

Implicit-Explicit Schemes for Fast Animation with Particle Systems

B. Eberhardt O. Eitzmuß

M. Hauth

Wilhelm-Schickard-Institut

Universität Tübingen, Germany

E-mail: {beberh, etzmuss, mhauth}@gris.uni-tuebingen.de

Abstract

Particle systems have been widely employed to animate deformable objects. In order to achieve real time capable systems often simplifications have been made to reduce the computational costs for solving the ODE at the expense of numerical and physical correctness. Implicit-Explicit (IMEX) methods provide a way to solve partly stiff systems efficiently, if the system meets some requirements. These methods allow the solution of the differential equation for particle systems to be computed both correctly and very quickly. Here we use an IMEX method to simulate draping textiles. In particular, our approach does not require any post-correction and works for very stiff materials.

1 Introduction

Since the mid of the 80's the animation of deformable models has attracted more and more attention in computer graphics. Terzopoulos and Fleischer [11] proposed a continuous model for deformable objects and solved the resulting PDEs with finite differences. Later on, particle systems became a very common model, in particular for two-dimensional deformable objects like textiles [3, 6, 9, 2]. All approaches require an ordinary differential equation to be solved.

Although Terzopoulos and Fleischer used a semi-implicit numerical method, i.e. explicit in space and implicit in time, later on explicit methods became popular. These require considerably less work per step, because there are no systems of equations to be solved.

Unfortunately explicit methods allow only small time steps for stiff particle systems. With the work of Baraff and Witkin [2], implicit time stepping underwent a renaissance in animation, because these implicit methods allow large time steps without loss of stability. Traditionally in computer graphics realistic behaviour is more

important than short computation time, whereas in virtual reality applications interactive frame rates are necessary. But since the presentation of this work, the barrier has been torn down and no one will accept hours of computation for the animation of deformable objects. Recent work [4, 8] focused on reducing computational costs per step while preserving the stability properties of the implicit algorithm.

The major drawback of implicit methods is, that each step requires a – generally nonlinear – system to be solved. Necessarily [7] at each step a Newton-like method is used to solve this system. To achieve interactive frame rates in computer animation several approaches have been presented to circumvent this problem. Some simplifications were introduced to allow very fast performance of an implicit step. But as a consequence of these simplifications the solution is not any more the solution of the ODE originally posed. Even if the results are visually pleasing, it is doubtful that specific materials can be modelled correctly in this way.

In this paper we propose the use of IMEX methods to solve the arising differential equation both fast and correctly. IMEX methods work on so called split ODEs and consist of two methods, an implicit and an explicit one, the first one being applied to stiff parts of the ODE the second one to the nonstiff parts. Such methods have been used in numerical analysis, for instance to solve convection-diffusion problems [1]. The main idea is to split the right-hand-side function f of the ODE into a – preferably linear – stiff part and a nonlinear nonstiff part. This way only a linear system has to be solved at each time step.

We also compare our method to the approaches taken by other authors.

2 Numerical methods

All above approaches require an initial value problem of an ordinary differential equation of the abstract form

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

to be solved. We will specialise the system and exploit its structure below.

2.1 Multistep methods

As the discussed IMEX methods are derived from multistep methods we will describe these briefly. Such a method with k steps is of the form

$$\sum_{j=0}^k \alpha_j Y_{n+j} = h \sum_{j=0}^k \beta_j f_{n+j}, \quad (1)$$

$$f_{n+j} := f(t_{n+j}, Y_{n+j}),$$

where h is the (time-) step size and Y_i is the numerical solution at time t_i . The coefficient α_k is required to be nonzero. Important special cases are the class of Adams

methods where $\alpha_0 = \dots = \alpha_{k-2} = 0$:

$$Y_{n+k} = Y_{n+k-1} + h \sum_{j=0}^k \beta_j f_{n+j} \quad (2)$$

and the class of BDF-methods (**backward differentiation formulas**) with $\beta_0 = \dots = \beta_{k-1} = 0$:

$$\sum_{j=0}^k \alpha_j Y_{n+j} = h \beta_k f_{n+k}. \quad (3)$$

A multi-step method is called implicit, if $\beta_k \neq 0$. An implicit method requires the solution of a nonlinear system of equations, as Y_{n+k} is needed to evaluate f for f_{n+k} . These equations have to be solved by Newton's method or a Newton-like scheme. The use of fixed point iteration is not suitable for stiff ODEs [7].

