GAMES103 - Intro to Physics-Based Animation

(Based on Unity, C# lang)

> Huamin Wang (games103@style3D.com)   Video; Lecture site

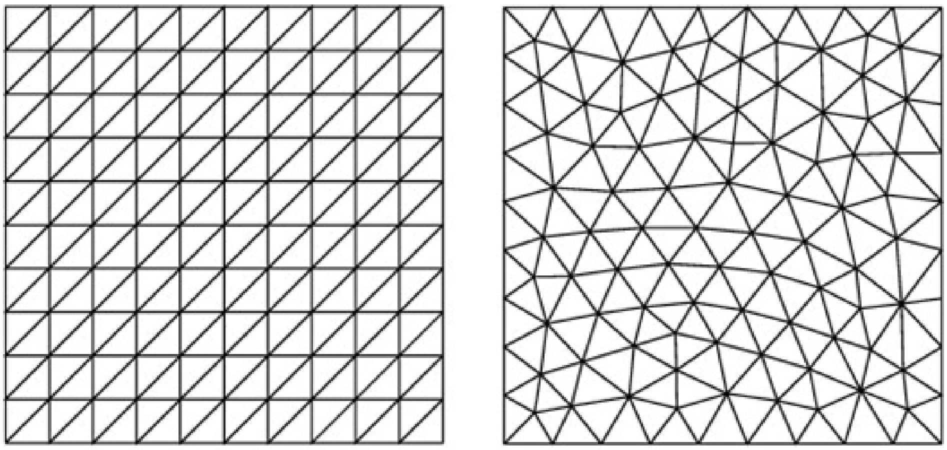
Lecture 1 Introduction

Graphics

Geometry

- **Mesh**: Triangle mesh is the foundation of graphics

  Vertices (nodes) + Elements (triangles, polygons, tetrahedra...)

  (a) Structured mesh   (b) Unstructured mesh

- **Point Cloud**: simple, can be raw data from surface scan.

  But problems in mesh reconstruction, (re)-sampling, neighborhood search ...

- **Grid**: often acquired from volumetric scan, e.g., CT

  Problems in memory cost (octree?), volumetric rendering

Rendering

- Photorealistic Rendering (Ray Tracing)
- Non-Photorealistic Rendering

Material Scan

- Body Scan by a Light Stage

Animations

- Character Animation
- Physics-Based Animation

Physics-Based Animation Topics

- **Rigid Bodies** [contact / fracture] - Mesh / *Particle in fracture (to avoid remeshing)
- **Cloth and Hair** [clothes / hair] - Mesh / *Grid (to simplify contacts)
- **Soft Bodies** [elastic / plastic] - Mesh
- **Fluids** [smoke / drops and waves / splashes] - Mesh in drops and waves (RT) / Particle (RT) in smoke and splashes / Grid (universal)

Lecture 2 Math Background

> **Vector, Matrix and Tensor Calculus**

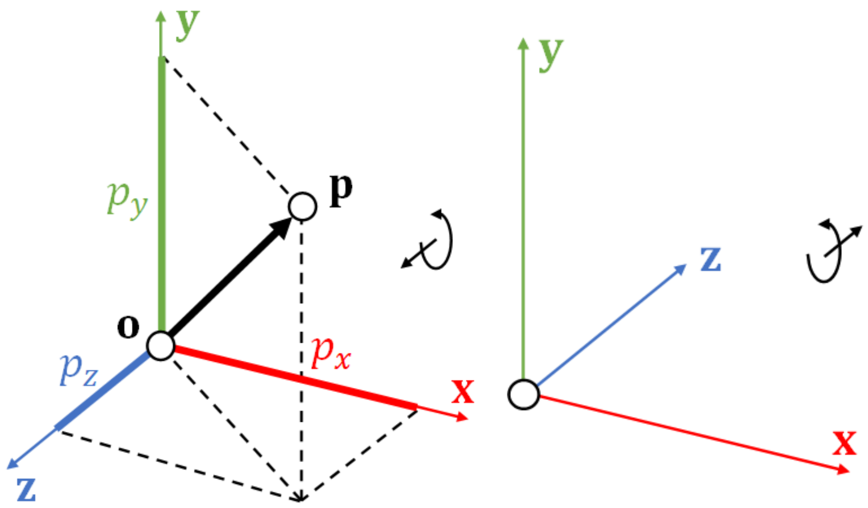
Vectors

Definitions

A geometric entity endowed with magnitude and direction

$$\mathbf{p} = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \in \mathbb{R}^3 \; ; \quad \mathbf{o} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

- Right-Hand System (OpenGL, Research, ...)
- Left-Hand System (Unity, DirectX, ...) => screen space



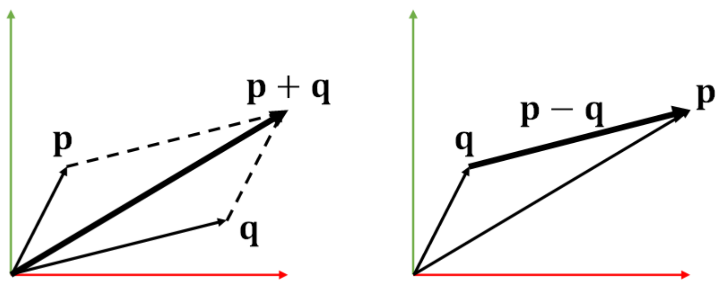
Can also be stacked up to form a high-dim vector -> describe the state of an obj (not a geometric vector but a stacked vector)

Arithematic

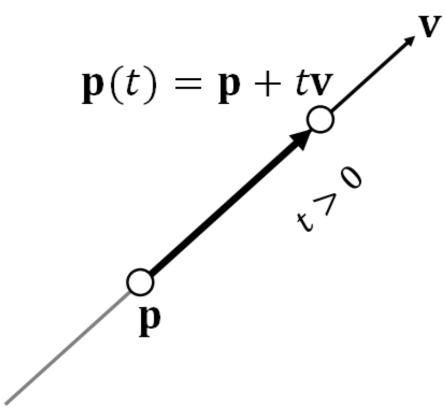
Addition and Substraction

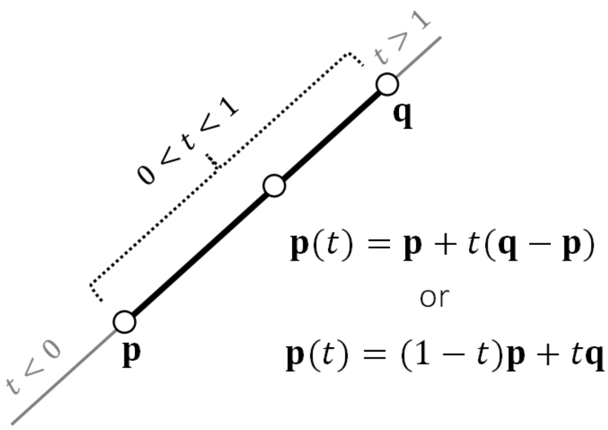
(commutative)

$$\mathbf{p} \pm \mathbf{q} = \begin{bmatrix} p_x \pm q_x \\ p_y \pm q_y \\ p_z \pm q_z \end{bmatrix} \; ; \quad \mathbf{p} + \mathbf{q} = \mathbf{q} + \mathbf{p}$$



Example: $\mathbf{p}(t) = \mathbf{p} + t\mathbf{v}$ to represent the movement of a particle. Segment: $0 < t < 1$; Ray: $0 < t$; Line: $t \in \mathbb{R}$ (t is an interpolant)





Vector Norm

Magnitude of a vector (length)

- **1-norm**: $||\mathbf{p}||_1 = |p_x| + |p_y| + |p_z|$
- **Euclidean (2) norm** (default): $||\mathbf{p}||_2 = (p_x^2 + p_y^2 + p_z^2)^{1/2}$
- **p-norm**: $||\mathbf{p}||_p = (p_x^p + p_y^p + p_z^p)^{1/p}$
- **Inifinity norm** (Maximum): $||\mathbf{p}||_\infty = \max(|p_x|, |p_x|, |p_x|)$

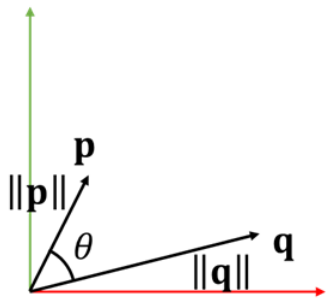
Usage:

- **Distance** between $\mathbf{p}$ and $\mathbf{q}$: $||\mathbf{q} - \mathbf{p}||$
- **Unit Vector**: $||\mathbf{p}|| = 1$
- **Normalization**: $\bar{\mathbf{p}} = \mathbf{p}/||\mathbf{p}||$ as $||\bar{\mathbf{p}}|| = ||\mathbf{p}||/||\mathbf{p}|| = 1$

Dot Product

(inner product)

$$< \mathbf{p}, \mathbf{q} >\equiv \mathbf{p} \cdot \mathbf{q} = p_x q_x + p_y q_y + p_z q_z = \mathbf{p}^{\mathrm{T}}\mathbf{q} = ||\mathbf{p}|| \; ||\mathbf{q}|| \cos\theta$$



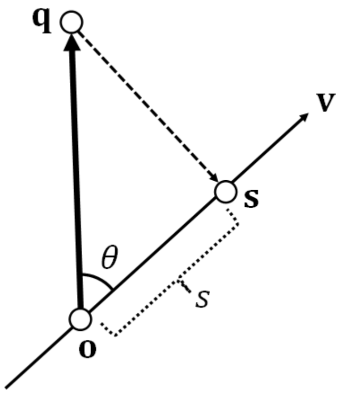
Properties:

- $\mathbf{p} \cdot \mathbf{q} = \mathbf{q} \cdot \mathbf{p}$
- $\mathbf{p} \cdot (\mathbf{q} + \mathbf{r}) = \mathbf{p} \cdot \mathbf{q} + \mathbf{p} \cdot \mathbf{r}$
- $\mathbf{p} \cdot \mathbf{p} = ||\mathbf{p}||_2^2$, an alternative way to write norm
- If $\mathbf{p} \cdot \mathbf{q} = 0$ and $\mathbf{p}, \mathbf{q} \neq 0$, then $\cos\theta = 0$ => **orthogonal**

Example: Particle-Line Projection

By def: $s = ||\mathbf{q} - \mathbf{o}|| \;\; \cos\theta$ => $\mathbf{s} = \mathbf{o} - s\bar{\mathbf{v}}$

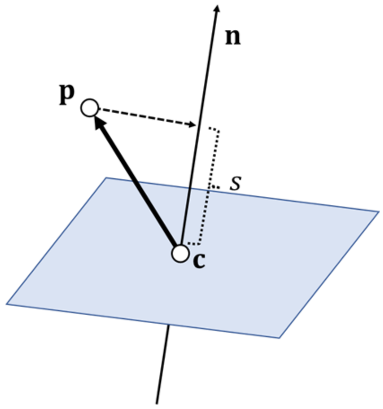
$$\begin{aligned} s &= ||\mathbf{q} - \mathbf{o}|| \;\; \cos\theta \\ &= ||\mathbf{q} - \mathbf{o}|| \; ||\mathbf{v}|| \cos\theta \;/||\mathbf{v}|| \\ &= (\mathbf{q} - \mathbf{o})^{\mathrm{T}}\mathbf{v}/||\mathbf{v}|| \\ &= (\mathbf{q} - \mathbf{o})^{\mathrm{T}}\bar{\mathbf{v}} \quad \text{(normalized)} \end{aligned}$$



Example: Plane Representation

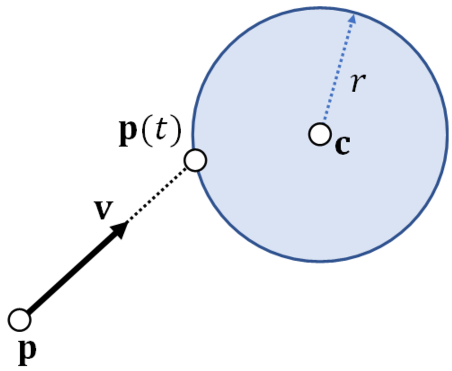
Check the relationship between point p and plane (s - the signed distance to the plane -> collision check / ...; $\mathbf{n}$ - normal)

$$s = (\mathbf{p} - \mathbf{c})^{\mathrm{T}}\mathbf{n} \begin{cases} > 0 & \text{Above the plane} \\ = 0 & \text{On the plane} \\ < 0 & \text{Below the plane} \end{cases}$$



Example: Particle-Sphere Collision

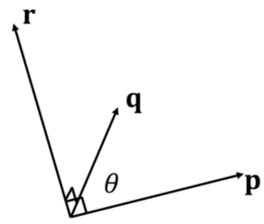
$$\begin{aligned} ||\mathbf{p}(t) - \mathbf{c}||^2 &= r^2 \\ (\mathbf{p} - \mathbf{c} + t\mathbf{v}) \cdot (\mathbf{p} - \mathbf{c} + t\mathbf{v}) &= r^2 \\ (\mathbf{v} \cdot \mathbf{v})t^2 + 2(\mathbf{p} - \mathbf{c}) \cdot \mathbf{v}t + (\mathbf{p} - \mathbf{c}) \cdot (\mathbf{p} - \mathbf{c}) - r^2 &= 0 \end{aligned}$$



t is the root -> No root (no collision) / One root (tangentially) / Two roots (the first point, smaller t)

Cross Product

$$\mathbf{r} = \mathbf{p} \times \mathbf{q} = \begin{bmatrix} p_y q_z - p_z q_y \\ p_z q_x - p_x q_z \\ p_x q_y - p_y q_x \end{bmatrix}$$

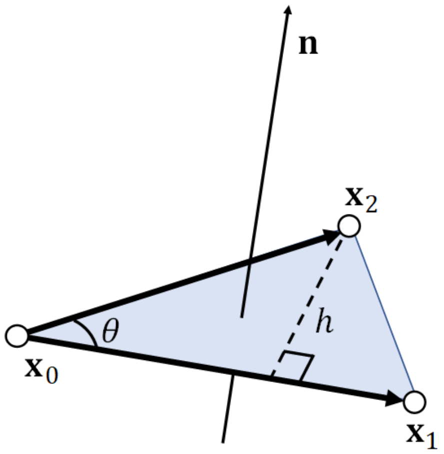


Properties

- $\mathbf{r}\cdot\mathbf{p}=0$; $\mathbf{r}\cdot\mathbf{q}=0$; $||\mathbf{r}||=||\mathbf{p}||\,||\mathbf{q}||\,\sin\theta$
- $\mathbf{p}\times\mathbf{q}=-\mathbf{q}\times\mathbf{p}$
- $\mathbf{p}\times(\mathbf{q}+\mathbf{r})=\mathbf{p}\times\mathbf{q}+\mathbf{p}\times\mathbf{r}$
- If $\mathbf{p}\times\mathbf{q}=\mathbf{0}$ and $\mathbf{p},\ \mathbf{q}\neq 0$, then $\sin\theta=0$, $\mathbf{p}$ & $\mathbf{q}$ are **parallel** (direction can be opposite)

