질량-스프링 네트워크의 안정적 시뮬레이션

강영민 / 동명대학교

1. 암시적 적분법을 통한 안정적 시뮬레이션

1.1. 암시적 해법이란 무엇인가?

수치해석에서 시간에 의존적인 미분 방정식의 수치적 근사해를 얻기 위해 사용하는 방법으로 명시적 기법(explicit method)와 암시적 기법(implicit method)이 있다.

명시적 기법은 다음 시간에서의 시스템 상태를 계산할 때, 현재의 상태를 이용하여 계산하는 방식이다. 반면 암시적 기법은 현재의 상태와 미래의 상태를 모두 포함한 방정식을 풀어서 해를 구하는 방식이다.

Y^t가 현재 시간 t에서의 상태라고 하고, 우리가 구하려는 미래의 상태를 Y^{t+h}라고 하자. 이때, h는 우리가 계산하려는 미래의 시간이 지금부터 얼마나 흘렀는지를 나타내는 작은 시간 간격이다. 명시적 기법은 현재의 상태를 다음과 같이 어떤 함수 Φ에 단순히 입력함으로써 해를 얻을 수 있는 방식이다.

$$Y^{t+h} = \Phi(Y^t)$$

반면, 암시적 기법은 다음과 같은 방정식을 풀어 미래의 상태 Y^{t+h}를 계산한다.

$$\Psi(Y^t, Y^{t+h}) = 0$$

이러한 조건을 만족하는 해를 찾기 위해 암시적 기법은 명시적 기법에 비해 계산을 더 요구한다. 그럼에도 불구하고 암시적 기법을 사용하는 이유는 명시적 기법이 불안정성 문제에 취약하기 때문이다. 질량-스프링 모델은 불안정성 문제가 발생하는 전형적인 물리 모델이다. 이러한 모델을 합리적인 수준의 시간 간격 h를 사용하면서도 안정적으로 시뮬레이션하는 일은 명시적 기법으로는 거의 불가능하다. 따라서 질량-스프링 모델의 시뮬레이션은 암시적 기법으로 해를 구한다.

1.2. 질량-스프링 모델의 시뮬레이션의 암시적 기법 표현

스프링의 힘은 후크(Hooke)의 법칙에 따라 쉽게 구할 수 있다. 이렇게 구한 힘을 $\mathbf{f}^t$를 적분하여 속도를 갱신하고, 이 속도를 이용하여 위치를 갱신하는 방법을 간단한 오일러(Euler) 수치적분으로 구현하면 다음과 같다.

$$\begin{pmatrix}\mathbf{v}^{t+h}\\ \mathbf{x}^{t+h}\end{pmatrix} = \begin{pmatrix}\mathbf{v}^t + h\mathbf{M}^{-1}\mathbf{f}^t\\ \mathbf{x}^t + h\mathbf{v}^{t+h}\end{pmatrix}$$

상태는 속도와 위치로 구성되며, 이를 갱신해 나가는 것이 시뮬레이션이라고 할 수 있다. 이때 속도 벡터 $\mathbf{v}$ 라는 것은 하나의 입자가 가진 속도가 아니라, n 개의 입자가 가진 속도를 모두 포함한 $3n$ 개의 원소를 가진 벡터 $[\mathbf{v}_{1.x}\ \mathbf{v}_{1.y}\ \mathbf{v}_{1.z}\ \mathbf{v}_{2.x}\ \mathbf{v}_{2.y}\ \mathbf{v}_{2.z}\ \cdots\ \mathbf{v}_{n.x}\ \mathbf{v}_{n.y}\ \mathbf{v}_{n.z}]^{\mathrm{T}}$를 의미한다. 행렬 $\mathbf{M}$은 질량 행렬로 n 개의 입자에 대응하는 n개의 3×3행렬이 대각 성분으로 자리한다. i 번째 입자의 질량이 m_i라고 하고 이 입자에 해당하는 질량 행렬 $\mathbf{M}_i\in\mathbb{R}^{3\times3}$은 대각행렬이며, 그 값과 역행렬은 다음과 같다.

$$\mathbf{M}_i=\begin{pmatrix} m_i & 0 & 0 \\ 0 & m_i & 0 \\ 0 & 0 & m_i \end{pmatrix} \quad \mathbf{M}_i^{-1}=\begin{pmatrix} 1/m_i & 0 & 0 \\ 0 & 1/m_i & 0 \\ 0 & 0 & 1/m_i \end{pmatrix} \quad , \mathbf{M}_i\in\mathbb{R}^{3\times3}$$

전체 시스템에서 사용되는 질량 행렬 $\mathbf{M}$은 이 $\mathbf{M}_i$를 대각성분으로 하는 행렬로 다음과 같다.

$$\mathbf{M}=\begin{pmatrix} \mathbf{M}_1 & 0 & \cdots & 0 \\ 0 & \mathbf{M}_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & \mathbf{M}_n \end{pmatrix} \quad \mathbf{M}^{-1}=\begin{pmatrix} \mathbf{M}_1^{-1} & 0 & \cdots & 0 \\ 0 & \mathbf{M}_2^{-1} & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & \mathbf{M}_n^{-1} \end{pmatrix} \quad , \mathbf{M}\in\mathbb{R}^{3n\times3n}$$

이 식에서 입자의 위치 $\mathbf{x}$를 갱신하는 것은 특별한 문제를 일으키지 않는다. 속도를 갱신하는 부분만을 보면 새로운 속도 상태 $\mathbf{v}^{t+h}$를 계산하기 위해 현재의 상태에서 바로 얻을 수 있는 $\mathbf{v}^t$와 $\mathbf{f}^t$만을 이용하는 명시적 기법을 사용하고 있음을 알 수 있다. 이 식을 암시적 기법으로 바꾸려면, 현재의 힘이 아니라 미래 상태에서 결정되는 미래의 힘 $\mathbf{f}^{t+h}$를 사용하면 된다. 이 방법은 암시적 기법 중에서도 역 오일러 방법(backward Euler method)라고 부르는 방법이다.

역 오일러 방법은 어떤 상태 함수 y의 미분 dy/dt가 $f(t,y)$라고 할 때 현재 상태 y^t를 알고 있을 때에 다음 상태 y^{t+h}를 다음과 같은 방법을 구하는 것이다.

$$y^{t+h}=y^t+hf(t+h,y^{t+h})$$

원래의 오일러 기법은 $y^{t+h}=y^t+hf(t,y^t)$로 표현된다.

역 오일러 방법으로 질량-스프링 모델의 상태 갱신식을 표현하면 다음과 같은 식을 얻게 된다.

$$\begin{pmatrix} \mathbf{v}^{t+h} \\ \mathbf{x}^{t+h} \end{pmatrix}=\begin{pmatrix} \mathbf{v}^t+h\mathbf{M}^{-1}\mathbf{f}^{t+h} \\ \mathbf{x}^t+h\mathbf{v}^{t+h} \end{pmatrix}$$

위치를 갱신하는 식은 속도를 갱신하여 얻은 값을 그대로 적용하면 되고, 앞서 살펴본 명시적 기법과 차이가 없기 때문에 속도의 갱신에만 집중하자. 이 식은 속도의 변화를 구하는 다음과 같이 바꾸어 표현할 수 있다.

$$\Delta\mathbf{v}^{t+h}=h\mathbf{M}^{-1}\mathbf{f}^{t+h}$$

달라진 것은 $\mathbf{f}^t$를 $\mathbf{f}^{t+h}$로 바꾼 것 밖에 없다. 그러나 이것은 미래의 상태로 결정되는 힘을 의미하기 때문에 아직 우리가 알지 못 하는 값이다. 따라서 이 값은 근사치를 구해서 사용할 수밖에 없다. 이를 위해 우리는 다음과 같이 테일러 급수(Taylor series)를 이용하여 미래 힘의 근사치를 구한다.

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

이때, n 개의 입자에 작용하는 n 개의 힘으로 이루어진 힘 벡터 $\mathbf{f}$을 위치 벡터로 미분한 $\partial \mathbf{f}^t / \partial \mathbf{x}$는 3×3 크기의 작은 요소 행렬 $\mathbf{J}_{ij}$가 $n \times n$개 있는 야코비(Jacobi) 행렬이며, $\mathbf{J}$로 표현할 수 있다. 행렬의 i 행 j열 요소는 i 질점에 작용하는 힘 $\mathbf{f}_i$을 질점 j의 위치 $\mathbf{x}_j$로 미분한 값이다. 즉, $\mathbf{f}_i$는 i에 연결된 모든 스프링이 만들어내는 힘의 총합이지만, $\mathbf{x}_j$에 대한 편미분에서 고려될 힘은 스프링 (i,j)에 의해 발생하는 힘 $\mathbf{f}_{ij}$ 뿐이다. 따라서 $\mathbf{J}_{ij}$는 $\partial \mathbf{f}_{ij}^t / \partial \mathbf{x}_j$이다.

하나의 힘이 3차원 벡터, 즉 $\mathbb{R}^3$의 원소이므로 $\mathbf{J}_{ij}$는 $\mathbb{R}^3$의 원소인 벡터를 $\mathbb{R}^3$의 원소인 벡터로 미분한 것이고, 이것은 $\mathbb{R}^{3\times 3}$의 원소인 행렬이 된다. $\mathbf{J}$ 행렬은 이 $\mathbf{J}_{ij}$를 $n \times n$ 개 가지고 있기 때문에 $\mathbf{J} \in \mathbb{R}^{3n \times 3n}$ 이 된다. 특히 $\mathbf{J}_{ii}$는 이 행렬의 대각성분이 되는 $\mathbb{R}^{3\times 3}$ 행렬이며, $\partial \mathbf{f}_i^t / \partial \mathbf{x}_i$이므로 다음과 같다. 아래 식에서 S는 스프링의 집합을 의미한다.

$$\mathbf{J}_{ii} = \frac{\partial \mathbf{f}_i}{\partial \mathbf{x}_i} = \sum_{(i,j)\in S} \frac{\partial \mathbf{f}_{ij}}{\partial \mathbf{x}_i} = -\sum_{(i,j)\in S} \frac{\partial \mathbf{f}_{ji}}{\partial \mathbf{x}_i} = -\sum_{(i,j)\in S} \mathbf{J}_{ij}$$

그러면 이제 속도의 변화는 다음과 같은 식으로 표현된다.

$$\Delta \mathbf{v}^{t+h} = h\mathbf{M}^{-1}\left(\mathbf{f}^t + \mathbf{J}\Delta \mathbf{x}^{t+h}\right)$$

미래의 위치인 $\Delta \mathbf{x}^{t+h}$는 $h(\mathbf{v}^t + \Delta \mathbf{v}^{t+h})$로 표현할 수 있으므로, 이를 대입하여 넣어 다음 식을 얻는다.

$$\begin{aligned}\Delta \mathbf{v}^{t+h} &= h\mathbf{M}^{-1}(\mathbf{f}^t + h\mathbf{J}(\mathbf{v}^t + \Delta \mathbf{v}^{t+h})) \\ &= h\mathbf{M}^{-1}\mathbf{f}^t + h^2\mathbf{M}^{-1}\mathbf{J}\mathbf{v}^t + h^2\mathbf{M}^{-1}\Delta \mathbf{v}^{t+h}\end{aligned}$$

우리가 계산하여 얻고 싶은 미지수는 $\Delta \mathbf{v}^{t+h}$에 담긴 값들이다. 이를 계산하기 위한 선형 시스템 꼴로 바꾸면 다음과 같다.