This nonlinear system is given by

$$0 = Y_{n+k} - h \beta_{n+k} f_{n+k} + a_n, \quad (4)$$

where $a_n = \sum_{j=0}^{k-1} (\alpha_{n+j} Y_{n+j} - h \beta_{n+j} f_{n+j})$. This system has to be solved for Y_{n+k} .

Newton's method applied to (4) leads to systems of linear equations of the form

$$(I - h \beta_{n+k} J) \Delta Y^{(i)} = b^{(i)}, \quad (5)$$

where $J = \frac{\partial}{\partial y} f(t_{n+k}, y(t_{n+k}))$, $Y_{n+k}^{(i)} = Y_{n+k}^{(i-1)} + \Delta Y^{(i)}$ and $b^{(i)} = -Y_{n+k}^{(i-1)} + h \beta_{n+k} f_{n+k} + a_n$.

The well known forward and backward Euler schemes fit in the multistep framework. The forward Euler method turns out to be the explicit Adams method for $k = 1$ and $\beta_1 = 0$, the backward Euler method is the BDF-method for $k = 1$, $\alpha_0 = -1$, $\alpha_1 = 1$, and $\beta_1 = 1$.

These methods were the first ones especially developed to deal with stiff equations.

2.2 IMEX-Schemes

In order to reduce the computational work, we look at a more specialised setting with a splittable right hand side (split ODE)

$$y'(t) = f(t, y(t)) + g(t, y(t)), \quad (6)$$

where g incorporates the stiff part of the system and f the nonstiff remainder. As recently discussed in numerical analysis literature [1], it is possible to combine two schemes for solving (6). This can be done such that the stability properties of an implicit method for g are nearly preserved while gaining the ease of the computation of an explicit method for f . A thorough analysis for combining multistep methods can be

found in [1]. For a constant step size h we thus obtain the general formula (compare to (1))

$$\begin{aligned} Y_{n+k} + \sum_{j=0}^{k-1} \alpha_j Y_{n+j} \\ = h \sum_{j=0}^{k-1} \beta_j f_{n+j} + \sum_{j=0}^k \widehat{\beta}_j g_{n+j}, \end{aligned} \quad (7)$$

where the β_j and $\widehat{\beta}_j$ are the coefficients of the explicit and the implicit method, respectively.

Therefore as above Newton's method is used to solve

$$G(Y_{n+k}) = Y_{n+k} - h\widehat{\beta}_{n+k}g_{n+k} + \widehat{a}_n, \quad (8)$$

with \widehat{a}_n defined analogously to (4). As a linear system, we now get

$$(I - h\widehat{\beta}\widehat{J})\Delta Y^{(i)} = \widehat{b}^{(i)},$$

where $\widehat{J} = \frac{\partial}{\partial y}g(t_{n+k}, y(t_{n+k}))$ is the Jacobian of g and contains all stiff eigenvalues of the system. By this approach we may save a lot of work because we don't have to compute the whole Jacobian.

A further improvement can be achieved if g is a linear operator with respect to y , i.e. with $A(t) := \frac{\partial}{\partial y}g(t)$

$$g(t, y) = A(t)y.$$

In this case Newton's method reduces to the solution of a single linear system

$$(I - h\widehat{\beta}A(t))Y_{n+k} = \widehat{b}. \quad (9)$$

Since it is possible to formulate g with a symmetric Jacobian, conjugate gradient method (*cg*) should be used, whereas stationary methods like the Jacobi-method are not suitable because of the large (i.e. stiff) eigenvalues of $A(t)$. *cg* has the advantage of a fast convergence in the direction of the major eigenvalues, which represent the stiff components.

