PAPER

Theories for Mass-Spring Simulation in Computer Graphics: Stability, Costs and Improvements

Mikio SHINYA[†a)], ***Member***

SUMMARY Spring-mass systems are widely used in computer animation to model soft objects. Although the systems can be numerically solved either by explicit methods or implicit methods, it has been difficult to obtain stable results from explicit methods. This paper describes detailed discussion on stabilizing explicit methods in spring-mass simulation. The simulation procedures are modeled as a linear digital system, and system stability is mathematically defined. This allows us to develop theories of simulation stability. The application of these theories to explicit methods allows them to become as stable as implicit methods. Furthermore, a faster explicit method is proposed. Experiments confirm the theories and demonstrate the efficiency of the proposed methods.

key words: *computer graphics, computer animation, physically-based animation*

1. Introduction

Physics-based methods have greatly enriched computer animation since they enable the realistic visual simulation of rigid bodies, flexible objects, fluids, brittle objects, and so on. Among them, spring-mass systems have played a central role in the simulation of soft objects such as cloth, where an object is modeled as a set of particles inter-connected by springs.

The spring-mass systems are described as ordinary differential equation systems, which can be numerically solved either by explicit methods or implicit methods. In earlier studies, the explicit methods were rather preferred because of its simplicity. However, avoiding numerical instability was a difficult task in the explicit methods, where the time step and the damping factors had to be carefully chosen by hand. Baraff and Witkin pointed out that the high stiffness of spring-mass systems is the major source of instability, and introduced the implicit Euler method for cloth simulation [1]. This greatly improved the stability and an efficient use of the conjugate gradient method made it practical in cloth animation. With their success, recent important work such as [2], [3] has been mainly implemented on the implicit methods, and the explicit methods seem to have been almost abandoned.

However, the explicit methods still have several advantages. Since the implicit methods need to evaluate the differentials of forces, it is difficult to handle non-analytic, procedural forces such as friction. The explicit methods, on the other hand, only need the force itself, and can easily deal with such situations. Another point is with numerical accuracy. The stable features of the implicit methods do not necessarily guarantee their accuracy, and the explicit methods can be more accurate, as demonstrated by Volino [4]. Therefore, if the explicit methods can be stably executed, they may be reevaluated as attractive alternatives to the implicit methods in spring-mass simulation.

Although numerical stabilities on ordinary differential equations have been well studied in the numeric computation field [5], [6], the results are not ready for graphics researchers to apply their particular problems such as spring-mass simulation. Recently, the author reported that exlicit spring-mass simulation can be stabilized by eigen analyses [7]. This paper gives detailed discussion of this stabilization method and proposes an improved explicit simulation method based the stability theory.

We first focus on linear features of the systems and mathematically define system stability. This results in the stability conditions for explicit and implicit methods. Based on this information, we present the maximum time step for the explicit methods and the equations needed to determine the damping coefficients. Based on these results, we develop a stable and fast explicit method, called the linearized explicit Euler method. Experiments show that this method is even faster than the implicit methods.

The stability conditions allow us to analyze the time complexity of the explicit/implicit methods, and it is shown that the complexity of both is proportional to the number of particles and the square root of the maximum eigen value of the spring coefficient matrix. The main contribution of this paper is to provide stability theories directly applicable to simulation-based computer animation, and novelty in the context of general numerical computation is not claimed.

2. Theories on Stability

Stability analysis is well established in the fields of digital control and digital filtering. In this section, spring-mass simulation is regarded as a digital linear system, which allows us to apply their well-tried formulations.

2.1 Dynamic Equations and Linearization

In general, particle-based mechanics can be formulated as a differential equation system, such as,

$$M\ddot{x}_w(t) = f(x_w(t), \dot{x}_w(t)), \qquad (1)$$

Manuscript received June 1, 2004.
Manuscript revised October 23, 2004.
[†]The author is with the Department of Information Science, Toho University, Funabashi-shi, 274–8510 Japan.
a) E-mail: shinya@is.sci.toho-u.ac.jp
DOI: 10.1093/ietisy/e88–d.4.767

where the $3n$-dimensional vectors x and f represent the position and applied force of n particles, and the $3n \times 3n$ matrix M represents the mass of each particle.

