An Introduction to Physics-based Animation

Adam Bargteil
University of Maryland, Baltimore County

Tamar Shinar
University of California, Riverside
I. A Simple Start: Particle Dynamics
Let’s jump right in and consider the problem of tracing a particle through a velocity field.
\[ [x,y] = \text{meshgrid}(0:.5:2\pi,0:.5:2\pi); \]

\[ \text{quiver}(x,y,x,5\cos(x)) \]
```matlab
octave:1>
octave:1>
[x,y] = meshgrid(0:.5:2*pi,0:.5:2*pi);
quiver(x,y,x,5*cos(x))
```
```matlab
[x,y] = meshgrid(0:.5:2*pi,0:.5:2*pi);
quiver(x,y,x,5*cos(x))
```
\[ \text{x, y} = \text{meshgrid}(0:0.5:2\pi, 0:0.5:2\pi); \]
\[ \text{quiver}(\text{x}, \text{y}, \text{x}, 5\cos(\text{x})) \]
Initial Value Problem

\[ x_p(0) = x_0 \]

\[ \frac{d x_p(t)}{d t} = v(x_p, t) \]
Initial Value Problem

\[ x_p(0) = x_0 \]

\[ \frac{d x_p(t)}{d t} = v(x_p, t) \]
Initial Value Problem

\[ x_p(0) = x_0 \]
\[ \frac{dx_p(t)}{dt} = v(x_p, t) \]
Initial Value Problem

\[ x_p(0) = x_0 \]

\[ \frac{dx_p(t)}{dt} = v(x_p, t) \]
Initial Value Problem

\[ x_p(0) = x_0 \]

\[ \frac{dx_p(t)}{dt} = v(x_p, t) \]

Change
Initial Value Problem

\[ x_p(0) = x_0 \]

\[ \frac{dx_p(t)}{dt} = v(x_p, t) \]

Change
Initial Value Problem

\[ x_p(0) = x_0 \]

\[ \frac{d x_p(t)}{d t} = v(x_p, t) \]

Change  \rightarrow  Difference
Initial Value Problem

\[ x_p(0) = x_0 \]

\[ \frac{dx_p(t)}{dt} = v(x_p, t) \]

Change \quad \rightarrow \quad Difference

Differential Equation
Initial Value Problem

\[ x_p(0) = x_0 \]

\[ \frac{d}{dt} x_p(t) = v(x_p, t) \]

Change \quad \rightarrow \quad Difference

Ordinary Differential Equation
Initial Value Problem

\[ x_p(0) = x_0 \]

\[ \frac{dx_p(t)}{dt} = v(x_p, t) \]

Change → Difference

First-order Ordinary Differential Equation
Initial Value Problem

\[ x_p(0) = x_0 \]

\[ \frac{dx_p(t)}{dt} = v(x_p, t) \]
Initial Value Problem

\[ x_p(0) = x_0 \]

\[ \frac{d x_p(t)}{d t} = v(x_p, t) \]

Simple
Initial Value Problem

\[ x_p(0) = x_0 \]

\[ \frac{d x_p(t)}{dt} = v(x_p, t) \]

Simple

Powerful
Initial Value Problem

\[ x_p(0) = x_0 \]

\[ \frac{dx_p(t)}{dt} = v(x_p, t) \]

Simple

Powerful

Instructive
Euler’s Method
The Derivative

\[ \frac{dx_p(t)}{dt} = \lim_{\epsilon \to 0} \frac{x_p(t + \epsilon) - x_p(t)}{\epsilon} \]
The Derivative

\[ \frac{dx_p(t)}{dt} = \lim_{\epsilon \to 0} \frac{x_p(t + \epsilon) - x_p(t)}{\epsilon} \]
The Derivative

\[
\frac{d\mathbf{x}_p(t)}{dt} = \lim_{\epsilon \to 0} \frac{\mathbf{x}_p(t + \epsilon) - \mathbf{x}_p(t)}{\epsilon}
\]
The Derivative

\[
\frac{dx_p(t)}{dt} = \lim_{\epsilon \to 0} \frac{x_p(t + \epsilon) - x_p(t)}{\epsilon}
\]
The Derivative

\[ \frac{d}{dt} x_p(t) = \lim_{\epsilon \to 0} \frac{x_p(t + \epsilon) - x_p(t)}{\epsilon} \]
The Derivative

\[
\frac{d\mathbf{x}_p(t)}{dt} = \lim_{\epsilon \to 0} \frac{\mathbf{x}_p(t + \epsilon) - \mathbf{x}_p(t)}{\epsilon}
\]
The Derivative

\[ \frac{dx_p(t)}{dt} = \lim_{\epsilon \to 0} \frac{x_p(t + \epsilon) - x_p(t)}{\epsilon} \]

\[ \epsilon \quad \rightarrow \quad \Delta t \]
The Derivative

\[ \frac{dx_p(t)}{dt} = \lim_{\epsilon \to 0} \frac{x_p(t + \epsilon) - x_p(t)}{\epsilon} \]

\[ \epsilon \quad \rightarrow \quad \Delta t \]

\[ \frac{dx_p(t)}{dt} \approx \frac{x_p(t + \Delta t) - x_p(t)}{\Delta t} \]
Euler’s Method

\[
\frac{dx_p(t)}{dt} \approx \frac{x_p(t + \Delta t) - x_p(t)}{\Delta t}
\]
Euler’s Method

\[
\frac{dx_p(t)}{dt} \approx \frac{x_p(t + \Delta t) - x_p(t)}{\Delta t} + \frac{dx_p(t)}{dt} = v(x_p, t)
\]
Euler’s Method

\[
\frac{dx_p(t)}{dt} \approx \frac{x_p(t + \Delta t) - x_p(t)}{\Delta t}
\]

\[
\frac{dx_p(t)}{dt} + \frac{dx_p(t)}{dt} = v(x_p, t)
\]

\[
\frac{x_p(t + \Delta t) - x_p(t)}{\Delta t} = v(x_p, t)
\]
Euler’s Method

\[
\frac{dx_p(t)}{dt} \approx \frac{x_p(t + \Delta t) - x_p(t)}{\Delta t}
\]

\[
\frac{dx_p(t)}{dt} = v(x_p, t)
\]

\[
\frac{x_p(t + \Delta t) - x_p(t)}{\Delta t} = v(x_p, t)
\]

\[
x_p(t + \Delta t) = x_p(t) + \Delta t \cdot v(x_p, t)
\]
Euler’s Method

\[
\frac{dx_p(t)}{dt} \approx \frac{x_p(t + \Delta t) - x_p(t)}{\Delta t} + \frac{dx_p(t)}{dt} = v(x_p, t)
\]

\[
\frac{x_p(t + \Delta t) - x_p(t)}{\Delta t} = v(x_p, t)
\]

\[
x_p(t + \Delta t) = x_p(t) + \Delta t \cdot v(x_p, t)
\]
Euler’s Method

\[
\frac{dx_p(t)}{dt} \approx \frac{x_p(t + \Delta t) - x_p(t)}{\Delta t}
\]

\[
\frac{dx_p(t)}{dt} + \frac{dx_p(t)}{dt} = \mathbf{v}(x_p, t)
\]

\[
\frac{x_p(t + \Delta t) - x_p(t)}{\Delta t} = \mathbf{v}(x_p, t)
\]

\[
x_p(t + \Delta t) = x_p(t) + \Delta t \cdot \mathbf{v}(x_p, t)
\]
Euler’s Method

\[
\frac{d\mathbf{x}_p(t)}{dt} \approx \frac{\mathbf{x}_p(t + \Delta t) - \mathbf{x}_p(t)}{\Delta t} + \frac{d\mathbf{x}_p(t)}{dt} = \mathbf{v}(\mathbf{x}_p, t)
\]

\[
\frac{\mathbf{x}_p(t + \Delta t) - \mathbf{x}_p(t)}{\Delta t} = \mathbf{v}(\mathbf{x}_p, t)
\]

\[
\mathbf{x}_p(t + \Delta t) = \mathbf{x}_p(t) + \Delta t \cdot \mathbf{v}(\mathbf{x}_p, t)
\]
The Great Tradeoff

\[
\frac{dx_p(t)}{dt} \approx \frac{x_p(t + \Delta t) - x_p(t)}{\Delta t}
\]
The Great Tradeoff

\[
\frac{d x_p(t)}{d t} \approx \frac{x_p(t + \Delta t) - x_p(t)}{\Delta t}
\]

As \( \Delta t \) decreases
The Great Tradeoff

\[ \frac{dx_p(t)}{dt} \approx \frac{x_p(t + \Delta t) - x_p(t)}{\Delta t} \]

As \( \Delta t \) decreases
the approximation gets better
The Great Tradeoff

\[
\frac{dx_p(t)}{dt} \approx \frac{x_p(t + \Delta t) - x_p(t)}{\Delta t}
\]

As \( \Delta t \) decreases
the approximation gets better
but
The Great Tradeoff

\[ \frac{dx_p(t)}{dt} \approx \frac{x_p(t + \Delta t) - x_p(t)}{\Delta t} \]

As \( \Delta t \) decreases,
the approximation gets better
but
the computational cost increases
Let’s consider another problem
In the real world
\[ f = ma \]
Another Initial Value Problem

\[ x_p(0) = x_0 \]

\[ \frac{d^2 x_p(t)}{dt^2} = \frac{f(x_p, t)}{m_p} \]
Another Initial Value Problem

\[ x_p(0) = x_0 \]

\[ \frac{d^2x_p(t)}{dt^2} = \frac{f(x_p, t)}{m_p} \]

Second-order Ordinary Differential Equation
Another Initial Value Problem

\[ x_p(0) = x_0 \]
\[ v_p(0) = v_0 \]
\[ \frac{dx_p(t)}{dt} = v(x_p, t) \]
\[ \frac{dv_p(t)}{dt} = \frac{f(x_p, t)}{m_p} \]
Another Initial Value Problem

\[ x_p(0) = x_0 \]
\[ v_p(0) = v_0 \]

\[ \frac{dx_p(t)}{dt} = v(x_p, t) \]
\[ \frac{dv_p(t)}{dt} = \frac{f(x_p, t)}{m_p} \]

Coupled First-order Ordinary Differential Equations
Euler’s Method (Again)

\[ \mathbf{v}_p(t + \Delta t) = \mathbf{v}_p(t) + \Delta t \cdot \frac{\mathbf{f}(\mathbf{x}_p, t)}{m_p} \]

\[ \mathbf{x}_p(t + \Delta t) = \mathbf{x}_p(t) + \Delta t \cdot \mathbf{v}_p(t + \Delta t) \]
Euler’s Method (Again)

\[ \mathbf{v}_p(t + \Delta t) = \mathbf{v}_p(t) + \Delta t \cdot \frac{\mathbf{f}(x_p, t)}{m_p} \]

\[ x_p(t + \Delta t) = x_p(t) + \Delta t \cdot \mathbf{v}_p(t + \Delta t) \]