$$\begin{aligned}\mathbf{M}\Delta \mathbf{v}^{t+h} - h^2\mathbf{J}\Delta \mathbf{v}^{t+h} &= h\mathbf{f}^t + h^2\mathbf{J}\mathbf{v}^t \\ (\mathbf{M} - h^2\mathbf{J})\Delta \mathbf{v}^{t+h} &= h\mathbf{f}^t + h^2\mathbf{J}\mathbf{v}^t\end{aligned}$$

이제 속도의 변화를 구하는 방법은 선형 시스템의 해를 푸는 문제가 된다.

$$\Delta \mathbf{v}^{t+h} = (\mathbf{M} - h^2\mathbf{J})^{-1}(h\mathbf{f}^t + h^2\mathbf{J}\mathbf{v}^t)$$

1.3. 힘의 야코비 행렬을 구하는 방법

역 오일러 기법을 통한 질량의 속도 갱신 방법은 힘의 도함수인 힘의 야코비 행렬 $\mathbf{J}$를 구하는 것이 필요하다. 이것은 모두 $n \times n$ 개의 $\mathbf{J}_{ij}$를 가지고 각각의 $\mathbf{J}_{ij}$는 힘 $\mathbf{f}_i$를 $\mathbf{x}_j$로 미분한 것이다. 이때 $i \neq j$라고 가정하면 $\mathbf{J}_{ij}$는 비대각 성분이고, 질점 i가 j와 스프링으로 연결되어 있지 않다면 $\mathbf{J}_{ij}$는 0 행렬이다. 의미 있는 $\mathbf{J}_{ij}$ 성분은 $(i,j) \in S$인 경우뿐이다. 이때 S는 스프링의 집합이며, (i,j)가 이 집합에 원소라는 것은 질점 i와 j가 스프링으로 연결되어 있다는 의미이다. 이때 $\mathbf{f}_{ij}(\mathbf{x}_j - \mathbf{x}_i) = -\mathbf{f}_{ji}(\mathbf{x}_i - \mathbf{x}_j)$이므로 $\mathbf{J}_{ij}$는 $\mathbf{J}_{ji}$와 동일하다.

$$\mathbf{J}_{ij} = \frac{\partial \mathbf{f}_{ij}(\mathbf{x}_j - \mathbf{x}_i)}{\partial \mathbf{x}_j} = \frac{-\partial \mathbf{f}_{ji}(\mathbf{x}_i - \mathbf{x}_j)}{\partial \mathbf{x}_j} = -\frac{-\partial \mathbf{f}_{ji}(\mathbf{x}_i - \mathbf{x}_j)}{\partial \mathbf{x}_i} = \mathbf{J}_{ji}$$

따라서 대각성분을 제외한 의미 있는 $\mathbf{J}_{ij}$는 모두 스프링의 개수인 $|S|$ 개 뿐이다. 대각성분인 $\mathbf{J}_{ii}$는 모두 n 개가 있으며, 이들은 연결되어 있는 모든 스프링 (i,j)에 대해 미분한 결과를 모두 더해야 한다. 이때 미분은 j가 아니라 i에 대해 미분하므로, $\mathbf{J}_{ij}$와 부호가 반대가 된다. 따라서 $\mathbf{J}_{ii}$는 다음과 같이 구할 수 있다.

$$\mathbf{J}_{ii} = \quad - \sum_{(i,j) \in S} \mathbf{J}_{ij}$$

이제 하나의 비대각 성분 $\mathbf{J}_{ij}$를 구하는 방법을 살펴 보자. 물론 $(i,j) \in S$일 경우에만 계산하면 된다. 질점 i와 j를 연결하는 스프링 (i,j)에 의해 질점 i에 작용하는 힘 $\mathbf{f}_{ij}$는 다음과 같이 계산한다. 이 식에서 l_{ij}^0는 스프링 (i,j)의 휴지(休止) 상태에서의 길이이다.

$$\mathbf{f}_{ij}^t = \kappa(|\mathbf{x}_j^t - \mathbf{x}_i^t| - l_{ij}^0)\frac{(\mathbf{x}_j - \mathbf{x}_i)}{|\mathbf{x}_j - \mathbf{x}_i|}$$

따라서 이 힘을 질점 j의 위치 $\mathbf{x}_j$로 미분한 $\mathbf{J}_{ij}$는 다음과 같다.

$$\mathbf{J}_{ij} = \frac{\partial \mathbf{f}_{ij}^t}{\partial \mathbf{x}_j} = \kappa \frac{\partial\left(\frac{|\mathbf{x}_j^t - \mathbf{x}_i^t| - l_{ij}^0}{|\mathbf{x}_j - \mathbf{x}_i|}(\mathbf{x}_j - \mathbf{x}_i)\right)}{\partial \mathbf{x}_j} = \kappa \frac{\partial(\phi(\mathbf{x}_j)\psi(\mathbf{x}_j))}{\partial \mathbf{x}_j} = \kappa\left(\frac{\partial \phi(\mathbf{x}_j)}{\partial \mathbf{x}_j}\psi(\mathbf{x}_j) + \frac{\partial \psi(\mathbf{x}_j)}{\partial \mathbf{x}_j}\phi(\mathbf{x}_j)\right)$$

$$where \;\; \phi(\mathbf{x}_j) = \frac{|\mathbf{x}_j^t - \mathbf{x}_i^t| - l_{ij}^0}{|\mathbf{x}_j - \mathbf{x}_i|} \quad , \quad \psi(\mathbf{x}_j) = \mathbf{x}_j - \mathbf{x}_i$$

이때 $h(\mathbf{x}_j) = \mathbf{x}_j - \mathbf{x}_i$이므로, $\partial h(\mathbf{x}_j)/\partial \mathbf{x}_j$는 간단히 $\mathbf{I}_{33}$임을 알 수 있다. 따라서 이 미분의 핵심은 $\partial g(\mathbf{x}_j)/\partial \mathbf{x}_j$를 구하는 것이다. 이때, 스프링 (i,j)의 현재 시간 t에서의 길이를 의미하는 $|\mathbf{x}_j - \mathbf{x}_i|$를 l_{ij}^t라는

함수로 표현하면, 이 미분은 다음과 같다.

$$\frac{\partial \phi(\mathbf{x}_j)}{\partial \mathbf{x}_j} = \frac{\partial}{\partial \mathbf{x}_j}(1 - l_{ij}^0 / l_{ij}^t) = -\, l_{ij}^0 \frac{\partial}{\partial \mathbf{x}_j}(1/l_{ij}^t) = \frac{l_{ij}^0}{{l_{ij}^t}^2} \frac{\partial}{\partial \mathbf{x}_j} l_{ij}^t$$

여기서 아직 미분되지 않은 부분 $\partial l_{ij}^t / \partial \mathbf{x}_j$는 다음과 같이 구한다.

$$\frac{\partial}{\partial \mathbf{x}_j} l_{ij}^t = \frac{\partial}{\partial \mathbf{x}_j} \sqrt{(\mathbf{x}_j - \mathbf{x}_i)^{\mathrm{T}} (\mathbf{x}_j - \mathbf{x}_i)} = \frac{1}{2\sqrt{(\mathbf{x}_j - \mathbf{x}_i)^{\mathrm{T}} (\mathbf{x}_j - \mathbf{x}_i)}} \frac{\partial}{\partial \mathbf{x}_j} (\mathbf{x}_j - \mathbf{x}_i)^{\mathrm{T}} (\mathbf{x}_j - \mathbf{x}_i)$$

$\partial \mathbf{x}^{\mathrm{T}} \mathbf{x} / \partial \mathbf{x} = 2\mathbf{x}^{\mathrm{T}}$ 이므로 위의 미분은 다음과 같이 정리할 수 있다.

$$\frac{\partial}{\partial \mathbf{x}_j} l_{ij}^t = \frac{(\mathbf{x}_j - \mathbf{x}_i)^{\mathrm{T}}}{\sqrt{(\mathbf{x}_j - \mathbf{x}_i)^{\mathrm{T}} (\mathbf{x}_j - \mathbf{x}_i)}} = \frac{(\mathbf{x}_j - \mathbf{x}_i)^{\mathrm{T}}}{l_{ij}^t}$$

따라서 $\partial \phi(\mathbf{x}_j) / \partial \mathbf{x}_j$는 다음과 같다.

$$\frac{\partial \phi(\mathbf{x}_j)}{\partial \mathbf{x}_j} = \frac{l_{ij}^0}{{l_{ij}^t}^3} (\mathbf{x}_j - \mathbf{x}_i)^{\mathrm{T}}$$

지금까지의 결과를 모두 모으면 다음과 같이 $\mathbf{J}_{ij}$를 구할 수 있다.

$$\mathbf{J}_{ij} = \kappa \left(\frac{l_{ij}^0}{{l_{ij}^t}^3} (\mathbf{x}_j - \mathbf{x}_i)(\mathbf{x}_j - \mathbf{x}_i)^{\mathrm{T}} + \frac{l_{ij}^t - l_{ij}^0}{l_{ij}^t} \mathrm{I}_{33} \right)$$

1.4. 암시적 기법의 구현

암시적 기법을 구현하기 위해서는 힘의 야코비 행렬인 $\mathbf{J}$를 구하는 것이 중요하며, 이 행렬이 포함된 선형방정식을 풀면 된다. 이를 질량-스프링 모델을 구성하는 각 질점의 위치와 스프링 연결 정보, 그리고 각 질점의 속도와 질점에 가해지는 힘은 일반적인 방법으로 쉽게 구할 수 있으며, 이러한 정보를 알고 있을 경우 다음 알고리즘을 이용하여 각 질점의 속도와 위치를 갱신할 수 있다. 이 방법은 댐핑(damping)이 없을 경우 안정적인 결과를 보인다. 댐핑이 포함될 경우에는 댐핑을 유발하는 힘의 미분이 포함되지 않았기 때문에 스프링의 힘과 비교했을 때에 댐핑이 중요한 역할을 수행한다면 불안정성이 나타날 수도 있다. 댐핑에 대한 미분은 나중에 고려하기로 한다.

입력으로는 우선 n, e, κ, h가 필요하며, 이들은 각각 질점 개수, 스프의 개수, 스프링 상수, 시간 간격을

의미한다. 다음으로 각 질점의 상태를 나타내는 $\mathbf{v}$와 $\mathbf{x}$가 필요하다. $\mathbf{v}$는 각 질점의 속도 n 개가 모여 이뤄진 벡터로 $\mathbf{v}[i] \in \mathbb{R}^3$이다. $\mathbf{x}$ 역시 비슷하게 각 질점의 3차원 좌표가 n 개가 모여 이뤄진 벡터로 $\mathbf{x}[i] \in \mathbb{R}^3$이며, 질점에 작용하는 힘은 $\mathbf{f}$로 표현하며 역시 각 질점에 작용하는 힘 n 개가 모여 이뤄진 벡터이며, $\mathbf{f}[i] \in \mathbb{R}^3$이다. m도 질점의 특징을 나타내는 정보이며, 각 질점의 질량으로 이뤄진 배열이다. 따라서 $m[i]$는 i 질점의 질량을 의미한다.

다음으로 필요한 입력은 스프링과 관련된 정보 $\mathbf{s}$와 l_0이다. $\mathbf{s}$는 두 질점 i와 j를 연결하는 스프링을 표현하는 (i,j)의 배열이며, 원소의 개수는 e 개이며, 각 원소는 (i,j) 형태로 표현되는 정수 쌍이다. l_0는 스프링의 휴지 길이 배열로서 $l_0[i]$는 i 번째 스프링, 즉 $\mathbf{s}[i]$의 휴지상태 길이를 저장하고 있다.

