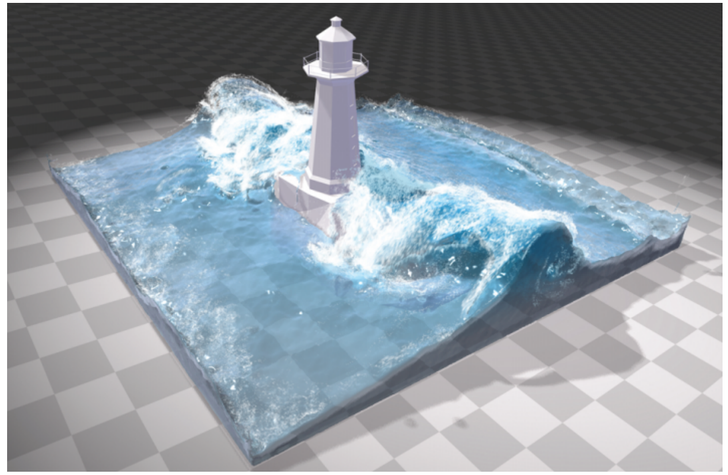
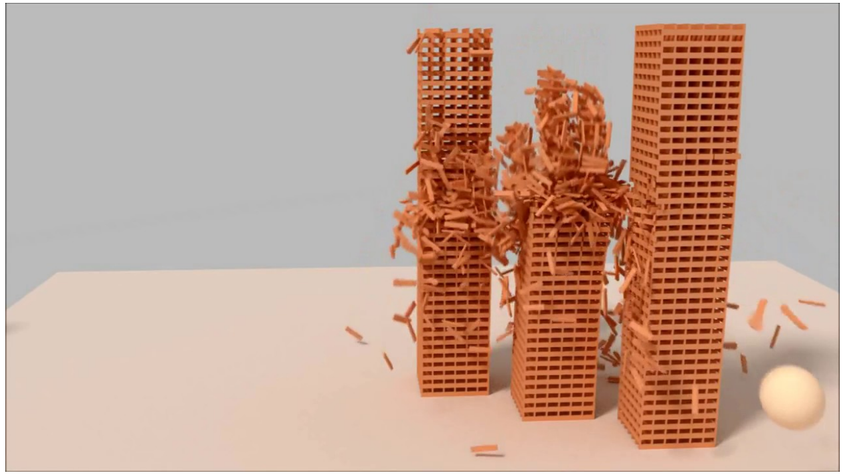
Physically based simulation

When physically based simulation is needed

- Accurate dynamics
- Teadious to model by hand or procedurally
  - Multiple interacting elements: ex. Multiple collisions: rigid bodies, hairs, etc.
  - Complex animated geometry: Cloths, fluids





General metohdology

1. Description of the system - Describe system by some parameters (positions, speed, orientation, etc).

- State of the system is known at time $t = 0$ - *Initial value problem in time*
- State of the system may be constrained in space - *Boundary value problem in space*

2. Evolution Link the evolution of the system to forces or constraints using dynamic principles and conservation laws

$\Rightarrow$ Differential equation

3. Numerical Solution Solve the differential equation using numerical iterative approaches.

Note: Fundamentally different that direct approach controling the trajectories at key-frames

- The system is set at an initial step
- We let the numerical solution build the space-time trajectory for us

(+) Allows to model complex behavior

(-) Lack of control on the result

Physically based models

1- Particles

2- Rigid bodies

3- Continuum models

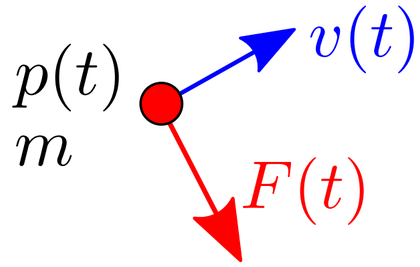
Physically-based particle system

1. Description

Particle is fully described by: Position p, Speed v, Mass m

fundamental quantities: position and linear momentum $P = m\,v$

Momentum preserved through time when no external forces are applied



2. Evolution

- Fundamental principle of the dynamics

Force applied on particle $F(p, v, t)$

$$\begin{cases} p'(t) = v(t) \\ P'(t) = m\,v'(t) = F(p, v, t) \end{cases}$$

- Conservation of energy (ex. kinetic energy $(1/2\,m\,v^2)$+potential energy = const, etc.)

- Lagragian, or Hamiltonian

Physically-based particle system

3. Numerical Solution

ODE (Ordinary Differential Equation) formulated as an **Initial Value Problem**

$$\text{ex. } \begin{cases} p'(t) = v(t) \\ m v'(t) = F(p, v, t) \end{cases}, \quad \text{with } v(0) = v_0,\ p(0) = p_0$$

- Discretize in time $t^k = k\,h, \quad h = \Delta t =$ time step.

  $\Rightarrow$ Build a discrete numerical solution $p^k = p(t^k), v^k = v(t^k)$.

- We can consider initially the following iterative scheme

$$\begin{cases} v^{k+1} = v^k + h\,F(p^k, v^k, t^k) \\ p^{k+1} = p^k + h\,v^{k+1} \end{cases}$$

Simple to implement, reasonably OK for simple examples (more details later).

Physically-based particle system

Pro

(+) Simple to implement, and control.

(+) Efficient to compute, scalable

(+) Highly adaptable from simple particle to rigid and deformable models

Cons

(-) Limited accuracy - highly simplified model from physical point of view.

$\Rightarrow$ Dominant model in CG production for general purpose deformable model animation.

Common use: Lots of sparsely interacting particles

Physically based models

1- Particles

2- Rigid bodies

3- Continuum models

Orientation and angular speed

In addition of position and speed, rigid body is defined by its **orientation** r and **angular speed** ω.

- In matrix form $r \in \mathbb{R}^{3\times 3}$

- Angular speed $\omega = (\omega_x, \omega_y, \omega_z) \in \mathbb{R}^3$

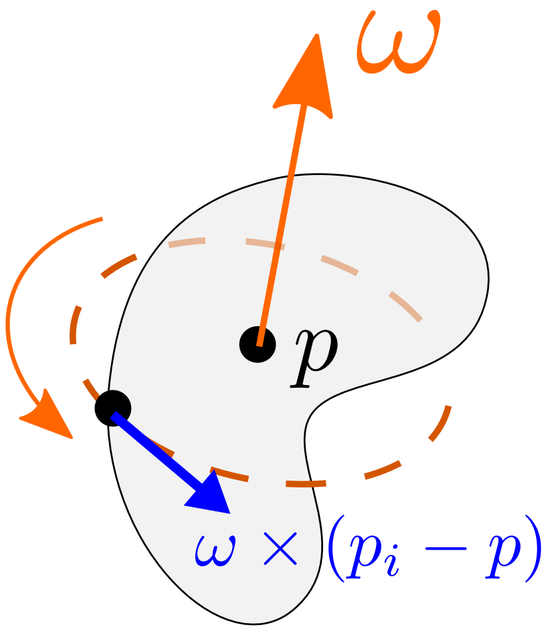
  Every vector $p_i(t)$ of the rigid body has a speed $p_i'(t) = v(t) + \omega(t) \times (p_i(t) - p(t))$

- Matrix expression of the angular speed

$$\mathrm{M}_\omega = \begin{pmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{pmatrix}$$

- Derivative of orientation matrix

  $r'(t) = \mathrm{M}_\omega(t)\, r(t)$



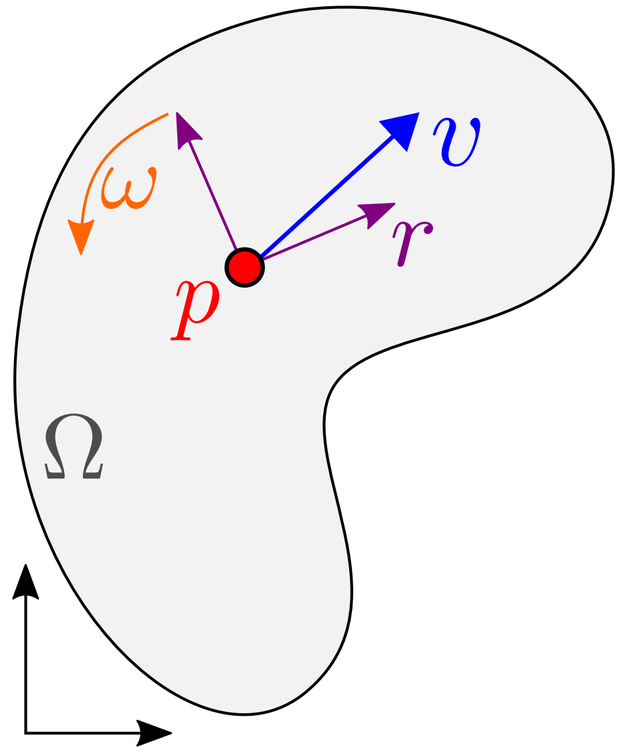
Rigid bodies system description

Similar to particle: Position $p \in \mathbb{R}^3$ and speed $v \in \mathbb{R}^3$ of the center of mass. Mass of the solid $m \in \mathbb{R}$.

- $p'(t) = v(t)$
- Fundamental quantity: position p and linear momentum $P = m\,v$.

Addition of rigid body: **Orientation** $r \in \mathbb{R}^{3\times3}$ (in its matrix formulation), and **angular speed** $w \in \mathbb{R}^3$ of the solid.

- $r'(t) = \mathrm{M}_\omega(t)\,r(t)$
- Fundamental quantity: orientation r and **angular momentum** $L = I\,\omega \in \mathbb{R}^3$, with $I \in \mathbb{R}^{3\times3}$ **inertia tensor** of the solid.



Intuition

- Solid acts like a particle at its center of mass + orientation and angular speed
- Mass $\rightarrow$ resistance to change of linear speed
- Inertia tensor ($\simeq$ angular mass) $\rightarrow$ resistance to change of angular speed

Definition of quantities

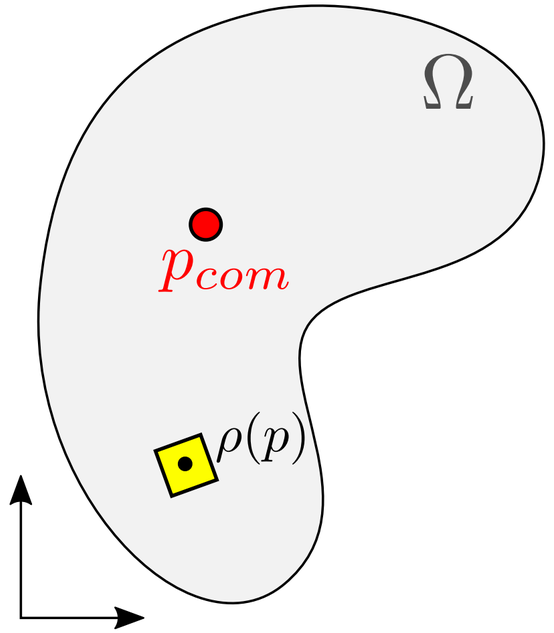
- Solid defined within a domain $\Omega \subset \mathbb{R}^3$
- With a density of mass $\rho(p)$ at each point $p \in \Omega$

- Total mass of the solid m

$$m = \int_{p \in \Omega} \rho(p) \, \mathrm{d}\Omega$$

- Position of the center of mass p_{com}

$$p_{com} = \frac{1}{m} \int_{p \in \Omega} \rho(p) \, p \, \mathrm{d}\Omega$$



Definition of inertia

Inertia tensor expressed at point q

Calling r the local coordinate $r = p - q$

$$I = \int_{r \in \Omega} \rho(r) \begin{pmatrix} I_{xx}(r) & I_{xy}(r) & I_{xz}(r) \\ I_{xy}(r) & I_{yy}(r) & I_{yz}(r) \\ I_{xz}(r) & I_{yz}(r) & I_{zz}(r) \end{pmatrix} \mathrm{d}\Omega = \int_{r \in \Omega} \rho(r) \begin{pmatrix} r_y^2 + r_z^2 & -r_x r_y & -r_x r_z \\ -r_x r_y & r_x^2 + r_z^2 & -r_y r_z \\ -r_x r_z & -r_y r_z & r_x^2 + r_y^2 \end{pmatrix} \mathrm{d}\Omega$$

$$I = \int_{r \in \Omega} \rho(r) \left(r^T r \, \mathrm{Id} - r \, r^T \right) \mathrm{d}\Omega$$

Rem.

- I is usually expressed at the center of mass $q = p_{com}$
- I depends on the body orientation. Given a rotation r: $I = r \, I_0 \, r^T$

  $\Rightarrow$ compute once I_0 in a rest position, then update it using r
- There exist a frame in which I is diagonal (principle axes of inertia). Corresponds to eigenvectors of matrix I.

Forces and torques on a solid

- Given a local force $f(p)$ acting on a position $p \in \Omega$
- Contribute to 2 global components applied on the body:

  - Total **external force** F applied on the center of mass

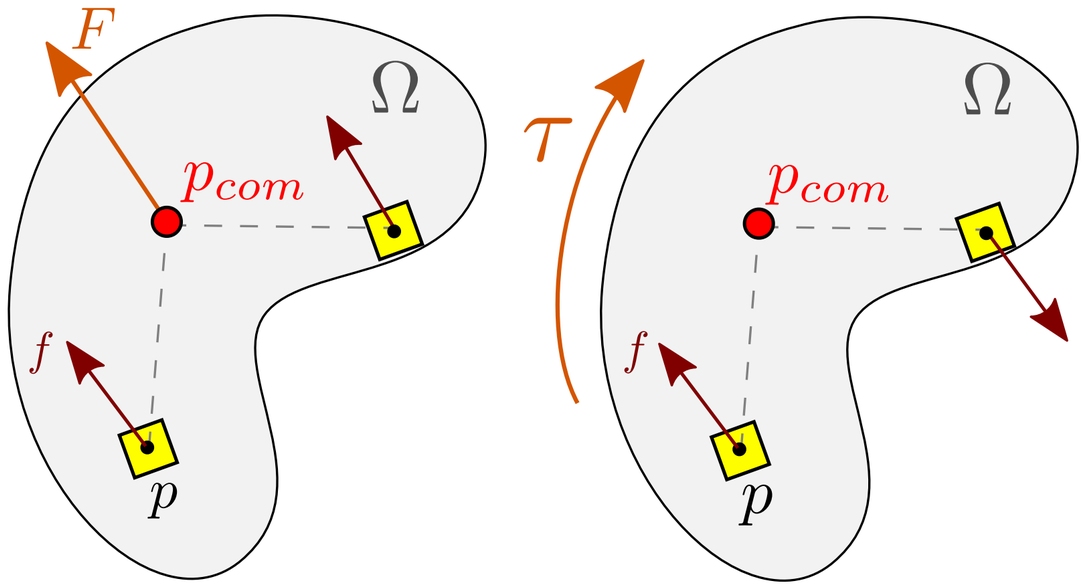
    Induce linear displacement of COM

    $$F = \int_{p \in \Omega} f(p)\,\mathrm{d}p$$

  - **Torque** τ applied on the body

    Induce spinning of the solid around its COM

    $$\tau = \int_{p \in \Omega} (p - p_{com}) \times f(p)\,\mathrm{d}\Omega$$



Dynamics

Similarly to particles

Force F is related to the change of linear momentum $F(t) = P'(t) = (m\,v)'(t)$

Specific to rigid bodies

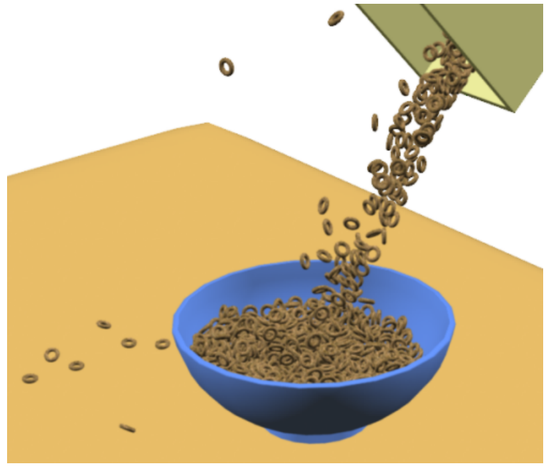
Torque τ is related to the change of angular momentum $\tau(t) = L'(t) = (I\,\omega)'(t)$

Equation of Motion

Fundamental principle of dynamics for rigid body

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} p \\ P \\ r \\ L \end{pmatrix}(t) = \begin{pmatrix} v(t) \\ F(t) \\ \omega(t)\,r(t) \\ \tau(t) \end{pmatrix}$$

Summary: Rigid solid dynamics



1. Description

State vector $u(t) = (p(t), P(t) = m\,v(t), r(t), L(t) = I(t)\,\omega(t))$

Initial condition $u(0) = (p_0, m\,v_0, r_0, I_0\,\omega_0)$

2. Temporal Evolution

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} p \\ P \\ r \\ L \end{pmatrix}(t) = \begin{pmatrix} v(t) \\ F(t) \\ \omega(t)\,r(t) \\ \tau(t) \end{pmatrix}$$

- $I(t) = r(t)\,I_0\,r(t)^T$
- $\omega(t) = I^{-1}(t)\,L(t)$

In practice:

- Shape and density ρ are known
- Compute I_0 for the given shape
- p_0, v_0, r_0, ω_0 are Initial Conditions

Iterate over t^k

- Compute $F(t^k)$ and $\tau(t^k)$ given external forces
- Compute $I(t^k) = r(t^k)\,I_0\,r(t^k)^T$
- Compute $\omega(t^k) = I^{-1}(t^k)\,L(t^k)$
- Update state vector $p^{k+1}, P^{k+1}, r^{k+1}, L^{k+1}$ at t^{k+1}

Use of quaternion

Relation $r'(t) = \omega(t)\, r(t)$ leads in numerical drift from rotation matrix

ex. $r^{k+1} = (\mathrm{Id} + h\,\omega^k)\, r^k$ (explicit scheme)

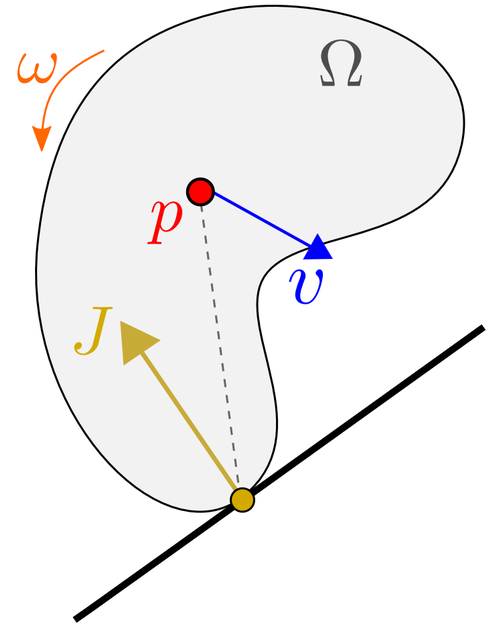
Using quaternion leads to more robust behavior

- Quaternion expression: $q'(t) = \frac{1}{2}\, q_\omega(t)\, q(t)$, with $q_\omega(t) = (\omega(t)\,,\ \ 0))$

- Quaternion is forced to keep a unit norm

ex. $\begin{cases} q^{k+1} = q^k + \frac{1}{2}\, q_\omega^k\, q^k \\ q^{k+1} = q^{k+1} / \|q^{k+1}\| \end{cases}$

Collisions in rigid bodies

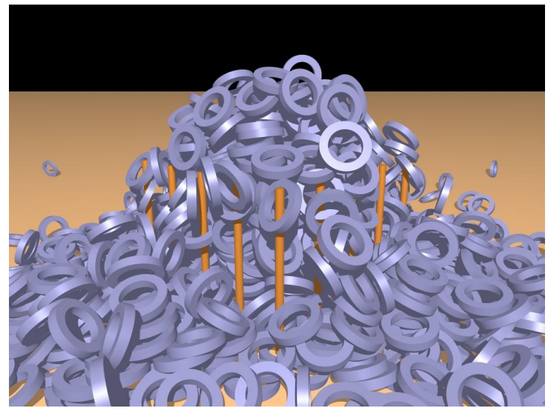


Collisions at position p change both linear and angular velocity

Similarly to particle: use of impulse (sudden change of speed) J.

