Lecture 2: More Time Integration and Some Rigid Bodies

Jan 9, 2014

Today’s Plan

- A few more details on presentations/discussions
- Wrap up some slides from last time
- More on time integration
- Intro to rigid bodies

Presentation scheduling

First pair of presenters and papers for Thurs, Jan 16:

- Dominik: “Non-convex rigid bodies with stacking”
- Li: TBD

Remainder still to be determined. I’ve heard from 6 of you so far.

If you send me a draft of your slides 24-48 hours prior, I will try to provide some suggestions.

Presentations

A typical format is…

- Motivate the topic/problem
- Briefly outline key related work
- Describe the main contributions of the paper
- Show and discuss results
    - e.g. animations, graphs, comparisons to theory or experiment, etc.
- Provide a critique of the paper (both good and bad)
- Conclude briefly

Presentations - Tips

- Don’t explain every tiny detail – focus on core/novel contributions
- Prefer diagrams and images (and your voice) over lots of tiny text
- Avoid overwhelming with equations
- Talk to the audience, not the slides
- Do not recycle the authors slides. (Figures, graphs, results are fine.)
- Practice!

Analyzing a paper

Imagine you are a reviewer deciding whether to accept or reject…

Questions to ask yourself:

- Did the authors clearly motivate the problem?
- Are the contributions truly novel wrt. previous work?
- How *substantial* are the contributions?
- Why did the authors make the [technical/design/theoretical] choices they did?
- Do the results actually achieve/support what was claimed?
- Are the writing and figures clear?

Reviews

See SIGGRAPH review format here:
http://s2013.siggraph.org/submitters/technical-papers/review-form
(Skip “reviewer expertise” and “private comments”)

Length: Aim for a few paragraphs.

MIT’s Fredo Durand has some tips for reviewing papers:
- http://people.csail.mit.edu/fredo/review.pdf

Keshav offers some great strategies for reading papers (and more good references):
http://blizzard.cs.uwaterloo.ca/keshav/home/Papers/data/07/paper-reading.pdf

Discussions

- Everyone should read the papers and bring comments/questions/critiques.
- Bring a PDF or print-out to refer to.
- This is a good chance to dive further into technical details, things that remain unclear, debate the merit of the work, etc.
- Presenter can guide the discussion, but need not have *all* the answers.

Picking up from 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.

Forces

What physical forces might we use to drive a particle system?

- Gravity
- Wind
- Springs / Elasticity
- Damping / Viscosity
- Friction
- Collisions/Contact
- …

Given the set of forces $F_1, F_2, \dots, F_n$, sum to get net force on a particle.

Forces: Damping/Viscosity

Often helpful to gradually slow the motion, avoid oscillating forever.

Approximates internal friction or energy dissipation.

Typically opposes the motion of the particle(s), and is proportional to the velocity. For example…

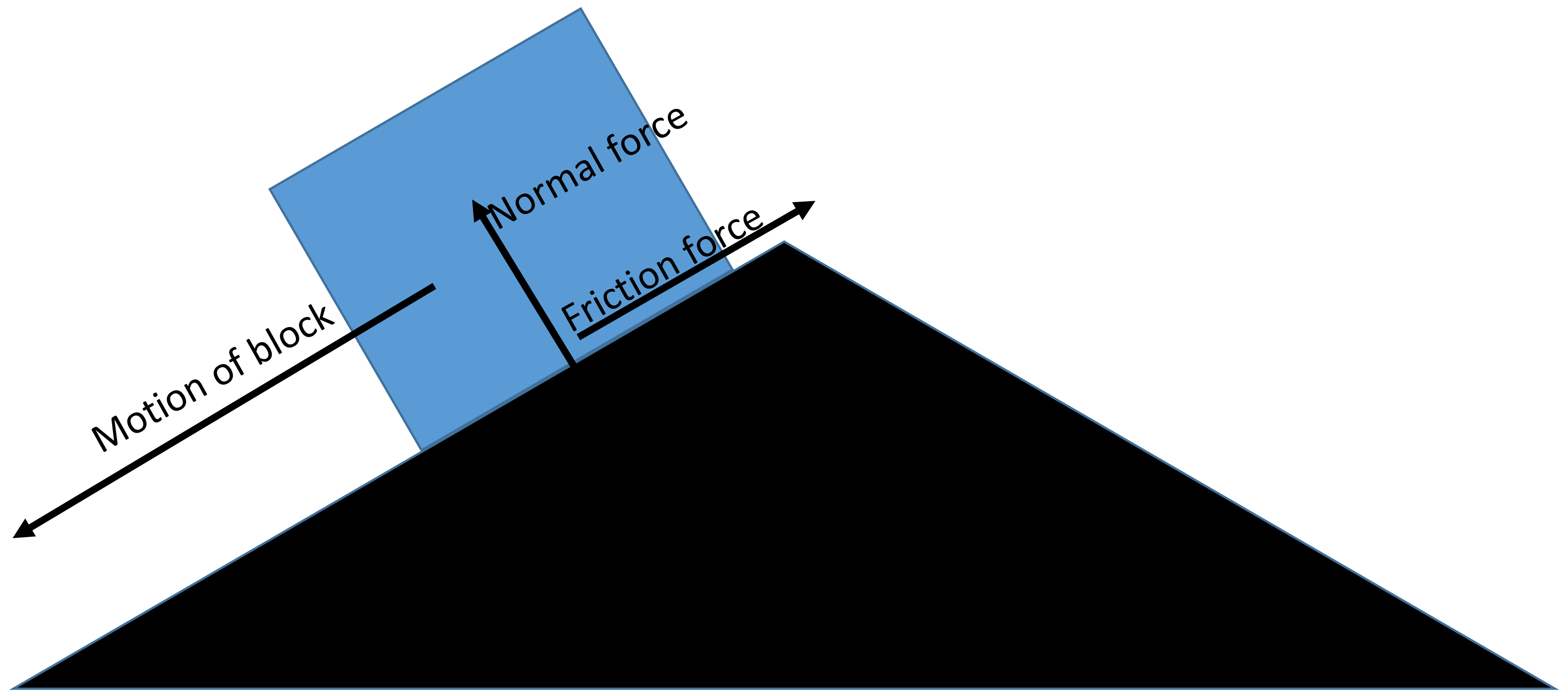
$$F = -k_d V$$

Note: Too much can look unrealistic/gooey.

Forces: Normal & Friction (Contact forces)

Normal force: Resists interpenetration

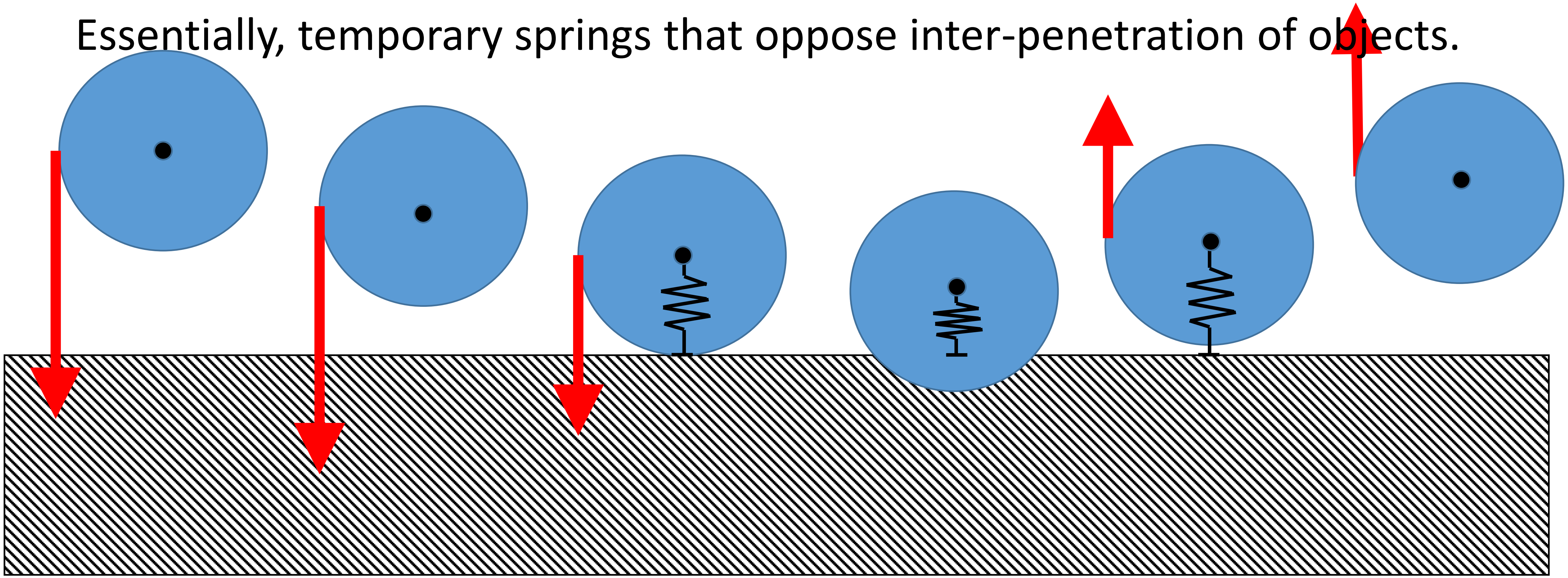
Friction force: Resists relative sliding between materials.



