# Explicit but Stable Spring-Damper Model with Harmonic Oscillation

**Conference Paper** · July 2017

CITATIONS
0

READS
8

**1 author:**

Young-Min Kang
Tongmyong University
**39** PUBLICATIONS **237** CITATIONS

SEE PROFILE

# Explicit but Stable Spring-Damper Model with Harmonic Oscillation

Young-Min Kang
Dept. of Game Engineering
Tongmyong University, Busan, KOREA
ymkang@tu.ac.kr

## ABSTRACT

The mass-spring model is the most commonly used model for the animation of cloth and the physical simulation of various deformable objects. Although the method is simple, this model has, however, serious problems. The spring model tends to be numerically unstable. In other words, when you try to simulate a spring that does not stretch well, you will have to use very small time steps or to solve very large linear systems generated by implicit integration. In this paper, we propose a new simulation model that improves numerical stability and physical reality in order to perform large scale mass-spring model. The method can be easily parallelized. We implemented the method in CUDA environments which can be used on general personal computer. The proposed method is based on a harmonic oscillation model.

## KEYWORDS

Mass-Spring, Stability, Harmonic Oscillation

## 1 INTRODUCTION

This paper proposes a model with improved numerical stability for efficient simulation of a mass-spring model, which is frequently used to perform physical simulations of deformable objects. The method of this paper reduces the numerical instability of the Euler method by applying method for better integration of spring force based on harmonic oscillator model. In general, the problem of numerical instability is resolved by applying implicit integration, but this method additionally causes a problem of solving a large-scale linear system. Especially, when the number of mass points increases, the number of components constituting the matrix of the linear system is increased to the square of the number of particle points. Therefore, the mass-spring model cannot be easily simulated in parallel fashion In this paper, a simulation method that can be easily implemented in CUDA environment was proposed. The CUDA environment is becoming more and more common in personal computer environments. Because of the parallelism and the numerical stability, the simulation method can efficiently produce plausible motions in real-time.

## 2 RELATED WORK

The mass spring model is a model that has been actively studied in the field of cloth animation since it was first formulated as a deformable object [11]. Among the various studies to solve the difficulties in real-time animation of deformable object model[12, 9, 10], the most important break-through was the implicit integration approach. Applying this implicit integration method to cloth animation, a linear system solution becomes a problem [1, 2]. The problem is that this linear system is a sparse matrix, but it has to deal with a very large matrix. A Several efficient schemes have been proposed to reduce the computational burden of the linear system solution including these matrices and use them for real-time cloth animation [8, 7, 4]. However, these techniques sacrifice accuracy to obtain real-time performance It was difficult for a complex model with a large number of particle points. A real-time animation technique of a character model fully dressed by Cordier has been proposed [3]. However, this technique did not improve the performance of the physics simulation it-

self. In this paper, we propose a method that can obtain stable results and calculate the state change of each material point in a form suitable for parallel processing.

## 3 PROBLEM

The simplest numerical integration technique is an explicit Euler integral. This method approximates the particle velocity of particles in the following way.

$$\frac{\mathbf{v}^{t+h} - \mathbf{v}^t}{h} \simeq \frac{d}{dt}\mathbf{v}^t = \frac{1}{m}\mathbf{f}^t \tag{1}$$

Therefore, the simulation is to calculate the velocity of the next state using the current velocity and the current force in the following manner.

$$\mathbf{v}^{t+h} = \mathbf{v}^t + \frac{h}{m}\mathbf{f}^t \tag{2}$$

This technique has serious numerical instability problems. This is because $\mathbf{f}^t h$ is not an accurate integration. To solve this problem, the implicit integration method performs the numerical integration using the force of the following state, that is, $\mathbf{f}^{t+h}$. However, the power of this future is unknown in the present state, and it is inevitable to use Taylor expansion. This approximation can be performed as follows.

$$\mathbf{f}^{t+h} = \mathbf{f}^t + \frac{\partial \mathbf{f}}{\partial \mathbf{x}}\Delta\mathbf{x} \tag{3}$$

There is another problem that $\partial\mathbf{f}/\partial\mathbf{x}$ is not a vector. When we have $n$ mass-points, $\partial\mathbf{f}/\partial\mathbf{x}$ becomes $n \times n$ matrix. Therefore, the problem is essentially a large linear system problem with $O(n^2)$ matrix.

