

# **Investigations into the Simulation of Cloth**

Luu Huy Danh Vo

A thesis

in

The Department of Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements for the Degree of

Master of Computer Science at

Concordia University

Montreal, Quebec, Canada

June 2006

© Luu Huy Danh Vo, 2006



Library and  
Archives Canada

Bibliothèque et  
Archives Canada

Published Heritage  
Branch

Direction du  
Patrimoine de l'édition

395 Wellington Street  
Ottawa ON K1A 0N4  
Canada

395, rue Wellington  
Ottawa ON K1A 0N4  
Canada

*Your file* *Votre référence*  
*ISBN: 978-0-494-28957-0*  
*Our file* *Notre référence*  
*ISBN: 978-0-494-28957-0*

**NOTICE:**

The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

**AVIS:**

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protègent cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

---

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

  
**Canada**

# Abstract

## Investigations into the Simulation of Cloth

Luu Huy Danh Vo

The subject of cloth simulation has seen many advances since the mid 1980's. It would certainly be interesting to know what aspects can be simulated, how they are simulated, as well as what the limitations are. In this thesis, we set the goal to explore the task of building cloth simulators; and hopefully, identify any missing part and its solutions. We begin with the necessary overview of the topic and the reference of related works and their contributions. Two popular models are selected as the base for the construction of two respective cloth simulators. One model is mass-spring type while the second model referenced defines its mechanical forces from energy condition functions. Each model presented brings along a multitude of approaches and techniques to solve the different subtasks that the challenge of building a cloth simulator consists of: The internal mechanisms, the integration method, damping, constraint handling techniques, collision detection and response all figure among the tasks involved in cloth simulation. The two constructed simulators serve for testing and comparison purposes. The results and analyses allow the exploration of additional possible solutions in the quest of finding the most appropriate methods. Finally, we propose the definition of an additional mechanical property of cloth to help simulate the curves and wrinkles observed on real cloth. And, further discussions of possible solutions are presented for certain cloth behaviors and aspects of cloth simulation that were left unspecified previously.

# Table of Contents

<b>List of Figures</b>	vii
<b>List of Tables</b>	xi
<b>CHAPTER 1: INTRODUCTION.....</b>	<b>1</b>
<b>1.1 OVERVIEW .....</b>	<b>1</b>
<b>1.2 MOTIVATION.....</b>	<b>1</b>
<b>1.3 OBJECTIVES AND METHODOLOGY.....</b>	<b>2</b>
<b>1.4 THESIS ORGANIZATION.....</b>	<b>3</b>
<b>CHAPTER 2: BACKGROUND .....</b>	<b>5</b>
<b>2.1. WHAT IS CLOTH? .....</b>	<b>5</b>
<b>2.2. OVERVIEW OF THE TERMINOLOGIES AND TOPICS INVOLVED.....</b>	<b>7</b>
<b>2.3. THE PROBLEM DESCRIPTION: SUBTASKS .....</b>	<b>10</b>
<b>CHAPTER 3: RELATED WORK - OVERVIEW .....</b>	<b>17</b>
<b>CHAPTER 4: RELATED WORK – DETAILED DISCUSSION OF ALGORITHMS.....</b>	<b>23</b>
<b>4.1. PROVOT’S MODEL.....</b>	<b>23</b>
4.1.1. The three types of springs .....	23
4.1.2. The Internal Dynamics, the External Forces, and the System Evolution.....	26
4.1.3. The “super-elasticity” problem .....	28
4.1.4. Collision detection and response .....	29
<b>4.2. BARAFF &amp; WITKIN’S MODEL .....</b>	<b>40</b>
4.2.1. The Condition Functions .....	41
4.2.1.1 Stretch Condition.....	41
4.2.1.2 Shear Condition.....	43
4.2.1.3 Bend Condition .....	43

4.2.2. Forces and force derivatives .....	44
4.2.3. Implicit Integration .....	45
4.2.3.1 The accuracy problem with larger time steps.....	46
4.2.3.2 Deduction of an Implicit System.....	47
4.2.4. Damping Forces.....	50
4.2.5. Constraint handling, collision detection and response.....	51
4.2.6. Adaptive time stepping .....	54
4.2.7. A modified Conjugate Gradient Method .....	55
<b>CHAPTER 5: DESIGN .....</b>	<b>56</b>
<b>5.1. SIMULATOR 1: A MASS-SPRING MODEL .....</b>	<b>56</b>
<b>5.2. SIMULATOR 2: AN ENERGY-CONDITION FUNCTION MODEL .....</b>	<b>60</b>
<b>CHAPTER 6: IMPLEMENTATION DETAILS.....</b>	<b>62</b>
<b>6.1. DETAILS OF SIMULATOR 1 .....</b>	<b>62</b>
<b>6.2. DETAILS OF SIMULATOR 2 .....</b>	<b>73</b>
<b>CHAPTER 7: RESULTS .....</b>	<b>81</b>
<b>7.1. COMPARING THE TWO SIMULATORS.....</b>	<b>81</b>
7.1.1. Some Test Results.....	81
7.1.2. Visual Differences and Behavioral Dissimilarities.....	120
7.1.3. Quantitative analysis.....	127
<b>7.2. ALTERNATIVE NUMERICAL INTEGRATION METHODS.....</b>	<b>133</b>
7.2.1. The 4 <sup>th</sup> Order Runge-Kutta Integration Method .....	136
7.2.2. The Verlet Integration Scheme .....	138
7.2.3. The Mixed Explicit/Implicit Integration method.....	141
7.2.4. A Different Approach to the Instability Problem.....	143
<b>7.3. ADDITIONAL COLLISION DETECTION AND RESPONSE SOLUTIONS .....</b>	<b>153</b>
7.3.1. An Easy Point-Triangle Collision Detection Method.....	155
7.3.2. The Fast Elimination-based Test for Triangle-triangle Intersection .....	159
<b>7.4. THE 4TH CLOTH MECHANICAL PROPERTY: PROXIMITY INDUCED CURVATURES</b> <b>.....</b>	<b>162</b>
7.4.1. The missing cloth property .....	164

7.4.2. Studying and modeling the behavior .....	166
7.4.3. The Theta rule.....	170
7.4.4. Energy Minimization & the four improvements.....	175
7.4.5. A Better Compression Energy Minimization.....	179
7.4.6. Computing k1 and k2.....	190
7.4.7. The pros and cons of this new formulation.....	193
<b>7.5. ORTHOGONAL RESISTANCE .....</b>	<b>195</b>
7.5.1. The mass-constraint mechanism.....	197
7.5.2. The realignment assumption .....	202
<b>7.6. WRINKLES AND OTHER CONSIDERATIONS FOR GARMENT ANIMATION .....</b>	<b>212</b>
7.6.1. The Additional Requirements for Simulating Complex Garments.....	212
7.6.2. Past Strategies for Simulating Wrinkles .....	213
7.6.3. Hypotheses about wrinkles in a particle-based model.....	215
<b>CHAPTER 8: CONCLUSIONS AND FUTURE WORK.....</b>	<b>221</b>
<b>BIBLIOGRAPHY .....</b>	<b>227</b>
<b>APPENDIX A .....</b>	<b>230</b>
<b>APPENDIX B .....</b>	<b>237</b>
<b>APPENDIX C .....</b>	<b>243</b>

## List of Figures

Figure 2.1. Interlacements of warps and wefts .....	6
Figure 2.2. The three basic weave structures .....	7
Figure 4.1. The three types of springs connected to a node: A) Structural springs, B) Shear springs, C) Flexion springs. ....	25
Figure 4.2. A mass-spring network of size m columns by n rows.....	26
Figure 4.3. The computation of a parent cone from two children cones. ....	34
Figure 4.4. The angle $\theta$ between the two normal vectors of adjacent triangles.....	44
Figure 5.1. From left to right: The mesh structure, the underlying spring system, and the rectangular piece of cloth simulated. ....	57
Figure 6.1. A component diagram for the first cloth simulator. ....	63
Figure 6.2. The component diagram for the second simulator. ....	76
Figure 7.1. The cloth in its initial state. ....	83
Figure 7.2. The first set of captured frames for test#1, simulator #1, at times: a) 1, b) 2, c) 3, d) 5, e) 7, f) 100 seconds.....	86
Figure 7.3. The second set of captured frames for test#1, simulator #1, at times: a) 3, b) 5, c) 7 seconds.....	88
Figure 7.4. A third set of captured frames showing instances of instability at times a) 13 and b) 14 seconds.....	89
Figure 7.5. A captured frame for test#1, simulator #1, after twenty-six seconds with damping.....	90
Figure 7.6. The system is in trouble when post-step modifications are active but damping is not, at times a) 3, and b) 8 seconds. ....	93
Figure 7.7. The system with inverse dynamics in place at times: a) 2, b) 4, c) 68 seconds. ....	95
Figure 7.8. The system with both post-step modifications and damping functions active. ....	96
Figure 7.9. Two frames captured when wind is added at times: a) 6 and b) 7 seconds....	97
Figure 7.10. The cloth viewed from different angles.....	98

Figure 7.11. The shear springs in action: a) 0, b) 1, c) 2, d) 4, e) 8 seconds. ....	100
Figure 7.12. The flexion springs in action: a) 0, b) 1, c) 2, d) 3, e) 7 seconds. ....	102
Figure 7.13. The limitation of flexion springs. ....	103
Figure 7.14. The first set of frames captured for test#1, simulator #2, simulating the stretch condition. ....	108
Figure 7.15. The second set of frames captured for test#1, simulator #2, simulating the shear condition. ....	112
Figure 7.16. The third set of frames captured for test#1, simulator #2, simulating the bend condition. ....	115
Figure 7.17. The three types of triangle pairs of the cloth mesh. ....	116
Figure 7.18. Nodes H and G are left out by the bend condition. ....	117
Figure 7.19. The cloth is much overstretched under the wind. ....	118
Figure 7.20. The second model behaves similarly to the first model under similar conditions. ....	119
Figure 7.21. A fourth type of triangle pairs to complete the bend condition. ....	121
Figure 7.22. A mesh structure with one more node per quad. ....	121
Figure 7.23. The interference of flexion springs on in-plane forces. ....	123
Figure 7.24. Type B' is nothing but type B for meshes rotated 180 degrees about the vertical axis. ....	124
Figure 7.25. A cloth quad elongated vertically. ....	124
Figure 7.26. A concern particular to defining internal forces over triangles instead of using springs. ....	127
Figure 7.27. The relationships of a node and its neighbors under a mass-spring model on the left, and under the condition model, on the right. ....	128
Figure 7.28. The two half-spaces. ....	145
Figure 7.29. A multiple-pass elongation-rate limiter. ....	147
Figure 7.30. The force transfer cone. ....	149
Figure 7.31. Testing the force transfer on two mesh sizes. ....	152
Figure 7.32. A real piece of cloth hung with its two top corners closer together than their "natural" flat distance. ....	165
Figure 7.33. An instance of cloth simulation without realistic out-of-plane motions. .	167



Figure 7.34. The angle theta of nodes on a same yarn or on a set of consecutive shear springs. ....	171
Figure 7.35. The strict application of the theta rule. ....	173
Figure 7.36. Less accentuated curves. ....	173
Figure 7.37. The strict application of the theta rule over a finer cloth mesh object. ....	174
Figure 7.38. Less accentuated curvature for a finer cloth mesh object. ....	174
Figure 7.39. Scenario 1.1.1.1 : the original formulation. ....	181
Figure 7.40. Scenario 1.1.1.1 : the new formulation. ....	182
Figure 7.41. Scenario 1.0.1.1 : the original formulation. ....	183
Figure 7.42. Scenario 1.0.1.1 : the new formulation. ....	183
Figure 7.43. Scenario 1.1.0.1 : the original formulation. ....	184
Figure 7.44. Scenario 1.1.0.1 : the new formulation. ....	185
Figure 7.45. Scenario 1.1.1.0 : the original formulation. ....	186
Figure 7.46. Scenario 1.1.1.0 : the new formulation. ....	186
Figure 7.47. Scenario 0.0.1.1 : the original formulation. ....	187
Figure 7.48. Scenario 0.0.1.1 : the new formulation. ....	188
Figure 7.49. Scenario 1.0.1.0 : the original formulation. ....	189
Figure 7.50. Scenario 1.0.1.0 : the new formulation. ....	189
Figure 7.51. Matching the final edge length of the new formulation with the original.	191
Figure 7.52. A cloth hanging from the two end points of a diagonal in the original simulator. ....	201
Figure 7.53. A cloth hanging from the two end points of a diagonal with the new mass constraint mechanism. ....	202
Figure 7.54. Simulating the first part of the scenario on the original cloth model at times: A) 0 second, B) 2, C) 4, D) 8, E) 12, and F) 15 seconds. ....	204
Figure 7.55. The instance captured by Figure 7.54 viewed from a different angle. ....	205
Figure 7.56. Simulating the second part of the scenario on the original cloth model at times: A) 18, B) 22, C) 26, D) 28, E) 30, and F) 40 seconds. ....	206
Figure 7.57. The first realignment method when the length of segment AE is greater or equal to its natural length. ....	208

Figure 7.58. The second realignment method when the length of segment AE is shorter than its natural length..... 209

Figure 7.59. Simulating the second part of the scenario on the improved cloth model at times: A) 18, B) 22, C) 26, D) 28, E) 30, and F) 40 seconds..... 211

Figure 7.60. Two flat pieces of cloth. .... 216

Figure 7.61. Undulations..... 216

Figure 7.62. The memory effect defined as a set of relationships between cloth nodes. 219

## List of Tables

Table 7.1. The default parameter values for the first simulator.....	82
Table 7.2. The active components for the first simulator. ....	82
Table 7.3. The default parameter values for the second simulator. ....	104
Table 7.4. The active components for the second simulator.....	105
Table 7.5. The relative overhead tests for components of simulator 1.....	130
Table 7.6. The relative overhead tests for components of simulator 2.....	132
Table C.1. The Modified Preconditioned Conjugate Gradient Method.....	245

# **Chapter 1: Introduction**

## **1.1 Overview**

Cloth modeling has caught the eye of many researches in the computer graphics community since the mid-1980s. Great interest in modeling cloth, its structures, and its perceivable behaviors for use in computer-generated images and animations has since spawned many advances and algorithms over the past several years. Of course, the computer graphics community is not the only one to show interest in this matter, let alone the first to dedicate its research for this topic. The textile industry, with its textile mechanics and engineering community, was undoubtedly the first to research cloth modeling and started as early as the first half of the twentieth century.

Nowadays, while the fashion industry, which the textile industry supplies, aims at obtaining computer-aided 2D cloth prototyping tools and garment design applications, the computer graphics and animation industries concern themselves with physical simulation of cloth and dressing virtual actors, nonetheless, all seek the common goal of successful cloth simulation with appropriate cloth modeling.

## **1.2 Motivation**

This work details the various challenges of devising an appropriate cloth simulator and presents the most commonly suggested and main solutions while

addressing specific issues left out by previous work in this area, issues that were possibly considered too case-specific.

We believe however that publishing detailed solutions to these issues will help eliminate repeating problems most cloth simulations eventually face and will help extend the previously proposed models. In short, this work digs out aspects of the problem that were not explored or specified while progressing through the whole challenge.

### ***1.3 Objectives and Methodology***

The purpose of this thesis is to both detail the whole problem and present a set of associated solutions and models, and even suggest hybrid solutions to achieve different cloth simulation goals. To attain this objective, many of the solutions presented by previous work will first be extracted in the following chapters; then, the solutions will be tried out for the purpose of analysis and for finding suitable answers to the various concerns encountered in the progress of building a cloth simulation. So, at each of the steps presented in Chapter 2, several alternatives are considered, analyzed, and several small corresponding cloth simulation programs are built in order to validate or confirm the hypothesis of such solutions referred. More specifically, the practical work is to devise a basic cloth simulator whose components can be upgraded and modified to incorporate the different cloth models and simulation techniques that will be discussed in this thesis. Then, the most meaningful observations are noted, the respective models and solutions employed are detailed in the following chapters, and further analysis of the results and suggestions are given in subsequent sections of this work.

## **1.4 Thesis Organization**

The thesis is presented in eight chapters. After this first introductory chapter, Chapter 2 starts with some brief background information of the subject by describing cloth, and presents the various aspects as well as terminologies of cloth simulation. The whole problem is then portrayed in subtasks.

Chapter 3 gives an overview of several related work as well as their relevance to this subject.

Chapter 4 discusses in more detail two of the cloth models that are at the base of many developed models and cloth simulators. For both models, the internal mechanisms will be detailed; their respective numerical integration method will be explained, and their constraint enforcement techniques and approach to collision detection and response will also be presented.

Chapter 5 onwards presents our practical work alongside our analyses of the results. More specifically, Chapter 5 describes the design of two simulators we built in order to try out the theoretical models.

Chapter 6 gives the implementation details of the two simulators, and explains the simulator components on a practical level.

Two set of tests will be performed with the two simulators and the results collected are then presented in Chapter 7. The two simulators are compared on different levels. Both quantitative and qualitative analyses of the cloth simulators and their models are performed. Then, alternative solutions for the numerical integration method and

collision detection and response are presented, tested, and analyzed. Chapter 7 then discusses some more specific problems we believe were left cloudy by previous work, and ends with a discussion over wrinkles and the considerations for garment simulation.

Finally, Chapter 8 presents the conclusions and discusses about future work.

## Chapter 2: Background

### 2.1. *What is Cloth?*

We are in contact with cloth constantly, in the form of clothing or in the other forms that we use in our homes. While we easily see its numerous characteristics, it is not as easy to explain them and it is even less obvious to recreate the attributes and behaviors of cloth with undisputed certainty for a simulation.

The characteristics of a fabric depend greatly of course from what it was made but also on how it was made. Thus, in order to construct a fine cloth model, knowing how cloth is formed, the underlying structure is helpful.

The two most common types of fabrics are woven fabrics and knit fabrics. This work only deals with woven fabrics, the more popular of the two in terms of the amount of work dedicated to it.

Cloth is woven from *yarns* or *threads*. The two terms are used interchangeably, though yarn is usually understood to be heavier and thicker. Yarn is a group of filaments, fibers, twisted together to form a continuous strand.

There are natural fibers and synthetic ones. Natural fibers come from an animal or a vegetable source while synthetic fibers are man-made and usually from polymers. When spinning, two or more fibers may be combined, for example, cotton mixed with polyester, or the yarn can be made up of only one fiber.



To describe very briefly what a textile designer has to choose at this point, depending on the number and the type of fibers spun, the yarn can be soft or hard, stiff or flexible, shiny or dull, textured or smooth, thick or thin, or, a combination of these features.

Next, the yarns are taken into the loom for a weaving process that has numerous steps and details [HOU00]. The result is a structure of interlacements of *warps* and *wefts* with a certain designed weave pattern. [Figure 2.1] The three basic weave structures are plain, twill, and satin weaves. [Figure 2.2] Then the piece of fabric is taken into a few additional and finishing steps.

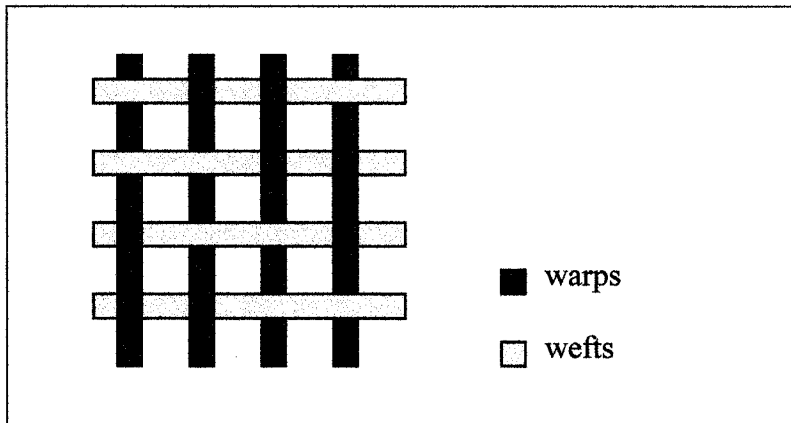


Figure 2.1. Interlacements of warps and wefts

The fabric's inherent qualities are based on those many variables of its construction such as the choice of fiber, the yarn weight, the color, the weave pattern, and so on. In the end, the fabric will yield a certain *look* and *hand*. The *look* is its visual appearance. The *hand* refers to how the fabric feels when we pick it up and lightly crush it in our hand. [p.2 of HOU00]

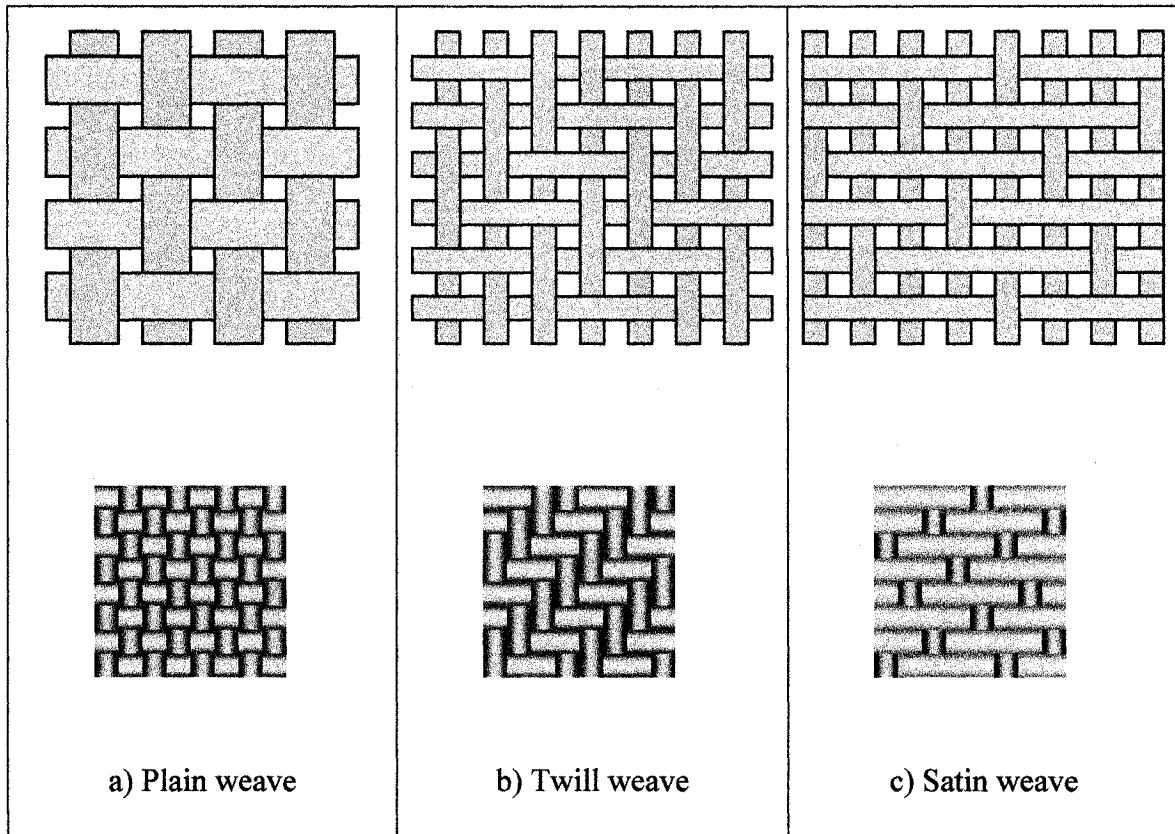


Figure 2.2. The three basic weave structures

## 2.2. Overview of the terminologies and topics involved

We assume that the reader already has some sound background knowledge of most past computer algorithms and physical or mathematical concepts presented in this work; but, for reassurance, we would like to briefly remind some concepts and define a few terminologies used in this work.

The terms energy-based, force-based, particle-based, are used regularly in this work to qualify cloth models. The meaning of a system being *energy-based* is that energy functions are defined to express how much energy, or in other words potential, can force the system or the object involved to change state, to change position or

configuration in order to reduce this potential, this energy. Obviously, to reach the current state in which the energy is not zero, an intervention external to the object must have taken place. The goal with energy-based systems is most commonly the minimization of this energy once all the factors involved are known. To relate to the topic of cloth simulation in particular, an energy-based cloth model would account for factors such as gravity, wind, and surrounding objects that could interact with the cloth, then formulate energy functions to express the behavioral properties of the cloth, and try to determine how the piece of cloth would find a resting state in that environment so that its energy would be minimal. Some cloth draping techniques mentioned in the later chapters elaborate on this principle and are easily capable of showing pieces of cloth draped over furniture in a living room. Now, in order to simulate a physical system over time with dynamics, forces must be obtained and an integration method must be chosen to advance the simulation in time. Forces can be divided between internal and external forces. Internal dynamics are the mechanisms explaining the behaviors and characteristics of the system representing the simulated object, cloth in the current case; and, the forces generated by these mechanisms are categorized as internal forces. The remaining forces influencing the state of the system are labeled as external forces. A model involving forces is referred to as being *force-based*. The formulation of internal forces can directly be obtained if variants of known mechanisms in physics such as springs are used or, alternatively, the energy functions derived from an energy-based model can be differentiated with respect to their respective variable such as positions or angles in order to get the internal forces. In many physical simulations where the object simulated is both deformable and possesses a pertinent surface or volume, it is often

preferable to use a limited number of sample points to represent the object and its properties. When these sample points are given a mass and are considered as discreet smaller objects to be simulated, then they are behaving like particles and the model is then said to be *particle-based*. It then becomes possible to observe and analyze the behaviors and the state of a particular part of the system without losing track of the overall whole. When a system such as cloth is simulated in an environment where there are other physical objects that could interact, collisions can then occur between parts of the cloth and parts of these rigid objects, this type of collisions is referred to as *cloth-object* collisions. In the specific case where the object interacting with the cloth is a human body, then the collision is said to be a cloth-body collision. With simulated deformable objects such cloth, collisions between parts of the same object becomes an issue to resolve and to treat appropriately for realism and correctness. We call this second kind of collision either cloth-cloth collisions or simply cloth *self-collision*. Regardless of its type, every collision is a phenomenon that is resolved quickly and yet includes many physical events occurring almost instantly. A complete collision response method must be able to deal with the contact between the two colliding objects, the possible sliding motion between the surfaces of these objects, the frictional force limiting this sliding motion that represents the roughness of the surfaces, the bouncing effect after impact if the collision is not considered to be a perfectly inelastic collision, else, the energy loss is considered total as if it were entirely absorbed and dissipated, and a perfectly inelastic collision offers no bouncing forces between the two objects in collision.

We described briefly in the previous section what cloth is made of on a scale less obvious to the human eye. The weaving patterns, the number of fibers twisted per thread, the fiber dimensions and weight, their colors, the fiber properties, and more characteristics of each distinct type of fabric have a direct effect on how light will reflect, refract, and yield color. Thus, the fabric properties not only determine how the cloth will behave mechanically and what shape can be obtained but also how it will look in terms of shininess, in terms of colors and texture. Many researches study these latter properties of fabrics that give cloth its visual qualities, and we refer to this set of cloth properties as optical properties.

### ***2.3. The Problem Description: Subtasks***

The cloth simulation challenge as a whole can be subdivided into smaller steps and these subtasks can each be approached with different methods and involve choosing between various solutions. The problem can be broken into the following steps:

- Adopting a geometry-based or physical-based approach,
- choosing an appropriate representation model,
- defining the internal dynamics of the system,
- selecting the numerical integration method,
- deciding how and whether to use damping and post-step constraints and corrective measures,
- determining suitable collision detection and handling methods,
- defining the optical properties chosen for rendering ,

- and in the more specific case of garment simulation, garment design and further analyses need to be addressed.

## **Geometry-based versus Physically-based**

First, the problems encountered depend entirely on what is sought to be achieved, this is where cloth simulation techniques are initially categorized. *Geometry-based* techniques are entirely dedicated in reproducing the look of cloth and to make it look realistic or accurate. Whereas *physically based* techniques also emphasize representing the realistic and correct behavior and characteristics of the simulated cloth. Depending on the emphasis chosen and the representation goals, either philosophy will prevail but numerous work now suggest hybrid techniques obviously.

## **Representation Models**

The next step consists of deciding on how to represent cloth and how it should be understood in order to study its behaviors. Cloth, on the macro scale level, is a relatively thin and continuous surface, a deformable object. Two representation categories are the particle-based models and the finite element models. The *particle-based* models represent cloth with sample points or nodes, very likely illustrating weave interlacements, crossings, and these particles will have a mass associated according to the surface area of the cloth they represent. The particles, along with their joining edges form the mesh

structure of the cloth, and different mesh structures can be used. Certain models use a regular triangle-mesh while others employ irregular triangle meshes, etc. Either linear or non-linear springs connecting the nodes are usually employed to simulate the elastic properties of cloth. And, equilibrium of both external and internal forces is enforced at each nodal point, therefore locally. *Finite-Element Continuum* models on the other hand treat cloth like a continuous surface and make no ad hoc assumptions regarding calculations of stress and strain quantities. These approaches usually follow from well-established disciplines of continuum mechanics. Finite-element models still take discrete points on the cloth surface, but then a system of nonlinear equations governing displacements and surface orientation at the points is solved using an iterative technique such as the Newton Raphson method, and the equilibrium of external and internal forces is enforced globally, as opposed to particle-models that enforce local equilibriums.

This work mainly concerns itself with particle models though some suggestions may refer to solutions employed by finite-element models.

## **Internal Dynamics**

*Internal Dynamics* is the core of the simulation where the models differ in their methods and approach in recreating the mechanical behaviors and characteristics of cloth. Of course, this only concerns the physical-based family of methods and not the geometric-based one, whose only concern is the shape and look of the simulated cloth.

Whether the model chosen is to be force-based or energy-based, the next step is to determine how to obtain these energies or forces from the cloth. Mass-spring models and

condition-function models are very likely the two most popular formulations of internal dynamics, internal forces. The two models mainly referred to in this thesis are Provot's mass-spring model and Baraff Witkin's condition-function model. They are certainly not the only models, but since one makes use of springs and the other defined the concept of condition functions over triangles, they represent a decent base for different methods and hybrid ones. To recapitulate, this step consisted of defining how the cloth reacts to imbalance from its equilibrium state, an imbalance generated by external factors and constraints. The definition of these internal forces, reactions coming from the fabric's mechanical properties, must without any doubt correspond with the representation model chosen in the previous step.

## **Numerical Integration Methods**

The next problem resides in the integration method used. Given the forces generated by the internal dynamics, external forces, and physical constraints, it would seem that the only remaining task is to update the new state of the cloth, that is, its position and velocity. Although, at first glance, it might seem that the task is trivial and that the simple use of the forward Euler integration method would be able to answer most of the cases, there are issues nonetheless that required researchers to consider alternative solutions and suggest different integration schemes. The issue of accuracy and the time step problems are discussed more thoroughly in chapter 4. The implicit integration scheme will also be explained under chapter 4 where the Baraff Witkin model is presented. Different integration methods exist such as the following explicit methods:



Euler Integration (also equals Euler forward integration), Midpoint Integration (Midpoint rule), Trapezoidal Rule, Simpson's Rule, Runge-Kutta, more precisely its fourth order variant; and the implicit method of Backward Euler.

## **Damping and Post-Step Corrections**

In the real world, the energy in a mechanical system like cloth would inevitably in part be dissipated, be absorbed or transformed. However, details such as friction, viscosity, air drag are not inherently dealt with by the model unless they are explicitly taken care of and added to the simulation. Taking for instance a mass-spring model, the springs will constantly oscillate around their equilibrium points since the energy remains ever existent unless the system is *damped*. This was the first reason to devise a formulation for damping. Some well-defined damping methods such as viscous damping can be used or other damping methods can also be formulated. The second reason that justifies the popular use of damping in cloth simulations is its partial help to the time step and accuracy problem first referred in the numerical integration method step. Damping, in any of its forms, helps correct some of what the internal dynamics and integration method together failed to simulate realistically. Additional and alternative solutions could be the use of *post-step corrective measures* to adjust the unsatisfactory state of the model prior to displaying it. For example, corrective methods such as the one suggested by Provot are used to deal with over-elongated or compressed springs. Under any model, corrective measures are tools used in an ad hoc way. It remains however true that

avoiding or rather preventing a problem is better than remedying it and some research does indeed apply this philosophy.

## **Collision Detection and Handling**

Collision detection and handling is a two-part step; First, detection, then, response. Additionally, there are two types of collisions a cloth simulator might need to address. Collisions between the cloth and the simulated surrounding environment such as objects and bodies the cloth is draped over must be dealt with, yet, to achieve a realistically accurate modeling of fabrics, self-collision of cloth with itself, must also be handled.

As one could have guessed, the collision detection problem in cloth simulation is not only a necessity for more complex and realistic behaviors but also a heavy overhead that must be dealt with. In fact, many papers state collision detection and handling to be the bottleneck of the cloth simulation challenge. All known optimization techniques can be considered; but, not all can be of equal significance or of any use in this specific case. However, various works suggested optimization techniques such as bounding volume hierarchies, space subdivision such as voxelizations, coarse mesh dynamics, multi-layers approach, and many more to be effective.

## Optical Properties

In addition to the issues related to the physical and mechanical simulation of cloth, a significant number of researchers have showed interest in studying the optical properties of cloth and the different fabrics. The majority of research on cloth simulation focuses on the visual representation of cloth physical behavior, and in order to simplify the problem, a “generic” fabric is often utilized to represent the behavior of fabrics in general.

The optical property problem here is not exactly what the geometric-based techniques are worried with; those had goal to make the cloth look good in terms of shape while the optical properties concern more the problem of procedural texturing and therefore, the distribution of light on the fabrics etc. It therefore brings the study down from our usual object-scale level to the yarn-scale and micro-scale levels.

At the yarn-scale level, the cloth structure is studied, revealing optical properties from structural characteristics such as the interlacement and weave structure, how its warps and wefts are placed, yarn thickness, twisting tightness, etc. Finally, at the micro-scale level, the light’s reflectance behavior on the fabrics, the roughness or smoothness of yarns, the cross-section’s shape etc is studied. Fibers have all the translucent, reflective and refractive characteristics. And so, the use of BRDFs(Bi-directional Reflectance Distribution Functions), functions describing the illumination properties of a surface in terms of the incident and reflected angles, is even called upon to obtain the exact behavior of light on those fabrics.

## Chapter 3: Related Work - Overview

Many researches and work have been made over the topic of cloth simulation and modeling since the mid 1980s. While it is true that most work had to cover superficially the many aspects of the whole problem of cloth simulation in order to present a comprehensive result, it is often the case that each particular work addressed a more specific side of the challenge with more emphasis; Therefore, providing us with many alternative approaches to each of the subtasks of the whole challenge. This chapter will introduce briefly each of the previous work referenced.

Carignan et al. [CAR92] designed cloth pieces in their work with polygonal panels in 2D then seamed and attached them to virtual actors' bodies in 3D using two kinds of dynamic constraints and forces; and, a physical-based model described by Terzopoulos et al. [TER87] is used for its internal dynamics along with the collision processing to animate clothes on synthetic actors in motion. Both cloth self-collision and collisions between cloth and the human body were handled with perfectly inelastic response in Carignan et al's work. Self-collision detection was performed by comparing pairs of particles. Their work consisted of taking the approach of a tailor and providing a system that, given cloth properties specified by the user, would automatically compute the less user friendly and less obvious parameters such as the mass of a node, the stretching resistance, the time step size, the bending resistance, or the stretching energy dissipation rate.

In 1992, House and Breen [HOU92] developed a particle model for cloth drape representing the micro-mechanical structure of cloth based on the observation that cloth is best described as a mechanism rather than a continuous model as assumed by continuum models. In this particle representation, each particle represents a yarn crossing point and the mechanical interactions occurring between those points such as compression and stretching, out-of-plane bending, and in-plane bending, shear or trellising, are modeled and each is characterized by a strain energy function that is computed based on simple geometric relationships among local particle neighbors. There are three phases to their simulation run over small time steps. The dynamics of each particle falling under gravity and interacting with its surrounding environment is first computed, then, energy-minimization is used to enforce inter particle constraints; finally, a correction to velocities is performed to account the movements of the second phase. Their model can be extended to include dynamics by using a force-based approach formulation introducing non-linear springs on the yarn segments between particles.

Provot presented a mass-spring model in [PRO95] defining explicitly three types of springs to deal with stretch, shear, and flexion or bending properties of cloth. In his following work [PRO97], he presented a cloth self-collision handling that dealt with both point-triangle and edge-edge types of collisions. He suggested the use of bounding box hierarchies and a hierarchy of cones, to represent the curvature levels of cloth, as optimization techniques for self-collision detection. Provot also described contact,

friction and dealt with the consistency issue involved in multiple collisions via an algorithm using “zones of impact.”

Cordero [COO01] presented a spring model very similar to Provot’s but with a slightly different formulation allowing more differentiation between in-plane and out-of-plane forces. Also, with this formulation, it is easier to notice and redefine the non-linear elastic behaviors of cloth to reflect a particular fabric.

Baraff and Witkin [BAR98] presented a physical-based model using condition-functions instead of springs to derive internal forces. Their work addresses mainly the problem of numerical instability by suggesting an implicit integration method. With the three condition functions expressing cloth’s stretching, shearing, and bending resistance properties were associated three respective damping forces. Baraff and Witkin also suggested the use of mass modification over other mechanisms to enforce constraints. They also contributed a modified conjugate gradient method adapted to their model to solve the system involved. Finally, an adaptive time stepping algorithm is detailed to guarantee a visually pleasant and numerically stable solution.

Villard and Borouchaki [VIL02] developed an adaptive meshing technique in order to reduce the computational time that a fine discretization would require otherwise while being able to refine the mesh for more accurate representation when and where it is needed. Their mechanical-based cloth model is inspired from Provot’s but appropriate terms were added to the stretch and shear force formulations to answer the adaptive mesh

refinement method. The bend condition is formulated differently, using an analogy from beams mechanical behaviors instead of simply using springs and an explicit integration scheme was adopted. Their refinement criterion is the curvature level at a node versus a predetermined threshold deviation value allowed against the tangent plane. If the curvature exceeded the threshold value, subdivisions of the concerned elements around the target node will produce a mesh refinement yielding additional nodes, some of which will be categorized as “active” and will take role in the mechanical simulation while others will be labeled “virtual” and will solely serve the mesh topology. To sum up, their model will start with a coarse quad mesh structure and will refine this wherever the curvature level requires a finer representation.

Oshita and Makinouchi [OSH01] combined dynamic simulation using a small number of particles and geometric techniques to smooth the coarse mesh into a fine mesh in order to simulate twists and wrinkles. Their simulation algorithm assumes the model to be represented by a coarse triangle mesh and can be divided into five steps. First, positions of the particles of the coarse mesh are computed. Next, surface generation i.e. smoothing is performed using point-normal (PN) triangles method replacing the triangular faces with three-sided cubic Bezier patches. The PN triangles provide means of improving visual quality by smoothing out silhouette edges and providing more sample points for vertex shading operations by replacing original flat triangles with curved shapes that are each re-triangulated into smaller flat triangles. Then, particle normal control deforms the cloth to reflect the elastic forces applied on the particles and edge length control is performed to keep the curve lengths equal to the original edge

lengths. Finally, triangular tessellation is applied for a smoother cloth surface to display. The dynamics model used is of a spring model similar to Provot's, but only elastic stretch forces are considered and the integration method chosen is the 4<sup>th</sup> order Runge-Kutta. Collision detection between cloth and solid objects is done for both vertex-faces and edge-edge pairs with the coarse mesh.

Bridson et al. [BRI03] stated that the key for high visual details from dynamic folds and wrinkles depends on a good model for bending, and a robust self-collision strategy for wrinkling and folding. And so, they presented an accurate model for bending based on the dihedral angle between the two normal vectors of the pair of adjacent triangles and its rate of change as in Baraff and Witkin [BAR98] with a formulation that allows non-zero rest angles. Their other major contribution is a mixed explicit/implicit integration scheme where the simplicity of an explicit method such as the 4<sup>th</sup> order Runge-Kutta is used to handle elastic forces, forces that are independent to velocity, while forces that are velocity dependent such as damping forces are handled via the implicit method of Backward Euler for its accuracy. An algorithm for efficient and robust process regarding collisions, contact and friction was described in [BRI02] combining a fail-safe geometric collision method with a fast non-stiff repulsion force method allowing the modeling of cloth thickness and both static and kinetic friction.

Volino and Magnenat-Thalmann contributed several work including survey papers and solution bundles in the topic of interactive cloth simulation techniques [VOL97][VOL98], a guideline to an efficient implementation of implicit numerical



methods and constraint-based collision response in [VOL00], a quantitative analysis of the efficiency of different integration methods for cloth simulation [VOL01], and numerous other work. Nadia Magnenat-Thalmann and her teams at MIRAlab went even further into studying the optical properties of fabrics, suggested a suitable Bi-directional Reflectance Distribution Function (BRDF), devised a technique for visualizing woven clothes in real-time and rendering a variety of weave patterns [THA03], and presented a technique for creating procedural textures of the twists of thread. Also, in the co-authored paper with Cordier [COR02], they proposed a hybrid model of a physical based approach and geometric deformations that answered real-time constraints while focusing on visual features of cloth that an observer would be sensitive to. Their model avoided as many heavy calculations as possible to satisfy the real-time requirement and used a multi-layer approach in which clothes are divided into three layers, each simulated via its own dedicated method.

Fuhrmann et al. [FUR03] replaced internal forces with constraints in their cloth model along with an Explicit Verlet integration scheme allowing the usage of large time steps without instability. An efficient algorithm for avoidance deals with cloth self-collision instead of a collision correction method. The algorithm employs repulsion strings based on a triangle-size dependent repulsion distance and a bounding hierarchy of particles.

## Chapter 4: Related Work – detailed discussion of algorithms

In this chapter we present the two main models we based our implementations on. Each model formulates the physical and mechanical properties of cloth differently, and employs different algorithms to complete the task.

### 4.1. *Provot's model*

The first model discussed is based on the work of Xavier Provot[PRO95]. The model is physical-based, particle-based, and even more precisely, a *mass-spring* type of model. The cloth object is first approximated to a deformable surface composed of a network of virtual masses and springs. More specifically, three types of springs are defined to link the masses: structural, shear, and flexion springs.

#### 4.1.1. The three types of springs

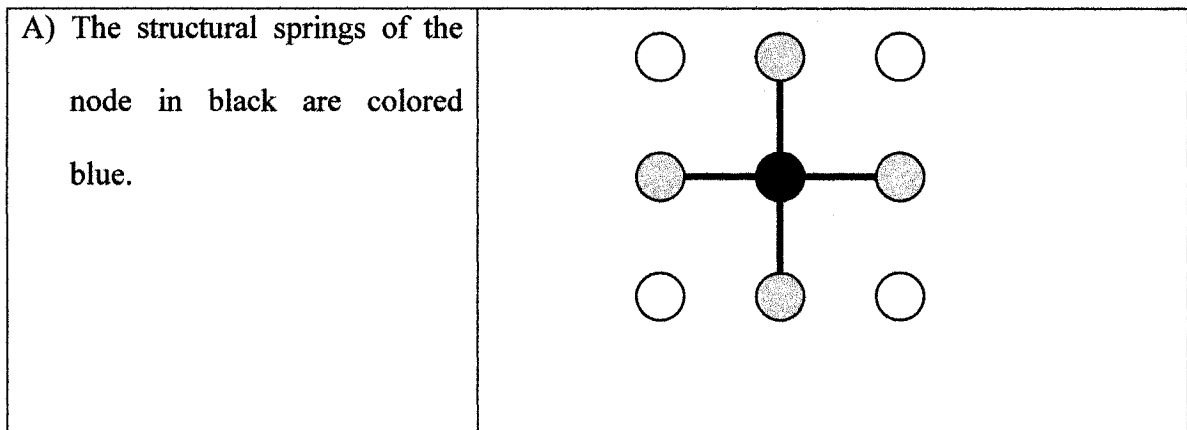
*Structural springs* are the springs connected from each node to its four directly adjacent neighbors acting against stretch or compression in those directions. These springs are also referred to as the stretch springs of this model. (Fig.4.1.A) *Shear springs* are the ones connecting to the four diagonal neighbors to handle diagonal stretch. (Fig.4.1.B) *Flexion springs* are defined as going from each node to its four neighbors

that are two apart from itself, i.e., skipping its directly adjacent neighbors (Fig.4.1.C)

These springs are designed to illustrate the bending resistance the fabric offers.

*The model is not considered as a continuous surface that will have to be discretized, but rather as a discrete structure of elements where each mass-point and each spring can be handled individually. [PRO 95]*

Each of the three types of springs has its associated stiffness coefficient and the force generated from the spring is computed with respect to the difference in length from its equilibrium often said “original” length. Let the model be described by an  $m \times n$  mesh such as described in Fig.4.2, and let  $P_{i,j}(t)$  represent the position of mass or point  $P_{i,j}$  at time  $t$ .



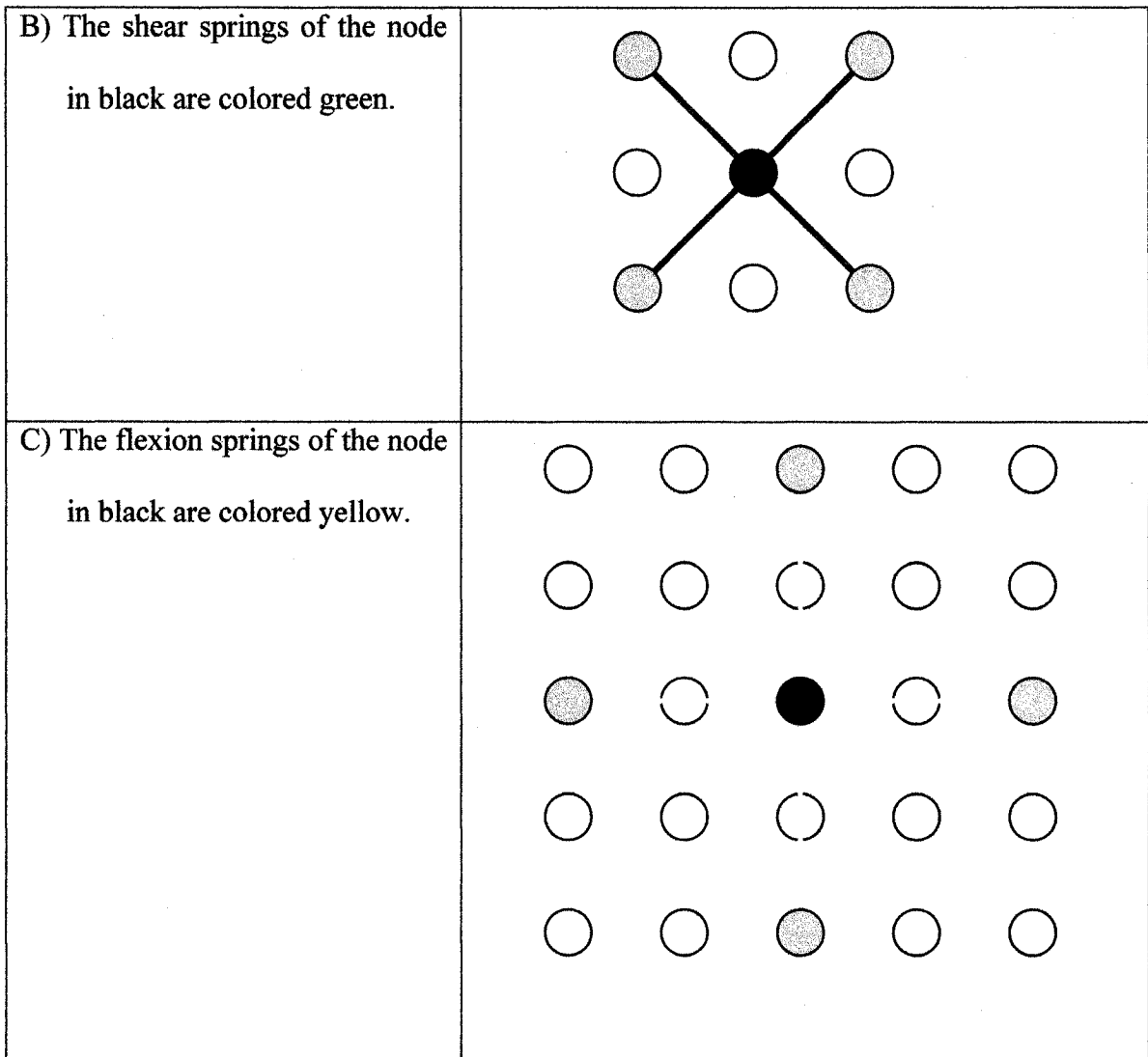


Figure 4.1. The three types of springs connected to a node: A) Structural springs, B) Shear springs, C) Flexion springs.

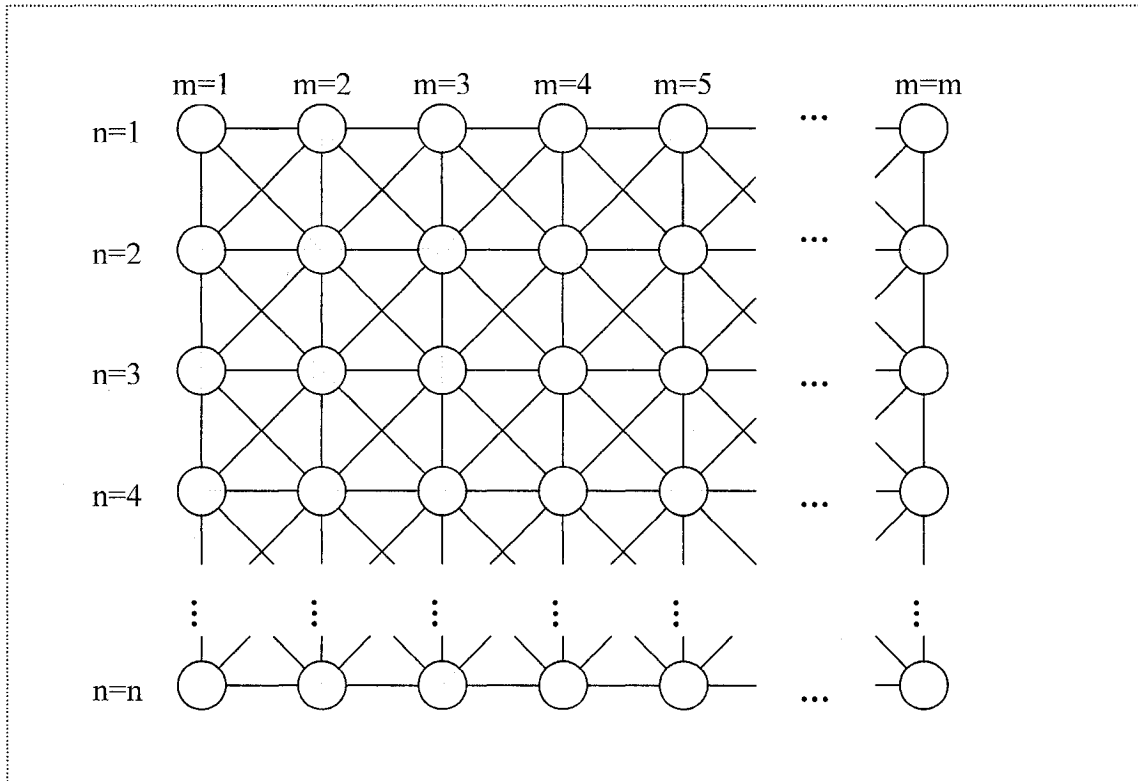


Figure 4.2. A mass-spring network of size  $m$  columns by  $n$  rows.

#### 4.1.2. The Internal Dynamics, the External Forces, and the System Evolution

The evolution of the system is governed by the fundamental law of dynamics:

$$F_{ij} = m a_{ij} \quad (\text{Equation 4.1})$$

where  $m$  is the mass of point  $P_{ij}$  and  $a_{ij}$  is the acceleration by the force  $F_{ij}$ . The force  $F_{ij}$  describes the summation of both external and internal forces acting on point  $P_{ij}$ . The *internal forces* result from the tensions of all the springs linked to  $P_{ij}$ , and can be expressed as follows:

$$F_{\text{internal}}(P_{ij}) = - \sum_{(k,l) \in R} K_{ij,k,l} [L_{ij,k,l} - L_{ij,k,l}^0 \frac{L_{i,j,k,l}}{\|L_{i,j,k,l}\|}] \quad (\text{Equation 4.2})$$

where the forces generated by all springs of any of the three types connecting particle  $P_{i,j}$  to its neighbors identified by pairs  $(k,l)$  and grouped under the set  $R$  are summed up.  $L_{i,j,k,l}$  represents the vector  $P_{k,l} - P_{i,j}$  while  $L^0_{i,j,k,l}$  is the natural length of this spring linking the two masses and  $K_{i,j,k,l}$  is the stiffness coefficient associated with the appropriate type of spring.

Provot's model will next use *viscous damping* to approximate the dissipation of the mechanical energy of the system:

$$F_{\text{damp}}(P_{i,j}) = -K_{\text{damp}} v_{i,j} \quad (\text{Equation 4.3})$$

where  $K_{\text{damp}}$  is the damping coefficient and  $v_{i,j}$  is the velocity of point  $P_{i,j}$  relative to the world space origin in which the cloth is animated.

The *external forces* are described similarly to most work in general; for instance, gravity is simply given by:

$$F_{\text{gravity}}(P_{i,j}) = mg \quad (\text{Equation 4.4})$$

where  $g$  is the gravitational acceleration.

In addition, air stream, wind, or the description of viscous fluid in which the cloth could be moving is expressed by:

$$F_{\text{fluid}}(P_{i,j}) = C_{\text{fluid}} [n_{i,j} \cdot (u_{\text{fluid}} - v_{i,j})] n_{i,j} \quad (\text{Equation 4.5})$$

where  $n_{i,j}$  is the unit normal vector on the surface at point  $P_{i,j}$ ,  $u_{\text{fluid}}$  the uniform velocity that the fluid exerts, and  $C_{\text{fluid}}$ , a chosen coefficient for the fluid.

The integration scheme and method in which this work was first introduced is the simple Euler method where  $h$  would be the time step chosen:

$$a_{i,j}(t+h) = \frac{1}{m} F_{i,j}(t)$$

$$v_{i,j}(t+h) = v_{i,j}(t) + h a_{i,j}(t+h)$$

$$P_{i,j}(t+h) = P_{i,j}(t) + h v_{i,j}(t+h)$$

(Method 4.6)

Provot also makes reference to *dynamics inverse procedures* described by Carignan *et al.* [CAR92] to deal with constrained points such as points of a hanging curtain being fixed to a rod where the movements of the cloth object are not entirely caused by analytically computable forces. While the integration of the fundamental law of dynamics allows the displacement of a point to be computed from the force applied to it, the inverse problem is sought here: the displacement is wished and known in advance to be null, and so will its velocity be, and so, the necessary force can be computed.

### 4.1.3. The “super-elasticity” problem

Unrealistic elongation is often detected around hanging points and is labeled as “Super-elasticity”. To remedy to this problem, the model would keep track of the deformation rate of springs and upon detection of a rate exceeding a threshold value or percentage, such as a ten percent, dynamic inverse procedures are used to recompute the state so that the deformation rate does not exceed this value. This adjustment procedure

is also an ad hoc principle since deformation rates are computed only after the numerical integration of each time-step is performed.

#### **4.1.4. Collision detection and response**

Provot presented a complete method for handling cloth self-collision in [PRO 97]. His method deals with four aspects: collision detection, optimization of detection, collision response, and conservation of collision consistency. In fact, regarding collision detection and response, both the general case of a collision involving a cloth object and another moving object in the scene and the particular case of self-collision are basically handled the same way; the difference lies in the optimization of self-collision detection. All objects are assumed to be represented by triangles.