Impulse split into

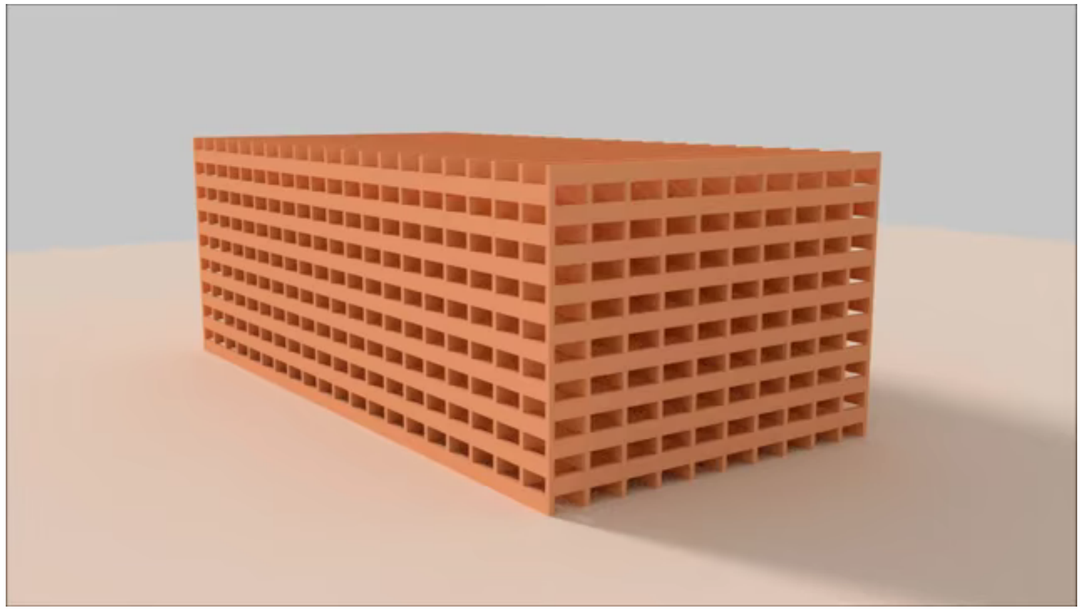
- *Force impulse* $\Delta P = m\,\Delta v = J$
- *Torque impulse* $\Delta L = I\,\Delta\omega = (p - p_{com}) \times J$



More details [D. Barff. Physically Based Modeling. SIGGRAPH Course Notes 1999]

Rigid bodies usage

- Standard usage for rigid bodies motions
  - Limited to non-deformable shapes
- Common in VFX (explosions), and *simulation games* (cars, airplanes, etc).
- Standard library: Bullet physics (ex. used in Blender).



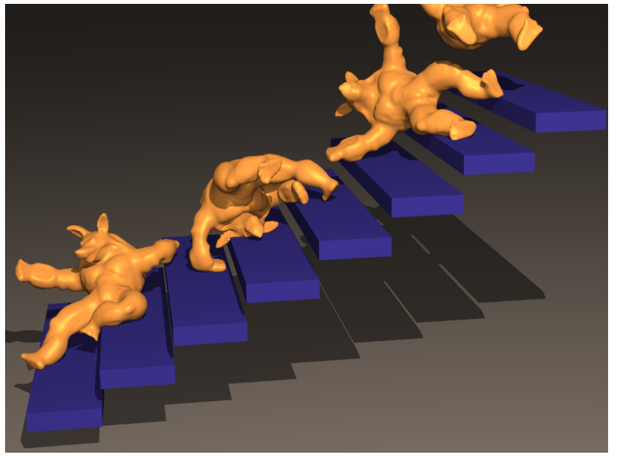
Physically based models

1- Particles

2- Rigid bodies

3- Continuum models

Deformation of a continuous shape



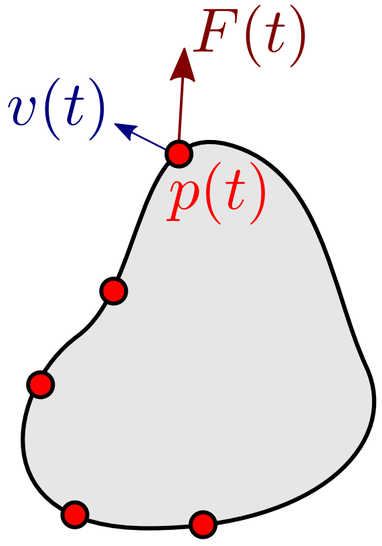
Every part of the shape can be deformed

ex. Describing elastic shapes, visco-elastic shapes, fluids, etc.

Two ways to describe the deforming object

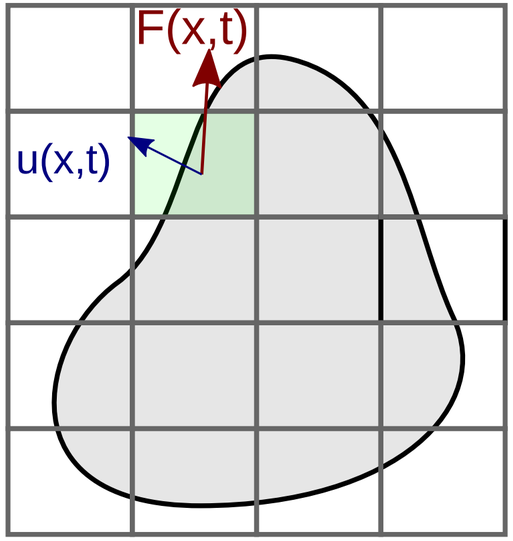
1. Lagrange representation

Positions follow the object deformation



2. Euler representation

Positions are fixed in 3D space

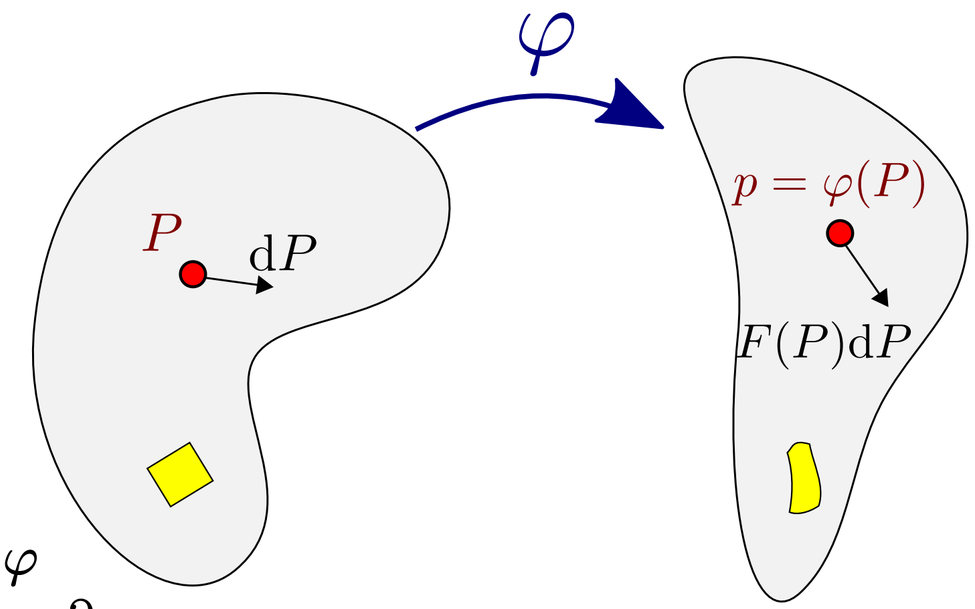


Deformation the Lagrangian description

Deformation map $\varphi : \mathbb{R}^3 \to \mathbb{R}^3$ such that $p = \varphi(P)$

P position in the reference undeformed shape

p position in the deformed configuration.



Deformation Gradient F

- $F(P) = \dfrac{\partial \varphi}{\partial P}(P) = \dfrac{\partial p}{\partial P} \in \mathbb{R}^{3\times 3}$
- Characterizes the local deformation associated to φ

  Position $P + \mathrm{d}P$ is mapped into $\varphi(P + \mathrm{d}P) \simeq p + \dfrac{\partial \varphi}{\partial P}\,\mathrm{d}P$

- $F(P) = \begin{pmatrix} \frac{\partial \varphi_x}{\partial X} & \frac{\partial \varphi_x}{\partial Y} & \frac{\partial \varphi_x}{\partial Z} \\ \frac{\partial \varphi_y}{\partial X} & \frac{\partial \varphi_y}{\partial Y} & \frac{\partial \varphi_y}{\partial Z} \\ \frac{\partial \varphi_z}{\partial X} & \frac{\partial \varphi_z}{\partial Y} & \frac{\partial \varphi_z}{\partial Z} \end{pmatrix}$

Strain

Deformation gradient F describe both

- Rigid transformation (translation, rotation) - not related to material effort
- Any other deformation inducing local length change - related to material effort

Strain ϵ is a measure of deformation ignoring rigid transformation.

Several possible measure of strain

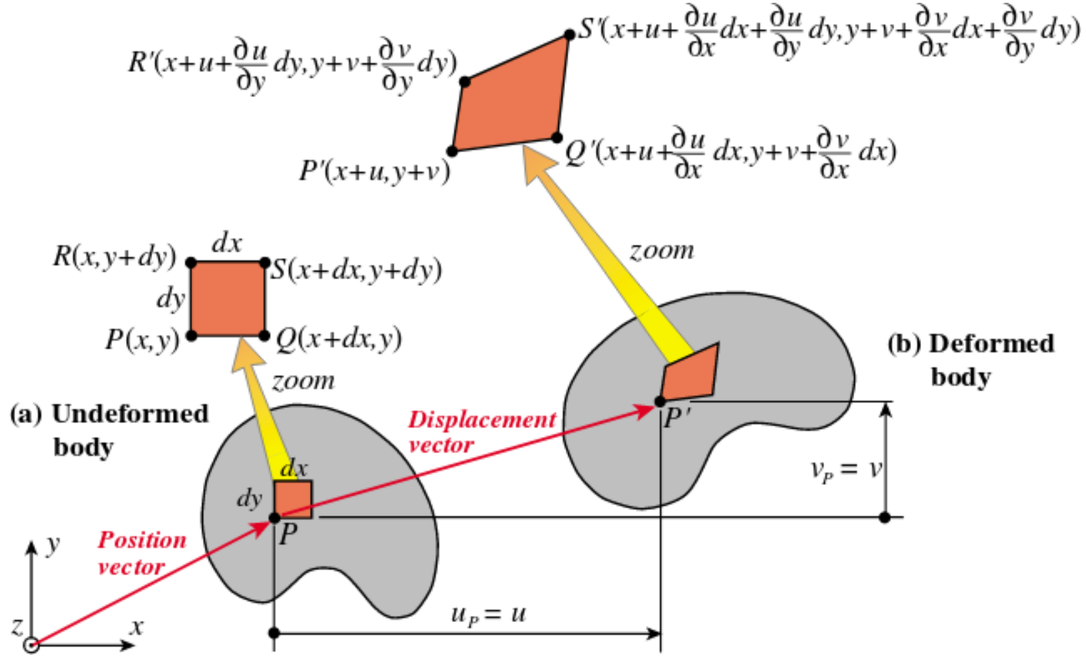
- Green strain tensor $\epsilon = \frac{1}{2}\left(F\,F^T - \mathrm{Id}\right)$

  (+) If φ is a rotation $F = R \Rightarrow \epsilon = 0$

  (-) Non linear in p

- Linearized Cauchy strain $\epsilon = \frac{1}{2}(F^T + F) - \mathrm{Id}$

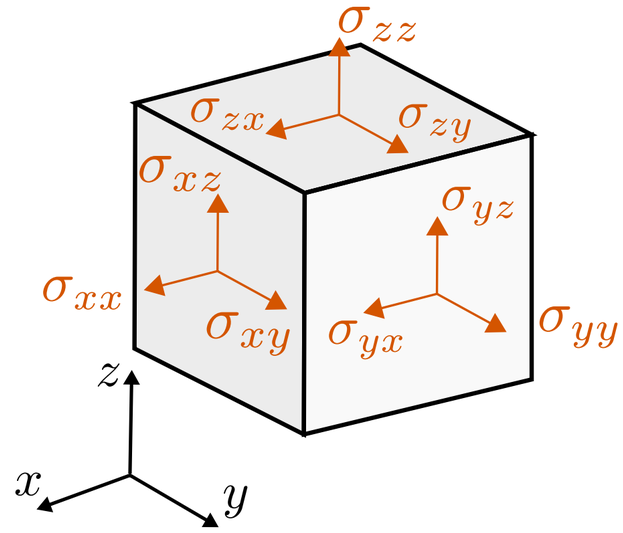
  Used for small deformations



Stress

Stress $\sigma \in \mathbb{R}^{3\times 3}$ describes internal forces (per area unit) induced by the local deformation (strain) in any direction

Constitutive Relation: Relation between stress and strain, characterize a type of material.



For linear constitutive relation:

$\sigma_{ij} = \sum_{k,l} C_{ijkl}\, \epsilon_{kl}$, C: stiffness tensor (81 coefficients)

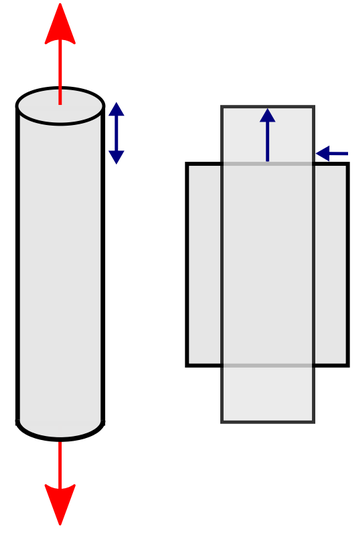
- Strain energy $U(\epsilon) = \sum_{i,j,k,l} C_{ijkl}\, \epsilon_{ij}\, \epsilon_{kl}/2$

For homogeneous isotropic elastic material, constitutive relation simplifies to

$\sigma = 2\mu\, \epsilon + \lambda \mathrm{tr}(\epsilon)\, \mathrm{Id}$, (μ, λ): Lamé parameters

Related to common mechanical modulus : Young' modulus Y and Poisson's ratio ν

$\mu = \frac{Y}{2(1+\nu)}, \lambda = \frac{Y\nu}{(1+\nu)(1-2\nu)}$



Evolution equation

Fundamental principle of dynamics in the entire volume Ω

Change of momentum = External forces (in volume) + Traction (stress applied on exterior surface normals)

$$\Rightarrow \underbrace{\int_\Omega \rho\, p''(t)\,\mathrm{d}\Omega}_{\text{Change of momentum}} = \underbrace{\int_\Omega F(t)\,\mathrm{d}\Omega}_{\text{External forces}} + \underbrace{\int_{\partial\Omega} \sigma\, n\,\mathrm{d}S}_{\text{Traction force on the boundary}}$$

Using divergence theorem $\displaystyle\int_{\partial\Omega} \sigma\, n\,\mathrm{d}S = \int_\Omega \mathrm{div}(\sigma)\,\mathrm{d}\Omega$

Equation in volume satisfied at each position $p \in \Omega$

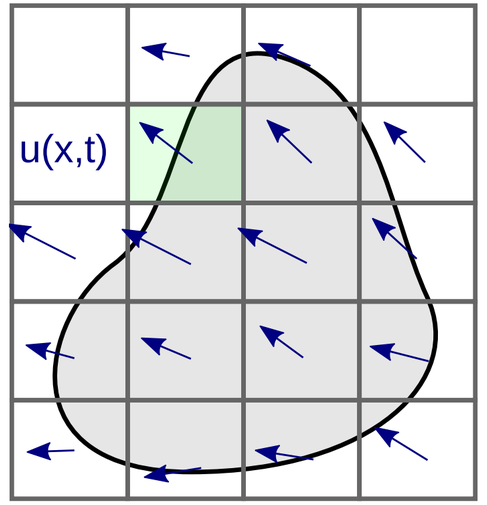
$$\boxed{\rho\, p''(t) = F(t) + \mathrm{div}(\sigma(t))} \qquad \sigma = \begin{pmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \sigma_{zz} \end{pmatrix} \qquad \mathrm{div}(\sigma) = \begin{pmatrix} \frac{\partial \sigma_{xx}}{\partial x} + \frac{\partial \sigma_{yx}}{\partial y} + \frac{\partial \sigma_{zx}}{\partial z} \\ \frac{\partial \sigma_{xy}}{\partial x} + \frac{\partial \sigma_{yy}}{\partial y} + \frac{\partial \sigma_{zy}}{\partial z} \\ \frac{\partial \sigma_{xz}}{\partial x} + \frac{\partial \sigma_{yz}}{\partial y} + \frac{\partial \sigma_{zz}}{\partial z} \end{pmatrix}$$

Euler formulation

In Euler formulation quantities are expressed at fixed position in 3D space.

