PHYSICS ENGINE

Homepage: https://sites.google.com/site/doc4code/

Email: sj6219@hotmail.com

2011/09/19

물리엔진이 하는 일은 거대한 연립 방정식을 풀어서, 그 해를 찾아내고, 그 결과를 게임에 적용하는 것이다.

물리 엔진을 이해하려면 수학과 물리학에 대한 기초지식이 필요하다. 그래서, 처음에 시작할 엄두를 내지 못하는 경우가 많다. 하지만, 실제 공부해보면 그리 어려운 수학이 필요하지 않다. 공업수학 중 벡터와 행렬에 대해 공부했다면, 나머지는 몰라도 괜찮다.

물리엔진의 원리를 이해하기는 어렵지만, 실제 구현은 더 간단하다. 단, 충돌 부분의 알고리즘은 다른 부분보다 더 복잡하다. 이 문서에서는 충돌 부분에 대한 자세한 설명을 생략하고, 충돌 부분을 제외한 물리 엔진에 대해서 설명하겠다. 충돌 알고리즘은 [9] Gino van den Bergen 에 매우 잘 설명되어 있어서, 이 책을 읽는 게 더 나을 것이다.

목차

1 KINEMATICS(물리 운동학)

여기서는 Rigid Body(강체)의 운동에 대해서 알아보자. 주로 고등학교 때 배운 물리 법칙은 선형 운동에 대해서만 다루는데, 실제 운동학에서는 회전 운동에 대해서도 계산해 주어야 한다.

선형 운동과 회전 운동은 비슷하게 대응되는 부분이 많기 때문에, 서로 비교해 가면서 공부하면 좋다.

MASS(질량) & CENTER OF MASS(무게중심)

무게 중심은 운동학에서 그 물체의 기준이 되는 점이다.

물체가 N 개의 입자로 이루어져 있고,

i 번째 입자의 질량을 m_i 라고 하면 전체 질량은

$$m = \sum_i m_i$$

이다.

입자의 위치를 $\mathbf{x}_i$라고 할 때, 무게 중심은

$$\mathbf{x} = \frac{\sum_i m_i \mathbf{x}_i}{m}$$

(식 1-1)

로 정의한다.

LINEAR VELOCITY(속도)

물체의 무게 중심을 시간에 대해서 미분한 것을 linear velocity 라고 정의한다.

$$\mathbf{v}(t) = \dot{\mathbf{x}}(t)$$

SKEW MATRIX

어떤 $\mathbf{d} = \begin{bmatrix} d_x \\ d_y \\ d_z \end{bmatrix}$에 대하여

$$\text{Skew}(\mathbf{d}) = \begin{bmatrix} 0 & -d_z & d_y \\ d_z & 0 & -d_x \\ -d_y & d_x & 0 \end{bmatrix}$$

(식 1-2)

로 정의한다.

임의의 3 차원 벡터 $\mathbf{r} = \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix}$에 대하여,

$$\text{Skew}(\mathbf{d})\,\mathbf{r} = \mathbf{d} \times \mathbf{r}$$

(식 1-3)

이 성립한다. 좌변은 3x3 행렬과 3x1 행렬의 곱이고, 우변은 3 차원 벡터와 3 차원 벡터의 cross product 이다.

그리고, 회전축이 $\mathbf{d} = \begin{bmatrix} d_x \\ d_y \\ d_z \end{bmatrix}$ 이고, 회전각이 θ 인 경우에 회전 행렬 R 은

$$R = E + \sin(\theta)\ \text{Skew}(\mathbf{d}) + (1 - \cos\theta)\text{Skew}(\mathbf{d})^2$$

$$E = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

(식 1-4)

로 계산할 수 있다. 다음 절에서 (식 1-4)을 증명하겠다.

ROTATION MATRIX(회전 행렬)

회전 행렬을 구하는 방법에 대해서 알아보자.

우선 회전축이 z 축이고 회전각이 θ 인 경우부터 계산해 보자.

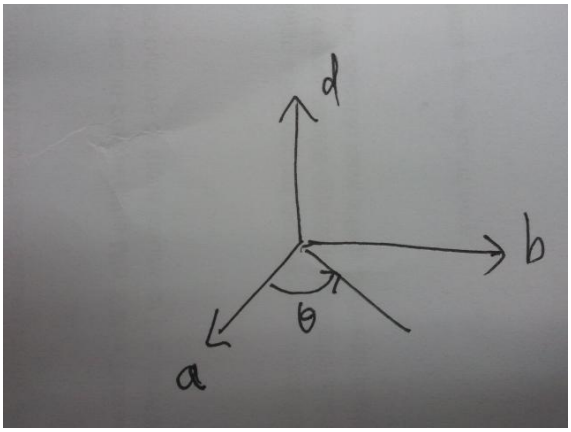
고등학교 때 배운 수학을 이용하면

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

(식 1-5)

이다.

이번에는 회전축이 $\mathbf{d} = \begin{bmatrix} d_x \\ d_y \\ d_z \end{bmatrix}$ 이고, 회전각이 θ 인 경우에 대해서 알아보자.



(그림 1-1)

(그림 1-1)에서 3 차원 벡터 $\mathbf{a}, \mathbf{b}, \mathbf{d}$ 는 서로 직각이고 크기가 1 이다.

일단 좌표 계를 $\mathbf{a}, \mathbf{b}, \mathbf{d}$ 로 바꾼 후, 회전시킨 다음, 원래 좌표 계로 변환하면 된다.

회전 행렬(Rotation Matrix)은

$$R = [\mathbf{a} \quad \mathbf{b} \quad \mathbf{d}] \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} [\mathbf{a} \quad \mathbf{b} \quad \mathbf{d}]^{-1}$$

(식 1-6)

위에서, $\mathbf{a} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}$ 이고 $\mathbf{b} = \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix}$ 일 때, $\begin{bmatrix} a_x & b_x & d_x \\ a_y & b_y & d_y \\ a_z & b_z & d_z \end{bmatrix}$ 를 $[\mathbf{a} \quad \mathbf{b} \quad \mathbf{d}]$ 로 간단히 표기하였다.

그런데, $[\mathbf{a} \quad \mathbf{b} \quad \mathbf{d}]$ 도 회전행렬이기 때문에 $[\mathbf{a} \quad \mathbf{b} \quad \mathbf{d}]^{-1} = \begin{bmatrix} \mathbf{a}^T \\ \mathbf{b}^T \\ \mathbf{d}^T \end{bmatrix}$ 이다.

마찬가지로, $\begin{bmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ d_x & d_y & d_z \end{bmatrix}$ 를 $\begin{bmatrix} \mathbf{a^T} \\ \mathbf{b^T} \\ \mathbf{d^T} \end{bmatrix}$ 로 표기했다. 그래서

$$R = [\mathbf{a} \quad \mathbf{b} \quad \mathbf{d}] \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{a^T} \\ \mathbf{b^T} \\ \mathbf{d^T} \end{bmatrix}$$

$$= [\mathbf{a} \quad \mathbf{b} \quad \mathbf{d}] \begin{bmatrix} \cos(\theta)\,\mathbf{a^T} - \sin(\theta)\,\mathbf{b^T} \\ \sin(\theta)\,\mathbf{a^T} + \cos(\theta)\,\mathbf{b^T} \\ \mathbf{d^T} \end{bmatrix}$$

$$= \mathbf{a}(\cos(\theta)\,\mathbf{a^T} - \sin(\theta)\,\mathbf{b^T}) + \mathbf{b}(\sin(\theta)\mathbf{a^T} + \cos(\theta)\,\mathbf{b^T}) + \mathbf{d}\mathbf{d^T}$$

$$= \cos(\theta)\,(\mathbf{a}\mathbf{a^T} + \mathbf{b}\mathbf{b^T}) + \sin(\theta)\,(\mathbf{b}\mathbf{a^T} - \mathbf{a}\mathbf{b^T}) + \mathbf{d}\mathbf{d^T}$$

(식 1-7)

그런데,

$$E = [\mathbf{a} \quad \mathbf{b} \quad \mathbf{d}][\mathbf{a} \quad \mathbf{b} \quad \mathbf{d}]^{-1}$$

$$= [\mathbf{a} \quad \mathbf{b} \quad \mathbf{d}] \begin{bmatrix} \mathbf{a^T} \\ \mathbf{b^T} \\ \mathbf{d^T} \end{bmatrix}$$

$$= \mathbf{a}\mathbf{a^T} + \mathbf{b}\mathbf{b^T} + \mathbf{d}\mathbf{d^T}$$

(식 1-8)

이고, 임의의 3 차원 벡터 $\mathbf{v}$ 에 대해서

$$\mathbf{v} = E\mathbf{v}$$

$$= [\mathbf{a} \quad \mathbf{b} \quad \mathbf{d}] \begin{bmatrix} \mathbf{a^T} \\ \mathbf{b^T} \\ \mathbf{d^T} \end{bmatrix} \mathbf{v}$$

$$= [\mathbf{a} \quad \mathbf{b} \quad \mathbf{d}] \begin{bmatrix} \mathbf{a^T}\mathbf{v} \\ \mathbf{b^T}\mathbf{v} \\ \mathbf{d^T}\mathbf{v} \end{bmatrix}$$

$$= [\mathbf{a} \quad \mathbf{b} \quad \mathbf{d}] \begin{bmatrix} \mathbf{a} \cdot \mathbf{v} \\ \mathbf{b} \cdot \mathbf{v} \\ \mathbf{d} \cdot \mathbf{v} \end{bmatrix}$$

$$= (\mathbf{a} \cdot \mathbf{v})\mathbf{a} + (\mathbf{b} \cdot \mathbf{v})\mathbf{b} + (\mathbf{d} \cdot \mathbf{v})\mathbf{d}$$

(식 1-9)

로 표현할 수 있다.

그리고,

$$\mathbf{d} \times \mathbf{v} = \mathbf{d} \times \big((\mathbf{a} \cdot \mathbf{v})\mathbf{a} + (\mathbf{b} \cdot \mathbf{v})\mathbf{b} + (\mathbf{d} \cdot \mathbf{v})\mathbf{d}\big) = (\mathbf{a} \cdot \mathbf{v})\,(\mathbf{d} \times \mathbf{a}) + (\mathbf{b} \cdot \mathbf{v})\,(\mathbf{d} \times \mathbf{b}) + (\mathbf{d} \cdot \mathbf{v})\,(\mathbf{d} \times \mathbf{d})$$

(식 1-10)

여기서 $\mathbf{d}\times\mathbf{a}=\mathbf{b},\mathbf{d}\times\mathbf{b}=-\mathbf{a},\mathbf{d}\times\mathbf{d}=\mathbf{0}$ 이므로

$$\mathbf{d}\times\mathbf{v}=(\mathbf{a}\cdot\mathbf{v})\,\mathbf{b}-(\mathbf{b}\cdot\mathbf{v})\,\mathbf{a}$$

(식 1-11)

이 성립한다.

또한, (식 1-2)에 의해

$$\begin{aligned}\mathrm{Skew}(\mathbf{d})\mathbf{v}&=\mathbf{d}\times\mathbf{v}\\&=(\mathbf{a}\cdot\mathbf{v})\,\mathbf{b}-(\mathbf{b}\cdot\mathbf{v})\,\mathbf{a}\\&=\mathbf{b}\mathbf{a}^{\mathbf{T}}\mathbf{v}-\mathbf{a}\mathbf{b}^{\mathbf{T}}\mathbf{v}\end{aligned}$$

(식 1-12)

이다.

(식 1-12)는 모든 $\mathbf{v}$ 에 대해 성립하므로

$$\mathrm{Skew}(\mathbf{d})=\mathbf{b}\mathbf{a}^{\mathbf{T}}-\mathbf{a}\mathbf{b}^{\mathbf{T}}$$

(식 1-13)

이다. 그 다음

$$\mathrm{Skew}(\mathbf{d})^2\mathbf{v}=\mathbf{d}\times(\mathbf{d}\times\mathbf{v})$$

(식 1-11) 적용

$$\begin{aligned}&=\mathbf{d}\times\big((\mathbf{a}\cdot\mathbf{v})\,\mathbf{b}-(\mathbf{b}\cdot\mathbf{v})\,\mathbf{a}\big)\\&=(\mathbf{a}\cdot\mathbf{v})\,(\mathbf{d}\times\mathbf{b})-(\mathbf{b}\cdot\mathbf{v})(\mathbf{d}\times\mathbf{a})\\&=-(\mathbf{a}\cdot\mathbf{v})\,\mathbf{a}-(\mathbf{b}\cdot\mathbf{v})\mathbf{b}\\&=-\mathbf{a}^{\mathbf{T}}\mathbf{v}-\mathbf{b}\mathbf{b}^{\mathbf{T}}\mathbf{v}\\&=(-\mathbf{a}\mathbf{a}^{\mathbf{T}}-\mathbf{b}\mathbf{b}^{\mathbf{T}})\mathbf{v}\end{aligned}$$

(식 1-8) 적용

$$=(\mathbf{d}\mathbf{d}^{\mathbf{T}}-\mathbf{E})\mathbf{v}$$

(식 1-14)

마찬가지로,

$$\text{Skew}(\mathbf{d})^2 = \mathbf{dd^T} - \mathbf{E}$$

(식 1-15)

다시 **(식 1-7)**로 돌아가서,

$$\mathrm{R} = \cos(\theta)\,(\mathbf{aa^T} + \mathbf{bb^T}) + \sin(\theta)\,(\mathbf{ba^T} - \mathbf{ab^T}) + \mathbf{dd^T}$$

(식 1-8) 적용

$$= \cos(\theta)\,(\mathbf{E} - \mathbf{dd^T}) + \sin(\theta)\,(\mathbf{ba^T} - \mathbf{ab^T}) + \mathbf{dd^T}$$

(식 1-13) 적용

$$= \cos(\theta)\,(\mathbf{E} - \mathbf{dd^T}) + \sin(\theta)\,\text{Skew}(\mathbf{d}) + \mathbf{dd^T}$$

(식 1-15) 적용

$$= -\cos(\theta)\,\text{Skew}(\mathbf{d})^2 + \sin(\theta)\,\text{Skew}(\mathbf{d}) + \text{Skew}(\mathbf{d})^2 + \mathrm{E}$$

$$= \mathrm{E} + \sin(\theta)\,\text{Skew}(\mathbf{d}) + (1 - \cos(\theta))\text{Skew}(\mathbf{d})^2$$

이제 **(식 1-4)**가 증명되었다.

QUATERNION(사원수)

Quaternion 은 회전 행렬을 다른 방식으로 간단히 표현하기 위해서 사용된다.

Quaternion 은 4 개의 원소로 이루어진다.

$\mathrm{q} = \mathrm{w} + \mathrm{x}\mathbf{i} + \mathrm{y}\mathbf{j} + \mathrm{z}\mathbf{k}$ 에서 $\mathrm{w}, \mathrm{x}, \mathrm{y}, \mathrm{z}$는 실수(real)이다. $(\mathrm{w}, \mathrm{x}, \mathrm{y}, \mathrm{z})$ 의 4 차원 벡터로 표현되기도 한다.

Quaternion 의 합은

$$\begin{aligned} \mathrm{q}_0 + \mathrm{q}_1 &= (\mathrm{w}_0 + \mathrm{x}_0\mathbf{i} + \mathrm{y}_0\mathbf{j} + \mathrm{z}_0\mathbf{k}) + (\mathrm{w}_1 + \mathrm{x}_1\mathbf{i} + \mathrm{y}_1\mathbf{j} + \mathrm{z}_1\mathbf{k}) \\ &= (\mathrm{w}_0 + \mathrm{w}_1) + (\mathrm{x}_0 + \mathrm{x}_1)\mathbf{i} + (\mathrm{y}_0 + \mathrm{y}_1)\mathbf{j} + (\mathrm{z}_0 + \mathrm{z}_1)\mathbf{k} \end{aligned}$$

(식 1-16)

으로 정의된다.

또, Quaternion 의 곱은

$$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = -1$$

$$\mathbf{ij} = -\mathbf{ji} = \mathbf{k}$$

$$\mathbf{jk} = -\mathbf{kj} = \mathbf{i}$$

$$\mathbf{ki} = -\mathbf{ik} = \mathbf{j}$$

를 이용해 다음과 같이 정의된다.

$$\begin{aligned} q_0 q_1 &= (w_0 + x_0\mathbf{i} + y_0\mathbf{j} + z_0\mathbf{k})(w_1 + x_1\mathbf{i} + y_1\mathbf{j} + z_1\mathbf{k}) \\ &= (w_0 w_1 - x_0 x_1 - y_0 y_1 - z_0 z_1) \\ &+ (w_0 x_1 + w_1 x_0 + y_0 z_1 - z_0 y_1)\mathbf{i} \\ &+ (w_0 y_1 + w_1 y_0 + z_0 x_1 - x_0 z_1)\mathbf{j} \\ &+ (w_0 z_1 + w_1 z_0 + x_0 y_1 - y_0 x_1)\mathbf{k} \end{aligned}$$

(식 1-17)

그리고, 3 차원 벡터 $\mathbf{v} = (v_x\ v_y\ v_z)$ 에 대하여 $\hat{\mathbf{v}} = v_x\mathbf{i} + v_y\mathbf{j} + v_z\mathbf{k}$ 으로 표시한다.

그러면, quaternion 의 곱은

$$(w_0 + \hat{\mathbf{v}_0})(w_1 + \hat{\mathbf{v}_1}) = (w_0 w_1 - \mathbf{v}_0 \cdot \mathbf{v}_1) + w_0\hat{\mathbf{v}_1} + w_1\hat{\mathbf{v}_0} + \widehat{\mathbf{v}_0 \times \mathbf{v}_1}$$

(식 1-18)

로 표시된다.

그리고, $q = w + \hat{\mathbf{v}}$ 에 대하여, $q^* = w - \hat{\mathbf{v}}$ 를 conjugate 라고 부르고, q^* 로 표시한다.

QUATERNION 와 회전행렬과의 관계

Quaternion 은 회전 행렬을 간단히 표현할 때 자주 사용된다.

Quaternion q 가

$$q = \cos\left(\frac{\theta}{2}\right) + \hat{\mathbf{d}}\sin\left(\frac{\theta}{2}\right)$$

(식 1-19)

라고 하자. 여기서 $\mathbf{d}$ 는 길이가 1 인 3 차원벡터이다.