Force f, generally a non-linear function, was linearized by Baraff for easier handling. When

$$x_w = x_0 + x, \quad \dot{x}_w = v_0 + v,$$

and $|x|$ and $|v|$ are sufficiently small, we can make the following linear approximation:

$$f(x_w, v_w) \simeq f(x_0, v_0) + (\partial f/\partial x)(x_0)x + (\partial f/\partial v)(v_0)v. \tag{2}$$

The dynamic equation is then approximated by a set of linear equations as follows:

$$M\ddot{x} = f_0 - Kx - Dv, \tag{3}$$

where

$$K = -(\partial f/\partial x)(x_0, v_0), \tag{4}$$
$$D = -(\partial f/\partial v)(x_0, v_0), \tag{5}$$
$$f_0 = f(x_0, v_0). \tag{6}$$

By setting $K' = M^{-1}K$, $D' = M^{-1}D$, and $f'_0 = M^{-1}f_0$, we have a linear differential equation system,

$$\dot{x} = v, \tag{7}$$
$$\dot{v} = -K'x - D'v + f'_0. \tag{8}$$

2.2 State Equations

In the Euler methods, the continuous differentials are replaced by discrete differences, such as

$$\dot{x} = (x[t+1] - x[t])/h, \tag{9}$$

where $x[t]$ represents the sampled values and h is the time step. The left side of the dynamic equation is discretized in this way, but we still have a choice with respect to the right-hand side, whether to take the values at t or at $t+1$. Explicit methods take the values at t, while implicit methods take those at $t+1$.

(1) Explicit Euler method

Applying the above mentioned replacements to Eq. (8) yields a set of linear algebraic equations representing the explicit Euler method:

$$x[t+1] = x[t] + hv[t], \tag{10}$$
$$v[t+1] = v[t] - h(K'x[t] + D'v[t]) + hf', \tag{11}$$

or in a more compact matrix form

$$w[t+1] = Aw[t] + hg_0, \tag{12}$$

where

$$w[t] = \begin{pmatrix} x \\ v \end{pmatrix}[t] \tag{13}$$

$$g_0 = \begin{pmatrix} 0 \\ f'_0 \end{pmatrix} \tag{14}$$

$$A = \begin{pmatrix} 1 & h \\ -hK' & -hD' + 1 \end{pmatrix}. \tag{15}$$

Matrix A defines the system and is called the *state transition matrix*. Vector w and Eq. (12) are called, respectively, the *state* and the *state equation*.

(2) Implicit Euler method

In a similar way, taking the right-hand side values at $t+1$ yields the state equation for the implicit Euler method as follows:

$$w[t+1] = A_{iE}w[t] + hg'_0, \tag{16}$$
$$A_{iE} = (1 + h^2K' + hD')^{-1}\begin{pmatrix} 1 + hD' & h \\ -hK' & 1 \end{pmatrix}.$$

The solution, $v[t+1]$, can be obtained from the lower half of Eq. (16),

$$v[t+1] = (1 + h^2K' + hD')^{-1} (hK'x[t] + (hD' + 1)v[t] + hf'_0). \tag{17}$$

The calculation is no longer straightforward unlike that of the explicit methods, and a linear equation system (17) should be numerically solved at each time step.

(3) Explicit Runge-Kutta methods

The replacement of differentials, Eq. (9), is not the only way. While the Euler methods simply applies a linear extrapolation of $\int f dt \simeq hf$, the Runge-Kutta methods apply higher-order extrapolation.

The second-order Runge-Kutta method, also called the mid-point method, solves the equation by using [8]

$$k_1 = hAw_0$$
$$k_2 = hA(w_0 + k_1/2)$$
$$w(t+h) = w_0 + k_2$$

for $w_0 = w(t)$. Thus, the state equation is described as

$$w[n+1] = w(t+h) = (1 + hA + (h^2/2)A^2)w[n], \tag{18}$$

In a similar way, we have the state equation for the fourth-order Runge-Kutta method as

$$w[n+1] = (1 + hA + (h^2/2)A^2 + (h^3/6)A^3 + (h^4/24)A^4)w[n]. \tag{19}$$

2.3 System Stability

In digital control/filtering theories, system stability is defined by the state equations. Let us follow their formulation.

