Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2006)
M.-P. Cani, J. O'Brien (Editors)

Simple Linear Bending Stiffness in Particle Systems

Pascal Volino and Nadia Magnenat-Thalmann

MIRALab, University of Geneva, Switzerland.

Abstract

In cloth simulation particle systems models, bending stiffness involve forces which are either represented as additional springs, or as out-of-plane forces along surface normals. While the former solution is quite inaccurate for both small and large curvatures, the latter requires significant computation and is unsuited for fast computation required by real-time or interactive systems. We present a linear approach which combines simplicity and accuracy, being perfectly suited for the implicit integration schemes used in particle systems.

Categories and Subject Descriptors: I.6 [Computing Methodologies]: Simulation and Modeling; I.3.5 [Computing Methodologies]: Computer Graphics - Computational Geometry and Object Modeling - Physically-Based Modeling.

1. Introduction

Tensile stiffness is fairly easy to simulate, as it involves computing deformations and forces within mesh elements. On the other hand, the simulation of bending stiffness is more complicated, as several adjacent mesh elements have to be considered simultaneously. Furthermore, this necessitates the action of out-of-plane forces that are usually more expensive to compute than in-plane forces.

Several solutions have been proposed in the literature, representing two main approaches. The first is to use crossover springs that extend the surface, opposing transversal bending [Pro95] [EWS96]. The second is to evaluate precisely the angle between adjacent mesh elements and to create between them normal forces that oppose this angle through opposite bending momentum [GHDS03] [BMF03] [VCT95] [TW06]. This approach can reach similar accuracy as grid continuum-mechanics [TPBF87] [BHW94] and grid particle system derivatives [BW98] which are fairly complex to evaluate.

The crossover spring approach (Fig.1 top) is believed to be the simplest. It is typically implemented in spring-mass mechanical representations, allowing a homogeneous simulation system. Unfortunately, this approach is also very inaccurate. First, when bending is low, crossover springs are quite unable to produce significant out-of-plane forces. Therefore, they are unsuited for the simulation of surfaces with high bending stiffness with an acceptable accuracy. Furthermore, they highly interfere with the tensile stiffness of the surface. Another issue is their "side-flipping" that may occur in highly bent situations, producing collapses and very large nonsymmetrical deformations, a situation which may be exacerbated with the use of nonsymmetrical mesh elements. This is why this approach is mostly used with regular grid representations of the surface [Pro95] [EWS96]. On the other hand, the normal force approach (Fig.1 bottom) can accurately represent bending stiffness, and there is also no significant interference with the tensile stiffness of the surface. However, the computational price of this method is much higher. First, evaluation of polygon normals is required, and evaluation of the bending angle often goes through significant computation [BMF03] [GHDS03] often involving trigonometric functions. Then, the preservation of total rotational momentum requires a constant recomputation of the coefficients distributing the forces over the particles according to the current position of the particles. Therefore, the computation of the Jacobian required for implicit integration methods may get fairly complicated, and is often carried out through approximations that degrade the numerical performance.

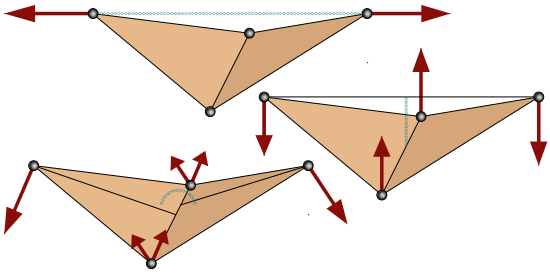

Fig.1: Three ways for creating bending stiffness in a triangle mesh: Using tensile crossover springs over mesh edges (top), using forces along triangle normals (bottom), and, as we propose, using forces evaluated from a weighted sum of vertex positions (right).

Through this work, we propose an alternative approach which combines fairly good accuracy for representing quantitative bending stiffness with a very simple and efficient computational procedure.

