Stability Analysis of Filtered Mass-Spring Systems

Alexandre Ahmad, Samir Adly, Olivier Terraz, Djamchid Ghazanfarpour

Stability Analysis of Filtered Mass-Spring Systems

Alexandre Ahmad and Samir Adly and Olivier Terraz and Djamchid Ghazanfarpour

University of Limoges, XLIM UMR CNRS 6172

Abstract

Mass-spring systems simulations rely on the numerical integration method used for solving the resulting ordinary differential equations. Implicit schemes, which solve such equations, are unconditionally stable and are thus widely used. Part of this stability is due to force filtering which is inherent to the implicit formulation and is referred to artificial damping. We extract this artificial damping and we analyse frequencies. This analysis enables us to define a non arbitrary damping value and a stability criterion in accordance to filtering. This directly comes from a decrease of velocity vectors' eigenvalues resulting in an increase of the time step in the same proportion. Moreover we applied a simple filtering model reproducing artificial damping to explicit schemes and results reveal an increase of the time step. Implementation of this method is straightforward for existing physically based simulators. Applications to cloth and fish animations are presented.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism–Animation G.1.7 [Numerical Analysis]: Ordinary Differential Equations

1. Introduction

Physically based animations are widely used nowadays in the field of computer graphics. Adding principles of physics to control motion not only helps artists in their tasks, but also is a challenging problem for scientists. When dealing with mass-spring systems, an important topic concerns the displacement, i.e how does a particle move from one position to another. According to Newton's second law of motion, the system of equations is transformed into a system of ordinary differential equations (ODE).

The numerical integration of ODEs is not an easy task. Explicit methods are the simplest ones for solving these equations, but they exhibit poor stability properties: the time step has to be small enough to ensure convergence, i.e. many iterations have to be computed per frame. This makes the explicit scheme often useless in practise. Because of its unconditional stability, implicit formulation is more efficient: large steps can be taken. Even though using this method linear systems have to be solved (which is computationally expensive), it still computes stiff issues much faster than explicit schemes. Part of this stability relies on filtering which is inherent to the implicit formulation and is often referred to artificial damping [Kas95, MDDB01].

When a spring is compressed or stretched, it produces counter forces in order to get back to its initial length. This leads to an oscillation phase before stabilisation, which in the case of cloth simulation produces unrealistically bouncy behaviours. Damping attenuates oscillations, in other words it reduces the bounciness. As it is well-known in the computer graphics community, damping increases arbitrarily the stability of the numerical integration. After analysis of implicit filtering, i.e. artificial damping, we show how this damping increases the stability of the numerical integration. We then define a simple filtering model for explicit schemes and a stability criterion: according to the filter's value, maximal time step for a stable numerical integration can be computed.

We applied the proposed filter to the most used explicit methods and results reveal a computational time acceleration of 20% in average. Since our damping proposal is achieved in a post-treatment, then the numerical integration scheme used is independent. Experiments show that for practical stiff examples, computational time comparisons are close when using our approach and an implicit method. We also present results on irregular meshes with varying mass and spring coefficients.

This paper is organised as follows: an overview of previ-

ous work is presented in section 2. Section 3 describes the problem of force propagation. We then present our filtering proposal in section 4 and frequency analysis is made in section 5. Section 6 discusses the computation of our stability criterion. Time comparisons of explicit filtered/unfiltered and implicit schemes are made in section 7. Finally, conclusions and future work are discussed in section 8.

2. Previous work

Computing animations when dealing with mass-spring structures rely on the important choice of the numerical method. An overview of integration schemes applied to computer graphics can be found in [VMT01, HES03]. Stability and accuracy are the two main criteria for choosing an integration method. As denoted in [NMK*05], in the field of physically based animations, stability is often much more important than accuracy. We chose to focus on stability.

The two families of integration methods, i.e. explicit and implicit, can be divided into two categories: multisteps and unisteps methods. Multisteps methods make use of two or more previous output values to compute the next one. They are not well suited for animations since previous values are no longer consistent in the case of user interactions or collisions. Most methods used in computer graphics are then unistep models (notably excepting BDF2 -implicit multistep- see [HE01, CK02]).

[BW98] showed the power of implicit modelling for cloth simulation (although similar schemes were already used for deformable models in [Fle87, TF88]). Since then, improvements of this stable model focused mostly on computational optimisations. To accelerate the computation of the linear system's solution, [VT00] proposed to exploit matrix sparsity by using a matrix-free data structure. [MDDB01] suggested to decompose forces into linear and non-linear terms and then to precompute the linear system's inverse matrix, in order to achieve real-time performance. [CK02] pointed out another source of instability: buckling. Applied to cloth animation, the authors proposed a model simulating this phenomenon which shows stable animations and nice buckling effects. Since then, buckling instability and realistic wrinkles has been a major focus [VMT06, TWS06].