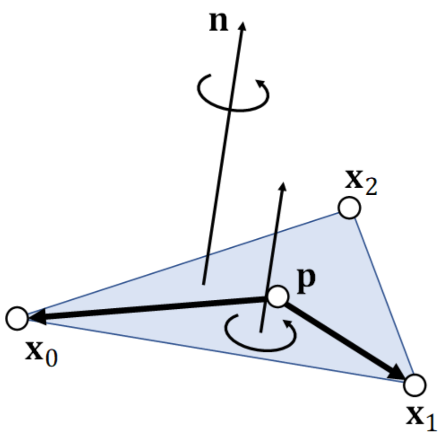
Example: Trinagle Normal and Area

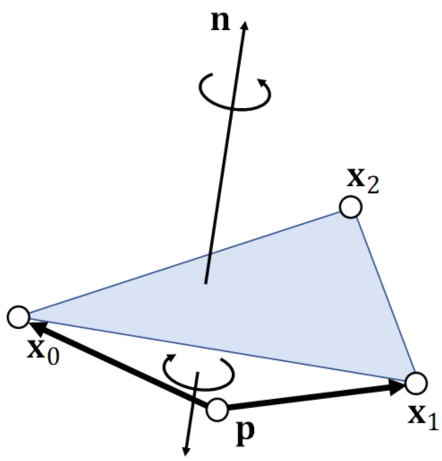
- Edge vectors: $\mathbf{x}_{10}=\mathbf{x}_1-\mathbf{x}_0$ & $\mathbf{x}_{20}=\mathbf{x}_2-\mathbf{x}_0$
- Normal: $\mathbf{n}=(\mathbf{x}_{10}\times\mathbf{x}_{20})/||\mathbf{x}_{10}\times\mathbf{x}_{20}||$ (dep on the topological order)
- Area: $A=||\mathbf{x}_{10}||h/2=||\mathbf{x}_{10}\times\mathbf{x}_{20}||/2$



Example: Trianle Inside / Outside Test (Co-plane)

- If inside $\mathbf{x}_0\mathbf{x}_1$: $(\mathbf{x}_0-\mathbf{p})\times(\mathbf{x}_1-\mathbf{p})\cdot\mathbf{n}>0$ (Same normal as the main triangle)
- If outside $\mathbf{x}_0\mathbf{x}_1$: $(\mathbf{x}_0-\mathbf{p})\times(\mathbf{x}_1-\mathbf{p})\cdot\mathbf{n}<0$





Example: Barycentric Coordinates

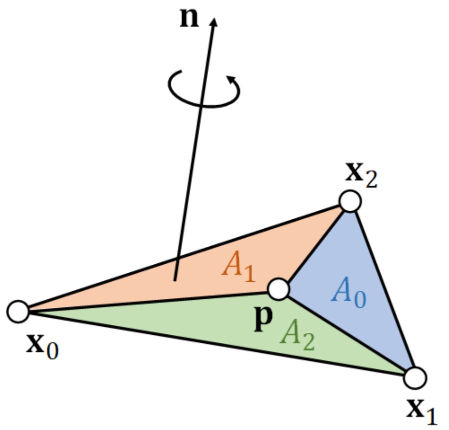
$$\frac{1}{2}(\mathbf{x}_0-\mathbf{p})\times(\mathbf{x}_1-\mathbf{p})\cdot\mathbf{n}=\begin{cases}\frac{1}{2}||(\mathbf{x}_0-\mathbf{p})\times(\mathbf{x}_1-\mathbf{p})|| & \text{inside}\\ -\frac{1}{2}||(\mathbf{x}_0-\mathbf{p})\times(\mathbf{x}_1-\mathbf{p})|| & \text{outside}\end{cases}$$

Signed Areas:

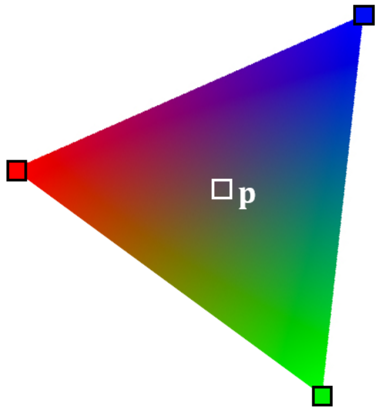
$$
\begin{aligned}
A_2 &= \frac{1}{2}(\mathbf{x}_0 - \mathbf{p}) \times (\mathbf{x}_1 - \mathbf{p}) \cdot \mathbf{n} \\
A_0 &= \frac{1}{2}(\mathbf{x}_1 - \mathbf{p}) \times (\mathbf{x}_2 - \mathbf{p}) \cdot \mathbf{n} \\
A_1 &= \frac{1}{2}(\mathbf{x}_2 - \mathbf{p}) \times (\mathbf{x}_0 - \mathbf{p}) \cdot \mathbf{n} \\
A &= A_0 + A_1 + A_2
\end{aligned}
$$

Barycentric weights of $\mathbf{p}$: $b_0 = A_0/A$, $b_1 = A_1/A$, $b_2 = A_2/A$ $(b_0 + b_1 + b_2 = 1)$

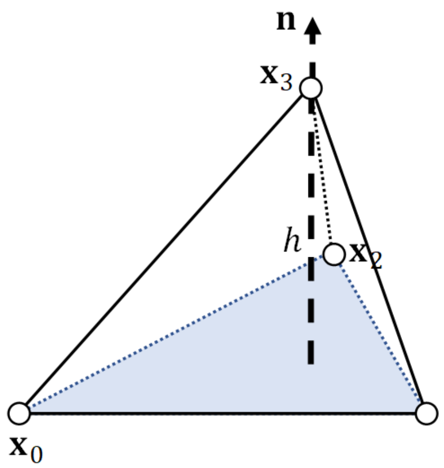
Barycentric Interpolation: $\mathbf{p} = b_0\mathbf{x}_0 + b_1\mathbf{x}_1 + b_2\mathbf{x}_2$



=> **Gourand Shading**: Using barycentric interpolation (no longer popular)

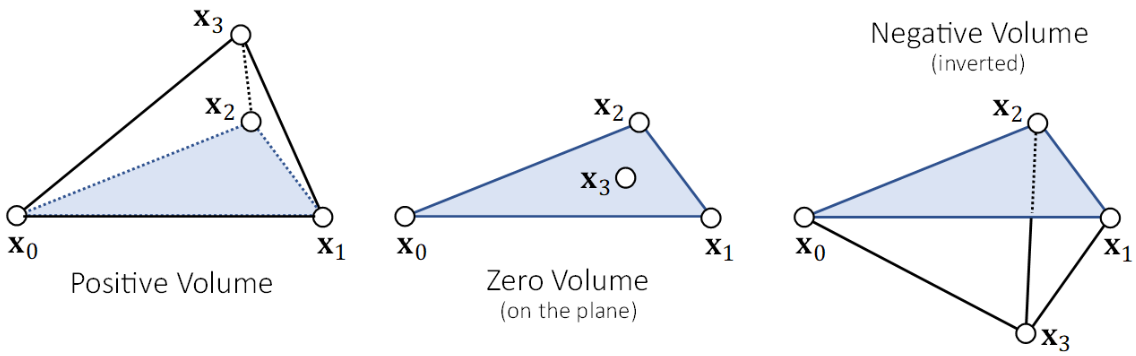


Example: Tetrahedral Volume



- Edge vectors: $\mathbf{x}_{10} = \mathbf{x}_1 - \mathbf{x}_0$ & $\mathbf{x}_{20} = \mathbf{x}_2 - \mathbf{x}_0$ & $\mathbf{x}_{20} = \mathbf{x}_2 - \mathbf{x}_0$
- Base triangle area: $A = \frac{1}{2}||\mathbf{x}_{10} - \mathbf{x}_{20}||$
- Height: $h = \mathbf{x}_{30} \cdot \mathbf{n} = \mathbf{x}_{30} \cdot \frac{\mathbf{x}_{10} - \mathbf{x}_{20}}{||\mathbf{x}_{10} - \mathbf{x}_{20}||}$
- Volume: (signed)

$$V = \frac{1}{3}hA = \frac{1}{6}\mathbf{x}_{30} \cdot \mathbf{x}_{10} \times \mathbf{x}_{20} = \frac{1}{6}\begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \mathbf{x}_3 & \mathbf{x}_0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$



Example: Barycentric Weight in Tetrahedra

$\mathbf{p}$ splits the tetrahedron into 4 tetrahedra: $V_0 = \mathrm{Vol}(\mathbf{x}_3, \mathbf{x}_2, \mathbf{x}_1, \mathbf{p})$, ...

$\mathbf{p}$ inside: if and only if $V_0, V_1, V_2, V_3 > 0$

Barycentric weights: $b_n = V_n/V,\ \mathbf{p} = b_0\mathbf{x}_0 + b_1\mathbf{x}_1 + b_2\mathbf{x}_2 + b_3\mathbf{x}_3$

Example: Particle-Triangle Intersection

- First find t when particle hits the plane: $(\mathbf{p}(t) - \mathbf{x}_0) \cdot \mathbf{x}_{10} \times \mathbf{x}_{20} = 0$, where $\mathbf{p}(t) = \mathbf{p} + t\mathbf{v}$

$$t = \frac{(\mathbf{p} - \mathbf{x}_0) \cdot \mathbf{x}_{10} \times \mathbf{x}_{20}}{\mathbf{v} \cdot \mathbf{x}_{10} \times \mathbf{x}_{20}}$$

- Check $\mathbf{p}(t)$ inside or outside

Matrices

Definitions

$$\mathbf{A} = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{bmatrix} = [\mathbf{a}_0 \quad \mathbf{a}_1 \quad \mathbf{a}_2] \in \mathbb{R}^{3\times 3}$$

- Transpose / Diagonal / Identity / Symmetric:

$$\mathbf{A}^{\mathrm{T}} = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{bmatrix}; \qquad \begin{bmatrix} a_{00} & & \\ & a_{11} & \\ & & a_{22} \end{bmatrix}; \qquad \mathbf{I} = \begin{bmatrix} 1 & & \\ & 1 & \\ & & 1 \end{bmatrix}; \qquad \mathbf{A}^{\mathrm{T}} = \mathbf{A}$$

Orthogonality

An orthogonal matrix is a matrix made of orthogonal unit vectors.

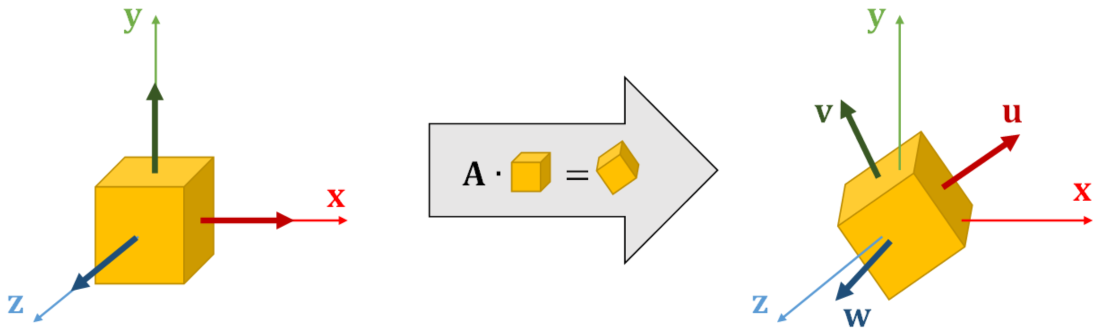
$$\mathbf{A} = [\mathbf{a}_0 \quad \mathbf{a}_1 \quad \mathbf{a}_2]\text{, such that } \mathbf{a}_i^{\mathrm{T}}\mathbf{a}_j = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{if } i \neq j \end{cases}$$

$$\mathbf{A}^{\mathrm{T}}\mathbf{A} = \begin{bmatrix} \mathbf{a}_0^{\mathrm{T}} \\ \mathbf{a}_1^{\mathrm{T}} \\ \mathbf{a}_2^{\mathrm{T}} \end{bmatrix} [\mathbf{a}_0 \quad \mathbf{a}_1 \quad \mathbf{a}_2] = \begin{bmatrix} \mathbf{a}_0^{\mathrm{T}}\mathbf{a}_0 & \mathbf{a}_0^{\mathrm{T}}\mathbf{a}_1 & \mathbf{a}_0^{\mathrm{T}}\mathbf{a}_2 \\ \mathbf{a}_1^{\mathrm{T}}\mathbf{a}_0 & \mathbf{a}_1^{\mathrm{T}}\mathbf{a}_1 & \mathbf{a}_1^{\mathrm{T}}\mathbf{a}_2 \\ \mathbf{a}_2^{\mathrm{T}}\mathbf{a}_0 & \mathbf{a}_2^{\mathrm{T}}\mathbf{a}_1 & \mathbf{a}_2^{\mathrm{T}}\mathbf{a}_2 \end{bmatrix} = \mathbf{I}; \quad \mathbf{A}^{\mathrm{T}} = \mathbf{A}^{-1}$$

Matrix Transformation

- **Rotation** can be represented by an **orthogonal** matrix

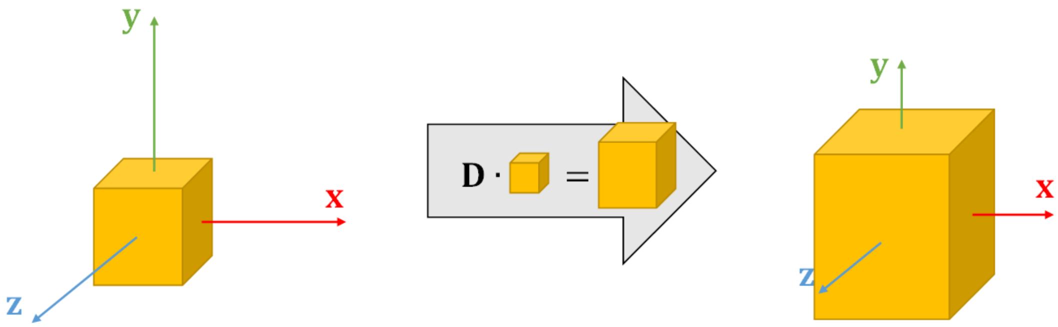
  (can represent local -> world)



(Considering of local coord. vect.)

$$\left.\begin{aligned}\mathbf{A}\mathbf{x} &= \mathbf{u}\\ \mathbf{A}\mathbf{y} &= \mathbf{v}\\ \mathbf{A}\mathbf{z} &= \mathbf{w}\end{aligned}\right\} \Rightarrow \mathbf{A} = [\mathbf{u} \quad \mathbf{v} \quad \mathbf{w}]$$

- **Scaling** can be represented by a **diagonal** matrix



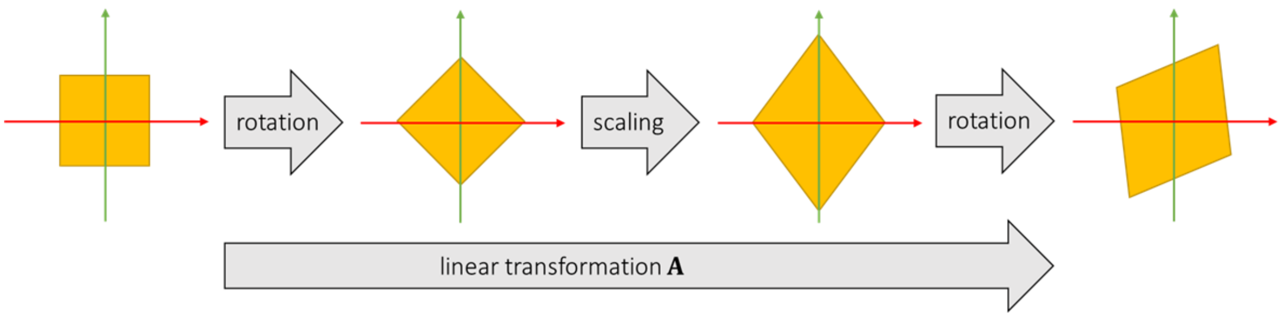
(Consisting of scaling factors)

$$\mathbf{D} = \begin{bmatrix} d_x & & \\ & d_y & \\ & & d_z \end{bmatrix}$$

- **Singular Value Decomposition**

  A matrix can be decomposed $\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^{\mathrm{T}}$ ($\mathbf{D}$ - Diagonal, $\mathbf{U}$ & $\mathbf{V}$ - Orthogonal)

  Rotation ($\mathbf{V}^{\mathrm{T}}$) -> Scaling ($\mathbf{D}$) -> Rotation (All can be decomposed as rotaion and scaling (even in 3D))



- **Eigenvalue Decomposition**

  Only consider **symmetric** matrices: $\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{U}^{-1} = \mathbf{U}\mathbf{D}\mathbf{U}^{\mathrm{T}}$