First, we defined the stability of the simulation system in the following way: *when finite force* $\|f\| < \infty$ *is applied for a finite duration, state* w *(the position and velocity) is also finite,* $\|w\| < \infty$. In linear systems, this is equivalent to the statement that *from any initial state $w[0]$, state $w[t]$ tends to zero if no force is applied.*

Using state transition matrix A, we can formulate this by:

$$w[n] = Aw[n-1] = A^n w[0] \to 0 \tag{20}$$

as $n \to \infty$. In the one-dimensional case, both w and A are scalars, and $|A| < 1$ is the necessary and sufficient condition. In the general case, A is a matrix and it can be proved that *the system is stable if and only if all the absolute value of A's eigen values is less than 1.* Therefore, eigen analysis of the state transition matrix, Eq. (15) provides the stability conditions.

3. Stability Conditions

It is known that explicit methods are not always stable even though the exact solution is finite [5], [6]. This section analyzes the eigen values of Eq. (15) and provides stability conditions directly applicable to spring-mass simulation. We start from the simplest case where both x and v are scalars, and follow this with a generalization. Further, we also analyze higher-order Runge-Kutta methods in the spring-mass simulation.

It might be thought preferable to specify physical parameters based on physical measurements or material data. However, it is generally hard to obtain those for damping factors because of their dependence on geometric shapes, surface states, and so on. In visual simulation, high frequency behavior is not visually noticeable, and there is some room to choose damping factors without significant visual artifacts. Thus, damping factors are often designed to improve stability [1]. According to this idea, this section also presents conditions for damping factors as well as time-step.

In visual simulation, mass and stiffness is critical to visual appearance of objects, they are specified based on material data or kinematic features. However, it is hard to get physical data for damping factors since they also depends on object shapes as well.

3.1 2D Case

Let us consider the simplest case where both $D' = d$ and $K' = k$ is a scalar. In this case, the state transition matrix A is a 2×2 matrix and the eigen values can be easily calculated from a quadratic equation. By using dimension-less variables

$$X = hd, \quad Y = h^2 k, \quad D = X^2 - 4Y, \tag{21}$$

the eigen values λ can be expressed as

$$\lambda = (1 - X/2) \pm \sqrt{D}/2.$$

The inequality $|\lambda| < 1$ yields the stable region in X-Y plane as:

$$Y < X \tag{22}$$

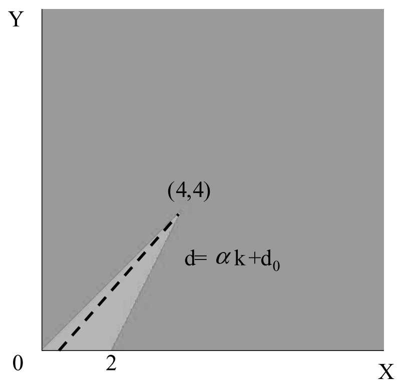

Fig. 1 Theoretical stable condition for the explicit Euler method. The light gray region is the stable region.

$$Y > 2X - 4 \tag{23}$$
$$Y > 0 \tag{24}$$

This stable region is shown in Fig. 1. An outline of the calculation is described in the Appendix. When h, d, and k hold (X, Y) within this region, stable simulation is guaranteed. The divergence was also numerically tested and an identical result was obtained. As seen in the figure, the region is tight and the selection of the damping factors is critical.

In most cases, the spring constant and the mass are determined by the materials. Thus, let us consider how to decide the maximum time-step and the damping factor with the goal of stable simulation.

(1) maximum time-step

As seen in the figure, the maximum value of Y is 4. Therefore, from the definition (Eq. (21)), the maximum time-step h_{max} for given k is given by

$$h_{max} = 2/\sqrt{k}. \tag{25}$$

(2) damping factor

When $Y = 4$, X should be 4, thus,

$$d_{max} = 4/h_{max} \tag{26}$$

guarantees stable simulation.

