Comparing Efficiency of Integration Methods for Cloth Simulation

Pascal VOLINO, Nadia MAGNENAT-THALMANN
MIRALab, C.U.I., University of Geneva - CH-1211, Switzerland
Web: http://miralabwww.unige.ch
Email: [pascal|thalmann]@cui.unige.ch
Phone: -41 (22) 705 77 63 Fax: -41 (22) 705 77 80

Abstract

Any cloth simulation system needs efficient numerical methods for integrating the equations that describe the mechanical behavior of the discrete representation of the cloth. Choosing the adequate method should be done with full knowledge of the advantages and weaknesses of the main techniques. This paper presents a quantitative comparison of the efficiency of the most common integration techniques used for cloth simulation, and raises the key considerations for optimal implementations depending on the practical kind of simulation problematic.

Keywords: *Cloth simulation, numerical integration, implicit integration, Euler, Midpoint, Runge-Kutta.*

1. Introduction

The correct choice of the simulation method and its implementation is a very important issue in the design of an efficient cloth simulation system. Among the available methods, there are finite elements methods [EIS 96], continuum mechanics [TER 87] or particle systems [BRE 94]. We will focus on the latter, which has shown to bring the best compromise between accuracy and speed for highly deformable objects such as cloth [VOL 95] [VOL 97].

A particle system represents the mechanical system as a set of punctual masses. The cloth surface shape is represented by the geometry between neighboring particles. The mechanical behavior is represented as interaction forces between the particles, which depend on the relative position and speed of the particles, measuring deformation and deformation speed. Various models exist for this representation, which rank from the simple spring-mass representation (spring forces between particle couples depending on the distance between the particles) to accurate surface or volume models (involving complex interactions between several neighboring particles). The laws ruling these interactions also rank from linear to highly nonlinear involving discontinuities and hysteretic curves.

The evolution of the system is computed numerically from these equations that form a large and sparse ordinary differential equation system, which, through adequate modeling, is also first-order. This numerical system has to be integrated numerically, for finally obtaining the evolution of the mechanical system along time, usually as a sequence of successive positions of the object along regular time intervals. Various numerical methods related to integration of numerical ordinary first-order differential systems are available for this purpose [PRE 92].

The aim of this study is not to describe the implementation of these methods, which has already been carried out extensively in [EBE 96] [VOL 97] [BAR 98] [VOL 00], and with some adaptations in [DES 99] [EBE 00] [KAN 00]. It rather intends to evaluate quantitatively the performance of the main integration methods in terms of speed and accuracy. Using a "typical" cloth object made of a common fabric material, we compare the computation speed and accuracy of each integration methods depending several simulation contexts, giving the reader an overview of the performance he can expect from each method.

2. Integration Methods

The choice of the adequate integration method has to be carried out using various considerations related to the kind of problem to be simulated. Among those considerations, there are:

* The size of the problem, mostly related to the number of particles used to describe the mechanical system.
* The desired accuracy, which reflects the allowable numerical tolerance between the computed solution and the theoretical evolution expected from the mechanical model.
* The simulation context, which can either be an extensive computation of the motion along time requiring accurate evaluation of all the dynamical factors, or a simple relaxation process where the simulation has to converge to the static rest state as quickly as possible.
* The stiffness of the problem, mainly related to the rigidity of the particle interactions and the size of the chosen time step, which translated into the "difficulty" the numerical method has to compute the evolution numerically, and which practically causes inaccuracy and instability problems.
* The time an iteration takes to compute, and the number of mechanical derivations (computation of particle forces from their position and speed) the methods requires to compute one.

The literature is abundant about various integration methods which aim to solve linear systems of first-order ordinary differential equations [PRE 92]. One can easily turn the second-order systems relating dynamical mechanical systems into first-order systems by constructing a state vector defined by the concatenation of position and speed states of the system, such as to fit the

requirements of any of these algorithms. Among all the available methods, we can identify several main classes:

* **Explicit methods**, which compute the state of the next time step out of a direct extrapolation of the previous states using derivative evaluations.
* **Implicit methods**, which deduce the state of the next time step from an equation system expressing some "reversibility" from the extrapolated solution.
* **Low-order methods**, which use a reduced number of evaluations for computing simple low-order extrapolations, leading to quickly computed, but inaccurate iterations.
* **High-order methods**, which use several evaluations to compute high-order solutions that get much more accurate as the time step is reduced.

