to 100s, although the resulting animation was very choppy and it required hundreds of time steps for the fabric to settle down to the final shape. Such a large time step is not meaningful and cannot possibly generate realistic animation because the derivatives have no significance after 100 seconds. A marginally acceptable animation with self-collision enabled was obtained with $\Delta t = 0.2s$. Several snapshots taken during this animation are shown in Figure 7.

Although there is no established method or system for validating the dynamic motion of cloth, our technique produced animations that are visually quite believable. It is noteworthy that these animations were obtained using a reasonable, practical amount of computation.

7 Conclusion

The groundbreaking work of [Baraff and Witkin 1998] on implicit time stepping greatly reduced the computational cost of integrating the stiff equations used in simulating textile fabrics, and thereby provided a practical solution for animating clothed characters. However, the phenomenon of buckling, which is another source of instability and a crucial property in cloth deformation, has been largely ignored until now. If the buckling problem cannot be handled appropriately, natural cloth animation would require enormous (almost impractical) amount of computation.

This paper reports the first study of a stable and practical method to handle the post-buckling instability without introducing a damping force into the dynamic simulation. The proposed method was shown to produce very realistic motion of clothes made from a range of fabric types using a uniform time step size. In particular, the power of the new method was shown in the animation of a light and thin cloth where sensitive response of cloth is required, which was very difficult to produce using the previous methods.

The tremendously increased stability of our algorithm allowed the simulation of the motion of cloth with time steps of 0.2 seconds or longer which could not be achieved in previous methods.

Acknowledgments

This work was supported by Korea Ministry of Information and Communication. This work was also partially supported by Automation and Systems Research Institute at Seoul National University, and the Brain Korea 21 Project.

References


Figure 8: Snapshots from Animations 1~4. Each animation corresponding to each row shows different materials with different parameters.
Numerical Techniques for Cloth Simulation

Michael Hauth
WSI/GRIS, Universität Tübingen

Outline

1. Introduction
   i. Spatial discretisation
   ii. Ordinary Differential Equations
2. Integration methods:
   i. Explicit Euler
   ii. Midpoint and Runge-Kutta methods
   iii. Examples
3. Stability and the Implicit Euler method
4. Implicit methods
   i. Implicit Midpoint method
   ii. Multistep methods
   iii. Verlet method
   iv. Examples
5. Nonlinear Systems
   i. Fixpoint Iteration
   ii. Newton method and variants
6. Constrained Linear Systems and cg
Physically Based Animation

Physical System

Particle System  PDE

\[\text{Finite Elements} \quad \text{Finite Differences}\]

Ordinary Differential Equation

Particle Systems

Modeling physical behavior as forces between mass-points

\[\Rightarrow \text{Ordinary Differential Equation}\]

Example: Mass-spring system
Finite Differences

Energy formulation + variational ansatz/ weak partial differential equation equation, e.g. from continuum mechanics.

Replace spatial differences by a finite difference stencil, e.g.

\[ \frac{\partial}{\partial x} f(x_0) = \frac{f(x_1) - 2f(x_0) + f(x_{-1})}{2h} \]

Usually: terms over edges and pointwise convergence

Problem: Unstructured meshes

Finite Elements

Energy formulation + variational ansatz/ weak partial differential equation equation, e.g. from continuum mechanics.

Replace functions by piecewise polynomial approximations over a discretisation of the domain

\[ u = \sum_i \mu_i b_i \]

Usually: terms over area elements and integral norm convergence.
Physically Based Animation

Physical System

Particle System

PDE

Finite Elements

Finite Differences

Ordinary Differential Equation

Outline

1. Introduction
   i. Spatial discretisation
   ii. Ordinary Differential Equations
2. Integration methods:
   i. Explicit Euler
   ii. Midpoint and Runge-Kutta methods
   iii. Examples
3. Stability and the Implicit Euler method
4. Implicit methods
   i. Implicit Midpoint method
   ii. Multistep methods
   iii. Verlet method
   iv. Examples
5. Nonlinear Systems
   i. Fixpoint Iteration
   ii. Newton method and variants
6. Constrained Linear Systems and cg
General ODE

2\textsuperscript{nd} order mechanical system:

\[ m \frac{d}{dt} x'(t) = f_v(t, x, x') \]
\[ \frac{d}{dt} x(t) = x'(t) \]

Reduction to 1\textsuperscript{st} order

\[ \frac{d}{dt} \begin{bmatrix} x(t) \\ v(t) \end{bmatrix} = \frac{I d}{m} \cdot v(t) \]
\[ \frac{d}{dt} v(t) = f(t, v(t)) \]

Example 1

The simplest Ordinary Differential Equation:

\[ y' = \lambda y \quad \lambda \in \{2, -15\} \]
\[ y(0) = 1 \]

Solution:

\[ y(t) = e^{\lambda t} y_0 \]
Example 2

An overdamped wave equation:

\[ y'' = \frac{\lambda}{2} y + \lambda y', \quad \lambda = -5 \]

Reduction to 1st order:

\[
\begin{align*}
\frac{d}{dt} \begin{bmatrix} y' \\ y'' \end{bmatrix} &= \begin{bmatrix} 0 & 1 \\ \frac{\lambda}{2} & \lambda \end{bmatrix} \begin{bmatrix} y' \\ y'' \end{bmatrix}
\end{align*}
\]

Solution:

\[
y(t) = \frac{\sqrt{15}}{15} e^{-t} + \frac{\sqrt{15}}{15} e^{t} \]

Example 3

A simple spring system:

\[
\frac{d^2 z}{dt^2} = \frac{k}{m} (l_0 - z) + \frac{d}{dt} \frac{dz}{dt} + g_z
\]

Reduction to 1st order:

\[
\begin{align*}
\frac{d}{dt} \begin{bmatrix} z \\ v_z \end{bmatrix} &= \begin{bmatrix} 0 & I \\ -\frac{k}{m} I & \frac{d}{m} I \end{bmatrix} \begin{bmatrix} z \\ v_z \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{k}{m} l_0 + g_z \end{bmatrix}
\end{align*}
\]

Parameter:

\[ k = 1000, l_0 = -1, d = 10, g = -10, m = 1 \]
Example 3 (contd.)

\[ \frac{d^2 z}{dt^2} = \frac{k}{m} (l_0 - z) + \frac{d}{dt} \frac{dz}{dt} + g_z \]

Analytical and a numerical solution:

Outline

1. Introduction
   i. Spatial discretisation
   ii. Ordinary Differential Equations
2. Integration methods:
   i. Explicit Euler
   ii. Midpoint and Runge-Kutta methods
   iii. Examples
3. Stability and the Implicit Euler method
4. Implicit methods
   i. Implicit Midpoint method
   ii. Multistep methods
   iii. Verlet method
   iv. Examples
5. Nonlinear Systems
   i. Fixpoint Iteration
   ii. Newton method and variants
6. Constrained Linear Systems and cg
Solution of ODEs

Numerical solution of a ODEs
Structure of the solver

Integration formula
Nonlinear Solver
Linear Solver
Residuum control
Steps size control
Constraints

Evaluating a Method

Criteria:
- Convergence
- Accuracy
- Stability
- Efficiency
Numerical Integration

Abstract problem:
\[ y' = f(t, y) \]

Goal:
Find approximations \( Y_n \approx y(t_n) \), with \( t_n = t_0 + nh \)

Explicit Euler:
\[ Y_{n+1} = Y_n + hf(t_n, Y_n) \]

Analysis by Taylor-Expansion

Taylor expansion of the true solution:
\[ y(t+h) = y(t) + hy'(t) + h^2/2 y''(t) + O(h^3) \]

For
\[ Y_{n+1} = Y_n + hf(t_n, Y_n) \]

Local error:
\[ \| y(t_1) - Y_1 \| = O(h^2) \]

Global error:
\[ \| y(t_n) - Y_n \| = O(h) \]

Definition: The method has order 1
Geometrical Interpretation

\[ Y_{n+1} = Y_n + hf(t_n, Y_n) \]

Geometrical Interpretation (contd.)

\[ Y_{n+1} = Y_n + hf(t_n, Y_n) \]
Geometrical Interpretation (contd.)

\[ Y_{n+1} = Y_n + hf(t_n, Y_n) \]

\[ Y_1 = y_0 + hf(t + h/2, y(t + h/2)) \]

Outline

1. Introduction
   i. Spatial discretisation
   ii. Ordinary Differential Equations
2. Integration methods:
   i. Explicit Euler
   ii. Midpoint and Runge-Kutta methods
   iii. Examples
3. Stability and the Implicit Euler method
4. Implicit methods
   i. Implicit Midpoint method
   ii. Multistep methods
   iii. Verlet method
   iv. Examples
5. Nonlinear Systems
   i. Fixpoint Iteration
   ii. Newton method and variants
6. Constrained Linear Systems and cg
**Explicit Midpoint Rule**

“Midpoint Integration”:

\[ Y_{n+1} = Y_n + hf(t_n + h/2, y(t_n + h/2)) \]

Idea: \( y(t_n + h/2) \approx Y_n + h/2 f(Y_n) \) (Euler step)

Method:

\[ k_1 = Y_n + \frac{h}{2} f(t_n, Y_n) \]
\[ Y_{n+1} = Y_n + hf(t_n + h/2, k_1) \]

**Order: 2**

---

**Runge-Kutta Methods**

Generalisation:
Let \( a_{ij} \) (coefficients), \( b_i \) (weights), \( c_j \) (abscissae) be given.

\[ k_i = Y_n + h \sum_{j=1}^{s} a_{ij} k_j, \quad k_i' = f(t_n + c_i h, k_i) \quad i = 1..s \]
\[ Y_{n+1} = Y_n + h \sum_{i=1}^{s} b_i k_i' \]

Runge-Kutta Table:

<table>
<thead>
<tr>
<th>( c_i )</th>
<th>( a_{11} )</th>
<th>( a_{12} )</th>
<th>( \cdots )</th>
<th>( a_{1s} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( c_2 )</td>
<td>( a_{21} )</td>
<td>( a_{22} )</td>
<td>( \cdots )</td>
<td>( a_{2s} )</td>
</tr>
<tr>
<td>( \vdots )</td>
<td>( \vdots )</td>
<td>( \ddots )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( c_s )</td>
<td>( a_{s1} )</td>
<td>( a_{s2} )</td>
<td>( \cdots )</td>
<td>( a_{ss} )</td>
</tr>
<tr>
<td>( b_1 )</td>
<td>( b_2 )</td>
<td>( \cdots )</td>
<td>( b_s )</td>
<td></td>
</tr>
</tbody>
</table>
6.2.2002

Examples of RKM

\[ k_i = Y_n + h \sum_{j=1}^{s} a_{ij} k_j', \quad k_i' = f(t_n + c_i h, k_i) \quad i = 1..s \]

\[ Y_{n+1} = Y_n + h \sum_{j=1}^{s} b_j k_j' \]

Order 2:

\[
\begin{array}{ccc}
0 & 0 & 0 \\
\frac{1}{2} & \frac{1}{2} & 0 \\
0 & 0 & 1 \\
\end{array}
\]

explicit midpoint

Order 4:

\[
\begin{array}{cccc}
0 & 1 & 0 & 0 \\
\frac{1}{3} & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\
\frac{1}{2} & 0 & \frac{1}{2} & 0 \\
1 & 0 & 0 & 1 \\
\frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \\
\end{array}
\]

"The" RK4 method

6.2.2002

Outline

1. Introduction
   i. Spatial discretisation
   ii. Ordinary Differential Equations
2. Integration methods:
   i. Explicit Euler
   ii. Midpoint and Runge-Kutta methods
   iii. Examples
3. Stability and the Implicit Euler method
4. Implicit methods
   i. Implicit Midpoint method
   ii. Multistep methods
   iii. Verlet method
   iv. Examples
5. Nonlinear Systems
   i. Fixpoint Iteration
   ii. Newton method and variants
6. Constrained Linear Systems and cg
Example 1a: \( y' = 2y \)

Example 1b: \( y' = -5y \)
Example 2: \[ y'' = \frac{-5}{2}y - 5y' \]

Example 3: \[ \frac{d^2z}{dt^2} = \frac{k}{m}(z - z_t) + \frac{d}{dt}\left(\frac{d}{dt}z\right) + g_z \]
What went wrong?

Analysing the simplest example:

\[ y' = \lambda y \]

Explicit Euler:

\[ Y_{n+1} = Y_n + h\lambda \quad Y_n = (1 + h\lambda) Y_n \]

Solving the recursion:

\[ Y_{n+1} = (1 + h\lambda)^n Y_0 \]

Step size restriction:

For \( \lambda < 0 \):

\[ Y_n < \infty \iff |1 + h \lambda| < 1 \]

Outline

1. Introduction
   i. Spatial discretisation
   ii. Ordinary Differential Equations
2. Integration methods:
   i. Explicit Euler
   ii. Midpoint and Runge-Kutta methods
   iii. Examples
3. Stability and the Implicit Euler method
4. Implicit methods
   i. Implicit Midpoint method
   ii. Multistep methods
   iii. Verlet method
   iv. Examples
5. Nonlinear Systems
   i. Fixpoint Iteration
   ii. Newton method and variants
6. Constrained Linear Systems and cg
**Numerical Integration**

Abstract problem:

\[ y' = f(t, y) \]

Numerical methods:

Explicit Euler: \[ Y_{n+1} = Y_n + hf(t_n, Y_n) \]

Implicit Euler: \[ Z_{n+1} = Z_n + hf(t_{n+1}, Z_{n+1}) \]

\[ Y_n = Z_n \]

**Linear Stability Analysis**

Dahlquists Equation:

\[ y' = \lambda y \quad \text{with solution} \quad y = e^{\lambda t} y_0 \]

Applying Euler:

\[ Y_n = (1 + h\lambda)^n Y_0 < \infty \quad \Leftrightarrow \quad |1 + h\lambda| < 1 \]

\[ Z_n = (1 - h\lambda)^n Z_0 < \infty \quad \Leftrightarrow \quad |1 - h\lambda|^{-1} < 1 \]
Relevance of Dahlquists Equation

Abstract setting:
\[ y' = f(t, y) \]

Linearise locally around a solution by Taylor expansion:
\[
y' = f(t, y) = f(t, \tilde{y}(t)) + \frac{\partial}{\partial y} f(t, \tilde{y}(t)) (\tilde{y}(t) - y(t)) + \ldots
\]
\[
= \text{const} + J(\tilde{y}(t) - y(t)) + \ldots
\]

Use an eigenbasis of \( J \) to obtain
\[
\hat{y}_n' = \lambda_n \hat{y}_n \quad n = 1 \ldots N
\]

Observation: All passive physical systems come to rest
\[ \Rightarrow y(t) < \infty \quad \text{and} \quad \text{Re} \lambda < 0 \]

A-Stability

For fixed \( \lambda \) condition for stepsize \( h \)

stable for \( h, \lambda : \Leftrightarrow \) solution of Dahlquist is bounded

A-stable : \( \Leftrightarrow \) stable for all \( \text{Re} \lambda < 0 \) and \( h > 0 \)

**stability region**: all \( h\lambda \) with bounded solution \( (h > 0) \)
Recall:
\[ Y_1 = \mathcal{R}(h\lambda) \ Y_0 \quad \text{with} \quad \mathcal{R}(z) = (1 + z) \quad \text{(ex. Euler)} \]
\[ \mathcal{R}(z) = (1 - z)^{-1} \quad \text{(im. Euler)} \]

\( \mathcal{R} \) is the **stability function**.

