Implicit and Explicit Methods of Cloth Simulation

Gavin Hayler, Shaun Bangay and Adele Lobb

Abstract

This paper examines two methods of integration used in cloth modelling, looking at their good and bad points mainly from an implementation viewpoint. The maths behind the methods is looked at and a way to implement the methods is presented through psuedo code. It is concluded that the explicit methods are good for real-time speed-oriented applications, whereas realism-oriented applications would need to use an implicit method.

1 Introduction

Cloth modelling is a highly researched area of computer graphics, and has changed much over the past few years. Modelling cloth in a virtual world is not a trivial task, as there are many different facets that need to be combined to provide a complete solution. This paper will be exploring some of the different methods used within the area of cloth dynamics which involves how the cloth moves and reacts to forces applied to it. The methods that will be explored are explicit and implicit integration.

2 Related Work

2.1 Physical Model

A large amount of research has gone into the field of cloth simulation, and many different solutions to the problem have been devised. For the most part, when developing a solution, the researcher needs to decide on a physical model for the cloth and on an integration method.

The physical model refers to the way that the cloth or particle system is represented logically. A number of different methods have been used in the past (EISCHEN et al. 1996; BARAFF and WITKIN 1998), but these models have been superseded by a system of particles which is used to represent the cloth. This method is highly intuitive since for the cloth to be drawn, it needs to be represented as a mesh which consists of a number of vertices. Those vertex positions can then be used as the particle positions in the system. This model is defined well by Choi *et al.* (CHOI and KO 2002), who draw their model from an idea that started with Breen *et al.* (BREEN et al. 1994) who were first to apply a particle system to the simulation of cloth. Choi *et al.* (CHOI and KO 2002) describe the model as a mesh of interconnected particles which approximate the cloth. These particles are connected by a system of springs with certain attributes assigned to them,

depending on the type of interaction that the spring is dealing with (stretching or compression). In general, variations of this physical model are now used throughout the cloth modelling field - by game developers and CGI animators alike. (JAKOBSEN 2001; CHOI and KO 2002; ETZMUSS et al. 2001; BREEN et al. 1994)

2.2 Integration Method

The integration method is a method which takes the forces on the particles, their velocities and positions and calculates where the particle must move to next, and what its velocity will be. There are mainly two methods of integration in use in the cloth simulation field - explicit and implicit integration. For some time, explicit integration dominated the field, possibly because it is the most obvious and intuitive solution, until Baraff and Witkin (BARAFF and WITKIN 1998) introduced the idea of implicit integration into cloth modelling.

Explicit integration attempts to solve the system based on the information available at time t_0. The reason that the time step has to be small to maintain stability, is that there will not be any huge changes in the system between each step which could corrupt the results (BARAFF and WITKIN 1998). The more particles that exist in the system, the smaller the time step needed to keep the stability of the system (OSHITA and MAKINOUCHI 2001). In general, game developers try to keep the number of vertices for each object to a minimum, making games an excellent application for explicit integration, as shown by the explicit method used by Jakobsen (JAKOBSEN 2001) in the physics engine for the game Hitman: Codename 47.

The new implicit method introduced by Baraff *et al. (*BARAFF *and* WITKIN *1998)* allows large time steps, and is able to handle large numbers of particles without much variance in the running times (ie: 5%). This method is largely used for accurate simulation of high resolution models, where the application is not intended to be real-time; namely animations.

Choi and Ko (CHOI and KO 2002) implement a variant of Baraff and Witkin's implicit integration, and demonstrate that they are able to use a time step of up to 100 seconds and still keep the system stable, even though the animation is choppy. However, when they use a more realistic time step, they are able to achieve most convincing results using a clothed animated character.

3 Physical Model

The physical model that is used for this research paper is based on the model that is mentioned in Section 2.1, that is the most widely used. Particles are used to represent the cloth vertices, and springs are set up between certain particles. The intricasies of the model are taken from the paper by Choi *et al.* (CHOI and KO 2002). In this particular model, there are two different types of springs used in the cloth, based on the type of interaction they are attempting to model: springs for compression and springs for stretching. The mathematical formulas for these springs are different, because Choi *et al.* decided that the stretching and compression interactions were different to warrant it, whereas often these two formulas are very similar, if not the same.

