

# Physics-based Simulation

- simple (independent particles), or complex (robust colliding, stacking, sliding 3D rigid bodies)
- many many simulators!
  - PhysX (Unity, Unreal), Bullet, Open Dynamics Engine, MuJoCo, Havok, Box2D, Chipmunk, OpenSim, RBDL, Simulink (MATLAB), ADAMS, SD/FAST, DART, Vortex, SOFA, Avatar, Project Chrono, Cannon.js, ...
  - many course projects, theses, abandon-ware

# Resources

- <https://processing.org/examples/> see “Simulate”; 2D particle systems
- Non-convex rigid bodies with stacking 3D collision processing and stacking  
[http://www.cs.ubc.ca/~rbridson/docs/rigid\\_bodies.pdf](http://www.cs.ubc.ca/~rbridson/docs/rigid_bodies.pdf)
- Physically-based Modeling, course notes, SIGGRAPH 2001, Baraff & Witkin  
<http://www.pixar.com/companyinfo/research/pbm2001/>
- Doug James CS 5643 course notes  
<http://www.cs.cornell.edu/courses/cs5643/2015sp/>
- Rigid Body Dynamics, Chris Hecker [http://chrishecker.com/Rigid\\_Body\\_Dynamics](http://chrishecker.com/Rigid_Body_Dynamics)
- Video game physics tutorial  
<https://www.toptal.com/game/video-game-physics-part-i-an-introduction-to-rigid-body-dynamics>
- Box2D javascript live demos <http://heikobehrens.net/misc/box2d.js/examples/>
- Rigid body collisions javascript demo <https://www.myphysicslab.com/engine2D/collision-en.html>
- Rigid Body Collision Response, Michael Manzke, course slides  
<https://www.scss.tcd.ie/Michael.Manzke/CS7057/cs7057-1516-09-CollisionResponse-mm.pdf>
- Rigid Body Dynamics Algorithms. Roy Featherstone, 2008
- [Particle-based Fluid Simulation for Interactive Applications](#), SCA 2003, PDF
- Stable Fluids, Jos Stam, SIGGRAPH 1999. interactive demo:  
<https://29a.ch/2012/12/16/webgl-fluid-simulation>

# Simulation Basics

- simulation loop
  - Equations of Motion

sum forces & torques;  
solve for accelerations
  - numerical integration

update positions, velocities
  - collision detection
  - collision resolution
- basic particle sim

$$dt = t_{i+1} - t_i$$

$$\mathbf{v}(t_{i+1}) = \mathbf{v}(t_i) + (\mathbf{f}(t_i)/m)dt$$

$$\mathbf{p}(t_{i+1}) = \mathbf{p}(t_i) + \mathbf{v}(t_{i+1})dt$$

# Physically-based Simulation – movement governed by forces

- particle systems
  - fire, water, smoke, (fluids, cloth, hair)
  - heuristics-based or more principled
- rigid-body simulation
  - blocks, robots, humans
- continuum systems
  - deformable solids
  - fluids, cloth, hair
- group movement
  - flocks, crowds