Looking at the explicit category, forward Euler has been used to reproduce the motion of snakes and worms in [Mil88]. Runge-Kutta models (of order 2 or more) make use of intermediate step values, which leads to better accuracy and stability. The Verlet integration method is probably the most used explicit method in the field of computer graphics (see [BFA02, KANB03]). The use of central differencing makes this model more stable while preserving the simplicity of forward Euler. Recently, emphasis is given to a mix of the two categories, IMplicit/EXplicit, i.e. IMEX. [TT94] solved the linear system explicitly in space and implicitly in time. [EEH00, BMF03] split forces into linear and non-linear terms, respectively solved implicitly and explicitly. The work described in [BA04] uses a criterion to determine explicit instability. When detected, the system is solved implicitly. To accelerate solving, the authors use graph decomposition to subdivide the linear system's matrix into smaller matrices, resulting in a faster computation.

Artificial Damping has been analysed in [Kas95, MDDB01]. Most models using explicit schemes make use of incorrect spring damping. As denoted in [BA04], projected damping is a correct model while incorrect spring damping damps rigid body rotation. But because it is simple and often deliver the desire effect, incorrect spring damping is generally used.

Despite the fact that emphasis is made on implicit/IMEX approaches, explicit methods are still subject to active research. [KANB03] explain their advantages for film production. The main drawback of explicit schemes is that stiff problems lead to instabilities. [Shi05] proposed to linearize the forces in order to stabilise explicit methods. The force matrix is evaluated a few times per frame at regular intervals or when the system diverges. Even though this method shows competitive results, it is unclear how the divergence is detected. With the same interest to enlarge the stability domain, we propose a frequency analysis of cloth animations computed by explicit and implicit schemes and conclude that high frequencies of motion lead to explicit instability. To postpone this instability, we add artificial damping, i.e. a low pass filter, as in the implicit (stable) scheme. Thus instability is postponed and detected *because* high frequencies are removed. Experiments highlighted that examples having important eigenvalues (10^6) computed explicitly (filtered) and implicitly revealed similar computational times. Obviously, for really high stiffness (10^9), implicit solving is much faster, since filtering is not the only source of the problem.

3. The Force Problem

This section shows the force propagation problem encountered by explicit ODEs when applied to mass-spring systems (see also [Kas95,MDDB01]). Figure 1 illustrates the 1D case for a rope.

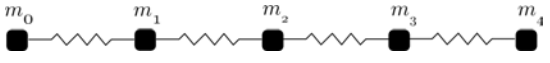

Figure 1: *A 1D case of a mass-spring system.*

A set of evenly spaced particles $\{m_0,\ldots,m_4\}$, with the same mass m are interconnected by springs with stiffness $km > 0$. At time t_0 no forces are acting on particles, springs are at their rest lengths, i.e. the system is in a static equilibrium. In the case of a small stiffness km, if a force acts over m_0 and pulls it to the left, then m_0 should move freely without strongly affecting the whole structure. But if km is big, then the entire structure should move instantly, due to the propagation of forces.

3.1. ODE System

Particle-based mechanics can be formulated as a differential equation system of the form:

$$\mathbf{M}\ddot{x} = f(x,\dot{x}) \tag{1}$$

where $x,\dot{x},\ddot{x}$ represent respectively the position, velocity and acceleration vectors, of size $3n$ (working in 3D space), n being the number of particles, f is the force vector and $\mathbf{M}$ is a diagonal mass matrix of size $3n \times 3n$, i.e. $diag(\mathbf{M}) = m_1,m_1,m_1,m_2,m_2,m_2,\ldots$. Defining $v = \dot{x}$, equation 1 is rewritten as:

$$\frac{d}{dt}\begin{pmatrix} x \\ v \end{pmatrix} = \begin{pmatrix} v \\ \mathbf{M}^{-1}f(x,v) \end{pmatrix} \tag{2}$$

Next section analyses the behaviour of explicit and implicit integration formulations applied to equation 2.