  Also can be defined: Let $\mathbf{U} = [\cdots \quad \mathbf{u}_i \quad \cdots]$, $\mathbf{u}_i$ is the eigenvector of d_i

$$\mathbf{A}\mathbf{u}_i = \mathbf{U}\mathbf{D}\mathbf{U}^{\mathrm{T}}\mathbf{u}_i = \mathbf{U}\mathbf{D}\begin{bmatrix}\vdots\\0\\1\\0\\\vdots\end{bmatrix} = \mathbf{U}\begin{bmatrix}\vdots\\0\\d_i\\0\\\vdots\end{bmatrix} = d_i\mathbf{u}_i$$

  For asymmetric matrices -> eigenvalue can be imaginary nums

- **Symmetric Positive Definiteness (s.p.d.)**

-> Linear system

For a s.p.d., if only if all its eigenvalues are positive

- $\mathbf{A}$ is s.p.d. if and only if: $\mathbf{v}^{\mathrm{T}}\mathbf{A}\mathbf{v} > 0$, for any $\mathbf{v} \neq 0$
- $\mathbf{A}$ is symmetric semi-definite if and only if: $\mathbf{v}^{\mathrm{T}}\mathbf{A}\mathbf{v} \geq 0$, for any $\mathbf{v} \neq 0$

Meaning:

- for $d > 0$ => $\mathbf{v}^{\mathrm{T}} d\mathbf{v} > 0$ for any $\mathbf{v} \neq 0$;
- for $d_0, d_1, \ldots > 0$ (for any $\mathbf{v} \neq 0$)

$$\Rightarrow \mathbf{v}^{\mathrm{T}}\mathbf{D}\mathbf{v} = \mathbf{v}^{\mathrm{T}} \begin{bmatrix} \ddots & & \\ & d_i & \\ & & \ddots \end{bmatrix} \mathbf{v} = \sum d_i v_i^2 > 0$$

- for $d_0, d_1, \ldots > 0$ with a $\mathbf{U}$ orthogonal (for any $\mathbf{v} \neq 0$)

$$\Rightarrow \mathbf{v}^{\mathrm{T}} \left(\mathbf{U}\mathbf{D}\mathbf{U}^{\mathrm{T}}\right)\mathbf{v} = \mathbf{v}^{\mathrm{T}}\mathbf{U}\mathbf{U}^{\mathrm{T}} \left(\mathbf{U}\mathbf{D}\mathbf{U}^{\mathrm{T}}\right)\mathbf{U}\mathbf{U}^{\mathrm{T}}\mathbf{v} = (\mathbf{U}^{\mathrm{T}}\mathbf{v})^{\mathrm{T}}(\mathbf{D})(\mathbf{U}^{\mathrm{T}}\mathbf{v}) > 0$$

In practice, a **diagonally domiant** matrix is p.d. ($a_{ii} > \sum_{i \neq j} |a_{ij}|$ for all i)

- **Properties**:
- $\mathbf{A}$ is s.p.d, then $\mathbf{B} = \begin{bmatrix} \mathbf{A} & -\mathbf{A} \\ -\mathbf{A} & \mathbf{A} \end{bmatrix}$ is symmetric semi-definite.

Linear Solver

$\mathbf{A}\mathbf{x} = \mathbf{b}$ ($\mathbf{A}$ - Square matrix; $\mathbf{x}$ - Unknown to be found; $\mathbf{b}$ - Boundary vector)

It's expensive to compute $\mathbf{A}^{-1}$, especially if $\mathbf{A}$ is large and sparse (Cannot use $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$)

Direct Linear Solver

LU factorization (Alt: Choleskey, LDL$^{\mathrm{T}}$, etc.)

$$\mathbf{A} = \mathbf{L}\mathbf{U} = \underbrace{\begin{bmatrix} l_{00} & & \\ l_{10} & l_{11} & \\ \vdots & \cdots & \ddots \end{bmatrix}}_{\mathbf{L}} \underbrace{\begin{bmatrix} \ddots & \cdots & \vdots \\ & u_{n-1,n-1} & u_{n-1,n} \\ & & u_{n,n} \end{bmatrix}}_{\mathbf{U}}$$

First solve: (up -> down)

$$\mathbf{L}\mathbf{y} = \mathbf{b} \Rightarrow \begin{bmatrix} l_{00} & & \\ l_{10} & l_{11} & \\ \vdots & \cdots & \ddots \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ \vdots \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \\ \vdots \end{bmatrix} \quad \Rightarrow \begin{cases} y_0 = b_0 / l_{00} \\ y_1 = (b_1 - l_{10} y_0)/l_{11} \\ \ldots \end{cases}$$

Then solve: (down -> up)

$$\mathbf{U}\mathbf{x} = \mathbf{y} \Rightarrow \begin{bmatrix} \ddots & \cdots & \vdots \\ & u_{n-1,n-1} & u_{n-1,n} \\ & & u_{n,n} \end{bmatrix} \begin{bmatrix} \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} \vdots \\ y_{n-1} \\ y_n \end{bmatrix} \quad \Rightarrow \begin{cases} x_n = y_n / u_{n,n} \\ x_{n-1} = (y_{n-1} - u_{n-1,n} x_n)/u_{n-1,n-1} \\ \ldots \end{cases}$$

Properties:

- When $\mathbf{A}$ is sparse, $\mathbf{L}$ & $\mathbf{U}$ are not that sparse, dep on the **permutation** (modify the order) -> MATLAB (LUP)
- 2 steps: factorization & solving. if want more systems with the same $\mathbf{A}$, factorization could be done once (Save costs)
- Hard to **parallelized** (Intel MKL PARDISO)

Iterative Linear Solver

$$\mathbf{x}^{[k+1]} = \mathbf{x}^{[k]} + \alpha \mathbf{M}^{-1}(\mathbf{b} - \mathbf{A}\mathbf{x}^{[k]})$$

(α - relaxation; $\mathbf{M}$ - Iterative matrix; $\mathbf{b} - \mathbf{A}\mathbf{x}^{[k]}$ - Residual error (for perfect solution -> = 0))

Converge property: ($\mathbf{b} - \mathbf{A}\mathbf{x}^{[0]} = \text{const}$ at first)

$$\begin{aligned}\mathbf{b} - \mathbf{A}\mathbf{x}^{[k+1]} &= \mathbf{b} - \mathbf{A}\mathbf{x}^{[k]} - \alpha\mathbf{A}\mathbf{M}^{-1}\left(\mathbf{b} - \mathbf{A}\mathbf{x}^{[k]}\right) \\ &= \left(\mathbf{I} - \alpha\mathbf{A}\mathbf{M}^{-1}\right)\left(\mathbf{b} - \mathbf{A}\mathbf{x}^{[k]}\right) = \left(\mathbf{I} - \alpha\mathbf{A}\mathbf{M}^{-1}\right)^{k+1}\left(\mathbf{b} - \mathbf{A}\mathbf{x}^{[0]}\right)\end{aligned}$$

So: ($\rho\left(\mathbf{I} - \alpha\mathbf{A}\mathbf{M}^{-1}\right)$ - Spectral radius (largest absolute value of eigenvalues))

$$\mathbf{b} - \mathbf{A}\mathbf{x}^{[k+1]} \to \mathbf{0}, \text{ if } \rho\left(\mathbf{I} - \alpha\mathbf{A}\mathbf{M}^{-1}\right) < 1$$

For $\mathbf{M}$, must be easy to solve, e.g. $\mathbf{M} = \operatorname{diag}(\mathbf{A})$ (**Jacobi**), or $\mathbf{M} = \operatorname{lower}(\mathbf{A})$ (**Gauss-Seidel**)

Accelerate converge methods: Chebyshev, Conjugate Gradient, ...

Properties:

- Simple; Fast for inexact sol; Parallelable
- Convergence condition (not converge for every matrix); Slow for exact solutions

Tensor Calculus

Basic Concepts

- 1st-Order Derivatives

  If $f(\mathbf{x}) \in \mathbb{R}$, then

$$\mathrm{d}f = \frac{\partial f}{\partial x}\mathrm{d}x + \frac{\partial f}{\partial y}\mathrm{d}y + \frac{\partial f}{\partial z}\mathrm{d}z = \begin{bmatrix}\frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} & \frac{\partial f}{\partial z}\end{bmatrix}\begin{bmatrix}\mathrm{d}x \\ \mathrm{d}y \\ \mathrm{d}z\end{bmatrix}; \quad \frac{\partial f}{\partial \mathbf{x}} = \begin{bmatrix}\frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} & \frac{\partial f}{\partial z}\end{bmatrix} \text{ or } \boldsymbol{\nabla} f(\mathbf{x}) = \begin{bmatrix}\frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \\ \frac{\partial f}{\partial z}\end{bmatrix}$$

  If $\mathbf{f}(\mathbf{x}) = \begin{bmatrix}f(\mathbf{x}) \\ g(\mathbf{x}) \\ h(\mathbf{x})\end{bmatrix} \in \mathbb{R}^3$, then

$$\text{Jacobian: } \mathbf{J}(\mathbf{x}) = \frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \begin{bmatrix}\frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} & \frac{\partial f}{\partial z} \\ \frac{\partial g}{\partial x} & \frac{\partial g}{\partial y} & \frac{\partial g}{\partial z} \\ \frac{\partial h}{\partial x} & \frac{\partial h}{\partial y} & \frac{\partial h}{\partial z}\end{bmatrix}; \quad \text{Divergence: } \boldsymbol{\nabla}\cdot\mathbf{f} = \frac{\partial f}{\partial x} + \frac{\partial g}{\partial y} + \frac{\partial h}{\partial z}; \quad \text{Curl: } \boldsymbol{\nabla}\times\mathbf{f} = \begin{bmatrix}\frac{\partial h}{\partial y} - \frac{\partial g}{\partial z} \\ \frac{\partial f}{\partial z} - \frac{\partial h}{\partial x} \\ \frac{\partial g}{\partial x} - \frac{\partial f}{\partial y}\end{bmatrix}$$

- 2nd-Order Derivatives

  If $f(\mathbf{x}) \in \mathbb{R}$, then (Hessian is symmetric, tangent stiffness)

$$\text{Hessian: } \mathbf{H} = \mathbf{J}(\nabla f(\mathbf{x})) = \begin{bmatrix}\frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial x \partial z} \\ \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial y^2} & \frac{\partial^2 f}{\partial y \partial z} \\ \frac{\partial^2 f}{\partial x \partial z} & \frac{\partial^2 f}{\partial y \partial z} & \frac{\partial^2 f}{\partial z^2}\end{bmatrix}; \quad \text{Laplacian: } \nabla\cdot\nabla f(\mathbf{x}) = \nabla^2 f(\mathbf{x}) = \Delta f(\mathbf{x}) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} + \frac{\partial^2 f}{\partial z^2}$$

- Taylor Expansion

  If $f(x) \in \mathbb{R}$, then

$$f(x) = f(x_0) + \frac{\partial f(x_0)}{\partial x}(x - x_0) + \frac{1}{2}\frac{\partial f^2(x_0)}{\partial x^2}(x - x_0)^2 + \cdots$$

  If (vector func) $f(\mathbf{x}) \in \mathbb{R}$, then

$$
\begin{aligned}
f(\mathbf{x}) &= f\left(\mathbf{x}_0\right)+\frac{\partial f\left(\mathbf{x}_0\right)}{\partial \mathbf{x}}\left(\mathbf{x}-\mathbf{x}_0\right)+\frac{1}{2}\left(\mathbf{x}-\mathbf{x}_0\right)^{\mathrm{T}} \frac{\partial f^2\left(\mathbf{x}_0\right)}{\partial \mathbf{x}^2}\left(\mathbf{x}-\mathbf{x}_0\right)+\cdots \\
&= f\left(\mathbf{x}_0\right)+\nabla f\left(\mathbf{x}_0\right) \cdot\left(\mathbf{x}-\mathbf{x}_0\right)+\frac{1}{2}\left(\mathbf{x}-\mathbf{x}_0\right)^{\mathrm{T}} \mathbf{H}\left(\mathbf{x}-\mathbf{x}_0\right)+\cdots
\end{aligned}
$$

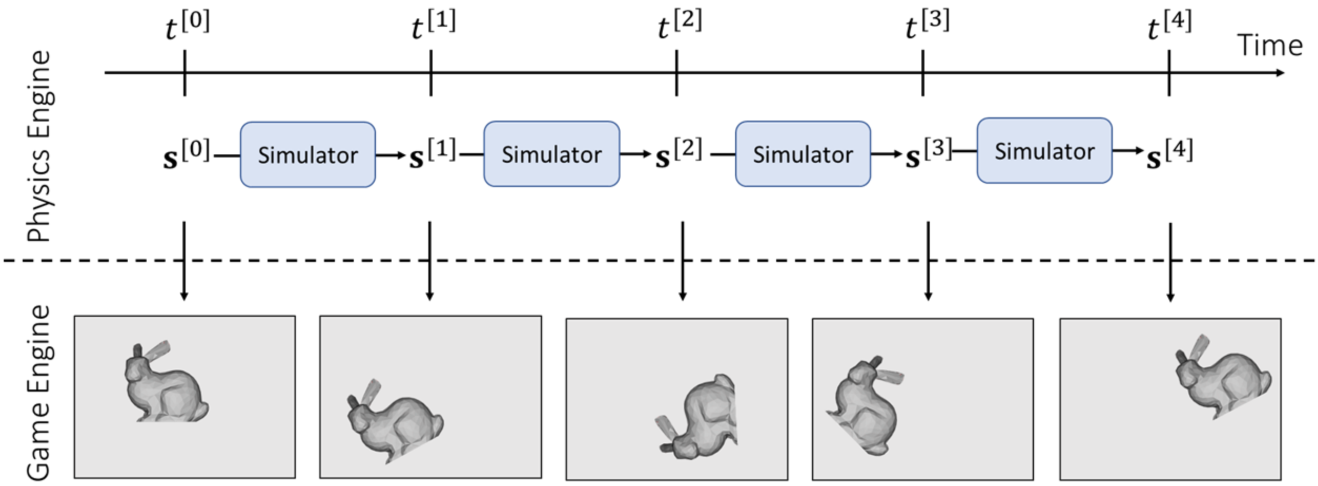
For $\mathbf{H}$ is s.p.d., second order derivative > 0 => interesting properties (to be discussed)

Lecture 3 Rigid Body Dynamics

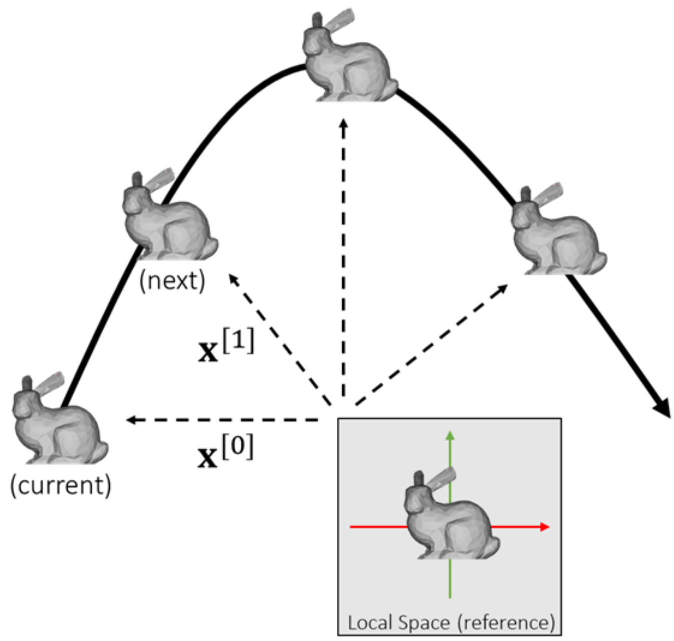
> (Single rigid body: dynamics / rotation / ...)

