Numerical Methods for Cloth Simulation

DUARTE DAVID, University of Saarland

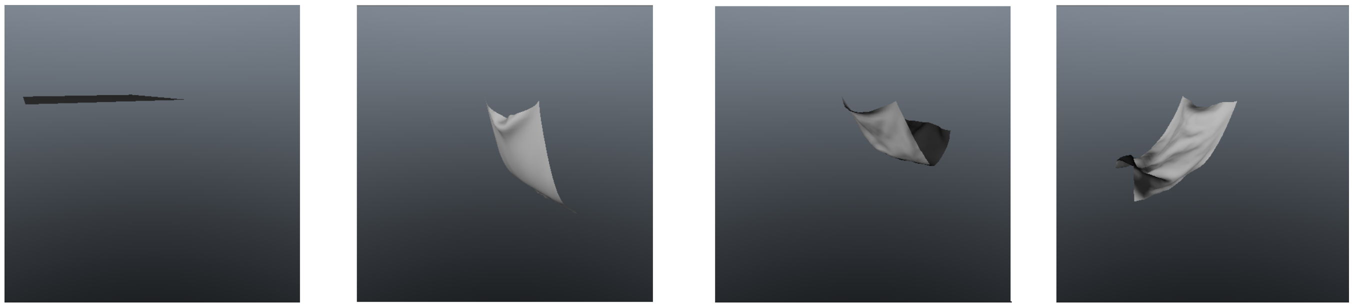

Fig. 1. Simulation of a 2x2 meters sq. piece of cloth over three seconds.

We summarize a physical model for cloth simulation and experiment various numerical algorithms to solve the resulting ODE.

Additional Key Words and Phrases: Physically-based Animation, Cloth Simulation, Differential Equations

1 INTRODUCTION

Cloth modelling and simulation is an important topic both within the textile fabrication community and the computer graphics community. While computational fabrication seeks to simulate accurate cloth for computer assisted design and preview of real cloth, in Computer Graphics accuracy is often less important than efficiency and visual realism.

In Computer Graphics and Animation, cloth simulation and animation is used to represent the clothes of virtual characters, as well as other forms of cloth in the environment, adding to the realism of simulated scenes.

Through this report, we will start by stating the model on which we base our simulation. We will then list the numerical methods used to solve it. Afterwards, we conclude by looking at the simulation results, looking at the total energy of the cloth along the simulation time for each of the methods and comparing the error.

2 CLOTH MODELLING

We use a particle model to describe cloth, similar to the one employed by [Provot 1996]. We consider planar rectangles as the basis for our cloth. We discretize it into a grid, and consider each vertex of the grid a particle with mass $m = \frac{total_{mass}}{n_{particles}}$, where $total_{mass}$ is the mass of the cloth and $n_{particles}$ is the number of vertices on the grid. The force applied to a particle i can be written as $Force_i = F_{internal,i} + F_{external,i}$, the sum of internal and external forces applied on it.

2.1 Internal Forces

If we don't consider dissipative phenomena, we can write

$$F_{internal,i} = -\frac{dU}{dx_i}$$

where x_i is the position of particle i, and the potential energy $U = U_{stretch} + U_{trellis} + U_{bending}$ [Breen et al. 1994], which is the sum a potential energy due to stretching, shearing and bending the cloth,respectively. Each of these energies may be decomposed as a sum of potential energy of "springs" connecting neighbouring particles. $U_{stretch}$ is the sum of the potential energies of springs connecting vertically and horizontally adjacent particles. $U_{trellis}$ is the sum of the potential energies of springs connecting diagonally adjacent particles. $U_{bending}$ is the sum of the potential energies of springs connecting particles (i, j) to $(i, j + 2)$ and (i, j) to $(i + 2, j)$, where i,j are the positions in the cloth grid.

Figure 2 shows the diagram of the springs.