The idea of our method is the following: We compute a "bending vector" that represents the bending of the surface through a simple linear combination of particle positions, and we then redistribute this vector as particle forces according to the bending stiffness of the surface (Fig.1 right). We will show that this scheme preserves total translational and rotational momentum conservation without the need of recomputing the distribution

coefficients according to the current position of the particles. This leads to a very simple computation process which is perfectly linear, and thus very well adapted to implicit numerical integration.

In the following, we describe the our linear bending stiffness model using irregular triangle meshes, which correspond to the most general context used in simulation of deformable surfaces. However, all concepts exposed hereafter can be transposed without major difficulties to simulations that are based on surface representations made of regular grids.

2. Computational Details

We start from two adjacent triangles ($\mathbf{P_A}$,$\mathbf{P_C}$,$\mathbf{P_D}$) and ($\mathbf{P_B}$,$\mathbf{P_D}$,$\mathbf{P_C}$) as shown in Fig.2. Their common edge has a length noted $\mathbf{l}$ and their respective heights relatively to vertices $\mathbf{P_A}$ and $\mathbf{P_B}$ are noted $\mathbf{h_A}$ and $\mathbf{h_B}$.

2.1. Measuring Local Curvature

The two adjacent triangles approximate a curved surface that contains the four vertices of the two triangles, and we assume that the surface is only curved *orthogonally* to the edge ($\mathbf{P_C}$,$\mathbf{P_D}$) (Fig.2 left). This is indeed not an obvious assumption, since *any kind* of surface curvature may produce some bending around the edge. However, this choice can be assumed as being the best, as the direction of the edge bending matches the curvature of the surface.

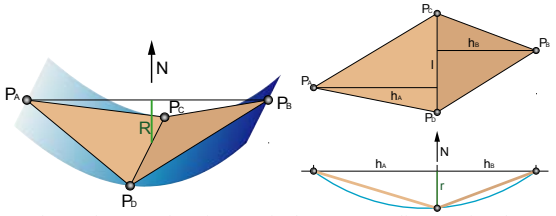

Fig.2: The curved surface equivalent to two adjacent triangles (left), and the computation of its curvature (right).

Our goal is now to estimate this curvature from the *height difference* noted $\mathbf{r}$ between the edges ($\mathbf{P_A}$,$\mathbf{P_B}$) and ($\mathbf{P_C}$,$\mathbf{P_D}$) (Fig.2). As we restrict ourselves to linear bending, we assume that the bending stiffness remains constant whatever the amount of curvature, and therefore we evaluate it assuming the edge angle between the adjacent triangles remain small. In these conditions, we can "measure" the curvature $\boldsymbol{\gamma}$ of the surface by fitting a quadric polynomial having a second-order coefficient $\boldsymbol{\gamma}/\mathbf{2}$ through the three points ($-\mathbf{h_A}$,$\mathbf{0}$), ($\mathbf{0}$,$-\mathbf{r}$), ($\mathbf{h_B}$,$\mathbf{0}$) as follows (Fig.2 right):

$$\begin{aligned} \tfrac{1}{2}\gamma h_A^2 - \beta h_A &= r \\ \tfrac{1}{2}\gamma h_B^2 + \beta h_B &= r \end{aligned} \qquad (1)$$

Solving this system, we obtain the curvature:

$$\gamma = \frac{2\,r}{h_A\,h_B} \qquad (2)$$

Now, we need to compute the height difference $\mathbf{r}$ from the current position of the triangles. Again in the context of small edge angle, this is approximated through the projected length of a *bending vector* $\mathbf{R}$ on the approximate normal of the surface $\mathbf{N}$ (normalized to unit length) so as (Fig.2):

$$r = R \cdot N \qquad (3)$$

The bending vector $\mathbf{R}$ indeed represents a kind of "second-order deformation difference" between the two elements, and its normal component represents the actual surface bending. It can be computed as a simple linear combination of vertex positions, as follows:

$$R = \alpha_A\,P_A + \alpha_B\,P_B + \alpha_C\,P_C + \alpha_D\,P_D \qquad (4)$$

The coefficients are computed through the segment intersection rule (Fig.3):

$$\begin{aligned} \alpha_A &= \frac{|N_B|}{|N_A|+|N_B|} = \frac{h_B}{h_A+h_B} & \alpha_C &= -\frac{|N_D|}{|N_C|+|N_D|} \\ \alpha_B &= \frac{|N_A|}{|N_A|+|N_B|} = \frac{h_A}{h_A+h_B} & \alpha_D &= -\frac{|N_C|}{|N_C|+|N_D|} \end{aligned} \qquad (5)$$

