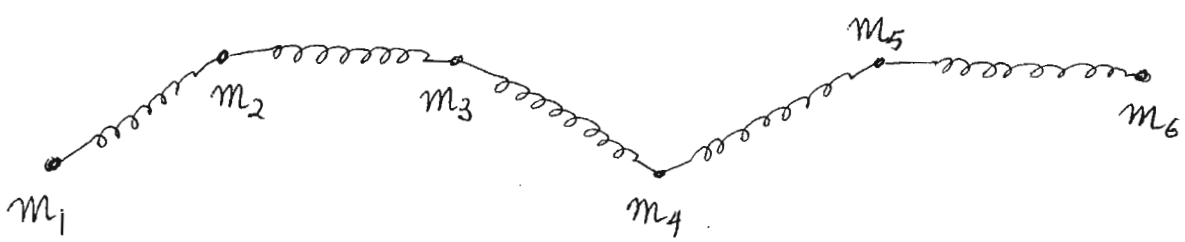
Mass-spring systems

The mass-spring model is one of the easiest ways to endow a geometric model with some physical properties. It approximates the material composition of a meshed object as follows:

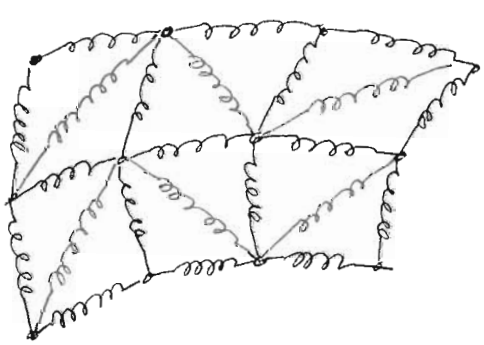
- "Point masses" are associated with each vertex of the mesh
- Massless springs are used to connect the point masses and provide "solid" stuctural support

examples

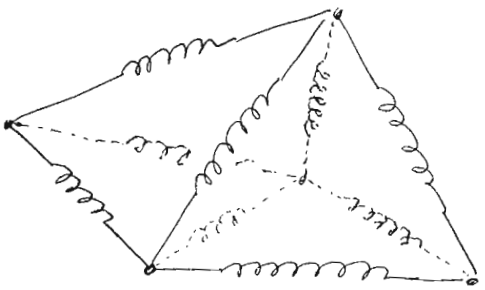
(1D) Rope / hair strand / wire ...



(2D) Cloth ...



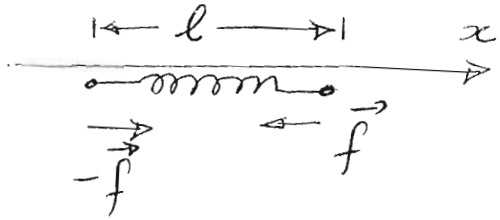
(3D) Elastic Solids...



More connectivity options than just along mesh edges; move on that on later lectures.

Mechanics of an ideal, massless spring

Baseline reference: Hooke's law



$f = -k \cdot l$ (zero rest length)

$f = -k(l - l_0)$ (finite rest length)

l = current length of spring

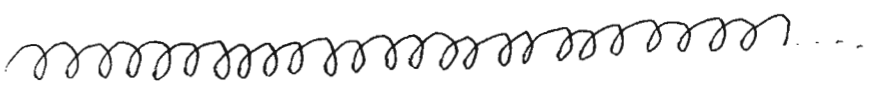
(l_0 = "natural" length of spring or "rest-length")

→ Generalization: The "spring constant" encapsulates jointly:

- → The material strength of the spring construction
- → The size (length) of the spring.

We would like the flexibility of decoupling these 2 properties. This would allow us to model "custom-length" springs, cut out of the same material.

e.g. Imagine we "cut-our-own-spring" from a continuous coiled wire...

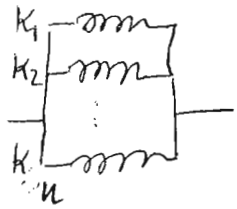


↝ What is the spring constant, if I cut this length?

↝ ... How about this length?