Rigid bodies: Assume no deformations

The goal of simulation is to update the state var. $\mathbf{s}^{[k]}$ ove



Translation Motion



For translation motion, the state variable contains the position $\mathbf{x}$ and the velocity $\mathbf{v}$ (M - Mass; Force can be the function of pos, vel, t, ...) -> Solve integral

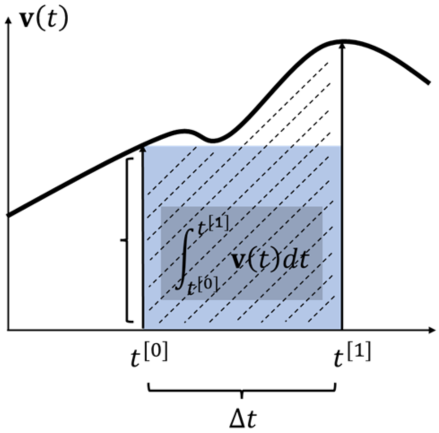
$$
\begin{cases}
\mathbf{v}\left(t^{[1]}\right)=\mathbf{v}\left(t^{[0]}\right)+M^{-1} \int_{t^{[0]}}^{t^{[1]}} \mathbf{f}(\mathbf{x}(t), \mathbf{v}(t), t)\, \mathrm{d} t \\
\mathbf{x}\left(t^{[1]}\right)=\mathbf{x}\left(t^{[0]}\right)+\int_{t^{[0]}}^{t^{[1]}} \mathbf{v}(t)\, \mathrm{d} t
\end{cases}
$$

Integration Methods Explained

By def, the integral of $\mathbf{x}(t) = \int \mathbf{v}(t)\, \mathrm{d}t$ is the area.

- **Explicit Euler** (1st-order accurate) sets the height at $t^{[0]}$

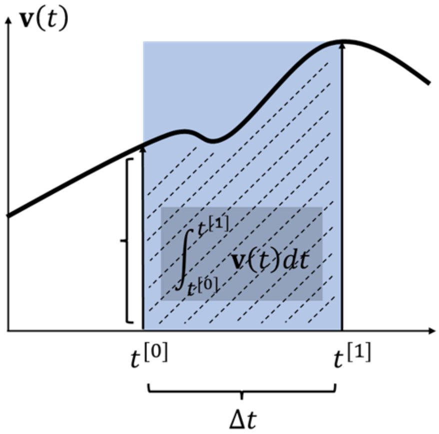
$$\int_{t^{[0]}}^{t^{[1]}} \mathbf{v}(t)\, \mathrm{d}t \approx \Delta t\, \mathbf{v}(t^{[0]})$$



  Use Taylor Expansion: $\int_{t^{[0]}}^{t^{[1]}} \mathbf{v}(t)dt = \Delta t \mathbf{v}\left(t^{[0]}\right) + \frac{\Delta t^2}{2}\mathbf{v}'\left(t^{[0]}\right) + \cdots = \Delta t \mathbf{v}\left(t^{[0]}\right) + \mathcal{O}(\Delta t^2)$ (Error: $\mathcal{O}(\Delta t^2)$)

- **Implicit Euler** (1st-order accurate): sets the height at $t^{[1]}$

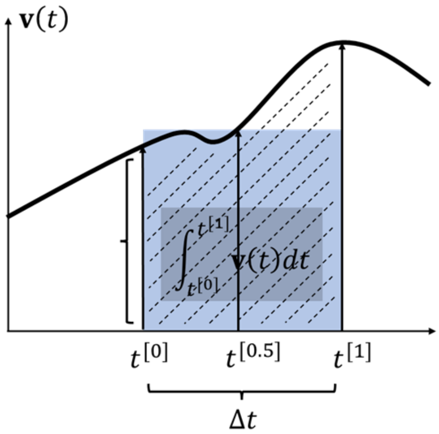
$$\int_{t^{[0]}}^{t^{[1]}} \mathbf{v}(t)\, \mathrm{d}t \approx \Delta t\, \mathbf{v}(t^{[1]})$$



  Taylor: $\int_{t^{[0]}}^{t^{[1]}} \mathbf{v}(t)dt = \Delta t \mathbf{v}\left(t^{[1]}\right) + \frac{\Delta t^2}{2}\mathbf{v}'\left(t^{[1]}\right) + \cdots = \Delta t \mathbf{v}\left(t^{[1]}\right) + \mathcal{O}(\Delta t^2)$ (Error: $\mathcal{O}(\Delta t^2)$)

- **Mid-point** (2nd-order accurate): sets at $t^{[0.5]}$

$$\int_{t^{[0]}}^{t^{[1]}} \mathbf{v}(t)\, \mathrm{d}t \approx \Delta t\, \mathbf{v}(t^{[0.5]})$$



Taylor:

$$
\begin{aligned}
\int_{t^{[0]}}^{t^{[1]}} \mathbf{v}(t) dt &= \int_{t^{[0]}}^{t^{[0.5]}} \mathbf{v}(t)\ \mathrm{d}t + \int_{t^{[0.5]}}^{t^{[1]}} \mathbf{v}(t)\ \mathrm{d}t \\
&= \frac{1}{2}\Delta t \mathbf{v}\left(t^{[0.5]}\right) - \frac{\Delta t^2}{2}\mathbf{v}'^{\left(t^{[0.5]}\right)} + O\left(\Delta t^3\right) + \frac{1}{2}\Delta t \mathbf{v}\left(t^{[0.5]}\right) + \frac{\Delta t^2}{2}\mathbf{v}'^{\left(t^{[0.5]}\right)} + O\left(\Delta t^3\right) \\
&= \Delta t\ \mathbf{v}(t^{[0.5]}) + \mathcal{O}(\Delta t^3)
\end{aligned}
$$

- Final Method in this case: **Semi-implicit** (Mid-point)

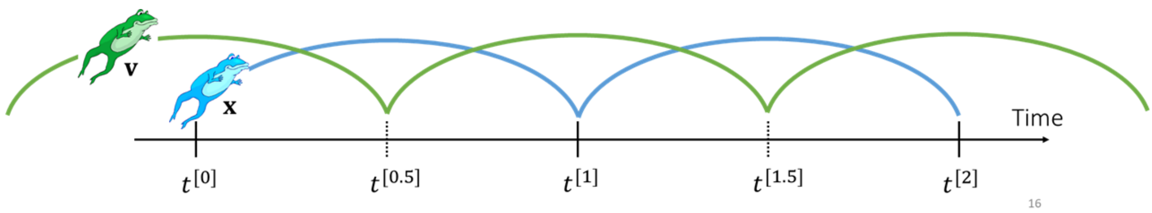
  Velocity -> Explicit; Position -> Implicit

$$
\begin{cases}
\mathbf{v}^{[1]} = \mathbf{v}^{[0]} + \Delta t M^{-1}\mathbf{f}^{[0]} \\
\mathbf{x}^{[1]} = \mathbf{x}^{[0]} + \Delta t \mathbf{v}^{[1]}
\end{cases}
$$

  Alternative: **Leapfrog Integration**

  v and x not overlap (Mid-points)

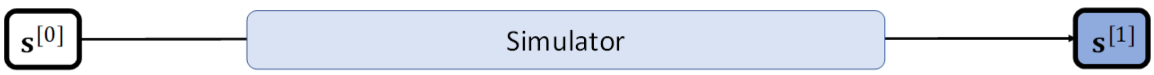
$$
\begin{cases}
\mathbf{v}^{[0.5]} = \mathbf{v}^{[-0.5]} + \Delta t M^{-1}\mathbf{f}^{[0]} \\
\mathbf{x}^{[1]} = \mathbf{x}^{[0]} + \Delta t \mathbf{v}^{[0.5]}
\end{cases}
$$



Type of Forces

- **Gravity Force**: $\mathbf{f}^{[0]}_{\text{gravity}} = M\mathbf{g}$
- Drag Force: $\mathbf{f}^{[0]}_{\text{drag}} = -\sigma \mathbf{v}^{[0]}$ (σ - drag coefficient) -> Reduced by following
- Use a coefficient to replace the drag force: $\mathbf{v}^{[1]} = \alpha \mathbf{v}^{[0]}$ (α - **decay coefficient**)

Translation Only Simulation



Steps: (Mass M and Timestep Δt are user spec var)

- $\mathbf{f}_i^{[0]} \leftarrow \text{Force}(\mathbf{x}_i^{[0]}, \mathbf{v}_i^{[0]})$
- $\mathbf{f}^{[0]} \leftarrow \sum \mathbf{f}_i^{[0]}$
- $\mathbf{v}^{[1]} \leftarrow \mathbf{v}^{[0]} + \Delta t M^{-1}\mathbf{f}^{[0]}$
- $\mathbf{x}^{[1]} \leftarrow \mathbf{x}^{[0]} + \Delta t \mathbf{v}^{[1]}$

Rotation Motion

Rotation Representations

Rotation Represented by Matrix

$$
\mathbf{R} = \begin{bmatrix} r_{00} & r_{01} & r_{02} \\ r_{10} & r_{11} & r_{12} \\ r_{20} & r_{21} & r_{22} \end{bmatrix}
$$

Suitable in graphics, rotation for vertices

Not suitable for dynamics:

- Too much redundancy: 9 elem, 3 dof
- Not intuitive
- Defining its time derivative (rotational vel) is difficult

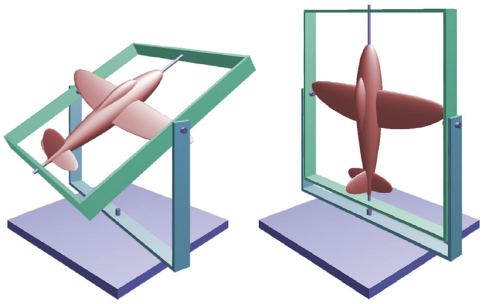
Rotation Represented by Euler Angles

Use 3 axial rotations to represent one general rotation. Each axial rotation uses an angle.

Used in Unity. (the order is rotaion-by-Z / X / Y) Intuitive.

Not suitable for dynamics:

- Lose DOFs in certain statues: Gimbal lock



- Defining its time derivative is difficult

Rotation Represented by Quaternion

Complex multiplications: In the complex system, two numbers represent a 2D point. => Quaternion: i, j, k are imaginary numbers (3D space) Four numbers represent a 3D point (with multiplication and division).

Complex multiplications

	$\mathbf{1}$	$\mathbf{i}$
$\mathbf{1}$	1	$\mathbf{i}$
$\mathbf{i}$	$\mathbf{i}$	-1

Quaternion multiplications

	$\mathbf{1}$	$\mathbf{i}$	$\mathbf{j}$	$\mathbf{k}$
$\mathbf{1}$	1	$\mathbf{i}$	$\mathbf{j}$	$\mathbf{k}$
$\mathbf{i}$	$\mathbf{i}$	-1	$\mathbf{k}$	$-\mathbf{j}$
$\mathbf{j}$	$\mathbf{j}$	$-\mathbf{k}$	-1	$\mathbf{i}$
$\mathbf{k}$	$\mathbf{k}$	$\mathbf{j}$	$-\mathbf{i}$	-1

Arithematic

Let $\mathbf{q} = [s \quad \mathbf{v}]$ be a quaternion made of 2 parts: a scalar s and a 3D vector $\mathbf{v}$ for $\mathbf{ijk}$

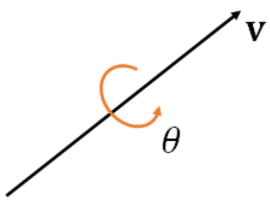
- $a\mathbf{q} = [as \quad a\mathbf{v}]$ Scalar-quaternion Multiplication
- $\mathbf{q}_1 \pm \mathbf{q}_2 = [s_1 \pm s_2 \quad \mathbf{v}_1 \pm \mathbf{v}_2]$ Addition/Subtraction (Same as vector)
- $\mathbf{q}_1 \times \mathbf{q}_2 = [s_1 s_2 - \mathbf{v}_1 \cdot \mathbf{v}_2 \quad s_1\mathbf{v}_2 + s_2\mathbf{v}_1 + \mathbf{v}_1 \times \mathbf{v}_2]$ Multiplication
- $||\mathbf{q}|| = \sqrt{s^2 + \mathbf{v} \cdot \mathbf{v}}$ Magnitude

In Unity: provide multiplication, but no addition/subtraction/...; Use w, x, y, z -> s, v

Representation

Rotate around $\mathbf{v}$ by angle θ

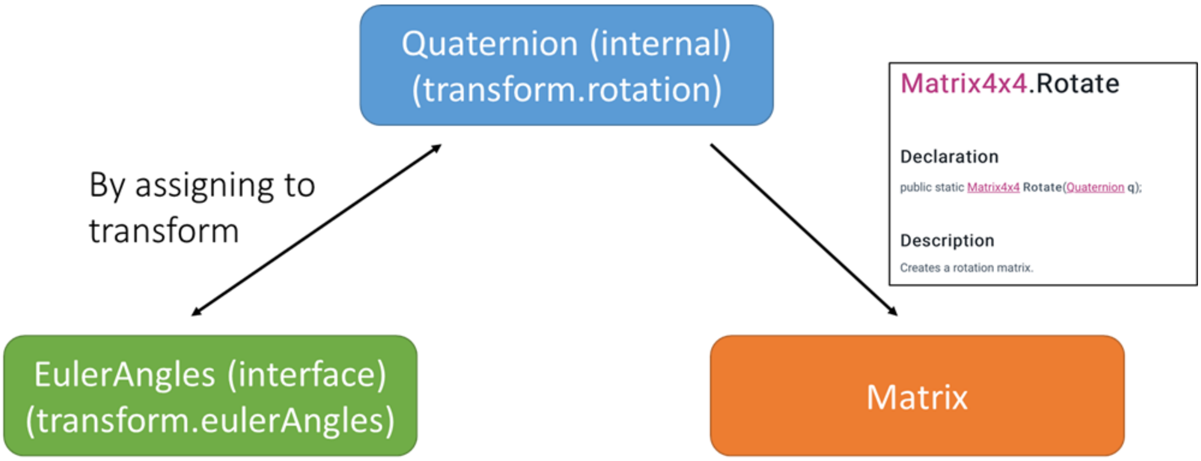
$$\begin{cases} \mathbf{q} = [\cos\frac{\theta}{2} \quad \mathbf{v}] \\ \|\mathbf{q}\| = 1 \end{cases} \Rightarrow \begin{cases} \mathbf{q} = [\cos\frac{\theta}{2} \quad \mathbf{v}] \\ \|\mathbf{v}\|^2 = \sin^2\frac{\theta}{2} \end{cases}$$



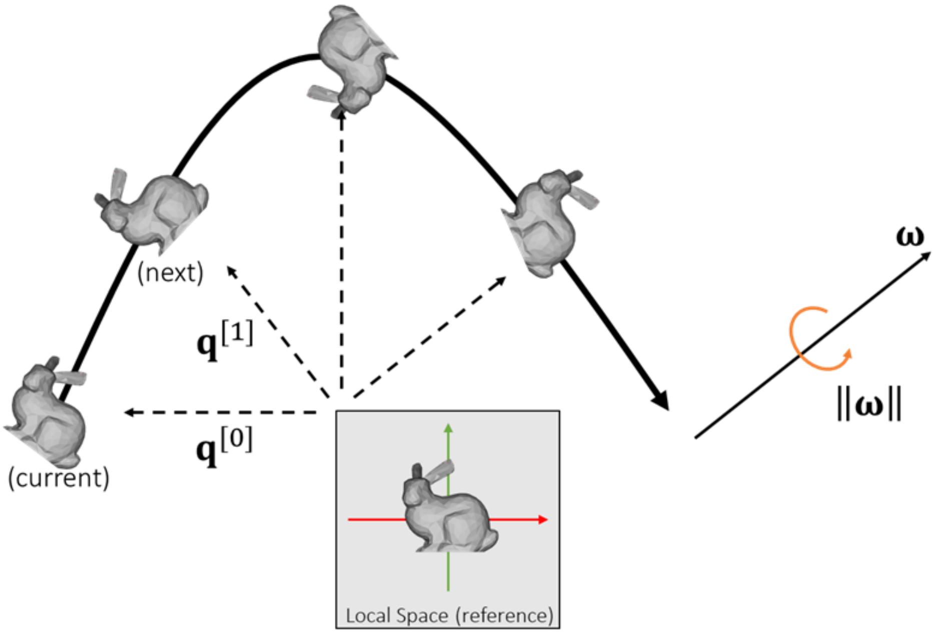
Convertible to the matrix:

$$\mathbf{R} = \begin{bmatrix} s^2+x^2-y^2-z^2 & 2(xy-sz) & 2(xz+sy) \\ 2(xy+sz) & s^2-x^2+y^2-z^2 & 2(yz-sx) \\ 2(xz-sy) & 2(yz+sx) & s^2-x^2-y^2+z^2 \end{bmatrix}$$