Although implicit integration always guarantees a stable result, this large-scale linear system solution is difficult to implement and is not suitable for parallel processing. In order to solve this problem, we propose a new technique which has a greatly improved numerical stability and follows an explicit integration method which can be calculated independently for each particle. The basic idea is that the numerical instability was caused by an inaccurate numerical integral, which is to be corrected to perform a more accurate numerical integration. The accurate velocity change can be described as follows:

$$\mathbf{v}^{t_0+h} = \mathbf{v}^{t_0} + \frac{1}{m}\int_{t_0}^{t_0+h} \mathbf{f}^t dt \tag{4}$$

The problem is how to obtain this exact integral. Integrating the spring force defined in the Hooke's law results in a problem because it does not take into account the varying forces during the integration period. Therefore, we considered a model considering this. This model is based on the assumption that the movement of the spring will cause a harmonic vibration. Harmonic oscillation is described analytically and accurate integration is possible. Therefore, the problem is to obtain the value of equation 4 analytically.

To solve the problem, simple harmonic oscillation without the consideration of damping force was considered [6]. However, damping was not considered in the previous work. In this paper, we propose a new method that correctly integrates the spring forces taking the damping force into account, and the better time stepping strategy is also proposed.

The proposed method takes into consideration only the state of adjacent material points, as in the explicit integration method. This approach is well suited for parallelism. Therefore, we implemented the proposed technique in CUDA environment. When simulating particles in the GTX 590 GPU environment with

CUDA, it was possible to simulate a complex mesh of 16,384 particles with a performance of more than 66 frames per second.

## 4 HARMONIC OSCILLATION

Suppose two particles $i$ and $j$ are linked with a spring. The spring can be described as $(i, j)$, and the locations of the particles are denoted as $\mathbf{x}_i$ and $\mathbf{x}_j$ respectively. The velocities and masses are similarly described as $\mathbf{v}_i$, $\mathbf{v}_j$, $m_i$, and $m_j$.

The vector from $i$ to $j$ is $\mathbf{x}_{ij} = \mathbf{x}_j - \mathbf{x}_i$, and the relative velocity can be similarly described as $\mathbf{v}_j i = \mathbf{v}_j - \mathbf{v}_i$. Let us denote a normalized vector with $hat$ as follows:

$$\hat{\mathbf{x}}_{ji} = \frac{\mathbf{x}_{ji}}{|\mathbf{x}_{ji}|} \tag{5}$$

The location of a particle is a function of time, and usually denoted as $\mathbf{x}_i^t$. Other vectors describing the physical states of particles are similiarty expressed. The rest length of a spring is denoted $l^0$, and the length at time $t$ is $l^t$ and equals to $|\mathbf{x}_{ji}^t|$. The difference between the current length and the rest length is denoted as $\delta^t = l^t - l^0$. The stiffness of the spring is $\kappa$.

A simple example of a harmonic oscillation is the situation where a mass point in 1D space is linked to a static object with a spring. The location of the mass point can be described with a scalar value $x$, and it oscillates in accordance with the equation $x = A \sin \omega t$ where $\omega = \sqrt{\kappa/m}$.

When two mass-points are linked, we can describe the elongation of the spring as follows:

$$\delta^t = A \sin \omega t = A \sin \left( \sqrt{\frac{\kappa(m_i + m_j)}{m_i m_j}} t \right) \tag{6}$$

In other words, the spring with two mass-points $m_i$ and $m_j$ can be described as a simple harmonic oscillation where a static object holds a dynamic mass-point of $(m_i m_j)/(m_i + m_j)$. If $m_i$ approach to infinity (static), this mass expresion becomes $m_j$.