10 행의 outerProduct 함수는 두 벡터를 곱해 텐서(tensor)를 얻는 텐서곱으로서의 외적을 의미하며, 어떤 두 벡터 $\mathbf{a}$와 $\mathbf{b}$의 텐서곱은 $\mathbf{a}\mathbf{b}^{\mathrm{T}}$와 같다. 17 행의 computeMatrixInverse는 입력으로 넘어온 행렬의 역행렬을 구해 반환하는 함수이다.

1.5. 효율적 애니메이션을 위한 근사해 구하기

암시적 기법은 $(\mathbf{M}-h^2\mathbf{J})\Delta\mathbf{v}^{t+h} = h\mathbf{f}^t + h^2\mathbf{J}\mathbf{v}^t$라는 선형 방정식의 해를 푸는 것이다. 이때 $\mathbf{A} = (\mathbf{M}-h^2\mathbf{J})$라 하고, $\mathbf{b} = h\mathbf{f}^t + h^2\mathbf{J}\mathbf{v}^t$라고 하면 이 선형 방정식은 $\Delta\mathbf{v}^{t+h} = \mathbf{A}^{-1}\mathbf{b}$로 표현할 수 있다.

이 문제는 벡터 $\mathbf{b}$와 행렬 $\mathbf{A}$의 역행렬 $\mathbf{A}^{-1}$을 구하는 문제가 된다. 그런데 $\mathbf{b}$를 계산하는 데에는 $O(n^2)$의 시간복잡도가 요구되며, 일반적으로 $n \times n$ 크기 행렬의 역행렬을 구하는 것은 $O(n^3)$의 시간 복잡도가 요구된다. 따라서 단순한 역행렬 계산법으로는 이 문제를 효율적으로 해결할 수 없다. 많은 경우 질량-스프링을 구성하는 입자의 수가 수 천 개에 이르기도 하기 때문에 행렬의 크기가 매우 커지고, 이 행렬의 역행렬을 구하는 것이 실시간이나 효율적 시간 내에 이루어지지 않는다. 따라서 우리는 이 $\mathbf{A}^{-1}$를 효율적으로 계산하는 방법을 고려해야 한다.

이때 $\mathbf{A}$ 행렬은 많은 성분이 0인 희소행렬(sparse matrix)이다. 그 이유는 $\mathbf{M}$ 행렬의 경우 대각성분 이외의 값은 모두 0인 대각행렬이며, 힘의 야코비 행렬인 $\mathbf{J}$의 경우에는 3×3 크기의 부행렬(submatrix)를 하나의 성분으로 간주할 때 i행 j열 성분이 0행렬이 아니려면 질점 i와 j가 연결되어 있을 때 뿐이다. 따라서 모두 n 개의 질점이 존재하고, 스프링의 개수가 e라고 하면, $\mathbf{J}$ 행렬에 0 행렬이 아닌 3×3 부행렬은 모두 $n+2e$개 존재한다. 이것은 일반적인 질량-스프링 모델을 고려할 때, $\mathbf{J}$의 성분 개수인 $n \times n$보다 훨씬 적은 수이다. 또한 $\mathbf{A}$의 i번째 대각 성분은 질점 i의 질량 행렬인 $m_i\mathbf{I}_{33}$의 값에 $h^2\mathbf{J}_{ii}$를 뺀 $m_i\mathbf{I}_{33} - h^2\mathbf{J}_{ii}$의 값인데, $\mathbf{J}_{ii}$의 값은 $-\boldsymbol{\Sigma}_{(i,j)\in E}\,\mathbf{J}_{ij}$ 의 값이므로, 대각성분은 $m_i\mathbf{I}_{33} + h^2\boldsymbol{\Sigma}_{(i,j)\in E}\,\mathbf{J}_{ij}$로 한 행에서 대각성분의 값이 다른 비대각성분의 값을 모두 더한 값에 질량이 더해진 형태이므로 대각성분이 선형시스템의 특징을 좌우한다고 할 수 있다. 다시 말해, 대각성분만을 취해도 선형시스템의 특성이 많이 유지된다고 할 수 있다.

만약 $\mathbf{A}$와 대각행렬이라고 가정하면 $\mathbf{A}^{-1}$을 구하기 위해 어렵게 역행렬 구하기 알고리즘을 사용할 필요가 없다. 대각행렬의 역행렬은 각 대각성분의 역수만 구하면 되기 때문이다. 우리의 시스템은 하나의 성분이 스칼라(scalar) 값이 아니고 $\mathbb{R}^{3\times3}$의 행렬이므로 $\mathbf{A}_{ii}^{-1}$을 i행 i열 대각성분에 두면 $\mathbf{A}^{-1}$을 구할 수 있다. 다시 말해

속도의 변화 $\Delta \mathbf{v}$는 선형 시스템 풀이가 아니라 다음과 같이 각각의 질점 별로 개별적으로 계산이 가능하다.

$$\Delta \mathbf{v}_i = \mathbf{A}_{ii}^{-1} \mathbf{b}_i$$

문제는 $\mathbf{A}$가 대각성분이 중요한 역할을 수행하는 행렬이기는 하지만, 완전한 대각행렬은 아니라는 점이다. 이럴 경우에 위의 방식은 실제 해에 근접한 근사해가 된다. 그러나 $\mathbf{A}$가 대각행렬 $\mathbf{D}$와 비대각성분으로 구성된 나머지 행렬 $\mathbf{R}$로 구분했을 때, $\mathbf{R}$이 0이 아니고 의미있는 값을 가진다면 $\mathbf{A} = (\mathbf{D} + \mathbf{R})$라고 표현할 수 있다. 따라서 선형 시스템은 $(\mathbf{D} + \mathbf{R})\Delta \mathbf{v} = \mathbf{b}$로 표현할 수 있고, 다음과 같은 관계식을 얻을 수 있다.

$$\begin{aligned} \mathbf{D}\Delta \mathbf{v} + \mathbf{R}\Delta \mathbf{v} &= \mathbf{b} \\ \mathbf{D}\Delta \mathbf{v} &= \mathbf{b} - \mathbf{R}\Delta \mathbf{v} \\ \Delta \mathbf{v} &= \mathbf{D}^{-1}\mathbf{b} - \mathbf{D}^{-1}\mathbf{R}\Delta \mathbf{v} \end{aligned}$$

그러면 $\Delta \mathbf{v}_i = \mathbf{A}_{ii}^{-1} \mathbf{b}_i$는 $\mathbf{D}_{ii}^{-1} \mathbf{b}_i$와 같은 값이므로 이렇게 얻은 $\Delta \mathbf{v}$는 다음과 같은 오차 ϵ를 가진다.

$$\epsilon = \mathbf{D}^{-1}\mathbf{R}\Delta \mathbf{v}$$

이 오차는 $\mathbf{R}$이 0 행렬에 가까워지면 0에 근접하게 된다. 그런데 이 $\mathbf{R}$이 0행렬은 아니지만 $\mathbf{D}$보다는 중요하지 않은 경우, 즉 대각성분이 특성을 좌우하는 대각지배행렬(diagonally dominant matrix)의 경우 이 오차를 순차적으로 줄여나가는 방법이 있다. 이것을 야코비 반복법이라고 한다.

야코비 반복법은 우리가 구하려고 하는 벡터를 좌우변에 동시에 나타나게 한 다음 식을 이용한다.

$$\Delta \mathbf{v} = \mathbf{D}^{-1}(\mathbf{b} - \mathbf{R}\Delta \mathbf{v})$$

여기서 좌변의 미지수 벡터는 반복법을 통해 새롭게 계산할 것이고, 우변의 미지수 벡터는 우리가 이미 추정한 값이다. 즉 k번째 반복을 통해 얻은 미지수 벡터가 $\Delta \mathbf{v}^{(k)}$라면 야코비 반복법은 다음과 같은 방식으로 미지수 벡터를 반복적으로 갱신하는 것이다.

$$\Delta \mathbf{v}^{(k+1)} = \mathbf{D}^{-1}(\mathbf{b} - \mathbf{R}\Delta \mathbf{v}^{(k)})$$

알고리즘 2는 이러한 야코비 반복법을 이용하여 근사해를 구하는 방법이다. 반복의 횟수가 k라고 하면 이 방법은 $O(k(n+e))$의 시간 복잡도를 갖는다.

알고리즘 1.	암시적 기법을 이용한 속도와 위치의 갱신

Function Implicit_Integration

n: 질점 수, e: 스프링 수, κ: 스프링 상수, h: 시간간격

$\mathbf{v}$: 속도, $\mathbf{x}$: 위치, $\mathbf{f}$: 힘, m: 질량, $\mathbf{s}$: 스프링 정보, l_0: 휴지상태 길이

```
01  M ← array(n×n, element: R^{3×3}) : initialize with 0
02  J ← array(n×n, element: R^{3×3}) : initialize with 0
03
04  M_ii = m[i]I_33  for each i from 0 to n−1
05
06  for spring from 0 to e−1 :
07      (i,j) = s[spring]
08      x_ji = x[j] − x[i]
09      l^t = |x_ji|
10      J[i][j] = κ*[ ((l^t - l_0[spring])/l^t)*I_33 +
11                    (l_0[spring]/(l^t*l^t*l^t))*outerProduct(x_ji , x_ji)]
12      J[j][i] = J[i][j]
13      J[i][i] ,  J[j][j] = J[i][i] − J[i][j] ,  J[j][j] − J[i][j]
14
15  A ← array(n×n, element: R^{3×3}) : initialize with 0
16  A ← M + h^2 J
17  dV ← array(n, element: R^3)
18  A^{-1} = computeMatrixInverse(A)
19  dV = A^{-1}(hf + h^2 Jv)
20
21  for i from 0 to n−1 :
22      v[i] = v[i] + dV[i]
23      x[i] = x[i] + h*v[i]
```

알고리즘 2.	암시적 기법의 근사해를 구해 속도와 위치의 갱신

Function Approx_Implicit_Integration

n: 질점 수, e: 스프링 수, κ: 스프링 상수, h: 시간간격

$\mathbf{v}$: 속도, $\mathbf{x}$: 위치, $\mathbf{f}$: 힘, m: 질량, $\mathbf{s}$: 스프링 정보, l_0: 휴지상태 길이

```
01  A_diag ← array(n, element: R^3×3) : initialize with 0
02  J_diag ← array(n, element: R^3×3) : initialize with 0
03  J_ij ← array(e, element: R^3×3) : initialize with 0
04
05  for each spring from 0 to e−1 :
06      (i,j) = s[spring]
07      x_ji = x[j] − x[i]
08      l^t = |x_ji|
09      J_ij[spring] = κ*[ ((l^t - l_0[spring])/l^t)*I_33 +
10                  (l_0[spring]/(l^t*l^t*l^t))*outerProduct(x_ji , x_ji)]
11      J_diag[i] −= J_ij[spring] ,  J_diag[j] −= J_ij[spring]
12
13  A_diag[i] = m[i]I_33 + h*hJ_diag[i]  for each i from 0 to n−1
14
15  b ← array(n, element: R^3) : initialize with 0
16   dV ←array(n, element: R^3)
17
18  for each i from 0 to n−1 :
19      b[i] = h*f^t[i] + h*h*J_diag[i]*v[i]
20  for each spring from 0 to e−1 :
21      (i,j) = s[spring]
22      b[i] = b[i] + h*h*J_ij[spring]*v[i]
23      b[j] = b[j] + h*h*J_ij[spring]*v[j]
24
25  for each i from 0 to n−1 :
26      dv[i] = A_diag[i]^−1*b[i]
27
28  for user−defined iterations
29      for each spring from 0 to e−1 :
30              (i,j) = s[spring]
31              b[i] += h*h*J_ij[spring]*dv[j]
32              b[j] += h*h*J_ij[spring]*dv[i]
33      for each i from 0 to n−1 :
34              dv[i] = A_diag[i]^−1*b[i]
35
36   for i from 0 to n−1 :
37      v[i] = v[i] + dV[i]
38      x[i] = x[i] + h*v[i]
```

1.6. 댐핑(damping)이 포함된 힘의 야코비(Jacobi) 행렬

질량-스프링 모델이 안정적으로 에너지 평형 상태에 도달하도록 하기 위해서는 적절한 댐핑(damping) 힘을 추가하는 것이 좋다는 것이 잘 알려져 있다. 댐핑은 시스템의 에너지를 소산(消散)하게 하는 힘으로 질점 하나의 경우에는 항력(抗力, drag force)의 형태가 된다. 항력은 항력계수 c_d를 질점의 속도에 곱해 얻을 수 있고, 그 방향은 이동하는 방향의 반대 방향이 된다. 즉 하나의 입자가 저항을 받아 에너지를 잃도록 하는 힘 $\mathbf{f}_d$는 다음과 같이 표현할 수 있다.

$$\mathbf{f}_d = -c_d\mathbf{v}$$

그런데 이러한 항력은 질량-스프링 모델에 댐핑으로 사용하면, 질량-스프링 모델이 스프링 운동을 통해 에너지를 잃는 것이 아니라 밀도가 높은 유체 속을 운동하는 것과 같은 결과가 나온다. 이러한 문제를 피하기 위해서 질량 스프링 모델에서는 스프링으로 연결된 두 질점 i와 j 이 가지는 상대속도를 이용해 댐핑을 구현한다.