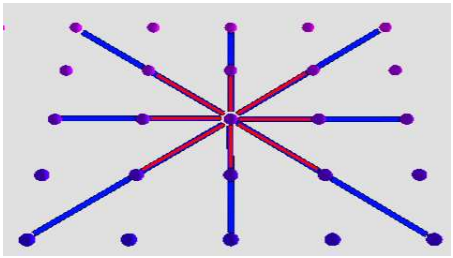

Figure 1: Physical Model of Cloth

3.1 Stretching

Particles that are directly adjacent to each other are used to simulate the stretching interaction, and particles that are two particles apart, are used to simulate the compression interaction. In Figure 1, the red lines represent the springs used to connect the adjacent particles for the purpose of calculating the stretching interaction, and the blue lines represent the springs used for compression interaction.

Therefore, each particle is connected to a number of other particles for the purpose of calculating the forces acting on that particle. Each of these forces is calculated and then added together to produce the resultant force on that particle. The stretching forces are calculated using the following formula:

$$f_i = \begin{cases} k_s(|\mathbf{x}_{ij}| - L)\frac{\mathbf{x}_{ij}}{|\mathbf{x}_{ij}|} & : \quad |\mathbf{x}_{ij}| \geq L \\ 0 & : \quad |\mathbf{x}_{ij}| < L \end{cases} \tag{1}$$

where k_s is the spring constant, $\mathbf{x}_{ij} = \mathbf{x}_j - \mathbf{x}_i$ and L is the rest length of the spring (the length when there are no forces acting on it). What this means is that a force is exerted if the current distance between two particles is greater than the rest length between them (ie: they are being stretched apart).

3.2 Compression

The compression interaction forces are computed with the following formula:

$$f_i = f_b(|\mathbf{x}_{ij}|)\frac{\mathbf{x}_{ij}}{|\mathbf{x}_{ij}|} \tag{2}$$

where

$$f_b(|x_{ij}|) = k_b\kappa^2(\cos\frac{\kappa L}{2} - \text{sinc}(\frac{\kappa L}{2}))^{-1} \tag{3}$$

and the curvature k of an arc can be expressed in terms of the distance between two points. k can be calculated using the following formula: $\kappa = \frac{2}{L}\text{sinc}^{-1}(\frac{|\mathbf{x}_{ij}|}{L})$, where $\text{sinc}(x) = \frac{\sin(x)}{x}$.

Unfortunately, since there is no useful definition for the function $\text{sinc}^{-1}(x)$, the best way to use it in implementation is to create a lookup table with all the values that could be needed, and write a function that looks up the correct value when the corresponding value is passed to it.

In real systems, geometric imperfections in the structure cause the fabric to begin buckling when the compressive force is first applied. To model this characteristic, another function (f_b^*) is used in the implementation, which simulates this behaviour at small compressive forces, and as the force increases behaves as f_b would.

$$f_b^* = \begin{cases} c_b(|\mathbf{x}_{ij}|) - L & : \quad f_b < c_b(|\mathbf{x}_{ij}| - L) \\ f_b & : \quad \text{otherwise} \end{cases} \tag{4}$$

where c_b is an arbitrary compression constant, similar to our spring constant k_s.

3.3 Damping

One final attribute needs to be added to the model of the cloth: damping. Many models in the past have introduced a damping term into the model in order to keep it stable, but the side-effect of this is that the cloth begins to move unrealistically. The damping term introduced into the model by Choi *et al.* is merely to model the natural property of cloth to dissipate its energy. Without this term, the simulated cloth can move in large unrealistic oscillations.

The damping force exerted on particle i due to the interaction from particle j is shown below.

$$f_i = -k_d(\mathbf{v}_i - \mathbf{v}_j) \tag{5}$$

where k_d is a damping constant of our choice and $\mathbf{v}_i$ and $\mathbf{v}_j$ are the velocites of the two particles. This damping force is added to every interaction force that is calculated.

3.4 Pseudo Code

A pseudo code example for the force calculations would be as follows:

Add external forces to each particle (eg: wind),
For each particle, calculate the stretching force and add a damping force before adding it to the particle,
For each particle, calculate the compression force as done for the stretching force.

After completing this procedure for the particles, each particles contains the resultant force that is being applied to it taking into account al types of forces.

4 Integration

Integration is the process in cloth simulation of calculating the next position that each particle must move to and its velocity based on the forces acting upon that particle, its current position and its current velocity. A time step is chosen beforehand, and the system is advanced from the current point in time to the state it would be in one time step in the future. Therefore, once all the forces have been calculated using the different interactions between the particles, these values are then used to calculate the particles' next position and next velocity in the integration step.