In order to determine $A$, energy conservation is applied. Total energy is the sum of kinetic energy and potential energy. The total energy of the spring is $\kappa A^2/2$. The potential energy is $\kappa \delta^{t2}/2$. The kinetic energy can be described as follows:

$$\frac{1}{2} \left( \frac{m_i m_j}{m_i + m_j} \right) \dot{\delta^t}^2 \tag{7}$$

Therefore, the energy conservation can be described as follows:

$$\kappa A^2 = \kappa \delta^{t2} + \left( \frac{m_i m_j}{m_i + m_j} \right) \dot{\delta^t}^2 \tag{8}$$

We can then find the amplitude $A$ as follows:

$$A = \sqrt{\delta^{t2} + \frac{m_i m_j}{\kappa(m_i + m_j)} \dot{\delta^t}^2} \tag{9}$$

In order to solve the above equation, the derivative of the spring deformation should be computed ad follows:

$$\frac{d}{dt}\delta^t = \frac{d}{dt}(l^t - l^0) = \frac{d}{dt} l^t \tag{10}$$

Therfore, we can compute the derivative as follows:

$$\begin{aligned} \frac{d}{dt}\delta^t &= \frac{d}{dt}\sqrt{\mathbf{x}_{ji}^{t\,\mathrm{T}} \mathbf{x}_{ji}^t} \\ &= \frac{1}{2} \mathbf{x}_{ji}^{t\,\mathrm{T}} \mathbf{x}_{ji}^{t\,-0.5} \frac{d}{dt} \mathbf{x}_{ji}^{t\,\mathrm{T}} \mathbf{x}_{ji}^t \\ &= \frac{1}{2} \mathbf{x}_{ji}^{t\,\mathrm{T}} \mathbf{x}_{ji}^{t\,-0.5} (2 \mathbf{x}_{ji}^{t\,\mathrm{T}} \mathbf{v}_{ji}^t) \end{aligned} \tag{11}$$

ISBN: 978-1-941968-42-0 ©2017 SDIWC

$$= \frac{{\mathbf{x}_{ji}^t}^{\mathrm{T}}\mathbf{v}_{ji}^t}{\sqrt{{\mathbf{x}_{ji}^t}^{\mathrm{T}}\mathbf{x}_{ji}^t}}$$
$$= {\hat{\mathbf{x}}_{ji}^t}^{\mathrm{T}}\mathbf{v}_{ji}^t$$

The amplitude $A$ can then be easily obatined.

The harmonic oscillation can be described with two time variables $t$ and $T$ as $\delta^t = A\sin\omega T$ where $t$ is the normal time in the simulation world and $T$ is the time defined in the oscillation period. Therefore, $T$ ranges from 0 to $2\pi/\omega$. In order to correctly integrate the mass-spring model based on harmonic oscillation, $T$ corresponding to the current time $t$ must be computed as follows:

$$T = \frac{1}{\omega}\sin^{-1}(\delta^t/A) \tag{12}$$

Now, the spring force along the spring $(i,j)$, $f_{ij}$ causing the harmonic oscillation can be analytically integrated as follows:

$$\begin{aligned}\int_{t_0}^{t_0+h} {f_{ij}}^t dt &= \int_{t_0}^{t_0+h} \kappa\delta^t dt \\ &= \kappa A\int_{T_0}^{T_0+h}\sin\omega T dT \\ &= -\frac{\kappa A}{\omega}\cos\omega T|_{T_0}^{T_0+h}\end{aligned} \tag{13}$$

By exploiting the integration above, we can simulate the mass-spring model more accurately. Let us define a variable $\phi$ which is the velocity change during the time interval can be obtained by dividing the integrated force with total mass.

