

Physically-based Modelling

Dynamics and Numerical Integration

Young-Min Kang
Tongmyong University

Dynamics

- ❖ Dynamics
 - ❖ force-based understanding of motions
- ❖ Important formula
 - ❖ $\mathbf{f} = m\mathbf{a}$
- ❖ Rigid body

$$\boldsymbol{\tau} = \mathbf{I}\dot{\boldsymbol{\omega}}$$

Integration of Equations of Motion

❖ Newton's 2nd law of motion

❖ $\mathbf{f} = m\mathbf{a}$

❖ In other words,

$$\mathbf{f} = m \frac{d\mathbf{v}}{dt}$$

❖ $d\mathbf{v} = ?$

$$\frac{\mathbf{f} dt}{m} = d\mathbf{v}$$

$$\int_{t_1}^{t_2} \frac{\mathbf{f} dt}{m} = \int_{\mathbf{v}_1}^{\mathbf{v}_2} d\mathbf{v}$$

if force is constant

$$\frac{\mathbf{f}}{m} (t_2 - t_1) = \mathbf{v}_2 - \mathbf{v}_1$$

$$\frac{\mathbf{f}}{m} \Delta t = \Delta \mathbf{v}$$

Initial Condition Problem

- ❖ Initial conditions
 - ❖ $x(t), v(t)$
 - ❖ location and velocity at current time t
- ❖ Problem
 - ❖ predict $x(t+dt)$ and $v(t+dt)$
 - ❖ predict the location and velocity after a small time step dt
- ❖ Repeat with Initial conditions = $x(t+dt), v(t+dt)$

Velocity update

- ❖ Initial condition of velocity

- ❖ $\mathbf{v}_1 = \mathbf{v}(t)$

- ❖ Velocity to be found

- ❖ $\mathbf{v}_2 = \mathbf{v}(t + \Delta t)$

- ❖ Velocity at the next frame

$$\mathbf{v}(t + \Delta t) = \mathbf{v}(t) + \frac{\mathbf{f}(t)}{m} \Delta t$$

- ❖ or

$$\mathbf{v}(t + \Delta t) = \mathbf{v}(t) + \mathbf{a}(t) \Delta t$$

$$\frac{\mathbf{f}}{m} (t_2 - t_1) = \mathbf{v}_2 - \mathbf{v}_1$$

$$\frac{\mathbf{f}}{m} \Delta t = \Delta \mathbf{v}$$

Position Update

- ❖ Velocity and Position

$$\mathbf{v} = d\mathbf{s}/dt \qquad \mathbf{v}dt = d\mathbf{s}$$

- ❖ Numerical Integration

$$\mathbf{v}(t + \Delta t)\Delta t = \Delta\mathbf{s}$$

- ❖ If force (or acceleration) is known at time t
 - ❖ We can “approximately” predict velocity and location at time $t+dt$

Realtime Simulation

- ❖ compute force: \mathbf{f}
- ❖ compute acceleration: $\mathbf{a} = \mathbf{f} / m$
- ❖ predict the velocity after time step
 - ❖ Euler integration

$$\mathbf{v}(t + \Delta t) = \mathbf{a}\Delta t$$

- ❖ predict the location after time step
 - ❖ Euler integration

$$\mathbf{x}(t + \Delta t) = \mathbf{v}(t + \Delta t)\Delta t$$

Simulation Code - main.cpp

```
void keyboardFunction(unsigned char key, int x, int y) {
    if (key == 27) exit(0);
    switch (key) {
        case 's':
            if(!myWatch.bRunning()) { Simulator->start(); myWatch.start(); }
            else { myWatch.stop(); Simulator->stop(); }
            break;
        case 'p':
            myWatch.pause(); Simulator->pause();
            break;
        case 'r':
            myWatch.resume(); Simulator->resume();
        default:
            break;
    }
}

void displayFunction(void) {
    ...
    // check DT (in microsecond) from StopWatch and store it to "deltaTime" (in seconds)
    deltaTime = myWatch.checkAndComputeDT() / 1000000.0;
    currentTime = myWatch.getTotalElapsedTime() / 1000000.0;
    Simulator->actions(deltaTime, currentTime);
    glutSwapBuffers();
}
```