그러면, $q\hat{\mathbf{v}}q^*$의 실수 부분의 값은 항상 0 이다. 그리고, R 이 회전축이 $\mathbf{d}$ 이고 회전각이 θ 일 때의

회전행렬이고, 3 차원벡터 $\mathbf{u}$ 가 $\hat{\mathbf{u}} = q\hat{\mathbf{v}}q^*$ 라면, $\mathbf{u} = R\mathbf{v}$인 성질이 있다. 이제 증명해보자.

$q = \cos\left(\frac{\theta}{2}\right) + \sin\left(\frac{\theta}{2}\right) d_x\mathbf{i} + \sin\left(\frac{\theta}{2}\right) d_y\mathbf{j} + \sin\left(\frac{\theta}{2}\right) d_z\mathbf{k} = w + x\mathbf{i} + y\mathbf{j} + z\mathbf{k}$, $\hat{\mathbf{v}} = v_x\,\mathbf{i} + v_y\,\mathbf{j} + v_z\mathbf{k}$ 라고 하면,

$$q\hat{\mathbf{v}}q^* = (w + x\mathbf{i} + y\mathbf{j} + z\mathbf{k})(\,v_x\,\mathbf{i} + v_y\,\mathbf{j} + v_z\mathbf{k})(w - x\mathbf{i} - y\mathbf{j} - z\mathbf{k})$$

$$= \left((-xv_x - yv_y - zv_z) + (wv_x + yv_z - zv_y)\mathbf{i} + (wv_y + zv_x - xv_z)\mathbf{j} + (wv_z + xv_y - yv_x)\mathbf{k}\right)(w - x\mathbf{i} - y\mathbf{j} - z\mathbf{k})$$

$$= (-wxv_x - wyv_y - wzv_z + wxv_x + xyv_z - xzv_y + wyv_y + yzv_x - xyv_z + wzv_z + xzv_y - yzv_x)$$

$$+ \left((x^2v_x + xyv_y + xzv_z) + (w^2v_x + wyv_z - wzv_y) + (-wzv_y - z^2v_x + xzv_z) + (wyv_z + xyv_y - y^2v_x)\right)\mathbf{i}$$

$$+ \left((xyv_x + y^2v_y + yzv_z) + (w^2v_y + wzv_x - wxv_z) + (-wxv_z - x^2v_y + xyv_x) + (wzv_x + yzv_z - z^2v_y)\right)\mathbf{j}$$

$$+ \left((xzv_x + yzv_y + z^2v_z) + (w^2v_z + wxv_y - wyv_x) + (-wyv_x - y^2v_z + yzv_y) + (wxv_y + xzv_x - x^2v_z)\right)\mathbf{k}$$

$$= (0)$$

$$+ \left((x^2 + w^2 - z^2 - y^2)v_x + (2xy - 2wz)v_y + (2xz + 2wy)v_z\right)\mathbf{i}$$

$$+ \left((2xy + 2wz)v_x + (y^2 + w^2 - x^2 - z^2)v_y + (2yz - 2wx)v_z\right)\mathbf{j}$$

$$+ \left((2xz - 2wy)v_x + (2yz + 2wx)v_y + (z^2 + w^2 - y^2 - x^2)v_z\right)\mathbf{k}$$

여기서, $q\hat{\mathbf{v}}q^*$의 실수 부분의 값은 0 임을 알 수 있다.

또한, $w^2 + x^2 + y^2 + z^2 = 1$ 이기 때문에

$$q\hat{\mathbf{v}}q^* = \hat{\mathbf{u}}$$

$$= \left((1 - 2y^2 - 2z^2)v_x + (2xy - 2wz)v_y + (2xz + 2wy)v_z\right)\mathbf{i}$$

$$+ \left((2xy + 2wz)v_x + (1 - 2x^2 - 2z^2)v_y + (2yz - 2wx)v_z\right)\mathbf{j}$$

$$+ \left((2xz - 2wy)v_x + (2yz + 2wx)v_y + (1 - 2x^2 - 2y^2)v_z\right)\mathbf{k}$$

(식 1-20)

여기서,

$$R' = \begin{bmatrix} 1 - 2y^2 - 2z^2 & 2xy - 2wz & 2xz + 2wy \\ 2xy + 2wz & 1 - 2x^2 - 2z^2 & 2yz - 2wx \\ 2xz - 2wy & 2yz + 2wx & 1 - 2x^2 - 2y^2 \end{bmatrix}$$

(식 1-21)

으로 표시하면

$$\mathbf{u} = \mathrm{R}'\mathbf{v}$$

가 된다. 이제 R′가 회전행렬 R 과 같다는 것을 증명하기만 하면 된다.

(식 1-4) 로 부터

$$\mathrm{R} = \mathrm{E} + \sin(\theta)\ \mathrm{Skew}(\mathbf{d}) + (1 - \cos(\theta))\mathrm{Skew}(\mathbf{d})^2$$

$$= \mathrm{E} + \sin(\theta)\begin{bmatrix} 0 & -\mathrm{d_z} & \mathrm{d_y} \\ \mathrm{d_z} & 0 & -\mathrm{d_x} \\ -\mathrm{d_y} & \mathrm{d_x} & 0 \end{bmatrix} + (1 - \cos(\theta))\begin{bmatrix} -\mathrm{d_y}^2 - \mathrm{d_z}^2 & \mathrm{d_x d_y} & \mathrm{d_x d_z} \\ \mathrm{d_x d_y} & -\mathrm{d_x}^2 - \mathrm{d_z}^2 & \mathrm{d_y d_z} \\ \mathrm{d_x d_z} & \mathrm{d_y d_z} & -\mathrm{d_x}^2 - \mathrm{d_y}^2 \end{bmatrix}$$

$\sin\theta = 2\sin\left(\frac{\theta}{2}\right)\cos\left(\frac{\theta}{2}\right), 1 - \cos\theta = 2\sin^2\left(\frac{\theta}{2}\right)$ 를 이용하면

$$= \mathrm{E} + \begin{bmatrix} 0 & -2\sin\left(\frac{\theta}{2}\right)\cos\left(\frac{\theta}{2}\right)\mathrm{d_z} & 2\sin\left(\frac{\theta}{2}\right)\cos\left(\frac{\theta}{2}\right)\mathrm{d_y} \\ 2\sin\left(\frac{\theta}{2}\right)\cos\left(\frac{\theta}{2}\right)\mathrm{d_z} & 0 & -2\sin\left(\frac{\theta}{2}\right)\cos\left(\frac{\theta}{2}\right)\mathrm{d_x} \\ -2\sin\left(\frac{\theta}{2}\right)\cos\left(\frac{\theta}{2}\right)\mathrm{d_y} & 2\sin\left(\frac{\theta}{2}\right)\cos\left(\frac{\theta}{2}\right)\mathrm{d_x} & 0 \end{bmatrix}$$

$$+ \begin{bmatrix} -2\sin^2\left(\frac{\theta}{2}\right)\mathrm{d_y}^2 - 2\sin^2\left(\frac{\theta}{2}\right)\mathrm{d_z}^2 & 2\sin^2\left(\frac{\theta}{2}\right)\mathrm{d_x d_y} & 2\sin^2\left(\frac{\theta}{2}\right)\mathrm{d_x d_z} \\ 2\sin^2\left(\frac{\theta}{2}\right)\mathrm{d_x d_y} & -2\sin^2\left(\frac{\theta}{2}\right)\mathrm{d_x}^2 - 2\sin^2\left(\frac{\theta}{2}\right)\mathrm{d_z}^2 & 2\sin^2\left(\frac{\theta}{2}\right)\mathrm{d_y d_z} \\ 2\sin^2\left(\frac{\theta}{2}\right)\mathrm{d_x d_z} & 2\sin^2\left(\frac{\theta}{2}\right)\mathrm{d_y d_z} & -2\sin^2\left(\frac{\theta}{2}\right)\mathrm{d_x}^2 - 2\sin^2\left(\frac{\theta}{2}\right)\mathrm{d_y}^2 \end{bmatrix}$$

$\cos\left(\frac{\theta}{2}\right) = \mathrm{w},\ \sin\left(\frac{\theta}{2}\right)\mathrm{d_x} = \mathrm{x},\ \sin\left(\frac{\theta}{2}\right)\mathrm{d_y} = \mathrm{y},\ \sin\left(\frac{\theta}{2}\right)\mathrm{d_z} = \mathrm{z}$를 대입하면

$$= \mathrm{E} + \begin{bmatrix} 0 & -2\mathrm{wz} & 2\mathrm{wy} \\ 2\mathrm{wz} & 0 & -2\mathrm{wx} \\ -2\mathrm{wy} & 2\mathrm{wx} & 0 \end{bmatrix} + \begin{bmatrix} -2\mathrm{y}^2 - 2\mathrm{z}^2 & 2\mathrm{xy} & 2\mathrm{xz} \\ 2\mathrm{xy} & -2\mathrm{x}^2 - 2\mathrm{z}^2 & 2\mathrm{yz} \\ 2\mathrm{xz} & 2\mathrm{yz} & -2\mathrm{x}^2 - 2\mathrm{y}^2 \end{bmatrix}$$

$$= \begin{bmatrix} 1 - 2\mathrm{y}^2 - 2\mathrm{z}^2 & 2\mathrm{xy} - 2\mathrm{wz} & 2\mathrm{xz} + 2\mathrm{wy} \\ 2\mathrm{xy} + 2\mathrm{wz} & 1 - 2\mathrm{x}^2 - 2\mathrm{z}^2 & 2\mathrm{yz} - 2\mathrm{wx} \\ 2\mathrm{xz} - 2\mathrm{wy} & 2\mathrm{yz} + 2\mathrm{wx} & 1 - 2\mathrm{x}^2 - 2\mathrm{y}^2 \end{bmatrix} = \mathrm{R}'$$

(식 1-22)

증명 끝!!!

ANGULAR VELOCITY(각속도)

i 번째 입자의 무게 중심 $\mathbf{x}$ 에 대한 상대위치를 $\mathbf{r}_i$ 로 표현하면, $\mathbf{r}_i$ 는 시간에 대한 함수로 표현할 수 있다.

$$\mathbf{r}_i(t) = \mathbf{x}_i(t) - \mathbf{x}(t)$$

(식 1-23)

회전 행렬을 $R(t)$로 표시하면,

$$\mathbf{r}_i(t) = R(t)\,\mathbf{r}_i(0)$$

(식 1-24)

또, i 번째 입자의 위치 $\mathbf{x}_i(t)$ 는

$$\mathbf{x}_i(t) = \mathbf{x}(t) + R(t)\,\mathbf{r}_i(0)$$

(식 1-25)

가 된다.

(식 1-24)를 시간에 대하여 미분하면

$$\dot{\mathbf{r}}_i(t) = \dot{R}(t)\mathbf{r}_i(0) = \dot{R}(t)R^T(t)\,R(t)\mathbf{r}_i(0)$$

(식 1-24) 적용

$$= \dot{R}(t)R^T(t)\mathbf{r}_i(t)$$

(식 1-26)

그런데,

$$R(t)R^T(t) = E$$

이다.

양변을 미분하면,

$$\dot{R}(t)R^T(t) + R(t)\dot{R^T}(t) = 0$$

이기 때문에

$$\left(\dot{R}(t)R^T(t)\right)^T = R(t)\dot{R^T}(t) = -\dot{R}(t)R^T(t)$$

그러므로, $\dot{R}(t)R^T(t)$ 는 skew-symmetric 행렬인 것을 알 수 있다. 즉

$$\mathrm{Skew}\big(\boldsymbol{\omega}(t)\big) = \dot{R}(t)R^{T}(t)$$

(식 1-27)

을 만족하는 $\boldsymbol{\omega}(t)$가 존재한다. 이 $\boldsymbol{\omega}(t)$를 angular velocity 라고 정의한다.

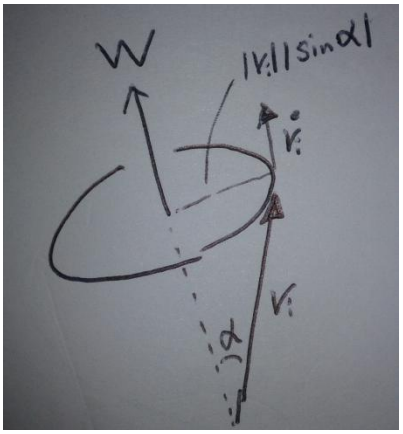
다시, (식 1-26)에 대입하면,

$$\dot{\mathbf{r}}_i(t) = \mathrm{Skew}\big(\boldsymbol{\omega}(t)\big)\, r_i(t) = \boldsymbol{\omega}(t) \times \mathbf{r}_i(t)$$

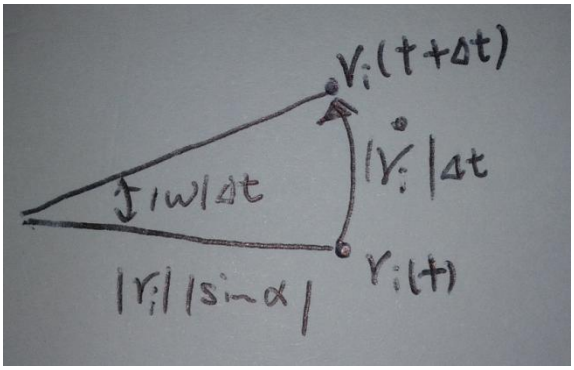
(식 1-28)

가 성립한다.

$\mathbf{r}_i(t)$ 와 $\boldsymbol{\omega}(t)$가 이루는 각을 $\alpha(t)$라고 하면



(그림 1-2)



(그림 1-3)

(그림 1-2)와 (그림 1-3)를 (식 1-28)과 비교해서 보면,

$$|\dot{\mathbf{r}}_i(t)| = \left|\sin(\alpha(t))\right| |\mathbf{r}_i(t)| \, |\boldsymbol{\omega}(t)|$$

이라는 것을 알 수 있다.

그러므로, 회전 축은 $\frac{\boldsymbol{\omega}(t)}{|\boldsymbol{\omega}(t)|}$ 이고, 회전 각속도는 $|\boldsymbol{\omega}(t)|$가 된다.

기하학적인 해석 대신에, 대수적인 방법으로 다시 증명해 보자.

이제, 회전축의 방향을 $\mathbf{d}(t) = \begin{bmatrix} d_x(t) \\ d_y(t) \\ d_z(t) \end{bmatrix}$ 라 하고, 회전 각속도를 $\theta(t)$라고 하자.

$$\mathrm{Skew}(\boldsymbol{\omega}(t)) = \dot{R}(t)R^T(t)$$

$$= \left(\lim_{h\to 0}\left(\frac{R(t+h) - R(t)}{h}\right)\right) R^T(t)$$

그런데, **(식 1-4)**를 적용하면, h 가 0 에 가까울 때,

$R(t+h) = \left(E + \sin(\theta(t)\,h)\,\mathrm{Skew}(\mathbf{d}(t)) + (1 - \cos(\theta(t)\,h))\mathrm{Skew}(\mathbf{d}(t))^2\right) R(t)$ 이므로

$$= \lim_{h\to 0}\left(\frac{\left(E + \sin(\theta(t)\,h)\,\mathrm{Skew}(\mathbf{d}(t)) + (1 - \cos(\theta(t)\,h))\mathrm{Skew}(\mathbf{d}(t))^2\right) R(t) - R(t)}{h}\right) R^T(t)$$

$$= \lim_{h\to 0}\left(\frac{\left(\sin(\theta(t)\,h)\,\mathrm{Skew}(\mathbf{d}(t)) + (1 - \cos(\theta(t)\,h))\mathrm{Skew}(\mathbf{d}(t))^2\right)}{h}\right) R(t)R^T(t)$$

L'Hôpital 의 정리를 적용하면

$$= \lim_{h\to 0}\left(\theta(t)\cos(\theta(t)\,h)\,\mathrm{Skew}(\mathbf{d}(t)) + \theta(t)\sin(\theta(t)\,h)\,\mathrm{Skew}(\mathbf{d}(t))^2\right)$$

$$= \theta(t)\,\mathrm{Skew}(\mathbf{d}(t))$$

$$= \mathrm{Skew}(\theta(t)\,\mathbf{d}(t))$$

그러므로,

$$\mathrm{Skew}(\boldsymbol{\omega}(t)) = \mathrm{Skew}(\theta(t)\,\mathbf{d}(t))$$

$$\boldsymbol{\omega}(t) = \theta(t)\,\mathbf{d}(t)$$

이다.

즉, $\boldsymbol{\omega}$ 의 방향은 회전축과 방향이 같고, 그 크기는 회전 각속도와 같다.

DERIVATIVES OF QUATERNION

이제 Quaternion 을 미분해보자.

$$q(t) = \cos\left(\frac{\theta(t)}{2}\right) + \widehat{\mathbf{d}(t)}\sin\left(\frac{\theta(t)}{2}\right)$$

$q(t)$는 회전축이 $\mathbf{d}(t)$, 각이 $\theta(t)$ 이다. 그리고 $\mathbf{d}(t)$의 크기는 1 이다.

Angular Velocity 를 $\boldsymbol{\omega}(t)$로 표시하면, 회전축의 방향은 $\frac{\boldsymbol{\omega}(t)}{|\boldsymbol{\omega}(t)|}$이고 회전 각속도는 $|\boldsymbol{\omega}(t)|$ 이다. 그런데, $q(t+h)q(t)^*$ 는 $q(t)$를 $q(t+h)$로 회전시키는 사원수이다. 이 회전의 방향을 $\mathbf{v}(t,h)$, 회전각을 $\varphi(t,h)$로 표기하면,

$$q(t+h)q(t)^* = \cos\left(\frac{\varphi(t,h)}{2}\right) + \widehat{\mathbf{v}(t,h)}\sin\left(\frac{\varphi(t,h)}{2}\right)$$

이다. 또

$$\mathbf{v}(t,0) = \frac{\boldsymbol{\omega}(t)}{|\boldsymbol{\omega}(t)|}$$

$$\varphi(t,0) = 0$$

$$\frac{\partial}{\partial h}\varphi(t,0) = |\boldsymbol{\omega}(t)|$$

를 만족한다.