어떤 스프링 (i,j)가 연결하는 두 질점 i와 j가 가진 속도가 각각 $\mathbf{v}_i$와 $\mathbf{v}_j$라면, 질점 j에 대한 질점 i의 상대속도 $\mathbf{v}_i-\mathbf{v}_j$이며, 이 스프링의 움직임에 의해 질점 i와 j에 가해지는 댐핑 힘 $\mathbf{f}_{ij}^d$와 $\mathbf{f}_{ji}^d$는 다음과 같다.

$$\mathbf{f}_{ij}^d = -c_d(\mathbf{v}_i-\mathbf{v}_j)$$
$$\mathbf{f}_{ji}^d = -c_d(\mathbf{v}_j-\mathbf{v}_i)$$

따라서 (i,j) 스프링으로 연결된 두 질점 i와 j의 관계에서 질량 i에 작용하는 힘은 다음과 같이 고쳐 쓸 수 있다.

$$\mathbf{f}_{ij}^t = \kappa(|\mathbf{x}_j^t-\mathbf{x}_i^t|-l_{ij}^0)\frac{(\mathbf{x}_j-\mathbf{x}_i)}{|\mathbf{x}_j-\mathbf{x}_i|}-c_d(\mathbf{v}_i-\mathbf{v}_j)$$

이 힘은 위치뿐만이 아니라 속도에도 의존적인 함수이므로, 이제 미래 상태의 힘 $\mathbf{f}^{t+h}$를 계산할 때에 새로운 미분이 추가되어야 한다. 즉, 위치에 대한 힘의 미분뿐만 아니라, 속도에 대한 미분도 포함한 다음과 같은 식을 사용해야 한다.

$$\mathbf{f}^{t+h}(\mathbf{x},\mathbf{v}) = \mathbf{f}^t(\mathbf{x},\mathbf{v}) + \frac{\partial\mathbf{f}^t(\mathbf{x},\mathbf{v})}{\partial\mathbf{x}}\Delta\mathbf{x}^{t+h} + \frac{\partial\mathbf{f}^t(\mathbf{x},\mathbf{v})}{\partial\mathbf{v}}\Delta\mathbf{v}^{t+h}$$

이때 다음과 같이 위치에 대한 미분을 $\mathbf{J}^{\mathbf{x}}$라고 하고, 속도에 대한 미분을 $\mathbf{J}^{\mathbf{v}}$라고 하자.

$$\mathbf{J}^{\mathbf{x}} = \frac{\partial\mathbf{f}^t(\mathbf{x},\mathbf{v})}{\partial\mathbf{x}}$$
$$\mathbf{J}^{\mathbf{v}} = \frac{\partial\mathbf{f}^t(\mathbf{x},\mathbf{v})}{\partial\mathbf{v}}$$

여기서 나타난 $\mathbf{J}^{\mathbf{x}}$는 앞서 살펴본 $\mathbf{J}$와 동일하며 비대각성분 $\mathbf{J}_{ij}^{\mathbf{x}}$와, 대각성분 $\mathbf{J}_{ii}^{\mathbf{x}}$는 다음과 같다.

$$\mathbf{J}_{ij}^{\mathbf{x}} = \kappa\left(\frac{l_{ij}^0}{l_{ij}^{t\,3}}(\mathbf{x}_j - \mathbf{x}_i)(\mathbf{x}_j - \mathbf{x}_i)^{\mathrm{T}} + \frac{l_{ij}^t - l_{ij}^0}{l_{ij}^t}\mathbf{I}_{33}\right) \quad , \quad \mathbf{J}_{ii}^{\mathbf{x}} = \; -\sum_{(i,j)\in S}\mathbf{J}_{ij}^{\mathbf{x}}$$

우리가 새롭게 구해야 하는 $\mathbf{J}^{\mathbf{v}}$는 $\partial \mathbf{f}_{ij}^d / \partial \mathbf{v}$로서 $\mathbf{J}_{ij}^{\mathbf{v}}$는 다음과 같이 구할 수 있다.

$$\mathbf{J}_{ij}^{\mathbf{v}} = \; -c_d\frac{\partial(\mathbf{v}_i - \mathbf{v}_j)}{\mathbf{v}_j} = c_d\mathbf{I}_{33}$$

이 $\mathbf{J}^{\mathbf{v}}$의 대각성분 $\mathbf{J}_{ii}^{\mathbf{v}}$는 i 행의 비대각성분을 모두 더한 값에 부호를 반대로 하면 된다. 이것은 결국 질점 i에 연결된 스프링의 개수 λ_i에 의해 다음과 같이 결정된다.

$$\mathbf{J}_{ii}^{\mathbf{v}} = -\lambda c_d\mathbf{I}_{33}$$

이상과 같이 $\mathbf{J}^{\mathbf{v}}$를 쉽게 구할 수 있다. 이를 포함하여 암시적 기법을 새롭게 정리하면 속도의 변화는 다음과 같은 문제를 푸는 것이 된다.

$$\Delta\mathbf{v}^{t+h} = h\mathbf{M}^{-1}[\mathbf{f}^t + h\mathbf{J}^{\mathbf{x}}(\mathbf{v}^t + \Delta\mathbf{v}^{t+h}) + \mathbf{J}^{\mathbf{v}}\Delta\mathbf{v}^{t+h}]$$

이를 일반적인 선형 시스템 꼴로 바꾸면 쉽게 다음과 같은 식을 얻을 수 있다.

$$(\mathbf{M} - h^2\mathbf{J}^{\mathbf{x}} - h\mathbf{J}^{\mathbf{v}})\Delta\mathbf{v}^{t+h} = h\mathbf{f}^t + h^2\mathbf{J}^{\mathbf{x}}\mathbf{v}^t$$

암시적 기법을 통해 속도의 변화를 계산하는 것은 결국 다음을 계산하는 것이다.

$$\Delta\mathbf{v}^{t+h} = (\mathbf{M} - h^2\mathbf{J}^{\mathbf{x}} - h\mathbf{J}^{\mathbf{v}})^{-1}(h\mathbf{f}^t + h^2\mathbf{J}^{\mathbf{x}}\mathbf{v}^t)$$

알고리즘 3과 4는 이러한 댐핑을 고려하지 않았던 알고리즘 1과 2를 각각 개선하여 댐핑 역시 고려되고, 그 미분이 활용되도록 한 것이다. 이러한 방식으로 개선하면 댐핑 힘을 어떻게 적용하든지 시스템이 불안정해지지 않는다.

알고리즘 3.	댐핑을 포함한 모델에서 암시적 기법을 이용한 속도와 위치의 갱신

Function Implicit_Integration

n: 질점 수, e: 스프링 수, κ: 스프링 상수, c_d: 댐핑 상수, h: 시간간격

$\mathbf{v}$: 속도, $\mathbf{x}$: 위치, $\mathbf{f}$: 힘, m: 질량, $\mathbf{s}$: 스프링 정보, l_0: 휴지상태 길이

00 $\mathbf{M} \leftarrow array(n \times n,\ element: \mathbb{R}^{3\times 3})$: initialize with 0
01 $\mathbf{Jx} \leftarrow array(n \times n,\ element: \mathbb{R}^{3\times 3})$: initialize with 0
02 $\mathbf{Jv} \leftarrow array(n \times n,\ element: \mathbb{R}^{3\times 3})$: initialize with 0
03
04 $\mathbf{M}_{ii} = m[i]\mathbf{I}_{33}$ for each i from 0 to $n-1$
05
06 for $spring$ from 0 to $e-1$:
07 $(i,j) = \mathbf{s}[spring]$
08 $\mathbf{x}_{ji} = \mathbf{x}[j] - \mathbf{x}[i]$
09 $l^t = |\mathbf{x}_{ji}|$
10 $\mathbf{Jx}[i][j] = \kappa * [\ ((l^t - l_0[spring])/l^t) * \mathbf{I}_{33} +$
11 $(l_0[spring]/(l^t * l^t * l^t)) *$ outerProduct($\mathbf{x}_{ji}$, $\mathbf{x}_{ji}$)]
12 $\mathbf{Jv}[i][j] = c_d\mathbf{I}_{33}$
13 $\mathbf{Jx}[j][i] = \mathbf{Jx}[i][j]$
14 $\mathbf{Jv}[j][i] = \mathbf{Jv}[i][j]$
15 $\mathbf{Jx}[i][i]$, $\mathbf{Jx}[j][j] = \mathbf{Jx}[i][i] - \mathbf{Jx}[i][j]$, $\mathbf{Jx}[j][j] - \mathbf{Jx}[i][j]$
16 $\mathbf{Jv}[i][i]$, $\mathbf{Jv}[j][j] = \mathbf{Jv}[i][i] - \mathbf{Jv}[i][j]$, $\mathbf{Jv}[j][j] - \mathbf{Jv}[i][j]$
17
18 $\mathbf{A} \leftarrow array(n \times n,\ element: \mathbb{R}^{3\times 3})$: initialize with 0
19 $\mathbf{A} \leftarrow \mathbf{M} + h^2\mathbf{Jx} + h\mathbf{Jv}$
20 $d\mathbf{V} \leftarrow array(n,\ element: \mathbb{R}^3)$
21 $\mathbf{A}^{-1} =$ computeMatrixInverse($\mathbf{A}$)
22 $d\mathbf{V} = \mathbf{A}^{-1}(h\mathbf{f} + h^2\mathbf{Jv})$
23
24 for i from 0 to $n-1$:
25 $\mathbf{v}[i] = \mathbf{v}[i] + d\mathbf{V}[i]$
26 $\mathbf{x}[i] = \mathbf{x}[i] + h * \mathbf{v}[i]$