Symplectic Euler
struct Particle {
    double mass;
    Eigen::Vector3d pos, vel, frc;
};

foreach (p : particles) {
    p.frc = 0.0;
}

foreach (f : forces) {
    foreach (p : forces.affectedParticles) {
        p.frc += f.computeForce(p);
    }
}

foreach (p : particle) {
    p.vel += dt * p.frc / p.mass;
    p.pos += dt * p.vel;
}
Check our Karl Sim’s *Particle Dreams*
Let’s Add Springs!
Springs

\[ f = -kd \]
Springs

\[ f_p = -kx_p \]
Springs

\[ f_p = -kx_p \]

non-zero rest length?
Springs

\[ \mathbf{f}_p = -k \mathbf{x}_p \]

\[ \mathbf{f}_p = -k (\| \mathbf{x}_p \|) \frac{\mathbf{x}_p}{\| \mathbf{x}_p \|} \]
Springs

\[ \mathbf{f}_p = -k \mathbf{x}_p \]

\[ \mathbf{f}_p = -k \left( \| \mathbf{x}_p \| \right) \frac{\mathbf{x}_p}{\| \mathbf{x}_p \|} \]

\[ \mathbf{f}_p = -k \left( \| \mathbf{x}_p \| - r \right) \frac{\mathbf{x}_p}{\| \mathbf{x}_p \|} \]
Springs

\[ f_p = -k \left( \| x_p \| - r \right) \frac{x_p}{\| x_p \|} \]
Springs

\[ f_p = -k (\|x_p\| - r) \frac{x_p}{\|x_p\|} \]

Strain \( \left( \frac{\|x_p\|}{r} - 1 \right) \)
Springs

\[ f_p = -k \left( \|x_p\| - r \right) \frac{x_p}{\|x_p\|} \]

Strain \[ \left( \frac{\|x_p\|}{r} - 1 \right) \]
Springs

\[ f_p = -k \left( \frac{\|x_p\|}{r} - 1 \right) \frac{x_p}{\|x_p\|} \]
Springs

\[ f_p = -k \left( \frac{\|x_p\|}{r} - 1 \right) \frac{x_p}{\|x_p\|} \]

arbitrary connection?
Springs

\[ f_p = -k \left( \frac{\|x_p\|}{r} - 1 \right) \frac{x_p}{\|x_p\|} \]

\[ f_p = k \left( \frac{\|x_q - x_p\|}{r} - 1 \right) \frac{x_q - x_p}{\|x_q - x_p\|} \]
Springs

\[ \mathbf{f}_p = -k \left( \frac{\| \mathbf{x}_p \|}{r} - 1 \right) \frac{\mathbf{x}_p}{\| \mathbf{x}_p \|} \]

\[ \mathbf{f}_p = k \left( \frac{\| \mathbf{x}_q - \mathbf{x}_p \|}{r} - 1 \right) \frac{\mathbf{x}_q - \mathbf{x}_p}{\| \mathbf{x}_q - \mathbf{x}_p \|} \]

\[ \mathbf{f}_q = -\mathbf{f}_p \]
Damping

\[ f_p = k \left( \frac{\|x_q - x_p\|}{r} - 1 \right) \frac{x_q - x_p}{\|x_q - x_p\|} \]
Damping

\[ \mathbf{f}_p = k \left( \frac{\left\| \mathbf{x}_q - \mathbf{x}_p \right\|}{r} - 1 \right) \frac{\mathbf{x}_q - \mathbf{x}_p}{\left\| \mathbf{x}_q - \mathbf{x}_p \right\|} \]

\[ \mathbf{f}_p = k_d \left( \frac{\mathbf{v}_q - \mathbf{v}_p}{r} \cdot \frac{\mathbf{x}_q - \mathbf{x}_p}{\left\| \mathbf{x}_q - \mathbf{x}_p \right\|} \right) \frac{\mathbf{x}_q - \mathbf{x}_p}{\left\| \mathbf{x}_q - \mathbf{x}_p \right\|} \]
Damping

\[ f_p = k \left( \frac{\|x_q - x_p\|}{r} - 1 \right) \frac{x_q - x_p}{\|x_q - x_p\|} \]

\[ f_p = k_d \left( \frac{v_q - v_p}{r} \cdot \frac{x_q - x_p}{\|x_q - x_p\|} \right) \frac{x_q - x_p}{\|x_q - x_p\|} \]

relative velocity
Damping

\[ f_p = k \left( \frac{\|x_q - x_p\|}{r} - 1 \right) \frac{x_q - x_p}{\|x_q - x_p\|} \]

\[ f_p = k_d \left( \frac{v_q - v_p}{r} \cdot \frac{x_q - x_p}{\|x_q - x_p\|} \right) \frac{x_q - x_p}{\|x_q - x_p\|} \]

\begin{align*}
\text{relative velocity} & \quad \text{spring direction} \\
\text{spring} & \quad \text{relative}
\end{align*}
Damping

\[ f_p = k \left( \frac{\|x_q - x_p\|}{r} - 1 \right) \frac{x_q - x_p}{\|x_q - x_p\|} \]

\[ f_p = k_d \left( \frac{v_q - v_p}{r} \cdot \frac{x_q - x_p}{\|x_q - x_p\|} \right) \frac{x_q - x_p}{\|x_q - x_p\|} \]

\( f_p = \left[ k_s \left( \frac{\|x_q - x_p\|}{r} - 1 \right) + k_d \left( \frac{(v_q - v_p) \cdot (x_q - x_p)}{r \|x_q - x_p\|} \right) \right] \frac{x_q - x_p}{\|x_q - x_p\|} \)
```cpp
foreach (p : particles) {
    p.frc = 0.0;
}

foreach (s : springs) {
    Eigen::Vector3d d = particles[s->j].pos - particles[s->i].pos;
    double l = d.norm();
    Eigen::Vector3d v = particles[s->j].vel - particles[s->i].vel;
    Eigen::Vector3d frc = (params.k_s*((l / s->r) - 1.0) +
              params.k_d*(v.dot(d)/(l*s->r))) * (d/l);
    particles[s.i].frc += frc
    particles[s.j].frc -= frc
}

foreach (p : particle) {
    p.vel += dt * p.frc / p.mass;
    p.pos += dt * p.vel;
}
```
Live Demo
II. Mathematical Models
Newton’s Laws
Newtonian Mechanics

- Published in *Principia*, 1687
- Includes three laws of motion:
  - inertia
  - \( f = ma \)
  - action/reaction
- Idealized particle or point mass
Newton’s First Law

A body persists at rest or in uniform motion in a straight line unless acted upon by a force

❖ Law of Inertia
❖ Defines an inertial frame of reference
Newton’s Second Law \((f = ma)\)

The rate of change of momentum of a body is directly proportional to the force applied to the body

\[
\frac{d}{dt}mv(t) = ma = f(t)
\]

❖ The basis for evolving a system of interacting particles
Newton’s Second Law \((f = ma)\)

- leads to a system of ODEs

\[
\dot{x}(t) = v(t) \\
\dot{v}(t) = a(t) = \frac{1}{m} f(t)
\]

\[
f(t) = f_e(t) + f_g(t) + f_s(t)
\]

\[
ma = f_e + f_g + f_s
\]
Newton’s Second Law \((f = ma)\)

- To model a system of particles,
- characterize all the forces on each particle
- Start with some initial conditions and apply \(f = ma\) to evolve in time

\[
ma = f_e + f_g + f_s
\]
Newton’s Third Law (Action/Reaction)

For every action, there is an equal and opposite reaction

- If body 1 applies force $f$ to body 2, then body 2 applies force $-f$ to body 1
Newton’s Third Law (Action/Reaction)

For every action, there is an equal and opposite reaction

- If body 1 applies force $\mathbf{f}$ to body 2, then body 2 applies force $-\mathbf{f}$ to body 1

- Example: Two particles connected by a spring force, equal/opposite pair of forces
Newton’s Third Law (Action/Reaction)

For every action, there is an equal and opposite reaction

- If body 1 applies force $\mathbf{f}$ to body 2, then body 2 applies force $-\mathbf{f}$ to body 1

- Example: In the collision of two particles, equal/opposite pair of forces prevents interpenetration
Newton’s Third Law (Action/Reaction)

For every action, there is an equal and opposite reaction

- If body 1 applies force $\mathbf{f}$ to body 2, then body 2 applies force $-\mathbf{f}$ to body 1

- Example: Particle resting on a surface, equal/opposite pair of forces prevents interpenetration
Newton’s Third Law (Action/Reaction)

For every action, there is an equal and opposite reaction

- If body 1 applies force $\mathbf{f}$ to body 2, then body 2 applies force $-\mathbf{f}$ to body 1

- Example: Particle resting on a surface, equal/opposite pair of forces prevents interpenetration
Alternative: Variational Mechanics

- Newtonian Mechanics is one formulation of classical mechanics
  - Based on vectors in Cartesian space
- Another set of approaches is called “variational” and is based on a principle of least action
  - Variational approaches let you use any set of coordinates
Conservation Laws
Conserved Quantities

- Cannot be created or destroyed!
- Includes mass, linear momentum, angular momentum, and energy
Conservation Laws

- Used in deriving evolution equations
- Inform choice of discrete approximation to continuous equations
- Implications for visual quality, numerical accuracy, and stability
Conservation of Mass

- Mass not created or destroyed (inexact)
- Mass naturally conserved in particle-based methods
  - Particles carry mass with them as they move
- Grid-based methods sometimes have issues with proper mass conservation
Conservation of Momentum

❖ By Newton’s second law, if there is no net force on a body, i.e., \( f = 0 \)

\[
\frac{d}{dt} m\mathbf{v} = 0
\]

\( \Rightarrow m\mathbf{v}(t) = \text{constant} \)

❖ So the momentum of the particle is conserved

❖ Similarly, if there is no net torque on a body, angular momentum is constant
Conservation of Momentum

- Newton’s third law equal/opposite also implies conservation of linear and angular momentum

\[
\frac{d}{dt} \mathbf{P}(t) = f_1 + (-f_1) = 0
\]
Conservation of Momentum

- Newton’s third law equal/opposite also implies conservation of linear and angular momentum

\[
\frac{d}{dt} \mathbf{L}(t) = \mathbf{r}_1 \times \mathbf{f}_1 + \mathbf{r}_2 \times (-\mathbf{f}_1) = (\mathbf{r}_1 - \mathbf{r}_2) \times \mathbf{f}_1 = 0
\]
Conservation of Momentum