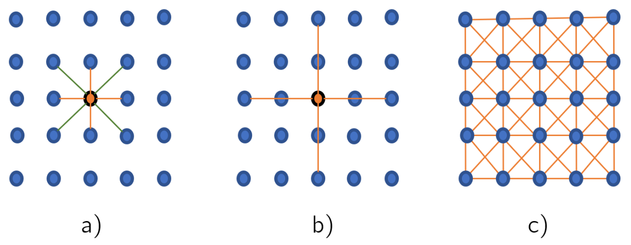

Fig. 2. Diagram of the springs in the model. a) represents the springs that counteract stretching (blue) and shearing (green), acting on the orange particle. b) represents the springs that counteract bending. c) represents all the springs in a 4 by 4 mesh of cloth

$U_{spring,i,j}$, the potential energy for a particle i connected to a particle j, is given by Hooke's Law and is

$$U_{spring,i,j} = \frac{1}{2}k(||(x_j - x_i)||_2 - r)^2$$

where r is the rest length of the spring (in our case, the length in the initial conditions) and k is the spring's stiffness.

The force exerted by such a spring on particle i is given by

$$F_{spring,i,j} = -\frac{dU_spring,i,j}{dx_i} = k\frac{(x_j - x_i)}{||x_j - x_i||_2}(||x_j - x_i||_2 - r)$$

For $U_{stretch}$ and $U_{trellis}$, the springs' stiffness describe the elasticity of the cloth, and we have $k = \frac{1}{elasticity}$. For $U_{bending}$, the springs' stiffness describe the bending stiffness of the cloth, and we have that $k = bendingStiffness$.

2.1.1 Dissipative Forces. We also consider a force that opposes the velocity in the direction of the springs (thus creating damped spring). Such force on a particle i caused by a spring j is given by $F_{i,j} = -n_j(v_i \cdot n_j)b$, where n_j is the unit vector in the direction of the spring, v_i is the velocity of particle i and b is the damping coefficient.

2.2 External Forces

2.2.1 Gravity. We consider a constant gravitational acceleration, $F_{gravity,i} = (0, 0, -9.8)m_i$.

2.2.2 Air Resistance. We subdivide the grid into triangles. Let T_i be the set of triangles for which i is one of its vertices' index, and consider the following drag force:

$$F_{airresistance,i} = -\frac{b}{m_i}\Sigma_{t \in T_i}(|n_t \cdot v_i|angle_{t,i}\frac{1}{\pi}area_t)$$

where b is the drag coefficient,n_t is the triangle normal, v_i is the velocity of vertex i, $angle_{t,i}$ is the angle of vertex i at triangle t and $area_t$ is the area of triangle t.[Provot 1996][House 2017].

2.3 Collisions

One usual way of treating collisions is to, at every simulation step, check if any object is colliding. If so, apply instantaneous collision response. However, doing so introduces a discontinuity in our system, as the velocity variables stop being continuous. Moreover, because we can only describe this force in terms of impulse (instantaneous velocity change), as opposed to an acceleration, our integrators cannot deal directly with it.

Instead, we treat collisions using the penalty method. For every 2 pairs of potential collisions (i.e.: between every 2 primitives that may collide with each other), it creates a potential energy

$$U_{penalty} = (1/2)k(||(x_2 - x_1)||_2 - T)^2$$

whenever the two closest points (x2 and x1) are closer than a certain threshold distance T. k is the "stiffness" of the collision. The force exerted on primitive 1 by the collision with primitive 2 is $F_{penalty}, 1, 2 = -\frac{dU_{penalty}}{dx_i} = k\frac{(x_2-x_1)}{||x_2-x_1||_2}(||x_2 - x_1||_2 - T)$. Similar to the damped springs, we add a damping force along the direction of this force to create inelastic collisions.

2.4 System of ODEs

Let F_i be the force exerted on a particle i, x_i its position and v_i its velocity, which are three dimensional vector. The solution x at a time t, $x(t) = (x_0(t), x_1(t), ..., x_n(t), v_0(t), v_1(t), .., v_n(t))$ is a $2 \cdot 3 \cdot n$ dimensional vector.