Let  $t_0$  be a time where the system is in collision-free state. Now, we want to consider the interval  $[t_0, t_0+dt]$ . The next assumption is that the numerical integration method used approximates that each node moves at constant velocity during the time interval  $[t_0, t_0+dt]$  such as the explicit Euler method does. Then, there are two possible types of collisions possible to detect: a “point-triangle” where a node and a triangle would come in contact, and an “edge-edge” collision involving a triangle edge intersecting the edge of a different triangle.



### Detecting the “point-triangle” collisions:

Let pair of point and triangle verified be denoted by  $P(t)$  for the point, and  $A(t), B(t), C(t)$  for the three vertices of the triangle. The first idea is if a collision occurred during the interval  $[t_0, t_0+dt]$ , there will be a time  $t$  in that interval when the point  $P(t)$  will lie in that triangle and this condition can be written as:

First, let  $V$  be the velocity of the point  $P(t)$ ,  $V_A$ ,  $V_B$ , and  $V_C$  be the respective velocities of the triangle's vertices  $A(t), B(t)$ , and  $C(t)$  during that interval. It follows that  $A(t)=A(t_0)+tV_A$ ,  $B(t)=B(t_0)+tV_B$ ,  $C(t)=C(t_0)+tV_C$ . Then,

There exists  $t \in [t_0, t_0+dt]$  such that there are  $u, v \in [0, 1]$ ,

$$u+v \leq 1 \text{ and}$$

$$AP(t) = uAB(t) + vAC(t)$$

*(Equation 4.7)*

where  $AP(t)$  is the vector  $P(t)-A(t)$ ,  $AB(t) = B(t)-A(t)$ ,  $AC(t) = C(t)-A(t)$ .

The above system has two equations and three unknowns. The next idea is to use a second condition expressing that  $P$  belongs to the triangle  $ABC$ . Knowing that the plane of the triangle has normal vector expressible via the vector product  $N(t) = AB(t) \times AC(t)$ , the following expression would also be satisfied at the time of collision ‘ $t$ ’ for point  $P$ :

$$AP(t) \cdot N(t) = 0 \quad \text{(Equation 4.8)}$$

That second relation would only imply, if satisfied, that the point P is coplanar to the triangle ABC, thus being a *necessary* but *not a sufficient* relation i.e. this condition should be tested first. In trying to solve for t, the above dot product is a third degree equation since  $N(t)$  is a second degree t term, and  $AP(t)$ , a first degree. Consequently, three values of t can be obtained, but only those belonging to the interval  $[t_0, t_0+dt]$  would be of interest. The candidate values do not yet imply the occurrence of a collision, but only a co-planarity of the point P and the triangle's vertices. The next step is to substitute the retained values back into the first equation 4.7 making it a linear system of two unknowns and two equations. The resulting is possibly one or several triplets (t,u,v). The solution with the smallest value of t is chosen, since it would represent the collision that occurred the soonest.

### **Detecting the “edge-edge” collisions:**

Let us denote a first edge of the cloth with  $AB(t)$  and a second edge by  $CD(t)$ . Now, we want to find out whether these two edges will cross and have a common point during the interval  $[t_0, t_0+dt]$ . In other words, there will be collision if and only if:

There exists  $t \in [t_0, t_0+dt]$  such that there exist  $u, v \in [0, 1]$  and

$$uAB(t) = v CD(t)$$

(Equation 4.9)

Again, similarly to the case of “point-triangle” type of collisions, this first equation alone leads to a non linear system requiring us to seek an additional relation to simplify it. If

collision indeed occurred at some time  $t$  in the interval, then it would follow that the four points A,B,C, and D would then lie on the same plane. Thus, forming for example another edge  $AC(t)$  from the points A and C,  $AC(t)$  would also be coplanar with  $AB(t)$  and  $CD(t)$ , which can be written as:

$$( AB(t) \times CD(t) ) \cdot AC(t) = 0 \quad (\text{Equation 4.10})$$

Hence, this third degree equation also becomes a necessary relation whose values of  $t$  that fall within the interval  $[t_0, t_0+dt]$  would be substituted back in the equation 4.9 to solve for  $u$  and  $v$ .

## **Collision Detection Optimization**

### **Bounding Boxes Hierarchy:**

Without any optimization, the collision detection between a cloth object represented by  $N$  nodes and another object in the scene with  $M$  nodes is of complexity  $\mathcal{O}(MN)$  while the detection of the cloth's self-collision itself has  $\mathcal{O}(N^2)$  complexity. Therefore, cloth self-collision greatly limits the performance of the simulation. The performance of the simulation will be unacceptable for large values of  $N$ ; conversely, a small value of  $N$  would yield a low sampling resolution of the cloth surface producing a coarse representation of cloth. Consequently, optimization over collision detection is especially recommended for cloth self-collision. The first simple optimization is to start with *bounding boxes*. We begin by recursively partitioning the cloth triangles into zones

based on their position in 2D texture coordinates or what is referred to as UV space. These zones will be bounded by the boxes each iteration. The box or zone hierarchy was built up during the recursive partitioning. The whole cloth is the root of the tree. Since the zones are initially in 2D, each zone will be subdivided into four smaller zones, and this recursive subdivision is performed until the zones consist each of only a pair of adjacent triangles that form a quad. Now, the collision algorithm would parse the bounding box tree while rapidly eliminating collision tests that involve two zones whose bounding boxes do not even intersect. To accurately detect intersections of bounding boxes during the interval  $[t_0, t_0+dt]$ , the box will not only bound the position of the zone at time  $t_0$ , but both the zone's positions at time  $t_0$  and  $t_0+dt$ .

#### **Surface curvature and self-collision detection:**

The following self-collision detection optimization suggested by Provot is inspired by Volino et al.'s work in [VMT94]. The solution is based on the property that any given zone qualified as having a "low curvature" cannot self-intersect itself and therefore, all the zones it includes do not intersect each other. The term "curvature" utilized for a zone is evaluated by the set of normals of the triangles belonging to that zone with the following method. A tree hierarchy of cones will be constructed. A *cone* will include a set a triangle normals; and, the angle  $\alpha$  at cone's vertex (Figure 4.3) representing half of the cone's opening angle will be used to evaluate the curvature of the zone represented. If  $\alpha < \pi$ , the zone represented by that cone cannot self-intersect. The construction of the cone tree starts from the leaf nodes at the bottom of the tree where a node is simply a

cone containing a single normal vector. In the other words, the only “zone” included in each tree node at this moment is a triangle. So,  $\alpha=0$ . Then, the tree is eventually constructed by recursively forming up new branches by taking pairs of cones whose zones are adjacent and making them descendants of a new parent node. Let two children nodes have the curvature measures  $\alpha_1$  and  $\alpha_2$ , and let  $\beta$  be the angle between the two axes of the descendant cones. Then, the parent cone’s curvature value is computed as:

$$\alpha = \beta/2 + \max\{\alpha_1, \alpha_2\} \quad (\text{Equation 4.11})$$

The process is continued until the root cone includes all the cloth’s normal vectors of all the zones.

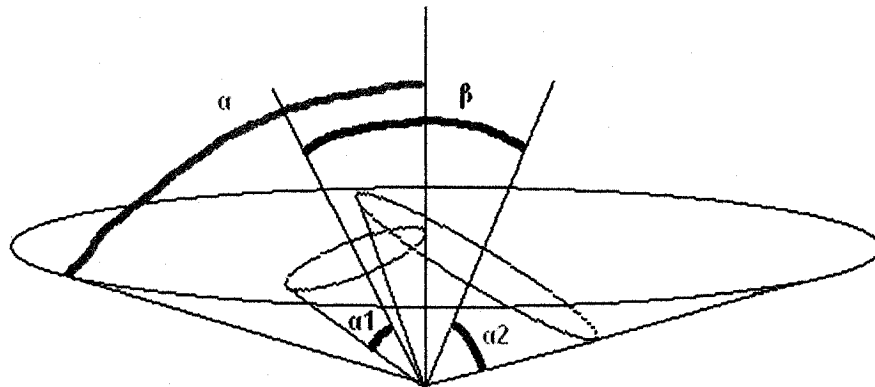


Figure 4.3. The computation of a parent cone from two children cones.

## Collision Response

### Contact and Friction

“When two objects collide, there is a time at which they are in contact.”[PRO97]

Let P be a point on the cloth entering in collision with a point H of a rigid object’s surface with normal vector N. When P=H, there is *contact* and let F be the force applied to P such that it allowed and kept the contact; Then, the perpendicular component of F to the surface at H is

$$F_N = (F \cdot N) N \quad (\text{Equation 4.12})$$

and the tangential component would be the remnant force  $F_T$  trying to move P from H on the rigid object surface

$$F_T = F - F_N \quad (\text{Equation 4.13})$$

The laws of Coulombian friction are:

If  $\|F_T\| \geq k_f \|F_N\|$ , there is sliding contact with friction, and P moves parallel to the surface under the action of the force

$$F_S = F_T - k_f \|F_N\| u_T \quad (\text{Equation 4.14})$$

where  $u_T = F_T / \|F_T\|$ .

$k_f$  is the friction coefficient and

- ◇ if  $k_f = 0$ , sliding occurs without friction i.e. only the tangential force determines the sliding force applied to P.
- ◇ if  $k_f = \infty$ , there is no sliding at all.

The coefficient  $k_f$  can be understood as a characteristic of the fabric's friction behavior.

The next part in a collision after there is contact is the *impact*. The problem is the force  $F_{\text{imp}}$  generated by the impact may be unknown. In the case of a “point-triangle” collision where the triangle is motionless, the impact of the point on the triangle, and vice-versa by Newton's second law, is indeed unknown. The point P's velocity  $v$  before the shock is known but not the velocity  $v'$  afterwards. While we could describe the force applied to the point by the triangle with

$$F_c = m (v' - v)/dt \quad (\text{Equation 4.15})$$

and the term  $(v' - v)/dt$  being the acceleration of the point during the interval,  $v'$  remains what we are seeking.

If we made the approximation that the forces implied are proportional to the velocities of impact to evaluate the force involved during impact, then we can take the previous equations 4.12 to 4.14 and replace  $F$  with  $v$  and  $F_s$  with  $v'$  in order to obtain  $v'$ , the velocity of the point P right after its collision with the triangle. Since velocities should be considered constant during the interval  $[t_0, t_0 + dt]$  and that  $v'$  is now known, it will be used instead of  $v$  into computing the new position  $P[t_0 + dt]$  resulting at the end of the interval. Doing so is equivalent to considering that the collision precisely takes place at time  $t_0$  whatever the true collision time  $t$  would be between the interval  $[t_0, t_0 + dt]$ .

Impact has been in part described above, but its collateral bouncing effect and the associated energy dissipation must also be considered. On one hand, if a collision is said

to be elastic, there will be no dissipation of energy. On the other hand, if it is inelastic, then, there will be some energy dissipation. But, if the collision is perfectly inelastic, the entire energy of collision will be dissipated. Concerning energy dissipation, the velocity of point P after the shock can be described as  $v' = v_T - k_d v_N$  where  $0 \leq k_d \leq 1$  is the dissipation coefficient,  $v_T$ , the tangential velocity and  $v_N$ , the perpendicular component of the velocity to the surface. The dissipation coefficient is another mechanical property of the fabric.

The total collision response can be formulated by considering all the elements described above. The velocity of a point P before a collision with a motionless object  $v = v_T + v_N$  becomes after collision:

if  $\|v_T\| \geq k_f \|v_N\|$ , then  $v' = v_T - k_f \|v_N\| \frac{v_T}{\|v_T\|} - k_d v_N$

and there will be sliding contact with friction, bouncing and dissipation.

if  $\|v_T\| < k_f \|v_N\|$ , then  $v' = -k_d v_N$

and there will only be the bouncing collateral effect with dissipation.

*(Equation 4.16)*

In the case of self-collision, the velocity referenced should be the one of the center of mass of both colliding elements. The normal vector used in the case of a “point-triangle” collision is simply the triangle’s, whereas the case of an “edge-edge” will use the vectorial product of the two edges.

To sum up this task of dealing with collision detection and handling suggested by Provot, the possible occurrence of a collision was first determined, then the sliding



contact with friction, the impact generated displacement or rather the position at time  $t_0+dt$  after the shock was determined as part of the collision response, and also the final velocity at the end of the interval after considering the bouncing collateral effect with energy dissipation was solved.

### **Consistency of multiple collisions:**

Modifications by the previous collision handling algorithm may have created other collisions. Such cases where solving collisions generated additional collisions within the interval  $[t_0, t_0 +dt]$  is labeled multiple-collisions. And, these newly generated collisions that happened within the same interval must also be solved or collision consistency will not be maintained. Remember that collision response altered the position of the cloth at time  $t_0+dt$ . One more collision detection pass will have to be carried out to determine whether or not the alteration produced a new collision state. If not, the algorithm may just proceed to the next step, else, collision handling must once again be called upon and the entire process of collision detection and response must once again iterate. But, nothing guarantees that this iterative method will converge; So, Provot provided a solution using “zones of impact”.

A zone of impact is a set of points that either were involved in the same collision, be it “point-triangle” or “edge-edge” type, or were part in two different collisions that involve one or more points in common. During each iteration, if two zones of impact happened to share one or more points, they are merged to form a single larger zone of impact. The iterative method will stop when all zones of impact stop growing and remain

stable, at which point, they are said to be circumscribed. This method now converges since the multiple collisions are generally local or the zones of impact can grow up by merging until they all become one huge zone that includes the whole cloth mesh.

The previous iterative method of determining zones of impact does converge, but it does not entirely solve the handling of multiple collisions. To make sure that no interpenetration occurs within a zone of impact, Provot suggests to treat all elements included within a zone of impact “as a whole” rather than individually, and to suppose a perfectly inelastic impact and non-sliding contact for the collision response. This hypothesis is justified by the observation that elements involved in multiple collisions have their movement made difficult by these collision interferences since they are all in contact with each other. In other words, they are considered to remain fixed with respect to each other so that no further collision occurs within the zone. The zones of impact are treated like rigid objects during the interval  $[t_0, t_0+dt]$  and each zone will be attributed a group velocity  $V_G$  and a group angular velocity  $\Omega_G$ .  $V_G$  is the mean velocity of the  $n$  points in the zone of impact while  $\Omega_G$  is computed by reference to the geometric center  $G$  of the zone. Then, the collision response for all points  $M$  in this zone of impact  $Z_c$  is given by its new velocity:

$$\text{For all } M \in Z_c, V_M = V_G + \Omega_G \times GM \quad (\text{Equation 4.17})$$

where  $GM$  is the vector from  $G$  to  $M$ .

The whole collision detection and handling algorithm can be divided into three phases. The first phase consists of detecting either type of collisions and computing the

collision response as usual without considering the zones of impact. The second phase starts by another collision detection pass, and then memorizes the zones of impact where newly generated collisions appeared. Next, the zones of impact are handled with the previously described new collision response methods. The third phase is simply to iterate the second phase until the zones of impact stop growing, that is, no more new collisions can be detected during the time interval.

## **4.2. Baraff & Witkin's model**

The second model we present is based on the work by David Baraff and Andrew Witkin[BAR98]. Their model is also physically-based, but it does not employ springs to compute internal dynamics. Instead, internal forces are derived from condition functions over triangles; hence, the model uses a triangular mesh. Although their model is not limited to the use of regular triangular meshes, most previous implementations of this model chose a regular triangular mesh. Their work addresses mainly the problem of numerical instability and of cloth simulations requiring small time steps by suggesting a use of an implicit integration method.

The model's internal forces are derived from a simple continuum formulation common to all approaches that "physically-based cloth simulation is formulated as a time-varying partial differential equation which, after discretization, is numerically solved as an ordinary differential equation" where the acceleration of a cloth system is expressed as follows:

$$\ddot{x} = M^{-1} \left( -\frac{\partial E}{\partial x} + F \right) \quad (\text{Equation 4.18})$$

$M^{-1}$  is a diagonal matrix representing the mass distribution of the cloth,  $x$ , its geometric state,  $E$  is the internal energy of the cloth as a function of  $x$ , and  $F$ , a function of both its position  $x$  and velocity  $x'$ , describes the external forces.

Before elaborating further, the condition functions generating the internal forces will be examined.

### 4.2.1. The Condition Functions

Similarly to most previous work in cloth simulation, this model will deal with the three mechanical properties of cloth: the stretch, shear and bending resistances of the fabric. For each of these properties, a *condition function* will be associated, and this function will reach the value zero when the cloth is in a rest state. Consequently, these condition functions represent the energy level of the cloth being away from its equilibrium state in some sense. Additionally to its changing world space position  $x_i$ , every cloth particle is given a fixed plane coordinate  $(u_i, v_i)$ . Let  $W(u, v)$  be a continuous function that maps from  $uv$  space to world space.

#### 4.2.1.1 Stretch Condition

Stretch can be measured at any point on the cloth surface via 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, and there is no stretch if  $\|W_u\|=1$ . Similarly for  $W_v$  in

the  $v$  direction. Applying this to a triangle, let  $i$ ,  $j$ , and  $k$  be the three vertices of a triangle, then define:

$$\Delta x_1 = x_j - x_i \quad ,$$

$$\Delta x_2 = x_k - x_i \quad ,$$

$$\text{and, } \Delta u_1 = u_j - u_i \quad ,$$

$$\Delta u_2 = u_k - u_i \quad ,$$

and similarly for  $\Delta v_1$  and  $\Delta v_2$ ,

$$\Delta v_1 = v_j - v_i \quad ,$$

$$\Delta v_2 = v_k - v_i \quad .$$

Then, let

$$\Delta x_1 = W_u \Delta u_1 + W_v \Delta v_1,$$

$$\Delta x_2 = W_u \Delta u_2 + W_v \Delta v_2$$

And, reformulating to solve for  $W_u$  and  $W_v$  yields

$$\begin{pmatrix} W_u & W_v \end{pmatrix} = \begin{pmatrix} \Delta x_1 & \Delta x_2 \end{pmatrix} \begin{pmatrix} \Delta u_1 & \Delta u_2 \\ \Delta v_1 & \Delta v_2 \end{pmatrix}^{-1} \quad . \quad (\text{Equation 4.19})$$

Note that  $x_1$  and  $x_2$  vary but since  $u$  and  $v$  coordinates are constant; the matrix in the above equation remains the same in time. Thus, we can think of  $W_u$  and  $W_v$  as functions of  $x$  or more precisely of  $x_i$ ,  $x_j$ , and  $x_k$ . The condition function for stretch energy is defined as:

$$C(x) = a \begin{pmatrix} \|W_u(x)\| - b_u \\ \|W_v(x)\| - b_v \end{pmatrix} \quad (\text{Equation 4.20})$$

where  $a$  is the triangle's area in  $uv$  coordinates and is thus also constant. Usually,  $b_u=b_v=1$ ; but, suppose we increase  $b_u$ ,  $W_u$  would seek a larger value for the condition to reach zero, thus lengthen the cloth in the  $u$  direction. Conversely, reducing either  $b_u$  or  $b_v$  would shorten the cloth in that direction.

#### 4.2.1.2 Shear Condition

To express how cloth resists in-plane shearing, the extent to which a cloth triangle is sheared can be measured with the inner product of  $W_u$  and  $W_v$ . In its rest state, the inner product is obviously zero since  $W_u$  and  $W_v$  are perpendicular. Also  $W_u^T W_v$  is reasonably a form of approximation to the sine of the shear angle. So, the shearing condition is given by

$$C(x) = a W_u(x)^T W_v(x) \quad (\text{Equation 4.21})$$

where  $a$  is once again the triangle's area in the  $uv$  plane.

#### 4.2.1.3 Bend Condition

The bend condition is measured via the *dihedral angle* between pairs of adjacent triangles. The condition depends only on the four particles defining the two adjoining triangles (see Figure 4.4) and is defined simply with

$$C(x) = \theta \quad (\text{Equation 4.22})$$

where  $\theta$  is in fact the angle between the two triangles' normal unit vectors  $n_1$  and  $n_2$ .

Additionally, the following two relations will be referred to:

$$\cos \theta = n_1 \cdot n_2 \quad \text{and} \quad \sin \theta = (n_1 \times n_2) \cdot e \quad (\text{Equation 4.23})$$

where  $e$  is the unit vector parallel to the common edge shared by particles  $j$  and  $k$ .

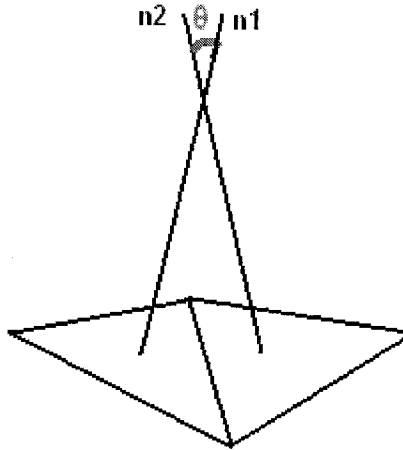


Figure 4.4. The angle  $\theta$  between the two normal vectors of adjacent triangles.

#### 4.2.2. Forces and force derivatives

Now that the condition functions are defined, the forces are obtained as follows. Each condition  $C(x)$  that the cloth seeks to minimize in order to reach an equilibrium state is associated an energy function

$$E_C(x) = \frac{k}{2} C(x)^T C(x) \quad (\text{Equation 4.24})$$

where  $k$  is the respective coefficient. Then, for each particle  $i$  that  $C$  depends on,

$$f_i = - \frac{\partial E_c}{\partial x_i} = -k \frac{\partial C(x)}{\partial x_i} C(x) \quad (\text{Equation 4.25})$$

and with this notation, it is understood that  $f_i$  is a vector  $\in \mathbb{R}^3$ .

The force derivative will be used extensively in the implicit integration scheme presented hereafter and is defined by

$$K_{ij} = \frac{\partial f_i}{\partial x_j} = -k \left( \frac{\partial C(x)}{\partial x_i} \frac{\partial C(x)}{\partial x_j} + \frac{\partial^2 C(x)}{\partial x_i \partial x_j} C(x) \right) \quad (\text{Equation 4.26})$$

for each pair of particles  $i$  and  $j$  that  $C$  depends on. Each  $K_{ij}$  is a 3x3 matrix.

### 4.2.3. Implicit Integration

We take advantage of this topic under the current model and organize it into two sections to fully explain the problem relating to numerical integration accuracy, stability, time step sizes and the reason for an implicit integration scheme. In addition to collision detection and response, the second bottleneck in cloth simulation seems to be the limitation of the time step size. Most cloth simulation models use small time step sizes because larger step sizes can lessen the numerical accuracy of the simulation, or worse, jeopardize its stability. Smaller time steps however mean more computational time for the simulation since to simulate a same amount of time now requires more iterations. While this limitation does not restrain grandiose applications such movie making, it does however threaten applications with real-time constraints. The first section will explain why small time steps are used, or conversely, how is the accuracy problem arising with large time



steps. The second section will detail the solution to this problem, the implicit integration scheme and more precisely, the Backward Euler method.

#### 4.2.3.1 The accuracy problem with larger time steps

With cloth simulation in general, the instability problem that can be encountered when using larger time steps is due in big part to the inaccuracy of the explicit integration methods such as explicit Euler. We know from basic calculus that interval integration is the approximation of the “area” under the curve via many little rectangles whose individual area is  $f(x) dx$ , and as  $dx$  gets infinitesimally small, this approximation becomes very accurate. But, with computers, we cannot afford such infinitely small  $dx$  which, in the case of cloth simulation is often identified as the time step size  $dt$  or  $h$ . As  $h$  becomes bigger, the less accurate are those approximations. If we looked at the popular Euler integration method, the same concern shows up. Each iteration, the method basically computes the following:

$$A_n = F_n / m$$

$$V_n = V_{n-1} + dt \times A_n$$

$$P_n = P_{n-1} + dt \times V_n \quad (\text{Method 4.27})$$

where  $F_n$  is the resulting force applied on the concerned object or element,  $A_n$ , its acceleration,  $V_n$ , its velocity, and  $P_n$ , its position or geometric state at time  $n$ . However,  $F_n$  is defined as  $F_n = F(P_{n-1})$ . But looking carefully, the term  $P_{n-1}$  currently being used as

the argument to compute the current force is in turn  $P_{n-1} = P_{n-2} + dt \times V_{n-1}$ , therefore once more dependent of the time step size. Continuing back further,

$$V_{n-1} = V_{n-2} + dt \times A_{n-1}$$

$$A_{n-1} = F_{n-1} / m$$

$$F_{n-1} = F(P_{n-2}) \quad \text{then again, } P_{n-2} = P_{n-3} + dt \times V_{n-2}, \text{ and so on.}$$

So, a big time step  $dt$  implies a big change in the velocity between two time frames, a big leap in positions and then the next time frame's force is computed with this position ... the bigger the leap, the less information it carries to the next time step. Thus, unless  $F(P)$  is a constant or is a linear function, different time step sizes will yield different results.

### 4.2.3.2 Deduction of an Implicit System

David Baraff and Andrew Witkin were not the first researchers to employ the use of implicit integration for cloth simulation since Terzopoulos et al. [TER87] did in 1987, but the two models are different and apparently, the implicit integration scheme has been ignored by most of the community since then until the more recent work. The implicit integration scheme can easily be presented by comparing the general explicit Euler method and the implicit method presented in [BAR98], named Backward Euler integration. Unlike the Explicit Euler integration that computes

$$v^{t+h} = v^t + h a^t \quad \text{where } f^t = m a^t$$

$$\text{and} \quad x^{t+h} = x^t + h v^{t+h},$$

(Equation 4.28)

Implicit Euler uses and evaluates the forces and therefore their accelerations or velocity derivatives at the end of the time step  $t+h$  instead of the beginning  $t$ . So,

$$v^{t+h} = v^t + ha^{t+h} \quad \text{where } f^{t+h} = ma^{t+h}$$

and

$$x^{t+h} = x^t + hv^{t+h}$$

(Equation 4.29)

with  $t$  being what is known now at the beginning of the time step. The state and information at the beginning of the time step can also be regarded as information that was computed by the end of the last time step. Similarly, what we seek to obtain at time  $t+h$  can equivalently be regarded as either the update at the end of the current time step or the knowledge to the next time step. It is then observable that the difference between the two integration schemes is that the implicit method does not use past known forces to integrate, but present and unknown forces that have to be guessed and approximated.

Let  $f$  be a function of position and velocity  $f=f(p,v)$ , then  $f^{t+h}=f(p^{t+h},v^{t+h})$ , but both  $v^{t+h}$  and  $p^{t+h}$  are also what is sought and are, therefore, unknowns.

The displacement in position and the change in velocity can be expressed as

$$\Delta v = h \times f/m = ha = h M^{-1} f(x^t + \Delta x, v^t + \Delta v)$$

and

$$\Delta x = h(v^t + \Delta v)$$

(Equation 4.30)

Now, let us suppose that force  $f$  is only a function of position, and then we would still need the position at the end of time step which we do not have yet.

$$x^{t+h} = x^t + \Delta x$$

But, remember that  $\Delta x = h v^{t+h} = h(v^t + \Delta v)$

So, all really result in determining  $\Delta v$ .

Going back to the equation  $\Delta v = h M^{-1} f(x^t + \Delta x, v^t + \Delta v)$  where neither  $\Delta x$  nor  $\Delta v$  are known, we approximate  $f$  by applying a Taylor series expansion to  $f$  but keeping only a first order approximation.

$$f(x^t + \Delta x, v^t + \Delta v) \sim f^t + (\partial f / \partial x) \Delta x + (\partial f / \partial v) \Delta v + \text{the rest ignored}$$

*(Equation 4.31)*

Consequently, it becomes

$$\Delta v = h M^{-1} (f^t + (\partial f / \partial x) \Delta x + (\partial f / \partial v) \Delta v)$$

*(Equation 4.32)*

Let us reduce it to one unknown by substituting  $\Delta x = h(v^t + \Delta v)$  into the above equation yielding:

$$\Delta v = h M^{-1} (f^t + h(\partial f / \partial x) \times (v^t + \Delta v) + (\partial f / \partial v) \Delta v)$$

*(Equation 4.33)*

Finally, let  $I$  denote the ID matrix and gathering  $\Delta v$  on the left side, we get the linear system:

$$(I - h^2 M^{-1} (\partial f / \partial x) - h M^{-1} (\partial f / \partial v)) \times \Delta v = h M^{-1} (f^t + h(\partial f / \partial x) v^t)$$

*(Equation 4.34)*

Thus, we see the need to compute the force derivatives with respect to position and the force derivatives with respect to velocity.

Next, let us recall that specifically for the currently referred model where forces are determined via  $f_i = -\partial E / \partial x_i = -k(\partial C(x) / \partial x) \times C(x)$  where  $C(x)$  is the condition function which we want to be zero, that is, the rest state is reached when  $C(x)=0$  thus where the energy  $E$  is minimal,  $C(x)$  does not depend on the velocity  $v$ , then  $\partial f / \partial v = 0$  and what remains to be solved is only

$$(I - h^2 M^{-1}(\partial f / \partial x)) \times \Delta v = h M^{-1} (f^t + h(\partial f / \partial x)v^t)$$

(Equation 4.35)

#### 4.2.4. Damping Forces

According to Baraff & Witkin's model, each condition is accompanied by its own damping force, and each damping force should only damp its corresponding condition-generated forces. For instance, stretch damping should not damp any motion other than the ones caused specifically by stretch or compression. Yet, a suitable well-chosen damping force function of both position and velocity must be at hand to prevent anomalous in-plane oscillations of the particles. The damping function should not be defined in terms of energy  $E$ , but in terms of the condition functions  $C(x)$  used to define energy. The proposed form for the damping force is:

$$d = -k_d \frac{\partial C(x)}{\partial x} C'(x) \quad (\text{Equation 4.36})$$

where  $k_d$  is the damping coefficient associated with the condition and  $C'(x)$  is the derivative of  $C(x)$  with respect to time. The equation can also be seen as

$$d = -k_d \frac{\partial C(x)}{\partial x} \frac{\partial C(x)}{\partial t},$$

$$\text{or } d = -k_d \frac{\partial C(x)}{\partial x} \left( \frac{\partial C(x)}{\partial x} \frac{\partial x}{\partial t} \right),$$

$$\text{or equivalently } d = -k_d \frac{\partial C(x)}{\partial x} \left( \frac{\partial C(x)}{\partial x} x' \right)$$

(Equation 4.37)

where  $x' = \frac{\partial x}{\partial t} = v$ , the velocity. Thus, the damping strength depends on the component

of the velocity that is in the  $\frac{\partial C(x)}{\partial x}$  direction and the damping force acts in the opposite

direction of  $\frac{\partial C(x)}{\partial x}$ . Since  $\frac{\partial f}{\partial x}$  and  $\frac{\partial f}{\partial v}$  are required and damping is now a contributing

force to the cloth system, the terms  $\frac{\partial d}{\partial x}$  and  $\frac{\partial d}{\partial v}$  as well as the condition specific

derivatives, their terms, and further implementation details are given in the appendices.

[APPENDIX A]

#### 4.2.5. Constraint handling, collision detection and response

A particle may be constrained or completely unconstrained in addition to being subject to forces. Also, a particle may be constrained in one, two or three dimensions, and it is the particle's acceleration or change in velocity that is constrained. But first, Baraff and Witkin do not use the following constraint enforcement mechanisms for reasons explained thoroughly in [BAR98]:

- Reduced coordinates
- Penalty methods
- Lagrange multipliers

In brief, the reduced coordinate method of constraints is rejected because changing the number of coordinates on a particle basis alters the size of derivative matrices and complicates the system immensely. The penalty method consists of adding more stiffness to the system essentially by adding stiff springs. Baraff and Witkin do not discard this method entirely and even consider it for the case of cloth self-interactions; but, while adding additional stiffness to the system, this mechanism does not guarantee constraint enforcement. The third constraint method, Lagrange multipliers, is simply avoided because its use would imply augmenting the linear system with additional variables, the multipliers, and equations, the constraint conditions. The result would turn the current definite system into an indefinite one and double the running time, thus, degrading the performance of the simulation.

Instead, the mechanism they used is *Mass Modification*. That constraint enforcement mechanism employs the concept of *inverse mass* introduced in their work. It is noticeable that the term  $M^{-1}$  is used in the model's equations instead of  $M$  to illustrate the use of inverse mass. For a particle, it is simply  $x_i'' = \frac{1}{m_i} f_i$  to describe a particle's acceleration. So, with the use of inverse mass, it is straightforward to constrain this particle by altering its mass. For example, making the term  $\frac{1}{m_i}$  tend towards zero by giving it an infinite mass will prevent a particle's velocity from changing. This is exactly equivalent to making it ignore the forces exerted on it.

In more detail, it is possible to constrain a particle's acceleration in only one or two dimensions, and obviously in all three like previously. If we stopped thinking of a particle's mass as a scalar and express its inverse in matrix form instead, it then becomes possible for instance to limit its change in velocity along the z dimension via

$$\mathbf{x}_i'' = \begin{pmatrix} \frac{1}{m_i} & 0 & 0 \\ 0 & \frac{1}{m_i} & 0 \\ 0 & 0 & 0 \end{pmatrix} \mathbf{f}_i \quad (\text{Equation 4.38})$$

An unconstrained particle would therefore have its inverse mass's 3x3 matrix expressible by  $\frac{1}{m_i} \mathbf{I}$  where  $\mathbf{I}$  is a 3x3 identity matrix. Going into even further detail, it is possible to prevent a particle from accelerating along a unit vector  $\mathbf{p} \in \mathbb{R}^3$  by defining its inverse mass matrix with  $\frac{1}{m_i} (\mathbf{I} - \mathbf{p} \mathbf{p}^T)$ . Also, preventing a particle from accelerating in either  $\mathbf{p}$  or  $\mathbf{q}$  direction where  $\mathbf{p}$  and  $\mathbf{q}$  are two mutually orthogonal unit vectors in  $\mathbb{R}^3$  is done via defining its matrix as  $\frac{1}{m_i} (\mathbf{I} - \mathbf{p} \mathbf{p}^T - \mathbf{q} \mathbf{q}^T)$ .

Now, let  $\mathbf{W}$  be the modified version of  $\mathbf{M}^{-1}$ . If we wanted to constrain a particle to a non-zero change in velocity  $\mathbf{z}_i \in \mathbb{R}^3$  and let all  $\mathbf{z}_i$ 's be represented by  $\mathbf{z} \in \mathbb{R}^{3n}$ , then, the previous system becomes

$$(\mathbf{I} - \mathbf{h}^2 \mathbf{W} (\partial \mathbf{f} / \partial \mathbf{x}) - \mathbf{h}^2 \mathbf{W} (\partial \mathbf{f} / \partial \mathbf{x})) \times \Delta \mathbf{v} = \mathbf{h} \mathbf{W} (\mathbf{f}^t + \mathbf{h} (\partial \mathbf{f} / \partial \mathbf{x}) \mathbf{v}^t) + \mathbf{z} \quad (\text{Equation 4.39})$$

The authors of this model stated that nothing substantial was added to the subject of collision detection and response from their work. Similarly to the first collision process



described in this work, cloth self-collision is detected by pairs of either type point-triangle or edge-edge of collisions. The optimization technique used to avoid  $O(n^2)$  comparisons is a coherency-based bounding-box approach. The response process for cloth-cloth collision is the use of strong damped springs to push the cloth apart when collision is detected. Though not strictly a frictional force, a form of dissipative force tangent to contact is used to reduce sliding. As for collisions between the cloth and rigid objects, detection is done between particles and the solid object faces while the solid objects are grouped into hierarchical bounding box tree for optimization. The corresponding collision response constrains the cloth particle's velocity to be made consistent with the velocity of the solid object. Also, the position alteration is adapted to the backward Euler step via the term  $y_i$ , a vector in  $R^3$ , to represent the correction. [BAR98].

#### **4.2.6. Adaptive time stepping**

Although this second model is more complex than the first model presented in this work, it still opts for an animation goal and not an engineering one. In other words, a visually pleasant and numerically stable solution is given preference over an accurate solution. Instability should be identified and avoided before it is seen on the display. It is clear that like for most models, the stiffness and greatest potential instability factor arises mostly from the strong stretch forces, so an adaptive time stepping solution is suggested:

- Keep track of the stretch terms.
- If any triangle undergoes a drastic change in its stretch condition in either u or v direction, discard the proposed state, reduce the step size, and try again.
- After two successes, retry to increase the step size. If it fails again, reduce the step size and wait longer before trying to increase again.

#### **4.2.7. A modified Conjugate Gradient Method**

For relatively small systems in terms of the number of particles, the linear system of equations of the cloth simulation expressed as  $Ax=b$  can directly be solved with methods such as Gaussian elimination; However, when the number of particles reaches the hundreds or thousands, iterative methods such as the conjugate gradient method would be preferable. The authors describe a modified version of this method in order to include their constraint handling approach. This modified conjugate gradient method is a relevant addition to Baraff and Witkin's cloth model but because we do not develop on it any further, we will leave its detailed discussion in the APPENDIX C.

## Chapter 5: Design

In this chapter, and in Chapters 6 and 7, we describe work actually carried out for this thesis. Our starting point was the algorithms discussed in Chapters 3 and 4 but, in each case, we experimented with extensions and modifications to these algorithms.

To begin with, two cloth simulators are implemented, each carrying out the properties and algorithms of the model described in Section 4.1 and the one in Section 4.2 respectively. The two simulators will first allow us to compare two different formulations of cloth mechanical behaviors and internal forces, and see how various integration schemes influence the outcome of a simulation. The simulators will then serve as a basis to determine further concerns encountered while constructing the models and performing tests with them.

### ***5.1. Simulator 1: A mass-spring model***

The first cloth simulator follows the mass-spring model described in Chapter 4 in most of its aspects. Cloth will be represented using particles or nodes linked with linear springs. The same three types of springs, namely structural, shear, and flexion, are defined to model the behaviors of stretch, shear, and bend with cloth. A regular triangle mesh structure is used to give shape to a rectangular piece of cloth. [Figure 5.1] Very small numbers of particles are tested for this simulator. The default setting constructs the

grid with simply eight times eight, sixty-four, particles. The alternative mesh resolution tested is sixteen by sixteen. The tests targeted using this first system range from simulating a piece of cloth hanging from one or more of its corners, a realistic looking flag flapping in the wind, to cloth draping and interactions with other rigid objects in the scene.

The mass of each particle is determined using the following formula:

$$\text{mass} = \rho \frac{1}{m} \frac{1}{n} \frac{1}{2} \frac{1}{3} 6 = \frac{\rho}{mn} \quad (\text{Equation 5.1})$$

The term  $\rho$  is the density of the cloth in Kg/m<sup>2</sup> units. To explain the remaining factors composing the particle's mass,  $m$  and  $n$  are here respectively the number of particles there is per row and the number of rows the rectangular cloth grid has. Since every triangle in a regular triangle mesh originally shares the same size, one third of the mass residing in the triangle's area is attributed to a particle's mass whether the particle is positioned in the middle of the grid or belongs to a side edge, therefore, avoiding lighter cloth sides. And, every particle under our mesh structure that is neither on one side nor in a corner is connected to six triangles.

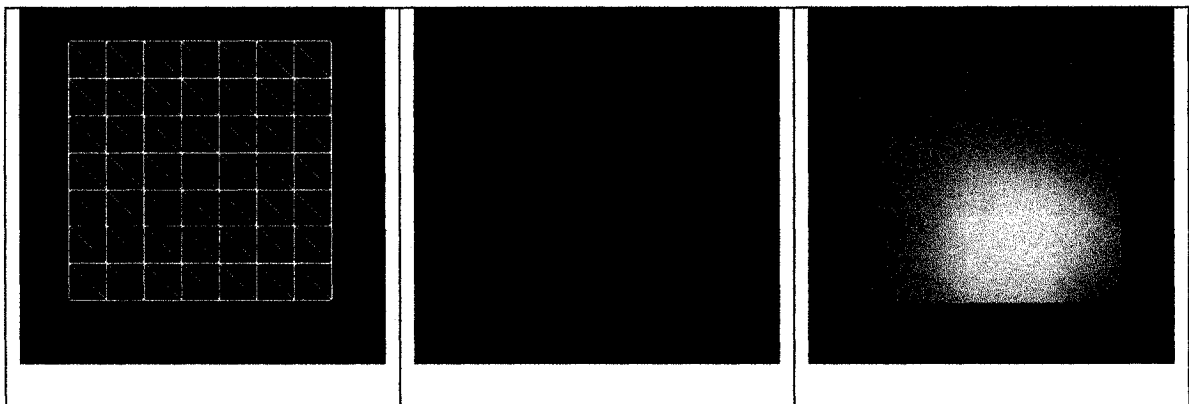


Figure 5.1. From left to right: The mesh structure, the underlying spring system, and the rectangular piece of cloth simulated.

The internal forces generated by the springs are obtained with an equivalent formulation to the one presented in Chapter 3. Let  $i$  and  $j$  now be particle indices, then

$$F_{\text{internal}}(\mathbf{P}_i) = - \sum_{j \in S} k_{ij} \left[ L_{ij} - L_{ij}^0 \frac{\mathbf{L}_{ij}}{\|\mathbf{L}_{ij}\|} \right] \quad (\text{Equation 5.2})$$

The total forces due to mechanical reactions internal to the cloth on a particle with index  $i$  is obtained by summing for all indices  $j$  belonging to a set  $S$  whose particles are linked to particle  $i$  via a spring of one of the three types described previously. Then,  $k_{ij}$  is the associated stiffness coefficient of the type of spring linking the two particles,  $L_{ij}$ , the vector from particle  $i$  to particle  $j$ ,  $\|\mathbf{L}_{ij}\|$ , its current length, and  $L_{ij}^0$ , its natural length when in a rest state.

Gravity and wind will be the two present external forces in the surrounding environment of the cloth simulated. Other rigid objects will also pose physical constraints to the cloth. The simulations will advance in time using the explicit Euler integration method. Every particle will see its velocity derivative, its velocity, and its positional state evolve in time. We will be able to try different time step sizes with the simulator and observe the effects they have on the motion of the cloth object, the difference in shape, and the stability of the system.

If the phenomenon of “super-elongation” referred in Chapter 4 were to occur, deformation rates will be limited with a threshold percentage value no greater than fifteen percent, and post-step repositioning will be called upon. While we also believe that only a very limited proportion of fabric types can even allow elongations reaching a significant percentage, unlike certain authors, we will still allow a non-zero threshold

value in order to study the stretching behavior. However, several advantages are obtained by discarding entirely the stretching behavior of cloth both in terms of implementation ease and in terms of overhead, stability and speed of the simulation because stretch can, in some cases, be hard to notice. For example, in the case of computer animations where the actors wearing virtually simulated clothes move continuously and the states of the cloth, its shape, change rapidly; then, allowing or disallowing the clothes to stretch by a five percent will no longer make any perceivable difference in a ten second movie sequence. Nevertheless, among the tests to be performed, this simulator will be able to show final drapes of cloth in its environment as well as capture static poses of any instant in simulation, so it will be possible to visualize the stretching properties of the fabric with significance. Viscous damping will be used as first remedy to deal with the occurrence of possible oscillations with this spring model.

In detail, the super-elasticity problem is dealt in our work by simply identifying the springs, edges, whose elongation exceeds a predetermined threshold value, say ten percent of the so-called natural length of the edge, and then applying a positional correction of each end node towards its counterpart by half the excess unless the latter is under positional constraint which requires the point to take the full excess correction. Our method is a post-step corrective measure that is a simplified adaptation of the solution suggested by Provot.

Collision detection between scene objects and cloth or cloth self-collision detection is handled in the same way for our simulator. Both Point-triangle and edge-edge types of collisions will be detected. We do not plan to use any optimization method for the

moment. This decision may be revised if the performance of the simulations does not meet real-time constraints. But, due to the small number of nodes, it will likely not necessitate optimizations.

## ***5.2. Simulator 2: An energy-condition function model***

A second simulator was designed to achieve the same simulation goals as the first, under the same constraints, but, using a different cloth model and another integration method so that we are given at least two different systems to compare, two points of view to the same problem. This second simulator will allow the choice between remaining with the previous Explicit Euler method and utilizing the new implicit integration alternative method. The version in which the prior integration scheme can be kept allows us to compare the internal dynamics of two cloth models alone. Then, activating the implicit integration allows the analysis of the two integration methods on the same simulator. The cloth model this simulator implements resembles the one described in Section 4.2, with a few exceptions. First, our data structures differ from the ones used in the references. Given a cloth object made up of  $n$  particles, instead of constructing matrices of size  $n \times n$  where each entry is a vector, and using these sparse but huge structures to later solve the system of equations, we will use loops of size  $n \times n$ , and solve for each particle per iteration of the loop. The work is the same, but the structures utilized are not. Next, after a few observed tests were performed with the first simulator as well as with the explicit integration version of this second simulator, we observed that the instability that would require an implicit solution mostly originates from in-plane

forces, and mainly from the strongest of all, stretch. So, after implementing the stretch condition in the implicit integration version of the simulator, we decided to also implement the shear condition, but to leave out the bending condition to remain in its explicit version only. Our justification is first a matter of simplicity; secondly, those out-of-plane forces were not causing significant instability compared to stretch forces. Additional data structures need to be built for this second model. For example, each particle will have its force derivatives with respect to positions and velocities computed and stored during the simulation. In reality, most of the design for this second simulator has already been established by reusing the first simulator and modifying the appropriate components, since after all; every cloth simulator can be designed following the steps and tasks we described in section 2.3 and will show a similar structural layout.



## Chapter 6: Implementation details

### 6.1. Details of Simulator 1

The simulator can be structured into the components and the functionalities depicted in the diagram of Figure 6.1. Such a functional layout of the simulator agrees with our subdivision of the cloth simulation problem into tasks described in the third part of Chapter 2.

First, we initialize the data structures of the simulator to reflect the starting state of the cloth object:

- Node array initialization
- Construction of springs
- Construction of surface triangles

Then, the simulation loop is established as a set of components that are repeated for every frame to be displayed:

- Internal forces of cloth springs
- External forces and constraints
- Damping
- Numerical integration: Explicit Euler
- Post-step modifications
- Collision detection and response
- Rendering

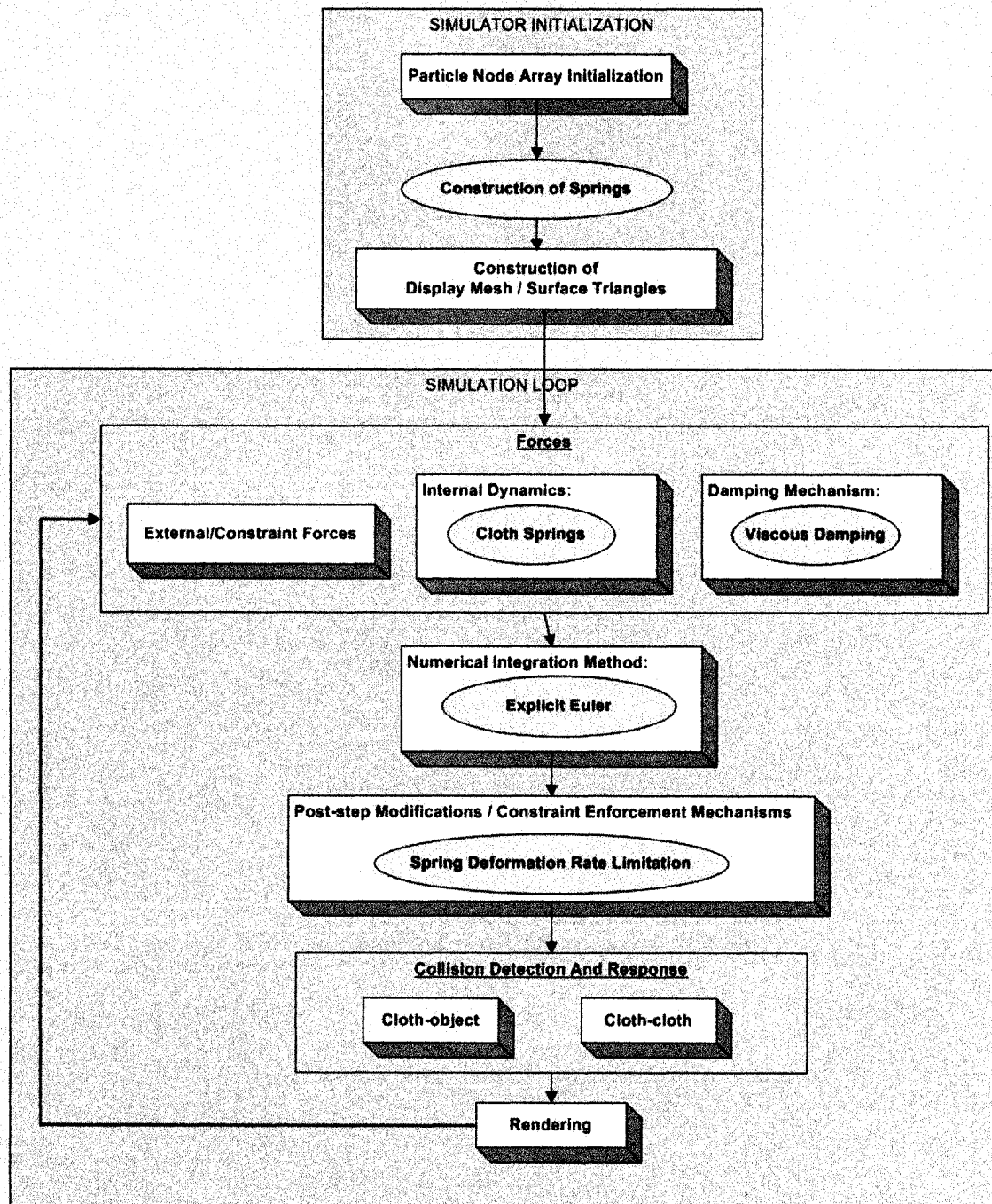


Figure 6.1. A component diagram for the first cloth simulator.

If we described the simulator using object-oriented terminology, then those components listed above can be thought as methods of the big simulator class, and the

data structures like particles, edges and springs, surface triangles, and other rigid objects will be classes. The three methods in initialization are used in that order to generate the starting state of the cloth object in its world space. We start by positioning particles and giving them their mass. Although we could give the cloth object any random starting configuration, it would only make sense to build a starting state that is presumably a rest state for the cloth, or, any state that would be realistic yet simple. Starting states where the particles are at great distances or unrealistic positions causing powerful stretch, shear, or bending forces will only produce fast and chaotic motions in the first few frames. For our simulator, a few starting states were made available for testing and among them, the two simplest are most often used. The first sets all the particles coplanar to the  $xy$ -plane and all evenly spaced in both axes yielding a simple rectangular flat surface where gravity pulls downward in the negative  $y$  direction. The other simple configuration makes that same flat rectangular surface parallel to the  $xz$ -plane instead and orients it facing upward in the positive  $y$  direction. To perform a simple test of the three mechanical properties of the cloth model individually without any external factors, we will see that it suffices to configure an initial state for the cloth where the particular type of springs tested is overstretched while every external force is deactivated. To base our first simulator on Provot's mass-spring model, edges will be placed between every pair of adjacent particles and diagonal adjacent neighbors once the particles are in place; and, the edge lengths and the particle nodes they are attached to will also be recorded as information. Springs will then be defined over these edges; in fact, it is preferable to just merge the edge class and the spring class, and only keep the latter because the spring class would already carry all the information of the edge class plus the stiffness

coefficient. As presented in Chapter 3, three distinct sets of springs will be constructed and stored into their respective storage lists. Finally, because the surface normal vectors are required for proper rendering and also for computing the effects of fluids and similar forces on the cloth surface, it becomes necessary to find a way to record the information of the triangles defining the cloth surface. It makes sense simply to use the particles as nodes and the edges previously defined for the triangles.

Once all the data structures and initial information required for our cloth object are set, we can start constructing the simulation loop. The first three methods in the simulation loop that precede the integration component all compute forces applied to the cloth object based on the information known at the beginning of the time step about the state of the cloth, thus, the order in which the three methods appear can be permuted without harm.

Let us take a closer look at each of the methods in the simulation loop:

- For a mass-spring model, the internal forces component will go through verifying every spring composing the cloth and gather forces caused by either elongation or compression over these springs. For each spring, the method will store the generated forces to the sums of forces acting on the concerned two particles located as ending nodes of the spring. Equivalently, every particle will have the forces caused by the twelve springs it is connected to be summed.
- External forces describe all the factors and constraints originating from the environment surrounding the cloth object that produce or modify the forces acting on the cloth. In fact, any force not produced from the internal mechanisms describing the properties and

behaviors of cloth are categorized as external. These forces are responsible for moving the cloth out of its equilibrium and rest state and are the first to trigger the internal dynamics to react. The external sources of forces can be as simple as gravity itself, wind, or other fluids in which the cloth is simulated, or come from more specific physical constraints such as parts of the cloth being fixed to a rigid object in the scene. The cloth-object collisions themselves are also external factors, but because the description of collision handling can possibly reflect additional cloth properties like surface rigidity, it is handled separately. The external forces defined for our simulators are gravity, wind, and the possibility to fix any of the four corners of the rectangular piece of cloth. Gravity is simulated simply by adding to each particle a force in the negative direction of  $y$  with strength equaling the particles mass multiplied by a gravity acceleration factor of 9.8 meters per second<sup>2</sup>. We fix a cloth particle to a world position by maintaining the total forces applied to it at zero, in both the internal force and external force methods. In other words, we assume that a fixed position constraint means that an opposite force stronger than any other force applied to the particle can be generated and can counter its movement.

- Damping methods are located before the integration methods because their formulations generally create forces, and forces must be accounted for before the numerical integration takes place. Damping is used to subdue the oscillations of the springs since real cloth would allow very little oscillations if any to be seen. Thus, subduing the oscillations as quickly as possible would make the simulation more visually realistic. Of course, damping formulations that are velocity dependent such as the current viscous damping method only come into play once the cloth's velocity is no longer zero. Viscous damping

is a basic and fast solution to start with in order to maintain the simulation stable. Without any form of damping, we will see in the following chapter that the current system under gravity would make the cloth bounce up and down over its equilibrium point, or even worse, with a too large time step and additional forces other than gravity, the system could completely lose its stability and eventually diverge. For each particle, we compute its viscous damping force by multiplying its velocity vector by minus one times the damping coefficient.

- The physical system simulated evolves with the chosen basic Euler integration method. During each iteration, the simulation loop will compute and display one animation frame. The method will update the state for each particle starting by taking the total forces computed by the previous three methods and then sequentially update the change in velocity, the new velocity, and the new position. What should be remembered about explicit methods is that the positions and velocities fed to compute the forces are the ones known at the beginning of the time step, whereas the ones to be obtained by the update are of the next time step or equivalently, of the end of this time step. Once the resulting positions are calculated, we reset the forces on the particle to zero for the next iteration. The resetting does not have to be performed at this time, but if no forces were computed after the numerical integration, then it would be appropriate to execute this task as soon as possible.
- Post-step corrective measures are very likely the last means for eliminating artifacts and enforcing a visually realistic simulation where the mechanisms of the described model failed. With the current example, the model utilizes linear springs to represent the in-plane stretching behavior of fabrics while the actual real behavior of cloth is rather non-

linear. The consequence of using such a model is to face the dilemma of either using springs that are too stiff, that have high value coefficients, and encounter more instability problems, or, to use smaller spring stiffness values and get artifacts such as the super-elongation of cloth edges. With the second choice, it is still possible to remedy to these artifacts before the display phase. We chose to explicitly apply positional corrections in the same way Provot suggested in his paper. A threshold value between one percent and fifteen percent is set depending on the stretching properties of the fabric we try to simulate. Whenever the deformation rate of a spring exceeds this maximum allowed value of ours, we will pull the two ending nodes of that spring back towards each other each by half the excess so that the spring length equals the maximum allowed. To be consistent with our handling of the positional constraints of nodes that must remain fixed during the time interval, particles under such constraints will not be corrected and the particle on the other end, if loose, will have to be corrected by the full excess length. In the case where elongation beyond the threshold value occurred on a spring whose two ends are both constrained as fixed, we decide to leave the state as it is but, instead we will just have to make sure that no such constraints are set. Because the resulting particle positions have now been determined by these corrections, it is then the particle velocities and change in velocities that need to be updated. Such an update in the reverse order of our integration method is called Inverse Dynamics procedures [PRO95]. Inverse dynamics procedures are used to make all the information of the state of the particles consistent. This way, for each particle, moving from its last position with the updated velocity will result in the position recorded. In our case, the inverse dynamics need only to stop once the correct velocities are obtained because forces are being reset after the

numerical integration method. At first, we thought it would be of no harm to skip this step and omit including the inverse dynamics procedures into this component, and so, our initial design left out this task. The disadvantages of skipping these steps will be presented in Chapter 7.