Now:
Method is A-Stable \( \Leftrightarrow |\mathcal{R}(z)| < 1 \) for all \( \Re(z) < 0 \)

It can be shown:

The stability function \( \mathcal{R} \) of all explicit Runge-Kutta methods is given by a **polynomial**.

\( \ddot{y} \) \( \mathcal{R} \) is unbounded on the negative halfplane

\( \ddot{y} \) The stability region (where \( \| \mathcal{R}(z) \| < 1 \)) is bounded

\( \ddot{y} \) No explicit Runge-Kutta Method can be A-stable
Stability of RKM

Outline

1. Introduction
   i. Spatial discretisation
   ii. Ordinary Differential Equations
2. Integration methods:
   i. Explicit Euler
   ii. Midpoint and Runge-Kutta methods
   iii. Examples
3. Stability and the Implicit Euler method
4. Implicit methods
   i. Implicit Midpoint method
   ii. Multistep methods
   iii. Verlet method
   iv. Examples
5. Nonlinear Systems
   i. Fixpoint Iteration
   ii. Newton method and variants
6. Constrained Linear Systems and cg
“Midpoint Integration”:

\[ Y_{n+1} = Y_n + hf\left(t_n + h/2, y(t_n + h/2)\right) \]

Linear approximation: \( y(t_n+h/2) \approx 1/2 \left( Y_{n+1} + Y_n \right) \)

\[ Y_{n+1} = Y_n + hf \left( t_n + \frac{1}{2} h, \frac{Y_{n+1} + Y_n}{2} \right) \]

Taylor analysis shows: Order 2.

Formulation as a Runge-Kutta method:

\[ k_1 = f\left( t_n + \frac{1}{2} h, \frac{Y_{n+1} + Y_n}{2} \right) \]

\[ Y_{n+1} = Y_n + hk_1 \]

Stability function: \( R(z) = \frac{1+z/2}{1-z} \)

\[ \hat{\mathcal{Y}} \] A-stable
Outline

1. Introduction
   i. Spatial discretisation
   ii. Ordinary Differential Equations
2. Integration methods:
   i. Explicit Euler
   ii. Midpoint and Runge-Kutta methods
   iii. Examples
3. Stability and the Implicit Euler method
4. Implicit methods
   i. Implicit Midpoint method
   ii. Multistep methods
   iii. Verlet method
   iv. Examples
5. Nonlinear Systems
   i. Fixpoint Iteration
   ii. Newton method and variants
6. Constrained Linear Systems and cg

Multistep Methods

General Multistep Method:

\[ \sum_{j=0}^{k} \alpha_j Y_{k_j+1} = h \sum_{j=0}^{k} \beta_j f_{k_j+1} \]

Adams Methods: (order \( k+1 \))

\[ Y_1 = Y_0 + h \sum_{j=0}^{k} \beta_j f_{k_j+1} \]

BDF Methods: (order \( k \))

\[ \sum_{j=0}^{k} \alpha_j Y_{k_j+1} = h \beta_k f_1 \]
Outline

1. Introduction
   i. Spatial discretisation
   ii. Ordinary Differential Equations
2. Integration methods:
   i. Explicit Euler
   ii. Midpoint and Runge-Kutta methods
   iii. Examples
3. Stability and the Implicit Euler method
4. Implicit methods
   i. Implicit Midpoint method
   ii. Multistep methods
   iii. Verlet method
   iv. Examples
5. Nonlinear Systems
   i. Fixpoint Iteration
   ii. Newton method and variants
6. Constrained Linear Systems and cg

2nd Order ODE: Verlet

For a pure 2nd order system:

\[ x'' = f(x, x(t)) \]

*Direct* approximation of \( x'' \):

\[
f(t_n) = x''(t_n) = \frac{x_{n+1} - 2x_n + x_{n-1}}{h^2} = \frac{1}{h} \left( \frac{x_{n+1} - x_n}{h} \right) - \left( \frac{x_n - x_{n-1}}{h} \right)
\]

\( = \frac{1}{h} (v_{n+1/2} - v_{n-1/2}) \)
From this:

\[ v_{n+1/2} = v_{n-1/2} + hf(t_n) \]
\[ x_{n+1} = x_n + h v_{n+1/2} \]
Numerical Techniques for Cloth Simulation;
M. Hauth, WSI/GRIS

Example 1a: $y' = 2y$

Example 1b: $y' = -5y$
Example 2: \( y'' = \frac{-5}{2} y - 5y' \)

Example 2a: Undamped wave equation \( y'' = \frac{-5}{2} y' \)
Numerical Techniques for Cloth Simulation;
M. Hauth, WSI/GRIS

Example 3: \[
\frac{d^2 z}{dt^2} = \frac{k}{m} (v_i - z) + \frac{d}{m} \frac{dz}{dt} + g_x;
\]

Work-Precision Diagram

In Detail: Stability for Ex.3
In Detail: Stability for Ex.3

- Exact solution
- Euler, implicit, h=0.02
- Euler, explicit, h=0.02

Integration formula
Stepsize control

\[
\begin{bmatrix}
0 \\
1 \\
0 \\
\end{bmatrix} z
+ \begin{bmatrix}
0 \\
\frac{1}{\Delta t} \\
0 \\
\end{bmatrix} \begin{bmatrix}
v_1 \\
v_2 \\
v_3 \\
\end{bmatrix}
+ \begin{bmatrix}
0 \\
\frac{1}{\Delta t} \\
0 \\
\end{bmatrix} \begin{bmatrix}
\Delta t_1 + g_1 \\
\Delta t_2 + g_2 \\
\Delta t_3 + g_3 \\
\end{bmatrix}
\]

Euler, explicit
h=0.010
h=0.00125

Re
Im
In Detail: Stability for Ex.3

Linear Stability: Conclusion

Implicit A-stable methods are

- More expensive
- But stable with large stepsizes.
- Superior for low accuracy requirements, when stability limits the admissible stepsizes.

The Dahlquist equation is the central tool in evaluating stability properties.
Excursion: Stability for Non-Linear Equations

L-Stability: $\Leftrightarrow \mathcal{R}(\infty) = 0$

Another test equation: $y' = -\alpha(y - \cos t), \quad y_0 = 0$

Solution: $y = \cos t$

\[ \alpha = 1000 \]
Excursion: Stability for Non-Linear Equations

L-Stability: $\Re(-\infty)=0$

Another test equation: $y' = -\alpha (y - \cos t), \quad y_0 = 0$

Solution: $y \approx \cos t$

Excursion: Stability for Non-Linear Equations

L-Stability: $\Re(-\infty)=0$

Another test equation: $y' = -\alpha (y - \cos t), \quad y_0 = 0$

Solution: $y \approx \cos t$
Summary of Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>A-stable</th>
<th>L-Stable</th>
<th>Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expl. Euler</td>
<td>No</td>
<td>No</td>
<td>1</td>
</tr>
<tr>
<td>Imp. Euler</td>
<td>Yes</td>
<td>Yes</td>
<td>1</td>
</tr>
<tr>
<td>Explicit Midpoint</td>
<td>No</td>
<td>No</td>
<td>2</td>
</tr>
<tr>
<td>RK4</td>
<td>No</td>
<td>No</td>
<td>4</td>
</tr>
<tr>
<td>Adams-Bashforth(κ) (explicit)</td>
<td>No</td>
<td>No</td>
<td>κ</td>
</tr>
<tr>
<td>Adams-Moulton(κ) (implicit)</td>
<td>No</td>
<td>No</td>
<td>κ+1</td>
</tr>
<tr>
<td>Impl. Midpoint</td>
<td>Yes</td>
<td>No</td>
<td>2</td>
</tr>
<tr>
<td>BDF(2)</td>
<td>Yes</td>
<td>Yes</td>
<td>2</td>
</tr>
<tr>
<td>BDF κ≤6</td>
<td>A(α) (Sector)</td>
<td>Yes</td>
<td>κ</td>
</tr>
</tbody>
</table>

Selecting a Method

- **explicit Euler for evaluation of correctness.**
- Does the solution suddenly diverge for h>h_\text{crit}? (Yes/No)

  - System is supposed to be stiff. Are the solutions nearly oscillatory? (Yes/No)
  - Integrators: Verlet
  - Numerical or analytical Jacobian needed.
  - Integrators: Implicit Euler, Implicit Midpoint BDF

  - System is supposed to be non-stiff. Is accuracy more important than fast computation times? (Yes/No)
  - Accurate Integrators:
    - Explicit Midpoint Adams
    - RK4
  - Fast Integrators:
    - Explicit Euler
    - Implicit Midpoint Adams
Outline

1. Introduction
   i. Spatial discretisation
   ii. Ordinary Differential Equations
2. Integration methods:
   i. Explicit Euler
   ii. Midpoint and Runge-Kutta methods
   iii. Examples
3. Stability and the Implicit Euler method
4. Implicit methods
   i. Implicit Midpoint method
   ii. Multistep methods
   iii. Verlet method
   iv. Examples
5. Nonlinear Systems
   i. Fixpoint Iteration
   ii. Newton method and variants
6. Constrained Linear Systems and cg

Solving Nonlinear Systems

We have to solve

\[
Y_1 - hf(Y_1) - Y_0 = 0 \quad \text{(Euler)}
\]

\[
G(Y)
\]

\[
Y_1 - hf\left(\frac{Y_1 + Y_0}{2}\right) - Y_0 = 0 \quad \text{(midpoint)}
\]

\[
Y_1 - \frac{2}{3} hf(Y_1) + \left(-\frac{4}{3} Y_0 + \frac{1}{3} Y_{-1}\right) = 0 \quad \text{(BDF(2))}
\]
Fixpoint Iteration

The most simple method for \( Y := hf(t, Y) + Y_0 : \)

\[
Y^{(n+1)} := hf(Y^{(n)}) + Y_0
\]

converges \( \iff \| hf(y) - hf(z) \| < \| z - y \| \)

Therefore for implicit Euler

\[
\| h(f(y) - f(z)) \| < 1
\]

and for the Dahlquist example:

\[
|h\lambda| < 1
\]

Again an unwanted restriction on the stepsize!

Outline

1. Introduction
   i. Spatial discretisation
   ii. Ordinary Differential Equations
2. Integration methods:
   i. Explicit Euler
   ii. Midpoint and Runge-Kutta methods
   iii. Examples
3. Stability and the Implicit Euler method
4. Implicit methods
   i. Implicit Midpoint method
   ii. Multistep methods
   iii. Verlet method
   iv. Examples
5. Nonlinear Systems
   i. Fixpoint Iteration
   ii. Newton method and variants
6. Constrained Linear Systems and cg
Newton’s Method

Newton’s method:

For \( k = 1, 2, \ldots \) until convergence do

1. Compute \( G(y^{(k)}) \) \( O(N) \)
2. Compute \( J^{(k)} = \frac{\partial}{\partial y} G(y^{(k)}) \) \( O(N) - O(N^2) \)
3. Solve \( J^{(k)} s^{(k)} = -G(y^{(k)}) \) Gauss: \( O(N^3) \)
4. Update \( y^{(k+1)} = y^{(k)} + s^{(k)} \) \( O(N) \)

Simplified Newton Method

Newton’s method:

2. Compute \( J^{(k)} = \frac{\partial}{\partial y} G(y^{(k)}) \) \( O(N) - O(N^2) \)
   Factor \( J^{(k)} s^{(k)} = -G(y^{(k)}) \) Gauss: \( O(N^2) \)

For \( k = 1, 2, \ldots \) until convergence do

1. Compute \( G(y^{(k)}) \) \( O(N) \)

3. Solve \( J^{(k)} s^{(k)} = -G(y^{(k)}) \) \( O(N) \)
4. Update \( y^{(k+1)} = y^{(k)} + s^{(k)} \) \( O(N) \)
Newton’s method:

For \( k = 1, 2, \ldots \) until convergence do

1. Compute \( \mathbf{G}(\mathbf{y}^{(k)}) \) \( \mathcal{O}(N) \)
2. Compute \( \mathbf{J}^{(k)} = \frac{\partial}{\partial \mathbf{y}} \mathbf{G}(\mathbf{y}^{(k)}) \) \( \mathcal{O}(N) \) (sparse!)
3. Solve \( \mathbf{J}^{(k)} \mathbf{s}^{(k)} \approx -\mathbf{G}(\mathbf{y}^{(k)}) \) Iterative method: \( \mathcal{O}(N) \)
4. Update \( \mathbf{y}^{(k+1)} = \mathbf{y}^{(k)} + \mathbf{s}^{(k)} \) \( \mathcal{O}(N) \)

\( \mathbf{y}^{(k)} \)

e.g. preconditioned Krylov methods like cg, or Jacobi/Gauss-Seidel Iteration, Multigrid.

(Un-) fortunately this changes the convergence behaviour!