$$\begin{aligned}\phi_{ij} &= \frac{1}{2\omega(m_i+m_j)}\int_{t_0}^{t_0+h} f_{ij}^t dt \\ &= -\frac{\kappa A[\cos\omega(T_0+h) - cos\omega T_0]}{2\omega(m_i+m_j)}\end{aligned} \tag{14}$$

The velocity change $\phi_{ij}$ is distributed to the linked mass-points in accordance with their masses. Let us denote the velocity changes magnitude for the mass-points caused by the spring $(i,j)$ as $\nu_i$ and $\nu_j$. We can easily find the following relations:

$$\begin{aligned}\nu_i + \nu_j &= \phi_{ij} \\ \nu_i m_j &= \nu_j m_i\end{aligned} \tag{15}$$

We can finaly determine the velocity change of each mass-point as follows:

$$\begin{aligned}\nu_i &= \phi_{ij} mj/(m_i+m_j) \\ \nu_j &= \phi_{ij} mi/(m_i+m_j)\end{aligned} \tag{16}$$

The velocity change of each mass-spring is actually the sum of the all the velocity changes cause by linked springs. Therefore, the velocity change of each mass-point by spring oscillation, $d\mathbf{v}_i^s$ , can be computed as follows

$$d\mathbf{v}_i^s = \sum_{(i,j)\in E}\frac{\phi_{ij} mj}{m_i+m_j}\hat{\mathbf{x}}_{ij} \tag{17}$$

where, $E$ is the set of springs.

The velocity changes computed by integrating the oscillating spring forces makes the simulation more stable. The simulation steps can be described as follows:

## 5 DAMPING

Damping is usually employed for stability and more plausible animation. The integration scheme in the previous section did not take the damping into account. However, the damping can be easily incorporated. It is well-known that the damped spring oscillates as follows:

$$\delta^t = Ae^{-\alpha t}\sin\omega_d t \tag{18}$$

**Algorithm 1:** Simulation Steps

**UpdateMass-SpringState**
**Data**: $dt$: In
**begin**
  compute $\phi_{ij}$ for every spring $(i, j)$
  **for** *all particle* $i$ **do**
    compute spring motion for particle $i$
      $d\mathbf{v}_i^s = \sum_{(i,j)\in E} \frac{\phi_{ij} mj}{m_i + m_j} \hat{\mathbf{x}}_{ij}$
    compute velocity change for particle $i$
      $\Delta\mathbf{v}_i = (d\mathbf{v}_i^s + \mathbf{f}_{external})h$
    update velocity for particle $i$
      $\mathbf{v}_i += \Delta\mathbf{v}_i$
    update location for particle $i$
      $\mathbf{x}_i += \Delta\mathbf{x}_i$

where the variable $\alpha$ is a coefficient propotional to the damping force, and $\omega_d$ is the dampened frequency of the oscillation because of the damping force. In the first subsection, $phi$ will be computed, and $\alpha$ and $\omega_d$ will be computed in the following subsection.

### 5.1 Computing $phi$ with damping

The force integration can then be rewritten with the damping as follows:

$$\kappa A \int_{t_0}^{t_0+h} e^{-\alpha t} \sin \omega_d t dt \tag{19}$$

With the assistance of calculus techniques, we can obtain the following form:

$$\kappa A \frac{e^{-\alpha t}}{\alpha^2 + \omega_d^2} (-\alpha \sin \omega_d t - \omega_d \cos \omega_d t)|_{t_0}^{t_0+h} \tag{20}$$

