CS888 Lecture 2: More Time Integration and Some Rigid Bodies

Jan 6, 2016

Reminder: Time Slot selection

I’ve created an online sign-up sheet:

http://www.slyreply.com/app/sheets/vr72g87b8q9j/

Please pick a date from the given list.

If you select a paper too, state it in the comment section, and ensure that it correspond to the topic for that week (rigid bodies, deformables, cloth, etc.)

I may juggle the order slightly (within topics) if it makes more sense.

Last time…

Recap:

- Time integration schemes (FE, RK2, …) enable us to solve for the evolution of a physical system, given the forces driving it.
- We discussed basic particle systems, identified some typical forces.

Plan

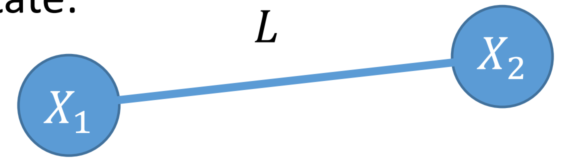
- Wrap up mass-springs intro, mention a couple more common forces.
- Further explore time integration and stability.
- Start on rigid bodies.

Hooke's Law for 3D springs

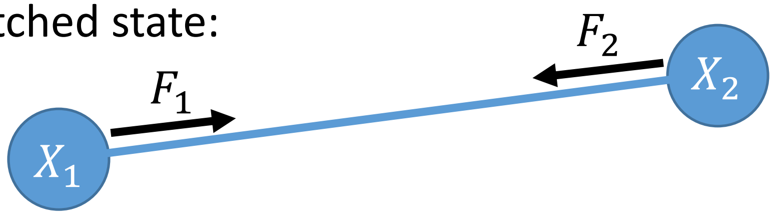
For a spring joining 2 particles with position vectors X_1 and X_2:

$$F_1 = -F_2 = -k\underbrace{(\|X_1 - X_2\| - L)}_{\text{Displacement}}\overbrace{\frac{X_1 - X_2}{\|X_1 - X_2\|}}^{\text{Direction}}$$

Rest state:

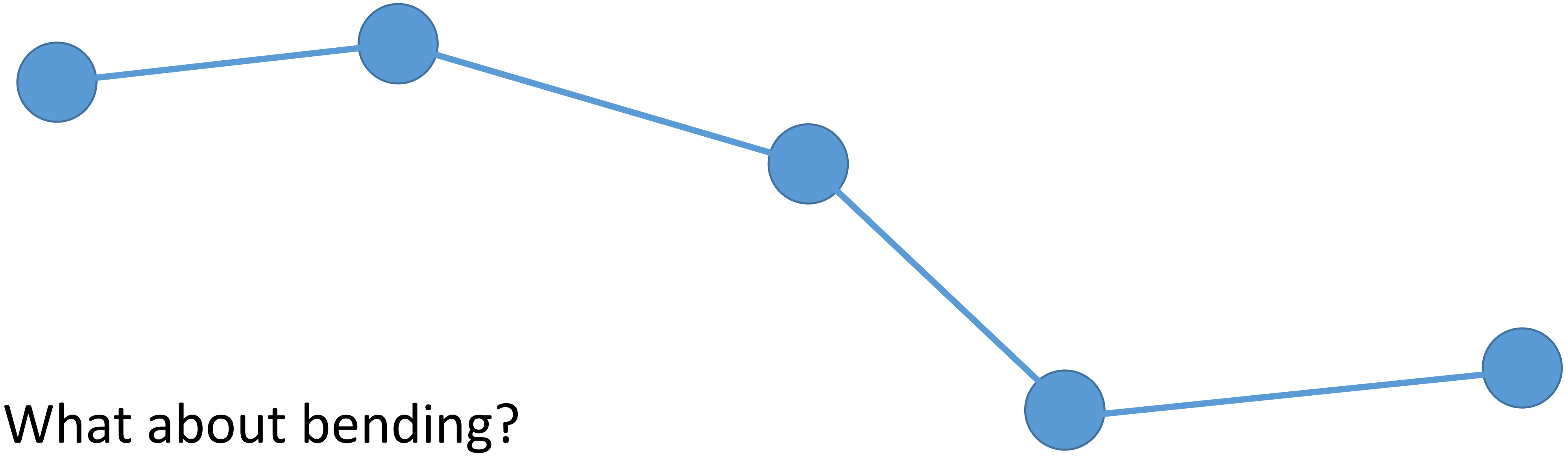


Stretched state:



Springs for Hair

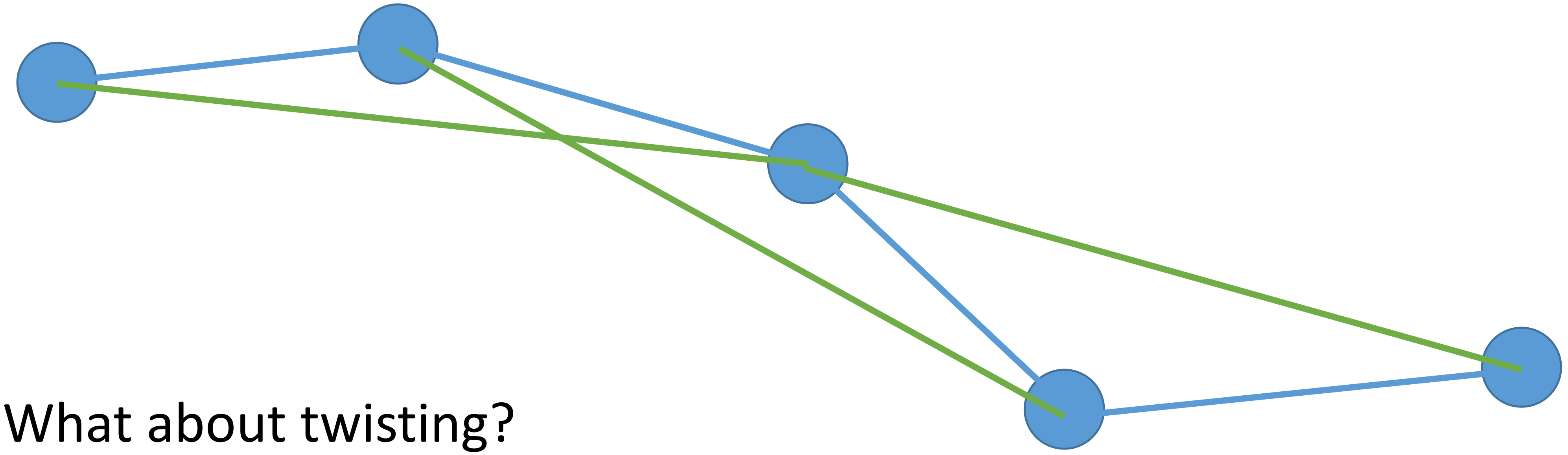
A single chain of masses and springs can model *stretching* of a strand of hair.



What about bending?

Springs for Hair

Simple model: Add *alternating* springs. This discourages the hair from collapsing when you bend it.



What about twisting?

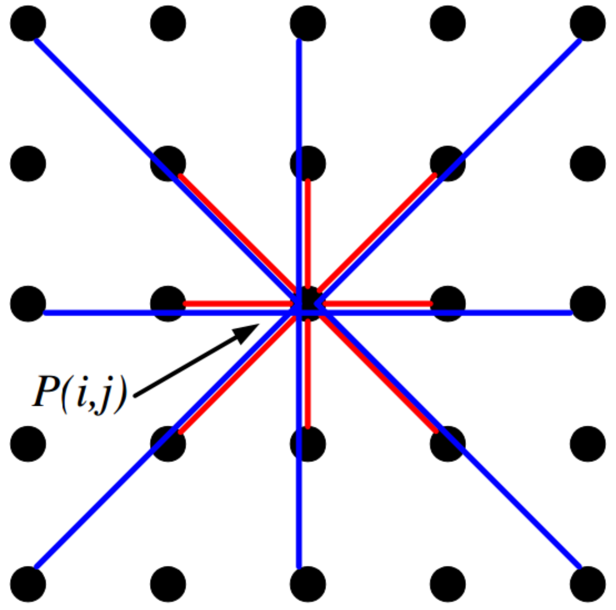
See “A Mass Spring Model for Hair Simulation” [Selle et al. 2008]

[Selle et al. 2008]

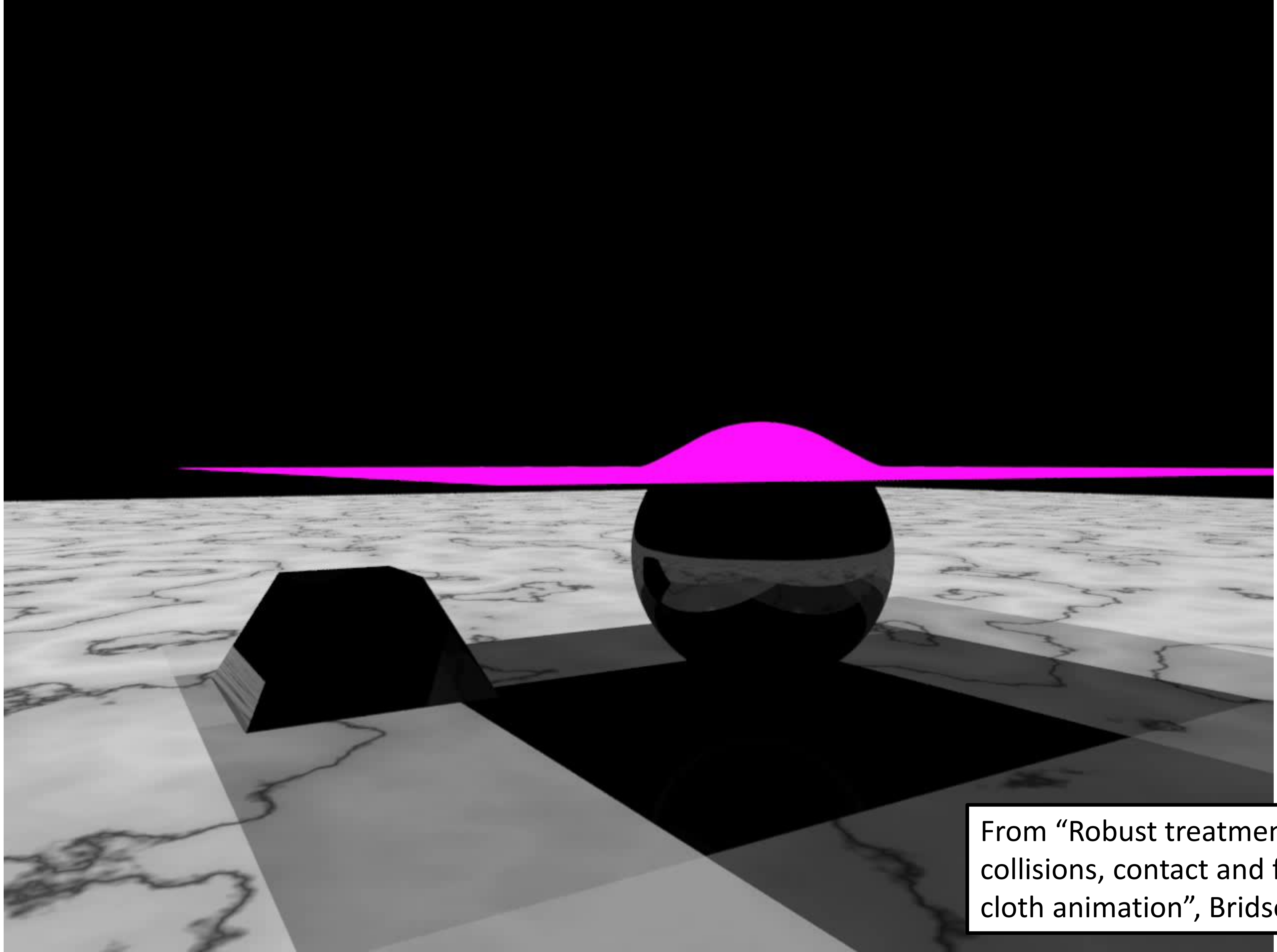
Springs for Cloth

Using a grid of particles, a similar approach has been used to model cloth.

- Red springs model stretching/shearing.
- Blue springs model bending.



From “Stable but responsive cloth”, Choi and Ko, 2002.