- Collision detection and response can either precede the post-step modification method in the sequence of computation, or, follow after it. The choice depends on whether it has priority over the post-step modification method or not. The reason behind considering the chronological order between these two methods is explained by the observation that both methods will make modifications to the final particle positions and velocities of the time interval. The particle velocities have an impact on the cloth's state for the next time step, while the particle positions have an immediate consequence on what is displayed this time step. The problem is that these two methods may interfere with each other. If collision detection and response were placed first in time, nothing guarantees that the post-step correction method would not create new collisions that would then be displayed without correction. Similarly, the collision response method may be unable to both solve a collision and keep the length of each spring equal to or below its capped value. We judge that the worse case between the two would be to see an unsolved collision state where two surfaces would go through each other. Compared to a super-elongation artifact, an object poking through the cloth or vice-versa is much less attractive and consequently, we decided to choose the lesser of the two evils and place the collision handling method as final method before rendering.

We used Provot's approach described in Chapter 4 for collision detection and handling. We decided however to apply a few implementation variances while building



up both the point-triangle and edge-edge collision detection algorithms. Just like Provot's approach, we started by constructing the cubic function of variable time  $t$  out of the two conditions describing each type of collision in Section 4.1.4. We found out that solving a cubic equation was significantly more complex than solving a quadratic equation. There are known methods such as Cardano's, but our implementation of this method was not always successful in producing the correct result. So instead of solving the cubic, we decided to simply determine whether there is a solution or not to the cubic equation within the time interval concerned. In case there was a cubic solution in the time interval, there are two different situations:

First, the cubic root is either a local minimum or a maximum. In this situation, we compute the derivative of the cubic equation with respect to  $t$ , and find the two roots of this quadratic equation. We then verify if either root falls within the interval and, if so, determine whether it is also a cubic root by substituting it back into the cubic equation.

The second situation is a root where the cubic curve crosses the  $Y=0$  line. If such a root existed within the interval, then substituting both the beginning interval value  $t_0$  to obtain  $Y_1$  and the interval's end value  $(t_0+dt)$  for  $Y_2$ , would yield two values  $Y_1$  and  $Y_2$  of opposite signs. In which case, we would use a bisection method and recursively try to locate the "exact" value of the cubic root until the sub-interval is small enough to our satisfaction. In our implementation, we stopped once the sub-interval was one eighth of the original time interval size. At this point, we arbitrarily estimated the cubic root to be the mid-value of this sub-interval.

The only case left off is if either or both of the starting and ending values of the interval were a cubic root of the type described in the second situation. Then, their sign

could possibly not be opposite. This exception is easily covered by verifying if either time values were in fact a cubic root we are looking for.

With either type of collisions, point-triangle or edge-edge, the substitution step back into the respective first condition actually meant performing an inclusion test. For the point-triangle type of collisions, this was of no concern because its inclusion test assumes coplanarity and our approximation approach would simply imply performing the inclusion test between the triangle and the projection of the point on the triangle's plane instead of the real point itself. But, since, the approximation is relatively close to the actual result, meaning the position of the point approximated is either on the triangle's plane or very close to it, this does not lead to an erroneous result. However, for the edge-edge intersection test, an approximation is never good enough. This particular "inclusion" test does not make any assumption such as coplanarity; either two edges intersect at a single point, on a line segment, or do not intersect at all. Thus, using approximate positions at a time that is not exactly equal to the collision time will always yield a false result. Instead of performing the simple edge-edge intersection test, we had to determine if the distance between the two lines were small enough and also find their two closest points and test if both points would belong to the two segments of the edges concerned. Both our point-triangle and edge-edge collision detection functions worked for collision between the cloth and moving objects in the scene as well as for cloth self-collisions in our simulator.

Given the small number of nodes used in our simulator, the simulator did not require the implementation of the optimization techniques. The frame rate displayed remained satisfactory without optimization. Nevertheless, optimizations for collision

detection would become necessary for the simulation of more complex garments or for a larger number of nodes.

We also adopted Provot's approach mainly for collision response. We obtained satisfactory results using this method for collisions between scene objects and the cloth object. But, for cloth self-collision, due to time constraints we did not implement the handling of multiple-collision handling accurately and were unable to maintain collision consistency when multiple collisions occurred and caused additional collisions by solving them. This informs us that cloth self-collision handling cannot be complete without addressing the issue of consistency of multiple collisions. We include the implementation of this characteristic among our future projects.

- Given our data structures and classes, the rendering component can offer the display of either the simulated cloth or the underlying structures modeling it. Different textures can be mapped onto the triangles of the cloth surface that were defined. The normal vector is calculated for each particle as the average of the normal vectors of all the triangles to which the particle also belongs. Each particle is drawn as a node of a surface triangle using the positions computed for the end of the time step. We decided to define the triangles by dividing each cloth quad with the edge going from the top left node to the bottom right node as illustrated on the leftmost image of Figure 5.1. The quad splitting could have alternatively been from the bottom left node to the top right, as long as no additional nodes need to be defined other than the particles; that is equivalent to saying that the number of nodes used to model the cloth mechanically and the number of nodes used to display are the same. No tessellation or geometrical techniques were used to refine the resolution of the mesh. In addition to displaying the cloth simulated with

triangles, the component also allows the display of the particles and the three types of springs: stretch, shear, and flexion; each, in a different color.

## **6.2. Details of Simulator 2**

The structural layout of the second simulator is similar to the first except a few components are added or replaced by methods corresponding to the second model.

[Figure 6.2]

In the initialization section, the terms  $df/dp$  and  $df/dv$  need to be defined for every particle while springs are no longer used:

- Node array initialization
- Construction of surface triangles
- Initialization of the terms  $df/dp$  and  $df/dv$

To be more precise, both  $df/dp$  and  $df/dv$  are arrays of size  $n \times n$  where each entry is a three by three matrix because each entry can be identified as  $df[i]/dp[j]$  or  $df[i]/dv[j]$  where  $i$  and  $j$  are indices going from 0 to  $n-1$  and represent the derivative of a force applied on a particle  $i$  with respect to the position or velocity of particle  $j$ . Indeed, many entries of either  $df/dp$  or  $df/dv$  are null matrices and those two big structures are sparse since only pairs of particles  $i$  and  $j$  that belong to the same triangle can have a mechanical relationships according to this model and would give a non-zero  $df[i]/dp[j]$  or  $df[i]/dv[j]$  entry.

In the simulation loop, because the internal forces are not obtained from a system of springs but from the derivatives of condition functions, a new method need to be in place and compute them for this new model. The terms  $df/dp$  and  $df/dv$  need to be computed each iteration before the numerical integration. The integration method is changed from the regular explicit to the more accurate implicit Euler method. Nonetheless, the sequence of computation remains similar for the simulation loop:

- Internal forces from condition functions
- External forces and constraints
- Compute  $df/dp$  and  $df/dv$
- Damping
- Numerical integration: Backward Euler method
- Post-step modifications
- Collision detection and response
- Rendering

It is always possible to revert to the explicit Euler method and keep every other component active, depending on the version of integration scheme that is to be tested.

- The component calculating the forces originating from internal dynamics for this model is more complex than the one used in the first simulator. Our implementation employs a hierarchy of methods to complete this task because of the number of unknown terms involved hereafter. Basically, every term that is unknown at the time it is called upon will have a corresponding method calculate its value, whether it is a scalar, a vector, or a matrix. The root method starts by establishing a loop for each of the three conditions

modeling stretch, shear, and bend properties of cloth. The reason for distinguishing one condition per loop is because while the loops of stretch and shear conditions would proceed on a per triangle basis, the loop for bend condition must proceed by pairs of adjacent triangles. But despite the differences, all three loops will calculate forces as described by equations 4.20 to 4.22 in section 4.2.2 and will add the forces exerted by these mechanisms on the corresponding three particles per iteration, or four in the case of bend condition.

- Let us start with the stretch condition. For each triangle of the cloth mesh, there are three particles identified by indices  $i,j,k$  for which the method must gather the stretch forces. The three particles affect one another in that the condition depends on all three particles. For this first condition in particular, given the definition of the stretch condition function, we can decompose the stretch force into its  $u$ -coordinate term plus its  $v$ -coordinate term, describing respectively the stretch versus each axis in the  $uv$  space.

Let  $m$  be one of the three indices  $i,j,k$ ; then, the stretch condition forces on particle  $m$  are given by

$$f_m = -k_s \frac{\partial C_u}{\partial P_m} C_u + -k_s \frac{\partial C_v}{\partial P_m} C_v . \quad (\text{Equation 6.1})$$

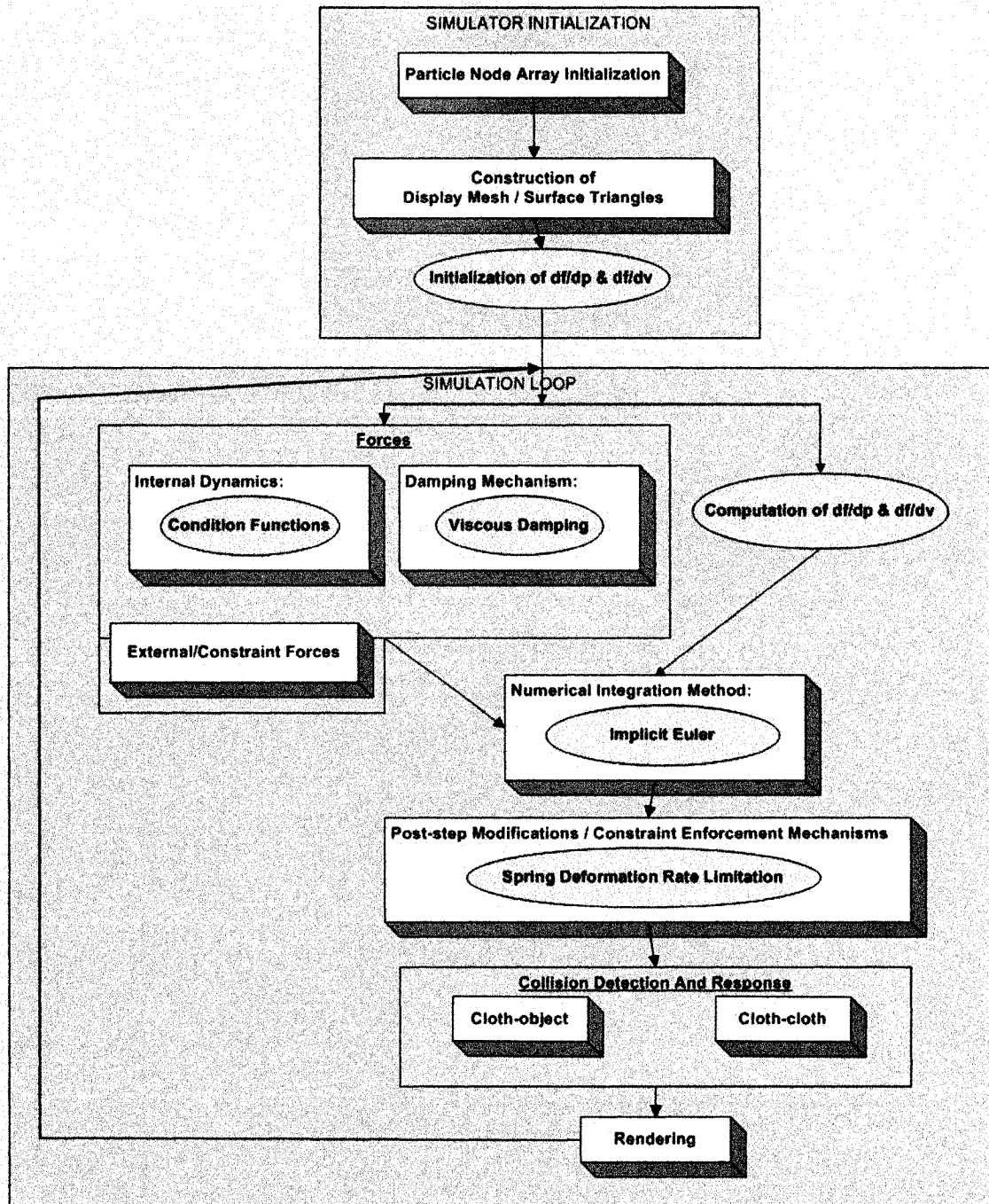


Figure 6.2. The component diagram for the second simulator.

The first derivatives of the stretch condition function with respect to the position  $P_m$  of particle  $m$  are given as

$$\frac{\partial C_u}{\partial P_m} = \alpha \frac{\partial W_u}{\partial P_m} \frac{W_u}{\|W_u\|} \quad \text{and} \quad \frac{\partial C_v}{\partial P_m} = \alpha \frac{\partial W_v}{\partial P_m} \frac{W_v}{\|W_v\|}$$

(Equation 6.2)

where  $\alpha = a^{3/4}$  and  $a$  is the constant triangle area in  $uv$  coordinates. The reason for replacing  $a$  with  $\alpha$  is justified in [PRI03] as a scale invariance solution, and therefore, we will replace  $a$  by  $\alpha$  in all our equations.

- The loop structure is identical for the shear condition and equation 4.21 of section 4.2.2 can be used straightforwardly to obtain these forces for the particle  $m$ . The derivative of this condition with respect to the particle position  $P_m$  is

$$\frac{\partial C}{\partial P_m} = \alpha \left( \frac{\partial W_u}{\partial P_m} W_v + \frac{\partial W_v}{\partial P_m} W_u \right). \quad (\text{Equation 6.3})$$

- For the bend condition, the loop must process a pair of adjacent triangles each iteration and the derivative we seek is

$$\frac{\partial C}{\partial P_m} = \cos \theta \frac{\partial \sin \theta}{\partial P_m} - \sin \theta \frac{\partial \cos \theta}{\partial P_m}. \quad (\text{Equation 6.4})$$

- We then define two methods to compute the terms  $W_u$  and  $W_v$ . We expand the equation given in Chapter 4 further:

$$(W_u \ W_v) = (\Delta x_1 \ \Delta x_2) \begin{pmatrix} \Delta u_1 & \Delta u_2 \\ \Delta v_1 & \Delta v_2 \end{pmatrix}^{-1}$$



$$\Leftrightarrow (W_u \ W_v) = (\Delta x_1 \ \Delta x_2) \begin{pmatrix} \frac{\Delta v_2}{|\det|} & -\frac{\Delta u_2}{|\det|} \\ -\frac{\Delta v_1}{|\det|} & \frac{\Delta u_1}{|\det|} \end{pmatrix} \text{ by Cramer's rule.}$$

(Equation 6.5)

The denominator  $|\det| = \Delta u_1 \Delta v_2 - \Delta v_1 \Delta u_2$  and because  $a = (\Delta u_1 \Delta v_2 - \Delta v_1 \Delta u_2)/2$  then  $|\det|=2a$ . So, our two new methods compute  $W_u = (\Delta x_1 \Delta v_2 - \Delta x_2 \Delta v_1)/2a$  and  $W_v = (-\Delta x_1 \Delta u_2 + \Delta x_2 \Delta u_1)/2a$ .

For the definitions of the deeper levels of the method hierarchy, please refer to appendices. [\[APPENDIX A\]](#)

Every time the forces of a condition are calculated for a particle, they are added to the total forces applied on that particle for the numerical solver to use during a time step.

- For the moment, both the components computing the external forces and the constraints enforcement mechanisms are from the same as in the first simulator.
- The components that compute  $df/dp$  and  $df/dv$  have a loop structure similar to the one above computing the internal forces. For the particles involved in the same condition function over a triangle or a pair of triangles, we compute all their terms  $df[i]/dp[j]$  also identified in the reference and in Chapter 4 as  $K_{ij}$ . If we took the stretch condition for example, three particles are involved for each condition over a triangle, so the indices  $i$  and  $j$  each have three values and nine terms  $K_{ij}$  need to be computed. The equation 4.26 in Chapter 4

$$K_{ij} = \frac{\partial f_i}{\partial x_j} = -k \left( \frac{\partial C(x)}{\partial x_i} \frac{\partial C(x)^T}{\partial x_j} + \frac{\partial^2 C(x)}{\partial x_i \partial x_j} C(x) \right)$$

requires the implementation of methods to compute the second derivative term  $\frac{\partial^2 C(\mathbf{x})}{\partial x_i \partial x_j}$

for each condition. We leave the definition of these methods also to the Appendices.

#### [APPENDIX A]

The derivative of forces applied on a particle with respect to its velocity are null since the condition functions do not depend on the particle velocity in the formulation of this model; consequently, the terms of  $df/dv$  are all null matrices and need not to be computed further. All the  $n \times n$   $K_{ij}$  terms that were computed by this component only come into consideration for the implicit numerical method, not for an explicit one. We stated previously that we decided to leave the bend condition among the three conditions to remain in an explicit implementation instead of including it for the implicit version. Therefore, at this step, we halted implementation of methods for this condition at the second derivative term, making it simply return a null matrix, and deactivating the methods upward in the hierarchy so that the terms  $K_{ij}$  originating from this bend condition are also null matrices.

- We kept the same viscous damping formulation from the first simulator instead of damping the forces from each of the three conditions independently like suggested in Chapter 4 for this second model. Our decision not to implement the three distinct damping methods respective to each condition is based on the observation depicted in the next chapter that the implicit integration alone already adds much stability to this model, but, the computation cost involved reduces the real-time performance of the model greatly. Therefore, adding three distinct damping methods will not necessarily add any immediate simulation quality or performance gain.

- For the implicit integration version of this numerical solver component, since  $df/dv=0$  and because we kept the previous viscous damping method, what remains to be solved for each particle is

$$(I - h^2 M^{-1}(\partial f/\partial x)) \times \Delta v = h M^{-1} (f^t + h(\partial f/\partial x)v^t),$$

and that is exactly equation 4.35. Then, let us think of it in the form

$$(\text{matrix A}) \times \Delta v = (\text{vector B}),$$

then we can solve this system  $Ax=B$  with  $x= A^{-1}B$ . So, the component builds a loop treating one particle per iteration. We observe the term  $\partial f/\partial x$  on both sides of the equation, it would therefore be appropriate to start by obtaining this term for each particle. We assume that the term  $\partial f/\partial x$  corresponding for a particle  $k$  is the sum of all the three by three matrices  $K_{ij}$  or  $df[i]/dp[j]$  where  $j=k$  i.e. all the force derivatives with respect to the position of particle  $k$ . The remaining terms of matrix  $A$  are also known so let us go to the right-hand side of the system to solve. On the right side of the equality, every term is straightforwardly obtained apart from the total forces applied on the particle, but we have obtained them previously for each particle via the two components calculating the external and internal forces. The system is ready to be solved and the change in velocity for each particle can be obtained. After which, computing the updated velocity and position for each particle is a trivial task.

- We do not include the adaptive time step option described in the reference and the remaining three components following the numerical integration step are the same as in the first simulator and everything is ready to be rendered.

## **Chapter 7: Results**

### ***7.1. Comparing the two simulators***

In this first section of the seventh chapter, we present visual results of the simulations, some overhead analyses, and a comparison of the two simulators on different levels. We start by presenting the visual results obtained from a set of tests performed on the two simulators.

#### **7.1.1. Some Test Results**

The parameter values shown in the table 7.1 below and the components active in table 7.2 are the default initial setting for all tests over simulator number one. Each test can change one or few of these values but will in general keep the remaining default values.

The default starting state of the cloth will be positioned as shown in Figure 7.1; the chosen initial positioning is just plainly flat and parallel to the XY plane.

<u>Parameter</u>	<u>Value</u>
number of particles	64
time step size	0.1s
$k_{\text{stretch}} = k_{\text{structural}}$	0.005
$k_{\text{shear}}$	0.005
$k_{\text{flexion}}$	0.005
damping coefficient	0.003608
particle mass	0.000651
gravitational acceleration factor	9.8
gravity active	true
wind active	true
Structural spring color	blue
Shear spring color	green
Flexion spring color	yellow
maximum overstretch threshold %	0.15

*Table 7.1. The default parameter values for the first simulator.*

<u>Component or sub-component</u>	<u>active value</u>
Node array initialization	true
Construction of springs	true
Construction of surface triangles	true
Internal forces of cloth springs	true
External forces	true
Positional constraints	true
Damping	true
Numerical Integration	true
Post-step corrective measures	true
Collision detection and response	false
Rendering	true

*Table 7.2. The active components for the first simulator.*

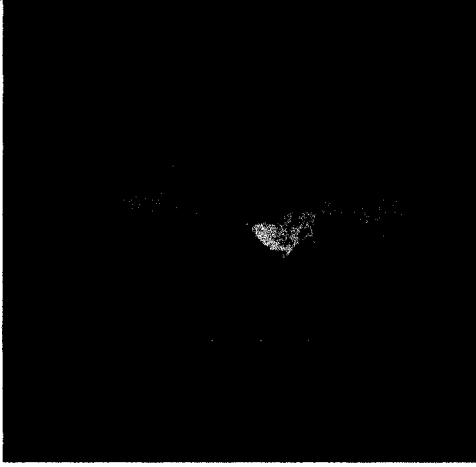
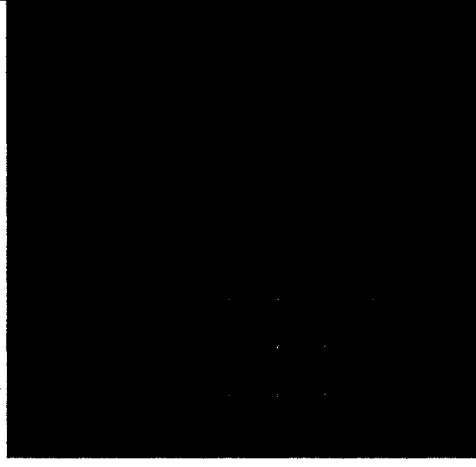
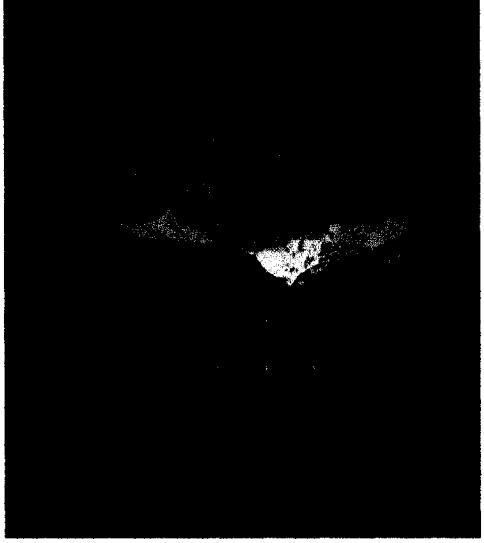
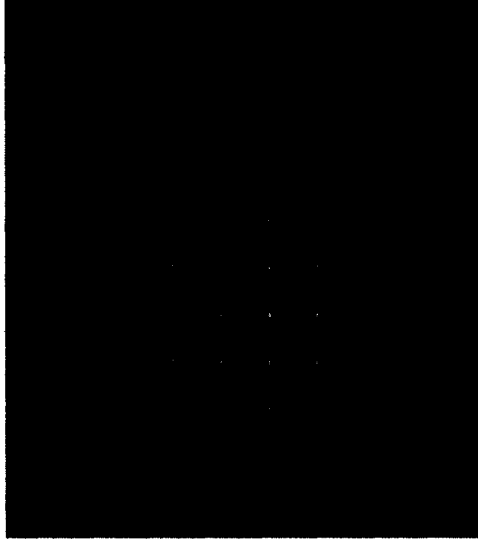
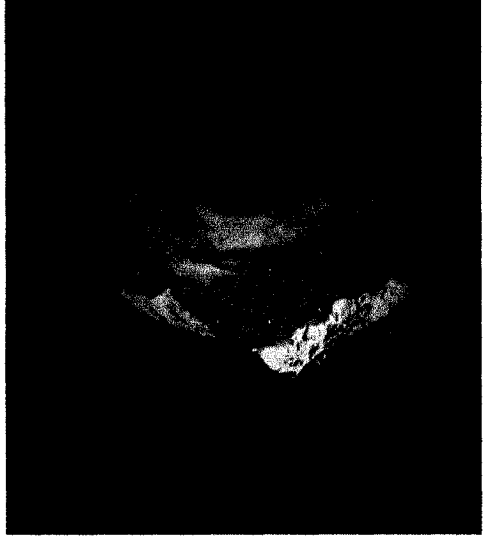
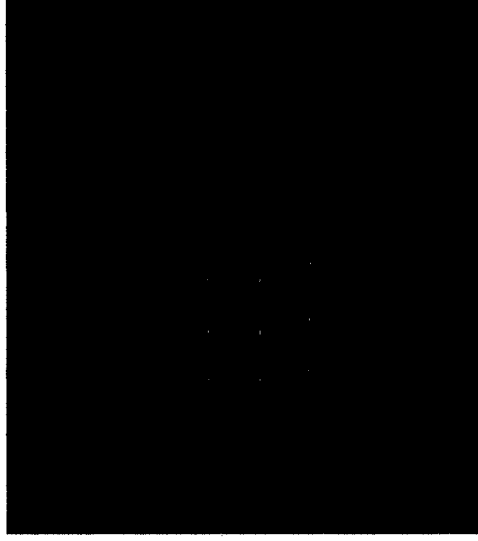
Time value	Textured cloth surface displayed	The system of springs
0 sec		


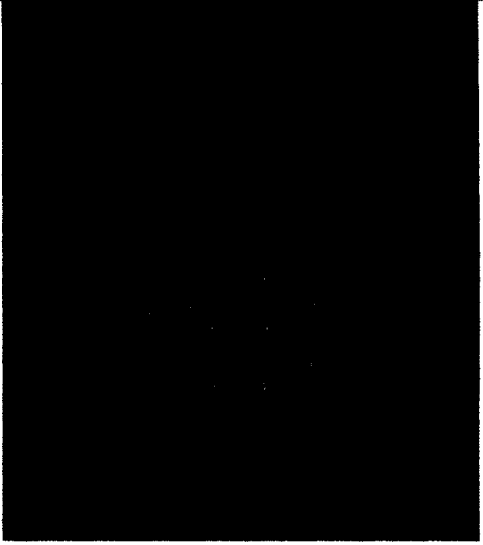

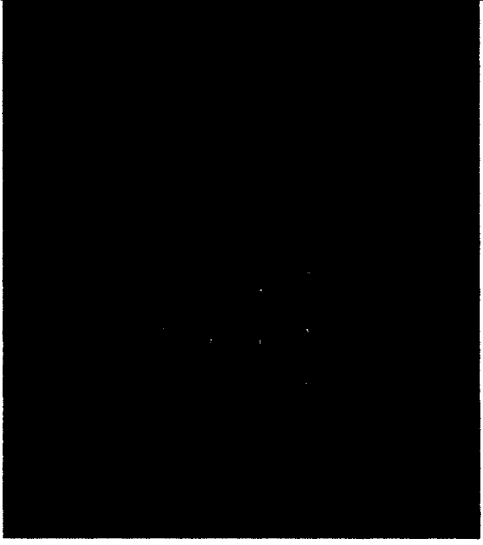
Figure 7.1. The cloth in its initial state.

- Simulator1, Test1: No damping and no post-step modifications

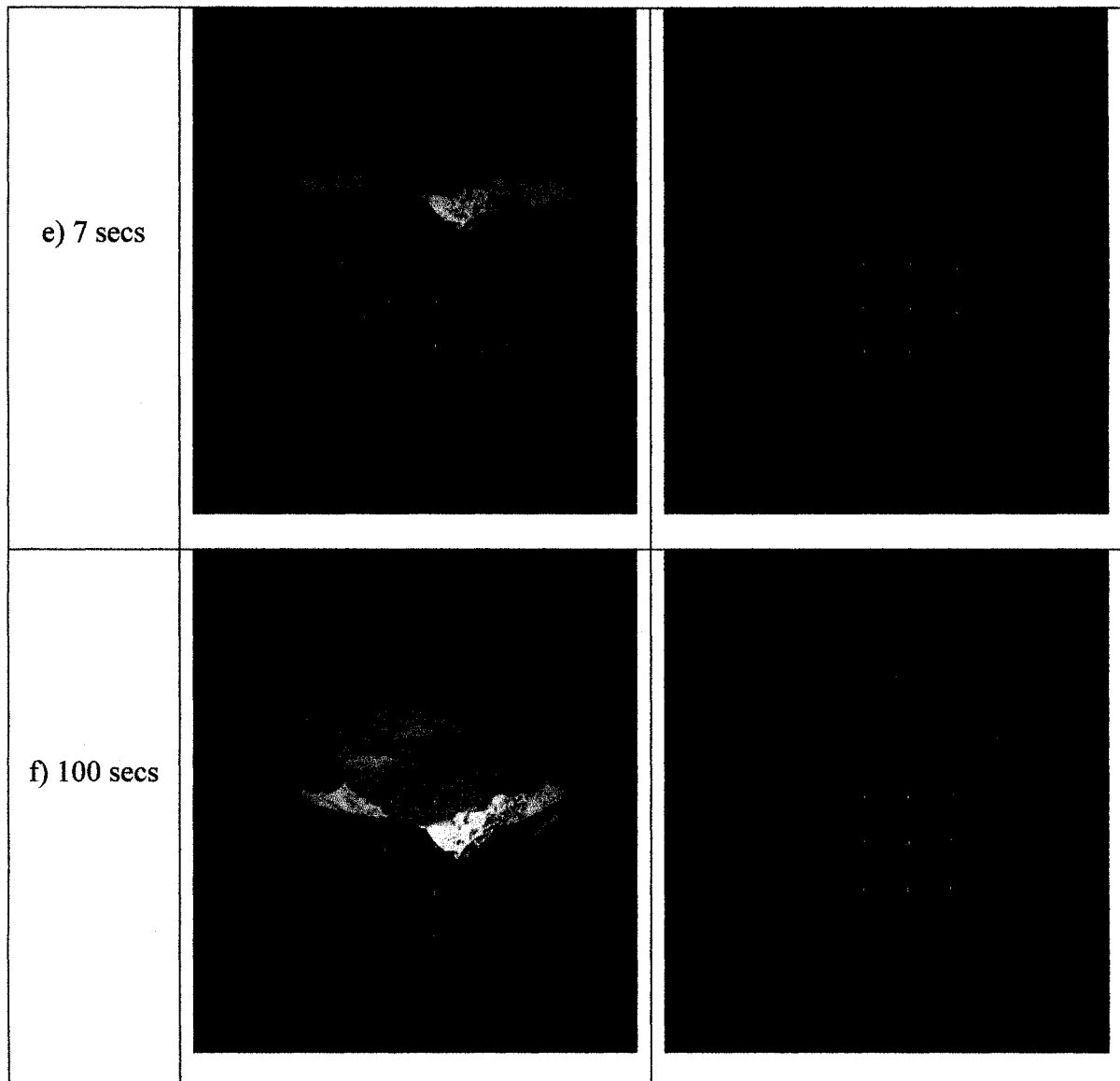
The first test will simulate the piece of cloth hanging from both its top left and top right corners without any active wind. The objective is to visualize how the model would fare in terms of its stretching behaviors versus the pulling of gravity. We start with absolutely no damping nor post-step correction so we can observe the internal forces in action on their own. Figures 7.2.a to 7.2.f illustrate the evolution of the system at six different time values. The cloth appears to overstretch with the pull of gravity; then, bounces up and down but never higher than its starting state. The system still remains consistent as far as one hundred seconds, but there are oscillations within the internal surface of the cloth forming up gradually, making the particles move back and forth by small amounts in all directions while remaining coplanar. The oscillations are small but they are disturbingly annoying, giving the cloth surface a slimy look like a glutinous substance being shaken. Surprisingly, despite all the oscillations happening on the cloth surface, there is still no

chaos after 200 seconds, the piece of cloth remains coherent if a single instant or picture is taken of it, but the motion does not resemble real cloth at all. The bouncing up and down is the result of the structural springs oscillating without damping interventions and the glutinous-like motion results from the oscillations of the undamped shearing springs.

Time value	Textured cloth surface displayed	The system of springs
a) 1 sec		
b) 2 secs		

c) 3 secs		
d) 5 secs		

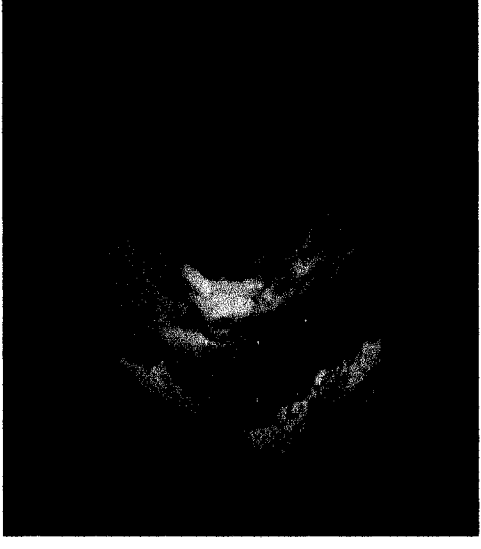
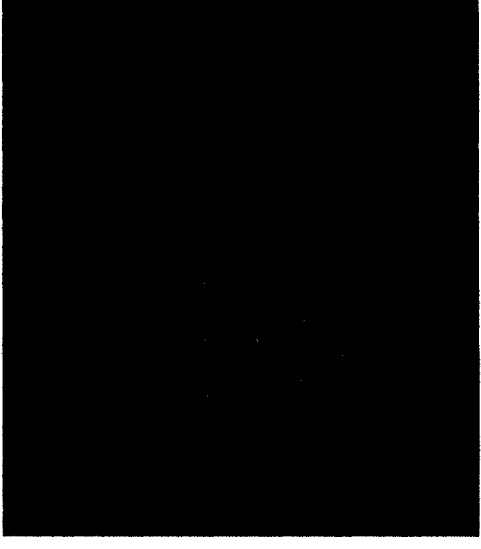
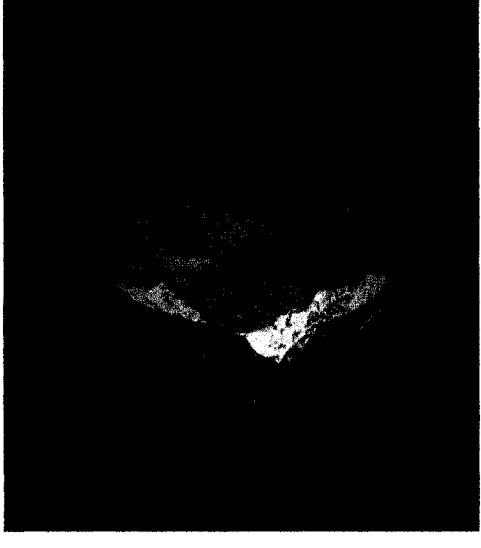
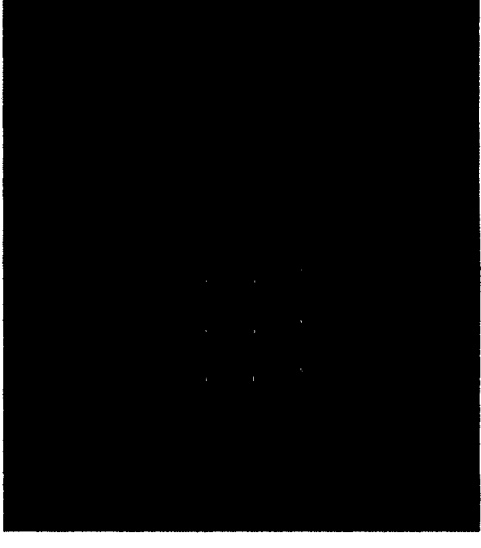


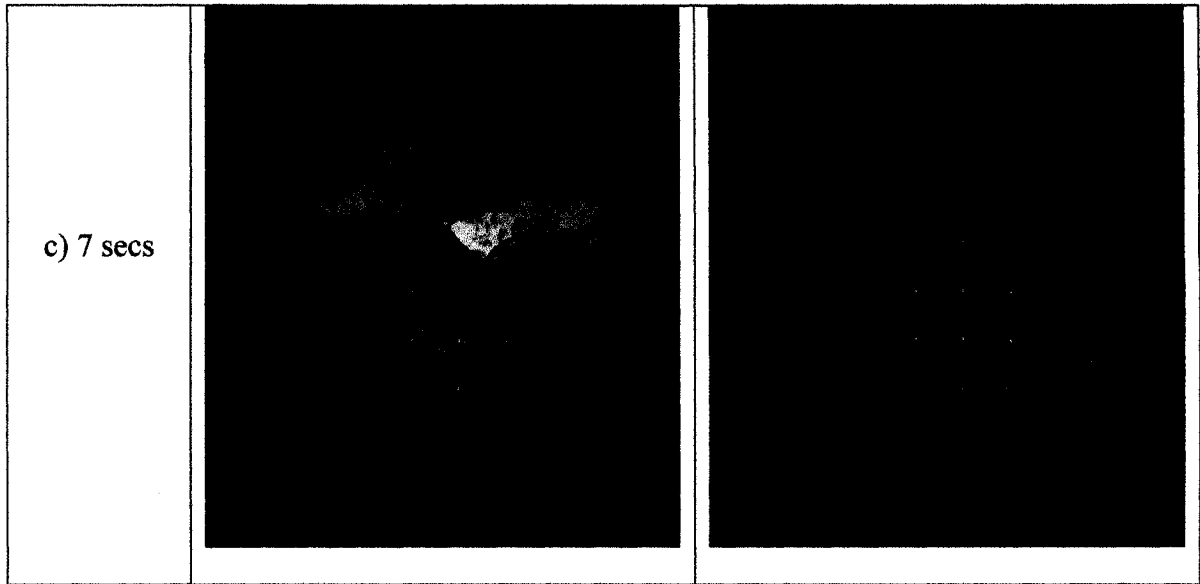


*Figure 7.2. The first set of captured frames for test#1, simulator #1, at times: a) 1, b) 2, c) 3, d) 5, e) 7, f) 100 seconds.*

We now augment the stiffness coefficient for the stretching springs to the value 0.010 and what we observe with Figures 7.3.a to 7.3.c is that the evolution of the system is almost identical until the fifth second where the stronger structural springs have pulled the cloth up more than with the previous stiffness value. Actually, if we looked very carefully and compared the figures at time three seconds from both coefficient values, we could see that the stronger stretch resistance value let the cloth remain at a slightly higher vertical



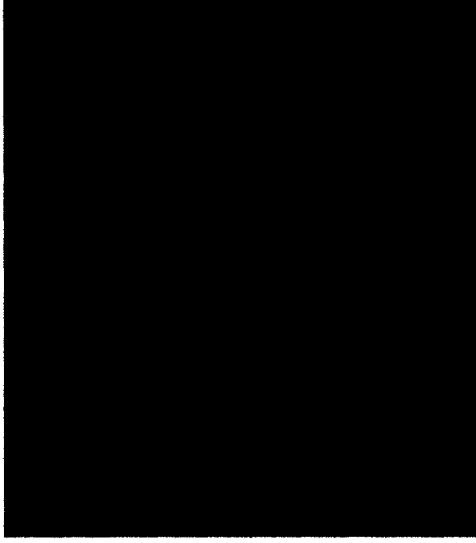

position, but the amount is not obvious. The cloth bounces up to the exact same maximum vertical position value but the moment that happens is earlier by approximately half a second and, at seven seconds, it has already dropped down vertically by a minuscule amount with this new structural stiffness value.

Time value	Textured cloth surface displayed	The system of springs
a) 3 secs		
b) 5 secs		



*Figure 7.3. The second set of captured frames for test#1, simulator #1, at times: a) 3, b) 5, c) 7 seconds.*

We can keep on increasing the structural spring's coefficient and observe less vertical stretching and an earlier bouncing up of the cloth as the adjusted value increases while the remaining behaviors of the cloth remain similar; But, as we increase that value, the slimy movements also increase, and once the coefficient reaches a value of 0.045, the system can no longer maintain its stability after twelve seconds. The cloth particles oscillate too much; the whole cloth looks like it is shaking and about to explode. The system starts diverging in between the thirteenth and fourteenth second of the simulation run.

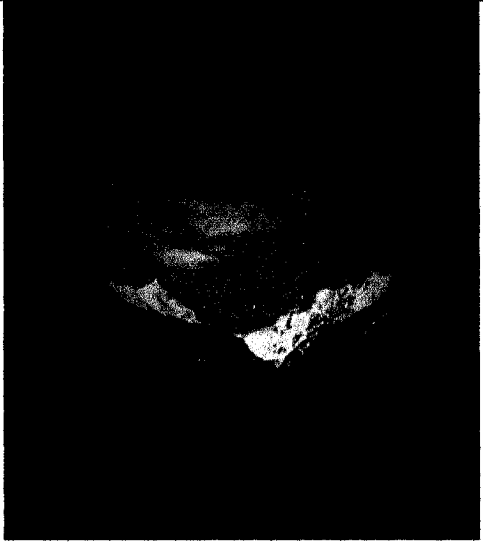

Time value	Textured cloth surface displayed	The system of springs
a) 13 secs		
b) 14 secs		

*Figure 7.4. A third set of captured frames showing instances of instability at times a) 13 and b) 14 seconds.*

- Simulator1, Test2: Damping active but not post-step modifications

We now reactivate the damping component. The cloth still reaches an overstretched state at time shown in figure 7.5 but bouncing has been subdued. The cloth just stops at its maximum overstretched position after slowly stretching downward, its velocity

seemingly absorbed at that point exactly as if the stretch resistance forces matched the force of gravity. Since we did not change the stiffness of the springs, the reduction in over elongation is caused by viscous damping reducing the effect of gravity on the cloth. Although it is possibly visually pleasing to see the evolution of the cloth drop at such speed for our analysis, we are reminded that David Baraff and Andrew Witkin specified in their work that damping of internal forces must distinctively damp only their corresponding force mechanisms, in this case the structural springs, and nothing else; because in reality, it does not take cloth this much time to reach its lowest position.

Time value	Textured cloth surface displayed	The system of springs
26 secs		

*Figure 7.5. A captured frame for test#1, simulator #1, after twenty-six seconds with damping.*

Let us now examine the value of the damping coefficient and test the outcome of modifying this value.

- We start by setting the damping coefficient to a value of 0.0001. The cloth started dropping then bouncing up and down but the amplitude decreased bounce after bounce. It reached a lower position before the first bounce with this weaker

damping force. After the sixth cycle of bouncing around forty seconds, the amplitude was reduced to a very small amount, and after one minute and fifteen seconds, about eleven cycles took place and the oscillations were visually subdued.

- Now, if the coefficient were doubled to 0.0002, the number of cycles of bouncing up and down were reduced to four, the amplitude decreased more quickly, and the overstretch is also less than with the previous coefficient value.
- The next damping coefficient value tried is increased to 0.001 and the cloth does not seem to go back up. Our simulated piece of fabric stops at its lowest position reached after seven seconds. The maximum elongation the cloth object drops to looks the same as the one when the default value was used. It is nearly impossible to visually observe any oscillations although technically speaking, some low-amplitude oscillations may have occurred. This will be explained shortly.

The default value we set for the damping coefficient 0.003608 is very close to the critical damping coefficient needed for this simulation system. The three values tested just previously produced an under-damped system thus, partially solving for the oscillation problem but not subduing it completely. We obtained the critical damping coefficient  $C_v$  for this system with the formula given in mechanical physics for a spring system


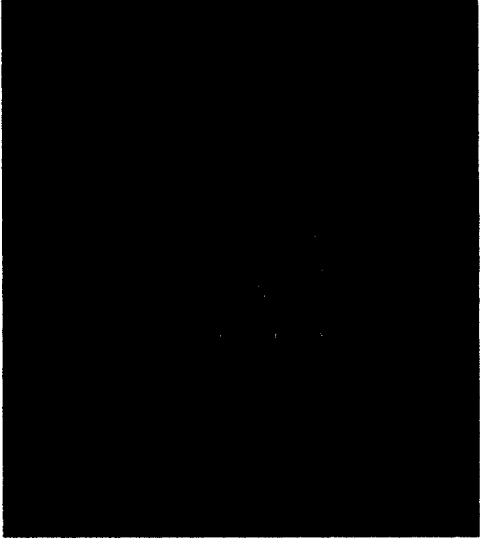
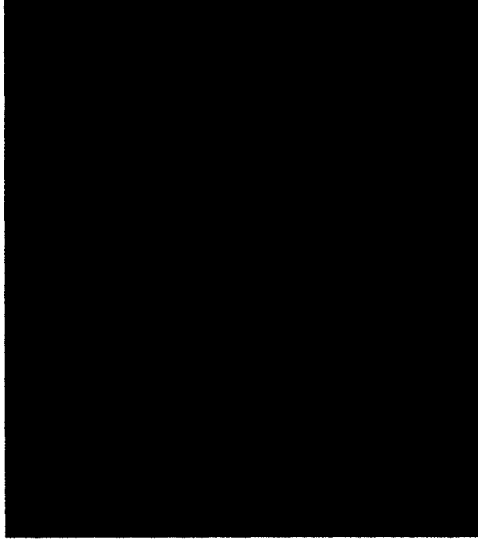
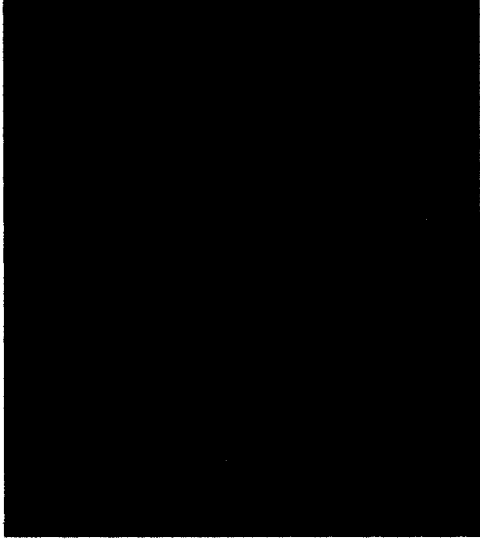
$$C_v = (4mk)^{1/2} \quad (\text{Equation 7.1})$$

where  $m$  is the mass, and  $k$  is the spring stiffness coefficient. For the value of  $k$ , we opted to let the maximum value of the stiffness coefficients of all three types of springs determine it. Though our default setting has all three stiffness coefficients equal, we

believe that taking the maximum value, which usually comes from the stretching property of fabric, can more easily help deal with the instability problem as a general solution.

- Simulator1, Test3: No damping but yes to post-step modifications

We now do the reverse option of the second test and again deactivate the damping method and instead utilize the post-integration component that limits the length of structural and shear springs to a threshold value. The result is not very appealing. Since there is no damping, the cloth started to drop during the first second similarly like in test number one. But soon after, we could observe the effect of our post-step position correction method holding the particles from dropping further. (Figure 7.6.a) The order in which particles were processed caused biases in the corrections of the particle positions and started causing positional artifacts. After eight seconds, the system started diverging and losing its coherency. (Figure 7.6.b) This shows two facts relating to the stability of the current simulator. First, the damping component can work without the post-step position correction method, but the latter, built as is, cannot work without damping. Secondly, it forces us to reconsider including inverse dynamics procedures for post-step position modifications, a step we first thought we could skip without meaningful harm to the system's stability.

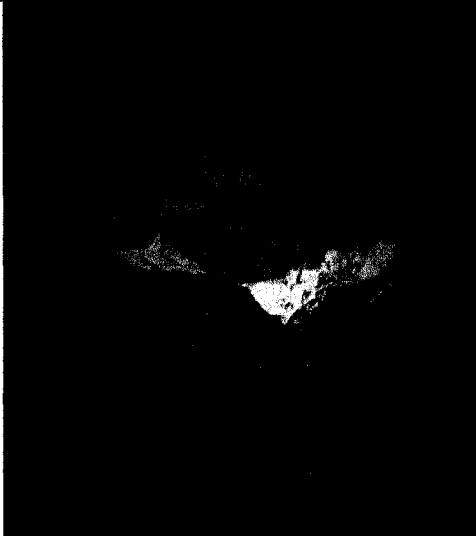
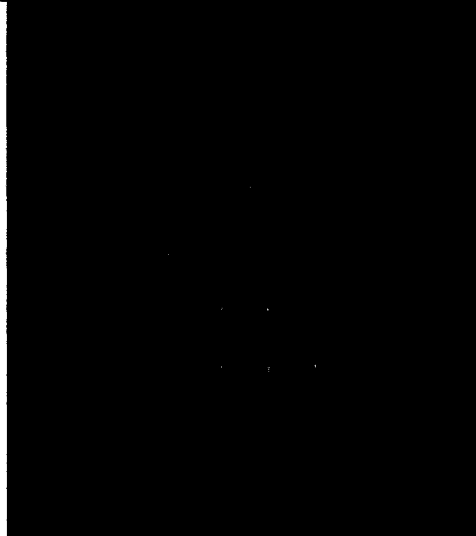
Time value	Textured cloth surface displayed	The system of springs
a) 3 secs		
b) 8 secs		

*Figure 7.6. The system is in trouble when post-step modifications are active but damping is not, at times a) 3, and b) 8 seconds.*

Let us perform more testing but now with the inverse dynamics procedures in place to adjust the particle velocities. Immediately, we can observe that constraint functions work more efficiently and limit the stretching rates of all springs without causing any instability. The cloth does not drop any lower than shown in Figure 7.7a. Without damping, the cloth bounces up and down as expected. But, differently from test number



one, the cloth does not bounce up by very much, and it eventually stops. The fact that the movement upward reaches a lower point is easily explained by the constraint method limiting the stretching rate and therefore also the forces of stretching springs pulling back on the following time step. The second observation that it now stops even after a long amount of time shows that without either damping or the stretching constraints, the struggle between the external force of gravity and the internal force of the structural springs, given the time step size used, the system acts just like a perpetual pendulum. Again, Figure 7.7.a and 7.7.b show the minimum and maximum vertical positions the cloth drops and bounces to. The amplitude of those vertical motions gradually decreases over time and the system stabilizes in its vertical motions yet does not stop entirely after sixty eight seconds. (Figure 7.7.c)

Time value	Textured cloth surface displayed	The system of springs
a) 2 secs		

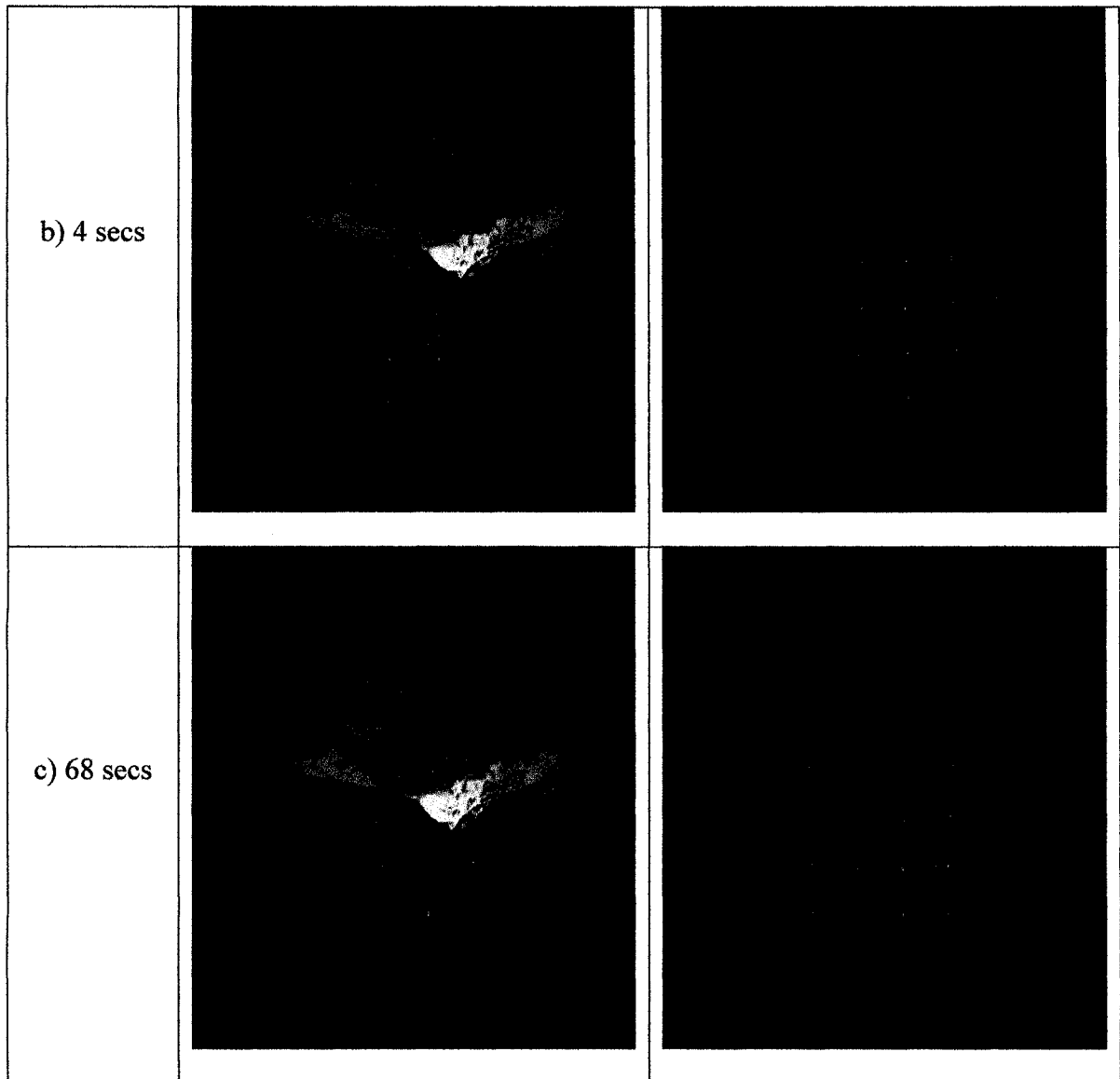


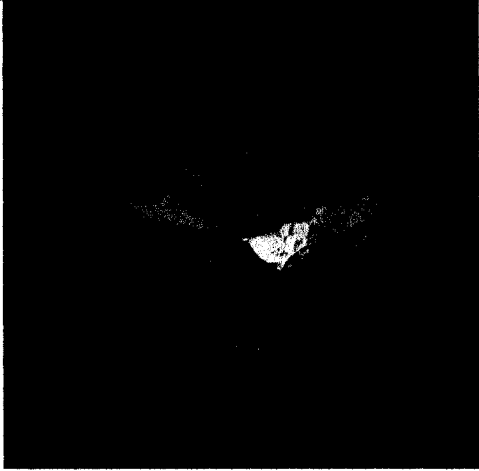
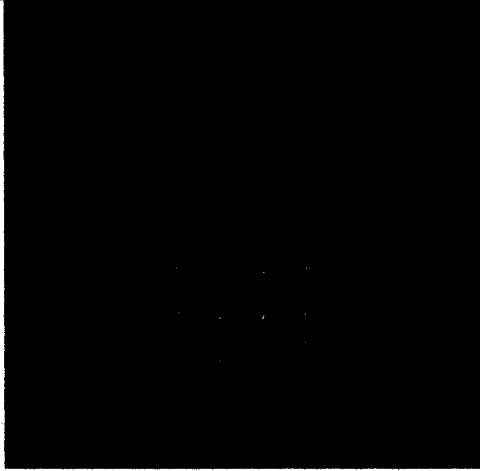
Figure 7.7. The system with inverse dynamics in place at times: a) 2, b) 4, c) 68 seconds.