알고리즘 4.	댐핑을 포함한 모델에서 암시적 기법의 근사해 구하기

Function Approx_Implicit_Integration

n: 질점 수, e: 스프링 수, κ: 스프링 상수, h: 시간간격

$\mathbf{v}$: 속도, $\mathbf{x}$: 위치, $\mathbf{f}$: 힘, m: 질량, $\mathbf{s}$: 스프링 정보, l_0: 휴지상태 길이

```
00  A_diag ← array(n, element: R^3×3) : initialize with 0
01  Jx_diag ← array(n, element: R^3×3) : initialize with 0
02  Jv_diag ← array(n, element: R^3×3) : initialize with 0
03  Jx_ij ← array(e, element: R^3×3) : initialize with 0
04  Jv_ij ← array(e, element: R^3×3) : initialize with 0
05
06  for each spring from 0 to e−1 :
07      (i,j) = s[spring]
08      x_ji = x[j] − x[i]
09      l^t = |x_ji|
10      Jx_ij[spring] = κ*[ ((l^t - l_0[spring])/l^t)*I_33 +
11                  (l_0[spring]/(l^t*l^t*l^t))*outerProduct(x_ji , x_ji)]
12      Jx_diag[i] −= Jx_ij[spring] ,   Jx_diag[j] −= Jx_ij[spring]
13      Jv_ij[spring] = c_d I_33
14      Jv_diag[i] −= Jv_ij[spring] ,   Jv_diag[j] −= Jv_ij[spring]
15
16  A_diag[i] = m[i]I_33 + h*hJx_diag[i] + h*Jv_diag[i]  for each i from 0 to n−1
17
18  b ← array(n, element: R^3) : initialize with 0
19  dV ←array(n, element: R^3)
20
21  for each i from 0 to n−1 :
22      b[i] = h*f^t[i] + h*h*Jx_diag[i]*v[i]
23  for each spring from 0 to e−1 :
24      (i,j) = s[spring]
25      b[i] = b[i] + h*h*Jx_ij[spring]*v[i]
26      b[j] = b[j] + h*h*Jx_ij[spring]*v[j]
27
28  for each i from 0 to n−1 :
29      dv[i] = A_diag[i]^−1 * b[i]
30
31  for user−defined iterations
32      for each spring from 0 to e−1 :
33              (i,j) = s[spring]
34              b[i] += h*h*Jx_ij[spring]*dv[j] + h*Jv_ij[spring]*dv[j]
35              b[j] += h*h*Jx_ij[spring]*dv[i] + h*Jv_ij[spring]*dv[i]
36      for each i from 0 to n−1 :
37              dv[i] = A_diag[i]^−1 * b[i]
38
39   for i from 0 to n−1 :
40      v[i] = v[i] + dV[i]
41      x[i] = x[i] + h*v[i]
```

2. 조화진동 모델을 적용한 안정적 시뮬레이션

2.1. 명시적 기법의 불안정성

명시적 기법은 $\mathbf{v}_i^{t+h} = \mathbf{v}_i^t + h\mathbf{f}_i^t/m_i$와 같이 현재 우리가 알고 있는 미분 추정치를 사용하여 수치 적분하는데, 이 미분 추정치가 시간 간격 h 동안 일정하게 유지되지 않고, 변화하는 함수이기 때문에 부정확성을 초래한다. 그리고, 이 부정확성은 시스템을 불안정하게 만드는 원인이 된다.

암시적 적분법은 이를 해결하기 위해 테일러 전개를 통해 미래의 미분치를 근사하여 추정하고, 이를 적분에 이용하기 때문에 안정성을 확보할 수 있었다. 그러나, 암시적 기법은 선형시스템을 풀어야 하는 새로운 문제를 일으키는 것을 앞서 살펴 보았다.

만약 상태의 미분이 어떻게 변화하는지를 잘 알고, 시간간격 내에서 상태의 이 미분치를 더 정확하게 적분하는 방법이 있다면, 명시적 적분을 사용하더라도 안정성을 크게 높일 수 있을 것이다. 즉 다음과 같은 적분이 가능하다면 선형시스템 풀이 없이 안정적 시뮬레이션이 가능하다.

$$\mathbf{v}^{t+h} = \mathbf{v}^t + \frac{1}{m}\int_t^{t+h} \mathbf{f}(t)dt$$

이것은 결국 함수 $\mathbf{f}(t)$의 해석적으로 다룰 수 있는 닫힌 형태(closed form) 표현을 안다면 이를 해석적으로 적분하여 새로운 상태를 얻을 수 있다는 것이다. 훅(Hooke)의 법칙에 의해 계산되는 스프링 힘은 현재의 스프링 상태에서 얼마만큼의 힘이 발생하는지는 알려주지만, 이 힘이 어떻게 변화할 것인지는 전혀 알려주지 않는다. 따라서 힘을 정확히 적분하기 위해서는 훅의 법칙에 의해 결정되는 힘이 아니라, 스프링의 운동에 대한 해석에 기반한 힘 모델이 필요하다.

정적인 객체에 스프링으로 연결된 질량의 운동은 조화진동(harmonic oscillation)으로 잘 모델링되어 있다. 이 조화진동은 해석적으로 다룰 수 있으며, 정확한 적분이 가능하다.

2.2. 조화진동(調和振動, harmonic oscillation)

조화 진동의 단순한 형태는 정적인 객체에 질량 m을 가진 질점이 스프링 강도 κ의 스프링으로 연결되어 있을 때에, 이 질점이 1차원 공간에서 진동하는 것이다. 질점의 위치는 스칼라(scalar) 값인 x로 표현되며, 다음과 같은 식으로 그 진동을 표현할 수 있다.

$$x = A\sin\omega t$$

이때 A는 진동의 폭이며, ω는 $\sqrt{\kappa/m}$로 결정된다.

우리가 다루는 문제는 하나의 질점이 정적 객체에 붙어 있는 것이 아니라, 운동하는 두 질점이 스프링의 양

쪽에 붙어 있는 모델이다. 스프링 (i,j)에 의해 두 질점 i와 j가 연결되어 있고, 각각의 위치를 $\mathbf{x}_i$와 $\mathbf{x}_j$로 표현했다. 속도는 $\mathbf{v}_i$와 $\mathbf{v}_j$이며, 각각의 질량은 m_i와 m_j로 표현된다.

질점 i에서 질점 j로 가는 벡터 $\mathbf{x}_j - \mathbf{x}_i$는 간략히 $\mathbf{x}_{ji}$로 표현하고, 비슷하게 상대속도 역시 $\mathbf{v}_j - \mathbf{v}_i$를 $\mathbf{v}_{ji}$로 표현하자. 질점의 위치는 시간에 대한 함수이므로 시간 t에서의 질점 i 위치를 $\mathbf{x}_i^t$와 같이 표현한다. 시간에 의존적인 다른 물리적 상태 역시 비슷한 방식으로 표현하자. (i,j) 스프링의 휴지상태 길이와 현재 시간 t에서의 길이는 앞서 사용한 방식과 같이 l_{ij}^0와 l_{ij}^t로 표현된다. 이는 $|\mathbf{x}_{ij}|$와 같은 값이다.

어떤 스프링 (i,j)가 일으키는 힘의 크기를 결정하는 핵심적 요소는 스프링의 변형 정도 δ_{ij}^t이다. 이 값은 $l_{ij}^t - l_{ij}^0$의 값이다.

두 질점이 연결되어 있을 때에, 이 변형 정도는 조화진동 모델에 따라 다음과 같은 진동을 할 것이다.

$$\delta_{ij}^t = A_{ij}\sin\omega_{ij}t = A_{ij}\sin\left[\left(\sqrt{\frac{\kappa(m_i+m_j)}{m_i m_j}}\right)t\right]$$

이 식에서 m_i가 무한에 접근하면 ω_{ij}는 $\sqrt{\kappa/m_j}$에 접근하고, 정적인 객체에 연결된 j의 단순 조화진동으로 접근하게 된다. m_j가 무한에 접근할 경우에는 질점 i의 단순 조화진동이 된다. 따라서 $m_i m_j/(m_i+m_j)$를 스프링으로 연결된 두 질점 i와 j의 조화진동에서 질량으로 작용하는 M_{ij}라고 볼 수 있다. 그러면 위의 진동은 다음과 같이 표현할 수 있다.

$$\delta_{ij}^t = A_{ij}\sin\sqrt{\frac{\kappa}{M_{ij}}}\,t$$

이 조화진동 모델에서 결정되지 않은 것은 스프링 (i,j)의 진동의 폭인 A_{ij}이다. 이 진폭을 결정하기 위해서는 에너지 보존 법칙이 적용된다. 스프링의 전체 에너지는 운동 에너지와 위치 에너지로 구성된다. 진폭을 알면 스프링의 총 에너지를 알 수 있는데, 스프링 에너지 총합 E_{tot}은 $\kappa A_{ij}^2/2$이다.

운동 에너지 E_k는 두 질점의 질량과 스프링 변형의 속도 $\dot{\delta}_{ij}^t$에 의해 다음과 같이 계산된다.

$$E_k = \frac{1}{2}M_{ij}\dot{\delta}_{ij}^{t\,2} = \frac{1}{2}\left(\frac{m_i m_j}{m_i+m_j}\right)\dot{\delta}_{ij}^{t\,2}$$

위치 에너지 E_p는 스프링 변형에 의해 결정되며 다음과 같다.

$$E_p = \frac{1}{2}\kappa\delta_{ij}^{t\,2}$$

따라서 에너지 보존 법칙에 따라 스프링의 총 에너지 E_{tot}은 $E_k + E_p$와 일치하여야 하므로 다음을 얻는다.

$$\kappa A_{ij}^2 = \kappa {\delta_{ij}^t}^2 + M_{ij}\dot{\delta}_{ij}^{t^2} = \kappa {\delta_{ij}^t}^2 + \left(\frac{m_i m_j}{m_i + m_j}\right)\dot{\delta}_{ij}^{t^2}$$

따라서 스프링 (i,j)의 진폭은 다음과 같다.

$$A_{ij} = \sqrt{{\delta_{ij}^t}^2 + \frac{M_{ij}}{\kappa}\dot{\delta}_{ij}^{t^2}} = \sqrt{{\delta_{ij}^t}^2 + \left(\frac{m_i m_j}{\kappa(m_i + m_j)}\right)\dot{\delta}_{ij}^{t^2}}$$

위의 식을 풀기 위해서는 스프링 변형의 미분치인 $\dot{\delta}_{ij}^t$를 알아야 한다. 이 값은 다음과 같다.

$$\frac{d}{dt}\delta_{ij}^t = \frac{d}{dt}(l_{ij}^t - l_{ij}^0) = \frac{d}{dt}l_{ij}^t$$

이 값은 다음과 같이 구할 수 있다.

$$\begin{aligned}\frac{d}{dt}l_{ij}^t &= \frac{d}{dt}\sqrt{{\mathbf{x}_{ji}^t}^{\mathrm{T}}\mathbf{x}_{ji}^t} \\ &= \frac{1}{2}{\mathbf{x}_{ji}^t}^{\mathrm{T}}{\mathbf{x}_{ji}^t}^{-\frac{1}{2}}\frac{d}{dt}{\mathbf{x}_{ji}^t}^{\mathrm{T}}\mathbf{x}_{ji}^t \\ &= \frac{1}{2}{\mathbf{x}_{ji}^t}^{\mathrm{T}}{\mathbf{x}_{ji}^t}^{-\frac{1}{2}} \cdot 2{\mathbf{x}_{ji}^t}^{\mathrm{T}}\mathbf{v}_{ji}^t \\ &= {\hat{\mathbf{x}}_{ji}^t}^{\mathrm{T}}\mathbf{v}_{ji}^t\end{aligned}$$

이때 $\hat{\mathbf{x}}$는 $\mathbf{x}$를 정규화한 벡터로 $\mathbf{x}/|\mathbf{x}|$를 의미한다. 이렇게 $\dot{\delta}_{ij}^t$를 계산하고 나면, 진폭을 쉽게 계산할 수 있다.