그러므로,

$$\frac{d}{dt}q(t) = \lim_{h\to 0}\left(\frac{q(t+h)-q(t)}{h}\right)q(t)^*\,q(t)$$

$$= \lim_{h\to 0}\left(\frac{\cos\left(\frac{\varphi(t,h)}{2}\right) + \widehat{\mathbf{v}(t,h)}\sin\left(\frac{\varphi(t,h)}{2}\right) - 1}{h}\right)q(t)$$

L'Hôpital 의 정리를 적용하면

$$= \lim_{h\to 0}\left(\left(-\frac{1}{2}\frac{\partial}{\partial h}\varphi(t,h)\sin\left(\frac{\varphi(t,h)}{2}\right) + \frac{\partial}{\partial h}\widehat{\mathbf{v}(t,h)}\sin\left(\frac{\varphi(t,h)}{2}\right) + \frac{\widehat{\mathbf{v}(t,h)}}{2}\frac{\partial}{\partial h}\varphi(t,h)\cos\left(\frac{\varphi(t,h)}{2}\right)\right)q(t)\right)$$

$$= \frac{1}{2}\widehat{\boldsymbol{\omega}(t)}q(t)$$

(식 1-29)

VELOCITY OF PARTICLE

i 번째 입자의 속도 $\mathbf{v}_i(t)$ 를 구해보자.

(식 1-25)로 돌아가자.

$$\mathbf{x}_i(t) = \mathbf{x}(t) + R(t)\,\mathbf{r}_i(0)$$

시간에 대해 미분하면

$$\mathbf{v}_i(t) = \dot{\mathbf{x}}_i(t) = \dot{\mathbf{x}}(t) + \dot{R}(t)\mathbf{r}_i(0)$$

(식 1-28) 적용

무게 중심의 속도 $\dot{\mathbf{x}}(t)$ 를 $\mathbf{v}(t)$ 로 표시하면

$$= \mathbf{v}(t) + \boldsymbol{\omega}(t) \times \mathbf{r}_i(t)$$

$$= \mathbf{v}(t) + \boldsymbol{\omega}(t) \times \big(\mathbf{x}_i(t) - \mathbf{x}(t)\big)$$

(식 1-30)

입자 i 의 속도를 보면 선형 운동으로 인한 $\mathbf{v}(t)$ 와 회전 운동으로 인한 $\boldsymbol{\omega}(t) \times \big(\mathbf{x}_i(t) - \mathbf{x}(t)\big)$ 의 합으로 이루어진다.

FORCE(힘) & TORQUE(돌림 힘)

물체는 힘(Force)에 의해 속도가 변한다.

i 번째 입자에 가해지는 힘을 $\mathbf{F}_i$ 라고 하면

물체 전체에 가해지는 힘 $\mathbf{F}$ 은

$$\mathbf{F} = \sum_i \mathbf{F}_i$$

(식 1-31)

그리고, i 번째 입자에 대해서, Torque $\boldsymbol{\tau}_i$ 를

$$\boldsymbol{\tau}_i = (\mathbf{x}_i - \mathbf{x}) \times \mathbf{F}_i = \mathbf{r}_i \times \mathbf{F}_i$$

(식 1-32)

로 정의한다.

전체 torque $\boldsymbol{\tau}$ 는

$$\boldsymbol{\tau} = \sum_i \boldsymbol{\tau}_i = \sum_i (\mathbf{x}_i - \mathbf{x}) \times \mathbf{F}_i$$

(식 1-33)

INERTIA TENSOR(관성텐서)

i 번째 입자의 무게중심에 대한 상대위치를 $\mathbf{r}_i = (\mathbf{x}_i - \mathbf{x}) = \begin{bmatrix} \alpha_i \\ \beta_i \\ \gamma_i \end{bmatrix}$ 로 표시할 때, 다음과 같이 I 를 그 물체의 Inertia Tensor 로 정의한다.

$$\mathrm{I} = \begin{bmatrix} \mathrm{I}_{xx} & -\mathrm{I}_{xy} & -\mathrm{I}_{xz} \\ -\mathrm{I}_{xy} & \mathrm{I}_{yy} & -\mathrm{I}_{yz} \\ -\mathrm{I}_{xz} & -\mathrm{I}_{yz} & \mathrm{I}_{zz} \end{bmatrix}$$

$$\mathrm{I}_{xx} = \sum_i \mathrm{m}_i \, (\beta_i{}^2 + \gamma_i{}^2)$$

$$\mathrm{I}_{yy} = \sum_i \mathrm{m}_i \, (\alpha_i{}^2 + \gamma_i{}^2)$$

$$\mathrm{I}_{zz} = \sum_i \mathrm{m}_i \, (\alpha_i{}^2 + \beta_i{}^2)$$

$$\mathrm{I}_{xy} = \sum_i \mathrm{m}_i \, \alpha_i \, \beta_i$$

$$\mathrm{I}_{xz} = \sum_i \mathrm{m}_i \, \alpha_i \, \gamma_i$$

$$\mathrm{I}_{yz} = \sum_i \mathrm{m}_i \, \beta_i \, \gamma_i$$

(식 1-34)

그리고,

$$\mathrm{I} = \sum_{\mathrm{i}} \mathrm{m_i}(\mathbf{r_i}^{\mathrm{T}}\mathbf{r_i}\,\mathrm{E} - \mathbf{r_i}\mathbf{r_i}^{\mathrm{T}})$$

(식 1-35)

$$\mathrm{I} = \sum_{\mathrm{i}} -\mathrm{m_i}\mathrm{Skew}(\mathbf{r_i})\,\mathrm{Skew}(\mathbf{r_i})$$

(식 1-36)

로 표시하면 더 편할 때가 있다. E 는 3×3 단위행렬이다.

그런데, Inertia Tensor 는 질량과 같이 일정한 값을 가지는 것이 아니라, 시간이 변하면 그 값도 변한다.

$$\mathrm{I(t)} = \sum_{\mathrm{i}} \mathrm{m_i}(\mathbf{r_i}(\mathrm{t})^{\mathrm{T}}\mathbf{r_i}(\mathrm{t})\,\mathrm{E} - \mathbf{r_i}(\mathrm{t})\mathbf{r_i}(\mathrm{t})^{\mathrm{T}})$$

$\mathbf{r_i}(\mathrm{t}) = \mathrm{R(t)}\mathbf{r_i}(0)$ 을 대입하면

$$\mathrm{I(t)} = \sum_{\mathrm{i}} \mathrm{m_i}(\mathbf{r_i}(0)^{\mathrm{T}}\mathrm{R(t)}^{\mathrm{T}}\mathrm{R(t)}\mathbf{r_i}(0)\mathrm{E} - \mathrm{R(t)}\mathbf{r_i}(0)\mathbf{r_i}(0)^{\mathrm{T}}\mathrm{R(t)}^{\mathrm{T}})$$

$$= \sum_{\mathrm{i}} \mathrm{m_i}(\mathbf{r_i}(0)^{\mathrm{T}}\mathbf{r_i}(0)\mathrm{E} - \mathrm{R(t)}\mathbf{r_i}(0)\mathbf{r_i}(0)^{\mathrm{T}}\mathrm{R(t)}^{\mathrm{T}})$$

$\mathbf{r_i}(0)^{\mathrm{T}}\mathbf{r_i}(0)$는 값이 Scalar 값이므로

$$\mathrm{I(t)} = \sum_{\mathrm{i}} \mathrm{m_i}(\mathrm{R(t)}\mathbf{r_i}(0)^{\mathrm{T}}\mathbf{r_i}(0)\mathrm{E}\,\mathrm{R(t)}^{\mathrm{T}} - \mathrm{R(t)}\mathbf{r_i}(0)\mathbf{r_i}(0)^{\mathrm{T}}\mathrm{R(t)}^{\mathrm{T}})$$

$$= \mathrm{R(t)}\left(\sum_{\mathrm{i}} \mathrm{m_i}(\mathbf{r_i}(0)^{\mathrm{T}}\mathbf{r_i}(0)\mathrm{E} - \mathbf{r_i}(0)\mathbf{r_i}(0)^{\mathrm{T}})\right)\mathrm{R(t)}^{\mathrm{T}}$$

$$= \mathrm{R(t)I(0)R(t)}^{\mathrm{T}}$$

(식 1-37)

(식 1-37)을 이용해서, Inertia Tensor 초기값과 회전 행렬을 이용해 시간에 따라 바뀌는 Inertia Tensor 의 값을 계산할 수 있다.

일반적인 Mesh 를 입력으로 주어졌을 때 무게 중심, Inertia Tensor 를 구하는 알고리즘이 [4] David H. Eberly. 의 2.5.5. 에 설명되어 있다.

https://sites.google.com/site/doc4code/source/PolyhedralMassProperties.pdf 에 내용을 복사해 두었다.

여기서는 꼭 필요한 알고리즘이 아니어서 생략했다. 이 알고리즘의 구현은 간단하지만, 기본 원리가 Green's Theorem 과 Divergence Theorem 를 이용하고 있어서 원리를 이해하기는 어렵다. Green's Theorem 과 Divergence Theorem 은 공업수학 중간쯤에 나온다.

LINEAR MOMENTUM(선운동량)

i 번째 입자에 대해서,

$$\mathbf{P}_i = m_i \mathbf{v}_i$$

와 같이 linear momentum 을 정의한다.

전체 linear momentum 은

$$\mathbf{P} = \sum_i \mathbf{P}_i = \sum_i m_i \mathbf{v}_i$$

(식 1-13) 적용

$$= \sum_i (m_i \mathbf{v} + \boldsymbol{\omega} \times m_i \mathbf{r}_i) = \mathbf{v}\left(\sum_i m_i\right) + \boldsymbol{\omega} \times \left(\sum_i m_i \mathbf{r}_i\right)$$

(식 1-11)에 의해 $\sum_i m_i \mathbf{r}_i = \sum_i \left(m_i(\mathbf{x}_i - \mathbf{x})\right) = (\sum_i m_i \mathbf{x}_i) - \mathbf{x}\sum_i m_i = 0$이므로

$$= m\mathbf{v}$$

(식 1-38)

그리고, 미분한 후, Newton 의 법칙을 적용하면

$$\dot{\mathbf{P}} = \sum_i \dot{\mathbf{P}}_i = \sum_i m_i \dot{\mathbf{v}}_i = \sum_i \mathbf{F}_i$$

그리고, **(식 1-38)**를 미분하면

$$\dot{\mathbf{P}} = m\dot{\mathbf{v}} = m\mathbf{a} = \mathbf{F}$$

(식 1-39)

(식 1-31)과 같아진다.

ANGULAR MOMENTUM(각운동량)

i 번째 입자에 대해서,

$$\mathbf{L}_i = \mathbf{r}_i \times \mathbf{P}_i$$

(식 1-40)

을 Angular Momentum 이라고 정의한다.

Angular Momentum 의 총합 $\mathbf{L}$ 은

$$\mathbf{L} = \sum_i \mathbf{r}_i \times \mathbf{P}_i$$

$$= \sum_i m_i \mathbf{r}_i \times \mathbf{v}_i$$

$$= \sum_i \left(m_i \mathbf{r}_i \times \mathbf{v} + m_i \mathbf{r}_i \times (\boldsymbol{\omega} \times \mathbf{r}_i) \right)$$

$$= \left(\sum_i m_i \mathbf{r}_i \right) \times \mathbf{v} + \left(\sum_i -m_i \mathbf{r}_i \times (\mathbf{r}_i \times \boldsymbol{\omega}) \right)$$

$$= \sum_i (-m_i \mathrm{Skew}(\mathbf{r}_i)\ \mathrm{Skew}(\mathbf{r}_i)\boldsymbol{\omega}) = \left(\sum_i -m_i \mathrm{Skew}(\mathbf{r}_i)\mathrm{Skew}(\mathbf{r}_i) \right) \boldsymbol{\omega}$$

$$= \mathrm{I}\boldsymbol{\omega}$$

(식 1-41)

Angular Momentum 을 미분하면

$$\dot{\mathbf{L}} = \sum_i \frac{d\mathbf{r}_i \times \mathbf{P}_i}{dt}$$

$$= \sum_i (\dot{\mathbf{r}}_i \times \mathbf{P}_i + \mathbf{r}_i \times \dot{\mathbf{P}}_i) = \sum_i (\dot{\mathbf{r}}_i \times m_i \mathbf{v}_i) + \sum_i (\mathbf{r}_i \times \dot{\mathbf{P}}_i)$$

(식 1-42)

앞부분을 먼저 계산하면

$$\sum_i (\dot{\mathbf{r}}_i \times m_i \mathbf{v}_i) = \sum_i \left(\dot{\mathbf{r}}_i \times m_i \frac{d(\mathbf{x} + \mathbf{r}_i)}{dt}\right) = \sum_i (\dot{\mathbf{r}}_i \times m_i (\dot{\mathbf{x}} + \dot{\mathbf{r}}_i)) = \left(\sum_i m_i \dot{\mathbf{r}}_i\right) \times \dot{\mathbf{x}} + \sum_i m_i (\dot{\mathbf{r}}_i \times \dot{\mathbf{r}}_i) = 0$$

위에서 $\sum_i m_i \mathbf{r}_i = 0$ 이므로 $\sum_i m_i \dot{\mathbf{r}}_i = 0$ 이다.

(식 1-42)의 뒷부분을 마저 계산하면

$$\dot{\mathbf{L}} = \sum_i (\mathbf{r}_i \times \dot{\mathbf{P}}_i) = \sum_i (\mathbf{r}_i \times \mathbf{F}_i) = \sum_i \boldsymbol{\tau}_i = \boldsymbol{\tau}$$

(식 1-43)

ANGULAR ACCELERATION(각 가속도)

이제 $\dot{\boldsymbol{\omega}}$ 를 구해보자. (식 1-41)에 의해 $\boldsymbol{\omega} = I^{-1}\mathbf{L}$ 이다. 그러므로,

$$\dot{\boldsymbol{\omega}}(t) = \dot{I^{-1}}(t)\mathbf{L}(t) + I^{-1}(t)\dot{\mathbf{L}}(t)$$

(식 1-44)

(식 1-37)에 의해 $I^{-1}(t) = R(t)I^{-1}(0)R(t)^T$ 이므로

$$\dot{I^{-1}}(t) = \dot{R}(t)I^{-1}(0)R(t)^T + R(t)I^{-1}(0)\dot{R}(t)^T$$

(식 1-45)

$\dot{R}(t) = \text{skew}(\boldsymbol{\omega}(t))R(t)$ 이기 때문에

$$\dot{R}(t)^T = R(t)^T \text{skew}(\boldsymbol{\omega}(t))^T = -R(t)^T \text{skew}(\boldsymbol{\omega}(t))$$

(식 1-45)에 적용하면

$$\dot{I^{-1}}(t) = \dot{R}(t)I^{-1}(0)R(t)^T - R(t)I^{-1}(0)\dot{R}(t)^T$$

$$= \text{skew}(\boldsymbol{\omega}(t))R(t)I^{-1}(0)R(t)^T - R(t)I^{-1}(0)R(t)^T \text{skew}(\boldsymbol{\omega}(t))$$

$$= \text{skew}(\boldsymbol{\omega}(t))I^{-1}(t) - I^{-1}(t)\,\text{skew}(\boldsymbol{\omega}(t))$$

다시 **(식 1-44)**에 적용하면

$$\dot{\boldsymbol{\omega}}(t) = \dot{I^{-1}}(t)\mathbf{L}(t) + I^{-1}(t)\dot{\mathbf{L}}(t)$$

$$= \left(\text{skew}\left(\boldsymbol{\omega}(t)\right)I^{-1}(t) - I^{-1}(t)\,\text{skew}\left(\boldsymbol{\omega}(t)\right)\right)\mathbf{L}(t) + I^{-1}(t)\dot{\mathbf{L}}(t)$$

$$= \text{skew}\left(\boldsymbol{\omega}(t)\right)I^{-1}(t)\,\mathbf{L}(t) - I^{-1}(t)\,\text{skew}\left(\boldsymbol{\omega}(t)\right)\mathbf{L}(t) + I^{-1}(t)\dot{\mathbf{L}}(t)$$

$I^{-1}(t)\,\mathbf{L}(t) = \boldsymbol{\omega}(t)$ 이고 $\text{skew}\left(\boldsymbol{\omega}(t)\right)\boldsymbol{\omega}(t) = 0$ 이므로

$$\dot{\boldsymbol{\omega}}(t) = -I^{-1}(t)\,\text{skew}\left(\boldsymbol{\omega}(t)\right)\mathbf{L}(t) + I^{-1}(t)\dot{\mathbf{L}}(t)$$

$$= I^{-1}(t)\left(-\boldsymbol{\omega}(t) \times \mathbf{L}(t) + \dot{\mathbf{L}}(t)\right)$$

$$= I^{-1}(t)\left(\mathbf{L}(t) \times \boldsymbol{\omega}(t) + \dot{\mathbf{L}}(t)\right)$$

(식 1-46)

KINETIC ENERGY(운동에너지)

이번에는 운동에너지에 대해서 알아보자.

각 입자의 운동에너지는 $\frac{1}{2}m_i\mathbf{v}_i{}^T\mathbf{v}_i$으로 정의한다.