The system of ODEs can then be written as

$$(x_0', x_1', ..., x_n', v_0', v_1', .., v_n') =$$
$$x' = f(x, t) =$$
$$f(v_0, v_1, ..., v_n, a_0, a_1, .., a_n, t)$$

where a_i is the acceleration of particle i.

2.5 Limitations

One of the limitations of our modelling is that we don't consider internal collisions. If we considered internal collisions, a single naive evaluation of the forces would have $O(n^2)$ operations, where n is the number of faces, as we would need to compute the collision forces between every face and every vertex. We can decrease the complexity by noting that these "collision" forces act only on a limited radius (otherwise they are 0), and by introducing acceleration structures for spatial queries, like kd-tress, we can apply only the non-zero forces, with a complexity of $O(nlog(n))$. However, this is beyond the scope of this project.

Another limitation is that modelling the various potential energies using masses and springs only allows us to deal with a very specific type of grid-like meshes. In computer graphics, usually one wants to compute the deformation energy of arbitrary meshes made of triangles. To do so, we could have instead considered the deformation of each triangle and computed the deformation gradient, using it to derive shearing and stretching forces. We could then have used the angle at each edge to compute the bending forces [Baraff and Witkin 1998].

Another problem comes from our handling of collisions, as they do not enforce the constraints precisely, and introduce additional stiffness to the system.

3 NUMERICAL SOLVERS

We use various numerical ODE solvers to solve the ODE obtained in the previous section.

- Explicit Euler - consistency order 1
- Implicit Euler - consistency order 1
- Symplectic Euler - consistency order 1
- Störmer-Verlet method - consistency order 2
- "The" 4th order Runge-Kutta method (RK4) - consistency order 4
- Adams' Method of order 4 (using RK4 for the first 4 steps) - consistency order 4

Of these, the only implicit method is *Implicit Euler.* Implicit methods require finding the fixed point of a function. We assume the function is a contractive mapping within the domain and image set in which we are evaluating that function (which only happens for a sufficiently small timestep) and compute the fixed point iteratively according to Banach's Fixed Point theorem. We stop the iteration if we are sufficiently close to the fixed point or if we have alreadydone more than 10 iterations, as each iteration requires one evaluation of the right hand side of the ODE, f, which is very costly. One better way to do this would be to find the fixed point using Newton's Method. However, that would require computing the jacobian matrix of the forces, which proved to be too hard to implement for this project.

3.1 Computational Cost

The main bottleneck with each of these methods is the evaluation of the the right hand side of the ODE, f. As such, we can summarize their cost in terms of the number of times they have to evaluate f.

Our implementation of $ImplicitEuler$ typically requires 10 evaluations of f (number of iterations to find the fixed point) for every step. $RK4$ requires 4 evaluations of f for every step. All others require only one evaluation of f. We should note that our implementation of the Störmer-Verlet method only evaluates f once, as it computes the position and velocity at different points in time (on a staggered grid: the velocity is evaluated at the middle of the timestep, while the position is evaluated at the beginning).

4 DISCUSSION

4.1 Experiments

We considered 4 test scenes:

- Scene 1: A square cloth in free fall, hitting two capsules
- Scene 2: A square cloth attached by 2 vertices
- Scene 3:A square cloth attached by 3 vertices
- Scene 4: A square cloth attached by 2 vertices, hitting a capsule

And we varied various physical parameters of these scenes (damping coefficient, air drag...). The results obtained showcase the visual realism of our model, as the cloth deforms in a realistic way. The cloth is a 2x2 meters sheet, with 0.2kg, 0.2 elasticity, 0.2 bending stiffness (except in scene 3, where it is 1.0). The air resistance coefficient is 50 and the spring damping coefficient is around 0.001;

To test the accuracy of each of the numerical schemes, we ran 120000 steps of RK4 for various scenes to obtain the solution at $t = 4s$, which we consider the ground truth. We then ran the other methods and compare the difference in the squared error between them.

4.2 Results

4.2.1 Energy. We consider the second scene described above, without any dissipative force. The energy of the system (sum of potential and kinetic ($\sum_i \frac{1}{2} m_i v_i^2$) energy) should be constant, as all forces are conservative. Figures 3 and 4 show the results.