Deformation described by velocity $u(p,t)$ at a given 3D fixed point $p=(x,y,z)$ at time t.

- Do not require anymore a reference shape
- Usefull for heavily deforming shapes (ex. fluids, gaz).



- Change of speed during dt

$$\frac{\mathrm{d}u}{\mathrm{d}t}(p,t)=\frac{\partial u}{\partial t}+\frac{\partial p}{\partial t}\cdot\frac{\partial u}{\partial p}=\frac{\partial u}{\partial t}+u\cdot\nabla u\,.$$ Called *material derivative*.

- Similarily to Lagrangian derivation:
  - Stress-rate tensor ϵ (rate of change of deformation in a neighborhood of a point) expressed with respect to u $\epsilon=\frac{1}{2}\left(\nabla u+\nabla u^T\right)$
  - Strain-rate tensor σ (rate of change of deformation in a neighborhood of a point).

Equation of motion for a fluid

- Fundamental principle of dynamics on linear momentum

$$\rho \frac{\mathrm{d}u}{\mathrm{d}t} = F + \mathrm{div}(\sigma)$$

$$\Rightarrow \rho \frac{\partial u}{\partial t} = F + \mathrm{div}(\sigma) - \rho\, u \cdot \nabla u\,. \qquad \text{The term } u \cdot \nabla u \text{ is called } \textit{convection}.$$

- External force: weight $F = \rho\, g$

- Stress decomposed into

$$\sigma = \sigma_{viscous} + \sigma_{pressure}$$

$$\sigma_{pressure} = -p\,\mathrm{Id} \text{ } \textit{(pressure acts along normal of surface elements)}$$

$$\rho \frac{\partial u}{\partial t} = \rho\, g - \rho\, u \cdot \nabla u + \mathrm{div}\left(\sigma_{viscous} - p\,\mathrm{Id}\right)$$

$$\Rightarrow \rho \frac{\partial u}{\partial t} = \rho\, g - \rho\, u \cdot \nabla u - \nabla p + \mathrm{div}\left(\sigma_{viscous}\right)$$

Navier-Stokes equation

- Newtonian fluid $\Rightarrow$ Linear relation between strain-rate ϵ and stress-rate $\sigma_{viscous}$
  - $\sigma_{viscous} = 2\mu\,\epsilon = \mu\left(\nabla u + \nabla u^T\right)$, μ constant viscosity parameter
- Incompressible fluid $\Rightarrow \operatorname{div}(u) = 0$

Equation of motion

$$\Rightarrow \rho\,\frac{\partial u}{\partial t} = \rho\,g - \rho\,u\cdot\nabla u - \nabla p + \operatorname{div}\left(\mu\left(\nabla u + \nabla u^T\right)\right)$$

- Noting that $\operatorname{div}(\nabla u^T) = \nabla \operatorname{div}(u) = 0$
- And $\operatorname{div}(\nabla u) = \triangle u$
- Set $\nu = \mu/\rho$

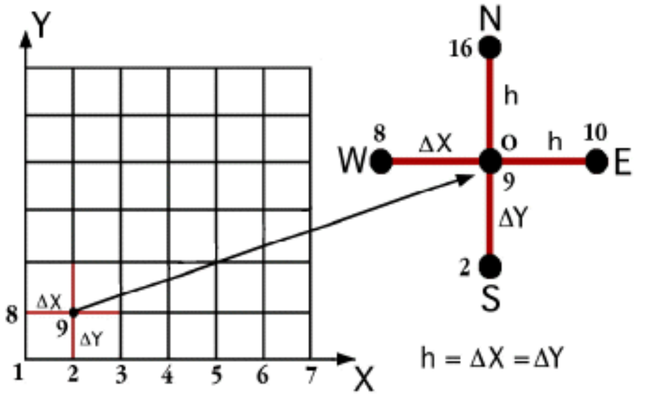
$$\Rightarrow \frac{\partial u}{\partial t} = g - u\cdot\nabla u - \frac{1}{\rho}\nabla p + \nu\triangle u$$

Navier-Stokes equation for incompressible Newtonian fluid.

Numerical solutions

Lagrangian and Euler description leads to PDE (Partial Differential Equations)

In general: no explicit solutions $\Rightarrow$ approximate numerical solution

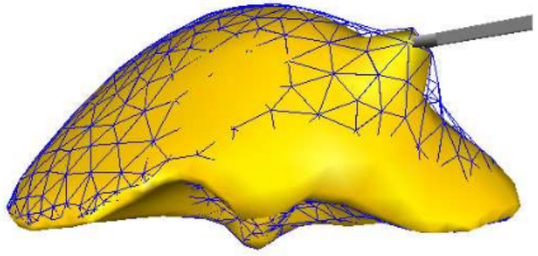


Finite Differences (FD)

- Discretize in space on a grid Δx and time Δt
- Use numerical approximation of derivatives using masks in space and time
  - (+) Very general, simple to setup
  - (+) Works well with rectangular grid (ex. Euler description)
  - (-) Difficult to handle shape boundaries
  - (-) Instabilities

Finite elements method (FEM)

- Discretize the shape into simple elements. Build continuous function on each element.
  In CG: Elements are tetrahedron (in volume). Continuous function are barycentric coordinates (linear interpolation functions).
- Integrate PDE over each element (weak formulation), leads to a linear syst
  - (+) Handle boundaries (ex. Deforming solid)
  - (+) Guarantee on accuracy
  - (-) Complex to set up, and computationally heavy
  - (-) Requires good quality meshing



Physically-based simulation

Particle based systems

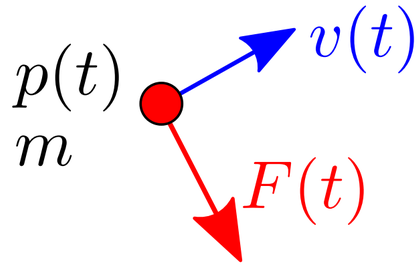
Reminder - Physically-based particle system

1. Description

Particle is fully described by: Position p, Speed v, Mass m

fundamental quantities: position and linear momentum $P = m\,v$

Momentum preserved through time when no external forces are applied



2. Evolution

- Fundamental principle of the dynamics

  Force applied on particle $F(p, v, t)$

$$\begin{cases} p'(t) = v(t) \\ P'(t) = m\,v'(t) = F(p, v, t) \end{cases}$$

- Conservation of energy (ex. kinetic energy $(1/2\,m\,v^2)$+potential energy = const, etc.)

- Lagragian, or Hamiltonian

Reminder - Physically-based particle system

3. Numerical Solution

ODE (Ordinary Differential Equation) formulated as an **Initial Value Problem**

$$\text{ex. } \begin{cases} p'(t) = v(t) \\ m v'(t) = F(p, v, t) \end{cases}, \quad \text{with } v(0) = v_0,\ p(0) = p_0$$

- Discretize in time $t^k = k\,h, \quad h = \Delta t =$ time step.

  $\Rightarrow$ Build a discrete numerical solution $p^k = p(t^k), v^k = v(t^k)$.

- We can consider initially the following iterative scheme

$$\begin{cases} v^{k+1} = v^k + h\,F(p^k, v^k, t^k)/m \\ p^{k+1} = p^k + h\,v^{k+1} \end{cases}$$

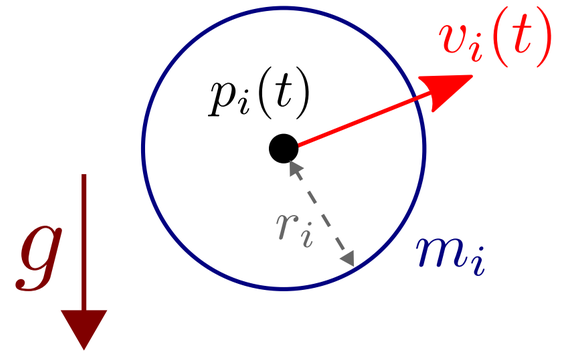
Simple to implement, reasonably OK for simple examples (more details later).

Example : Rigid spheres

System modeling

Particles modeling the center of hard spheres.

- Spheres can collide with surrounding obstacles
- Spheres can collide with each others



- *System*: N particles with position p_i, speed v_i, mass m_i, modeling a sphere of radius r_i.
  - Initial conditions $p_i(0) = p_i^0$, $v_i(0) = v_i^0$

- *Forces*: Single gravity forces $F_i = m_i\, g$. Collisions handled by *impulses*.

- *Temporal evolution*: Fundamental principle of dynamics $v_i(t) = p_i'(t)$, $v_i'(t) = g$

- *Numerical solution*

$$\begin{cases} v^{k+1} = v^k + h\, g \\ p^{k+1} = p^k + h\, v^{k+1} \end{cases}$$

Collision with a plane

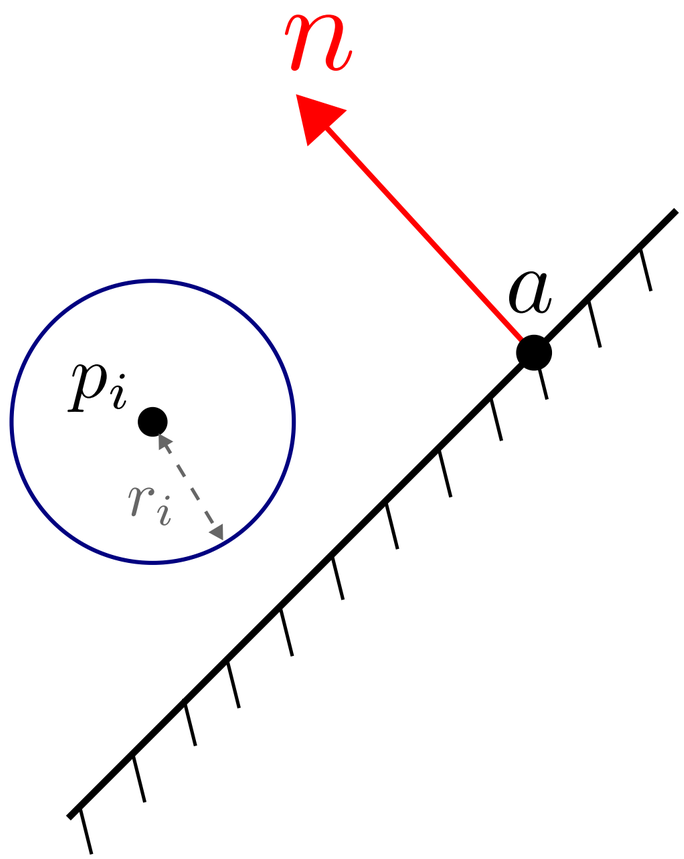
Plane $\mathcal{P}$: parameterized using a point a and its normal n.

$$\{p \in \mathbb{R}^3 \in \mathcal{P} \Rightarrow (p - a) \cdot n = 0\}$$

- Sphere above plane : $(p_i - a) \cdot n > r_i$
- Sphere in collision: $(p_i - a) \cdot n \leq r_i$

- Collision detection algorithm

```
for(int i=0; i<N; ++i)
{
    float detection = dot(p[i]-a, n);
    if (detection <= r[i])
    {
        // ... collision response
    }
}
```



What should we do when a collision is detected

Collision response with plane

Suppose exact contact: $(p_i - a) \cdot n_i = r_i$

Collision response = **Update speed**

Split $v = v_{//} + v_\perp$

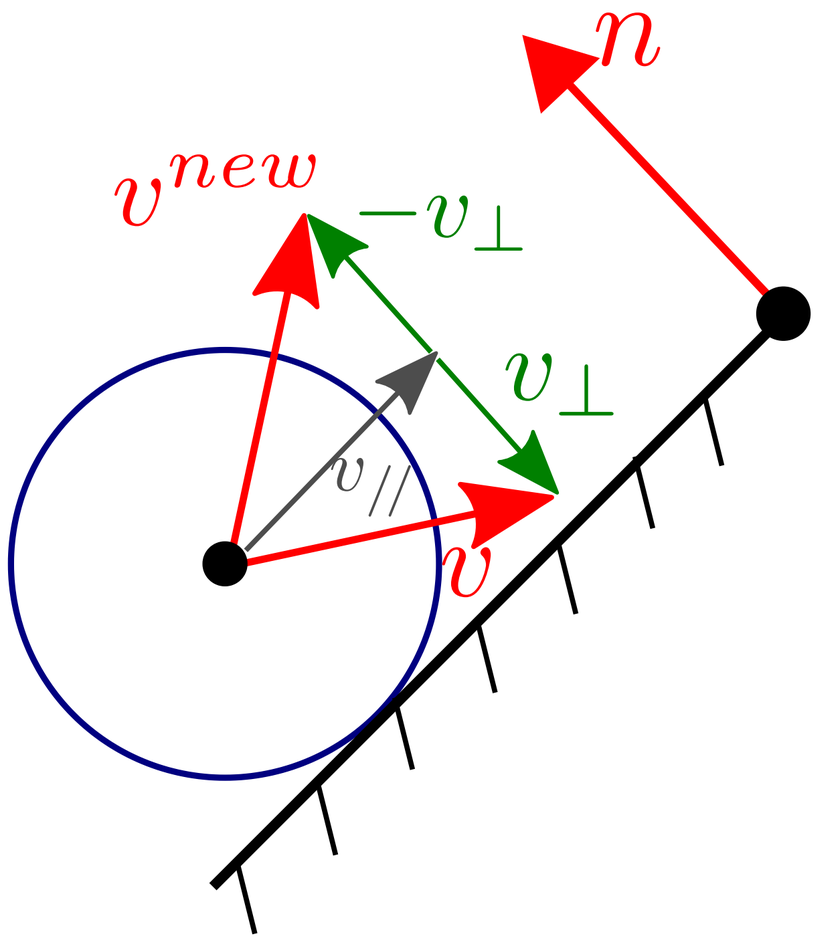
- $v_\perp = (v \cdot n)\, n$
- $v_{//} = v - (v \cdot n) n$

New speed

$v^{new} = \alpha\, v_{//} - \beta\, v_\perp = \alpha\, v - (\beta + \alpha)\, (v \cdot n) n$

$1 - \alpha \in [0, 1]$ Speed loss in $//$ direction (friction)

$1 - \beta \in [0, 1]$ Speed loss in $\perp$ direction (impact)



Result: Collision response

Applying collision response on speed only

Collision response with plane : position

In real case (discrete time) no exact contact, but penetration $(p_i - a) \cdot n_i < r_i$

$\Rightarrow$ Need to compute collision response at contact point.

Three possibilities

(1) Correct position in projecting on the constaint

(+) Simple to implement

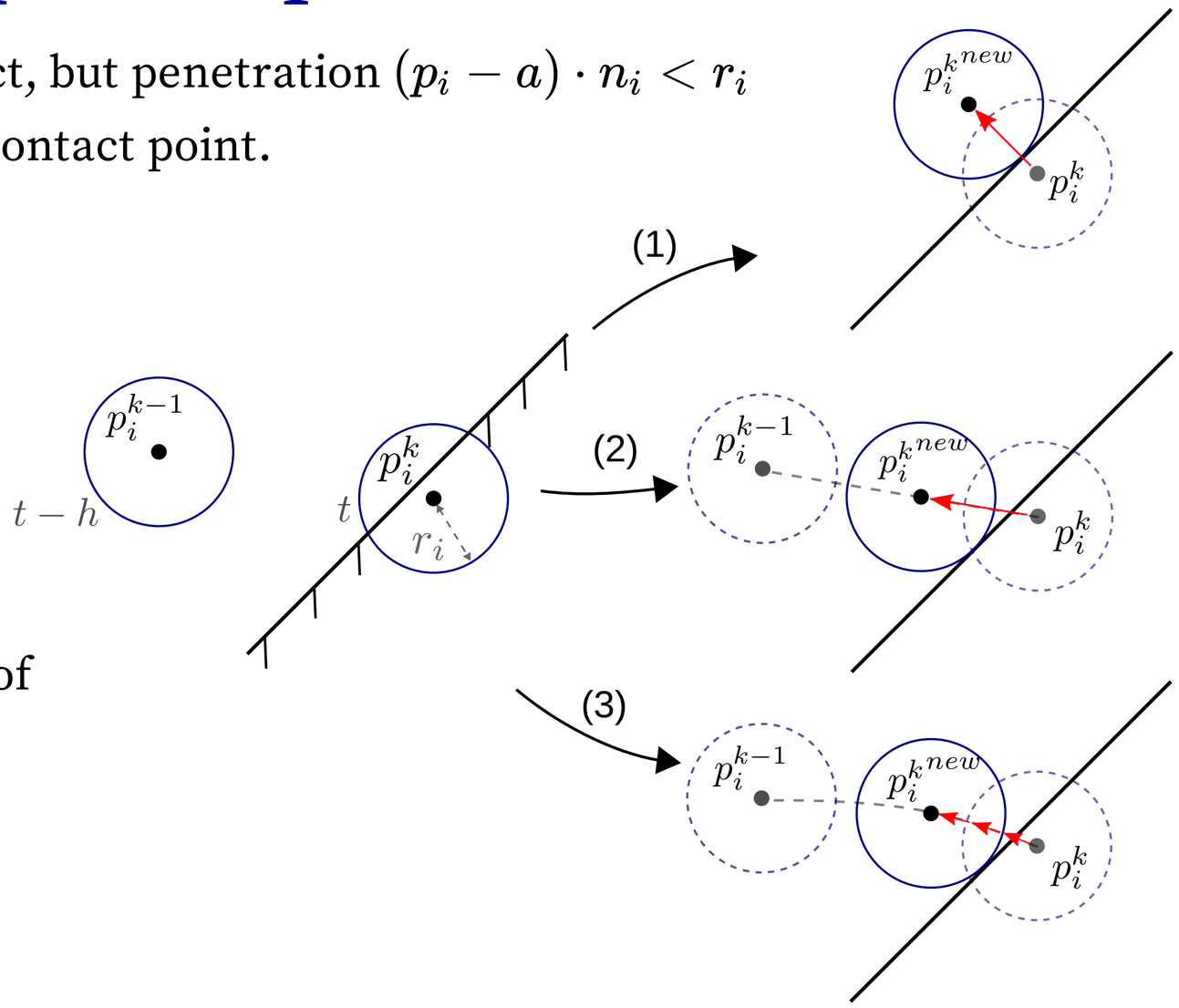
(-) Physically incorrect position

(2) Approximate the correct position

(3) Go backward in to find exact instant of collision

(+) Physically correct

(-) Computationally heavy (binary search, etc.)