Forces: Collisions

One strategy for modeling collisions is *penalty/repulsion forces*.

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



Time Integration, Revisited

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:

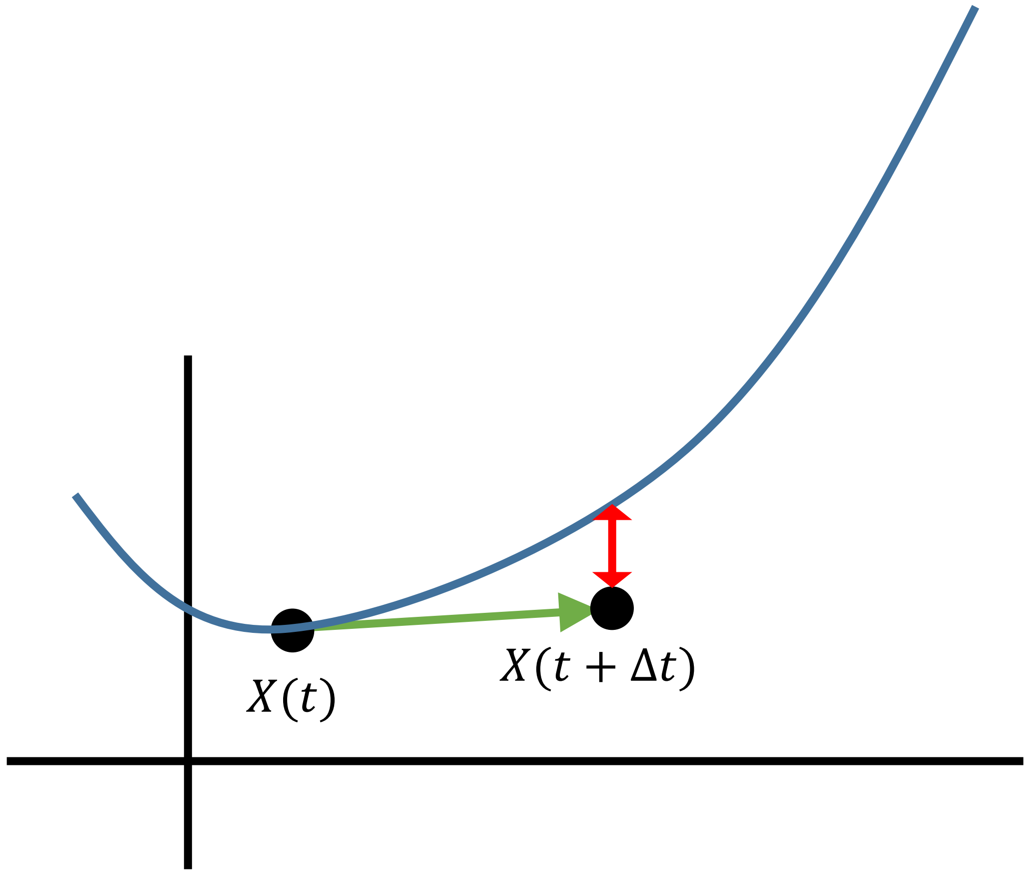
- A brief look at Forward Euler error and stability
- *Implicit* integration basics

Error of Forward Euler

For dx/dt = V, forward Euler is:

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

How can we determine the error made in ***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…

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

This is just velocity, V.

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:

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

Therefore the $\mathrm{O}(\Delta t^2)$ term dominates the error.

What about accumulated error?

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 $n = \frac{t^n - t^0}{\Delta t} = O(\frac{1}{\Delta t})$ steps.

The global error is thus $O(\Delta t)$, and we say Forward Euler is a 1st order accurate method.

- A more precise argument depends on properties of V itself.

Similarly, the midpoint method (RK2) is 2nd order.

Stability of Forward Euler

We saw that 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 linear *test equation*: $\frac{dx}{dt} = \lambda x$

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

Behavior after many steps?

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

Stability of Forward Euler

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

When does it blow up (grow)? When does it decay?

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

This can be shown to imply: $0 < \Delta t < \frac{2}{|\lambda|}$

Larger systems

Larger, possibly nonlinear systems can be locally approximated by a linear differential equation…

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

…where *A* is now a matrix, *x* is a vector. (e.g. vector fields, springs)

How to analyze stability?

Matrix Eigenvalues

Recall: *eigenvalues* characterize how a matrix acts on corresponding *eigenvectors*.

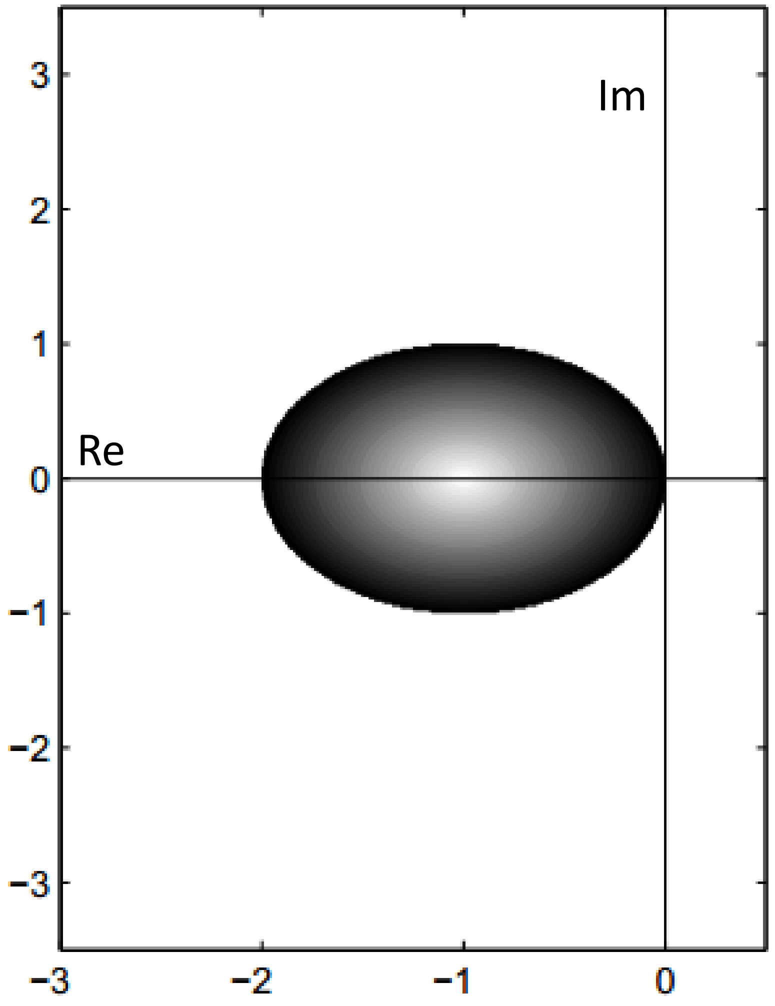
$$Av = \lambda v$$

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

Stability Regions

Eigenvalues of a problem may even be complex…

We can visualize the absolute *region of stability* on the *complex plane*, by plotting whether stability is satisfied for different values of the product $\Delta t \lambda$.