Rotation Representations in Unity



Rotation Motion



Use a 3D vector $\boldsymbol{\omega}$ to denote **angular velocity**:

The dir of $\boldsymbol{\omega}$ -> the axis; The magnitude of $\boldsymbol{\omega}$ -> the speed (sim to the representation of quaternion)

Torque and Inertia

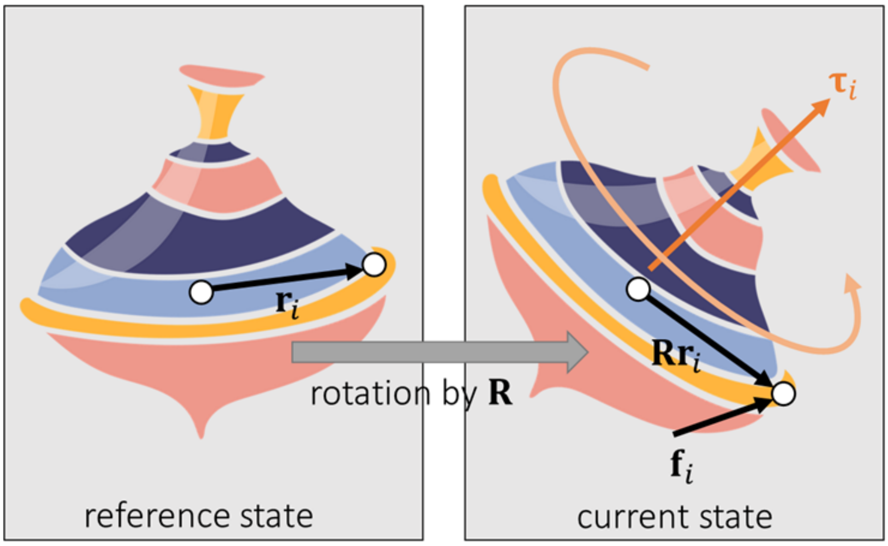
Torque

(Original state: $\mathbf{r}_i$ -> Rotated: $\mathbf{R}\mathbf{r}_i$, $\mathbf{f}_i$ is a force)

The rotational equiv of a force, describing the rotational **tendency** caused by a force.

$\boldsymbol{\tau}_i$: perpendicular to both $\mathbf{R}\mathbf{r}_i$ and $\mathbf{f}_i$; proportional to $||\mathbf{R}\mathbf{r}_i||$ and $||\mathbf{f}_i||$; proportional to $\sin\theta$ (the angle of the two vectors)

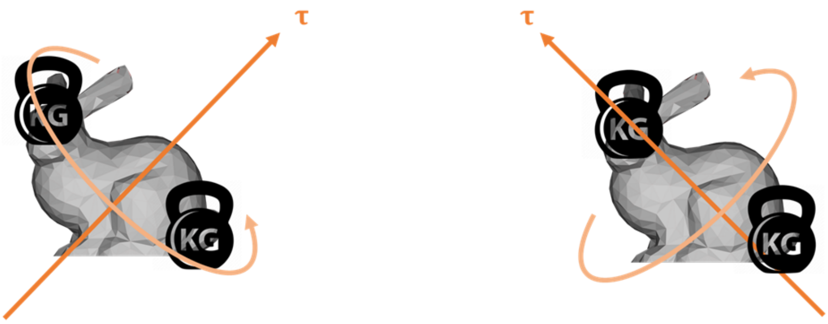
$$\boldsymbol{\tau}_i \leftarrow \mathbf{R}\mathbf{r}_i \times \mathbf{f}_i$$



Inertia

Describes the **resistance** to rotational tendency caused by torque (not const)

Left side (heavier point far away from the torque) has higher resistance (inertia) to the rotational tendency, slower rotation



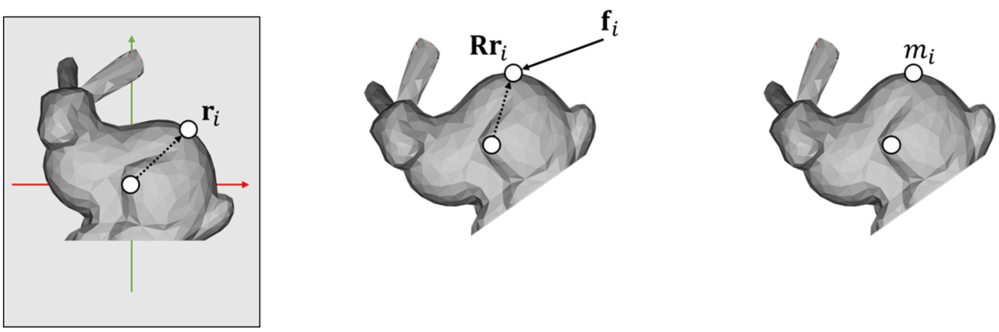
Ref state inertia, change with rotation (dep on pose). But no need to re-compute every time

$$\mathbf{I}_{\mathbf{ref}} = \sum m_i \left(\mathbf{r}_i^{\mathrm{T}}\mathbf{r}_i\mathbf{1} - \mathbf{r}_i\mathbf{r}_i^{\mathrm{T}}\right)$$

($\mathbf{1}$ - 3x3 identity matrix)

$$\begin{aligned}\mathbf{I} &= \sum m_i \left(\mathbf{r}_i^{\mathrm{T}}\mathbf{R}^{\mathrm{T}}\mathbf{R}\mathbf{r}_i\mathbf{1} - \mathbf{R}\mathbf{r}_i\boldsymbol{r}_i^{\mathrm{T}}\mathbf{R}^{\mathrm{T}}\right)\\ &= \sum m_i \left(\mathbf{R}\mathbf{r}_i^{\mathrm{T}}\mathbf{r}_i\mathbf{1}\mathbf{R}^{\mathrm{T}} - \mathbf{R}\mathbf{r}_i\boldsymbol{r}_i^{\mathrm{T}}\mathbf{R}^{\mathrm{T}}\right)\\ &= \sum m_i\mathbf{R} \left(\mathbf{r}_i^{\mathrm{T}}\mathbf{r}_i\mathbf{1} - \mathbf{r}_i\boldsymbol{r}_i^{\mathrm{T}}\right)\mathbf{R}^{\mathrm{T}}\\ &= \mathbf{R}\mathbf{I}_{\mathbf{ref}}\mathbf{R}^{\mathrm{T}}\end{aligned}$$

Use torque to represent



- **Torque** on a spec point: $\boldsymbol{\tau}_i = (\mathbf{R}\mathbf{r}_i) \times \mathbf{f}_i$

  Total torque: $\boldsymbol{\tau}_i = \sum \boldsymbol{\tau}_i$
- The rotational equivalent of **mass** is called **inertia** $\mathbf{I}$ (Result: 3x3):

  Reference inertia: $\mathbf{I}_{\text{ref}} = \sum m_i(\mathbf{r}_i^{\mathrm{T}}\mathbf{r}_i\mathbf{1} - \mathbf{r}_i\mathbf{r}_i^{\mathrm{T}})$

Current inertia: $\mathbf{I} = \mathbf{R}\mathbf{I}_{\text{ref}}\mathbf{R}^{\text{T}}$

Rigid Body Simulation

Translational and Rotational Motion

- Translation (Linear)

  States: velocity $\mathbf{v}$ and position $\mathbf{x}$ (`transform.position` in Unity)

  Physical Quantities: mass M and force $\mathbf{f}$

$$\begin{cases} \mathbf{v}^{[1]} = \mathbf{v}^{[0]} + \Delta t M^{-1}\mathbf{f}^{[0]} \\ \mathbf{x}^{[1]} = \mathbf{x}^{[0]} + \Delta t \mathbf{v}^{[1]} \end{cases}$$

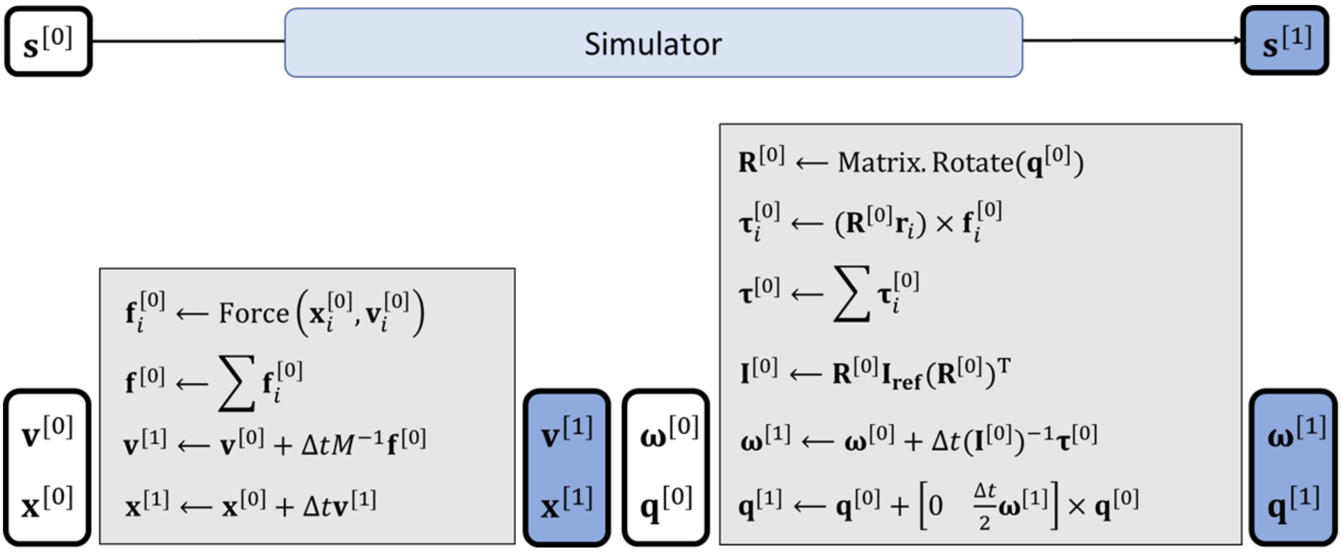
- Rotational (Angular): Better normalize when $||\mathbf{q}|| \neq 1$ (in Unity automatically)

  States: angular velocity $\boldsymbol{\omega}$ and quaternion $\mathbf{q}$ (`transform.rotation` in Unity)

  Physical Quantites: inertia $\mathbf{I}$ and torque $\boldsymbol{\tau}$

$$\begin{cases} \boldsymbol{\omega}^{[1]} = \boldsymbol{\omega}^{[0]} + \Delta t \left(\mathbf{I}^{[0]}\right)^{-1} \tau^{[0]} \\ \mathbf{q}^{[1]} = \mathbf{q}^{[0]} + \begin{bmatrix} 0 & \frac{\Delta t}{2}\boldsymbol{\omega}^{[1]} \end{bmatrix} \times \mathbf{q}^{[0]} \end{cases}$$

Rigid Body Simulation Process



In Unity: No 3x3 matrices, only 4x4 (use 4x4 and set the last col / line); Provide `.inverse` to inverse; ...

Implementation

In practice, we update the same state var $\mathbf{s} = \{\mathbf{v}, \mathbf{x}, \boldsymbol{\omega}, \mathbf{q}\}$

Issues

- Translation is easier, code translation first
- Using a const $\boldsymbol{\omega}$ first while testing update $\mathbf{q}$, in this case the object will spin constantly
- Gravity does NOT cause torque (except for air drag force)

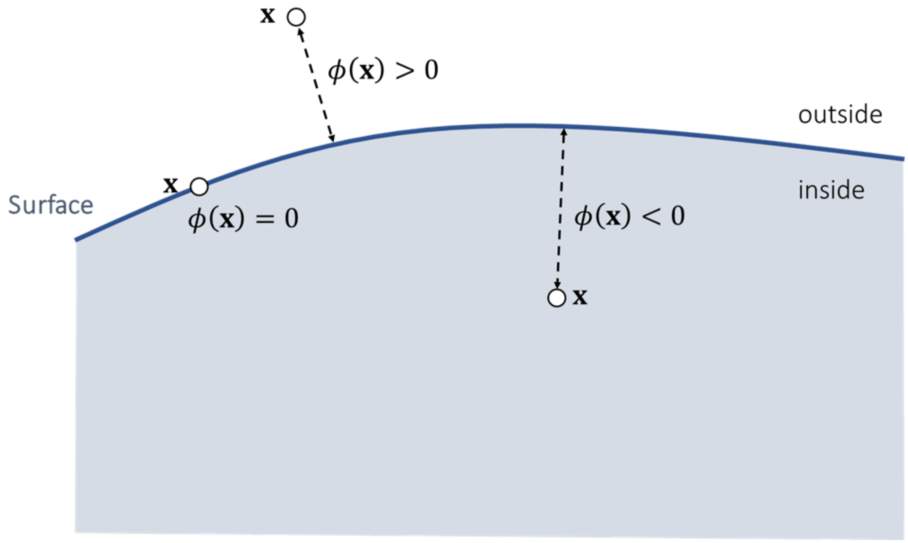
Lecture 4 Rigid Body Contacts (Lab 1)

Particle Collision Detection and Response

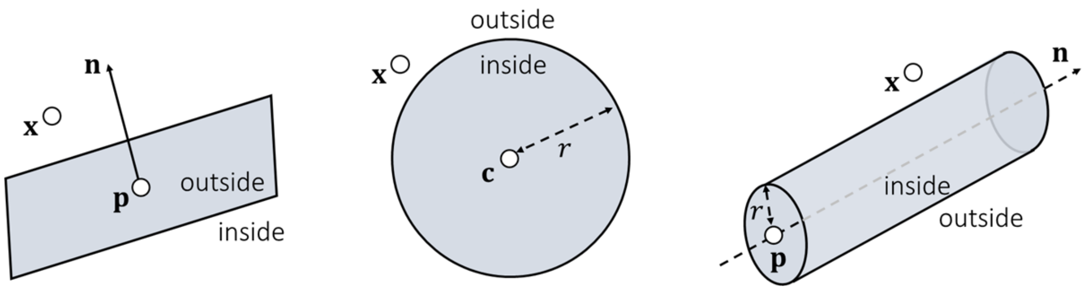
Distance Functions

Signed Distance Function

Use a signed distance func $\phi(\mathbf{x})$ to define the distance indicating which side as well (corresponding to 0 surface)



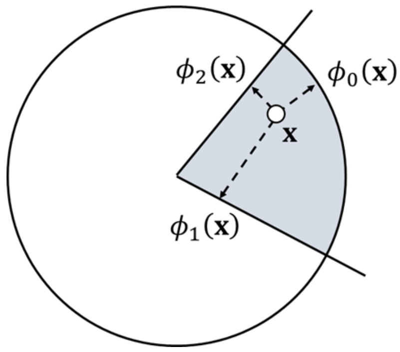
Examples



$\phi(\mathbf{x}) = (\mathbf{x} - \mathbf{p}) \cdot \mathbf{n}$ $\qquad$ $\phi(\mathbf{x}) = ||\mathbf{x} - \mathbf{c}|| - r$ $\qquad$ $\phi(\mathbf{x}) = \sqrt{||\mathbf{x} - \mathbf{p}||^2 - ((\mathbf{x} - \mathbf{p}) \cdot \mathbf{n})^2} - \mathbf{r}$