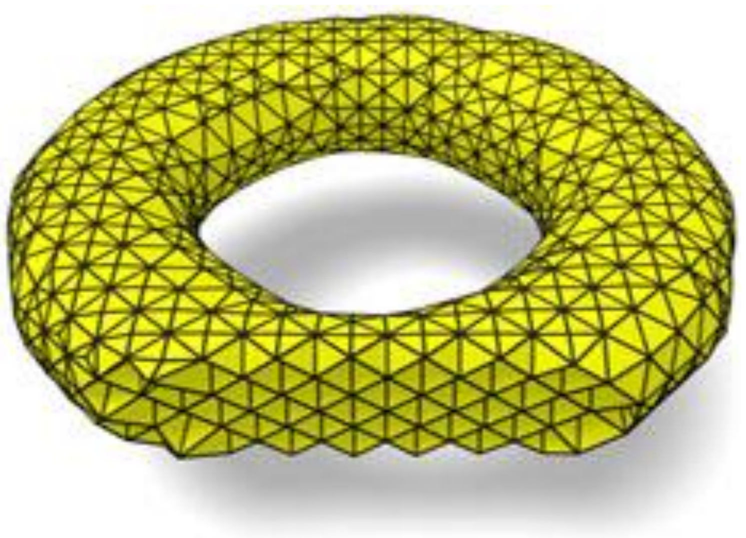
From “Robust treatment of collisions, contact and friction for cloth animation”, Bridson et al. 2002

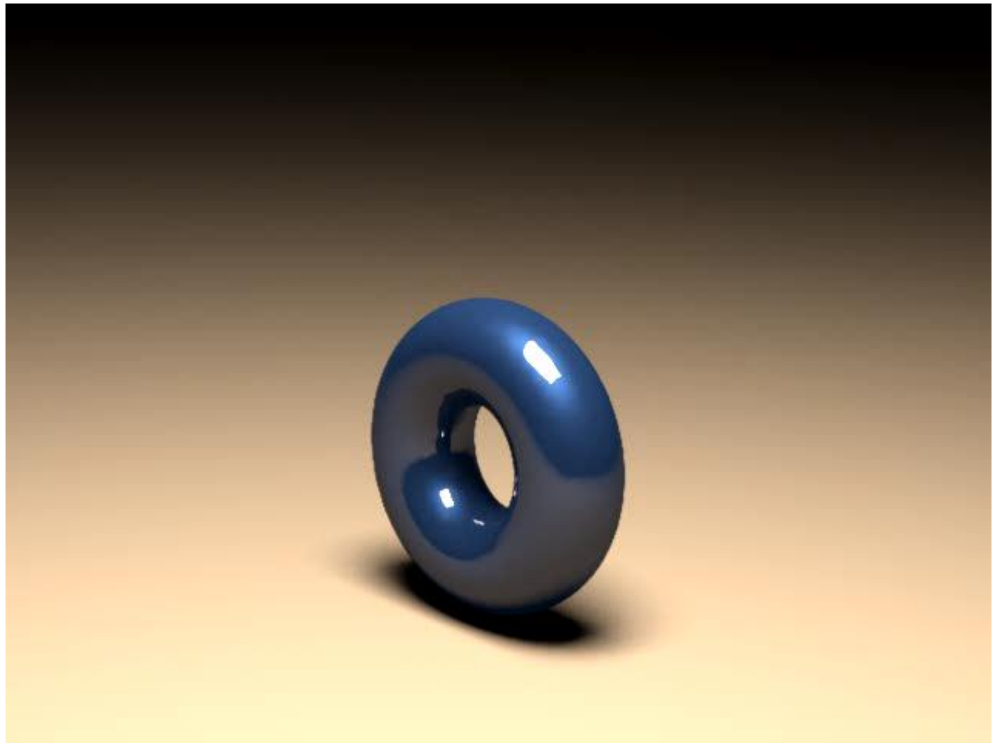
Springs for 3D Deformables

3D networks of springs form a simple model for *volumetric* deformable objects (AKA “soft body dynamics”).

- Typically connected in a mesh of tetrahedra or hexahedra.

E.g.



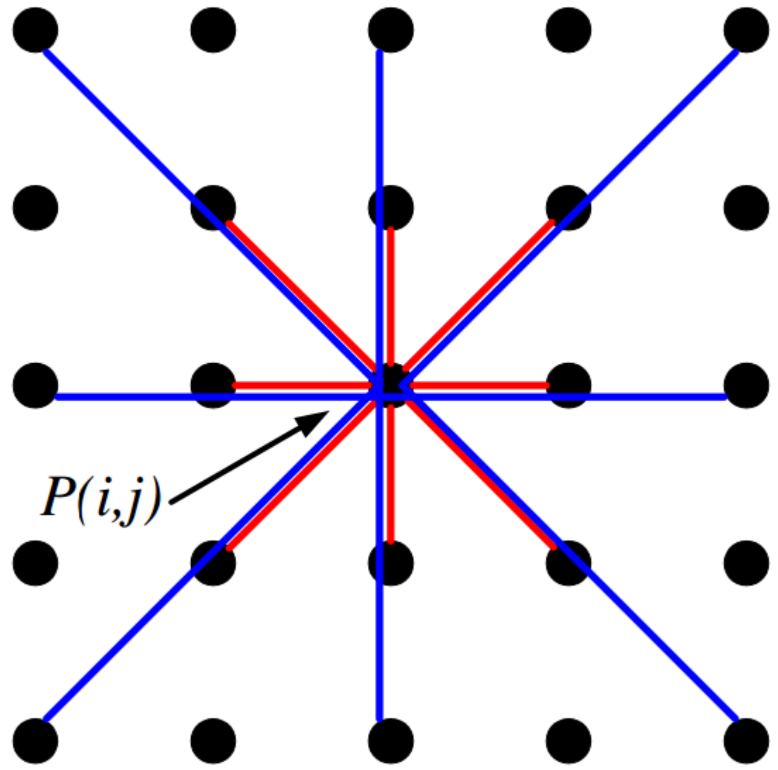


A volumetric mass-spring animation [Selle et al. 2008]

Problems with springs…

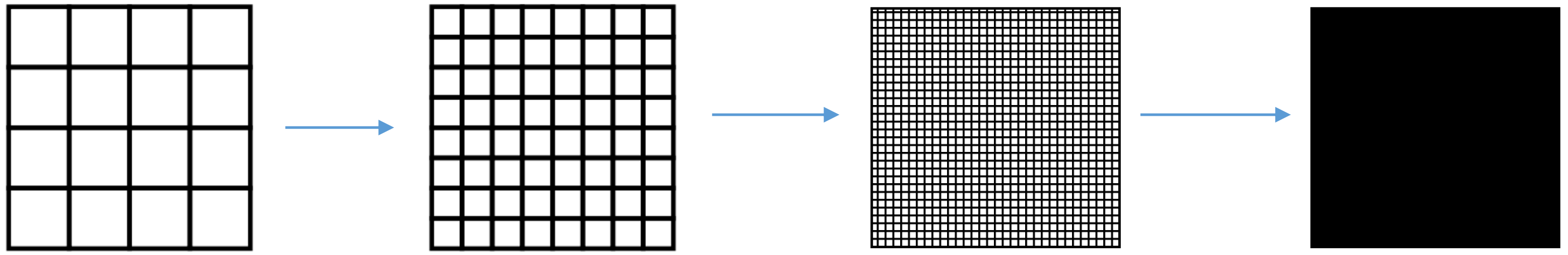
For cloth/shells, can we control stretching and bending strength *independently*?

No! "Bending" springs also affect stretchiness in a non-trivial way.



Problems with springs…

If we double the cloth mesh resolution (particle density), will the behaviour be *consistent*?



No! Ideally, the motion should approach some “true” solution as we *refine*.
In general, simple mass-spring system do not *converge*.

More consistent *finite difference, finite element*, or *finite volume* approaches are preferred in VFX, and some games have gone this way too.

More Forces: Damping/Viscosity

Approximates internal friction or energy dissipation (to heat).

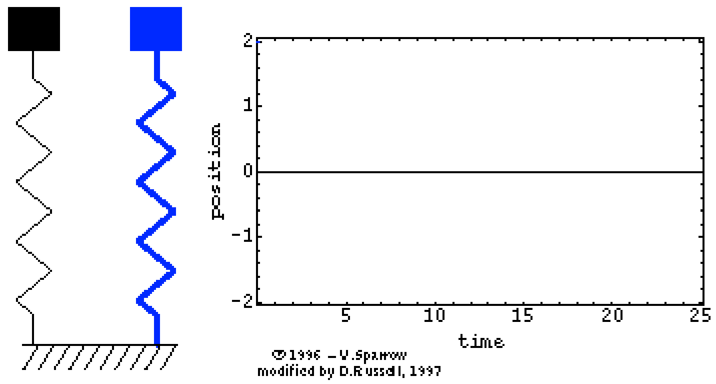
Gradually slows the motion; avoids oscillating forever.

Damping typically… (1) opposes the motion of the system, and (2) is proportional to velocity. For example…

$$F = -k_d V$$

with the damping coefficient k_d.

Notice similarity to 1D spring force, $F = -k\Delta x$.



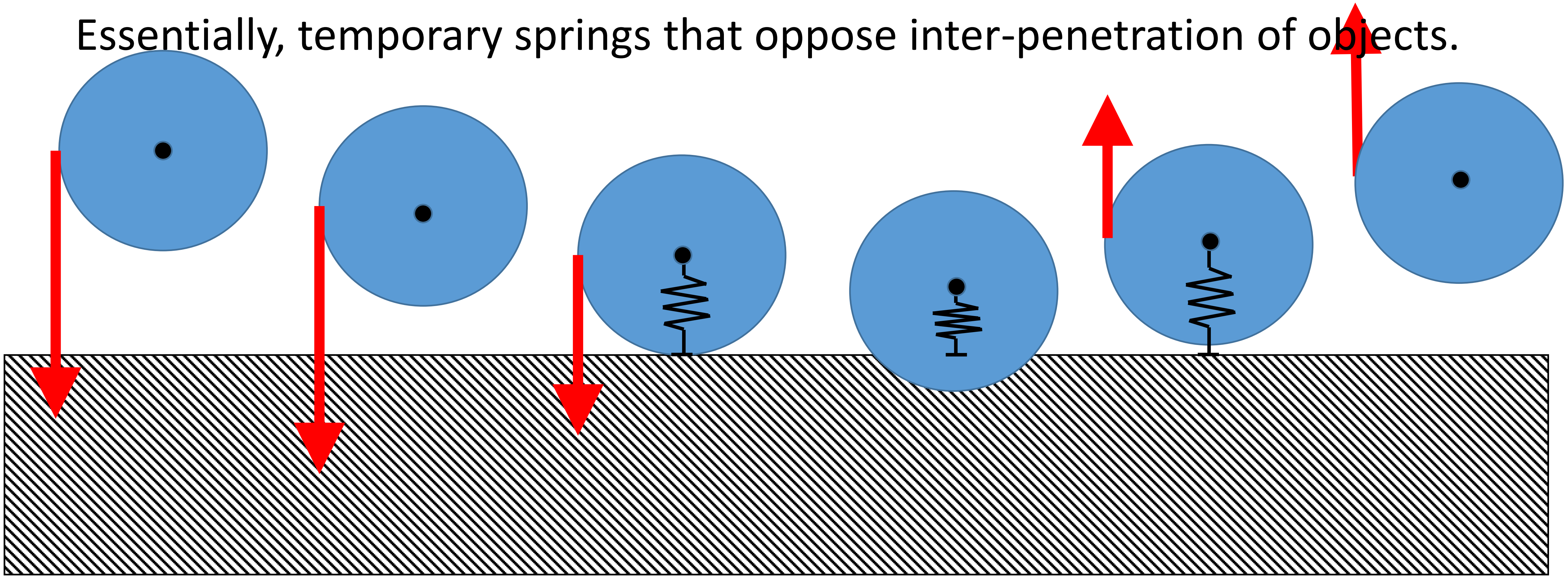
Source Link

Too much damping can sometimes look unrealistic/gooey.

More Forces: Collisions

One strategy to handle collisions is *penalty/repulsion forces*.

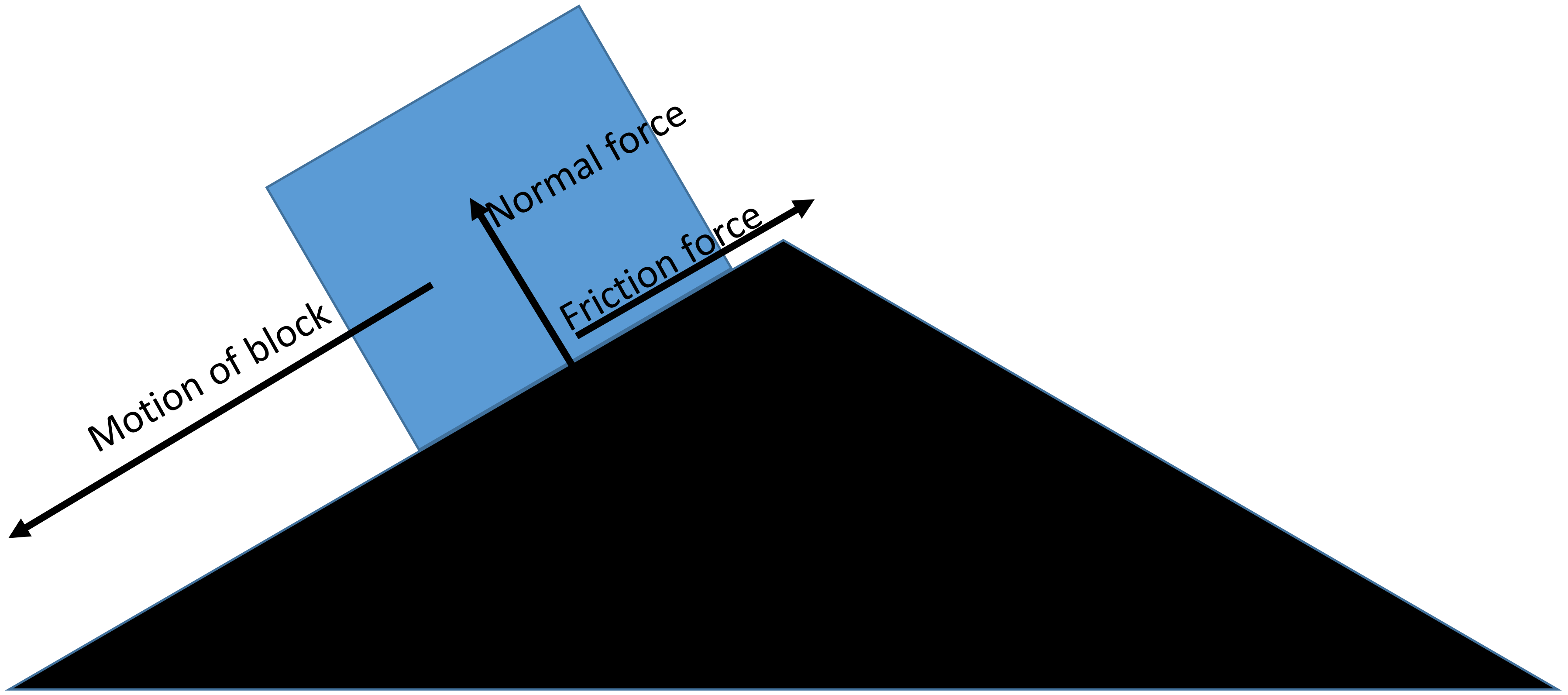
Essentially, temporary springs that oppose inter-penetration of objects.