전체 운동에너지 E 는

$$E = \sum_i \frac{1}{2}m_i\mathbf{v}_i{}^T\mathbf{v}_i = \sum_i \frac{1}{2}m_i\mathbf{v}_i{}^T\mathbf{v}_i$$

(식 1-30) 적용

$$\sum_i \frac{1}{2}m_i(\mathbf{v} + \boldsymbol{\omega} \times \mathbf{r}_i)^T(\mathbf{v} + \boldsymbol{\omega} \times \mathbf{r}_i)$$

$$= \sum_i \frac{1}{2}m_i(\mathbf{v} - \text{skew}(\mathbf{r}_i)\boldsymbol{\omega})^T(\mathbf{v} - \text{skew}(\mathbf{r}_i)\boldsymbol{\omega})$$

$$= \sum_i \left(\frac{1}{2}m_i\mathbf{v}^T\mathbf{v} - m_i\mathbf{v}^T\text{skew}(\mathbf{r}_i)\boldsymbol{\omega} + \frac{1}{2}\boldsymbol{\omega}^T\text{skew}(\mathbf{r}_i)^T\text{skew}(\mathbf{r}_i)\boldsymbol{\omega}\right)$$

$\text{skew}(\mathbf{r}_i)^T = -\text{skew}(\mathbf{r}_i)$ 를 적용하면

$$= \frac{1}{2}\mathbf{v}^{T}\left(\sum_{i} m_i\right)\mathbf{v} - \mathbf{v}^{T}\mathrm{skew}\left(\sum_{i} m_i\mathbf{r}_i\right)\boldsymbol{\omega} + \frac{1}{2}\boldsymbol{\omega}^{T}\left(\sum_{i} -m_i\mathrm{skew}(\mathbf{r}_i)\mathrm{skew}(\mathbf{r}_i)\right)\boldsymbol{\omega}$$

$\sum_i m_i\mathbf{r}_i = 0$ 와 **(식 1-36)**을 적용하면

$$= \frac{1}{2}\mathbf{v}^{T}m\mathbf{v} + \frac{1}{2}\boldsymbol{\omega}^{T}I\boldsymbol{\omega}$$

(식 1-47)

이번에는 운동에너지를 미분해 보자.

$$\dot{E} = \frac{1}{2}(\mathbf{v}\cdot\dot{m}\mathbf{v}) + \frac{1}{2}(\boldsymbol{\omega}\cdot\dot{I}\boldsymbol{\omega})$$

$$= \frac{1}{2}m\dot{\mathbf{v}}\cdot\mathbf{v} + \frac{1}{2}m\mathbf{v}\cdot\dot{\mathbf{v}} + \frac{1}{2}\dot{\boldsymbol{\omega}}\cdot I\boldsymbol{\omega} + \frac{1}{2}\boldsymbol{\omega}\cdot(I\dot{\boldsymbol{\omega}})$$

(식 1-39)에서 $m\dot{\mathbf{v}} = \mathbf{F}$, **(식 1-41)**, **(식 1-44)**에서 $(I\dot{\boldsymbol{\omega}}) = \boldsymbol{\tau}$

$$= \mathbf{F}\cdot\mathbf{v} + \frac{1}{2}\dot{\boldsymbol{\omega}}\cdot I\boldsymbol{\omega} + \frac{1}{2}\boldsymbol{\omega}\cdot\boldsymbol{\tau}$$

$\dot{\boldsymbol{\omega}}\cdot I\boldsymbol{\omega} = (I\boldsymbol{\omega})^{T}\dot{\boldsymbol{\omega}} = \boldsymbol{\omega}^{T}I\dot{\boldsymbol{\omega}}$이고, **(식 1-46)**을 적용하면

$$= \mathbf{F}\cdot\mathbf{v} + \frac{1}{2}\boldsymbol{\omega}^{T}\left(\mathbf{L}\times\boldsymbol{\omega} + \dot{\mathbf{L}}\right) + \frac{1}{2}\boldsymbol{\tau}\cdot\boldsymbol{\omega}$$

$\boldsymbol{\omega}^{T}(\mathbf{L}\times\boldsymbol{\omega}) = \boldsymbol{\omega}\cdot(\mathbf{L}\times\boldsymbol{\omega}) = 0$ 이고 $\boldsymbol{\omega}^{T}\dot{\mathbf{L}} = \boldsymbol{\tau}\cdot\boldsymbol{\omega}$ 이므로

$$\dot{E} = \mathbf{F}\cdot\mathbf{v} + \boldsymbol{\tau}\cdot\boldsymbol{\omega}$$

(식 1-48)

ACCELERATION OF PARTICLE

마지막으로, 입자의 가속도 $\ddot{\mathbf{x}}_i(t)$ 에 대해서 살펴보자.

(식 1-30)에 의해

$$\dot{\mathbf{x}}_i(t) = \mathbf{v}(t) + \boldsymbol{\omega}(t)\times\mathbf{r}_i(t)$$

$$\ddot{\mathbf{x}}_i(t) = \dot{\mathbf{v}}(t) + \dot{\boldsymbol{\omega}}(t)\times\mathbf{r}_i(t) + \boldsymbol{\omega}(t)\times\dot{\mathbf{r}}_i(t)$$

$$= \dot{\mathbf{v}}(t) + \dot{\boldsymbol{\omega}}(t)\times\mathbf{r}_i(t) + \boldsymbol{\omega}(t)\times\left(\boldsymbol{\omega}(t)\times\mathbf{r}_i(t)\right)$$

(식 1-49)

첫 번째 항은 물체의 선형운동에 의한 가속도를 나타낸다.

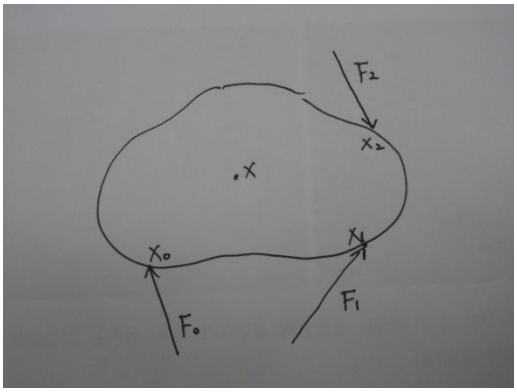
그리고, 두 번째 항은 tangential acceleration 으로 물체의 회전 속도의 증감에 따른 가속도를 의미한다. 그리고, 마지막 항은 물체의 centripetal acceleration(구심력)을 나타낸다. 회전을 유지하려면, 구심력에 의해 입자를 안쪽으로 끌어들여야 한다.

이로써, 어려운 물리 공부는 끝났다. 이제 물리 엔진을 만들기만 하면 된다. 기대하시라……ㅎㅎ

2 UNCONSTRAINED MOTION

이번 장에서는 물체가 가해지는 힘이 입력으로 정해질 때, 물체의 운동을 계산하는 방법에 대해서 다루어 보겠다.

FORCE & TORQUE



(그림 2-1)

(그림 2-1)에서 물체의 무게 중심을 $\mathbf{x}$ 라고 하자. $\mathbf{F}_0$ 의 힘이 $\mathbf{x}_0$ 위치에 가해지고, $\mathbf{F}_1$ 은 $\mathbf{x}_1$에, $\mathbf{F}_2$ 은 $\mathbf{x}_2$에 힘이 가해지고 있다고 가정하자.

그러면, (식 1-31)과 (식 1-33)에서 정의한 것처럼, 이 물체에 가해지는 힘은 $\mathbf{F} = \mathbf{F}_0 + \mathbf{F}_1 + \mathbf{F}_2$ 이고 토크는 $\boldsymbol{\tau} = (\mathbf{x}_0 - \mathbf{x}) \times \mathbf{F}_0 + (\mathbf{x}_1 - \mathbf{x}) \times \mathbf{F}_1 + (\mathbf{x}_2 - \mathbf{x}) \times \mathbf{F}_2$ 가 된다.

ACCELERATION

물체의 운동 방정식을 구해보자.

질량 m, Inertia Tensor 초기값 $\mathrm{I}(0)$, 무게중심 초기 위치 $\mathbf{x}(0)$, 초기 방향 $\mathrm{q}(0)$, 무게 중심의 초기 속도 $\mathbf{v}(0)$, 초기 angular velocity $\boldsymbol{\omega}(0)$, Force $\mathbf{F}(\mathrm{t})$, Torque $\boldsymbol{\tau}(\mathrm{t})$ 이 입력으로 주어질 때, 물체의 운동을 계산해보자.

우선 acceleration 부터 구해야 한다. (식 1-39)에 의해 선형 가속도는 쉽게 구할 수 있다.

$$\dot{\mathbf{v}}(\mathrm{t}) = \mathrm{m}^{-1}\mathbf{F}(\mathrm{t})$$

(식 2-1)

또 (식 1-46)을 이용해 angular acceleration $\dot{\boldsymbol{\omega}}(\mathrm{t})$ 을 계산할 수 있다.

$$\dot{\boldsymbol{\omega}}(\mathrm{t}) = \mathrm{I}^{-1}(\mathrm{t})\left(\mathbf{L}(\mathrm{t}) \times \boldsymbol{\omega}(\mathrm{t}) + \dot{\mathbf{L}}(\mathrm{t})\right)$$

하지만, 실제로 물리 엔진을 구현할 때 정확한 $\dot{\boldsymbol{\omega}}(\mathrm{t})$ 를 계산하지 않는다. $\mathbf{L}(\mathrm{t}) \times \boldsymbol{\omega}(\mathrm{t}) \approx 0$ 으로 간주해서 간단히 계산한다. 또 이렇게 해야 안정된 계산 결과를 얻을 수 있다. 잘못하면 계산 결과가 들쑥날쑥 요동칠 수 있다.

$$\dot{\boldsymbol{\omega}}(\mathrm{t}) \approx \mathrm{I}^{-1}(\mathrm{t})\dot{\mathbf{L}}(\mathrm{t}) = \mathrm{I}^{-1}(\mathrm{t})\boldsymbol{\tau}(\mathrm{t})$$

(식 2-2)

대부분의 물리엔진에서, (식 2-2)를 이용해 angular acceleration 을 계산한다.

INTEGRATION

적분하는데도 여러 가지 방법이 있다. 물리 엔진에서는 간단한 방법을 사용해 적분한다.

$$F(t) = \int_0^t f(\tau)d\tau$$

을 구해 보자.

$$F(t) \approx F(0) + \sum_{i=0}^{n-1} f(i\Delta t)\Delta t, \qquad \Delta t = \frac{t}{n}$$

으로 계산할 수 있다.

즉,

$$F(t + \Delta t) = F(t) + f(t)\Delta t$$

을 반복적으로 적용하여 $F(t)$를 구한다.

그리고, 이런 식으로 계산하는 것을 Explicit Method 라고 부른다.

경우에 따라서는,

$$F(t) \approx F(0) + \sum_{i=0}^{n-1} f\big((i+1)\Delta t\big)\Delta t, \qquad \Delta t = \frac{t}{n}$$

로 계산하기도 한다. 이것은 Implicit Method 라고 한다.

이 때는

$$F(t + \Delta t) = F(t) + f(t + \Delta t)\Delta t$$

을 이용해 $F(t)$를 구한다

Implicit Method 가 Explicit Method 에 비하여 더 안정적인 결과값을 구할 수 있어서,물리엔진에서는 가급적 Implicit Method 를 사용한다.

VELOCITY

$\mathbf{v}(t)$ 는 $\dot{\mathbf{v}}(t)$ 를 integration(적분)해서 구한다. $\mathbf{v}(t) = \int \dot{\mathbf{v}}(t)dt$ 이다.

이 때, Explicit Method 를 사용한다.

컴퓨터에서 구현할 때는

$$\mathbf{v}(t + \Delta t) = \mathbf{v}(t) + \dot{\mathbf{v}}(t)\Delta t = \mathbf{v}(t) + m^{-1}\mathbf{F}(t)\Delta t$$

(식 2-3)

으로 계산한다.

마찬가지로, $\boldsymbol{\omega}(t)$ 는

$$\boldsymbol{\omega}(t + \Delta t) = \boldsymbol{\omega}(t) + \dot{\boldsymbol{\omega}}(t)\Delta t \approx \boldsymbol{\omega}(t) + I^{-1}(t)\boldsymbol{\tau}(t)\Delta t$$

(식 2-4)

로 구한다.

POSITION & ROTATION

마지막으로, 위치와 방향을 구하면 모든 계산이 끝난다.

Explicit 방식을 사용하면,

$$\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \mathbf{v}(t)\Delta t$$

로 위치를 계산할 수 있다.

Implicit 방식을 사용하면,

$$\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \mathbf{v}(t + \Delta t)\Delta t$$

(식 2-5)

이 된다. Implicit 방식이 Explicit 방식보다 안정적이므로, 대부분의 물리엔진에서는 **(식 2-5)**를 이용해 위치를 계산한다.

(식 1-29)를 보면

$$q(t+\Delta t) = q(t) + \dot{q}(t)\Delta t = q(t) + \frac{1}{2}\hat{\boldsymbol{\omega}}(t+\Delta t)q(t)\Delta t$$

(식 2-6)

을 이용해 방향을 구한다.

ROTATION MATRIX

Quaternion 의 값을 알면, **(식 1-22)**를 이용해 회전 행렬을 구할 수 있다.

$q(t+\Delta t) = q_w(t+\Delta t) + q_x(t+\Delta t)\mathbf{i} + q_y(t+\Delta t)\mathbf{j} + q_z(t+\Delta t)\mathbf{k}$ 로 표시하면

$$R(t+\Delta t)$$

$$= \begin{bmatrix} 1-2\left(q_y(t+\Delta t)\right)^2 - 2\left(q_z(t+\Delta t)\right)^2 & 2q_x(t+\Delta t)q_y(t+\Delta t) - 2q_w(t+\Delta t)q_z(t+\Delta t) & 2q_x(t+\Delta t)q_z(t+\Delta t) + 2q_w(t+\Delta t)q_y(t+\Delta t) \\ 2q_x(t+\Delta t)q_y(t+\Delta t) + 2q_w(t+\Delta t)q_z(t+\Delta t) & 1-2\left(q_x(t+\Delta t)\right)^2 - 2\left(q_z(t+\Delta t)\right)^2 & 2q_y(t+\Delta t)q_z(t+\Delta t) - 2q_w(t+\Delta t)q_x(t+\Delta t) \\ 2q_x(t+\Delta t)q_z(t+\Delta t) - 2q_w(t+\Delta t)q_y(t+\Delta t) & 2q_y(t+\Delta t)q_z(t+\Delta t) + 2q_w(t+\Delta t)q_x(t+\Delta t) & 1-2\left(q_x(t+\Delta t)\right)^2 - 2\left(q_y(t+\Delta t)\right)^2 \end{bmatrix}$$

(식 2-7)

로 쉽게 구할 수 있다.

INERTIA TENSOR

Inertia Tensor 는 **(식 1-37)**을 이용해 계산한다.

$$I(t+\Delta t) = R(t+\Delta t)I(0)R(t+\Delta t)^T$$

(식 2-8)

결론

지금까지의 식을 종합하면,$\mathbf{F}(t), \boldsymbol{\tau}(t), \mathbf{v}(t), \boldsymbol{\omega}(t), \mathbf{x}(t), q(t), m, I(t)$의 값을 알면 $\mathbf{F}(t+\Delta t), \boldsymbol{\tau}(t+\Delta t), \mathbf{v}(t+\Delta t), \boldsymbol{\omega}(t+\Delta t), \mathbf{x}(t+\Delta t), q(t+\Delta t), R(t+\Delta t), I(t+\Delta t)$의 값도 계산할 수 있다.

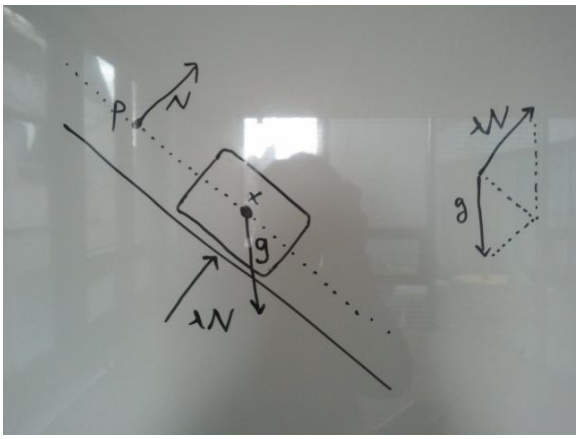
$\mathbf{F}(t), \boldsymbol{\tau}(t), \mathbf{v}(0), \boldsymbol{\omega}(0), \mathbf{x}(0), q(0), m, I(0)$ 의 값을 알면, 모든 순간의 $\mathbf{x}(t), R(t)$를 구할 수 있다.

다음 장에서는 $\mathbf{F}(t), \boldsymbol{\tau}(t)$를 구하는 방법에 대해서 알아볼 것이다.

3 CONSTRAINED DYNAMICS

앞장에서는 물체에 가해지는 힘이 입력으로 주어졌을 때, 그 물체의 운동을 계산하는 것을 배웠다. 하지만, 모든 힘이 주어지지 않는 경우가 많다. 그럴 때는 힘을 먼저 계산한 후 앞에서 공부한 방법대로 계산해야 한다.

그림을 보면서 설명하겠다.



(그림 3-1)

(그림 3-1)에서 네모난 물체가 경사면을 따라서 미끄러지고 있다. 이 물체와 바닥 면이 끈과 같이 특수한 장치로 연결되어서 서로 떨어지지 않는다고 가정하자.

이 때 중력 $\mathbf{g}$ 가 물체에 가해지고, 물체의 무게 중심은 $\mathbf{x}(t)$ 라고 가정하자.

경사면의 방향은 $\mathbf{N}$ 이라고 하자. $\mathbf{N}$ 의 크기는 1 이다. 그림에서 점선으로 표시된 경사면이 점 $\mathbf{p}$ 를 지난다.

외부에서 이 물체에 가해지는 힘은 $\mathbf{g}$ 뿐이다. 하지만, 물체 상호간의 작용 반작용으로 또 다른 힘이 가해진다. 이 반작용 힘의 방향은 $\mathbf{N}$ 이다. 그리고 그 크기를 λ라고 표시하자. 네모난 물체에는 $\lambda\mathbf{N}$ 의 힘이 가해지고, 경사면에는 $-\lambda\mathbf{N}$ 의 힘이 가해진다. 이렇게 외부의 힘뿐 아니라 내부의 힘도 계산해야 하는 경우가 발생한다. 우리가 해야 할 일은 λ의 값을 계산하는 것이다. 변수 λ 값이 너무 작으면 물체가 경사면 아래로 파고드는 비정상적인 사태가 발생할 것이다. 그리고, 변수 λ의 값이 너무 크면 경사면 위로 튀어 오르게 된다. 그래서 가급적 λ 값이 작으면서 경사면 아래로 충돌하지 않도록 하는 값을 구해야 한다.

모든 시간 t 에 대해서

$$(\mathbf{x}(t) - \mathbf{p}) \cdot \mathbf{N} = 0$$

(식 3-1)

를 만족해야 한다.

미분하면

$$\mathbf{v}(t) \cdot \mathbf{N} = 0$$

(식 3-2)

먼저 우리가 해야 할 일은 (식 3-2)를 만족시킬 수 있는 $\lambda(t)$를 구하는 것이다.

어떤 순간 t 에 $\mathbf{v}(t) \cdot \mathbf{N} = 0$ 라고 가정하자.

순간 $t + \Delta t$에도 (식 3-2)를 만족해야 한다.

$$\mathbf{v}(t + \Delta t) \cdot \mathbf{N} = 0$$

(식 3-3)

$\mathbf{v}(t + \Delta t) = \mathbf{v}(t) + +m^{-1}(\mathbf{g} + \lambda(t)\mathbf{N})\Delta t$ 를 대입하면

$$(\mathbf{v}(t) + m^{-1}(\mathbf{g} + \lambda(t)\mathbf{N})\Delta t) \cdot \mathbf{N} = 0$$

$$\mathbf{v}(t) \cdot \mathbf{N} + m^{-1}(\mathbf{g} \cdot \mathbf{N})\Delta t + m^{-1}\lambda(t)\Delta t = 0$$

$$\lambda(t) = -\mathbf{g} \cdot \mathbf{N}$$

(식 3-4)

이렇게 매 순간 λ 의 값을 구할 수 있다.