Using the normals:

$$\begin{aligned} N_A &= (P_A - P_C)\wedge(P_A - P_D) & N_C &= (P_C - P_B)\wedge(P_C - P_A) \\ N_B &= (P_B - P_D)\wedge(P_B - P_C) & N_D &= (P_D - P_A)\wedge(P_D - P_B) \end{aligned} \qquad (6)$$

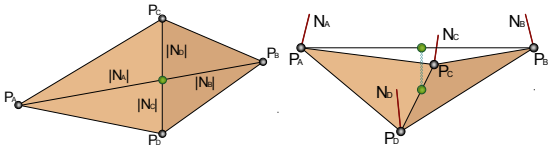

Fig.3: Computing the particle weights from the polygon normals using the segment intersection rule.

2.2. Applying Bending Momentum

The main idea of our linear bending stiffness scheme is to apply forces on the particles that directly oppose the bending vector $\mathbf{R}$ of the current deformation, *without* projection along $\mathbf{N}$, or any other intermediate computations that would explicitly evaluate the actual values of the bending strain and stress.

Thus, we consider that the bending forces $\mathbf{F_A}$, $\mathbf{F_B}$, $\mathbf{F_C}$, $\mathbf{F_D}$ are applied on the vertices $\mathbf{P_A}$, $\mathbf{P_B}$, $\mathbf{P_C}$, $\mathbf{P_D}$ respectively along the vector $\mathbf{R}$. That can be done as follows, with a stiffness coefficient $\boldsymbol{\lambda}$ that would bring adequate scaling:

$$\begin{aligned} F_A &= -\lambda\,\alpha_A\,R & F_C &= -\lambda\,\alpha_C\,R \\ F_B &= -\lambda\,\alpha_B\,R & F_D &= -\lambda\,\alpha_D\,R \end{aligned} \qquad (7)$$

This distribution, which uses the same coefficients as (4), has been chosen for satisfying total mechanical momentum conservation in the system, as detailed in Section 2.3.

Finally, we need to compute the value of the stiffness coefficient $\boldsymbol{\lambda}$ according to the linear bending stiffness modulus $\boldsymbol{\mu}$ of the surface and the geometry of the triangles.

The bending momentum created by the forces $\mathbf{F_A}$ and $\mathbf{F_B}$ applied on respectively $\mathbf{P_A}$ and $\mathbf{P_B}$ around the edge ($\mathbf{P_C}$,$\mathbf{P_D}$) can be expressed from the height difference $\mathbf{r}$, through (7), (5) and (3) as follows:

$$h_A\,F_A\cdot N = -h_B\,F_B\cdot N = \lambda\,\frac{h_A\,h_B}{h_A+h_B}\,R\cdot N = \lambda\,\frac{h_A\,h_B}{h_A+h_B}\,r \qquad (8)$$

The bending momentum also results from the bending stiffness modulus μ of the bent surface of curvature γ applied over the length l of the edge ($\mathbf{P_C}$,$\mathbf{P_D}$). From this, using (2):

$$l\,\mu\,\gamma = \frac{2\,l\,\mu\,r}{h_A\,h_B} \tag{9}$$

A non-obvious issue is to take into account how adjacent edge bends combine together for describing the actual surface curvature. Energetic considerations, mainly detailed by [GHDS03] with results from [MDSB03] suggest that λ should be evaluated by equating (8) with *one third* of (9). Therefore:

$$\lambda = \frac{2}{3}\frac{h_A + h_B}{(h_A\,h_B)^2}\,l\,\mu \tag{10}$$

2.3. Checking Momentum Conservation

Whatever the expected accuracy of the represented stiffness, an essential feature of any mechanical modeling of internal forces is the conservation of total translational and rotational mechanical momentum. Failing to comply with this will produce unrealistic "ghost forces" that can bring mechanical objects to unrealistic equilibrium postures or perpetual motion.