# Ordinary Differential Equations

$$\frac{dX(t)}{dt} = f(X(t), t)$$

Simulation as a path  
through state-space,  
driven by a vector field

Given  $X_0 = X(t_0)$   
Compute  $X(t)$  for  $t > t_0$

# Newtonian Physics as first-order ODE

- motion of one particle
- motion of many particles

# Particle Forces

- gravity

$$\|\mathbf{F}_{ij}\| = \frac{G m_i m_j}{r^2}$$

$$f^{(i)} = \begin{pmatrix} 0 \\ 0 \\ -m_i G \end{pmatrix}$$

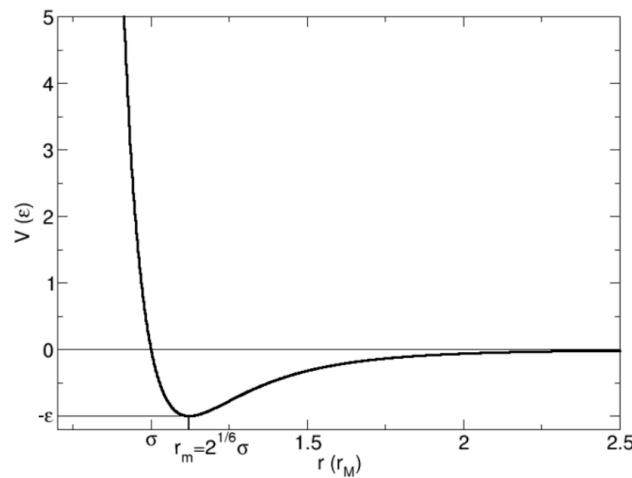
- viscous damping

$$f^{(i)} = -d\mathbf{v}^{(i)}$$

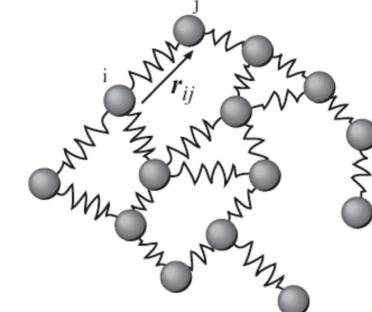
- springs & dampers

$$F = -k_p x - k_d v$$

- Lennard-Jones potentials



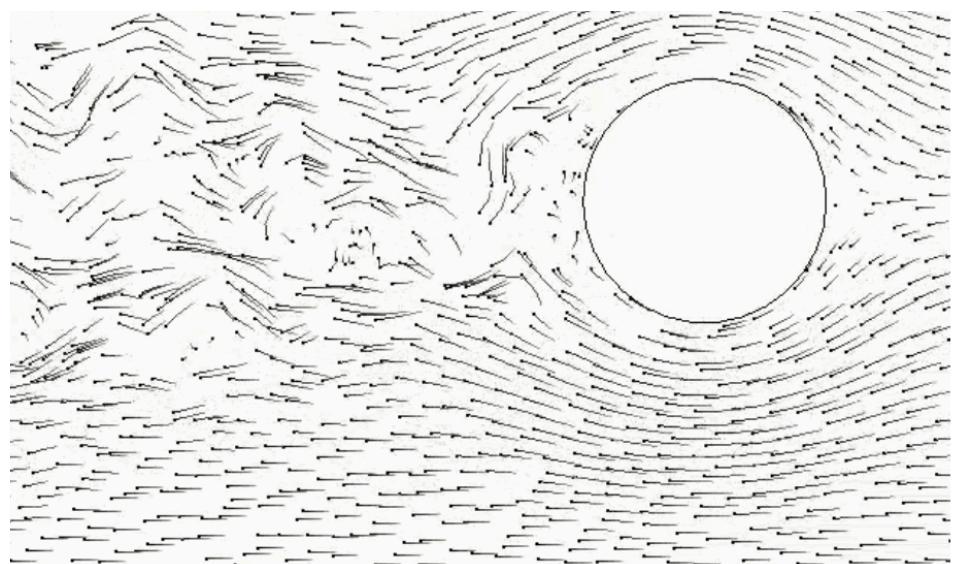
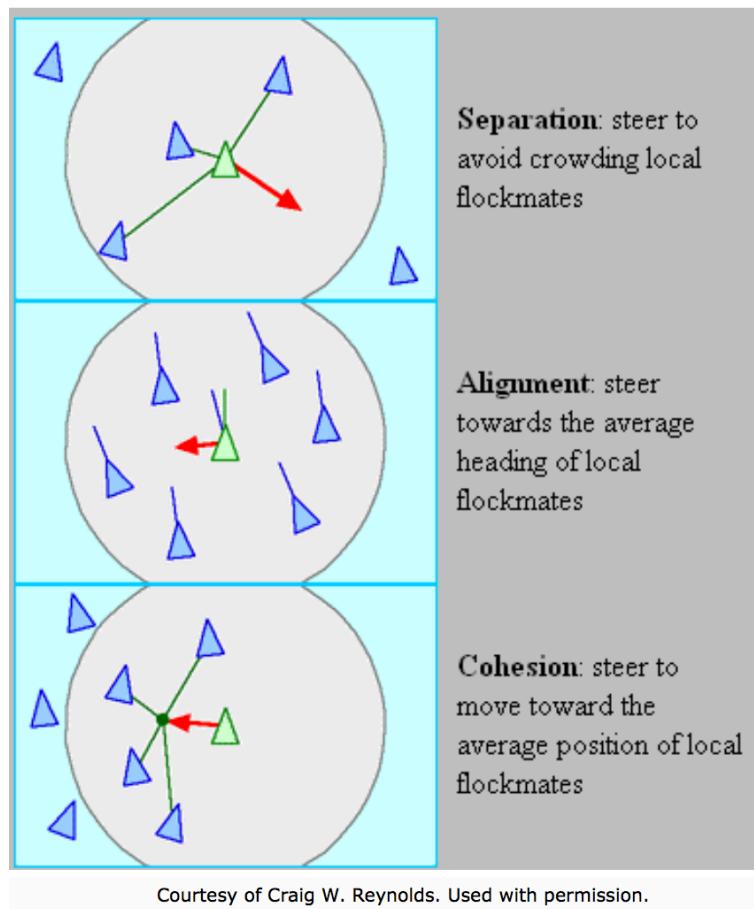
$$\mathbf{f}_a = - \left[ k_s (|\mathbf{l}| - r) + k_d \frac{\dot{\mathbf{l}} \cdot \mathbf{l}}{|\mathbf{l}|} \right] \frac{\mathbf{l}}{|\mathbf{l}|}, \quad \mathbf{f}_b = -\mathbf{f}_a,$$



# Proxy Forces

- behavior forces:  
flocking birds, schooling fish, etc.  
[“Boids”, Craig Reynolds, SIGGRAPH 1987]

Curl noise for procedural fluid flow  
R. Bridson, J. Hourihan, M. Nordenstam,  
Proc. SIGGRAPH 2007.



# ODE Numerical Integration: Explicit (Forward) Euler

$$\frac{dX}{dt} = f(X)$$

$$\Delta X = \Delta t f(X)$$

$$t_1 = t_0 + h$$

$$X_1 = X_0 + h f(X_0, t_0)$$

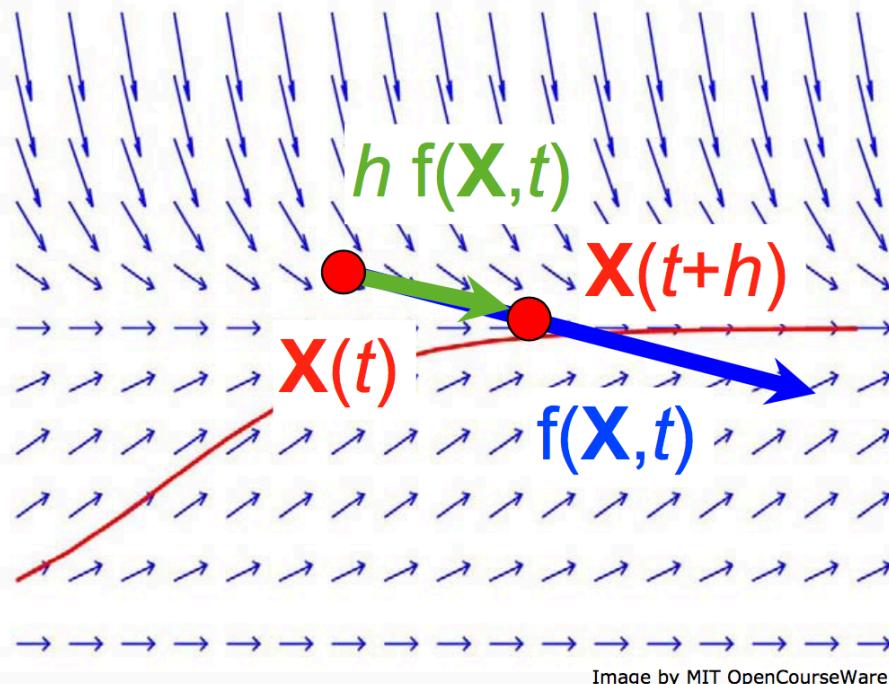


Image by MIT OpenCourseWare.