그 다음 앞 장에서 배운 방법대로 물체의 운동을 계산하면 된다.

그리고 힘 $\lambda\mathbf{N}$ 과 물체의 속도 $\mathbf{v}(t)$는 서로 직각이기 때문에 $\lambda\mathbf{N}$ 때문에 물체의 운동에너지가 증가(또는 감소)하지 않는다.

CONSTRAINTS

물체간의 제한 조건을 만족시킬 수 있도록 물체 상호간에 발생하는 힘을 계산해 줘야 한다.

n 개의 물체가 있다고 가정하고, i 번째 물체의 무게중심의 속도를 $\mathbf{v}_i$, angular velocity $\boldsymbol{\omega}_i$라고 하자.

전체 물체의 velocity 를 하나의 6n 차원 벡터로 표시할 수 있다.

$$\mathbf{V} = \begin{bmatrix} \mathbf{v}_1 \\ \boldsymbol{\omega}_1 \\ \mathbf{v}_2 \\ \boldsymbol{\omega}_2 \\ \cdots \\ \cdots \\ \mathbf{v}_n \\ \boldsymbol{\omega}_n \end{bmatrix} (6n \times 1)$$

(식 3-5)

Constraint 는 1 개나 2 개의 물체의 관계에 대한 조건식이다.

Constraint 가 s 개 있고, k 번째 조건이 i 번째 물체와 j 번째 물체간의 관계에 대한 조건이라고 하면

$$C_k(\mathbf{x}_i, q_i, \mathbf{x}_j, q_j) = 0$$

$$1 \le k \le s$$

(식 3-6)

$\mathbf{x}_i = \begin{bmatrix} \mathbf{x}_{i,x} \\ \mathbf{x}_{i,y} \\ \mathbf{x}_{i,z} \end{bmatrix}$, $q_i = q_{i,w} + q_{i,x}\mathbf{i} + q_{i,y}\mathbf{j} + q_{i,z}\mathbf{k}$ 로 표시하고, 미분하면

$$\dot{C_k} = \frac{\partial C_k}{\partial \mathbf{x}_{i,x}}\dot{\mathbf{x}_{i,x}} + \frac{\partial C_k}{\partial \mathbf{x}_{i,y}}\dot{\mathbf{x}_{i,y}} + \frac{\partial C_k}{\partial \mathbf{x}_{i,z}}\dot{\mathbf{x}_{i,z}} + \frac{\partial C_k}{\partial q_{i,w}}\dot{q_{i,w}} + \frac{\partial C_k}{\partial q_{i,x}}\dot{q_{i,x}} + \frac{\partial C_k}{\partial q_{i,y}}\dot{q_{i,y}} + \frac{\partial C_k}{\partial q_{i,z}}\dot{q_{i,z}}$$

$$+\frac{\partial C_k}{\partial \mathbf{x}_{j,x}}\dot{\mathbf{x}}_{j,x}+\frac{\partial C_k}{\partial \mathbf{x}_{j,y}}\dot{\mathbf{x}}_{j,y}+\frac{\partial C_k}{\partial \mathbf{x}_{j,z}}\dot{\mathbf{x}}_{j,z}+\frac{\partial C_k}{\partial q_{j,w}}\dot{q}_{j,w}+\frac{\partial C_k}{\partial q_{j,x}}\dot{q}_{i,x}+\frac{\partial C_k}{\partial q_{j,y}}\dot{q}_{j,y}+\frac{\partial C_k}{\partial q_{j,z}}\dot{q}_{j,z}=0$$

$\dot{q}_i=\frac{1}{2}\widehat{\omega_i}q_i$ **(식 1-29)**이기 때문에,

$$\dot{q}_{i,w}=-\frac{1}{2}\boldsymbol{\omega}_{i,x}q_{i,x}-\frac{1}{2}\boldsymbol{\omega}_{i,y}q_{i,y}-\frac{1}{2}\boldsymbol{\omega}_{i,z}q_{i,x}$$

$$\dot{q}_{i,x}=\frac{1}{2}q_{i,w}\boldsymbol{\omega}_{i,x}+\frac{1}{2}q_{i,z}\boldsymbol{\omega}_{i,y}-\frac{1}{2}q_{i,y}\boldsymbol{\omega}_{i,z}$$

$$\dot{q}_{i,y}=\frac{1}{2}q_{i,w}\boldsymbol{\omega}_{i,y}+\frac{1}{2}q_{i,x}\boldsymbol{\omega}_{i,z}-\frac{1}{2}q_{i,z}\boldsymbol{\omega}_{i,x}$$

$$\dot{q}_{i,z}=\frac{1}{2}q_{i,w}\boldsymbol{\omega}_{i,z}+\frac{1}{2}q_{i,y}\boldsymbol{\omega}_{i,x}-\frac{1}{2}q_{i,x}\boldsymbol{\omega}_{i,y}$$

를 대입하면,

$$\frac{\partial C_k}{\partial \mathbf{x}_{i,x}}\mathbf{v}_{i,x}+\frac{\partial C_k}{\partial \mathbf{x}_{i,y}}\mathbf{v}_{i,y}+\frac{\partial C_k}{\partial \mathbf{x}_{i,z}}\mathbf{v}_{i,z}$$

$$+\left(-\frac{1}{2}q_{i,x}\frac{\partial C_k}{\partial q_{i,w}}+\frac{1}{2}q_{i,w}\frac{\partial C_k}{\partial q_{i,x}}-\frac{1}{2}q_{i,z}\frac{\partial C_k}{\partial q_{i,y}}+\frac{1}{2}q_{i,y}\frac{\partial C_k}{\partial q_{i,z}}\right)\boldsymbol{\omega}_{i,x}$$

$$+\left(-\frac{1}{2}q_{i,y}\frac{\partial C_k}{\partial q_{i,w}}+\frac{1}{2}q_{i,z}\frac{\partial C_k}{\partial q_{i,x}}+\frac{1}{2}q_{i,w}\frac{\partial C_k}{\partial q_{i,y}}-\frac{1}{2}q_{i,x}\frac{\partial C_k}{\partial q_{i,z}}\right)\boldsymbol{\omega}_{i,y}$$

$$+\left(-\frac{1}{2}q_{i,z}\frac{\partial C_k}{\partial q_{i,w}}-\frac{1}{2}q_{i,y}\frac{\partial C_k}{\partial q_{i,x}}+\frac{1}{2}q_{i,x}\frac{\partial C_k}{\partial q_{i,y}}+\frac{1}{2}q_{i,w}\frac{\partial C_k}{\partial q_{i,z}}\right)\boldsymbol{\omega}_{i,z}$$

$$+\frac{\partial C_k}{\partial \mathbf{x}_{j,x}}\mathbf{v}_{j,x}+\frac{\partial C_k}{\partial \mathbf{x}_{j,y}}\mathbf{v}_{j,y}+\frac{\partial C_k}{\partial \mathbf{x}_{j,z}}\mathbf{v}_{j,z}$$

$$+\left(-\frac{1}{2}q_{j,x}\frac{\partial C_k}{\partial q_{j,w}}+\frac{1}{2}q_{j,w}\frac{\partial C_k}{\partial q_{j,x}}-\frac{1}{2}q_{j,z}\frac{\partial C_k}{\partial q_{j,y}}+\frac{1}{2}q_{j,y}\frac{\partial C_k}{\partial q_{j,z}}\right)\boldsymbol{\omega}_{j,x}$$

$$+\left(-\frac{1}{2}q_{j,y}\frac{\partial C_k}{\partial q_{j,w}}+\frac{1}{2}q_{j,z}\frac{\partial C_k}{\partial q_{j,x}}+\frac{1}{2}q_{j,w}\frac{\partial C_k}{\partial q_{j,y}}-\frac{1}{2}q_{j,x}\frac{\partial C_k}{\partial q_{j,z}}\right)\boldsymbol{\omega}_{j,y}$$

$$+\left(-\frac{1}{2}q_{j,z}\frac{\partial C_k}{\partial q_{j,w}}-\frac{1}{2}q_{j,y}\frac{\partial C_k}{\partial q_{j,x}}+\frac{1}{2}q_{j,x}\frac{\partial C_k}{\partial q_{j,y}}+\frac{1}{2}q_{j,w}\frac{\partial C_k}{\partial q_{j,z}}\right)\boldsymbol{\omega}_{j,z}$$

$$=0$$

여기서, 계수를 $J_{k,6i+1}$, $J_{k,6i+2}$, $J_{k,6i+3}$, $J_{k,6i+4}$, $J_{k,6i+5}$, $J_{k,6i+6}$, $J_{k,6j+1}$, $J_{k,6j+2}$, $J_{k,6j+3}$, $J_{k,6j+4}$, $J_{k,6j+5}$, $J_{k,6j+6}$로 표현하면,

$$J_{k,6i+1}\mathbf{v}_{i,x}+J_{k,6i+2}\mathbf{v}_{i,y}+J_{k,6i+3}\mathbf{v}_{i,z}+J_{k,6i+4}\boldsymbol{\omega}_{i,x}+J_{k,6i+5}\boldsymbol{\omega}_{i,y}+J_{k,6i+6}\boldsymbol{\omega}_{i,z}$$

$$+ J_{k,6j+1}\mathbf{v}_{i,x} + J_{k,6j+2}\mathbf{v}_{i,y} + J_{k,6j+3}\mathbf{v}_{i,z} + J_{k,6j+4}\boldsymbol{\omega}_{j,x} + J_{k,6j+5}\boldsymbol{\omega}_{j,y} + J_{k,6j+6}\boldsymbol{\omega}_{j,z}$$

$$= 0$$

(식 3-7)

그러므로 **(식 3-6)**은 **(식 3-7)**형식으로 변환될 수 있다. **(식 3-6)**형식은 위치와 방향에 관한 조건식이고, **(식 3-7)**은 속도와 각속도에 관한 조건식이다. 물리엔진에서는 **(식 3-7)**형태로 constraint 를 표현한다.

constraint 는 물리엔진의 입력으로 들어간다. 물리엔진에서 하는 일은 입력된 constraint 를 만족하는 방정식을 풀어 주는 것이다. 프로그래머가 API 에 의해 constraint 를 입력할 수도 있고, 엔진의 충돌부분에서 추가하기도 한다. 또, hinge 와 같은 constraint 를 구현하는 코드에서도 추가한다.

나머지 $\beta \notin \{6i+1, 6i+2, 6i+3, 6i+4, 6i+5, 6i+6, 6j+1, 6j+2, 6j+3, 6j+4, 6j+5, 6j+6\}$에 대하여,

$$J_{k,\beta} = 0$$

라고 정하면, **(식 3-7)**은

$$[J_{k,1} \quad J_{k,2} \quad \cdots \quad J_{k,6n}]\mathbf{V} = 0$$

로 표현될 수 있다. 또, s 개의 Constraint 을 하나로 통합하면,

$$\begin{bmatrix} J_{1,1} & J_{1,2} & \cdots & J_{1,6n} \\ J_{2,1} & J_{2,2} & \cdots & J_{2,6n} \\ \cdots & \cdots & \cdots & \cdots \\ J_{s,1} & J_{s,2} & \cdots & J_{s,6n} \end{bmatrix} \mathbf{V} = \mathbf{0}$$

$$J\mathbf{V} = \mathbf{0}$$

(식 3-8)

로 표시된다. 여기서, J 는 $s \times 6n$ 행렬이고, $\mathbf{0}$ 은 $s \times 1$ 행렬이다.

이때, J 를 Jacobian 행렬이라고 부른다.

CONSTRAINT FORCE

힘에는 시스템 외부에서 주어지는 힘과 Constraint 때문에 시스템 내부에서 생기는 힘이 있다.

외부에서 발생하는 힘은 아래와 같이 표시된다.

$$\mathbf{F}_{\text{ext}} = \begin{bmatrix} \mathbf{f}_1 \\ \boldsymbol{\tau}_1 \\ \mathbf{f}_2 \\ \boldsymbol{\tau}_2 \\ \cdots \\ \cdots \\ \mathbf{f}_n \\ \boldsymbol{\tau}_n \end{bmatrix} (6n \text{ x } 1)$$

(식 3-9)

각 Constraint 를 만족하기 위해서 물체 상호간에 내부적인 힘이 작용한다.

i 번째 물체에 가해지는 Constraint Force 를 $\mathbf{f}_{ci}$로 표기하면, 전체 Constraint Force 는

$$\mathbf{F}_c = \begin{bmatrix} \mathbf{f}_{c1} \\ \boldsymbol{\tau}_{c1} \\ \mathbf{f}_{c2} \\ \boldsymbol{\tau}_{c2} \\ \cdots \\ \cdots \\ \mathbf{f}_{cn} \\ \boldsymbol{\tau}_{cn} \end{bmatrix} (6n \text{ x } 1)$$

(식 3-10)

로 표시된다.

그런데, **(식 1-48)**에 따르면 $\mathbf{F}_c \cdot \mathbf{V} = 0$ 이면 운동 에너지에 변화를 주지 않는다. 만약, 임의의 $s \times 1$ 행렬 $\boldsymbol{\lambda}$에 대하여

$$\mathbf{F}_c = \mathrm{J}^{\mathrm{T}}\boldsymbol{\lambda}$$

(식 3-11)

이고, **(식 3-8)** $\mathrm{J}\mathbf{V} = \mathbf{0}$ 을 만족한다면,

$$\mathbf{F}_c{}^{\mathrm{T}} = \boldsymbol{\lambda}^{\mathrm{T}}\mathrm{J}$$

$$\boldsymbol{F}_c{}^{T}\boldsymbol{V} = \boldsymbol{\lambda}^{T}\boldsymbol{J}\boldsymbol{V}$$

(식 3-8) 적용

$$\mathbf{F}_c \cdot \mathbf{V} = \boldsymbol{\lambda}^{\mathrm{T}}\mathbf{0} = 0$$

그래서, $\mathbf{F}_c$ 는 운동 에너지에 영향을 주지 않는 시스템 내부의 힘이다. 물리엔진에서는 미지수 벡터 $\boldsymbol{\lambda}$를 구하는 일을 한다. s 개의 변수를 가지는 연립 방정식을 푸는 것이다.

E 가 3×3 단위 행렬이라고 하면, 질량과 Inertia Tensor 는

$$M = \begin{bmatrix} m_1E & 0 & 0 & 0 & \dots & \dots & 0 & 0 \\ 0 & I_1 & 0 & 0 & \dots & \dots & 0 & 0 \\ 0 & 0 & m_2E & 0 & \dots & \dots & 0 & 0 \\ 0 & 0 & 0 & I_2 & \dots & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & \dots & \dots & m_nE & 0 \\ 0 & 0 & 0 & 0 & \dots & \dots & 0 & I_n \end{bmatrix} (6n \text{ x } 6n)$$

(식 3-12)

로 표현한다.

(그림 3-1)로 돌아가서 적용해 보면

$$\mathbf{V} = \begin{bmatrix} \mathbf{v}(t) \\ \boldsymbol{\omega}(t) \end{bmatrix}$$

$$J = [N_x \quad N_y \quad N_z \quad 0 \quad 0 \quad 0] = [\mathbf{N}^T \quad \mathbf{0}] \ (1 \text{ x } 6)$$

$$\mathbf{F}_{ext} = \begin{bmatrix} \mathbf{g} \\ \mathbf{0} \end{bmatrix} (6 \text{ x } 1)$$

$$\mathbf{F}_c = J^T\lambda = \begin{bmatrix} \mathbf{N} \\ \mathbf{0} \end{bmatrix}\lambda$$

가 된다.

CONSTRAINT BIAS

(그림 3-1)의 문제에서, 계산상의 오차나 기타 이유로 인하여, **(식 3-1)**이 성립하지 않는 경우가 발생한다.

$$\varepsilon = (\mathbf{x}(t) - \mathbf{p}) \cdot \mathbf{N}$$

(식 3-13)

원래 정상적인 경우 ε 의 값은 0 이어야 하지만, 오차가 있는 경우가 있다. 그래서, **(식 3-3)**를 변형하여 오차를 보정한다.

$$\mathbf{v}(t+\Delta t)\cdot\mathbf{N}=-\beta\varepsilon$$

(식 3-14)

여기서, $-\beta\varepsilon$ 를 Constraint Bias 라고 한다. β의 값이 0 이면 오차가 줄어들지 않고 유지된다. β 의 값이 1 이면 오차가 즉시 교정되지만, 물리엔진이 불안정해지고 결과가 들쑥날쑥 할 수 있다. 그래서, 안정된 결과값을 가지도록 적당한 값을 0 과 1 사이에서 tuning 해야 한다.

마찬가지로 **(식 3-8)**도

$$J(t)\mathbf{V}(t+\Delta t)=\boldsymbol{\zeta}(t)$$

(식 3-15)

로 바뀐다. $\boldsymbol{\zeta}(t)$ 는 $s\times 1$ 행렬로 각 물체의 위치와 시간의 함수이다. $\boldsymbol{\zeta}(t)$ 의 각 행에는 해당 Constraint bias 값을 가진다.

EQUATION OF MOTION

이제 전체 방정식을 풀어보자.

$M\dot{\mathbf{V}}=\mathbf{F}_{ext}+\mathbf{F}_c$이므로

$$\mathbf{V}(t+\Delta t)=\mathbf{V}(t)+M^{-1}(t)\left(\mathbf{F}_{ext}(t)+\mathbf{F}_c(t)\right)\Delta t$$

(식 3-11) 적용

$$\mathbf{V}(t+\Delta t)=\mathbf{V}(t)+M^{-1}(t)\left(\mathbf{F}_{ext}(t)+J^{T}(t)\boldsymbol{\lambda}(t)\right)\Delta t$$

(식 3-15) 적용

$$J(t)\mathbf{V}(t)+J(t)M^{-1}(t)\left(\mathbf{F}_{ext}(t)+J^{T}(t)\boldsymbol{\lambda}(t)\right)\Delta t=\boldsymbol{\zeta}(t)$$

$$J(t)M^{-1}(t)J^{T}(t)\,\boldsymbol{\lambda}(t)=\frac{1}{\Delta t}\boldsymbol{\zeta}(t)-\frac{1}{\Delta t}J(t)\mathbf{V}(t)-J(t)M^{-1}(t)\mathbf{F}_{ext}(t)$$

그러므로

$$J(t)B(t)\boldsymbol{\lambda}(t)=\boldsymbol{\eta}(t)$$

단,

$$B(t) = M^{-1}(t)J^{T}(t), \qquad \boldsymbol{\eta}(t) = \frac{1}{\Delta t}\boldsymbol{\zeta}(t) - J(t)\left(\frac{1}{\Delta t}\mathbf{V}(t) + M^{-1}(t)\mathbf{F}_{ext}(t)\right)$$

(식 3-16)

으로 정리할 수 있다.