Result: Projecting position on plane

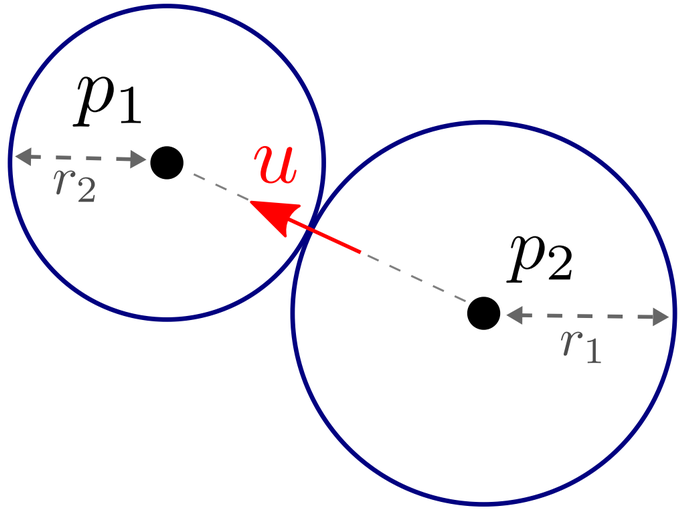
$p_i^{new} = p_i + d\,n$

$d = r_i - (p_i - a) \cdot n_i$: distance of penetration

Collision between sphere

Given 2 spheres $(p_1, v_1, r_1, m_1), (p_2, v_2, r_2, m_2)$.

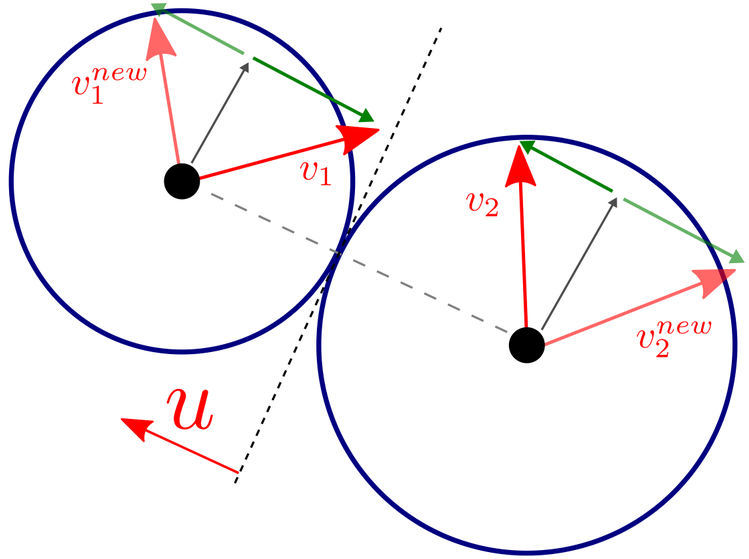
Collision when $\|p_1 - p_2\| \leq r_1 + r_2$



Response to collision in speed

$$\begin{cases} v_1^{new} = v_1 - \dfrac{j}{m_1} u \\ v_2^{new} = v_2 + \dfrac{j}{m_2} u \end{cases}$$

Rem. If $m_1 = m_2$, spheres switch between their $\perp$ speed



j: impulse (sudden change of momentum)

$$j = 2 \frac{(v_1 - v_2) \cdot u}{\frac{1}{m_1} + \frac{1}{m_2}}$$

$$u = \frac{p_1 - p_2}{\|p_1 - p_2\|}$$

Collision response with sphere - Derivation

- Conservation of momentum of the center of mass (COM)

$m_1 v_1 + m_2 v_2 = m_1 v_1^{new} + m_2 v_2^{new}$

$\Rightarrow m_2(v_2^{new} - v_2) = -m_1(v_1^{new} - v_1) = J$, J: the impulse

The impulse is orthogonal to the separating plane between the two surfaces

$J = j\,u,\ u = (p_1 - p_2)/\|p_1 - p_2\|$

- Conservation of the kinetic energy (elastic collision)

$1/2\,m_1\,v_1^2 + 1/2\,m_2\,v_2^2 = 1/2\,m_1\,(v_1^{new})^2 + 1/2\,m_2\,(v_2^{new})^2$

$\Rightarrow m_1 v_1^2 + m_2 v_2^2 = m_1\left(v_1 - \frac{j}{m_1}u\right)^2 + m_2\left(v_2 + \frac{j}{m_2}u\right)^2$

$\Rightarrow 0 = -2\,j\,v_1 \cdot u + \frac{j^2}{m_1} + 2\,j\,v_2 \cdot u + \frac{j^2}{m_2}$

$\Rightarrow j = 2\frac{v_1 - v_2}{1/m_1 + 1/m_2} \cdot u$

Final result

Reprojection on the contact surface

$$p_1^{new} = p_1 + \frac{m_2}{m_1+m_2} d\,u$$
$$p_2^{new} = p_2 - \frac{m_1}{m_1+m_2} d\,u$$
$$d = r_1 + r_2 - \|p_1 - p_2\|$$

Numerical solution of ODE

General formulation

Consider relation given a system of **first order** differential equation

Mechanical systems are often expressed as

- *scalar equation of second order in* p
- *system of first order in* (p, v)

In general, we can write

$$u'(t) = \mathcal{F}(u(t), t)$$

If $\mathcal{F}$ is an affine function in u

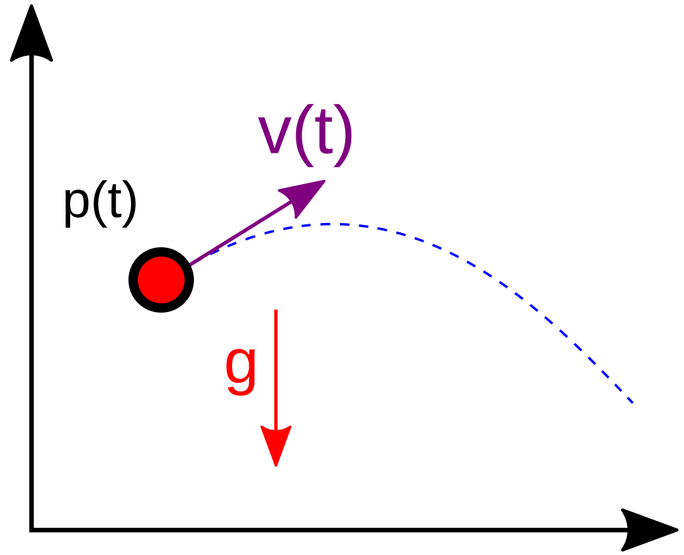
$$u'(t) = \mathrm{A}(t)\, u(t) + b(t)$$

When A is constant through time

$$u'(t) = \mathrm{A}\, u(t) + b(t)$$

Example: Free fall under gravity

- Force $F(t) = m\,g$
- Second order differential equation: $p''(t) = g$
- First order system $\underbrace{\begin{pmatrix} p \\ v \end{pmatrix}'(t)}_{u'(t)} = \underbrace{\begin{pmatrix} v(t) \\ g \end{pmatrix}}_{\mathcal{F}(u,t)}$
- Linear system $\underbrace{\begin{pmatrix} p \\ v \end{pmatrix}'(t)}_{u'(t)} = \underbrace{\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}}_{\mathrm{A}} \underbrace{\begin{pmatrix} p \\ v \end{pmatrix}(t)}_{u(t)} + \underbrace{\begin{pmatrix} 0 \\ g \end{pmatrix}}_{b(t)}$

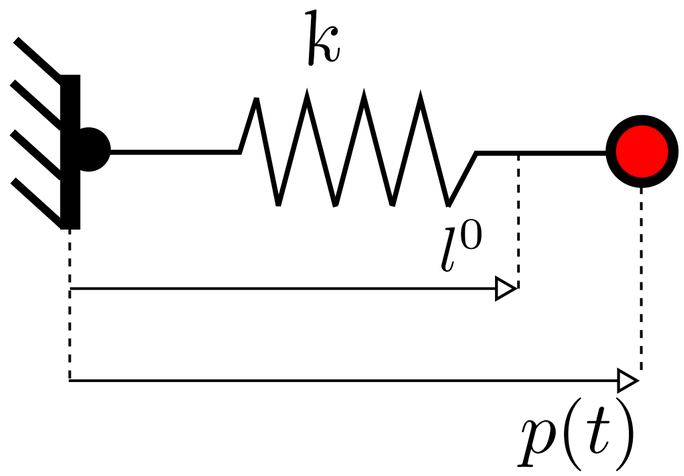


- Exact solution known: $p(t) = \frac{1}{2} g\,t^2 + v_0\,t + p_0$

Note: variables are vectors - matrix can be expressed by block, or per components.

Example: 1D spring (/Harmonic oscillator)

- Force $F(t) = -k\,(p(t) - l^0)$, k spring stiffness, l^0 rest length
- Second order differential equation: $m\,p''(t) + k\,p(t) = k\,l^0$



- First order system $\underbrace{\begin{pmatrix} p \\ v \end{pmatrix}'(t)}_{u'(t)} = \underbrace{\begin{pmatrix} v(t) \\ -k/m\,(p(t) - l^0) \end{pmatrix}}_{\mathcal{F}(u,t)}$

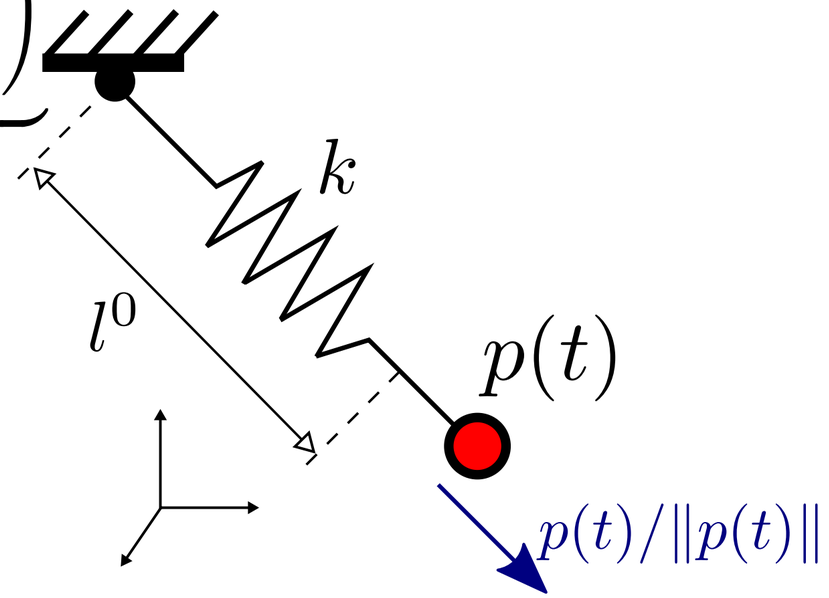
- Linear system $\underbrace{\begin{pmatrix} p \\ v \end{pmatrix}'(t)}_{u'(t)} = \underbrace{\begin{pmatrix} 0 & 1 \\ -k/m & 0 \end{pmatrix}}_{\mathrm{A}} \underbrace{\begin{pmatrix} p \\ v \end{pmatrix}(t)}_{u(t)} + \underbrace{\begin{pmatrix} 0 \\ k/m\,l^0 \end{pmatrix}}_{b}$

- Exact solution known: $p(t) = A\,\sin(\omega\,t + \varphi) + l^0$

$$\omega = \sqrt{k/m},\ A^2 = (p^0 - l^0)^2 + \left(\frac{v_0}{\omega}\right)^2,\ \tan(\varphi) = \frac{p^0 - l^0}{v^0/\omega}$$

Example: 3D mass spring

- Force $F(t) = m\,g - k\,(\|p(t)\| - l^0)\,\frac{p(t)}{\|p(t)\|}$, k spring stiffness, l^0 rest length
- Second order differential equation: $m\,p''(t) = mg - k\,(\|p(t)\| - l^0)\,\frac{p(t)}{\|p(t)\|}$
- First order system $\underbrace{\begin{pmatrix} p \\ v \end{pmatrix}'(t)}_{u'(t)} = \underbrace{\begin{pmatrix} v(t) \\ g - k/m\,(\|p(t)\| - l^0)\,\frac{p(t)}{\|p(t)\|} \end{pmatrix}}_{\mathcal{F}(u,t)}$
- Not linear
- No simple explicit solution



Existence of solution

Solving the physical system = Solving the **Initial Value Problem** (IVP)

- Find the solution of $u'(t) = \mathcal{F}(u, t), t \geq 0$
- With initial condition $u(t = 0) = u_0$

Cauchy-Lipschitz (/Picard-Lindelof) theorem states that

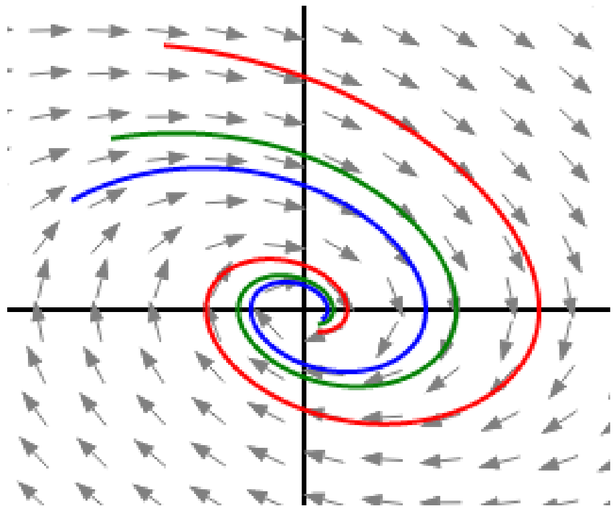
If $\mathcal{F}$ is Lipschitz with respect to u and continuous with respect to t

Then there exists a unique solution $u(t)$.

The solution is called the **integral curve** of the IVP.

- $\mathcal{F}$ can be seen as a vector field

  (Vector field in 6D for $p(t) \in \mathbb{R}^3, v(t) \in \mathbb{R}^3$)
- Solution is a path along this vector field passing by u_0



Solving the ODE exactly

Equation is linear

Homogeneous, constant coefficients

$u'(t) = \mathrm{A}u(t), u(0) = u_0$

$\Rightarrow u(t) = u_0 \ \exp(\mathrm{A}\, t)$

Non-Homogeneous, constant coefficients

$u'(t) = \mathrm{A}u(t) + b(t), u(0) = u_0$

$$\Rightarrow u(t) = u_0 \ \exp(\mathrm{A}\, t) + \exp(\mathrm{A}\, t) \int_{t'=0}^{t} b(t') \exp(-\mathrm{A}\, t')\, \mathrm{d}t'$$

Homogeneous, variable coefficients

$u'(t) = \mathrm{A}(t)\, u(t), u(0) = u_0$

$\Rightarrow$ No closed-form solution in the general case

- *Rem.* Unfortunately, in general, $u(t)$ is **not** $u_0 \ \exp\left(\int_0^t \mathrm{A}(t')\, \mathrm{d}t'\right)$

Equation is non linear

No general method

$\Rightarrow$ Numerical approaches are required most of the time

Numerical solution

1st order Explicit Euler

Naive numerical scheme: Approximation of the derivative

$$\frac{u^{k+1} - u^k}{h} = \mathcal{F}(u^k, t^k)$$

$$\Rightarrow u^{k+1} = u^k + h\,\mathcal{F}(u^k, t^k)$$

In the linear case

$$u^{k+1} = (\mathrm{I} + h\,\mathrm{A})\; u^k + h\,b^k$$

Pro : very easy to implement

Is u^k a good approximation of the true solution $\tilde{u}(t^k)$?