3.2 *N*-Dimensional Cases

When K' and D' are simultaneously diagonalizable matrices, modal analysis can be applied, which decomposes the problem into a set of 2D problems (See the Appendix). Let k_i and d_i be the i'th eigen values of K' and D', respectively. Modal analysis shows that the simulation is stable when the stability condition (Eq. (24)) is satisfied by all pairs of k_i and d_i.

The maximum time-step h_{max} is thus given by

$$h_{max} = 2/\sqrt{k_0}, \tag{27}$$

where k_0 denotes the maximum eigen value of K'.

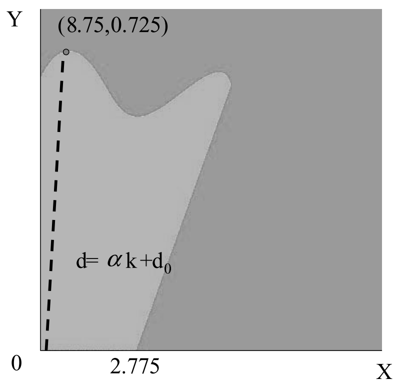

Fig. 2 Stable condition for the explicit 4th-order Runge-Kutta method.

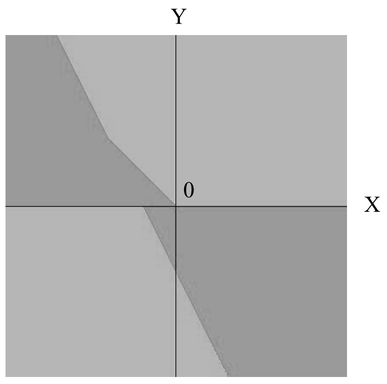

Fig. 3 Stable condition for the implicit Euler method.

Although damping factors d_i may be determined one by one, it is more convenient to parameterize them. We found the Rayleigh damping expression [9]

$$D' = d_0 I + \alpha K' \tag{28}$$

to be simple and useful. Constant term d_0 represents air resistance, for example, and dominates the low-frequency, and hence visually significant, motion. The second term stabilizes the high frequency modes, and we experimentally confirmed that its visual influence is very small. Therefore, we can adjust α to maximize time-step without introducing visual artifacts.

Equation $d = d_0 + \alpha k$ is a straight line in Fig. 1, and is described by

$$X = d_0 h + (\alpha/h)Y \tag{29}$$

When the line passes through points (X_{max}, Y_{max}) and $(hd_0, 0)$, the line segment $0 < Y < h^2 k_0$ lies in the stable region, as shown in Fig. 1. This leads to

$$\alpha = h(X_{max} - d_0 h)/Y_{max}, \tag{30}$$

$$= h(4 - d_0 h)/4 \tag{31}$$

$$hd_0 < 2. \tag{32}$$

These are the stable conditions for the damping factors. Note that d_0 can be determined based on the 'terminal velocity' of the particles in the free fall situation, and the second inequality, Eq. (32), automatically holds in most cases (See the Appendix).

3.3 Stability of Higher-Order Explicit Methods

With a small amount of additional calculations, it is also possible to analyze higher-order Runge-Kutta methods. Only the results are described here, the details are presented in the Appendix. The stable region for the fourth-order Runge-Kutta method is expressed by 8th-order algebraic equations, as is shown in Fig. 2. The maximum $Y_{max} \simeq 8.75$ is obtained at $X_{max} \simeq 0.725$. Therefore, the time-step and α are determined by

$$h_{max} = \sqrt{8.75}/\sqrt{k_0},$$

$$\alpha = h(0.725 - d_0 h)/8.75. \tag{33}$$

Note that the maximum time step is $\sqrt{8.75}/2 = 1.48$ times longer than that of the Euler method. This supports Volino's observation that higher-order methods are more stable than lower-order methods [4].

3.4 Stability of Implicit Methods

It is known that the implicit Euler method is stable whenever D' and K' are positive definite [5], [6]. Figure 3 shows the stable area. It also shows the over-stability of the method: in region $d < 0, k < 0$, although a physical system is unstable, it is forced to converge by the method.