COMPUTING JACOBIAN

(식 3-16)의 방정식을 풀기 위해서는, $J(t)\left(\frac{1}{\Delta t}\mathbf{V}(t) + M^{-1}\mathbf{F}_{ext}(t)\right)$ 를 계산해야 한다. 그런데, Jacobian 행렬의 $s \times 6n$ 원소 중 대부분이 0 이다. 행렬의 각 행은 최대 12 개까지 0 이 아닌 원소를 가지고 나머지는 0 이다. 이 성질을 이용하여 Jacobian 행렬의 자료구조를 정하면, Jacobian 행렬과 $6n \times 1$ 행렬의 곱을 계산할 때 더 효율적으로 계산할 수 있다.

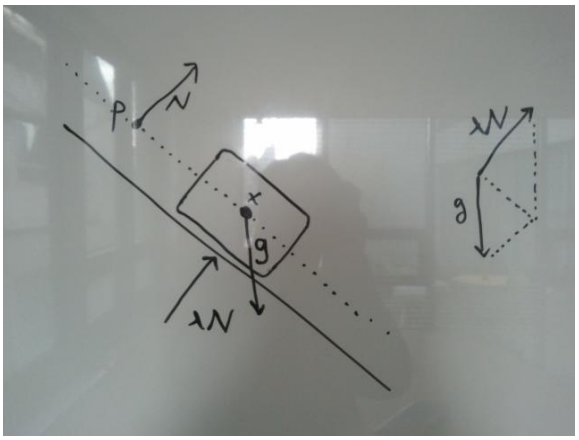
```
float [s] Multiply(Vector6 Jsp[s][2], int Jmap[s][2], Vector6 V[n])
{
        float     Result[s];
        for (int i = 0; i < s; i++)
                int b1 = Jmap [i][0];
                int b2 = Jmap [i][1];
                float   Sum = 0;
                Sum = Sum + Jsp[i][0] · V[b1];
                if   b2 >= 0
                        Sum = Sum + Jsp[i][1] · V[b2]
                Result[i] = Sum
        return Result
}
```

메모리를 절약하기 위해, 전체 Jacobian 행렬을 $s \times 6n$ 배열대신에 Jsp 배열과 Jmap 배열로 압축해 표현한다. Jsp 배열은 $s \times 12$행렬이고, Jmap 은 $s \times 2$ 행렬을 나타낸다. Jmap 에는 Jacobian 의 각 행이 어떤 물체간의 관계식인지 관한 정보를 저장한다. Jmap 의 i 번째의 행의 값이 b1 과 b2 라면, Jacobian 행렬의 i 번째 행은 b1 번째 물체와 b2 번째 물체의 관계식이 된다. 이때, Jacobian 행렬의 i 번째 행은 6*b1 열부터 6*b1+5 열까지와 6*b2 열부터 6*b2+5 열까지만 0 이 아닌 값을 가질 수 있고, 나머지 열의 값은 0 이다.

INEQUALITY CONSTRAINT

Constraint 가 등식인 경우도 있지만, 부등식인 경우도 있다.

Distance constraint(두 물체가 일정 거리에 있어야 하는 조건), revolute joints, prismatic joint 의 경우에는 constraint 가 등식이다. Contact constraint 나 joint angle limit 의 경우에는 부등식이다.



(그림 3-2)

이번에는 물체가 바닥 면에서 떨어질 수 있는 경우까지 생각해 보자. 앞에서는 설명을 간단하게 하기 위해 Constraint 가 등식인 경우에 대해서만 설명했다. 이번엔 Constraint 가 부등식인 경우에 대해 알아보겠다.

(그림 3-2)에서 다음 조건을 만족해야 한다.

$$(\mathbf{x}(t) - \mathbf{p}) \cdot \mathbf{N} \geq 0$$

(식 3-17)

그런데, 어떤 순간 t 에 $(\mathbf{x}(t) - \mathbf{p}) \cdot \mathbf{N} > 0$ 라고 하고, Δt가 충분히 작다면,

$$(\mathbf{x}(t + \Delta t) - \mathbf{p}) \cdot \mathbf{N} \geq 0$$

(식 3-18)

일 것이다. 그러므로 $(\mathbf{x}(t)-\mathbf{p})\cdot\mathbf{N}>0$ 인 경우에는 특별히 Constraint Force 를 생성하지 않아도 된다. 그냥 외부 힘만 이용하여 물체의 운동을 계산하면 된다. 그래서, $(\mathbf{x}(t)-\mathbf{p})\cdot\mathbf{N}=0$ 경우만 생각해보자.

어떤 순간 t 에 $(\mathbf{x}(t)-\mathbf{p})\cdot\mathbf{N}=0$ 라면

$$\mathbf{v}(t+\Delta t)\cdot\mathbf{N}\geq 0$$

(식 3-19)

이어야 (식 3-18)을 만족시킬 수 있다. $\mathbf{x}(t+\Delta t)=\mathbf{x}(t)+\mathbf{v}(t+\Delta t)\Delta t$ 이기 때문이다.
그래서, (식 3-19)는 다음과 같이 된다.

$\mathbf{v}(t+\Delta t)=\mathbf{v}(t)+m^{-1}(\mathbf{g}+\lambda\mathbf{N})\Delta t$ 이므로

$$(\mathbf{v}(t)+m^{-1}(\mathbf{g}+\lambda\mathbf{N})\Delta t)\cdot\mathbf{N}\geq 0$$

$\alpha=m^{-1}\Delta t,\ \beta=\mathbf{v}(t)\cdot\mathbf{N}+m^{-1}\Delta t\,\mathbf{g}\cdot\mathbf{N}$ 로 표현하면

$$\alpha\lambda+\beta\geq 0$$

(식 3-20)

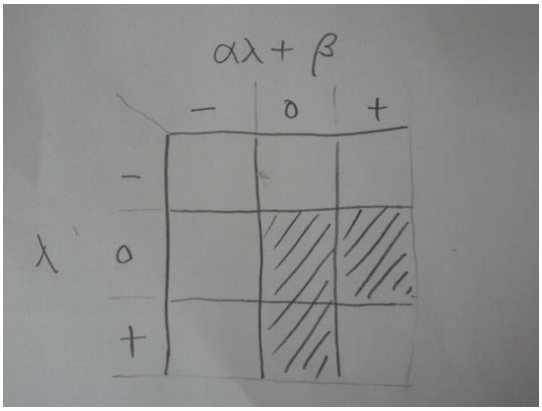
그리고, $\mathbf{v}(t+\Delta t)\cdot\mathbf{N}>0$ 이라면 $\lambda=0$ 이어야 한다. 물체가 바닥 면에서 떨어지기 때문에, 반작용의 힘이 없어지기 때문이다. 또, $\lambda>0$ 이라면 $\mathbf{v}(t+\Delta t)\cdot\mathbf{N}=0$ 이어야 한다. 물체가 바닥 면에 붙어있어야 하기 때문이다. 식으로 표현하면,

$$(\mathbf{v}(t+\Delta t)\cdot\mathbf{N}\geq 0)\ \text{and}\ (\lambda\geq 0)\ \text{and}\ \big((\mathbf{v}(t+\Delta t)\cdot\mathbf{N})\lambda=0\big)$$

λ의 조건 식으로 표현하면

$$(\alpha\lambda+\beta\geq 0)\ \text{and}\ (\lambda\geq 0)\text{and}\ \big((\alpha\lambda+\beta)\lambda=0\big)$$

(식 3-21)



(그림 3-3)

(그림 3-3)에서 $\alpha\lambda+\beta$, λ의 부호의 조합이 빗금 친 부분이어야 한다.

(식 3-21)을 다르게 표현하면

$$(\lambda \geq 0) \text{ and } (\lambda > 0 \rightarrow (\alpha\lambda+\beta) = 0) \text{ and } (\lambda = 0 \rightarrow (\alpha\lambda+\beta) \geq 0)$$

(식 3-22)

즉, 위 식을 만족하는 λ의 부등식을 풀면 된다.

엄밀하게 말해서, 물리엔진에서 모든 형식의 부등식 문제를 풀 필요는 없다. (식 3-22)과 같이 특수한 형식의 부등식 문제만 풀 수 있으면 충분하다. 이런 형태의 문제를 MLCP 라고 하는데, 뒤에서 더 설명하겠다.

이제 (식 3-16)이 부등식 constraint 의 경우까지 포함하도록 확장해 보자.

(식 3-15)에서

$$\mathbf{U}(t) = J(t)\mathbf{V}(t+\Delta t) - \boldsymbol{\zeta}(t)$$

로 바꾸면, (식 3-16)도

$$\mathbf{W}(t) = \frac{1}{\Delta t}\mathbf{U}(t) = J(t)B(t)\boldsymbol{\lambda}(t) - \boldsymbol{\eta}(t)$$

$$B(t) = M^{-1}(t)J^{T}(t), \qquad \boldsymbol{\eta}(t) = \frac{1}{\Delta t}\boldsymbol{\zeta}(t) - J(t)\left(\frac{1}{\Delta t}\mathbf{V}(t) + M^{-1}(t)\mathbf{F}_{ext}(t)\right)$$

(식 3-23)

로 바뀐다.

$\mathbf{U}(t)$와 $\mathbf{W}(t)$의 i 번째 행을 $U_i(t)$와 $W_i(t)$라고 표현하면, $U_i(t)$와 $W_i(t)$의 부호는 같다.

그래서, i 번째 constraint 의 조건이

$$(\lambda_i \geq 0) \text{ and } (\lambda_i > 0 \rightarrow U_i(t) = 0) \text{ and } (\lambda_i = 0 \rightarrow U_i(t) \geq 0)$$

이라고 한다면, 위의 조건은

$$(\lambda_i \geq 0) \text{ and } (\lambda_i > 0 \rightarrow W_i(t) = 0) \text{ and } (\lambda_i = 0 \rightarrow W_i(t) \geq 0)$$

와 필요충분조건이다.

(식 3-23)은 다음절에 나오는 MLCP 로 풀 수 있다.

MIXED LINEAR COMPLEMENTARY PROBLEM (MLCP)

A 가 $s \times s$ 행렬이고, $\mathbf{B}$ 와 $\mathbf{W}$, $\boldsymbol{\lambda}$ 를 $s \times 1$ 행렬일 때, $\mathbf{W} = A\boldsymbol{\lambda} - \mathbf{B}$ 만족해야 한다고 하자. $\mathbf{W}$ 의 i 번째 행이 W_i라고 하고, $\boldsymbol{\lambda}$의 i 번째 행을 λ_i라고 표시하자. 다음 조건을 모두 만족하는 $\boldsymbol{\lambda}$ 를 푸는 문제를 Mixed Linear Complementary Problem 라고 부른다.

$$\forall i : \lambda^-_i \leq \lambda_i \leq \lambda^+_i$$

$$\forall i : \lambda^-_i < \lambda_i < \lambda^+_i \rightarrow W_i = 0$$

$$\forall i : \lambda_i = \lambda^-_i \rightarrow W_i \geq 0$$

$$\forall i : \lambda_i = \lambda^+_i \rightarrow W_i \leq 0$$

(식 3-24)

MLCP 조건에서 λ_i가 한계 값 λ^-_i와 λ^+_i 이 아닐 때에는 W_i가 0 이어야 한다. MLCP 는 다음에 설명할 Projected Gauss-Seidel 방법 등으로 풀 수 있다.

λ^+_i (upper bound :상한 값)과 λ^-_ilower bound:하한 값)을 적당히 설정하면 여러 가지 형식의 Constraint 를 동일한 MLCP 로 풀 수 있다.

(식 3-16)로 돌아가서,

$$\mathbf{W} = JB\boldsymbol{\lambda} - \boldsymbol{\eta} = \mathbf{0}$$

(식 3-25)

를 MLCP 문제로 변환할 수 있는지 알아보자.

$\mathbf{W}$ 의 i 번째 행을 W_i 라고 하면, $W_i = 0$ 이어야 한다.

$\lambda^-_i = -\infty, \lambda^+_i = \infty$로 설정하면, (식 3-24)의 첫 번째 조건은 항상 성립하고, 세 번째와 네 번째 조건도 가정이 거짓이므로 항상 성립한다. 그리고, 두 번째 조건은 $W_i = 0$이 된다. 그러므로, (식 3-25)를 MLCP 에 의해서 풀 수 있다.

물리엔진에서 사용되는 constraint 는 다음 세 형식으로 분류할 수 있다.

i) $W_i = 0$

ii) $(\lambda_i \geq 0)$ and $(\lambda_i > 0 \rightarrow W_i = 0)$ and $(\lambda_i = 0 \rightarrow W_i \geq 0)$

iii) $(\lambda^-_i \leq \lambda_i \leq \lambda^+_i)$ and $(\lambda^-_i < \lambda_i < \lambda^+_i \rightarrow W_i = 0)$
and $(\lambda_i = \lambda^-_i \rightarrow W_i \geq 0)$ and $(\lambda_i = \lambda^+_i \rightarrow W_i \leq 0)$

(식 3-26)

첫 번째 경우에는 (식 3-25)와 같은 Equality constraint 이고, 두 번째는 (식 3-22)의 Contact Constraint 에 해당한다. (식 3-24)에 $\lambda^-_i = 0, \lambda^+_i = \infty$을 대입하면, 두 번째 형식의 constraint 를 MLCP 에 의해 풀 수 있다. 세 번째 형식은 접촉한 두 물체의 마찰력을 구현할 때 사용된다. 뒤에 나오는 friction constraint 부분에서 마찰력에 관해 설명하겠다.

이렇게 여러 가지 형식의 Constraint 를 동일한 MLCP 방법으로 풀 수 있다

GAUSS-SEIDEL METHOD

MLCP 를 푸는 방법을 배우기 전에, 우선 기초가 되는 Gauss-Seidel 방법에 대해서 알아보자.

Gauss-Seidel 은 연립방정식을 푸는 방법 중 하나이다.

A 가 $s \times s$ 행렬, B 와 $\boldsymbol{\lambda}$를 $s \times 1$ 행렬이라고 할 때, 연립 방정식 $A\boldsymbol{\lambda} = B$ 를 풀 때 이용된다.

```
float [s] GaussSeidel(float A[s][s], float B[s], float λexp[s])
{
        float    λ[s] = λexp;
```

```
        for (int Iter = 0; Iter < MAX_ITERATION; Iter++)
                for (int i = 0; i < s; i++)
                        float Delta = B[i];
                        for (int j = 0; j < s; j++)
                                Delta = Delta - A[i][j] * λ[j]
                        Delta = Delta / A[i][i]
                        λ[i] = λ[i] + Delta
        return λ
}
```

방정식의 해의 예상 값 $\boldsymbol{\lambda}_{exp}$부터 시작해서 점점 해에 접근해나간다.

A 가 Positive definite Matrix 에 경우, 여러 번 반복하면 $A\boldsymbol{\lambda} - B$는 점점 $\mathbf{0}$ 에 수렴한다.

자세한 것은 공업수학 책을 참고하기 바란다. 왜 수렴하는 지 알고 싶으면, **[13] Ren-Cang Li, Iterative Schemes and Their Convergence For a Symmetric Positive Definite Matrix** 를 보면 된다.

PROJECTED GAUSS-SEIDEL METHOD

Project Gauss-Seidel 방법은 Gauss-Seidel 을 확장해서 (식 3-24)의 MLCP 해를 구하는데 이용된다.

$$\mathbf{W} = JB\boldsymbol{\lambda} - \boldsymbol{\eta}, \qquad B = M^{-1}J^{T}$$

JB 가 positive definite 이라면 해는 수렴한다. 최소한 JB 는 positive semi definite 이다. JB 가 positive semi definite 인 경우에도 꽤 만족할 만할 결과를 얻을 수 있다.

```
float [s] ProjectedGaussSeidel(Vector6 Jsp[s][2], int Jmap[s][2], Matrix6x6 M^-1[n], float η[s],
        float λ^-[s],  float λ^+[s],  float λ_exp[s])
{
        Vector6 Bsp^T[s][2] = Convert(Jsp, Jmap, M^-1);          // B = M^-1 J^T, B^T = JM^-1
        float   λ[s] = λ_exp;
        Vector6 a[n] = TransposeMultiply (Bsp^T, Jmap, λ);      // a = B λ

        float   d[s];
        for (int i=0; i<s; i++)
                d[i] = Jsp[i][0] · Bsp^T[i][0]+ Jsp[i][1] · Bsp^T[i][1]

        for (int Iter = 0; Iter < MAX_ITERATION; Iter++)
                for (int i=0; i<s; i++)
                        int b1 = Jmap [i][0];
                        int b2 = Jmap[i][1];
                        float Delta = (η[i]-Jsp[i][0] · a[b1] -Jsp[i][1] · a[b2])/d[i];
                        float Old = λ[i];
                        λ[i] = max(λ^-[i], min(λ[i]+Delta, λ^+[i])
```

```
                    float Diff = λ[i] – Old;
                    a[b1] = a[b1] + Diff * Bsp^T[i][0]
                    a[b2] = a[b2] + Diff * Bsp^T[i][1]

        return λ
}

Vector6[s][2] Convert(Vector6 Jsp[s][2], int Jmap[s][2], Matrix6x6 M^-1[n])
{
        Vector6  Result[s][2];
        for (int i = 0; i < s; i++)
                int b1 = Jmap[i][0];
                int b2 = Jmap[i][1];
                Result[i][0] = M^-1[b1] * Jsp[i][0]
                Result[i][1] = M^-1[b2] * Jsp[i][1]
        return Result
}

Vector6 [n] TransposeMultiply(Vector6 Bsp^T[s][2], int Jmap[s][2], Vector6 λ[s])
{
        Vector6  Result[n];
        for (int i = 0; i < n; i++)
                Result[i] = 0
        for (int i = 0; i < s; i++)
                int b1 = Jmap [i][0];
                int b2 = Jmap [i][1];
                Result[b1] = Result[b1] + λ[b1] * Bsp^T[i][0]
                Result[b2] = Result[b2] + λ[b2] * Bsp^T[i][1]
        return Result
}
```

위 알고리즘의 동작원리를 이해하기 어려울 수 있다. 여기서 입력 JB 는 Gauss-Seidel 의 A 에 해당하고, ProjectedGaussSeidel() 함수의

```
float Delta = (η[i]-Jsp[i][0] · a[b1] -Jsp[i][1] · a[b2])/d[i];
```

부분은 GaussSeidel() 함수의

```
float Delta = (B[i] - ∑_j^s A[i][j] * λ[j]) / A[i][i];
```

에 해당되어, 알고리즘이 유사한 것을 알 수 있다.