Sophisticated formulation of the inaccuracy:

Inexact Newton method:

For \( k = 1, 2, \ldots \) until convergence do

1. Compute \( \mathbf{G}(\mathbf{y}^{(k)}) \)
2. Compute \( \mathbf{J}^{(k)} = \frac{\partial}{\partial \mathbf{y}} \mathbf{G}(\mathbf{y}^{(k)}) \)
3. Solve \( \mathbf{J}^{(k)} \mathbf{s}^{(k)} \approx -\mathbf{G}(\mathbf{y}^{(k)}) \) such that \( \mathbf{y}^{(k+1)} = \mathbf{y}^{(k)} + \mathbf{s}^{(k)} \)
4. Update \( \mathbf{y}^{(k+1)} = \mathbf{y}^{(k)} + \mathbf{s}^{(k)} \)

Converges if and only if Newton converges and \( \eta^{(k)} \leq \eta < 1 \)
Simplified Inexact Newton

Classical Newton too costly to implement
⇒ Combine with simplified Newton with $J^{(k)} \approx J^{(0)}$

Generalisation: take any constant $J \approx J^{(0)}$ and write

$$J_s^{(k)} = -G(Y^{(k)}) + (J - J^{(k)})s^{(k)} - r^{(k)} = -G(Y^{(k)}) + \hat{r}^{(k)}$$

Simplified inexact Newton method:

2. Compute \( J \approx \frac{1}{\eta} G(y^{(0)}) \) \( \mathcal{O}(N) \)

For \( k=1,2,... \) until convergence do

1. Compute \( G(y^{(k)}) \) \( \mathcal{O}(N) \)

3. Solve \( J_s^{(k)} = -G(y^{(k)}) + r^{(k)} \) such that \( \|r^{(k)}\| \leq \eta \|r^{(0)}\| \) \( \mathcal{O}(N) \)

4. Update \( Y^{(k+1)} = Y^{(k)} + s^{(k)} \) \( \mathcal{O}(N) \)

Computation of the Jacobian

- Analytical
- Numerical differentiation:

$$J(y) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(y) & \ldots & \frac{\partial f}{\partial x_N}(y) \end{bmatrix} \quad \text{with} \quad \frac{\partial f}{\partial x_i}(y) = \frac{f(y + \epsilon e_i) - f(y)}{\epsilon}$$

- For cg we need only matrix vector products:

$$J(y)v = \frac{f(y + \epsilon v) - f(y)}{\epsilon}$$

(“matrix free method”)
Application to Cloth

We choose

\[
J = I - h \gamma \begin{bmatrix} 0 & I \\ k \Delta & d \Delta \end{bmatrix}
\]

thus faster matrix computation

faster matrix multiplication (3x)

Convergence behaviour gives hint when to change stepsize!

We choose

\[
\eta = 0.02 \quad \text{or} \quad \eta = \text{const} \cdot \sqrt{\|J\|_2}
\]

Nonlinear Systems: Summary

- Newton-type Method necessary.
- Newton can be used with an approximated Jacobian.
- Newton can be used with an iterative method as an inner solver.
Outline

1. Introduction
   i. Spatial discretisation
   ii. Ordinary Differential Equations
2. Integration methods:
   i. Explicit Euler
   ii. Midpoint and Runge-Kutta methods
   iii. Examples
3. Stability and the Implicit Euler method
4. Implicit methods
   i. Implicit Midpoint method
   ii. Multistep methods
   iii. Verlet method
   iv. Examples
5. Nonlinear Systems
   i. Fixpoint Iteration
   ii. Newton method and variants
6. Constrained Linear Systems and cg

Solving the linear systems

conjugate gradient (cg) for solving $As = G$

optimal (Krylov-) method for s.p.d. systems

**Preconditioning:** Solve $M^{-1}As = M^{-1}G$

- diagonal preconditioner/diagonal scaling $M := \text{diag}(A)$
  (Baraff’98, Volino’00)
- ICC (incomplete Cholesky factorisation) (EEH’00)
  $A \approx CC^T$ $M := CC^T$
- SSOR (successive symmetric overrelaxation)
  $A \approx D - L - L^T$ $M := (D - L)^{-1}(D - L^T)$
- Block diagonal preconditioning

SIGGRAPH 2003 Course #29: Clothing Simulation and Animation,
Ko/Breen/Hauth/Fedkiew/House
Comparing Preconditioners

<table>
<thead>
<tr>
<th>Preconditioner</th>
<th>#Newton</th>
<th>#cg</th>
<th>solver time (s)</th>
<th>refactor time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diag</td>
<td>898</td>
<td>2677</td>
<td>7.64</td>
<td>0</td>
</tr>
<tr>
<td>LCC</td>
<td>532</td>
<td>837</td>
<td>4.27</td>
<td>0.25</td>
</tr>
<tr>
<td>SSOR</td>
<td>494</td>
<td>813</td>
<td>4.03</td>
<td>0.2</td>
</tr>
</tbody>
</table>

800 triangles, 1s using $h=0.02$, no stepsize control
Artificial refactoring in each integration step

Adding constraints

The idea of filtered cg (Baraff’98):

$$s^{k+1} = s^k + \alpha_k d^k$$
$$r^{k+1} = G - Ax^{k+1} = r^k - \alpha_k Ad^k$$

Construct $P$ which “filters” unwanted update directions

$$s^{k+1} = s^k + \alpha_k P d^k$$
$$r^{k+1} = r^k - \alpha_k PAd^k$$

In fact this solves:

$$\begin{pmatrix} PAP \end{pmatrix}s = (PG)$$
Enforcing constraints

Complete process:

$Y_0$  Newton  $Y_1$

$\text{constrain}$

Integration formula

Summary

- Implicit methods allow large time steps.
- Implicit methods are superior for stiff equations and low accuracy.
- Explicit methods are easier to implement and an option for high accuracy.
- The Efficiency of implicit methods critically depends on the (non-) linear system solver.
- Using cg, the Jacobian is not needed in matrix form and can be approximated.
1 Spatial discretisation

The modelling of physical systems often leads to partial or directly to ordinary differential equations. The solution of these equations usually is a dominant part of the total computational costs for a simulation or animation, therefore being the main focus of this chapter. Before addressing the numerical solution of ODEs - the temporal discretization or (time) integration - we take a brief look at the numerical techniques used for spatial discretization. Three techniques, not entirely unrelated, dominate the field. They are normally classified as particle systems, finite difference and finite element methods.

1.1 Particle Systems

When using the particle system paradigm, the discretization is already a part of the physical modelling process, as the continuous object is immediately represented as a set of discrete points with finite masses. Physical properties are specified by directly defining forces between these mass points. Typical representatives of this approach, that is very popular in cloth simulations, are mass-spring-damper systems [21, 26] and particle systems with forces defined directly by measured curves [6] or low order polynomial fits of this data [2].

1.2 Finite Differences

Another physical modelling concept is to specify physical behavior by minimizing some energy functionals defined on a continuous solid. The arising equations normally contain derivatives with respect to space and time variables. Replacing the spatial derivatives with finite differences, e.g.

\[
\frac{\partial}{\partial x} f \rightarrow \frac{1}{2h} (f(x + h) - f(x - h))
\]

and

\[
\frac{\partial^2}{\partial x^2} f \rightarrow \frac{1}{h^2} (f(x + h) - 2f(x) + f(x - h))
\]

leads to finite difference formulations. This replacement is easily accomplished in 1-space or on structured grids in any dimension. It becomes harder to define finite difference approximations on unstructured meshes [18], often finite element techniques are used for deriving appropriate schemes [19]. An important trait of finite difference schemes is, that the terms on the right hand side of the ODE are given for a point and its neighbors, i.e. are specified on the edges of the discretization. The resulting equations are structured very similar to these from particle systems, indeed finite difference techniques can be used to derive a particle system from continuous equations [9].

1.3 Finite Elements

Finite elements, in their beginning, were designed to overcome the difficulties of finite differences with unstructured meshes. They also start with a continuous model, usually given as an integral equation in a weak form, e.g. resulting form a variational ansatz. The
functions are then replaced by a piecewise polynomial approximation over the unstructured discretization of the domain. The number of nodes per element and the polynomial degree of the shape functions can be varied on a single grid. Curvilinear grids allow a very good shape fitting. For visual simulations one usually prefers piecewise linear approximations over triangle, quad-, hexa- or tetrahedral meshes. The finite element method is the preferred solution technique in numerical analysis and engineering applications because of its versatility, sound derivation and superior convergence properties with respect to integral norms, often natural for the problem at hand. The arising equations are given by terms formed over the area elements of the mesh. Their drawback is that these improvements are paid for by an increased computational effort, compared to the previous techniques. Also the masses may not be concentrated on the points of the mesh, as in the previous cases, leading to an implicit ODE

\[ Mx' = f(x) \]  

with a non-diagonal and even singular mass matrix \( M \). A common technique is to use mass lumping, to make \( M \) diagonal and reduce computational costs \[24, 8\].

For the following discussion we assume that the ODE is given in the general explicit form

\[ x' = f(x) \text{ or } x'' = f(x, x') \]  

before we will come back to the special setting \( (6) \) for eventually gaining computational advantages.

2 Methods for Numerical Integration

As we have seen in the previous sections mechanical systems are often given as a second order ordinary differential equation accompanied by initial values

\[
\begin{align*}
  x''(t) &= f_v(t, x(t), x'(t)), \\
  x(t_0) &= x_0, \quad x'(t_0) = v_0.
\end{align*}
\]

The differential equation can be transformed into a first order system by introducing velocities as a separate variable:

\[
\begin{bmatrix}
  x(t) \\
  v(t)
\end{bmatrix}' = \begin{bmatrix}
  f_v(t, x(t), v(t)) \\
  v(t)
\end{bmatrix}, \quad \begin{bmatrix}
  x(t_0) \\
  v(t_0)
\end{bmatrix} = \begin{bmatrix}
  x_0 \\
  v_0
\end{bmatrix}.
\]

For the next few sections it will be convenient to write this ODE in the more abstract form

\[ y'(t) = f(t, y(t)), \quad y(t_0) = y_0, \]

and we will come back to the special setting \( (6) \) for eventually gaining computational advantages.

Throughout the following discussion we will use the following examples:

1. \( y' = \lambda y, \quad y(0) = 1 \) with \( \lambda = 2, -15 \) for \( t \in [0, 1] \) (figure 1).

2. The overdamped wave equation \( y'' = \lambda/2y + \lambda y' \) with \( \lambda = -5 \) for \( t \in [0.10] \) and starting values \( y(0) = 0, \quad y'(0) = 1 \). It has the analytical solution

\[
y(t) = \frac{1}{15} \sqrt{15} e^{1/2(-5+\sqrt{15})t} - \frac{1}{15} \sqrt{15} e^{-1/2(5+\sqrt{15})t}.
\]

3. This example is based on a simple mechanical system (figure 2(a)): A particle \( p \) with mass \( m \) connected to the origin using a spring with stiffness \( k \), damping \( d \) and rest length \( l_0 \), is pulled down by gravity. This setting is described by the
ODE
\[ \frac{d^2 z}{dt^2} = \frac{k (l_0 - z)}{m} - \frac{d}{m} \frac{dz}{dt} + g_z. \] (8)

We set the parameters \( m = 0.1, k = 100, l_0 = -1, d = 1, g = -10 \) \( v_{0,z} = -5 \) and simulate the interval \( t \in [0, 2] \) (figure 2(b)).

2.1 Explicit methods

The oldest and most simple method of integration is the so called forward or explicit Euler method. Time is discretised into slices of length \( h \). To get a formula for advancing a time step \( h \), the differential quotient on the left hand side of (7) is replaced by the forward difference quotient
\[ \frac{y(t + h) - y(t)}{h} \approx y'(t) = f(t, y(t)). \] (9)

Thus we get the integration formula for advancing a single timestep
\[ y(t + h) = y(t) + hf(t, y(t)). \] (10)

Iterating this method gives a sequence of numerical approximations \( Y_n \approx y(t_n) =: y(t_0 + nh) \). Geometrically this method can be interpreted as straightly following the tangent of the solution and then recalculating the slope for the next step.

There are several criteria for evaluating an integration method:
- convergence
- accuracy
- stability
- efficiency

Convergence means that for \( h \to 0 \) the numerical solutions \( Y_n \) meet the analytical. All useful methods must be convergent, so we won’t discuss non-convergent methods or criteria for convergence. More interesting is the accuracy or order of a method. By this we mean how fast a method converges for \( h \to 0 \), or with other words how accurate the solution is for
But how do we find $k_i$? From a given $h$. By using a taylor expansion for the exact solution after a single time step

$$y(t + h) = y(t) + h y'(t) + h^2/2 y''(t) + O(h^3)$$  (11)

we find that for the numerical approximation $Y_1$ produced by an explicit Euler step

$$y(t_1) - Y_1 = O(h^2).$$  (12)

If we continue the method using the numerical solution $Y_1$ as a starting value for the next time step we lose[13] a power of $h$ for the global error

$$y(t_n) - Y_n = O(h).$$  (13)

This means that the explicit Euler method converges linearly or has order 1. We will analyze the stability and efficiency of the method later.

As a next step we introduce methods of higher order. For this a centered difference estimation for $y'(t + h/2)$ (7) is used

$$\frac{y(t + h) - y(t)}{h} \approx y'(t + h/2) = f(t + h/2, y(t + h/2)).$$  (14)

resulting in the iteration scheme

$$Y_{n+1} = Y_n + h f(t + h/2, y(t_n + h/2)).$$  (15)

But how do we find $k_1 \approx y(t_n + h/2)$? For an estimation we use an explicit Euler step to get

$$k_1 = Y_n + \frac{h}{2} f(t, Y_n)$$  (16)

$$Y_{n+1} = Y_n + hf(t + h/2, k_1),$$  (17)

the so called explicit midpoint rule. The estimation by forward Euler, although not very accurate, is good enough, as the function evaluation is multiplied by the timestep to advance to the next approximation. So by a taylor expansion we find a local error of $O(h^3)$ leading to a global error of

$$Y_n - y(t_n) = O(h^2)$$  (18)

for the explicit midpoint rule.

Generalizing the idea of using function evaluations at $s$ intermediate points $t + c_j h$ leads to Runge-Kutta methods. They are defined by a Runge-Kutta matrix $(a_{ij})$, weights $b_i$, abscissae $c_j$ and the equations

$$k_i = Y_n + h \sum_{j=1}^{s} a_{ij} k_j'$$

$$Y_{n+1} = Y_n + h \sum_{i=1}^{s} b_i k_i'$$  (19)

The coefficient set can comfortably be specified as shown in table 1. If the matrix $(a_{ij})$ is strictly upper,

<table>
<thead>
<tr>
<th>$c_1$</th>
<th>$a_{11}$</th>
<th>$a_{12}$</th>
<th>...</th>
<th>$a_{1s}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_2$</td>
<td>$a_{21}$</td>
<td>$a_{22}$</td>
<td>...</td>
<td>$a_{2s}$</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
<td></td>
<td>...</td>
</tr>
<tr>
<td>$c_s$</td>
<td>$a_{s1}$</td>
<td>$a_{s2}$</td>
<td>...</td>
<td>$a_{ss}$</td>
</tr>
</tbody>
</table>

\[ b_1 \quad b_2 \quad ... \quad b_s \]

Table 1: General Runge–Kutta method

The most famous scheme is the method by Runge and Kutta given in table 2(b). Table 2(a) shows the explicit midpoint rule interpreted as a Runge-Kutta method. The method by Runge and Kutta possesses order 4.

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1/2</th>
<th>1/2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/2</td>
<td>1/2</td>
<td>0</td>
<td>1/2</td>
<td>1/2</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

(a) Midpoint

(b) RK4

Table 2: Explicit midpoint and "the" Runge-Kutta method.