2.1. Scope of the Study

We shall restrict our consideration to three different methods which explore the range of these classes, and which seem to fit the best the requirements set for cloth simulation problems, in terms of implementation simplicity and efficiency for particle systems using large numbers of particles that interact sparsely and with a constant topology. The methods that we consider for this application are:

* The **explicit Midpoint method**, which is a simple low-order explicit method. It requires two mechanical derivations per iteration and returns a second-order accurate solution relative to the time step. It also requires two storages of the state vector. We preferred this method to the still simpler first-order Euler method, because of the obvious gains of accuracy and stability which, despite the additional mechanical evaluation, makes it largely more efficient. We implemented this method for garment simulation in [VOL 95].
* The **explicit Runge-Kutta method**, implemented in its fifth-order version with error evaluation [PRE 92]. It requires five mechanical derivations per iteration, as well as five storages of the state vector. This method is supposed to provide high accuracy, which increases significantly as the time step is reduced. This method was experimented in [EBE 96] and [VOL 97].
* The **Backward Euler method**, which is the implicit implementation of its simple forward counterpart. It requires one mechanical evaluation and the resolution of a sparse linear system per iteration, as well as one storage of the system state additionally to those required for the system resolution algorithm. This method is supposed to provide approximate results that are not subject to numerical instability as the time step is increased. We implemented this method combined with a Conjugate Gradient algorithm using linear system matrix products computed on the fly, as described in [VOL 00], and thus able to take into account the anisotropy and nonlinearities of the mechanical model as the actual Hessian matric is used for each current state of the mechanical system. No initial matrix setup is required, suppressing also the need of separating linear and nonlinear components as discussed in [EBE 00].

We have also carried out some preliminary tests with the **Rosenbrook method**, which is an implicit implementation of a fourth-order Runge-Kutta method. It is supposed to combine the stability of implicit methods with the accuracy of high-order methods. We implemented this method using the algorithm described in [PRE 92], but preliminary experiments have shown very deceptive results, and the gain of accuracy did not compensate the large calculations required for each iteration, whereas increased instability problems did not allow time steps much larger than those used for good accuracy with backward Euler.

We did not consider in our tests the methods aimed toward simplifications which might highly approximate and degrade the dynamic behavior of deformable models, such as implicit integration with precomputed inverse matrices [DES 99] which involves high simplification and linrarization of the Hessian matrix and which also becomes very unpractical for large matrix sizes (the inverse of a sparse matrix is not necessarily sparse). We simulated such algorithm using accurate resolution on an accordingly approximated constant matrix, and we found that these approximations produced more simulation errors (on dynamic behavior of wrinkles and motion damping particularly) than producing a quick and rough linear system solution using a reduced number of Conjugate Gradient iterations with an accurate matrix. Even more drastic simplifications [KAN 00] reduce the matrices to their diagonal component.

2.2. Implementation

All these methods were implemented in a single framework, which allows the simulation of cloth objects, using two different discrete mechanical representations:

* **A complete surface elasticity model** which allows the simulation of anisotropic elasticity (weft and warp Young modulus, shearing, Poisson coefficient) and associated viscosity. Bending is also implemented, but not taken into account in this study. The base element of this simulation is a triangle of the mesh describing the surface, and the elasticity laws are computed as interactions between the three vertices of a triangle reflecting all the mechanical behavior curves which, for this study, are restricted to be linear.
* **A simplified spring-mass model** which represents an approximated equivalent elasticity model using linear viscoelastic springs connecting the particle couples describing each edge of the surface mesh. This model is one of the simplest that a cloth simulation application would use.

The implementation also supports collision detection and response, which were disabled for these tests. An object-oriented framework written in C++ integrate all these technologies into a single application allowing simulation of cloth objects of any shape with specified parameters.

The application is run on a SGI Octane having a 200 MHz R100000 processor, and enough memory for working without swapping. Performance timings are done on the mechanical computation only, and do not take into account display and data structure management.

3. Performance

Performance is a key issue in choosing the adequate integration method, as cloth simulation usually involves

very large mechanical systems described by a huge number of variables, and the numerical resolution of the system is therefore critical to the total computation time.

Performance actually depends on several factors:

* The **computation time** taken for one iteration of the algorithm. This depends on the complexity of the method, and also related to the number of times the forces of the system have to de derived from the system state using the laws of mechanics.
* The **time step** for one iteration, which represents the time discretization required to reach a given accuracy or numerical stability for a given method.
* The **desired accuracy** of the resolution, which may be coarse if only static equilibrium is to be computed, or high if the whole dynamic evolution of the system is desired. Accuracy increases along with time step reduction as better as the method is high-order.
* The **numerical stability** of the method, which also limits the time step for a given method and a given mechanical system.

These factors describe our investigation field in the following sections.