Same holds for a collection of interacting particles!
Conservation of Energy

- Initial energy (potential)
  \[ mgh_0 \]
- Conservation of energy
  \[ \frac{1}{2}mv(t)^2 + mgh(t) = mgh_0 \]
- Speed when hits
  \[ v(t) = \sqrt{2gh_0} \]
Conservation of Energy

- Initial energy (potential)
  \[ mgh_0 \]
- Conservation of energy
  \[ \frac{1}{2}mv(t)^2 + mgh(t) = mgh_0 \]
- Speed when hits
  \[ v(t) = \sqrt{2gh_0} \]
Conservation of Energy

- Different numerical schemes have different energy conservation properties.
- In many schemes, energy grows or decays nonphysically,
  - instability (blow up), or
  - motion too damped.
Conservation Laws for Continua

- At large enough length scales, model matter as a continuum rather than set of discrete particles
Conservation Laws for Continua

- To derive evolution equations for continua, consider arbitrary control volume $\Omega$
Conservation Laws for Continua

- **Conservation of Mass**
  \[
  \frac{d}{dt} \int_{\Omega} \rho dV = - \int_{\partial\Omega} \rho \mathbf{v} \cdot \mathbf{n} dS
  \]

- **Continuity Equation**
  \[
  \Rightarrow \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0
  \]
Conservation Laws for Continua

- Local conservation law

\[ \frac{\partial u}{\partial t} + \nabla \cdot f(u) = 0 \]
Rigid Bodies
Rigid Body Idealization

- If deformation is negligible, a rigid body approximation is more efficient than a soft body model.
Rigid Body Idealization

- Elastic forces are replaced with constraints that particles in the body remain a fixed distance apart.
Rigid Body Idealization

- 3n DoF are replaced with 6 DoF! position and orientation $\mathbf{x}(t), \mathbf{R}(t)$
Rigid Body Kinematics

- Center of mass

\[ \mathbf{x}_{\text{com}} = \frac{\sum_{i=1}^{N} m_i \mathbf{x}_i}{\sum_{i=1}^{N} m_i}. \]
Rigid Body Coordinates

Object Space

World Space
Linear and Angular Velocity

- Linear velocity

$$\mathbf{v}(t) = \dot{\mathbf{x}}(t)$$
Linear and Angular Velocity

- Angular velocity

$$\omega(t)$$
Rigid Body Coordinates

particle position \( \mathbf{p}(t) = \mathbf{x}(t) + \mathbf{R}(t)\mathbf{r}_0 \)

particle velocity \( \dot{\mathbf{p}}(t) = \mathbf{v}(t) + \mathbf{\omega}(t) \times \mathbf{r}(t) \)
Linear and Angular Momentum

- Linear Momentum

\[ \mathbf{P}(t) = \sum_{i=1}^{N} m_i \mathbf{v}_i(t) \]

(c.o.m. origin) \( \Rightarrow \mathbf{P}(t) = m \mathbf{v}(t) \)

- Angular Momentum

\[ \mathbf{L}(t) = \sum_{i=1}^{N} \mathbf{r}_i(t) \times m_i \mathbf{v}_i(t) \]

(c.o.m. origin) \( \Rightarrow \mathbf{L}(t) = \mathbf{I}(t) \omega(t) \)

\[ \mathbf{I}(t) : \text{ inertia tensor} \]
Rigid Body Inertia Tensor

\[
\mathbf{I}(t) = \sum_{i=1}^{N} m_i \left( \mathbf{r}_i^T \mathbf{r}_i \delta - \mathbf{r}_i \mathbf{r}_i^T \right)
\]

\[
= \mathbf{R}(t) \sum_{i=1}^{N} m_i \left( \mathbf{r}_{0i}^T \mathbf{r}_{0i} \delta - \mathbf{r}_{0i} \mathbf{r}_{0i}^T \right) \mathbf{R}(t)^T
\]

\[
= \mathbf{R}(t) \mathbf{I}_0 \mathbf{R}(t)^T.
\]
Linear and Angular Momentum

- No net force =>
  - linear momentum and velocity constant
- No net torque =>
  - angular momentum constant
  - angular velocity not necessarily constant

\[ P(t) = m \mathbf{v}(t) \]
\[ L(t) = I(t) \omega(t) \]
Newton’s Second Law for Rigid Bodies

\[ \frac{d}{dt} \begin{pmatrix} \mathbf{P}(t) \\ \mathbf{L}(t) \end{pmatrix} = \begin{pmatrix} f(t) \\ \tau(t) \end{pmatrix}, \quad f(t) = \sum f_i \]
\[ \tau(t) = \sum \mathbf{r}_i \times f_i \]

- Summary

\[ \frac{d}{dt} \begin{pmatrix} \mathbf{x}(t) \\ \mathbf{R}(t) \\ \mathbf{P}(t) \\ \mathbf{L}(t) \end{pmatrix} = \begin{pmatrix} \mathbf{v}(t) \\ \omega^*(t) \mathbf{R}(t) \\ f(t) \\ \tau(t) \end{pmatrix} \]
Soft Bodies
Adding Elasticity and Damping

\[ f = ma \]

\[ K(x - u) + D(v) + Ma = f_{ext} \]

in terms of displacements \( d \)

\[ K(d) + D(\dot{d}) + M\ddot{d} = f_{ext} \]
Adding Elasticity and Damping

\[ m\mathbf{a} = \mathbf{f} \]

\[ \mathbf{K}(\mathbf{x} - \mathbf{u}) + \mathbf{D}(\mathbf{v}) + \mathbf{M}\ddot{\mathbf{a}} = \mathbf{f}_{ext} \]

in terms of displacements \( \mathbf{d} \)

\[ \mathbf{K}(\mathbf{d}) + \mathbf{D}(\dot{\mathbf{d}}) + \mathbf{M}\ddot{\mathbf{d}} = \mathbf{f}_{ext} \]
Adding Elasticity and Damping

\[ ma = f \]

\[ K(x - u) + D(v) + Ma = f_{ext} \]

in terms of displacements \(d\)

\[ K(d) + D(\dot{d}) + M\ddot{d} = f_{ext} \]
Adding Elasticity and Damping

\[ ma = f \]

\[ K(x - u) + D(v) + Ma = f_{\text{ext}} \]

in terms of displacements \( d \)

\[ K(d) + D(\dot{d}) + M\ddot{d} = f_{\text{ext}} \]
Adding Elasticity and Damping

\[ m a = f \]

\[ K(x - u) + D(v) + Ma = f_{ext} \]

in terms of displacements \( d \)

\[ K(d) + D(\dot{d}) + M\ddot{d} = f_{ext} \]
Adding Elasticity and Damping

\[ ma = f \]

\[ K(x - u) + D(v) + Ma = f_{ext} \]

in terms of displacements \( d \)

\[ K(d) + D(\dot{d}) + M\ddot{d} = f_{ext} \]
Generalization of spring stiffness, called stiffness matrix
Generalization of spring stiffness, called *stiffness matrix*

- Sparse, Symmetric, Diagonally Dominant, Row / Col sums 0
Generalization of spring stiffness, called stiffness matrix

- Sparse, Symmetric, Diagonally Dominant, Row/Col sums 0

- Positive semi-definite
Generalization of spring stiffness, called stiffness matrix

- Sparse, Symmetric, Diagonally Dominant, Row/Col sums 0
- Positive semi-definite
- Negative Jacobian of forces
Generalization of spring stiffness, called *stiffness matrix*

- Sparse, Symmetric, Diagonally Dominant, Row/Col sums 0
- Positive semi-definite
- Negative Jacobian of forces
- Negative Hessian of energy
Generalization of spring stiffness, called *stiffness* matrix

- Sparse, Symmetric, Diagonally Dominant, Row/Col sums 0
- Positive semi-definite
- Negative Jacobian of forces
- Negative Hessian of energy

\[ K = \frac{-\partial f_i}{\partial x_j} = \frac{-\partial \eta}{\partial x_i \partial x_j} \]
So how do we calculate elastic forces?
Elastic Forces Cookbook

- Deformation Function (world pos, rest position)
- Deformation Gradient (deformation function)
- Strain (deformation gradient)
- Stress (strain)
- Energy (stress, strain)
- Forces (energy)
Deformation and Its Gradient

$u_0$, $u_1$, $u_2$
Deformation and Its Gradient

\[ x(u) = x(u_0) + F(u - u_0) \]
Deformation and Its Gradient

\[ x(u) = x(u_0) + F(u - u_0) \]

\[ \frac{\partial x}{\partial u} = F \]
Deformation and Its Gradient

\[ x(u) = x(u_0) + F(u - u_0) \]

\[ \frac{\partial x}{\partial u} = F \]
Strain is
Strain is

- a measure of deformation
Strain is

- a measure of deformation
- dimensionless (i.e. has no units)
Strain is

- a measure of deformation
- dimensionless (i.e. has no units)
- a function of the deformation gradient
Strain Metrics

- Green’s finite  \[ \epsilon = \frac{1}{2} (F^T F - I) \]

- Cauchy’s infinitesimal  \[ \epsilon = \frac{1}{2} (F^T + F) - I \]

- Co-rotated  \[ \epsilon = \frac{1}{2} (\tilde{F}^T + \tilde{F}) - I \]

where  \[ F = Q \tilde{F} \]
Stress
Stress

- stress is a function of strain
Stress

- Stress is a function of strain.
- Stress has units of (Newton / meter$^2$).
Stress

- stress is a function of strain
- stress has units of (Newton / meter\(^2\))
- models of stress-strain relationships can be highly complex, especially with organic materials
Linear Stress-strain Relationships

General, linear

$$\sigma = C\varepsilon$$
Linear Stress-strain Relationships

General, linear isotropic material

\[ \sigma = C\varepsilon \]

\[ \sigma = \lambda \text{Tr} (\varepsilon) \mathbf{I} + 2\mu \varepsilon \]
Elastic Potential, Traction, Force

\[ \eta = \frac{1}{2} \sigma : \epsilon = \frac{1}{2} \sum_{i,j} \sigma_{ij} \epsilon_{ij} \]
Elastic Potential, Traction, Force

\[ \eta = \frac{1}{2} \sigma : \epsilon = \frac{1}{2} \sum_{i,j} \sigma_{ij} \epsilon_{ij} \]

\[ \tau = \sigma n \]
Elastic Potential, Traction, Force

\[ \eta = \frac{1}{2} \sigma : \epsilon = \frac{1}{2} \sum_{i,j} \sigma_{ij} \epsilon_{ij} \]

\[ \tau = \sigma \mathbf{n} \]

\[ \mathbf{f}_i = -\frac{\partial \eta}{\partial \mathbf{x}_i} \]
Elastic Potential, Traction, Force

\[ \eta = \frac{1}{2} \sigma : \epsilon = \frac{1}{2} \sum_{i,j} \sigma_{ij} \epsilon_{ij} \]

\[ \tau = \sigma n \]

\[ f_i = -\frac{\partial \eta}{\partial x_i} \]

\[ f = \int_{\partial R} \sigma n \, dS \]
Plasticity

\[ F = F_e F_p \]
Fluids
Fluids

- Take shape of container
- Can’t support shear stress
Fluids can support normal stress.
Navier-Stokes Equations

\[ \rho(u_t + u \cdot \nabla u) = -\nabla p + \mu \Delta u + f \]
\[ \nabla \cdot u = 0 \]