3.2. Explicit Integration and Problem

Forward Euler formula is of the form:

$$\begin{pmatrix} x^{t+\Delta t} \\ v^{t+\Delta t} \end{pmatrix} = \begin{pmatrix} x^t \\ v^t \end{pmatrix} + \Delta t \begin{pmatrix} v^t \\ \mathbf{M}^{-1}f(x^t,v^t) \end{pmatrix} \tag{3}$$

Knowing initial conditions equation 3 can be solved iteratively. Spring forces act only on the two end connected particles. One can see that v^t (the simple extension of forward Euler known as forward-backward (FB) Euler uses $v^{t+\Delta t}$) is used to update $x^{t+\Delta t}$, not x^t. Consequently a displacement of particle m_0 will affect particle m_1 after 2 iterations. Forces acting on m_0 are consequently propagated to m_4 after 8 iterations. The resulting effect is that particles have a certain freedom of motion, without immediately affecting the whole structure, leading to local variations. In the real world, propagation of forces is instantaneous. This is not the case with forward Euler, unless computing $2(n-1)^2$ iterations per step (which makes it impractical). This drawback of explicit schemes can be solved using an implicit method.

3.3. The Implicit Explanation

The most known implicit integration scheme is to be considered as backward Euler, and is expressed as follows:

$$\begin{pmatrix} x^{t+\Delta t} \\ v^{t+\Delta t} \end{pmatrix} = \begin{pmatrix} x^t \\ v^t \end{pmatrix} + \Delta t \begin{pmatrix} v^{t+\Delta t} \\ \mathbf{M}^{-1}f(x^{t+\Delta t},v^{t+\Delta t}) \end{pmatrix} \tag{4}$$

In this case $f(x^{t+\Delta t},v^{t+\Delta t})$ has to be approximated using linearization for example (see [BW98]). It is important to note that the second row of equation 4 can be rewritten into the following linear system:

$$\mathbf{A}v = b \tag{5}$$

where $\mathbf{A}$ is a the effective system matrix, usually sparse, symmetric and positive definite (or transformed to be), v,b are respectively the velocity and effective load vectors. Different methods for solving linear systems can be found in the literature [PTVF92]. Just to name a few, the Cholesky factorisation and the conjugate gradient are the most used algorithms in computer animation, with a large preference for the last one thanks to its speed. Considering the Cholesky factorisation, the $\mathbf{A}$ matrix is decomposed into two triangular matrices, a lower one $\mathbf{L}$ and its transpose $\mathbf{L}^\mathbf{T}$ (see [PTVF92] for details):

$$\mathbf{A} = \mathbf{L}\mathbf{L}^\mathbf{T} \tag{6}$$

If we consider the example shown in figure 1 then $\mathbf{A}$ and $\mathbf{L}$ have the following shape:

$$\mathbf{A} = \begin{pmatrix} a_{0,0} & a_{1,0} & & & \mathbf{0} \\ a_{1,0} & a_{1,1} & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & a_{3,3} & a_{3,4} \\ \mathbf{0} & & & a_{4,3} & a_{4,4} \end{pmatrix} \quad \mathbf{L} = \begin{pmatrix} l_{0,0} & & & & \mathbf{0} \\ l_{1,0} & l_{1,1} & & & \\ & l_{2,1} & \ddots & & \\ & & \ddots & l_{3,3} & \\ \mathbf{0} & & & l_{4,3} & l_{4,4} \end{pmatrix}$$

We obtain the solution v by using a classical forward (equation 7) and backward (equation 8) substitution:

$$\mathbf{L}y = b \tag{7}$$

$$\mathbf{L}^\mathbf{T}v = y \tag{8}$$

Analysis of this computation will help us to understand how an implicit method propagate forces. Considering equation 7 applied to our example, the first component is trivially $y_0 = b_0/l_{0,0}$. Other components are calculated with the recurrence $y_i = \left(b_i - y_{i-1}l_{i,i-1}\right)/l_{i,i}$. As denoted in [Kas95], the case where $l_{i,i-1} = 1/a$ and $l_{i,i} = (a-1)/a$ is a representative behaviour away from the boundaries. The recurrence can then be rewritten into:

$$y_i = \lambda b_i + (1-\lambda)y_{i-1} \tag{9}$$

where $\lambda = a(a-1)^{-1}$. Equation 9 is a simple recursive filter. The output y_i is a blend between the previous output y_{i-1} and the current input b_i. In fact this is a low-pass filtering of smoothing size determined by λ. If neighbour particles in the range of λ have similar force variations, which is typically the case when using small time steps, then filtering will have almost no effect. On the contrary, filtering takes all of its meaning when forces show different behaviours.

Low-pass filtering occurs when solving implicit formulation. In other words, particles propagate forces to their neighbours, and during one time step all the neighbours in the range of λ take into account this propagation. The larger the time step is, the more important the smoothing range will be and consequently the low-pass filtering. If the time step is small, then filtering is not operating and no artificial damping is added.