Simulation Code - Simulator.cpp

```
#include "Simulator.h"
#include <stdlib.h>
#include <stdio.h>
// Constructor
CSimulator::CSimulator(): bRunning(false) { }
CSimulator::~~CSimulator() { }

void CSimulator::actions(double dt, double currentTime) {
    if(bRunning) {
        doBeforeSimulation(dt, currentTime);
        doSimulation(dt, currentTime);
        doAfterSimulation(dt, currentTime);
    }
    visualize();
}
// Control Event Handlers
void CSimulator::start() {
    this->init();
    bRunning = true;
}
void CSimulator::stop() {
    bRunning = false;
    this->clean();
}
void CSimulator::pause() {}
void CSimulator::resume() {}
```

Simulation Code - DynamicSimulator.cpp

```
#include "DynamicSimulator.h"

CDynamicSimulator::CDynamicSimulator() : CSimulator() {}

void CDynamicSimulator::init() {
    for(int i=0;i<NUMPARTS;i++)particle[i].randomInit();
}

void CDynamicSimulator::clean() {}

void CDynamicSimulator::doBeforeSimulation(double dt, double currentTime) {}

void CDynamicSimulator::doSimulation(double dt, double currentTime) {
    for(int i=0;i<NUMPARTS;i++)
        particle[i].simulate(dt, currentTime);
}

void CDynamicSimulator::doAfterSimulation(double dt, double currentTime) {}

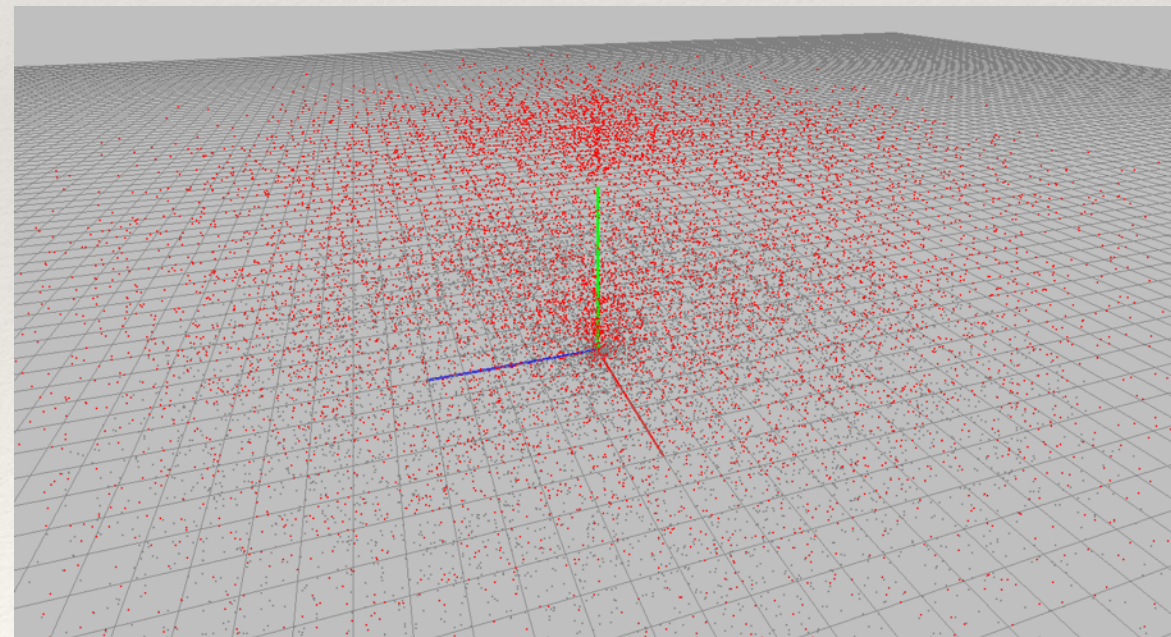
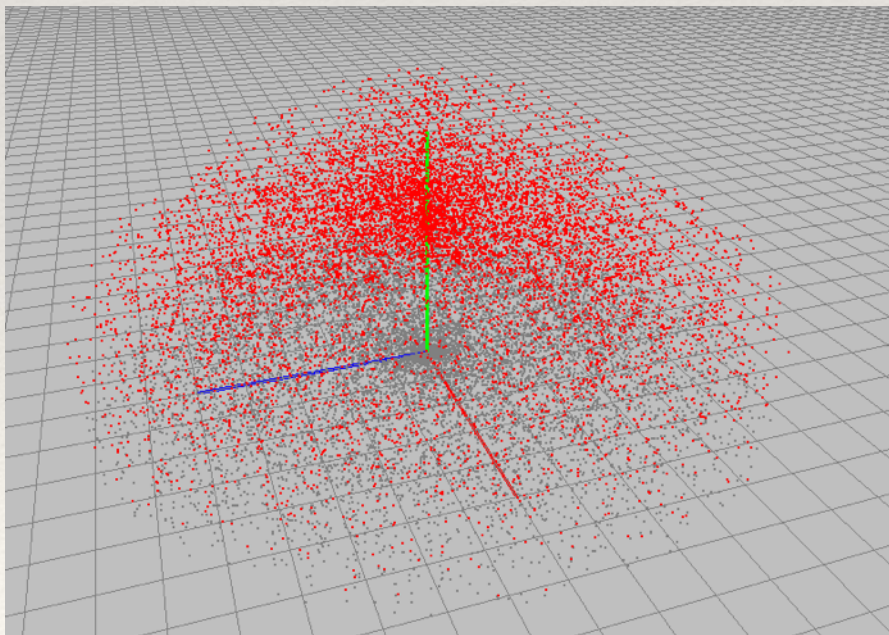
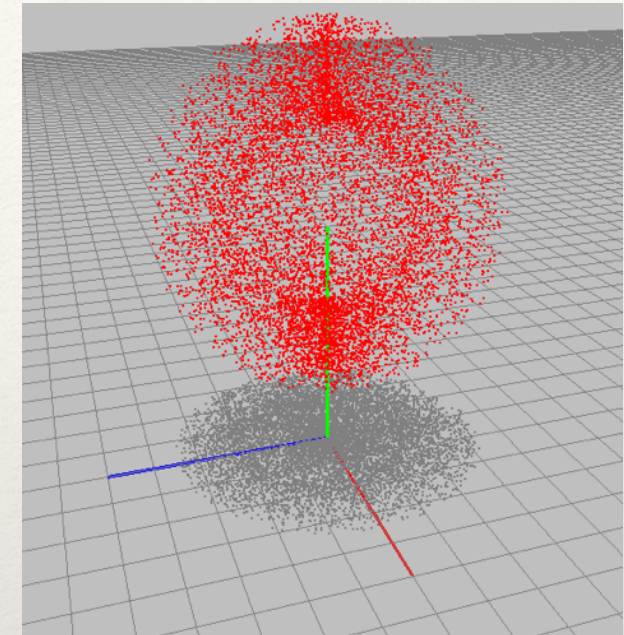
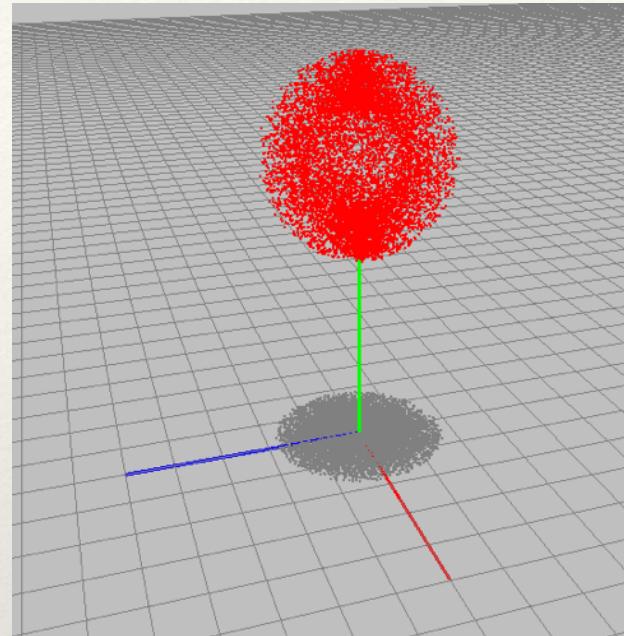
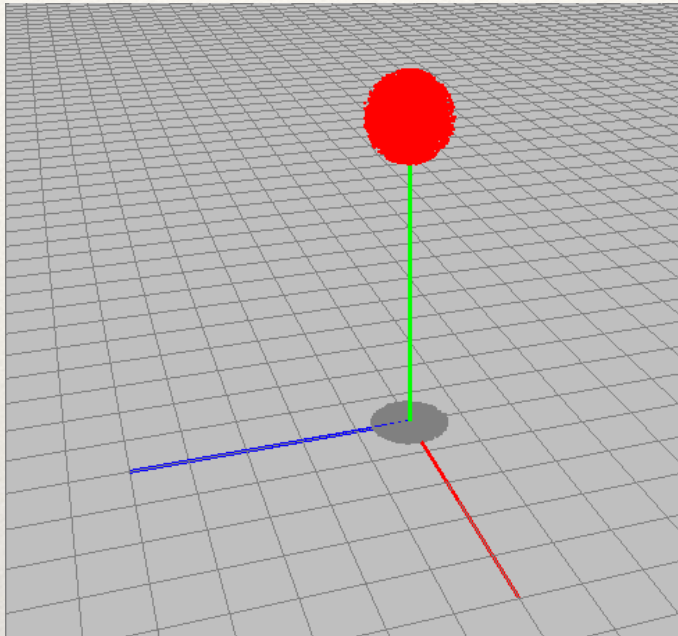
void CDynamicSimulator::visualize(void) {
    for(int i=0;i<NUMPARTS;i++) {
        particle[i].drawWithGL(POINT_DRAW);
    }
}
```