If it is not possible to isolate the stiff components in a linear operator, one can use an inexact simplified Newton method for the solution of (8) (or (4) respectively).

2.3 Exploiting the second order structure

The system under discussion has the special structure of being defined by an ODE of second order, i.e.

$$x'' = \widetilde{f}(t, x, x').$$

By the auxiliary variable

$$v := x', \quad (10)$$

one gets an equivalent system of first order and doubled dimension

$$\begin{bmatrix} x \\ v \end{bmatrix}' = \begin{bmatrix} v \\ \tilde{f}(t, x, v) \end{bmatrix}.$$

Thus the Jacobian has the special form

$$\frac{\partial}{\partial [x, v]^T} f(t, [x, v]^T) = \begin{bmatrix} 0 & I \\ A_x & A_v \end{bmatrix}.$$

with $A_x = \frac{\partial}{\partial x} \tilde{f}(t, x, v)$ and $A_v = \frac{\partial}{\partial v} \tilde{f}(t, x, v)$. We can exploit this by analogously splitting $Y_{n+k}^{(i)}$ and $b^{(i)}$ from (9) (or (5) respectively) in their \cdot_x and \cdot_v parts, thus solving (we omit the Newton index)

$$\begin{bmatrix} I & -h\beta I \\ -h\beta A_x & I - h\beta A_v \end{bmatrix} \begin{bmatrix} \Delta Y_{n+k, x} \\ \Delta Y_{n+k, v} \end{bmatrix} = \begin{bmatrix} b_x \\ b_v \end{bmatrix}$$

in each iteration. By taking advantage of the upper half, i.e. $\Delta Y_{n+k, x} = b_x + h\beta \Delta Y_{n+k, v}$ we are left with the computation of a solution for

$$\begin{aligned} (I - (h\beta)^2 A_x - h\beta A_v) \Delta Y_{n+k, v} \\ = b_v - h\beta A_x b_x. \end{aligned}$$

The dimension of this system is reduced to the original dimension of the second order ODE. Thus we save a considerable amount of computing time.

3 Comparison of methods

All these ideas, although not presented in the context of IMEX schemes, are implicitly present in previous work.

Baraff and Witkin [2] formulate nonlinear constraints but only use their linear approximation to obtain a linear system of equations. This way the system to be solved in an implicit Euler step also becomes linear and can be solved efficiently by a *cg*-method. This method corresponds to the solution of a nonlinear system with only one Newton iteration. Because the nonlinear part is not integrated, with high stiffness one may encounter similar problems as we had to deal with.

Provot [9] proposed a simple model only incorporating linear springs. This model was used by Desbrun et al. [4] who used also the implicit Euler method. But instead of linearising the whole system they split it in a linear and nonlinear part and use a precomputed inverse of the system matrix for solving the linear part of the equations.

They don't aim at solving the equation completely, as they don't integrate the nonlinear term. Instead a correcting force is introduced to preserve the angular momentum approximately. The use of a precomputed inverse prohibits a change of step size h and changes of the elastic moduli.

Based on this work Kang et al.[8] did some further simplification to avoid solving the linear system. In order to update the solution vector in one step they divide each row by its diagonal entry of the matrix of the linear system. Therefore they just perform a single iteration of a Jacobi-like scheme for solving the linear equation (5). This may be too little to solve this equation or to damp the stiffer modes, especially considering the fact, that the convergence behaviour strongly depends on special properties of the system matrix. $I - h\tilde{\beta}A$ is in fact diagonal dominant, actually by the '1' of the identity. But the stiffer the equation the less this one counts for, and convergence may be hardly conceivable.

In [9], [4],[8] a post-correction step is used to restrict the spring elongations to a maximum value. Thus they can use only moderately stiff springs and obtain a cloth-like behaviour after performing this post-correction. Unfortunately physical soundness is not guaranteed by this modification of the numerical solution.