More Forces: Contact & Friction

Normal contact force: Resists interpenetration.

Friction force: Resists *relative* sliding between materials.



A Basic Particle Simulation Loop (w/ F.E.)

```
$t = 0$
Initialize starting state (positions, velocities, etc.)
$while\ (t < t_{end})\ \{$
    Compute and sum current net force vector, $F$, per particle.
    Update particle positions: $x(t + \Delta t) = x(t) + \Delta t \mathrm{V}(\mathrm{t})$.
    Update particle velocities: $V(t + \Delta t) = V(t) + \frac{\Delta t}{m} F$.
    Increment time: $t = t + \Delta t$.
$\}$
```

Time Integration, Continued

A Little More Time Integration

Last time:

- Forward Euler (FE) and Midpoint (RK2) schemes
- Numerical time integration introduces error
- Forward Euler (and similar) can sometimes blow up!

Today:

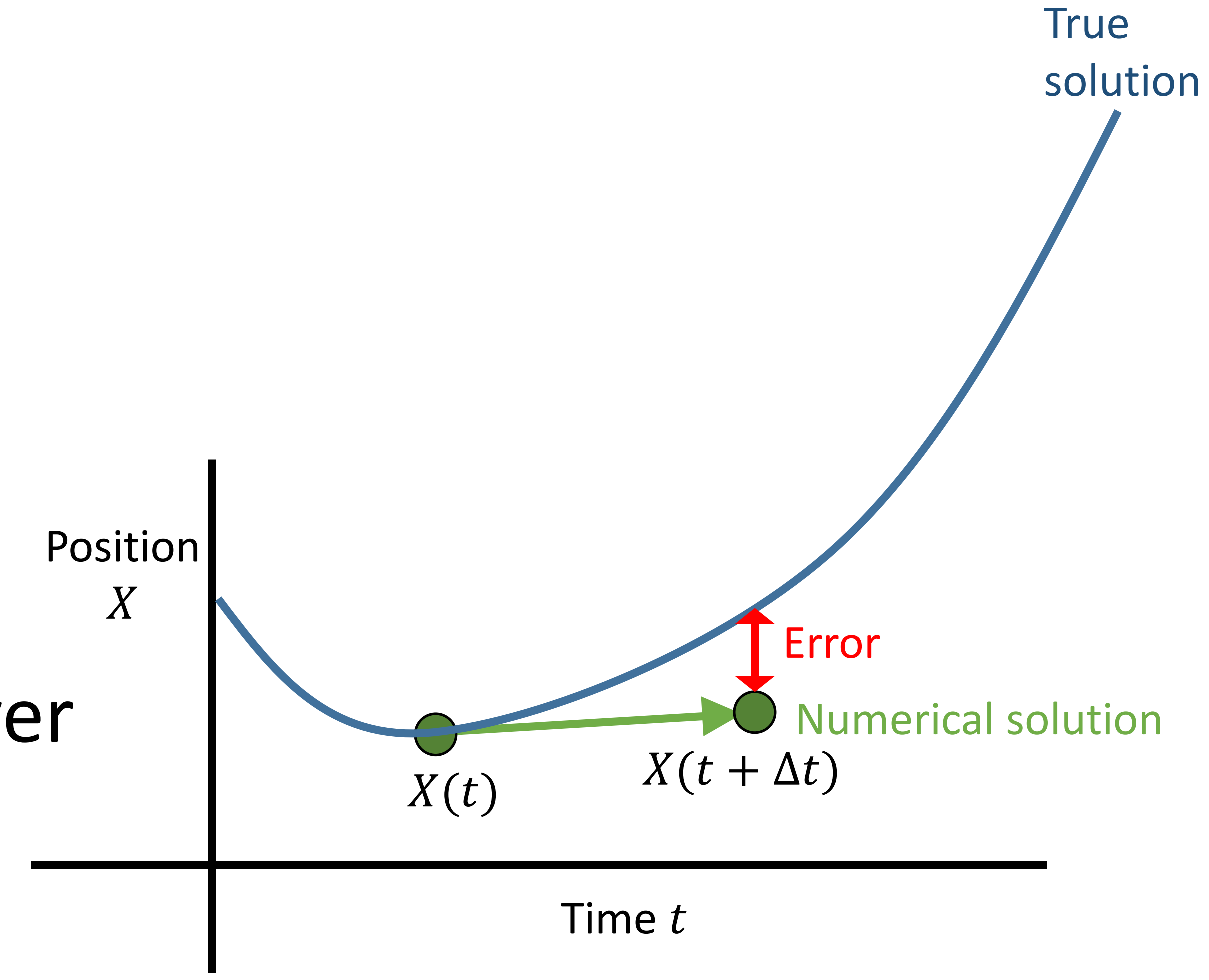
- A brief look at Forward Euler error and stability
- *Implicit* integration basics

Error of Forward Euler

For $\frac{dx}{dt} = V$, forward Euler is:

$$X(t + \Delta t) = X(t) + V(t)\Delta t$$

What is the error accumulated over ***one step***?



Error of Forward Euler

What is the *true solution* at $X(t + \Delta t)$?

We don't know it exactly, but we can approximate it with a *Taylor series* expansion:

Higher order terms, negligible for small Δt…

$$X_{true}(t + \Delta t) = X(t) + \underbrace{\Delta t X'(t)}_{\text{This is just velocity, V.}} + \frac{\Delta t^2 X''(t)}{2} + O(\Delta t^3)$$

Error of Forward Euler

$$X_{true}(t+\Delta t) = X(t) + \Delta t X'(t) + \frac{\Delta t^2 X''(t)}{2} + O(\Delta t^3)$$

$$X_{FE}(t+\Delta t) = X(t) + \Delta t X'(t)$$

The difference is called the local truncation error:

$$X_{true} - X_{FE} = \frac{\Delta t^2 X''(t)}{2} + O(\Delta t^3)$$

For small Δt, the $\mathrm{O}(\Delta t^2)$ term dominates the error.

How does error accumulate?

What is the *total* error after n steps, at time $t^n = t^0 + n\Delta t$?

Each step incurs $O(\Delta t^2)$ error. To get to t^n we take $n = \frac{t^n - t^0}{\Delta t} = O\left(\frac{1}{\Delta t}\right)$ steps.

The product gives a global error of $O(\Delta t)$; we say Forward Euler is *1st order accurate*. (Rigourous analysis depends on properties of V itself.)

Meaning: Halving Δt should reduce observed error by a factor of 2.

FE v.s. RK2

Forward Euler error is 1st order, $O(\Delta t)$.

RungeKutta2 (explicit midpoint) is 2nd order, $O(\Delta t^2)$.

→RK2 is asymptotically more accurate, i.e. approaches the exact solution faster as the timestep is reduced (“under refinement”).

Stability of Forward Euler

FE can “blow up” for large time steps, e.g. recall: $\frac{dx}{dt} = -x$.

So how large is too large?

Apply FE to the 1D linear *test equation*: $\frac{dx}{dt} = \lambda x$

Result is: $x_{n+1} = x_n + \Delta t \lambda x_n = x_n(1 + \Delta t \lambda)$

Behaviour after *many* steps starting with x_0?

$$x_n = x_0(1 + \Delta t \lambda)^n$$

Stability of Forward Euler

So, we have: $x_n = x_0(1 + \Delta t\lambda)^n$

When does this blow up (increase exponentially)? When does it decay?

For stability, we therefore need: $|1 + \Delta t\lambda| < 1$

(Assuming λ is real and negative…) this implies: $0 < \Delta t < \frac{2}{|\lambda|}$.

Larger systems

Larger, possibly nonlinear systems (e.g. particle systems, etc.) can be locally approximated by a similar linear differential equation…

$$\frac{d\boldsymbol{x}}{dt} = A\boldsymbol{x}$$

…where *A* is a matrix, ***x*** is a vector. (i.e. 1 entry per *degree of freedom*.)

How to analyze stability?

Matrix Eigenvalues

Eigenvalues characterize how a matrix A acts on corresponding *eigenvectors* v.

$$Av = \lambda v$$

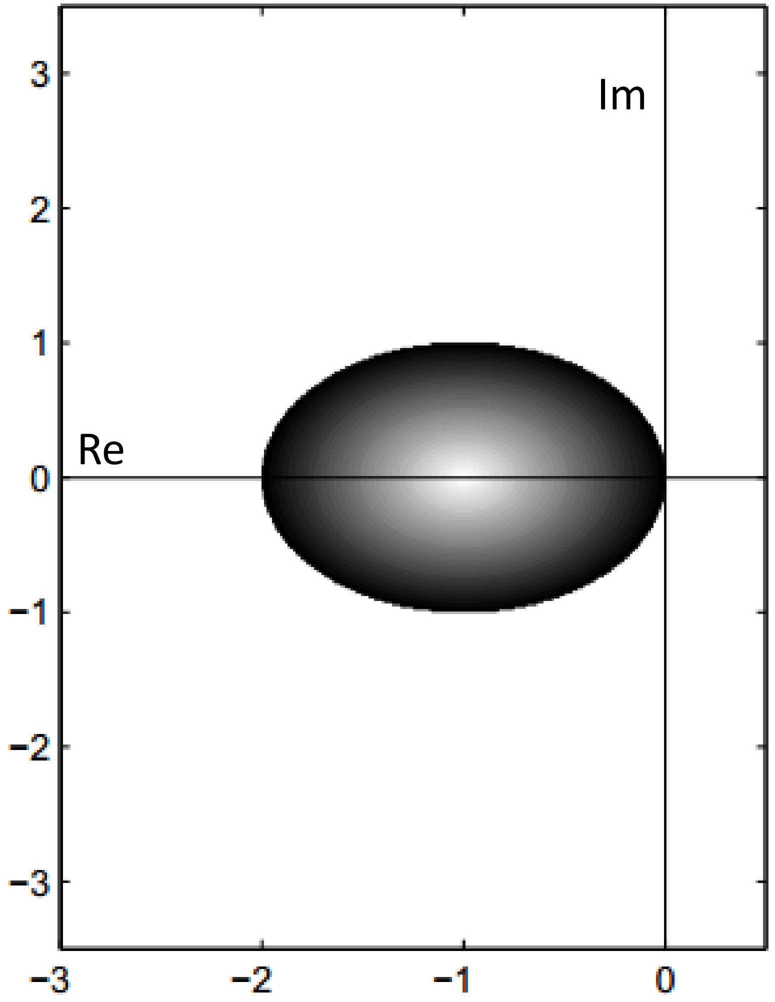
By analyzing stability of each *eigenvalue,* λ, independently, we can find a stable timestep for the most restrictive “mode”.

Stability Regions

Eigenvalues of a problem can be *complex* numbers…

We visualize the absolute *region of stability* on the *complex plane*, by plotting where the stability criterion is satisfied (by the product $\Delta t\lambda$.)

Forward Euler stability region



FE: $|1 + \Delta t\lambda| < 1$

Impact of Stability

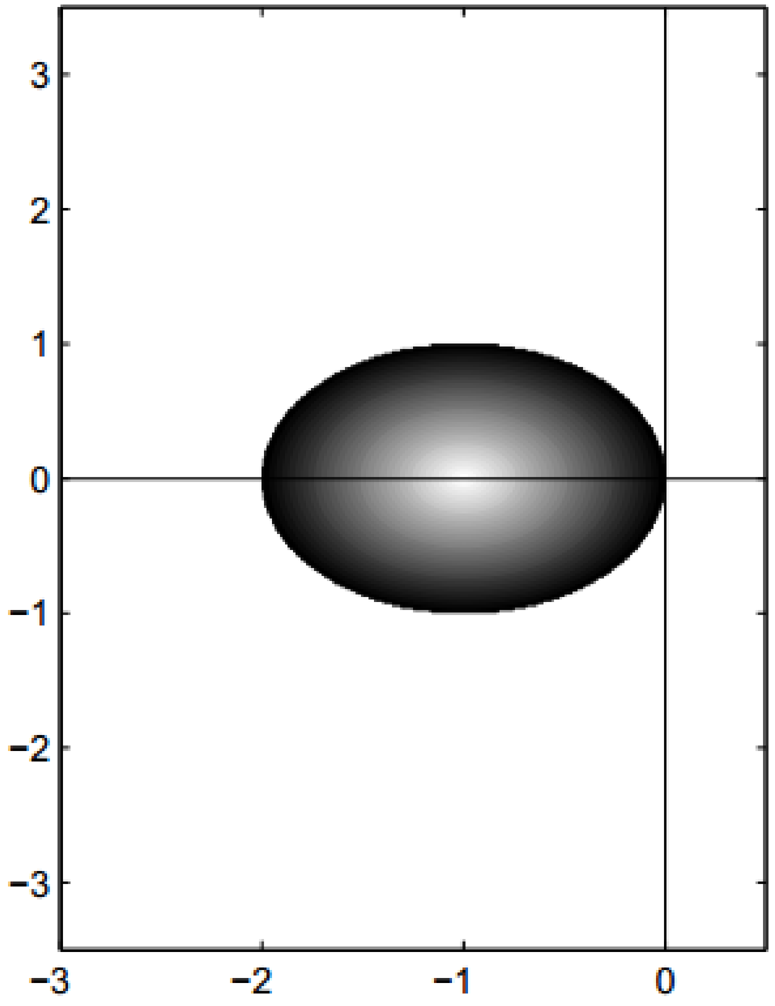
Accelerating or decaying motion corresponds to *real* eigenvalue components.