Explicit Euler - study case

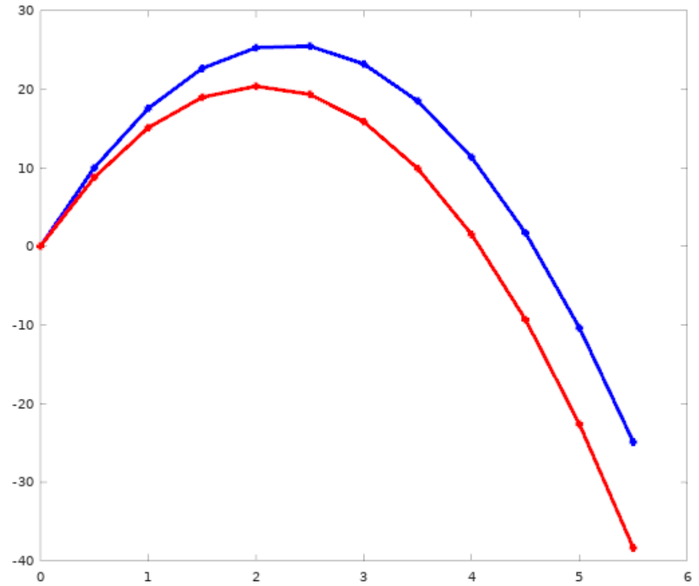
Free fall under gravity

- True solution $\tilde{u}(k\,h) = p_0 + (k\,h)v_0 + \frac{(k\,h)^2}{2}\,g$

- Numerical scheme: $\begin{pmatrix} p \\ v \end{pmatrix}^{k+1} = \begin{pmatrix} 1 & h \\ 0 & 1 \end{pmatrix} \begin{pmatrix} p \\ v \end{pmatrix}^{k} + \begin{pmatrix} 0 \\ h\,g \end{pmatrix}$

- Numerical solution: $p^k = p_0 + k\,h\,v_0 + \frac{k(k-1)}{2}\,h^2\,g$

$\Rightarrow$ **Not exact** : Error $e^k = |u^k - \tilde{u}(k\,h)| = \frac{g}{2} k\,h^2$



red: true solution

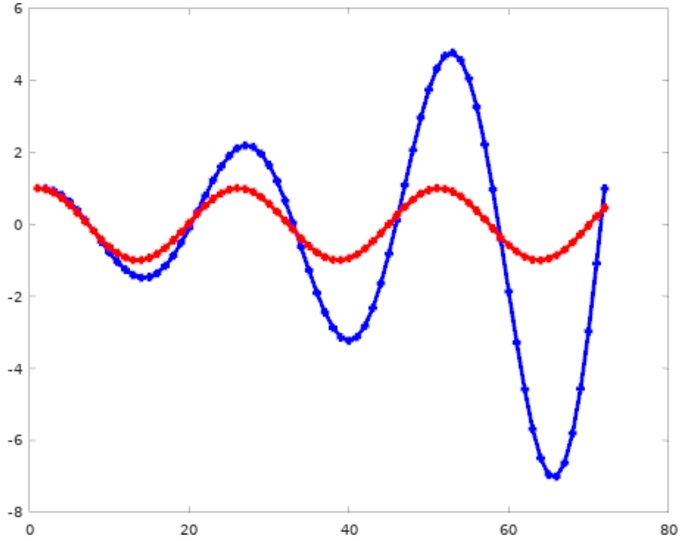
blue: numerical solution

Explicit Euler - study case

1D spring

- True solution: permanent oscillation

- Numerical scheme: $\begin{pmatrix} p \\ v \end{pmatrix}^{k+1} = \begin{pmatrix} 1 & h \\ -K/m\,h & 1 \end{pmatrix} \begin{pmatrix} p \\ v \end{pmatrix}^{k} + \begin{pmatrix} 0 \\ K/m\,h\,l^0 \end{pmatrix}$



red: true solution

red: numerical solution

- Numerical solution diverge to ∞
- Worse than bad accuracy for Graphics

Explicit Euler - study case

1D spring: Analysis of the system energy

- Energy $E = \frac{1}{2}mv^2 + \frac{K}{2}(p - l^0)^2$

$$E^{k+1} = \frac{1}{2}m\left(-\frac{K}{m}\Delta t\left(p^k - l^0\right) + v^k\right)^2 + \frac{1}{2}K\left(p^k + \Delta t\, v^k - l^0\right)^2$$

$$E^{k+1} = \underbrace{\frac{1}{2}m\left(v^k\right)^2 + \frac{1}{2}K(p^k - l^0)^2}_{E^k} + \frac{1}{2}\left[\underbrace{\frac{K^2}{m}(\Delta t)^2(p^k - l^0)^2}_{>0} - \underbrace{2\,K\Delta t\left(p^k - l^0\right)\,v^k + 2K\Delta t\left(p^k - l^0\right)v^k}_{=0} + \underbrace{K(\Delta t)^2\left(v^k\right)^2}_{>0}\right]$$

$$E^{k+1} = E^k + \epsilon\left(\Delta t\right)^2, \epsilon > 0$$

$\Rightarrow$ gain of energy

$\Rightarrow$ divergence

Accuracy of a numerical method - general definition

- Define the **local truncation error** τ:

  Error accumulated during one step, assuming perfect knowledge of the true solution

  $\tau_k = \|\tilde{u}(t^k) - u^k\|$, assuming $u^{k-1} = \tilde{u}(t^{k-1})$

- A numerical scheme is said to be accurate of order k, if its local truncation error is in $\mathcal{O}(h^{k+1})$.

Explicit Euler is of order 1*, at every step we add an error in* $\mathcal{O}(h^2)$.

Stability of a numerical method - general definition

- Classical stability of a method studied on $u'(t) = \lambda u(t), \lambda \in \mathbb{C}$.
  - The true solution $\tilde{u}(t) = \exp(\lambda t)$ converge if $\mathbb{R}_e(\lambda) \leq 0$.
  - For linear system, λ refers to eigenvalues of A.
  - For non linear system, λ refers to eigenvalues of the Jacobian of $\mathcal{F}$

- A numerical method is *unconditionnaly stable* if $\mathbb{R}_e(\lambda) \leq 0 \Rightarrow$ stable discrete solution.
- Otherwise, it is conditionnally stable/unstable.
- Region of stability: Set of conditions on λ such that the discrete solution doesn't diverge.

Rem.

- A numerical solution can diverge even when the true ODE solution converge when using unstable numerical method.
- Converseley, a numerical solution can converge even when the true ODE solution diverge when using stable numerical method.
- Stability ! = Accuracy.

Stability analysis of explicit Euler

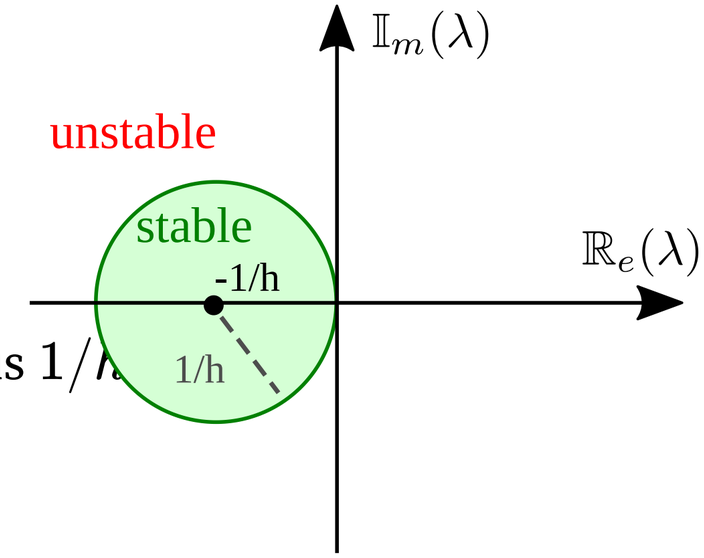
$u'(t) = \lambda\, u(t)$

$\Rightarrow u^{k+1} = u^k + \lambda\, u^k$ using explicit Euler

$\Rightarrow u^{k+1} = (1 + \lambda\, h)\, u^k$

$\Rightarrow$ Stable if $|1 + \lambda\, h| \leq 1$ (conditionnal stability)

$|1/h + \lambda| \leq 1/h$: Interior of a disc centered on $(-1/h, 0)$ with radius $1/h$



Rem. For 1D elastic spring

$\lambda = \pm i\sqrt{K/m}$

$\Rightarrow |1 + i\sqrt{K/m}h| = \sqrt{1 + K/m\, h^2} > 1$

$\Rightarrow$ Explicit euler is always unstable on the elastic spring problem.

Other approach: Implicit method

Other approach: Implicit Euler

Explicit Euler

$u^{k+1} = u^k + h\,\mathcal{F}(u^k, t^k)$

u^k is known to compute u^{k+1}

- In the linear case

$u^{k+1} = (\mathrm{I} + h\,\mathrm{A})\,u^k + h\,b(t^k)$

Implicit Euler

$u^{k+1} = u^k + h\,\mathcal{F}(u^{k+1}, t^{k+1})$

u^{k+1} appears in RHS

$\Rightarrow$ Need to solve $u^{k+1} - h\,\mathcal{F}(u^{k+1}, t^{k+1}) = u^k$

- In the linear case : solve a linear system

$u^{k+1} = u^k + h\,\left(\mathrm{A}u^{k+1} + b(t^{k+1})\right)$

$\Rightarrow u^{k+1} = (\mathrm{I} - h\,\mathrm{A})^{-1}\,\left(u^k + h\,b(t^{k+1})\right)$

Implicit Euler - study case

Free fall under gravity

- True solution $\tilde{u}(k\,\Delta t) = p_0 + (k\,\Delta t)v_0 + \frac{(k\,\Delta t)^2}{2}\,g$

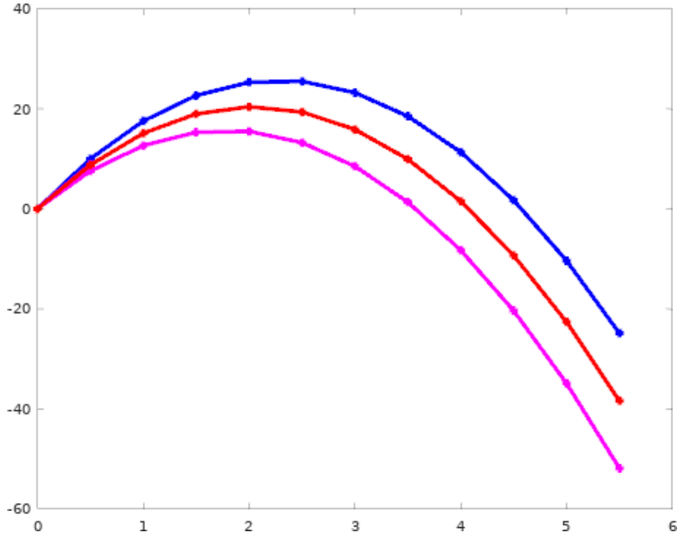
- Numerical scheme: $\begin{pmatrix} p \\ v \end{pmatrix}^{k+1} = \begin{pmatrix} 1 & -h \\ 0 & 1 \end{pmatrix}^{-1} \left(\begin{pmatrix} p \\ v \end{pmatrix}^{k} + \begin{pmatrix} 0 \\ g \end{pmatrix} \right)$

$$\Rightarrow \begin{pmatrix} p \\ v \end{pmatrix}^{k+1} = \begin{pmatrix} 1 & h \\ 0 & 1 \end{pmatrix} \begin{pmatrix} p \\ v \end{pmatrix}^{k} + \begin{pmatrix} h^2\,g \\ h\,g \end{pmatrix}$$

- Numerical solution: $p^k = p_0 + (k\,\Delta t)v_0 + \frac{k(k+1)}{2}\,(\Delta t)^2\,g$

$\Rightarrow$ Not exact : Error $e^k = |u^k - \tilde{u}(k\,\Delta t)| = \frac{k}{2}(\Delta t)^2 g$

Same error magnitude than explicit Euler



- *red: True solution*
- *magenta: Implicit Euler*
- *blue: Explicit Euler*

Implicit Euler - study case

1D spring

- True solution: permanent oscillations

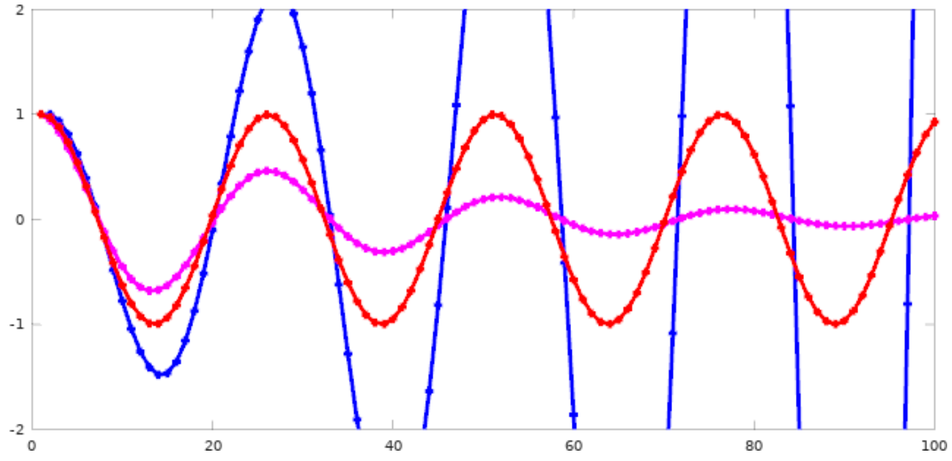
- Numerical scheme: $\begin{pmatrix} p \\ v \end{pmatrix}^{k+1} = \begin{pmatrix} 1 & -h \\ K/m\,h & 1 \end{pmatrix}^{-1} \left(\begin{pmatrix} p \\ v \end{pmatrix}^{k} + \begin{pmatrix} 0 \\ K/m\,h\,l^0 \end{pmatrix} \right)$

$$\Rightarrow \begin{pmatrix} p \\ v \end{pmatrix}^{k+1} = \frac{1}{1+h^2\,K/m} \left(\begin{pmatrix} 1 & h \\ -K/m\,h & 1 \end{pmatrix} \begin{pmatrix} p \\ v \end{pmatrix}^{k} + \begin{pmatrix} K/m\,h^2\,l^0 \\ K/m\,h\,l^0 \end{pmatrix} \right)$$

Eigenvalues of $(\mathrm{I} - \mathrm{A}h)^{-1}$ are $\frac{1 \pm i\sqrt{\frac{K}{m}}h}{1+\frac{K}{m}h^2}$

$$\Rightarrow \left| \frac{1 \pm i\sqrt{\frac{K}{m}}h}{1+\frac{K}{m}h^2} \right| = \frac{1}{\sqrt{1+\frac{K}{m}h^2}} < 1 \Rightarrow \text{always converge}$$

Even if the true solution oscillates



Stability analysis of Implicit Euler

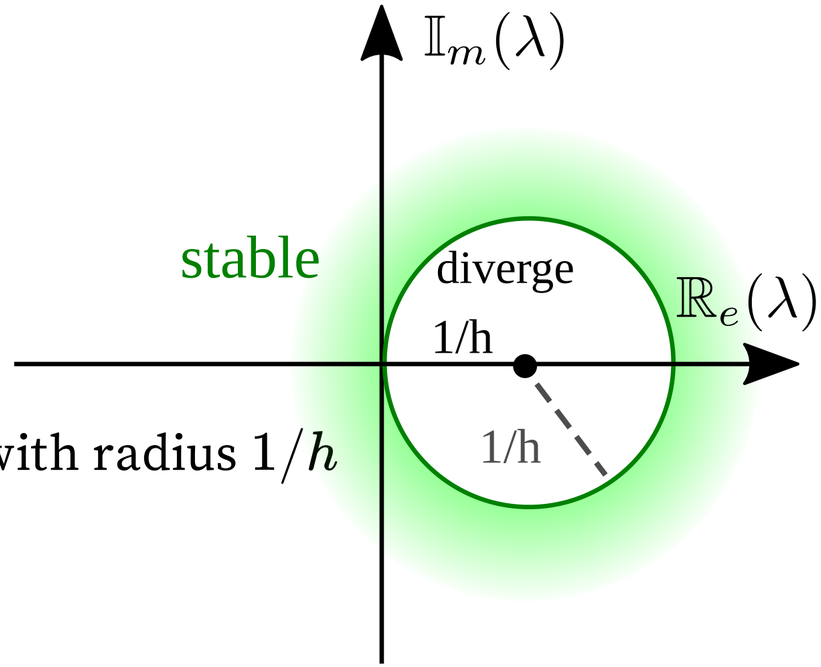
What are the general conditions for which $u'(t) = \lambda\, u(t)$ converge using Implicit Euler ?

$u^{k+1} = u^k + h\,\lambda\, u^{k+1}$

$\Rightarrow (1 - h\,\lambda)\, u^{k+1} = u^k$

$\Rightarrow u^{k+1} = \frac{1}{1-h\,\lambda}\, u^k$

Stability condition: $\left|\frac{1}{1-h\lambda}\right| \leq 1 \Rightarrow |1 - h\lambda| \geq 1$



$|1/h - \lambda| \geq 1/h$: exterior of a disc centered on $(1/h, 0)$ with radius $1/h$

$\Rightarrow$ Fully enclose $\mathbb{R}_e(\lambda) \leq 0$

$\Rightarrow$ Implicit euler is unconditionally stable

Rem. May converges even when the true solution does not.

Comparison Explicit Euler / Implicit Euler

Explicit Euler

- $u^k = u^k + h\,\mathcal{F}(u^k, t^k)$
- $u^k = (\mathrm{I} + h\,\mathrm{A})\,u^k + h\,b(t^k)$
- Accuracy: 1st Order
- Stability:

  Conditional $|1 + h\lambda| \geq 1$

  Solution to spring model diverges

Implicit Euler

- $u^{k+1} = u^k + h\,\mathcal{F}(u^{k+1}, t^{k+1})$
- $u^k = (\mathrm{I} - h\,\mathrm{A})^{-1}\left(u^k + h\,b(t^{k+1})\right)$
- Accuracy: 1st Order
- Stability:

  Unconditionnaly stable

  Solution to spring model converge to l^0

Higher order method

Higher accuracy can be achived using Taylor expansion.

ex. $u^{k+1} = u^k + h\,\mathcal{F}(u^k, t^k) + \dfrac{h^2}{2}\dfrac{\mathrm{d}\mathcal{F}}{\mathrm{d}t}(u^k, t^k)$

- Second order accurate: **2nd order explicit Euler**

- Can achieve arbitrary accuracy.
- In practice: computing derivatives of $\mathcal{F}$ is complex.

$\Rightarrow$ Not often used

Runge-Kutta methods

Runge-Kutta numerical methods allow to

- Reach higher order accuracy
- Only involve knowledge of $\mathcal{F}$, without its derivatives
- Involves several evaluation of $\mathcal{F}$ at various points

2nd order Runge-Kutta: RK2

$$u^{k+1} = u^k + \frac{1}{2}\left(k_1 + k_2\right)$$
$$k_1 = h\,\mathcal{F}(u^k, t^k)$$
$$k_2 = h\,\mathcal{F}(u^k + k_1, t^k + h)$$

Accuracy, stability ?

Computing Accuracy

Reminder: Local truncation error

$\tau_k = \|\tilde{u}(t^k + h) - u(t^k + h)\|$

Assuming true solution at time t_k: $\tilde{u}(t^k) = u(t^k)$ (and all its derivatives).