This question is straightforward for spring-based models: Any exactly aligned opposite forces have a null translational and rotational mechanical momentum contribution. However, the normal-based models need a constant update of the force distribution coefficients according to the current positions of the particles for ensuring null mechanical momentum contribution.

Regarding our model, the question is trivial for the translational momentum. Since we have from (5):

$$\alpha_A + \alpha_B + \alpha_C + \alpha_D = 0 \tag{11}$$

Using (7), we have the following translational momentum contribution:

$$\begin{aligned} &F_A + F_B + F_C + F_D \\ &= -\lambda\,(\alpha_A + \alpha_B + \alpha_C + \alpha_D)\,R = 0 \end{aligned} \tag{12}$$

What about the rotational momentum contribution? Using (7) and (4), it can be computed as follows:

$$\begin{aligned} &P_A \wedge F_A + P_B \wedge F_B + P_C \wedge F_C + P_D \wedge F_D \\ &= -\lambda\,(\alpha_A\,P_A + \alpha_B\,P_B + \alpha_C\,P_C + \alpha_D\,P_D) \wedge R \\ &= -\lambda\,R \wedge R = 0 \end{aligned} \tag{13}$$

This is indeed an essential feature of our model: Only (4) and (7) are required for having exactly total mechanical momentum conservation, both translational and rotational, *whatever the current position of the particles* and whatever the actual way of computing coefficients, provided that (11) is fulfilled. Thus, momentum conservation is not broken by having the coefficients α_A, α_B, α_C, α_D computed *independently* from the current position of the particles.

The simplification of our model is therefore the following: The coefficients α_A, α_B, α_C, α_D can be *precomputed* using the *initial* geometry of the mesh. Then, for each iterations, we only need to compute the linear evaluations (4) and (7), which is quick and straightforward. Indeed, this is also why our bending model is perfectly linear. In the context of cloth simulation, where the 2D fabric coordinates of the mesh vertices are usually available, it is wise to use these for precomputing the coefficients through (6) and (5), as they represent exactly the mesh in its undeformed state.

2.4. Computing the Jacobian

As discussed above, our model is implemented so that the coefficients α_A, α_B, α_C, α_D do not depend on the current position of the vertices $\mathbf{P_A}$, $\mathbf{P_B}$, $\mathbf{P_C}$, $\mathbf{P_D}$. In these conditions, the Jacobian of the bending stiffness forces is simple and straightforward to compute from (4) and (7) without any approximation. This is done as follows, with $\mathbf{I}$ denoting the identity matrix, and with any $\mathbf{J}$ and $\mathbf{K}$ among ($\mathbf{A}$,$\mathbf{B}$,$\mathbf{C}$,$\mathbf{D}$):

$$\frac{\partial F_J}{\partial P_K} = -\lambda\,\alpha_J\,\alpha_K\,I \tag{14}$$

Thanks to a perfectly linear bending model, the Jacobian of all bending forces is constant and totally independent from the current particle positions. This allows efficient numerical resolution through usual implicit numerical resolution methods [BW98] [EEH00] [HE01] with good convergence properties, along with possible algorithmic optimizations for performing the computation quickly.

2.5. Algorithm Summary

Precomputation: For each internal edge of the mesh:

- Evaluate α_A, α_B, α_C, α_D using (6) and (5).
- Evaluate λ from the bending stiffness μ using (10).
- Eventually, pre-evaluate the Jacobian contribution (14).

Computation: For each internal edge of the mesh:

- Evaluate the bending vector $\mathbf{R}$ using (4).
- Evaluate particle forces $\mathbf{F_A}$, $\mathbf{F_B}$, $\mathbf{F_C}$, $\mathbf{F_D}$ using (7).