Images from  
MIT-OCW:

Wojciech Matusik, and Frédo Durand. 6.837 Computer Graphics. Fall 2012. Massachusetts Institute of Technology: MIT OpenCourseWare, <https://ocw.mit.edu>. License: Creative Commons BY-NC-SA.

# Explicit Euler: problems

Solution spirals outwards no matter how small the time step !

(although smaller time steps are still better)

Or can lead to instabilities:

[Physically Based Animation  
Witkin & Baraff, SIGGRAPH course notes]

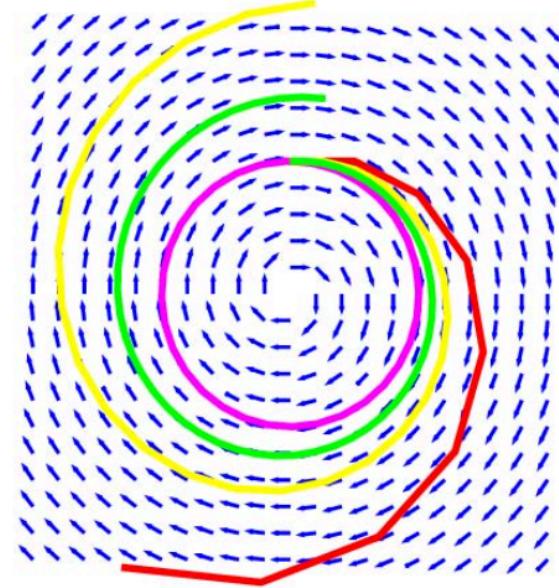
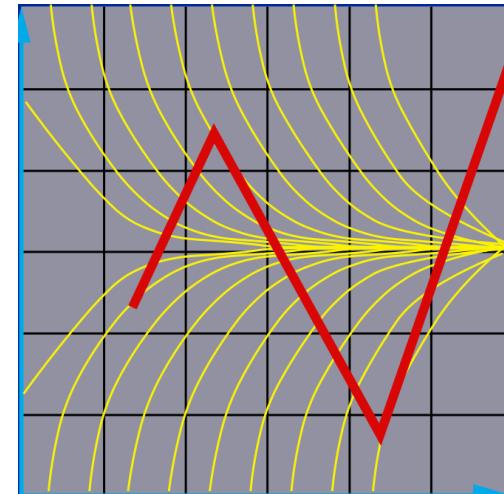
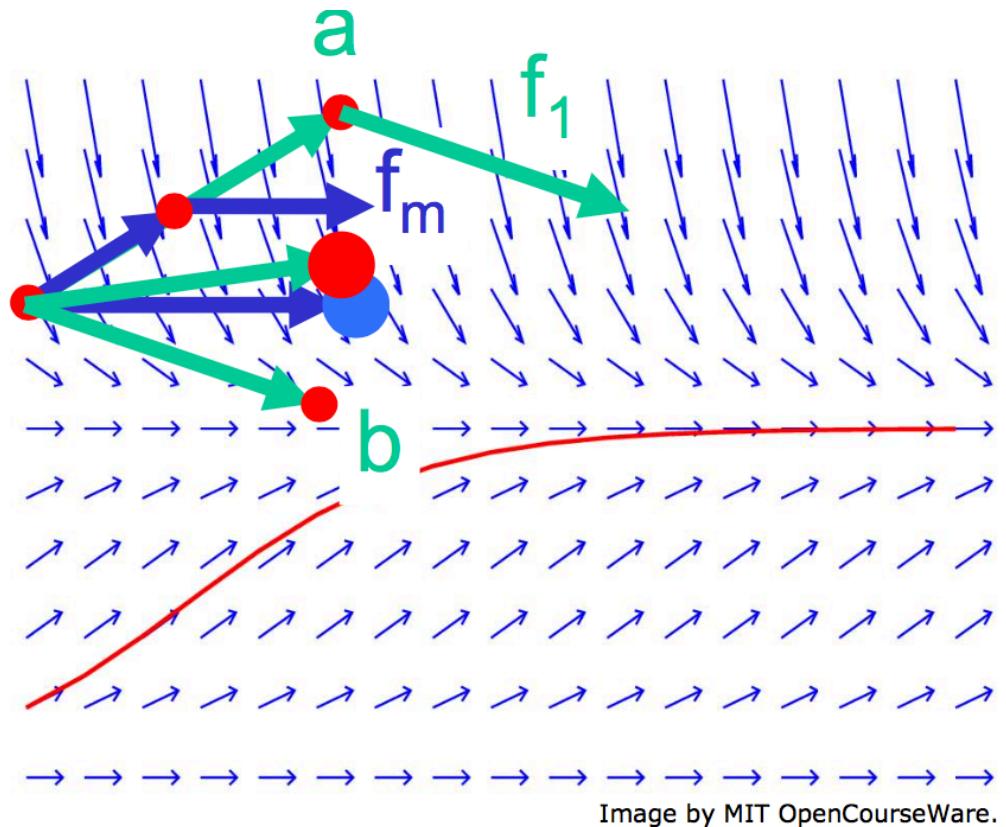


Image by MIT OpenCourseWare.