조화 진동은 두 개의 시간 변수 t와 T로 표현할 수 있다. 이 두 시간 변수를 이용하여 진동을 표현하면 다음과 같다.

$$\delta_{ij}^t = A_{ij}\sin\omega_{ij}T$$

이때 시간 변수 t는 일반적인 시간이며, T는 진동 주기 내에서 정의되는 시간으로서 0에서 $2\pi/\omega$의 범위내에 있는 시간이다. 조화진동에 기반한 모델을 정확히 적분하기 위해서는 이 T 변수를 이용하여 적분하여야 하며, 이 값은 다음과 같이 구할 수 있다.

$$T = \frac{1}{\omega}\sin^{-1}(\delta_{ij}^t/A_{ij})$$

이제 스프링 (i,j)가 조화진동을 일으키기 위해 이 스프링 방향으로 생성되는 힘의 크기 f_{ij}^t를 시간 t_0에서 t_0+h 구간에서 다음과 같이 해석적으로 적분할 수 있다. 이때 f_{ij}^t는 스프링 방향으로의 힘의 크기이므로 스칼라 값이다.

$$\begin{aligned}\int_{t_0}^{t_0+h} f_{ij}^t dt &= \int_{t_0}^{t_0+h} \kappa \delta_{ij}^t dt \\ &= \kappa A_{ij} \int_{T_0}^{T_0+h} \sin\omega_{ij} t \ dt \\ &= -\frac{\kappa A_{ij}}{\omega_{ij}} \cos\omega t \Big|_{T_0}^{T_0+h}\end{aligned}$$

이 적분을 이용하여 질량-스프링 모델을 더욱 정확히 시뮬레이션할 수 있다. 하나의 스프링 (i,j)가 발생시키는 힘을 적분한 이 양을 ϕ_{ij}라고 정의하면 이 값은 다음과 같이 계산할 수 있다.

$$\begin{aligned}\phi_{ij} &= \int_{T_0}^{T_0+h} f_{ij}^t dt \\ &\frac{\kappa A_{ij}}{\omega}[\cos\omega(T_0+h) - \cos\omega T_0]\end{aligned}$$

이 스프링이 발생시킨 힘의 적분은 각각의 질점 i,j에 나뉘어 져야 하고, 각각의 질점 i,j에 가해진 힘의 적분총량을 $\boldsymbol{\phi}_i$와 $\boldsymbol{\phi}_j$는 다음과 같이 ϕ_{ij}를 균등하게 나눈 값을 두 질점을 묶고 있는 스프링 방향으로 서로 반대가 되도록 적용하면 된다. 따라서 특정 질점 i에 가해지는 힘의 총량은 다음과 같이 계산할 수 있다. 이때 $\mathbf{f}^{ext}$는 외부에서 가해지는 외력이다.

$$\boldsymbol{\phi}_i = \frac{1}{2m_i} \sum_{(i,j)\in S} \phi_{ij} \widehat{\mathbf{x}_{ij}} + \mathbf{f}^{ext} h$$

각 질점의 속도 변화는 그 질점에 연결된 모든 스프링에 의해 생성되는 속도변화를 모은 것이기 때문에 다음과 같이 질점 i의 속도 변화를 구할 수 있다. 이때, $\mathbf{f}^{ext}$는 시스템 외부에서 가해지는 외력이다.

$$d\mathbf{v}_i = \frac{\boldsymbol{\phi}_i}{m_i}$$

이때 S는 질량-스프링 모델의 스프링 집합이다. 이러한 방법으로 속도를 갱신할 경우 명시적 오일러 기법과 같이 현재 알려진 미분치를 그대로 수치적분하는 것에 비해 더욱 안정적인 결과를 얻을 수 있다. 그리고, 이 방법은 각각의 스프링 단위로 힘을 계산하듯이 속도의 변화를 계산하기 때문에 전체적인 시간 복잡도가 암시적 적분법에 비해 매우 낮다는 장점이 있다. 이 방식은 명시적 기법이지만 안정적인 방법이라 할 수 있다.

2.3. 댐핑을 고려한 조화진동 적분

댐핑은 안정성 뿐만 아니라 에너지를 적절히 소산시키며 더욱 사실적인 동작을 생성하는 데에 필요하다. 앞서 살펴본 조화진동 기반 시뮬레이션은 댐핑을 고려하지 않고 있다. 그러나 사실적인 애니메이션을 위해서는 댐핑을 반드시 고려해야 하며, 이 댐핑이 포함된 힘을 적분하는 것이 필요하다. 댐핑이 포함될 경우 스프링 진동이 다음과 같다는 것이 잘 알려져 있다.

$$\delta_{ij}^t = A_{ij} e^{-\alpha_{ij} t} \sin \omega_{ij}^d t$$

이때, α는 댐핑 힘에 비례하는 계수이며, ω_{ij}^d는 댐핑으로 지연된 진동 주파수라고 할 수 있다. 우리는 우선 이 식에서 유도되는 속도의 변화 ϕ_{ij}를 계산한 뒤에, α와 ω_{ij}^d의 값을 계산하는 법을 살펴볼 것이다.

힘의 적분은 다음과 같이 댐핑을 포함한 식의 적분으로 바뀐다.

$$\int_{t_0}^{t_0+h} f_{ij}^t dt = \kappa A_{ij} \int_{T_0}^{T_0+h} e^{-\alpha_{ij} t} \sin \omega_{ij}^d t \; dt$$

잘 알려진 미적분 기술을 이용하여 우리는 그 결과가 다음과 같음을 알 수 있다.

$$\kappa A_{ij} \frac{e^{-\alpha_{ij} t}}{\alpha_{ij}^2 + \omega_{ij}^{d\,2}} (-\alpha_{ij} \sin \omega_{ij}^d t - \omega_{ij}^d \cos \omega_{ij}^d t) \Bigg|_{T_0}^{T_0+h}$$

따라서, 스프링 (i,j)가 발생시킨 힘의 크기를 적분한 결과는 다음과 같다.

$$\begin{aligned} &\kappa A_{ij} \frac{e^{-\alpha_{ij} T_0} e^{-\alpha_{ij} h}}{\alpha_{ij}^2 + \omega_{ij}^{d^2}} [-\alpha_{ij} \sin(\omega_{ij}^d T_0 + \omega_{ij}^d h) \\ &\qquad - \omega_{ij}^d \cos(\omega_{ij}^d T_0 + \omega_{ij}^d h)] \\ &- \kappa A_{ij} \frac{e^{\alpha_{ij} T_0}}{\alpha_{ij}^2 + \omega_{ij}^{d^2}} [-\alpha_{ij} \sin \omega_{ij}^d T_0 - \omega_{ij}^d \cos \omega_{ij}^d h] \end{aligned}$$

이것은 달리 표현하면 다음과 같다.

$$\kappa A_{ij}\frac{e^{-\alpha_{ij}T_0}e^{-\alpha_{ij}h}}{\alpha_{ij}^2+\omega_{ij}^{d^2}}\,[-\alpha_{ij}\sin\omega_{ij}^d T_0\cos\omega_{ij}^d h-\alpha_{ij}\cos\omega_{ij}^d T_0\sin\omega_{ij}^d h$$
$$-\omega_{ij}^d\cos\omega_{ij}^d T_0\cos\omega_{ij}^d h+\omega_{ij}^d\sin\omega_{ij}^d T_0\sin\omega_{ij}^d h)]$$
$$-\kappa A_{ij}\frac{e^{-\alpha_{ij}T_0}}{\alpha_{ij}^2+\omega_{ij}^{d^2}}\,[-\alpha_{ij}\sin\omega_{ij}^d T_0-\omega_{ij}^d\cos\omega_{ij}^d h]$$

간략한 표현을 위해 다음과 같이 정의하자.

$$C^t=\frac{e^{-\alpha_{ij}T_0}}{\alpha_{ij}^2+\omega_{ij}^{d\,2}}$$
$$C^h=\frac{e^{-\alpha_{ij}h}}{\alpha_{ij}^2+\omega_{ij}^{d\,2}}$$
$$s^t=\sin\omega_{ij}^d T_0\ ,\quad s^h=\sin\omega_{ij}^d h$$
$$c^t=\cos\omega_{ij}^d T_0\ ,\quad c^h=\cos\omega_{ij}^d h$$

스프링 (i,j)의 변형 속도 변화 ϕ_{ij}는 이 정의를 이용하여 다음과 같이 표현할 수 있다.

$$\begin{aligned}\phi_{ij}=&-\kappa A\,C^t\\&(\ \ s^t[C^h\alpha_{ij}c^h-C^h\omega_{ij}^d s^h-\alpha_{ij}]\\&+\ c^t[C^h\alpha_{ij}s^h+C^h\omega_{ij}^d c^h-\omega_{ij}^d])\end{aligned}$$

이제 이 값을 적용하여 질량-스프링 모델의 움직임을 시뮬레이션 할 수 있다. 그런데 이 식에서 아직 정해지지 않은 값은 α_{ij}와 ω_{ij}^d이다. 이 두 값을 구하여야만 스프링의 변형 속도를 계산할 수 있다. 댐핑이 포함된 단순한 조화진동은 다음과 같은 미분방정식으로 표현된다.

$$\ddot{\mathbf{x}}+\frac{c_d}{m}\dot{\mathbf{x}}+\frac{\kappa}{m}\mathbf{x}=0$$

이때 c_d는 댐핑 계수이다. 이 미분방정식을 푸는 것은 잘 알려져 있다. 댐핑이 없을 경우에 진동의 각속도 ω가 $\sqrt{\kappa/m}$으로 알려져 있다. 댐핑이 적용되었을 때의 각속도 ω^d를 구하기 위해 우선 ξ를 $c_d/2\sqrt{\kappa m}$으로 정의하면, 위의 미분방정식은 아래와 같이 표현할 수 있다.

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

이 미분방정식의 해는 다음과 같다.

$$\mathbf{x}^t=Ae^{-\xi\omega t}\cos\omega\sqrt{1-\xi^2}\,t$$

여기서 $\xi\omega$가 α이며, $\omega\sqrt{1-\xi^2}$이 댐핑에 의해 지연된 각속도 ω^d가 된다. 앞서 살펴본 바와 같이 두 개의 질량이 연결된 스프링에서의 진동 각속도 ω가 $\sqrt{\kappa/M_{ij}} = \sqrt{\kappa(m_i+m_{j)/}m_im_j}$이고 ξ값은 두 질량이 스프링에 미치는 질량 효과인 $m_im_j/(m_i+m_j)$를 $c_d/2\sqrt{\kappa m}$의 m으로 적용하여 구할 수 있다. 따라서 특정 스프링 (i,j)의 α와 ω^d의 값은 아래와 같이 구할 수 있다.

$$\xi_{ij} = \frac{c_d}{2\sqrt{\kappa M_{ij}}} = \frac{c_d}{2\sqrt{\frac{\kappa m_i m_j}{m_i+mj}}}$$

$$\alpha_{ij} = \xi_{ij}\omega_{ij} = \frac{c_d}{2\sqrt{\kappa M_{ij}}}\sqrt{\frac{k}{M_{ij}}} = \frac{c_d}{2M_{ij}} = \frac{c_d(m_i+m_j)}{2m_im_j}$$

$$\omega_{ij}^d = \sqrt{\kappa/M_{ij}}\sqrt{1-\frac{c_d^2}{4\kappa M_{ij}}} = \sqrt{\frac{\kappa(m_i+m_j)}{m_im_j}}\sqrt{1-\frac{c_d^2(m_i+m_j)}{4\kappa m_im_j}}$$

이 값들을 이용하여 시뮬레이션을 수행할 수 있다. 이러한 값을 이용하여 실제 시뮬레이션을 수행하는 방법은 알고리즘 5에 나타나 있다.