General methodology to compute accuracy

(1) Compute Taylor expansion of the numerical scheme

- Express $u(t^k + h)$ at position (u^k, t^k)

(2) Compute Taylor expansion of the exact differential equation

- Express $\tilde{u}(t^k + h)$ at position (u^k, t^k), given $\tilde{u}'(t^k) = \mathcal{F}(u^k, t^k)$

(3) Compare both expressions

RK2 - Accuracy

(1) Compute Taylor expansion of the numerical scheme

RK2 definition: $u(t^k + h) = u^k + \frac{1}{2}(k_1 + k_2)$

$k_1 = h\,\mathcal{F}(u^k, t^k)$

$k_2 = h\,\mathcal{F}(u^k + k_1, t^k + h)$

$$\Rightarrow k_2 = h\left(\mathcal{F}(u^k, t^k) + k_1\,\frac{\partial \mathcal{F}}{\partial u}(u^k, t^k) + h\,\frac{\partial \mathcal{F}}{\partial t}(u^k, t^k) + \mathcal{O}(h^2)\right) \quad \text{(differential on } k_1 \text{ and on } h\text{)}$$

$$\Rightarrow k_2 = h\mathcal{F} + h^2\mathcal{F}\frac{\partial F}{\partial u} + h^2\frac{\partial \mathcal{F}}{\partial t} + \mathcal{O}(h^3) \qquad \textit{(dropping parameters } (u^k, t^k)\textit{)}$$

$$u(t^k + h) = u^k + \frac{1}{2}\left[h\mathcal{F} + \left(h\,\mathcal{F} + h^2\,\frac{\partial \mathcal{F}}{\partial u}\,\mathcal{F} + h^2\,\frac{\partial \mathcal{F}}{\partial t} + \mathcal{O}(h^3)\right)\right]$$

$$\Rightarrow u(t^k + h) = u^k + h\,\mathcal{F} + \frac{h^2}{2}\left(\frac{\partial \mathcal{F}}{\partial u}\,\mathcal{F} + \frac{\partial F}{\partial t}\right) + \mathcal{O}(h^3)$$

RK2 - Accuracy

(2) Compute Taylor expansion of the exact differential equation

$$\tilde{u}(t^k + h) = \tilde{u}(t^k) + h\,\tilde{u}'(t^k) + \frac{h^2}{2}\tilde{u}''(t^k) + \mathcal{O}(h^3)$$

Assuming exact solution at t^k: $\tilde{u}^{(n)}(t^k) = u^{(n)}(t^k)$ + Injecting the ODE relation $u'(t^k) = \mathcal{F}(u^k, t^k)$

$$\Rightarrow \tilde{u}(t^k + h) = u(t^k) + h\,\mathcal{F}(u^k, t^k) + \frac{h^2}{2}\frac{\mathrm{d}\mathcal{F}}{\mathrm{d}t}(u^k, t^k) + \mathcal{O}(h^3)$$

$$\Rightarrow \tilde{u}(t^k + h) = u(t^k) + h\,\mathcal{F}(u^k, t^k) + \frac{h^2}{2}\left(\frac{\partial \mathcal{F}}{\partial u}(u^k, t^k)\,u'(t^k) + \frac{\partial \mathcal{F}}{\partial t}(u^k, t^k)\right) + \mathcal{O}(h^3)$$

Drop parameter $(u^k, t^k) \Rightarrow \tilde{u}(t^k + h) = u^k + h\,\mathcal{F} + \frac{h^2}{2}\left(\frac{\partial \mathcal{F}}{\partial u}\,\mathcal{F} + \frac{\partial F}{\partial t}\right) + \mathcal{O}(h^3)$

(3) Compare both epxressions

$$\tau_k = \|\tilde{u}(t^k + h) - u(t^k + h)\| = \mathcal{O}(h^3)$$

$\Rightarrow$ **RK2 is 2nd order accurate**

RK2 - Stability

Study $u'(t) = \lambda\, u(t) \Rightarrow \mathcal{F}(u,t) = \lambda\, u(t)$

- $k_1 = h\,\mathcal{F}(u^k, t^k) = h\,\lambda\,u^k$
- $k_2 = h\,\mathcal{F}(u^k + k_1, t^k + h) = h\,\lambda\,(u^k + h\lambda\,u^k) = (h\lambda + h^2\lambda^2)\,u^k$

$\Rightarrow u^{k+1} = u^k + \frac{1}{2}(k_1 + k_2)$

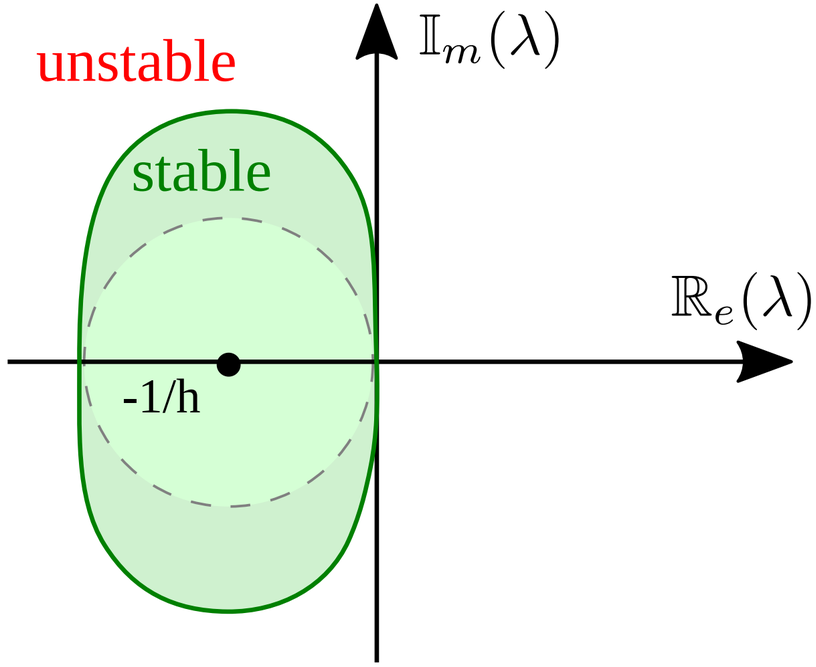
$\Rightarrow u^{k+1} = (1 + h\lambda + \dfrac{h^2}{2}\lambda^2)\,u^k$

Stability condition $\left|1 + h\lambda + \dfrac{h^2}{2}\lambda^2\right| \leq 1$

$\Rightarrow$ RK2 is conditionnaly stable (explicit method)

More stable than explicit Euler

(Still unstable for oscillatory 1D spring)



Other Runge-Kutta methods

Midpoint method (2nd order accuracy)

$$
\begin{aligned}
k_1 &= h\,\mathcal{F}(u^k, t^k) \\
k_2 &= h\,\mathcal{F}(u^k + k_1/2, t^k + h/2) \\
u^{k+1} &= u^k + h\,k_2
\end{aligned}
$$

RK4 (Classical Runge-Kutta)

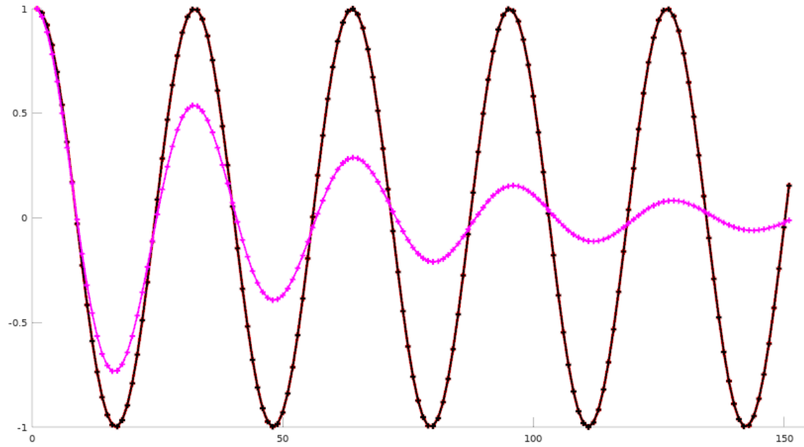
$$
\begin{aligned}
k_1 &= h\,\mathcal{F}(u^k, t^k) \\
k_2 &= h\,\mathcal{F}\left(u^k + \frac{k_1}{2}, t^k + \frac{h}{2}\right) \\
k_3 &= h\,\mathcal{F}\left(u^k + \frac{k_2}{2}, t^k + \frac{h}{2}\right) \\
k_4 &= h\,\mathcal{F}\left(u^k + k_3, t^k + h\right) \\
u^{k+1} &= u^k + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)
\end{aligned}
$$

RK4 - Case study

- Rk4 conditionally stable for oscillating spring

  Not permanent oscillations: Slight decrease of magnitude through time

- Large improvement of accuracy compared to implicit Euler



$h = 0.2$, error rk4 $\simeq 10^{-4}$

- *red: true solution*
- *black: rk4*
- *magenta: implicit Euler*

Stability region: $\left|1 + h\lambda + \frac{h^2}{2}\lambda + \frac{h^3}{6}\lambda^3 + \frac{1}{24}h^4\lambda^4\right| \leq 1$

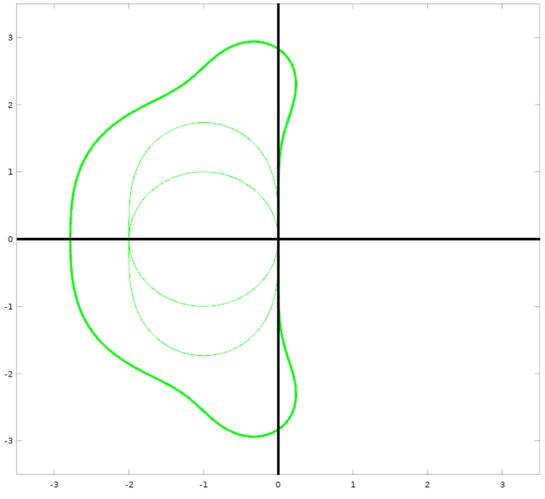
In the case of oscillating spring: $\lambda = i\sqrt{K/m} = i\omega$

Stable if $\left|1 + ih\omega - \frac{h^2}{2}\omega^2 - i\frac{h^3}{6}\omega^3 + \frac{h^4}{24}\omega^4\right| \leq 1$

$\Rightarrow 1 - \frac{h^6\omega^6}{72} + \frac{h^8\omega^8}{576} \leq 1$

$\Rightarrow h^6\,\omega^6\,(h^2\omega^2 - 8)/576 \leq 0$

$\Rightarrow h \leq \frac{2^{3/2}}{\omega} = \frac{2^{3/2}}{\sqrt{K/m}} \simeq \frac{2.8}{\sqrt{K/m}}$



Dormand Prince (ode45)

$$
\begin{aligned}
k_1 &= h\,\mathcal{F}(u^k, t^k) \\
k_2 &= h\,\mathcal{F}\left(u^k + \frac{1}{5}k_1, t^k + \frac{1}{5}h\right) \\
k_3 &= h\,\mathcal{F}\left(u^k + \frac{3}{40}k_1 + \frac{9}{40}k_2, t^k + \frac{3}{10}h\right) \\
k_4 &= h\,\mathcal{F}\left(u^k + \frac{44}{45}k_1 - \frac{56}{15}k_2 + \frac{32}{9}k_3, t^k + \frac{4}{5}h\right) \\
k_5 &= h\,\mathcal{F}\left(u^k + \frac{19372}{6561}k_1 - \frac{25360}{2187}k_2 + \frac{64448}{6561}k_3 - \frac{212}{729}k_4, t^k + \frac{8}{9}h\right) \\
k_6 &= h\,\mathcal{F}\left(u^k + \frac{9017}{3168}k_1 - \frac{355}{33}k_2 + \frac{46732}{5247}k_3 + \frac{49}{176}k_4 - \frac{5103}{18656}k_5, t^k + h\right) \\
k_7 &= h\,\mathcal{F}\left(u^k + \frac{35}{384}k_1 + \frac{500}{1113}k_3 + \frac{125}{192}k_4 - \frac{2187}{6784}k_5 + \frac{11}{84}k_6, t^k + h\right)
\end{aligned}
$$

$$
u_5^{k+1} = u^k + h\left(\frac{35}{384}k_1 + \frac{500}{1113}k_3 + \frac{125}{192}k_4 - \frac{2187}{6784}k_5 + \frac{11}{84}k_6\right)
$$

$$
u_4^{k+1} = u^k + h\left(\frac{5179}{57600}k_1 + \frac{7571}{16695}k_3 + \frac{393}{640}k_4 - \frac{92097}{339200}k_5 + \frac{187}{2100}k_6 + \frac{1}{40}k_7\right)
$$

u_5^{k+1} is 5th order, u_4^{k+1} is 4th order $\Rightarrow$ Automatic error estimation for adaptive time steps.

Embedded method

[J. Dormand, P. Prince. A family of embedded Runge-Kutta formulae. J. of Computation and Applied Mathematics, 1980].

Symplectic methods / semi-implicit

Introduction to symplectic methods

Standard approaches trade-off

- Explicit methods: (+) Simple to compute, (-) limited stability
- Implicit methods: (-) Hard to compute (especially on non linear functions), (+) very stable
- Oscillatory systems are not easy to model
  (-) Numerical solution either diverge or converge.

Symplectic approach

- *Remark*: Mechanical systems have position and speed variables
  - Derivative of position is linear w/r speed
  - Derivative of speed is more complex (forces - non linear)

$\Rightarrow$ *General idea*: separate treatment of speed and position

Semi-implicit:

- Implicit scheme for position p^{k+1} (linear part)
- Explicit scheme for speed v^{k+1} (non linear part)

$\Rightarrow$ In practice: use speed v^{k+1} to evaluate p^{k+1}.

Pro

(+) As simple as explicit method to implement
(+) Improved stability
(+) Well adapted to oscillatory systems

Semi implicit method

Simplest semi-implicit method: **Semi-implicit Euler** / **Verlet**

General case

$$v^{k+1} = v^k + h\,\mathcal{F}_v(p^k, v^k, t^k)$$
$$p^{k+1} = p^k + h\,\mathcal{F}_p(p^k, v^{k+1}, t^k)$$

Application to classical mechanical cases

$p'(t) = v(t), m\,v'(t) = F(p, v, t)$

$$\Rightarrow \begin{cases} v^{k+1} = v^k + h\,F(p^k, v^k, t^k)/m \\ p^{k+1} = p^k + h\,v^{k+1} \end{cases}$$

(+) Trivially easy to convert explicit Euler to semi-implicit Euler

Expressed using positions only

$$p^{k+1} = p^k + h(v^k + h\,F(p^k, v^k, t^k))$$
$$p^{k+1} = p^k + h\left(\frac{p^k - p^{k-1}}{h} + h\,F(p^k, v^k, t^k)/m\right)$$
$$\Rightarrow p^{k+1} = 2\,p^k - p^{k-1} + h^2\,F(p, v, t)/m$$

1st order accurate (like explicit/implicit Euler) in position and speed.

Stability on oscillatory system

1D spring system: $F(p^k, v^k, t^k) = -K\,p^k$

$p^{k+1} = 2\,p^k - p^{k-1} - h^2\,K/m\,p^k$

$\Rightarrow p^{k+1} = (2 - h^2\,K/m)\,p^k - p^{k-1}$

Stable and permanent oscillation when $h < \frac{2}{\sqrt{K/m}}$

Demonstration

- Study the roots of the associated caracteristic equation

- $x^2 + (y-2)x + 1 = 0$, with $y = h^2\,K/m$

- Discriminant $\Delta = y(y-4)$, with roots $r = \frac{(2-y) \pm \sqrt{y(y-4)}}{2}$

- Two cases:

  $\Delta < 0 \Rightarrow y \in]0,4[\ : r = \frac{(2-y) \pm i\sqrt{y(4-y)}}{2} \Rightarrow |r| = 1/2\,\sqrt{(2-y)^2 + y(4-y)} = 1 \Rightarrow$ Permanent oscillations (stable)

  $\Delta \geq 0 \Rightarrow y \leq 0,$ or $y \geq 4\ : |r| > 1$ Unstable solution

- Finally, stable for $y \in]0,4[\Rightarrow h^2 K/m < 4 \Rightarrow h < \frac{2}{\sqrt{K/m}}$, and system oscillate permanently

Higher order symplectic

2nd order Velocity Verlet

$$\begin{cases} v^{k+1/2} & = v^k + \frac{h}{2} F(p^k, v^k, t^k)/m \\ p^{k+1} & = p^k + h\, v^{k+1/2} \\ v^{k+1} & = v^{k+1/2} + \frac{h}{2} F(p^{k+1}, v^{k+1/2}, t^{k+1})/m \end{cases}$$

- Remains stable with permanent oscillation for $h \leq \frac{2}{\sqrt{K/m}}$

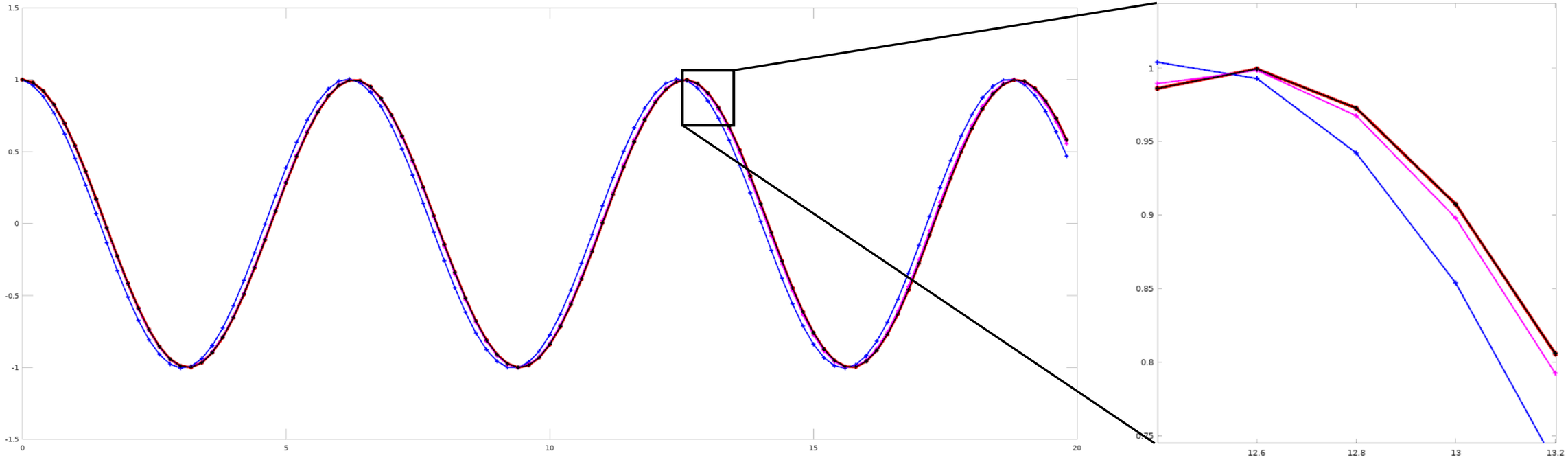
- Still simple to implement

Example for 1D elastic spring

```
for(int k=0; k<N; ++k) {
    v = v + h/2 * (-K/m*p);
    p = p + h*v;
    v = v + h/2 * (-K/m*p);
}
```