Simulation Code - Particle.cpp

```
void CParticle::simulate(double dt, double et) {
    // Update velocity: Euler Integration of acceleration
    vel = vel + dt*gravity;
    // Update position: Euler Integration of velocity
    loc = loc + dt*vel;

    // collision handling
    if(loc[1]<0) {
        loc.set(loc[0], -0.9*loc[1], loc[2]);
        if(vel[1]<0) vel.set(vel[0], -0.9*vel[1], vel[2]);
    }
    setPosition(loc[0], loc[1], loc[2]);
}
```


Result



Forces

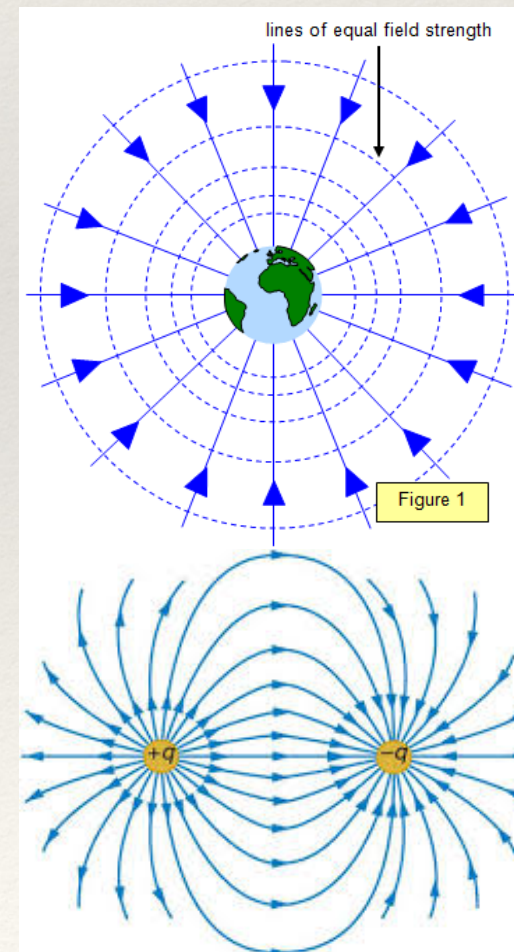
- ❖ Forces
 - ❖ cause motion
 - ❖ very important in dynamics
- ❖ Various force models

Concepts

- ❖ Force field : example - gravitational force
- ❖ Friction: a contact force that resists motion
- ❖ Fluid dynamic drag: resistance to objects moving in fluid
- ❖ Pressure: force per area
- ❖ Buoyancy: force pushing objects “upwards” when immersed in a fluid
- ❖ Springs and dampers: elastically ties objects
- ❖ Force applies torque to an object, and make it “rotate.”