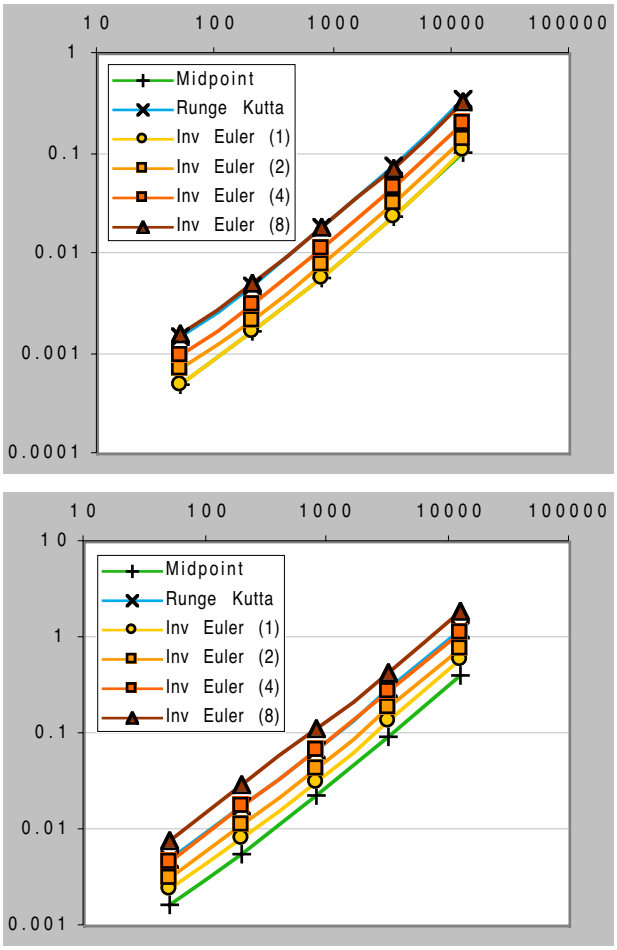

Fig.1. Computation time per iteration for the various integration methods.
on SGI Octane R10000 200MHz
for simplified model (top) and accurate model.(botton)
Time (vertical, seconds) per Polygons (horizontal, number).

3.1. Computation Time

The total computation time is the time required for computing one iteration times the number of iterations. Our first investigation is to evaluate the iteration computation time for each of these methods.

For these measurements, we have simulated a square of fabric with a given discretization both with the accurate and simplified models, using the Midpoint, the Runge-Kutta and the Backward Euler methods, with 1, 2, 4, 8 iterations in the Conjugate Gradient algorithm for the latter, and measured computation time (Fig.1).

From these tests, we can see that with our implementation, the computation times per mesh polygon are roughly the following:

Method	Accurate	Simplified
Midpoint	**32 µs**	**8 µs**
Runge-Kutta	**95 µs**	**25 µs**
Back. Euler (No CG iter.)	**31 µs**	**5 µs**
Back. Euler (Per CG iter.)	**16 µs**	**2.5 µs**

The most important fact to note is that the application of the Backward Euler method with a reduced number of Conjugate Gradient iterations compares very well with the traditional explicit methods. With one iteration only, it is barely worse than the very simple explicit Midpoint method. Our implementation, described in [VOL 00] does not explicitly construct the matrix of the system to be resolved by the Conjugate Gradient, but computes "on the fly" the product of this matrix with vectors when needed by the Conjugate Gradient algorithm. This gives a very efficient implementation when using a low number of Conjugate Gradient iterations (no heavy preprocessing for building the matrix), which is often sufficient for most applications.

These tests will help us to choose the method that gives the best compromise between accuracy and computation speed, as discussed in the next section.

3.2. Dynamic Accuracy

For measuring accuracy and numerical stability of the algorithms, we need to set up a "standard" material on which the experiments are carried out, as well as the rules allowing to extrapolate the results to any material of different size and parameters.

In the scope of our study, we restrict the experimentation to linear metric elasticity of an isotropic cloth material, described by a Young modulus **E** and a surface density **d**. For the simulation, the surface square is discretized into elements which roughly have the length **l**, and the computation is carried out with time steps of size **t**.

Defining the Condition Coefficient

Thanks to the linearity of the equations describing linear elasticity, we reduce the number of parameters describing a problem using proportionality laws, and we compute a "condition coefficient" **K** which illustrates the acceleration of a mesh element with normalization to the

problem mass, simulation time step and element size, as follows:

$$\mathbf{K} = \mathbf{E}\,\mathbf{d}^{-1}\,\mathbf{l}^{-2}\,\mathbf{t}^{2} \qquad (1)$$

This non-dimensional coefficient actually measures the "numerical difficulty" of computing an iteration of time step **t** with the given problem. We checked experimentally with our implementation that any scaling of a simulation along distance, time and mass which leaves **K** unchanged does not change anything to the simulation result.

A typical cloth simulation problem could involve a cotton fabric cloth surface, which typically have a density $\mathbf{d = 0.1\ kg.m^{-2}}$ and a Young modulus $\mathbf{E = 20\ N.m^{-1}}$. Given a discretization into elements averaging one centimeter and a simulation time step of ten milliseconds, the condition coefficient of the problem computed with (1) is $\mathbf{K = 200}$.

It is possible to define similar coefficients related to bending and viscosity modulus. The corresponding **K** coefficients are respectively multiplied by additional $\mathbf{l^{-2}}$ and **t** factors. In simulations that consider simultaneously all these forms of mechanical behaviors, the dominant **K** coefficient rules the "numerical difficulty" of the problem.