Comparison between approaches

Small $h = 0.2/\omega$

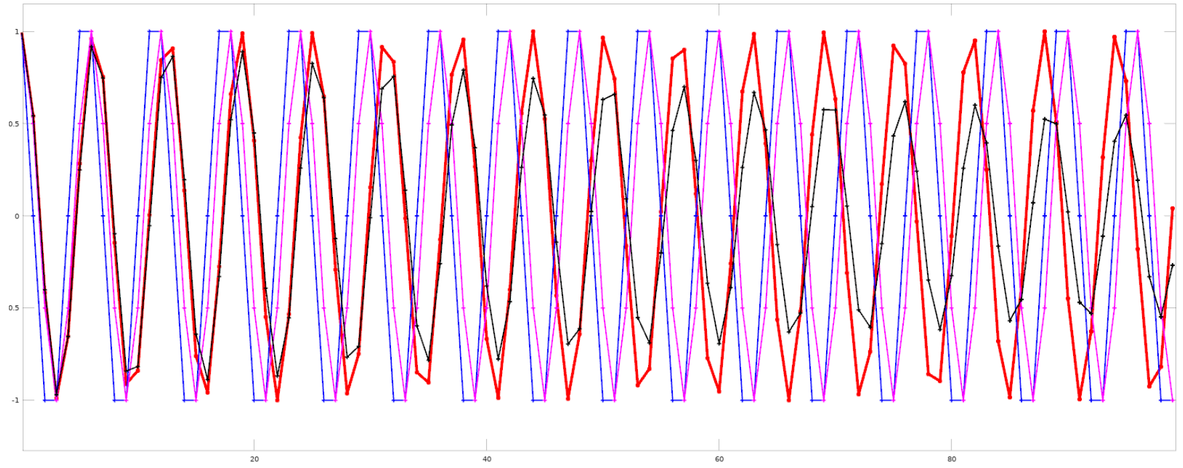


True solution , *Semi-implicit Euler* , *Velocity Verlet* , *Runge-Kutta RK4*

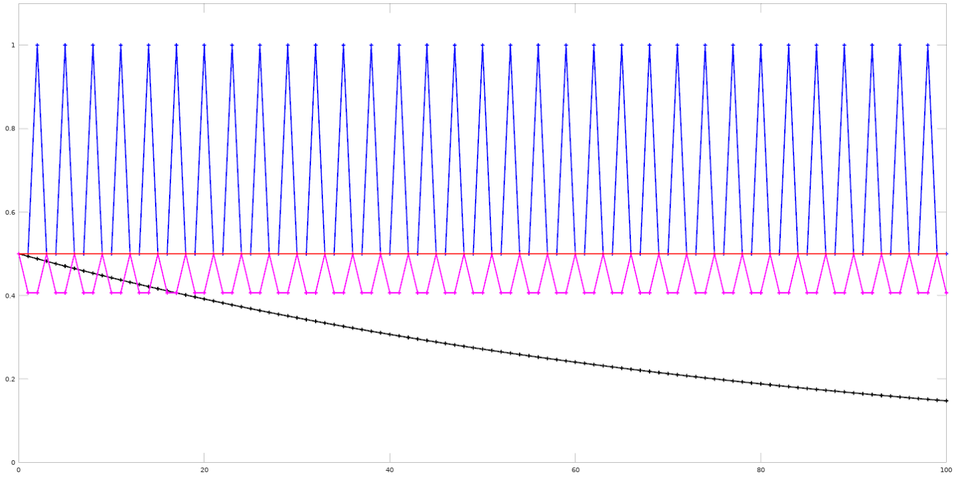
- RK4 - best behavior (undistinguishable from true solution)

Comparison between approaches

Larger $h = 1.0/\omega$



Temporal evolution of p^k



Temporal evolution of energy

$$E = 1/2m\,(v^k)^2 + 1/2K(p^k)^2$$

True solution , *Semi-implicit Euler* , *Velocity Verlet* , *Runge-Kutta RK4*

- RK4 loose energy and p^k converge toward l^0
- Symplectic integrator keep oscillating

Summary and extensions

Numerical integration of ODE

General formulation: $u'(t) = \mathcal{F}(u,t), u(t) = (p(t), v(t))$.

Explicit Euler

$u^{k+1} = u^k + \Delta t\, \mathcal{F}(u^k, t^k)$

(+) Easy to implement

(-) Worst scheme in all cases (divergence, low accuracy)

Explicit Runge-Kutta

$u^{k+1} = u^k + \Delta t \sum_j \alpha_j k_j$

(+) Good **accuracy**

(+) Efficient to apply

(+/-) Stability OK for non-stiff problem, diverge on stiff problem

(-) Artificial damping for constant energy system

Implicit methods

$u^{k+1} = u^k + \Delta t\, \mathcal{F}(u^{k+1}, t^{k+1})$

(+) Good to deal with **stiff problem** - very stable

(-) Add numerical damping (converge even if solution oscillates)

(-) Hard/computationally costly to apply on non linear problem

Symplectic integrator

$v^{k+1} = v^k + \Delta t\, F^k/m$

$p^{k+1} = p^k + \Delta t\, v^{k+1}$

(+) Handle well constant energy system, preserves energy (Hamiltonian systems)

(+) Simple and efficient to implement

(-) Less accurate than RK

(-) Diverge on stiff problem

Extension - Automatic time steping

Ajust h to limit errors at each step

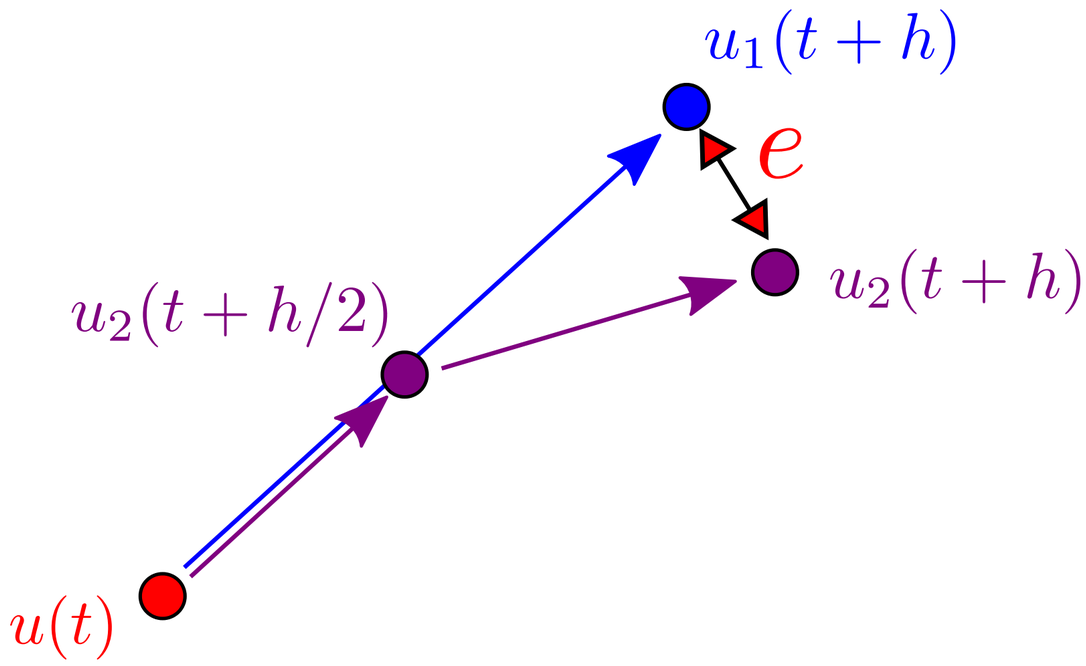
(1) Evaluate error in comparing:

- u_1^k computed using one step h
- u_2^k computed using two steps $h/2$.
- $e^k = \|u_1^k - u_2^k\|$

(2) Using embedded method $e^k = \|u_1^k - u_2^k\|$

Update

- $h^{new} = A\ h$ when $e^k < C_1$, $(A > 1)$
- $h^{new} = h/A$ when $e^k > C_2$



Implicit Euler on non-linear problem

$$u^{k+1} = u^k + h\,\mathcal{F}(u^{k+1}, t^{k+1})$$

- For high dimensional $\mathcal{F}$ - no explicit known inverse function

- Idea: Linearize $\mathcal{F}$ at u^k, assuming $\|u^{k+1} - u^k\|$ small.

$$u^{k+1} = u^k + h\left(\mathcal{F}(u^k, t^{k+1}) + (u^{k+1} - u^k)\frac{\partial \mathcal{F}}{\partial u}(u^k, t^{k+1})\right)$$

$$\Rightarrow \left(\mathrm{I} - h\,\tfrac{\partial \mathcal{F}}{\partial u}(u^k, t^{k+1})\right) u^{k+1} = \left(\mathrm{I} - h\,\tfrac{\partial \mathcal{F}}{\partial u}(u^k, t^{k+1})\right) u^k + h\,\mathcal{F}(u^k, t^{k+1})$$

$$u^{k+1} = u^k + \left(\mathrm{I} - h\,\frac{\partial \mathcal{F}}{\partial u}(u^k, t^{k+1})\right)^{-1} \mathcal{F}(u^k, t^{k+1})$$

- In theory only conditionally stable; In practice: very stable

- Requires the expression of Jacobian matrix $\frac{\partial \mathcal{F}}{\partial u}$ - large sparse matrix

- Requires inversion of a linear system at every steps

$\Rightarrow$ Used for stable simulation with large time step

Implicit Euler on non-linear problem - particle system

For standard particles systems

- $p^{k+1} = p^k + h\, v^k$
- $v^{k+1} = v^k + h\, \mathrm{M}^{-1}\, F(p^{k+1}, v^{k+1})$

With M the diagonal matrix of particles masses, assuming force F doesn't depends on t.

Linear approximation on force F

$$\mathrm{M}\, v^{k+1} = \mathrm{M}\, v^k + h\Big(F(p^k, v^k) + \underbrace{(p^{k+1} - p^k)}_{h\, v^{k+1}} \frac{\partial F}{\partial p}(p^k, v^k) + (v^{k+1} - v^k) \frac{\partial F}{\partial v}(p^k, v^k)\Big)$$

$$\Rightarrow \left(\mathrm{M} - h\frac{\partial F}{\partial v} - h^2 \frac{\partial F}{\partial p}\right)\, v^{k+1} = \left(\mathrm{M} - h\, \frac{\partial F}{\partial v}\right) + h\, F$$

[D. Baraff, A. Witkin. Large steps in Cloth Simulation. ACM SIGGRAPH 98]

Position based dynamics (PBD)

Non-linear constraints (non penetration, constant length, etc)

- can make explicit integration diverging (arbitrary gain of energy)
- are hard to integrate within implicit integrators

PBD - Use symplectic integrator expressed using position only

- Speed is computed implicitly as $(p^{k+1} - p^k)/h$
- Handle constraints using explicit projection of positions.

```
PBD algorithm
For all k
    Integrate position (without constraints) p[k+1]=integrator(p[k],p[k-1],t}
    For all constraints
        Project p[k+1] on constraints
    Compute new speed if needed v[k+1]=(p[k+1]-p[k])/h
```

(+) Unconditionnaly stable even for stiff constraints

(+) Simple to implement

(-) Low accuracy, no energy preserving

Popular in Computer Graphics

[M. Muller et al. Position Based Dynamics. VRIPHYS 2006]

Extensions: XPBD, Projective dynamics, ...

Physically-based simulation

Deformable models

Material model

Elasticity: Shape goes back toward its original rest position when external forces are removed.

-Purely elastic models don't loose energy when deformed (potential ↔ kinetic)

Plasticity: Oppositve of elasticity. Plastic material don't come back to their original shape (/change their rest position during deformation).

- Ductile material - can allow large amount of plastic deformation without breaking (plastic)
- Brittle - Opposite (glass, ceramics)

Viscosity: Resistance to flow (usually for fluid, ex. honey)

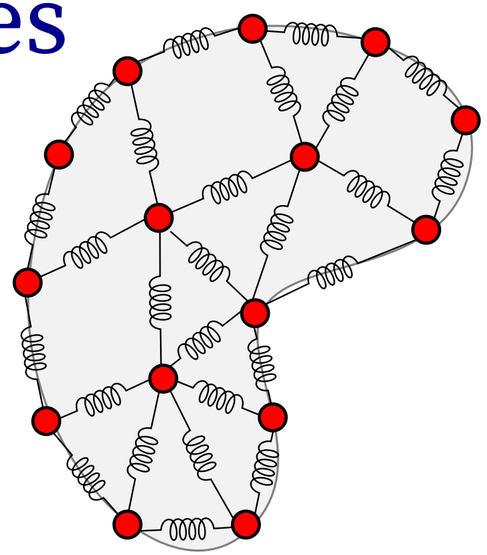
In reality

- *Elasto-plastic materials*: Allow elastic behavior for small deformation, and plastic at larger one.
- *Visco-elastic materials*: Elastic properties with delay.

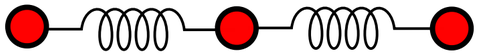
Modeling elastic shapes with particles

Spring mass systems

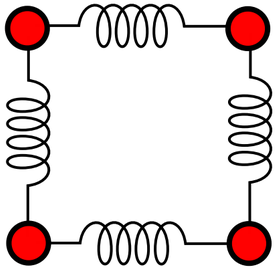
- Particles (position, speed, mass): samples on shape
- Springs : link closed-by particles in the reference shape



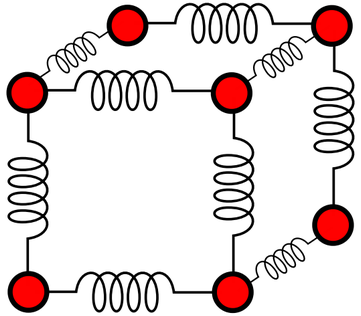
1D curve structure



2D surface structure



3D volume structure

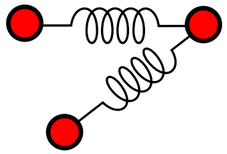


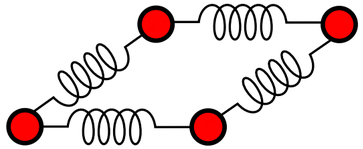
Spring structure

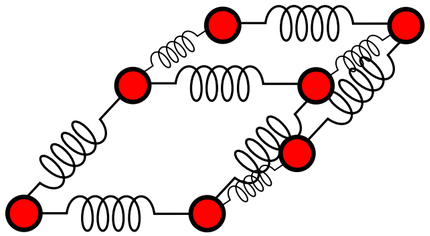
How to model spring connectivity ?

- **Structural springs:** 1-ring neighbors springs ($\simeq$ mesh edges)

  (+) Limit elongation/contraction, (-) Allows shearing, and bending

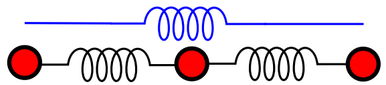


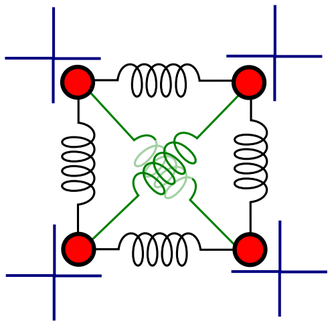


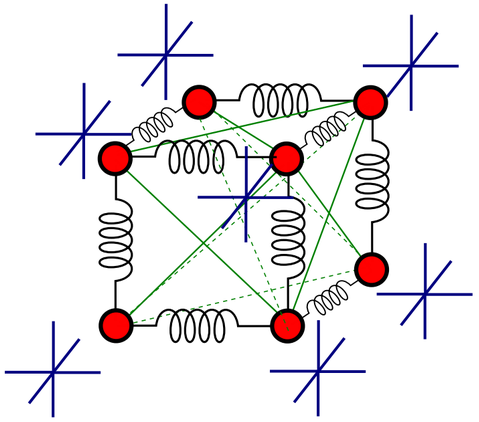


$\Rightarrow$ Add extra springs connectivity

- **Shearing springs**: Diagonal links
- **Bending springs**: 2-ring neighborhood



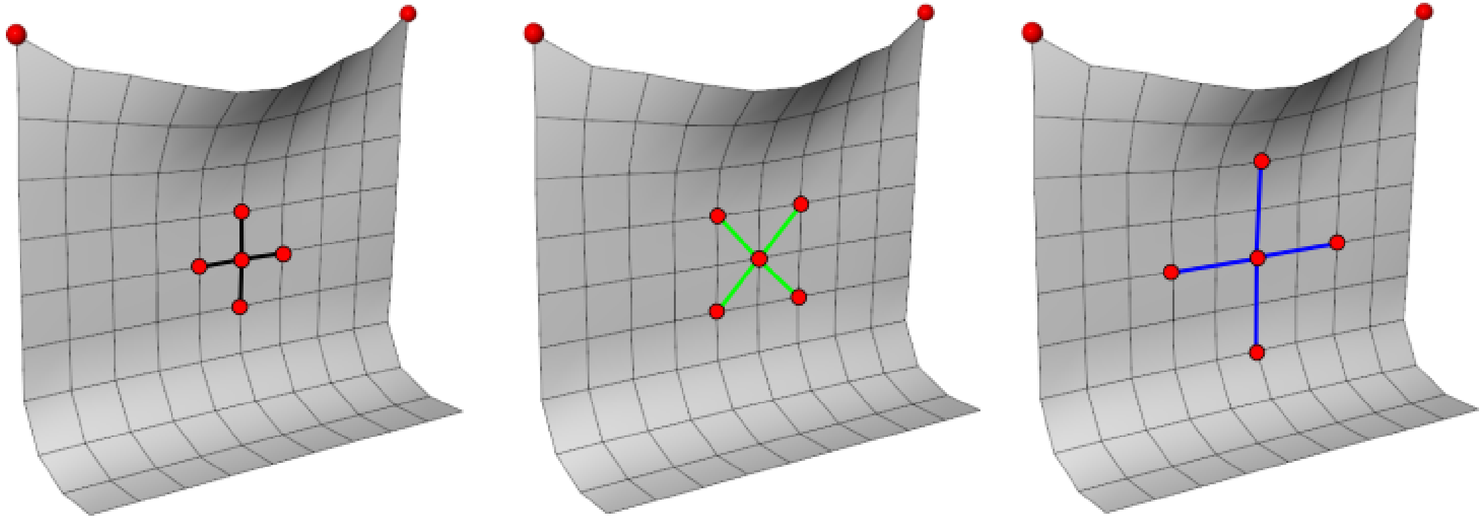




Cloth Simulation

Mass-spring cloth simulation

- Particles are sampled on a $N \times N$ grid.
  - Each particle has a mass m ($m_{cloth} = N^2\, m$)
- Set structural, shearing and bending springs.