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

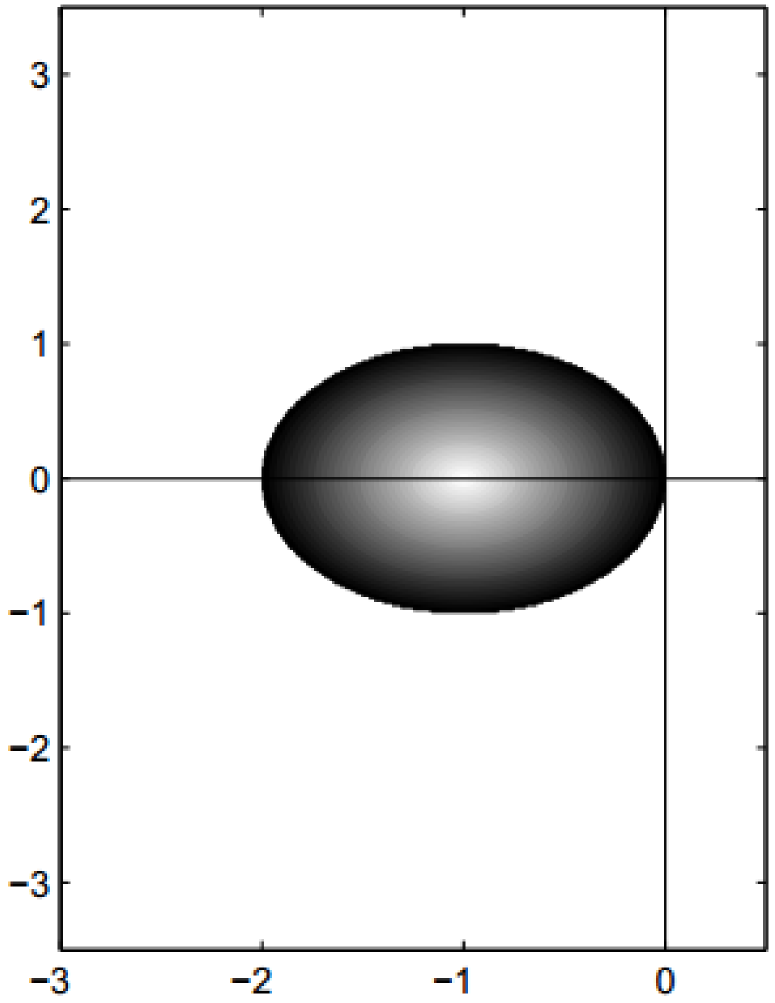
Impact of Stability

Rotational/oscillatory motion corresponds to *imaginary* eigenvalues.

Often the stability restriction can be very (or impossibly!) limiting.

e.g. Rotational velocity fields, springs, etc.

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



Imaginary Eigenvalues

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

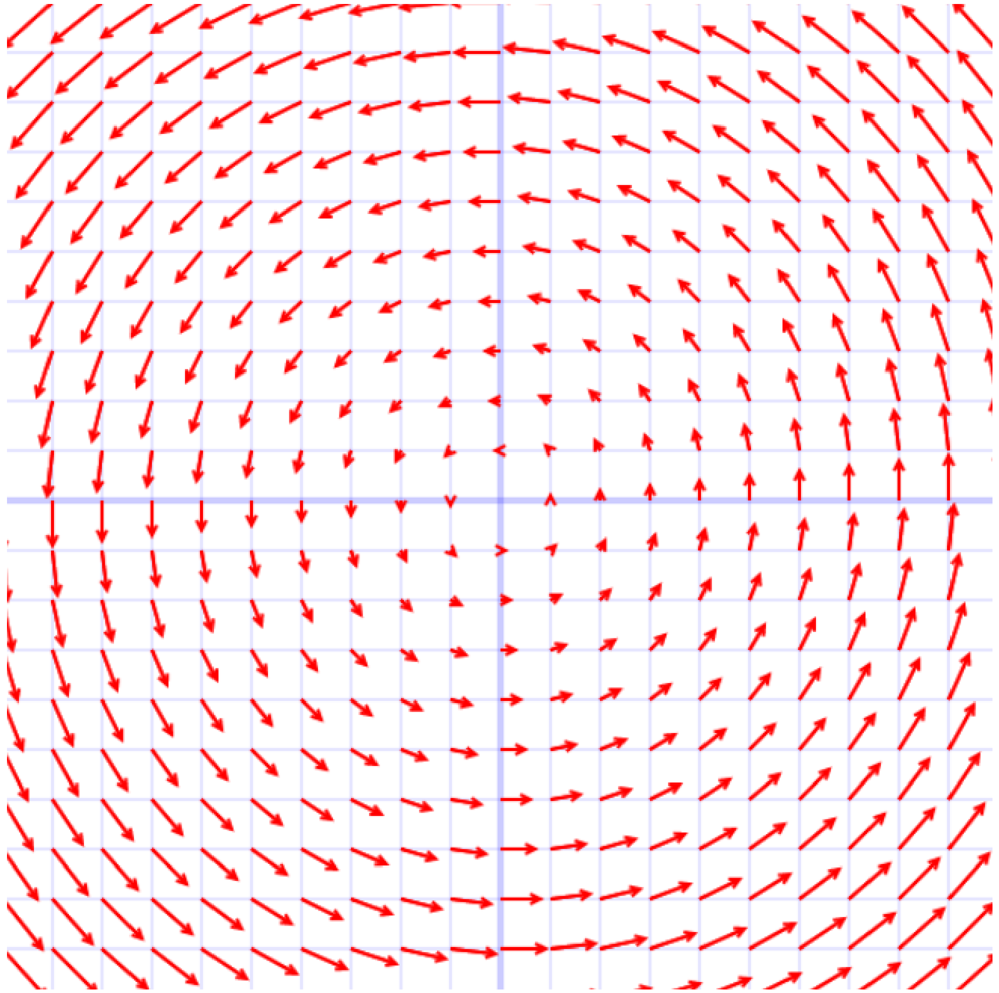
Then…

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

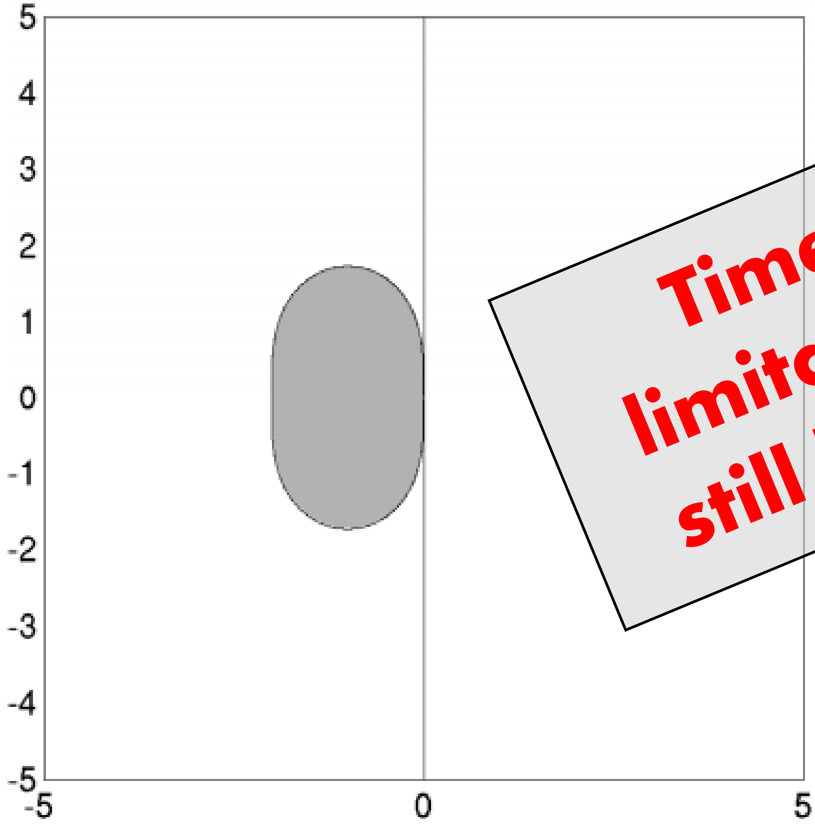
$A = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$ has strictly imaginary eigenvalues, $\lambda = \pm i$.

Forward Euler is *never* stable here; stability region doesn’t include these points, which lie on the (vertical) imaginary axis.