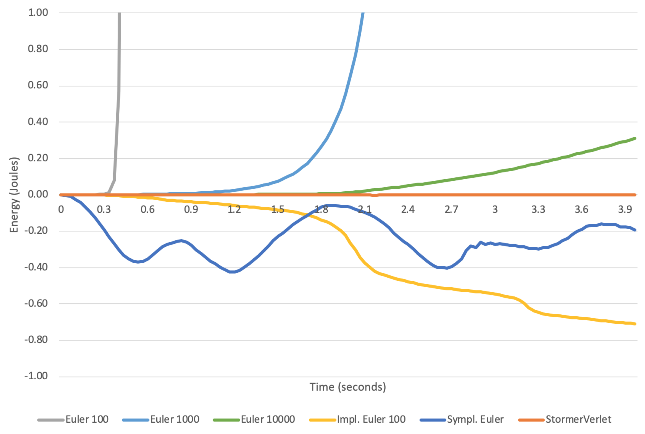

Fig. 3. Energy along the time of the trajectory, computed with the different integrators, with 100 steps per frame (and 30 frames per second. "Euler 100", "Euler 1000" and "Euler 10000" refers to *Explicit Euler* with 100, 1000, and 10000 steps per frame.

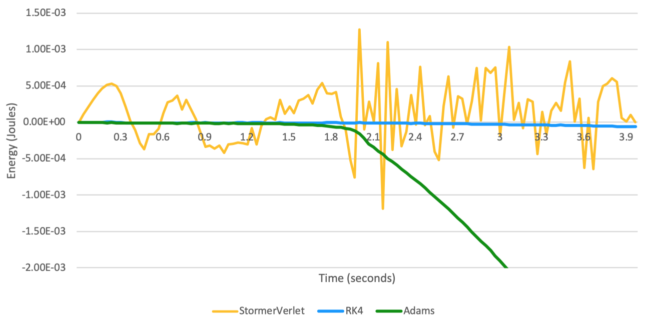

Fig. 4. Energy along the time of the trajectory, computed with the different integrators, with 100 steps per frame (and 30 frames per second.

It should be noted that the energy of the *Störmer-Verlet* method is not entirely accurate, as the position and velocity are not evaluated at the same point in time. However, they are close enough that is should not make a significant difference.

With *Explicit Euler*,due to the stiffness of the system, the energy quickly explodes. With *Implicit Euler*, the energy decreases. The symplectic methods (*Symplectic Euler* and *Störmer-Verlet*) neither systematically decrease nor increase the energy, with the energy instead oscillating. The error of the *Störmer-Verlet* method is significantly lower than *Symplectic Euler*. Adams' method starts with a very low error, but at around 1.8s (when the cloth "whiplashes" and the forces are greater) it makes an error which is then propagated, resulting in loss of energy. *RK4* has the least error, when it comes to energy conservation. All of these observations are according to our expectations.

4.2.2 Total error. The table below shows the final error for various scenes, at $t = 4seconds$. The error is computed as

$$\Sigma_i((p_i - p_{gt,i})^2 + (v_i - v_{gt,i})^2)$$

where p_i and v_i are the position and velocity of particle i, $p_{gt,i}$ and $v_{gt,i}$ are the ground truth position and velocity of particle i, all evaluated at $t = 4s$.

	Exp. Euler*	Imp. Euler**	Symp. Euler	Störmer-Verlet	RK4	Adams'
Scene 1	51987.50339	247.2503302	401.8342639	154.9491741	232.234782	2441.342397
Scene 2	73039.02593	44.23256231	457.1130984	70.78140386	55.6142788	62.50184305
Scene 3	0.770608416	2.241996207	7.944865596	0.011765561	0.000574313	0.006577008
Scene 4	12.98317294	7.075791357	16.5678817	8.478083835	8.238787012	247.4730069

*1000 steps per frame.
** 50 steps per frame.

Fig. 5. Error of the integrators with respect to each scene at $t = 4s$, computed with RK4 at 1000 steps per frame. Unless stated otherwise, each of the integrators ran at 100 steps per frame, for 120 frames (4 seconds).