Therefore, the integrated force magnitude for spring $(i, j)$ can be described as follows:

$$\begin{aligned}
&\kappa A \frac{e^{-\alpha t_0} e^{-\alpha h}}{\alpha^2 + \omega_d^2} \quad [-\alpha \sin(\omega_d t_0 + \omega_d h) \\
&\qquad\qquad -\omega_d \cos(\omega_d t_0 + \omega_d h)] \\
&-\kappa A \frac{e^{-\alpha t_0}}{\alpha^2 + \omega_d^2} \quad (-\alpha \sin \omega_d t_0 - \omega_d \cos \omega_d h) \\
&= \kappa A \frac{e^{-\alpha t_0} e^{-\alpha h}}{\alpha^2 + \omega_d^2} \quad (-\alpha \sin \omega_d t_0 \cos \omega_d h \\
&\qquad\qquad -\alpha \cos \omega_d t_0 \sin \omega_d h \\
&\qquad\qquad -\omega_d \cos \omega_d t_0 \cos \omega_d h \\
&\qquad\qquad +\omega_d \sin \omega_d t_0 \sin \omega_d h)) \\
&-\kappa A \frac{e^{-\alpha t_0}}{\alpha^2 + \omega_d^2} \quad (-\alpha \sin \omega_d t_0 - \omega_d \cos \omega_d h)
\end{aligned}$$

For the simplicity, let us denote as follows:

$$\begin{aligned}
C^t &= \frac{e^{-\alpha t_0}}{\alpha^2 + \omega_d^2} \\
C^h &= \frac{e^{-\alpha h}}{\alpha^2 + \omega_d^2} \\
s_\omega^t = \sin \omega_d t_0, &\quad s_\omega^h = \sin \omega_d h \\
c_\omega^t = \cos \omega_d t_0, &\quad c_\omega^h = \cos \omega_d h
\end{aligned} \tag{22}$$

The $\phi$ for a spring $(i, j)$ can then be described as follows:

$$\begin{aligned}
\phi_{ij} = \quad & -\frac{\kappa A}{2(m_i + m_j)} C^t \\
& ( \quad s_\omega^t [C^h \alpha c_\omega^h - C^h \omega_d s_\omega^h - \alpha] \\
& + \quad c_\omega^t [C^h \alpha s_\omega^h + C^h \omega_d c_\omega^h - \omega_d])
\end{aligned} \tag{23}$$

Now we can apply the previous algorithm to obtain mass-spring motion with damping based on the $\phi$ values.

### 5.2 Determination of $\alpha$ and $\omega_d$

Simple harmonic oscillaction with damping force can be expressed as follows:

$$\ddot{\mathbf{x}} + \frac{k_d}{m} \dot{\mathbf{x}} + \frac{\kappa}{m} \mathbf{x} = 0 \tag{24}$$

where $k_d$ is a damping coefficient.

Without damping, the frequency of the oscillation $\omega$ is $\sqrt{\kappa/m}$. In order to compute the dampened frequency, let us define a substitution value $\xi$ as $d/2\sqrt{\kappa m}$. Then the equation of motion can be described as follows:

$$\ddot{\mathbf{x}} + 2\xi\omega\dot{\mathbf{x}} + \omega^2\mathbf{x} = 0 \tag{25}$$

The solution of the equation is known as follows:

$$\mathbf{x}^t = Ae^{-\xi\omega t}\cos(\omega\sqrt{1-\xi^2}t) \tag{26}$$

Here, $\xi\omega$ is $\alpha$, and $\omega\sqrt{1-\xi^2}$ is the dampened frequence $\omega_d$. As we have seen before, the frequency of oscillation by two linked mass-points is computed as $\omega = \sqrt{k(m_i+m_j)/m_i m_j}$. Therefore, we can determine the values as follows [5]:

$$\begin{aligned} \xi &= \frac{k_d}{2\sqrt{\frac{\kappa m_i m_j}{m_i+m_j}}} \qquad (27)\\ \alpha &= \xi\omega = \frac{k_d(m_i+m_j)}{2m_i m_j} \\ \omega_d &= \sqrt{\frac{\kappa(m_i+m_j)}{m_i m_j}}\sqrt{1-\frac{k_d^2(m_i+m_j)}{2\kappa m_1 m_2}} \end{aligned}$$

With these values, we can simulate the harmonic oscillation of the mass-spring-damper model.