\( \rho \): density \hspace{1cm} \( p \): pressure \hspace{1cm} \( f \): forces

\( u \): velocity \hspace{1cm} \( \mu \): viscosity
Navier-Stokes Equations

\[ \rho \left( u_t + u \cdot \nabla u \right) = - \nabla p + \mu \Delta u + f \]

\[ \nabla \cdot u = 0 \]

\( \rho \): density  \quad \( p \): pressure  \quad \( f \): forces

\( u \): velocity  \quad \( \mu \): viscosity
Navier-Stokes Equations

\[
\rho \left( u_t + u \cdot \nabla u \right) = - \nabla p + \mu \Delta u + f
\]

\[\nabla \cdot u = 0\]

\(\rho\): density \quad \rho\): pressure \quad f\): forces

\(u\): velocity \quad \mu\): viscosity
Navier-Stokes Equations

\[ \rho \left( u_t + u \cdot \nabla u \right) = - \nabla p + \mu \Delta u + f \]

\[ \nabla \cdot u = 0 \]

\( \rho \): density  \( p \): pressure  \( f \): forces

\( u \): velocity  \( \mu \): viscosity
Navier-Stokes Equations

\[
\rho \left( u_t + u \cdot \nabla u \right) = - \nabla p + \mu \Delta u + f
\]

\[\nabla \cdot u = 0\]

\(\rho: \text{density}\)  \(p: \text{pressure}\)  \(f: \text{forces}\)

\(u: \text{velocity}\)  \(\mu: \text{viscosity}\)
Navier-Stokes Equations

\[
\rho \left( u_t + u \cdot \nabla u \right) = - \nabla p + \mu \Delta u + f
\]

\[\nabla \cdot u = 0\]

\(\rho: \) density \hspace{1cm} \(p: \) pressure \hspace{1cm} \(f: \) forces

\(u: \) velocity \hspace{1cm} \(\mu: \) viscosity
Navier-Stokes Equations

\[ \rho \left( u_t + u \cdot \nabla u \right) = -\nabla p + \mu \Delta u + f \]

\[ \nabla \cdot u = 0 \]

\( \rho \): density  \( p \): pressure  \( f \): forces

\( u \): velocity  \( \mu \): viscosity
Material Derivative

\[ \rho \left( u_t + u \cdot \nabla u \right) = -\nabla p + \mu \Delta u + f \]

\[ \nabla \cdot u = 0 \]
Material Derivative

\[ u(x, t) \]
Material Derivative

\[ a_p(t) = \frac{d}{dt} v_p(t) \]
\[ = \frac{d}{dt} u(x_p(t), t) \]
\[ = \left( \frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} \frac{dx_p}{dt} \right) \]
Material Derivative

\[ a_p(t) = \frac{d}{dt} v_p(t) \]

\[ = \frac{d}{dt} u(x_p(t), t) \]

\[ = \left( \frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} \frac{dx_p}{dt} \right) \]

\[ = u_t + u \cdot \nabla u \]

\[ \frac{Du}{Dt} = u_t + u \cdot \nabla u \]
Pressure Forces

\[ \rho \left( u_t + u \cdot \nabla u \right) = - \nabla p + \mu \Delta u + f \]

\[ \nabla \cdot u = 0 \]

- maintains fluid volume, resisting compression/expansion
- forces fluid from areas of high pressure to low pressure
Viscous Forces

\[ \rho \left( \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} \right) = -\nabla p + \mu \Delta \mathbf{u} + \mathbf{f} \]

\[ \nabla \cdot \mathbf{u} = 0 \]

- internal friction
- Laplacian measures difference from neighbors

<table>
<thead>
<tr>
<th>( \mu )</th>
<th>(mPa \cdot s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mu_{\text{air}} )</td>
<td>( 1.8 \times 10^{-2} )</td>
</tr>
<tr>
<td>( \mu_{\text{water}} )</td>
<td>1</td>
</tr>
<tr>
<td>( \mu_{\text{honey}} )</td>
<td>( 5 \times 10^3 )</td>
</tr>
</tbody>
</table>
Viscous Forces: Solid Boundaries

\[ \rho \left( u_t + u \cdot \nabla u \right) = - \nabla p + \mu \Delta u + f \]
\[ \nabla \cdot u = 0 \]

- no-slip boundary condition

\[ u(x, t) = V(x, t), \quad x \in \Gamma \]
External Forces

\[ m \underbrace{\left( u_t + u \cdot \nabla u \right)}_{\text{a}} = - \underbrace{\nabla p + \mu \Delta u + f}_{\text{f}} \]

\[ \nabla \cdot u = 0 \]

- Gravity, surface tension, interaction forces, control forces, embedded structures
Incompressibility

\[ \rho (u_t + u \cdot \nabla u) = -\nabla p + \mu \Delta u + f \]

\[ \nabla \cdot u = 0 \]

- Conservation of mass
  \[ \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho u) = 0 \]

- Constant density
  \[ \Rightarrow \nabla \cdot u = 0 \]
Incompressibility

\[\rho \left( u_t + u \cdot \nabla u \right) = - \nabla p + \mu \Delta u + f\]

\[\nabla \cdot u = 0\]

- Net flow through boundary must be zero

\[0 = \int_{\Omega} \nabla \cdot u = \int_{\partial \Omega} u \cdot n\]
Incompressibility

\[
\rho \left( u_t + u \cdot \nabla u \right) = - \nabla p + \mu \Delta u + f
\]

\[
\nabla \cdot u = 0
\]

- Net flow through boundary must be zero

\[
0 = \int_{\Omega} \nabla \cdot u = \int_{\partial \Omega} u \cdot n
\]
III. Spatial Discretization
Lagrangian vs. Eulerian
Reference Frames

- An Eulerian reference frame is fixed.
- A Lagrangian reference frame moves with the material.
Reference Frames

Eulerian

Lagrangian
Reference Frames

Eulerian

Lagrangian
Reference Frames

Eulerian

\[ \frac{\partial y}{\partial t} + \mathbf{u} \cdot \nabla y \]

Lagrangian

\[ \frac{dy}{dt} \]
Grids, Meshes, and Particles
# Regular Grids

<table>
<thead>
<tr>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>✗ Simple</td>
<td>✗ Difficult to track shape over time</td>
</tr>
<tr>
<td>✗ Fast operations (e.g. point location)</td>
<td>✗ Difficult to handle non-grid-aligned boundaries</td>
</tr>
<tr>
<td>✗ Can take advantage of structure for efficiency</td>
<td></td>
</tr>
</tbody>
</table>
Staggered Grid
## Meshes (Simplicial Complexes)

<table>
<thead>
<tr>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Easy to map to previous points in time</td>
<td>- Difficult to generate meshes</td>
</tr>
<tr>
<td>- Can conform to boundaries</td>
<td>- Difficult to perform some operations (e.g. point location)</td>
</tr>
</tbody>
</table>
# Particles

<table>
<thead>
<tr>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple</td>
<td>Difficult to perform integration because they don’t partition space</td>
</tr>
<tr>
<td>Easy to map to previous points in time</td>
<td></td>
</tr>
</tbody>
</table>
Hybrid Structures

Advantages

❖ Advantages of underlying structures

Disadvantages

❖ Complexity
❖ Computational and accuracy costs from mapping between structures
Interpolation
Interpolation

- Samples stored at discrete points on grid, mesh, or particles
- Elsewhere, must be interpolated there
Interpolation

- Example: semi-Lagrangian advection
Interpolation

- Example: semi-Lagrangian advection
Example: semi-Lagrangian advection
Interpolation

- Example: semi-Lagrangian advection
Linear Interpolation (1D)

- $f(0) = f_1$
- $f(1) = f_2$
- Weights sum to 1
- Weight is length of opposite point

$$f(t) = (1 - t)f_1 + tf_2.$$
Bilinear Interpolation (2D)

- \( f(0, 0) = f_1, \ldots \)
- \( f(0, 1) = f_4 \)
- weights sum to 1
- weight is area
  opposite point

\[
\begin{align*}
f(s, t) &= s(1-t)f_1 + t(1-s)f_2 + (1-s)(1-t)f_3 + stf_4
\end{align*}
\]
Bilinear Interpolation (2D)

\[ f(s, t) = (1 - s)(1 - t)f_1 + s(1 - t)f_2 + st f_3 + (1 - s)t f_4 \]
Bilinear Interpolation (2D)

\[ f(s, t) = (1 - s)(1 - t)f_1 + s(1 - t)f_2 + stf_3 + (1 - s)t f_4 \]
Bilinear Interpolation (2D)

\[ f(s, t) = (1 - s)(1 - t)f_1 + s(1 - t)f_2 + stf_3 + (1 - s)t f_4 \]
Bilinear Interpolation (2D)

\[ f_5 = (1 - t)f_1 + tf_4 \quad f_6 = (1 - t)f_2 + tf_3 \]

\[ f(s, t) = (1 - s)(1 - t)f_1 + s(1 - t)f_2 + stf_3 + (1 - s)tf_4 \]
Trilinear Interpolation (3D)

\[ f(s, t, u) = 
(1 - s)(1 - t)(1 - u)f_1 
+ s(1 - t)(1 - u)f_2 
+ st(1 - u)f_3 
+ (1 - s)t(1 - u)f_4 
+ (1 - s)(1 - t)uf_5 
+ s(1 - t)uf_6 
+ stuf_7 
+ (1 - s)tuf_8 \]
Trilinear Interpolation (3D)

\[ f(s, t, u) = \\
(1 - s)(1 - t)(1 - u)f_1 \\
+ s(1 - t)(1 - u)f_2 \\
+ st(1 - u)f_3 \\
+ (1 - s)t(1 - u)f_4 \\
+ (1 - s)(1 - t)uf_5 \\
+ s(1 - t)uf_6 \\
+ stu f_7 \\
+ (1 - s)tuf_8 \]
Trilinear Interpolation (3D)