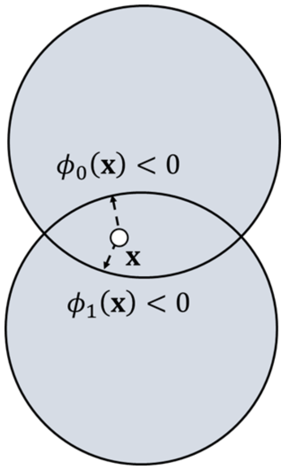
Intersection of Signed Distance Functions

=> Bool operations



if $\phi_0(\mathbf{x}) < 0$ and $\phi_1(\mathbf{x}) < 0$ and $\phi_2(\mathbf{x}) < 0$
  then inside
  $\phi(\mathbf{x}) = \max(\phi_0\mathbf{x}, \phi_1(\mathbf{x}), \phi_2(\mathbf{x}))$ // all val are negative, the max one is the closest
else
  $\phi(\mathbf{x}) = ?$ // not relavent (no collision)

Union of Signed Distance Functions



if $\phi_0(\mathbf{x}) < 0$ or $\phi_1(\mathbf{x}) < 0$
  then inside
  $\phi(\mathbf{x}) \approx \min(\phi_0(\mathbf{x}), \phi_1(\mathbf{x}))$ // approximate -> correct near outer boundary
else outside
  $\phi(\mathbf{x}) = \min(\phi_0(\mathbf{x}), \phi_1(\mathbf{x}))$

-> We can consider collision detection with the union of two objects as collision detection with two separate objects

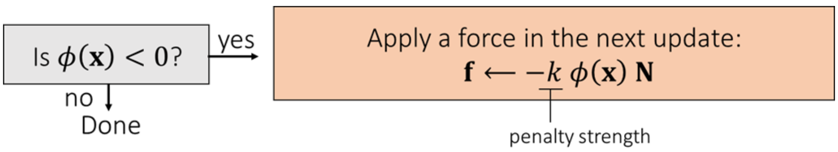
Penalty methods

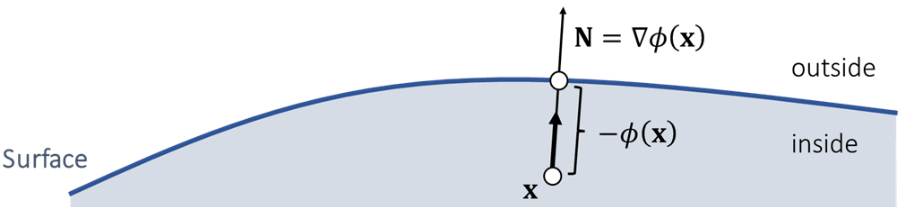
(Implicit integration is better)

Quadratic Penalty Method

Check if collide - Yes -> Apply a force at the point (in the next update)

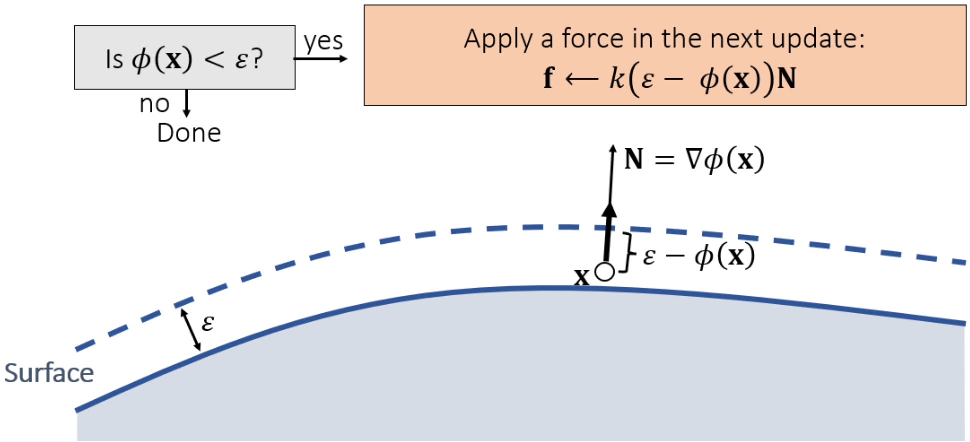
For **quadratic** penalty (strength) potential, the force is **linear**





Problem: Already inside -> cause artifacts

=> Add **buffer** help less the penetration issue (cannot strictly prevent)

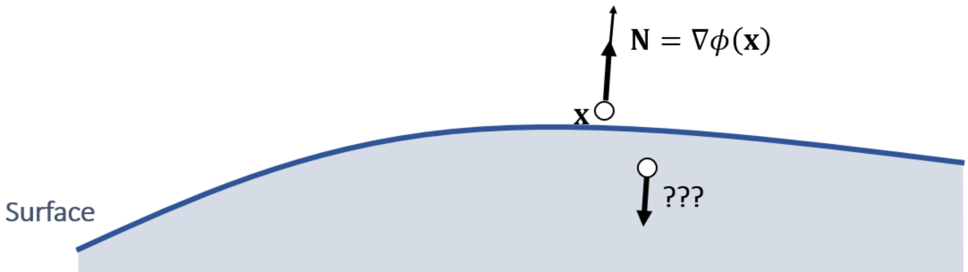


Problem: k too low -> buffer not work well; k too high -> too much force generated (overshooting)

Log-Barrier Penalty Method

Ensures that the force can be large enough, but assumes $\phi(\mathbf{x}) < 0$ never happens => By adjusting Δt

Always apply the penalty force: $\mathbf{f} \leftarrow \rho \frac{1}{\phi(\mathbf{x})} \mathbf{N}$ (ρ - Barrier strength)



Problems: cannot prevent overshooting when very close to the surface; when penatration occurs, the penatration will be higher and higher (-> smaller step size -> higher costs)

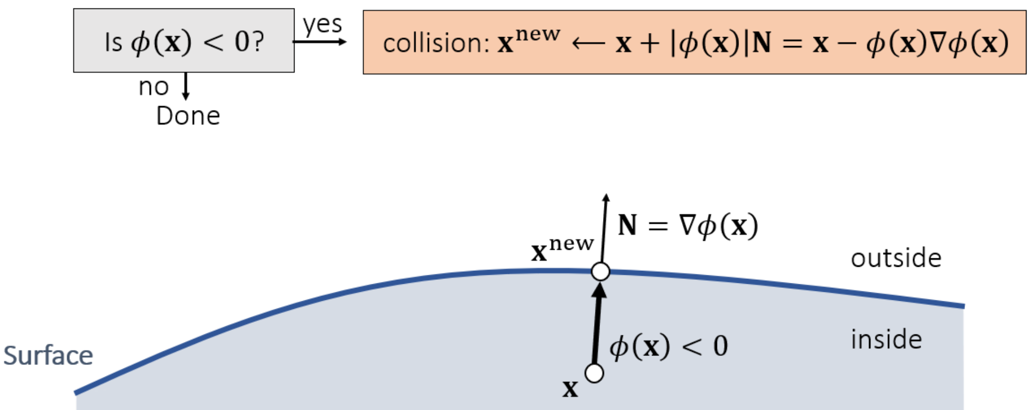
=> Log-Barrier limited within a buffer as well to solve

Frictional contacts are difficult to handle

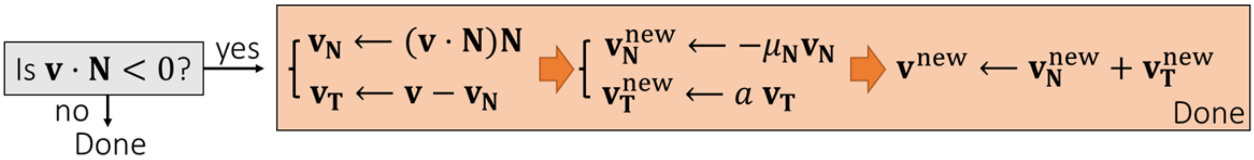
Impulse method

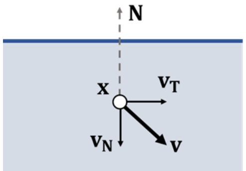
Update the vel and pos as the collision occurs

Changing the **position**:



Changing the **velocity**: ($\mu_{\mathbf{N}}$ - bounce coefficient, $\in [0, 1]$, a - frictional decay of vel)





a should be minimized but not violating Coulomb's law

$$\left\|\mathbf{v}_\mathbf{T}^{\text{new}} - \mathbf{v_T}\right\| \le \mu_\mathbf{T}\left\|\mathbf{v}_\mathbf{N}^{\text{new}} - \mathbf{v_N}\right\|$$
$$(1-a)\left\|\mathbf{v_T}\right\| \le \mu_\mathbf{T}\,(1+\mu_\mathbf{N})\left\|\mathbf{v_N}\right\|$$

Therefore:

$$a \longleftarrow \max(1 - \underbrace{\mu_\mathbf{T}\,(1+\mu_\mathbf{N})\left\|\mathbf{v_N}\right\|/\left\|\mathbf{v_T}\right\|}_{\text{dynamic friction}},\ \underbrace{0}_{\text{static friction}}\ )$$

Can precisely control the friction effects

Rigid Collision Detection and Response by Impulse

Rigid Body Collision Detection

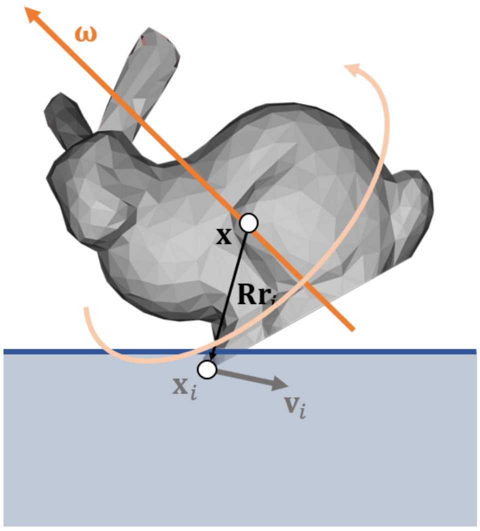
When the body is made of many vertices, test each vertex: $\mathbf{x}_i \leftarrow \mathbf{x} + \mathbf{R}\mathbf{r}_i$ (from the mass center to the vertices)

=> detection: transverse every point if $\phi(\mathbf{x}) < 0$

Rigid Body Collision Response by Impulse

Vertex i: ($\mathbf{v}$ - linear vel; $\boldsymbol{\omega}$ - angular vel)

$$\begin{cases}\mathbf{x}_i \leftarrow \mathbf{x} + \mathbf{R}\mathbf{r}_i & \text{(Position)}\\ \mathbf{v}_i \leftarrow \mathbf{v} + \boldsymbol{\omega}\times\mathbf{R}\mathbf{r}_i & \text{(Velocity)}\end{cases}$$



But cannot modify $\mathbf{x}_i$ and $\mathbf{v}_i$ directly since they are not state var. (in Lec.3: 4 var are vel of mass center / pos of mass center / rotational pose / angular vel)

Applying an impulse j at vertex i: ($\Delta\mathbf{v} = \frac{\mathbf{j}}{M}$ (Newton's Law); $\mathbf{R}\mathbf{r}_i \times \mathbf{j}$ - torque induced by **j**)

$$\begin{cases}\mathbf{v}^{\text{new}} \leftarrow \mathbf{v} + \dfrac{1}{M}\mathbf{j}\\ \boldsymbol{\omega}^{\text{new}} \leftarrow \boldsymbol{\omega} + \mathbf{I}^{-1}(\mathbf{R}\mathbf{r}_i \times \mathbf{j})\end{cases}$$

$$
\begin{aligned}
\Rightarrow \mathbf{v}_i^{\text{new}} &= \mathbf{v}^{\text{new}} + \boldsymbol{\omega}^{\text{new}} \times \mathbf{R}\mathbf{r}_i \\
&= \mathbf{v} + \frac{1}{M}\mathbf{j} + \left(\boldsymbol{\omega} + \mathbf{I}^{-1}\left(\mathbf{R}\mathbf{r}_i \times \mathbf{j}\right)\right) \times \mathbf{R}\mathbf{r}_i \\
&= \mathbf{v}_i + \frac{1}{M}\mathbf{j} - \left(\mathbf{R}\mathbf{r}_i\right) \times \left(\mathbf{I}^{-1}\left(\mathbf{R}\mathbf{r}_i \times \mathbf{j}\right)\right)
\end{aligned}
$$

Cross Product as a Matrix Product

Convert the cross prod $\mathbf{r}\times$ into a matrix prod $\mathbf{r}^*$

$$
\mathbf{r} \times \mathbf{q} = \begin{bmatrix} r_y q_z - r_z q_y \\ r_z q_x - r_x q_z \\ r_x q_y - r_y q_x \end{bmatrix} = \begin{bmatrix} 0 & -r_z & r_y \\ r_z & 0 & -r_x \\ -r_y & r_x & 0 \end{bmatrix} \begin{bmatrix} q_x \\ q_y \\ q_z \end{bmatrix} = \mathbf{r}^* \mathbf{q}
$$

In our case:

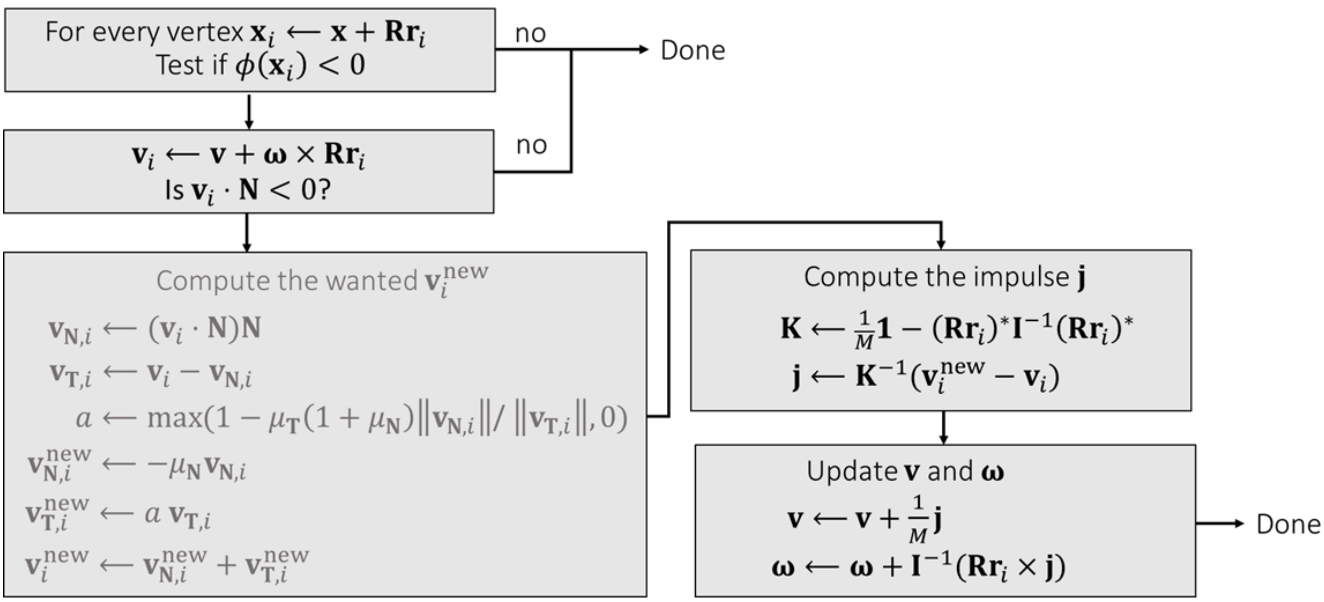
$$
\Rightarrow \mathbf{v}_i^{\text{new}} = \mathbf{v}_i^{\text{new}} = \mathbf{v}_i + \frac{1}{M}\mathbf{j} - (\mathbf{R}\mathbf{r}_i)^* \mathbf{I}^{-1} (\mathbf{R}\mathbf{r}_i)^* \mathbf{j}
$$