From this point on, we will use the updated version of the post-step modification component that includes the inverse dynamics steps for the constraint method.

- Simulator1, Test4: Both damping active and post-step modifications active

We now make both the previously discussed components active simultaneously. The cloth drops and stabilizes with an acceptable elongation shown in Figure 7.8 after about

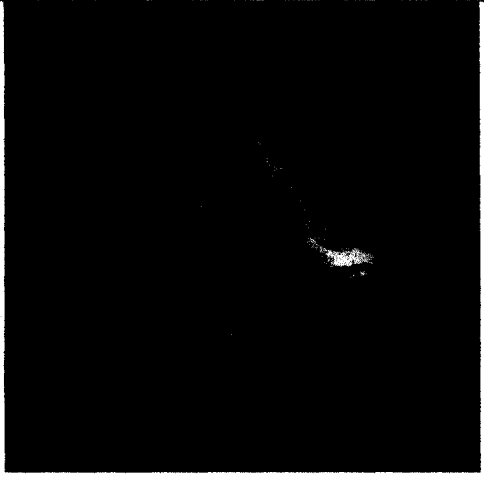
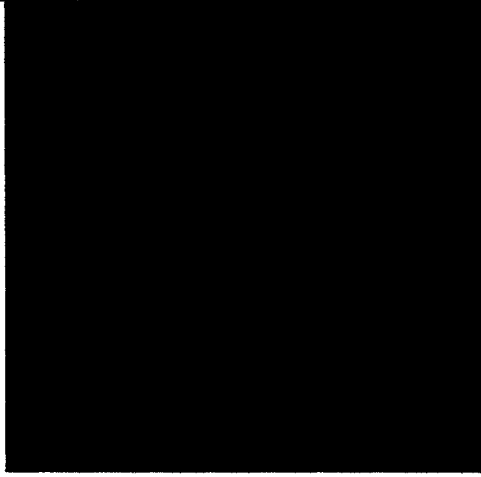
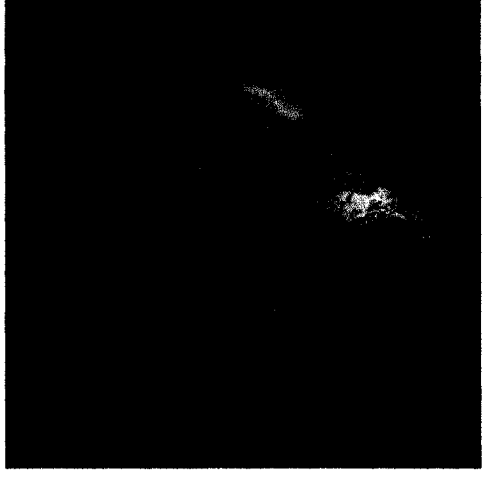

seven seconds of the simulation. If compared to the previous test where no damping was used, the cloth drops too slowly here and the drop shown by the previous test would be more appropriate in terms of speed, but the elimination of oscillating motions is neat. In all, it gives us an incentive to reconsider maybe utilizing a different definition for damping or damp each mechanism independently instead of formulating a damping affecting all forces in play, a requirement specified in Baraff & Witkin's work.

Time value	Textured cloth surface displayed	The system of springs
7 secs		

*Figure 7.8. The system with both post-step modifications and damping functions active.*

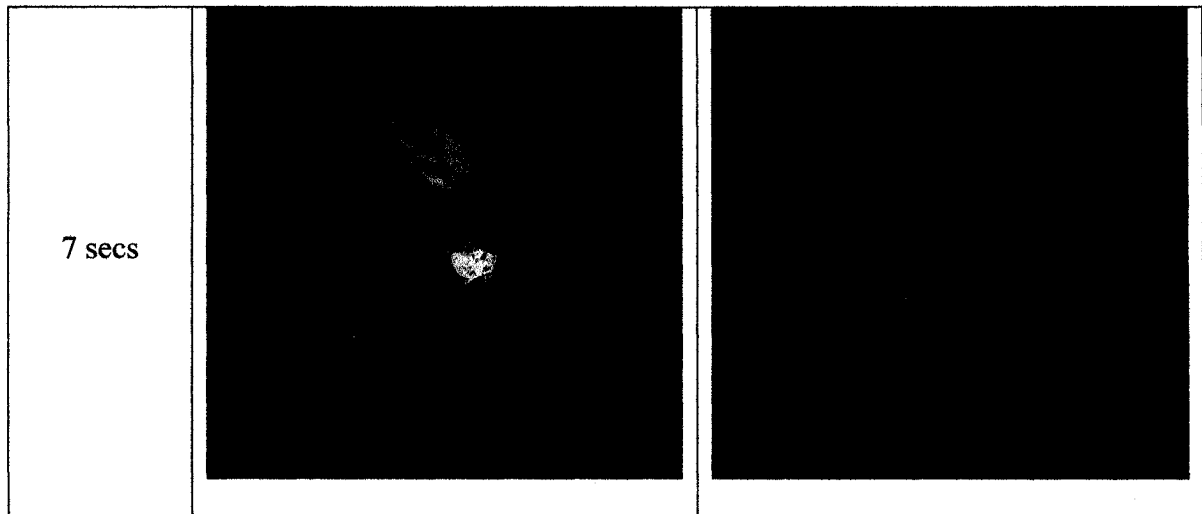
- Simulator1, Test5: Wind

To the previous setting we now add wind as an additional external force. Two frames with one second difference are shown in Figures 7.9.a and 7.9.b.

Time value	Textured cloth surface displayed	The system of springs
a) 6 secs		
b) 7 secs		

*Figure 7.9. Two frames captured when wind is added at times: a) 6 and b) 7 seconds.*


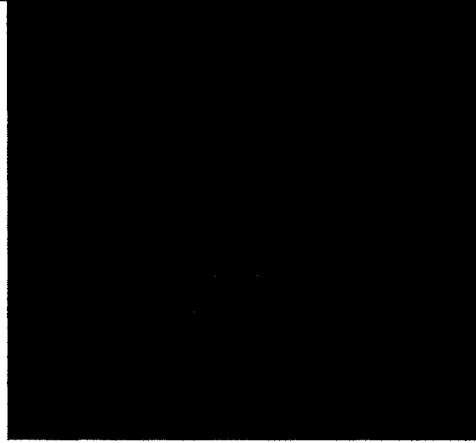




Figures 7.9.b and 7.10 both show the same state for the cloth at seven seconds but with two different angles. The system behaves correctly and remains stable. The display rate, the motions of the cloth as well as its elastic behaviors are coherent. The animation of the cloth can simulate a flag flapping in the wind with enough accurateness.



*Figure 7.10. The cloth viewed from different angles.*

- Simulator1, Test6: Shear and Bending properties

The tests performed so far have not isolated the shearing and bending behaviors of our cloth model. It is not absolutely necessary to disable the two other types of springs to test one type, say the flexion. A proper setting can clearly show the targeted type of springs in action. To test the shear springs, we disable all external forces including gravity, we also disable damping and the post-step constraint mechanism, we do not fix any nodes, and we position the cloth like in Figure 7.11.a where every structural spring is kept at natural length and where the shearing angle is set to be about thirty degrees.

Time value	Textured cloth surface displayed	The system of springs
a) 0 sec		
b) 1 sec		
c) 2 secs		

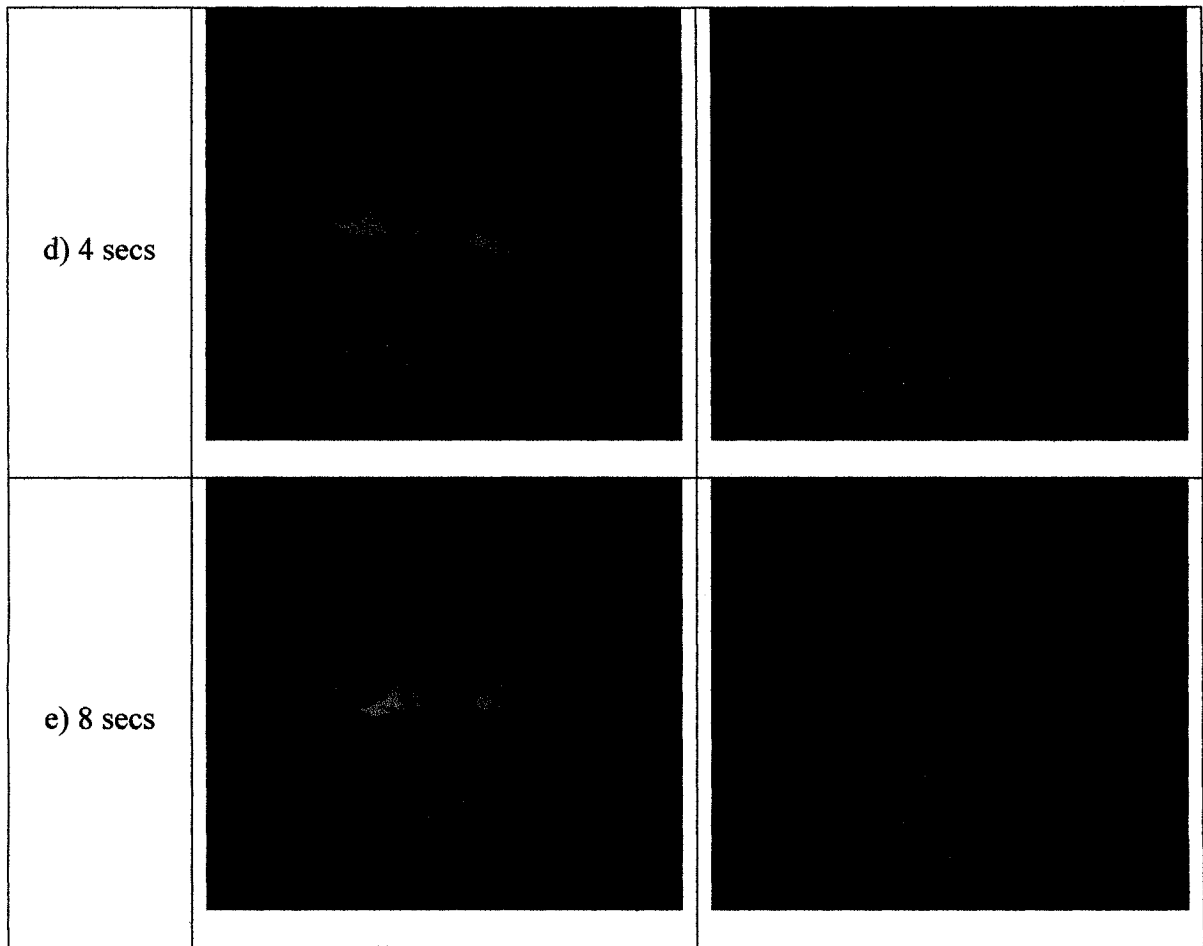
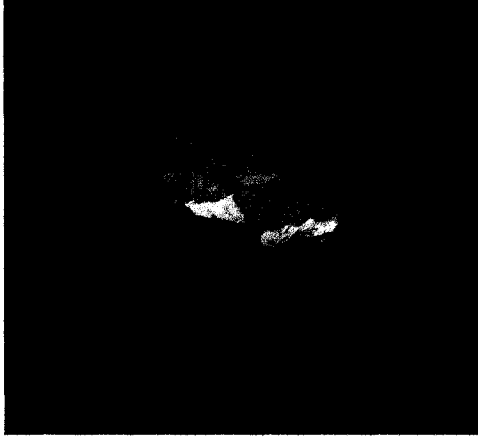
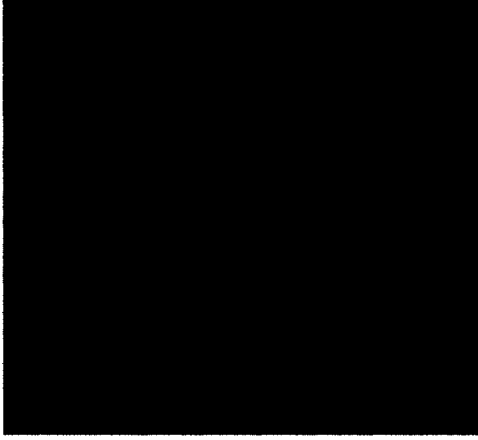
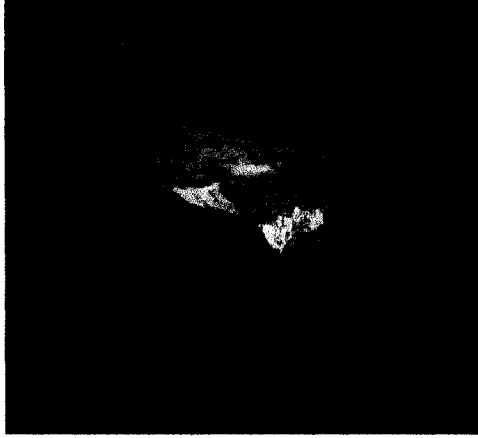
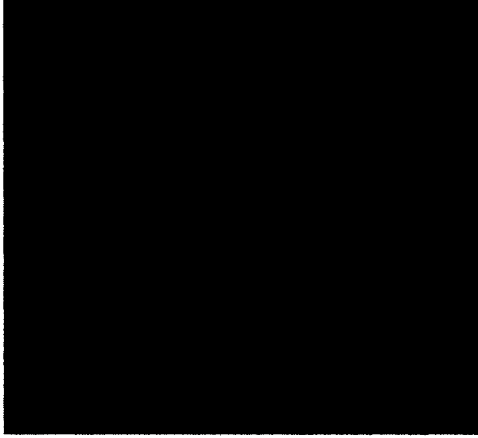
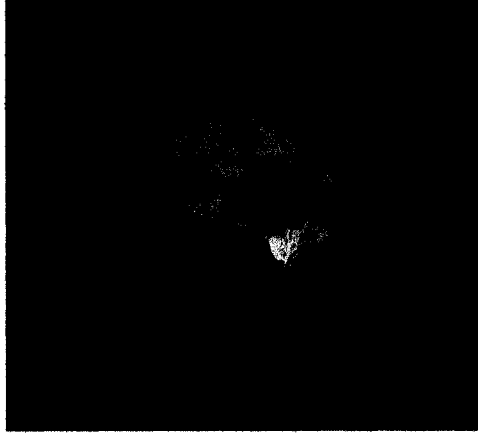
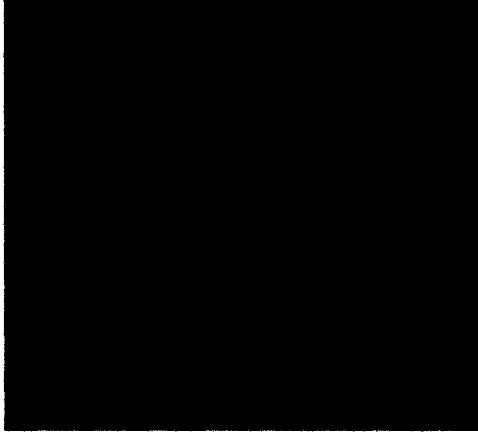


Figure 7.11. The shear springs in action: a) 0, b) 1, c) 2, d) 4, e) 8 seconds.

The results conform to what was expected at this point. The elongated shear springs pulled the two top left and bottom right extremities towards the center of the cloth object while the two other extremities were pushed away from the center because their springs were shorter than the natural length of shear springs. And, since no damping or position corrections were applied, springs were elongated at times and the cloth object kept contracting upon itself then expanding continuously like a living jellyfish-like life form.

To visualize the effects of flexion springs, we will keep the same settings and also start with structural and shear springs at their natural lengths. The initial positioning will be

like shown in Figure 7.12.a below where our camera will rotate about the Y-axis almost 45 degrees to look at the cloth object a bit from the side.

Time value	Textured cloth surface displayed	The system of springs
a) 0 sec		
b) 1 sec		
c) 2 secs		



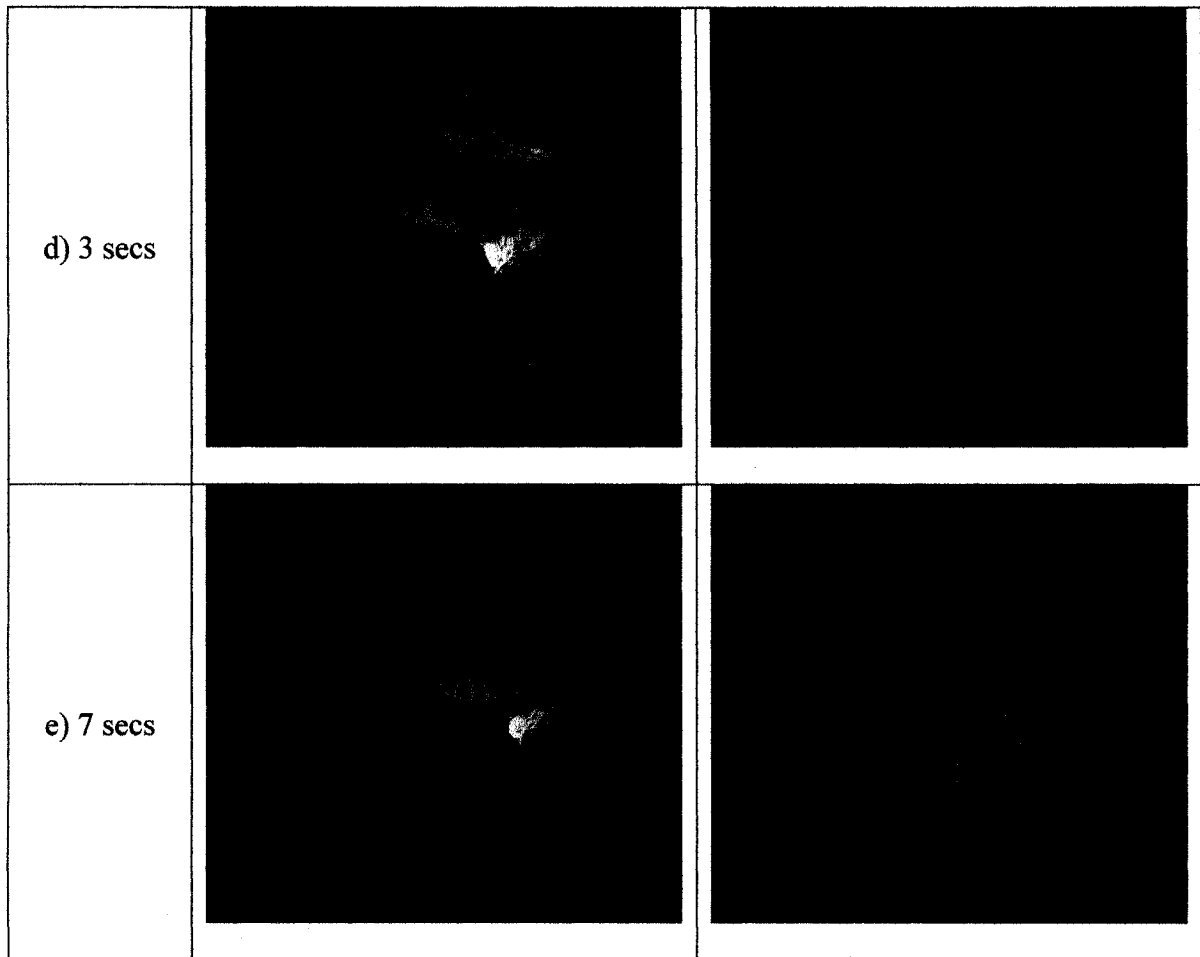
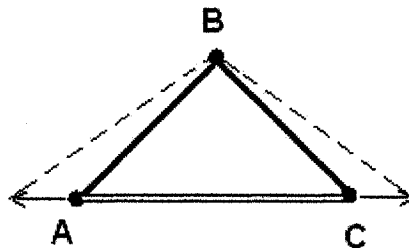


Figure 7.12. The flexion springs in action: a) 0, b) 1, c) 2, d) 3, e) 7 seconds.

We observe that the flexion springs in the color yellow truly act as invisible mechanisms as their actions have an effect on the resulting shape of the cloth mesh; yet, they do not make up the mesh itself but it is the structural springs that happen to coincide with the edges embodying the mesh that give it shape. The flexion springs defined so are not of much use when the particles are coplanar. Even if their definition in theory can counter in-plane elongations or compressions, the structural springs, being much stiffer, are likely to intervene and precede the flexion springs in simulating the in-plane behaviors of fabrics.

There is an important observation that can be made out of the figure 7.13 below. Flexion springs alone do not simulate the bending resistance well. It is the combination of these springs and the two other types, but mainly structural ones, which allows proper modeling of the bending property. Given a state requiring the intervention of the flexion springs like depicted in Figure 7.13, all that the flexion spring located on the edge AC would do is to push the particles on nodes A and C such that the edge AC returns to its original length. Though that reaction brought the angle ABC towards realigning the three particles, they are not realigned yet, but the flexion springs can not help further on their own. The structural springs located on the edges AB and BC are elongated by the reaction of the flexion spring AC, thus, they will pull their particles closer on the next iteration. It then becomes clear that it is the actions of the flexion springs followed by the structural springs alternatively as such that will be able to realign the three particles.



*Figure 7.13. The limitation of flexion springs.*

We can conclude this first set of tests by confirming once again that without any form of damping and without post-step constraint enforcements, the cloth does not behave realistically, elongations beyond acceptable amounts and continuous oscillations prevent the correct modeling of the mechanical behaviors of fabric.

We now proceed to a second set of tests for the second simulator. The two following tables (Tables 7.3 and 7.4) contain the default parameter values and the listing of active components for the subsequent tests performed on the second simulator.

The tests we applied to the second simulator were similar to the tests applied to the first simulator and we describe them in the same order.

<u>Parameter</u>	<u>Value</u>
number of particles	64
time step size	0.1s
$k_{\text{stretch}}$	0.0050
$k_{\text{shear}}$	0.0001
$k_{\text{bend}}$	0.0001
viscous damping coefficient	0.001614
particle mass	0.000651
gravitational acceleration factor	9.8
gravity active	true
wind active	true
maximum overstretch threshold %	0.15

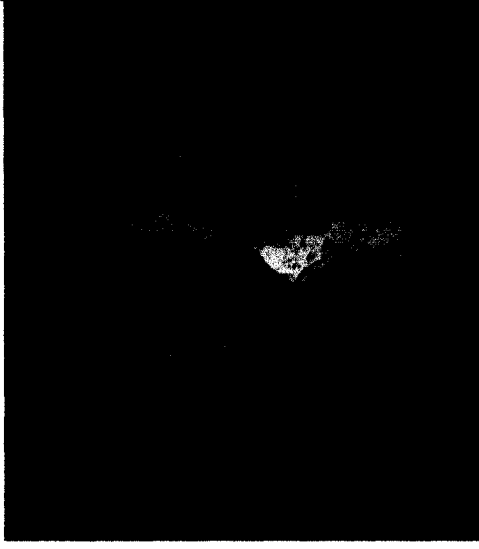

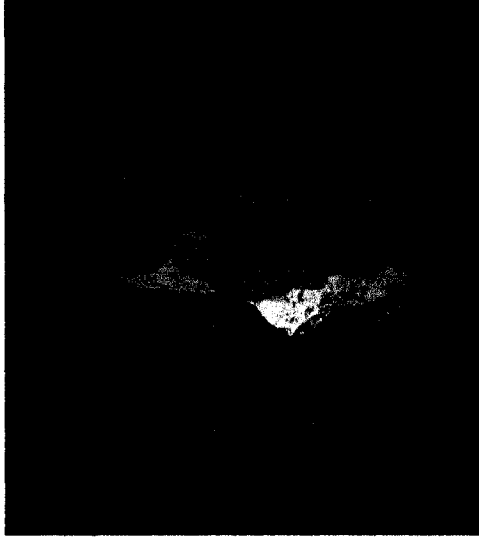
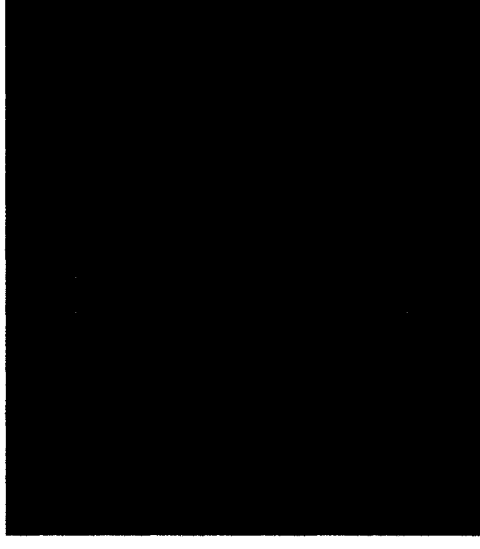
*Table 7.3. The default parameter values for the second simulator.*

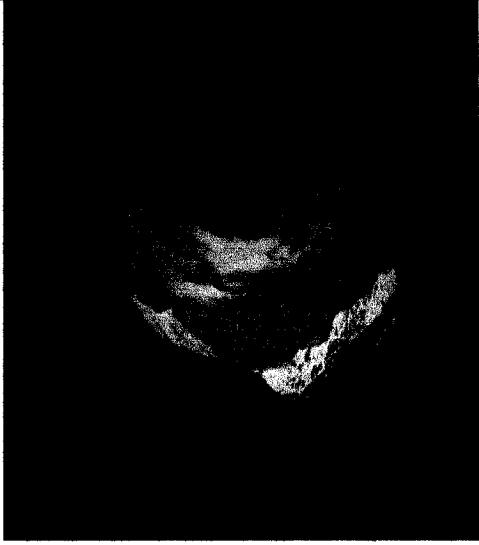
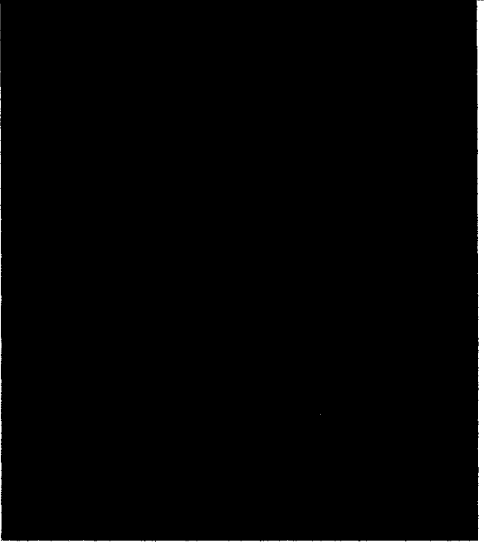
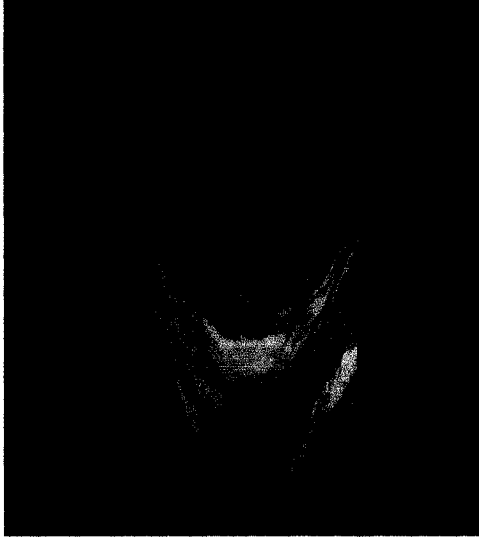
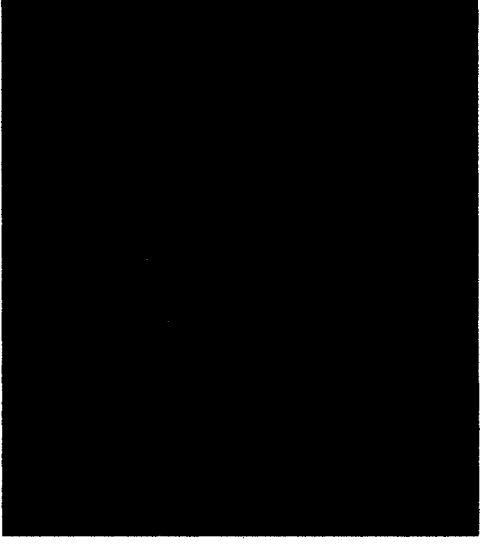
<u>Component or sub-component</u>	<u>active value</u>
Node array initialization	true
Construction of springs	false
Construction of surface triangles	true
Initialization of $df/dp$ and $df/dv$	false
Computations of $df/dp$ and $df/dv$	false
Internal forces from condition functions	true
External forces	true
Positional constraints	true
Viscous Damping	false
Explicit Numerical Integration	true
Implicit Numerical Integration	false
Post-step corrective measures	false
Collision detection and response	false
Rendering	true

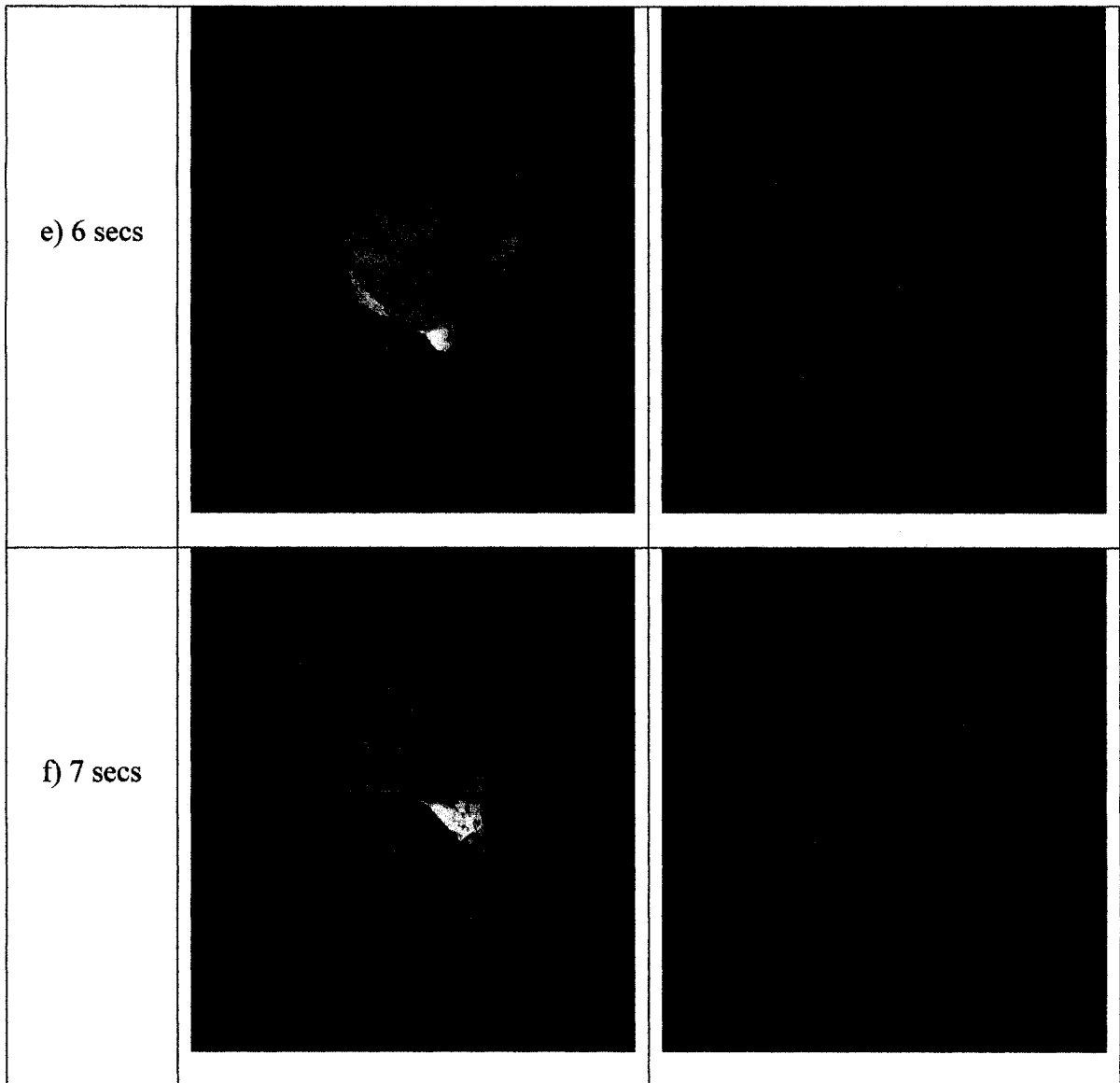
*Table 7.4. The active components for the second simulator.*

- Simulator2, Test1: Each condition under the explicit numerical method

We will test each of the three conditions under the Explicit Euler integration. Again, we start by only allowing the internal dynamics of the model to react against the external force of gravity and simulate the stretching behaviors of a piece of cloth hanging from its two top corners.

Time value	Textured cloth surface displayed	The triangles underneath
a) 0 sec		
b) 1 sec		

c) 2 secs		
d) 3 secs		



*Figure 7.14. The first set of frames captured for test#1, simulator #2, simulating the stretch condition.*


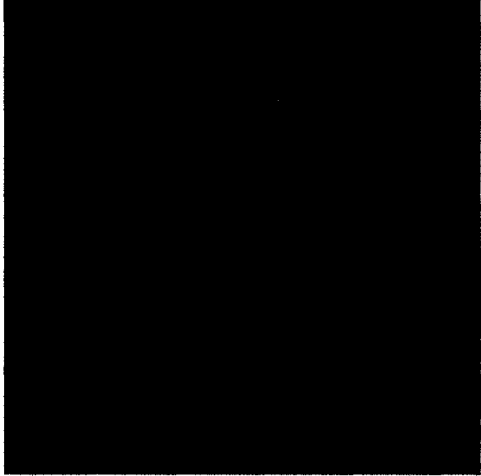
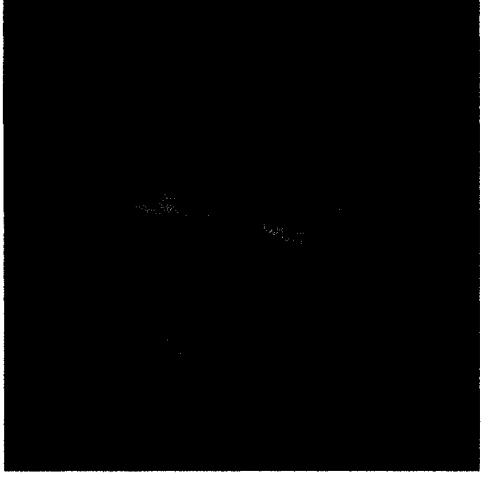
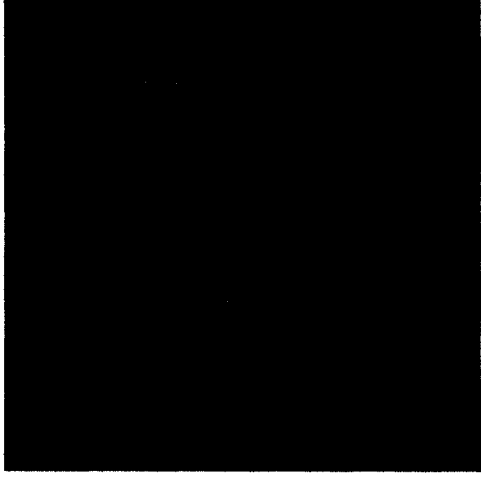
The results are partly similar to what was obtained with simulator one except for two observations. First, the cloth does not bounce back up as high. This could be explained by the fact that we also see the cloth stretch downwards more than in the first simulator, that is, the forces created by the stretch condition are not as strong. The remedy could have been simple: an increase in the stiffness coefficient of the stretch condition. Unfortunately, increasing  $k_{\text{stretch}}$  beyond the value 0.005 under the current explicit

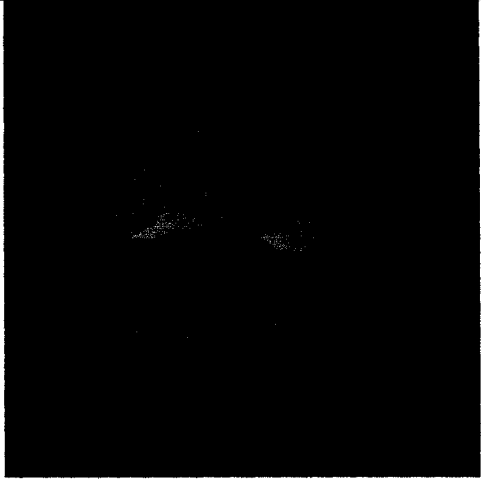
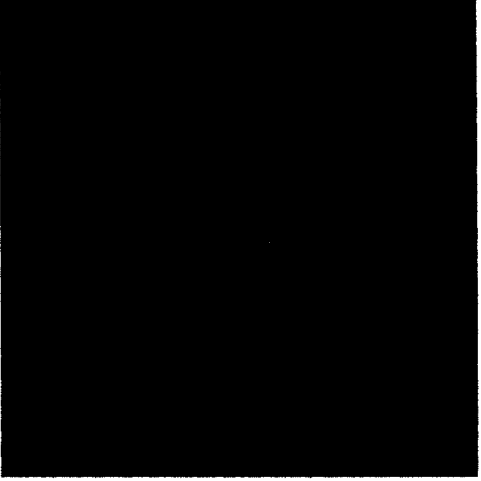
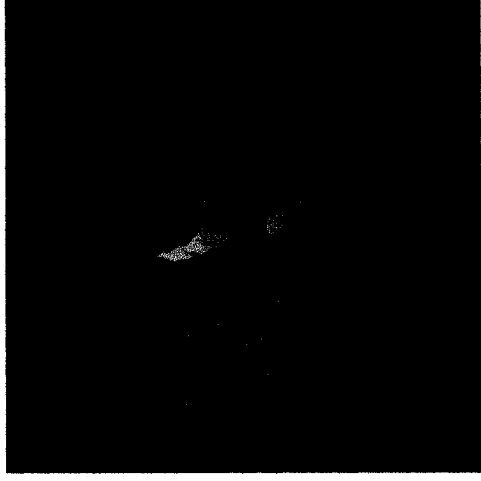
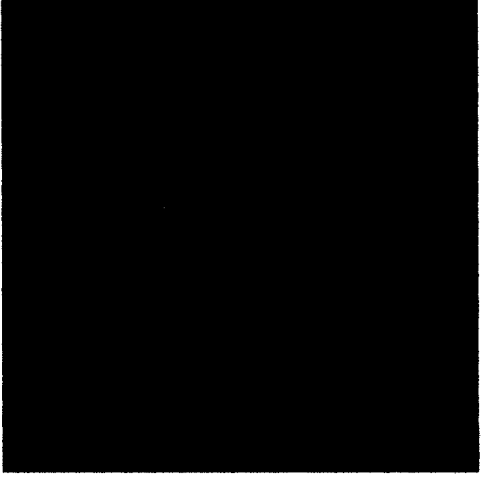
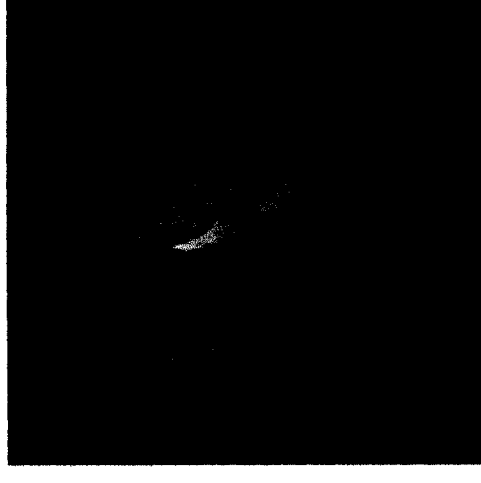
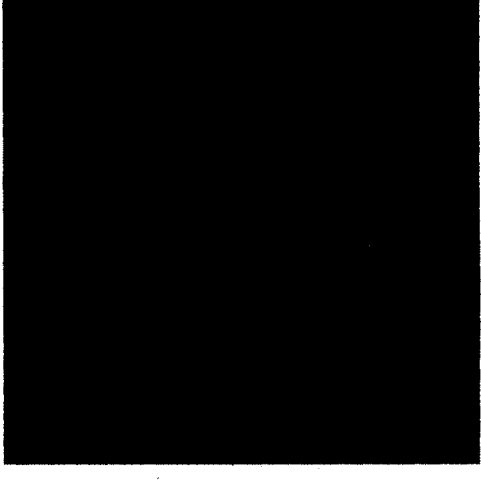
numerical method makes the system susceptible to divergence. Second, the bouncing motion upwards is not perfectly symmetric versus the vertical axis; there is a small bias to the right. A problem similar to this one will be observed under the bend condition also. These are issues that depend on how the triangle mesh is structured and how the formulation of conditions over triangles is handled with respect to the mesh structure. This problem will be discussed further in the following section.

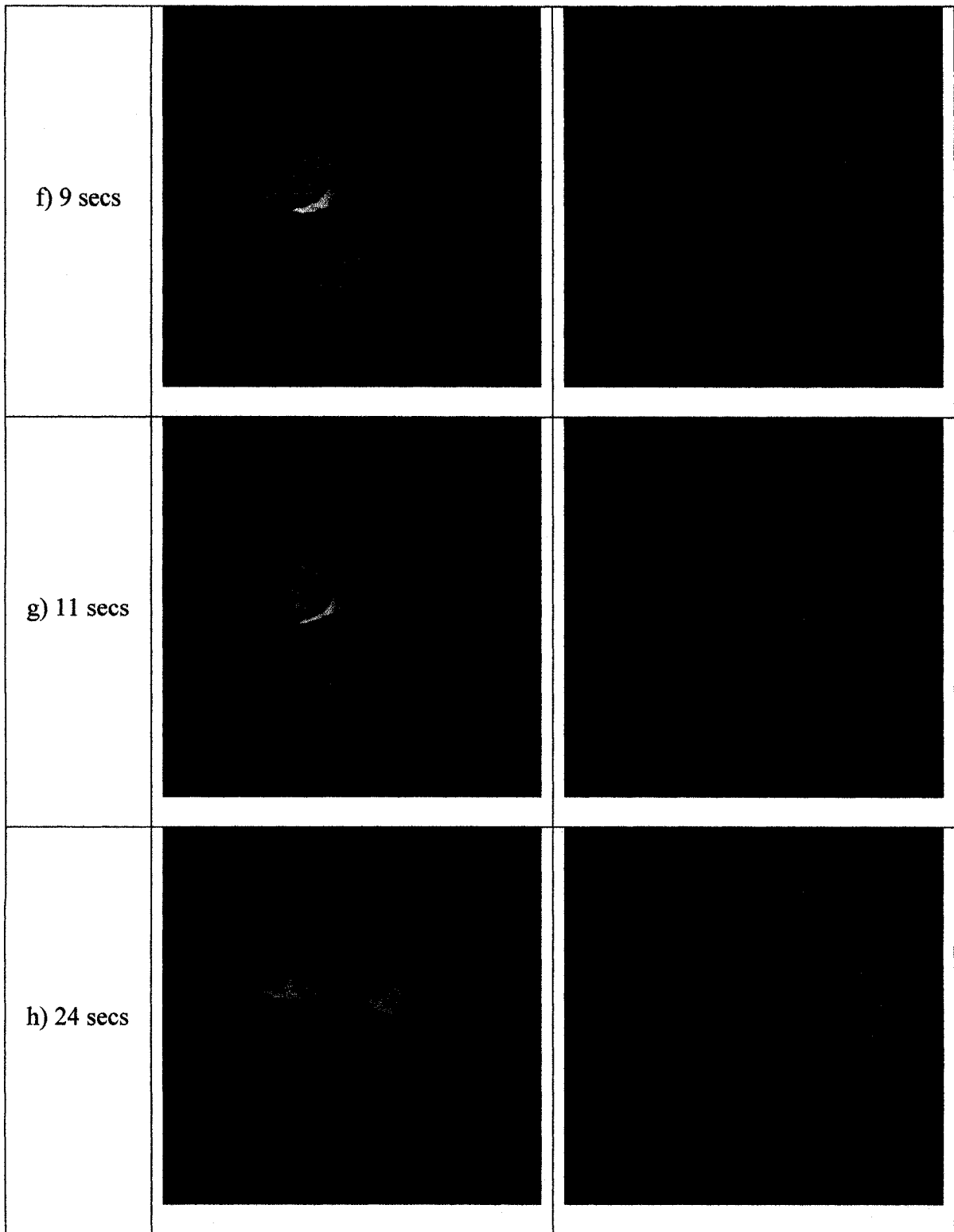
Next, the effect of gravity is disabled to test shear and bend conditions. The same initial positioning used in our first set of tests over simulator number one is repeated here to test shear and bend on this second model. Figures 7.15.a to 7.15.g illustrate the first eleven seconds of evolution of the system under shear forces. The cloth contracts on the axis running from its top left corner to its bottom right corner while expanding on the perpendicular one. Since there is no damping, the inverse motion completing a cycle of oscillation can be constructed by taking the same set of figures but reversing the order from eleven seconds to zero seconds. To be exact, the cloth never really comes back to the state shown in Figure 7.15.a at time zero. Instead, it reaches a similar state as shown in Figure 7.15.h at around twenty four seconds of simulation. Without damping or additional intervention, we can observe one more visual artifact, not representative of real cloth. Between time nine and thirteen seconds, the shear forces may have reached equilibrium or even pushing values, no longer pulling on the contracting axis; But, similarly to the case of our first simulator where a mechanical spring system needed damping, it is here also the case that while the forces no longer exerted a pull, the acceleration and velocities have not been made null, so the cloth nodes can pass over their equilibrium positions. In the case where the nodes are located towards the center of



the contracting axis, the nodes just pass through each other and the mesh structure is temporarily wrong. While a spring system can behave like that, real cloth cannot. This artifact can be dealt with two solutions. Either collision detection-and-handling process can handle that or the property of curve formation discussed in Section 7.4 can also solve the artifact. There is no shear spring with this model nor is there any diagonal edge in the orthogonal direction but when this artifact occurs, if we compared that to the first model, it is as if the shear spring reached a an opposite direction vector.


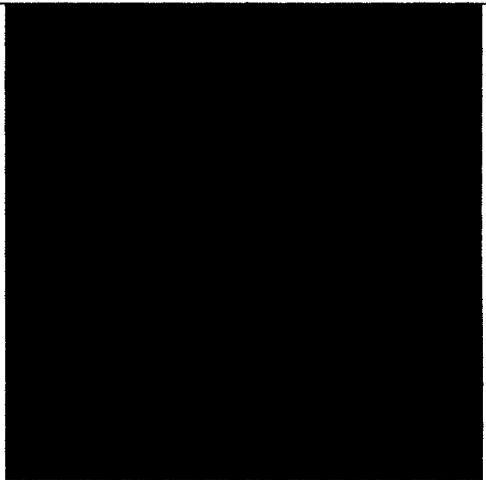
Time value	Textured cloth surface displayed	The triangles underneath
a) 0 sec		
b) 2 secs		

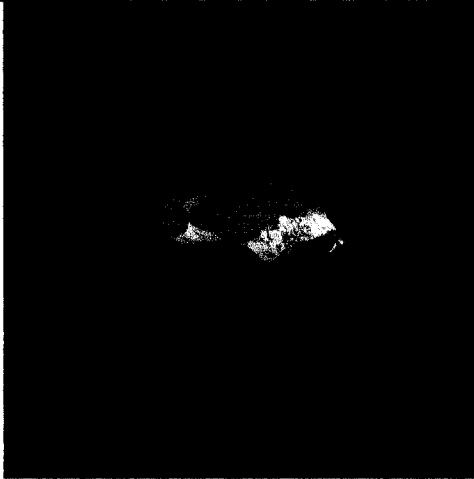
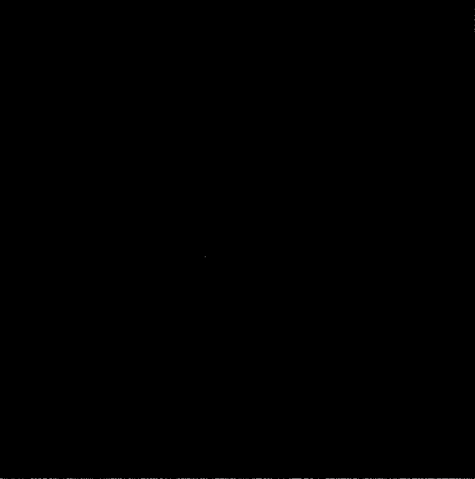
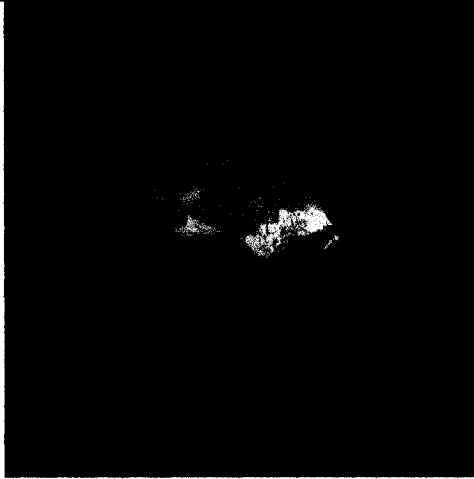
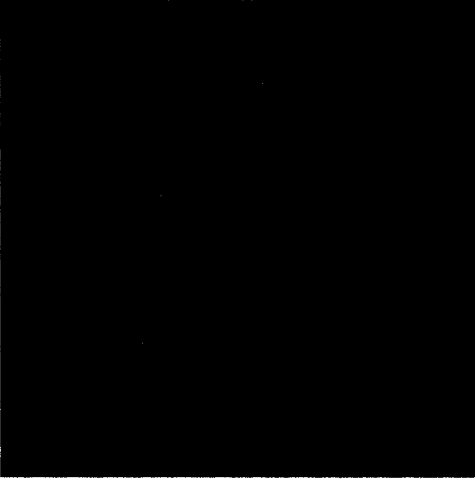

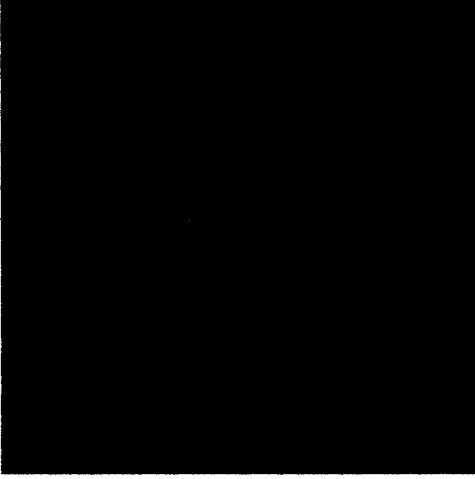
c) 4 secs		
d) 7 secs		
e) 8 secs		

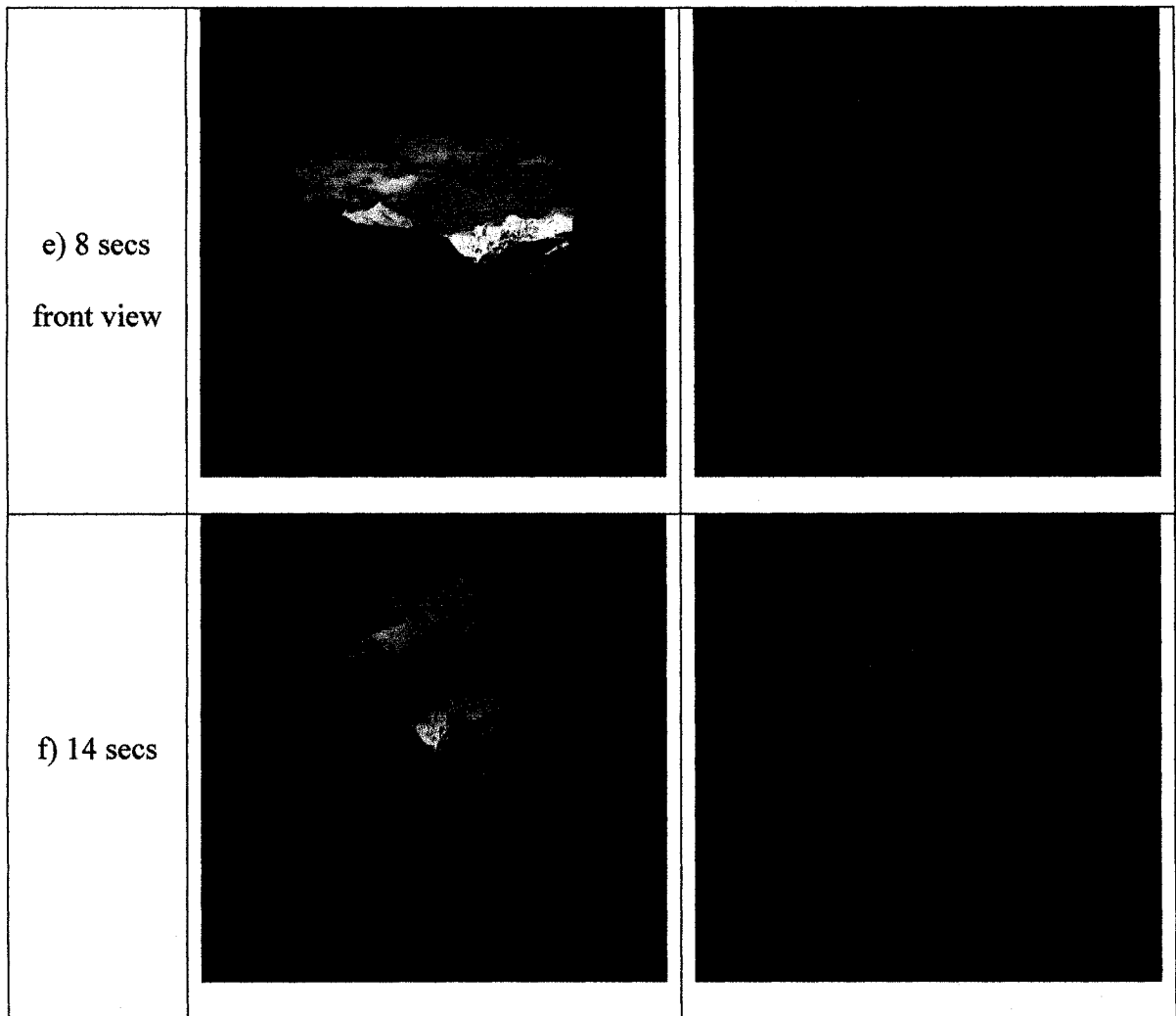


*Figure 7.15. The second set of frames captured for test#1, simulator #2, simulating the shear condition.*

For the bend condition, with the stiffness coefficient we set, after fourteen seconds, the cloth more or less tends towards a flat state with most of its dihedral angles near their resting values. In comparison to the spring model illustrated in figures 7.12 of the first simulator test set, we observe that this model's formulation of the bend property emphasizes entirely on bringing the dihedral angle back to its rest value, and we set this resting value at 180 degrees. Consequently, the right angles we set unrealistically as an initial state on the cloth are more rapidly replaced by smooth and low-curvature surfaces than the first simulator was able to produce. Fourteen seconds may seem a bit long but we must remember that cloth is more permissive for its bending resistance than for its other two properties.