4 Modelling a mass-spring system

We note that the stiffness in particle systems that model textiles is due to the springs that account for tension forces (Provot [9] calls these springs structural springs). Therefore it is sufficient to solve only for these springs implicitly and use the explicit scheme for the remaining forces.

Forces for linear springs between two particles at x_i and x_j are given by

$$F(x) = k_{ij}(\|x_i - x_j\| - l_{ij}) \frac{x_i - x_j}{\|x_i - x_j\|}, \quad (11)$$

where k_{ij} is the elastic modulus of this spring and l_{ij} its rest length.

Desbrun et al. [4] split these forces into a linear part

$$F_1(x) = k_{ij}(x_i - x_j) \quad (12)$$

and a non-linear part

$$F_2(x) = -k_{ij} \cdot l_{ij} \frac{x_i - x_j}{\|x_i - x_j\|}. \quad (13)$$

In our approach F_1 is included in g , whereas F_2 is treated by the explicit method as part of f . This is possible because F_2 has a constant absolute value and its Jacobian does not contribute significantly. The solution of the IMEX-scheme remains stable under all circumstances and for all feasible parameters, although F_2 is treated explicitly.

Additionally, we need damping forces to account for energy dissipation. These are modelled for each structural spring by

$$F_d(x) = d_{ij}(v_i - v_j), \quad (14)$$

where d_{ij} is the damping coefficient. These dissipative forces are stiff as well as linear, hence they are incorporated in g .

Thus we get linear forces

$$k_{ij}(x_i - x_j) + d_{ij}(v_i - v_j) \quad (15)$$

for each spring acting on particle i .

Defining matrices

$$(K)_{ij} = \begin{cases} \sum_{i \neq j} k_{ij} & \text{if } i = j, \\ -k_{ij} & \text{if } i \neq j \end{cases} \quad (16)$$

and, analogously,

$$(D)_{ij} = \begin{cases} \sum_{i \neq j} d_{ij} & \text{if } i = j, \\ -d_{ij} & \text{if } i \neq j \end{cases} \quad (17)$$

the linear forces can be written by

$$F_{\text{lin}}(x, v) = Kx + Dv \quad (18)$$

These matrices represent the discretized Laplacian operator, which models the diffusion in a deformable model. They were already used by Desbrun et al. in [4].

The explicit forces contained in f can be arbitrary non-stiff forces. Besides F_2 , there can be forces due to bending and shearing depending on angles (e.g.[6]), which are typically several magnitudes smaller compared to tension forces and therefore can be integrated explicitly. Our approach has the major advantage, that the particle system is not anymore restricted to springs described by (11) as in Provot's model [9], but also allows triangle meshes to be animated.

External forces due to air resistance, wind and gravitation are treated explicitly as well.

5 Setting up a linear system

Here we describe how an IMEX first order method is applied to the described particle system. It uses the implicit and explicit form of Euler's methods:

$$y^{l+1} = y^l + hf(y^l) + hg(y^{l+1}) \quad (19)$$

First, we follow section 2.3 and reduce the system from $6n$ coordinates to $3n$ coordinates by applying the integration scheme to the lower row of equation (10). The implicit Euler method for $\frac{dx}{dt} = v$ gives

$$x^{l+1} = x^l + hv^{l+1}. \quad (20)$$

Applying (19) to our system and substituting for x^{l+1}

$$\begin{aligned}
Iv^{l+1} - h\frac{1}{m}(Kx^{l+1} + Dv^{l+1}) &= \\
v^l + h\frac{1}{m}F_{\text{expl}}(x^l, v^l) & \\
\Leftrightarrow & \\
(I - h^2\frac{1}{m}K - h\frac{1}{m}D)v^{l+1} &= \\
v^l + h\frac{1}{m}F_{\text{expl}}(x^l, v^l) - h\frac{1}{m}Kx^l & \quad (21)
\end{aligned}$$

This is a symmetric linear system that can be solved efficiently by the *cg* method. One *cg* iteration has complexity of a sparse matrix-vector multiplication, which is linear in the number of nonzero entries in the sparse matrix. With sparse matrices in general this is even less expensive than multiplying with an inverse which can have $O(n^2)$ nonzero elements, due to the loss of sparsity on inversion.