4. Linearized Explicit Euler Method

Following the above discussion, we have the optimized time-step and damping factors. However, the execution time is still larger than that of the implicit method as seen in the next section. We found that the key to the speed of the implicit method is its linear approximation used in the force calculation. The calculation of the non-linear force and the maintenance of geometric properties such as normal vectors are much more expensive than sparse matrix-vector multiplications. Reducing the number of non-linear estimations allows the explicit Euler method to be accelerated as well. Based on this consideration, we introduce the linearization scheme into the explicit method. At each time-step, h_l, required by stability equation Eq. (25), the force is evaluated by matrix-vector products as in

$$f = -K(x - x_0) - D(v - v_0) + f_0. \tag{34}$$

Matrices K and D are estimated at a longer time interval, h_{nl}; that is, only as frequently as in the implicit Euler method, for example, three times per frame. As seen in the next section, this algorithm is even faster than the implicit methods. To avoid instability due to non-linearity, K and D are re-evaluated whenever signs of divergence are detected. The procedure is detailed below:

1) Evaluate K and D.

2) Repeat the following while $dt < h_{nl}$:

   2.1) Calculate the force using the linear approximation, Eq. (34).

   2.2) Calculate $w = (x, v)$.

   2.3) Check divergence (e.g., $||v_i|| >$ th), and if detected, discard the latest result and go back to Step 1 to re-evaluate K and D.

   2.4) $dt+ = h_l$.

5. Experiments and Discussion

In this section, the efficiency of the methods in spring-mass simulation is experimentally compared and their time complexity is analyzed to present fundamental data useful when animation systems are designed.

5.1 Time Complexities

The factors determining the execution time are the number of particles, n, and the maximum eigen value of K', denoted by k_0. The CPU time was measured for the implicit Euler (IMP), the explicit Euler (EU), the linearized explicit Euler methods described in the previous section (LEU), and the explicit 4th-order Runge-Kutta method (RK4). The test sequence was the draping action of a cloth under gravity, similar to Volino's experiment [4]. Collision handling was turned off. The test object is a square mesh, shown in Fig. 4. The physical model adopts Baraff's potentials for shear, bend, and stretch [1]. The time-step for the implicit Euler is kept at 0.01 seconds because a too large time-step makes it impractical to deal with collision and non-linearity. For the explicit methods, on the other hand, the time-step is set to the maximum according to Eqs. (27) and (33). In the linearized explicit Euler method, the interval of non-linear evaluation is also set at 0.01 seconds.

The results are shown in Figs. 5 and 6. In the first experiment, we changed the number of particles, while the spring coefficients of the particles are kept constant; this holds the maximum eigen value k_0 almost constant. In the second experiment, the number of particles was 16×16, and the spring coefficients were controlled. As shown in the figure, the linearized explicit Euler method is the fastest; twice as fast as the implicit Euler method. The explicit Runge-Kutta method is also faster than the naive explicit Euler method, since it takes advantages of linear force estimation and longer time-steps.

The figures also show that the growth in the required cpu-time for all methods is basically proportional to both n and $\sqrt{k_0}$, and thus, the time complexity of the methods is $O(nk_0^{1/2})$. This is rather obvious for the explicit methods because the time-step is proportional to the inverse of $\sqrt{k_0}$, and each step costs $O(n)$. The cost of the implicit Euler method is more complicated and is discussed in the following section.

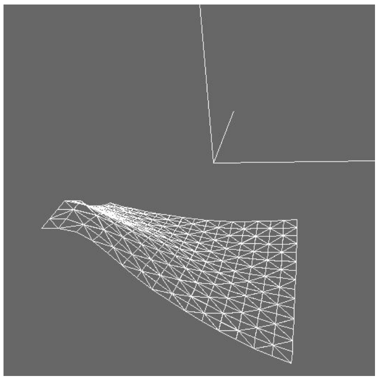
Fig. 4 Draping cloth mesh.

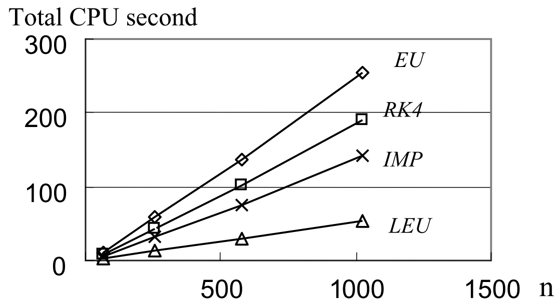

Fig. 5 Computation time versus number of the particles.