## 6 EXPERIMENTS

The system used to implement this technique was an intel 3.47GHz i7 CPU using Microsoft's Window 7 operating system, 24G RAM, and GTX590 GPU environment. Although it is a 12-core system, it did not perform parallel processing on the CPU, and parallel processing was performed using the GPU.

In order to compare the stability of the harmonic oscillation-based integration with simple Euler integration of spring force, we used the experimental setting shown in Fig. 1. In this setting, we can control the masses of mass-points, and the stiffness of each spring.

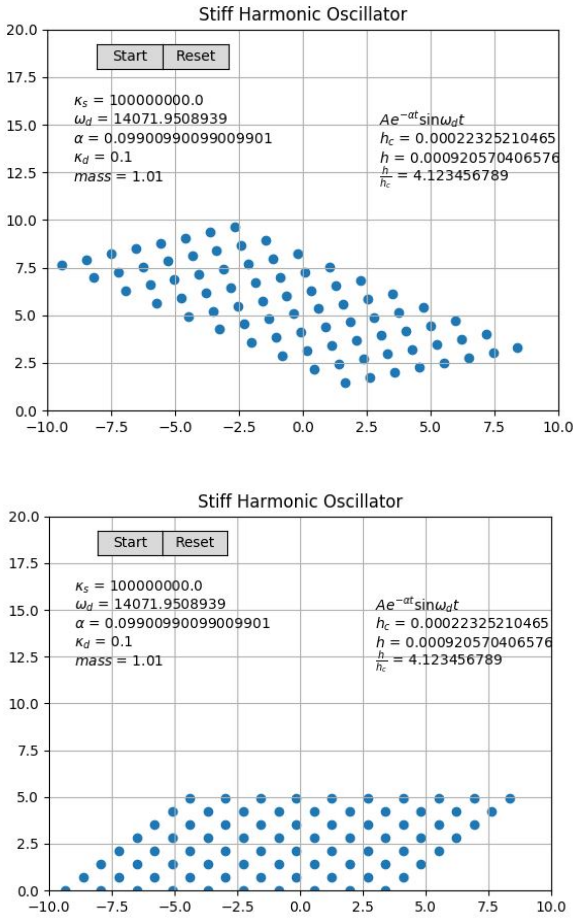



**Figure 1.** Experimental system

The experimental system integrated the spring force based on the harmonic oscillation model described in this paper, and showed more stable property compared traditional Euler method. In this simulation, we could increase the spring constant large enough to make the mass-spring model behaves almost like a rigid body as shown in Fig. 1.

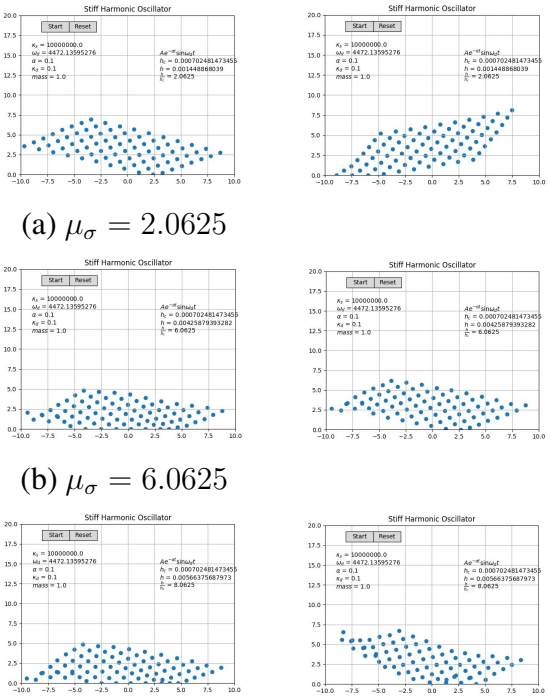
In order to understand the stability of the system, a critical time step should be defined. The motion of a spring is by its nature vibrating with the frequency of $\omega$ without damping or $\omega_d$ with damping. It is well known that we can represent the signal with a frequency $\omega$ by sampling at least twice in the cycling period, $2\pi/\omega$. Therefore, we have to reduce the time interval to be smaller than $\pi/\omega$. We can define this time