λ 를 구할 때 λ_{exp}부터 시작해서 해에 접근시킨다. 실제 물리엔진에서는 매 프레임마다 입력이 거의 비슷하므로, 이전 해를 보관했다가 그 값에서부터 접근하면, 답에 더 빨리 수렴한다.

MLCP 를 푸는 또 다른 방법은 [3] David Baraff 의 논문에 자세히 설명되어 있고, [6] ODE 엔진에 구현된 코드가 있다. 그 논문의 방법을 이해하면, Projected Gauss-Seidel 을 이해하는 데도 도움이 될 것이다. (너무 자세하게 설명하게 되어서, 오히려 어려울 수 있다)

그러나, Projected Gauss-Seidel 이 더 단순하고 구현하기가 간단해서, 최근 물리엔진에서 많이 사용되고 있다.

4 CONTACT MODEL

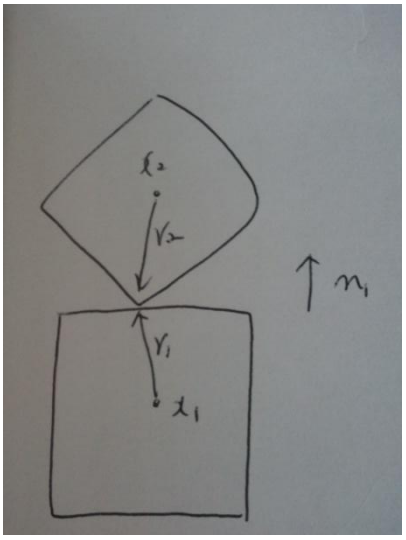
이번 장에서는 두 물체간의 충돌을 어떻게 처리하는지에 대해서 공부해보자.

우선 어떤 물체간에 충돌이 발생하는지 알아야 한다. 각각의 순간마다 접촉하고 있는 물체를 찾아내야 한다. 두 물체 간에 접촉하고 있다면, 접촉하고 있는 위치, 접촉면도 함께 찾는다. 그 다음 이런 충돌 정보를 처리한다. 충돌을 찾아내는 알고리즘은 복잡하므로 설명을 생략하겠다.

그 대신 그 결과값을 물리엔진에서 어떻게 처리하는지에 대해서 알아보겠다.

CONTACT CONSTRAINT

두 물체간에 일어나는 일반적인 충돌에서의 Jacobian 계수를 구해보자.



(그림 4-1)

(그림 4-1)에서

$$C_i = (\mathbf{x}_2 + \mathbf{r}_2 - \mathbf{x}_1 - \mathbf{r}_1) \cdot \mathbf{n}_1 \geq 0$$

(식 4-1)

(식 1-28) 에 의하면 $\dot{\mathbf{r}}_2 = \boldsymbol{\omega}_2 \times \mathbf{r}_2$이므로, 미분하면

$$\dot{C}_i = (\mathbf{v}_2 + \boldsymbol{\omega}_2 \times \mathbf{r}_2 - \mathbf{v}_1 - \boldsymbol{\omega}_1 \times \mathbf{r}_1) \cdot \mathbf{n}_1 + (\mathbf{x}_2 + \mathbf{r}_2 - \mathbf{x}_1 - \mathbf{r}_1) \cdot (\boldsymbol{\omega}_1 \times \mathbf{n}_1) \geq 0$$

(식 4-2)

$(\boldsymbol{\omega}_2 \times \mathbf{r}_2) \cdot \mathbf{n}_1 = \boldsymbol{\omega}_2 \cdot (\mathbf{r}_2 \times \mathbf{n}_1)$이다.

또, $\mathbf{x}_2 + \mathbf{r}_2 - \mathbf{x}_1 - \mathbf{r}_1 \approx \mathbf{0}$ 이므로, $(\mathbf{x}_2 + \mathbf{r}_2 - \mathbf{x}_1 - \mathbf{r}_1) \cdot (\boldsymbol{\omega}_1 \times \mathbf{n}_1) \approx 0$ 이다. 그래서,

$$J_i \mathbf{V} = \begin{bmatrix} -\mathbf{n}_1{}^T & -(\mathbf{r}_1 \times \mathbf{n}_1)^T & \mathbf{n}_1{}^T & (\mathbf{r}_2 \times \mathbf{n}_1)^T \end{bmatrix} \begin{bmatrix} \mathbf{v}_1 \\ \boldsymbol{\omega}_1 \\ \mathbf{v}_2 \\ \boldsymbol{\omega}_2 \end{bmatrix} \geq 0, \qquad 0 \leq \lambda_i < \infty$$

(식 4-3)

PENETRATION

한 물체가 다른 물체 내부로 뚫고 침투하게 되면 그 만큼 보정을 해 줘야 한다.

그래서, (식 4-2)를

$$\zeta_i = -\beta \min(C_i, \quad 0)$$

(식 4-4)

$$\dot{C}_i = (\mathbf{v}_2 + \boldsymbol{\omega}_2 \times \mathbf{r}_2 - \mathbf{v}_1 - \boldsymbol{\omega}_1 \times \mathbf{r}_1) \cdot \mathbf{n}_1 + (\mathbf{x}_2 + \mathbf{r}_2 - \mathbf{x}_1 - \mathbf{r}_1) \cdot (\boldsymbol{\omega}_1 \times \mathbf{n}_1) \geq \frac{\zeta_i}{\Delta t}$$

(식 4-5)

로 변형한다. 그러면, (식 4-3)도

$$J_i(t)\mathbf{V}(t + \Delta t) = \begin{bmatrix} -\mathbf{n}_1{}^T & -(\mathbf{r}_1 \times \mathbf{n}_1)^T & \mathbf{n}_1{}^T & (\mathbf{r}_2 \times \mathbf{n}_1)^T \end{bmatrix} \begin{bmatrix} \mathbf{v}_1 \\ \boldsymbol{\omega}_1 \\ \mathbf{v}_2 \\ \boldsymbol{\omega}_2 \end{bmatrix} \geq \zeta_i(t), \qquad 0 \leq \lambda_i < \infty$$

(식 4-6)

로 변형된다.

Constraint Bias 의 β 를 적당히 설정해서 조금씩 양쪽 물체를 떨어뜨리면서 오차를 보정한다.

FRICTION CONSTRAINT

마찰력은 두 개의 Constraint 를 이용해 구현한다.

접촉면의 수직방향이 $\mathbf{n}$ 이고, 접촉면에 접하는 두 개의 벡터를 $\mathbf{u}_1, \mathbf{u}_2$ 라고 하고 $\mathbf{u}_1, \mathbf{u}_2, \mathbf{n}$ 이 서로 수직인 단위벡터라고 하면

$$\mathbf{r}_1 = \mathbf{c} - \mathbf{x}_1$$

$$\mathbf{r}_2 = \mathbf{c} - \mathbf{x}_2$$

$$\dot{C}_i = (\mathbf{v}_2 + \boldsymbol{\omega}_2 \times \mathbf{r}_2 - \mathbf{v}_1 - \boldsymbol{\omega}_1 \times \mathbf{r}_1) \cdot \mathbf{u}_1$$

$$\dot{C}_j = (\mathbf{v}_2 + \boldsymbol{\omega}_2 \times \mathbf{r}_2 - \mathbf{v}_1 - \boldsymbol{\omega}_1 \times \mathbf{r}_1) \cdot \mathbf{u}_2$$

$$J_{i,j}\mathbf{V} = \begin{bmatrix} -\mathbf{u}_1{}^T & -(\mathbf{r}_1 \times \mathbf{u}_1)^T & \mathbf{u}_1{}^T & (\mathbf{r}_2 \times \mathbf{u}_1)^T \\ -\mathbf{u}_2{}^T & -(\mathbf{r}_1 \times \mathbf{u}_2)^T & \mathbf{u}_2{}^T & (\mathbf{r}_2 \times \mathbf{u}_2)^T \end{bmatrix} \begin{bmatrix} \mathbf{v}_1 \\ \boldsymbol{\omega}_1 \\ \mathbf{v}_2 \\ \boldsymbol{\omega}_2 \end{bmatrix} = \mathbf{0}$$

여기서, $\mathbf{c}$ 는 두 물체의 접촉 점이고, $\mathbf{x}_1$과 $\mathbf{x}_2$는 두 물체의 무게중심이다.

이 때, 물체 1 에 가해지는 마찰력은 $-\lambda_i \mathbf{u}_1 - \lambda_j \mathbf{u}_2$이 되고, 물체 2 에는 $\lambda_i \mathbf{u}_1 + \lambda_j \mathbf{u}_2$의 마찰력이 가해진다. 원래 마찰력의 최대값은 수직 방향으로 가해지는 힘에 비례해야 하지만, 아래와 같이 간단하게 λ_i, λ_j 의 범위를 제한해서 구현할 수도 있다.

$$-\mu m_c g \le \lambda_i \le \mu m_c g$$

$$-\mu m_c g \le \lambda_j \le \mu m_c g$$

여기서 μ는 두 물체간의 마찰 계수(friction coefficient)이고, m_c는 질량이고 g 는 중력 가속도 계수($9.8\ {}^{m}/_{sec^2}$)이다.

그러나, 대부분의 물리엔진에서는 좀 더 정확한 계산을 위해 Contact Constraint 에 의한 반작용 힘을 먼저 계산한 후, 그 크기가 λ_c라면

$$-\mu\lambda_c \le \lambda_i \le \mu\lambda_c$$

$$-\mu\lambda_c \le \lambda_j \le \mu\lambda_c$$

로 마찰력의 최대/최소값을 정한다.

즉, **(식 3-26)**의 세 번째 형식에

$$\lambda^{-}_{i} = -\mu\lambda_c$$

$$\lambda^{+}_{i} = \mu\lambda_c$$

$$\lambda^{-}_{j} = -\mu\lambda_c$$

$$\lambda^{+}_{j} = \mu\lambda_c$$

를 대입해서 마찰력을 계산한다.

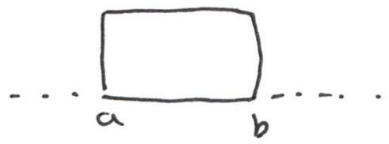
엄밀히 말하면 위 문제는 MLCP 가 아니다. 계산도중 λ_c가 계속 변하므로 λ^{-}_{i}, λ^{+}_{i}는 상수가 아니기 때문이다. 하지만, Project Gauss-Seidel 알고리즘을 약간 수정하면 위 문제도 풀 수 있다. 우선, λ_i,λ_j를 계산하기 전에 λ_c를 먼저 계산할 수 있도록, Jacobian 행렬에서 Friction constraint 를 Contact constraint 보다 아래쪽 행에 배치한다. 그리고, λ^{-}_{i}, λ^{+}_{i}, λ^{-}_{j}, λ^{+}_{j}의 값을 읽어 들일 때, 바뀐 λ_c값을 반영하도록 알고리즘을 수정한다.

CONTACT CACHING

Projected Gauss-Seidel 방법을 보면, 추측 값 $\boldsymbol{\lambda_0}$가 입력으로 들어간다. 이전 결과 $\boldsymbol{\lambda}$를 보관하고 있다가, 다음 번에 계산할 때 $\boldsymbol{\lambda_0}$ 의 입력 값으로 이용하면, 더 빨리 결과 값에 수렴할 수 있다.

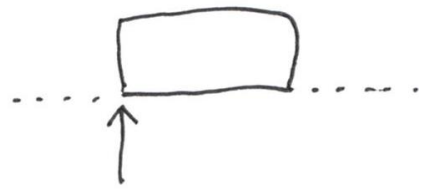
또, Contact Cache 를 이용하는 예에 대해서 살펴보자.

2 차원인 경우에 대해 설명하겠다.



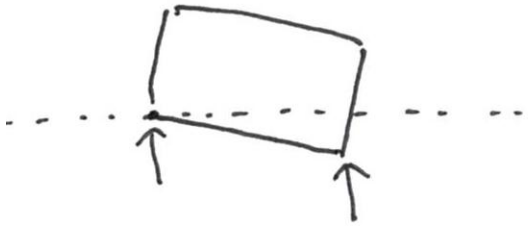
(그림 4-2)

네모난 물체가 아래 바닥 면과 충돌하려고 하는 순간이다. 물체가 바닥 면 위에서 유지하려면 점 a 와 점 b 두 군데서의 충돌 정보가 필요하다. 하지만, 충돌 알고리즘에서는 가장 깊숙하게 충돌한 위치 한 개만 찾아낼 수 있다. 그래서 점 a (또는 점 b)만 찾아낸다.



(그림 4-3)

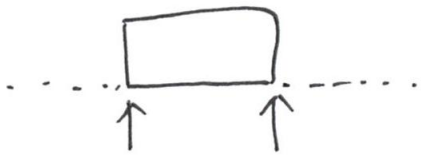
그래서 물리 엔진에서는 **(그림 4-3)**와 같이 반작용 힘을 계산한다.



(그림 4-4)

그 결과, **(그림 4-4)**같이 점 b 가 바닥 면 밑으로 침투하여 충돌하게 된다. 그 다음, 충돌 처리 부분에서 가장 깊숙이 충돌한 점 b 를 찾아낸다. 이 때 점 a 는 아직 충돌하고 있는 상태이므로, cache 에 저장되어 있다. 그러면 점 a 와 점 b 모두 물리엔진에서 Contact Constraint 로 처리된다.

그래서, 양쪽에서 힘을 받는다. 점 b 가 침투되어 있는 부분은 Constraint bias 에 의해 보정된다.



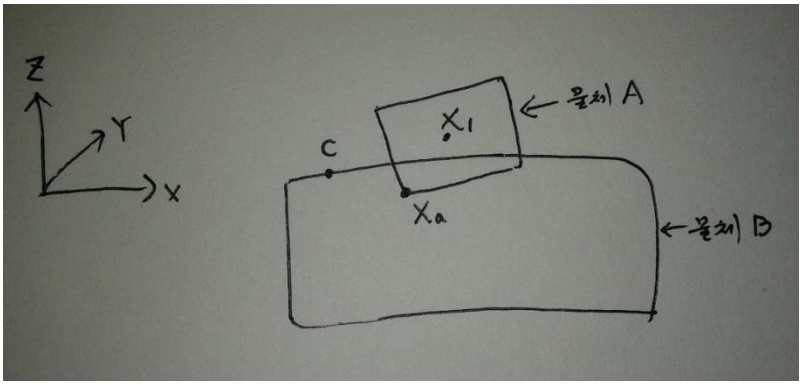
(그림 4-5)

그 다음 (그림 4-5)처럼 양쪽에서 힘을 받는 안정적인 상태가 되어 더 이상 움직이지 않는다.

5 총정리

지금까지 배운 것을 종합해서 실제 물리엔진이 동작하는 예를 연구해 보자.

$$T = \boldsymbol{\tau}$$



(그림 5-1)

(그림 5-1)에서 물체 A 의 질량을 m_1, 관성텐서를 $I_1(\tau)$이라고 하자.

그리고 물체 B 는 움직이지 않는 고정된 물체라고 가정하자.

그러면, 물체 B 의 질량은 $m_2{}^{-1} = 0$, 관성텐서는 $I_2{}^{-1}(\tau) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$가 되고,

$$M^{-1}(\tau) = \begin{bmatrix} m_1{}^{-1}E & 0 & 0 & 0 \\ 0 & I_1{}^{-1}(\tau) & 0 & 0 \\ 0 & 0 & m_2{}^{-1}E & 0 \\ 0 & 0 & 0 & I_2{}^{-1}(\tau) \end{bmatrix}$$

(식 5-1)

가 된다.

물체 A 에 중력의 힘이 주어지면 $\mathbf{f}_1(\tau) = m_1 \begin{bmatrix} 0 \\ 0 \\ -9.8\ m/sec^2 \end{bmatrix}$ 이고,

$$\mathbf{F}_{ext}(\tau) = \begin{bmatrix} \mathbf{f}_1(\tau) \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} (12\ x\ 1)$$

(식 5-2)

가 된다.

그리고, 초기 상태의 속도가 0 이라고 하면,

$$\mathbf{V}(\tau) = \begin{bmatrix} \mathbf{v}_1(\tau) \\ \boldsymbol{\omega}_1(\tau) \\ \mathbf{v}_2(\tau) \\ \boldsymbol{\omega}_2(\tau) \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} (12\ x\ 1)$$

(식 5-3)

가 된다.

물리 엔진에서는 각 프레임마다 물체간의 충돌 여부를 검사해서, 충돌 점과 충돌 면을 구한다.

그림에서 충돌 면의 방향은 $\mathbf{N} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$이고, 충돌 평면은 $\mathbf{N} \cdot \mathbf{x} \geq \mathbf{N} \cdot \mathbf{c}$ 가 된다. 또 충돌 점 $\mathbf{x}_a(\tau)$ 는 $\mathbf{N} \cdot \mathbf{x}_a(\tau) < \mathbf{N} \cdot \mathbf{c}$ 이므로 Constraint 를 위반하고 있는 상태다.

물체 A 의 무게중심을 $\mathbf{x}_1$, 속도를 $\mathbf{v}_1$, 각속도를 $\boldsymbol{\omega}_1$이라고 하면 충돌 점 $\mathbf{x}_a$ 의 속도는

$$\mathbf{v}_a(t) = \mathbf{v}_1(t) + \boldsymbol{\omega}_1(t) \times \left(\mathbf{x}_a(\tau) - \mathbf{x}_1(\tau)\right)$$

(식 5-4)

그리고, Constraint 는

$$\mathbf{N} \cdot \mathbf{v}_a(t) \geq 0$$

(식 5-5)

여야 한다.