★ From basic spring mechanics

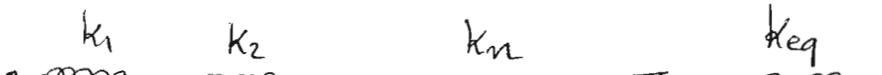
Parallel connection:



$\equiv$ 

$$k_{eq} = k_1 + k_2 + \cdots k_n$$

Serial connection:



$\equiv$

$$\frac{1}{k_{eq}} = \frac{1}{k_1} + \frac{1}{k_2} + \cdots + \frac{1}{k_n}$$

↳ Thus: • Connecting 2 identical springs in-line produces <u>half</u> the spring constant

• Connecting 3 identical springs: $k_{new} = k/3$

⇒ Spring constant is <u>inversely proportional</u> to coil length!

$k = \frac{\hat{k}}{l_0}$ → "Young's modulus"

l_0 → "Natural" or "rest" length

Thus Hooke's law (for a finite rest-length spring) becomes:

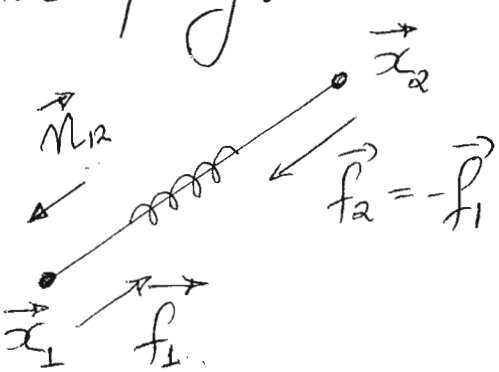
$$f = -\frac{\hat{k}}{l_0}(l - l_0)$$

(for the less-frequent case of zero-restlength springs, we retain the formulation with the "spring constant" $\hat{k}$, instead, i.e $f = -kl$)

When there is no risk of confusion, we will drop the "^" from the Young's modulus, and simply write:

$$f = -\frac{k}{l_0}(l - l_0) = -k\left(\frac{l}{l_0} - 1\right)$$

Directionality: The formula above defines the magnitude of the force, but does not concisely convey the direction it acts on. Naturally, this direction is along the length of the spring:

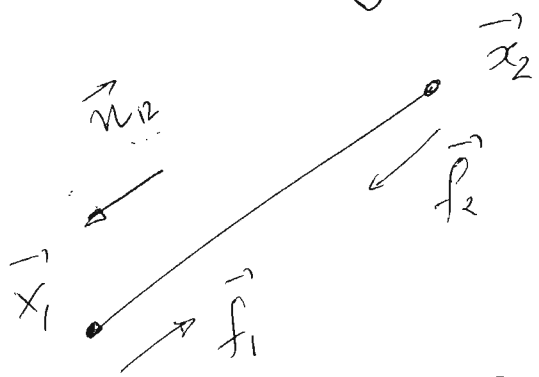


We disambiguate this by using the unit normal $\vec{n}_{12}$ in the direction of the spring:

$$\vec{n}_{12} = \frac{x_1 - x_2}{\|x_1 - x_2\|}$$

[Remember: The magnitude (or Euclidean Norm) of a vector $\vec{v} = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$ is $\|\vec{v}\|_2 = \sqrt{x^2+y^2+z^2}$]

Putting everything together, we get:



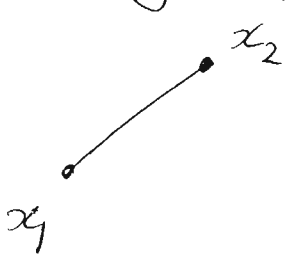
$$\vec{f}_1 = -k\left(\frac{l}{l_0} - 1\right)\vec{n}_{12}$$

or $$\vec{f}_1 = -k\left(\frac{\|\vec{x}_1 - \vec{x}_2\|_2}{l_0} - 1\right)\frac{\vec{x}_1 - \vec{x}_2}{\|\vec{x}_1 - \vec{x}_2\|}$$

or $$f_1 = -\frac{k}{l_0}\left(\frac{l}{l_0} - 1\right)(\vec{x}_1 - \vec{x}_2)$$

In any case $\vec{f}_2 = -\vec{f}_1$.

Zero-length springs



$$\vec{f}_1 = -k(\vec{x}_1 - \vec{x}_2)$$

↑ "Spring constant" not "Young's Modulus"!

Damping: Due to internal friction any realistic spring will lose energy (as heat) when stretched or compressed. This internal friction manifests as a force that resists elongation or contraction of the spring.