The Free Fall Test

Our first test intends to measure the accuracy of the various methods in the context of accurate dynamic simulation. In such kind of simulation, the interest is to reproduce exactly the motion of a cloth object along time, the accuracy of its evolution being the key of the realism of an animation involving simulated cloth.

In this experiment, we let a horizontal fabric square of **1.0 m** side length, initially still, fall by its own weight, under a gravitation field of $\mathbf{10\ m.s^{-2}}$ acceleration (Fig.2). The discretization was set to roughly 400 elements ($\mathbf{l = 0.05\ m}$), and its Young modulus was altered in order to obtain various **K** values, with a simulation time step of $\mathbf{t = 0.01\ s}$.

When using implicit methods, we perform a preconditioning of the system state variables of the linear system to be resolved using the inverse square root of the mass of the corresponding particle. This allows the iterations of the Conjugate Gradient algorithm to distribute the resolution numerical errors as evenly as possible between the particles, so that to obtain for instance a fall speed that does not depend on the mass of the particle.

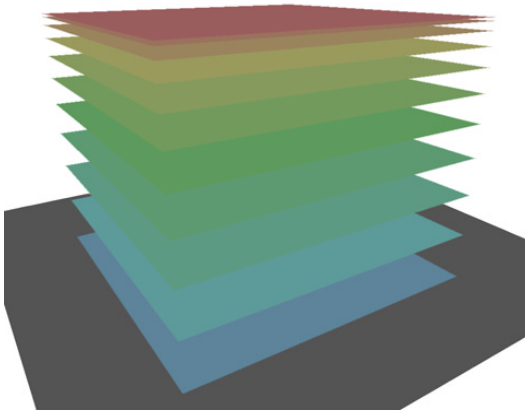

Fig.2. The free fall test: A horizontal fabric square, initially at rest, is dropped from **1 m** height in a gravity field of $\mathbf{10\ m.s^{-2}}$.

We measure the time it takes for this fabric piece to fall a height of **1 m**. Without any additional external forces considered (no aerodynamic interactions), we expect this to happen in a constant time of **0.45 s**.

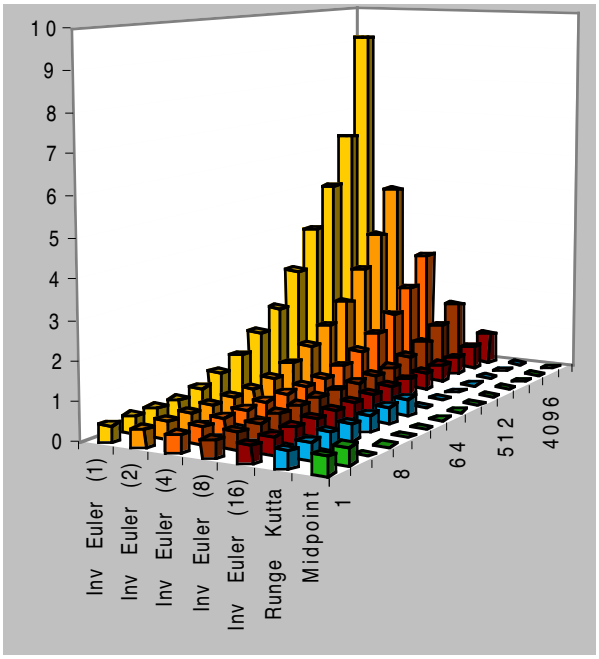

Fig.3. Fall time (vertical, seconds) with various **K** values (right) and different integration methods (left).
Theoretical time: **0.45 s**. Null values indicate numerical instability.

Several interesting facts arise from this experiment. As a matter of numerical stability, the Midpoint method supports **K** values up to almost **3** whereas the Runge-Kutta method supports **K** values up to almost **100**. This indicates that with Runge-Kutta, it is possible to use simulation time steps which are almost six times larger than with Midpoint. Given the fact that a Runge-Kutta iteration takes only three times more computation than a Midpoint iteration (Fig.1), the Runge-Kutta method seems to be computationally two times more efficient than the Midpoint method.

As a matter of simulation accuracy, both Midpoint and Runge-Kutta seem to preserve accuracy correctly within their range of numerical stability. While the implicit Euler method seems stable for any **K** value, its accuracy is however very degraded by high **K** values and reduced numbers of Conjugate Gradient iterations. More precisely, we see that accuracy is well preserved with one Conjugate Gradient iteration up to a **K** value of **4**, and increasing the iteration number **n** times also increases the **K** value $\mathbf{n^2}$ times for the same accuracy.

From this, we can see that the Inverse Euler method needs at least four Conjugate Gradient iterations to reach the accuracy of the Runge-Kutta method. We also see that similar requirement of accuracy bring the two methods in parity in terms of computation time (Fig.1).