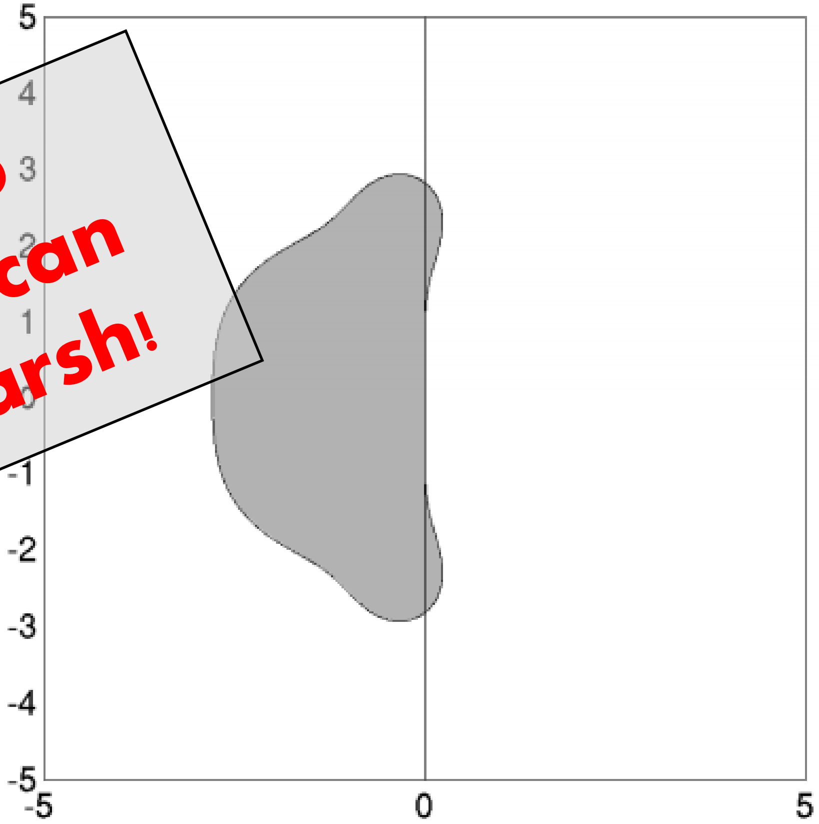
Try a 1D spring, also.



Stability Regions for other methods



Midpoint (AKA Runge Kutta 2)



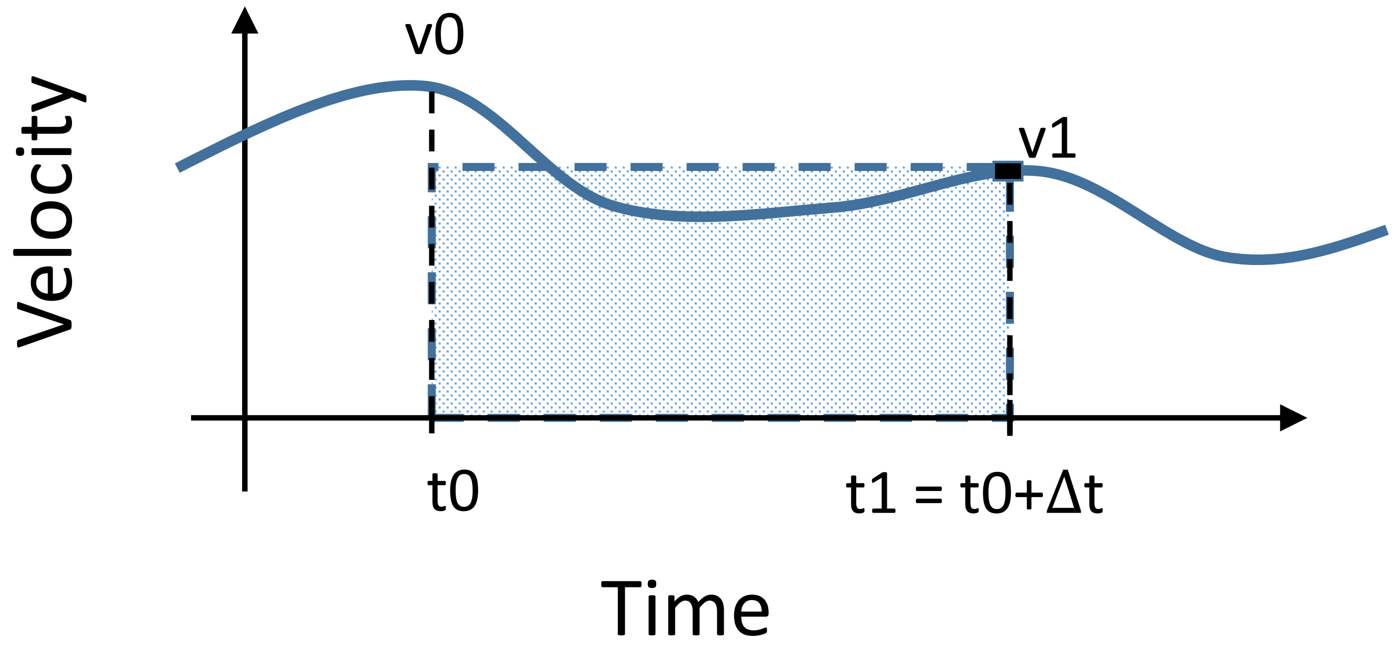
Runge Kutta 4
(A similar 4th order scheme)

Time step limitation can still be harsh!

Implicit (or backward) Euler

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

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



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 did know 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: $\frac{dx}{dt} = \lambda x$

Backward Euler gives:

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

This is an *implicit* equation for $X(t + \Delta t)$.

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

For a single step:

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

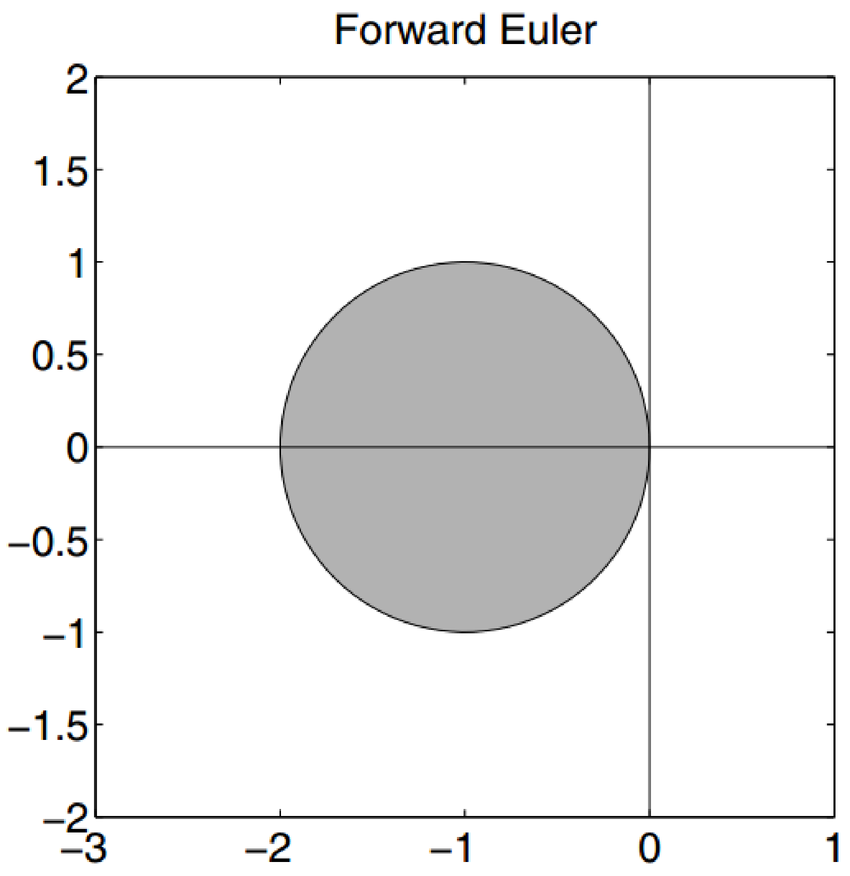
So for step *n*, we have:

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

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

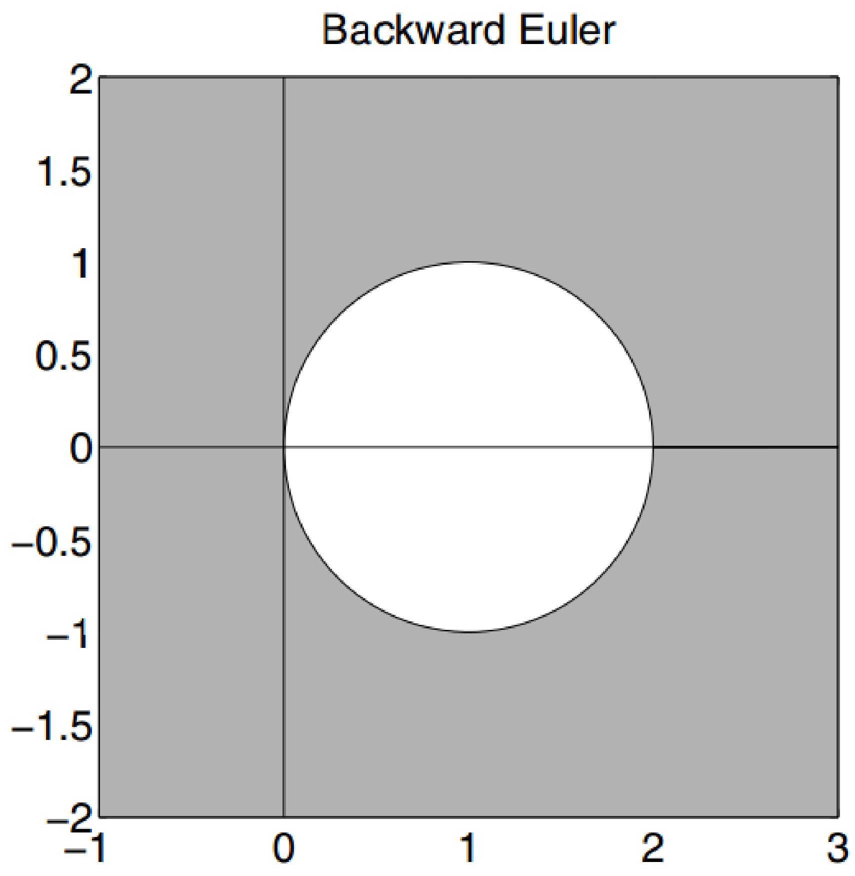
$$|1 - \lambda\Delta t| \geq 1$$

Stability Regions – Explicit vs. Implicit Euler



(a)

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



(b)

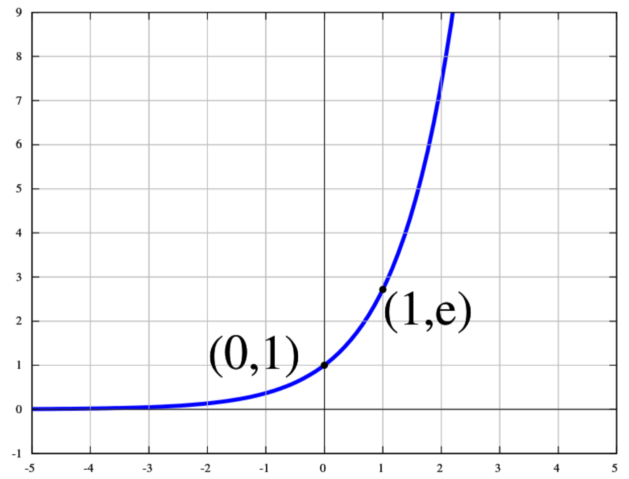
$$|1 - \lambda \Delta t| \geq 1$$

Implicit Euler

Can sometimes behave stably even when the “physics”/problem is not!

e.g.,

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



Solution is e^t, which grows exponentially.

(Imagine a physical system that gains speed exponentially.)

This corresponds to the right half of the complex plane.

BE gives:

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

If you take large steps, the numerical solution will decay in magnitude, which doesn’t match the true solution!

Implicit Euler

In practice:

- BE is stable for large time steps.
- The price is overly *damped* motion (rapid loss of energy).

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

This system 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

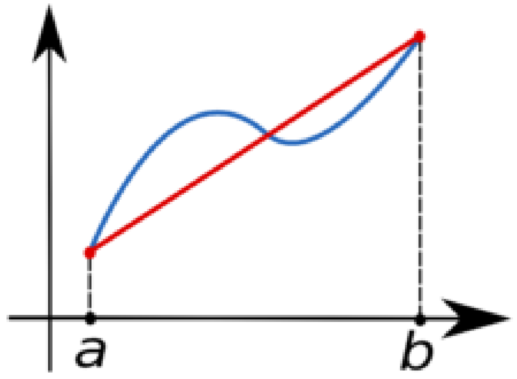
Other schemes mix explicit and implicit integration.

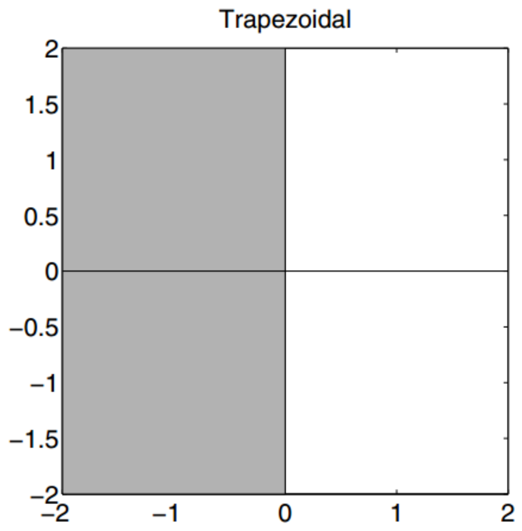
Trapezoidal rule is half a step of FE, half a step of BE!

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

Stable when the "physics" is.

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





Caveat…

Technically, these stability analyses only apply to linear problems.

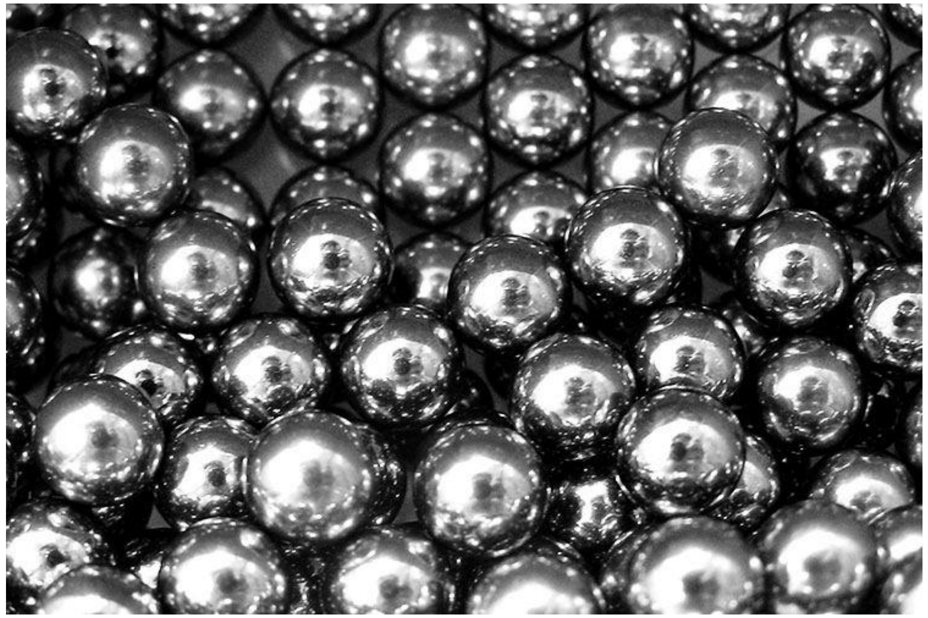
But, they are often indicative of behaviour for nonlinear problems too.

Rigid Bodies

Rigid Bodies

For modeling *very* stiff objects, a mass-spring/deformable system is problematic… Why?

- Explicit methods, will need very small time steps.
- Implicit methods, equations become ill-conditioned (hard to solve)
- Deformation happens fast, but is not the (visually) interesting part!