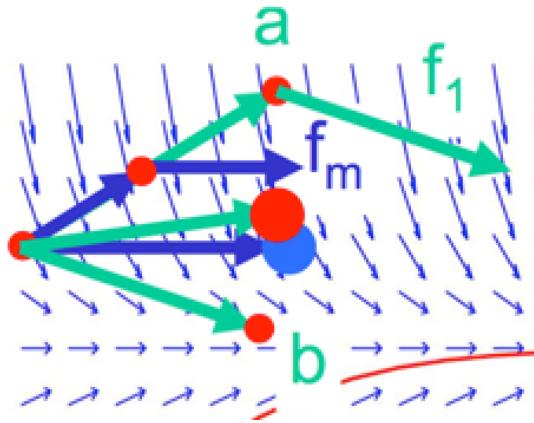
# Midpoint & Trapezoid methods

- **Midpoint:**
  - $\frac{1}{2}$  Euler step
  - evaluate  $f_m$
  - full step using  $f_m$
- **Trapezoid:**
  - Euler step (a)
  - evaluate  $f_1$
  - full step using  $f_1$  (b)
  - average (a) and (b)
- Not exactly same result,  
but same order of accuracy



Second order accurate;  
Midpoint method also known as  
“second order Runge-Kutta”

[slide in part from MIT-OCW]



### EXPLICIT EULER

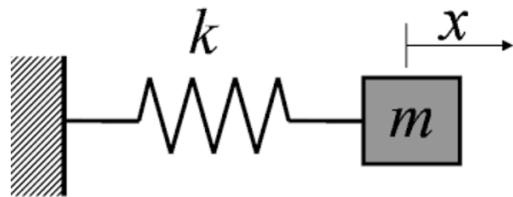
```
void takeStep(ParticleSystem* ps, float h)
{
    velocities = ps->getStateVelocities()
    positions = ps->getStatePositions()
    forces = ps->getForces(positions, velocities)
    masses = ps->getMasses()
    accelerations = forces / masses
    newPositions = positions + h*velocities
    newVelocities = velocities + h*accelerations
    ps->setStatePositions(newPositions)
    ps->setStateVelocities(newVelocities)
}
```

# Code

### MIDPOINT METHOD

```
void takeStep(ParticleSystem* ps, float h)
{
    velocities = ps->getStateVelocities()
    positions = ps->getStatePositions()
    forces = ps->getForces(positions, velocities)
    masses = ps->getMasses()
    accelerations = forces / masses
    midPositions = positions + 0.5*h*velocities
    midVelocities = velocities + 0.5*h*accelerations
    midForces = ps->getForces(midPositions, midVelocities)
    midAccelerations = midForces / masses
    newPositions = positions + 0.5*h*midVelocities
    newVelocities = velocities + 0.5*h*midAccelerations
    ps->setStatePositions(newPositions)
    ps->setStateVelocities(newVelocities)
}
```

# Implicit Euler



forward “explicit” Euler

$$x_{n+1} = x_n + h v_n$$

$$v_{n+1} = v_n + h \left( \frac{-kx_n}{m} \right)$$

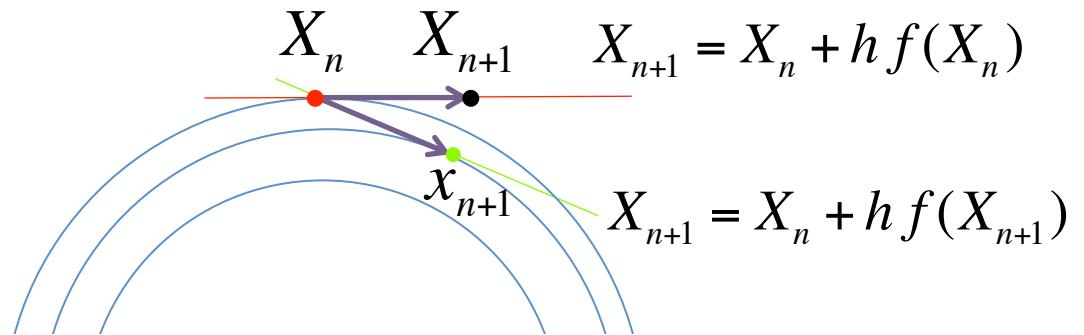
backward “implicit” Euler

$$x_{n+1} = x_n + h v_{n+1}$$

$$v_{n+1} = v_n + h \left( \frac{-kx_{n+1}}{m} \right)$$

Idea: Use the derivative at the destination

Problem: We don't know the destination yet !!



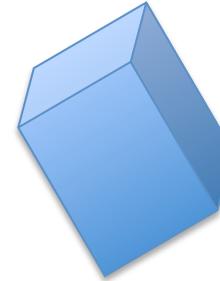


# Rigid Body Dynamics

- from particles to rigid bodies...



Newton's equations of motion



Newton-Euler equations of motion

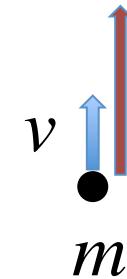
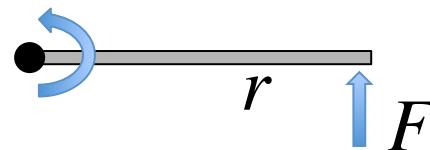
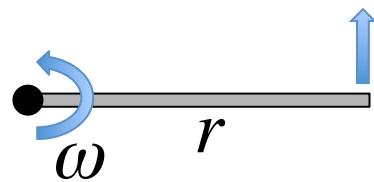
# Preliminaries