Rotational/oscillatory motion corresponds to *imaginary* components.

Often the stability restriction for a given method can be very (or impossibly!) limiting.

e.g. F.E. for pure rotational velocities, ideal springs, etc.

Forward Euler stability region



FE: $|1 + \Delta t \lambda| < 1$

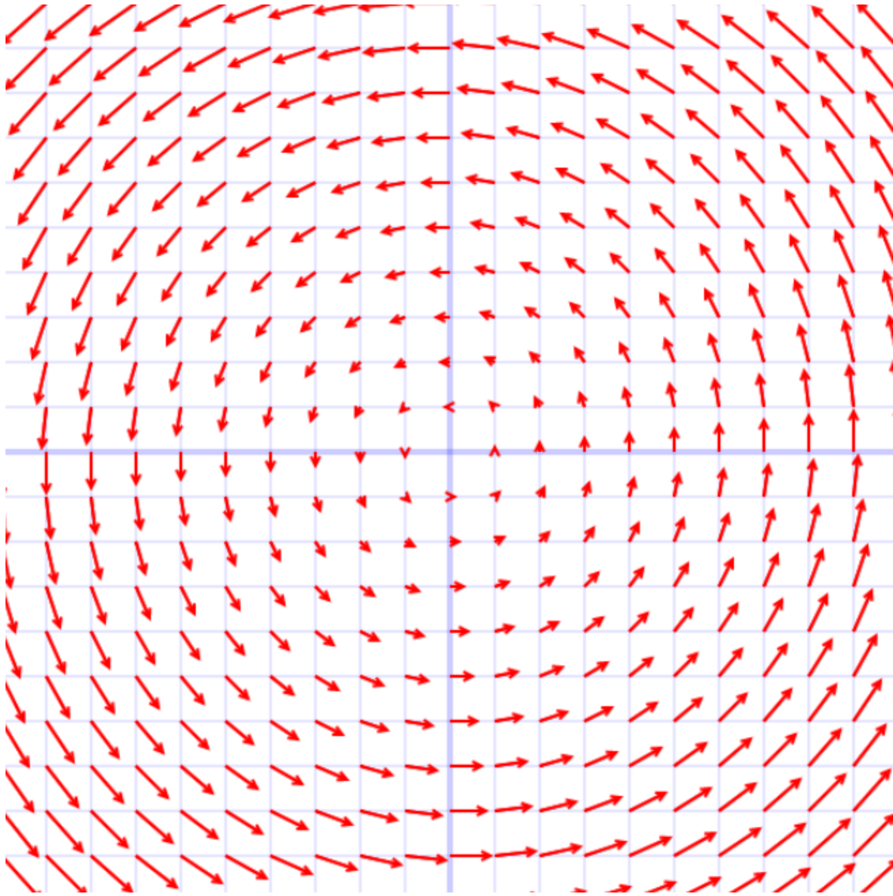
Imaginary Eigenvalues

Recall the rotational vector field $V = (-y, x)$.

The differential equation is…

$$\frac{d}{dt}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix}$$

The matrix $A = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$ has *strictly imaginary* eigenvalues, $\lambda = \pm i$.



Imaginary eigenvalues

If $\lambda = \pm i$, where does $\Delta t \lambda$ lie in this picture?

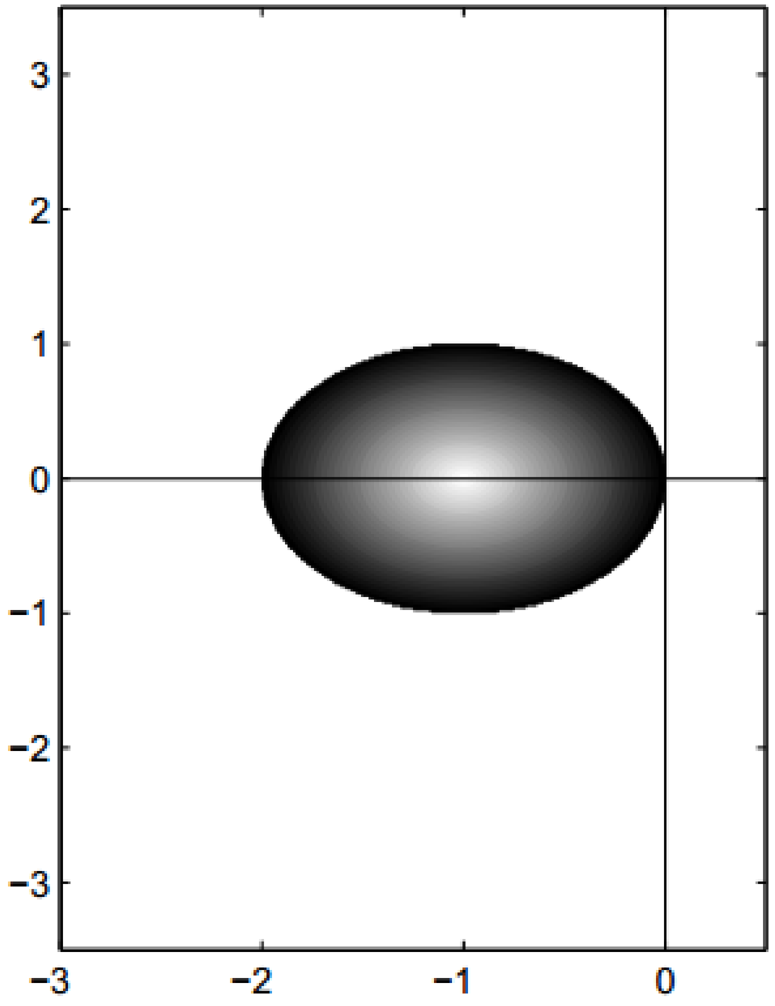
On the imaginary axis, *outside the stability region*.

F.E. is *never* stable on purely rotational motion!

Likewise, for a purely oscillating 1D mass-spring:

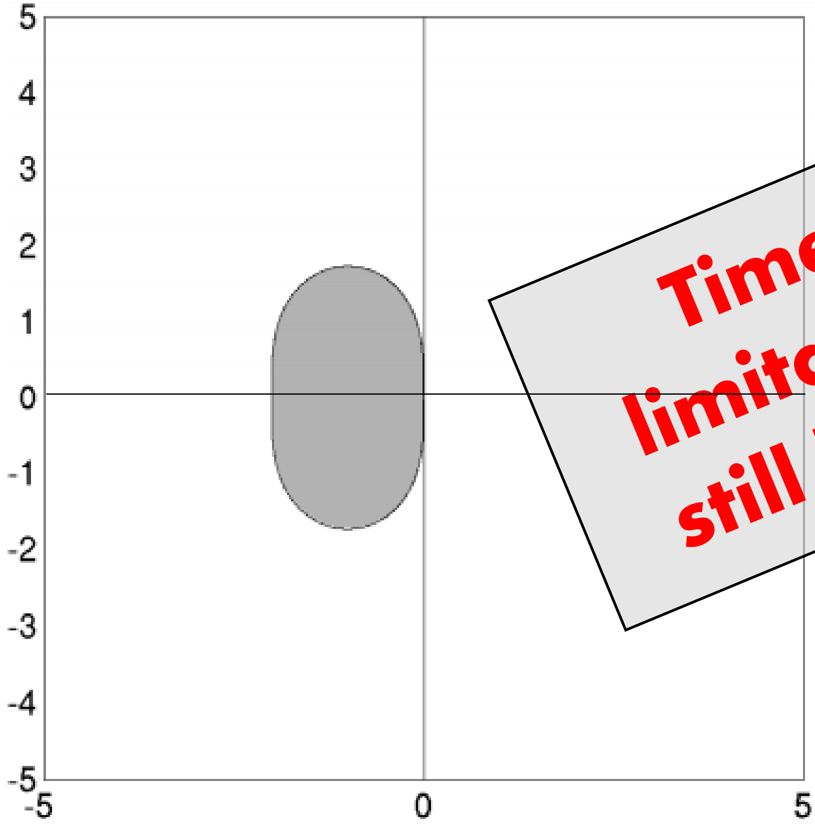
$$\frac{d}{dt}\begin{pmatrix} x \\ V \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -k/m & 0 \end{pmatrix}\begin{pmatrix} x \\ V \end{pmatrix}$$

Forward Euler stability region



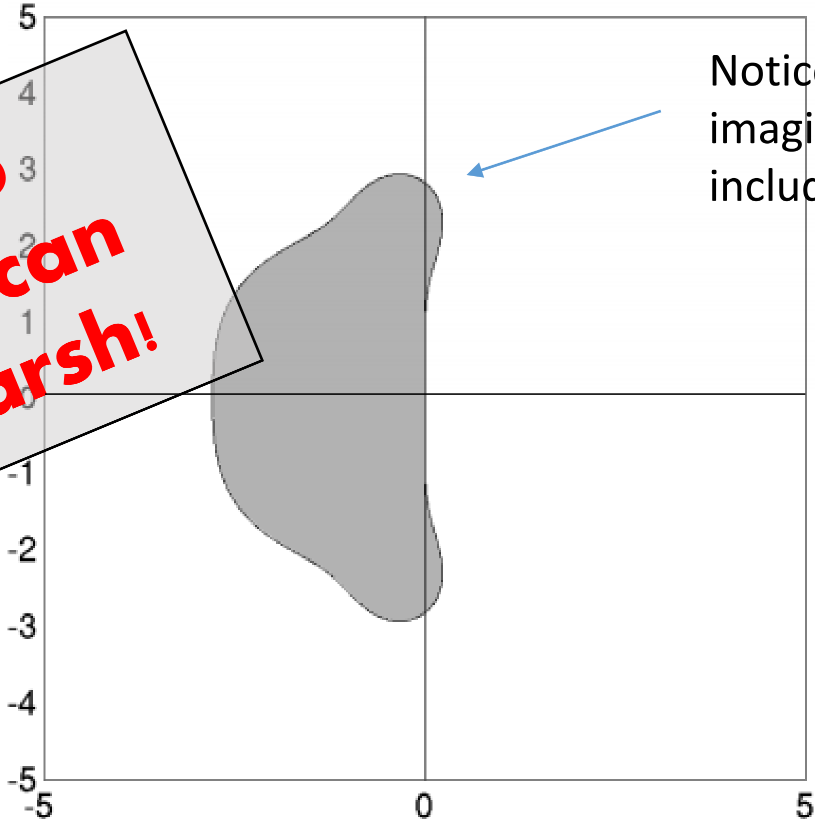
FE: $|1 + \Delta t \lambda| < 1$

Stability Regions for other (explicit) methods



Midpoint (AKA Runge Kutta 2)

Time step limitation can still be harsh!



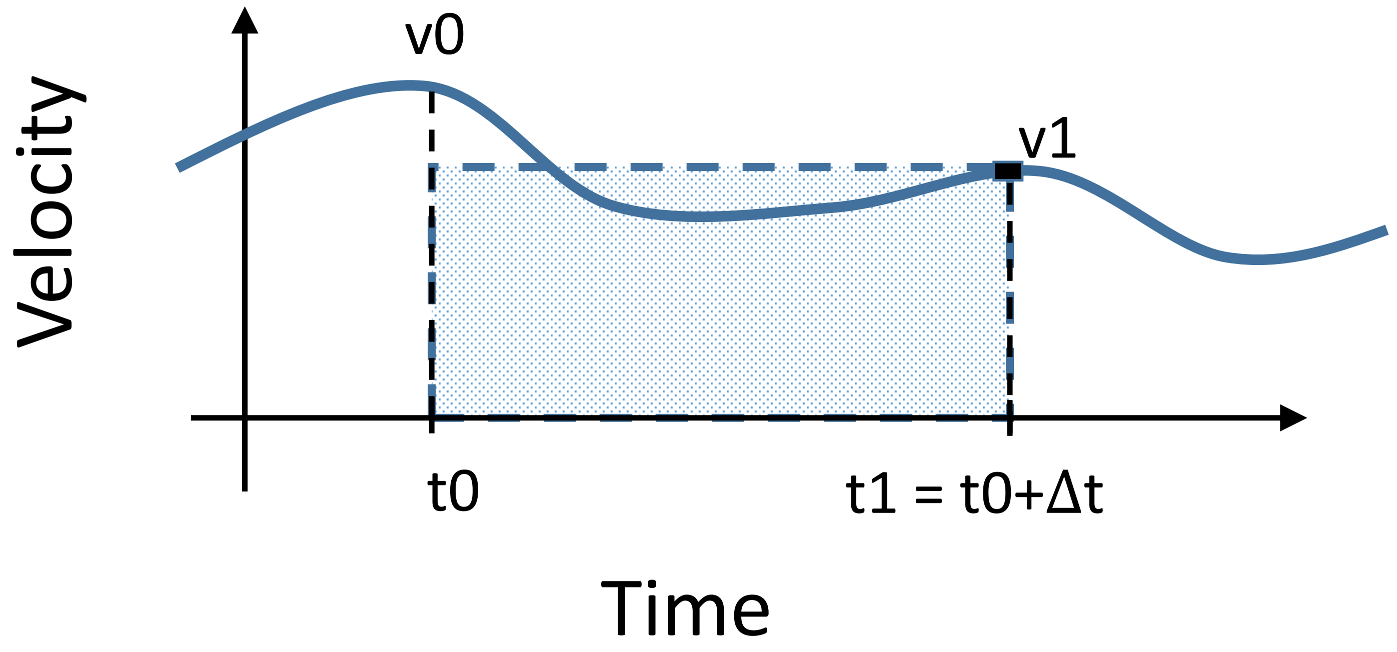
Notice parts of the imaginary axis are included here.