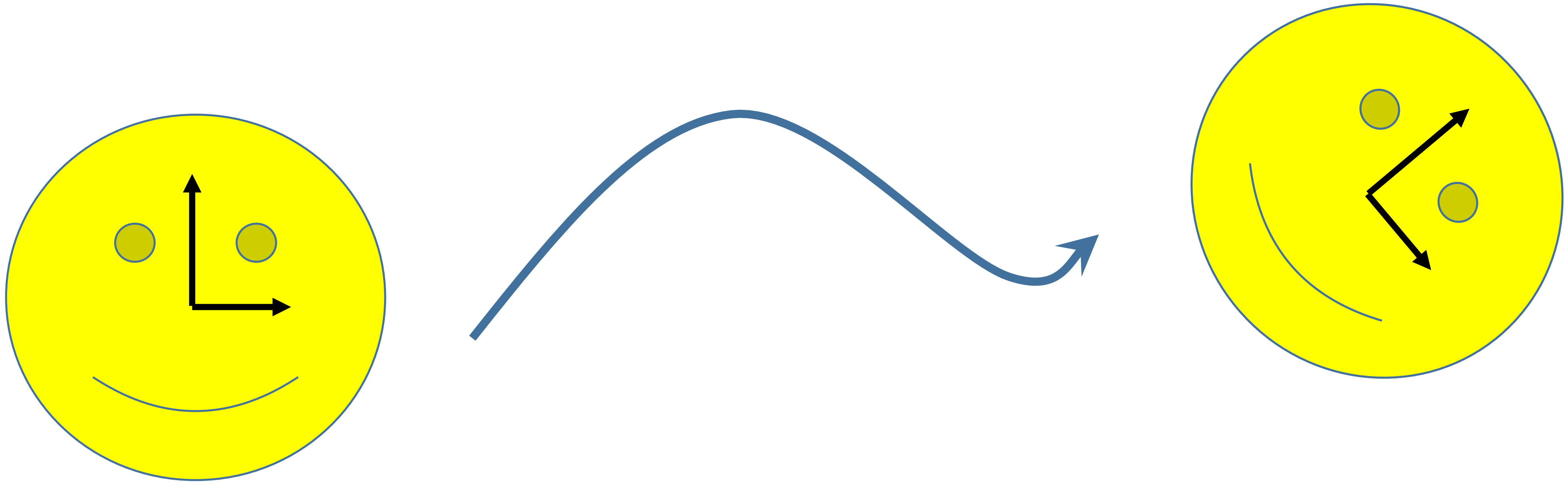
Abstraction

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

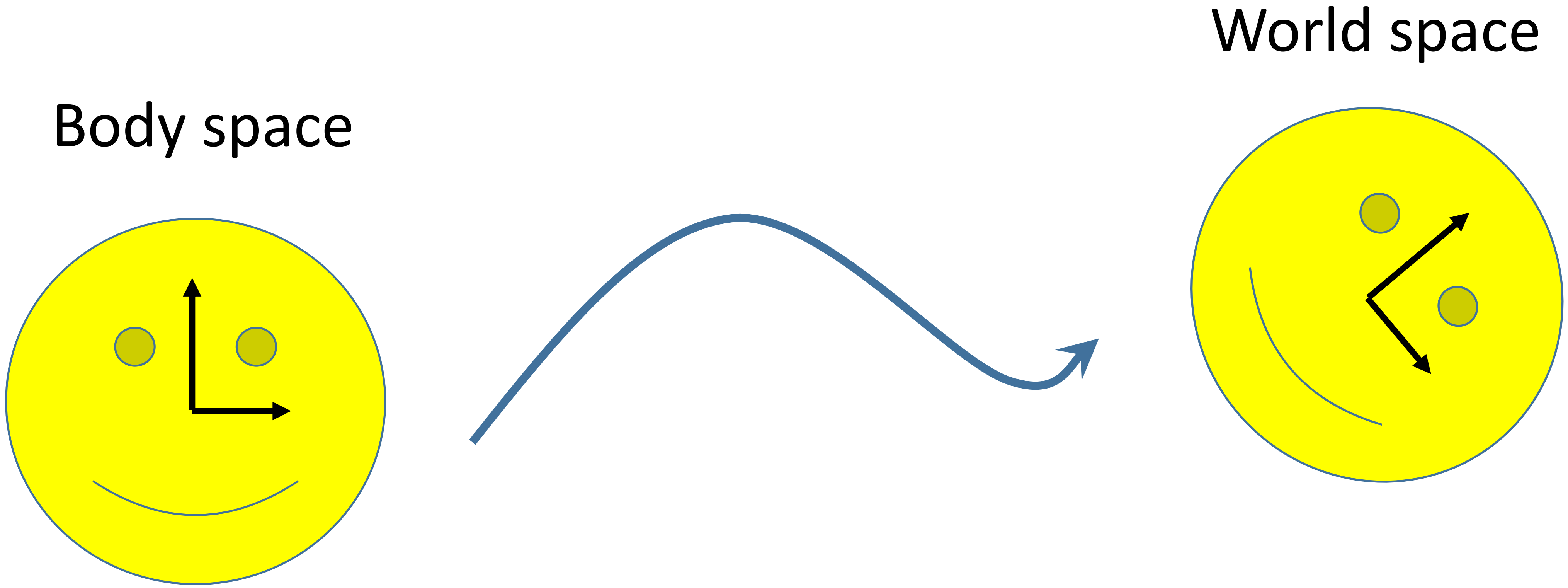
Assume that there is **no** deformation at all, i.e., perfectly rigid. What does this buy us?

We can represent the state of the rigid body with a single position **and** orientation, and only evolve these quantities.

Rigid body state



Rigid body state



We can use a standard affine transformation to represent the position and orientation of the body at any time.
For any point on the body, we can find its current position in world space.

Points on the rigid body

If:

- p_0 is the 3D position of a point in body space
- centre of mass is the 3D position vector, c(t)
- orientation is represented by a rotation matrix, R(t)

Then current *world position* of the particle, p(t) is:

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

Evolving Position – Easy!

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:

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

How do we evolve the *orientation*?

Representing Orientation

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, but suffer from drift).
- Quaternions are 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: $$\frac{d}{dt}\begin{pmatrix} X \\ V \end{pmatrix} = \begin{pmatrix} V \\ F/M \end{pmatrix}$$

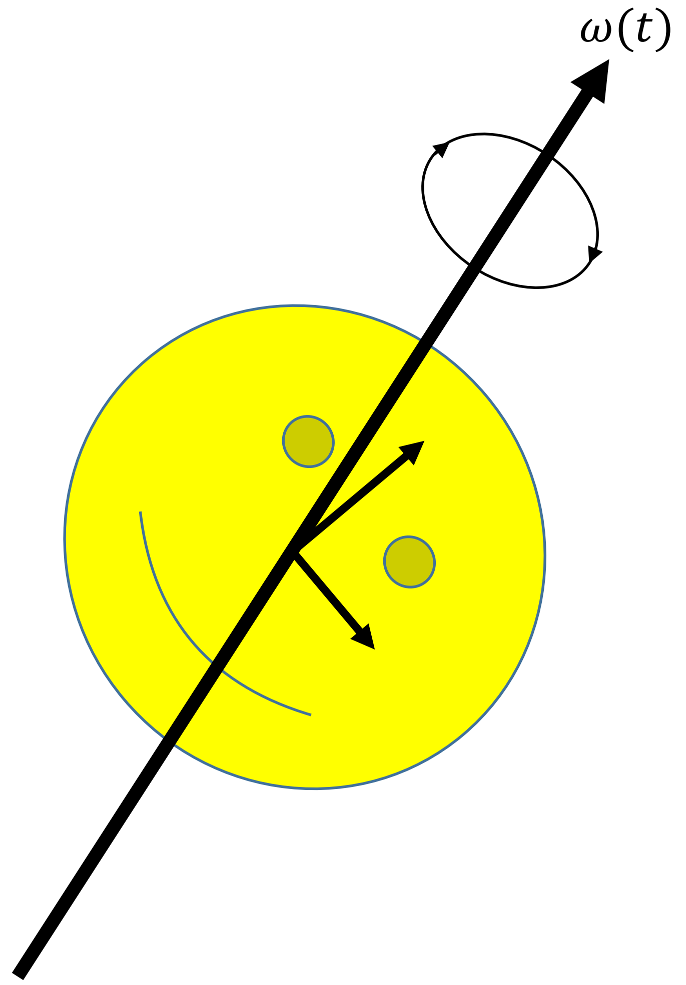
Orientation: $$\frac{d}{dt}\begin{pmatrix} R \\ ? \end{pmatrix} = \begin{pmatrix} ? \\ ? \end{pmatrix}$$

Angular Velocity

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

Direction: the axis about which we are rotating.

Magnitude: the speed at which we are rotating.



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

There is a (somewhat) convenient relation between angular velocity ω and rate of change of the rotation matrix.

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

Denote the skew-symmetric matrix above by ω^*.
Then we have: $\dot{R} = \omega^* R$

(See Pixar notes for derivation/intuition.)