- cross product via a matrix multiply

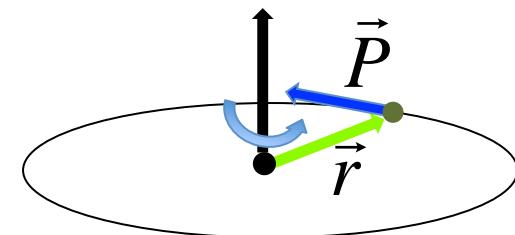
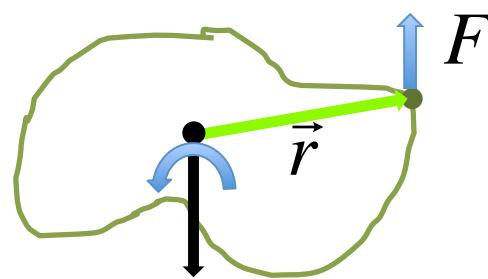
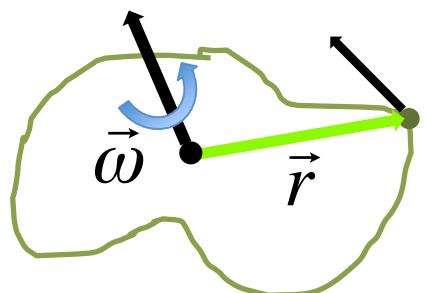
$$\tilde{a} = \begin{pmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{pmatrix}$$

# Kinematics of Rotation

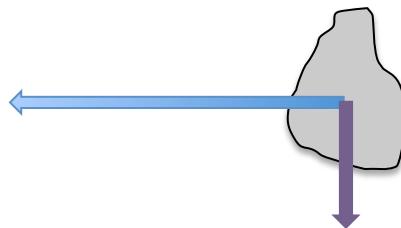
- Intuitively, with scalars:



- More generally:

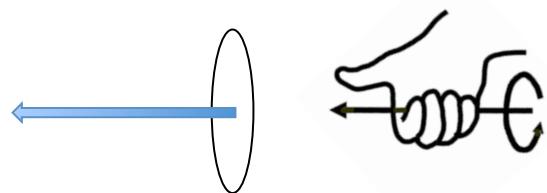


# Newton's Law

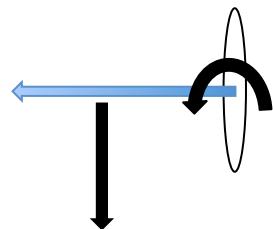


# Euler's Law

top view



side view



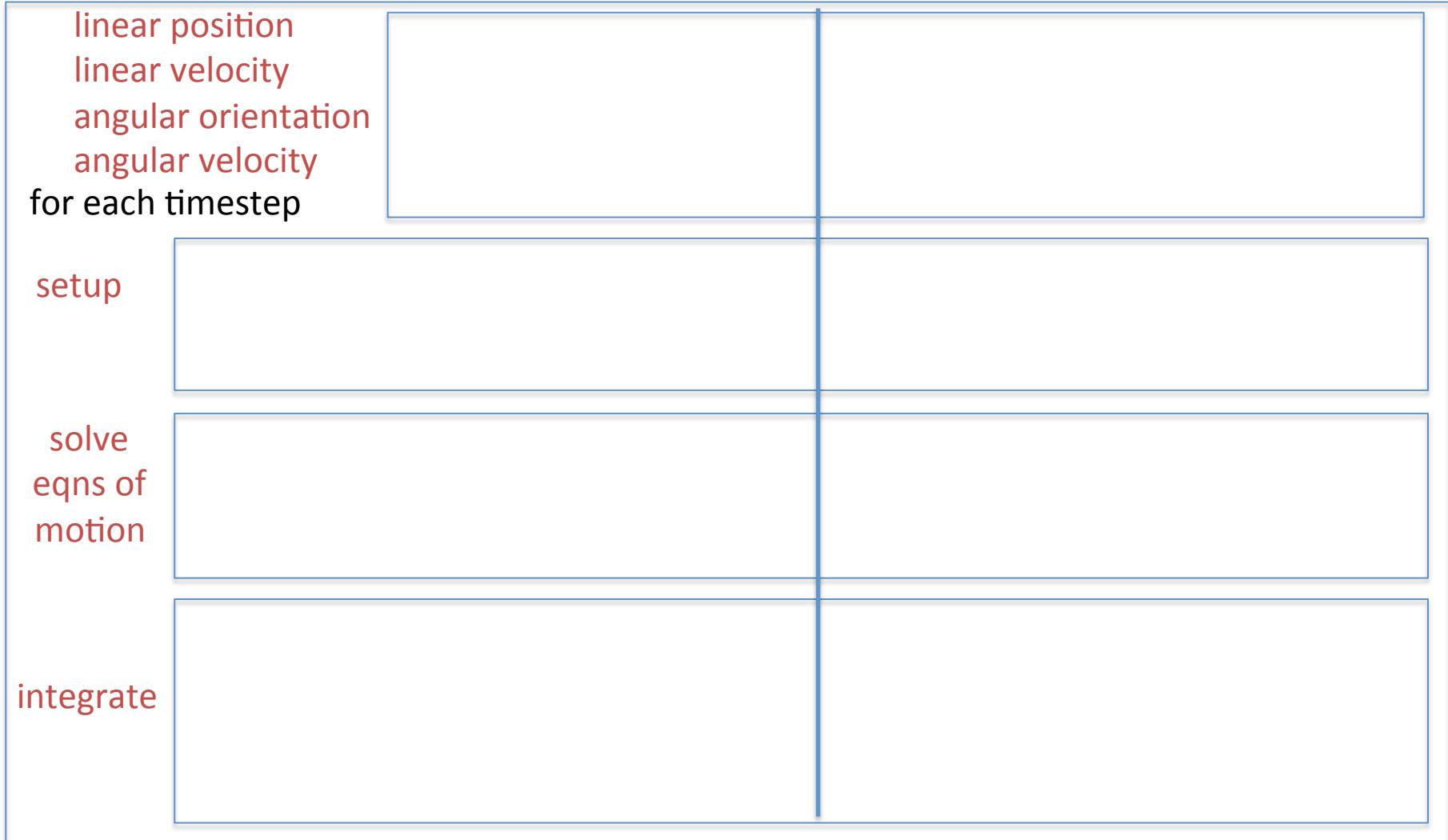
# Angular Momentum of a Set of Particles

# Inertia Tensor

# Newton-Euler Equations of Motion

# Updating the Inertia Tensor

# Simulation Loop



# Rigid Body Collisions

- Collision Detection
  - broad phase: e.g., AABBs, bounding spheres
  - narrow phase: detailed checks
- Collision Response
  - collision impulses
  - constraint forces: resting, sliding, hinges, ....