Time value	Textured cloth surface displayed	The triangles underneath
a) 0 sec		

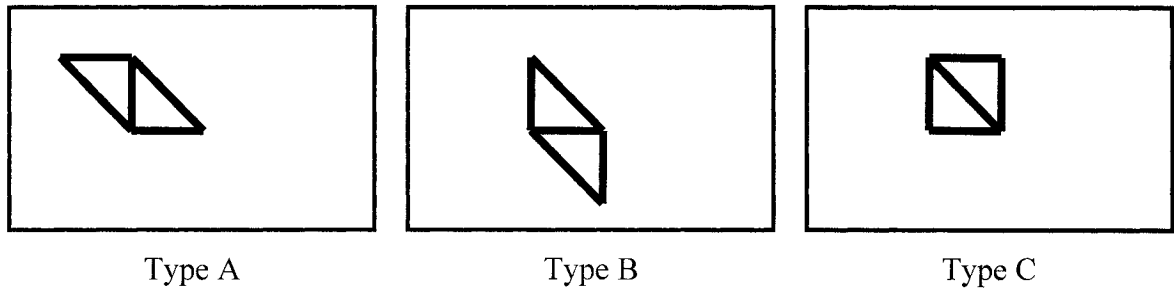
b) 4 secs		
c) 6 secs		
d) 8 secs		



*Figure 7.16. The third set of frames captured for test#1, simulator #2, simulating the bend condition.*

The bending resistance is satisfactorily modeled by this condition function except for one thing. Given the starting position we gave to the cloth, the reaction should be uniform for all nodes on a same vertical height. As a result, nodes located on a same row should therefore have the same y and z values; but, as Figure 7.16.e shows the cloth after eight seconds from a front view, that is not the case. Since the formulation of this condition over each pair of triangles holds nicely and since the derivation of forces applied on each of the triangle's particles is well defined, the problem resides in how the triangle mesh is structured. With our current mesh structure, there are three kinds of triangle pairs.

(Figure 7.17) For the purpose of easy reference, we will identify them as type A, B, and C.



*Figure 7.17. The three types of triangle pairs of the cloth mesh.*

Now, let us suppose a similar cloth mesh structure in Figure 7.18 where the dihedral angle on the line from node P to node Q is not equal to a rest angle value. As a result, all the type-B pairs of triangles whose common edge lies on line PQ will exert bend condition forces on their respective particles. However, it is clearly observed that the two nodes G and H are left out without any bending resistance forces while it would be common sense that they are submitted to the same internal forces.

An equivalently similar problem can happen with the type C pairs of triangles on the other axis. These are small issues specific to the triangle mesh structure but they still must be taken care of for the model to function correctly. One simple solution is to detect whether or not the triangle pair is located on the side and apply the corresponding forces on the nodes that would otherwise have been missed out like nodes G and H.

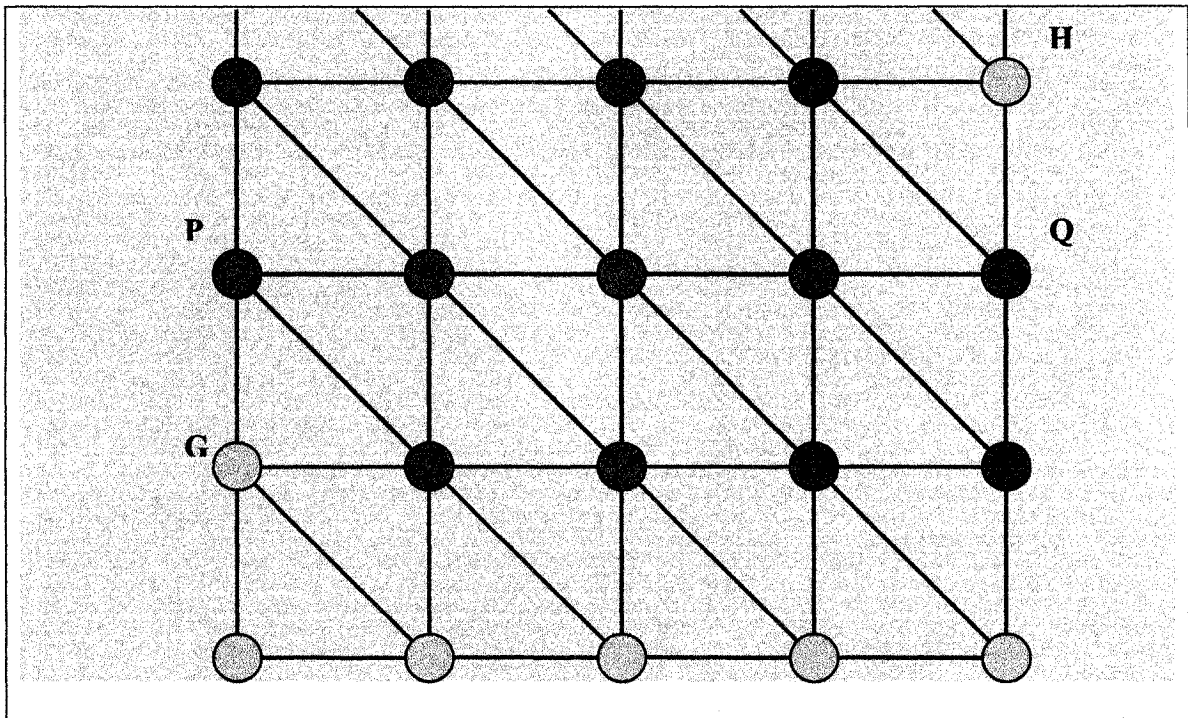


Figure 7.18. Nodes H and G are left out by the bend condition.

- Simulator2, Test2: A flag under the wind

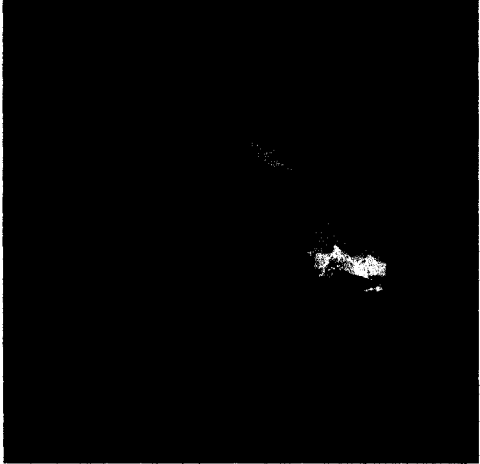
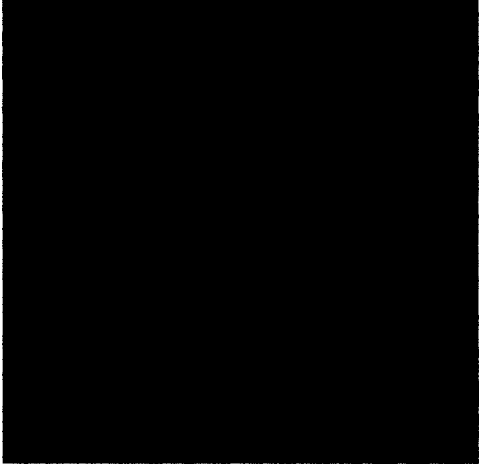
This test consists of simulating again a flag flapping under the wind but this time using the second simulator. The external forces of gravity and wind will be reactivated along with the viscous damping and post-step constraint mechanism components. Without any form of damping, the simulator cannot perform this test without losing its stability, and without a post-step modification mechanism, the cloth is much overstretched under the wind as shown in figure 7.19. But with both the previously mentioned components active, the simulator behaves similarly to the first simulator for this test. It is in fact very difficult to tell them apart without running the tests side by side. This proves that under



similar settings and under the same integration method, the two cloth models represent the three basic cloth properties, stretch, shear, and bend, in a similar manner.



*Figure 7.19. The cloth is much overstretched under the wind.*

Time value	Textured cloth surface displayed	The triangles underneath
a) 6 secs		

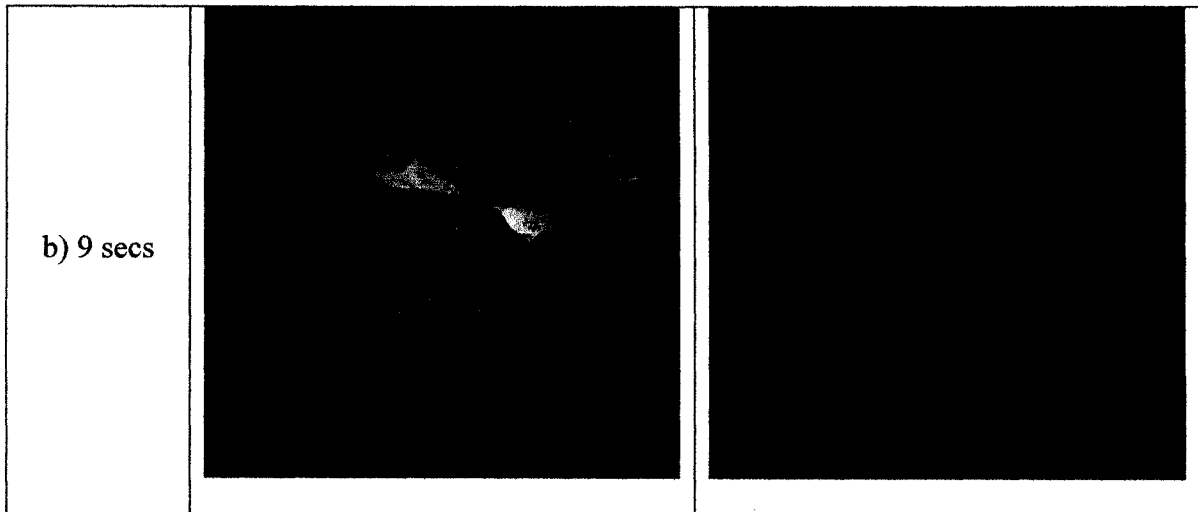


Figure 7.20. The second model behaves similarly to the first model under similar conditions.

- Simulator2, Test3: Implicit Integration

The use of an implicit numerical integration method is what really differentiates this model from the first. In order to utilize the implicit integration method, the components initializing and computing the terms  $df/dp$  and  $df/dv$  must be enabled and the integration method must be replaced by the Backward Euler one.

The implicit time integration is meant to solve the instability issue. Unfortunately, at this point, our results remain inconclusive. We can noticeably observe a higher computation overhead, but the gain in stability is not evident. The system still remains incapable of subduing oscillations without the damping method, and, it still diverges under the same time step size that caused the explicit version to diverge. Even though the system does last a few seconds more before it becomes unstable, the instability is not completely solved and the performance trade-off does not seem satisfactory. We believe there are at least two reasons explaining this lack of convincing results. First, the number of nodes in our simulator may be too small to visibly

demonstrate the gain of using an implicit time integration method. Moreover, with our current implementations, the stability of the system seems to depend more on the use of damping methods and constraint enforcement mechanisms than it depends on the numerical integration solution. Also, the flaw may as well originate from possible inaccuracies of our implementation for the lower level methods of the implicit integration scheme such as the second derivative terms. Nevertheless, more conclusive results can be obtained from making small changes to the implicit method described previously. But, this solution cannot be presented before the alternative solution of “Mixed Implicit/Explicit” is presented in section 7.2.3.

### **7.1.2. Visual Differences and Behavioral Dissimilarities**

The two simulators may have yielded resembling visual results given the same piece of cloth simulated and the duplicate settings placed for the two set of tests, but the small differences that were perceivable give a glimpse to much larger underlying dissimilarities of the two models.

- It is observable that, given the current mesh structure, the second model does not define any bending resistance in the direction orthogonal to the blue segment shown in Figure 7.21 below. To simulate the bend condition more completely, it would be wise to go beyond the limitation of the mesh structure used for display and consider a fourth type of pair of triangles shown on the right of Figure 7.21. Processing this new type of triangle pairs will eliminate additional bias that would have been created by our mesh structure otherwise.

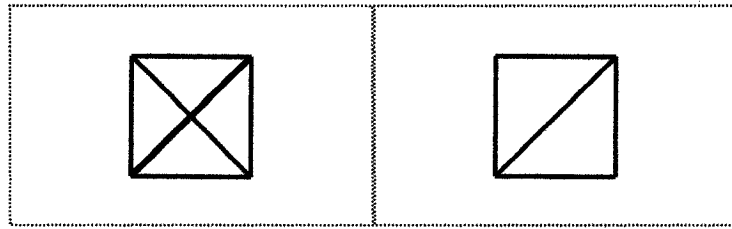


Figure 7.21. A fourth type of triangle pairs to complete the bend condition.

We avoided using triangle mesh structure like the one shown below (Figure 7.22) for display and for the other two properties because the intersection formed by the two diagonal segments of each quad adds a new node that must be accounted for the mechanical calculations in the simulation.

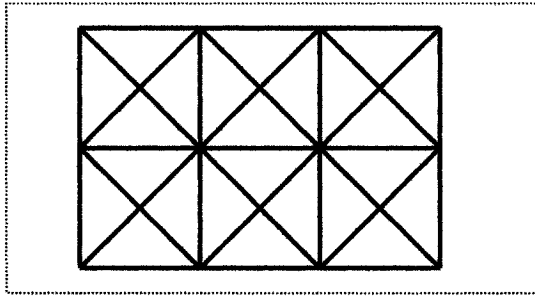


Figure 7.22. A mesh structure with one more node per quad.

- The first cloth model on the contrary did not use its displayed mesh structure for mechanical computations. For this reason, the first simulator avoids problems such as a loss of uniformity and a lack of symmetry. However, the first model needs a rectangular mesh structure on which it can install its springs on the warp and weft axes, whereas the second model's conditions, except for shear, can be defined over any type of triangle mesh, be it regular or irregular. The shear condition, although possibly adaptable to any irregular mesh, was originally defined for a regular triangle mesh structure. Without distinct warp and weft axes, the shear springs of the mass-spring model lose their meaning and use entirely.

- Again, while the first simulator uses flexion springs to simulate the bend resistance property of cloth, the second simulator worked a condition function defined over the dihedral angle between pairs of adjacent surface triangles. The visual results showed that working on the angle between two surfaces gave out a smoother motion than using springs. Additionally, the spring model only covered bending along its warp and weft axes, not considering possible bending resistance along any diagonal axes. The second model as we have mentioned previously can be modified to calculate bending forces along both diagonal axes of each quad in the mesh structure. Relating to the bend property of cloth, the second simulator has another advantage over the first in that its bend condition function is sufficient to handle the property on its own unlike the flexion springs that alone, cannot satisfactorily simulate bending property. Also, flexion springs can interfere with in-plane forces. Figure 7.23 below shows a state in which there is shearing. The shear springs are of course reacting; however, we can see that the flexion springs are also shortened, making them exert repulsion forces on their nodes. The motion resulting from these forces may not look wrong, but the interference of the bend property when there is no actual bending happening makes it very hard to distinguish out-of-plane and in-plane forces. This difficulty in isolating the results of two different cloth properties renders the control and representation of different fabrics inefficient.

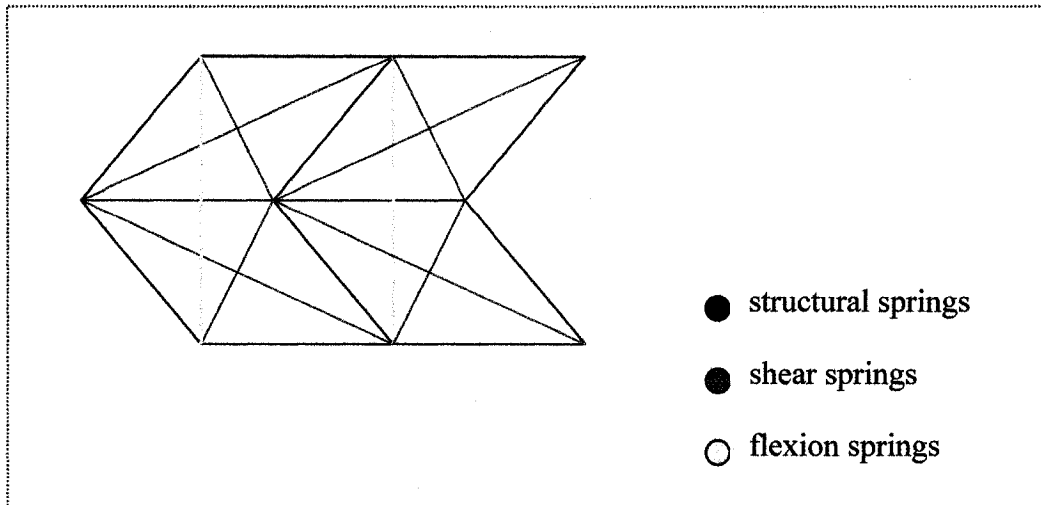


Figure 7.23. *The interference of flexion springs on in-plane forces.*

- We have mentioned in the previous section and shown with Figure 7.18 that a few nodes can be incorrectly left out of the bend resistance computation with the second model and the problem is related to the mesh structure used. This problem occurs whenever there are bending forces on either the warp or weft axes near the sides of the cloth mesh. A garment or complex piece of cloth will have many of its parts seamed together so the occurrence of this problem is reduced but not eliminated because a piece of cloth will still have edges and nodes located on the sides unless the cloth is seamed closed like a sphere.
- We should not confuse the fourth type of triangle pairs defined previously to handle bend in the other diagonal direction and the type of pairs shown on the right of Figure 7.24. That type is nothing but a deviant of type-B but just for meshes defined by rotating our cloth mesh one hundred and eighty degrees about the vertical axis. Adding the type of triangle pairs that we can refer to as B' cannot solve the current problem. We end up by accumulating twice the bend condition forces on every node but those located on the sides.

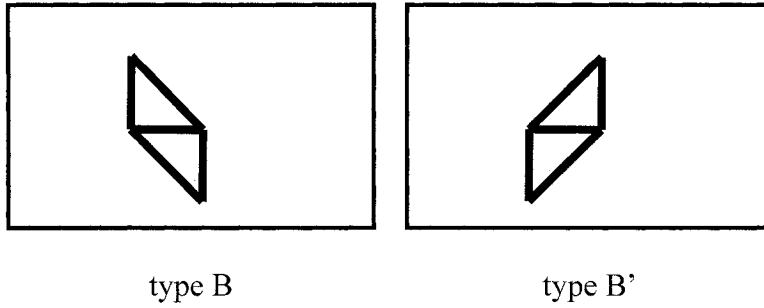


Figure 7.24. Type B' is nothing but type B for meshes rotated 180 degrees about the vertical axis.

Hence, the simplest solution is to take care of the nodes left out if the triangle pair processed of either type A or B is located on a side.

- There is a concern with using shear springs, and this concern also consists a meaningful difference between spring models and models that define the shear condition with the shearing angle like the second model does.

Given cloth quad being stretched in the vertical direction as follows:

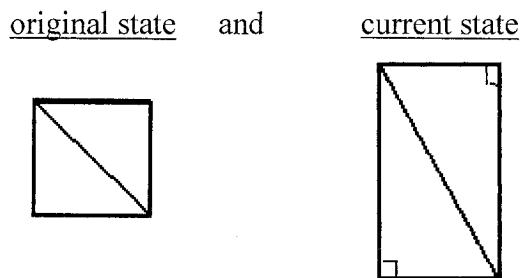


Figure 7.25. A cloth quad elongated vertically.

Under Baraff Witkin's shear condition  $\mathbf{C} = \mathbf{a} \mathbf{W} \mathbf{u} \bullet \mathbf{W} \mathbf{v}$ , this is a zero shear force while with models such as Provot's using shear springs, the diagonal spring there in the middle would have pulled back because its condition was violated. But in reality, it is common

sense to state that there was only vertical stretch conditions being incited and that, once settled, would not involve any shear condition being unsatisfied.

Care must be taken as to what this last observation meant because different solutions to this concern lead to completely different paths.

A first path would be to forfeit the use of diagonal springs if a shear violation were considered two stretch violations. However, this path is not exactly a solution because while it remains true that stretch often occurs on the cloth near a sheared area, nothing guarantees that the case where the stretch springs are at their natural length but the shearing angle is not null cannot happen. In any case, leaving the task of modeling the shear and in-plane bending properties to stretch springs is inaccurate and unpredictable. If shear springs were discarded, then they would need to be replaced by a different formulation for shear resistance to deal with the excessive in-plane angle between yarns.

A related alternative is to simply disregard the pull from shear springs but keep the push from those since they become necessary when dealing with proximity induced curves discussed in Section 7.4. Again, this idea cannot entirely omit the use of shear springs; it only directly solves the case illustrated previously. When shearing occurs, it is likely that one diagonal spring of a cloth quad will be over-elongated while the other will be shortened. This second path suggests only considering the repulsion of the shortened spring and leaves the pull of the elongated spring to the stretch springs that are connected to the nodes concerned. A different path would not involve replacing or discarding this type of spring but concerns the overall algorithm's order. Instead of simply gathering both external forces and internal forces resulting from unsatisfied conditions each step,

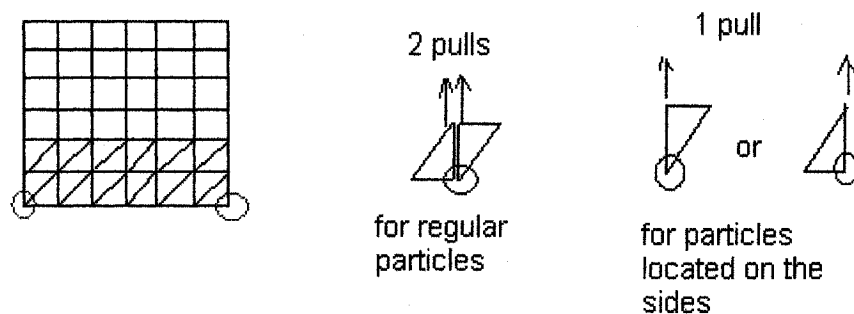


solve for stretch before shear condition is checked. Though, this approach would involve the numerical integration solver twice; once for everything but the shear property, and once more to obtain the shear forces. The current best solution would be to replace shear springs with a formulation based on the shear angle like the one described in the second model, and thus utilizing a hybrid model to represent cloth.

- Defining the stretch condition over a triangle, as in the second model, instead of a spring has some drawback and the solution to this concern may be an arbitrary and unjustified decision. While testing the vertical stretch condition, errors in the visualization revealed an important fact: the two corners or nodes would only receive about half of the pulling force against stretch. After a few additional test runs combined of vertical, horizontal, and other stretch directions, the observation could be finally related to the way the stretch condition is defined. Unlike most particles, nodes located on either of the sides only receive pull from one triangle over which they are defined when either the force is vertical or horizontal. Figure 7.26 illustrates the problem where all but the particles on the sides obtain forces from two triangles against in-plane stretch forces in each direction.

Let the two triangles in Figure 7.26 adjacent to a regular node be labeled  $t_A$  and  $t_B$ . Then, one immediate solution to this is to get some force from the triangle  $t_A$  and some from  $t_B$ , this way ... all triangles receive about the same force under the same stretch condition. However, some questions arise from adopting such a solution. First, how much are we supposed to take from each triangle? Then, let us suppose without loss of generality that we decide to take half from each, what would justify such a choice? In fact, this problem resembles the one with the bend condition where certain nodes were

left out of the forces computed in that it relates on the mesh structure of the cloth. The solution is to handle nodes on the sides differently from the other nodes once again. Whether we decide to halve the forces applied from each of the triangles  $t_A$  and  $t_B$  for all non-side nodes or we choose to double the stretch condition forces applied to side nodes is just as arbitrary yet a choice is necessary to solve the problem without changing the cloth model.



*Figure 7.26. A concern particular to defining internal forces over triangles instead of using springs.*

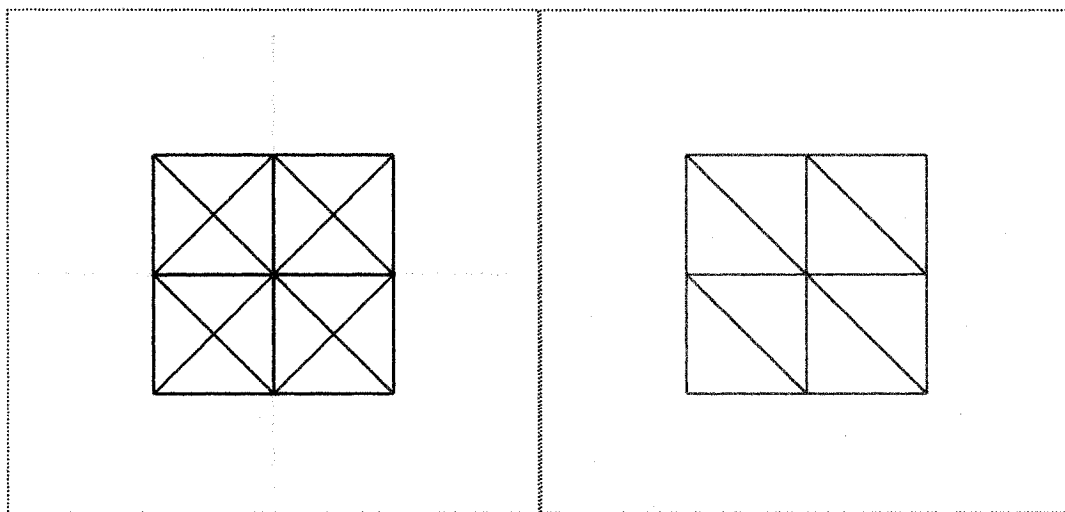
When running the simulation as a whole, the noticeable effect of this unsolved concern is that sides of the cloth appear more rubber-like.

### 7.1.3. Quantitative analysis

In this section, we present some quantitative results and a brief analysis of these numbers. The previous two sections allowed us to compare the visual outcome of the two simulators and judge their accuracy to reproduce cloth behaviors; we were also able to point out the dissimilarities of the two models. The topics of this section are performance and computation overhead; yet, the goal is not to determine which simulator

or model is more efficient. Rather, the objective is to successfully locate the concerns for computation overhead and performance issues within each of the two simulators.

To start, a node in the first simulator undergoes a smaller number of mechanical relationships than a same node does with the second simulator. This does not involve the computation size load of each relationship yet. But, just in terms of number of relationships to be computed, the second simulator is already showing a possible greater computational burden. Figure 7.27 shows that nodes in the mass-spring model each are connected to eight edges and twelve springs. Since springs determine the mechanical relationships of particles in this model, that means twelve relationships per node for internal forces. On the other hand, nodes in the second model are connected only to six edges and six adjacent triangles. However, the stretch and shear condition each need to process these six triangles and the bend condition needs to process six pairs of adjacent triangles connected to this same node; so, a total of eighteen relationships must be accounted for each node in the second simulator.



*Figure 7.27. The relationships of a node and its neighbors under a mass-spring model on the left, and under the condition model, on the right.*

We believe it is necessary to break each simulator back into its components and be able to identify the computation overhead associated with each component. Once the relative computation overhead of each simulator component becomes known, the efforts in optimizing for performance or employing alternative solutions can more easily be focused. The subsequent sections of this chapter, after 7.1, study alternative solutions to some of these components.

◇ **Component Overhead Tests for Simulator 1**

The component overhead tests will be performed as follows: All but the collision and detection component will be active; and, the simulator will be run for a predetermined amount of simulation time. The corresponding real time for that simulation run will be recorded. Then, each subsequent test will disable a single component while leaving every other component active; and, run for the same amount of simulation time while the real time elapsed will also be recorded. Collision and detection is an obvious computation load and can require optimization and efficient solutions. We do not include this component for the overhead tests. Also, initializing components will not be disabled. We are currently only interested in determining the relative overhead of each of the component in the simulation loop.

The tests were run on a Pentium 4 1.7Ghz system with 512 RAM. The time step size used is 0.1 seconds, the first set of runs will use a mesh of size 8x8 and a simulation time of 60 seconds, whereas the second set of runs will simulate 30 seconds with a 16x16 node

grid. The second column in Table 7.5 displays the result for the first set and the third column, the second set.

Component disabled	Real-time required to simulate (seconds)	
	8 x 8 nodes 60 simulated seconds	16 x 16 nodes 30 simulated seconds
None (all active)	20.5	50
Internal Forces	17.5	18
External Forces	20	47.5
Fixed Nodes	20.5	50
Damping	20.5	50
Explicit Integration	20	48
Post-step modifications	20.5	50
Inverse Dynamics	20.5	50
Normal vectors for each vertex	19.5	47
rendering	20	48

Table 7.5. The relative overhead tests for components of simulator 1.

We are well aware that the results are greatly dependent on the implementation methods used as well as numerous other factors; but, the accuracy of these time values is not what we seek to obtain. The relationships emerging from the analysis of these values are what pertain to us. The results are interpreted as follows. Components with a smaller time-value have a relatively more significant computation overhead associated, whereas components whose value is close to the time value of the simulator run with all components active have a relatively small computation overhead. As shown in Table 7.5, most of the components listed have a relatively easily manageable computation overhead except for the internal force module. The larger number of nodes allows us to observe the drastic speed up the simulator gains by dropping the computation of internal forces.

Thus, as the number of nodes increases, it becomes crucial to optimize the implementation of this simulator component as much as possible; since, the deactivation of this module for cloth simulation is not an option.

We quickly discovered the implementation mistake that increased the computation load of that component for our simulator, a problem that may have been missed out given the initially small number of nodes involved in the simulation. The method was implemented to compute, for each node, the forces obtained from the springs to which the node is attached. However, the identification of the springs was done through loop form verification; a wasteful implementation tactic indeed. We have stated above that each node is in exactly twelve relationships for this mass-spring model. Therefore, exactly twelve relationships should be calculated for internal forces of each node. The updated version of this component allows our simulator to run with the following results: 19.5 seconds with all components active instead of 20.5 seconds for the 8x8 mesh and 60-second setting, and, 31 seconds instead of 50 seconds for the second setting.

#### ◇ **Component Overhead Tests for Simulator 2**

The overhead tests for the second simulator are performed similarly. The time step size remains 0.1 seconds; the mesh sizes and the simulation time targeted for each test set are the same as before. The only difference is that the second model has the option to switch to the implicit Euler integration method. So, the first row of values still represents the real time needed for the simulator to run with all its components active with the explicit

scheme, while the second row will now contain the real time values of this same setting but with the implicit time integration component instead of the explicit one.

Component disabled	Real-time required to simulate (seconds)	
	8 x 8 nodes 60 simulated seconds	16 x 16 nodes 30 simulated seconds
None (explicit)	30	70
None (implicit)	104	300 + (over 5 minutes)
Internal Forces	19	20
External Forces	29.5	69.5
Fixed Nodes	30	70
Damping	30	70
Explicit Integration	30	70
Post-step modifications	30	70
Inverse Dynamics	30	70
Normal vectors for each vertex	29	66
rendering	29	65

Table 7.6. The relative overhead tests for components of simulator 2.

The implicit integration scheme requires the definition of the terms  $df/dp$  and  $df/dv$  which in turn require the computations of the second derivatives of the condition functions. In short, the implicit method requires the computation of many more terms before the final position update can be calculated. The computation of each additional term adds to the overhead thus making this integration method more expensive. For a relatively small number of nodes, the ratio of real time over simulation time is still below two. But, as the number of nodes becomes larger, the time needed to compute and to display a frame is about two seconds per frame, a performance that may not be feasible for real-time applications. For the most part of the results, the relative overhead distribution among

components seems similar to the first simulator; except, the deactivation of the explicit integration component does not seem to speed up this second simulator at all.

## ***7.2. Alternative numerical integration methods***

Section 7.1.3 just showed that the numerical integration component of a cloth simulator can limit its performance. Ideally, we would wish for a method that is accurate and yet does not induce any heavy computational overhead. Instead, the two numerical methods used so far on our simulators, the regular explicit Euler and the implicit Euler, illustrated a tradeoff between speed and accuracy. Perhaps, we could find a solution with better performance, or more accuracy and stability could be gained, or, simply, a different method could produce the same visual results while simplifying the implementation load required.

To find out, we opt to try the following additional numerical methods and approaches:

- The 4<sup>th</sup> order Runge-Kutta integration method
- The Verlet integration scheme
- A mixed Explicit/Implicit method
- A different strategy to the instability problem

The three different numerical methods were tested over a variant of the first mass-spring simulator. The simulator was modified to work with its new integration method while keeping most of its other components such as internal dynamics and collision detection methods as similar as possible.



Here is a summary of the results:

- The 4<sup>th</sup> order Runge-Kutta integration algorithm can maintain stability without requiring any damping method. Also, this scheme offers more accuracy and stability over the regular explicit Euler method without giving up as much performance as the implicit Euler does. However, our formulation of wind no longer worked as well under this scheme and needed to be revised. In short, this integration worked for our simulator in the most part and brings a more beneficial trade-off for cloth simulation in general if compared to either the explicit Euler or the implicit Euler schemes
- The system stability and performance of the cloth simulator implemented under the Verlet integration scheme were similar to those of the regular Euler method. We believe that the motions of the cloth would be closer to those of real cloth under this scheme because both the state of the previous time step and the current state are used to compute the next time step's; but our test results cannot confirm this speculation. However, this integration scheme has one observable advantage over the regular Euler scheme: its associated damping strategy immediately acts in the same time step and does not produce forces that need to be accounted for during the next time step.
- The mixed Explicit/Implicit integration scheme produced surprisingly stable results. The system stability improved much by combining the accuracy of the implicit method for velocity-dependent forces and the simplicity of explicit methods for all other forces. Its performance was slightly improved over a full

implicit method. However, this mixed scheme also requires the implementation of the several definitions required under an implicit scheme. The mid-step velocity computation seems to be a key strategy to benefit from the stability offered from the implicit scheme.

- Our analysis also shows that the instability of a cloth system is due mostly to excessive external forces that are not appropriately handled. By defining new mechanisms such as our force-transfer mechanism to prevent artifacts like node cross-overs, the cloth simulator can improve its stability significantly without having to change the numerical integration method.

In conclusion, the mixed Explicit/Implicit or the 4<sup>th</sup> order Runge-Kutta schemes are both advantageous alternatives, and upgrading our cloth simulator from a regular Euler integration method to either of these would improve its overall performance. Our test results correspond with those of other references using these two methods. Additionally, excessive external forces ought to be handled by new mechanisms such as our force-transfer formulation to reduce the possibility of system instability. This last approach does not require upgrading a current simulator to any new integration scheme and can easily be adapted to most particle-based cloth simulator.

Sub-sections 7.2.1 to 7.2.4 offer a more detailed discussion of these alternative solutions.

### 7.2.1. The 4<sup>th</sup> Order Runge-Kutta Integration Method

The first alternative integration method tested is the 4<sup>th</sup> order Runge-Kutta integration scheme. This scheme is one of the standard algorithms to solve differential equations. Similarly to the mid-point algorithm, approximations to the object's state within the time interval are used in addition to the knowledge known at the beginning of the interval in order to obtain the resulting state.

The algorithm is as follows:

$$X_{n+1} = X_n + (h/6) \times (K_1 + 2K_2 + 2K_3 + K_4)$$

with

- 1)  $K_1 = f(t_n, X_n)$
- 2)  $K_2 = f(t_n + h/2, X_n + (h/2) \times K_1)$
- 3)  $K_3 = f(t_n + h/2, X_n + (h/2) \times K_2)$
- 4)  $K_4 = f(t_n + h, X_n + h \times K_3)$

*(Algorithm 7.2)*

The algorithm is easy to implement once all its terms above are identified and understood.  $X_n$  is the particle's position from the end of previous time step or equivalently, what is known at the beginning of this time step. Let the velocity be expressible by

$X' = f(t,x) = dx/dt$ . Then, this is a four-step algorithm where each step computes a  $K_i$  (i from 1 to 4) representing a trial-step velocity of that interval.  $K_1$  represents the velocity computed using the knowledge of the position at the beginning of the time interval;  $K_2$  is

the velocity at the mid-point of the interval using the position updated with the knowledge of velocity  $K_1$ ;  $K_3$  is also a midpoint-velocity but this time referring to a midpoint position obtained with  $K_2$ ; and finally,  $K_4$  is the velocity at the end of the time step computed with the knowledge of  $K_3$ . The next positional state  $X_{n+1}$  will be computed as a linear combination of the position at the beginning of the time step and these four trial-step velocities.

Surprisingly, the simulator implemented under this integration method can even maintain nearly full stability without any damping method. Also, without any mechanism to limit elongation and without damping, the cloth also starts by bouncing up and down for the stretch test, but unlike with the explicit Euler method, the oscillations are quickly reduced, and the cloth eventually stabilizes itself. However, our wind mechanism was no longer accurate with this integration method and the behavior of the wind effect was rather abnormal but did not render the system unstable. In general, the computation overhead increased but fortunately, so did the accuracy and stability. Since our simulator does not use a large number of particles, the performance remained satisfactory. This trade-off observed corresponds with the analysis of Volino and Magnenat-Thalmann's paper [VOL00] comparing the efficiency of various integration methods for cloth simulation.

## 7.2.2. The Verlet Integration Scheme

The Verlet integration scheme suggested in Paul Meli's work [MEL02] and previously presented by Jakobsen [JAK01] originates from the domain of molecular dynamics. There is more than one Verlet integration scheme, and all were developed specifically for solving the set of Newton's equations:

$$\frac{dx}{dt} = v$$

$$\frac{dv}{dt} = g(x)$$

In our cloth simulation case, a particle's acceleration may be given directly by dividing the forces applied on the particle by its mass, but the internal forces of the cloth are in turn functions of the particle's position, so what this integration scheme seeks to answer would also suit our goals. The basic version of this method starts by defining the particle positions  $x$  as a function  $f$  of time  $t$ . A particle's position at time  $t$  is given by  $x_t = f(t)$ ; Then,  $f$  is approximated with a third order Taylor expansion in the vicinity of  $t=t_0$

$$f(t) = f(t_0) + (t-t_0)f'(t_0) + \frac{1}{2!}(t-t_0)^2f''(t_0) + \frac{1}{3!}(t-t_0)^3f^{(3)}(t_0) + O((t-t_0)^4).$$

*(Equation 7.3)*

This approximation is applied to  $f(t_0+h)$  giving

$$f(t+h) = x_{t_0+h} \approx f(t_0) + hf'(t_0) + \frac{1}{2}h^2f''(t_0) + \frac{1}{6}h^3f^{(3)}(t_0),$$

*(Equation 7.4)*

then, let us rewrite it in terms of particle position  $x_t$  being updated over time with step size  $h$ , and use the substitutions  $f'(t)=v_t$  and  $f''(t)=a_t$  for the particle velocity and change in velocity:

$$x_{t+h} = x_t + hv_t + \frac{1}{2} h^2 a_t + \frac{1}{6} h^3 f^{(3)}(t). \quad (\text{Equation 7.5})$$

Now, the same approximation is done to  $f(t_0-h)$  and yields a position update backwards in time expressed as

$$x_{t-h} = x_t - hv_t + \frac{1}{2} h^2 a_t - \frac{1}{6} h^3 f^{(3)}(t). \quad (\text{Equation 7.6})$$

If both expressions of the position update forward in time and the update backwards in time were summed, we would get

$$\begin{aligned} x_{t+h} + x_{t-h} &= 2x_t + h^2 a_t \\ \Leftrightarrow x_{t+h} &= 2x_t - x_{t-h} + h^2 a_t \end{aligned} \quad (\text{Equation 7.7})$$

We can observe that there are no velocity terms in this position update expression. The previous and current positions are used to compute the next time step's. In fact, the particle's velocity  $v_t$  has been implicitly approximated as  $\frac{x_t - x_{t-h}}{h}$  when the Verlet scheme summed its two Taylor series, else we would have observed the regular expression  $x_t + hv_t + \frac{1}{2} h^2 a_t$ .

A form of viscous damping can be added by rewriting the expression with

$$x_{t+h} = (2-\varepsilon)x_t - (1-\varepsilon)x_{t-h} + h^2 a_t = x_t + (1-\varepsilon)(x_t - x_{t-h}) + h^2 a_t$$

(Equation 7.8)

where  $\varepsilon$  is a small positive number such as 0.01. So, adding viscous damping consisted of scaling the approximation of velocity by a positive factor  $(1-\varepsilon)$  less than 1. If the velocities need to be updated for reasons like calculating forces caused by viscous drag, air drag, or other types of forces dependent on the particle velocities, the velocities can be computed by  $v_{t+h} = (x_{t+h} - x_t)/h$ .

We tried this time integration scheme for the mass-spring model and the results were similar to the regular explicit Euler method in terms of stability and performance. The longer the simulation ran, the larger was the difference in position for the piece of cloth from one scheme compared to the other. Although different, the motion under either scheme looked equivalently natural. Again, the velocity is implicitly understood and does not need to be computed under the Verlet scheme; and, both the previous and current positions are used in the computation of the next, thus the motion ought to be more natural. However, that is only speculation from our part; the benefits of this scheme versus the Euler method for the purpose of cloth simulation remain unclear from our tests. The artificial damping number  $\varepsilon$  needed to be as much as 0.25 in order to simulate the viscous damping necessary to make the simulation look natural. But, we observed that this artificial damping mechanism, unlike viscous damping, added no forces to the system and was immediately applied for the current time step instead of waiting for the next iteration for the contributions to take effect. This is possibly an advantage to using the Verlet method instead of Euler's basic scheme.

### 7.2.3. The Mixed Explicit/Implicit Integration method

Bridson et al. [BRI03] in their 2003 work presented a time integration method that combined the flexibility and simplicity of explicit methods with the accuracy and stability of implicit schemes. More specifically, explicit methods were used for the velocity independent forces like the elastic forces, whereas implicit methods were utilized to treat velocity-dependent forces like damping forces. Their algorithm was presented as follows:

- $\mathbf{v}^{n+1/2} = \mathbf{v}^n + \frac{\Delta t}{2} \mathbf{a}(t^n, \mathbf{x}^n, \mathbf{v}^n)$   $\Rightarrow$  (explicit)
- Modify  $\mathbf{v}^{n+1/2}$  into  $\mathbf{v}^{n+1/2}$  to limit strain
- $\mathbf{x}^{n+1} = \mathbf{x}^n + \Delta t \mathbf{v}^{n+1/2}$   $\Rightarrow$  (explicit)
- $\mathbf{v}^{n+1} = \mathbf{v}^{n+1/2} + \frac{\Delta t}{2} \mathbf{a}(t^{n+1}, \mathbf{x}^{n+1}, \mathbf{v}^{n+1})$   $\Rightarrow$  (implicit)
- Modify  $\mathbf{v}^{n+1}$  to limit strain *(Algorithm 7.9)*

with  $\mathbf{x}$  denoting positions,  $\mathbf{v}$  velocities, and  $\mathbf{a}$  accelerations.

However, oscillations may still be occurring because the first explicit half-step of the algorithm could have produced excessively large-value velocities before the later implicit half-step could have damped them down to stability. To remedy to these difficulties, the order of the velocity steps will be simply switched. The modified algorithm sequence starts with an implicit integration update for the velocity:



- $v^{n+1/2} = v^n + \frac{\Delta t}{2} a(t^n, x^n, v^{n+1/2})$   $\Rightarrow$  (implicit)
- Modify  $v^{n+1/2}$  to limit strain
- $x^{n+1} = x^n + \Delta t v^{n+1/2}$   $\Rightarrow$  (explicit)
- $v^{n+1} = v^{n+1/2} + \frac{\Delta t}{2} a(t^{n+1}, x^{n+1}, v^{n+1})$   $\Rightarrow$  (explicit)

(Algorithm 7.10)

But, it is necessary to note that the previous algorithm sequence does not remain stable or accurate for an adaptive time-step simulation. Instead, the authors describe an additional third approach for variable time step simulation. The description of this last approach follows immediately after the second in their work [BRI03].

The possible benefits of implementing such a mixed-scheme integration method were quite predictable. We estimated the results to be a more stable system than the explicit method while remaining less computationally expensive than a full implicit scheme. Nevertheless, due to the inconclusive results we previously obtained from the implicit version of our second simulator, we decided to adopt this scheme and analyze the resulting performance. But unlike the specifications stated above, we adopted this new scheme for all forces, whether they were velocity-dependent or not. To our surprise, the improvements were considerably better than expected. Oscillations could not yet be eliminated but the frequency of the oscillation was greatly reduced. The most obvious case was when the simulator ran without the damping method and the time step size was doubled from its default setting; the frequency of the oscillations was at least halved and the time in which the system remained stable could be prolonged considerably. Needless

to say, with the damping method and the constraint mechanism, the system's stability became a nearly guaranteed feature. Since our implementations have not changed for the implicit and explicit parts of this new method, we attribute the performance gain and stability improvement to the mid-step velocity calculation from which, the position update was computed. And, the layout of this time integration scheme allows our simulator to lessen the effect of excessive velocities by working out the velocities in two half-steps. In summary, this alternative time integration scheme unexpectedly allowed us to observe the benefits of the implicit scheme more clearly than our previous fully implicit scheme did.

#### **7.2.4. A Different Approach to the Instability Problem**

Thus far, we have presented some additional numerical solutions in the search for a way to diminish the instability of the system and allow larger time step sizes, to provide more numerical accuracy, or to simply compute each simulation step faster. We have observed that higher-order explicit methods such as the 4<sup>th</sup> order Runge-Kutta provided more accuracy and stability at the price of more computation overhead, and that implicit integration was a successful approach to obtain stability in exchange for the increase in implementation complexity it necessitates. In the previous perspective, we sought to subdue the instability via a different integration method because the large time step size and the inaccuracy of the current method were held responsible for this problem. But, from another point of view, the instability of a system originates from allowing the system to reach a critical state where either its reactions produce forces that are too strong, or the system's coherency is lost, and its mechanisms are incapable of making the

appropriate corrections. First, we ought to remember that we always assumed a realistic and stable starting simulation state for the system. So, in order for the system to even reach a problematic state, an external factor or intervention producing excessive forces must have occurred without proper handling. Then, because we describe cloth as a set of smaller mechanisms like springs or geometric conditions defined over triangles that interact with each other, it becomes very difficult for the cloth to handle every external factor and force instantly in a unified way. The system needs time for the information to spread from mechanism to mechanism; and, a number of time steps are required before every mechanism on the cloth can process a non-uniformly applied force, and before a complete, cloth-wide response can be given. Unlike real cloth that can answer to any number of forces regardless of their intensity or their location on its surface, the cloth models and their mechanisms can easily fail to prevent a node from passing through another and jeopardize the mesh structure. It then becomes necessary to handle forces of such intensity with special care.

We believe an excessive external force in the simulation that is not uniformly applied to all nodes can cause three different types of problems:

1. Super-elasticity or excessive elongation
2. Excessive repulsion forces
3. “Crossovers” and mesh incoherencies

In fact, when such an excessive force occurs on a cloth node, all three problems could occur simultaneously. Again, we represent cloth and its behaviors with nodes and the

mechanical relationships between these nodes. However, some of these relationships may have become too extreme or gone wrong following the fallout of an excessive force. We can categorize the relationships of a node with the rest of the cloth into two half-spaces and help clarify the problem.

Let us name the node A undergoing the excessive force  $F_A$ . Then, let  $F_A$  be used as a normal vector that would define the plane going through A. The 3D space is then divided into two half-spaces, in “front” of the plane and in the “back” of the plane. (Figure 7.28)

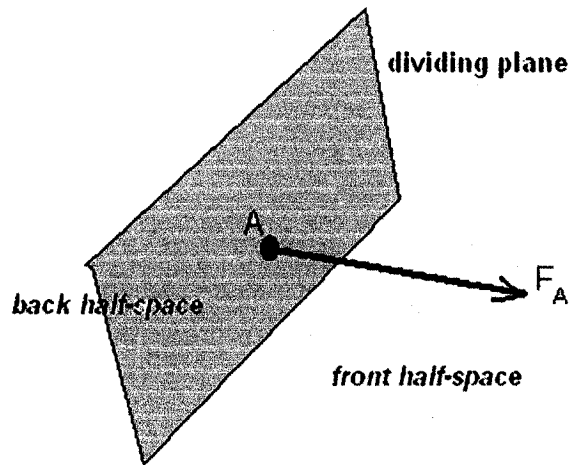


Figure 7.28. The two half-spaces.

At this point, we can also label these two half-spaces as the “Pull” half-space and “Push” half-space due to the relationships of A with the nodes in the back half-space; the nodes are likely characterized as being “pulled” and dragged along by A, whereas the nodes in the front half-space look like they are being pushed. The nodes on the dividing plane can also be understood as being pulled without loss of generality. Super-elasticity can

happen mostly amongst the nodes in the “Pull” half-space, while the two remaining problems occur in the “Push” half-space.

The first problem is therefore easily solved, even in the case where the forces are irregularly strong like  $F_A$ . We previously utilized a post-step corrective measure on the positions of nodes. The simulator component associated with this function was labeled “elongation rate limiter”, or simply “limiter”. Now, the only modification the limiter needs is the possibility to reiterate more than once per time step. Figure 7.29 shows that with the current definition of our limiter, a single correction pass would solve the elongation of segment CD but cause an elongation on segment BC. Consequently, super-elasticity would therefore persist and be displayed. Now, the lower part of Figure 7.29 also displays the result of applying subsequent correction passes within the same time step. Although a few additional steps still does not provide a fail-safe-solution, it does reduce the overall super-elasticity problem further. The number of times the limiter needs to operate within a same time step depends on the magnitude of the super-elasticity problem; so, it depends on the strength of the excessive force like the one applied on node D in Figure 7.29. To optimize the corrections, the limiter does not need to reiterate its entire work since it is easily observed that the newly created over-elongations are in close proximity to the previous ones. So, the subsequent passes can directly be performed in the neighborhood of the original super-elasticity occurrence.

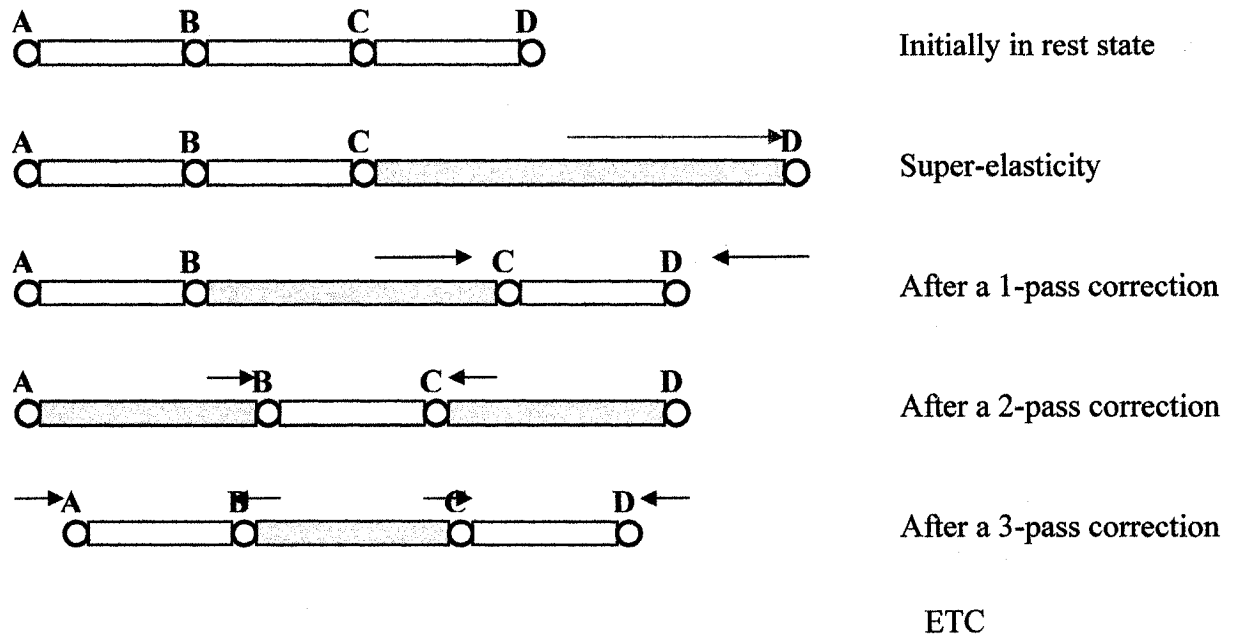


Figure 7.29. A multiple-pass elongation-rate limiter

Next, the two remaining types of problems occur in the pushing half-space. Excessively strong repulsion forces can be handled with the damping methods on internal forces, and the modeling of the cloth property described in 7.4 also helps reduce this problem. On the other hand, what worries us most is the occurrence of the third type of problems, “crossovers”, where obviously damping mechanisms appear to have been insufficient. So, we present an approach that focuses on treating this problem and the method can also be described as a force transfer mechanism.

Our approach processes external forces that are excessive. Although internal forces that are extreme can easily be treated in a similar way, we believe that it is preferable to treat them in conjunction with the damping of internal mechanisms.

The first step in this approach consists of determining the magnitude from which a force is characterized to be excessive. Let us name that variable by ‘c’. We can obtain the value of c by determining the minimum amount of change in position that would start

troubling us. It is obvious that a change in position greater than the natural distance separating two adjacent nodes would break the coherency of the mesh. But, the minimum value should be even smaller than this. We believe a change in position greater than half the natural length of a yarn spring would be the minimum value that could start causing trouble since two connected nodes moving in opposite directions for this distance can cross over one another within a single time step. Nonetheless, that is the extreme case and being overly precautionous. Once the worrisome change in position is set and the time step size is known to us, we can easily compute the corresponding force magnitude  $c$ .

Next, for each excessive force  $F_A$  such that  $F_A > c$ , we define a “cone” whose vertex is node A and spreads out in the “Push” half-space as illustrated in Figure 7.30. The center ray of this cone lies on the line through  $F_A$  and  $\theta$  is the angle from the center ray to a point on the surface of the conic surface.

The value of  $\theta$  is a function of the magnitude of  $F_A$ . Every other cloth node B such that  $B \neq A$  that falls within this conic volume is subject to a new force transferred from the effect of  $F_A$ . This new force is a function of at least two variables. First, it is inversely proportional to the distance separating A and B. Second, it depends on the angle  $\theta_B$  between the center ray and the line segment AB. The first reason behind considering such a conic volume is to identify nodes that are susceptible to suffer mesh incoherency due to the move of node A by the end of the current time step. The second justification of transmitting such forces to certain nodes is to provide an approximation to the reaction

of repulsive forces had the time step been reduced to prevent the incoherency. Without reducing the time step size, it is now possible to lessen the possibilities of a mesh incoherency and approximate the correct in-plane distortion that should have resulted on the cloth.

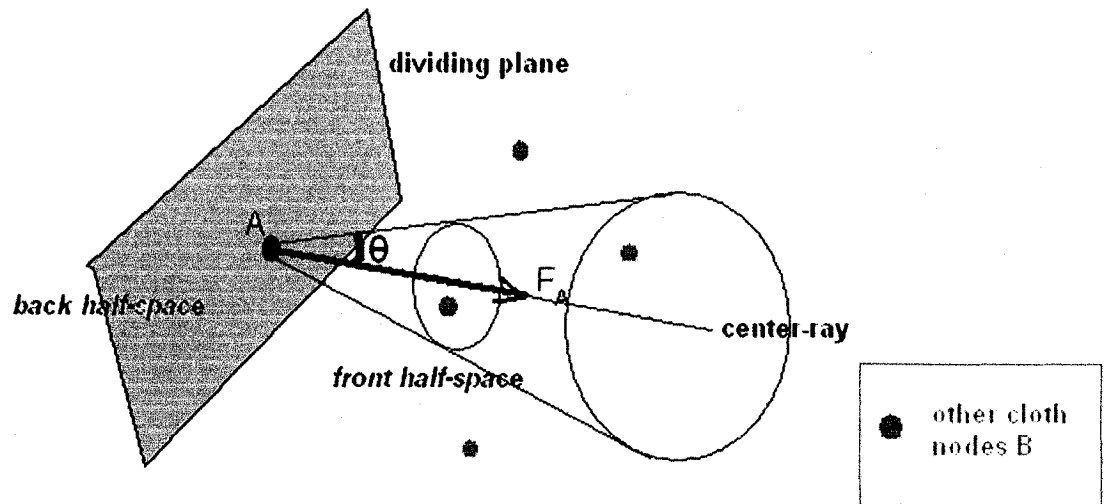


Figure 7.30. The force transfer cone.