By using algebraic relations for the coefficients, it is possible to construct explicit Runge-Kutta methods of arbitrary high order resulting in many inner stages with numerous evaluations. For most practical applications order 4 is sufficient.
(a) Example 1a: \( y' = 2y \)

(b) Example 1b: \( y' = -15y \)

(c) Example 2: \( y'' = -5y' - 5y \)

(d) Example 3

Figure 3: Work precision diagrams for the explicit Euler, explicit midpoint and RK4 methods.
Having constructed all these methods we apply them to our examples. The plots in figure 3 were produced by solving the examples using different timesteps and measuring the number of floating point operations needed for achieving the specified accuracy when compared with the (analytical) reference solution. In the work-precision diagram the $y$-axis shows the error $\|Y_{t_{\text{end}}} - y(t_{\text{end}})\|$ as a function of the required number of floating point operations. The first example with $\lambda = 2$ (figure 3(a)) shows exactly the expected behavior: when reducing the time step and thus investing more work, the numerical solutions converge towards the reference solution. Moreover the slope of the curves in the double logarithmic plot exactly matches the order of the method. In all other examples (figure 3(b)-3(d)) this behavior only shows up after an initial phase, where the solver produces completely wrong results. This is the point where stability comes into play. We will now analyze this by using the simplest example where it occurs, i.e. example 1 with $\lambda < 0$.

### 2.2 Stability

The equation for example 1 is called Dahlquist’s test equation

$$y' = \lambda y, \quad \lambda \in \mathbb{C}. \quad \text{(20)}$$

Its exact solution for an initial value $y(0) = y_0$ is given by

$$y(t) = e^{\lambda t} y_0. \quad \text{(21)}$$

This equation is a tool for understanding and evaluating the stability of integration methods. We have seen, that in the damped case characterized by $\text{Re}\lambda < 0$ convergence is only achieved for very small time steps. In this case, since the exponent is negative, the analytical solution is bounded for $t \to \infty$. Therefore one expects from a meaningful numerical method to deliver a bounded solution. An integration scheme that yields a bounded solution is called stable.

If we apply the forward Euler method with a fixed step size $h$ to (20), the n-th point of the numerical solution is given by:

$$Y_n = (1 + h\lambda)^n y_0 \quad \text{(22)}$$

It is bounded if and only if $|1 + h\lambda| < 1$, i.e. for $h\lambda$ in the unit ball around -1. A similar analysis can be carried out for the other methods and also results in restrictions of the admissible step size. This analysis explains the sharp bend in figures 3(b)-3(d). Only when the step size drops below a certain limit dictated by $\lambda$, i.e. by $h < \lambda^{-1}$ in case of the forward Euler method, the numerical solutions can converge. If the damping is increased, i.e. $\text{Re}\lambda \to -\infty$, then for the explicit Euler necessarily $h \to 0$ for the solution to be stable. This means the step size is artificially limited and it cannot be increased beyond the stability limit. This limits the flexibility of balancing work against accuracy.

### 2.3 The implicit Euler method

To construct a method that better suits our needs we go back to (7) and substitute the differential quotient by a backward difference quotient for $y(t + h)$

$$\frac{y(t + h) - y(t)}{h} \approx y'(t + h) = f(t + h, y(t + h)). \quad \text{(23)}$$

This results in the integration formula

$$Y_{n+1} = Y_n + hf(t + h, Y_{n+1}), \quad \text{(24)}$$

the so called backward or implicit Euler method. As its explicit variant this method can be shown to have order 1. Now the numerical solution only is given implicitly by the solution of the possibly nonlinear equation

$$Y_{n+1} - hf(t + h, Y_{n+1}) - Y_n = 0. \quad \text{(25)}$$

If we apply this method to the Dahlquist equation we get the recurrence formula

$$Y_n = (1 - h\lambda)^{-n} y_0. \quad \text{(26)}$$

The numerical solution $Y_n$ remains bounded for $|(1 - h\lambda)^{-1}| < 1$. If we assume $\lambda < 0$, this holds for arbitrary $h > 0$. Thus there is no restriction on step-size, the method is unconditionally stable. Figure 5 shows the work-precision diagrams for the implicit Euler method and our examples. We observe that we never loose stability and we especially do no miss the
solution by several orders of magnitude compared to the explicit methods. Of course the method loses accuracy when the time steps become large.

As a useful tool for visualizing the stability properties of a method we define the stability region $S$ to be the set of parameters, for which the integration method yields a bounded solution:

$$S := \{ z := h\lambda \in \mathbb{C} : \text{the numerical integration of equation (20) with step size } h \text{ and parameter } \lambda \text{ is stable} \}. \quad (27)$$

Methods that contain the complete left half-plane in $S$ are called $A$-stable or unconditionally stable. They are well suited for the stable integration of stiff equations. Obviously, the implicit Euler scheme is $A$-stable, whereas its explicit counterpart is not. The stability regions of all presented methods are shown in figure 4.

After reviewing the process that led us to the definition of the stability region, we can outline a more general idea that will allow us to determine the stability of more complex methods. The idea for analysing both Euler methods applied to (20) was to find a closed expression describing the stability function $\mathcal{R}$. This function maps the initial value $y_0$ to the value $Y_1$, performing a single step of the method

$$\mathcal{R} : y_0 \mapsto Y_1. \quad (28)$$

Thus $Y_n = R(h\lambda)^n y_0$. For the explicit Euler method we found in (22)

$$\mathcal{R}(z) = 1 + z, \quad (29)$$

for the implicit version in (26)

$$\mathcal{R}(z) = \frac{1}{1 - z}. \quad (30)$$

The definition for the stability region now reads

$$S = \{ z \in \mathbb{C} : |\mathcal{R}(z)| < 1 \}. \quad (31)$$

2.4 Methods of higher order

To find a higher order method, we go back to equation (15) and insert a linear interpolation term for $y(t + h/2)$. The resulting formula is taken as an implicit definition of $y(t + h)$. We get the implicit midpoint rule

$$Y_1 = Y_0 + hf \left( t + \frac{h}{2}, \frac{Y_1 + Y_0}{2} \right), \quad (32)$$

using a simplified notation for advancing one step, i.e. writing $Y_0$ and $Y_1$ instead of $Y_n$ and $Y_{n+1}$.

Alternatively the midpoint rule can be derived as a collocation method with $s = 1$ internal nodes, i.e. by constructing a polynomial interpolating the particle trajectories at a given, fixed set of $s$ nodes. This idea allows for the construction of implicit Runge-Kutta methods with arbitrary order. In contrast to explicit methods the matrix $(a_{ij})$ ceases to
be strictly lower triangular. These methods are computationally more expensive, so we just stick to the midpoint rule. Its stability function is given by
\[ R = \frac{1 + z/2}{1 - z/2} \] (33)
As \( R \leq 1 \) for any \( \text{Re} \, z < 0 \) the implicit midpoint rule is \( A \)-stable.
As another possible choice we now introduce multi-step methods. They are computationally inexpensive because they have no inner stages and some of them are \( A \)-stable. A multistep method with \( k \) steps is of the general form
\[ \sum_{j=0}^{k} \alpha_j Y_{k-j+1} = h \sum_{j=0}^{k} \beta_j f_{k-j+1}, \] (34)
with \( f_{n+j} := f(t_{n+j}, Y_{n+j}) \). Here we also have ‘history points’ with negative indices. The coefficient \( \alpha_k \) is required to be nonzero; for variable time step sizes the coefficients depend on the last stepsizes, which we have omitted here for the ease of demonstration. Important special cases are the class of \( Adams \) methods where \( \alpha_0 = \cdots = \alpha_{k-2} = 0: \)
\[ Y_1 = Y_0 + h \sum_{j=0}^{k} \beta_j f_{k-j+1} \] (35)
and the class of \( BDF \)-methods (backward differentiation formulas) with \( \beta_0 = \cdots = \beta_{k-1} = 0: \)
\[ \sum_{j=0}^{k} \alpha_j Y_{k-j+1} = h \beta_k f_1. \] (36)
If the formula involves the right-hand side \( f_1 \) at the new approximation point \( Y_1 \) the method is said to be implicit. \( BDF \)-methods are always implicit. The coefficients can again be constructed by a collocation approach. \( BDF \)-methods exist up to order 6, higher order methods loose consistency for any choice of coefficients[12].

The stability regions of implicit and explicit \( Adams \) methods are bounded and located around the origin, thus they are not interesting for large time steps. \( BDF \)-methods were the first to be developed to deal with stiff equations and possess an unbounded stability region covering a sector within the negative complex half-plane. Therefore they are among the most widely used methods today. For \( k+1 \) points, these methods possess order \( k+1 \) and only one nonlinear system has to be solved, whereas \( s \) coupled systems have to be solved for an \( s \)-stage implicit Runge-Kutta method.

The \( BDF \)-method for \( k = 1 \) is just the implicit Euler method, for \( k=2 \) the method is given as
\[ Y_1 = \frac{4}{3} Y_0 - \frac{1}{3} Y_{-1} + \frac{2}{3} h f(t+h, Y_1) \] (37)
The coefficients for higher order methods are given in table 3. The stability region of \( BDF(2) \) and the other implicit methods are shown in figure 4.

<table>
<thead>
<tr>
<th>( \alpha_0 )</th>
<th>( \alpha_1 )</th>
<th>( \alpha_2 )</th>
<th>( \alpha_3 )</th>
<th>( \alpha_4 )</th>
<th>( \alpha_5 )</th>
<th>( \alpha_6 )</th>
<th>( \beta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2</td>
<td>-2</td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>-3</td>
<td>-2</td>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>-4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>-5</td>
<td>5</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>-6</td>
<td>10</td>
<td>7</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 3: BDF Methods

### 2.5 The Verlet method

As a last method we will discuss a scheme commonly referred to as leapfrog or \( Störmer-Verlet \) method. It is especially efficient if (5) is given as the second order system
\[ x''(t) = f_v(t, x(t)), \] (38)
i.e. \( f_v(t, x(t), x'(t)) = f_v(t, x(t)) \). It is not applicable to general first order systems of the form (7).

To derive it, we use centered differences at a staggered grid (figure 6) i.e. we now approximate \( v \) at \( t+(2i+1)h/2 \) and \( x \) at \( t+ih \) by centered differences
\[ \frac{v_{n+1/2} - v_{n-1/2}}{h} = f(x_n) \] (39)
\[ \frac{x_{n+1} - x_{n}}{h} = v_{n+1/2} \] (40)
(a) Example 1a: $y' = 2y$

(b) Example 1b: $y' = -15y$

(c) Example 2: $y'' = -5y' - 5y$

(d) Example 3

Figure 5: Work precision diagrams for the implicit Euler, implicit midpoint, BDF(2) and Verlet (whenever possible) methods. The results for Euler are for comparison and the same as in figure 3.
thus

\[ v_{n+1/2} = v_{n-1/2} + hf(x_n) \]
\[ x_{n+1} = x_n + hv_{n+1/2}. \]

The method possesses order 2 as one can see by substituting (41) into (42) resulting in the second order centered difference

\[ \frac{x_{n+1} - 2x_n + x_{n-1}}{h^2} = f(x_n). \]

From this equation an alternative formulation of the Verlet scheme as a multistep method can be derived

\[ v_n - v_{n-1} = hf(x_n) \]
\[ x_{n+1} - x_n = hv_n, \]

which omits the half steps and staggered grids from above. Now for second order equations which do not possess the form of (38) one may replace \( f(x_n) \) by \( f(x_n, v_{n-1}) \) at the expense of some stability. Correctly the replacement had to be with \( f(x_n, v_n) \) but this would result in an implicit method. Now we can apply the method to examples 2 and 3.

2.6 Methods used in clothing literature

Terzopoulos et. al. [24] used a finite element formulation and a simple implicit Euler scheme to solve the arising ODE. Later publications focused on explicit integration methods, e.g. Eberhardt et al. [6] preferred RK4 and the Burlisch-Stoer extrapolation method as suggested in Press et al. [20]. Volino [4] used an explicit midpoint rule.

Implicit methods again became popular with the work of Baraff and Witkin. They used a linearized implicit Euler method and achieved simulations about an order of magnitude faster than explicit
methods. Although nonlinear constraints are formulated in the model, they only use their linear approximation to obtain a linear system of equations. This way the system to be solved in each time step also becomes linear and can be solved efficiently by a cg-method. This method corresponds to the solution of a nonlinear system with only one Newton iteration. Because the nonlinear part is not integrated, with high stiffness one may encounter similar slow downs as observed by Volino[28] and Eberhardt[7].

Provot[21] proposed a simple model only incorporating linear springs, combining it with an explicit method. This model was used by Desbrun et. al. [5] who also use only a linearized implicit method. But instead of linearizing the whole system they split it in a linear and nonlinear part and use a precomputed inverse of $A$ for solving the linear part of the equations. They don’t aim at solving the equation completely, as they don’t integrate the nonlinear term explicitly. Instead the angular momentum is corrected to account for the nonlinear part. With this algorithm one can neither change the stepsize $h$ nor deal with an $A$ depending on $t$.

Based on this work Kang et al.[17] did some further simplification to avoid solving the linear system. In order to update the solution vector in one step they divide by diagonal entry of the matrix of the linear system. Therefore they just do a single iteration of a Jacobi-like scheme for solving the linear equations. Again this may lead to artificial slowdowns.

Recently more advanced methods gain importance. Hauth et. al.[15] used BDF and the implicit midpoint rule, and Choi and Ko[3] also used BDF, both combining it with an iterative cg solver. In Hauth et. al. [16] there the complete BDF-2 algorithm, including variable step sizes, is presented in pseudocode, derived in the presented framework.

2.7 Selecting an efficient method

Which method is best for a certain application? This question is nearly impossible to answer a priori. The only choice is to try a set of methods and to evaluate which one performs best. Choosing the methods to try, though can be done based on theoretical considerations and observations of the problem at hand. A possible strategy is shown in figure 8.

The same statement holds for predicting the efficiency of a method. Generally implicit methods require more work per step. On the other hand one may be able to use time steps that are several magnitudes larger than the ones explicit methods would allow. Although accuracy will suffer, the integration won’t be unstable (see figure 2(b) for example 3). If evaluations of the right hand side function are cheap, a step with RK(4) is faster than an implicit step with BDF(4). On the other hand if it is cheap to compute a good sparse approximation to the jacobian, it may be more efficient to solve the linear system with a few cg iterations than to perform 4 full function evaluations.

3 Solving nonlinear systems

All implicit methods require the solution of a nonlinear system. The implicit Euler method for example reduces our integration problem to the solution of the nonlinear system

$$Y_1 - hf(Y_1) - Y_0 = 0.$$  (46)

The other methods yield a system of similar form, namely

$$Y_1 - hf\left(\frac{1}{2}(Y_1 + Y_0)\right) - Y_0 = 0$$  (47)

$$Y_1 - \frac{2}{3}f(Y_1) + \left(-\frac{2}{3}Y_0 + \frac{1}{3}Y_{-1}\right) = 0$$  (48)

for the midpoint and BDF(2) rule, respectively. This is a nonlinear system of dimension $6N$. It must be solved with Newton’s method to allow arbitrary step sizes independent of $\lambda$. Simpler methods for nonlinear systems would compensate the advantage of A-stability because the number of iterations would increase proportionally to the stiffness parameter $|\lambda|$. We now will work out an approach for implementing Newton’s method efficiently.