Runge Kutta 4
(A similar 4th order scheme)

Implicit (or backward) Euler

Another alternative scheme is **implicit Euler (**or **backward Euler)**.

Idea: use the (unknown!) velocity at the *end* of the timestep to approximate the integral.



Compare: Explicit vs. Implicit Euler

Explicit/Forward Euler:

$$X(t+\Delta t) = X(t) + V(t)\Delta t$$

Explicit schemes use known (or readily evaluated) data on the RHS.

Implicit/Backward Euler

$$X(t+\Delta t) = X(t) + \boxed{V(t+\Delta t)}\Delta t$$

Generally unknown. (If we knew the state at $t+\Delta t$, we'd be done already!)

Implicit schemes require **solving (implicit) equations.**

Stability – Backward Euler

Consider our test equation again: $\frac{dx}{dt} = \lambda x$

Applying Backward Euler gives:

$$X(t + \Delta t) = X(t) + \lambda \Delta t X(t + \Delta t)$$

This is an *implicit* equation for $X(t + \Delta t)$. i.e. it appears on both sides.

So we have to solve for it, in this case by rearranging:

$$X(t + \Delta t) = \frac{X(t)}{(1 - \lambda \Delta t)}$$

Stability – Backward Euler

So a single step of BE is:

$$X(t + \Delta t) = \frac{X(t)}{(1 - \lambda \Delta t)}$$

Proceeding as we did for FE, after *n* steps, we have:

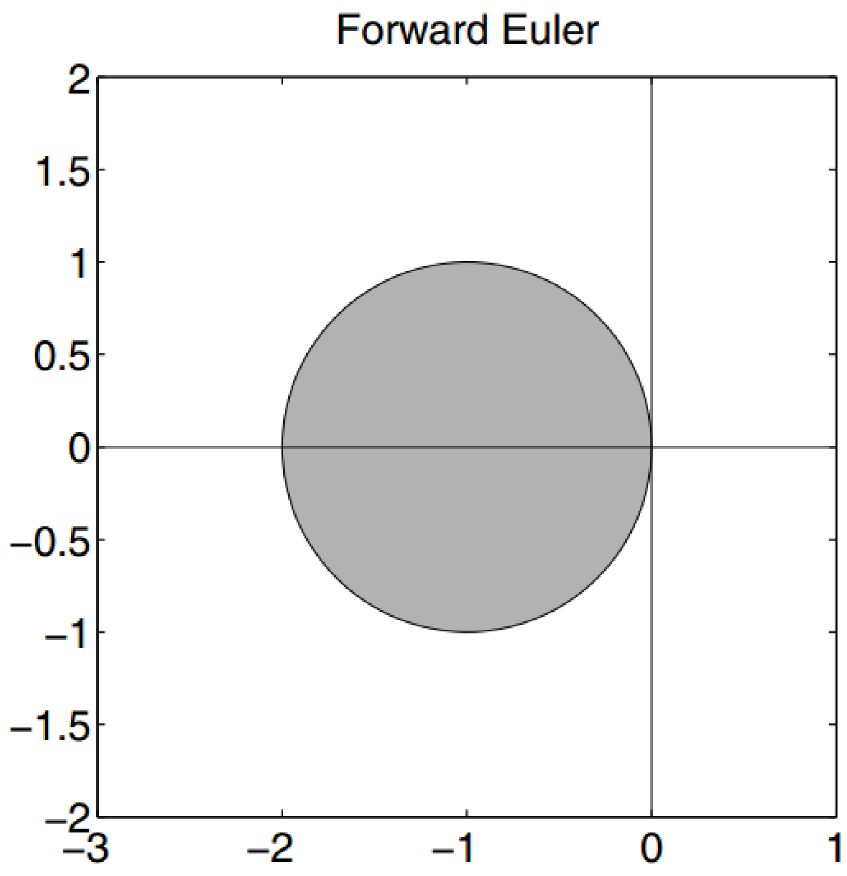
$$X_n = \frac{X_0}{(1 - \lambda \Delta t)^n}$$

When does this decay/decrease? (i.e. when is it stable?)

$$|1 - \lambda \Delta t| \geq 1$$

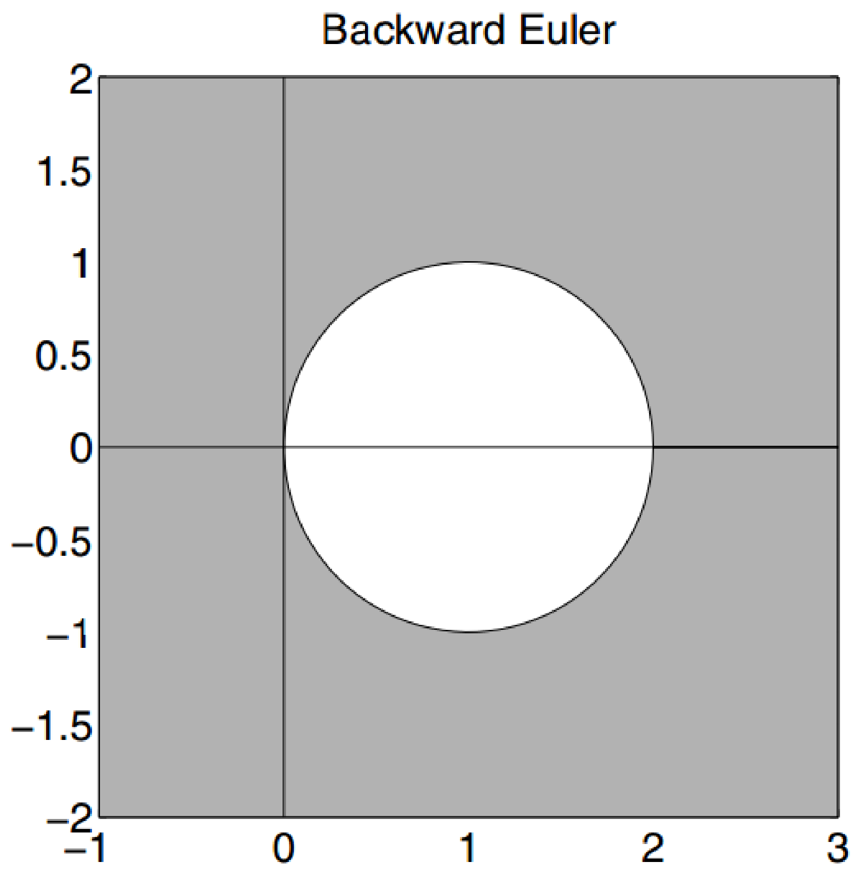
Stability Regions – Explicit vs. Implicit Euler

(a)



$$|1+\lambda\Delta t| \leq 1$$

(b)



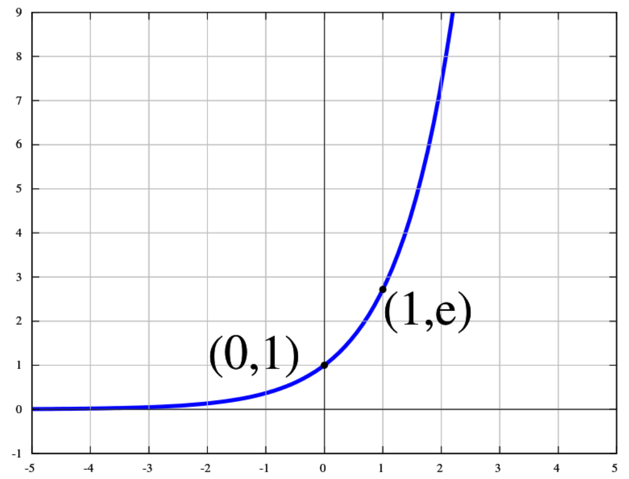
$$|1-\lambda\Delta t| \geq 1$$

Implicit Euler

B.E. sometimes behaves stably (i.e. decays) even when the “physics”/problem does not!

e.g.,

$$\frac{dx}{dt} = x$$



Solution is e^t, which grows exponentially.

(Imagine a physical system that gains energy exponentially.)

Lies on the right half of the complex plane, $Re(\Delta t\lambda) > 0$.

BE gives:

$$X_n = \frac{X_0}{(1 - \Delta t)^n}$$

If you take large steps ($\Delta t > 2$), the numerical solution will decay in magnitude, which doesn’t match the true solution! (Stability and accuracy are distinct issues…)

Implicit/Backward Euler

In practice:

- BE is stable for large time steps.
- The price is overly *damped* motion (artificial loss of energy).
- Quite common in graphics (and elsewhere).

For problems with multiple variables, need to solve a *linear system* of equations. (See CS370,CS475.)

The system to solve may also be ***non-linear***…

  - Need to iterate, using Newton’s method or other nonlinear solver.
  - Typically requires computing *derivatives* of the functions/forces.

Trapezoidal rule

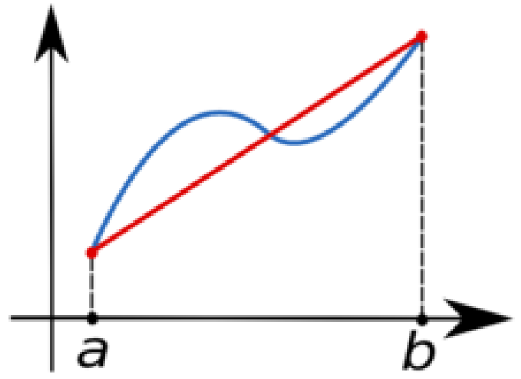
Some time integrations schemes mix explicit and implicit integration.

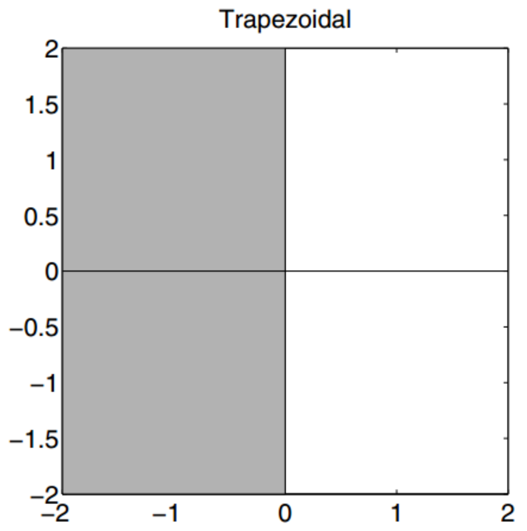
Trapezoidal rule is half a step of FE, half a step of BE!

$$X(t + \Delta t) = X(t) + \Delta t \left(\frac{V(t) + V(t + \Delta t)}{2} \right)$$

Stable when the problem is.

i.e., the left half of the complex plane.



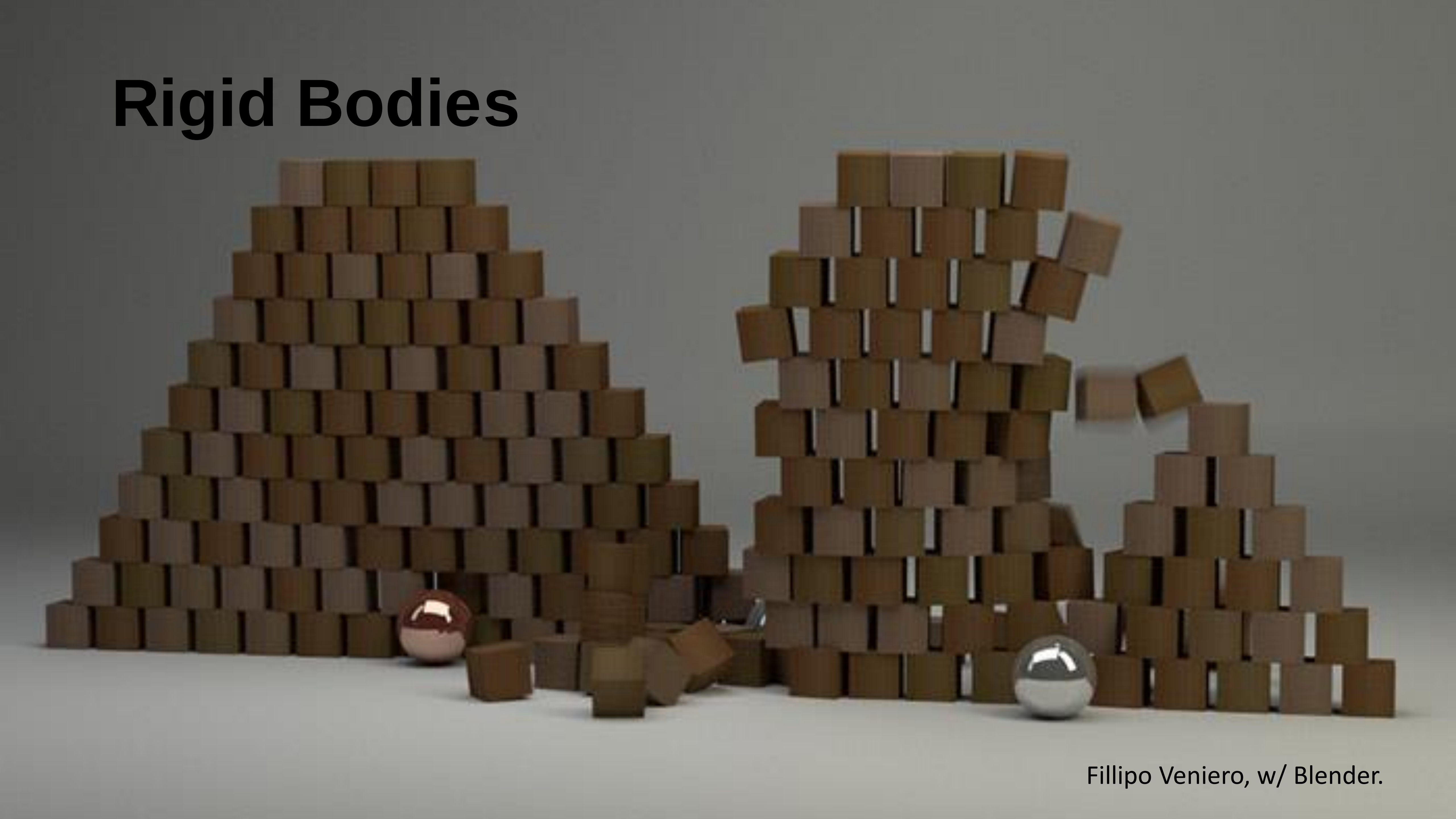


Caveat…

Technically, these stability analyses only apply to linear problems (or local linearizations).