There remain two concerns that the approach must address. First, although the external force  $F_A$  may be alarming, it is possible that once all the external and internal forces of this time step are gathered for node A, that the effective force applied on A is no longer of concern and that its magnitude has changed. Therefore, the detection of an excessive force such as  $F_A$  in the phase of external force determination will only alert the current new mechanism to remember to verify the forces on A just before the numerical integration step is taken. Let us name the effective final forces applied on A of this time step as  $F_A^*$ . Second, if a spring were defined on the segment AB and the length  $|AB|$  of the segment were smaller than the natural segment length  $L_{AB}^0$ , then repulsion forces



were already applied on pushing the node B away from A. And, the transfer of more forces from the effect of  $F_A^*$  to B would amplify the repulsion forces on B and increase the risk of obtaining excessively strong repulsion forces. We would be eliminating one problem but creating another. Likewise, if B were already moving the same way as  $F_A^*$  at an equal or greater velocity, it would be unlikely to undergo the pushing effect of  $F_A^*$ . Therefore, we need to determine the part of the forces  $F_B$  applied on B that already go in the direction of  $F_A^*$ ; then, we need to deduct that from the new force  $F_B^*$  that should be transferred to B by this mechanism. This can easily be done by computing the projection of  $F_B$  onto  $F_A^*$ . In the case, this deduction changes the direction of  $F_B^*$ , we simply set the net vector of  $F_B^*$  to be a null vector, and no effect of  $F_A^*$  is transferred to node B.

We summarize here the algorithm in point-form and pseudo-codes:

- The value of  $c$  is determined for this cloth simulation, time step and mesh size.
  - In the simulation loop
    - For each force  $F_A$  such that  $F_A > c$ , the index of node A is stored in a processing list.
    - The simulation proceeds to compute its forces as usual until the next step is the time integration, then we start the second part of this mechanism:
    - For each node A whose index figures in the processing list:
      - if( (excess= $|F_A^*| - c$ ) > 0) // if the force magnitude exceeds the  
 { // worrisome value, we continue the approach
      - The excess\_vector = ( $|F_A^*| - c$ )  $\times$   $F_A^*$ .unit();
      - Let the value of  $\cos \theta$  of the cone be  $w = (1 - 0.1)^{(\text{excess}/c)}$ ,
      - //The next step consists of identifying the nodes that fall within  
 // this cone
- For each node B:**
- if( ( $\cos \theta_B = \text{AB} \cdot F_A^*$ ) >  $w$ ) // we also calculate  $\cos \theta_B$  in the

- ```

( // same step
• // For a non-zero length of segment AB, we compute the
// gross force transfer

$$\mathbf{F}_{\rightarrow B} = \frac{L_{AB}^0 \cos \theta_B}{|AB|} \times \text{excess\_vector};$$

• // next, we calculate the projection of  $\mathbf{F}_B$  on  $\mathbf{F}_A^*$ 

$$\text{Proj}_{(\mathbf{F}_A^*)} \mathbf{F}_B = \frac{\mathbf{F}_A^* \cdot \mathbf{F}_B}{\mathbf{F}_A^* \cdot \mathbf{F}_A^*} \mathbf{F}_A^*;$$

// if the deduction reversed the direction
• if(  $(\mathbf{F}_{\rightarrow B} = \mathbf{F}_{\rightarrow B} - \text{Proj}_{(\mathbf{F}_A^*)} \mathbf{F}_B) \cdot \mathbf{F}_{\rightarrow B} \leq 0$ )

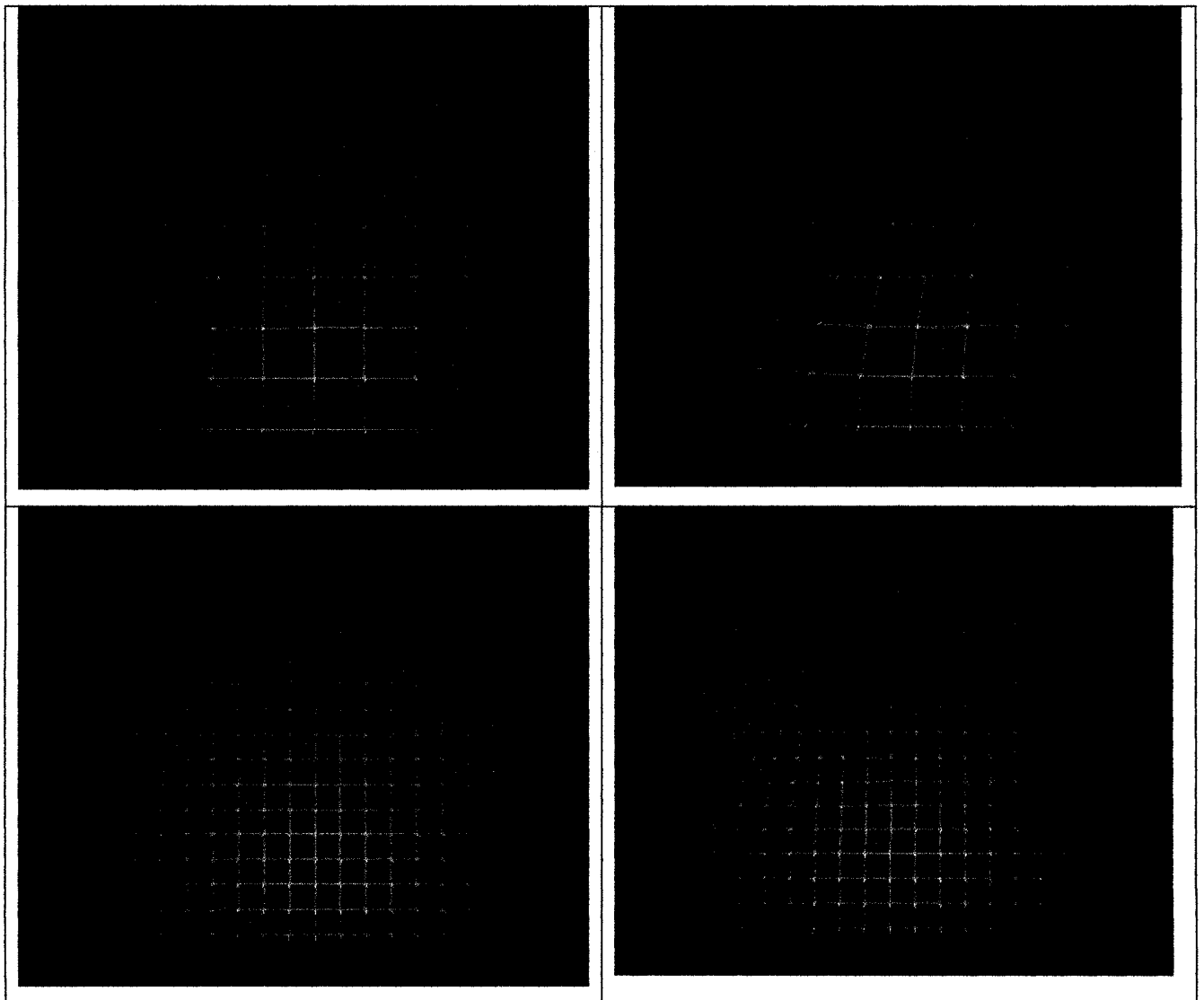
$$\mathbf{F}_{\rightarrow B} = (0,0,0);$$
 // there is no force to transfer
• // Next, we add this new force to B
B.Forces +=  $\mathbf{F}_{\rightarrow B}$ .
}
}

```
- Numerical time integration step.

(Algorithm 7.11)

We tested a simple case with this approach using the mass-spring simulator and two different mesh sizes. A relatively strong force would hit the leftmost node of the middle row in the mesh. The direction of the force would be towards the right and coplanar to the cloth surface. The force only hits that single node and is applied for a short duration of half a simulated second, just as if a rigid object hit the node from the left side very quickly. Figure 7.31 illustrates four instances of the test where the smaller of the two mesh sizes were used in the two top captured frames, and, for each row, the left frame represented the simulation without the force transfer while the right frame applied our new mechanism. In the left pictures, the node hit just goes through its right neighbor

and the mesh structure is unrealistic. Conversely, in the right pictures, the neighboring nodes are moved to conserve the mesh structure; and, the distortion on the cloth created by this approach represents a satisfactory approximation of how cloth would have behaved if the time step were made smaller in order for the internal mechanisms of the model to react properly. This approach reduces the occurrence of artifacts like those “crossovers” and helps maintain both the mesh coherency and the system stability. It allows the handling of excessive forces without changing the time step size.



*Figure 7.31. Testing the force transfer on two mesh sizes.*

### **7.3. Additional Collision Detection and Response Solutions**

The approach presented in Chapter 4 for collision detection and response is an accurate and complete method for dealing with both cloth-object collisions and cloth self-collision; yet, it requires solving a cubic equation for every pair of elements, point-triangle or edge-edge, which undergoes the collision detection test, and therefore increases the difficulty level for implementation. There exist different solutions that do not involve cubic equations and we considered three of them:

- An easy point-triangle collision detection method,
- a fast elimination-based test for triangle-triangle intersections,
- and the use of springs for collision prevention.

We analyzed and compared each of them versus Provot's approach in the hope of finding a suitable alternative that could avoid solving cubic equations.

Our studies could be summarized as follows:

- The first alternative method only deals with collisions between points and triangles. The type of collision between two edges is not covered by this approach. The method is simple though: it starts by determining whether a point has crossed a triangle's plane during the time step or not. Then, if the point did, the method continues by computing the intersection between the point's trajectory and the triangle plane. Last, it performs an inclusion test between the intersection

point and the triangle. This first strategy is indeed straightforward and does not seem to require much computational complexity for the simulator if compared to Provot's method. However, it has two weaknesses. First, it can only deal with static triangles. In other words, the triangle tested for collision must be assumed to remain at the same position during the time interval. Second, because of the first limitation, inaccurate results can be produced during the collision response phase.

- The second alternative approach is apparently a well-known strategy to detect collisions between pairs of triangles. This strategy does not need to differentiate between two types of collisions like point-triangle or edge-edge like Provot's approach does. This strategy is said to be "fast-elimination based" because it rapidly eliminates cases where triangles cannot intersect and consequently reduces the number of cases to be tested as we proceed through the steps of the method. Nevertheless, this strategy can only deal with static triangles.
- Our third possible alternative was the use of springs. Using springs, collision detection could instead become collision prevention or avoidance. And, collision response could also be computed at the same time. Thus, hitting two birds with one stone. Also, we believe it would also be possible to simulate the repulsion forces occurring when cloth buckles up using springs. However, the use of springs is a considerable overhead addition to the cloth simulator and does not appear to be a very optimal solution.

In the end, we judged Provot's approach more complete than any of the three methods above. Besides, our "trick" described in section 6.1 allowed us to implement collision detection and handling without having to actually solve the cubic equations. So, we decided not to try any alternative collision detection method for our cloth simulator.

Sections 7.3.1 and 7.3.2 present our analyses of the first and second alternative, respectively, collision detection methods in more detail. The use of springs for collision detection and response is presented in the [Appendix B](#).

### **7.3.1. An Easy Point-Triangle Collision Detection Method**

The first alternative solution addresses only the type of collisions between points and triangles. The method will be presented first. It will seem relatively much simpler than Provot's approach; then, we will analyze the differences and weakness of this newly presented method compared to Provot's.

We start by identifying the position of our point  $P$  at the beginning of the time interval by  $P(t)$  and at the end of the interval with  $P(t+dt)$ . Then, we define the normal vector to the plane of the triangle with a vector  $n = (a, b, c)$  where for every vector point  $x$  and another point  $x_0$  on that plane the plane equation  $n \cdot (x - x_0) = 0$  will be true. We then rewrite the plane equation as  $ax + by + cz + d = 0$  where  $d$  can be calculated with  $d = -ax_0 - by_0 - cz_0$ .

The next step is to determine whether the particle or point P has crossed over the plane of the triangle. The reason to test this first is obvious. If the particle entered into collision with the triangle within the time interval, it would have also crossed the plane of the triangle or would have ended on the plane. To perform this test, we compute the signed distance from the plane for both  $P(t)$  and  $P(t+dt)$ . The signed distance gives the shortest distance from a point to a plane and its value is positive if the point were on the same side of the plane as the plane's normal vector. If the point were on the opposite side to the normal vector, the signed distance would be negative, and the value zero would simply mean that the point is on the plane. The signed distance is given with the following formula:

$$D = \frac{ax_0 + by_0 + cz_0 + d}{\sqrt{a^2 + b^2 + c^2}} \quad (\text{Equation 7.11})$$

The most expensive part to compute is the denominator, with squares and a square root. However, the denominator is always positive and does not affect the sign. So, in practice, you have to compute only  $D=ax_0+by_0+cz_0+d$ .

Now, let  $D_t$  be the signed distance from the plane for  $P(t)$  and  $D_{t+dt}$  be the one for  $P(t+dt)$ . There are three possible cases: First, the product  $D_t \times D_{t+dt} > 0$ , and that means P never crossed the plane of the triangle. Second,  $D_t \times D_{t+dt} < 0$ , and that tells us the point P did cross the plane within the time interval. Finally, if  $D_t \times D_{t+dt} = 0$ , either  $D_t=0$  or  $D_{t+dt}=0$ , or both are zero and the point P simply moved on the plane. If the second or third case happened, we proceed to the next step, else, P never intersected with the triangle.

The next and last step consists of testing whether the intersection point, let us name it  $Q$ , between the line  $P(t)$  to  $P(t)$  and the plane is actually within the triangle in question. For the second case, we simply substitute the line equation of  $P$  into the plane equation. For the third case,  $Q$  was either  $P(t)$  or  $P(t+dt)$  whichever was at a signed distance of zero. For the special case where both were on the plane, both  $P(t)$  and  $P(t+dt)$  will be tested for inclusion within the triangle. We suggest the following point-triangle inclusion test. Its assumption is that the point is coplanar to the triangle; a condition already verified at this point. Let us now identify the three vertices of the triangle as  $A$ ,  $B$ , and  $C$ , given in a counter-clockwise order. Also define the vectors such as  $AB$  meaning the vector from  $A$  to  $B$  or  $B-A$ . Now the trick is as follows: Since we adopt the counter-clock wise definition for specifying the triangle vertices, the cross-product of the two vectors  $AB$  and  $AC$ ,  $AB \times AC$  defined a normal vector pointing out of the surface, and the plane's normal vector  $n$  would therefore also point out the same way. Then, we also know that two parallel vectors pointing in the same direction will have their dot product greater or equal to zero. Given this knowledge, we compute the cross-product  $AB \times AQ$ . That test alone tells us on which side of the line of vector  $AB$  the point  $Q$  is. If  $Q$  were on the inside of the triangle, the previous cross-product would yield a vector perpendicular to both  $AB$  and  $AQ$  that would point in the same way as the normal vector to the surface triangle. That is, the dot product between  $AB \times AQ$  and the normal vector  $n$  would be greater or equal to zero. Otherwise,  $Q$  would be outside of the triangle. That was the test versus the triangle edge on  $AB$ , we perform the same test for the remaining two edges, and if all three tests are satisfied, it would then indicate that  $Q$  is included in the triangle. To sum up, the point-triangle inclusion test is:



Q is included in the triangle ABC if

$$[(B-A) \times (Q-A)] \cdot n \geq 0 \quad \text{AND}$$

$$[(C-B) \times (Q-B)] \cdot n \geq 0 \quad \text{AND}$$

$$[(A-C) \times (Q-C)] \cdot n \geq 0 \quad .$$

*(Method 7.12)*

The most obvious difference we can observe between this method and Provot's is that there is no cubic equation to solve in this one. Unlike Provot's approach to collision detection, this method does not seek to find the exact time of possible collisions between points and triangles. However, that is the facet that makes Provot's approach more complete than methods like the current one. Let us recall once again that Provot's approach describes a cloth node as a point moving in 3D space during the time interval. Similarly, a triangle is also identified as a surface moving in time. The positions of both elements are described as functions of the time variable. So, the first result Provot's method sought is a time value within the interval, if any, when the point's linear trajectory and the triangle's plane would intersect; and, so on. With the exact time of collision detected, the corresponding states of both colliding objects at that very moment as well as their point of contact can be obtained precisely.

In contrast, this first alternative method only makes reference to the cloth triangle without the time variable to describe it. Only the point is described in time. In fact, we chose to select the position of the triangle at the end of the time interval for reference; this choice will be justified shortly. Consequently, this method can only guarantee an accurate treatment of collision detection between points and static triangles. But, it is observable that numerous other solutions for the topic of cloth animation also treat the problem using static triangles. Two explanations can justify this choice.

First, if a collision truly happened within the time interval but not at the beginning nor the end, the new method would fail to report this event; however, due to the structural nature of the object involved, a cloth mesh, it is very likely that the time interval ended with a collision between the same point and a different triangle nearby. With the exception of triangles located on the sides of the cloth, triangles are surrounded by their adjacent neighbors. Thus, if a particle went through the surface of a triangle, it is very likely that its trajectory intersected a nearby triangle by the end of the time interval.

Second, if the positions of static triangles were chosen at the end of the time interval as suggested, every intersection between points and triangles existing at that moment will be detected without exception and be resolved before the display phase. Nevertheless, there is a concern with the response phase where the appropriate collision response is calculated with the information obtained from the detection phase. Yet, it is then possible that the actual first triangle that collided with the point was not detected; instead, a neighboring triangle and its information were used because it was involved in the later collision at the end of the interval. Ultimately, the worry with methods using static triangles is that the simulation accuracy is reduced in the collision response. Fortunately, given that the time step size is not too large, this inaccuracy remains bearable in exchange for a simpler collision detection method.

### **7.3.2. The Fast Elimination-based Test for Triangle-triangle Intersection**

This second solution presented handles the collision detection between pairs of triangles in 3D space by rapidly eliminating the triangles that do not intersect. The

method does not process separately whether the collision is of type point-triangle or edge-edge; a same approach is taken for every pair of triangles, and the detection is performed at a specific time value, generally, the end of the time interval. Hence, this approach also refers to static triangles, but again, for the same reasons explained in the previous alternative solution; it becomes acceptable to use the positions of the triangles at the end of the time interval as reference. This approach is apparently well-known; so, we only present its main steps.

The approach starts by considering the two planes, identified as  $\Pi_1$  and  $\Pi_2$ , defined respectively by the first and second triangle. Depending on the relative position of one triangle versus the other, three general cases can be identified.

First, both the triangles lay on the same plane i.e.  $\Pi_1 = \Pi_2$ . Then, the case becomes verifying whether an in-plane intersection has occurred. Two coplanar triangles intersect if at least one edge from the first intersected an edge of the second, or if one triangle were contained within the other. But, instead of verifying these conditions, it would be more efficient to consider that the coplanarity condition implies that if two triangles intersected, at least one vertex from one of the triangles lie within the other triangle. Then, the worst possible case would be to perform six inclusion tests.

The second possible situation is that all three vertices of one triangle lie on the same side of the plane defined by the other triangle. In which case, the triangles do not intersect.

Conversely, the remaining situation is that the vertices of one triangle lie on different sides of the other triangle's plane. In this case, collision is possible and we must proceed to the next step. We compute the line intersection of the two planes  $\Pi_1$  and  $\Pi_2$ .

Then, we come back to each of the individual plane and compute the intersection of this line versus the corresponding triangle. For each triangle, the resulting intersection can be a point, a line segment, or nothing. As long as none of the two results is empty, we perform a last intersection test with them. If the two results in turn intersected or overlapped, the triangles intersected as well.

#### **7.4. The 4th cloth mechanical property: Proximity induced curvatures**

In this section, we discuss the regular and basic behavior of fabrics that have not been explicitly represented by previous cloth models. We will extend the previous models with the design of a method to simulate this behavior.

The particular basic cloth behavior we want to simulate is the formation of curves and folds when parts of a cloth are brought close together. This behavior is indeed a more general form of the buckling up phenomenon of cloth since the manifestation of this behavior starts as soon as two parts of cloth are closer than their natural flat distance.

To extend the previous cloth models and simulate this specific behavior, we divided the challenge into a series of analytical steps that allowed us to devise a re-formulation of the past cloth models and upgrade our cloth simulators.

The steps can be summarized as follows:

- We started by raising some observations based on comparing the two models we mainly referred to versus the behavior of real cloth under particular circumstances where real cloth produces curves while the two models do not. We also noted that continuum finite-element models did not encounter this difficulty thanks to their energy minimization strategy, and we based our approach on this principle.

- Next, we studied the behavior more thoroughly before trying to formulate any mechanism to simulate it. Upon closer analysis, this particular cloth behavior is strongly related to the bend property of cloth. In fact, it can be understood as a complement of the bend property described previously by past cloth models. This behavior constructs a bridging relationship between in-plane forces and out-of-plane cloth motions. Subsequently, we then explained some prerequisite characteristics to describe this behavior and presented a first general formulation that would extend a mass-spring cloth model with the modeling of this behavior.
- We then explained how to determine the displacement direction along the normal vector axis of a node by constructing a rule based on the dihedral angle at that node and we labeled that rule the *Theta* rule.
- Then, we discussed some improvements related to four common misconceptions brought along with previous cloth models in general. We also suggested differentiating between two “types” of energy involved with cloth.
- In the following step, we improved our formulation and justified it using the energy minimization principle applied in the vicinity of a mechanism. We also showed a set of scenarios to prove that our new formulation minimizes the compression energy better.
- In this next to last step, we presented the method to obtain the two important coefficients of our new formulation  $k_1$  and  $k_2$  such that the current mechanism is guaranteed not to produce more stretch energy than its previous formulation did.

- Finally, we summarized the benefits and possible weaknesses of our new solution. And, we also explained again the necessity of representing this behavior for correct cloth simulation.

We extended the implementation of our simulator with this new formulation and the results were visually appealing and more accurate of real cloth.

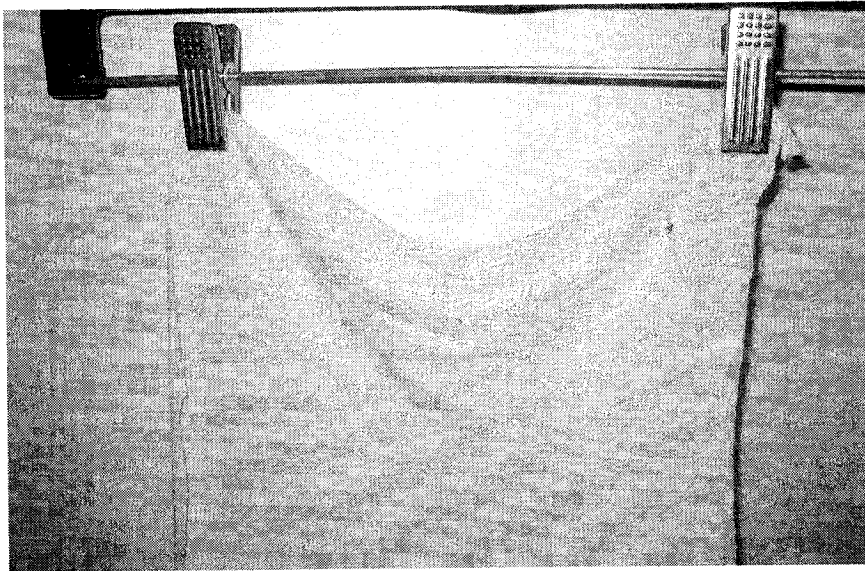
The following sections 7.4.1 to 7.4.7 discuss each of these steps in detail.

#### **7.4.1. The missing cloth property**

Many particle-based models often show associated implementation pictures with very nice 3D curvatures on the piece of cloth simulated. These curves can easily be explained given the circumstances in which the piece of cloth is simulated. The simulations often involve draping the piece of fabric over rigid objects where the handling of collisions and treatment of internal forces suffice to produce nice curves in three dimensions. Alternatively, the curves are results to direct external forces or manipulations such as wind acting on the cloth's particles in an out-of-plane fashion that would provoke the reaction of internal mechanisms.

We have alleged that the previous cloth models do treat bending resistance; however, it is not easy to identify how their formulation produces curves. In fact, this characteristic seems to have been left out or neglected by many models. Given the simple case that two extrema of the cloth are brought closer together, in real life, we

know that to produce curves like in Figure 7.32. But if the condition functions were taken as described without modification, then all that results from the models' formulations are in-plane repulsion forces, but none of the out-of-plane movements observed with real cloth. This observation is true for both of the models we presented earlier. Figure 7.33 illustrates once again an instance of a cloth simulation that does not take into account this property.



*Figure 7.32. A real piece of cloth hung with its two top corners closer together than their "natural" flat distance.*

Yet, energy models did not need to specifically treat this cloth property because their pursuit of the lowest energy state for the cloth particles already dealt with the concern. That is, with a state where nodes are in close proximity, repulsion energies are first induced, and then, the act of iteratively seeking particle positions to minimize the overall energy would eventually lead to an out-of-plane solution. The energy minimization principle is sound and practical for solving this issue. We will borrow this



idea from the continuum models as a guideline in the formulation of our solution, yet, without truly shifting the current model to an energy model.

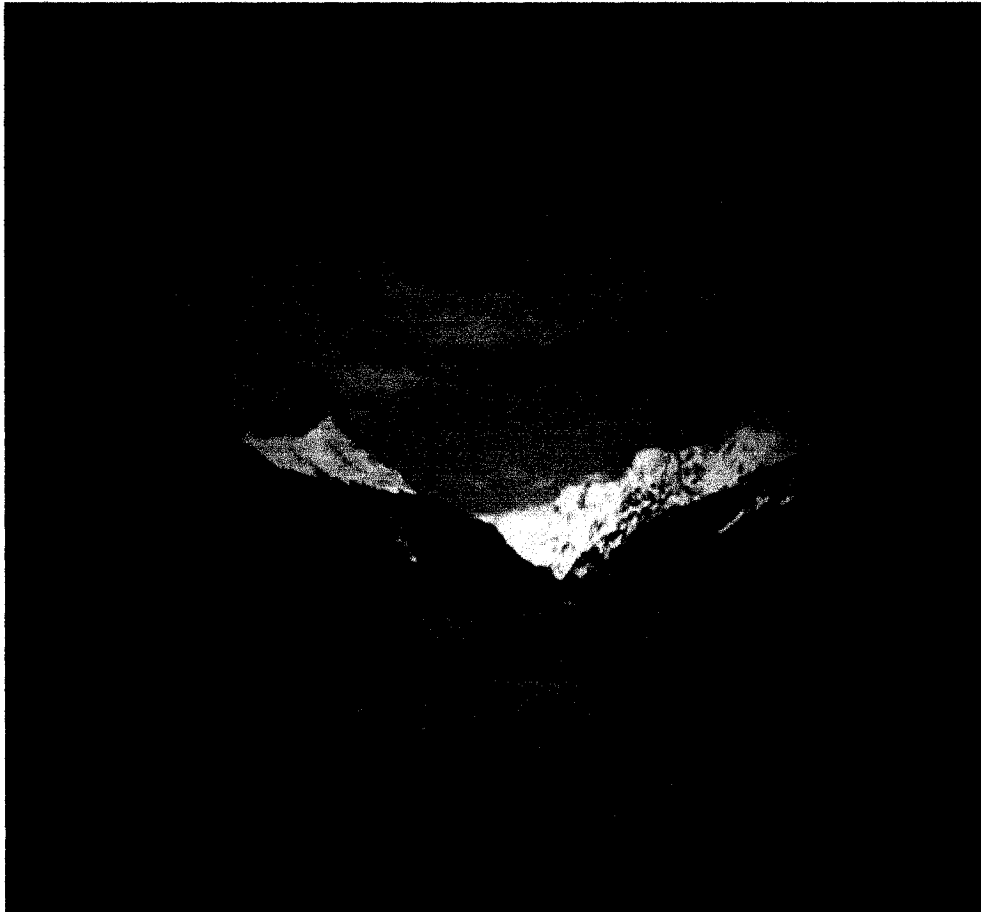
The earlier descriptions of bending resistance do not suffice. We believe that a cloth model cannot be “complete” without modeling this additional behavior of fabrics. Thus, we suggest a new formulation for this behavior we describe as *proximity induced curvatures*.

#### **7.4.2. Studying and modeling the behavior**

To extend the previous models with the simulation of this behavior, it now becomes necessary to formulate new conditions and redefine the internal mechanisms; consequently, we may first perceive this newly described behavior as a distinct and fourth property of cloth. However, on a closer look, the relationship of this behavior and the bending resistance described by previous model is such that the newly observed behavior is simply another manifestation of the bending property. We also believe that it will be easier to relate the fabric’s thickness, weight, rigidity and the bending resistance through this behavior than it was with the previous formulation. Even so, the treatment of this behavior does not replace the past formulations of bend resistance we studied before; it simply complements the overall description of cloth’s bending resistance better.

We observe that this behavior we refer to as “proximity induced curves” is a key feature that would allow us to tie a relationship between in-plane forces and out-of-plane motions of fabrics. And, it consists of an attempt to properly model and explain the

formation of curves due to in-plane compression of yarns. Actually, the word “compression” would still be inappropriate since cloth avoids compression by exhibiting the behavior that we discussing. Perhaps the qualifying words “proximity-induced” for the curves would be more self-explanatory.



*Figure 7.33. An instance of cloth simulation without realistic out-of-plane motions.*

We choose to describe this behavior based on a mass-spring model; however, it is clearly possible to extend the description to any model treating in-plane forces as a function of the length of the yarn segments between cloth nodes.

To start, let A and B be the two nodes connected via an in-plane property spring AB. When the length of AB is greater of equal to its natural length, the spring just exerts

its usual pulling behavior to resist stretching. But, when the current spring length is shorter than its natural length, instead of simply applying a pushing behavior to repel the two nodes, we insert an additional out-of-plane motion. It would be preferable that the new motion does not induce any additional compression to neighboring springs, and ought to have minimized consequences on the in-plane forces. Thus, we suggest setting the new out-of-plane motion along the axis of the node's normal vector. We will justify this choice later. Previously, the force produced by a shortened spring on its two nodes could be summarized as:

$$\begin{aligned} F_{\text{spring}} &= k_{\text{stiffness}} \times \text{direction\_vector\_along\_the\_spring} \times \text{compression\_rate} \\ &= k_{\text{stiffness}} \times \text{repulsion\_vector}. \end{aligned}$$

Now, we modify the formulation for a shortened spring into a two-part force description:

$$F_{\text{spring}} = k_{\text{stiffness}} \times \text{repulsion\_vector} + k_{\text{stiffness}} \times \text{Displacement\_rate} \times \text{Normal\_vector}.$$

Let  $L_{AB}^0$  denote the natural length of the spring AB,  $L_{AB}$ , its current length, and  $N$  be the “scaled” normal vector of node A; then, the force applied on a node A by its shortened spring AB is now expressed with:

$$F_{AB \rightarrow A} = k_1 \times k_{\text{stiffness}} \times \left[ (B-A) \times \left( 1 - \frac{L_{AB}^0}{L_{AB}} \right) \right] + k_2 \times k_{\text{stiffness}} \times \left[ \left( \frac{L_{AB}^0}{L_{AB}} - 1 \right) \times N \right]$$

*(Equation 7.13)*

where  $k_1$  and  $k_2$  are coefficients used to express the fabric's bending resistance property that we will describe shortly.

First, we specified the qualifier “scaled” for the vertex’s normal vector  $N$  because unlike the directional vector  $(B-A)$ , the normal vector of a vertex may not have been set to be automatically adjusted with the mesh resolution. Consequently, the new formulation would behave dependently on the number of nodes used to model the cloth; a result that would have not been satisfactory.

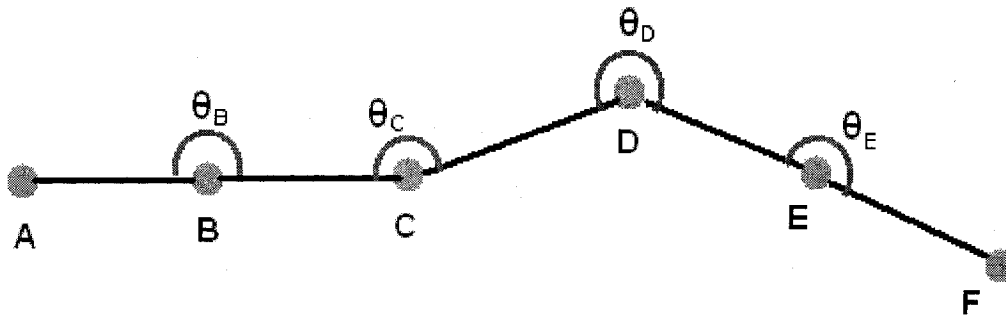
Obviously,  $k_1$  determines the relative amount of “push-back” and,  $k_2$ , the relative amount of “curve-up” the cloth reacts as a response to in-plane compression. Again, real cloth is not permissive versus the compression of its threads; the fabric will rather show a behavior where curves and eventually folds will be formed in order to maintain the length of its yarns. In our implementations, we like to refer to this whole reaction of cloth as the “push and curve.” The determination of  $k_1$  and  $k_2$  is crucial for our new formulation because of its direct impact on the perceivable bending resistance the cloth demonstrates. Our first experiments led us to believe that  $k_1$  and  $k_2$  are real numbers within the interval  $[0,1]$  and that their governing condition is  $k_1+k_2=1$ ; both these two assumptions produced visually appealing results, although this formulation was not justified. In truth, both these assumptions will be replaced with a more justifiable approach later on. After several more experimentations with real cloth, we observed that thicker fabrics allow fewer curves under this behavior and tend to push back more. And, heavier cloth was harder to curve or to fold; but, once curves or folds had been formed, they were tougher to undo with the cloth’s internal forces; an observation that remains nonetheless consistent with the previous models. Again, we may be prompted to directly associate the values of  $k_1$  and  $k_2$  with the cloth’s thickness, weight or even rigidity factors, but there is a better approach.

But, before we present this approach, we first need to first explain how we determine the displacement direction along the normal vector axis of the node.

### 7.4.3. The Theta rule

First, let us recall that the normal vector defined at a cloth node or mesh vertex, is the average of all the normal vectors of triangles adjacent to the node. Then, the next step is to determine if the node should move “up” in the direction of the normal vector or move “down” in the opposite direction. We define a rule based on the angle  $\theta$  of the yarn’s curve located at the node, and name that rule the “theta rule” for the purpose of easier reference. The angle  $\theta$  is calculated on the yarn where the spring is shortened. (Figure 7.28) But, for shear springs, we simply refer to the consecutive shear edges along a same direction because there are no shear yarns. The theta rule works as follows:

- If  $\theta = \pi$ , like with the nodes B or E in Figure 7.28, we arbitrary let the curving displacement direction vector of the node to be the normal vector N.
- Else if  $\theta > \pi$  as with node D, we also make it N.
- Else,  $\theta < \pi$  like in the case of node C, and the curving displacement direction will be -N.



*Figure 7.34. The angle theta of nodes on a same yarn or on a set of consecutive shear springs.*

One thing is certain with real cloth, softer fabrics curve more easily, whereas a more rigid fabric produces sharper curves. Combined with the thickness characteristic, we also observe under the same rigidity level that a thicker fabric will produce longer curves. All these different behaviors show the complexity of the bending resistance property of fabrics. Up to this point, there are at least the sharpness and the length of the curves to deal with, two distinct bend characteristics that can even produce additional combinations. Fortunately, with this new theta-rule in hand, we can at least gain control over the accentuation speed of the curves formed. We can now simulate a fabric that yields sharper, more accentuated and shorter curves; or contrarily, a fabric that allows less accentuated and therefore larger curves. By larger, we do not necessarily mean the amplitude of the curve but rather its span over a larger cloth surface, that is, longer. To model this curve span characteristic, we observe that the strict application of the theta-rule in fact accelerated the accentuation of the curves formed by proximity. Conversely, if a node were to communicate to its surrounding neighbors the result of its own theta-rule test, and made the neighboring nodes choose their curving displacement direction based on that decision instead of their own test result, the outcome is that the curves

formed on the simulated cloth are much less accentuated, and overall, fewer but larger curves are formed. The nodes closest to the sources of external forces are selected to start this overriding procedure of the previously established theta-rule. And, the greater the ranges of this overriding procedure, the longer the curves formed. In other words, that is also how far from the start of each curve forming the fabric will try to reduce the accentuation.

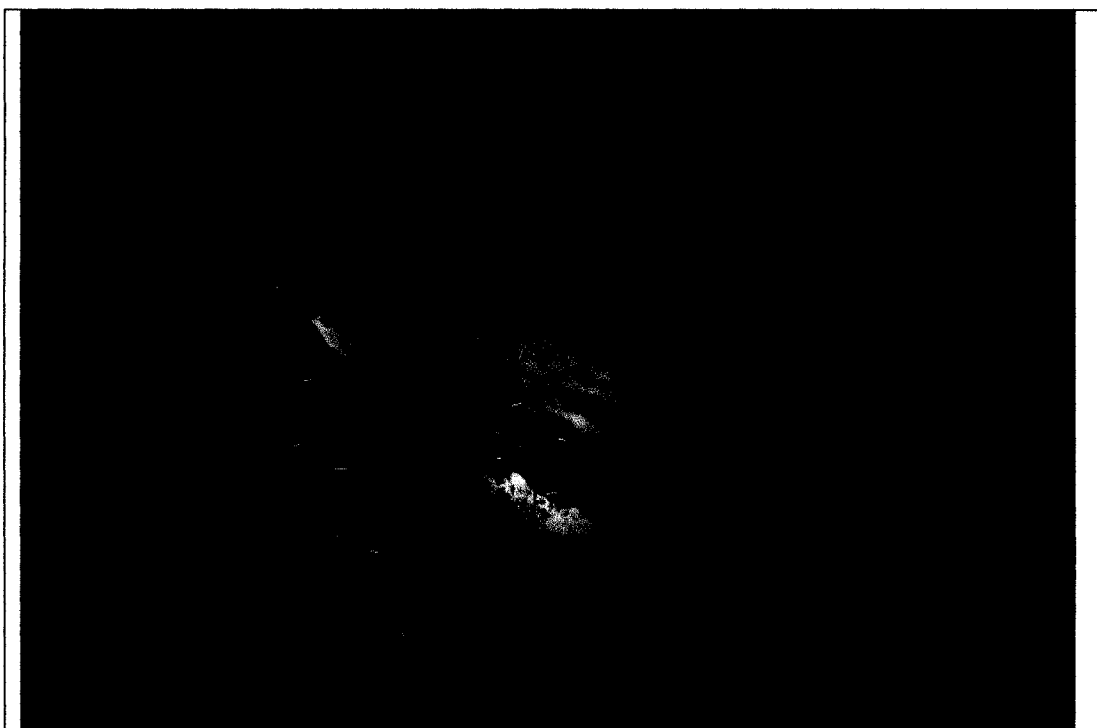
To test this hypothesis, we implemented the new formulation along with the theta rule option on top of our previous simulators and performed a few tests. Figure 7.35 shows the instance of curves formed on a hanging piece of cloth with a strict application of the theta-rule while figure 7.36 shows the same piece of cloth hanging but with less curve accentuation. Figures 7.37 and 7.38 show the same comparison but for a finer mesh piece of cloth. With these two later figures, we can also observe that the application of the theta-rule to accentuate curves will form more but smaller curves. Consequently, we can conclude that the strict application of the theta rule will accentuate curves faster and produce smaller but more curves if tested on a decent mesh size. The mesh size for Figures 7.37 and 7.38 were of 16 nodes by 16 nodes. Conversely, relaxing the application of the theta-rule yields a smaller amount curves and less pronounced curves.

By playing with the theta-rule, we were able to control the accentuation of the curves formed by proximity; however, the softness or rigidity characteristics of fabrics remain unclearly represented so far. Despite the fact that we have not yet been able to incorporate the simulation of fabric softness or rigidity into the current formulation, we

found out that manipulating the computation of normal vectors at vertices could provide us in some form what we desired.

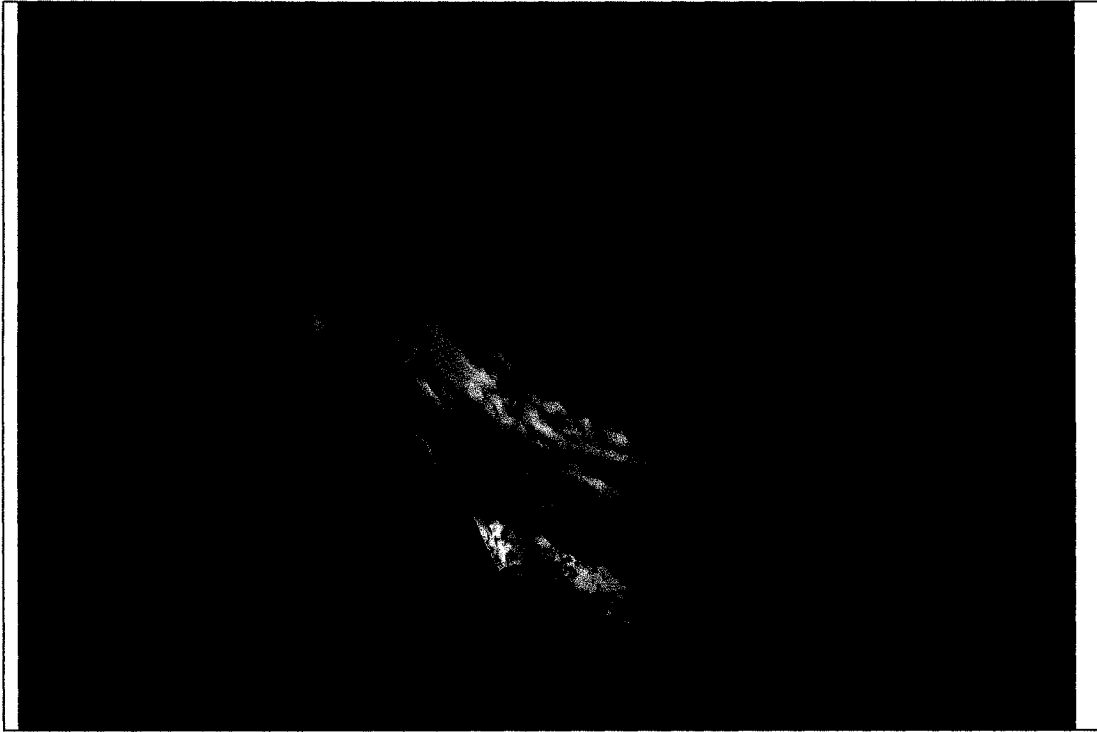


*Figure 7.35. The strict application of the theta rule.*



*Figure 7.36. Less accentuated curves.*





*Figure 7.37. The strict application of the theta rule over a finer cloth mesh object.*



*Figure 7.38. Less accentuated curvature for a finer cloth mesh object.*

#### **7.4.4. Energy Minimization & the four improvements**

In order to construct a better cloth model, there are at least four improvements to be made overall. We believe the previous models brought along four misconceptions by lack of specifications for behaviors such as the one we have been dealing with in this section. We will deal with these four common “mistakes” and be able to improve our model even further.

##### **Going beyond a one-to-one association**

The first false impression is to strictly associate a one-to-one correspondence between a mechanism, such as a condition function or a spring, and a single cloth property. The labeling of mechanisms can partly explain this possible misleading association. We are convinced that this wrong impression must be erased completely, and that a cloth model designer must be able to consider that various parts of a single mechanism can be used to simulate more than one property, and vice-versa, a cloth property can require several mechanisms to simulate it accurately. Taking for example the current case where the mechanism used to model cloth stretch resistance can be divided into two parts: the “push” and the “pull”. The “pull” behavior was appropriate to model stretch resistance; however, also associating the “push” behavior to this same cloth property was a grave mistake. We strongly suggest re-associating the “push” to the bend property. The justification is simple, a compressive force must have happened prior to obtain a “push” reaction; and, when such a force is exerted on real cloth, the cloth will be displaced linearly but will also be curving. The curving magnitude and specifics depend

on the bend property of the fabric, but it becomes clear that the reaction of curving or not characterizes the bend resistance. The “push” then becomes the counterpart of the curve formation resulting from the bend resistance property.

### **Different formulations for different parts**

Similarly, the second mistake is to utilize an identical formulation and the same stiffness coefficient for two distinct parts of a mechanism. Now, knowing that the “pull” and the “push” belong to different cloth properties, it is no longer appropriate to keep the previous formulation. The addition of the  $k_1$  and  $k_2$  coefficients as well as the out-of-plane motion are the new design we suggest. In fact, this change will satisfy our past doubt about using spring-like mechanisms to simulate cloth. The limitation in degree of simulation realism the linear springs were giving the model is undone with this new perception.

### **Reducing interference and overlap between mechanisms**

A third problem the previous models often had was to allow interference and overlap between their mechanisms. In the comparison of our two simulators, we previously stated that the mass-spring model had a bend resistance mechanism, namely flexion springs that interfered with the mechanisms of other cloth properties. Interference from one mechanism on the action of another can make the result less predictable; and, more importantly, such an overlap represents a redundancy problem.

Currently, both the referenced models show an overlap in the actions of their mechanisms. Let us take the simple example where three consecutive nodes A, B, and C on a straight yarn with the node B closer to A than C. Then, the current state would be an elongation of the edge BC and a shortened AB edge. The current mechanisms would fight compression on AB and fight stretch on BC. As a result, the mechanism on AB pushes B towards C, and the mechanism on BC pulls B towards C. None of the two mechanisms performed wrong; but, on an overall scale, the result is twice the reaction needed on node B: thus, that is an excess of forces applied on B that could cause oscillations later on. The redundancy problem may not have been obvious previously because a common reflex for a cloth model designer is to either decrease the stiffness coefficient, or to add damping; both solutions were subconsciously an imperfect approach to consider this overlap of the action of mechanism AB on mechanism BC. Instead, minimizing interference and redundancy would render the outcome of each mechanism more clearly, and we will show after the fourth concern is discussed that our new strategy does reduce overlaps between mechanisms.

### **Disregarding the mesh resolution limit**

The fourth suggestion we recommend is to not limit the modeling to the mesh resolution and distinguish the physical model from the graphical display. Clearly, the mesh resolution limits what we can display. However, let us remember that nodes are simply sample points of the cloth surface, and that the formulation and the modeling are for the real cloth object we want to simulate and not for simply a collection of nodes and springs.

Getting out of this misleading perception is easier said than done. Yet, there exists adaptive mesh techniques like the one described in Villard and Bourouchaki's work [VIL02] to overcome this limitation. So again, our point is: a cloth behavior modeling formulation ought to be based on the observed behaviors of real cloth only, and not be based on the limitation brought by the mesh resolution. The model should be assumed to model correctly a finer mesh resolution.

### **Two Types of Energy**

The next improvement we discuss does not exactly address a misconception. We simply suggest this hypothesis we believe would render the modeling of the bend resistance and curve formation easier to explain. We know from continuum models that the cloth seeks to minimize its energy by exhibiting the behaviors we observe. Concerning the in-plane energy, we suggest to clearly differentiate between compression-caused energy, or simply *compression energy*, and *stretch induced energy*. The reason we want to make this distinction is explained by the apparent dissimilar behaviors real cloth shows versus each of these types of energy. Cloth seems to tolerate the stretch energy more than it does for the compression energy. We easily observe stretch on real cloth, but we scarcely if ever see real compression with cloth. Actually, the compression state does occur on cloth with the yarn crossings sliding on the thread scale; however, it is not sustained long enough for the eye to perceive it with ease because the cloth will curve and fold in response to the compression energy it cannot keep. On the other hand, stretch can occur and be maintained as long as the cloth does

not tear. This observation made us construct the hypothesis that cloth tends to solve the compression with much more urgency than it does for the stretch energy. Or seen otherwise, cloth possesses the means to minimize all the compression energy on its surface while it may or may not always be able to resolve the stretch energy.

#### **7.4.5. A Better Compression Energy Minimization**

We still require a better way to obtain the values of  $k_1$  and  $k_2$ . If applied specifically to compression energy, the energy minimization idea from continuum models can explain the next part of our strategy. If we now assumed that the cloth as a whole always reacts so that the energy of its mechanisms, springs or internal forces, is minimized; then, on a local scale of a single mechanism and under the condition of shortened edge length, each mechanism will no longer simply try to minimize its energy alone, but will also act such that the energy in its vicinity remains minimal. If we labeled the original spring formulation as “Push only” and our new formulation as “Push and curve”, then while “Push only” minimized the energy caused by compression on the current mechanism by restoring the edge length, it did not address its impact on the surrounding mechanisms. On the other hand, “Push and curve” also performs the same minimization, but it will also reduce the probabilities of creating compression energy on the adjacent mechanisms by moving out-of-plane. And, the less in-plane interfering direction a vertex can choose for its neighborhood is its own normal vector direction.

Again, we only concern ourselves with compression energy at the moment. It is also obvious for our models that compression energy is proportional to the difference in length between a shortened edge and its natural length.

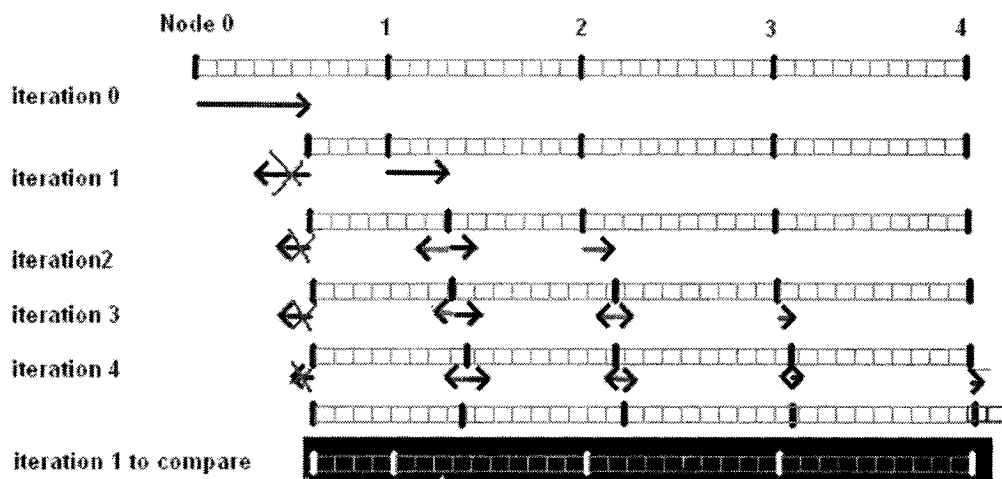
Now, let us define scenarios to illustrate the relative benefits of the new “Push and Curve” redesign over the original formulation of “Push only”. For each scenario, we will then compare a pair of 2D drawings illustrating respectively the two formulations on a yarn level to keep the visual comparison simple. The comparison could obviously be extended in 3D by reconnecting adjacent nodes and recreating the cloth surface; however, too many additional scenarios will have to be treated and each them will just be a more complex combination of single-yarn cases illustrated hereafter. We label each scenario in the following manner: *A.B.C.D* where each of these capital letters is a binary value whose meaning is shown in the matrix below.

|          | <u>0</u>                                                                                                                        | <u>1</u>                                                                                         |
|----------|---------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------|
| <b>A</b> | The compression is created by (equal) forces exerted from both sides of a same mechanism                                        | The compression is created by forces exerted only on one side of a mechanism.                    |
| <b>B</b> | The node(s) undergoing the initial compressive force is not constrained.                                                        | The node(s) undergoing the initial compressive force is constrained in position, that is, fixed. |
| <b>C</b> | The first dihedral angle encountered along the yarn is not 180 degrees. In other words, this yarn did not start fully straight. | The yarn started completely straight. All angles between edges were of 180 degrees.              |
| <b>D</b> | A force of opposite direction is exerted in the vicinity on the fourth node to the right.                                       | No opposite force is exerted in the vicinity.                                                    |

There are a total of fourteen possible scenarios with this labeling scheme. We selected to make only five out of fourteen of the comparisons to illustrate our point. Each of these drawings is a brief and approximate illustration yet sufficient to represent how each formulation behaves in different cases.

**Scenario 1.1.1.1**

Original in-plane formulation



*Figure 7.39. Scenario 1.1.1.1 : the original formulation*



## New formulation

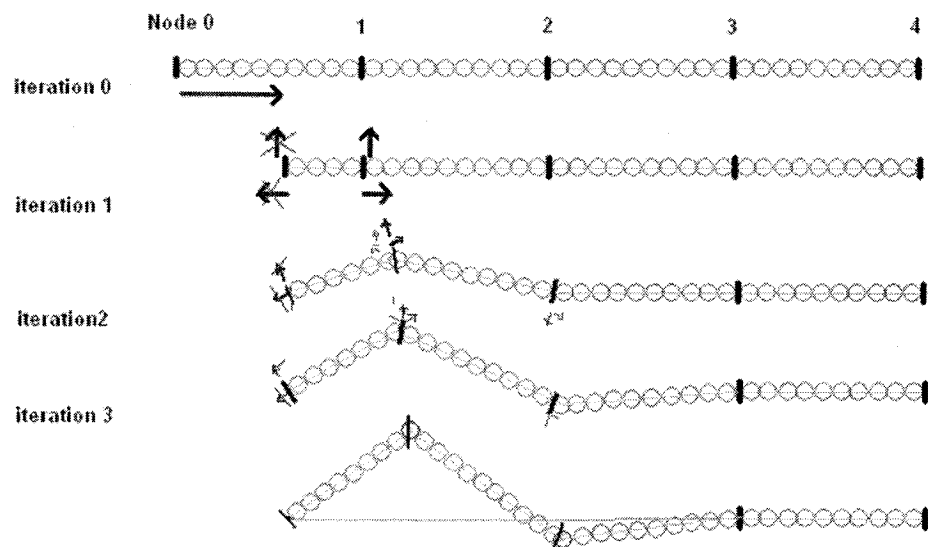


Figure 7.40. Scenario 1.1.1.1 : the new formulation

## Comparison for 1.1.1.1:

The original formulation does not even reduce the compression energy until the fourth iteration; all it does is transfer this energy from one mechanism to the next and the process takes four iterations. Also, the amount of compression energy solved on the fourth iteration is only 1/16 or 6.25% of the original compression energy created. The total yarn segment length is still short by 0.5625 of a natural edge length after four iterations. Thus, this is way too slow. Conversely, the new formulation solved most (over 80%) of the compression energy by the fourth iteration. And, although its first iteration did not solve much of the compression, it has already solved as much as the original formulation could have done by iteration 4. So, this scenario shows that the new formulation started reducing the compression energy earlier than the previous formulation and also reduced the compression faster.