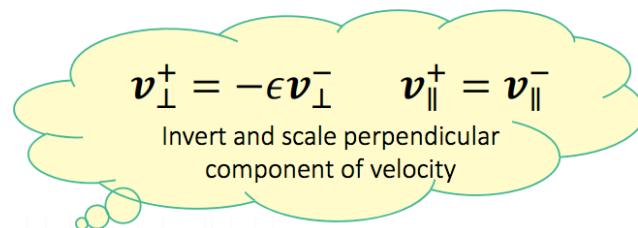
# Particle-Plane Collisions

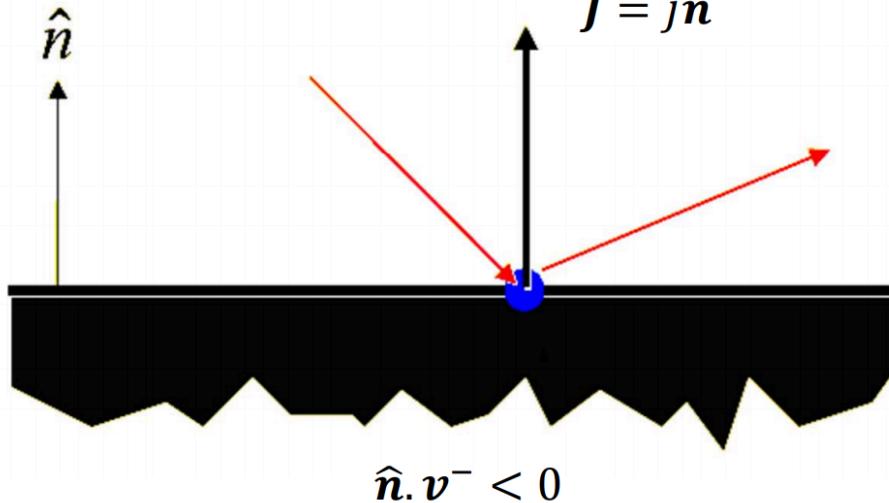
<https://www.scss.tcd.ie/Michael.Manzke/CS7057/cs7057-1516-09-CollisionResponse-mm.pdf>

17

## PARTICLE-PLANE: FRICTIONLESS IMPULSE RESPONSE

Change in velocity caused by applying an impulse in direction normal to plane, and of magnitude  $j$


$$\begin{aligned} v_{\perp}^{+} &= -\epsilon v_{\perp}^{-} & v_{\parallel}^{+} &= v_{\parallel}^{-} \\ \text{Invert and scale perpendicular component of velocity} \end{aligned}$$



$$v^+ = \frac{J}{m} + v^-$$

$$J = j\hat{n}$$

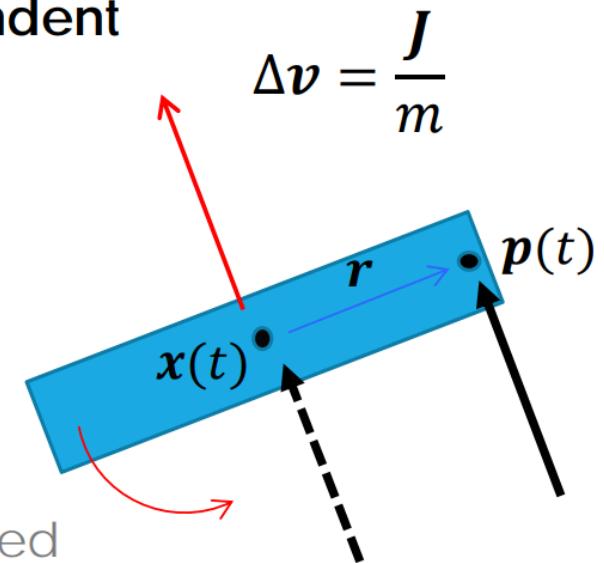
$$j = 1 + \epsilon$$

Not this easy for Rigid Bodies

## Impulse causes change in Linear and Angular velocity

The effect of an impulse (or for that matter a force) on an object's linear and angular momentum are independent

- Linear component: similar to particles
  - Causes a change in velocity inversely proportional to mass
  - As if force was applied at c.o.m.
- Angular component: impulsive torque
  - Causes change in angular velocity inversely proportional to moment of inertia (determined from inertial tensor)
  - Dependent on position of impulse



$$\begin{aligned}\Delta\omega &= I^{-1}(r \times J) \\ &= I^{-1}((p - x) \times J)\end{aligned}$$