알고리즘 5.	조화진동에 기반한 질량-스프링 시뮬레이션

Function Integration_HarmonicOscillation

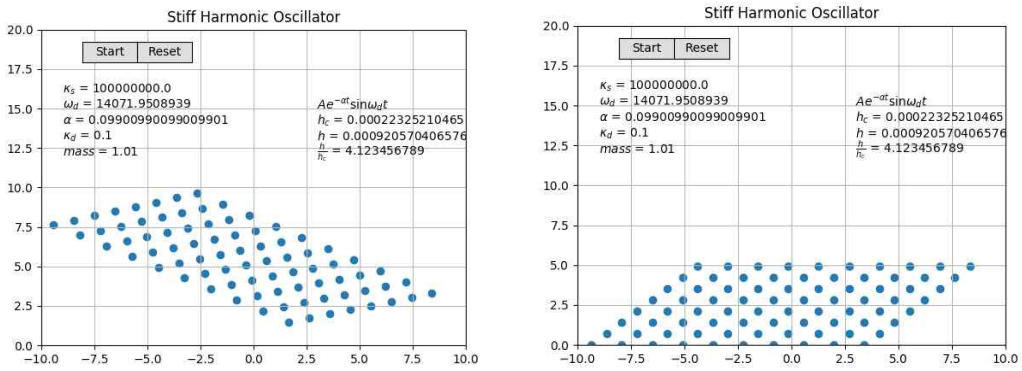
n: 질점 수, e: 스프링 수, κ: 스프링 상수, h: 시간간격

$\mathbf{v}$: 속도, $\mathbf{x}$: 위치, $\mathbf{f}^{ext}$: 스프링 외부 힘, m: 질량, $\mathbf{s}$: 스프링 정보, l_0: 스프링 휴지상태 길이

```
01  dv ← array(n, element: R^3) : initialize with 0
02  φ ← array(n, element: R^3) : initialize with 0
03  for each spring from 0 to e−1 :
04      (i,j) = s[spring]
05      x_ji = x[j] − x[i]
06      compute δ_ij and δ̇_ij with x_ji and l_0[spring]
07      compute A_ij and T_0 with δ_ij and δ̇_ij
08      compute ξ_ij, α_ij, ω^d_ij
09      compute φ_ij with ξ_ij, α_ij, ω^d_ij
10      x_ji = x[j] − x[i]
11      l^t = |x_ji|
12      φ[i] += φ_ij x_ij / 2l^t_ji
13      φ[j] -= φ_ij x_ij / 2l^t_ji
14  for each i from 0 to n−1 :
15      dv[i] = φ[i]/m[i]
16      v[i] += dv[i]
17      x[i] = x[i] + h*v[i]
```

2.4. 실험결과

이 값들을 이용하여 우리는 시뮬레이션을 수행할 수 있다. 이러한 값을 이용하여 실제 시뮬레이션을 수행하는 방법은 알고리즘 5에 나타나 있다. 그림 1은 우리 연구의 기법을 실험하기 위한 환경이다.



[그림 1. 조화진동 기반 시뮬레이션 환경]

시스템의 안정성을 분석하기 위해 임계 시간 h_c를 정의하였다. 스프링의 운동은 진동 각속도 ω에 의해 진동 주파수가 결정되는데, 신호는 이 주파수에 의해 정해지는 주기 내에 최소 두 번 이상의 샘플링이 이루어질 때, 안정적으로 복원이 가능하다. 스프링의 진동 각속도가 ω일 경우, 주기는 $2\pi/\omega$가 된다. 따라서 수치적분의 시간 간격은 π/ω보다 작은 값이 되어야 한다. 우리는 시간간격이 넘을 수 없는 이 값을 임계 시간 h_c로 정의하였다.

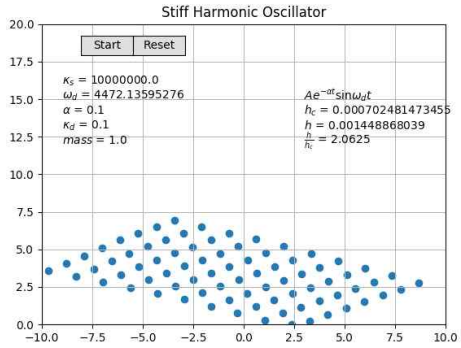
$$h_c = \frac{2\pi}{\omega}$$

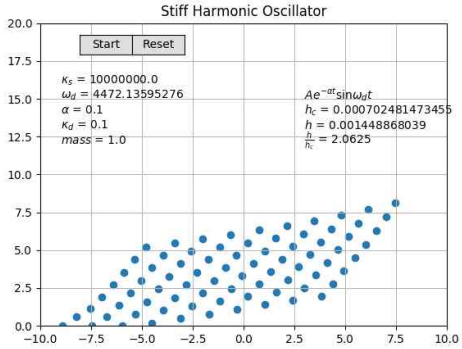
전통적인 오일러 기법은 시간 간격이 이 임계 시간보다 클 경우 불안정성을 보이게 된다. 본 연구에서 개발한 기법은 이 임계 시간보다 큰 간격을 사용하여도 안정적 시뮬레이션이 가능하다.

안정성의 정도를 측정하기 위해 μ_σ를 정의하였다. 이 값은 사용가능한 시간 간격을 임계 시간으로 나눈 것으로, 시스템의 안정성을 깨트리지 않고 임계 시간보다 얼마나 큰 시간 간격을 사용할 수 있는지를 보여준다.

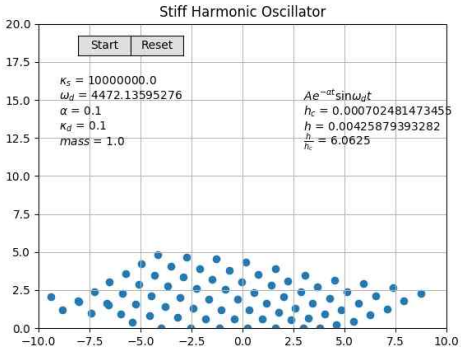
$$\mu_\sigma = \frac{h}{h_c}$$

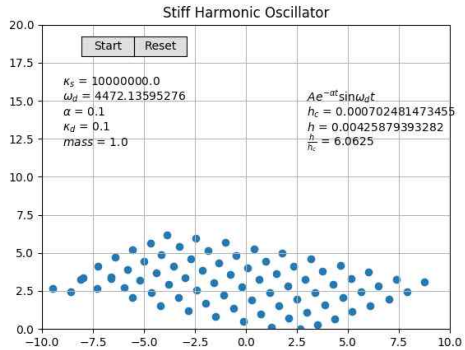
그림 2는 이러한 안정성 실험을 수행한 결과이다. 그림에서 볼 수 있는 바와 같이 임계 시간의 2배, 6배, 8배가 넘는 시간 간격을 사용했을 경우에도 안정적 시뮬레이션이 가능했다. 그림에서 확인할 수 있는 바와 같이 시간 간격을 크게 하여도 안정적 시뮬레이션이 가능하기에 효율적인 시뮬레이션이 필요한 곳에서 개발된 기법을 적용할 수 있다. 또한 이 기법은 행렬의 역행렬을 구하는 작업이 필요하지 않기 때문에 입자 단위의 병렬화가 용이하다는 장점이 있다. 따라서 병렬 처리 기법을 적용하여 대규모 복잡도를 가진 물체의 시뮬레이션을 수행하는 데에 적합하다.



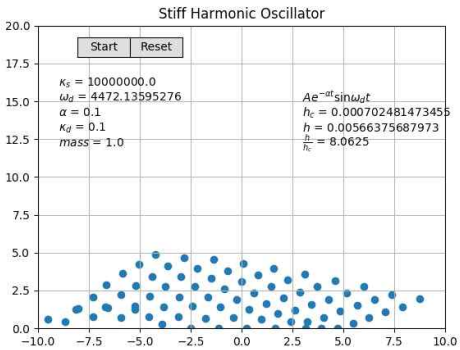


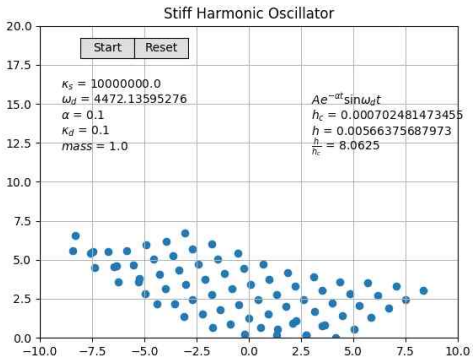
(a) $\mu_\sigma = 2.0625$





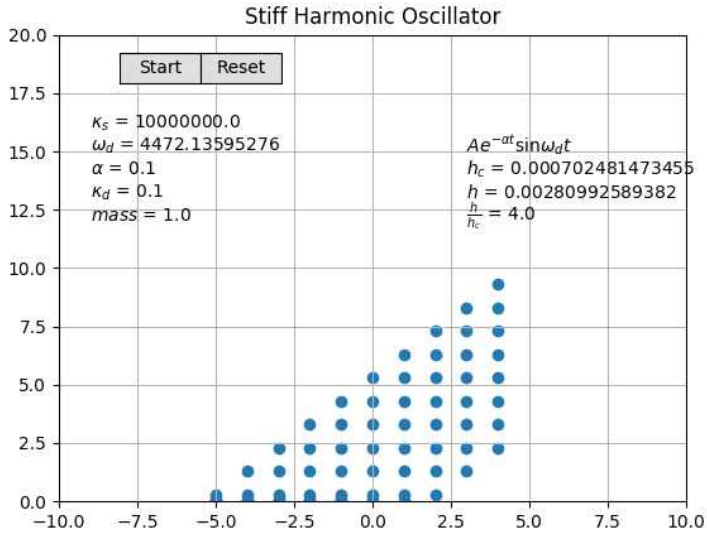
(b) $\mu_\sigma = 6.0625$





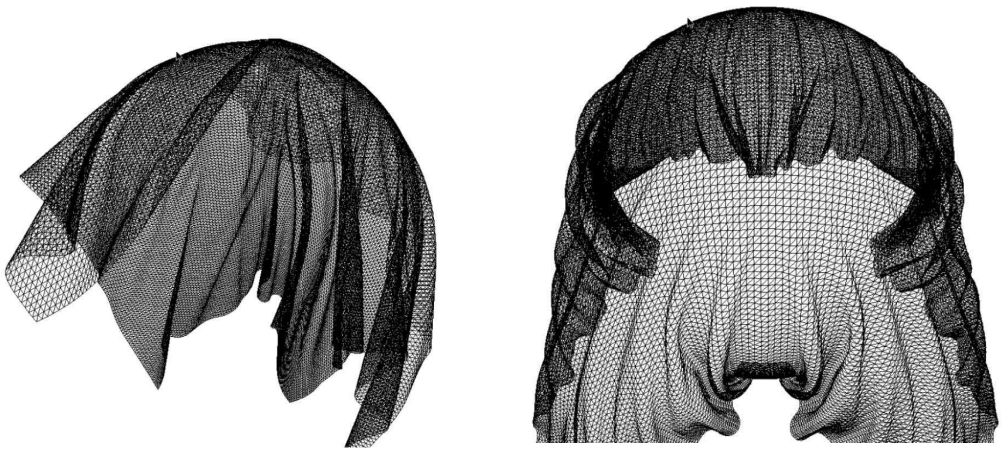
© $\mu_\sigma = 8.0625$

[그림 2. 조화진동 기반 스프링 시뮬레이션의 안정성 실험]



[그림 3. 임계 시간의 정수배 시간간격이 사용될 때의 문제]

임계 시간의 정수배인 시간간격을 사용할 경우에는 진동의 주기와 일치하여 적분된 힘의 총량이 0이 되어 버리게 된다. 따라서 그림 3과 같이 임계 시간의 4배인 시간 간격을 쓰면 스프링 힘이 항상 0이 되어 스프링으로 연결된 질점이 서로를 밀거나 당기지 못 하고 서로 무관한 입자처럼 행동하게 된다. 이러한 문제를 회피하기 위해서는 사용되는 시간간격이 임계 시간의 정수배가 되지 않도록 할 필요가 있다.