step as a critical time step $h_c$

$$h_c = \frac{\pi}{\omega} \tag{28}$$

The traditional Euler method shows instability when the time step is larger than this critical time step. The proposed method can employ time steps larger than the critical time step. The stability of the system $\mu_\sigma$can then be measured with the ratio of the possible time step size to the critical time step as follows:

$$\mu_\sigma = \frac{h}{h_c} \tag{29}$$



(a) $\mu_\sigma = 2.0625$

(b) $\mu_\sigma = 6.0625$

(c) $\mu_\sigma = 8.0625$

**Figure 2.** Stability Test

Fig. 2 shows the stability of the proposed method. We employed large time steps to simulate the mass-spring model shown in the figure. Fig. 2 (a), (b), and (c) are the result when we changed the value $\mu_\sigma$ to be $2.0625$, $6.0625$, and $8.0625$ respectively. As shown in the figure, even with the larger time steps compared with the critical time step, we could successfully simulated the model.

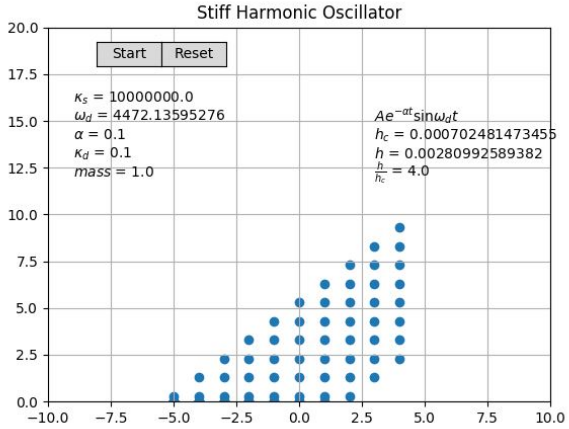



**Figure 3.** Integer mulitiplcation of $h_c$ as time step

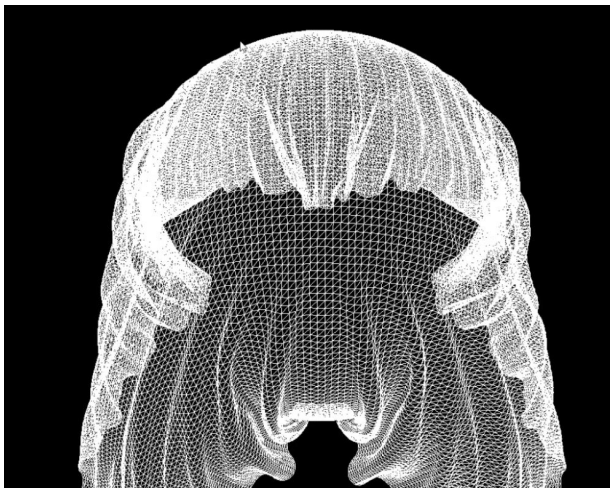
Fig. 3 shows the result when we use a time step which is an integer multiplication of critical time step. In this case, the integration of the force of spring becomes 0. Therefore, no spring motion is observed as shown in the figure. The proposed method can effectively compute the critical time step, and we can determine proper time steps based on the information.

The propose method is not only stable but also simple enough to be easily implemented in a parallel computing environments. We applied this method to a cloth animation and simulated the cloth in a parallel fashion with the assistance of GPU API. The result is shown in Fig. 4. As shown in the figure, the method can be successfully employed for massively large amount of mass-points. The number of mass-points used for this animation is 16,384.