그런데, t= τ 일 때 **(식 5-5)**의 Constraint 를 위반하고 있는 상태이므로, 에러를 보정해 주어야 한다.

$$\varepsilon = \min\left(\mathbf{N} \cdot \mathbf{x}_a(\tau) - \mathbf{N} \cdot \mathbf{c}, \quad 0\right)$$

$$\zeta_1(\tau) = -\beta\varepsilon, \quad 0 \leq \beta \leq 1$$

(식 5-6)

그래서, **(식 5-5)**의 Constraint 는

$$\mathbf{N} \cdot \mathbf{v}_a(t) \geq \zeta_1(\tau)$$

(식 5-7)

으로 바뀐다.
t= τ 일 때의 에러를 보정하여, $t = \tau + \Delta t$일 때 **(식 5-7)**의 constraint 를 만족해야 한다.

이제부터는 $t = \tau + \Delta t$ 일 때의 물체 A 의 무게중심 $\mathbf{x}_1(t)$, 회전행렬 $R_1(t)$, 속도 $\mathbf{v}_1(t)$, 회전 각속도 $\boldsymbol{\omega}_1(t)$를 구해보자.

(식 5-4)을 대입하면,

$$\mathbf{N} \cdot \mathbf{v}_1(t) + \mathbf{N} \cdot \left(\boldsymbol{\omega}_1(t) \times \left(\mathbf{x}_a(\tau) - \mathbf{x}_1(\tau)\right)\right) \geq \zeta_1(\tau)$$

(엄밀하게 말하면, 위 식에서 $\mathbf{x}_a(\tau + \Delta t) - \mathbf{x}_1(\tau + \Delta t)$를 사용해야 정확하지만, 아직 그 값을 구할 수 없으므로 $\mathbf{x}_a(\tau) - \mathbf{x}_1(\tau)$으로 대신한다.)

그런데, 임의의 벡터 $\mathbf{x}, \mathbf{y}, \mathbf{z}$에 대해 $\mathbf{Det}([\mathbf{x} \quad \mathbf{y} \quad \mathbf{z}]) = \mathbf{x} \cdot (\mathbf{y} \times \mathbf{z}) = (\mathbf{x} \times \mathbf{y}) \cdot \mathbf{z}$ 이므로

$$\mathbf{N} \cdot \mathbf{v}_1(t) + \left(\left(\mathbf{x}_a(\tau) - \mathbf{x}_1(\tau)\right) \times \mathbf{N}\right) \cdot \boldsymbol{\omega}_1(t) \geq \zeta_1(\tau)$$

$$\mathbf{N}^{\mathrm{T}} \mathbf{v}_1(t) + \left(\left(\mathbf{x}_a(\tau) - \mathbf{x}_1(\tau)\right) \times \mathbf{N}\right)^{\mathrm{T}} \boldsymbol{\omega}_1(t) \geq \zeta_1(\tau)$$

여기서,

$$J_1(\tau) = \left[\mathbf{N}^{\mathrm{T}} \quad \left(\left(\mathbf{x}_a(\tau) - \mathbf{x}_1(\tau)\right) \times \mathbf{N}\right)^{\mathrm{T}} \quad \mathbf{0}^{\mathrm{T}} \quad \mathbf{0}^{\mathrm{T}}\right] \ (1 \times 12)$$

(식 5-8)

라고 하면,

$$J_1(\tau)\,\mathbf{V}(t) \geq \zeta_1(\tau)$$

로 요약할 수 있다.

$$\boldsymbol{\lambda} = [\lambda_1]\ (1 \text{ x } 1)$$

(식 5-9)

를 **(식 3-11)**에 대입하면

$$\begin{bmatrix}\mathbf{f}_{c1}(\tau)\\ \boldsymbol{\tau}_{c1}(\tau)\end{bmatrix} = J_1{}^{T}(\tau)\,\lambda_1$$

이다.

그러므로, 반작용에 의해 물체 A 에 $\mathbf{f}_{c1}(\tau) = \lambda_1\mathbf{N}$ 의 힘과, $\boldsymbol{\tau}_{c1}(\tau) = \left(\mathbf{x}_a(\tau) - \mathbf{x}_1(\tau)\right) \times \lambda_1\mathbf{N}$ 의 Torque 가 추가되는 것을 알 수 있다. 즉 $\mathbf{x}_a(\tau)$ 지점에 $\lambda_1\mathbf{N}$ 의 반작용 힘이 가해진다. 즉 λ_1를 구하는 것은 이 반작용 힘의 크기를 구하는 것이다.

또, $\lambda_1 \geq 0$ 이어야 하고,

$$J_1(\tau)\,\mathbf{V}(t) \geq \zeta_1(\tau)$$

(식 5-10)

이어야 한다.

$$\lambda^{-}{}_1 = 0, \qquad \lambda^{+}{}_1 = \infty$$

로 설정하면 **(식 3-26)**의 두 번째 조건과 같아진다. 여기서 MLCP 의 성질에 의해

$$J_1(\tau)\,\mathbf{V}(t) > \zeta_1(\tau) \quad \rightarrow \quad \lambda_1 = 0$$

인데, 이것은 두 물체가 서로 떨어지게 되면 반작용 힘도 없어진다는 물리적 성질과 일치한다.

이제 MLCP 에 필요한 모든 입력이 준비되었다.

$$J(\tau) = [J_1(\tau)]$$

(식 5-11)

$$\boldsymbol{\zeta}(\tau) = \left[\zeta_1(\tau)\right]$$

(식 5-12)

이라고 하고, (식 3-23)에 적용하면,

$$\mathbf{W}(t) = J(\tau)B(\tau)\boldsymbol{\lambda} - \boldsymbol{\eta}(\tau)$$

$$B(\tau) = M^{-1}(\tau)J^{T}(\tau), \qquad \boldsymbol{\eta}(\tau) = \frac{1}{\Delta t}\boldsymbol{\zeta}(\tau) - J(\tau)\left(\frac{1}{\Delta t}\mathbf{V}(\tau) + M^{-1}(\tau)\mathbf{F}_{ext}(\tau)\right)$$

인데, 위 MLCP 방정식을 풀면, λ_1의 값을 구할 수 있다.

방정식을 풀어서 $\boldsymbol{\lambda}$를 구하고, 그 값을 (식 3-10)과 (식 3-11)에 대입하면

$$\mathbf{F}_c(\tau) = J^{T}(\tau)\boldsymbol{\lambda}$$

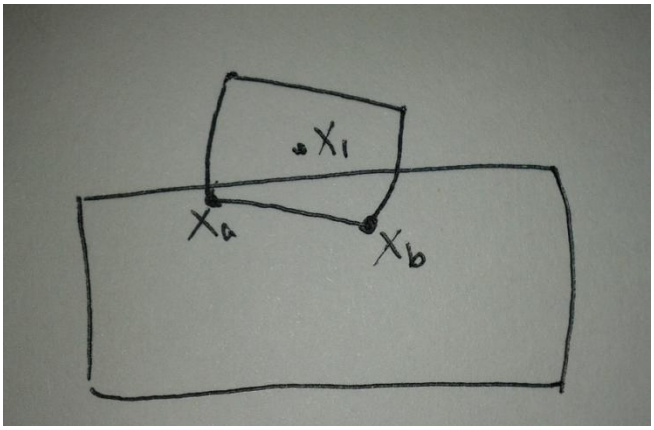
$$\mathbf{F}_c(\tau) = \begin{bmatrix} \mathbf{f}_{c1}(\tau) \\ \boldsymbol{\tau}_{c1}(\tau) \\ \mathbf{f}_{c2}(\tau) \\ \boldsymbol{\tau}_{c2}(\tau) \end{bmatrix} (12 \text{ x } 1)$$

즉, 물체 A 에는 Constraint Force $\mathbf{f}_{c1}(\tau)$와 Constraint Torque $\boldsymbol{\tau}_{c1}(\tau)$가 가해진다.

λ_1의 값을 알면, Force 와 Torque 를 구할 수 있고, 그렇게 되면 물체 A 의 무게중심 $\mathbf{x}_1(\tau + \Delta t)$,

회전행렬 $R_1(\tau + \Delta t)$, 속도 $\mathbf{v}_1(\tau + \Delta t)$, 회전 각속도 $\boldsymbol{\omega}_1(\tau + \Delta t)$ 를 구할 수 있다.

T = $\boldsymbol{\tau} + \Delta \mathbf{t}$



(그림 5-2)

$M^{-1}(\tau+\Delta t), \mathbf{F}_{ext}(\tau+\Delta t)$, $\mathbf{V}(\tau+\Delta t)$는 $t=\tau$일 때와 같은 방식으로 구하면 된다.

그런데, 충돌 점 $\mathbf{x}_a$의 위치가 약간 위로 이동된다.

$$\mathbf{x}_a(\tau+\Delta t) = \mathbf{x}_1(\tau+\Delta t) + R_1(\tau+\Delta t)\,\mathbf{r}_a$$

$$\mathbf{r}_a = {R_1}^T(\tau)\big(\mathbf{x}_a(\tau) - \mathbf{x}_1(\tau)\big)$$

Contact Caching 을 하면서 충돌시의 상대위치 $\mathbf{r}_a = {R_1}^T(\tau)\big(\mathbf{x}_a(\tau) - \mathbf{x}_1(\tau)\big)$와 충돌 평면을 보관해 두면, $\mathbf{x}_a$의 충돌 여부를 쉽게 알아낼 수 있다.

$$\zeta_1(\tau+\Delta t) = -\beta \min\left(\mathbf{N}\cdot\mathbf{x}_a(\tau+\Delta t) - \mathbf{N}\cdot\mathbf{c}, 0\right)$$

(식 5-13)

$$J_1(\tau+\Delta t) = \begin{bmatrix}\mathbf{N}^T & \Big(\big(\mathbf{x}_a(\tau+\Delta t) - \mathbf{x}_1(\tau+\Delta t)\big)\times\mathbf{N}\Big)^T & \mathbf{0}^T & \mathbf{0}^T\end{bmatrix}\ (1\text{ x }12)$$

(식 5-14)

$${\lambda^-}_1 = 0, \qquad {\lambda^+}_1 = \infty$$

(식 5-15)

물체 B 가 움직이지 않는다고 가정했으므로, $\mathbf{N}$ 의 값이 바뀌지 않았다.

그런데, 이번에는 $\mathbf{x}_b$가 더 깊숙하게 충돌하였다.

충돌 검사를 하면, 충돌 점 $\mathbf{x}_b(\tau+\Delta t)$와 충돌 평면의 정보를 알 수 있다.

그리고, 다음 frame 에 $\mathbf{x}_b$의 변경된 위치를 계산하기 쉽게 하기 위해 $\mathbf{r}_b$를 계산해둔다.

$$\mathbf{r}_b = {R_1}^T(\tau+\Delta t)\big(\mathbf{x}_b(\tau+\Delta t) - \mathbf{x}_1(\tau+\Delta t)\big)$$

두 번째 충돌 점에 대한 Constraint 처리도 비슷하게 이루어진다.

$$\zeta_2(\tau+\Delta t) = -\beta \min\left(\mathbf{N}\cdot\mathbf{x}_b(\tau+\Delta t) - \mathbf{N}\cdot\mathbf{c}, \qquad 0\right)$$

(식 5-16)

$$J_2(\tau+\Delta t) = \begin{bmatrix}\mathbf{N}^T & \Big(\big(\mathbf{x}_b(\tau+\Delta t) - \mathbf{x}_1(\tau+\Delta t)\big)\times\mathbf{N}\Big)^T & \mathbf{0}^T & \mathbf{0}^T\end{bmatrix}\ (1\text{ x }12)$$

(식 5-17)

$${\lambda^-}_2 = 0, \qquad {\lambda^+}_2 = \infty$$

(식 5-18)

이번에는 Constraint 가 2 개이므로,

$$\mathrm{J}(\tau+\Delta t)=\begin{bmatrix}\mathrm{J}_1(\tau+\Delta t)\\ \mathrm{J}_2(\tau+\Delta t)\end{bmatrix}\ (2\ \mathrm{x}\ 12)$$

(식 5-19)

$$\boldsymbol{\zeta}(\tau+\Delta t)=\begin{bmatrix}\zeta_1(\tau+\Delta t)\\ \zeta_2(\tau+\Delta t)\end{bmatrix}\ (2\ \mathrm{x}\ 1)$$

(식 5-20)

$$\boldsymbol{\lambda}=\begin{bmatrix}\lambda_1\\ \lambda_2\end{bmatrix}(2\ \mathrm{x}\ 1)$$

(식 5-21)

로 하면

$$\mathrm{J}(\tau+\Delta t)\mathrm{B}(\tau+\Delta t)\boldsymbol{\lambda}=\boldsymbol{\eta}(\tau+\Delta t)$$

$$\mathrm{B}(\tau+\Delta t)=\mathrm{M}^{-1}(\tau+\Delta t)\mathrm{J}^{\mathrm{T}}(\tau+\Delta t)$$

$$\boldsymbol{\eta}(\tau+\Delta t)=\frac{1}{\Delta t}\boldsymbol{\zeta}(\tau+\Delta t)-\mathrm{J}(\tau+\Delta t)\left(\frac{1}{\Delta t}\mathbf{V}(\tau+\Delta t)+\mathrm{M}^{-1}(\tau+\Delta t)\mathbf{F}_{\mathrm{ext}}(\tau+\Delta t)\right)$$

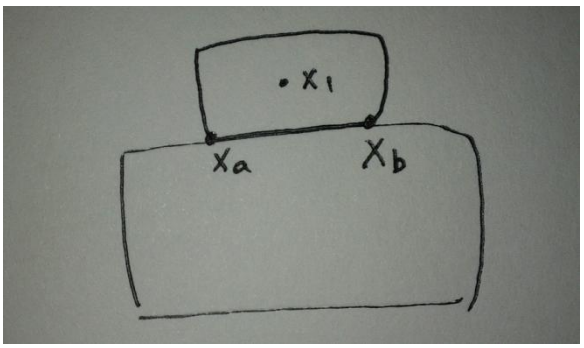
위 MLCP 방정식을 풀어서, λ_1과 λ_2를 구하면 된다.

T = $\boldsymbol{\tau+2\Delta t}$

3 차원인 경우에는, 또 하나의 충돌 점이 생긴다.

이번에는 Constraint 가 3 개가 되고, 3 개의 충돌 지점에서의 반작용 힘을 구하게 된다.

T = ∞



(그림 5-3)

이런 과정을 반복하게 되면, 물체 A 의 Constraint Force $\mathbf{f}_{c1}(t)$는 $m_1 \begin{bmatrix} 0 \\ 0 \\ 9.8\ m/sec^2 \end{bmatrix}$ 에 수렴하게 되고, Constraint Torque $\boldsymbol{\tau}_{c1}(t)$는 0 에 수렴한다. Force 와 Torque 가 균형을 이루게 되어 물체 A 는 움직이지 않게 된다. 그리고, 물체 A 가 물체 B 바깥쪽으로 나가게 되어서 Constraint 도 모두 만족시키게 된다. 하지만 어느 정도 떨어질 때까지 접촉하고 있는 것으로 간주하여, Contact Constraint 를 유지하여야 한다. 그렇지 않으면, 순간적으로 반작용 힘이 없어져서, 물리엔진이 불안정해진다.

6 결론

물리엔진에 관하여 공부하는 사람이 거의 없다 보니, 관련 자료를 찾기가 쉽지 않다. 나도 자료를 찾느라고 오랫동안 고생했었다. 아무쪼록 이 자료가 나중에 공부하는 후배들에게 도움이 되었으면 한다.

참고문서

[1] Erin Catto. Iterative dynamics with temporal coherence.

http://www.continuousphysics.com/ftp/pub/test/physics/papers/IterativeDynamics.pdf

box 2d 을 만든 Erin Catto 가 쓴 자료이다. 개인적으로 가장 함축적으로 잘 설명되어 있는 자료라고 생각한다. 하지만 물리에 대한 기초지식이 있어야 읽을 수 있다.

[2] http://www.cs.cmu.edu/~baraff/sigcourse/index.html Lecture Notes

[3] David Baraff. Fast contact force computation for nonpenetrating rigid bodies.

In Proceedings of SIGGRAPH 1994, pages 23–34, 1994.

David Baraff 의 논문을 읽어보면 초기 물리 엔진의 구현 방법에 대해 알 수 있다. MLCP(Mixed Linear Complementary Problem)의 해결방법이 Erin Catto 의 방법보다 복잡하고 어렵지만, 어떻게 해서 이런 방법을 나왔는지에 대해 추측해 볼 수 있다. Erin Catto 의 자료보다 좀 더 자세하게 설명되어 있다.

[4] David H. Eberly. Game Physics. Second Edition.

[5] David H. Eberly. 3D Game Engine Design

물리 엔진의 기초 지식을 알기 위해서는 책을 통해서 익히는 게 좋다. 기초 물리에 대하여 설명한 책이다.

[6] ODE http://ode.org/

[7] bullet http://www.bulletphysics.com

[8] box2d http://www.box2d.org

공개 물리엔진들이다. 소스를 참조할 수 있으므로 공부하기에 좋다. [8] box2d 는 2D 에 특화된 물리엔진이다. [7] bullet 엔진은 상용 게임에서도 사용될 정도로 잘 구현되어 있다.

[9] Gino van den Bergen. Collision Detection in Interactive 3D Environments

[10] Christer Ericson. Real-Time Collision Detection

충돌관련 알고리즘에 대한 설명을 하고 있는 책이다. [9] Gino van den Bergen.이 [10] Christer Ericson. 보다 더 얇고, EPA 알고리즘에 대한 설명이 더 들어있다. 특히 GJK, EPA 알고리즘 모두 중요하니 잘 알아두기 바란다.

[11] A Unified Framework for Rigid Body Dynamics (Chapter 1-6)

구글에서 문서를 찾을 수 있다. 다양한 물리 엔진의 구현방법에 대해서 상세하게 설명이 되어 있다.

[12] Roger A. Horn. Matrix Analysis

매트릭스에 관한 수학책이다. 다소 어려운 게 흠이지만, 시간 많고 수학을 좋아하면 읽어볼 만하다. 특히, eigen vector 를 이용해 매트릭스를 분해(Eigen decomposition)하는 방법이 아주 자세히 나와있다.

[13] Ren-Cang Li, Iterative Schemes and Their Convergence For a Symmetric Positive Definite Matrix

Gauss-Seidel Method 에서 해의 수렴 조건에 대한 증명이 잘 나와있다.
https://sites.google.com/site/doc4code/source/Iterative%20Schemes%20Convergence.pdf
에 복사해 두었다.