### Scenario 1.0.1.1

#### Original in-plane formulation

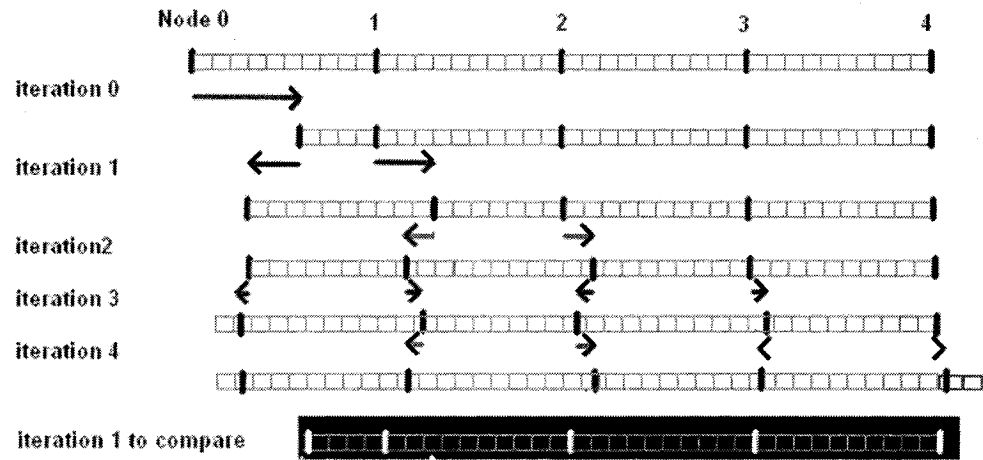


Figure 7.41. Scenario 1.0.1.1 : the original formulation

#### New formulation

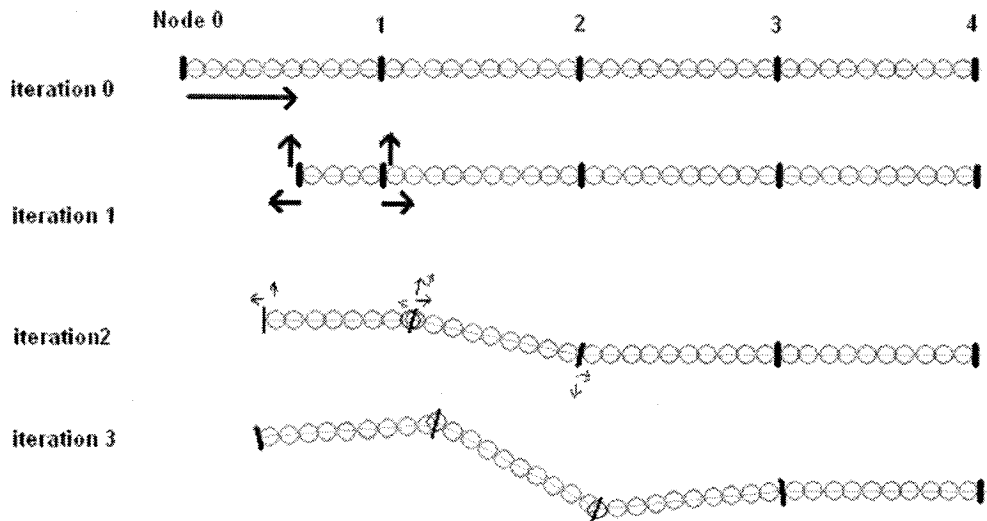


Figure 7.42. Scenario 1.0.1.1 : the new formulation

### Comparison for 1.0.1.1:

In this scenario, the original formulation started solving 50% the compression energy as soon as iteration 1, and solves a total of 4/6 or 66% of the initial compression by iteration 4. The new formulation also solves an overall of 50% of the initial compression after the first iteration, and ends up solving the same proportion of compression (4/6) also after four iterations. Both formulations seemingly perform equally for this scenario but the final shape of the yarn is different. The new formulation of course offers non linear motions that the original mechanism does not.

### Scenario 1.1.0.1

#### Original in-plane formulation

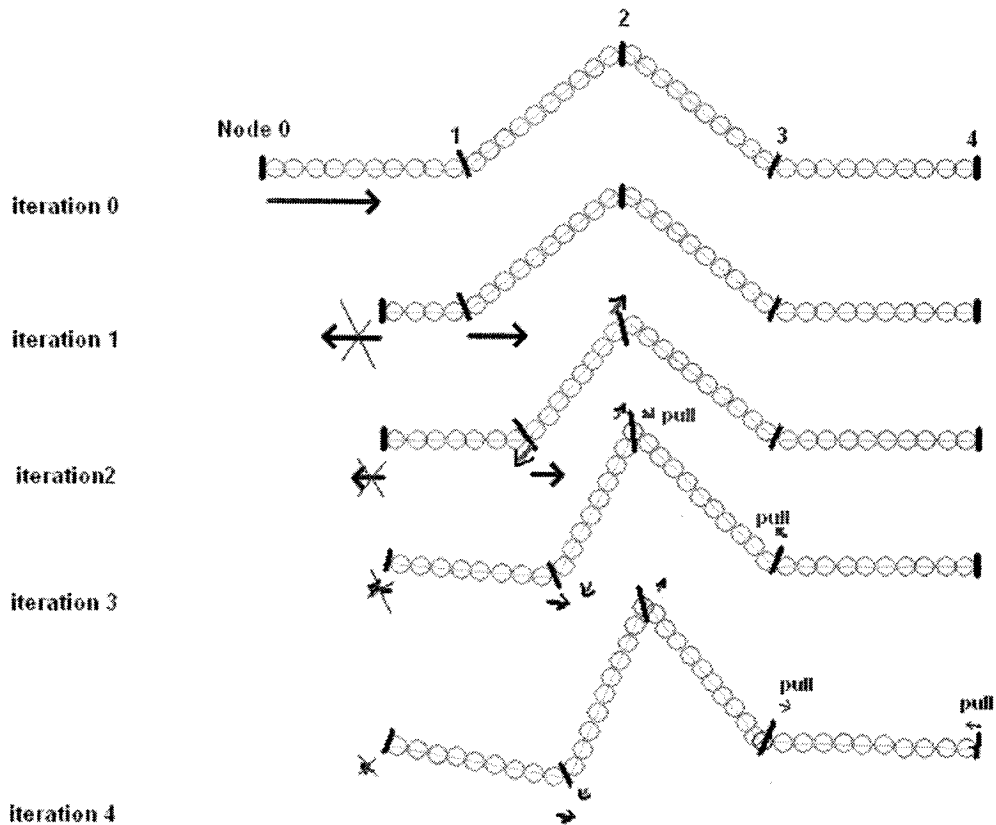
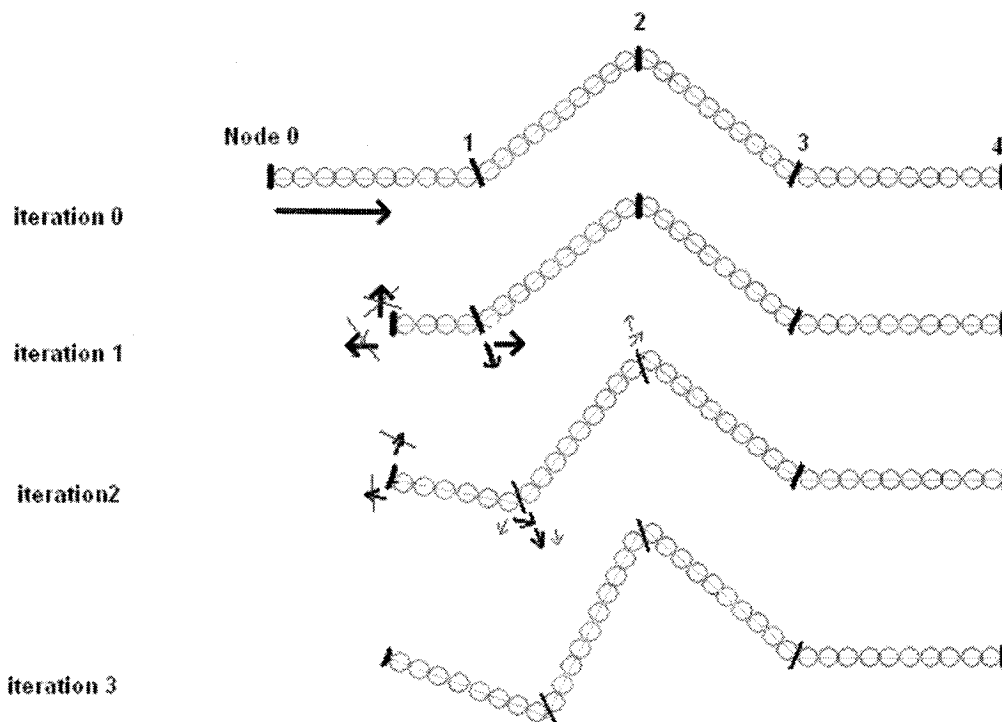


Figure 7.43. Scenario 1.1.0.1 : the original formulation

## New formulation



*Figure 7.44. Scenario 1.1.0.1 : the new formulation*

## Comparison for 1.1.0.1:

Both formulations reduce  $1/6$  of the overall compression right after the first iteration. Both formulations seem to perform equally at first; however, by the third iteration of the new formulation, we can observe that the overall compression energy reduction of the new formulation is an iteration ahead of the original formulation. And, the other difference between the new formulation's iteration 3 and the original's 4 is that the new formulation solves the energy within the first two dihedral angles and does not create any amount of energy on the third mechanism on the right unlike the original formulation.

### Scenario 1.1.1.0

#### Original in-plane formulation

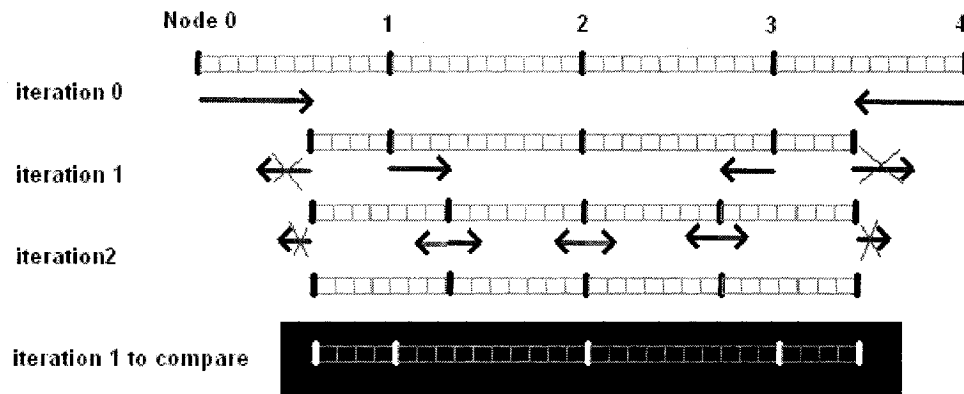


Figure 7.45. Scenario 1.1.1.0 : the original formulation

#### New formulation

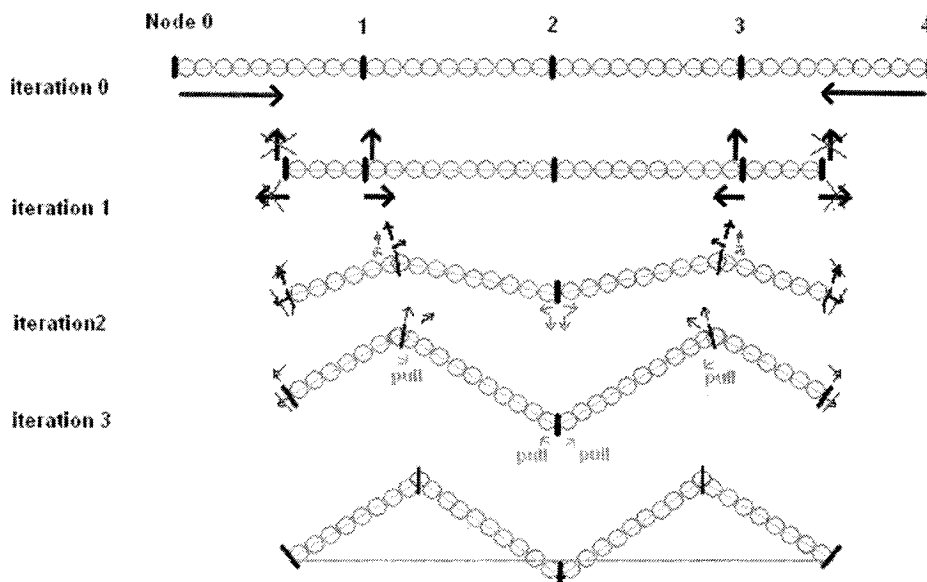


Figure 7.46. Scenario 1.1.1.0 : the new formulation

Comparison for 1.1.1.0:

It is clear by iteration 2 that the original formulation would not solve the compression energy at all. It can only equally distribute it among the three mechanisms stuck in-between the two constrained nodes. Conversely, the new formulation starts solving immediately after the first iteration. Although the overall compression is not reduced by more than 1/12, the new formulation proceeds towards a more promising solution than the original. However, a bit of stretch energy is created in the process. After the third iteration, the new formulation reduces as much as 8/12 or 2/3 of the overall compression.

Scenario 0.0.1.1

Original in-plane formulation

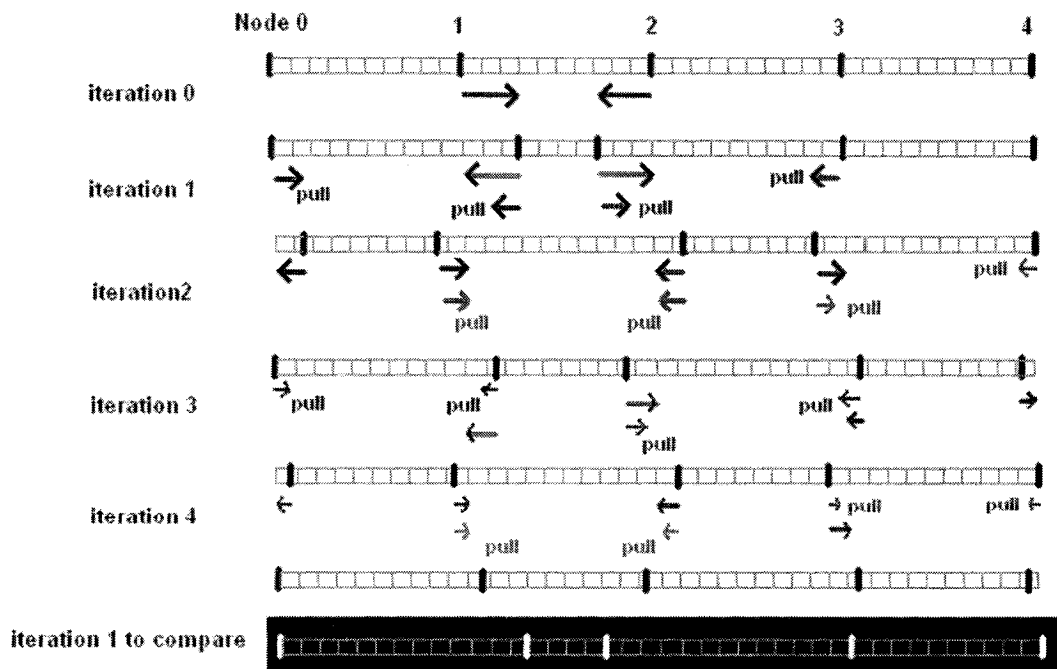


Figure 7.47. Scenario 0.0.1.1 : the original formulation

### New formulation

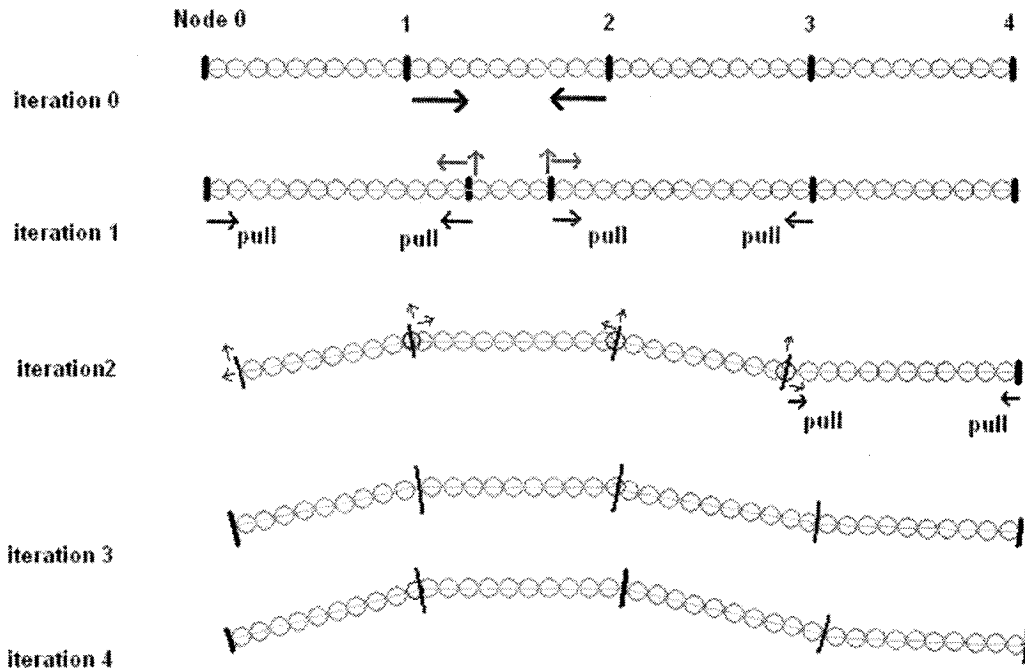


Figure 7.48. Scenario 0.0.1.1 : the new formulation

### Comparison for 0.0.1.1:

In this scenario, both formulations apparently end up reducing the same amount of compression and most of it. The difference would be once again the final shape of the yarn segment and the behavior portrayed. The new formulation shows that curvature is formed with the initial compression force whereas the original formulation does not create any such curve.

## Scenario 1.0.1.0

### Original in-plane formulation

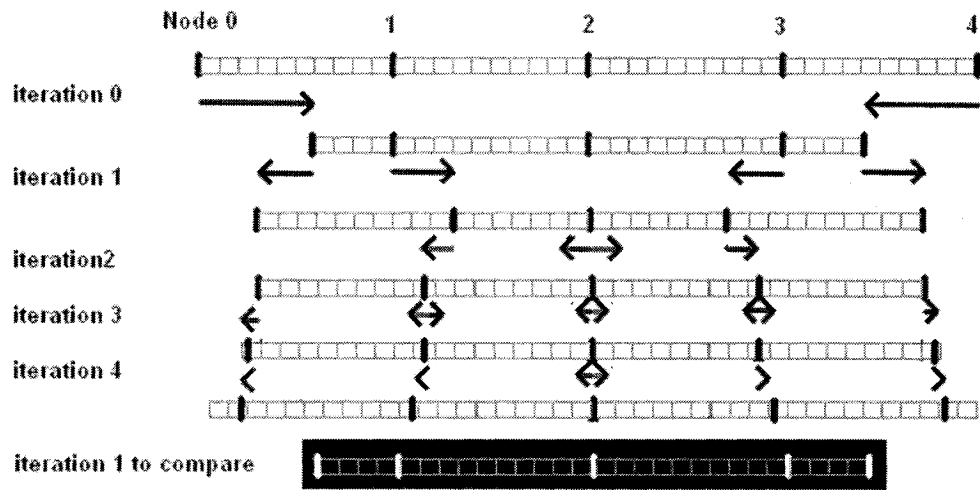


Figure 7.49. Scenario 1.0.1.0 : the original formulation

### New formulation

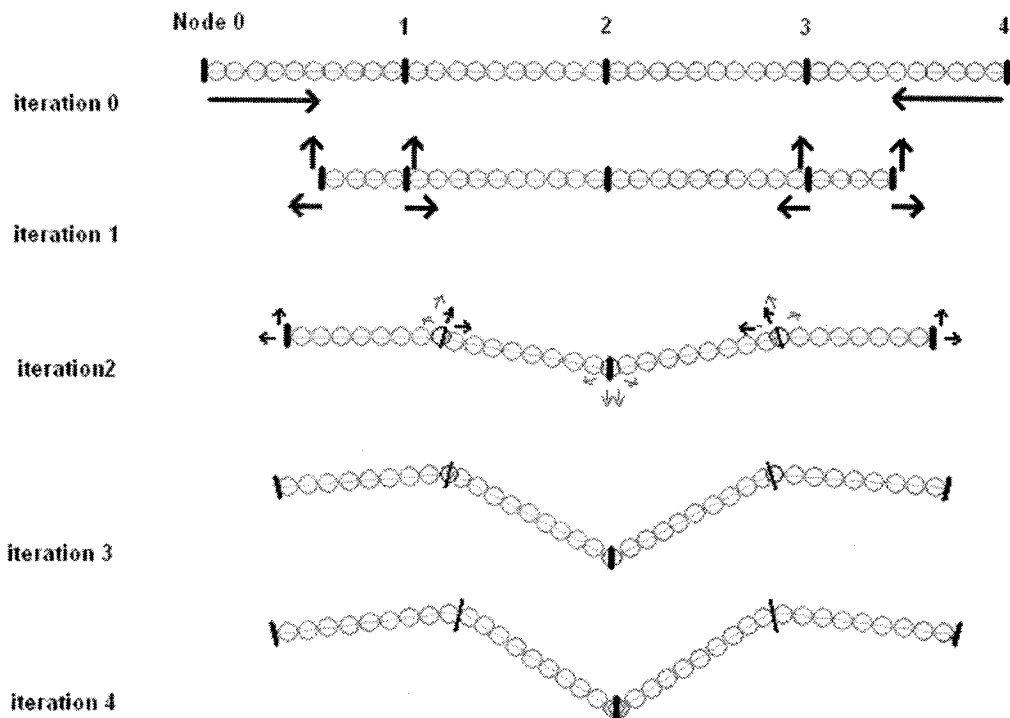


Figure 7.50. Scenario 1.0.1.0 : the new formulation



#### Comparison for 1.0.1.0:

The original formulation solved 8.5/12 or about 70.83% of the compression by the fourth iteration, whereas the new formulation performed slightly better by solving 9/12 or 75% of the overall compression. If we account the inaccuracy involved in these drawings, it could be stated that both formulations performed similarly in this scenario also with the exception of the resulting shape of the yarn segment.

The remaining nine scenarios are simply variants of the five shown above and do not fundamentally show anything different. Figures 7.39 to 7.50 showed that overall, the new formulation of “Push and curve” performs at least as well as the original formulation in solving the compression energy. And, in the scenarios where both formulations perform evenly, our new formulation depicts a behavior closer to how real cloth behaves with the formation of curves under compression.

#### **7.4.6. Computing $k_1$ and $k_2$**

Then, now what remains to be guaranteed is that “Push and curve” does not end up inducing more stretch energy on the current mechanism than “Push only” could possibly produce. One way to ensure this is by matching the final energy level on the current mechanism, or in other words, the final length of the edge ought to be the same as it would have been with “Push only”. This new condition also provides us with an approach to determine  $k_1$  and  $k_2$ .

Currently, we suggest re-expressing the relationship of  $k_1$  and  $k_2$  in form of a ratio instead of the first assumption  $k_1+k_2=1$ . And, the ratio should logically be a direct function of the bend coefficient used in the old formulation. In this manner, the bend property remains expressed by a single coefficient for each distinct fabric.

$$\frac{k_2}{k_1} = c = f(k_{\text{bend}}) \quad (\text{Equation 7.14})$$

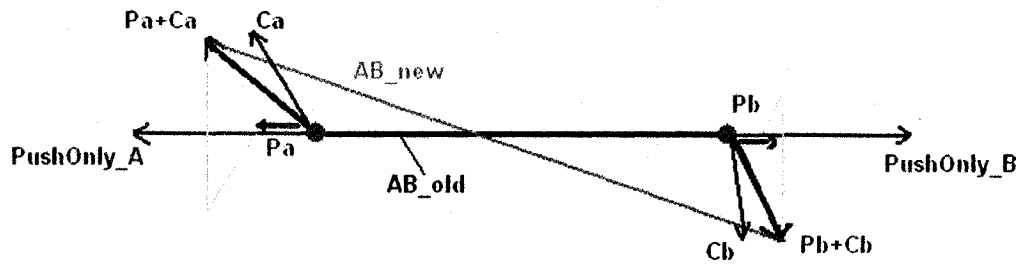


Figure 7.51. Matching the final edge length of the new formulation with the original.

To compute the values of  $k_1$  and  $k_2$ , we define the following terms:

- Let A and B be the two nodes on a shortened edge of this mechanism.
- Let  $AB_{\text{OLD}}$  be the vector B-A at the beginning of the time step,
- And  $AB_{\text{NEW}}$  also be B-A but after “push & curve”
- $L_{AB}^0$  is the natural length of edge AB.
- PushOnly\_A and PushOnly\_B are the force vectors generated by the original spring formulation on nodes A and B respectively. More precisely,

$$\text{PushOnly}_B = k_{\text{stiffness}} \left( (A-B) - \frac{L_{AB}^0}{|AB_{\text{OLD}}|} \times (A-B) \right) \quad \text{and}$$

$$\text{PushOnly}_A = k_{\text{stiffness}} \left( (B-A) - \frac{L_{AB}^0}{|AB_{\text{OLD}}|} \times (B-A) \right),$$

- $(N_A) = \left( \frac{L_{AB}^0}{|AB_{OLD}|} - 1 \right) \times k_{stiffness} \times N_A$  and  $(N_B) = \left( \frac{L_{AB}^0}{|AB_{OLD}|} - 1 \right) \times k_{stiffness} \times N_B$

where  $N_A$  and  $N_B$  are the scaled normal vectors at A and B.

- Then, we define for our new formulation  $P_A$  and  $C_A$  where  $P_A = k_1 \times \text{PushOnly}_A$ , and  $C_A = k_2 \times (N_A)$ .
- Similarly for  $P_B$  and  $C_B$ .

Recall that we currently desire the resulting energy on the mechanism AB to be exactly the same as with the original formulation; then, the final length of AB ought to be the same as it would have been with the actions of  $\text{PushOnly}_A$  and  $\text{PushOnly}_B$ . Let us assume for a moment that the time step size is equal to the square root of the node's mass and that the current velocity of the nodes are zero. Then, we can express the new condition as:

$$|AB_{NEW}| = |AB_{OLD} + P_B + C_B - P_A - C_A| \quad (\text{Equation 7.15})$$

Substituting in the terms of "push and curve" yields:

$$|AB_{OLD}| + 2 \times |\text{PushOnly}_A| = |AB_{OLD} + k_1 \times \text{PushOnly}_B + k_2 \times (N_B) - k_1 \times \text{PushOnly}_A - k_2 \times (N_A)|$$

Note that under the original formulation,  $\text{PushOnly}_A = -\text{PushOnly}_B$ . Then, at this point if we performed the substitution  $k_2 = k_1 c$ , we eventually obtain a simple quadratic equation to solve for  $k_1$ . Thus, we obtain  $k_1$  and  $k_2$  such that the both the stretch and compression energies match the original formulation whereas the compression energy produced on adjacent mechanisms is guaranteed to be at least as low.

#### **7.4.7. The pros and cons of this new formulation**

We have shown that our new formulation models cloth bending resistance and curve formation better. Our new strategy also minimizes the compression energy better. However, our approach remains far from being a true energy minimization algorithm like the continuum models are. The first obvious reason is that we do not perform any verification step to determine whether the result we compute minimizes the energy or not. Second, the strategy only tries to minimize the compression energy in the vicinity of the behavior, not on a cloth-wide range. Third, our strategy only focused on reducing the compression energy, not the stretch energy. Although our approach cannot warrant a global energy minimization of the cloth surface, we assumed the decision to add out-of-plane motion to the previous model was a justified necessity and a right choice based on the observed corollaries of cloth simulations that were built on continuum models. Their curves and folds looked natural. Also, real cloth does curve; thus, the final tradeoff of the possible stretch energy creation versus the compression energy elimination must have been an optimal reaction for the cloth by moving out-of-plane. We believe that this factual observation alone suffices to formulate an attempt like our strategy does for relating bending resistance, in-plane forces, and curve formations. Apart from the compression energy minimization, the other benefits are clear. First, reformulating our way reduced the overlap and interference between mechanisms simulating cloth. Second, the new strategy also reduced the possibility of in-plane oscillations and therefore system instability by much.

Lastly, our new formulation helps complement the bending resistance described by the previous models. The previous formulations of bend property presented in Chapter 4 were focused on describing the tendency of cloth to restore itself from a non-resting dihedral angle, and go back towards a more flat state by undoing curves. This new property solves the circumstance-specific weaknesses of the third property. For instance, given a simulation where the external forces acting on the cloth produce fast motions and rapidly changing states such as the simulation of a flag flapping in the wind, it was difficult to observe the effect of the bend property. Likewise, within a short sequence of simulation frames, or worse, within a single still frame, it was previously unattainable to distinguish different types of fabric's bending behaviors. Increasing the strength coefficient of the bend condition or increasing the value of the stiffness coefficient of flexion springs in order to quicken the results for the third cloth property was an unacceptable solution since we stated that the bend property of cloth was rather permissive. It is much easier to observe the bend properties of different fabrics through the formation of curves and folds. And, the "resistance" side of the bend property can evidently be deduced from the previous observation.

To sum up, our approach improves the modeling of cloth's bend resistance property, helps simulate the formation of curves, and bridges a relationship between in-plane forces and out-of-plane displacements.

## **7.5. Orthogonal Resistance**

In the quest of improving our current simulators further, we wanted to compare the resting shapes of real cloth with the ones produced by simulations under relatively similar constraints and circumstances. We conducted a small set of experiments under a variety of setups with simple cloth such as hanging a rectangular cotton piece of cloth from two random points on its surface; and then, changing the distance separating those two points on the cloth, comparing every time the final resting shape between the real cloth and the simulated object. We observed through the experiments that under the following specific circumstances, the simulated cloth did not behave like real cloth and its final shape was incorrectly reproduced. With real cloth, when two points on a cloth surface are held apart such that there is enough stretch occurring in-between them, the part of cloth along the line segment between the two constrained points seems to offer much greater resistance against any movement orthogonal to that line. In our set of experiments, this particular real cloth behavior was clearly noticeable when the two top corners of the cloth were pinned flat against the wall. We believe that the observations for this behavior can be re-interpreted in terms of a particle-based cloth model as follows: the nodes aligned in-between the two nodes held, or pulled apart, resist strongly the forces moving them from this alignment. The resistance starts occurring when the distance between the two nodes is greater or equal to their said “natural” distance when the cloth surface is flat. The greater the elongation rate between them, the stronger is this

resistance. In addition, this resistance is larger for a node located closer to either end, and receives a minimal value when it is in the exact middle-point of the line segment.

It would be possible to represent this behavior by using the trampoline analogy and allocate spring constants to simulate the resistance. However, due to all the instability concerns we have discussed previously; we prefer to forfeit adding additional spring-based mechanisms into the simulation, and avoid increasing the worry about forces, oscillations and damping. Instead, we successfully simulated this behavior with our cloth simulators by defining a new constraint enforcement mechanism.

First, our new enforcement method is constructed by combining the mass-constraint mechanism presented in [BAR98] with a formulation based on the analogy of beam physics regarding the distribution of this mass modification. Nonetheless, our new mechanism relies on the assumption that the nodes on a cloth yarn can be realigned correctly prior to the activation of the new mechanism. Since the previous model failed to model this characteristic realistically, we also suggest a realignment method to satisfy this assumption. To test our newly formulated mechanisms, we added these constraint enforcement features to our cloth simulator and the results were much closer to real cloth than previously.

The following two sub-sections 7.5.1 and 7.5.2 describe our new mass-constraint mechanism and the realignment assumption required respectively in more detail.

### 7.5.1. The mass-constraint mechanism

We suggest a method that combines the constraint enforcement mechanism that uses mass modification by Baraff & Witkin [BAR98] presented in Section 4.2, and the principle of deflection curve from beam physics. The positional constraints of the two end nodes of the cloth act similarly to the two supports of a beam. In physics, the beam deflection  $\delta$  of a beam with two supports is given as:

$$\delta = \frac{F A^2 (L - A)^2}{3EI} \quad (\text{Equation 7.14})$$

where  $F$  is the force applied on the beam; more precisely it is a load, a weight.  $L$  is the length of the beam,  $A$  the distance from the closest end, where a support is, to the point on the beam where the force is applied.  $A$  is assumed to be within the range of the interval  $[0, L/2]$ . Also,  $E$  is the Young's Modulus, and  $I$  the moment of inertia of the beam. In brief, the beam deflection is the amplitude of the curve of the beam caused by the load. The Young's Modulus, defined as stress divided by strain, is a ratio that varies with the material of the beam while the moment of inertia is determined by the beam's cross-sectional shape and thickness, but is not related to the length or the material of the beam.

The inverse-mass concept introduced along the mass-modification mechanism will serve as a tool for simulating the resistance. As the resistance grows larger, the inverse-mass would tend towards zero; therefore, limiting any change in velocity. We extend this constraint mechanism with the deflection curve notion in order to distribute



this resistance to the nodes concerned. Obviously, the yarn does not act like a beam and both the Young's modulus values and the moment of inertia of a yarn or thread are much inferior to the ones of a steel beam. But, the formulation of the deflection curve gives a fine basis to the shape of a cloth "line" under similar conditions.

To relate the two ideas, we start by redefining  $L$  and  $A$  in the context of our cloth simulation. We let  $L$  be the distance that separates the two constrained nodes. These two nodes will act as end nodes just like the two supports in the case of a beam. Then  $A$  will be defined as the distance separating a node and the closest end point. The distance  $A$  is obviously zero for the end points. Both variables  $L$  and  $A$  will need to be refined some more after we present equation 7.16. We set the mass increase to be inversely proportional to the deflection value. With the beam deflection equation we can observe that the equation can be rewritten as

$$\delta = F \frac{A^2(L - A)^2}{3L} \frac{1}{EI} \quad (\text{Equation 7.15})$$

where, omitting the force  $F$ , the first factor of the expression relates to where the position of the force applied, whereas the second factor relates to the characteristics of the beam. We can correlate these two factors with cloth's thread characteristics to formulate the mass modification for expressing the resistance. Thus, we can express the mass modification of a node aligned between the two end points inclusively as

$$\text{mass} = \text{mass} \times \left( 1 + \frac{3L'' EI''}{A^2(L - A)^2} \right) \quad (\text{Equation 7.16})$$

where the two terms  $E$  and  $I$  are now between double quotes and we make it a single coefficient value we will simply label  $EI$ . Like beams, a longer yarn segment and a larger

value of  $L$  would imply a larger deflection or in this case, less resistance. However, with the equation 7.16 above, it becomes clear that  $L$  and  $A$  cannot be defined as current distances between nodes. In the case of cloth, unlike beams, when  $L$  increases, more stretch is created on the yarn segment and a stronger resistance is generated against movement orthogonal to the stretch direction. We once thought of expressing both  $L$  and  $A$  in terms of the number of edges separating the cloth nodes; unfortunately, that approach is not scale-invariant versus the number of nodes used to sample the same yarn segment. In other words, that approach could not produce a formulation independent of the mesh resolution. Our solution is to simply let  $L$  and  $A$  be in their said “natural” distances when the cloth is in its flat resting state. This definition allows both a formulation that is independent of the mesh resolution, yields a larger value of  $L$  when for a longer yarn segment but does not increase  $L$  when the segment is overstretched.

Next,  $EI$  is a function of two set of parameters. The first set of parameters is the cloth’s mechanical characteristics expressed in our models as the stiffness or condition coefficients for stretch, shear, and bend. The second set of parameters is the current elongation ratio identified by  $\epsilon$ , and the maximum elongation ratio  $\epsilon_{MAX}$  allowed with the fabric. So,  $EI$  is a function

$$EI = f(\epsilon, \epsilon_{MAX}, k_{stretch}, k_{shear}, k_{bend})$$

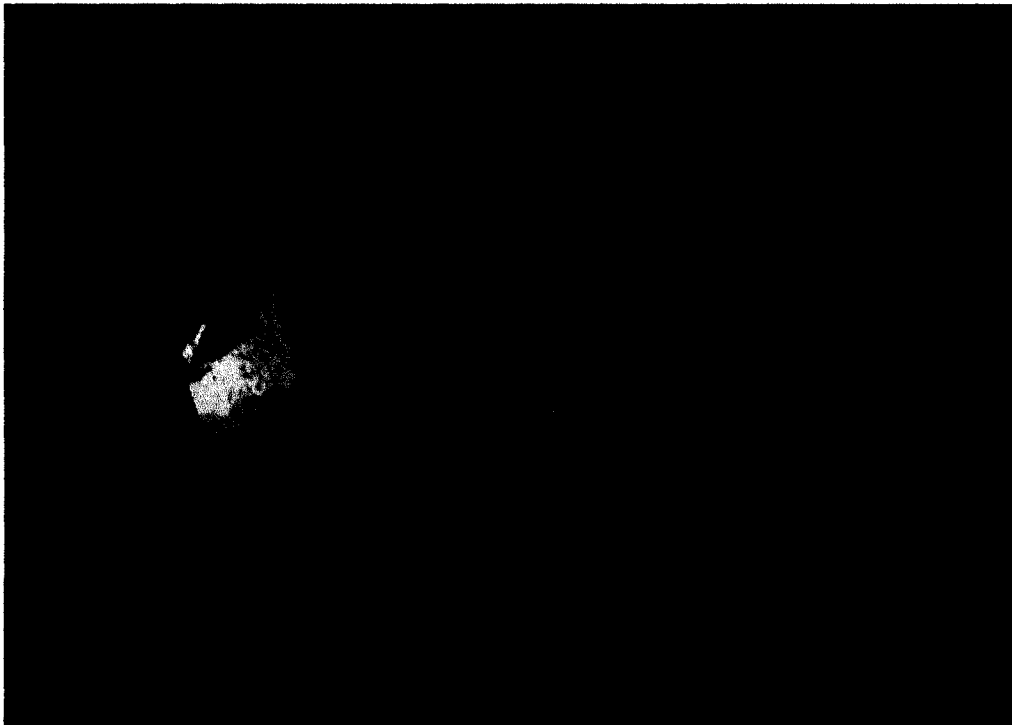
The specific details of this function depend critically on the values of each parameter set for the simulator; however, we obtained satisfactory results by defining the function such that  $EI$  was about 100. For our particular simulator settings and stiffness coefficients, we tried  $EI$  as a linear combination of the three cloth coefficients and  $\epsilon_{MAX}$ , but plus a higher power term for  $\epsilon$ , a hypothesis that gave us visually good enough results. More

specifically, a quadratic term with  $\varepsilon$  worked for us. Unfortunately, we were unsuccessful in drawing a more exact relationship between these parameters and their contribution with the mass modification. EI is set null if  $\varepsilon$  is less than zero. Moreover, in the case A was zero in the equation of the mass modification, we replaced the division by zero with a multiplication of the node's mass by a very big number. We found that multiplying the node's mass by a value of ten thousand was sufficient to simulate an infinite mass for our simulator considering the range of all the other values computed in the simulator. The final task is to determine the nodes aligned between the two constrained nodes in the case the line defined by those two were not along a warp or weft. We suggest the use of a scan line algorithm for this task such as the Bresenham's line rasterization algorithm. We implemented a simple variant of the midpoint scan-line algorithm for our simulator and it allowed the application of this mass-constraint mechanism over any pair of nodes on the cloth surface.

To summarize, the extended constraint mechanism we presented offers two advantages in addition to the capability to simulate another behavior for cloth. First, the resistance is formulated as a mean to absorb external forces instead of adding more forces to the system. Then, by the same token, over-elongation and stretch between the two constrained nodes will be limited further. However, the drawback of this method is that it would also greatly affect the internal forces of the stretch condition trying to pull back the nodes; nonetheless, this disadvantage is negligible for two reasons: First, if the distance between the two constrained nodes were greater than their natural distance and this state were maintained such that it became a position constraint for the two end nodes,

the internal forces were probably already insufficient to act against the external forces causing and maintaining the over-elongated state. Second, we left the super-elongation problem to a post-step position corrective method instead of utilizing stronger anti-stretch coefficients that would have required even heavier damping. It is necessary to point out that the mass of each node is reset after the end of each time step. In short, this mechanism does not truly interfere with others, and helps simulate an additional cloth behavior.

We implemented this constraint mechanism on top of our mass-spring cloth simulator and Figure 7.53 shows a simulation with this new mechanism whereas Figure 7.52 gives us a reference of our previous simulator without this mechanism.



*Figure 7.52. A cloth hanging from the two end points of a diagonal in the original simulator.*



*Figure 7.53. A cloth hanging from the two end points of a diagonal with the new mass constraint mechanism.*

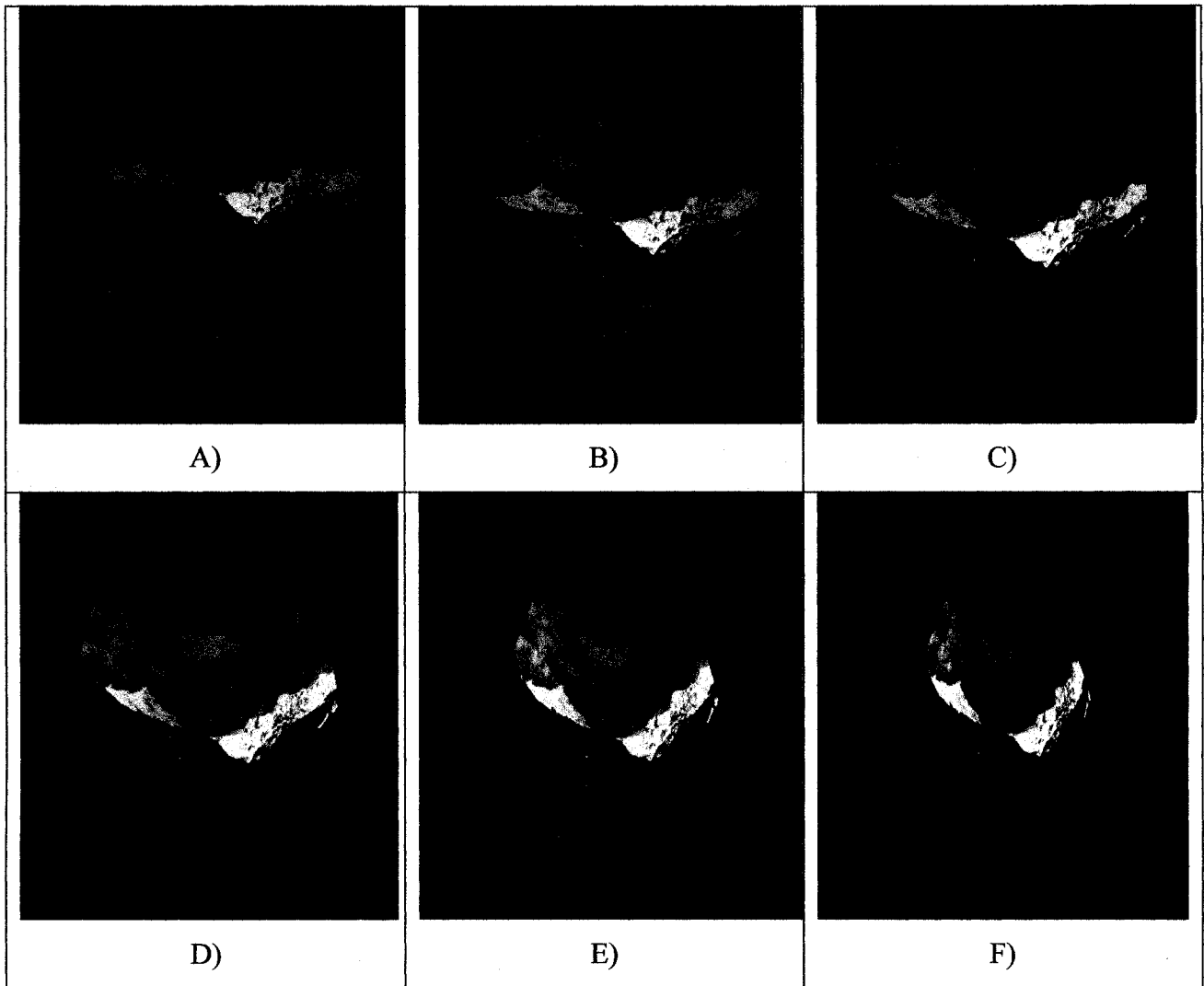
The constraint mechanism we presented in this section assumes that a proper realignment process has been done prior to using this mechanism. That is, if the distance  $L$  were greater or equal to the natural distance separating the two constrained nodes; then, an attempt to properly realign all the nodes on the yarn segment has been performed already. This requirement is discussed further in the next section.

### **7.5.2. The realignment assumption**

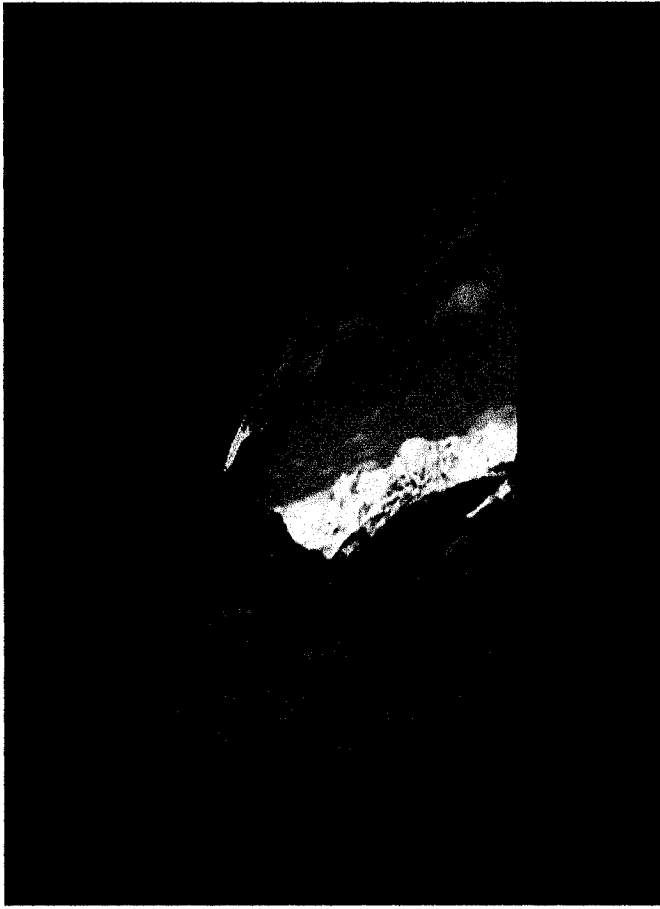
As stated by the end of the previous section, the mass-constraint mechanism assumes that the nodes that would be constrained were realigned or at least underwent a realignment attempt before the mass-constraint mechanism is triggered. Otherwise, once

the nodes are constrained by mass modification, they would then become much more resistant to any form of corrections using forces, and the nodes could retain a possibly incorrect state. This usually results in a yarn segment that remains much over elongated.

However, the necessary realignment is easier said than done. The two particle-based models we have employed to construct our simulators do not satisfy this condition. Let us construct a scenario based on the scope of what has been dealt with in this section 7.5 to illustrate this problem. Let the cloth object represent a small light rectangular piece of cloth. If both top corners of the cloth object were brought closer and then pulled back to their starting positions and even slightly further to instill some stretch along the top yarn, we would expect the cloth to mostly regain its original flat shape with maybe the exception of some slight effect of gravity producing small curving tendency downwards. But, Figures 7.54A to F and 7.56A to F show that the cloth models are not yielding the expected results. Figure 7.55 is just another point of view to the instance captured in Figure 7.54F. The cloth behaves acceptably while the corners are brought closer with that new formulation devised in section 7.4, but, upon the pull back motion, the cloth does not realign itself enough. We tried reducing the mass and the curving downward was indeed reduced, but, despite this effort, one evident weakness was always observable with these models: the realignment of the top yarn was excessively slow, and in most case, not satisfactory. And, increasing the number of nodes accentuated this problem even more.

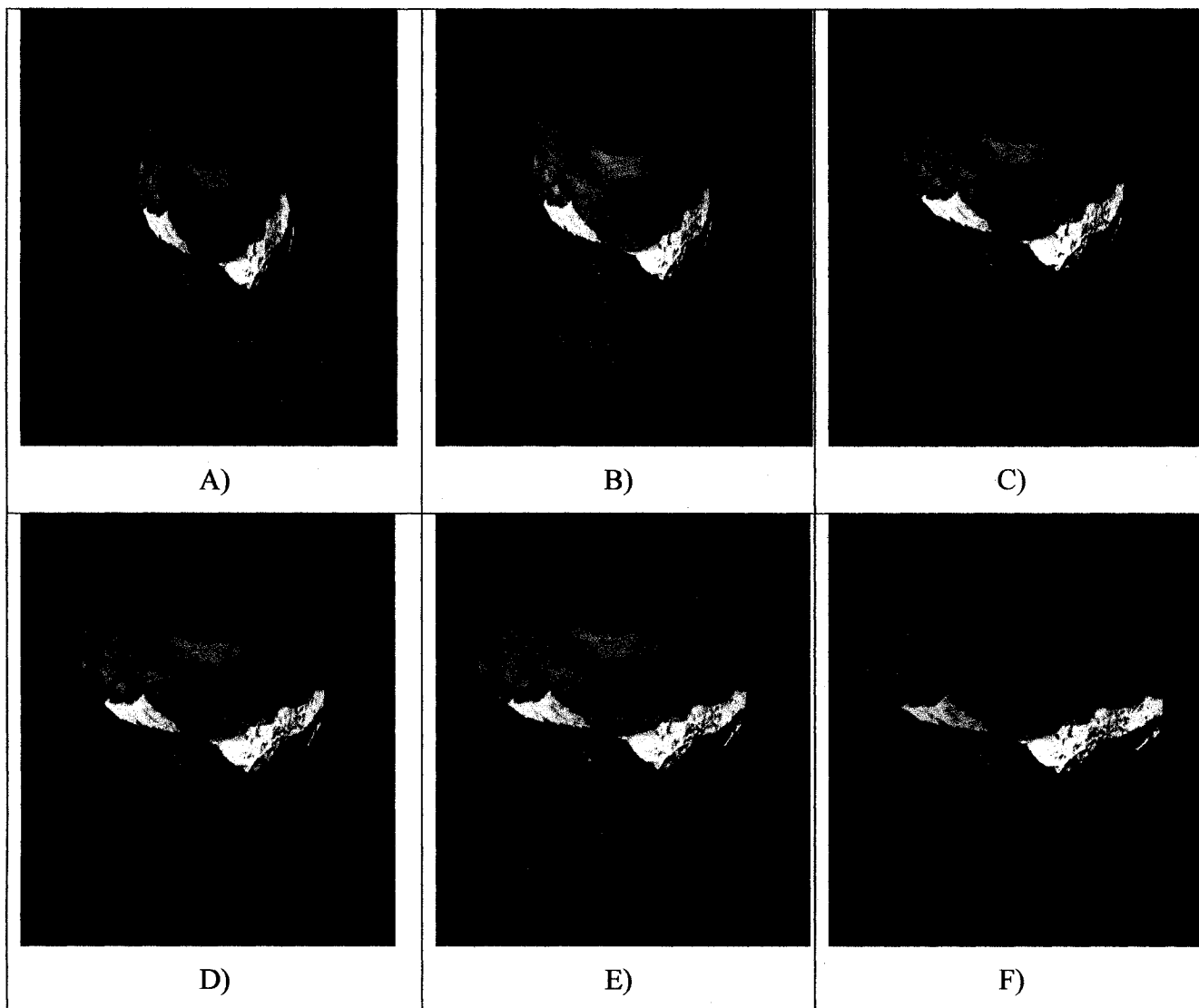


*Figure 7.54. Simulating the first part of the scenario on the original cloth model at times: A) 0 second, B) 2, C) 4, D) 8, E) 12, and F) 15 seconds.*



*Figure 7.55. The instance captured by Figure 7.54 viewed from a different angle.*





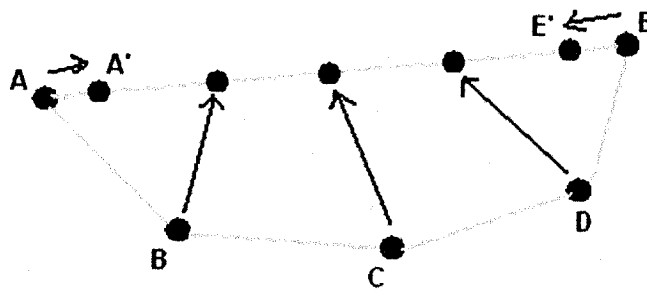
*Figure 7.56. Simulating the second part of the scenario on the original cloth model at times: A) 18, B) 22, C) 26, D) 28, E) 30, and F) 40 seconds.*

The failure to obtain the desired results can be explained by three characteristics and perhaps weaknesses of the two cloth models we used. The first reason explaining the unsuccessful realignment is the amount of time needed to communicate the information between mechanisms of the cloth model. This concern is not particular to the two cloth models we used but any model describing the physical object by a set of interacting mechanisms whose information transfer is from mechanism to mechanism will necessitate a certain number of time steps to spread the information of an event occurring

on one end of the cloth surface to the other end. We understand that real cloth also necessitates some time before the information is communicated along the cloth surface; however, it is certainly much faster. This composes some of the trade-off between a particle-based model and a continuum model. The continuum model reiterates in a same time step until a global equilibrium is reached and therefore, all the information has been accounted for on the cloth surface. But, the price is the multiple iterations required within a single time step. The second weakness relates to the first. Taking our current scenario, by the time the information carrying the realignment “order” goes from either corner node to the middle nodes of the top yarn, several iterations would have passed, and these latter nodes would have dropped quite too much and be late on the realignment already. But to make things worse, once they receive this information to realign, the best they can do is to receive pulling forces coming from the two adjacent mechanisms. And, obviously, those two linear force vectors are insufficient to properly help realign the nodes. This second weakness can be summarized by the fact that each mechanism formulated so can only try maintaining their edge length constant but cannot help maintain the greater yarn segment length constant. The third problem clearly revealed in scenarios like this one is that the model’s behavior is greatly dependent on the mesh resolution. A larger number of nodes also mean a larger number of mechanisms and thus, a larger number of time steps, before the same information is communicated through the cloth object. This is a serious problem because a higher mesh resolution is used in hope of simulating a more realistically looking cloth object, but the end result is a less accurate simulation due to the information being spread too slowly.

In the first subsection 7.5.1, we expressed the desire to avoid adding forces to the model. However, we do not see any way to go around this tactic without having to significantly redesign the cloth models. For the moment, we have come up with two simple complementary methods to solve scenarios resembling the current particular scenario and satisfy the realignment assumption.

Let us re-label the two corner nodes being pulled as A and E. Then, if the length of AE is greater or equal to their natural distance, the following realignment mechanism is performed like shown in Figure 7.57:



*Figure 7.57. The first realignment method when the length of segment AE is greater or equal to its natural length.*

In brief, we install a new spring on AE and compute its resulting position of A'E'. Then, a realignment vector displaces each node towards their ideal position on the resulting segment A'E'; and, the magnitude of this vector depends on the elongation rate of A'E'. We formulated these realignment forces in forms of springs whose stiffness coefficient was both dependent on the elongation rate on A'E' and also set to realign within a very few time steps.