2.6. Extending the Bending Stiffness Model

So far, we have described a model of isotropic bending stiffness. Anisotropic (direction-dependent) bending stiffness can be modeled by assigning to an edge of the mesh a stiffness which depends on the angle of the edge relatively to the surface coordinates. Hence, given weft and warp bending stiffness μ_U and μ_V, the equivalent bending stiffness μ orthogonally to the edge of angle θ relatively to the weft direction would be:

$$\mu = \mu_U\,\sin^2\theta + \mu_V\,\cos^2\theta \tag{15}$$

Another possible extension could be the consideration of nonlinear bending strain-stress laws. The local curvature of the surface may be approximated from the bending vector $\mathbf{R}$ using (2) and (3) by the following evaluation:

$$\gamma = \frac{2\,R \cdot N}{h_A\,h_B} \quad with \quad N = \frac{N_A + N_B}{|N_A + N_B|} = \frac{N_C + N_D}{|N_C + N_D|} \tag{16}$$

Eventually,this approximation may then be used for giving a nonlinear behavior to the bending strain-stress law, for instance by scaling $\mathbf{R}$ accordingly before applying (7).

Also, a rest curvature γ_0 could be given to the surface by adding an offset $\mathbf{R_0}$ to $\mathbf{R}$ before applying (7), with:

$$R_0 = \tfrac{1}{2}\, h_A\, h_B\, \gamma_0\, N \qquad (17)$$

Similarly, viscous stiffness can easily be added to the model through a bending vector "velocity" **R'** computed from the vertex velocities $\mathbf{P_A'}$, $\mathbf{P_B'}$, $\mathbf{P_C'}$, $\mathbf{P_D'}$ as follows, to be then redistributed similarly as viscous forces over the particles:

$$R' = \alpha_A\, P'_A + \alpha_B\, P'_B + \alpha_C\, P'_C + \alpha_D\, P'_D \qquad (18)$$

These two extensions would however introduce some rotational mechanical momentum in (13), as $\mathbf{R_0}$ and **R'** are not necessarily collinear to **R**. Although often small enough to be ignored in most practical cases, it should eventually be compensated through significant additional computation.

2.7. Side Effects on Tensile Stiffness

Only the normal-based method ensures a perfect decoupling between tensile and bending stiffness. On the other side, the spring-based model involves very large coupling, and is totally inappropriate for models that require good quantitative accuracy on the stiffness behavior of the mechanical surface. Meanwhile, the presented linear method does involve some coupling, which in practice shows up as two effects (Fig.4), as follows.

When the edge bending remains small, the linear method introduces some stiffness that does not only oppose curvature, but also any deformation and orientation difference between adjacent elements. Fortunately, this local effect does not introduce large-scale tensile stiffness in the surface as would the spring-based approach do, but only produces some local "fairing" of tensile deformations, producing no noticeable effects in the global draping.

When the edge bending gets large, the linear bending method introduces some tensile compression effects that might possibly "crunch" mesh elements which have low tensile stiffness compared to the bending stiffness. This effect is however unlikely to show noticeable effects with the tensile-bending stiffness ratio of usual materials. However, this outlines what could be the largest drawback of our model: It is quite inadequate to prevent the occurrence of self-collisions that result from wrinkle collapses and edge flipping through the sole use of large bending stiffness when edge bending may become large.

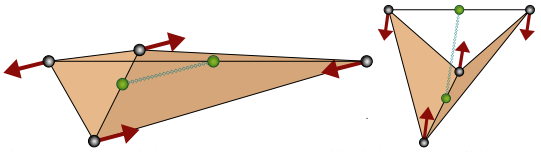

Fig.4: Bending stiffness may oppose tensile deformation difference for low edge bending (left), and produce tensile compression for high edge bending (right).

3. Results

This model has been implemented in a cloth simulation system which uses an accurate particle scheme of nonlinear anisotropic tensile surface elasticity representation, integrated with Implicit Euler [EEH00]. This system is being used for accurate garment simulation (Fig.7). For comparison, an alternate spring-based and normal-based bending stiffness models have also been implemented.