From (21) it can be seen that the stiffness of the system does not depend on k but on $\frac{k}{m}$. Note that this corresponds to the angular frequency of a harmonic oscillator that is given by $\sqrt{\frac{k}{m}}$. Therefore a smaller mass density of the material leads to stiffer differential equations. Furthermore, a finer discretization leads to a stiffer ODE as well, because a single particle of a finer mesh has a smaller mass.

The matrix of the system changes whenever the elastic moduli or damping coefficients change. As forces in textiles are not linear they must be approximated by piecewise linear springs (see [6, 5]). This is required for realistic textile modelling. Hence, in general, the matrix must be set up in each step.

The number of *cg* iterations depends on the condition number of the matrix and the starting value for the iteration. The condition number of the matrix is improved by an incomplete cholesky preconditioner. As an effect, our iteration converges in very few steps.

A starting value is computed by a predictor that extrapolates the new solution vector from the most recent values of the solution. Considering the single-step integration methods used we only use the last value and compute the predictor as follows:

$$v_{n+1} \approx 2v_n - v_{n-1} \quad (22)$$

6 Constraints and collision response

In every numerical setup for animation there must be a way to enforce constraints, in particular constraints imposed by collisions.

Baraff and Witkin [2] presented a very efficient method to enforce constraints in a *cg*-method. In each iteration of the *cg*-method the new direction is filtered such that

the solution does not alter in a constrained direction. Hence if necessary we can preset the velocity of a particle for the start value of the *cg*-method, because by constraining the velocity in all directions, it is guaranteed that this velocity is unchanged in the final solution returned by the *cg*-method.

However, we found that the repositioning suggested in [2] produced a severe increase in the number of *cg*-iterations in our system. Therefore, we dispense with any alteration of particle positions, because we can control the position of each particle at the next time using the described constraint enforcing mechanism.

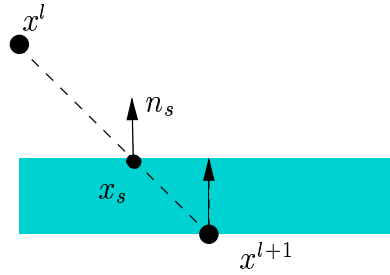


Figure 1: Correcting a particle position

If we want the particle to move back to the surface we constrain its velocity in the normal direction n_s of the penetrated surface

$$\pi(v^{l+1}) = \frac{1}{h}\pi(x_s - x^{l+1}),$$

where π is the projection onto the oriented direction n_s and x_s is the point where the particle penetrates the surface (see figure 1).

Since we constrain only one direction of the particle velocity the particle is still free to move in the other directions according to the forces acting on it. Therefore the velocity is only constrained in the direction normal to the penetrated surface such that the particle is driven back to the surface but can move freely on the surface.

This way we allow the textile to penetrate temporarily. In the visualisation of a frame auxiliary particles corresponding to penetrated particles are displayed such that no penetration is visible. This way any kind of collision, elastic or inelastic, can be modelled.

7 Dealing with very high stiffness

If we model textiles with extremely large elastic moduli like those that are obtained from the Kawabata experiments [3], we note that the solution computed remains stable

but the movement of the textile is reduced when large time steps are used and the animation almost comes to a standstill (unstable methods show a chaotic behaviour).

The reason for this happening is that the matrix of the linear system (21) does not have any entries that couple the different coordinate directions, i.e. we could also solve three independent systems. Changes in one direction cannot be compensated in another direction in the same integration step, because all coupling terms are contained in f . They appear only in the single computation of f on the right-hand side of the linear system (21).

In order to overcome this problem these coupling terms have to be updated within the linear solving procedure, i.e. the right-hand side of the system must be updated in the cg -method. Hence we start the cg -method with the initial b , do a few cg iterations, update b with the current approximative solution, do a few cg iterations and so on.