4.1 Explicit Integration

As was mentioned before, explicit integration was the most widely used method of integration for some time. It is still quite widely used for certain applications for one reason - it is fast. It is not a particularily stable method at large time steps or at large numbers of particles, but in real-time simulations, speed is the main driving force and the models used in such applications are usually low in detail.

The method of explicit integration that was used for this paper is a method called Verlet integration and is taken from Jakobsen (JAKOBSEN 2001). Verlet is different from the classic Euler integration which is defined as $\mathbf{x}^{n+1} = \mathbf{x}^n + \mathbf{v}^n(\triangle t)$ and $\mathbf{v}^{n+1} = \mathbf{v}^n + \mathbf{a}^n(\triangle t)$ where $\mathbf{x}^n$, $\mathbf{v}^n$, $\mathbf{x}^{n+1}$ and $\mathbf{v}^{n+1}$ are the current postion and velocity and the new position and velocity of the particle respectively, and $\triangle t$ is the time step. $\mathbf{a}^n$ is the acceleration of the particle and is computed using Newton's law: $\mathbf{f}^n = m\mathbf{a}^n$ where $\mathbf{f}^n$ is the accumulated force acting on the particle (as computed beforehand).

Verlet, on the other hand, does not use the velocity of the particle in the calculation. Instead, it uses the current postion $\mathbf{x}^n$ and the old position $\mathbf{x}^{n-1}$ to calculate the new position.

$$\mathbf{x}^{n+1} = 2\mathbf{x}^n - \mathbf{x}^{n-1} + \mathbf{a}(\triangle t) \tag{6}$$

$$\mathbf{x}^{n-1} = \mathbf{x}^n \tag{7}$$

Verlet integration is more stable than other explicit integration methods, because the velocity is implicitly given through $\mathbf{x}^n - \mathbf{x}^{n-1}$ (the distance travelled in the previous time step), which makes it harder for the velocity and position to get out of sync. This method is, however, not always accurate, as energy may dissipate or leave the system, but it is fast and relatively stable.

Pseudo Code:
For each particle
Calculate next position using past and current position
Old position takes current position value
Current position takes new position value
Next particle

4.2 Implicit Integration

Implicit integration was first introduced by Baraff and Witkin (BARAFF and WITKIN 1998) in 1998 and was a revolutionary step in cloth modelling. In constrast to the explicit methods which use current or past information about the particle to compute the next position and velocity, implicit integration estimates the next position and uses that in a calculation to see if the prediction was correct. This procedure is performed over and over until it is determined that the prediction is correct. The implicit integration that was used in this paper was a method described by Choi *et al.* (CHOI and KO 2002), which they term a semi-implicit integration method with a second-order backward difference formula (BDF). In their formula, they use the partial differentials $\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$ and $\frac{\partial \mathbf{f}}{\partial \mathbf{v}}$ which we approximate with $\frac{\mathbf{f}^n - \mathbf{f}^{n-1}}{\mathbf{x}^n - \mathbf{x}^{n-1}}$ and $\frac{\mathbf{f}^n - \mathbf{f}^{n-1}}{\mathbf{v}^n - \mathbf{v}^{n-1}}$ respectively. The change in position

and hence the next position can be calculated by solving the following equation for $(\mathbf{x}^{n+1} - \mathbf{x}^n)$:

$$(\mathbf{I} - \triangle t\frac{2}{3}\mathbf{M}^{-1}\frac{\partial \mathbf{f}}{\partial \mathbf{v}} - \triangle t^2\frac{4}{9}\mathbf{M}^{-1}\frac{\partial \mathbf{f}}{\partial \mathbf{x}})(\mathbf{x}^{n+1} - \mathbf{x}^n) = \frac{1}{3}(\mathbf{x}^n - \mathbf{x}^{n-1})$$

$$+\frac{\triangle t}{9}(8\mathbf{v}^n - 2\mathbf{v}^{n-1}) + \frac{4 \triangle t^2}{9}\mathbf{M}^{-1}(\mathbf{f}^n - \frac{\partial \mathbf{f}}{\partial \mathbf{v}}\mathbf{v}^n) - \frac{2 \triangle t}{9}\mathbf{M}^{-1}\frac{\partial \mathbf{f}}{\partial \mathbf{v}}(\mathbf{x}^n - \mathbf{x}^{n-1}) \quad (8)$$