We have carried out a number of tests for assessing the accuracy of the bending stiffness produced by our model. Using the drape test (Fig.6) test, we did not notice any significant difference of our model compared to the normal-based model. We expect some differences to show up as individual mesh edges start to show large bending angles, a situation which is however not visible in the drape test since tensile stiffness in the mesh actually limits the wrinkle details down to polygon size. We have also assessed mesh independence of the bending stiffness behavior through resolution tests (Fig.5). Comparing both models in this context, the vertical fall difference remains well below 1%.

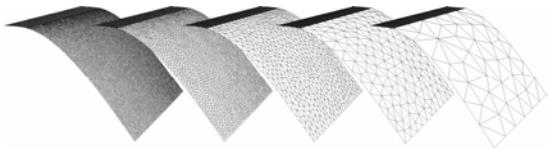

Fig.5: Bending stiffness is accurate enough for being fairly well resolution-independent.

Performance was measured as the time taken for particle bending force evaluation on a 10.000 polygon surface, measured on a 3GHz Pentium4 PC (in milliseconds):

Spring-Based	Normal-Based	Our Model
34 ms	190 ms	42 ms

The computation time of our model is quite comparable to the spring-based model (slightly more computation, but no square roots to evaluate), and way shorter than the normal-based model (normals and coefficients no not need to be evaluated at each iteration). Furthermore, our method also benefits from a very simple and precomputable Jacobian.

It should be also noted that thanks to its linearity, and therefore a constant Jacobian without any approximation, we also observe a significant performance increase when using implicit integration methods, as we can use significantly larger iteration time steps and less Conjugate Gradient iterations in the linear system resolution without excessively increasing the numerical error of the resolution (The amount is much dependent on many other factors of the simulation system). To assess this, we have compared the overall performance on the disk shown in Fig.6 top right (0.01 N.m bending stiffness), using 100 millisecond iteration timesteps. The underlying tensile model is a spring-mass model and the integration method is Implicit Euler [EEH00] (computation speed given in iterations per second, including rendering):

No Bending	Normal-Based	Our Model
14 itr / s	5 itr / s	11 itr / s

It can be noted that we were not able to perform correctly this simulation, which involves quite high bending stiffness, with the spring-based model (very inaccurate drape and numerical problems due to the very high spring stiffness).

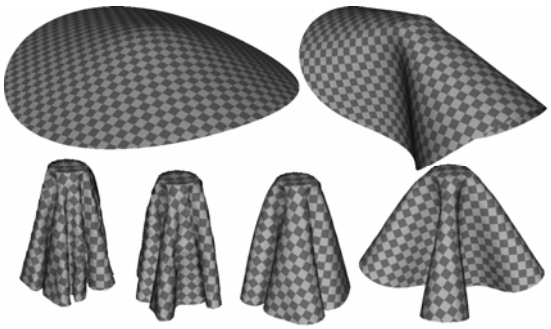

Fig.6: Drape test with various bending stiffness: 0.1, 0.01, 0.001, 0.0001, 0.00001, 0.000001 N.m. Tensile stiffness 100 N/m, density 0.1 kg/m^2, radius 1 m, approximately 4000 polygons.

4. Conclusion

In terms of computational simplicity and speed, our model competes very well with the simple crossover spring model, by requiring only simple linear operations for each iteration. Furthermore, no square roots or trigonometry are required, and the resulting Jacobian is constant and can be evaluated without approximation. This is indeed a huge computational advantage when using implicit numerical integration methods. The accuracy of our model is furthermore quite close to the accurate normal-based method, without requiring most of its complex computation.

In summary, we cannot identify any context where the use of the spring-based method would be more advisable comparatively to ours: Our method offers much better accuracy (particularly when dealing with low curvature and high bending stiffness) with similar or even significantly better computation time when considering implicit integration methods. Meanwhile, the only situations where we still advocate the use of the normal-based method are when mesh deformation constraints are large enough for producing large bending around mesh edges despite high bending stiffness, when this bending has to be explicitly considered for highly accurate nonlinear models, and when such models are used for preventing surface tangling using collision processing techniques such as those described in [BFA02].