This updated iteration converges as well as the conventional cg -method. The algorithm to solve $Ax = b(v, x)$ with error tolerance ϵ , preconditioner P and constraint operator C , which enforces the constraints as described in the previous section, is given by the following pseudo-code:

Algorithm 1: Updated and filtered cg

```
do
   $i = 0$ 
   $b = b(v, x^l + hv)$ 
   $r = C(b - Av)$ 
  while  $\|r\| \geq \epsilon \|b\|$ 
     $i = i + 1$ 
    solve  $P^{-1}z = r$ 
     $z = Cz$ 
    if  $i = 0$ 
       $\rho = \langle r, z \rangle$ 
    else
       $\beta = \frac{\rho}{\rho_1}$ 
       $p = z + \beta p$ 
     $q = Ap$ 
     $q = Cq$ 
     $\alpha = \frac{\rho}{\langle p, q \rangle}$ 
     $v = v + \alpha p$ 
     $r = r - \alpha q$ 
     $\rho_1 = \rho$ 
while ( $i > 0$ )
```

Only very few cg iterations are needed within one time step and the solver is still very efficient. For very stiff springs the simplified Newton method needs less cg iterations than plain cg would require for the same simulation.

We found it sufficient, to use a constant forcing term, i.e. we impose the residual tolerance

$$\|r\| \leq \epsilon \|b\|.$$

on the cg -iteration. Further improvements can be made by optimising this forcing term, which will be part of forthcoming work.

8 Results

We implemented our numerical algorithms with the Matrix Template Library (MTL [10]) by the University of Notre Dame. It provides efficient and flexible data structures for (sparse) matrix and vector operations, which are crucial for the simulation performance.

To evaluate the performance of our techniques we present two examples simulated with various parameters.

The first example is a tablecloth draping over a square table (figure 4 (a)-(c)). This example demonstrates the collision response described in section 6. The second one is a textile that is fixed at two corner points (figure 4 (d)).

#Particles	$\frac{k}{m}$	$\frac{d}{m}$	h	#cg	cg	matrix setup	solver
400	10^4	40	0.01s	192	1.14s	1.31s	3.34s
400	10^4	40	0.02s	233	1.04s	0.7s	2.16s
400	10^6	40	0.005s	2601	8.07s	2.67 s	12.47s
2704	10^6	40	0.01s	1344	32.29s	10.46s	49.64s

Figure 2: Square table, performance for simulation of one sec. on a Mips RS10000/250MHz

#Particles	$\frac{k}{m}$	$\frac{d}{m}$	h	#cg	cg	matrix setup	solver
400	10^4	40	0.01s	58	0.54s	1.3s	2.7s
400	10^6	40	0.001s	614	5.56s	12.66s	27.02s

Figure 3: hanging textile, performance for simulation of one sec. on a Mips RS10000/250MHz

The successive entries in table 2 and 3 are the number of particles used in the example, the elastic modulus/mass quotient, the damping coefficient/mass quotient of the structural springs, the time step h and the number of cg iterations. Further we measured computation times for the modified cg method, for the matrix setup and the overall times of the numerical solver (including matrix setup, cg and the computation of the new solution).

Although the number of cg iterations increases with the stiffness, we still get a good performance for stiff springs. Note that a smaller time step does not necessarily lead to an increase in the number of cg iterations.

9 Conclusion and further work

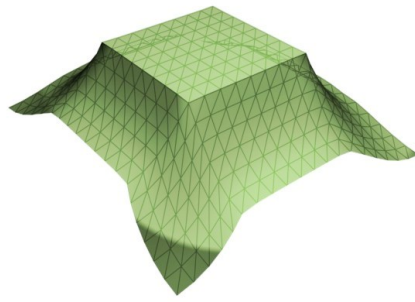
We have discussed a system that efficiently simulates the draping behaviour of textiles. With our approach, the arising differential equation is completely solved and nonlinear forces are treated correctly. This is especially important, as in the context of very high stiffness neglecting these forces would lead to wrong results.