However, it should be noted that the experiment was carried out using a uniformly discretized mesh, and uniform mechanical parameters. Real-world simulations do not have this regularity, and numerical instability with explicit methods occur in the stiffest regions of the mesh, which, even if they are marginal in the whole mechanical

system, may totally "explode" and destroy the simulation and therefore will rule the size of the largest time step possible. With implicit methods, the resulting inaccuracies may be unnoticed when taking a time step adapted to the average stiffness.

Anyhow, this experiment shows clearly that when accurate reproduction of dynamic motion is required, it is not possible to increase the time step of implicit methods as much as desired, as this cause very noticeable inaccuracy as weak forces will be "neglected" relatively to stiff forces. While this is not an issue for draping problems where only the final state is desired, this aspect has to be taken into account when accurate reproduction of the whole evolution is wanted. While implicit Euler is a good choice for robust simulation where accuracy is not really an issue, the explicit Runge-Kutta offers good possibilities of ensuring high accuracy because of its high-order solution, and also because is provides good possibilities integration error evaluation for efficient time step control which, by the way, is too context-sensitive for being pre-evaluated using only the knowledge of **K**.

Discretization and Computation Time

The condition coefficient value is a good indicator of the time step and accuracy that can be expected for a given problem with a given iteration time step. Considering a simulation involving elements **n** times smaller, maintaining accuracy and stability (preserving **K** constant in formula (1)) would require a time step **n** times smaller, and therefore **n** times as many iterations for simulating the mechanical system along a constant duration. Given the fact that there are also $\mathbf{n}^2$ times more elements to handle, the total computation time is finally multiplied by a drastic $\mathbf{n}^3$ (even $\mathbf{n}^4$ if curvature stiffness rule the simulation accuracy). While this factor is what cause explicit methods to become so inefficient with refined discretizations as this scaling has to be strictly observed for preventing instability, implicit methods are a bit more tolerant if only "visual" accuracy matters, accuracy which is not related to the size of the elements.

3.3. Draping Speed

Draping is another context of simulation, where only the final static equilibrium state of the mechanical system is to be computed. Here, the interest is to converge to the equilibrium state as quickly as possible, with minimum computation charge. As the full evolution of the cloth along time is not an interest, accuracy can be traded away for computation speed.

From the dynamic study described above, implicit methods should be quite strong on this point, as they do not suffer from numerical instability, and allow large time steps to be used at the expense of dynamic accuracy which can here be neglected.

The Draping Test

For this test, we let an initially horizontal fabric square, attached along one of its edges, fall by its own weight (Fig.4). The fabric is a cotton sample of $\mathbf{1\ m}$ side length, with a Young modulus $\mathbf{E = 20\ N.m^{-1}}$ and a density $\mathbf{d = 0.1\ kg.m^{-2}}$, discretized into 10000 polygons $\mathbf{l = 0.01\ m}$, and put in a gravitation field of $\mathbf{10\ m\ s^{-2}}$. Without any damping, we expect that in its first oscillation, the fabric reach a roughly vertical position after slightly more than half a second.

Our purpose is here to find the computation time necessary to obtain the fabric in its vertical position. For this, we count the number of computation iterations necessary for obtaining the fabric in its vertical position in its first oscillation, not being interested by the realism of this motion (Fig.5).

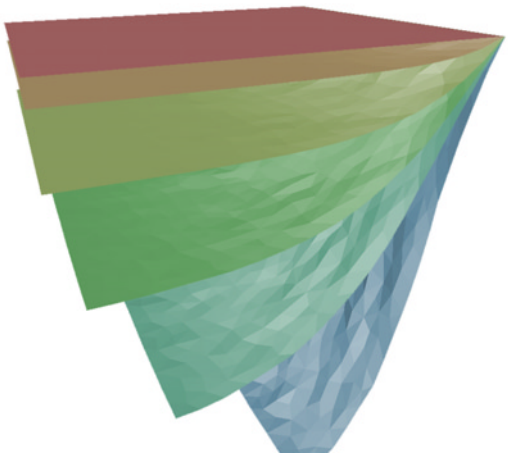

Fig.4. The draping test: An horizontal **1 m** square of fabric fixed along a side falling in a gravitation field of $\mathbf{10\ m\ s^{-2}}$.

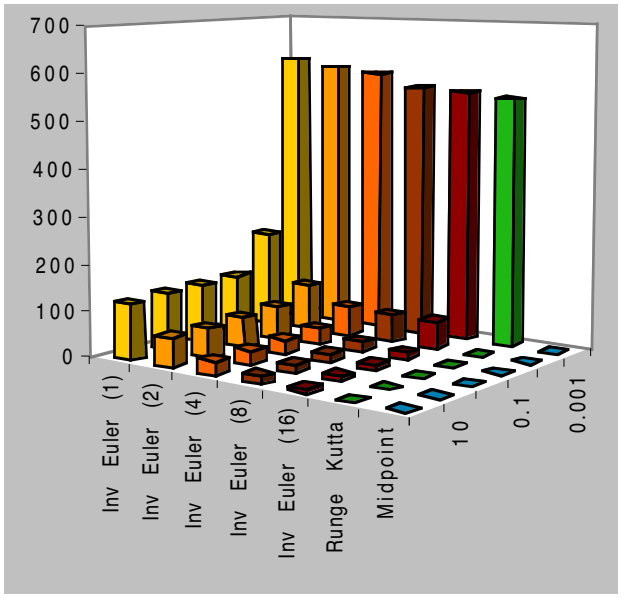

Fig.5. Number of iterations (vertical) required to get the fabric at vertical position, with various time steps (right, seconds), and integration methods (left). Null values indicate numerical instability.