Force field

- ❖ Force field
 - ❖ a vector field indicating the force exerted by on object on another
- ❖ Very good example
 - ❖ gravitational force field
 - ❖ electromagnetic force field



Gravitational force field

❖ universal gravitational force $|\mathbf{f}_u| = Gm_1m_2/r^2$

❖ G: gravitation constance $6.673 \times 10^{-11} (N \cdot m^2)/kg^2$

❖ r: distance between masses

❖ $m_{\{1,2\}}$: mass of each objects

❖ gravity on Earth

❖ mass of Earth: $5.98 \times 10^{24} kg$

❖ radius of Earth: $6.38 \times 10^6 m$

❖ gravitational acceleration

$$\frac{Gm_{earth}}{r^2} \simeq \left(\frac{6.673 \times 5.98}{6.38^2} \right) \times 10m/s^2 \simeq 9.8034m/s^2$$

friction

- ❖ resistance force due to the contacting surfaces

- ❖ contact force

- ❖ normal force: N is important

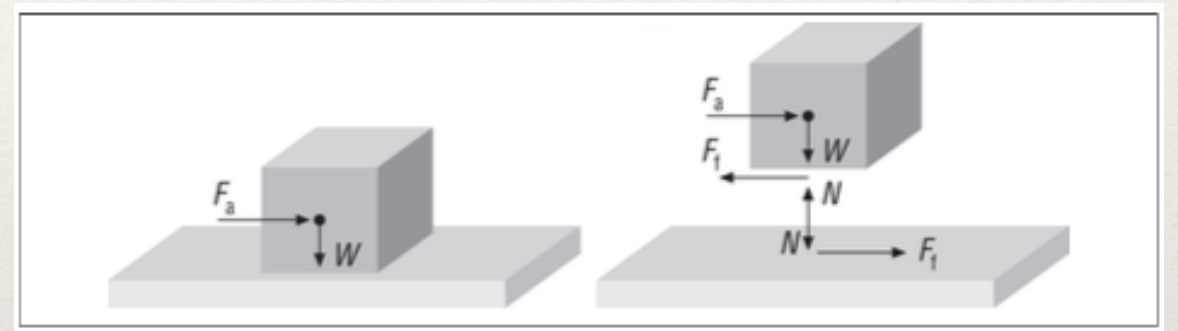
- ❖ two friction

- ❖ static friction: maximum friction

$$|\mathbf{f}_{max}| = \mu_s \mathbf{N}$$

- ❖ kinetic friction

$$|\mathbf{f}_k| = \mu_k \mathbf{N}$$



Coefficients of friction

❖ Well known surfaces

Surface condition	M_s	M_u	% difference
Dry glass on glass	0.94	0.4	54%
Dry iron on iron	1.1	0.15	86%
Dry rubber on pavement	0.55	0.4	27%
Dry steel on steel	0.78	0.42	46%
Dry Teflon on Teflon	0.04	0.04	—
Dry wood on wood	0.38	0.2	47%
Ice on ice	0.1	0.03	70%
Oiled steel on steel	0.10	0.08	20%

Fluid dynamic drag

- ❖ similar to friction
 - ❖ friction is also important component of drag
 - ❖ but, it's not the only one
- ❖ viscous drag for “slow-moving” objects (laminar)
 - ❖ $f = -C v$
- ❖ for “fast-moving” objects (turbulence)
 - ❖ $f = -C v^2$

Pressure

- ❖ Pressure is not force
 - ❖ pressure = force per unit area
 - ❖ $F = PA$ (force = pressure x area)
 - ❖ $P = F / A$
- ❖ Pressure is important in simulating
 - ❖ boats, hovercrafts,...