Recap so far…

Position update:

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

Corresponding orientation update:

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

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

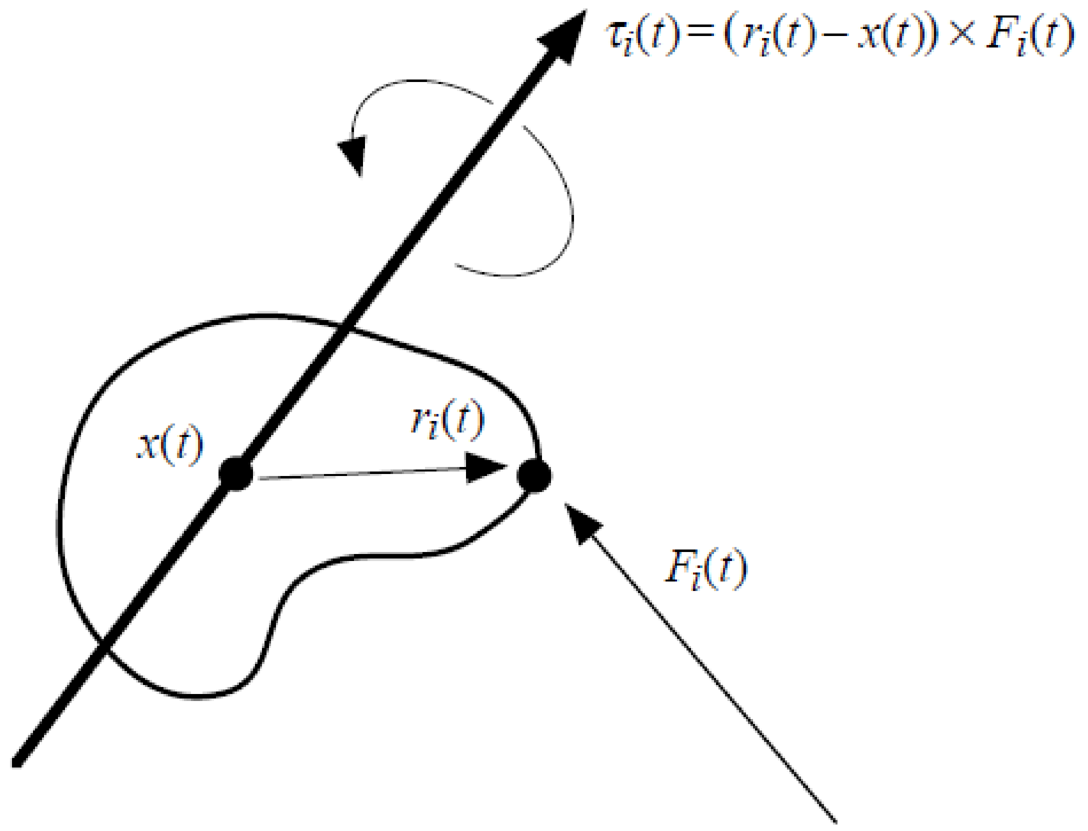
Torque

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.

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

Angular velocity is not conserved, in the absence of forces!

But angular momentum, L, is.

Changes in momentum

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

Linear:

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

Angular:

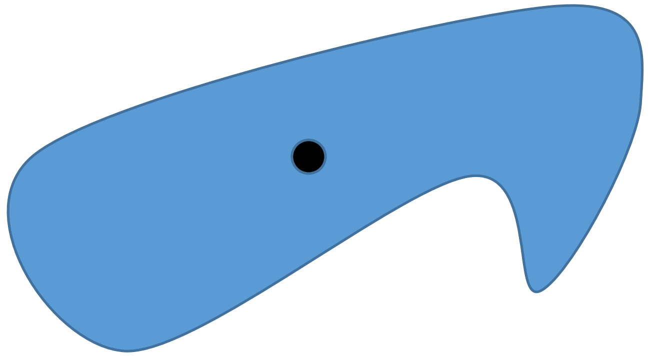
$$\frac{dL}{dt} = \tau$$

Angular momentum vs. Angular Velocity

The last piece of the puzzle is converting from angular velocity to 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. $\mathrm{P} = mV$)

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

Moment of inertia is also an integral:

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

For many common shapes, simple formulas exist.

Inertia Tensor *I*

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

Re-computing at each step would be costly…

Fortunately, it can be shown that:

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

(See Pixar notes for a derivation.)

Rotational Equations of Motion

We now have everything we need to fill in: $\frac{d}{dt}\binom{R}{?} = \binom{?}{?}$

$$\frac{d}{dt}\binom{R}{L} = \binom{\omega^* R}{\tau}$$

Notice 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

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

Try writing out the steps for forward Euler on a rigid body.

Collisions

Two main families, typically.

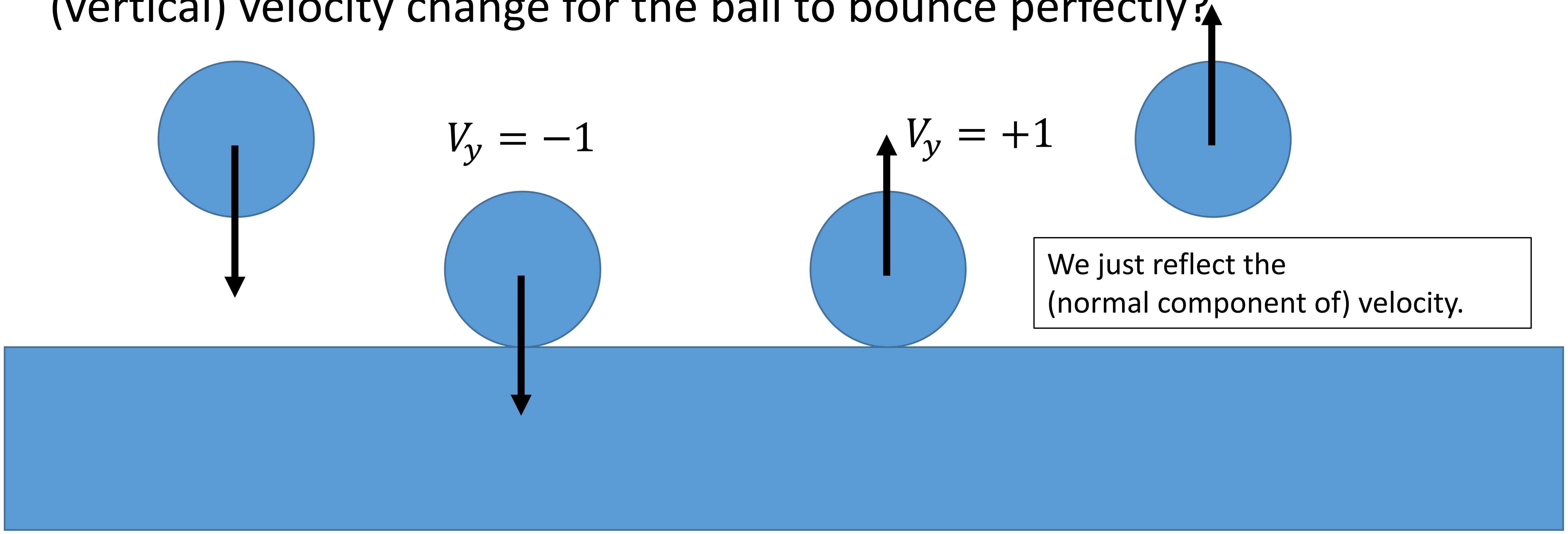
- Penalty/repulsion methods (briefly, last time)
- Impulse methods (briefly, this time)

Penalty method applies force over some 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 a *rubber ball* falling onto a ground plane. How must the (vertical) velocity change for the ball to bounce perfectly?