\[ f(s, t, u) = 
(1 - s)(1 - t)(1 - u)f_1 
+ s(1 - t)(1 - u)f_2 
+ st(1 - u)f_3 
+ (1 - s)t(1 - u)f_4 
+ (1 - s)(1 - t)uf_5 
+ s(1 - t)uf_6 
+ stu f_7 
+ (1 - s)tuf_8 \]
Barycentric Coordinates

\[ f_p = f(\alpha, \beta, \gamma) = \alpha f_a + \beta f_b + \gamma f_c \]
Barycentric Coordinates

\[ f_p = f(\alpha, \beta, \gamma) = \alpha f_a + \beta f_b + \gamma f_c \]

\[ \alpha = \frac{\text{area}(p, b, c)}{\text{area}(a, b, c)}, \]
\[ \beta = \frac{\text{area}(p, c, a)}{\text{area}(a, b, c)}, \]
\[ \gamma = \frac{\text{area}(p, a, b)}{\text{area}(a, b, c)}. \]
Barycentric Coordinates

\[ f_p = f(\alpha, \beta, \gamma) = \alpha f_a + \beta f_b + \gamma f_c \]

\[
\alpha = \frac{\text{area}(p, b, c)}{\text{area}(a, b, c)},
\]

\[
\beta = \frac{\text{area}(p, c, a)}{\text{area}(a, b, c)},
\]

\[
\gamma = \frac{\text{area}(p, a, b)}{\text{area}(a, b, c)}.
\]
Barycentric Coordinates

\[ f_p = f(\alpha, \beta, \gamma) = \alpha f_a + \beta f_b + \gamma f_c \]

\[ \alpha = \frac{\text{area}(p, b, c)}{\text{area}(a, b, c)}, \]

\[ \beta = \frac{\text{area}(p, c, a)}{\text{area}(a, b, c)}, \]

\[ \gamma = \frac{\text{area}(p, a, b)}{\text{area}(a, b, c)}. \]
Barycentric Coordinates

\[ f_p = f(\alpha, \beta, \gamma) = \alpha f_a + \beta f_b + \gamma f_c \]

\[ \alpha = \frac{\text{area}(p, b, c)}{\text{area}(a, b, c)} , \]
\[ \beta = \frac{\text{area}(p, c, a)}{\text{area}(a, b, c)} , \]
\[ \gamma = \frac{\text{area}(p, a, b)}{\text{area}(a, b, c)} . \]
Barycentric Coordinates

\[ f_p = f(\alpha, \beta, \gamma) = \alpha f_a + \beta f_b + \gamma f_c \]

\[ \alpha = \frac{\text{area}(p, b, c)}{\text{area}(a, b, c)}, \]

\[ \beta = \frac{\text{area}(p, c, a)}{\text{area}(a, b, c)}, \]

\[ \gamma = \frac{\text{area}(p, a, b)}{\text{area}(a, b, c)}. \]
Barycentric Coordinates

- Coordinates at a vertex

\[ \alpha = 1 \]
\[ \beta = 0 \]
\[ \gamma = 0 \]
Barycentric Coordinates

- Coordinates at a vertex
  \[ \alpha = 0 \]
  \[ \beta = 1 \]
  \[ \gamma = 0 \]
Barycentric Coordinates

- Coordinates at a vertex

\[ \alpha = 0 \]
\[ \beta = 0 \]
\[ \gamma = 1 \]
Barycentric Coordinates

- Coordinates on edge

\[ \alpha = 1 - t \]
\[ \beta = t \]
\[ \gamma = 0 \]
Barycentric Coordinates

- Coordinates on edge

\[ \alpha = 0 \]
\[ \beta = 1 - t \]
\[ \gamma = t \]
Barycentric Coordinates

- Coordinates on edge
  \[ \alpha = t \]
  \[ \beta = 0 \]
  \[ \gamma = 1 - t \]
Barycentric Coordinates

- Inside/outside test

\[ \alpha > 0, \beta > 0, \gamma > 0 \]
Barycentric Coordinates

- Inside/outside test

$$\alpha < 0, \beta > 0, \gamma > 0$$
Barycentric Coordinates

- Inside/outside test

\[ \alpha < 0, \beta > 0, \gamma > 0 \]
Polynomial Interpolation

\[ f(t) = at + b \]
Polynomial Interpolation

\[ f(t) = at^2 + bt + c \]
Polynomial Interpolation

- Lagrange interpolation
- Newton interpolation
- Same polynomial
- Different cost of construction and evaluation
Least Squares Approximation

- Approximate noisy or overdetermined data
Least Squares Approximation

- Given $m$ data points
  
  $$\left( x_1, f_1 \right), \left( x_2, f_2 \right), \ldots, \left( x_m, f_m \right)$$

- Given basis
  
  $$\phi_1(x), \ldots, \phi_n(x)$$

- Find coefficients $\alpha_1, \ldots, \alpha_n$

  $$\phi(x) = \alpha_1 \phi_1(x) + \ldots + \alpha_n \phi_n(x)$$
Least Squares Approximation

\((x_1, f_1), (x_2, f_2), \ldots, (x_m, f_m)\)

\(\phi(x) = \alpha_1 \phi_1(x) + \ldots + \alpha_n \phi_n(x)\)

- minimize sum of squared errors

\[\sum_{i=1}^{m} |\phi(x_i) - f_i|^2\]
Least Squares Approximation

\[ \text{argmin}_\alpha \| A\alpha - f \|_2^2 \]

- Normal equations

\[ A^T A\alpha = A^T f \]
Least Squares Approximation

- Weighted least squares
  \[ \sum_{i=1}^{m} w_i |\phi(x_i) - f_i|^2 \]

- Regularized least squares
  \[ \text{argmin}_\alpha \| A\alpha - f \|_2^2 + \| \Gamma\alpha \|_2^2 \]
Other Methods

- Bezier curves, splines
- Harmonic coordinates, mean value coordinates, Green coordinates
Finite Differences
Finite Difference Methods

- Used to discretize spatial derivatives

\[ \rho \left( u_t + u \cdot \nabla u \right) = -\nabla p + \mu \Delta u + f \]
\[ \nabla \cdot u = 0 \]
Finite Difference Methods

- Recall derivative of function
  \[
  \frac{d}{dx} f(x) = \lim_{h \to 0} \frac{f(x + h) - f(x)}{h}
  \]

- Finite difference approximation
  \[
  \frac{d}{dx} f(x) \approx \frac{f(x + h) - f(x)}{h}
  \]
  forward difference
Finite Difference Methods

\[ f'(x) \approx \frac{f(x + h) - f(x)}{h} \]

- Error decreases with decreasing \( h \)
- Taylor series

\[ f(x + h) = f(x) + hf'(x) + \frac{h^2}{2} f''(x) + \ldots + \frac{h^n}{n!} f^{(n)}(x) + \ldots \]

- Rearrange

\[ \frac{f(x + h) - f(x)}{h} = f'(x) + \frac{h}{2} f''(x) + \ldots \]
Finite Difference Methods

\[ f'(x) \approx \frac{f(x + h) - f(x)}{h} \]

- Error decreases with decreasing \( h \)
- Taylor series
  \[ f(x + h) = f(x) + hf'(x) + \frac{h^2}{2} f''(x) + \ldots + \frac{h^n}{n!} f^{(n)}(x) + \ldots \]
- Rearrange
  \[ \frac{f(x + h) - f(x)}{h} = f'(x) + O(h) \]

first order accurate
Forward and Backward Difference

- **Forward difference**
  \[ f'(x) \approx \frac{f(x + h) - f(x)}{h} \]

- **Backward difference**
  \[ f'(x) \approx \frac{f(x) - f(x - h)}{h} \]
Forward and Backward Difference

- Forward difference
  \[ f'(x) \approx \frac{f(x + h) - f(x)}{h} \]

- Backward difference
  \[ f'(x) \approx \frac{f(x) - f(x - h)}{h} \]
Central Difference Approximation

\[ f'(x) \approx \frac{f(x + h) - f(x - h)}{2h} \]

- Taylor series

\[
f(x + h) = f(x) + hf'(x) + \frac{h^2}{2} f''(x) + O(h^3),
\]

\[
f(x - h) = f(x) - hf'(x) + \frac{h^2}{2} f''(x) - O(h^3).
\]

- Combine, rearrange

\[
\frac{f(x + h) - f(x - h)}{2h} = f'(x) + O(h^2)
\]
Discretization Error

- **Forward difference**

\[
\frac{f(x + h) - f(x)}{h} = f'(x) + \frac{h}{2} f''(x) + \ldots
\]

- **Central difference**

\[
\frac{f(x + h) - f(x - h)}{2h} = f'(x) + \frac{h^2}{6} f'''(x) + \ldots
\]
Higher Derivatives

\[ f''(x) \approx \frac{f'(x + \frac{h}{2}) - f'(x - \frac{h}{2})}{h} \]

\[ \approx \frac{\frac{f(x+h) - f(x)}{h} - \frac{f(x) - f(x-h)}{h}}{h} \]

\[ \approx \frac{f(x + h) - 2f(x) + f(x - h)}{h^2} \]
Higher Derivatives

\[ f''(x) \approx \frac{f'(x + \frac{h}{2}) - f'(x - \frac{h}{2})}{\frac{h}{2}} \]

\[ \approx \frac{\frac{f(x+h)-f(x)}{h} - \frac{f(x)-f(x-h)}{h}}{h} \]

\[ \approx \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} \]
Higher Derivatives

\[
\frac{f''(x)}{h} \approx \frac{f'(x + \frac{h}{2}) - f'(x - \frac{h}{2})}{h} = \frac{f(x + h) - f(x)}{h} - \frac{f(x) - f(x - h)}{h} = \frac{f(x + h) - 2f(x) + f(x - h)}{h^2}
\]

3-point stencil
Laplacian Operator

\[ \rho (u_t + u \cdot \nabla u) = -\nabla p + \mu \Delta u + f \]
\[ \nabla \cdot u = 0 \]

- In 2D \[ \Delta u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \]

\[
\frac{\partial^2 u(x, y)}{\partial x^2} \approx \frac{u(x + h, y) - 2u(x, y) + u(x - h, y)}{h^2} \\
\frac{\partial^2 u(x, y)}{\partial y^2} \approx \frac{u(x, y + h) - 2u(x, y) + u(x, y - h)}{h^2}
\]
Laplacian Operator

\[ \Delta u \approx \frac{u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j}}{h^2} \]

5 - point stencil
Poisson Equation (1D)

\[ u_{xx} = f, \quad x \in \Omega \]
\[ u(x) = \bar{u}(x), \quad x \in \partial \Omega \]