Due to our flexible design the system is suited for virtual reality applications as well as for high accuracy simulation of cloth. As the results show, the approach, leaving the elastic moduli (and thus the matrix) constant, is capable of simulating several hundred particles in real time. If we use a more accurate model with high and varying stiffness to model the nonlinear behaviour of cloth, the computation time does not exceed several minutes and is still less expensive than a high quality rendering of that scene.

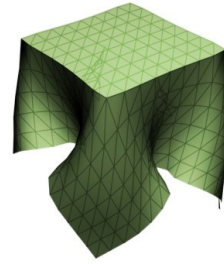
The presented model and the numerical treatment do not depend on the regular structure of the particle mesh. Rectangular as well as triangular meshes, which provide

additional freedom for modelling more complex shapes of textiles, may be used.

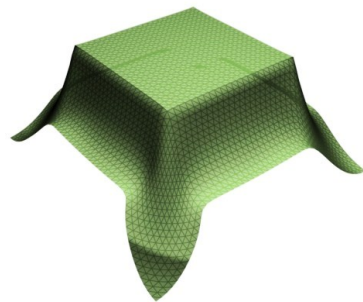
Further work will focus on algorithmic and numerical improvements as variable time-stepping and Jacobi-saving strategies will result in another speedup. Choosing an optimal preconditioner and a variable forcing term may prove valuable as well.



(a)



(b)



(c)



(d)

Figure 4: Examples

10 Acknowledgements

This work was partly supported by the DFG MoViTex and ElastoMedTrain grants.

We are grateful to Christian Lubich for many fruitful discussions. We also would like to thank the developers of MTL for making their library available.

References

- [1] U. M. Ascher, S. J. Ruuth, and B. T. Wetton. Implicit-explicit methods for time-dependent partial differential equations. *SIAM J. Numer. Anal.*, 32(3):797–823, 1995.
- [2] D. Baraff and A. Witkin. Large steps in cloth simulation. In M. Cohen, editor, *SIGGRAPH 98 Conference Proceedings*, Annual Conference Series, pages 43–54. ACM SIGGRAPH, Addison Wesley, July 1998. ISBN 0-89791-999-8.
- [3] D. E. Breen, D. H. House, and M. J. Wozny. Predicting the drape of woven cloth using interacting particles. In A. Glassner, editor, *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, Computer Graphics Proceedings, Annual Conference Series, pages 365–372. ACM SIGGRAPH, ACM Press, July 1994. ISBN 0-89791-667-0.
- [4] M. Desbrun, P. Schröder, and A. Barr. Interactive animation of structured deformable objects. In *Graphics Interface*, pages 1–8, June 1999.
- [5] B. Eberhardt and A. Weber. A particle system approach to knitted textiles. *Computers & Graphics*, 23(4):599–606, 1999.
- [6] B. Eberhardt, A. Weber, and W. Strasser. A fast, flexible, particle-system model for cloth draping. *IEEE Computer Graphics and Applications*, 16(5):52–60, Sept. 1996.
- [7] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II*. Springer-Verlag, Berlin, 1996.
- [8] Y.-M. Kang, J.-H. Choi, H.-G. Cho, D.-H. Lee, and C.-J. Park. Real-time animation technique for flexible and thin objects. In *WSCG*, pages 322–329, Feb. 2000.
- [9] X. Provot. Deformation constraints in a mass-spring model to describe rigid cloth behavior. In W. A. Davis and P. Prusinkiewicz, editors, *Graphics Interface '95*, pages 147–154. Canadian Information Processing Society, Canadian Human-Computer Communications Society, May 1995. ISBN 0-9695338-4-5.
- [10] J. G. Siek and A. Lumsdaine. The matrix template library: Generic components for high-performance scientific computing. *Computing in Science and Engeneering*, pages 70–78, Nov. 1999. Aavailable via ftp from <http://www.lsc.nd.edu/research/mtl>.

- [11] D. Terzopoulos and K. Fleischer. Deformable models. *The Visual Computer*, 4:306–331, 1988.