3.1 Newton’s method

For the nonlinear system $G(Y) = 0$ we compute a numerical solution by the following algorithm:
The error of the iterative solution of the linear system is formulated in terms of the residual, which is easily computationally accessible, whereas the actual error cannot be computed. The tolerance of the linear iteration is decreased proportionally to the monotonically decreasing residual of the nonlinear iteration. An analysis of this method[22] shows that it converges under rather weak additional assumptions. If the classical Newton method converges and the scalar tolerances $\eta^{(k)}$ are uniformly bounded by an $\eta < 1$, the inexact method converges. In literature the $\eta^{(k)}$ are referred to as forcing terms. Note that this additional assumption is also necessary: For $\eta = 1,$
s^{(k)} = 0 would be admissible and the iteration would stagnate.

The inexact method then at least converges linearly, whereas Newton converges superlinearly. By choosing the \( \eta_k \) to converge to zero sufficiently fast[22], the convergence of the inexact Newton method can be forced to have an order \( > 1 \). In a neighbourhood of the solution the convergence usually speeds up. By extrapolating the solution of the previous time step we obtain a good initial value for the new solution and the method converges quickly using the constant bound \( \eta = 0.02 \) without imposing a too strict tolerance on the linear solver.

### 3.3 Inexact simplified Newton methods

The efficiency of the Newton method can be further improved by another approximation. In the simplified version of Newton’s method the Jacobian \( J^{(k)} \) is approximated by \( J^{(0)} \). Such a scheme can be rewritten in the form of an inexact Newton method, if the linear system is written as follows and \( J \) is chosen as approximation to \( J^{(k)} \)

\[
J s^{(k)} = -G(Y^{(k)}) + (J - J^{(k)}) s^{(k)} + r^{(k)}
= -G(Y^{(k)}) + \tilde{r}^{(k)} \tag{49}
\]

The residual \( r^{(k)} \) is replaced by the larger \( \tilde{r}^{(k)} \), which can be bounded if \( J \approx J^{(k)} \). By choosing \( \tilde{\eta}^{(k)} \) appropriately, the method still converges. In fact, we trade some accuracy approximating \( J^{(k)} \) against accuracy in solving the linear system and up to a certain limit the method still behaves as before.

This degree of freedom can be further exploited be even not computing \( J^{(0)} \) but a sparser approximation of it. Hauth et. al.[15] exploit the idea previously used by Desbrun et al.[5] and approximate the Jacobian by the linear expression

\[
F_{\text{lin}}(x, v) = \sum_{j \in E} \left( \frac{k_{ij}}{l_{ij}} (x_i - x_j) + \frac{d_{ij}}{l_{ij}} (v_i - v_j) \right). \tag{50}
\]

This choice of the Jacobian has two major advantages over the full Jacobian. First, \( J \) is inexpensive to compute and only changes when either the material constants or the step size changes. Second, we reduce the entries in the Jacobian to approximately a third of the entries in the sparsity pattern of the full Jacobian. Hence an iteration of the linear solver only requires a third of the original time. Obviously this is a major speed-up for the solver. The resulting algorithm is surprisingly simple.

#### Algorithm 3: Inexact Simplified Newton’s Method

1. Compute \( J \approx \frac{1}{\Delta t} G(Y^{(0)}) \).
2. for \( k = 1, 2, \ldots \) until convergence do
   3. Compute \( G(Y^{(k)}) \).
   4. Find \( s^{(k)} \) with \( J s^{(k)} = -G(Y^{(k)}) + r^{(k)} \), such that \( \|r^{(k)}\| \leq \tilde{\eta} k \| G(Y^{(k)}) \| \).
   5. Update \( Y^{(k+1)} := Y^{(k)} + s^{(k)} \).

#### 3.4 Adaptive time stepping

Newton’s method can also be used to control the step size of the ODE solver. If the convergence of Newton’s method is poor, the time step \( h \) is reduced such that the solution of the previous time step is a better start value for the current time step and achieves a faster convergence. On the other hand, the number of Newton iterations necessary is a criteria for the behaviour of the integrator and has already been used for an order selection algorithm[14] of the Radau integrator. In our implementation the number of Newton iterations decides whether to increase or decrease the time step \( h \).

### 4 Linear Systems

In a last section we briefly present the last part of the puzzle, the solution of the remaining nonlinear system. We restrict the discussion to the most important iterative method used, the method of conjugate gradients. A more profound discussion can be found in the standard texts of Trefethen[25], Greenbaum[11] or Saad[23]. It uses only matrix-vector multiplications to solve a symmetric system of the form

\[
Ax = b, \tag{51}
\]
with \( A \in \mathbb{R}^{N \times N} \), \( x, b \in \mathbb{R}^N \). Thus the expenses for a matrix inversion is reduced from \( O(N^3) \) to a hopefully small number of multiplications with a complexity of \( O(N^2) \). In the present application, \( A \) is the Jacobian arising in Newton’s method, e.g. \( A = I - h \frac{\partial}{\partial Y} f(Y_0) \) for the implicit Euler method.

4.1 Preconditioning

What lets Krylov methods like cg fly, is the use of a cheaply invertible preconditioner \( M \approx A \) and the solution of

\[
M^{-1}Ax = M^{-1}b, \tag{52}
\]

which is equivalent to (51). The most prominent preconditioners in computer science literature are

- The diagonal preconditioner

\[
M := \text{diag}(A). \tag{53}
\]

This is the choice considered in the systems of Baraff[1] and Volino[27], also called diagonal scaling. This preconditioner is easily available and inexpensive to apply, as each application is just a division of each vector-component by the corresponding diagonal entry, resulting in \( N \) flops per application.

- The incomplete Cholesky factorisation (ICC) is computed by carrying out an approximate Cholesky factorisation of \( A \), i.e. formally factorising \( A \) and dropping all intermediate values not fitting the sparsity pattern. It is more expensive to compute and to apply, resulting in almost \( 2\text{nnz}(A) \) flops per application, where \( \text{nnz} \) denotes the set entries of \( A \).

- The successive-symmetric over-relaxation preconditioner (SSOR) is another iterative solving scheme for linear equations. The matrix formulation of one SSOR-step is given by the multiplication by

\[
M := (D - L)D(D - L^T), \tag{54}
\]

where \( A = D - L - L^T \) and \( D = \text{diag}(A) \), \( L \) strictly lower tridiagonal. Note that the inversion can be realised by inverting two triangular systems, as in practice one product with \( D \) is precomputed. Thus the SSOR-preconditioner is inexpensive to compute and the cost of an application is about the same as for the ICC-preconditioner.

- The block diagonal preconditioner, a generalisation of diagonal scaling where typically \( 3 \times 3 \) blocks of \( A \) are inverted.

4.2 Enforcing constraints for collision response

The incorporation of collisions distinguishes animation from classical problems in numerical analysis. The effects cannot be modelled a priori in the differential equation, since the collision reaction depends on the collisions that are detected in each time step. Therefore, the ODE solver has to incorporate a collision response according to the current collisions. This requires the ODE to be modified during run time in each time step.

A first and useful technique is to use constraints, i.e. to constrain the motion of a particle \( p_i \) during a time step in the direction \( c_i \), e.g. to allow no movement in the (negative) normal direction of a colliding surface.

Baraff and Witkin [1] presented a very efficient method to enforce constraints inside a cg-method. The cg method, like Newton’s method, is an update method. It starts with an initial guess and adds scalar multiples of a search direction. If we insure, that the initial guess obeys the constraints and each update is orthogonal to the constraint directions, then the final solution will also fulfill the constraints. Therefore in each iteration of the cg-method the new direction is filtered such that the solution does not alter in a constrained direction. We will give some theoretical background to illuminate this method.

Constraining positions and particles is equivalent to adding algebraic equations to the system. Therefore, a differential algebraic system has to be solved with different constraints (algebraic equations) in each step. Collision forces are given implicitly, i.e. the predefined positions and velocities result in constraint forces, which can be computed after each step.
The constraints can be implemented by multiplying the linear system matrix with the rank-deficient projector matrix
\[ P := I - \sum_i d_i d_i^T. \]
The vectors \( d_i \in \mathbb{R}^{3N} \) are constructed by inserting the constrained direction \( c_i \in \mathbb{R}^3 \) to constrain particle \( j \) at the position \( 3j \). Thus we seek an admissible solution of the system
\[ PAPx = Pb. \] (55)
Here the multiplication on the right restricts the forces to admissible forces, the left multiplication with \( P \) filters the velocities, the multiplication with \( A \) is the linearized transformation to accelerations, which are again filtered by \( P \). This system ceases to be positive definite, in fact it is nonnegative definite, but of course not of full rank. A unique solution exists if the system is expanded by the equation
\[ (I - P)x = 0, \] (56)
which states that the solution is admissible, i.e. has no components outside the nullspace \( \text{ker}(P) \). Effectively, the equality constrained least square problem
\[ \| PAPx - Pb \| \to \min, \quad \text{where } (I - P)x = 0 \] (57)
is solved. Since \( Pb \in \text{range}(PAP) \) the minimum is 0 and the cg method applied to \( PAP \) is able to find a solution of this singular linear system[10].

Algorithm 4: Filtered cg

while \( \| r \| \| b \| \leq \epsilon \)
\[ i = i + 1 \]
solve \( Mz = r \)
if \( i = 0 \)
\[ \rho = (r, z) \]
else
\[ \beta = \frac{\rho}{\rho_1} \]
\[ p = z + \beta p \]
\[ p = Pp \]
\[ q = Ap \]
\[ q = Pq \]
\[ \alpha = \frac{\rho}{(p,q)} \]
\[ v = v + \alpha p \]
\[ r = r - \alpha q \]
\[ \rho_1 = \rho \]
end

References


Abstract

We present an algorithm to efficiently and robustly process collisions, contact and friction in cloth simulation. It works with any technique for simulating the internal dynamics of the cloth, and allows true modeling of cloth thickness. We also show how our simulation data can be post-processed with a collision-aware subdivision scheme to produce smooth and interference free data for rendering.


Keywords: cloth, collision detection, collision response, contacts, kinetic friction, static friction, physically based animation

1 Introduction

Collisions are a major bottleneck in cloth simulation. Since all points are on the surface, all points may potentially collide with each other and the environment in any given time step. Moreover, for believable animation the number of particles is generally in the tens of thousands or more. Since cloth is very thin, even small interpenetrations can lead to cloth protruding from the wrong side. This is visually disturbing and can be difficult to correct after the fact either in the next time step or in post-processing. While rigid body simulations often have relatively few collisions per object (apart from resting contact), deformable bodies naturally give rise to large numbers of collisions varying in strength from resting contact to high speed impact. Two-dimensional manifolds like cloth are the worst case. Naïve methods for detecting and stopping every collision can quickly grind the simulation to a halt.

This paper presents a collision handling algorithm that works with any method for simulating the internal dynamics (i.e. stretching, shearing, and bending) to efficiently yet robustly produce visually complex motion free from interference as in figure 1. The key idea is to combine a fail-safe geometric collision method with a fast (non-stiff) repulsion force method that models cloth thickness as well as both static and kinetic friction. Ever since [Moore and Wilhelms 1988] proposed that repulsion forces are useful for contact whereas exact impulse-based treatment is useful for high velocity impact, authors have toyed with using both. For example, [Sims 1994] switched between instantaneous impulses for high velocities and penalty spring forces for low velocities to treat his evolving articulated rigid body creatures. Although similar in spirit to our approach, we always use both techniques in a fully hybridized and efficient manner.

We view repulsion forces, e.g. during resting contact, as a way to deal with this vast majority of collisions in a simple and efficient manner allowing us to use a more expensive but completely robust method to stop the few that remain. Since our repulsion forces handle most of the self-interaction, it is desirable to make them computationally efficient to apply. Therefore we propose using a repulsion spring model that is not stiff. In contrast, many authors use computationally expensive stiff repulsion springs, e.g. with force inversely
proportional to separation distance, since they do not have a robust alternative for stopping any remaining collisions. See e.g. [Moore and Wilhelms 1988; Lafleur et al. 1991; Baraff and Witkin 1998], although we note that some of the difficulties associated with the stiffness of the repulsion springs was partially alleviated by using an implicit method for the time integration in [Baraff and Witkin 1998].

Our robust geometric collision algorithm is the first scheme that guarantees no dynamic self-interference of cloth. [Moore and Wilhelms 1988] started in this direction proposing a hard-to-solve fifth order polynomial to detect point-face collisions. This was abandoned by the community until [Provot 1997] reduced it to a cubic and extended it to treat edge-edge collisions. However, Provot did not properly account for rounding errors, so self-intersection could still occur. Generally, these difficulties led the community to allow self-intersection, and then attempt to detect and correct it after the fact. For example, [Volino and Magnenat Thalmann 1995; Volino et al. 1995; Volino and Magnenat-Thalmann 1997; Volino et al. 2000] proposed a number of methods such as “most probable” orientations for collisions, i.e. nodes vote on which side they would like to be on and the majority wins. Although this gives no guarantee of physical consistency, or that the method even works, they did produce convincing simulations of a ribbon folding, garments crumpling in a dryer, and a stack of cloth. The complexity involved with unraveling self-intersection has led many to use large repulsion forces to keep the cloth well separated, but this leads to visual artifacts with cloth seemingly floating over itself at large distances with little or no friction. Since our method avoids nonphysical self-interference altogether, we do not need large repulsion forces or complicated and unreliable algorithms for detecting and fixing self-intersection. A further advantage over methods that allow self-intersection, even when they succeed in recovering from it, is that our cloth is always properly constrained. This is necessary to capture the bulk and small scale crumpling apparent in complex folds.

A key ingredient of our new algorithm is that we do not work directly with positions but only obtain positions by integrating velocities. Thus, given a current non-interfering state for our cloth, the collision handling problem can be reduced to finding velocities that guarantee a non-interfering state at the end of the time step. Moreover, given a current non-interfering state and a proposed set of new positions at the end of the time step, under a linear trajectory assumption we can compute a velocity to be used along with the initial state in our collision processing algorithm. This allows us to cleanly separate the time evolution of the internal cloth dynamics (and the environment around the cloth) from the collision processing algorithm. That is, the result of the time evolution is merely used as an initial guess for the final position of the cloth. Then this initial guess is modified to account for any collisions. This allows us to easily integrate our collision, contact and friction processing algorithm with a pre-existing cloth dynamics engine.