4. Explicit Application Proposal

In section 3.3 we have explained how implicit approaches apply low-pass filtering to force. This way, local variations i.e. high frequencies, are attenuated, and hence stability is

improved. As a solution to force propagation, we propose to diffuse velocity at each time step to particles' neighbours. Even though force filtering has a stronger physical meaning, we obtained similar results when filtering force or velocity. This is not surprising since velocity is computed directly from force. Filtering velocity allows the model's integration to be used as a post-processing unit without modifying the existing numerical scheme which can then be seen as a black box.

4.1. Our Filtering/Damping Proposition

Equation 9 shows an exponential filter generated by implicit solving. Since our goal is to represent the same behaviour, we applied an identical filtering. Indeed, other filters may be used. We experimented a Gaussian one, and observed similar results (larger time steps of the same proportion, motion damping is visually comparable). Exponential filter is defined by:

$$\beta(d) = \lambda e^{-\lambda|d|} \tag{10}$$

where d is the algebraic distance between two particles. We can observe in figure 2 that λ is responsible for filtering adjustment. Hence one can use adaptive filtering through the variation of λ. Attention should be given to very low-pass filtering ($\lambda < 1$), which leads to undesirable effects: motion is then too much damped. In this case current particle's velocity has almost as much influence as the particle's neighbours velocities, so there are no more local variations and there only remains a global motion, i.e. the whole structure is moving. Considering the velocity of the ith particle $v_i^{t+\Delta t}$,

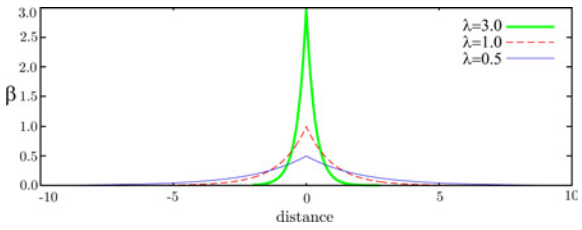

Figure 2: *Exponential filter.*

the new filtered velocity is computed as follows:

$$v_{i_{filtered}}^{t+\Delta t} = \sum_j \beta_j^{t+\Delta t} v_j^{t+\Delta t}$$

where $\beta_j^{t+\Delta t}$ correspond to the filter value of neighbouring particle j (see equation 10). Here we consider that particle i is a neighbour of itself. We point out that $\beta_j^{t+\Delta t}$ is time dependent and has to be normalised. Also, the exchange between two particles is not symmetric, and can produce undesirable effects. Thus we propose the following alternative which worked well with our examples:

$$v_{i_{filtered}}^{t+\Delta t} = \sum_j \beta_j^{t+\Delta t} \frac{v_j^{t+\Delta t} + v_i^{t+\Delta t}}{2}$$

Our algorithm is quite similar to the one proposed in [MDDB01]. Nevertheless, there are notable differences. Meyer *et al.* aim implicit optimisation by precomputing the inverse of a modified linear system's matrix **A** and using it as a force filtering. We do not have the same goal. We aim explicit stabilisation by defining a generic filter for explicit schemes in order to reduce the time step. To do so, we analyse implicit solving and extract the generated filter. Moreover, we present a frequency analysis.

4.2. Structure

We applied filtering on different mass-spring structures, i.e. clothes (we used the model proposed in [CK02]) and three dimensional objects with varying material properties such as the fish model presented in [TT94].

4.3. Implementation Discussion

Implementation of our method is straightforward for existing ODE solvers. Filtering operates as a post-processing unit. Our approach has proved to be working with all tested explicit methods: forward Euler, FB Euler, Verlet scheme and Runge-Kutta 4 (RK4). In our example we simulated fluid friction force, spring stiff force as in [TT94]. Filtering can be applied on a mass-spring structure with damping forces, although not necessary. In this case we recommend the projected damping model. We point out that our filtering schemes eliminates high frequencies, which occur in the in-plane (along the spring) and also in the out-plane. Projected damping attenuates spring oscillations in the in-plane, thus a double damping is done in the in-plane. This is not a problem since spring oscillations are often undesired.

5. Filtering Analysis

In this section we analyse the frequencies of motion generated by an explicit, an implicit and a filtered explicit scheme. We expect animations computed by a low damped explicit method to be fulfilled of details, since it allows local variations, while motion computed by implicit or filtered explicit methods is supposed to be damped. We made the following experiment: a 2D cloth mesh nailed at two corners is falling under gravity force during 5 seconds (see bottom figure 4(d)). Three simulations are being tested with respectively the following numerical methods: FB Euler, backward Euler and filtered FB Euler. Figure 3 shows our 2D to 1D transformation on a close-up pattern of the mesh (which is repeated on a plane). To analyse changes in velocities we will represent velocity field in RGB space instead of 3D world space.