But, they are often *indicative* of behaviour for nonlinear problems too.

Rigid Bodies

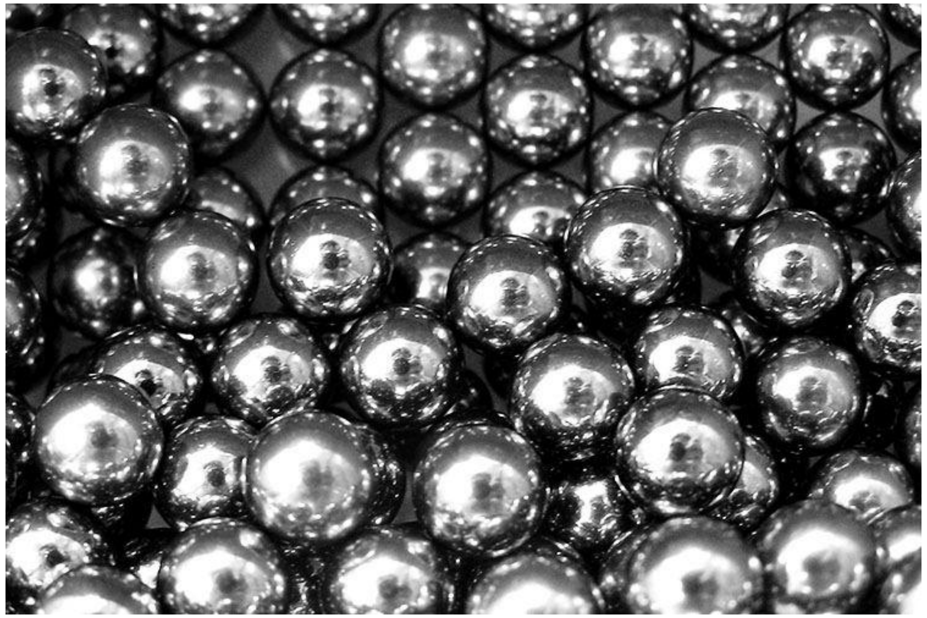


Fillipo Veniero, w/ Blender.

Rigid Bodies

For modeling ***very*** stiff objects (ie. large k), a mass-spring/deformable system is problematic… Why?

- Explicit methods: will need very small time steps to satisfy stability.
- Implicit methods: equations become ill-conditioned (harder to solve).
- Deformations happen very fast, but are not the (visually) interesting part!



Abstraction

Don’t represent/solve for things we don’t care about!

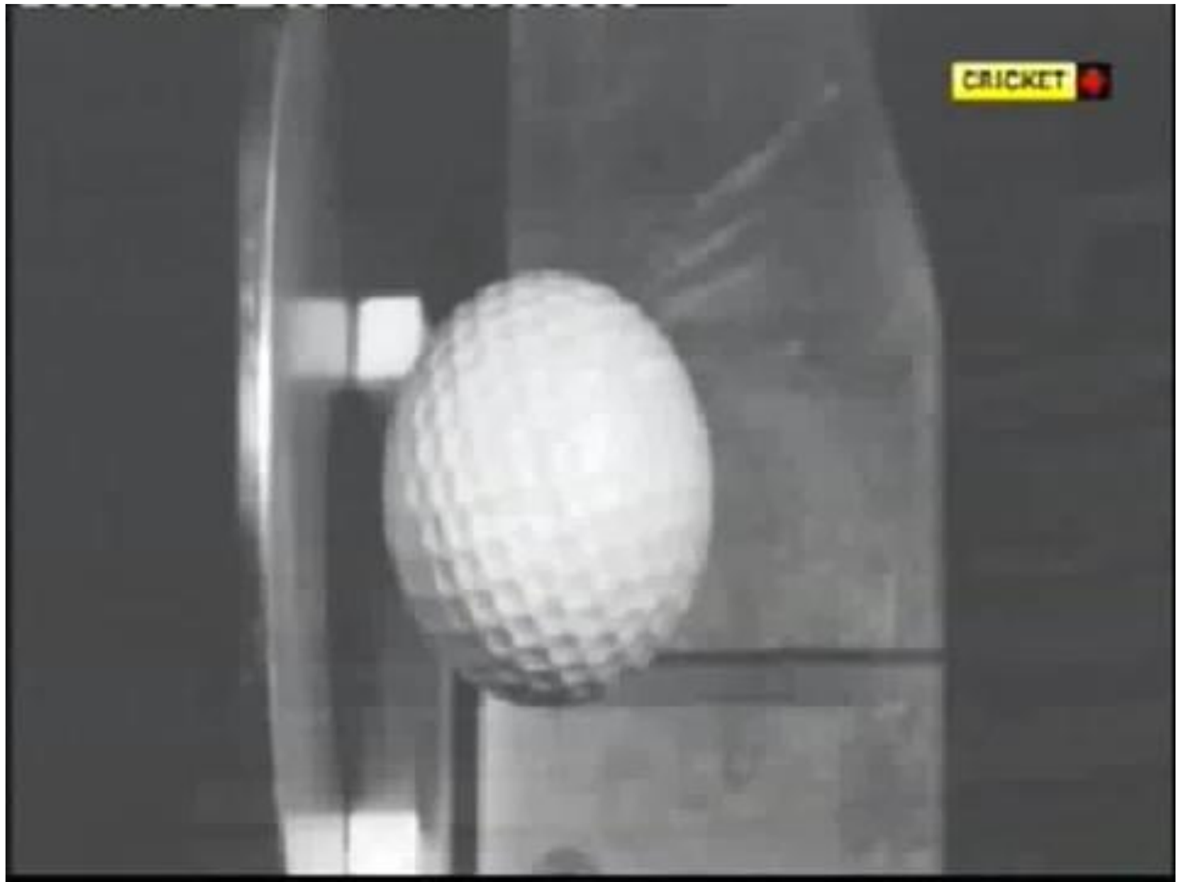
Assume that there is **no** deformation at all, i.e., perfectly rigid. What does this buy us?

The only possible motions are **translations** and **rotations**.

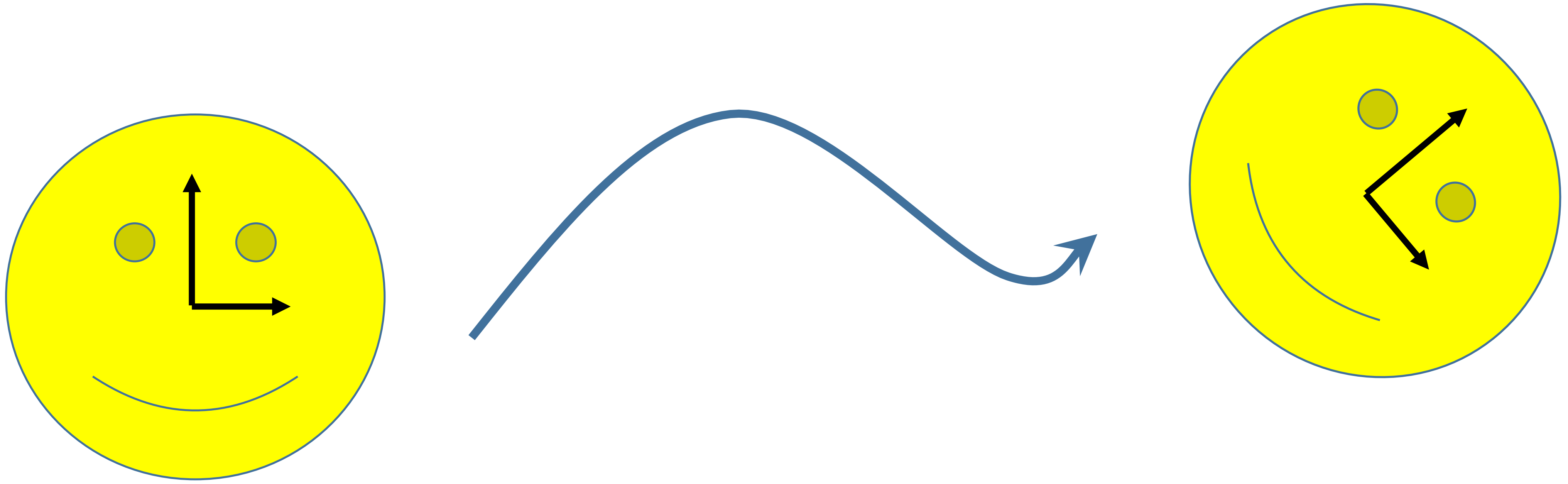
The *state* of the rigid body involves **position** and **orientation** info only; we evolve these quantities.

Abstraction

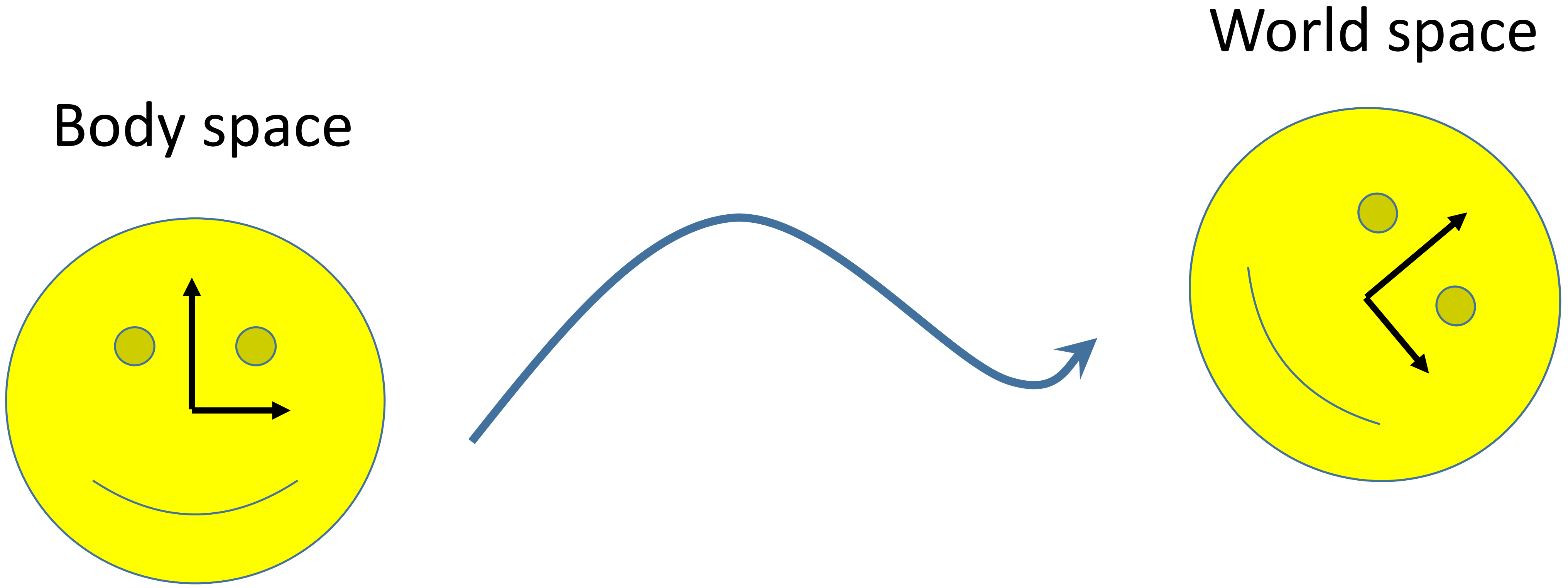
Don’t forget that this is an idealization; on different time scales and/or large enough forces, the deformations may be important and visible!



Rigid body state



Rigid body state



An affine transformation can represent the position and orientation of the body at any time, t.

This lets us find the current world position of any material point, p.

Points on the rigid body

If:

- p_0 is the 3D position of a given material point in body space;
- $c(t)$ is the current 3D position of the centre of mass (in world space);
- $R(t)$ is the current orientation of the body;

…then the current *world position* of the point, $p(t)$ is given by:

$$p(t) = c(t) + R(t)p_0$$

Evolving Position – Particle dynamics again!

Treat the centre of mass of the object as a single point/particle, having the full mass of the rigid body.

Solve the usual equations of motion for position and velocity

$$\frac{d}{dt}\begin{pmatrix} X \\ V \end{pmatrix} = \begin{pmatrix} V \\ F/M \end{pmatrix}$$

with your favourite time integration scheme.

How do we evolve the *orientation*?