Our approach to static and kinetic friction is based on the repulsion forces and is trivial to apply even for cloth-cloth interaction. Correctly preventing self-intersection and modeling static and kinetic friction, especially for cloth-cloth contacts, is essential for producing the detailed time-evolving folds and wrinkles exhibited by cloth. Our treatment gives highly realistic folds and wrinkles as demonstrated in the figures.

2 Other Work

There is a rich history of research on contact and collisions in the graphics community, and we cannot possibly cover it all due to space limitations. However, we will mention a number of these works where appropriate throughout the text. For example, [Baraff and Witkin 1998; Provot 1995; Provot 1997; Volino et al. 1995; Jimenez and Luciani 1993; Moore and Wilhelms 1988] are cited a number of times.

Baraff carried out a detailed study of numerical methods for rigid body motion with contact and friction in a series of papers [Baraff 1989; Baraff 1990; Baraff 1991; Baraff 1993; Baraff 1994]. [Gourret et al. 1989] simulated a hand grasping an object using a finite element model for both the hand and the object. They detected collision as the overlapping of volumetric objects and subsequently treated collision, contact and friction based on the size of the overlap (including reactive repulsive forces). [Mirtich and Canny 1995] showed that one could produce physically plausible results modeling colliding, rolling, sliding, and resting contact for rigid bodies as a series of collision impulses, or micro-collisions. For example, a block sitting on a table experiences many micro-collisions keeping it from sinking into the table. They used an infinitesimal collision time, Poison’s hypothesis, and a Coulomb friction model. [O’Brien and Hodgins 1999] used a finite element model to simulate elastic brittle objects producing impressive animations of fracture. Collisions were detected via static interpenetration and resolved with penalty forces. Although the penalty forces could be stiff, they stated that their finite element model was relatively just as stiff dictating a small time step anyhow.

Although we use triangles to represent our surface, other representations may be used. [Herzen et al. 1990] addressed collisions between parametric surfaces, [Grinspun and Schröder 2001] worked with subdivision surfaces, and an implicit surface formulation of contact and collision processing was demonstrated in [Gas-cuel 1993; Desbrun and Gascuel 1994].

3 Cloth Model

Since this paper is concerned with collisions, particularly self-collisions, we do not address internal cloth dynamics. Those interested in cloth modeling are referred to the survey article of [Ng and Grimsdale 1996] and the book of [House and Breen 2000]. We also make specific mention of the CAD apparel system in [Okabe et al. 1992], the work of [Breen et al. 1994] using experimentally determined measurements for cloth properties, and the seminal papers of [Terzopoulos et al. 1987; Terzopoulos and Fleischer 1988a; Terzopoulos and Fleischer 1988b; Terzopoulos and Witkin 1988] on constitutive modeling of deformable bodies for computer graphics. In addition, [Baraff and Witkin 1998] proposed an implicit time stepping method and generated convincing results despite dropping nonsymmetric terms from their matrix. Further approximations were made in [Desbrun et al. 1999], e.g. violating local preservation of angular momentum, in order to obtain interactive rates while sacrificing a little realism.

For the purposes of demonstrating our collision handling, we use a simple mass-spring model for the internal cloth dynamics, as opposed to a true constitutive model. However, in figure 4 we also illustrate the efficacy of our approach with a highly sophisticated model from an industrial production system. In our basic model, particles are arranged in a rectangular array with structural springs connecting immediate neighbors. Diagonal springs provide shear support, and springs connected to every other node (with a stabilization spring attached to the center node in between) resist bending. We make the edges and corners of the cloth slightly heavier by giving all particles the same mass instead of a mass proportional to the area of the surrounding cloth. The heavier edges and corners give the cloth an attractive flare similar to that of real cloth where tailors often make hems a little heavier. This basic cloth model has been used by many authors, e.g. [Provot 1995].

1The equal mass assumption also simplifies many of the formulas presented in this paper. Generalizing to the unequal mass case is straightforward.
4 Limiting the Strain and Strain Rate

Sometimes triangles are undesirably stretched or compressed by large percentages. A rule of thumb in computational mechanics is that a triangle edge should not change length by more than 10% in a single time step, see e.g. [Caramana et al. 1998]. This can be enforced by either adaptively decreasing the time step or nonphysically decreasing the strain rate. This rule of thumb is generally used to obtain better accuracy, as opposed to stability, and thus it is used in conjunction with implicit time stepping algorithms as well, see e.g. [Baraff and Witkin 1998].

[Provot 1995] addressed this issue in a novel way processing the cloth after each time step with an iterative algorithm that repairs excessively deformed triangles. This algorithm focused on the overall strain as opposed to the strain rate (although [Provot 1995] mistakenly referred to this as the deformation rate). [Provot 1995] looped through the mesh changing the positions of the nodes on edges that had deformed by over 10%. Since adjusting the position of one node affects the length of all the edges containing it, an iterative process was used. Good results were obtained even though the algorithm does not converge in certain situations, e.g. when all the boundaries of the cloth mesh have constrained (fixed) positions that force an expansion beyond 10%. [Provot 1995] illustrated that this iterative method was significantly more efficient than arbitrarily increasing the spring stiffness when one is dissatisfied with the degree of mesh deformation in a numerical simulation.

Although this method seems to work well, it involves moving nodes and can therefore induce self-intersection in the mesh. Thus, to fit this method into the framework of our collision processing algorithm, we adjust velocities instead of positions. At each time step, we calculate the candidate new spring lengths using the current velocities. Then we apply momentum-conserving corrective impulses to the velocities to ensure that all springs are deformed by a maximum of 10% from their rest length at the end of the time step (ignoring bending springs). These impulses influence the future strains of surrounding springs, and thus an iterative procedure is needed to guarantee that no spring deforms to over 10% of its rest length during the time step. This is essentially equivalent to using biphasic springs with a much stiffer spring constant beyond 10% deformation, and the iterative procedure is similar to using implicit time stepping when the stiffer spring constant is activated. This mimics the physical behavior of cloth (and skin!), which offers little resistance to small deformations but becomes stiff after a critical deformation is reached.

We apply this deformation limiting procedure using a Jacobi iterative approach (parallel rather than sequential), and although convergence is not guaranteed, generally only one or two iterations per time step are required for visually pleasing results. Although a Gauss-Seidel iterative approach (sequential rather than parallel) generally converges faster, it can introduce a noticeable bias according to which parts of the cloth are updated first (although this can be mitigated to some degree by using random orderings). Moreover, Jacobi style iteration is easier to parallelize for high performance.

In addition to the strain, we also limit the strain rate according to the rule of thumb mentioned above. Although, this is usually done by adaptively reducing the time step, these smaller time steps can lead to a loss of efficiency. To avoid slowing the simulation, we continually monitor the strain rate and use momentum-conserving impulses to reduce it so that springs do not change their current length by more than 10% during a time step. This trade-off of accuracy for efficiency does not seem to induce any unwanted visual artifacts and is similar to the traditional damping of an implicit time discretization of the equations. We use a Gauss-Seidel iterative approach in order to accelerate convergence. Only a few iterations are needed as the fastest deforming edges are rapidly damped to reasonable deformation rates. Convergence is not required since we can still adaptively reduce the time step, and after only a few iterations only a moderate reduction of the time step is necessary. [Volina et al. 1995] proposed a philosophically similar technique that monitors local mechanical energy variations and artificially distributes kinetic energy through momentum transfers in regions where instability might occur. Similarly, [Baraff and Witkin 1998] used their implicit time integration scheme to automatically damp the local energy generated by their treatment of collisions.

5 Time Discretization

We cleanly separate the time evolution of the internal cloth dynamics (and the environment around the cloth) from the collision processing algorithm. This allows us to easily integrate our collision, contact and friction processing algorithms with a pre-existing cloth dynamics engine. Starting at time \( t = 0 \) with cloth positions \( \mathbf{x}^0 \) and velocities \( \mathbf{v}^0 \), the algorithm is as follows. For \( n = 0, 1, 2, \ldots \):

- Select a collision time step size \( \Delta t \) and set \( \mathbf{v}^{n+1} = \mathbf{v}^n + \Delta t \mathbf{a}^c \)
- Advance to candidate positions \( \mathbf{x}^{n+1} \) and velocities \( \mathbf{v}^{n+1} \) at time \( t^{n+1} \) with the cloth internal dynamics
- Compute the average velocity \( \mathbf{v}^{n+1/2} = (\mathbf{x}^{n+1} - \mathbf{x}^n) / \Delta t \)
- Check \( \mathbf{x}^n \) for proximity (section 6), then apply impulse impulses (section 7.2) and friction (section 7.3) to the average velocity to get \( \mathbf{v}^{n+1/2} \)
- Check linear trajectories from \( \mathbf{x}^n \) with \( \mathbf{v}^{n+1/2} \) for collisions (section 6), resolving them with a final midpoint velocity \( \mathbf{v}^{n+1/2} \) (sections 7.4 and 7.5)
- Compute the final positions \( \mathbf{x}^{n+1} = \mathbf{x}^n + \Delta t \mathbf{v}^{n+1/2} \)
- If there were no repulsions or collisions, set \( \mathbf{v}^{n+1} = \mathbf{v}^{n+1/2} \)
- Otherwise, advance the midstep velocity \( \mathbf{v}^{n+1/2} \) to \( \mathbf{v}^{n+1} \) (section 7.6).

When repulsions or collisions appear, our method for determining the final velocities is essentially central time differencing [Hughes 1987]. In fact, we use central time differencing for our internal cloth dynamics as well, though we stress that any reasonable algorithm could be used for that purpose, e.g. one large implicit time step as in [Baraff and Witkin 1998] or many small explicit Runge-Kutta steps.

The algorithm is stable for any collision time step \( \Delta t \), thus \( \Delta t \) can be chosen adaptively in a straightforward manner. For example, we choose a minimum \( \Delta t \) as the time step of the cloth dynamics evolution and a maximum \( \Delta t \) on the order of one frame, and start with the maximum. We halve the time step when an actual collision occurs, i.e. the repulsion forces aren’t adequate, and try the time step over again only doing the full collision processing at the minimum \( \Delta t \). Whenever we get three successful time steps in a row we double \( \Delta t \) again. Adaptive time stepping was also addressed in [Baraff and Witkin 1998].

6 Proximity and Collision Detection

To accelerate the detection of proximities for repulsions and of intersections for collisions, we use an axis-aligned bounding box hierarchy. It is built bottom-up once at the beginning of the simulation using the topology of the cloth mesh. In one sweep we greedily pair off adjacent triangles to get parent nodes, then in a sweep in the opposite direction pair off these to get the next level up, and so on alternating sweep directions until we are left with one root node.

At each time step we calculate the extents of the axis-aligned bounding boxes for the repulsion calculation (section 7.2) by taking a box around each triangle enlarged by the thickness of the cloth (e.g. \( 10^{-3} \)m or 1mm), and then taking the union of the extents in each axis direction as we move up the hierarchy. We also recalculate the hierarchy for each iteration of the collision algorithm (section 7.4).
by taking a box around each triangle and its candidate position at the end of the step (since we have to cover the path that the triangle takes during the time step) enlarged by the rounding error tolerance (e.g. \(10^{-m}\), again merging as we move up the hierarchy. In both cases, we get a set of candidates for interference by intersecting the box to be tested with the root of the tree; only if it overlaps do we recursively check its children, proceeding until we reach the leaves of the tree. The leaves we reach indicate on which triangles the actual geometry tests need to be performed. These tests break down into checking points against triangular faces and edges against other edges (naturally we don’t compare a point against the triangle that contains it, or an edge against an edge that shares an endpoint). For more details on hierarchical methods and bounding volume hierarchies, see [Hahn 1988; Webb and Gigante 1992; Barequet et al. 1996; Gottschalk et al. 1996; Lin and Gottschalk 1998]. Further pruning of unnecessary tests between adjacent patches in low curvature regions of cloth is possible, at least for static proximity tests, see [Volino and Magnenat-Thalmann 1994; Volino et al. 1995].

In what follows we use the shorthand \(\vec{x}_i\) to mean \(\vec{x}_i - \vec{x}_j\).

To check if a point \(\vec{x}_i\) is closer than some thickness \(h\) to a triangle \(\vec{x}_1 \vec{x}_2 \vec{x}_3\) with normal \(\vec{n}\) we first check if the point is close to the plane containing the triangle: \([\vec{x}_1 \vec{n}] < h\). If so, we project the point onto the plane and compute the barycentric coordinates \(w_1, w_2, w_3\) with respect to the triangle:

\[
\begin{bmatrix}
\vec{x}_1 \cdot \vec{x}_1 \\
\vec{x}_1 \cdot \vec{x}_2 \\
\vec{x}_1 \cdot \vec{x}_3 \\
\vec{x}_2 \cdot \vec{x}_2 \\
\vec{x}_2 \cdot \vec{x}_3 \\
\vec{x}_3 \cdot \vec{x}_3 \\
\end{bmatrix}
\begin{bmatrix}
w_1 \\
w_2 \\
w_3 \\
\end{bmatrix} = \begin{bmatrix}
\vec{x}_1 \cdot \vec{x}_3 \\
\vec{x}_2 \cdot \vec{x}_3 \\
\vec{x}_3 \cdot \vec{x}_3 \\
\end{bmatrix}
\]

\(w_1 + w_2 + w_3 = 1\).

These are the normal equations for the least-squares problem of finding the point \(w_1 \vec{x}_1 + w_2 \vec{x}_2 + w_3 \vec{x}_3\) in the plane closest to \(\vec{x}_4\). If the barycentric coordinates are all contained within the interval \([-\hat{\delta}, \hat{\delta}]\), where \(\hat{\delta}\) is divided by a characteristic distance of the triangle, the point is close.

To check if an edge \(\vec{x}_i \vec{x}_j\) is close to another edge \(\vec{x}_i \vec{x}_k\), we find the pair of points, one on each edge, that are closest and check their distance. The search for the closest pair is easy by checking for the degenerate case of parallel edges, i.e. if \([\vec{x}_1 \times \vec{x}_3]\) is smaller than a round-off tolerance. If so, it reduces to a simple one-dimensional problem. Otherwise, we find the points on the infinite lines that are closest via the normal equations:

\[
\begin{bmatrix}
\vec{x}_1 \cdot \vec{x}_1 \\
\vec{x}_1 \cdot \vec{x}_2 \\
\vec{x}_1 \cdot \vec{x}_3 \\
\vec{x}_2 \cdot \vec{x}_2 \\
\vec{x}_2 \cdot \vec{x}_3 \\
\vec{x}_3 \cdot \vec{x}_3 \\
\end{bmatrix}
\begin{bmatrix}
a \\
b \\
\end{bmatrix} = \begin{bmatrix}
\vec{x}_1 \cdot \vec{x}_3 \\
\vec{x}_2 \cdot \vec{x}_3 \\
\vec{x}_3 \cdot \vec{x}_3 \\
\end{bmatrix}.
\]

If these points are on the finite edges we are done, otherwise we clamp them to the endpoints. The point that moved the most during clamping is guaranteed to be the one that answers the question, and the second point is found by projecting the first onto the second infinite line and clamping to the finite edge. The direction pointing from one point to the other is taken as the “normal” vector. We also save their relative positions along the edges, i.e. the weights \(0 \leq a, b \leq 1\) so that the points are \(\vec{x}_1 + a \vec{x}_2\) and \(\vec{x}_1 + b \vec{x}_3\).