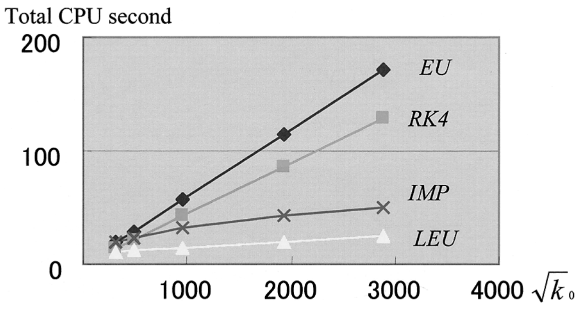

Fig. 6 Computation time versus square root of the maximum eigen value ($\sqrt{k_0}$).

5.2 Fuller Analysis of Implicit Euler

In the implicit Euler method, the linear equation system $H\delta_v = b$ should be solved at each time-step. Baraff adopted the preconditioned conjugate gradient (PCG) method to solve the equation. The PCG method iteratively solves the equation, and in each iteration, matrix-vector product Hc is calculated [1]. Since $H = (1 + h^2K' + hD')$ is very sparse, typically with just a constant number of non-zero elements per row, the cost of the multiplication is $O(n)$. Hence, the number of required iterations determines overall complexity.

It is known that the convergence ratio at the i'th iteration [10] is

$$\epsilon_k = ||e_i||/||e_0||$$

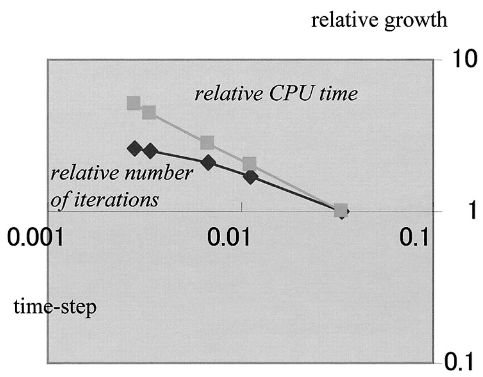

Fig. 7 Computation time and number of iterations versus time-step.

$$
\begin{aligned}
&< 2c^k/(1+c^{2k})\\
c &= (\sqrt{\sigma}-1)/(\sqrt{\sigma}+1)\\
\sigma &= \lambda_{max}/\lambda_{min}.
\end{aligned}
$$

When the error falls below the threshold ϵ_{th} after N iterations, we have

$$2N \sim \log(\epsilon_{th}/2)(\sqrt{\sigma}), \qquad (35)$$

for $\sigma \gg 1$. Therefore, the convergence rate depends on the square root of the ratio of the maximum and minimum eigen values. The maximum/minimum eigen values of H, μ_{min} and μ_{max}, can be roughly estimated as

$$
\begin{aligned}
\mu_{min} &\sim 1,\\
\mu_{max} &\sim h^2 k_0.
\end{aligned}
$$

By using Eq. (35), an estimate of the number of required iterations, N_{pcg}, is given by

$$N_{pcg} \sim O(hk_0^{1/2}). \qquad (36)$$

The estimation is based on the worst-case analysis and does not provide an exact value, but it does explain the results shown in Fig. 6.

Eq. (36) also raises the interesting fact that the required iterations per time-step are reduced when the time-step decreases. Figure 7 shows the growth in the total number of iterations and the total cpu time with respect to the time-step. As shown in the figure, the increase in required iterations is very mild when the time-step decreases. The evaluation cost of K pushes up the cpu time, but the growth is less than proportional in this range. This suggests that the adaptive time-stepping approach is still useful in stabilizing the system without large additional cost. Indeed, the positive definiteness of H, not that of K', is strictly required by the PCG method for convergence. Switching to other general, more expensive solvers like Gauss-Jordan can be fatal in terms of cost. In such cases, decreasing the time-step can keep the positive definiteness of H by reducing the influence of K' on H.