- At each interior node
  \[ \frac{u_{i-1} - 2u_i + u_{i+1}}{h^2} = f_i \]
- Boundary nodes
  \[ u_0 = \bar{u}(x_0), \quad u_7 = \bar{u}(x_7) \]
Poisson Equation (1D)

\[ u_{xx} = f, \quad x \in \Omega \]

\[ u(x) = \bar{u}(x), \quad x \in \partial \Omega \]

linear system

\[
\begin{pmatrix}
-2 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & -2 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & -2 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & -2 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & -2 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & -2 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & -2
\end{pmatrix}
\begin{pmatrix}
u_1 \\
u_2 \\
u_3 \\
u_4 \\
u_5 \\
u_6 \\
u_7
\end{pmatrix}
=
\begin{pmatrix}
f_1 - \frac{\bar{u}(x_0)}{h^2} \\
f_2 \\
f_3 \\
f_4 \\
f_5 \\
f_6 - \frac{\bar{u}(x_7)}{h^2}
\end{pmatrix}
\]
Finite Elements
Finite Elements

- Discretize the space of representable functions, instead of discretizing derivatives

Method:

- Discretize space into a finite set of elements
- Choose a set of basis functions over the elements (e.g. piecewise linear)
- *Galerkin* projection onto these basis functions
- Solve the problem
Let's try an example:
Linear Finite Elements for Elasticity
Choose Element Type: Triangle
Choose Basis Functions: Piecewise Linear
Project Deformation Function onto the Piecewise Linear Function Space
Project Deformation Function onto the Piecewise Linear Function Space

\[ x(u) = x(u_0) + F(u - u_0) \]
Solve the Problem
Solve the Problem

We know we can compute forces from $\mathbf{F}$
Solve the Problem

We know we can compute forces from $F$

But, how do we compute $F$?

$$( \text{in } x(u) = x(u_0) + F(u - u_0) )$$
\[ u = u_0 + \alpha (u_1 - u_0) + \beta (u_2 - u_0) \]
\[ u = u_0 + \alpha (u_1 - u_0) + \beta (u_2 - u_0) \]

\[ u = u_0 + (u_{10} \quad u_{20}) \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \]
\[ \mathbf{x} = \mathbf{x}_0 + \alpha (\mathbf{x}_1 - \mathbf{x}_0) + \beta (\mathbf{x}_2 - \mathbf{x}_0) \]
\[ x = x_0 + \alpha (x_1 - x_0) + \beta (x_2 - x_0) \]

\[ x = x_0 + \begin{pmatrix} x_{10} & x_{20} \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \]
Rest/Material

\[ \mathbf{u} = \mathbf{u}_0 + (\mathbf{u}_{10} \quad \mathbf{u}_{20}) \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \]

World

\[ \mathbf{x} = \mathbf{x}_0 + (\mathbf{x}_{10} \quad \mathbf{x}_{20}) \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \]
\[ u = u_0 + (u_{10} \ u_{20}) \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \]

\[ x = x_0 + (x_{10} \ x_{20}) \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \]

\[ x(u) = x_0 + (x_{10} \ x_{20}) \begin{pmatrix} u_{10} \\ u_{20} \end{pmatrix}^{-1} (u - u_0) \]
\[ u = u_0 + (u_{10} \quad u_{20}) \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \quad \text{and} \quad x = x_0 + (x_{10} \quad x_{20}) \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \]

\[ x(u) = x_0 + (x_{10} \quad x_{20}) (u_{10} \quad u_{20})^{-1} (u - u_0) \]

\[ F = (x_{10} \quad x_{20}) (u_{10} \quad u_{20})^{-1} \]
$F = \begin{pmatrix} x_{10} & x_{20} \end{pmatrix} \begin{pmatrix} u_{10} & u_{20} \end{pmatrix}^{-1}$
Solve the Problem

\[ \epsilon = \frac{1}{2} \left( \tilde{F}^T + \tilde{F} \right) - I \]

where \( F = Q \tilde{F} \)
Solve the Problem

\[ \epsilon = \frac{1}{2} \left( \tilde{F}^T + \tilde{F} \right) - I \]

where \( F = Q \tilde{F} \)

\[ \sigma = \lambda \text{Tr} (\epsilon) I + 2\mu \epsilon \]
Solve the Problem

\[ \epsilon = \frac{1}{2} \left( \tilde{F}^T + \tilde{F} \right) - \mathbf{I} \]

where \( \mathbf{F} = Q \tilde{F} \)

\[ \sigma = \lambda \text{Tr} (\epsilon) \mathbf{I} + 2\mu \epsilon \]

\[ \mathbf{f} = Q \sigma \mathbf{n}_i \]
Solve the Problem

\[ \epsilon = \frac{1}{2} \left( \tilde{F}^T + \tilde{F} \right) - \mathbf{I} \]

where \( \mathbf{F} = \mathbf{Q} \tilde{\mathbf{F}} \)

\[ \sigma = \lambda \text{Tr} ( \epsilon ) \mathbf{I} + 2 \mu \epsilon \]

\[ \mathbf{f} = \mathbf{Q} \sigma \mathbf{n}_i \]

where \( \mathbf{n}_i \)
are the normals of the opposite faces
in rest space
IV. Temporal Integration
Explicit Integration

Explicit formula for \((t+1)\)
in terms of quantities known at time \(t\)

\[ \mathbf{x}_p(t + \Delta t) = \mathbf{x}_p(t) + \Delta t \cdot \mathbf{v}(\mathbf{x}_p, t) \]

Note: everything on the right hand side is evaluated at time \(t\)
Choose Your Integration Scheme Wisely
Trapezoidal Rule vs. Midpoint Method

Forward Euler

Trapezoidal Rule

Midpoint Method
Trapazoidal Rule
```octave
[x,y] = meshgrid(0:.5:2*pi,0:.5:2*pi);
quiver(x,y,x,5*cos(x))
```
[x, y] = meshgrid(0:.5:2*pi, 0:.5:2*pi);
quiver(x, y, x, 5*cos(x))
octave:1>

\[
[x,y] = \text{meshgrid}(0:.5:2\pi,0:.5:2\pi);
\]

\[
\text{quiver}(x,y,x,5\cos(x))
\]
\[
[x, y] = \text{meshgrid}(0:.5:2\pi, 0:.5:2\pi);
\]

\[\text{quiver}(x, y, x, 5\cos(x))\]
Midpoint Method
octave:1>

octave:1>

$x, y = \text{meshgrid}(0:.5:2\pi, 0:.5:2\pi)$;

quiver($x, y, x, 5 \cos(x)$)
octave:1> 

octave:1> 
[x,y] = meshgrid(0:.5:2*pi,0:.5:2*pi);
quiver(x,y,x,5*cos(x))
$[x,y] = \text{meshgrid}(0:.5:2\pi,0:.5:2\pi);$

```matlab
\text{quiver}(x,y,x,5*\cos(x))$
```
octave:1> [x,y] = meshgrid(0:.5:2*pi,0:.5:2*pi);
octave:1> quiver(x,y,x,5*cos(x))
Both second-order Runge-Kutta methods (same accuracy)

Very different behavior

Trapezoidal rule is smoother, more damped looking

Midpoint Method keeps more energy, but can be noisy / aliased
Position Updates for

\[
d^2 x_p(t) \over dt^2 = f(x_p, t) \over m_p
\]
Three Position Updates

\[ x_{p}(t + \Delta t) = x_{p}(t) + dt \cdot v_{p}(t) \]

\[ x_{p}(t + \Delta t) = x_{p}(t) + \frac{dt}{2} \cdot (v_{p}(t) + v_{p}(t + \Delta t)) \]

\[ x_{p}(t + \Delta t) = x_{p}(t) + dt \cdot v_{p}(t + \Delta t) \]
“Stiff” Problems

Consider the IVP:

\[ x(0) = x_0 \quad v(0) = 0 \quad f = -kx \]
“Stiff” Problems

Consider the IVP:

\[ x(0) = x_0 \quad v(0) = 0 \quad f = -kx \]

After one time step:

\[ x(\Delta t) = \left( 1 - \frac{\Delta t^2 k}{m} \right) x_0 \]
“Stiff” Problems

Consider the IVP:

\[ x(0) = x_0 \quad v(0) = 0 \quad f = -kx \]

After one time step:

\[ x(\Delta t) = \left(1 - \frac{\Delta t^2 k}{m}\right) x_0 \]

If \( \Delta t > \sqrt{\frac{2m}{k}} \)

the spring will be more extended than when we started
If We Want to Take Bigger Timesteps
Implicit Integration

Replace:

\[
v(t + \Delta t) = v(t) + \Delta t \cdot M^{-1} f(x(t), t)
\]
\[
x(t + \Delta t) = x(t) + \Delta t \cdot v(t + \Delta t)
\]

With:

\[
v(t + \Delta t) = v(t) + \Delta t \cdot M^{-1} f(x(t + \Delta t), t + \Delta t)
\]
\[
x(t + \Delta t) = x(t) + \Delta t \cdot v(t + \Delta t)
\]
Implicit Integration

Replace:

\[ \mathbf{v}(t + \Delta t) = \mathbf{v}(t) + \Delta t \cdot M^{-1} \mathbf{f}(\mathbf{x}(t), t) \]
\[ \mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \Delta t \cdot \mathbf{v}(t + \Delta t) \]

With:

\[ \mathbf{v}(t + \Delta t) = \mathbf{v}(t) + \Delta t \cdot M^{-1} \mathbf{f}(\mathbf{x}(t + \Delta t), t + \Delta t) \]
\[ \mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \Delta t \cdot \mathbf{v}(t + \Delta t) \]
Applied to Soft Bodies

\[ K(x - x_0) + D(v) + Ma = f_{ext} \]
Applied to Soft Bodies

\[ K(x - x_0) + D(v) + Ma = f_{ext} \]

Linearize
Applied to Soft Bodies

\[ K(x - x_0) + D(v) + Ma = f_{ext} \]

Linearize

\[ Kx - Kx_0 + Dv + Ma = f_{ext} \]
Applied to Soft Bodies

\[ K(x - x_0) + D(v) + Ma = f_{ext} \]

Linearize

\[ Kx - Kx_0 + Dv + Ma = f_{ext} \]

\[ x(t + \Delta t) = x(t) + \Delta t \cdot v(t + \Delta t) \]
Applied to Soft Bodies

\[
K(x - x_0) + D(v) + Ma = f_{ext}
\]

Linearize

\[
Kx - Kx_0 + Dv + Ma = f_{ext}
\]

+ \[
x(t + \Delta t) = x(t) + \Delta t \cdot v(t + \Delta t)
\]

substitute into

\[
v(t + \Delta t) = v(t) + \Delta t \cdot M^{-1}f(x(t + \Delta t), t + \Delta t)
\]
Applied to Soft Bodies

\[ K(x - x_0) + D(v) + Ma = f_{ext} \]