To detect a collision between a moving point and a moving triangle, or between two moving edges, we first find the times at which the four points are coplanar assuming they move with constant velocity over the collision time step as in [Moore and Wilhelms 1988; Provot 1997; Doghri et al. 1998]. The latter two showed this reduces to finding the roots of a cubic equation,

\[(\vec{x}_2 + t \vec{v}_2) \times ((\vec{x}_3 + t \vec{r}_3) \times (\vec{x}_4 + t \vec{r}_4)) = 0\]

Any roots outside of the interval \([0, \Delta t]\) are discarded, and then the remaining roots are checked in increasing order with proximity tests at time \(t\). If the elements are closer than a small rounding error tolerance (\(10^{-m}\) for our simulations, which is 1000 times smaller than the cloth thickness), we register a collision. We likewise check for proximity at the end of the time step, \(t = \Delta t\), in case rounding errors hide a collision at the boundary between two time steps. While earlier works neglected rounding error, our approach guarantees (if collisions are resolved) that the cloth is separated by at least the rounding error tolerance at every time step and never self-intersects during time steps.

### 7 Contact and Collision Response

#### 7.1 Interpolating within the cloth

We often need to deal with two points from the cloth, computing their relative velocity or applying an impulse to them. However, we cannot directly look up or alter the velocities of such points, and instead must deal with the corners of the triangle or endpoints of the edges.

To compute the velocity of a point interior to a triangle or edge we use linear interpolation, which is exact for affine deformations (i.e. translation, rotation, and affine shearing and scaling). In particular, if a point interior to a triangle \(\vec{x}_1 \vec{x}_2 \vec{x}_3\) has barycentric coordinates \(w_1, w_2, w_3\) (calculated during proximity or collision detection) its interpolated velocity is \(w_1 \vec{v}_1 + w_2 \vec{v}_2 + w_3 \vec{v}_3\), and similarly if a point interior to an edge \(\vec{x}_1 \vec{x}_2\) is the fraction \(a\) along the edge then its interpolated velocity is \((1-a)\vec{v}_1 + a \vec{v}_2\). Note that we are finding the velocity of a specific piece of material involved in a contact or collision, i.e. the weights \(w_i\) or \(a\) are fixed so their time derivatives do not appear.

If an impulse of magnitude \(I\) in direction \(\vec{n}\) needs to be applied to two points in the cloth (i.e. \(\vec{H}\) to the first and \(-\vec{H}\) to the second), we instead apply impulses to the triangle corners or edge endpoints, weighted by the barycentric coordinates, so that the desired change in relative (interpolated) velocity for the two points is achieved. For the point-triangle case, where an interior point of triangle \(\vec{x}_1 \vec{x}_2 \vec{x}_3\) with barycentric coordinates \(w_1, w_2, w_3\) is interacting with point \(\vec{x}_4\), we compute adjusted impulses

\[
\vec{I} = \frac{2I}{w_1 + w_2 + w_3} \vec{v}_{i}^{\text{new}} = \vec{v}_i + w_i(\vec{I}/m)\vec{n} \quad i = 1, 2, 3
\]

\[
\vec{v}_4^{\text{new}} = \vec{v}_4 - (\vec{I}/m)\vec{n}
\]

assuming all nodes have mass \(m\). For the edge-edge case where a point with relative position \(a\) along the edge \(\vec{x}_1 \vec{x}_2\) interacts with a point with relative position \(b\) along the edge \(\vec{x}_3 \vec{x}_4\), the adjusted impulses are

\[
\vec{I} = \frac{2I}{a^2 + (1-a)^2 + b^2 + (1-b)^2} \vec{v}^{\text{new}}_1 = \vec{v}_1 + (1-a)(\vec{I}/m)\vec{n}
\]

\[
\vec{v}^{\text{new}}_3 = \vec{v}_3 - (1-b)(\vec{I}/m)\vec{n}
\]

Weighting the impulses in this way introduces appropriate torques for off-center interactions as well as giving continuity across triangle boundaries, and converges to the expected formulas when the interior points approach mesh nodes.

#### 7.2 Repulsions

Resolving the tens of thousands of collisions that can readily occur in folding and contact situations can be prohibitively expensive. This is why repulsion forces are mandatory. They dramatically reduce the number of collisions usually eliminating them altogether making our collision processing algorithm not only tractable but efficient. Our cloth is given a realistic thickness, e.g. 1mm, and repulsion forces are only applied between pieces of cloth that have this close proximity. As discussed in section 6, we use an axis-aligned bounding box hierarchy to make this proximity detection
efficient. Proximity is determined for both point-triangle pairs and edge-edge pairs. If a pair is close enough, then two kinds of repulsion forces are applied. The first is based on an inelastic collision, and the second is a spring based force.

[Baraff and Witkin 1992; Baraff and Witkin 1994] discussed collision modeling for deformable bodies pointing out that when a discretized rod collides with a wall, the endpoint of the rod should come impulsively to rest, i.e. the endpoint is subject to a completely inelastic collision impulse. Subsequently the rod stores energy in compression and then expands, separating from the wall. The loading and unloading of the elastic rod models a completely elastic collision. They pointed out that inelasticity can only be introduced by adding damping forces internal to the rod that dissipate energy due to the collision. Other authors have used completely inelastic collisions as well, such as [Desbrun et al. 1999] and [Carignon et al. 1992] (who used completely inelastic collisions to remove the “kicks” generated by the repulsion springs of [Lafleur et al. 1991]). We take a similar approach, modeling all cloth-cloth collisions and cloth-object collisions using an identically zero restitution coefficient in Poisson’s hypothesis. The energy stored in deformations of our mass-spring model when one of the nodes abruptly comes to rest against an object in its path is released as the cloth restores itself, causing it to bounce. In fact, most real-world cloth-cloth and cloth-object collisions are fairly inelastic, so even with a zero restitution coefficient one should take care to monitor the energy stored within the cloth. We do this intrinsically by limiting the strain rate as discussed in section 4.

Since our repulsion forces are meant to dramatically reduce the number of subsequent collisions, we incorporate a completely inelastic collision into our repulsion force. Suppose two points in the cloth, one inside a triangle and one a node of the mesh or both interior to mesh edges, are close and have relative velocity \( v_N \) in the normal direction which is less than zero, i.e. they are approaching each other. (See section 7.1 above for details on interpolating velocities interior to triangles and edges, and section 6 for details on the normal direction used in point-triangle and edge-edge interactions.) To stop the imminent collision we apply an inelastic impulse of magnitude \( I_r = m v_N / 2 \) in the normal direction. (See section 7.1 for how we distribute the impulse to the mesh nodes involved.)

Since our cloth model includes a realistic cloth thickness, we would like to ensure that pieces of the cloth are well separated at a distance on the order of this cloth thickness. This helps cloth to slide over itself (and objects) without the excessive snagging caused by the discretization errors resulting from the representation of smooth surfaces with discrete triangles. When pieces of cloth are too close together, there is a compression of cloth fibers, and a second repulsion force is applied to model this compression. The repulsion force is proportional to the overlap beyond the cloth thickness \( h \) (e.g. 1mm). However, since our robust collision handling algorithm (see section 7.4) will stop every intersection, our spring repulsion force is limited to a maximum when the objects touch, thus avoiding problems with stiffness. Furthermore, we limit our repulsion force so that objects are never propelled outside this overlap region in a single time step. This not only helps to reduce stiffness, but allows cloth in contact to stay close enough together to feel repulsion forces in subsequent time steps. This is important since the repulsion force magnitude is used to model friction, and thus friction forces are also felt in future time steps producing stable folds and wrinkles that add to the visual realism. Many other authors have used spring based repulsion forces, see e.g. [Jimenez and Luciani 1993; Marhefka and Orin 1996], but their methods suffered from undue stiffness since an arbitrary amount of interpenetration was allowed to occur. Again, this is alleviated in our model by the robust geometric collision algorithm that stops interpenetration resulting in a bound on the magnitude of the spring based repulsion force.

The spring based repulsion force is modeled with a spring of stiffness \( k \). As a simple heuristic, we found that matching the stiffness of the stretch springs in the cloth gave good results. The overlap is

\[
d = h - (x_4 - w x_1 - w_2 x_2 - w_3 x_3) \cdot \hat{n}
\]

giving a spring force of \( kd \) in the direction \( \hat{n} \). Multiplying by \( \Delta t \) gives the impulse. As discussed above, we limit this so that the relative velocity change will reduce the overlap by at most some fixed fraction (e.g. 1/10) in one \( \Delta t \) time step. If the normal component of relative velocity \( v_N \geq 1d/\Delta t \) already we apply no repulsion, otherwise we compute the impulse magnitude

\[
I_r = -\min\left(\Delta t k d, m \left( \frac{1d}{\Delta t} - v_N \right) \right)
\]

and distribute it to the mesh nodes as explained in section 7.1.

The repulsion forces can either be applied sequentially or all at once in a parallel update. One drawback of the parallel update is that situations involving multiple interactions can lead to undesirable behavior, e.g. as impulses from multiple inelastic collisions are added together. This can be alleviated to some degree by keeping track of the number of interactions a node is involved in and dividing the resulting impulses by that number. Another remedy consists of multiplying the inelastic collision impulses by a suitable relaxation parameter less than one. We found \( 0.25 \) works fine though the algorithm seems fairly insensitive to small changes in this value.

### 7.3 Friction

We use Coulomb’s model for friction, both static and kinetic, with a single friction parameter \( \mu \). The repulsion force \( F_k \) from section 7.2 is the negative of the normal force \( F_N \) pressing the cloth together. Therefore a friction force, in the direction of the pre-friction relative tangential velocity \( \vec{v}_T^{pre} \) but opposite to it, has magnitude at most \( \mu F_N \). This integrates to an impulse of magnitude \( \mu F_N \Delta t \) in the same direction, and thus a change in the relative tangential velocity of at most \( \mu F_N \Delta t / m \) where \( m \) is the mass (assumed equal for all particles involved). If this is larger than \( \vec{v}_T^{pre} \), then either the cloth was slipping and stopped due to kinetic friction, or was stuck and shouldn’t be allowed to start slipping due to static friction. Either way the new relative tangential velocity should be zero. If not, we can simply subtract this off the magnitude of the relative tangential velocity to account for kinetic friction slowing down the slipping. This calculation can be simplified by noting that \( F_N \Delta t / m \) is just \( \Delta v_N \), the change in relative velocity in the normal direction, which can be directly calculated in the repulsion algorithm. Then the decrease in the magnitude of the relative tangential velocity is \( \min(\mu \Delta v_N, |\vec{v}_T^{pre}|) \), i.e. our final relative tangential velocity is

\[
\vec{v}_T = \max\left(1 - \mu \frac{|\Delta v_N|}{|\vec{v}_T^{pre}|}, 0 \right) \vec{v}_T^{pre}
\]

We apply impulses to achieve this for both point-face proximities and edge-edge proximities.

In effect, we are modeling frictional contact with micro-collisions. We note that if we applied our friction algorithm to rigid bodies, it would solve the inclined plane problem as discussed in [Mirtich and Canny 1995]. A similar algorithm for kinetic friction was proposed by [Jimenez and Luciani 1993] who also calculated a normal force \( F_N \) from the magnitude of their spring repulsion force and subsequently used it to evaluate their Coulomb friction model. For static friction they attached a spring between the closest points of two objects when the normal force is nonzero. This attractive spring models static friction until the force exerted by this spring reaches the threshold \( \mu F_N \). Beyond this, the spring is removed and only kinetic friction applies.
7.4 Geometric Collisions

Repulsion forces alone cannot ensure that no interpenetrations will occur since positions are only checked at discrete moments in time. For robust collision handling, the path of the cloth between time steps must be considered as discussed in section 6.

Some authors back up simulations in time to treat collisions in chronological order, e.g. [Hahn 1988]. When a single time step may have thousands of collisions and contacts, as is characteristic of highly deformable bodies like cloth, this is quite expensive and can grind the simulation to a halt. The problem was addressed for rigid bodies by [Mirtich 2000] who processed the rigid bodies in parallel using a timewarp algorithm to back up just the objects that are involved in collisions while still evolving non-colliding objects forward in time. This method works well except when there are a small number of contact groups which unfortunately is the case for cloth as the entire piece of cloth has every node in contact with every other node through the mass-spring network.

Instead of rewinding the simulation to process one collision at a time, we resolve them all simultaneously with an iterative procedure as did [Volino et al. 1995; Provot 1997; Milenkovic and Schmidt 2001]. This does not give the same result as processing the collisions in chronological order. However, there is enough uncertainty in the collision modeling and error in the cloth discretization that we are already guaranteed to not get the exact physically correct answer. Instead we will obtain a physically plausible solution, i.e. one of many possible physically correct outcomes which may vary significantly with slight perturbations in initial conditions or the inclusion of unmodeled phenomena such as interactions between fuzzy strands of cloth. More details on sampling plausible solutions according to probability distributions reflecting a number of factors can be found in [Chenney and Forsyth 2000] who addressed the related problem of multiple colliding rigid bodies. Simulating plausible motion in chaotic scenarios was also addressed by [Milenkovic and Schmidt 2001] who studied problems where large numbers of rigid bodies were in a single contact group and employed iterative collision processing techniques, phrased as optimization procedures, in order to adjust the positions of the bodies to avoid overlap.

As discussed in section 6, our geometric collision processing algorithm is activated either when a collision actually occurs or when geometry (points and faces or edges and edges) is in (too) close proximity at the end of a time step. Thus, we need to account for both approaching and separating objects when a “collision” is registered. If the geometry is approaching, we apply a completely inelastic repulsion impulse. Otherwise, if the geometry is already separating (as may happen at the end of the time step, i.e. a close call rather than a true collision), we apply a spring based repulsion force. See section 7.2 for more details on both of these.

The collision impulses can either be applied sequentially or all at once in a parallel update. Once again, in the case of a parallel update, situations involving multiple interactions can lead to undesirable behavior. This can once again (similar to the repulsion forces) be alleviated by dividing the resulting impulses by the number of interactions, or by using a suitable relaxation parameter less than one (e.g. 0.25).