Therefore, replace with some matrix $\mathbf{K}$. Finally $\mathbf{j}$ can be computed with the following equations.

$$
\Rightarrow \mathbf{v}_i^{\text{new}} - \mathbf{v}_i = \mathbf{K}\mathbf{j}
$$

$$
\mathbf{K} \leftarrow \frac{1}{M}\mathbf{1} - (\mathbf{R}\mathbf{r}_i)^* \mathbf{I}^{-1} (\mathbf{R}\mathbf{r}_i)^*
$$

Implementation



Other details:

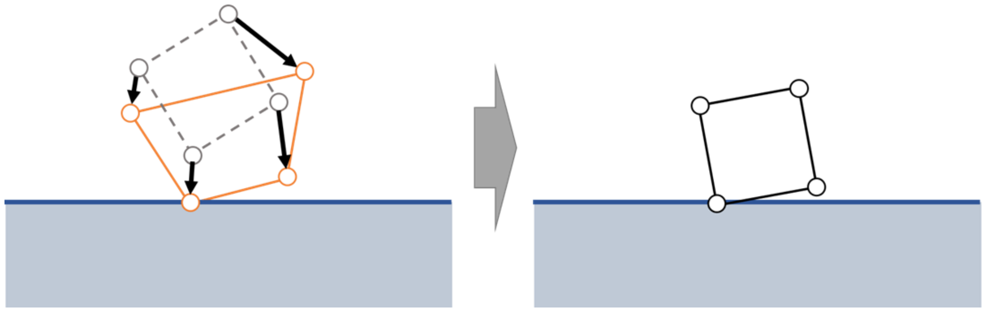
- For several vertices in collision, use their **average**
- Can **decrease the restitution** $\mu_{\mathbf{N}}$ to reduce oscillation
- Don't update the position: not linear

Shape Matching

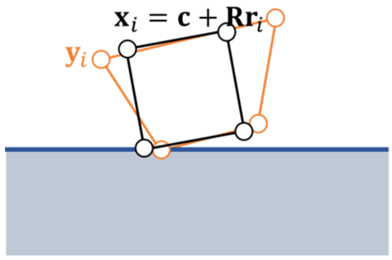
Basic Idea

Allow each vertex to have its own velocity, so it can move by itself

- Move vertices independently by its velocity, with collision and friction being handled (use the impulse method for every point)
- Enforce the rigidity constraint to become a rigid body again (IMPORTANT)



Mathematical Formulation



Want to find $\mathbf{c}$ and $\mathbf{R}$ ($\mathbf{c}$ - mass center): want the final rigid body (a square) close enough to the trapozoid

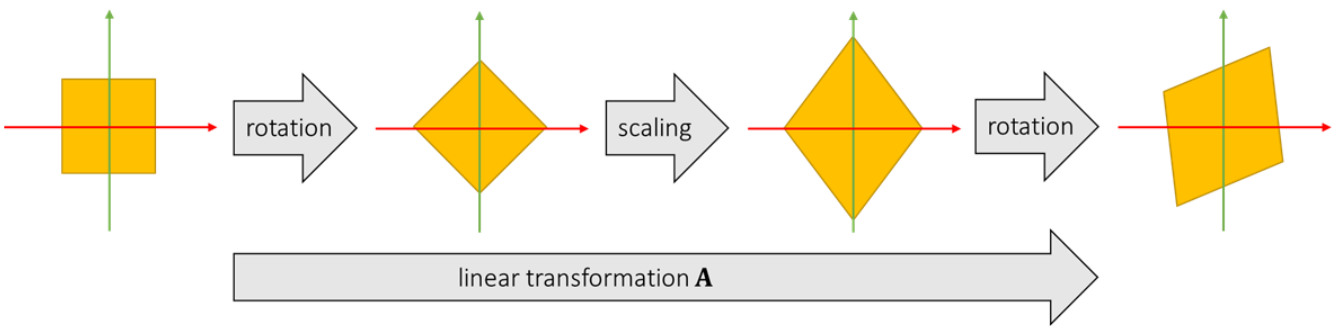
($\mathbf{A}$ - a matrix, not only corresponding to rotation, $\sum \mathbf{A}\mathbf{r}_i = 0$ since the mass center was set to 0; E - The objective, $= \frac{1}{2}\|\mathbf{c} + \mathbf{A}\mathbf{r}_i - \mathbf{y}_i\|^2$)

$$\{\mathbf{c}, \mathbf{R}\} = \operatorname{argmin} \sum_i \frac{1}{2}\|\mathbf{c} + \mathbf{R}\mathbf{r}_i - \mathbf{y}_i\|^2$$
$$\Rightarrow \{\mathbf{c}, \mathbf{A}\} = \operatorname{argmin} \sum_i \frac{1}{2}\|\mathbf{c} + \mathbf{A}\mathbf{r}_i - \mathbf{y}_i\|^2$$

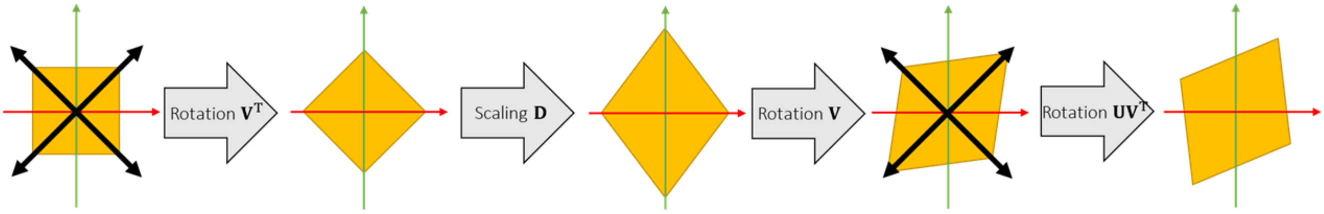
For mass center $\mathbf{c}$ and matrix $\mathbf{A}$ (Find derivatives):

$$\frac{\partial E}{\partial \mathbf{c}} = \sum_i \mathbf{c} + \cancel{\mathbf{A}\mathbf{r}_i} - \mathbf{y}_i = \sum_i \mathbf{c} - \mathbf{y}_i = \mathbf{0}$$
$$\Rightarrow \mathbf{c} = \frac{1}{N}\sum_i \mathbf{y}_i \quad (\text{average})$$
$$\frac{\partial E}{\partial \mathbf{A}} = \sum_i (\mathbf{c} + \mathbf{A}\mathbf{r}_i - \mathbf{y}_i)\mathbf{r}_i^{\mathrm{T}} = \mathbf{0}$$
$$\mathbf{A} = \left(\sum_i (\mathbf{y}_i - \mathbf{c})\mathbf{r}_i^{\mathrm{T}}\right)\left(\sum_i \mathbf{r}_i\mathbf{r}_i^{\mathrm{T}}\right)^{-1} \overset{\text{Polar Decomposition}}{=\!=\!=\!=} \underbrace{\mathbf{R}}_{\text{rotation}}\ \underbrace{\mathbf{S}}_{\text{deformation}}$$

Remind: Singular value decomposition: $\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^{\mathrm{T}}$ (rotaion, scaling and rotation)

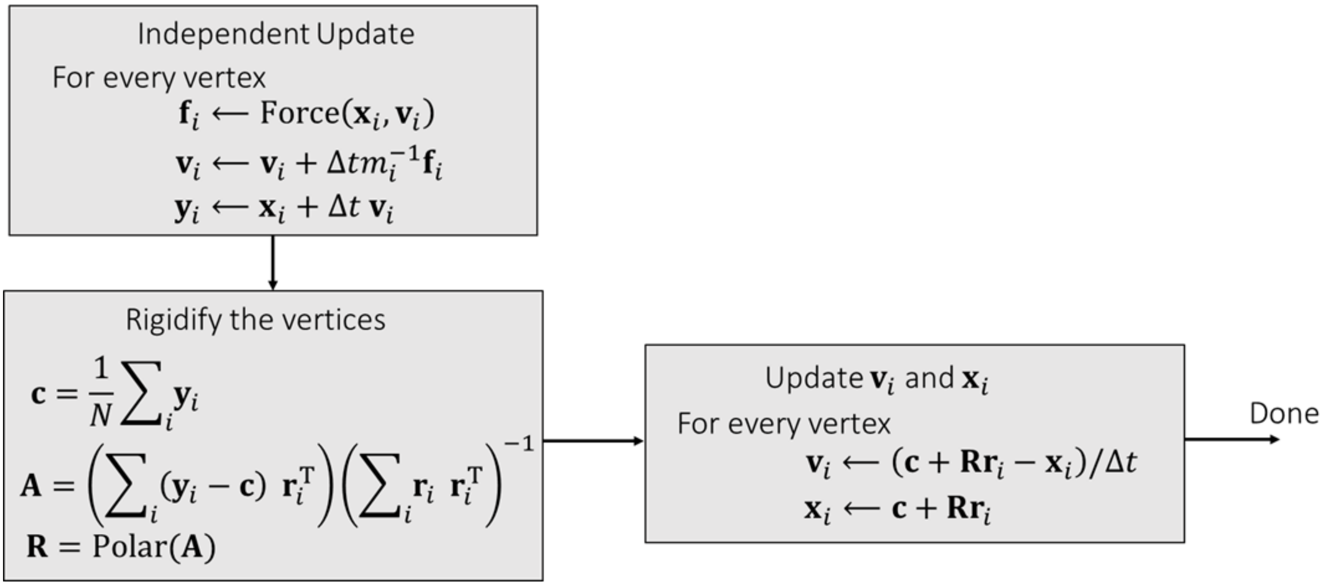


Rotate the object back before the final rotation: $\mathbf{A} = (\mathbf{U}\mathbf{V}^{\mathrm{T}})(\mathbf{V}\mathbf{D}\mathbf{V}^{\mathrm{T}}) = \mathbf{R}\mathbf{S}$ (Local deformation: $\mathbf{V}\mathbf{D}\mathbf{V}^{\mathrm{T}} = \mathbf{S}$)



Implementation

Physical quantities are attached to each vertex, not to the entire body.



(The function of $\mathrm{Polar}(\mathbf{A})$ is provided, ultilizing the polar deposition technique)

Properties:

- Easy to **implement** and **compatible with other nodal systems**: cloth, soft bodies, particle fluids, ...
- Difficult to **strictly enforce friction and other goals**. The rigidification process will destroy them (may require iter)
- More suitable when the **friction accuracy is unimportant**, i.e., buttons on clothes

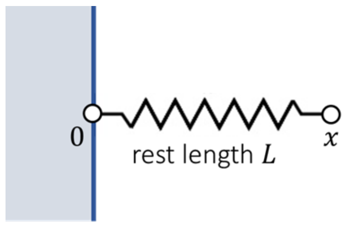
Lecture 5 Physics-Based Cloth Simulation

A Mass-Spring System

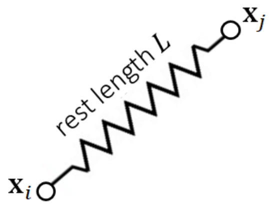
Spring Systems

An Ideal Spring

Satisfies Hooke's Law: The spring force tries to restore the rest length (k - Spring Stiffness)



$$E(x) = \frac{1}{2}k(x-L)^2\ ; \quad f(x) = -\frac{\mathrm{d}E}{\mathrm{d}x} = -k(x-L)$$

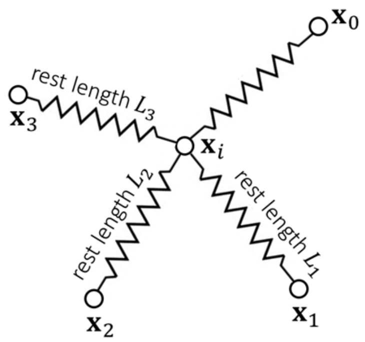


$$E(\mathbf{x}) = \frac{1}{2}k(\|\mathbf{x}_i - \mathbf{x}_j\| - L)$$

$$\mathbf{f}_i = -\mathbf{f}_j \qquad \begin{cases} \mathbf{f}_i(\mathbf{x}) = -\boldsymbol{\nabla}_i E = -k\left(\|\mathbf{x}_i - \mathbf{x}_j\| - L\right)\dfrac{\mathbf{x}_i - \mathbf{x}_j}{\|\mathbf{x}_i - \mathbf{x}_j\|} \\ \mathbf{f}_j(\mathbf{x}) = -\boldsymbol{\nabla}_j E = -k\left(\|\mathbf{x}_j - \mathbf{x}_i\| - L\right)\dfrac{\mathbf{x}_j - \mathbf{x}_i}{\|\mathbf{x}_j - \mathbf{i}_j\|} \end{cases}$$

Multiple Springs

The energies and forces can be simply summed up

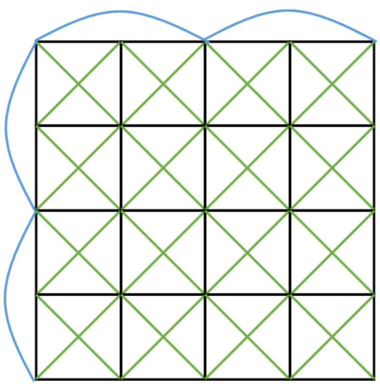


$$E = \sum_{j=0}^{3} E_j = \sum_{j=0}^{3}\left(\frac{1}{2}k(\|\mathbf{x}_i - \mathbf{x}_j\| - L)^2\right)$$

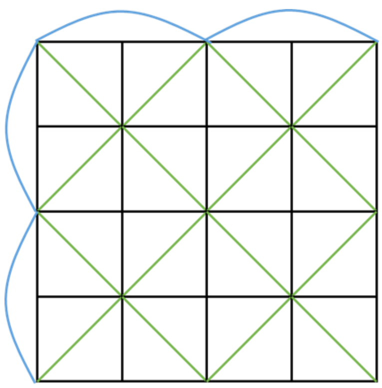
$$\mathbf{f}_i = -\nabla_i E = \sum_{j=0}^{3}\left(-k\left(\|\mathbf{x}_i - \mathbf{x}_j\| - L\right)\frac{\mathbf{x}_i - \mathbf{x}_j}{\|\mathbf{x}_i - \mathbf{x}_j\|}\right)$$

Structures in Simulations

Structured Spring Networks



A structured network

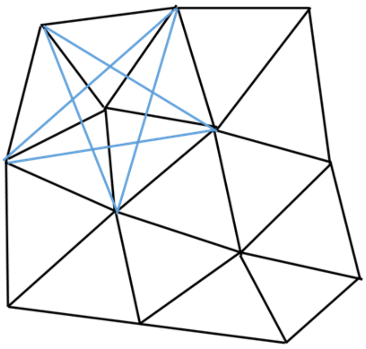


A simplified network

Unstructured Spring Networks

Unstructured triangle mesh

-> the edges into spring networks (usually in cloth simulations)

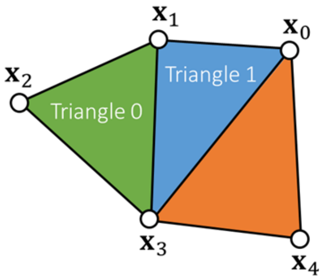


Blue lines for bending resistance (every neighboring triangle pair)

Triangle Mesh Representation

Two arrays: **Vertex** & **Triangle lines**

- Vertex list: $\{\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4\}$ (3D vectors)
- Triangle list: `{1, 2, 3, 0, 1, 3, 0, 3, 4}` (Index triples) => (`{1, 2, 3}` for Triangle 0; `{0, 1, 3}` for Triangle 1; ...)