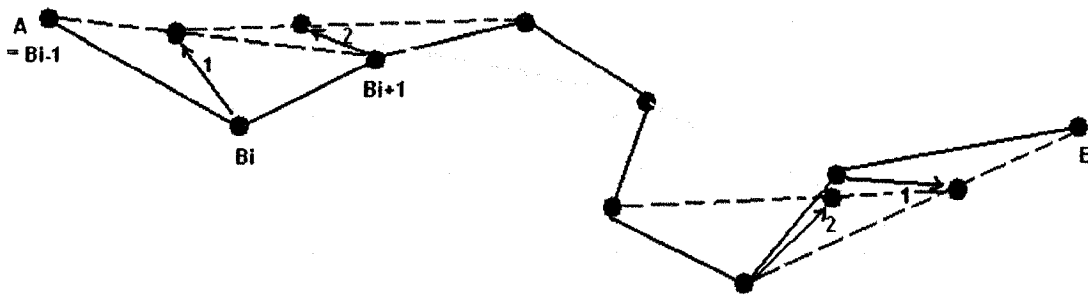
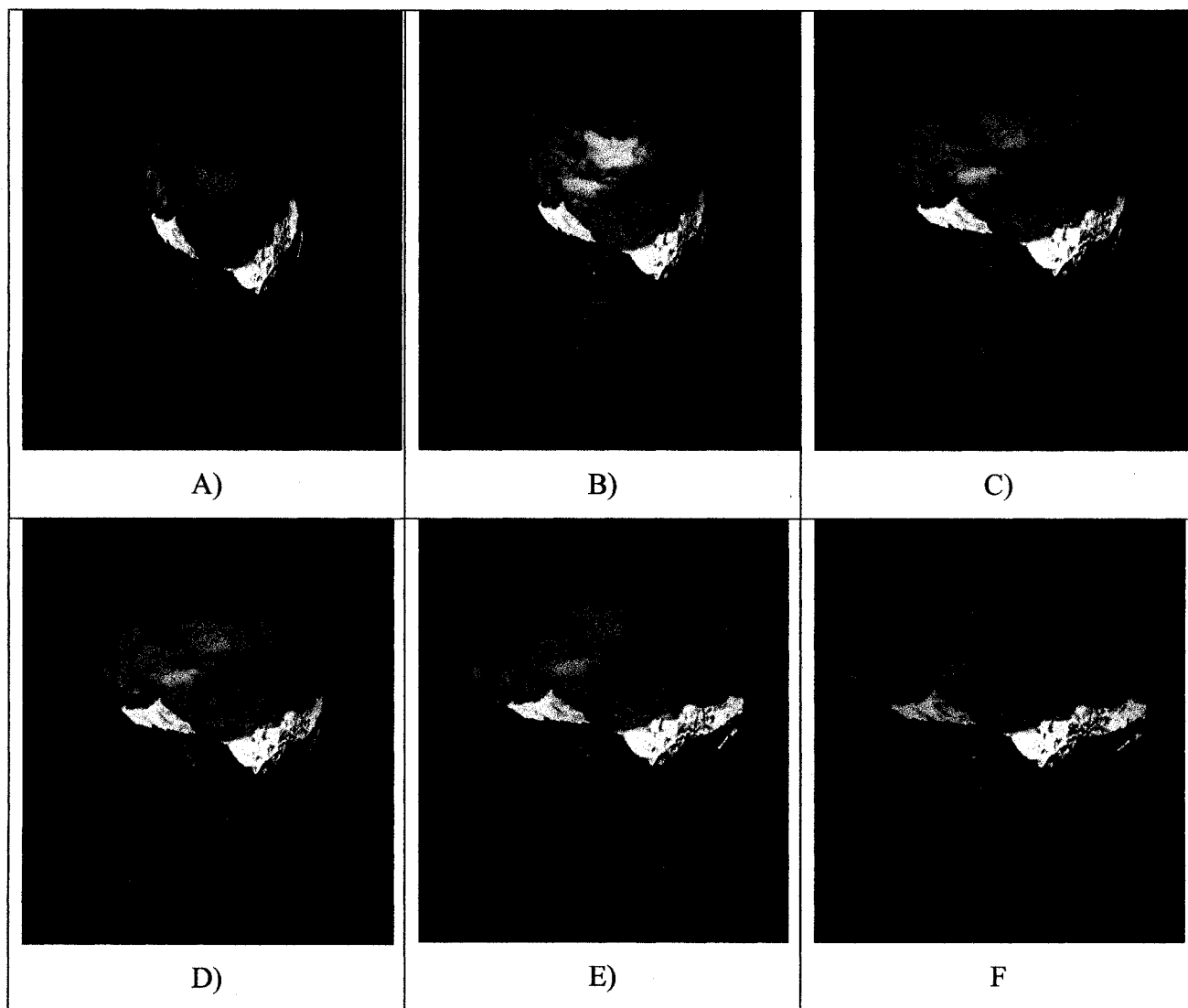


Figure 7.58. The second realignment method when the length of segment  $AE$  is shorter than its natural length.

Otherwise, if the length of segment  $AE$  were less than its natural length, a different method is used to realign the nodes. This second method, illustrated in Figure 7.58, does not require any condition about the current shape of segment  $AE$ . The segment  $AE$  could assume any starting shape. For each node  $B_i$  on the segment  $AE$  where the node  $B_{i-1}$  precedes it and  $B_{i+1}$  follows it, the method focuses on keeping the length of the edge  $B_{i-1}B_{i+1}$  instead of the smaller edges,  $B_{i-1}B$  or  $BB_{i+1}$ . This second method can be summarized as constructing a realignment force vector to pull node  $B$  towards a position  $B'$  on the edge  $B_{i-1}B_{i+1}$  such that the length of the smaller “left” edge  $B_{i-1}B$  is equal to its natural length. Note that we stated “left” as an example to express that this method proceeds in a particular order. The algorithm starts with the two end nodes  $A$  and  $E$  and proceeds inwards to realign its nodes as such until it reaches the middle nodes of the segment  $AE$ . One particular characteristic of this method is that it allows the simulation of a say cloth mesh  $M_1$ , whose resolution is doubled a coarser mesh  $M_2$ 's i.e. four times the total number of nodes, to look more similar to the simulation of  $M_2$ . The previous

models produced very dissimilar simulations under different levels of mesh resolution. In addition, this method yields a faster realignment for the constrained yarn.

The two methods combined for a same simulator give visually satisfactory results. Figures 7.59A to F show the same scenario but now with these realignment methods to help, and the results are closer to the real behavior of a relatively light and small rectangular piece of cloth as specified in this situation.



*Figure 7.59. Simulating the second part of the scenario on the improved cloth model at times: A) 18, B) 22, C) 26, D) 28, E) 30, and F) 40 seconds.*

## **7.6. Wrinkles and other Considerations for Garment Animation**

In this section, we discuss the apparent differences between the simulation of simple cloth and the simulation of more complex garments. We then mention a few strategies developed for simulating complex garments, and more precisely, for simulating wrinkles on cloth. We close the section by discussing some hypotheses on how wrinkles could be interpreted on our current particle-based cloth models.

### **7.6.1. The Additional Requirements for Simulating Complex Garments**

The models and behaviors of cloth presented so far in this thesis frequently serve as foundation for the simulation of more complex garments. However, the simulation of complex garments involves many more issues than the simulation of simple cloth. To list a few, the design of the garments themselves necessitates a certain level of pattern making and garment design knowledge; furthermore, the initial dressing of these garments over a virtual human body is a more sophisticated process than the usual draping of table cloth. Moreover, the accurate modeling of a specific garment may well require specifying the different types of fabric that make up the numerous parts and layers of a garment as well as the seaming constraints that structure it. These details must be considered to distinctively describe a particular garment and model it accordingly. In the animation phase, the interactions between the virtual bodies and the garments must first be considered; then, the interactions between the different parts and layers of the

garments themselves must also be accounted for. Thus, the scale of complexity for animating garments is far greater than it is for the animation of simple cloth. Different techniques need to be employed in order to keep the simulation performance viable without trading too much realism away. Nadia Magnenat-Thalmann and the teams at MiraLAB, Geneva, specialize much in the techniques of garment simulation. Among those techniques, the multi-layer approach [COR02] is an efficient method to simulate clothing on virtual actors by applying different simulation models and algorithms for each layer. A piece of garment is affiliated to a certain layer depending on its interactions with the body and its distance from the body's surface.

### **7.6.2. Past Strategies for Simulating Wrinkles**

There is a cloth feature that has an even more obvious impact on the level of realism for the simulation of complex garments than for the simulation of a simpler piece of cloth: wrinkles. Despite the addition of a method to simulate the formation of curves in Section 7.4, our capability to simulate wrinkles on cloth remains limited to the degree of detail we can afford. By the degree of detail, we mean the number of cloth nodes to simulate mechanically the wrinkles, the level of mesh resolution in order to use geometric techniques, and the quality and definition of the textures for texture-based techniques. In all, without these additional prerequisites for finer details, there are not enough folds and curves to simulate wrinkles appropriately.

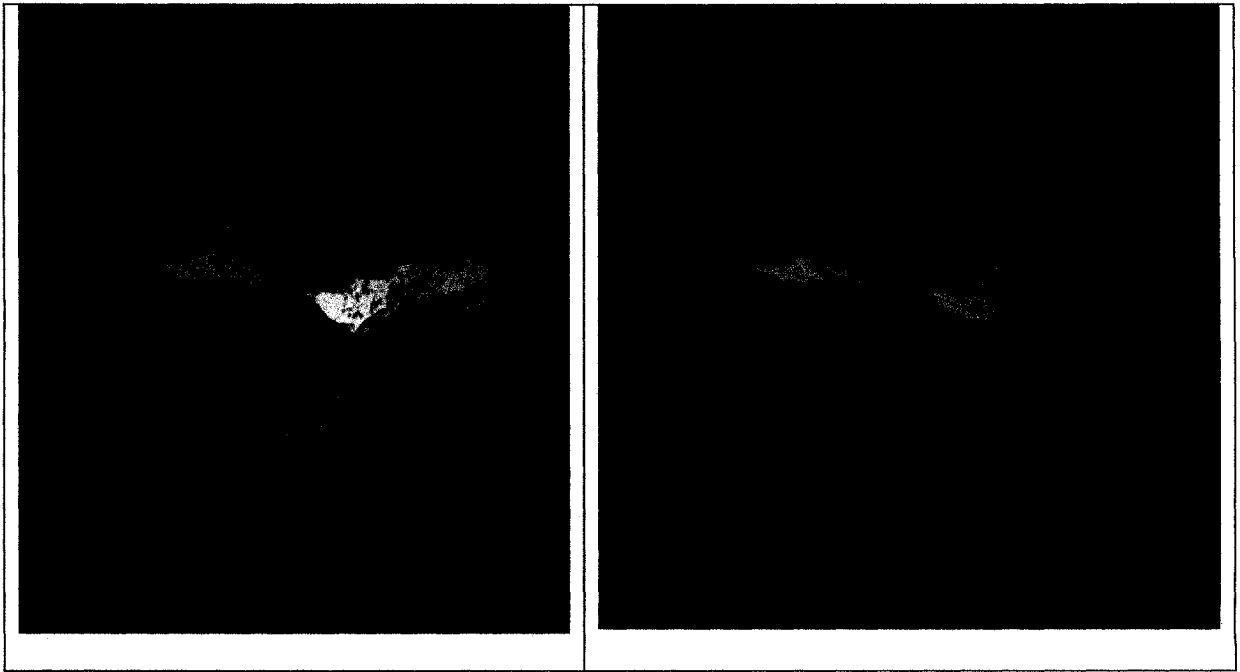


An increase in the number of mechanical nodes will ease our attempt to achieve wrinkles; unfortunately, it also increases our computation overhead greatly, a performance cost that we may not be able to support. It is possible to use geometric techniques such as the one described in Oshita's work [OSH01] to increase the level of detail to the cloth while remaining with a coarse mesh for the mechanical simulation and using a finer mesh obtained from geometric smoothing techniques for final display. Alternatively, the adaptive meshing technique presented in the work of Villard and Borouchaki [VIL02] can provide us with a way to obtain more curves and smoother curves. Their technique allows the dynamic refinement of the cloth mesh used within regions of the cloth where the degree of curvature requires it. However, this technique has a drawback for the simulation of wrinkles on complex cloth objects moving such as garments worn by actors in motion. In such simulations, the various movements of the actors change the wrinkles on the clothing too rapidly. Consequently, the method will rapidly attain the finest mesh resolution due to the high number of wrinkles visualized on the garments and the method will be very busy adapting itself to the constantly changing state of the cloth. Nonetheless, this method remains excellent for simulations where the cloth simulated does not continuously change its state. If the concern were not to obtain a fine mechanical simulation for wrinkles, then Hadap et al. [HAD00] describe an elegant geometric and texture-based approach to simulate wrinkles on clothes. Their approach can be summarized as follows: a user-defined wrinkle pattern is input as a bump map along with the initial un-deformed cloth triangle mesh. Then, the algorithm computes the modulation of the wrinkles based on a deformation transformation function per triangle. That deformation transformation was in turn obtained from the geometric changes of the

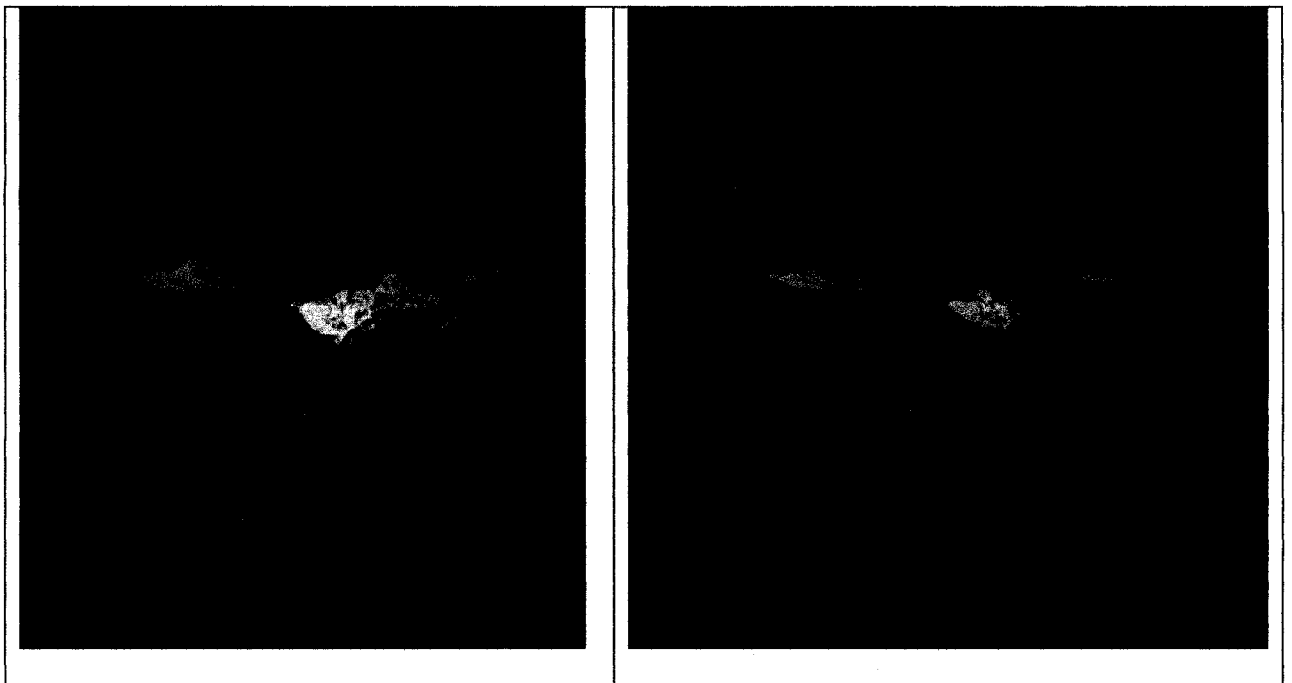
triangle of the new deformed mesh that was computed from the physical simulation of the cloth. In short, the wrinkles are appropriately modulated to reflect the physical changes of the cloth mesh; and, this technique allows the appearance of wrinkles to fit along the changes of the clothes. There are additional discussions in that work providing solutions involving the appropriate change in the direction of the wrinkles under varying circumstances and multi-fold wrinkles. The resulting accuracy of the representation of wrinkles however depends on the correctness of the initial user-defined wrinkle pattern or patterns. Although this is not exactly a drawback, the technique still requires a correct input that may not be known by the user. If the challenge were to determine the wrinkle pattern under the given conditions in which the cloth is being simulated, then the problem is not solved.

### **7.6.3. Hypotheses about wrinkles in a particle-based model**

The Figures 7.40 and 7.41 below show the relative increase in realism that an attempt in simulating wrinkles brings. Two completely flat pieces of cloth are compared with other pieces positioned the same way but with some undulations on their surfaces. Cloth surfaces are rarely entirely flat in real life unless the surface was previously ironed and further kept from the formation of wrinkles and creases. Cloth, through its many forms such as garments, curtains, flags, table clothes, inevitably has to interact with the surrounding environment and can rarely avoid the formation of wrinkles and creases on its surface.



*Figure 7.60. Two flat pieces of cloth.*



*Figure 7.61. Undulations.*

We discuss hereafter some hypothetical approach for modeling wrinkles on our particle-based cloth models without the use of further texturing or geometric techniques. Let us assume that the current mesh resolution was satisfactory and affordable to simulate wrinkles or creases; then, we would still need a way to simulate the formation of realistically looking wrinkles instead of just throwing in some random undulations on the cloth surface.

The following observations can be made with real cloth. When cloth buckles up, the curves and even the wrinkles have a more probable tendency to be formed from already existing small undulations and creases by either accentuating or enlarging the previously present curves. Additionally, this tendency is observed only when the buckling up happens along a direction relatively perpendicular to the previous curves, wrinkles, or creases. Under proper circumstances, it is as if the cloth memorized from where to start curving up and folding over itself. These observations can be verified in practice by folding a piece of cloth sharply enough to produce a small crease; then, hold two points on the cloth surface perpendicular to the newly shaped crease, and move the points back and forth towards each other.

Therefore, the first hypothesis we form about modeling wrinkles is that there exists a “memory effect” with cloth; and, the formation of wrinkles and creases can be associated with this hypothetical characteristic.

Next, we believe it is possible to attribute a “permanence level” to such a tendency. And, the higher this “permanence level” is, the higher the probability of the tendency to kick in. Creases for example consist among the most permanent of these effects on cloth while wrinkles have lower probabilities to be reproduced in the exact way from two similar buckling phenomena.

As a third hypothesis, we also believe it is feasible to model these “memory effects” by defining them as relationships between nodes on a same warp, weft, or shear “line” that are at least separated by an intermediary node. And a crease, for instance, consists of a set of such relationships. Thus, our third hypothesis is that wrinkles or creases can be modeled as a set of relationships between nodes that are not directly adjacent, and each relationship consists of a non-flat resting dihedral angle. Please note that there is no requirement that a wrinkle or crease be formed in a straight line on the cloth surface. But, each relationship of the set that composes the wrinkle is perpendicular to the wrinkle itself.

We observe with real cloth that wrinkles and creases temporarily disappear when the cloth surface that includes them is stretched in a direction that is more or less perpendicular to the wrinkles and creases. Then, we can extend our hypothetical modeling of wrinkles and creases further with this statement: Wrinkles and creases can only fully manifest themselves when there is no stretch occurring perpendicular to them.

The two nodes in each relationship can be separated by a certain number of intermediary nodes. Obviously, we can claim that this number is smaller for creases than it is for

wrinkles. If we added one to the previous number, we would obtain the length in number of edges of each relationship. Although we are unsure how to both accurately and effectively redistribute the curving displacement of these intermediary nodes when the relationship reactivates itself, it is clear that this relationship reactivates itself when its “length” is shorter or equal to its resting length where a non-flat resting state has now been associated. Again, a wrinkle or crease is a set of such relationships. Consequently, as with real cloth, when a crease is of a considerable length and the cloth surface is stretched perpendicular only over half of this length, we can still observe the presence of the other half of the crease over the cloth surface that was not stretched.

We label the maximum relationship length of a set that composes the wrinkle to be the wrinkle’s width, or, “the memory’s width”. And, the number of relationships in this set will determine how long the wrinkle is over the cloth surface and will be labeled as the memory’s length. (Figure 7.60)



*Figure 7.62. The memory effect defined as a set of relationships between cloth nodes.*

The “memory effect” idea is a way to accelerate the process of recreating wrinkles. Yet, we need to specify how these memory effects can be obtained. We distinguish three sources and methods to obtain the memory effects. First, they can be user-defined to recreate the wrinkles from another wrinkle pattern. Second, every time the cloth simulated produces a curve with an angle that becomes too sharp, below a certain value, we record the curve as a memory effect. Finally, a cloth simulation could be run once without worrying about memory effects and interact with a surrounding physical static environment while taking into account its seaming constraints in the case of a garment. The cloth would then take a final drape shape, and the folds, wrinkles, and curves formed could then be taken as candidate memory effects.

To recapitulate, we believe it is possible to extend the current particle-based cloth models and simulate wrinkles by defining them as sets of relationships between nodes, and explain this cloth behavior as if cloth memorized the wrinkles and creases. This hypothetical approach however requires that the mesh resolution be fine enough to represent the small wrinkles and creases.

## Chapter 8: Conclusions and Future Work

This thesis summarizes our work for determining helpful guidelines in the construction of cloth simulators. We started by dividing the whole challenge into subtasks and attempted to predefine the steps that are required for a cloth simulation by referring to related work of various authors. We then explored the different obstacles as well as the alternative solutions associated with each subtask. At each step, we studied and compared the different models and approaches by trying practical implementations of some of these techniques over our two cloth simulators, and, we analyzed the outcomes. Last but not least, throughout these steps, we were also able to point out opportunities for improvement and issues that previous work did not deal with explicitly. Moreover, we suggested solutions that had good practical results on our simulators.

Overall, much work has been done, with various degrees of success; even with all these guidelines, cloth simulation remains a difficult task. We observed throughout this work that simple models do not work well; at least not without further improvements and handling of special cases that would render the models more complex, and remove their simplicity. We observed the causes and symptoms of different issues encountered such as the instability problem and the various ways to deal with it. On one hand, the methods that we used to maintain system stability such as damping, or the change for more accurate numerical methods, or the direct management of excessive forces, all showed that neither of the two main cloth models we referred to could truly simulate all cloth



behaviors without a bit of tuning here and there. For instance, some simple phenomena (e.g., folds under compression) are tricky to get right, and the previous cloth models necessitated a modification to represent this real cloth behavior. On the other hand, accuracy is needed, but applications require speed; thus efficient algorithms and compromises are both needed, a requirement that is not simple to fulfill. We also observed that the understanding of the underlying physics is needed for good results. In all, it proves that cloth, although a common object in our daily lives is an object whose behaviors are very difficult to model and no simple model represents it fully.

In more detail, we started by presenting a comprehensive overview of Provot's mass-spring model which defines three specific types of springs (structural, shear and flexion) to treat the three fundamental properties of cloth (stretch, shear and bend) respectively. The model also treats the problem of "super-elasticity" and uses inverse dynamics procedures to complete the corrections of this issue. The author of this model also suggested a complete solution for collision detection and response as well as collision optimization techniques and the solution also addressed the coherency of multiple collisions. Then, we presented a detailed picture of Baraff and Witkin's model which defined three energy-based condition functions to model the three respective basic cloth properties. This model emphasized the upgrade of the integration method from an explicit scheme to an implicit scheme to achieve better accuracy and improve the system stability. The algorithm required, however, the computation of force derivatives and a hierarchy of detailed terms to solve for the change in velocity at each time step. The authors also discussed associating the damping to each condition specifically, using a

mass-constraint mechanism, an adaptive time stepping technique and a modified conjugate gradient method to elaborate their model further.

We constructed two simulators based on these two models and produced two respective sets of simulation test results to compare and analyze the differences between the two models. We were able to observe and explain the many visual differences and behavioral dissimilarities by looking at details such as the mechanisms and formulations used to model the properties of cloth as well as the mesh structures. We concluded that Baraff and Witkin's model had a slight advantage over a mass-spring model because its formulation for the bend resistance property was more efficient, and in general, it had less inter-mechanism interferences. But, it still had specific issues to be solved for the stretch condition. As for the integration schemes, it was theoretically clear that the implicit method would provide more accuracy and stability to the system. However, in practice, our results did not show conclusive gains against the performance trade-off up to that point. Thus, we then decided to study and try alternative integration methods further for our cloth simulators.

We tried the 4<sup>th</sup> order Runge-Kutta, the Verlet scheme, and a mixed explicit/implicit integration scheme as well. The first and third alternatives were clearly advantageous methods over a regular explicit Euler integration method. The mixed explicit/implicit scheme also allowed us to visibly observe the benefit of an implicit method in terms of system stability by reducing the performance overhead that a full implicit method required. In the trend of addressing the system stability issue, we pointed out the usual causes of system instability are unhandled excessive forces and

suggested a force transfer mechanism to treat the excessive forces and reduce the instability without having to switch integration methods.

Next, we performed a brief study for finding collision detection and response methods that would avoid solving cubic equations. However, we concluded in the end that using a work-around for solving cubic equations and using Provot's complete solution would be a more advantageous decision because the alternative methods were only able to treat static triangles.

Our various simulation tests and implementations allowed us to suggest two additional improvements to the previous cloth models. First, we extended the mass-spring model with a method based partly on the energy minimization principle used in continuum finite-elements models to reproduce the formation of curves under compression. Our method can be summarized as a thorough complement to the previous bend resistance formulation, and a re-definition of the relationships between cloth properties and the mechanisms utilized to model them. We showed in theory that our new formulation minimizes the compression energy better than the previous formulation did; and, in practice, we showed with our implementations that the cloth object simulated behaves more like real cloth with our new method to simulate curves.

Our second improvement to the cloth models borrows the mass-constraint enforcement method presented in Baraff and Witkin's work and combines it with an analogy to the formulation of the deflection curve of a beam bent under load to construct a new constraint mechanism. The new mechanism simulates the behavior of cloth resisting forces that attempt to displace it from a constrained alignment when the cloth is

stretched and constrained. Also, we complemented this enforcement formulation with a realignment method to ensure that the cloth behaves more realistically than with the previous models under these circumstances.

Last, we mentioned the additional difficulties involved in the simulation of complex garments as opposed to the simulation of simple cloth. One of the key topics is the modeling of wrinkles and creases. We referenced some past strategies and, finally, we presented our approach with a set of hypotheses on how to extend a current mass-spring model for the simulation of wrinkles and creases.

Previously, we stated that due to time and resource constraints, we opted to simplify our implementations of Baraff and Witkin's model by modifying a few aspects. We plan to implement a cloth simulator that would correspond to their model more entirely in the future. The model would also implement the bend resistance with all its implicit terms and would also separate the three damping forces as suggested by the authors of this model. Our cloth simulators used small mesh sizes; so, we were able to obtain satisfactory results with the speed without implementing any optimization on cloth-cloth and cloth-body collision detection and handling. But, our future work will certainly involve the studying and implementation of fast collision processing algorithms, especially for cloth-cloth collisions, that was claimed to be a bottleneck in cloth simulations by several work. We also plan to find out more ways of "cheating" to avoid frequent expensive calculations for the simulations of different cloth behaviors without having to trade off much accuracy. The formation of folds and wrinkles on cloth is

definitely a subject we would like to improve further in our future work; and, we aim to involve more complex garments in the simulation of the cloth behaviors.

## Bibliography

- Baraff D., Witkin A., *Large Steps in Cloth Simulation*, Computer Graphics (In Conference Proceedings of ACM SIGGRAPH), 43-54, 1998.
- Bridson R., Anderson J., Fedkiw R. *Robust Treatment of Collisions Contact and Friction for Cloth Animation*. SIGGRAPH'02, San Antonio, July 21-26, 2002.
- Bridson R., Marino S., Fedkiw R. *Simulation of Clothing with Folds and Wrinkles*. Eurographics/SIGGRAPH Symposium on Computer Animation, 2003.
- Carignan M., Yang Y., Magnenat-Thalmann N., Thalmann D. *Dressing Animated Synthetic Actors with Complex Deformable Clothes*. In *Proceedings of SIGGRAPH'92*, Computer Graphics, 1992, Vol.26, No.2, 99-104.
- Cordero J. *Realistic Cloth Animation Using the Mass-spring Model*. Department of Computer Languages and Systems, University of Seville, 2001.
- Cordier F., Magnenat-Thalmann N., *Real-time Animation of Dressed Virtual Humans*. EUROGRAPHICS, Volume 21, no.3, 327-335, 2002.
- Duncan B. *Fiber Facts*. Information Sheet 1250, MSU Coordinated Access to the Research and Extension System, Mississippi State University Extension Service, January 10, 2003.
- Fuhrmann A., Grob C., Luckas V. *Interactive Animation of Cloth including Self Collision*. Journal of WSCG, Vol.11, No.1, 141-148. Plzen, Czech Republic, February 3-7, 2003.
- Hadap S., Bangerter E., Volino P., Magnenat-Thalmann N., *Animating Wrinkles on Clothes*, MIRALab, CUI, University of Geneva, Switzerland, 2000.
- House D.H., Breen D.E., *Cloth Modeling and Animation*, A K Peters, Ltd., 2000.

- Jakobsen T. Advanced Character Physics – The Fysix Engine. Proceedings of Game Developer's Conference 2001, San Jose, 2001. URL:  
<http://www.ioi.dk/~tj/publications/AdvancedPhysics.htm> .
- Macri D., *Real-time Cloth*. In Game Developers Conference Proceedings, 2000.
- Magenat-Thalmann N., Adabala N., Fei G. *A Procedural Thread Texture Model*.  
Journal of Graphics Tools, 8(3), 33-40, 2003.
- Magenat-Thalmann N., Adabala N., Fei G. *Visualization of woven cloth*. Eurographics Symposium on Rendering 2003, 178-185, 2003.
- Magenat-Thalmann N., Adabala N., Fei G. *Real-time Rendering of Woven Clothes*.  
VRST'03, October 1-3, 2003, Osaka, JAPAN, 2003.
- Melis, P. *Real-Time Cloth Simulation in a 3D Virtual Environment*, University of Twente, The Netherlands, August 27, 2002.
- Oshita M., Makinouchi A. *Real-time Cloth Simulation with Sparse Particles and Curved Faces*. In Proceedings of Computer Animation 2001, 220-227, Seoul, Korea, November 2001.
- Pritchard D., *Implementing Baraff & Witkin's Cloth Simulation*, University of British Columbia, May 6, 2003.
- Provot, X. (1995), *Deformation Constraints in a Mass-Spring Model to Describe Rigid Cloth Behavior*. In *Graphics Interface '95*, 147-254, Quebec, Canada, 17-19 May 1995.
- Provot, X. (1997), *Collision and self-collision handling in cloth model dedicated to design garments*. Institut National de Recherche en Informatique et Automatique (INRIA), France.

- Shewchuk J.R., *An Introduction to the Conjugate Gradient Method Without the Agonizing Pain*, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA, August 1994.
- Terzopoulos D., Platt J.C., Barr A.H. *Elastically deformable models*. Computer Graphics (Proc.SIGGRAPH), 21:205-214, July 1987.
- Villard J., Borouchaki H., *Adaptive Meshing for Cloth Animation*. Proceedings 11<sup>th</sup> International Meshing RoundTable, Sandia National laboratories, 243-252, September 15-18, 2002.
- Volevich V., Kopylov E., Khodulev A., Karpenko O., *An Approach to Cloth Synthesis and Visualization*. The 7<sup>th</sup> International Conference on Computer Graphics and Visualization, Moscow, Russia, May 21-24, 1997.
- Volino P., Magnenat-Thalmann N., *Efficient self-collision detection on smoothly discretized surface animations using geometrical shape regularity*. In Computer Graphics Forum, volume 13, 155-166, 1994.
- Volino P., Magnenat-Thalmann N. *Developing Simulation Techniques for an Interactive Clothing System*. Virtual Systems and Multimedia (VSMM'97 proceedings), Geneva, Switzerland, 109-118, 1997.
- Volino P., Magnenat-Thalmann N., *Interactive Cloth Simulation: Problems and Solutions*. MIRAlab, University of Geneva, Geneva, Switzerland, 1998.
- Volino P., Magnenat-Thalmann N. *Implementing fast Cloth Simulation with Collision Response*. Computer Graphics International 2000, 257-266, 2000.
- Volino P., Magnenat-Thalmann N., *Comparing Efficiency of Integration Methods for Cloth Simulation*. MIRAlab, University of Geneva, Geneva, Switzerland, 2000.



## APPENDIX A

In this Appendix, we present the extended terms used by the implicit integration scheme presented in section 4.2.3 as well as the definition that relate to the implementation details of the second simulator regarding this implicit method for the second simulator in section 6.2.

### **The Damping Terms**

We begin by detailing the damping terms of an implicit integration system. We previously left off in section 4.2.4 with the equation 4.37 expressing the damping term again as

$$d = -k_d \frac{\partial C(x)}{\partial x} \frac{\partial C(x)}{\partial t}$$

In particular, for a particle  $i$  of the cloth mesh, the damping is expressed with

$$d_i = -k_d \frac{\partial C(x_i)}{\partial x_i} \frac{\partial C(x_i)}{\partial t} ;$$

But, it will no longer simply become

$$d_i = -k_d \frac{\partial C(x_i)}{\partial x_i} \left( \frac{\partial C(x_i)}{\partial x_i} \frac{\partial x_i}{\partial t} \right).$$

It has been noted in David Pritchard's work [PRI03] that computing the damping term  $d_i$

by interpreting  $\frac{\partial C(x_i)}{\partial t}$  directly as  $\left( \frac{\partial C(x_i)}{\partial x_i} \frac{\partial x_i}{\partial t} \right)$  does not yield correct results. Instead,

the derivative of the condition function at a particle  $i$  with respect to time must be reinterpreted as the following summation for all particles  $m$  that affect the condition function value computed at particle  $i$ :

$$d_i = -k_d \frac{\partial C(x_i)}{\partial x_i} \left( \sum_m \frac{\partial C(x_i)}{\partial x_m} \frac{\partial x_m}{\partial t} \right).$$

Now that we have determined the damping term, we must also consider the terms  $\frac{\partial d}{\partial x}$

and  $\frac{\partial d}{\partial v}$  as well because damping is now also contributing to the total forces generated over the cloth object and both the force derivatives with respect to positions and velocities must be calculated in an implicit system.

However, contrarily to the forces generated by the three condition functions, damping forces are not dependent on the positions of particles but only to their velocities.

Consequently, the term  $\frac{\partial d}{\partial x}$  is nil but the second term  $\frac{\partial d}{\partial v}$  is not, and must be detailed.

For each pair of cloth particles  $i$  and  $j$ , applying the chain rule, we would obtain

$$\frac{\partial d_i}{\partial v_j} = -k_d \left[ \frac{\partial^2 C(x_i)}{\partial x_i \partial v_j} \left( \sum_m \frac{\partial C(x_i)}{\partial x_m} \frac{\partial x_m}{\partial t} \right) + \frac{\partial C(x_i)}{\partial x_i} \left( \sum_m \frac{\partial \left( \frac{\partial C(x_i)}{\partial x_m} \frac{\partial x_m}{\partial t} \right)}{\partial v_j} \right) \right].$$

The first term in the expression above containing the second order derivative of the condition function is zero because none of the conditions depends on the particle velocities. And, if we expressed  $v_j$  as  $\frac{\partial x_j}{\partial t}$ , then the derivative of the summation term above is often zero unless  $m$  is equal to  $j$ , and unless particle  $j$  did play a role in the

condition function value of particle i, or it would also be nil. So, particle j must likely reside within the neighborhood of particle i; and possibly, particle j would belong to an adjacent triangle to particle i. Then, this derivative would reevaluate to:

$$\frac{\partial d_i}{\partial v_j} = -k_d \frac{\partial C(x_i)}{\partial x_i} \frac{\partial C(x_j)}{\partial x_j}.$$

### **Terms in the deeper levels of the method hierarchy**

In sections 4.2.3, 4.2.4, and at the start of this appendix, we described condition forces, damping forces as well as force derivatives with respect to positions and velocities. But, the expressions remained in their general forms, and they are insufficiently detailed for proper implementation guidelines required in section 6.2. We will provide the more detailed definitions of these terms for each of the three conditions.

#### The stretch condition:

In section 6.2, we presented the equation 6.1 for the stretch condition forces

$$f_m = -k_s \frac{\partial C_u}{\partial P_m} C_u + -k_s \frac{\partial C_v}{\partial P_m} C_v.$$

In the above equation, the stretch condition terms in u and v are simply computed as:

$$C_u = a*( \|W_u\| - b_u) \text{ and } C_v = a*( \|W_v\| - b_v)$$

where  $b_u$  and  $b_v$  are often set to 1 like in the second simulator.

And, with equation 6.2, we defined the condition derivatives in direction of u and v again

$$\text{as } \frac{\partial C_u}{\partial P_m} = \alpha \frac{\partial W_u}{\partial P_m} \frac{W_u}{\|W_u\|} \quad \text{and} \quad \frac{\partial C_v}{\partial P_m} = \alpha \frac{\partial W_v}{\partial P_m} \frac{W_v}{\|W_v\|}.$$

So, the terms that currently need to be detailed are  $\frac{\partial W_u}{\partial P_m}$ ,  $\frac{\partial W_v}{\partial P_m}$ ,  $W_u$ , and  $W_v$ .

We can obtain from equation 4.19 that the terms

$$W_u = \frac{\Delta x_1 \Delta v_2 - \Delta x_2 \Delta v_1}{\Delta u_1 \Delta v_2 - \Delta u_2 \Delta v_1}, \quad \text{and,}$$

$$W_v = \frac{-\Delta x_1 \Delta u_2 + \Delta x_2 \Delta u_1}{\Delta u_1 \Delta v_2 - \Delta u_2 \Delta v_1}.$$

These two terms are however more often used in their normalized forms.

Also, the divisor  $(\Delta u_1 \Delta v_2 - \Delta u_2 \Delta v_1)$  is also equal to twice the area of the triangle in uv-space i.e.  $2a$ . Note that the terms  $x_m$  and  $P_m$  are used interchangeably to represent the position of particle m in world space.

For the derivative terms of  $W_u$  and  $W_v$  with respect to the particle position  $x_m$ , there are three cases per triangle where m could take the three values i,j, and k.

For the case where  $m=i$ ,

$$\frac{\partial W_u}{\partial P_m} = \frac{-\Delta v_2 + \Delta v_1}{2a} \quad \text{and} \quad \frac{\partial W_v}{\partial P_m} = \frac{\Delta u_2 - \Delta u_1}{2a}.$$

For the case where  $m=j$ ,

$$\frac{\partial W_u}{\partial P_m} = \frac{\Delta v_2}{2a} \quad \text{and} \quad \frac{\partial W_v}{\partial P_m} = \frac{-\Delta u_2}{2a}.$$

Finally, for the case where  $m=k$ ,

$$\frac{\partial W_u}{\partial P_m} = \frac{-\Delta v_1}{2a} \quad \text{and} \quad \frac{\partial W_v}{\partial P_m} = \frac{\Delta u_1}{2a}.$$

Next, the derivative of forces with respect to positions shown in equation 4.26 would only contain a single term we have not yet determined for the stretch condition: the condition's second derivative with respect to positions of any pair of cloth particles  $i$  and  $j$ ,  $\frac{\partial^2 C(x_i)}{\partial x_i \partial x_j}$ . Again, for the stretch condition, it is preferable to split this definition in the

directions  $u$  and  $v$ .

$$\frac{\partial^2 C_u(x_i)}{\partial x_i \partial x_j} = \frac{\alpha}{\|W_u\|} \frac{\partial W_u}{\partial x_i} \frac{\partial W_u}{\partial x_j} \left( I - \left( \frac{W_u}{\|W_u\|} \right) \left( \frac{W_u}{\|W_u\|} \right)^T \right), \quad \text{and,}$$

$$\frac{\partial^2 C_v(x_i)}{\partial x_i \partial x_j} = \frac{\alpha}{\|W_v\|} \frac{\partial W_v}{\partial x_i} \frac{\partial W_v}{\partial x_j} \left( I - \left( \frac{W_v}{\|W_v\|} \right) \left( \frac{W_v}{\|W_v\|} \right)^T \right)$$

where  $\alpha = a^{3/4}$  and  $I$  is a 3x3 identity matrix.

### The Shear Condition:

The first derivative of the shear condition expressed in equation 6.3 does not contain any

new terms to be defined at this point:  $\frac{\partial C}{\partial P_m} = \alpha \left( \frac{\partial W_u}{\partial P_m} W_v + \frac{\partial W_v}{\partial P_m} W_u \right)$ . However, its

second derivative can be defined as follows.

$$\frac{\partial^2 C_u(x_i)}{\partial x_i \partial x_j} = \alpha \left( \frac{\partial W_u}{\partial x_i} \frac{\partial W_v}{\partial x_j} + \frac{\partial W_u}{\partial x_j} \frac{\partial W_v}{\partial x_i} \right).$$

We also noted in implementation that for the coordinate variables  $s, t \in \{x, y, z\}$ ,

the value of  $\frac{\partial^2 C_u(x_i)}{\partial x_{i,s} \partial x_{j,t}}$  is zero unless  $s=t$ .

### The Bend Condition:

To define the terms  $\frac{\partial \sin\theta}{\partial P_m}$  and  $\frac{\partial \cos\theta}{\partial P_m}$  shown in equation 6.4 of the first derivative of

the bend condition, we adopted Pritchard's implementation strategy [PRI03].

Let the term  $\hat{v}$  of a vector  $v$  represent its normalized form i.e.  $\hat{v} = v/|v|$ .

$$\frac{\partial \cos\theta}{\partial P_m} = \frac{\partial(\hat{n}^A \bullet \hat{n}^B)}{\partial x_m} = \frac{\partial \hat{n}^A}{\partial x_m} \bullet \hat{n}^B + \hat{n}^A \bullet \frac{\partial \hat{n}^B}{\partial x_m},$$

and,

$$\frac{\partial \sin \theta}{\partial P_m} = \frac{\partial (\hat{n}^A \times \hat{n}^B) \cdot \hat{e}}{\partial x_m} = \left( \frac{\partial \hat{n}^A}{\partial x_m} \times \hat{n}^B + \hat{n}^A \times \frac{\partial \hat{n}^B}{\partial x_m} \right) \cdot \hat{e} + (\hat{n}^A \times \hat{n}^B) \cdot \frac{\partial \hat{e}}{\partial x_m}$$

where we are working on the two adjacent triangles A and B sharing the common edge e.

Obviously,  $\frac{\partial \hat{n}^A}{\partial x_m} = \frac{1}{\|n^A\|} \frac{\partial n^A}{\partial x_m}$ ; and similarly for  $n^B$  and e. Then, for each coordinate s

$\in \{x, y, z\}$ , we then define the three terms:

$$\frac{\partial n^A}{\partial x_{m,s}} = S_s(q^A_m), \quad \frac{\partial n^B}{\partial x_{m,s}} = S_s(q^B_m), \quad \text{and} \quad \frac{\partial e}{\partial x_{m,s}} = q^e_m I_s,$$

where  $I_s$  is the  $s^{\text{th}}$  column of the 3x3 identity matrix and the meaning of  $S_s(v)$ , where v is a certain vector, is the transpose of the  $s^{\text{th}}$  row of  $S(v)$  into column vectors. And, for a vector v, the matrix  $S(v)$  takes the following definition:

$$S(v) = \begin{bmatrix} 0 & -v_z & v_y \\ v_z & 0 & -v_x \\ -v_y & v_x & 0 \end{bmatrix}$$

And, as defined above, the vector v will be one of the following three:

For the triangle A, it is  $q^A = [x_2 - x_1, x_0 - x_2, x_1 - x_0, 0]$ .

For the triangle B, it is  $q^B = [0, x_2 - x_3, x_3 - x_1, x_1 - x_2]$ .

And, for the edge e,  $q^e = [0, 1, -1, 0]$ .

The details for the second order derivatives of this condition will not be presented since we decided to leave the bend condition under the explicit scheme for simpler implementation of the second simulator. But, they can be found in the work of Pritchard [PRI03] for reference.

## **APPENDIX B**

In this appendix, we present the third alternative collision detection and response solution mentioned in section 7.3 in more detail.

### **Collision Prevention with Springs**

Springs and similar mechanisms can be used to treat collision, especially for the case of deformable objects like cloth. The approach becomes more of collision prevention than collision detection and response.

The use of stiff springs for handling collision is mentioned in many references, but most of the time, the approach is not detailed very much. We assume the lack of details for this approach resides in its simplicity and in some of the disadvantages that does not make it an efficient approach computationally. Nonetheless, it remains an alternative solution for handling collision detection and response all in one. To adopt a similar approach, we use repulsion springs. The flexion springs in Provot's model would correspond to what are closest to repulsion springs we have presented so far, but, unlike the other types of springs, repulsion springs, as their name indicates it, do not exert pulling forces but only repulsion forces. Additionally, repulsion springs are defined between each node and every other node in the cloth mesh that is not directly adjacent or diagonally adjacent. Thus, without further optimization techniques, if we let  $N$  be again the total number of nodes in this mesh, this approach already requires the establishment of  $N \times (N-9)$  repulsion springs. The complexity of this approach is  $O(N^2)$ , and we have



stated that such a complexity would either limit the number of nodes we could use or hinder the performance of the simulation.

Yet there is another reason, and possibly an advantage, to use repulsion springs other than for collision detection. All the methods presented so far in this work to model the properties of cloth only treat relationships between adjacent nodes or at most, nodes that are one node apart. These mechanisms and conditions are insufficient to model the behavior of cloth when it buckles up. When cloth is buckled up, it is possible to observe repulsion forces working against the buckling up. Neither the flexion springs from the first model nor the bend condition from the second were able to appropriately simulate this repulsion force behavior. And actually, neither could be any stronger because the bending resistance property defined between pairs of adjacent triangles is relatively a very permissive property of cloth. But, when cloth is buckled up, there is more than just a single dihedral angle in play; there are numerous folds and curves forming up within a restrained area.

Now, going back to the collision detection and response functionality of repulsion springs, we can simulate the thickness of the fabric with these new mechanisms. Although the initial terminology used is “spring”, a repulsion spring needs not to be limited to the linear type of springs. Repulsion springs can be made to declare an approaching colliding node once their length is below a certain value, and a collision response process can start. The important point is that these new springs must be designed to prevent nodes from approaching one another by more than that thickness distance. If a node were to head towards the middle-point of a triangle surface, the three

repulsion springs going from that node to the three vertices of the triangle would then act altogether to prevent the node from approaching more than the minimum distance allowed.

There is however a problem with using springs for collision detection. If the spring's range of activity started much larger than the thickness distance value, or in other words, if the spring started pushing back and preventing a node from coming closer by applying forces too soon, two concerns will be created. The first is that the repulsion springs will interfere with in-plane forces, just like the possible problem we had raised up with flexion springs earlier in Section 7.1.2. The second concern is that the motion of incoming colliding nodes will start to diminish too soon as if there were entering a slow-down zone when approaching the cloth surface and that is not very realistic. On the contrary, if we decided to reduce the range of effect of the repulsion springs to a value nearing the thickness value, then we risk being unable to intercept nodes incoming with high velocities. These nodes would go through the cloth surface within one time interval, and repulsion springs would be unable to correct these artifacts. This is a disadvantageous trade-off involved with the use of springs for collision prevention. Again, this approach is more of collision prevention than it is for detection and correction.

With the problem presented in the previous paragraph, it consequently becomes difficult to use the same functionality of repulsion springs that simulates the thickness of cloth to also handle the repulsion behavior occurring when cloth buckles up. We must add a second functionality to repulsion springs to handle this behavior. The phenomenon we are referring to when we say that cloth is buckled up is observable in real life for

instance when someone pulls up their sleeves, many folds and curves are formed around the level of the elbow. The formation of multiple curves clogged up together appears to provoke a tendency to push back the farther parts brought close. In other words, after enough folds and curves have been formed between two distant nodes, there exists enough force to yield a significant push back. But first, the relationship between the number of folds there are between two nodes and the repulsion strength between them does not seem to be linear; it starts incredibly small and increases very rapidly. The second observation is that a fold is only accounted into this buckling phenomenon if the two nodes brought close by it were not originally too far from each other. For instance, if we folded a large piece of cloth over itself evenly a few times, no significant repulsion force can be observed. On the other hand, if we folded a tenth of that cloth surface over itself for the same number of times, we might be able to observe repulsion occurring depending on the fabric's resistance to bending, thickness, and weight.

We believe that this behavior can be related to the bend resistance property of cloth along with its thickness and weight factors. A thicker cloth would resist bending much more. More of this aspect of cloth's bending resistance is discussed in detail in Section 7.4. A heavier fabric however, would allow weaker repulsion. That remains consistent with our formulation of the behavior of cloth against buckling up since pushing forces are generated from such a phenomenon, the larger the mass of each node, the less motion is instilled.

For the purpose of using repulsion springs to simulate the repulsion behavior against buckling up, we consequently do not need to define this second functionality for springs whose two nodes are distanced too far apart. The distance to which this

functionality would consider a fold to be part of the buckling up resistance depends on once again the properties of the fabric itself. To sum up, a repulsion spring is defined between each node and every other node that is not directly adjacent to it in order to prevent contact between nodes as well as contact between a node and a cloth triangle. For each node, only the springs that connect it to the nodes within a certain distance are given the second functionality to resist buckling up.

That second functionality given to repulsion springs to resist buckling can be formulated as follows. But first, let us remind that the first functionality of repulsion springs that prevents contact uses the same length value of activation for every spring. It does not matter how far the two nodes on the spring originally were; once the spring is shortened to that length value, the two nodes will be applied forces or corrections to keep them from approaching further. And, the functionality can be modified to prevent a node from going closer to a triangle's surface.

Now, unlike that first functionality, the second one has an activation length that is proportional to the original distance between the two nodes of the spring. And, the forces produced by this second functionality will be accumulated for the two nodes linked to the spring, but those forces will not be immediately be added into the total forces applied to these nodes; they will be stored independently until another condition described below is fulfilled. Let  $N_{sf}$  be the number of springs whose second function is activated. The simulation keeps track of  $N_{sf}$  at each node. When  $N_{sf}$  exceeds a certain value, there are enough folds and curves happening nearby to push that node.  $N_{sf}$  and the number of folds are generally not the same.  $N_{sf}$  is again the number of springs whose length is equal or below its respective minimum value while a fold could cause a node to get too

close to a cloth triangle; and therefore, a single fold could possibly increase a node's Nsf value by three. However, once Nsf reaches the value four for a node, at least two folds have been formed in the proximity of that node. Then, for instance, a fabric with a certain bending resistance property could require a minimum Nsf value of seven, and therefore three folds, within a ten-node radius range from a node before the repulsion forces are accounted for that node. Once a node has an Nsf value greater or equal to seven, then the previously stored forces from the seven repulsion springs are added to the forces applied on that node, and subsequent repulsion forces can be directly added also to the forces applied on that node. At the end of each time frame, the count is reset to zero for every node and the repulsion forces temporarily accumulated are all reset whether they were taken into effect or not.

To summarize, repulsion springs can be used to prevent contact between parts of cloth and can also be utilized to simulate the resistance against buckling up; and, we need not to worry about which side of the cloth is entering into contact with which side. Yet, this preventive approach cannot guarantee to catch or prevent every collision, and it cannot make corrections if a collision were not prevented. Nonetheless, it remains a simple approach.

## APPENDIX C

The normal preconditioned CG (Conjugate Gradient) method takes a symmetric positive-definite matrix  $A$ , a preconditioning matrix  $P$  of the same dimension as  $A$  and iteratively solves  $Ax=b$ ; more specifically in this case of a cloth simulation with implicit integration, the system to solve is  $A\Delta v=b$ . Shewchuk [SHE94] provides a detailed description of CG. Briefly, the preconditioner  $P$  must be easily invertible and speeds up the convergence of the method. As seen previously, constraint handling uses the  $W$  matrix instead of the  $M^{-1}$  to filter out the velocities in the constrained directions. The diagonal entries of  $W$  are  $W_{ii} = \frac{1}{m_i} S_i$  where  $S_i$  is defined depending on how many degrees of freedom are not constrained for the particle  $i$ . Let  $ndof(i)$  represent the number of degrees of freedom left for particle  $i$ . Then, let  $I$  be the identity matrix of same size as  $W$ ,  $p_i$  and  $q_i$ , two mutually orthogonal and prohibited directions to particle  $i$ , and define:

$$S_i = \begin{cases} I & \text{if } ndof(i)=3 \text{ or equivalently, this particle isn't constrained,} \\ (I - p_i p_i^T) & \text{if } ndof(i)=2, \text{ the particle is constrained along direction } p_i, \\ (I - p_i p_i^T - q_i q_i^T) & \text{if } ndof(i)=1, \text{ both directions } p_i \text{ and } q_i \text{ are prohibited,} \\ 0 & \text{if } ndof(i)=0, \text{ the particle is completely constrained in 3D} \\ & \text{space by three mutually orthogonal directions.} \end{cases}$$

Next, a filtering procedure over any given vector  $a$  is defined to constrain  $a$  in the same way  $W$  is used to constrain the cloth particles as follows:

procedure **filter**( $\mathbf{a}$ )

for  $i = 1$  to  $n$

$$\hat{\mathbf{a}}_i = \mathbf{S}_i \mathbf{a}_i$$

return  $\hat{\mathbf{a}}$

The term integer  $n$  above is the number of possible degrees of freedom a particle has or number of coordinates. Now, an invariant velocity constraint  $\mathbf{z}_i$  particular to each particle will be set at the beginning and maintained. Again  $\mathbf{z}_i$  represents a particle's specific velocity constraint such as the case of being temporarily attached to an external object whose velocity would be  $\mathbf{z}_i$  and all of them define  $\mathbf{z}$  for the entire cloth.

The modified Preconditioned Conjugate Gradient Method is given:

| line | Pseudo-code                                                                   | Details/Explanations                                                                                                                                |
|------|-------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
| 1    | procedure modified-pcg                                                        | If we let $\mathbf{x}_i$ be the $i^{\text{th}}$ step of the method, $\mathbf{d}_i$ , its search direction,                                          |
| 2    | $\Delta \mathbf{v} = \mathbf{z}$                                              | then, this is the invariant $\mathbf{x}_0 = \Delta \mathbf{v} = \mathbf{z}$ ,                                                                       |
| 3    | $\delta_0 = \text{filter}(\mathbf{b})^T \mathbf{P} \text{filter}(\mathbf{b})$ | the term $\delta$ will be explained on line 6                                                                                                       |
| 4    | $\mathbf{r} = \text{filter}(\mathbf{b} - \mathbf{A}\Delta \mathbf{v})$        | the starting (filtered) residual $\mathbf{r}_0 = \mathbf{r}_i = \mathbf{b} - \mathbf{A}\mathbf{x}_i$                                                |
| 5    | $\mathbf{c} = \text{filter}(\mathbf{P}^{-1}\mathbf{r})$                       | $\mathbf{c}$ is the search direction $\mathbf{d}_i$ , here it is the starting one $\mathbf{d}_0$ and it is preconditioned.                          |
| 6    | $\delta_{\text{new}} = \mathbf{r}^T \mathbf{c}$                               | Basically, this term $\delta$ is the inner product of the residual and its filtered preconditioned self.                                            |
| 7    | while $\delta_{\text{new}} > \epsilon^2 \delta_0$                             | Instead of explicitly defining the stop condition as the moment when the residual becomes smaller than a fraction number $\epsilon$ of the starting |

|    |                                                                                 |                                                                                                                                                                           |
|----|---------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|    |                                                                                 | residual, the term $\delta$ is used instead. The idea remains the same.                                                                                                   |
| 8  | { $q = \text{filter}(Ac)$                                                       | applying A to the search direction i.e. $q = Ad_i$                                                                                                                        |
| 9  | $\alpha = \frac{\delta_{\text{new}}}{c^T q}$                                    | This is the step size along the search direction $d_i$ to be taken.                                                                                                       |
| 10 | $\Delta v = \Delta v + \alpha c$                                                | Proceed to next step $x_{i+1} = x_i + \alpha_i d_i$                                                                                                                       |
| 11 | $r = r - \alpha q$                                                              | its new residual $r_{i+1} = r_i - \alpha_i (Ad_i)$                                                                                                                        |
| 12 | $s = P^{-1} r$                                                                  | $s$ is the new preconditioned residual that will serve at the end to obtain the new search direction                                                                      |
| 13 | $\delta_{\text{old}} = \delta_{\text{new}}$                                     | These three lines 13 to 15 would be equivalent to the following two steps. First, the Gram-Schmidt constant $B_{i+1} = \frac{r_{i+1}^T M^{-1} r_{i+1}}{r_i^T M^{-1} r_i}$ |
| 14 | $\delta_{\text{new}} = r^T s$                                                   | Then, there the new filtered search direction                                                                                                                             |
| 15 | $c = \text{filter}(s + \frac{\delta_{\text{new}}}{\delta_{\text{old}}} c)$<br>} | $d_{i+1} = r_{i+1} + B_{i+1} d_i$ is obtained.                                                                                                                            |

Table C.1. The Modified Preconditioned Conjugate Gradient Method.

The preconditioner  $P$  used in this model is simply a diagonal matrix whose entries are

$P_{ii} = \frac{1}{A_{ii}}$  so that the products involving  $P^{-1}$  are trivial. It is observable that if all the calls

to the procedure filter were removed and if the invariant starting constraint were zero, the method would be exactly the standard Preconditioned Conjugate Gradient method.