Representing Orientation

Choice of representation can be important…

2D: a single angle

3D:

- Euler angles (1 angle per axis). (Suffers from “gimbal lock”.)
- 3x3 rotation matrices, *R*. (We’ll use these for simplicity. 9 numbers for 3 DOFs…)
- Quaternions are also common in practice. (See the Baraff/Witkin notes for more.)

Main question: What are the corresponding equations of motion for orientation? i.e., How does *R* evolve?

Position: $$\frac{d}{dt}\begin{pmatrix} X \\ V \end{pmatrix} = \begin{pmatrix} V \\ F/M \end{pmatrix}$$

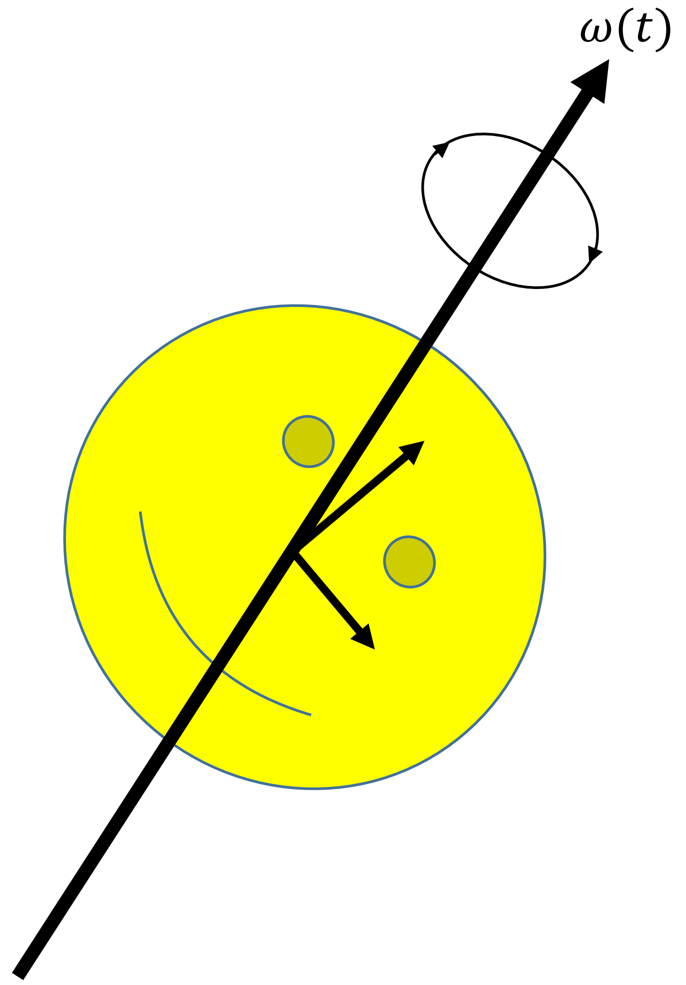
Orientation: $$\frac{d}{dt}\begin{pmatrix} R \\ ? \end{pmatrix} = \begin{pmatrix} ? \\ ? \end{pmatrix}$$

Angular Velocity

We can represent the rotational speed or ***angular velocity*** with a 3D vector, $\omega(t)$.

Direction: axis about which we are rotating.

Magnitude: speed at which we are rotating.



Rate of change of orientation, $\frac{dR}{dt}$

There is a nice relation between angular velocity vector, ω, and rate of change of the orientation/rotation matrix.

$$\frac{dR}{dt} = \begin{pmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{pmatrix} R$$

Denote the (skew-symmetric) matrix above by ω^*.

Then we have: $\frac{dR}{dt} = \omega^* R$

(See Pixar notes for a full derivation, and some intuition.)

So far…

Position update:

$$\frac{dX}{dt} = V$$

Analogous orientation update:

$$\frac{dR}{dt} = \omega^* R$$

Next: we need a rotational analog for velocity update, (i.e., $\frac{dV}{dt} = \frac{F}{m}$)

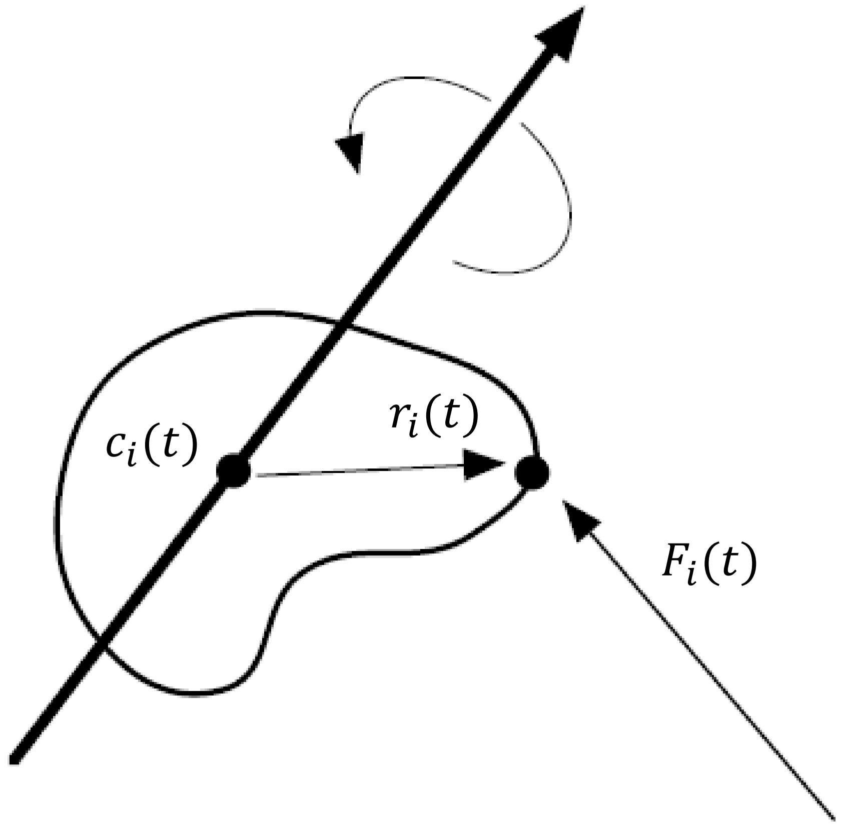
Torque

Torque, τ, is the rotational analog of force.

It is computed as:

$$\tau = (r - c) \times F$$

for a force *F* applied at a point r on the rigid body, where c is the centre of mass.



Angular momentum

Momentum is a useful *conserved* quantity (absent forces).

Linear momentum is $P(t) = mV(t)$.

Since linear velocity/momentum are proportional, and mass is typically constant, we often just work with linear/translational velocity.

However, ***angular velocity is not conserved***, even in the absence of forces! Trickier to work with.

But angular momentum, L, is conserved.

Changes in momentum

Momentum *does* change when there are forces/torques applied.

Linear:

$$m\frac{dV}{dt} = \frac{dP}{dt} = F$$

Angular:

$$\frac{dL}{dt} = \tau$$

Super! Great!

Analogy between position and orientation:

Position variables:

$$\frac{dX}{dt} = V$$

$$\left(m\frac{dV}{dt} = \right)\frac{dP}{dt} = F$$

Orientation variables:

$$\frac{dR}{dt} = \omega^* R$$

$$\frac{dL}{dt} = \tau$$

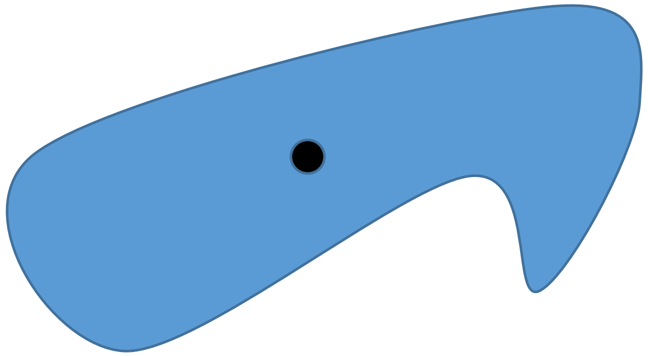
But how are R and L related?

Angular Momentum vs. Angular Velocity

The last piece of the puzzle is converting between angular velocity and angular momentum…

$$L = I\omega$$

I is the ***moment of inertia*** tensor, a 3x3 matrix that characterizes the *distribution* of mass relative to the centre of mass point.



Inertia Tensor I

The rotational counterpart of mass. (Compare $L = I\omega$ vs. $\mathrm{P} = mV$)

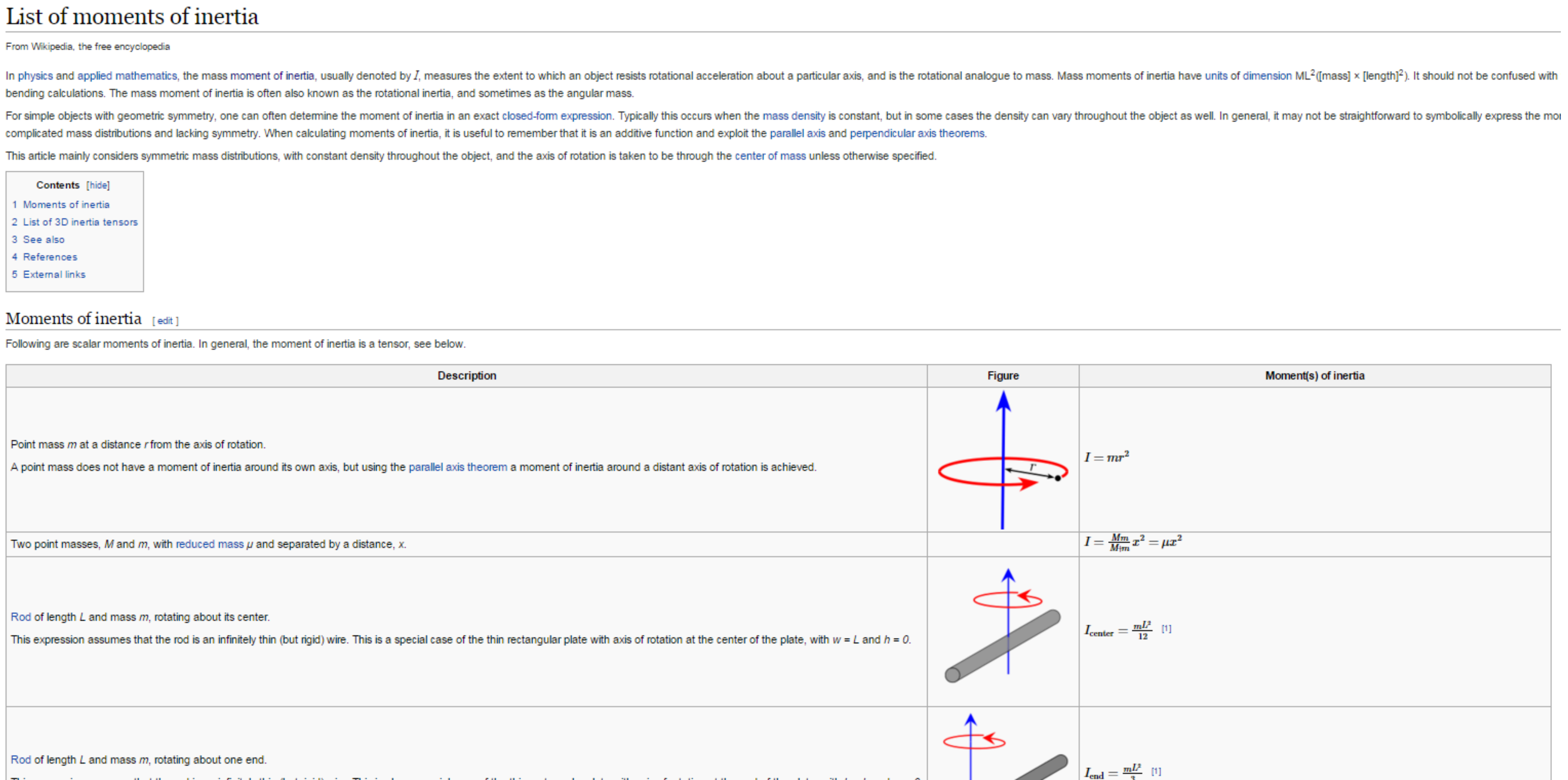
Recall: Mass of the rigid body is the integral of density $m = \int \rho \, dV$ over the object.

Moment of inertia is also an integral:

$$I = \int \rho \begin{pmatrix} r_y^2 + r_z^2 & r_x r_y & r_x r_z \\ r_x r_y & r_x^2 + r_z^2 & r_y r_z \\ r_x r_z & r_y r_z & r_x^2 + r_y^2 \end{pmatrix} dV$$

Finding Inertia Tensors

For many common shapes, simple formulas exist.
For complex shapes, you typically have to approximate the integral numerically.



List of moments of inertia

From Wikipedia, the free encyclopedia

In physics and applied mathematics, the mass moment of inertia, usually denoted by I, measures the extent to which an object resists rotational acceleration about a particular axis, and is the rotational analogue to mass. Mass moments of inertia have units of dimension ML^2([mass] × [length]^2). It should not be confused with bending calculations. The mass moment of inertia is often also known as the rotational inertia, and sometimes as the angular mass.

For simple objects with geometric symmetry, one can often determine the moment of inertia in an exact closed-form expression. Typically this occurs when the mass density is constant, but in some cases the density can vary throughout the object as well. In general, it may not be straightforward to symbolically express the mo complicated mass distributions and lacking symmetry. When calculating moments of inertia, it is useful to remember that it is an additive function and exploit the parallel axis and perpendicular axis theorems.