Linearize

\[ Kx - Kx_0 + Dv + Ma = f_{ext} \]

\[ x(t + \Delta t) = x(t) + \Delta t \cdot v(t + \Delta t) \]

substitute into

\[ v(t + \Delta t) = v(t) + \Delta t \cdot M^{-1}f(x(t + \Delta t), t + \Delta t) \]

re-arrange
Applied to Soft Bodies

\[
K(x - x_0) + D(v) + Ma = f_{ext}
\]

Linearize

\[
Kx - Kx_0 + Dv + Ma = f_{ext}
\]

\[
+ x(t + \Delta t) = x(t) + \Delta t \cdot v(t + \Delta t)
\]

substitute into

\[
v(t + \Delta t) = v(t) + \Delta t \cdot M^{-1}f(x(t + \Delta t), t + \Delta t)
\]

re-arrange

\[
(M + \Delta t^2 K + \Delta t D) v(t + \Delta t) = Mv(t) + \Delta t (-K (x(t) - x_0) + f_{ext})
\]
\((M + \Delta t^2K + \Delta tD) \ v(t + \Delta t) = Mv(t) + \Delta t \ (-K (x(t) - x_0) + f_{ext})\)
\[(M + \Delta t^2 K + \Delta t D) v(t + \Delta t) = Mv(t) + \Delta t (-K (x(t) - x_0) + f_{ext})\]

**Linear System**

Sparse, Symmetric
\[(M + \Delta t^2 K + \Delta t D) \mathbf{v}(t + \Delta t) = M \mathbf{v}(t) + \Delta t (-K (\mathbf{x}(t) - \mathbf{x}_0) + f_{ext})\]

\begin{align*}
\text{Linear System} & \quad \text{Momentum} \\
\text{Sparse, Symmetric} & 
\end{align*}
\[(M + \Delta t^2 K + \Delta t D) \mathbf{v}(t + \Delta t) = M \mathbf{v}(t) + \Delta t (-K(x(t) - x_0) + f_{ext})\]

Euler step of elastic and external forces

Linear System
Sparse, Symmetric

Momentum
V. Constraints
Constraints

- Geometric relationships that must be satisfied

\[ y = h \]
Constraints

- Constraint forces arise in response to other forces to maintain the constraint
Degrees of Freedom (DOF)

- Number of independent parameters describing configuration

\[ 3n \]
Degrees of Freedom (DOF)

- number of independent parameters describing configuration

\[ 3n - 1 \]

6
Unilateral/Bilateral Constraints

\[ g(x, t) \geq 0 \quad \text{and} \quad g(x, t) = 0 \]
Unilateral/Bilateral Constraints

\[ g(x, t) \geq 0 \]

\[ g(x, t) = 0 \]

\[ y_1 - h \geq 0 \]

\[ (x_1 + r_1) - (x_2 + r_2) = 0 \]
Soft vs. Hard Constraints

- **Soft constraint:** force competes with other forces in the system
- **Hard constraint:** force as strong as necessary to maintain constraint
Constraints: Solution Methods

- Maximal coordinates + auxiliary conditions
  - Include forces to maintain constraint
- Generalized coordinates (a.k.a., reduced coordinates, minimal coordinates)
  - Parameterize true DOF, respecting constraints
Penalty Methods

- Restoring force that acts to drive system to valid state

constraint: \((\mathbf{x}_1 + \mathbf{r}_1) - (\mathbf{x}_2 + \mathbf{r}_2) = 0\)

penalty force: \(\mathbf{f}_p = -k((\mathbf{x}_2 + \mathbf{r}_2) - (\mathbf{x}_1 + \mathbf{r}_1))\)

\[m\mathbf{a} = \mathbf{f} + \mathbf{f}_p\]
Penalty Methods

✓ Simple to add to solver

- Must tune parameters (k)
- Introduce stiff forces $\rightarrow$ smaller time step or implicit integration
- Constraint violation, oscillations
Lagrange Multiplier Methods

- Constraint force arises in response to other forces in the system
- Add unknowns to equations that represent strength of constraint force
- Add (differentiated) constraint equations
Constraint Implicit Surface

\[ \mathbf{x} = (x_1, x_2, \ldots, x_n) \]

- **Gradient**

\[ \nabla g(\mathbf{x}) = \begin{pmatrix} \frac{\partial g}{\partial x_1}(\mathbf{x}) \\ \frac{\partial g}{\partial x_2}(\mathbf{x}) \\ \vdots \\ \frac{\partial g}{\partial x_n}(\mathbf{x}) \end{pmatrix} \]

\[ g(\mathbf{x}) = 0 \]

- \( g(\mathbf{x}) < 0 \)
  - \( g(\mathbf{x}) > 0 \)
Constraint Force

- Constraint force
  \[ F_c \]

- Workless
  \[ F_c \cdot \delta x = 0 \]

  \[ \Rightarrow F_c = \lambda \nabla g(x) \]

  Lagrange multiplier

\[ g(x) = 0 \]
\[ g(x) < 0 \]
\[ g(x) > 0 \]
Equations of Motion

- m constraints

\[ g(x) = \begin{pmatrix} g_1(x) \\ g_2(x) \\ \vdots \\ g_m(x) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} = 0 \]

- Constraint force

\[ F_c = J^T \lambda = \begin{pmatrix} \nabla g_1 \\ \nabla g_2 \\ \vdots \\ \nabla g_m \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_m \end{pmatrix} \]

- Equations of motion

\[ Ma = F + F_c = F + J^T \lambda \]
How to find $\lambda$

- Constraint tells us valid positions
  \[ g(x) = 0 \]

- Differentiate to get valid velocities
  \[ \dot{g}(x) = J(x)v = 0 \]

- Differentiate again to get valid accelerations
  \[ \ddot{g}(x) = \ddot{J}v + J\ddot{a} = 0 \]
How to find $\lambda$

- Equations of motion

\[
Ma = F + F_c = F + J^T \lambda
\]

\[
a = M^{-1}(F + J^T \lambda)
\]
How to find $\lambda$

- Equations of motion

$$Ma = F + F_c = F + J^T \lambda$$

$$a = M^{-1}(F + J^T \lambda)$$

- Plug into expression for valid $a$

$$\dot{g}(x) = \dot{J}v + Ja = 0$$
How to find $\lambda$

- Equations of motion
  
  $$Ma = F + F_c = F + J^T\lambda$$
  
  $$a = M^{-1}(F + J^T\lambda)$$

- Plug into expression for valid $a$
  
  $$\dot{g}(x) = \dot{J}v + Ja = 0$$

- Rearrange
  
  $$(JM^{-1}J^T)\lambda = -JM^{-1}F - \dot{J}v$$
How to find $\lambda$

- **Rearrange**

\[
(JM^{-1}J^T)\lambda = -JM^{-1}F - Jv
\]

- **KKT System**

\[
\begin{pmatrix}
M & -J^T \\
-J & 0
\end{pmatrix}
\begin{pmatrix}
y \\
\lambda
\end{pmatrix}
=
\begin{pmatrix}
0 \\
-b
\end{pmatrix}
\]
Generalized Coordinates

- Instead of maximal coordinates \( x_1, x_2, \ldots, x_n \) along with auxiliary conditions and forces

- Generalized coordinates \( q_1, q_2, \ldots, q_N \) with \( N < n \) that take constraints into account
Generalized Coordinates

- Example: rigid body

\[ x_1, x_2, x_3, x_4 \]

+ constraints

\[ x, R \]
Transformation Equations

- Relate maximal and generalized coordinates

\[ x_1 = x_1(q_1, q_2, \ldots, q_N) \]
\[ x_2 = x_2(q_1, q_2, \ldots, q_N) \]
\[ \vdots \]
\[ x_n = x_n(q_1, q_2, \ldots, q_N) \]

\[ \mathbf{x} = \mathbf{x}(\mathbf{q}) \]

- Jacobian

\[ J(\mathbf{q}) = \frac{\partial \mathbf{x}}{\partial \mathbf{q}} \]
Generalized Forces

\[ \begin{align*}
  x_1 & \quad f_1 \\
  \vdots & \quad \vdots \\
  x_n & \quad f_n \\
  q_1 & \quad g_1 \\
  \vdots & \quad \vdots \\
  q_N & \quad g_N
\end{align*} \]
Generalized Forces

- Two sets of forces are dynamically equivalent if they do the same "virtual work"

\[
\begin{align*}
    f \cdot \delta x &= g \cdot \delta q \\
    f \cdot J \delta q &= g \cdot \delta q \\
    g &= J^T f
\end{align*}
\]
Generalized Forces: Rigid Body

\[ \dot{x} = v + \omega \times r = \begin{pmatrix} I & r^*T \end{pmatrix} \begin{pmatrix} v \\ \omega \end{pmatrix} \]

\[ g = J^T f = \begin{pmatrix} I \\ r^* \end{pmatrix} f = \begin{pmatrix} f \\ r \times f \end{pmatrix} \]
Equations of Motion

- Lagrange equations of motion

\[
\frac{d}{dt} \left( \frac{\partial T}{\partial \dot{q}} \right) = g \\
\left( \frac{d}{dt} (mv) = f \right)
\]

\[
T = \frac{1}{2} \dot{x}^T M \dot{x} = \frac{1}{2} (J \dot{q})^T M J \dot{q} = \frac{1}{2} \dot{q}^T J^T M J \dot{q}
\]

- Euler-Lagrange equations of motion, defining Lagrangian \( L = T - V \)

\[
\frac{d}{dt} \left( \frac{\partial L}{\partial \dot{q}} \right) - \frac{\partial L}{\partial q} = 0
\]
Example: Pendulum

❖ Transformation equations

\[ x(\theta) = l \sin \theta \]
\[ y(\theta) = -l \cos \theta \]

❖ Velocity

\[
\begin{pmatrix}
\dot{x}(\theta) \\
\dot{y}(\theta)
\end{pmatrix}
= \begin{pmatrix}
l \cos \theta \ddot{\theta} \\
l \sin \theta \ddot{\theta}
\end{pmatrix}
= J \ddot{\theta}
\]

❖ Lagrangian

\[
L = \frac{1}{2} ml^2 \dot{\theta}^2 - mgl \cos \theta
\]

❖ Equations of motion

\[
\ddot{\theta} + \frac{g}{l} \sin \theta = 0
\]
Constrained Rigid Body Systems

- Types of constraints
  - Articulation (joints)
  - Collisions
  - Resting and sliding contact
Impulse-Momentum Equations