**But what is the value of  $J$ ?**

# RIGID BODY COLLISION

<https://www.scss.tcd.ie/Michael.Manzke/CS7057/cs7057-1516-09-CollisionResponse-mm.pdf>

$$v_{rel}^+ = -\epsilon v_{rel}^-$$

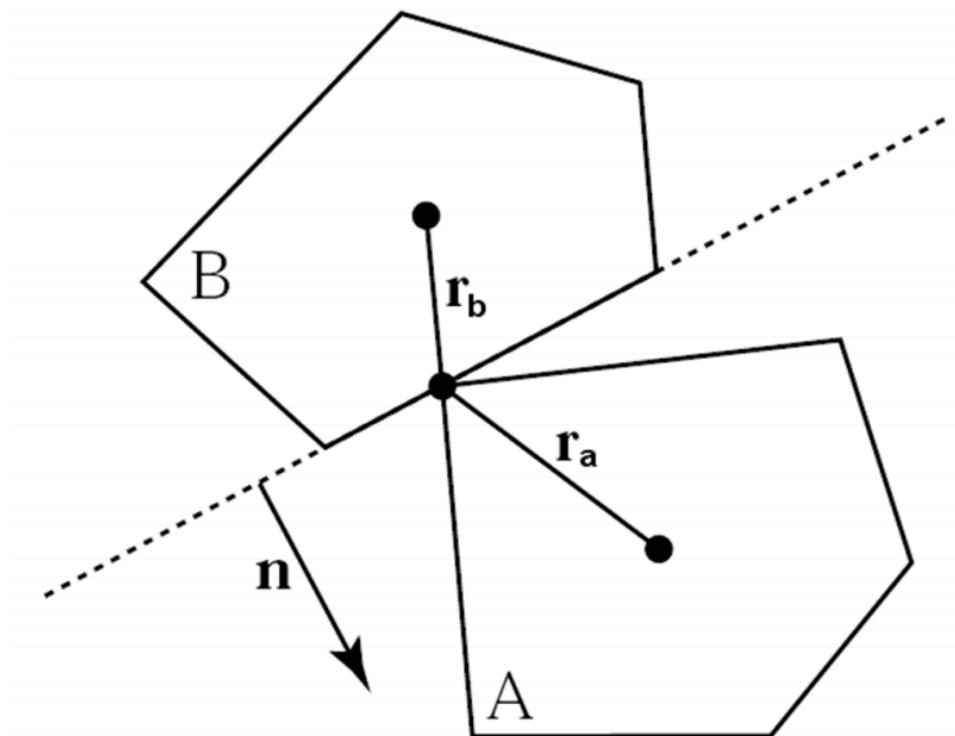
$$v_{rel} = \hat{n}(\dot{p}_A - \dot{p}_B)$$

$$\begin{aligned}\dot{p}_A &= v_A + \omega_A \times (p_A - x_A) \\ &= v_A + \omega_A \times r_A\end{aligned}$$

$$\begin{aligned}\dot{p}_A &= v_B + \omega_B \times (p_B - x_B) \\ &= v_A + \omega_A \times r_A\end{aligned}$$



Linear velocity  
component



Angular component: Linear velocity of point p due to its rotation. See previous slide

Post-collision velocity should be

$$\dot{\mathbf{p}}_A^+ = \mathbf{v}_A^+ + \boldsymbol{\omega}_A^+ \times \mathbf{r}_A$$

However we need to express this in terms of the previous state of the object (pre-collision values)

$$\begin{aligned} \mathbf{v}_A^+ &= \mathbf{v}_A^- + \Delta\mathbf{v} = \mathbf{v}_A^- + \frac{j\hat{\mathbf{n}}}{m_A} && \text{SEE SLIDE 19} \\ \boldsymbol{\omega}_A^+ &= \boldsymbol{\omega}_A^- + \Delta\boldsymbol{\omega} = \boldsymbol{\omega}_A^- + \mathbf{I}_A^{-1}(\mathbf{r}_A \times j\hat{\mathbf{n}}) \end{aligned}$$

$$\dot{\mathbf{p}}_A^+ = \mathbf{v}_A^- + \frac{j\hat{\mathbf{n}}}{m_A} + (\boldsymbol{\omega}_A^- + \mathbf{I}_A^{-1}(\mathbf{r}_A \times j\hat{\mathbf{n}})) \times \mathbf{r}_A$$

$$\begin{aligned} &= \boxed{\mathbf{v}_A^- + \boldsymbol{\omega}_A^- \times \mathbf{r}_A} + j \left( \frac{\hat{\mathbf{n}}}{m_A} + \mathbf{I}_A^{-1}(\mathbf{r}_A \times \hat{\mathbf{n}}) \right) \times \mathbf{r}_A \\ &\quad \downarrow \text{PREVIOUS SLIDE} \\ &= \dot{\mathbf{p}}_A^- + j \left( \frac{\hat{\mathbf{n}}}{m_A} + \mathbf{I}_A^{-1}(\mathbf{r}_A \times \hat{\mathbf{n}}) \right) \times \mathbf{r}_A \end{aligned}$$

Similarly (by Newton's 3<sup>rd</sup> Law: every reaction has equal and opposite reaction):

$$\dot{\mathbf{p}}_B^+ = \dot{\mathbf{p}}_B^- + j \left( \frac{\hat{\mathbf{n}}}{m_A} + \mathbf{I}_A^{-1}(\mathbf{r}_A \times \hat{\mathbf{n}}) \right) \times \mathbf{r}_A$$

<https://www.scss.tcd.ie/Michael.Manzke/CS7057/cs7057-1516-09-CollisionResponse-mm.pdf>

Putting this in  $v_{rel}^+ = \hat{\mathbf{n}}(\dot{\mathbf{p}}_A^+ - \dot{\mathbf{p}}_B^+)$  we get