This article mainly considers symmetric mass distributions, with constant density throughout the object, and the axis of rotation is taken to be through the center of mass unless otherwise specified.

Moments of inertia [edit]

Following are scalar moments of inertia. In general, the moment of inertia is a tensor, see below.

Description	Figure	Moment(s) of inertia
Point mass m at a distance r from the axis of rotation. A point mass does not have a moment of inertia around its own axis, but using the parallel axis theorem a moment of inertia around a distant axis of rotation is achieved.		$I = mr^2$
Two point masses, M and m, with reduced mass μ and separated by a distance, x.		$I = \frac{Mm}{M+m}x^2 = \mu x^2$
Rod of length L and mass m, rotating about its center. This expression assumes that the rod is an infinitely thin (but rigid) wire. This is a special case of the thin rectangular plate with axis of rotation at the center of the plate, with $w = L$ and $h = 0$.		$I_{\text{center}} = \frac{mL^2}{12}$ [1]
Rod of length L and mass m, rotating about one end.		$I_{\text{end}} = \frac{mL^2}{3}$ [1]

Inertia Tensor I

Subtlety: unlike mass, the current inertia tensor *I(t)* **in world space** changes over time!

Re-computing (the integral) for $I(t)$ at each step could be costly…

Fortunately, it can be shown that:

$$I(t) = R(t) I_{body} R(t)^T$$

(See Baraff/Witkin’s Pixar notes for a full derivation.)

Rotational Equations of Motion

We now have everything we need to evolve rotation, $\frac{d}{dt}\begin{pmatrix} R \\ ? \end{pmatrix} = \begin{pmatrix} ? \\ ? \end{pmatrix}$:

$$\frac{d}{dt}\begin{pmatrix} R \\ L \end{pmatrix} = \begin{pmatrix} \omega^* R \\ \tau \end{pmatrix}$$

Note: At each step we will need to recover the angular velocity ω from momentum L using:

$$I(t) = R(t) I_{body} R(t)^T \text{ and } \omega = I(t)^{-1} L(t)$$

Analogy between position and orientation:

Position variables:

$$\frac{dX}{dt} = V$$

$$\left(m\frac{dV}{dt} =\right)\frac{dP}{dt} = F$$

$$P = mV$$

Orientation variables:

$$\frac{dR}{dt} = \omega^{*}R$$

$$\frac{dL}{dt} = \tau$$

$$L = I\omega$$

Time integration for rigid bodies

Apply your favourite time integration scheme to *discretize* this full set of equations. (Forward Euler, midpoint, etc.)

Collisions

Two main families.

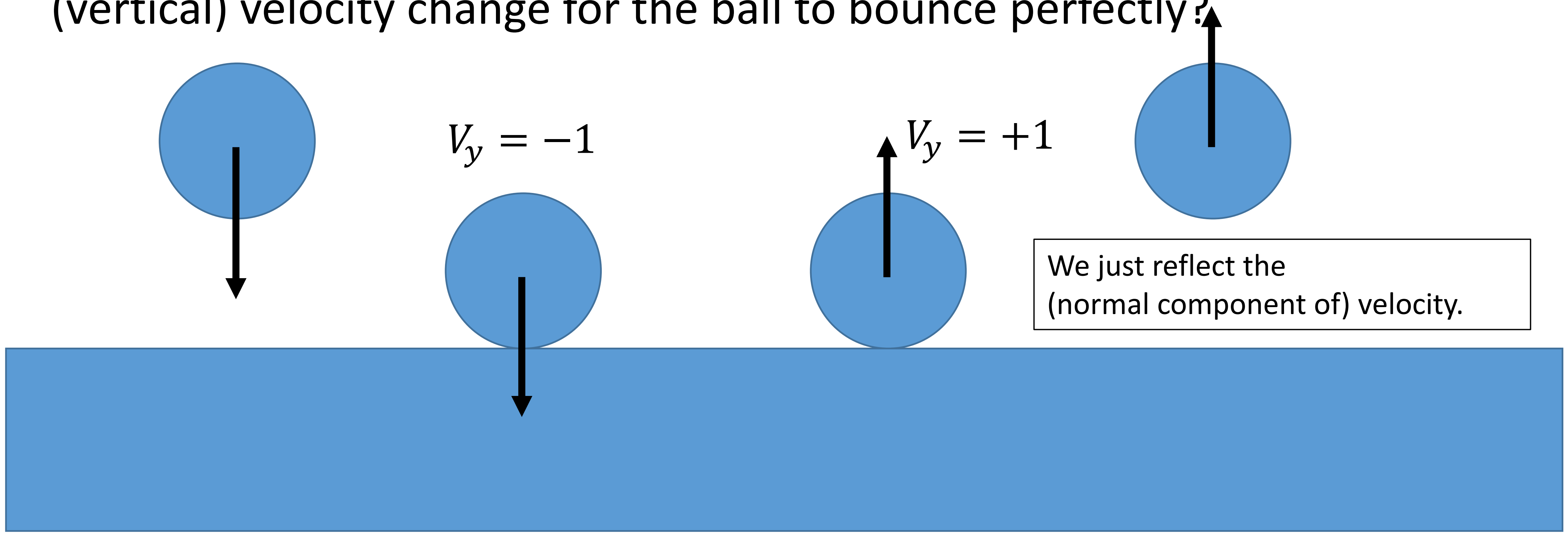
- Penalty/repulsion methods (briefly, last time)
- Impulse methods (briefly, this time)

Penalty method applies force over some finite period of contact, Δt.

Impulses are an instantaneous application of force over an infinitesimal period, i.e., conceptual limit as $\Delta t \to 0$.

Impulses – *Elastic* Collisions

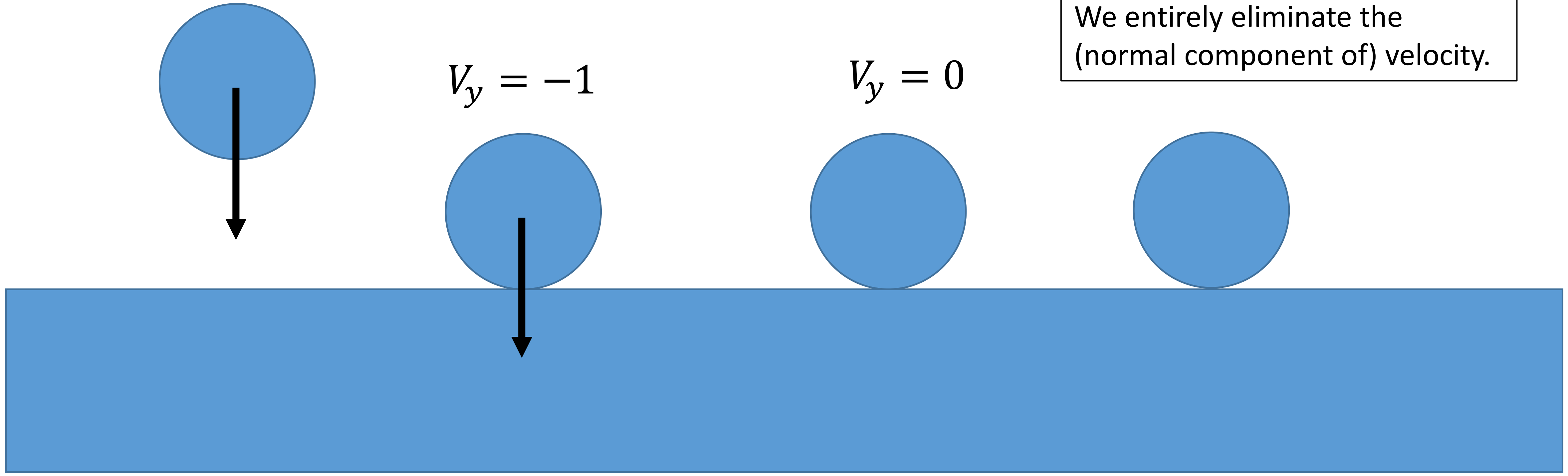
Consider an *ideal rubber ball* falling onto a flat plane. How must the (vertical) velocity change for the ball to bounce perfectly?



We just reflect the (normal component of) velocity.

Impulses – *Inelastic* Collisions

Consider a *heavy stone* falling onto a flat plane. Nearly all of the energy is dissipated in the collision.



We entirely eliminate the (normal component of) velocity.

Coefficient of Restitution

Most collisions are somewhere between.

Coefficient of restitution, $\epsilon \in [0,1]$, models the fraction of energy remaining after a collision. Putting it together…

$$V_{rel,n}^{after} = -\epsilon V_{rel,n}^{before}$$

Note! Only the (relative) velocity in the normal direction changes.

Tangential velocity is left alone (until you add friction…)

See: e.g. “Nonconvex Rigid Bodies with Stacking” [Guendelman 2003] for a summary of the general case, with friction and rotational collision impulses.

Some Interesting Recent Examples

- Drawn from [Kaufman et al. 2008] and [Smith et al. 2012], on our papers list.

Catenary Arch

$\mu = 0.6$

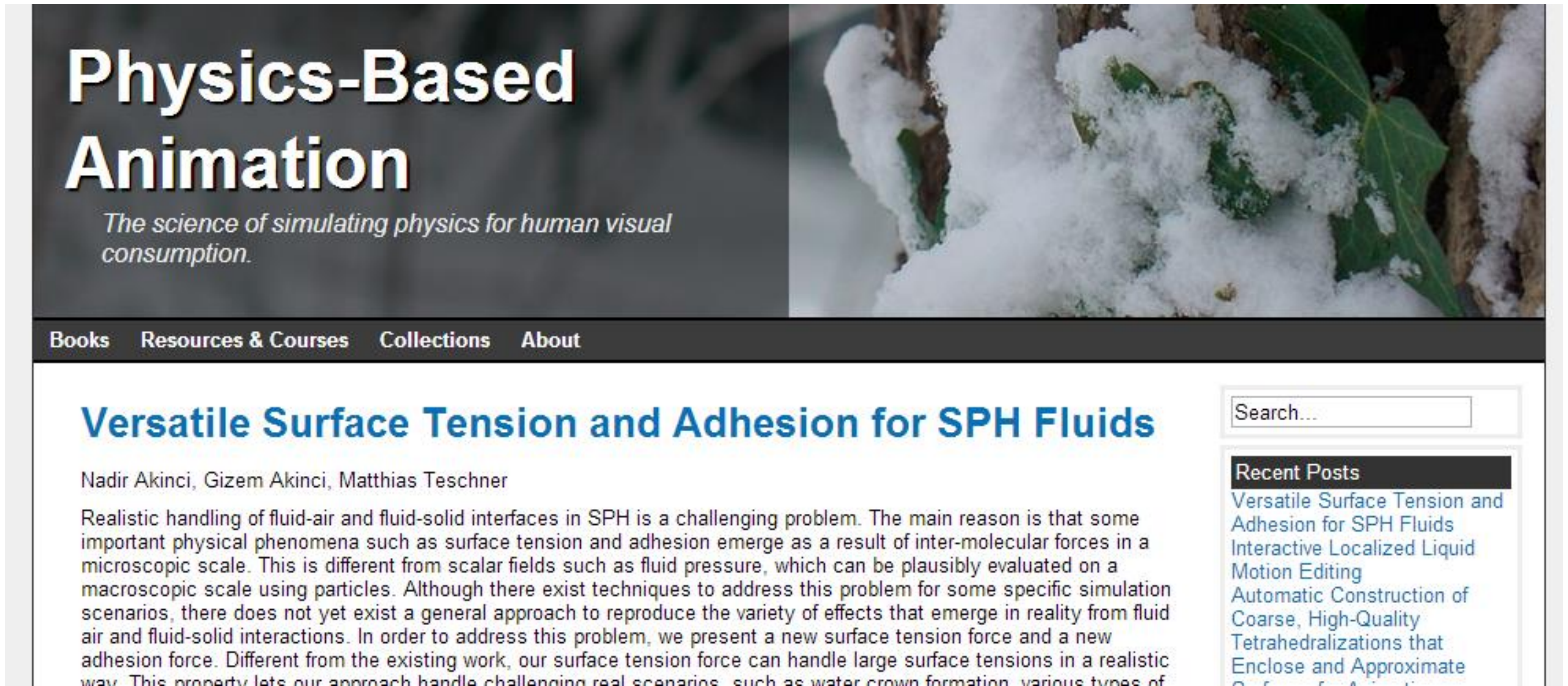
Decomposition: Rigid Bunny Drop

Scalability,
1,002,001 Balls,
Random Initial Velocities

Physics-Based Animation

If you're interested in following along with recent/new research papers, check out:

www.physicsbasedanimation.com



Physics-Based Animation

The science of simulating physics for human visual consumption.

Versatile Surface Tension and Adhesion for SPH Fluids

Nadir Akinci, Gizem Akinci, Matthias Teschner

Realistic handling of fluid-air and fluid-solid interfaces in SPH is a challenging problem. The main reason is that some important physical phenomena such as surface tension and adhesion emerge as a result of inter-molecular forces in a microscopic scale. This is different from scalar fields such as fluid pressure, which can be plausibly evaluated on a macroscopic scale using particles. Although there exist techniques to address this problem for some specific simulation scenarios, there does not yet exist a general approach to reproduce the variety of effects that emerge in reality from fluid air and fluid-solid interactions. In order to address this problem, we present a new surface tension force and a new adhesion force. Different from the existing work, our surface tension force can handle large surface tensions in a realistic way. This property lets our approach handle challenging real scenarios, such as water crown formation, various types of

Recent Posts

Versatile Surface Tension and Adhesion for SPH Fluids

Interactive Localized Liquid Motion Editing

Automatic Construction of Coarse, High-Quality Tetrahedralizations that Enclose and Approximate