Figure 4 shows velocity evolution throughout the experiment of the animation (no adaptive time stepping is used, all coefficients for the mass-spring structure are identical for all three simulations). Horizontal axis represents time with

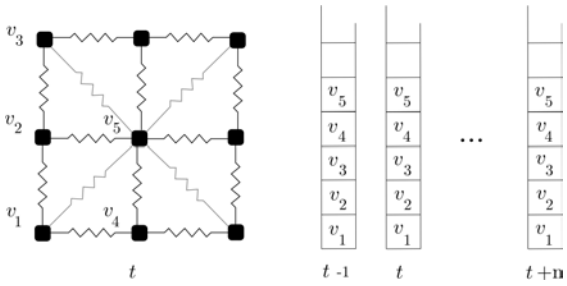

Figure 3: *At each time step, velocities are stored in one dimension vector for evolution analysis.*

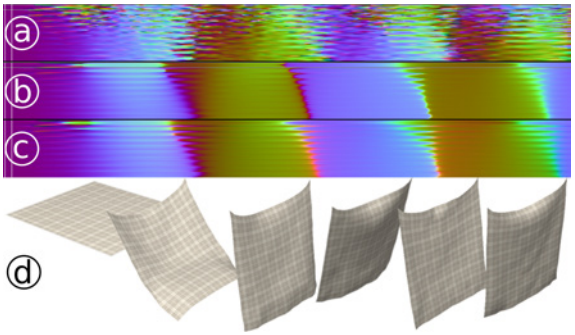

Figure 4: *Evolution of cloth particles' velocities computed by* (a) *FB Euler method* (b) *backward Euler (see [BW98])* (c) *FB Euler using our proposed filter (filtered FB Euler)* (d) *screenshots illustrate the 3D cloth experiment at corresponding times with the top images.*

a time step of 0.01s. We can clearly see that similar areas of same colours exist in all three pictures, and occurring almost at the same time. But one can distinguish disturbance on top image *(a)*, which is computed by FB Euler. They are due to local variations, i.e. high frequencies. As expected, the middle picture *(b)*, solved by backward Euler, contains few perturbations, since they are automatically filtered. We solved the linear system using Cholesky decomposition and the conjugate gradient method. Both computations generated filtering. The shape of the bottom image *(c)*, obtained using filtered FB Euler, is visually similar to the middle picture, i.e. the artificial damping is reproduced.

To visualise the frequency range, we performed a 2D Fourier transform on the whole rectangular mesh velocities (with the same cloth experiment). A screenshot of the resulting animations is shown in figure 5. Looking at the bottom row of figure 5, i.e. the frequency domain, the centre value (white pixel) corresponds to very low frequencies. The farther we go from the centre, the higher the frequencies are. Again, the predicted behaviours are illustrated: explicit FB Euler computation (left) contains higher frequencies than the two others. Lower frequencies (closer to the centre) are seen

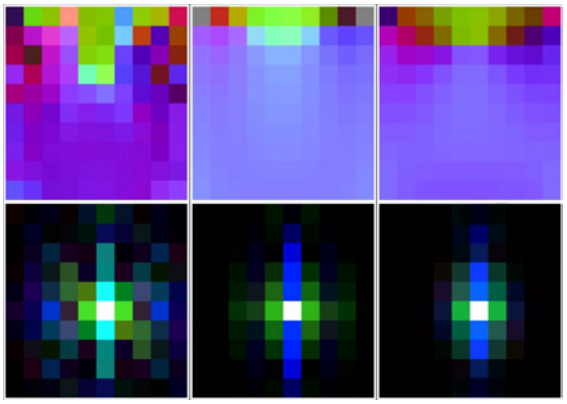

Figure 5: *Top row: 2D cloth structure projected as an image. Bottom row: their corresponding frequency domain. From left to right: forward, backward and filtered forward Euler.*

with the implicit scheme (middle). Almost identical results are achieved by filtering FB Euler scheme (right).

6. Stability Criterion

To determine the time step limit when adding artificial damping, we will use an eigenvalue analysis. For forward Euler, FB Euler and the Verlet scheme, maximum time step can be computed as follows:

$$\Delta t_{max} = \frac{2}{\sqrt{k_0}} \tag{11}$$

where k_0 denotes maximum eigenvalue computed from the stiffness matrix (see [HES03, Shi05]). A simple but efficient method for maximum eigenvalue computing is the power method. We modified this algorithm for our needs: we integrated filtering (see algorithm 1). Figure 6 shows the time step size related to a scalar λ. If λ is high (i.e. >10), the filter has almost no effect. On the contrary if λ is small (i.e. ≤ 1), the low pass filtering is very effective.