Our first finding is that the explicit methods seem quite not adapted for draping. Runge-Kutta requires more than 500 iterations for performing the simulation without instability, with the maximum allowed time step $\mathbf{t = 0.001\ s}$, which in fact corresponds to the maximum value of **K** coefficient experimented in the previous section.

The backward Euler method is robust enough to handle the problem without instability for any time step.

However, we see that larger time steps do not proportionally translate into fewer steps for performing the draping. As the time step becomes larger, and as the corresponding **K** coefficient exceeds the theoretical limit observed in the previous section, we quickly observe a "saturation" of the number of iterations to a constant which seems to be inversely proportional to the number of Conjugate Gradient iterations that were performed.

From this it is clear that when **K** exceeds the dynamic accuracy limit of a given implicit integration method, the time step does not really reflect a time interval anymore. In such case, the implicit method will only evaluate an approximation of the rest state of the mechanical system by linear extrapolation from the Hessian matrix, whose accuracy depends on the number of Conjugate Gradient iterations that were used to resolve the corresponding linear system. Hence, there is no real way to "cheat" on the time step for speeding up draping, even if dynamic accuracy is not a concern: The total number of Conjugate Gradient iterations for performing all the simulation iterations of a draping problem cannot go below an incompressible number, related to a kind of "total computational difficulty" for solving a draping problem, which in the case of our experiment seems around 100.

Still, this experiment shows the drastic advantage of using implicit methods for draping problems: With our implementation using the accurate elasticity model and the computation times measured in section 3.1, the draping could be computed in 30 seconds with Backward Euler with any large time step, compared to 150 seconds with Runge-Kutta when using an "optimal" time step.

3.4. Dealing with Nonlinear Models

Most mechanical simulations work with numerical equations that are not linear. There are two main reasons for such nonlinearity:

* The equations describing the mechanical behavior laws are not linear. For instance, the strain-stress relation describing elasticity may actually be complex curves, which furthermore may take into account time-dependent and hysteretic behaviors.
* During the simulation, the orientation of the mechanical elements change, and this modifies the expressions of the mechanical laws in the world coordinates.

While rarely causing numeric "explosions" as with explicit methods, nonlinearity may disrupt the stability of simulations integrated with implicit models with large disturbing vibrations, particularly when using large time steps that cause iterations to converge to the equilibrium state of the mechanical objects rather than simulating accurately their mechanical behavior. This can for instance be observed when simulating stretched flat surfaces without curvature forces. The reason for that is that the hypothetical equilibrium state is derived from the knowledge of the Hessian matrix, which relates the first-order evolution of the forces as the deformations change. Nonlinearity causes this matrix to change between the successive iterations, and this evaluation to be inaccurate, despite high system resolution accuracy that can be reached with numerous Conjugate Gradient iterations.

The solution for this is to approximate the Hessian matrix for taking into account the changes that may be observed from the change of the system state between successive iterations. While an underestimation of de derivatives may lead to an equilibrium state valuation too far from the current state, and by this cause instability, an overestimation of the derivatives will place this evaluation nearer to the current state, therefore stabilizing the simulation, at the expense of extra numerical damping and slow convergence. This is particularly true for drastic linearisations as for example used in [DES 99].

Knowledge of the expected state changes between successive time steps are required to perform this approximation correctly. With nonlinear mechanical behavior, one solution is to take the steepest parts of the curves as derivatives, whereas for the element orientation problem, isotropic derivatives considering force evolution equally in any directions may be considered. However, the more drastic these approximations are, the less accurate the simulation will be for dynamic simulations, and the slower the simulation will converge for draping problems.

A nice solution described in [EBE 00], which makes sense when efficiency relies on the use of a constant Hessian matrix, is to perform the implicit resolution on a linear constant approximation, and to simulate the nonlinear and variable component, unlikely to cause stiffness problems, using an explicit method.

3.5. Real Case Simulation

In order to test the efficiency of our model in the context of garment animation, the algorithms have been integrated in a 3D design framework allowing the management of complex garment objects in interaction with animated virtual characters. This integration has been carried out in the form of a 3DStudio Max plugin (Fig.6), running on a 500 MHz PentiumIII PC.

We have simulated a 2000 Polygon garment made of the cotton material described in Section 3. The mesh elements are roughly five centimeters in size, and therefore the resulting condition coefficient **K** is roughly **8** with a simulation time step of 10 milliseconds.