Buoyancy

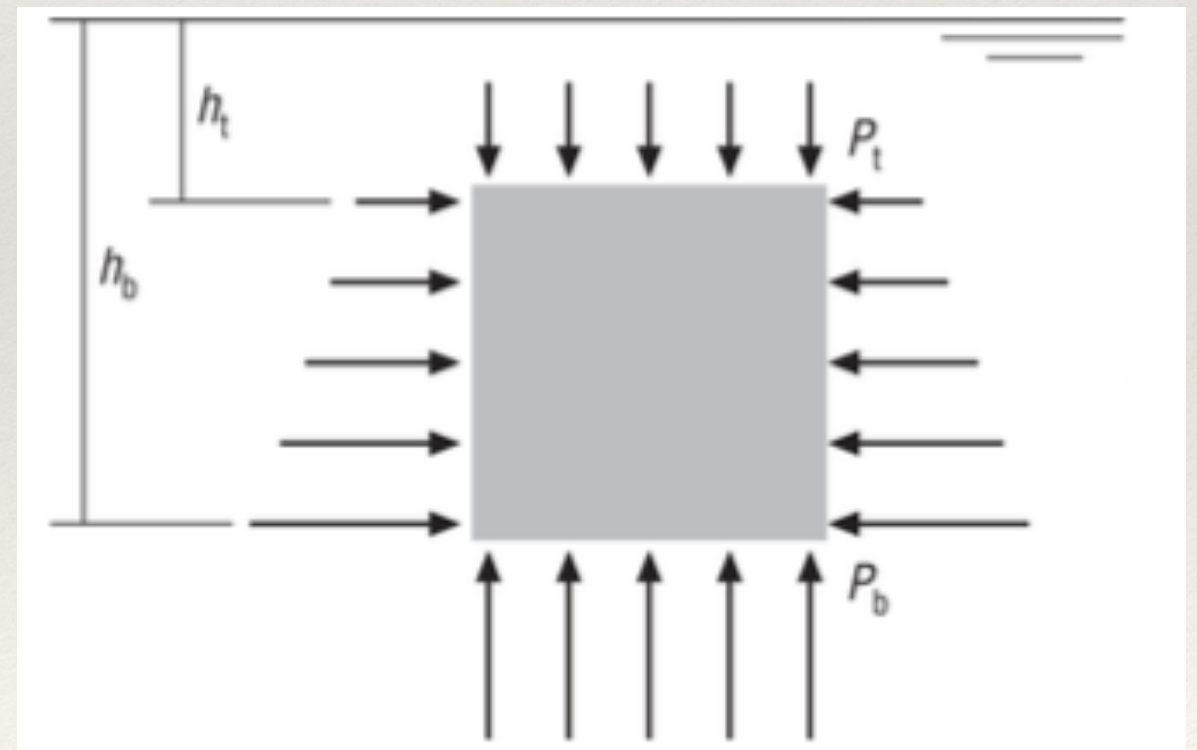
- ❖ Different pressure in fluid
- ❖ Horizontal net force = 0
- ❖ Vertical net force = bottom force - top force
- ❖ $F = PA$
- ❖ Pressures: function(density, gravity)

- ❖ on top surface

$$P_t = \rho g h_t$$

- ❖ on bottom surface

$$P_b = \rho g h_b$$



Buoyancy

❖ forces $\mathbf{f}_t = \mathbf{P}_t A_t = \rho \mathbf{g} h_t s^2$

$$\mathbf{f}_b = \mathbf{P}_b A_b = \rho \mathbf{g} h_b s^2$$

❖ difference

$$\mathbf{f}_b - \mathbf{f}_t = \rho \mathbf{g} h_b s^2 - \rho \mathbf{g} h_t s^2$$

$$= \rho \mathbf{g} (h_b - h_t) s^2$$

$$= -\rho \mathbf{g} s^3$$

$$= -\rho \mathbf{g} V \quad (V : \text{volume})$$

Spring force

- ❖ Hooke's law

- ❖ force needed to extend or compress a spring by a distance x is proportional to that distance

- ❖ $f = -k x$

- ❖ k : spring constant

- ❖ rest length of spring:

- ❖ current length of spring:

- ❖ force magnitude: $|\mathbf{f}| = k_s(L - r)$

- ❖ force direction: obj1 and obj2 are linked and located at x_1 and x_2 .