Similarly to implicit solving, the larger the time step is, the more important the filtering will be. When $\lambda = 1$, time step is doubled. This leads to an almost twice faster computation than when using unfiltered methods. Since time step is doubled, k_0 is consequently divided by 4 (see equation 11). If spring and mass coefficients are not modified throughout the animation (usually the case), then k_0 is of the same order of magnitude. This is consequently true for time steps. In fact this algorithm is executed once and for all, as a precomputation and require a few iterations for convergence.

7. Results

For all of our experiments, our test platform was an AMD 2800+ running Linux.

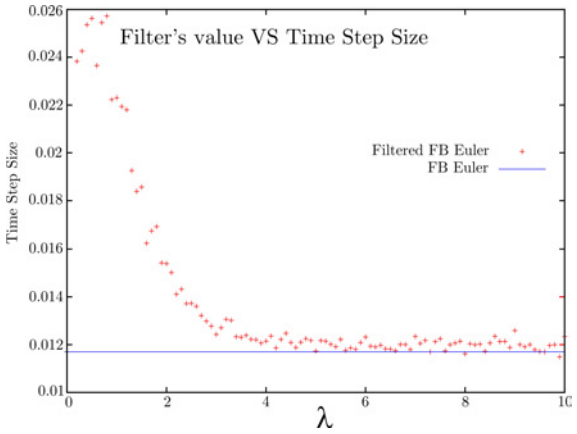

Figure 6: *Time step size is determined by the filtering coefficient* λ.

Algorithm 1 Modified Power Method.

Require: vector pos, lambda
1: v initial guess (random)
2: **for** $k = 1, 2, 3, \ldots$ until convergence **do**
3: $v \leftarrow A \cdot v$
4: $\alpha \leftarrow max(v)$
5: $v \leftarrow v/\alpha$
6: $v \leftarrow filter(pos, v, \lambda)$
7: **end for**
8: **return** α

7.1. Filter Analysis Discussion

Some of our examples worked well with a twice bigger time step which reduces by almost 50% time computations. But some examples did not because the force propagation is not the only source of instability. We point out that filtering attenuates velocity, particularly when numerical instability is detected: if a particle's velocity goes unstable, then it can be stabilised according to its neighbours' velocities (depending on the instability). If all neighbours' velocities go unstable, then filtering cannot attenuate velocities anymore, and the system is hence unstable. This is a typical case of stiff systems. All of our examples accepted a 1.25 larger time step. Using such a time step value without filtering leads to numerical instability. It is thus stable when filtering is applied. Since filtering is in $\mathcal{O}(n)$ (the number of neighbours is constant and is usually much smaller than the number of particles n), we experienced a 20% faster computation time (for a 25% larger time step, which means that filtering has a 5% cost).

7.2. Time Comparisons

To illustrate our model's efficiency, we tested the variation of the following parameters (with the experiment shown in figure 7 for a 10 seconds animation): k_0 and particle number (since all numerical methods are proportional to these two parameters). We tested explicit methods (FB Euler, RK4 and Verlet methods), an implicit one, backward Euler as in [BW98] and the IMEX method proposed in [EEH00]. We used the matrix-free data structure (see [VT00]) for both implicit and IMEX methods, in order to set up fair comparisons. We used a fixed time step (0.01s) for both the IMEX and the implicit scheme. For explicit methods, we precomputed the maximum time step (although ≤ 0.01) using equation 11 for FB Euler and Verlet method and used for RK4 a maximum step size determined by $\sqrt{8.75}/\sqrt{k_0}$ (see [Shi05]). We disabled collision detection for fair comparisons, since collisions are handled differently when using explicit or implicit methods. Top figure 8 shows time comparisons when increasing the number of particles (with $k_0 = 10^6$).

Results show a time acceleration of about 20% when filtering explicit methods (using a 1.25 larger time step size). The Verlet scheme is not shown for clarity but is subject to an equivalent speed up. We can see that the filtered FB Euler curve (see the top of figure 8) is very close to the backward Euler one. It is important to note that the conjugate gradient algorithm (where most of implicit solving time is spent) highly depends on the convergence criteria (error threshold ε and maximum number of iterations). In our case we used a relatively big $\varepsilon = 0.01$ (stable though), but when decreasing it, implicit solving takes more time for this same eigenvalue (filtered FB Euler solving is then faster).

The bottom figure 8 shows time comparison when increasing k_0 (due to stiffness). Implicit and IMEX methods become significantly more efficient when $k_0 > 10^6$, although with such stiffness, animations show a solid-like behaviour. In this case, explicit methods are to be prohibited.