5.3 Implementation Issues

In the explicit methods, maximum eigen value k_0 has to be calculated to determine the time-step. Although this calculation is expensive in general, the maximum eigen value of sparse matrices can be inexpensively calculated by the power method [10], [11]. This runs at $O(n)$ in practice. In most cases, k_0 does not increase drastically, thus k_0 is calculated as a pre-process. The time-step is then calculated for $k'_0 = F_{safe}k_0$ with safety factor F_{safe} (1.5 in our case). This is recalculated only when divergence is detected. The collision detection is similar to that of Baraff, with a hierarchical bounding box structure, and edge-edge/vertex-mesh collision detection.

The explicit methods can easily introduce frictional force, and a snap shot from a friction example sequence is shown in Fig. 8.

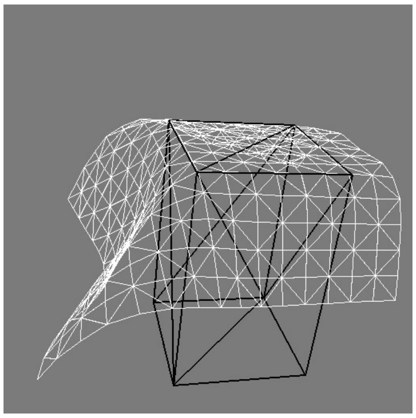

Fig. 8 An example of cloth deformation under frictional force between the box and cloth.

6. Conclusion

The stability theories introduced here is directly applicable to spring-mass simulation, and made the explicit methods as stable as the implicit methods in computer animation. Based on the theories, a faster explicit method was developed, and its efficiency was demonstrated by experiments.

The advantages of the explicit methods are:

- faster execution with linearization,
- easier handling of non-analytic, procedural forces such as friction.

Therefore, when Rayleigh damping is acceptable, the linearized explicit Euler (LEU) method is more efficient than the implicit method. When accuracy is important, the explicit Runge-Kutta method might be more appropriate.

This paper was based on the linear theories, but system non-linearity can be another source of instability. Future work includes the analysis of the non-linear properties to provide more general framework for physics-based animation.

Acknowledgments

The author would like to thank Makoto Tsukada, Emiko

Ishiwata, and Tian-Bo Deng for their helpful discussion. This research was partially supported by the Ministry of Education, Science, Sports and Calture, Grant-in-Aid for Science Research (C), 2003, 14550380.

References

[1] D. Baraff and A. Witkin, "Large steps in cloth animation," Proc. SIGGRAPH'98, pp.43–54, ACM Press/ACM SIGGRAPH, 1998.
[2] K. Choi and H. Ko, "Stable but resposive cloth," Proc. SIGGRAPH 2002, pp.604–611, ACM Press/ACM SIGGRAPH, 2002.
[3] R. Bridson, R. Fedkiw, and J. Anderson, "Robust treatment of collisions, contact snd friction for cloth animation," Proc. SIGGRAPH 2002, pp.594–603, ACM Press/ACM SIGGRAPH, 2002.
[4] P. Volino and N. Magnenat-Thalmann, "Comparing efficiency of integration methods for cloth simulation," Proc. CGI'01, 2001.
[5] E. Hairer and G. Wanner, Solving Ordinary Differential Equations II, second revised ed., Springer, 1996.
[6] T. Mitsui, Jyobibun houteishiki no suuchi kaihou, Iwanami Shoten, 2003. (in Japanese).
[7] M. Shinya, "Stabilizing explicit methods in spring-mass simulation," Proc. CGI2004, pp.528–531, 2004.
[8] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling, Numerical recipes in C, Cambridge University Press, 1988.
[9] D. James and D. Pai, "Dyrt: Dynamic response textures for real time deformation simulation with graphics hardware," Proc. SIGGRAPH 2002, pp.582–585, ACM Press/ACM SIGGRAPH, 2002.
[10] Y. Saad, Iterative methods for sparse linear systems, PWS Publishing, Boston, 1996.
[11] H. Togawa, Handbook of numerical methods, Science-Sha, 1992. (in Japanese).