We can see that the *Störmer-Verlet* integrator produces the most consistently good results.*Explicit Euler* only works in heavily damped scenes (even though it is doing 10 times as much work as *Störmer-Verlet*), otherwise it produces catastrophic results. As expected, *Symplectic Euler* produces always worse results and *Störmer-Verlet. RK4* and *Implicit Euler* produce results roughly on par with *Störmer-Verlet*, albeit at a greater computational cost. *Adams' method* depends a lot on the stiffness of the scene, as one bad step will more gravely affect subsequent steps.

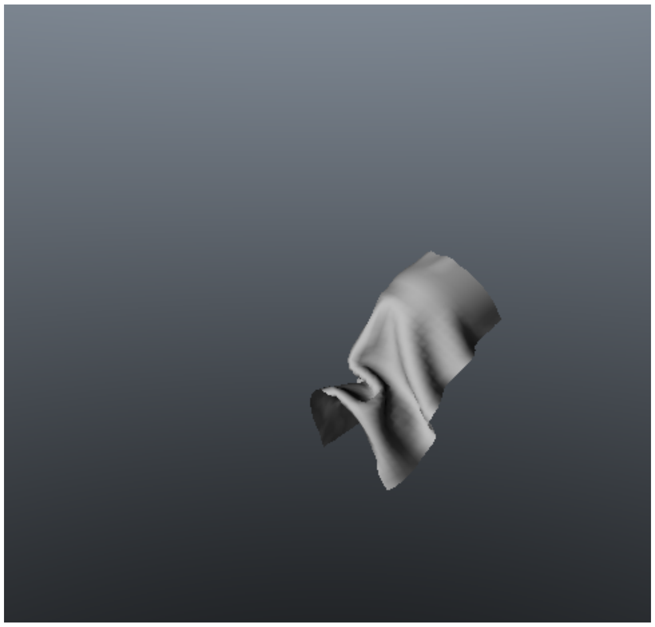

Fig. 6. Natural folding patterns are formed

4.2.3 Other results. The video that can be found at https://youtu.be/FVa_d85nywg showcases various results for various scenes with different physical parameters. Observe how the wrinkles characteristic to cloth are formed, as the grid deforms in a believable way. We also observe that, even though the consistency is worse than with *RK4*, the *Störmer-Verlet* method produces results on-par with *RK4*, even though it is much cheaper. We also observe that *Implicit Euler* produces very visually pleasing results, even though its current implementation is lacking in efficiency. In fact, all other methods showcase a certain high frequency "noise", while *Implicit Euler* does not show such artifact (note that the artifacts that may be seen in the video with *Implicit Euler* on scene 1 are due to the lack of self collisions, which causes the cloth to intersect itself).

5 CONCLUSIONS

We have observed the results of using a wide variety of numerical integrators to solve the ODE associated with a cloth simulation. Although many concessions were made when modelling the cloth, we still obtained reasonable results. We observed that explicit solvers, in particular *Explicit Euler* and *Adams' method*, tend to behave poorly with the stiffness of the system, while Symplectic and Implicit solvers behave better. In particular, we highlight the *Störmer-Verlet* integrator for producing very good results while having a very low computational cost.

REFERENCES

David Baraff and Andrew Witkin. 1998. Large Steps in Cloth Simulation. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '98)*. ACM, New York, NY, USA, 43–54. https://doi.org/10.1145/280814.280821

David E. Breen, Donald H. House, and Michael J. Wozny. 1994. Predicting the Drape of Woven Cloth Using Interacting Particles. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '94)*. ACM, New York, NY, USA, 365–372. https://doi.org/10.1145/192161.192259

Keyser J. House, D. 2017. Foundations of Physically Based Modeling and Animation. New York: A K Peters/CRC Press, https://doi.org/10.1201/9781315373140. (2017).

Xavier Provot. 1996. Deformation Constraints in a Mass-Spring Model to Describe Rigid Cloth Behavior. In *In Graphics Interface*. 147–154.