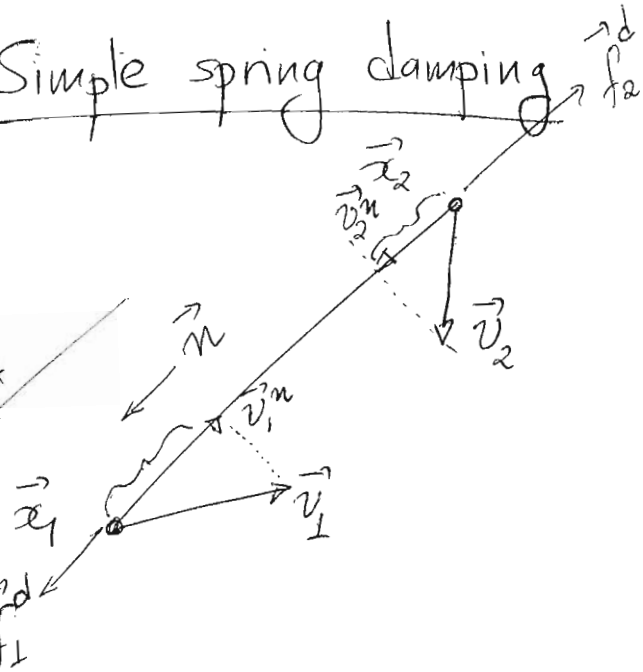
Damping is important, because:

→ Any realistic physical system exhibits it.

→ If we do not provide some systematic way to remove energy from the modeled system, numerical errors could slowly lead to a parasitic increase of energy, which is not discouraged by any explicit mechanism. This could cause the simulation to go unstable / explode.

Note: Although some damping mechanism is essential, the derivation of the mechanically accurate internal friction process may be very complicated. Thus in many cases (esp. in graphics) we employ simplified damping formulations, which serve the purpose of removing energy, albeit at a slighly inaccurat fashion compared to the real phenomenon.

Simple spring damping



Expansion (Contraction) Velocity

$$V_{rel} = v_1^n - v_2^n$$

(>0 when expanding)
(<0 when contracting

$$v_1^n = \vec{v}_1 \cdot \vec{n}$$

$$v_2^n = \vec{v}_2 \cdot \vec{n}$$

i.e. $$V_{rel} = (\vec{v}_1 - \vec{v}_2) \cdot \vec{n}$$

We will define:

$$\vec{f}_1^{\,d} = -b \cdot V_{rel} \cdot \vec{n}$$ ← the "damping" coefficient

$$= -b \left[(\vec{v}_1 - \vec{v}_2) \cdot \vec{n} \right] \vec{n}$$

$$= -b \, \vec{n} \left[\vec{n}^T (\vec{v}_1 - \vec{v}_2) \right]$$

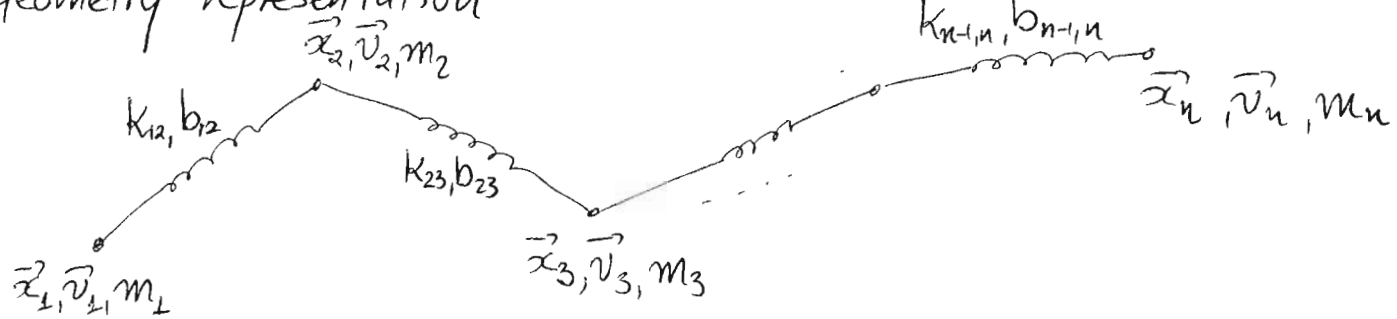
3×1 1×3 3×1

$$\boxed{\vec{f}_1^{\,d} = -b \underbrace{\vec{n}\vec{n}^T}_{3\times 3 \text{ matrix}} (\vec{v}_1 - \vec{v}_2)}$$