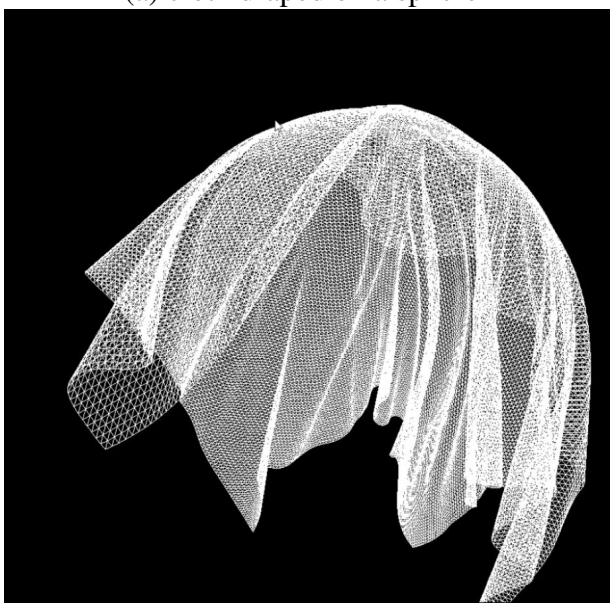
## 7 CONCLUSION

This paper proposed a simulation method of a mass-spring model with improved numerical stability based on a harmonic oscillation model. The proposed method is more stable than the simple explicit Euler method, and does not need to solve the linear system of the implicit method. These characteristics are suitable for parallel processing, and high-performance, high-quality simulation us-

ISBN: 978-1-941968-42-0 ©2017 SDIWC



(a) cloth draped on a sphere



(b) cloth sliding down on a sphere

**Figure 4.** Cloth animation with the proposed method

ing CUDA. The proposed technique can be implemented very simply in CUDA environment, and the experimental result is a technique to simulate the interactions of particles in massive mass-spring networks that were impossible in real-time applications.

## ACKNOWLEDGMENT

This work was supported in part by 2017 PLSI Program of Korea Institute of Science and Technology Information.

## REFERENCES

[1] D. Baraff and A. Witkin. Large steps in cloth simulation. *Proceedings of SIGGRAPH 98*, pages 43–54, July 1998.

[2] K.-J. Choi and H.-S. Ko. Stable but responsive cloth. *ACM Transactions on Graphics: Proceedings of SIGGRAPH 2002*, pages 604–611, 2002.

[3] F. Cordier and N. Magnenat-Thalmann. Real-time animation of dressed virtual humans. *Proceedings of Eurographics 2002*, 2002.

[4] M. Desbrun and M.-P. Gascuel. Animating soft substances with implicit surfaces. *Proceedings of SIGGRAPH 95*, pages 287–290, August 1995.

[5] D. H. House and J. C. Keyser. *Foundations of Physically Based Modeling and Animation*. CRC Press, 2016.

[6] Y.-M. Kang and C.-S. Cho. Photorealistic cloth in real-time applications. *Computer Animation and Virtual Worlds*, 23(3-4):253–265, 2012.

[7] Y.-M. Kang and H.-G. Cho. Real-time animation of complex virtual cloth with physical plausibility and numerical stability. *Presence - Teleoperators and Virtual Environments*, 13(6):668–680, 2004.

[8] Y.-M. Kang, J.-H. Choi, H.-G. Cho, and C.-J. Park. Fast and stable animation of cloth with an approximated implicit method. *Computer Graphics International 2000*, pages 247–255, 2000.

[9] M. Meyer, G. Debunne, M. Desbrun, and A. H. Bar. Interactive animation of cloth-like objects in virtual reality. *The Journal of Visulalization and Computer Animation*, 12:1–12, 2001.

[10] X. Provot. Deformation constraints in a mass-spring model to describe rigid cloth behavior. *Graphics Interface '95*, pages 147–154, May 1995.

[11] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically deformable models. *Computer Graphics (Proceedings of SIGGRAPH 87)*, 21(4):205–214, July 1987.

[12] P. Volino, N. Magnenat-Thalmann, S. Jianhua, and D. Thalmann. An evolving system for simulating clothes on virtual actors. *IEEE Computer Graphics & Applications*, 16(5):42–51, September 1996.