Not only edges but also an inner one (each triangle has 3 edges but there are repeated ones)

Topological Construction

Sort triangle edge triples: edge vertex index 0 & 1 and triangle index (index 0 < index)

- Triple list:

```
{{1, 2, 0}, {2, 3, 0}, {1, 3, 0},   // Green triangle: {1, 2} for edge index; {0} at the end for triangle index
 {0, 1, 1}, {1, 3, 1}, {0, 3, 1},
 {0, 3, 2}, {3, 4, 2}, {0, 4, 2}}
```

- Sorted triple list: `{{0, 1, 1}, {0, 3, 1}, {0, 3, 2}, {0, 4, 2}, {1, 2, 0}, {1, 3, 0}, {1, 3, 1}, {2, 3, 0},  {3, 4, 2}}`

  Repeated edges `{1, 3, 0}|{1, 3, 1}` & `{0, 3, 1}|{0, 3, 2}` shows in the neighbor => eliminate

- **Final edge list:** `{{0, 1}, {0, 3}, {0, 4}, {1, 2}, {1, 3}, {2, 3}, {3, 4}}`

  Neighboring triangle list: `{{1, 2}, {0, 1}}` (for bending) (or use neighboring edge list)

Integrators

Explicit Integration

Scheme

(Notations: m_i - Mass of vertex i; E - Edge list; L - Edge length list (pre-computed))

- For every vertex:
  - $\mathbf{f}_i \leftarrow \text{Force}(\mathbf{x}_i, \mathbf{v}_i)$
  - $\mathbf{v}_i \leftarrow \mathbf{v}_i + \Delta t m_i^{-1} \mathbf{f}_i$
  - $\mathbf{x}_i \leftarrow \mathbf{x}_i + \Delta t \mathbf{v}_i$
- To compute Spring Forces: (For every spring e)
  - $i \leftarrow E[e][0]$ (Spring index $[e]$ and vertex index $[0]$)
  - $j \leftarrow E[e][1]$

- $L_e \leftarrow L[e]$
- $\mathbf{f} \leftarrow -k(\|\mathbf{x}_i - \mathbf{x}_j\| - L_e)\frac{\mathbf{x}_i-\mathbf{x}_j}{\|\mathbf{x}_i-\mathbf{x}_j\|}$
- $\mathbf{f}_i \leftarrow \mathbf{f}_i + \mathbf{f}$
- $\mathbf{f}_j \leftarrow \mathbf{f}_j + \mathbf{f}$

Problem

Overshooting: when k and/or Δt is too large

Naive solution: Reduce Δt => Slow down the simulation

Implicit Integration

General Scheme (Euler Method)

Integrate both $\mathbf{x}$ and $\mathbf{v}$ implicitly ($\mathbf{M} \in \mathbb{R}^{3N\times 3N}$ - Mass matrix (usually diagonal))

$$\begin{cases}\mathbf{v}^{[1]} = \mathbf{v}^{[0]} + \Delta t\mathbf{M}^{-1}\mathbf{f}^{[1]} \\ \mathbf{x}^{[1]} = \mathbf{x}^{[0]} + \Delta t\mathbf{v}^{[1]}\end{cases} \quad \text{or} \quad \begin{cases}\mathbf{x}^{[1]} = \mathbf{x}^{[0]} + \Delta t\mathbf{v}^{[0]} + \Delta t^2\mathbf{M}^{-1}\mathbf{f}^{[1]} \\ \mathbf{v}^{[1]} = \left(\mathbf{x}^{[1]} - \mathbf{x}^{[0]}\right)/\Delta t\end{cases}$$

$\mathbf{v}^{[1]}$ & $\mathbf{x}^{[1]}$ are unknown for the current time step => Find the x and v:

Assume $\mathbf{f}$ dep only on $\mathbf{x}$ (homonomic): Solve the eqn (Problem: $\mathbf{f}$ may not be a linear function)

$$\mathbf{x}^{[1]} = \mathbf{x}^{[0]} + \Delta t\mathbf{v}^{[0]} + \Delta t^2\mathbf{M}^{-1}\mathbf{f}\left(\mathbf{x}^{[1]}\right)$$

The equation equiv to the following (where $\|\mathbf{x}\|^2_{\mathbf{M}} = \mathbf{x}^{\mathrm{T}}\mathbf{M}\mathbf{x}$) => **Optimization prob** => Numerical schemes (Usually only conservative forces can use this energy)

$$\mathbf{x}^{[1]} = \operatorname{argmin} F(\mathbf{x}) \quad \text{for} \quad F(\mathbf{x}) = \frac{1}{2\Delta t^2}\left\|\mathbf{x} - \mathbf{x}^{[0]} - \Delta t\mathbf{v}^{[0]}\right\|^2_{\mathbf{M}} + E(\mathbf{x})$$

Because: (applicable for every system; The first order der of $F(\mathbf{x})$ reaches 0 for the min pt.)

$$\nabla F\left(\mathbf{x}^{[1]}\right) = \frac{1}{\Delta t^2}\mathbf{M}\left(\mathbf{x}^{[1]} - \mathbf{x}^{[0]} - \Delta t\mathbf{v}^{[0]}\right) - \mathbf{f}\left(\mathbf{x}^{[1]}\right) = \mathbf{0} \Rightarrow \mathbf{x}^{[1]} - \mathbf{x}^{[0]} - \Delta t\mathbf{v}^{[0]} - \Delta t^2\mathbf{M}^{-1}\mathbf{f}\left(\mathbf{x}^{[1]}\right) = \mathbf{0}$$

The Optimization Problem

Newton-Raphson Method

Solving optimization problem: $x^{[1]} = \operatorname{argmin} F(x)$ ($F(x)$ is Lipschitz continuous)

Given a current $x^{(k)}$ we approx the goal by $0 = F'(x) \approx F'(x^{(k)}) + F''(x^{(k)})(x - x^{(k)})$ (Taylor Expansion)

(For 2D: $\mathbf{0} = \boldsymbol{\nabla} F(\mathbf{x}) \approx \boldsymbol{\nabla} F(\mathbf{x}^{(k)}) + \frac{\partial F^2(\mathbf{x}^{(k)})}{\partial \mathbf{x}^2}(\mathbf{x} - \mathbf{x}^{(k)})$)

Steps:

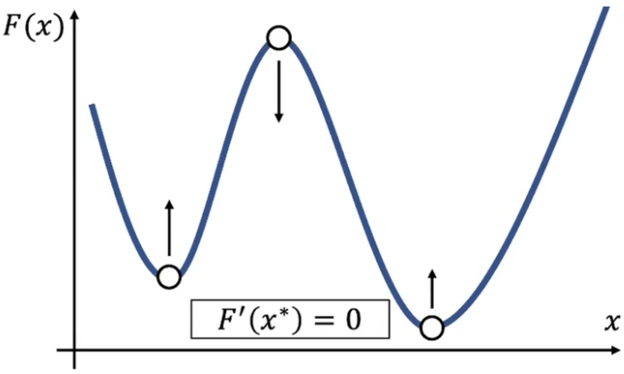
- Initialize $x^{(0)}$
- For $k = 0 \dots k$

  $\Delta x \leftarrow -(F''(x^{(k)}))^{-1}F'(x^{(k)})$ (For 2D: $\Delta\mathbf{x} \leftarrow -\left(\frac{\partial F^2(\mathbf{x}^{(k)})}{\partial \mathbf{x}^2}\right)^{-1}\boldsymbol{\nabla} F\left(\mathbf{x}^{(k)}\right)$)

  $x^{(k+1)} \leftarrow x^{(k)} + \Delta x$

  If $|\Delta x|$ is small then break
- $\mathbf{x}^{[1]} \leftarrow \mathbf{x}^{(k+1)}$

Newton's Method finds extremum, but it can be min or max => finds 2nd order derivative (at local min: $F''(x^*) > 0$; at max: $F''(x^*) < 0$)



For a function which second order derivative always larger than 0 ($F''(x) > 0$ everywhere) => $F(x)$ has only one minimum

Simulation by Newton's Method

For simulation: $F(\mathbf{x}) = \frac{1}{2\Delta t^2}\left\|\mathbf{x} - \mathbf{x}^{[0]} - \Delta t \mathbf{v}^{[0]}\right\|_M^2 + E(\mathbf{x})$

- Derivative:

$$\nabla F\left(\mathbf{x}^{(k)}\right) = \frac{1}{\Delta t^2}\mathbf{M}\left(\mathbf{x}^{(k)} - \mathbf{x}^{[0]} - \Delta t \mathbf{v}^{[0]}\right) - \mathbf{f}\left(\mathbf{x}^{(k)}\right)$$

- Force Hessian Matrix: ($\mathbf{H}(\mathbf{x}^{(k)})$ - Energy Hessian)

$$\frac{\partial^2 F\left(\mathbf{x}^{(k)}\right)}{\partial \mathbf{x}^2} = \frac{1}{\Delta t^2}\mathbf{M} + \mathbf{H}\left(\mathbf{x}^{(k)}\right)$$

Steps:

- Initialize $\mathbf{x}^{(0)}$, often as $\mathbf{x}^{[0]}$ or $\mathbf{x}^{[0]} + \Delta t \mathbf{v}^{[0]}$
- For $k = 0 \ldots k$:

  Solve $\left(\frac{1}{\Delta t^2}\mathbf{M} + \mathbf{H}\left(\mathbf{x}^{(k)}\right)\right)\mathbf{x} = \left(-\frac{1}{\Delta t^2}\mathbf{M}\left(\mathbf{x}^{(k)} - \mathbf{x}^{[0]} - \Delta t \mathbf{v}^{[0]}\right) + \mathbf{f}\left(\mathbf{x}^{(k)}\right)\right)$

  $\mathbf{x}^{(k+1)} \leftarrow \mathbf{x}^{(k)} + \Delta \mathbf{x}$

  If $\|\Delta \mathbf{x}\|$ is small then break
- $\mathbf{x}^{[1]} \leftarrow \mathbf{x}^{(k+1)}$
- $\mathbf{v}^{[1]} \leftarrow (\mathbf{x}^{[1]} - \mathbf{x}^{[0]})/\Delta t$

Spring Hessian

Hessian matrix is a second order derivative (sim to $\partial^2 F/\partial \mathbf{x}^2$) => if p.d. so that the **ONLY min point** is found and has no maximum (sufficient but not neccesary condition)

$$\mathbf{H}(\mathbf{x}) = \sum_{e=\{i,j\}} \begin{bmatrix} \frac{\partial^2 E}{\partial \mathbf{x}_i^2} & \frac{\partial^2 E}{\partial \mathbf{x}_i \partial \mathbf{x}_j} \\ \frac{\partial^2 E}{\partial \mathbf{x}_i \partial \mathbf{x}_j} & \frac{\partial^2 E}{\partial \mathbf{x}_j^2} \end{bmatrix} = \sum_{e=\{i,j\}} \begin{bmatrix} \mathbf{H}_e & -\mathbf{H}_e \\ -\mathbf{H}_e & \mathbf{H}_e \end{bmatrix}$$

The matrix in the last part is 3N x 3N, every vertex is a 3D vector. The first $\mathbf{H}_e$ is at (i, i), the $-\mathbf{H}_e$ in the first line is at (i, j), ...

Positive Definite: Dep on every $\mathbf{H}_e$ (3 x 3): @ Lec.2, P48

$$\mathbf{H}_e = k\frac{\mathbf{x}_{ij}\mathbf{x}_{ij}^{\mathrm{T}}}{\|\mathbf{x}_{ij}\|^2} + k\left(1 - \frac{L}{\|\mathbf{x}_{ij}\|}\right)\left(\mathbf{I} - \frac{\mathbf{x}_{ij}\mathbf{x}_{ij}^{\mathrm{T}}}{\|\mathbf{x}_{ij}\|^2}\right); \quad \mathbf{x}_{ij} = \mathbf{x}_i - \mathbf{x}_j$$

where $k\frac{\mathbf{x}_{ij}\mathbf{x}_{ij}^{\mathrm{T}}}{\|\mathbf{x}_{ij}\|^2}$ & $\left(\mathbf{I} - \frac{\mathbf{x}_{ij}\mathbf{x}_{ij}^{\mathrm{T}}}{\|\mathbf{x}_{ij}\|^2}\right)$ are already s.p.d. (Proof by multiplying by $\mathbf{v}^{\mathrm{T}}$ and $\mathbf{v}$ at both sides)

But $\left(1 - \frac{L}{\|\mathbf{x}_{ij}\|}\right)$ can be negative when $\|\mathbf{x}_{ij}\| < L_e$ (when compressed)

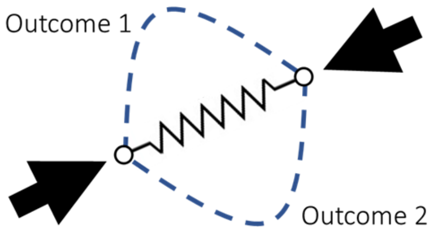
Conclusion: when stretched $\mathbf{H}_e$ is s.p.d. and when compressed $\mathbf{H}_e$ may not be s.p.d. => $\mathbf{A}$ may not be s.p.d. either

$$\mathbf{A} = \frac{1}{\Delta t^2}\mathbf{M} + \mathbf{H}(\mathbf{x}) = \underbrace{\frac{1}{\Delta t^2}\mathbf{M}}_{\text{s.p.d.}} + \sum_{e=\{i,j\}} \underbrace{\begin{bmatrix} \ddots & \vdots & \vdots & \\ & \mathbf{H}_e & -\mathbf{H}_e & \\ & -\mathbf{H}_e & \mathbf{H}_e & \\ & & & \ddots \end{bmatrix}}_{\text{may not be s.p.d.}}$$

(for smaller Δt => more p.d., actually sim to explicit integration with smaller time step => more stable)

Positive Definiteness of Hessian

When a spring is compressed, the spring Hessian may not be positive definite => Multiple minima



Only in 2D/3D: In 1D: $E(x) = \frac{1}{2}k(x-L)^2$ and $E''(x) = k > 0$

Enforcement of P.D.

Some linear solver may require $\mathbf{A}$ must be p.d. in $\mathbf{A}\Delta\mathbf{x} = \mathbf{b}$

Solution:

- Drop the ending term when $\|\mathbf{x}_{ij}\| < L_e$:

$$\mathbf{H}_e = k\frac{\mathbf{x}_{ij}\mathbf{x}_{ij}^{\mathrm{T}}}{\|\mathbf{x}_{ij}\|^2} + \cancel{k\left(1 - \frac{L}{\|\mathbf{x}_{ij}\|}\right)\left(\mathbf{I} - \frac{\mathbf{x}_{ij}\mathbf{x}_{ij}^{\mathrm{T}}}{\|\mathbf{x}_{ij}\|^2}\right)}$$

- *Choi and Ko. 2002. Stable But Responive Cloth. TOG (SIGGRAPH)*

Linear Solvers

Solving $\mathbf{A}\Delta\mathbf{x} = \mathbf{b}$

Jocobi Method

- $\Delta\mathbf{x} \leftarrow 0$

  For $k = 0 \ldots k$:

  $\mathbf{r} \leftarrow \mathbf{b} - \mathbf{A}\Delta\mathbf{x}$ // Residual error

  if $\|r\| < \varepsilon$ then break // Convergence condition ε

  $\Delta\mathbf{x} \leftarrow \Delta\mathbf{x} + \alpha\mathbf{D}^{-1}\mathbf{r}$ // Update by $\mathbf{D}$, the diagonal of $\mathbf{A}$

vanilla Jocobi method ($\alpha = 1$) has a tight convergence req on $\mathbf{A}$: Diagonal Dominant

Other Solvers

- Direct solvers (LU / LDLT / Cholesky / …) – [Intel MKL PARDISO]
  - One shot, expensive but worthy if need exact sol
  - Little restriction on $\mathbf{A}$
  - Mostly suitable on CPUs
- Iterative solvers
  - Expensive to solve exactly, but controllable
  - Convergence restriction on $\mathbf{A}$, typically positive definiteness
  - Suitable on both CPUs and GPUs
  - Easy to implement
  - Accelerable: Chebyshev, Nesterov, Conjugate Gradient …

After-Class Reading

- *Baraff and Witkin. 1998. Large Step in Cloth Simulation. SIGGRAPH.*

> One of the first papers using implicit integration. The paper proposes to use only one Newton iteration, i.e., solving only one linear system. This practice is fast, but can fail to converge.

Bending and Locking Issues

Co-rotational FEM