If the current position $\mathbf{x}^n$ is added to the result, it will produce the next position $\mathbf{x}^{n+1}$. Using the next position, we are then able to calculate the following velocity $\mathbf{v}^{n+1}$ by solving

$$\mathbf{v}^{n+1} = \frac{1}{\triangle t}(\frac{3}{2}\mathbf{x}^{n+1} - 2\mathbf{x}^n + \frac{1}{2}\mathbf{x}^{n-1}) \quad (9)$$

In the implementation, the change in position and the change in velocity is stored on each iteration and a new value is calculated each time. This process continues until the two values are within a pre-defined limit of each other, which then causes the program to break out of the loop.

Pseudo Code:

For each particle
  Reset delta values
  Do
    Calculate delta position and delta velocity
  Until difference between the old and new delta values is within a tolerable range
  New particle position and velocity calculated from delta values and assigned to particle
Next particle

In the above psuedo code, the delta values refer to the change in position $(\mathbf{x}^{n+1} - \mathbf{x}^n)$ and the change in velocity $(\mathbf{v}^{n+1} - \mathbf{v}^n)$.

5 Results

Figure 2 contains the results of the tests done for the explicit or verlet integration. The grid size refers to the number of particles in the system (ie: a 5 particle by 5 particle square grid); the FPS is the number of frames that were able to be rendered per second; and the stable timestep is the highest value that could be used as the timestep while still keeping the system stable.

As one can see from Figure 2, as the number of particles in the system increases, the frames per second (FPS) stays the same until the point at which the video card is no longer the bottle neck of the application, at which point the FPS decreases rapidly as the work continues to increase.The stable timestep value decreases almost linearly as the number of particles increase.

6 Conclusion

The explicit method that has been examined in this paper is surprisingly stable, as can be seen from Section 5, and would be most useful in an application where there are not going to be many particles in the system. Such an application would perhaps be a computer game, where models

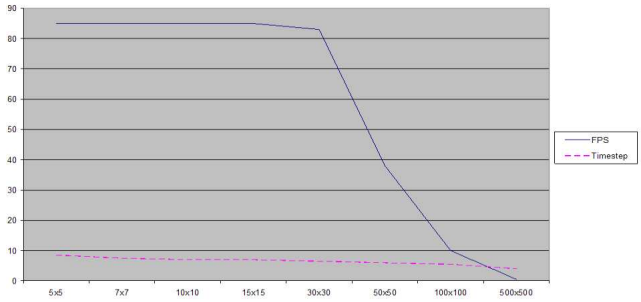

Figure 2: Explicit Integration Results

tend to be kept to as low a detail as possible to keep speed high. However, for use in an offline rendering environment, where high detail models are the norm, this method of integration would not be as well suited. A better method would be a highly stable method, such as the implicit method reviewed here.

References

BARAFF, DAVID and A. WITKIN (1998). Large steps in cloth simulation. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques* (1998), ACM Press, pp. 43–54.

BREEN, D. E., D. H. HOUSE and M. J. WOZNY (1994). Predicting the drape of woven cloth using interacting particles. *Proceedings of SIGGRAPH 1994* (1994), pp. 365–372.

CHOI, KWANG-JIN and H.-S. KO (2002). Stable but responsive cloth. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (San Antonio, Texas, 2002), San Antonio, Texas, ACM Press, pp. 604–611.

EISCHEN, J. W., S. DENG and T. G. CLAPP (1996). Finite-element modelling and control of flexible fabric parts. *IEEE Computer Graphics and Applications 16*, 5 (1996), pp. 71–80.

ETZMUSS, OLAF, M. HAUTH, M. KECKEISEN, S. KIMMERLE, J. MEZGER and M. WACKER (2001). A cloth modelling system for animated characters. Tech. rep., Wilhelm-Schickard-Institut für Informatik, Graphisch-Interaktive Systeme, Universität Tü, 2001.

JAKOBSEN, THOMAS (2001). Advanced character physics. In *Proceedings of Game Developer's Conference* (San Jose, 2001), San Jose.

OSHITA, M. and A. MAKINOUCHI (2001). Real-time cloth simulation with sparse particles and curved faces. In *Proceedings of Computer Animation 2001* (November 2001), pp. 220–227.