We applied our method to the fish structure (see figure 9), and as for the cloth experiments, bigger time steps and faster computations were in the same proportion. The simulations computed with the three methods, i.e. explicit, implicit and filtered explicit are visually similar.

8. Conclusions

Explicit methods exhibit interesting properties, such as simplicity and accuracy. Unfortunately they are unstable and require small time steps. For stiff systems, this leads to high computational times. We showed how implicit schemes generate low-pass filtering of forces, and thus avoiding some instabilities. Taking advantage of this, we applied a similar filtering to explicit schemes. Frequency analysis and results show that time steps can be increased from 25% to 200%, and consequently computational times are decreased from 20% to almost 50%. The drawback is that, as in the implicit case, artificial damping is added and hence animations look smoother (depending on the filter value). For practical exam-

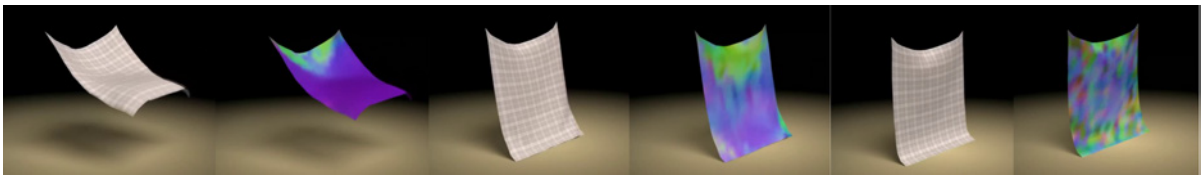

Figure 7: *Animation of a fallen drape. Every two images: a textured mesh and its corresponding velocity coloured mesh.*

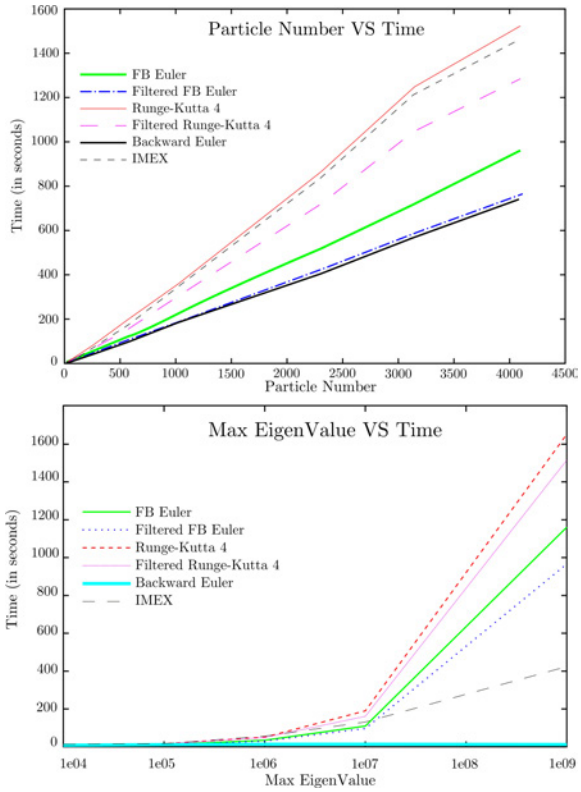

Figure 8: *Top: Computational times compared to the particle number for $k_0 = 10^6$. Bottom: Explicit methods have competitive computation times until eigenvalue reaches 10^6.*

ples ($k_0 \leq 10^6$) our method shows competitive results compared to an implicit solving. But explicit methods dealing with higher stiffness are much slower. To face this problem, we look forward to apply implicit stabilisation procedures to explicit methods, such as the linearization. We also plan to improve filtering by using more powerful filters, such as the Kalman filter, recently used in similar work [GM06]. A Von Neumann analysis could also be used to quantify frequencies and to keep only the desired ones. For better time performance, it could be possible to run our filtering algorithm on the GPU.

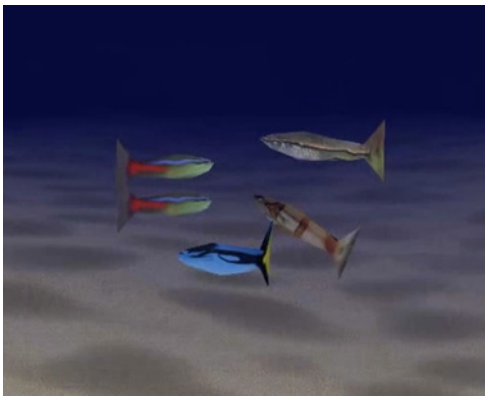

Figure 9: *Screenshot of our method applied to fish animation.*

References

[BA04] BOXERMAN E., ASCHER U.: Decomposing cloth. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation* (New York, NY, USA, 2004), ACM Press, pp. 153–161.