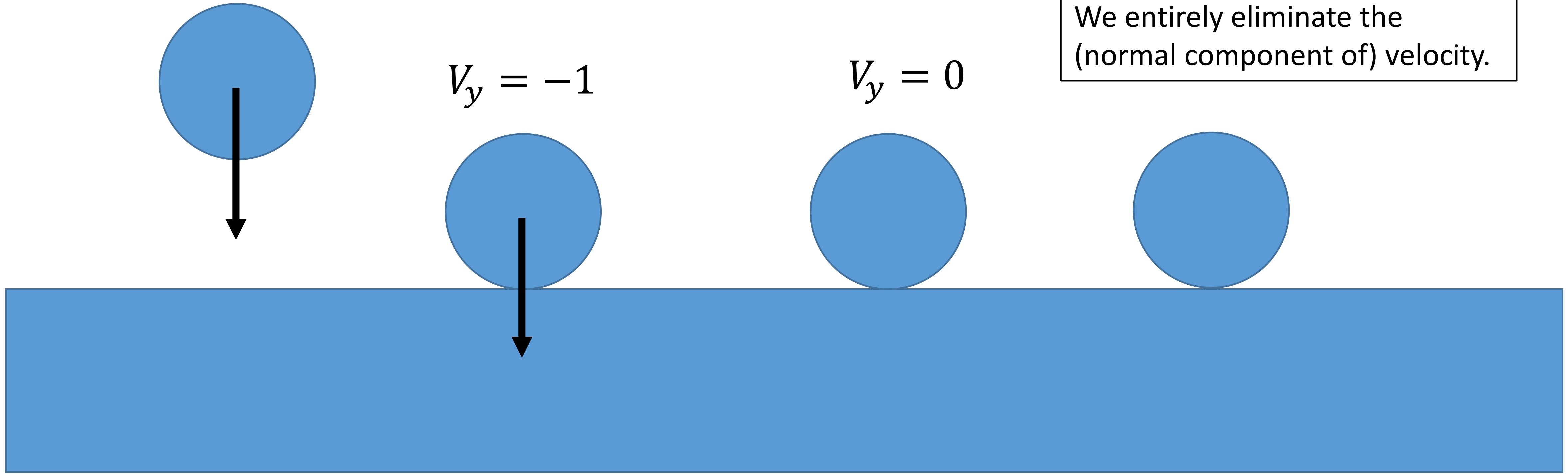


We just reflect the (normal component of) velocity.

Impulses – *Inelastic* Collisions

Consider a *heavy stone* falling onto a ground 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 (unless you add friction…)

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

Some Recent Examples

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

Catenary Arch

$\mu = 0.6$

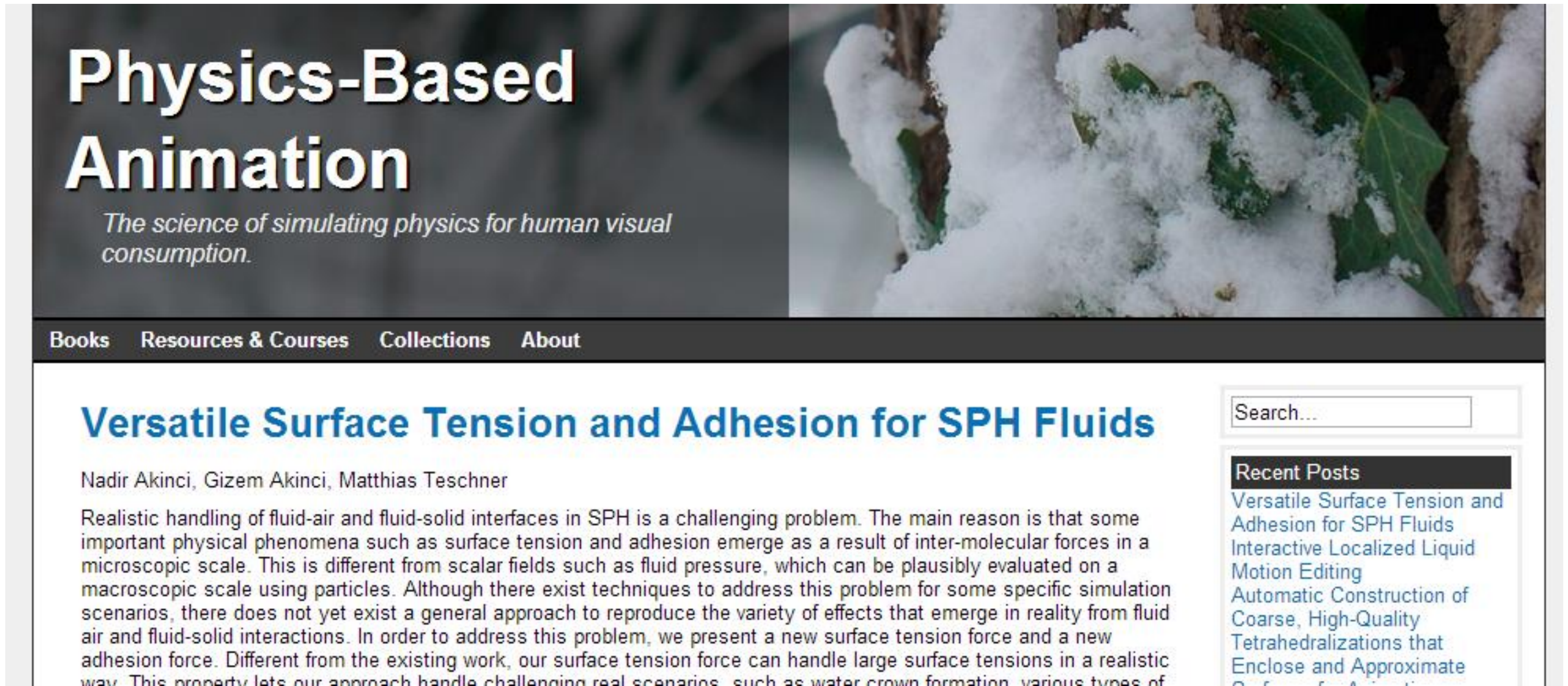
Decomposition: Rigid Bunny Drop

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

Physics-Based Animation

If you're interested in following recent/new research, 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

Where did that come from?

Consider how an arbitrary vector $r(t)$ is affected by the rotation.

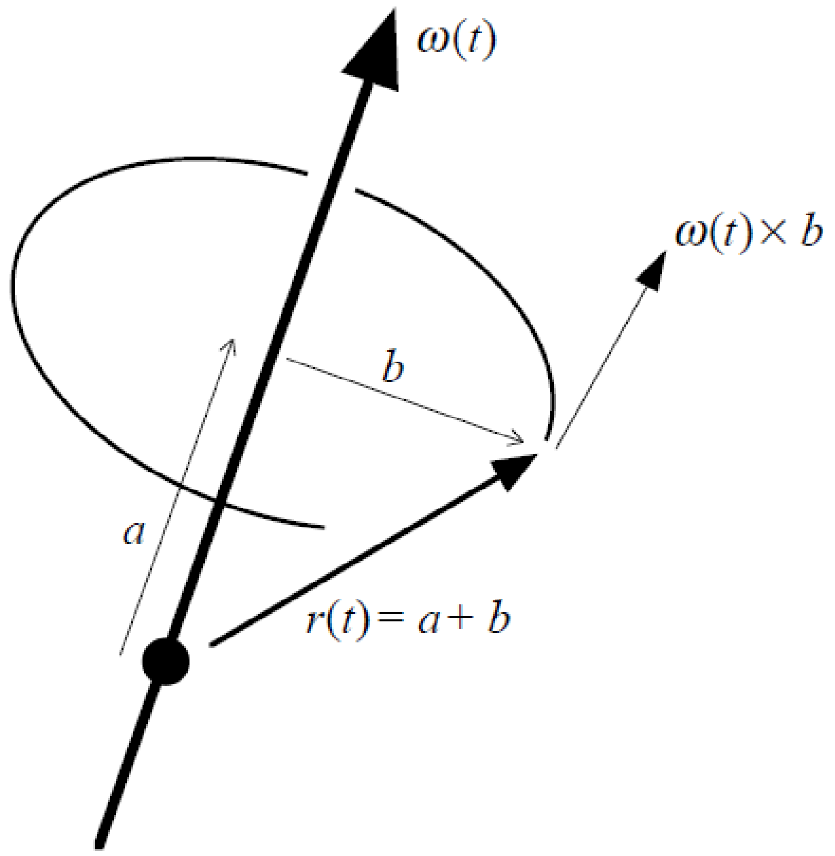
Split $r(t)$ into perpendicular $\perp$ and parallel $\parallel$ components, a and b.

$r(t)$ sweeps out a circle with radius $|b|$.

The tip velocity is $\omega(t) \times b$.

But $\omega(t) \parallel a$, so $\omega(t) \times a = 0$.

Thus… $\dot{r} = \omega \times (a + b) = \omega \times r$



Where did that come from? Cont'd.

Apply this rule to the three coordinate axes (r_x, r_y, r_z) of our rigid body in world space…

$$\dot{R} = (\omega \times r_x \quad \omega \times r_y \quad \omega \times r_z)$$

But these axes comprise our orientation matrix, R!

The cross product operation "$\omega \times$" can be equivalently expressed in matrix form as: $\begin{pmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{pmatrix}$

So this gives us back our solution, $\dot{R} = \begin{pmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{pmatrix} R = \omega^* R$