The cloth simulation process has two stages:

* The garment assembly stage, where the patterns are pulled together and seamed around the body. This is a draping problem involving to obtain a rest position of the garment as quickly as possible.
* The garment animation stage, where the motion of the garment is computed as the body is animated. The dynamical motion of the cloth is important here.

The garment assembly and seaming operations could be performed almost four times faster with the Backward Euler (2 minutes) than with Runge-Kutta (8 minutes), knowing that collision detection and response account for more than the half of the computation time, and actually limits the time step size when contact starts between the cloth and the body.

For the dynamical animation, comparable accuracy could be obtained between Runge-Kutta and Backward Euler using eight iterations of the Conjugate Gradient, which gave similar computation times. The backward Euler method however allowed the increase of the time step up to 0.1 seconds, where inaccuracies began to show up: Mostly, excessive "heaviness" of the fabric that failed to follow the motion of the body properly, garments folds that would not disappear quickly, as well as additional artifacts caused by inaccurate collision response.

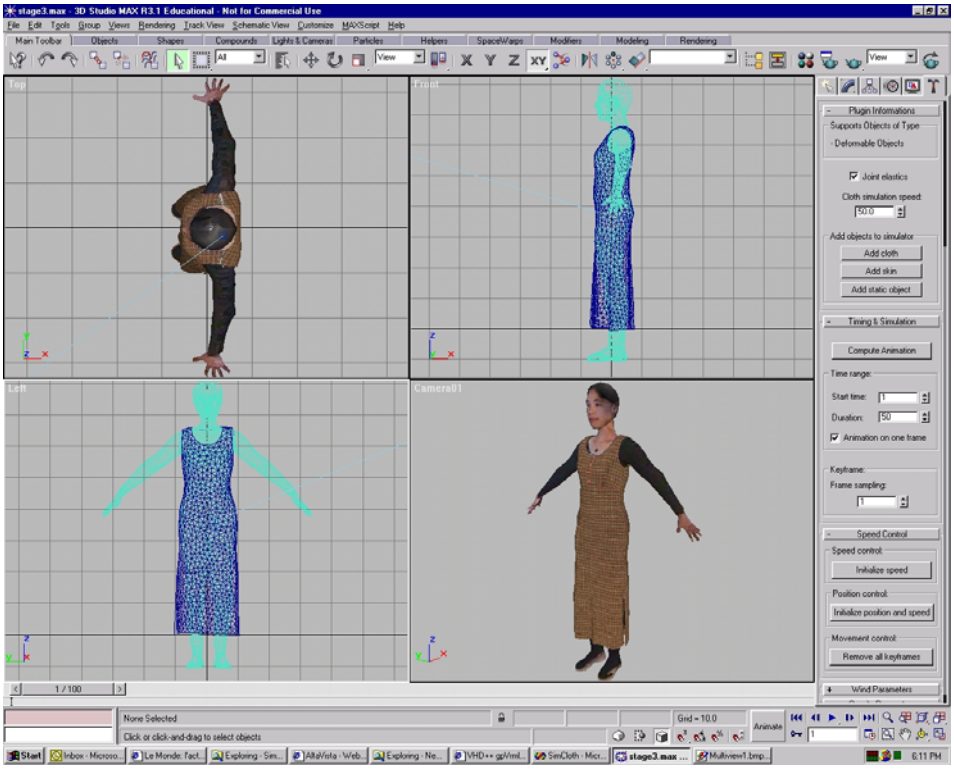

Fig.6. Garment simulation in the 3DStudio Max environment.

4. Conclusion

Recent literature has emphasized on the relevance of implicit methods for cloth simulation. The implicit Euler method seems effectively a good candidate for most situations involving cloth simulation, because of the robustness resulting from not being prone to numerical instability. This is particularly true when simulating very heterogeneous mechanical systems (elements of various sizes and various mechanical properties) where, using explicit models, the most critical elements would rule the time step size for all the simulation.

Contrary the perception of the implicit model iteration being slow because of the linear system resolution it involves, the inverse Euler iteration often proves to be faster than the explicit Runge-Kutta method of higher order, if an adequate approximate linear system resolution is implemented. A limited number of Conjugate Gradient iterations seems suitable for this. Furthermore, while increasing the time step seems not limited by instability with implicit methods, it should be kept in mind that this is still done at the expense of accuracy of the whole simulation. The number of iterations should also be set sensitively to the stiffness of the mechanical problem, for limiting the potential inaccuracies that become particularly visible when an accurate simulation of a dynamical system is wanted.

There is an obvious advantage of using implicit methods, and particularly the inverse Euler method, for draping problems where quick convergence to a rest position is required quickly. Our test have shown that the inverse Euler method allow to perform a draping problem almost ten times as fast as with the Runge-Kutta method. While not exactly reproducing real mechanical behavior, the simulation with large time steps provides a quite efficient convergence to equilibrium, and the numerical errors quite often act as extra damping, removing the need of adding them explicitly to the model.