While processing all the collisions that occurred during a time step, we may have inadvertently caused new secondary collisions to occur. In order to keep the cloth interference free, we must analyze the newly computed post-collision path of the cloth for possible collisions and process these as well. This means that the bounding box hierarchy needs to be adjusted to account for the new post-collision velocities. Then secondary collisions can be detected and corrected, etc., and the process continues until a final interference free state can be computed. Since relatively large bounding boxes that contain the moving triangles need to be recomputed for every iteration, and a cubic equation has to be solved for every possible collision, this may be expensive. Luckily, our repulsion forces tend to catch and treat almost all collisions making the iteration scheme here practical to apply even for high velocity cloth with many nodes and a high degree of folding and contact. Also, there are some multiple collision situations, such as a node sandwiched between two approaching triangles, that are resolved immediately if we apply impulses in parallel (but can iterate for a long time if they are applied sequentially instead, although Gauss-Seidel generally converges faster than Jacobi iteration). However, there are still situations where many iterations are required, so after a few iterations we switch to a failsafe method which quickly eliminates all collisions. We use the method proposed by [Provot 1997], but not followed through in the literature, possibly because the formulas proposed in [Provot 1997] do not give true rigid body motion. We give corrected versions below.

7.5 Rigid Impact Zones

[Provot 1997] proposed collecting the nodes involved in multiple collisions into “impact zones” which are treated as rigid bodies. This is justified by observing that when cloth bunches together friction will prevent most relative motion. Thus, after a few iterations of applying local impulses as outlined above, we instead switch to merging lists of nodes representing impact zones. Initially, each node in its own list. Then, when a point-face or edge-edge collision occurs, the lists containing the four involved nodes are merged together into one larger impact zone. The impact zones are grown until the cloth is collision free. The velocity of the nodes in the impact zone is derived from a rigid body motion that preserves linear and angular momentum. The formula for angular velocity given in [Provot 1997] is flawed, so we present a corrected version here.

To find the rigid body motion we first compute the initial center of mass of the impact zone and its average velocity

\[ \bar{x}_{CM} = \frac{\sum m_i \bar{x}_i}{\sum m_i}, \quad \bar{v}_{CM} = \frac{\sum m_i^{n+1/2} \bar{v}_i}{\sum m_i} \]

then the angular momentum of the nodes with respect to their center of mass

\[ \bar{L} = \sum m_i \left( \bar{x}_i - \bar{x}_{CM} \right) \times \left( \bar{v}_i^{n+1/2} - \bar{v}_{CM} \right) \]

and the \( 3 \times 3 \) inertia tensor of the current configuration of nodes (using \( \delta \) to represent the identity tensor)

\[ I = \sum m_i \left( \bar{x}_i^n - \bar{x}_{CM} \right)^2 \delta - \left( \bar{x}_i^n - \bar{x}_{CM} \right) \otimes \left( \bar{x}_i^n - \bar{x}_{CM} \right) \]

The rigid body angular velocity that would preserve angular momentum is \( \omega = \overset{-1}{I} \bar{L} \), so the new instantaneous velocity of node \( i \) is

\[ \bar{v}_{CM} + \omega \times \left( \bar{x}_i - \bar{x}_{CM} \right) \]

However, we want the average velocity over the time step of finite size \( \Delta t \), so that the update \( \bar{x}_i^{n+1} = \bar{x}_i^n + \Delta t \omega^{n+1/2} \) exactly corresponds to a rigid body motion, i.e. so that lengths and angles stay fixed. If this last condition is not enforced (it was not addressed in [Provot 1997]), then self-intersection can occur. Assuming that we can accept a small \( O(\Delta t) \) error in the axis and angle of the total rotation, we make the approximation that \( \omega \) stays constant over the time step. Then we find the fixed and rotating components of the position

\[ \bar{x}_F = \left( \bar{x}_i - \bar{x}_{CM} \right) \overset{\omega}{\delta}_{[0]} \overset{\omega}{\delta}_{[0]}, \quad \bar{x}_R = \bar{x}_i - \bar{x}_{CM} - \bar{x}_F \]

giving the final position

\[ \bar{x}_i^{n+1} = \bar{x}_{CM} + \bar{v}_{CM} + \bar{x}_F + \cos(\Delta t \omega) \bar{x}_R + \sin(\Delta t \omega) \overset{\omega}{\delta}_{[0]} \times \bar{x}_R. \]
The new average velocity is then \( v^{n+1/2} = (\bar{v}^{n+1} - \bar{v}^n) / \Delta t \).

Applied on its own, this impact zone method has a tendency to freeze the cloth into nonphysical clumps. However, a combination of our repulsion forces and the initial collision impulses tends to keep these impact zones small, isolated and infrequent. Moreover, once formed, they are short-lived as the repulsion forces tend to quickly separate the offending nodes.

### 7.6 Updating the Final Velocity

When there are repulsions or collisions, we need to update the velocity from \( v^{n+1/2} \) to \( v^{n+1} \). For central differencing we need to solve the implicit equation

\[
v^{n+1} = v^{n+1/2} + \frac{\Delta t}{2} a(t^{n+1}, x^{n+1}, v^{n+1})
\]

where \( a(t, x, v) \) is the acceleration.

In many cloth models, such as ours, the damping forces (hence accelerations) are linear in the velocities giving the linear system

\[
(f - \Delta t \frac{\partial a}{\partial v}) v^{n+1} = v^{n+1/2} + \frac{\Delta t}{2} a(t^{n+1}, x^{n+1}).
\]

Here \( \frac{\partial a}{\partial v} \) is the Jacobian matrix of accelerations with respect to velocities, and \( a(t, x) \) is the elastic component of acceleration, i.e., everything apart from damping. In any reasonable cloth model this matrix will be symmetric positive definite (or can be safely made so, see [Baraff and Witkin 1998]) after multiplying both sides by the mass matrix, i.e., converting velocities into momenta and accelerations into forces. Then the conjugate gradient algorithm (e.g., [Saad 1996]) can be used to solve for \( v^{n+1} \). For nonlinear damping Newton’s method can be used requiring similar linear solves.

As an alternative for the nonlinear case, or when solving the linear system proves too difficult, one can explicitly match the velocity forward in time. Starting with \( u_0 = v^{n+1/2} \) we advance

\[
u_{m+1} = u_m + k \bar{a}(v^{n+1}, x^{n+1}, u_{m+1})
\]

ending with \( v^{n+1} = u_M \) where \( \Delta t / 2 = Mk \). Each substep is resolved with fixed point iteration, starting with initial guess \( u_m^{(0)} = u_m \) for \( u_{m+1} \) and continuing with

\[
u_{m+1}^{(j+1)} = u_m + k \bar{a}(v^{n+1}, x^{n+1}, u_{m+1}^{(j)})
\]

for a given number of iterations. The substep size \( k \) is chosen with \( k \lambda_{max}(\bar{a}) < 1 \) in order to guarantee convergence. Note that this can be made more efficient by separating the calculation of acceleration into the elastic component \( a_e \) mentioned above (which does not change) and the damping component.

As mentioned in section 4, we monitor the strain rate in the cloth and introduce additional damping if necessary. This is important for dealing with especially difficult collision situations. While our algorithm can handle the penalty term, bad strain rates in the cloth due to the collision impulses can cause additional unwanted collisions in subsequent time steps slowing the simulation down and producing visually inaccurate results. In addition, when one node collides and impulsively changes its velocity so that it is dramatically out of sync with surrounding nodes, our strain rate limiting procedure puts the nodes back in sync reducing the velocity of surrounding nodes even though they have not yet been involved in a collision. This keeps our cloth from experiencing unnecessary stress and also increases the likelihood that repulsion forces will stop the surrounding nodes before they actually collide, again dramatically reducing the number of collisions that have to be dealt with.

### 8 Post-Processing with Subdivision

Sharp folds and wrinkles in the cloth mesh give undesirable angular features when rendered as plain triangles. For visually pleasing animations a smoother surface is desired. Some authors have directly simulated smooth surfaces instead of simple triangle meshes. For example, [Thingvold and Cohen 1992] used dynamic B-splines which allowed them to interactively refine the mesh in regions of interest associating the control points with their dynamic simulation mesh nodes. They derived a number of rules for when, where and how to refine, even detecting when mesh refinement would cause intersection and then either stopping refinement or backing up the simulation in time to avoid intersection. [DeRose et al. 1998] exploited the convex hull properties of their subdivision surface model of cloth to accelerate collision detection. [Grinspun and Schröder 2001] rigorously modeled thin manifolds with subdivision surfaces detecting collisions with their derived bounds on surface normals and refining the mesh as required to resolve them.

We propose a fast and simple yet collision-aware post-processing subdivision scheme to smooth our triangle mesh. Our post-processing scheme takes the existing intersection free simulation data and produces a finer, more detailed and smoother approximation to the manifold. We never have to back the mesh up in time or cease refinement, or even have to consider refinement at all in the simulation. Our algorithm efficiently works independently from the simulation on the positions recorded for each frame. In addition, each frame can be processed independently just as in a rendering pipeline.

Another motivation for our post-processing is found in [Howlett and Hewitt 1998] who addressed cloth collisions with volumetric objects. They ensured that cloth nodes remained outside the objects making the collision-handling algorithm faster and simpler, but allowed edge and face collisions with the objects. These were handled in a post-processing step before rendering where they added nonactive points and adjusted their positions to eliminate intersections. Although we treat cloth-object collisions in a more detailed manner, our subdivision approach naturally allows this clever “tie about it” strategy explained in [Baraff 2001] where small penetrations are allowed in the simulation but are corrected before rendering. [Howlett and Hewitt 1998] further processed their cloth in an attempt to preserve area, but we do not undertake this endeavor since the idea of subdivision is that it recovers the true geometry approximated by the coarse simulation mesh.

Our post-processing algorithm proceeds as follows. If we allowed intersection with objects in the simulation, we begin by adjusting the cloth positions in the given frame to eliminate them iterating back and forth with an adjustment to eliminate cloth-cloth intersections that those adjustments may have caused. We use the repulsions and collision impulses from sections 7.2 and 7.4. When this is finished, our original mesh is intersection free even accounting for rounding error. We then subdivide the mesh putting a new node at the midpoint of each edge. Since the original mesh was intersection free and the subdivided mesh lies exactly within the original mesh, the subdivided mesh is guaranteed to be intersection free as well.

Next we use the modified Loop subdivision scheme [Loop 2001] to find smoother positions for all the nodes of the subdivided mesh. Unfortunately, moving to these smoothed positions may create intersections. However, we can view the vector from a node’s original position to its smoothed position as a pseudo-velocity and apply our collision detection algorithms from section 6 to determine when the intersection would occur. We stop the nodes at that point (or just before that point to avoid difficulties with rounding error) as if they had inelastically collided. We of course need to check again to see if these adjustments to the smoothed positions caused new intersections. Typically only a few iterations are required to eliminate all
Figure 2: The friction between a rotating sphere and a piece of cloth draped over it creates a complex structure of wrinkles and folds. Intersections especially since the convex-hull property of the subdivision means intersections are unlikely in the first place. A solution is guaranteed to exist, since the new nodes can simply be left on the triangle they were created on. Once we have a smoothed but intersection free subdivided mesh, we can subdivide again continuing until the desired resolution is reached. Since the cloth is originally separated by a finite distance, but each step of subdivision smoothing moves the nodes exponentially less and less, we very quickly find no more adjustments need to be made.

We caution the reader that this post-processing technique performs exceptionally well because we use a fairly high resolution dynamic simulation mesh. The efficiency of our repulsion and collision processing algorithms allows the use of such a mesh, and we have not noticed any problems with visual artifacts. However, on a relatively coarser mesh, one should be aware of potential artifacts such as "popping" that might result from using this subdivision scheme.

9 Examples

We demonstrate several examples using our simple cloth model with highly complicated folding where most of the nodes (tens of thousands in the dynamics and hundreds of thousands after subdivision) are in close contact with each other as opposed to, say, the simple draping of a skirt about a mannequin. In figure 1, a curtain is draped over the ground and a sphere. Our biphasic spring model enables complex wrinkling and eliminates undue deformation. When the sphere moves up and away, the curtain flips back over on itself resulting in a large number of contacts and collisions. The highly complex structure of folds and wrinkles is stable due to our static friction model. When a second ball pushes through the complex structure eventually slipping underneath, the algorithm still efficiently and correctly resolves all contacts and collisions. Note how realistically the cloth unravels by the final frame.

Figure 2 illustrates our static and kinetic friction algorithm with a piece of cloth draped over a rotating sphere. Figure 3 shows a tablecloth draped over four table legs with no tabletop. The object-cloth contact is tricky due to the sharp corners of the legs particularly when a sphere descends through the cloth down onto the ground, but our repulsion forces prevent unnatural snagging. Simulation times were reasonable even for these complex examples. Typically, a piece of cloth with $150 \times 150$ nodes runs at about 2 minutes per frame on a 1.2GHz Pentium III. Finally, figure 4 shows the draping and folding of a robe around a digital character from a production animation system utilizing a number of our techniques.

10 Conclusions and Future Work

The synergy of efficient repulsion forces combined with robust geometric treatment of collisions has allowed us to efficiently simulate complex cloth motion. The prevention of self-intersection together with kinetic and static friction produces complex, yet stable folding and wrinkling unachievable by simpler approaches. In addition, our post-processing subdivides simulation data without introducing self-intersection resulting in even higher quality animations. Our algorithm makes few assumptions about the internal cloth dynamics, and thus can easily be incorporated into existing codes with advanced cloth models.

We are close to a fully parallel implementation exploiting the parallel nature of most of our scheme. Other areas we plan to develop include modeling different values for kinetic and static friction coefficients, adaptive meshing to better resolve folds, and optimization of the bounding volume hierarchy. Furthermore, we are eager to apply our techniques to characters with highly wrinkled loose fitting skin.

Two rather important problems that we have not addressed are the interactions between cloth with sharp objects and the behavior of cloth when trapped in between two solid deformable or rigid bodies. We refer the reader interested in sharp objects to the recent developments of [Kane et al. 1999; Pandolfi et al. 2002]. For the case of intersecting collision bodies additional technologies like the ones developed by Baraff, Witkin and Kass (Personal Communication 2002) are required.
References


BARAFF, D. 2001. Collision and contact. In SIGGRAPH 2001 Course Notes, ACM.


Figure 4: Frames from a production animation of a robe draped over a digital character.

11 Acknowledgement

Research supported in part by an ONR YIP and PECASE award N00014-01-1-0620. The first author was supported in part by a Stanford Graduate Fellowship.

The authors would like to thank Igor Neverov, Joseph Teran, and Neil Molino for their help producing the final animations, and Henrik Wann Jensen for the use of his dali rendering system.

The authors would like to thank Cliff Plumer, Andy Hendrickson, Sebastian Marino and Lucasfilm for their guidance and support, as well as the images in figure 4.