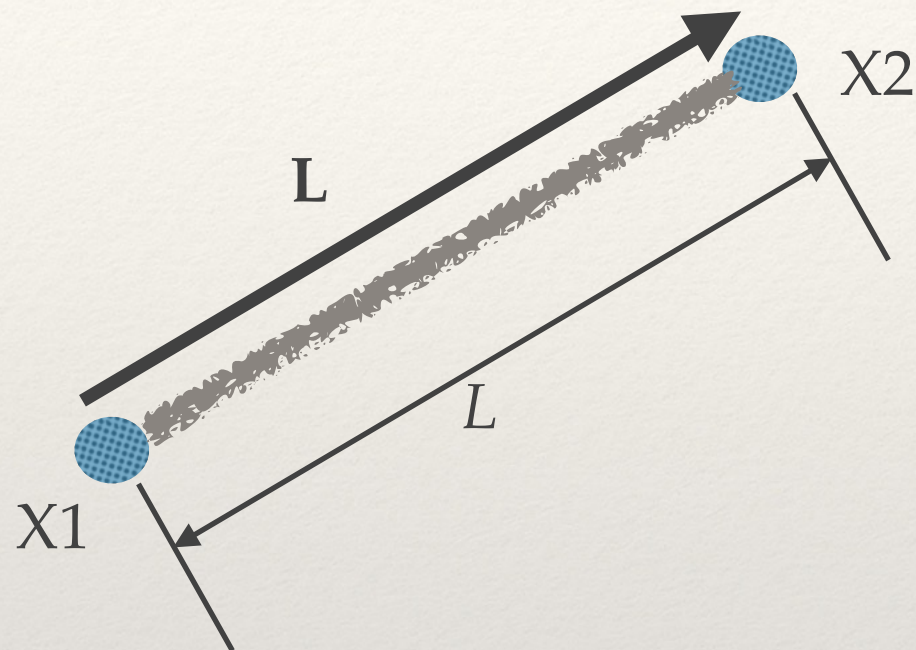
- ❖ $\frac{\mathbf{x}_1 - \mathbf{x}_2}{|\mathbf{x}_1 - \mathbf{x}_2|}$ and $-\frac{\mathbf{x}_1 - \mathbf{x}_2}{|\mathbf{x}_1 - \mathbf{x}_2|}$

Damper

- ❖ Springs do not oscillate forever
 - ❖ energy dissipation
 - ❖ simple model
 - ❖ damping force

$$\mathbf{f}_d = k_d(\mathbf{v}_1 - \mathbf{v}_2)$$

Spring and Damper



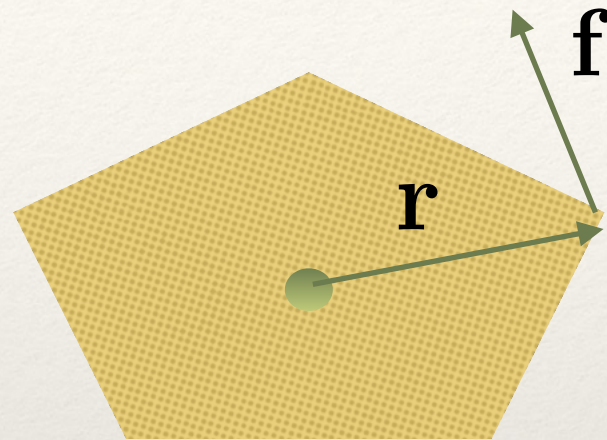
$$\mathbf{f}_1 = -\left(k_s(L - r) + k_d(\mathbf{v}_1 - \mathbf{v}_2) \cdot \frac{\mathbf{L}}{L}\right) \frac{\mathbf{L}}{L}$$

$$\mathbf{f}_2 = -\mathbf{f}_1$$

Force and Torque

Force and Torque

- ❖ Force
 - ❖ causes linear acceleration
- ❖ Torque
 - ❖ causes angular acceleration
- ❖ Torque: τ
 - ❖ a vector
 - ❖ magnitude
 - ❖ how quickly the angular velocity is changed
 - ❖ $|\mathbf{r} \times \mathbf{f}|$
 - ❖ direction
 - ❖ rotational axis = $(\mathbf{r} \times \mathbf{f}) / |\mathbf{r} \times \mathbf{f}|$



$$\tau = \mathbf{r} \times \mathbf{f}$$