As always $\vec{f}_2^{\,d} = -\vec{f}_1^{\,d}$

Simulating the dynamics of a hair strand

Geometry representation



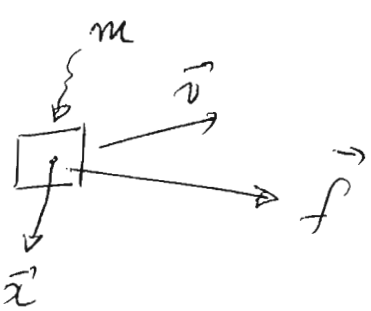
For simplicity: $m_1 = m_2 = \cdots = m_n = m$

$k_{12} = k_{23} = \cdots = k$

$b_{12} = b_{23} = \cdots = b$.

Storage $X = \begin{pmatrix} \vec{x}_1 \\ \vec{x}_2 \\ \vdots \\ \vec{x}_n \end{pmatrix}$ $V = \begin{pmatrix} \vec{v}_1 \\ \vec{v}_2 \\ \vdots \\ \vec{v}_n \end{pmatrix}$

Focus on a single body



Newton's 2nd law:

$$\vec{f}(t) = m \cdot \vec{a}(t) \quad \text{or} \quad \vec{f}(t) = m \cdot x''(t)$$

This is an ordinary differential equation for the unknown trajectory function $x(t)$

In fact it is a 2nd order ODE. We use the following trick to turn this into 2 first order equations, as follows:

Define $v(t) = x'(t)$

(thus $v'(t) = x''(t)$)

And write

$$x'(t) = v(t)$$

$$v'(t) = a(t) = \frac{1}{m} f(t)$$

$$= \frac{1}{m} f(x(t), v(t))$$

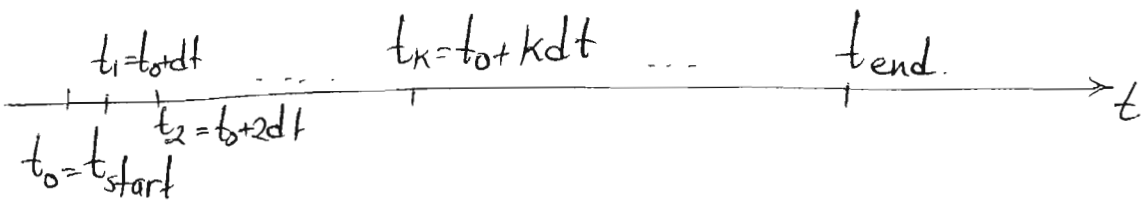
or

$$\begin{pmatrix} x(t) \\ v(t) \end{pmatrix}' = \begin{pmatrix} v(t) \\ (1/m) f\begin{pmatrix} x(t) \\ v(t) \end{pmatrix} \end{pmatrix}$$

If we call $Y(t) = \begin{pmatrix} x(t) \\ v(t) \end{pmatrix}$ this "combined" state variable then this looks like

$$Y'(t) = G(Y(t))$$

Then we subdivide our simulation timeline in "chunks" of size $\underline{dt}$



And our objective is to generate a sequence

$$\begin{bmatrix} x_1 = x(t_1) \\ v_1 = v(t_1) \end{bmatrix} \rightarrow \begin{bmatrix} x_2 = x(t_2) \\ v_2 = v(t_2) \end{bmatrix} \rightarrow \cdots \begin{bmatrix} x_n \\ v_n \end{bmatrix}.$$

Note The "Frame time" T_{frame} (i.e. the interval when we output a snapshot of the simulation is independent of $\underline{dt}$ e.g. we can take $dt = 0.001$ and $T_{frame} = 0.03$ (thus, output the result of every 30th "step").

With our "combined" State variable $Y(t) = \begin{pmatrix} x(t) \\ v(t) \end{pmatrix}$ we have

$$Y(t_0) \rightarrow Y(t_1) \rightarrow Y(t_2) \rightarrow \cdots \rightarrow Y(t_n).$$

The first Way to generate this sequence is the

Forward Euler Method