$$\begin{aligned}
 v_{rel}^+ &= \hat{\mathbf{n}} \left( \left( \boxed{\mathbf{p}_A^- + j \left( \frac{\hat{\mathbf{n}}}{m_A} + \mathbf{I}_A^{-1}(\mathbf{r}_A \times \hat{\mathbf{n}}) \right) \times \mathbf{r}_A} \right) - \boxed{\mathbf{p}_B^- + j \left( \frac{\hat{\mathbf{n}}}{m_B} + \mathbf{I}_B^{-1}(\mathbf{r}_B \times \hat{\mathbf{n}}) \right) \times \mathbf{r}_B} \right) \\
 &= \boxed{\hat{\mathbf{n}}(\dot{\mathbf{p}}_A^- - \dot{\mathbf{p}}_B^-)} + j \left( \frac{1}{m_A} + \frac{1}{m_B} + \hat{\mathbf{n}}(\mathbf{I}_A^{-1}(\mathbf{r}_A \times \hat{\mathbf{n}})) \times \mathbf{r}_A + \hat{\mathbf{n}}(\mathbf{I}_B^{-1}(\mathbf{r}_B \times \hat{\mathbf{n}})) \times \mathbf{r}_B \right) \\
 &= v_{rel}^- + j \left( \frac{1}{m_A} + \frac{1}{m_B} + \hat{\mathbf{n}}(\mathbf{I}_A^{-1}(\mathbf{r}_A \times \hat{\mathbf{n}})) \times \mathbf{r}_A + \hat{\mathbf{n}}(\mathbf{I}_B^{-1}(\mathbf{r}_B \times \hat{\mathbf{n}})) \times \mathbf{r}_B \right)
 \end{aligned}$$

From previous slide

And since  $v_{rel}^+ = -\epsilon v_{rel}^-$

$$\begin{aligned}
 -\epsilon v_{rel}^- &= v_{rel}^- + j \left( \frac{1}{m_A} + \frac{1}{m_B} + \hat{\mathbf{n}}(\mathbf{I}_A^{-1}(\mathbf{r}_A \times \hat{\mathbf{n}})) \times \mathbf{r}_A + \hat{\mathbf{n}}(\mathbf{I}_B^{-1}(\mathbf{r}_B \times \hat{\mathbf{n}})) \times \mathbf{r}_B \right) \\
 \text{giving } j &= \frac{-(1 + \epsilon) v_{rel}^-}{m_A^{-1} + m_B^{-1} + \hat{\mathbf{n}}(\mathbf{I}_A^{-1}(\mathbf{r}_A \times \hat{\mathbf{n}})) \times \mathbf{r}_A + \hat{\mathbf{n}}(\mathbf{I}_B^{-1}(\mathbf{r}_B \times \hat{\mathbf{n}})) \times \mathbf{r}_B}
 \end{aligned}$$

This gives us the impulse magnitude we were looking for

27

# IMPULSE MAGNITUDE EQUATION

WHERE THE VARIABLES COME FROM

$$j = \frac{-(1 + \epsilon) v_{rel}^-}{m_A^{-1} + m_B^{-1} + \hat{n} (I_A^{-1}(\mathbf{r}_A \times \hat{n})) \times \mathbf{r}_A + \hat{n} (I_B^{-1}(\mathbf{r}_B \times \hat{n})) \times \mathbf{r}_B}$$

$$\mathbf{I}_A^{-1} = \mathbf{R}_A \mathbf{I}_{bodyA}^{-1} \mathbf{R}_A^T \quad \text{and} \quad \mathbf{I}_B^{-1} = \mathbf{R}_B \mathbf{I}_{bodyB}^{-1} \mathbf{R}_B^T$$

Constants

$$\mathbf{r}_A = \mathbf{p}_A - \mathbf{x}_A \quad \text{and} \quad \mathbf{r}_B = \mathbf{p}_B - \mathbf{x}_B$$

Intermediate Variables

$$v_{rel}^- = \hat{n}(\dot{\mathbf{p}}_A^- - \dot{\mathbf{p}}_B^-)$$

Rigid Body State

$$\dot{\mathbf{p}}_A = \mathbf{v}_A + \boldsymbol{\omega}_A \times (\mathbf{p}_A - \mathbf{x}_A) \quad \text{and} \quad \dot{\mathbf{p}}_B = \mathbf{v}_B + \boldsymbol{\omega}_B \times (\mathbf{p}_B - \mathbf{x}_B)$$

Contact Model

$j$ : impulse magnitude

$\mathbf{v}_A, \mathbf{v}_B$ : velocity of A and B

$\epsilon$ : coefficient of restitution

$v_{rel}^-$ : pre-collision relative velocity of contact points

$m_A, m_B$ : mass of object A and B

$\mathbf{I}_A, \mathbf{I}_B$ : world space inertial tensor

$\mathbf{R}_A, \mathbf{R}_B$ : orientation of A and B

$\mathbf{I}_{bodyA}, \mathbf{I}_{bodyB}$ : object space inertial tensor

$\boldsymbol{\omega}_A, \boldsymbol{\omega}_B$ : angular velocity of A and B

$\hat{n}$ : contact plane normal

$\mathbf{x}_A, \mathbf{x}_B$ : centre of mass position of A and B

$\mathbf{p}_A, \mathbf{p}_B$ : contact point on A and B

28

# APPLYING IMPULSE

Impulse magnitude is given by:

$$j = \frac{-(1 + \epsilon) v_{rel}^-}{m_A^{-1} + m_B^{-1} + \hat{\mathbf{n}}(\mathbf{I}_A^{-1}(\mathbf{r}_A \times \hat{\mathbf{n}})) \times \mathbf{r}_A + \hat{\mathbf{n}}(\mathbf{I}_B^{-1}(\mathbf{r}_B \times \hat{\mathbf{n}})) \times \mathbf{r}_B}$$

The actual impulse vector is simply  $\mathbf{J} = j\hat{\mathbf{n}}$

This is applied to the objects as follows:

- Change in Linear momentum is directly equal to the impulse:

$$\Delta \mathbf{P} = \mathbf{J} \iff \Delta \mathbf{v} = \mathbf{J} m^{-1}$$

- Change in Angular momentum is equal to the impulsive torque ( $\tau_{\text{IMPULSE}}$ ):

$$\Delta \mathbf{L} = (\mathbf{r} \times \mathbf{J}) \iff \Delta \boldsymbol{\omega} = \mathbf{I}^{-1}(\mathbf{r} \times \mathbf{J})$$