[그림 4. 조화진동 기법을 적용한 질량-스프링 모델 병렬처리 시뮬레이션 결과]

그림 4는 조화진동 기법을 병렬적으로 계산하여 옷감 시뮬레이션에 적용한 결과이다. 매우 복잡한 메시 구조를 가진 옷감 모델을 잘 시뮬레이션 할 수 있다.

3. 근사 암시적 기법과 조화진동 모델의 하이브리드 기법

조화진동 모델은 오일러 기법과 같은 전통적인 명시적 기법에 힘의 적분을 더 나은 방식으로 수행하기 때문에 안정성이 다소 높아진다. 그러나, 이렇게 해서 얻어지는 안정성은 암시적 적분법에 비해 크게 떨어지는 수준이다. 따라서 안정성의 측면에서 이 기법은 그다지 매력있는 방법은 아니다.

그러나 암시적 기법은 그 안정성에도 불구하고 메시의 구조가 복잡해질 경우 시스템에 포함된 행렬의 크기가 너무 커지고, 이 행렬의 역행렬을 구하는 것이 쉽지 않게 된다. 이러한 문제를 해결하기 위해 야코비 반복법에 기반한 근사 암시적 기법이 제안되었다. 하지만, 이 야코비 반복법은 대각지배행렬에서만 그 수렴이 보장되어 있는데, 질량-스프링 모델이 만들어내는 선형시스템에 포함된 행렬이 대각지배행렬이라는 보장이 없다. 이러한 이유로 다음 반복식에서 k를 높인다고 수렴한다는 보장이 없다. 그리고, 수렴을 하는 경우에도 반복의 횟수가 늘어나면 전체적 성능이 저하될 수밖에 없다.

$$\Delta \mathbf{v}^{(k+1)} = \mathbf{D}^{-1}(\mathbf{b} - \mathbf{R}\Delta \mathbf{v}^{(k)})$$

이때, $\mathbf{b}$는 $\mathbf{f}^t h + h^2 \mathbf{J}\mathbf{v}$로서 현재 시간 t에서의 힘 $\mathbf{f}^t$와 점성력 $h\mathbf{J}^{\mathbf{x}}\mathbf{v}$를 단순히 시간간격 h로 오일러 적분한 것이다. 이 값이 만약 더 좋은 값 $\tilde{\mathbf{b}}$로 개선된다면 계산 효율을 높이기 위해서는 단 한 번의 계산이 이루어지는 다음과 같은 식을 적용할 수 있을 것이다.

$$\Delta \mathbf{v} = \mathbf{D}^{-1}\tilde{\mathbf{b}}$$

이러한 개선된 $\tilde{\mathbf{b}}$를 구하기 위해 기존의 $\mathbf{b}$와 조화진동 기반 모델에서 구한 $\boldsymbol{\phi}$의 가중합을 구한다. 간단한 방법은 각각의 가중치를 0.5로 두는 것이다.

따라서, 각 질점의 속도 변화는 다음과 같이 구할 수 있다.

$$\begin{aligned}\Delta \mathbf{v}_i &= \mathbf{D}_{ii}^{-1}\tilde{\mathbf{b}}_i \\ &= (\mathbf{M}_{ii} - h^2\mathbf{J}_{ii}^{\mathbf{x}} - h\mathbf{J}_{ii}^{\mathbf{v}})^{-1}[\frac{1}{2}(\mathbf{f}^t h + h^2\mathbf{J}_{ii}^{\mathbf{x}}\mathbf{v}_i + h^2\sum_{(i,j)\in S}\mathbf{J}_{ij}^{\mathbf{x}}\mathbf{v}_j) + \frac{1}{2}\boldsymbol{\phi}_i]\end{aligned}$$

이 기법은 암시적 기법의 장점과 조화진동 기반 모델의 장점을 함께 가지어 안정적이면서도 빠르게 상태를 갱신할 수 있다.

조건	오일러	조화진동	야코비 반복	암시적 기법	하이브리드
$n=5\times5$ $m_i=1kg$ $c_d=0.1,$ $h=0.01$ 크기 $1m\times1m$	5.7×10^3	7.5×10^3	6.3×10^4	9.5×10^5	6.5×10^7
	5,690	7,500	62,500	950,000	65,000,000
$n=5\times5$ $m_i=2kg$ $c_d=0.1,$ $h=0.01$ 크기 $1m\times1m$	1.1×10^4	1.5×10^4	1.5×10^5	1.9×10^6	9.5×10^8
	11,250	15,100	149,000	1,910,000	950,000,000

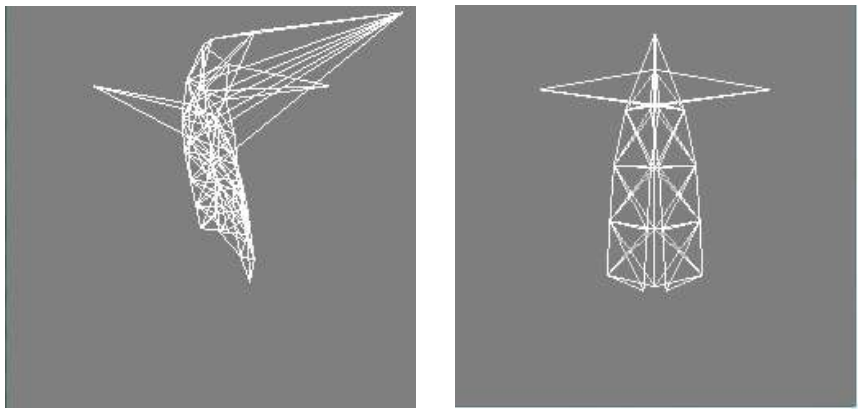
[표 1. 하이브리드 기법과 기타 기법의 안정성 비교를 위한 $\kappa_{\max}$]

입자수	오일러	조화진동	야코비 반복	암시적 기법	하이브리드
20×20	0.02	0.06	0.12	0.72	0.14
30×30	0.08	0.17	0.33	5.70	0.35
40×40	0.12	0.28	0.47	36.72	0.55

[표 2. 스프링-댐퍼 모델의 다수 입자를 갱신하는 데에 소요되는 시간 비교 (단위: s)]

표 1은 기법은 하이브리드 기법과 다른 기법들의 안정성을 비교한 것이다. 각각의 기법에서 주어진 조건에서 사용할 수 있는 최대 경직도 $\kappa_{\max}$를 측정하여 비교한 것으로 조화진동 기법의 안정성이 암시적 기법이나, 야코비 반복법에 의한 암시적 기법의 근사해를 사용하는 것보다 훨씬 떨어지는 것을 확인할 수 있다. 하지만, 이 조화진동 기법과 야코비 반복법의 대각행렬만 사용하는 방법을 합성한 하이브리드 기법의 경우 암시적 기법에 비해서도 더 나은 안정성을 보임을 확인할 수 있다. 암시적 기법의 경우 나타난 $\kappa_{\max}$가 시스템이 실패하는 지점이 아니라 경직도가 높은 스프링 때문에 모델이 접히어 펼쳐지지 않는 이상동작이 나타나는 시점으로 설정했다. 이러한 상황은 그림 5와 같은 경우이다. 암시적 기법의 경우 경직도를 크게 높일 경우 그림 5 (a)와 같이 시뮬레이션 자체가 실패하는 경우도 있지만, (b)와 같이 스프링 모델이 접혀 더 이상 동작하지 않는 상황이 나타난다. 우리는 이 상황도 경직도 한계를 넘은 것으로 간주하였다.

표 2는 각각의 기법이 주어진 입자수의 모델을 다룰 때에 상태를 갱신하는 데에 소요되는 시간을 비교한 것이다. 하이브리드 기법은 야코비 반복이나 조화진동 기법, 그리고 오일러 기법처럼 명시적 기법의 상태 갱신과 같은 수준의 연산을 필요로 하기 때문에 표에 나타난 바와 같은 효율적 성능을 보인다.



(a) 불안정성으로 시뮬레이션 실패　　(b) 모델이 접혀 동작하지 않는 경우

[그림 5. 불안정성이 발생한 결과]

알고리즘 6은 하이브리드 기법의 구현을 위한 의사코드이다.

알고리즘 6. 조화진동과 근사 암시적 기법의 하이브리드 기법

Function Approx_Implicit_Integration

n: 질점 수, e: 스프링 수, κ: 스프링 상수, h: 시간간격

$\mathbf{v}$: 속도, $\mathbf{x}$: 위치, $\mathbf{f}$: 힘, m: 질량, $\mathbf{s}$: 스프링 정보, l_0: 휴지상태 길이

```
01  A_diag ← array(n, element: R^3×3) : initialize with 0
02  J_diag ← array(n, element: R^3×3) : initialize with 0
03  J_ij ← array(e, element: R^3×3) : initialize with 0
04  φ ← array(n, element: R^3) : initialize with 0
05
06  for each spring from 0 to e−1 :
07      (i,j) = s[spring]
08      x_ji = x[j] − x[i]
09      compute δ_ij and δ̇_ij with x_ji and l_0[spring]
10      compute A_ij and T_0 with δ_ij and δ̇_ij
11      compute ξ_ij, α_ij, ω_ij^d
12      compute φ_ij with ξ_ij, α_ij, ω_ij^d
13      x_ji = x[j] − x[i]
14      l^t = |x_ji|
15      φ[i] += φ_ij x_ij / 2 l_ji^t
16      φ[j] -= φ_ij x_ij / 2 l_ji^t
17
18      J_ij[spring] = κ*[ ((l^t - l_0[spring])/l^t)*I_33 +
19                  (l_0[spring]/(l^t*l^t*l^t))*outerProduct(x_ji , x_ji)]
20      J_diag[i] −= J_ij[spring] ,  J_diag[j] −= J_ij[spring]
21
22  A_diag[i] = m[i]I_33 + h*hJ_diag[i]  for each i from 0 to n−1
23
24  b ← array(n, element: R^3) : initialize with 0
25  for each i from 0 to n−1 :
26      b[i] = h*f^t[i] + h*h*J_diag[i]*v[i]
27  for each spring from 0 to e−1 :
28      (i,j) = s[spring]
29      b[i] = b[i] + h*h*J_ij[spring]*v[i]
30      b[j] = b[j] + h*h*J_ij[spring]*v[j]
31  dV ← array(n, element: R^3) : initialize with 0
32  for each i from 0 to n−1 :
33      b[i] = (b[i] + φ[i])/2
34
35  for each i from 0 to n−1 :
36      dv[i] = A_diag[i]^−1 * b[i]
37      v[i] = v[i] + dV[i]
38      x[i] = x[i] + h*v[i]
```