Appendix A: Stability Conditions for Explicit Methods

For convenience, let us represent the state transition matrix as

$$A = 1 + hC.$$

Using C and its maximum eigen value μ, the stable condition $|1+\mu|^2 < 1$ is rewritten as:

$$\begin{aligned} 1 + 2\Re(\mu) + |\mu|^2 &< 1, \\ \Re(\mu) &< -|\mu|^2/2. \end{aligned}$$

For convenience, μ is represented in the form of

$$\mu = P + Q\sqrt{D}.$$

When $D < 0$,

$$\begin{aligned} |\mu|^2 &= P^2 - DQ^2, \\ \Re(\mu) &= P, \end{aligned}$$

and the condition is

$$P^2 - DQ^2 < -2P. \tag{A·1}$$

When $D \geq 0$, the conditions are:

$$-2 < P < 0 \tag{A·2}$$

$$P^2 > DQ^2 \tag{A·3}$$

$$(P+2)^2 > DQ^2 \tag{A·4}$$

Therefore, calculating P and Q provides the stability conditions for each method.

A.1 Explicit Euler

For the explicit Euler method,

$$A = \begin{pmatrix} 1 & h \\ -hk & -hd+1 \end{pmatrix}, \tag{A·5}$$

$$C = \begin{pmatrix} 0 & 1 \\ -k & -d \end{pmatrix}. \tag{A·6}$$

The eigen values, μ_1, can be calculated by using the quadratic equation,

$$\det(C - \mu_1 I) = \mu_1(\mu_1 + d) + k = 0,$$

Solving it yields:

$$P = -X/2, \quad Q = 1/2.$$

where X, Y, and D are defined by Eq. (21).

A.2 Fourth-Order Runge-Kutta

From the state Eq. (19), the eigen value for the fourth-order Runge-Kutta, μ_4, can be expressed by using that of the Euler method, μ_1, as:

$$h\mu_1 + (1/2)(h\mu_1)^2 + (1/6)(h\mu_1)^3 + (1/24)(h\mu_1)^4,$$

and then,

$$\begin{aligned} P &= -X/2 + X^2/4 - X^3/12 + X^4/48 - Y/2 + XY/4 \\ &\quad -X^2Y/12 + Y^2/24, \\ Q &= -1/2 + X/4 - X^2/12 + X^3/48 + Y/12 - XY/24. \end{aligned}$$

These provide the stability condition using Eqs. (A·1) and (A·4).

A.3 Simultaneously Diagonalizable Case

When D' and K' are simultaneously diagonalizable, all of their eigen vectors y_i are common:

$$D'y_i = d_i y_i, \quad K'y_i = k_i y_i.$$

Let us denote (x, v) as a linear combination of y_i as in:

$$\begin{pmatrix} x \\ v \end{pmatrix} = \begin{pmatrix} \sum x'_i y_i \\ \sum v'_i y_i \end{pmatrix}.$$

Putting them into the state equation yields a set of 2D equations:

$$\begin{pmatrix} x'_i \\ v'_i \end{pmatrix}[t+1] = \begin{pmatrix} 1 & h \\ -hk_i & -hd_i + 1 \end{pmatrix} \begin{pmatrix} x'_i \\ v'_i \end{pmatrix}[t].$$

This is in the same form as Eq. (A·6). Therefore, if all pairs of (k_i, d_i) satisfy the 2D stable condition, the original n-dimensional simulation is stable.

Appendix B: Terminal Velocity and d_0

Let us consider the physical meaning of d_0. When a particle falls under gravity, the dynamic equation is:

$$\dot{v} = g - d_0 v,$$

which approaches zero when $t \to \infty$. Setting $\dot{v} = 0$, we have the terminal velocity

$$v_\infty = g/d_0.$$

For example, if $v_\infty \sim 1$ m/sec, $d_0 \sim 10$ sec^{-1}. Since time-step h is usually $h \sim 10^{-2}$, then, $X_0 = hd_0 \sim 0.1$ in typical cases.

Mikio Shinya is currently a Professor at Department of Information Science, Toho University. He received a BSc in 1979, an MS in 1981, and a PhD in 1990 from Waseda University. He joined NTT Laboratories in 1981, and moved to Toho University in 2001. He was a visiting scientist at the University of Toronto in 1988–1989. His research interests include computer graphics and visual science.