[BFA02] BRIDSON R., FEDKIW R., ANDERSON J.: Robust treatment of collisions, contact and friction for cloth animation. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2002), ACM Press, pp. 594–603.

[BMF03] BRIDSON R., MARINO S., FEDKIW R.: Simulation of clothing with folds and wrinkles. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer animation* (Aire-la-Ville, Switzerland, 2003), Eurographics Association, pp. 28–36.

[BW98] BARAFF D., WITKIN A.: Large steps in cloth simulation. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1998), ACM Press, pp. 43–54.

[CK02] CHOI K.-J., KO H.-S.: Stable but responsive cloth. In *SIGGRAPH '02: Proceedings of the 29th annual*

conference on Computer graphics and interactive techniques (New York, NY, USA, 2002), ACM Press, pp. 604–611.

[EEH00] EBERHARDT B., ETZMUSS O., HAUTH M.: Implicit-explicit schemes for fast animation with particle systems. In *Eurographics Computer Animation and Simulation Workshop 2000* (2000).

[Fle87] FLEISHER D. T. . J. P. . A. B. . K.: Elastically deformable models. *Computer Graphics 21*, 4 (July 1987), 205–214.

[GM06] GROTE M. J., MAJDA A. J.: Stable time filtering of strongly unstable spatially extended systems. *Proceedings of the National Academy of Sciences of the United States of America 103*, 20 (2006), 7548–7553.

[HE01] HAUTH M., ETZMUSS O.: A high performance solver for the animation of deformable objects using advanced numerical methods. In *Proc. Eurographics 2001* (2001), Chalmers A., Rhyne T.-M., (Eds.), vol. 20(3) of *Computer Graphics Forum*, pp. 319–328.

[HES03] HAUTH M., ETZMUSS O., STRASSER W.: Analysis of numerical methods for the simulation of deformable models. *The Visual Computer 19*, 7-8 (2003), 581–600.

[KANB03] KAČIĆ-ALESIĆ Z., NORDENSTAM M., BULLOCK D.: A practical dynamics system. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, 2003), Eurographics Association, pp. 7–16.

[Kas95] KASS M.: An introduction to physically based modeling, chapter introduction to continuum dynamics for computer graphics. In *SIGGRAPH Course Notes* (1995).

[MDDB01] MEYER M., DEBUNNE G., DESBRUN M., BARR A. H.: Interactive animation of cloth-like objects in virtual reality. *Journal of Visualization and Computer Animation 12*, 1 (2001), 1–12.

[Mil88] MILLER G. S. P.: The motion dynamics of snakes and worms. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1988), ACM Press, pp. 169–173.

[NMK*05] NEALEN A., MÜLLER M., KEISER R., BOXERMANN E., CARLSON M.: Physically based deformable models in computer graphics. In *Proceedings of Eurographics 2005* (2005), pp. 91–94.

[PTVF92] PRESS W. H., TEUKOLSKY S. A., VETTERLING W. T., FLANNERY B. P.: *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, New York, NY, USA, 1992.

[Shi05] SHINYA M.: Theories for mass-spring simulation in computer graphics: Stability, costs and improvements. *IEICE - Trans. Inf. Syst. E88-D*, 4 (2005), 767–774.

[TF88] TERZOPOULOS D., FLEISCHER K.: Modeling inelastic deformation: viscolelasticity, plasticity, fracture. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1988), ACM Press, pp. 269–278.

[TT94] TU X., TERZOPOULOS D.: Artificial fishes: physics, locomotion, perception, behavior. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1994), ACM Press, pp. 43–50.

[TWS06] THOMASZEWSKI B., WACKER M., STRASSER W.: A consistent bending model for cloth simulation with corotational subdivision finite elements. In *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, 2006), Eurographics Association, pp. 107–116.

[VMT01] VOLINO P., MAGNENAT-THALMANN N.: Comparing efficiency of integration methods for cloth simulation. In *Computer Graphics International* (2001), pp. 265–274.

[VMT06] VOLINO P., MAGNENAT-THALMANN N.: Simple linear bending stiffness in particle systems. In *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, 2006), Eurographics Association, pp. 101–105.

[VT00] VOLINO P., THALMANN N. M.: Implementing fast cloth simulation with collision response. In *CGI '00: Proceedings of the International Conference on Computer Graphics* (Washington, DC, USA, 2000), IEEE Computer Society, p. 257.