For dynamic problems where accurate evolution of the mechanical system along time is needed, the advantage of implicit methods is less obvious. Their stability gives a false sense of efficiency, allowing obtaining quickly a result by “cheating” on the time step size. However, playing back the generated animation, artifacts quickly show up: Excessive damping, wrinkles and folds that fail to disappear, and even objects failing to fall correctly by their own weight. These artifacts are still augmented by the approximations made to the Hessian matrix, possibly in the purpose of reducing instability, while excessive reduction of the Conjugate Gradient iterations produce additional inaccuracy and slow convergence.

It seems that there is still some benefit in using the Backward Euler method than any other explicit method for dynamic simulations thanks to the reduced time it takes to compute one iteration, which also only requires one derivation of the particle forces from the state of the system. Our tests have shown a roughly doubled speed for the accuracy corresponding to the limit of stability of the Runge-Kutta method. We got substantial improvements through the implementation of the implicit Midpoint method [VOL 00], which however had the drawback of increasing the numerical instability problem, forcing additional use of isotropic force gradients, at the expense of accuracy.

The explicit methods have still their interest, and should be reserved for simulations requiring high accuracy

and particularly those where involving low mechanical damping and where mechanical energy conservation is important. Instability concerns will force parameters and time step size to ensure good accuracy for the simulation of all particles of the discrete mechanical representation, and therefore for the entire mechanical object. This may however require prohibitive computation times for very stiff and discretized models.

The 5th-order Runge-Kutta method das proven to be a good solution [EBE 96] [VOL 97], because of its high accuracy, and because it furthermore provides integration error evaluation, which is a very good hint to the very sensitive problem of optimal time step size determination. The simpler Midpoint method may have some interest only in very particular cases involving very loose materials with rough discretization, or when numerous fast iterations with small time steps are required for other reasons (high motion sampling, collision detection, very discontinuous models).

All these considerations should be carefully taken into account when designing a mechanical simulation engine, as they are the keys to efficient simulation, and therefore complex models that, for garment simulation, express fully visual experience of real fashion models.

We intend to pursue our investigations for dealing with damping in a more accurate way. This still remains an important issue to dynamic realism of cloth simulation models, which has to take into account viscosity, the dissipative effect of hysteretic behavior, as well as collision damping and friction. The integration methods have to be tuned to take precisely these effects into account.

Bibliography

[BAR 98] : **D. Baraff, A. Witkin**, "*Large Steps in Cloth Simulation*", Computer Graphics (SIGGRAPH'98 proceedings), Addison-Wesley, 32, pp 106-117, 1998.

[BRE 94] : **D.E. Breen, D.H. House, M.J. Wozny**, "*Predicting the Drape of Woven Cloth Using Interacting Particles*", Computer Graphics (SIGGRAPH'94 proceedings), Addison-Wesley, pp 365-372, July 1994.

[DES 99] : **M. Desbrun, P.Schröder, A. Barr**, "*Interactive Animation of Structured Deformable Objects*", Proceedings of Graphics Interface, 1999.

[EBE 96] : **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), pp 52-59, Sept. 1996.

[EBE 00] : **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, pp 137-151, 2000.

[EIS 96] : **J.W. Eischen, S. Deng, T.G. Clapp**, "*Finite-Element Modeling and Control of Flexible Fabric Parts*", Computer Graphics in Textiles and Apparel (IEEE Computer Graphics and Applications), pp 71-80, Sept. 1996.

[KAN 00] : **Y.M. Kang, J.H. Choi, H.G. Cho, D.H. Lee, C.J. Park**, "*Real-Time Animation Technique for Flexible and Thin Objects*", WSCG proceedings, pp 322-329, 2000.

[PRE 92] : **W.H. Press, W.T. Vetterling, S.A. Teukolsky, B.P. Flannery**, "Numerical Recipes in C", Second edition, Cambridge University Press, 1992.

[TER 87] : **D. Terzopoulos, J.C. Platt, H. Barr**, "*Elastically Deformable Models*", Computer Graphics (SIGGRAPH'97 proceedings), Addison-Wesley, 21, pp 205-214, 1987.

[VOL 95] : **P. Volino, M. Courchesne, N. Magnenat-Thalmann**, "*Versatile and Efficient Techniques for Simulating Cloth and Other Deformable Objects*", Computer Graphics (SIGGRAPH'95 proceedings), Addison-Wesley, pp 137-144, 1995.

[VOL 97] : **P. Volino, N. Magnenat-Thalmann**, "*Developing Simulation Techniques for an Interactive Clothing System*", Virtual Systems and Multimedia (VSMM'97 proceedings), Geneva, Switzerland, pp 109-118, 1997.

[VOL 00] : **P. Volino, N. Magnenat-Thalmann**, "*Implementing fast Cloth Simulation with Collision Response*", Computer Graphics International 2000, pp 257-266, 2000.