Still, the proposed model is a very good compromise for simulating accurately bending stiffness in many different contexts, and its simplicity and linearity opens the door to efficient implementations based on vector computation and dedicated hardware.

Acknowledgements

We are thankful to all MIRALab team participants to this work through their technical and artistic contributions. This research is funded by the European FP6 project LEAPFROG IP (NMP2-CT-2005-515810).

Bibliography

[BFA02] R. Bridson., R. Fedkiw, J. Anderson, Robust treatment of collisions, contact, and friction for cloth animation, *ACM Transactions on Graphics* (ACM. SIGGRAPH 2002 proceedings), pp 594–603, 2002.

[BHW94] D.E. Breen, D.H. House, M.J. Wozny, Predicting the Drap of Woven Cloth Using Interacting Particles, *Computer Graphics* (ACM SIGGRAPH 94 proceedings), Addison-Wesley, pp 365-372, 1994.

[BMF03] R. Bridson, S. Marino, R. Fedkiw, Simulation of Clothing with Folds and Wrinkles, *Eurographics-SIGGRAPH Symposium on Computer Animation*, pp 28-36, 2003.

[BW98] D. Baraff, A. Witkin, Large Steps in Cloth Simulation, *Computer Graphics* (ACM SIGGRAPH 98 proceedings), ACM Press, 32, pp 106-117, 1998.

[EEH00] B. Eberhardt, O. Etzmuss, M. Hauth, Implicit-Explicit Schemes for Fast Animation with Particles Systems, *Proceedings of the Eurographics workshop on Computer Animation and Simulation*, Springer-Verlag, pp 137-151, 2000.

[EWS96] B. Eberhardt, A. Weber, W. Strasser, A Fast, Flexible, Particle-System Model for Cloth Draping, *Computer Graphics in Textiles and Apparel* (IEEE Computer Graphics and Applications), IEEE Press, 16(5), pp 52-59, Sept. 1996.

[GHDS03] E. Grinspun, A. Hirani, M. Desbrun, P. Schröder, Discrete Shells, *ACM Symposium on Computer Animation*, 2003.

[HE01] M. Hauth, O. Etzmuss, A High Performance Solver for the Animation of Deformable Objects using Advanced Numerical Methods, *Eurographics 2001 proceedings*, Blackwell, 2001.

[MDSB03] M. Meyer, M. Desbrun, P. Schröder, A.H. Barr, Discrete differential-geometry operators for triangulated 2-manifolds, *Visualization and Mathematics III*, Springer-Verlag, pp 35–57, 2003.

[Pro95] X. Provot, Deformation Constraints in a Mass-Spring Model to Describe Rigid Cloth Behavior, *Proceedings of Graphics Interface*, pp 147-154, 1995.

[TPBF87] D. Terzopoulos, J. Platt, A. Barr, K. Fleischer, Elastically Deformable Models, *Computer Graphics* (ACM SIGGRAPH 87 proceedings), 21(4), pp 205-214, 1987.

[TW06] B. Thomaszewski, M. Wacker, Bending Models for Thin Flexible Objects, WSCG Short Communication Proceedings, 9(1), 2006.

[VCT95] P. Volino, M. Courchesne, N. Magnenat-Thalmann, Versatile and Efficient Techniques for Simulating Cloth and Other Deformable Objects, *Computer Graphics* (ACM SIGGRAPH 95 proceedings), pp 137-154, 1995.

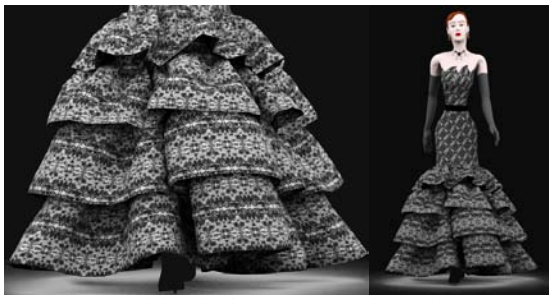

Fig.7: A fast model to simulate very efficiently complete garments with materials that may have fairly large bending stiffness.