- Acceleration-level formulation doesn’t work well with discontinuous velocities, frictional contact

- Instead, integrate $f = ma$ to get impulse-momentum formulation

\[
\int_{t_1}^{t_2} ma \, dt = \int_{t_1}^{t_2} f \, dt,
\]

\[
\Rightarrow m(v(t_2) - v(t_1)) = j
\]
Impulse-Momentum Equations

- Integrated, semi-discrete equations

\[ M \mathbf{V}^{n+1} = M \mathbf{V}^n + \Delta t \mathbf{F} + J^T \lambda^{n+1} \]

- Combine with velocity-level constraint equation \( J \mathbf{V} = 0 \)

\[
\begin{pmatrix}
M & -J^T \\
-J & 0
\end{pmatrix}
\begin{pmatrix}
\mathbf{V}^{n+1} \\
\mu^{n+1}
\end{pmatrix}
=
\begin{pmatrix}
M \mathbf{V}^n + \Delta t \mathbf{F} \\
0
\end{pmatrix}
\]

\[ J M^{-1} J^T \mu^{n+1} = -J \mathbf{V}^n - \Delta t J M^{-1} \mathbf{F} \]
Instead of solving global, coupled system, common to split update with non-constraint forces add constraint impulses

\[ M\dot{V}^* = MV^n + \Delta tF \]

\[ MV^{n+1} = MV^* + J^T \mu^{n+1} \]
Instead of solving global, coupled system, common to split update with non-constraint forces and add constraint impulses.

\[ M \mathbf{V}^* = M \mathbf{V}^n + \Delta t \mathbf{F} \]
\[ M \mathbf{V}^{n+1} = M \mathbf{V}^* + J^T \mu^{n+1} \]
Instead of solving global, coupled system, common to split update with non-constraint forces add constraint impulses

\[ M\mathbf{V}^* = M\mathbf{V}^n + \Delta t\mathbf{F} \]

\[ M\mathbf{V}^{n+1} = M\mathbf{V}^* + J^T \mu^{n+1} \]
Instead of solving global, coupled system, common to split update with non-constraint forces add constraint impulses

\[ M\mathbf{V}^* = M\mathbf{V}^n + \Delta t\mathbf{F} \]
\[ M\mathbf{V}^{n+1} = M\mathbf{V}^* + J^T\mu^{n+1} \]
Instead of solving global, coupled system, common to split update with non-constraint forces and add constraint impulses.

\[ M \mathbf{V}^* = M \mathbf{V}^n + \Delta t \mathbf{F} \]
\[ M \mathbf{V}^{n+1} = M \mathbf{V}^* + J^T \mu^{n+1} \]
Instead of solving global, coupled system, common to split update with non-constraint forces and add constraint impulses.

\[ M\mathbf{V}^* = M\mathbf{V}^n + \Delta t\mathbf{F} \]
\[ M\mathbf{V}^{n+1} = M\mathbf{V}^* + J^T \mu^{n+1} \]
Instead of solving global, coupled system, common to split update with non-constraint forces add constraint impulses.

\[
MV^* = MV^n + \Delta t F
\]

\[
MV^{n+1} = MV^* + J^T \mu^{n+1}
\]
Instead of solving global, coupled system, common to split update with non-constraint forces add constraint impulses

\[
M\mathbf{V}^* = M\mathbf{V}^n + \Delta t\mathbf{F} \\
M\mathbf{V}^{n+1} = M\mathbf{V}^* + J^T \mu^{n+1}
\]
Instead of solving global, coupled system, common to split update with non-constraint forces, add constraint impulses.

\[ M \mathbf{v}^* = M \mathbf{v}^n + \Delta t \mathbf{F} \]
\[ M \mathbf{v}^{n+1} = M \mathbf{v}^* + J^T \mathbf{\mu}^{n+1} \]
Impulse-Momentum Equations

- Instead of solving global, coupled system, common to split update with non-constraint forces add constraint impulses

\[ M \mathbf{V}^* = M \mathbf{V}^n + \Delta t \mathbf{F} \]
\[ M \mathbf{V}^{n+1} = M \mathbf{V}^* + J^T \mu^{n+1} \]
**Impulse-Momentum Equations**

- Instead of solving global, coupled system, common to split update with non-constraint forces and add constraint impulses.

\[
M \dot{V}^* = M V^n + \Delta t F
\]

\[
M V^{n+1} = M \dot{V}^* + J^T \mu^{n+1}
\]
Instead of solving global, coupled system, common to split update with non-constraint forces. Add constraint impulses.

\[ M \mathbf{V}^* = M \mathbf{V}^n + \Delta t \mathbf{F} \]
\[ M \mathbf{V}^{n+1} = M \mathbf{V}^* + J^T \mu^{n+1} \]

Iterative impulse solve, repeat fixed number of times or until tolerance met.
Global vs. Iterative Solve

- **Global**
  - need to solve larger linear system
  - LCP for inequality constraints

- **Iterative**
  - may be slow to converge
  - simple to do inequality constraints
Handling Drift With Stabilization

- Approach was based on velocity-level constraints lead to drift in positions

\[
\begin{pmatrix}
M & -J^T \\
-J & 0
\end{pmatrix}
\begin{pmatrix}
V^{n+1} \\
\mu^{n+1}
\end{pmatrix}
= \begin{pmatrix}
MV^n + \Delta tF \\
0
\end{pmatrix}
\]

- Correct drift with stabilization
  - E.g., Baumgarte stabilization \( \dot{g}(x) + \gamma g(x) = 0 \)
  - Modify positions and velocities so they satisfy constraints
Softening Constraints

- Also common to soften constraints

\[
\begin{pmatrix}
M & -J^T \\
-J & \gamma I
\end{pmatrix}
\begin{pmatrix}
V^{n+1} \\
\mu^{n+1}
\end{pmatrix}
= 
\begin{pmatrix}
M V^n + \Delta t F \\
-\frac{\beta}{\Delta t} g(X^n)
\end{pmatrix}
\]

- Stabilizes constraints
- Regularizes system
  - Better numerical properties
  - Handle redundant constraints
- Adds some compliance to constraint
Collisions and Contact
Collision Detection

- Polygonal geometry

- Separating axis theorem

- Convex decomposition
Collision Detection

- Signed distance field
  \[ \phi(x) \]
- Zero level set
  \[ \phi(x) = 0 \]
- Fast inside/outside tests
- Penetration depth \( \phi(x) \)
- Normals \( \nabla \phi(x) \)
Accelerating Collision Detection

- Bounding volumes
- Hierarchical bounding volumes
- Spatial partitions
Discrete vs. Continuous Collision Detection
Collision Response: Inelastic

- Newton’s third law (action/reaction)
  \[ m_1 v'_1 = m_1 v_1 + j \]
  \[ m_2 v'_2 = m_2 v_2 - j \]

- Assume inelastic (sticking)
  \[ v'_1 = v'_2 \]

- Solve:
  \[ j = \left( \frac{1}{m_1} + \frac{1}{m_2} \right)^{-1} (v_2 - v_1) \]
  \[ v'_1 = v'_2 = \left( \frac{m_1}{m_1 + m_2} \right) v_1 + \left( \frac{m_2}{m_1 + m_2} \right) v_2 \]
Collision Response: Elastic

- Newton’s third law (action/reaction)
  \[ m_1 v'_1 = m_1 v_1 + j \]
  \[ m_2 v'_2 = m_2 v_2 - j \]

- Assume elastic (bouncing)
  \[ (v'_2 - v'_1) = -\varepsilon (v_2 - v_1) \]

- Solve:
  \[ j = \left( \frac{1}{m_1} + \frac{1}{m_2} \right)^{-1} (1 + \varepsilon)(v_2 - v_1) \]
  \[ v'_1 = v_1 + \frac{1}{m_1} j, \quad v'_2 = v_2 - \frac{1}{m_2} j \]
Deformable Object Collisions

- Body compresses, stores energy and bounces
- Inelastic collision between particles and ground

\[ m_1 v_1' = m_1 v_1 + j \]
\[ m_2 v_2' = m_2 v_2 - j \]
\[ v_2' = v_1' \]
Deformable Object Collisions

- Body compresses, stores energy and bounces
- Inelastic collision between particles and ground

\[
\begin{align*}
    m_1 v_1' &= m_1 v_1 + j \\
    m_2 v_2' &= m_2 v_2 - j \\
    v_2' &= v_1'
\end{align*}
\]
Rigid Body Collisions

❖ Physically, similar to deformable body collisions

❖ But rigid idealization precludes storing energy

❖ Instead, algebraic collision laws for before/after collision

❖ Cases:
  ❖ Inelastic (sticking)
  ❖ Elastic: frictionless, with friction
Rigid Body Inelastic Collision

- Linear momentum
  \[ m_1 v'_{C1} = m_1 v_{C1} + j \]
  \[ m_2 v'_{C2} = m_2 v_{C2} - j \]

- Angular momentum
  \[ I_1 \omega'_1 = I_1 \omega_1 + r_1 \times j \]
  \[ I_2 \omega'_2 = I_2 \omega_1 - r_2 \times j \]

- Sticking
  \[ v'_{P1} = v'_{P2} \]

15 equations / unknowns
Rigid Body Frictionless Collision

- Linear momentum
  \[ m_1 v'_{C1} = m_1 v_{C1} + j \]
  \[ m_2 v'_{C2} = m_2 v_{C2} - j \]

- Angular momentum
  \[ I_1 \omega'_1 = I_1 \omega_1 + r_1 \times j \]
  \[ I_2 \omega'_2 = I_2 \omega_1 - r_2 \times j \]

- Elastic
  \[ j = jn \]
  \[ (v'_{P2} - v'_{P1}) \cdot n = -\epsilon(v_{P2} - v_{P1}) \cdot n \]
Rigid Body Frictional Collision

- **Coulomb friction model**
  \[ \mu \text{ coefficient of friction} \]

- **Static**
  \[ \| f_t \| \leq \mu \| f_n \| \]

- **Sliding**
  \[ f_t = -\mu \| f_n \| t \]
Rigid Body Frictional Collision

- Example [Guendelman et al. 2003]
- Elastic in normal direction
  \[
  (v'_{P2} - v'_{P1}) \cdot n = -\epsilon(v_{P2} - v_{P1}) \cdot n
  \]
- Assume sticking (+2 equations)
  \[
  (I - nn^T)(v'_{P2} - v'_{P1}) = 0
  \]
- If not admissible, assume sliding (-2 unknowns)
  \[
  j = j_n n - \mu j_n t
  \]
Thanks!

For notes, slides, and source code visit:
https://cal.cs.umbc.edu/Courses/PhysicsBasedAnimation

adamb@umbc.edu
shinar@cs.ucr.edu