Forces

- On each particle: gravity + drag + spring forces

- $F_i(p, v, t) = m_i\, g - \mu\, v_i(t) + \sum_{j \in \mathcal{V}_i} K_{ij} \left(\|p_j(t) - p_i(t)\| - L_{ij}^0 \right) \dfrac{p_j(t) - p_i(t)}{\|p_j(t) - p_i(t)\|}$

  - $\mathcal{V}_i$: neighborhood of particle i
  - L_{ij}^0: rest length of spring ij

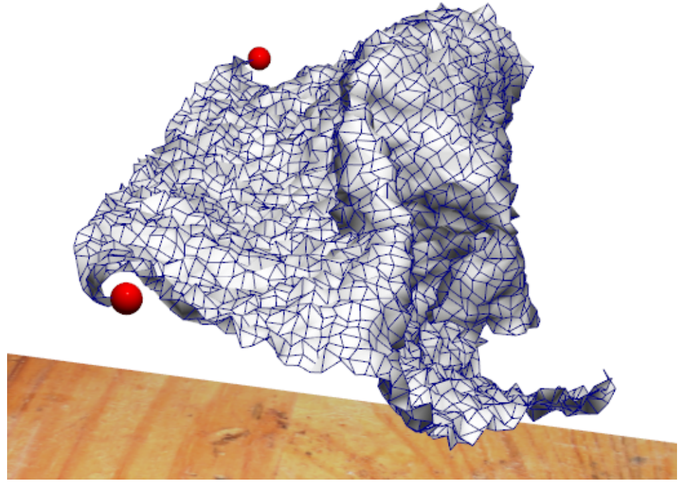
Associated ODE $\forall i, \begin{cases} p_i'(t) = v_i(t) \\ v_i'(t) = F_i(p, v, t)/m_i \end{cases}$

Note on Mass-Spring numerical solution

- Non-linear ODE

- Large K_{ij} : good length preservation, but stiff ODE
  $\Rightarrow$ divergence of explicit schemes.
- Avoid explicit Euler (divergence)

- Semi-implicit Euler/Verlet works fine for low K_{ij}
  Semi-implicit Euler + PBD allows simple integration + stable stiff springs
  [Muller et al. PBD, Inextensible clothing in Computer Games]



- RK4 more accurate (but higher complexity than Verlet)

- Implicit Euler : requires linearization, but very stable

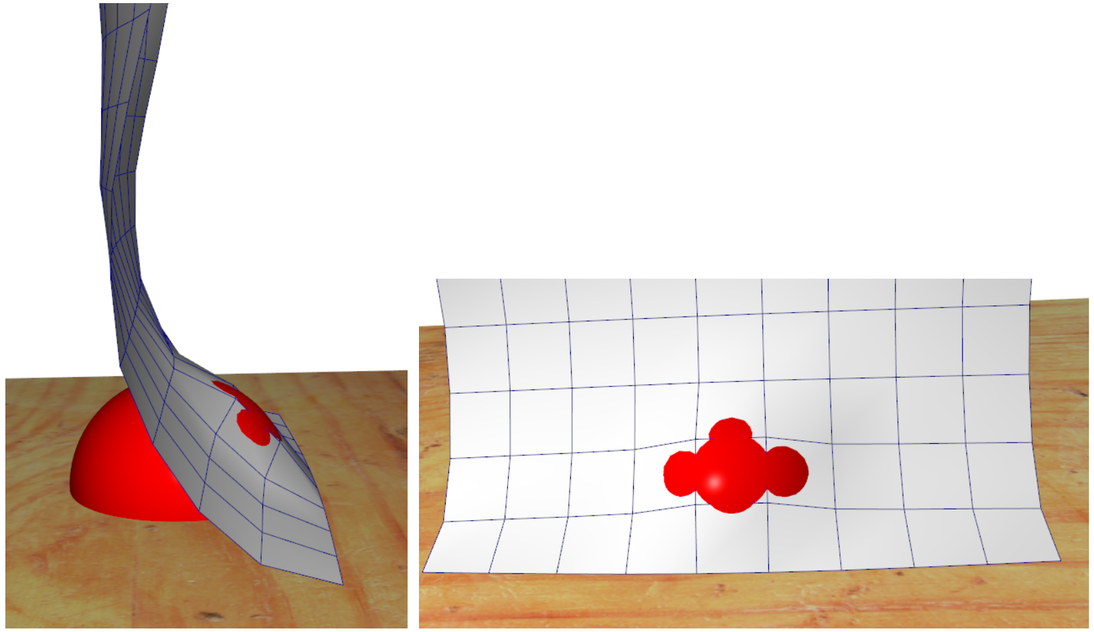
Collisions

- Simple approach : Handled as collision between particles and shapes

  (+) Simple and efficient

  (-) Collision may still appears within a triangle

  $\Rightarrow$ Exaustive approach: point-face + edge-edge



Limitation of mass spring model and continuous model

- Does mass-spring system converge toward a unique solution when sampling increase ?

  $\Rightarrow$ No :(

  Depends on the connectivity $\rightarrow$ bad for physical accuracy

Corollary

- Mass-springs work well for grid-mesh structure (draping)
- Less for arbitrary triangular meshes

1st improvment: Change toward energy formulation for bending springs (limits locking effect)

$$F = \frac{\partial E}{\partial p}$$

$$E = \frac{1}{2} K L \kappa^2, \kappa\text{: curvature}$$

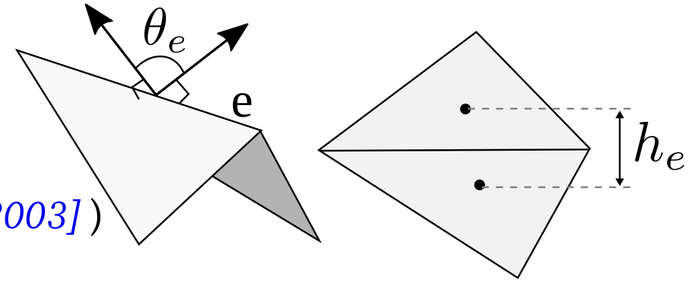
[Cho et al, Stable but Responsive Cloth, ACM SIGGRAPH 2002]

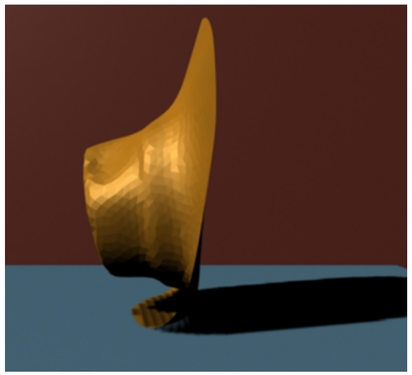
Triangle as continuous elements

- Defining Bending Energy between triangles
  - $W_B(x) = \sum_{edges\ e} (\theta_e - \theta_e^0) \frac{\|e^0\|}{h_e^0}$

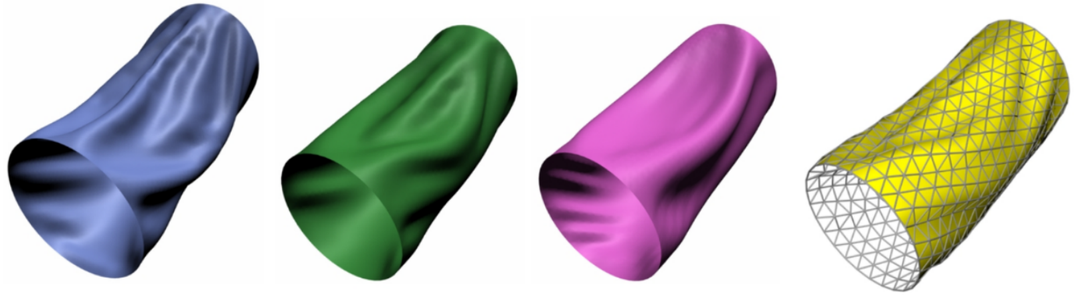
[E. Grinspun et al., Discrete Shells, SCA 2003]

(or expressed using forces in [R. Bridson et al., SCA 2003])



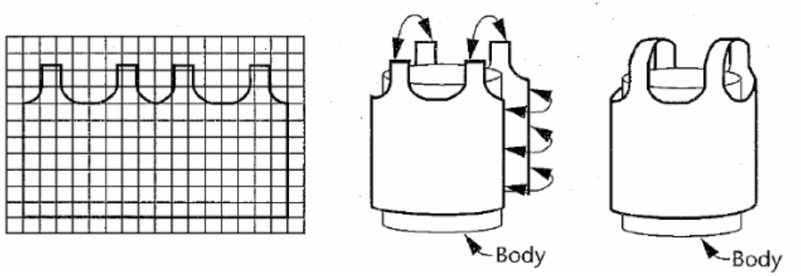


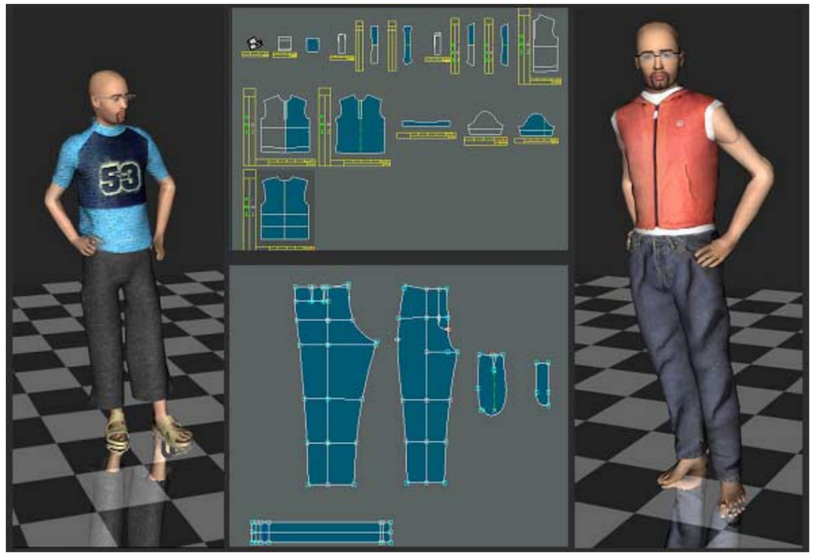
- Going toward full FEM numerical resolution
  - B. Thomaszewski et al. [SCA 2006], [VRIPHYS 2008], [EG 2009].



Clothing

- Stich 2D patterns together to generate full cloth
  - Cloths are developable material (preserve isometry to their 2D patterns)





[Thalman et al. 2002]

[Umetani et al., 2011]

Advanced collision

Self collision

- Handled as moving point in collision with moving triangle

- Given a triangle $P_1(t)P_2(t)P_3(t)$ and a point $P(t)$, with

$$P_k(t) = P_k(t_0) + v_{P_k}(t)$$

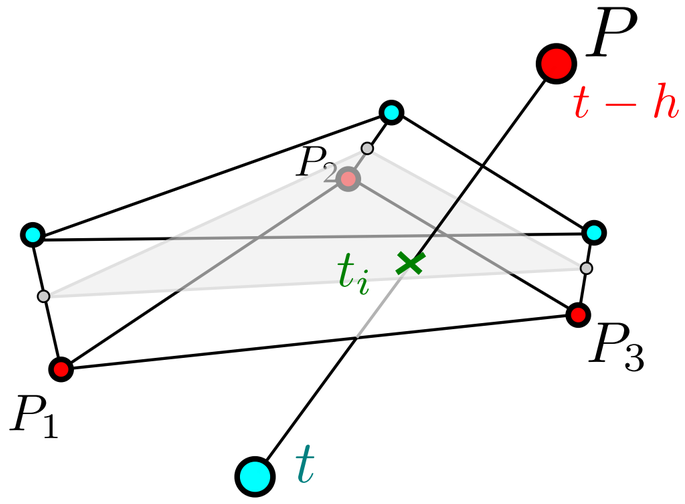
- $P(t_i)$ in collision with triangle $P_1(t_i)P_2(t_i)P_3(t_i)$ at time t_i if

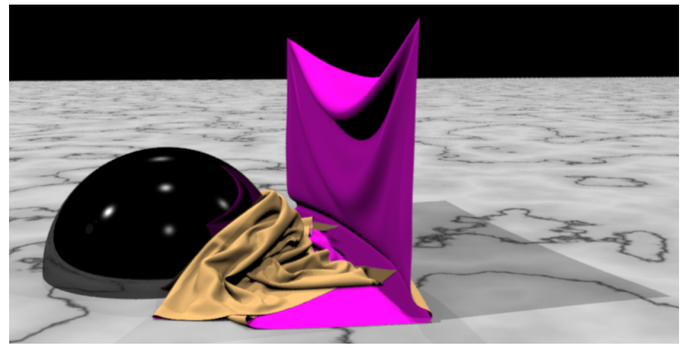
$$\exists(\alpha,\beta,\gamma) \in [0,1]^3,\ P(t_i) = \alpha\, P_1(t_i) + \beta\, P_2(t_i) + \gamma\, P_3(t_i)$$

$$\alpha + \beta + \gamma = 1$$

$$(P(t_i) - P_1(t_i)) \times n(t_i) = 0,\ n(t_i) \text{ normal to the triangle}$$

(necessary condition to help computation)





[X. Provot. Collision and self-collision handling in cloth model dedicated to design garments. Graphics Interface 1997.]
[R. Bridson et al. Robust Treatment of Collisions, Contact and Friction for Cloth Animation. ACM SIGGRAPH 2002]

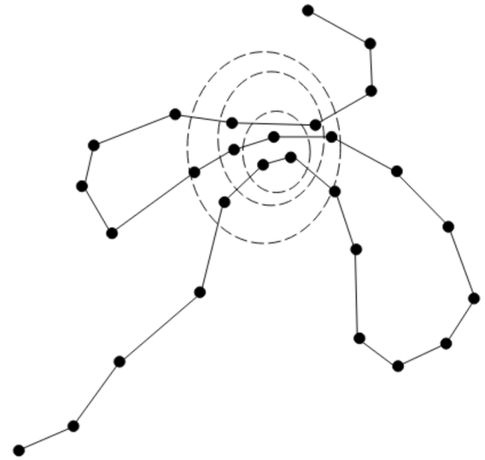
Advanced collision

Multiple collisions

- *Solving each collisions separately may lead to new ones without converging to collisions free state*

⇒ Rigid Impact Zone - Incremental approach

1. Detected collisions are grouped by proximity (same face, edge, etc)
2. Each group is handled as collision between rigid body
3. Treated groups are merged together.
4. Iterate at step 1 with larger groups



[Provot et al., 97]

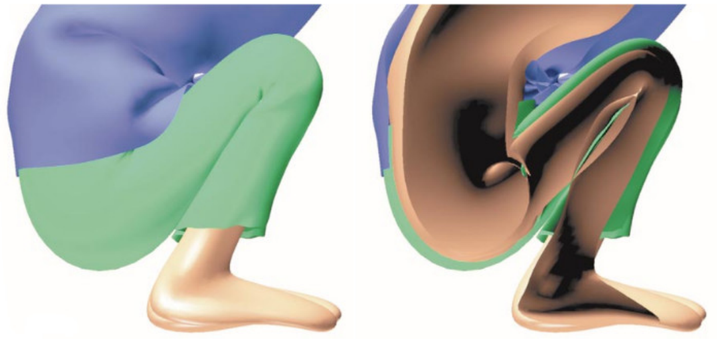
- Recently handled on the GPU

  [Tang et al., I-Cloth: Incremental Collision Handling for GPU-Based Interactive Cloth Simulation, ACM SIGGRAPH Asia 2018]

- Robust generic approach using tetrahedrization of air volume

  [Muller et al., Air Meshes for Robust Collision Handling, ACM SIGGRAPH 2015]

Untangling cloths



Specific problem of collision when no previous state is known.

How to solve collision ?

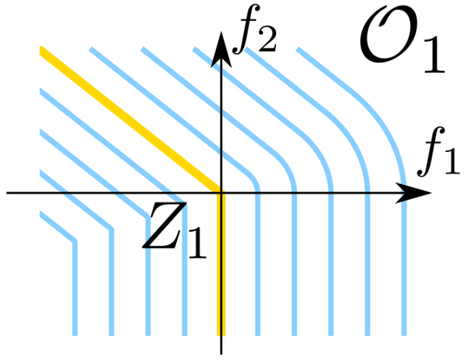
- Geometrical displacement + penalty based forces

  ex. Optimize intersection contour minimization

  [Baraf et al. 2002], [Volino et al 2006]

- Implicit surfaces

  [Buffet et al. 2019]



Wrinkles in cloth modeling

Obtaining cloths wrinkles requires high resolution simulations

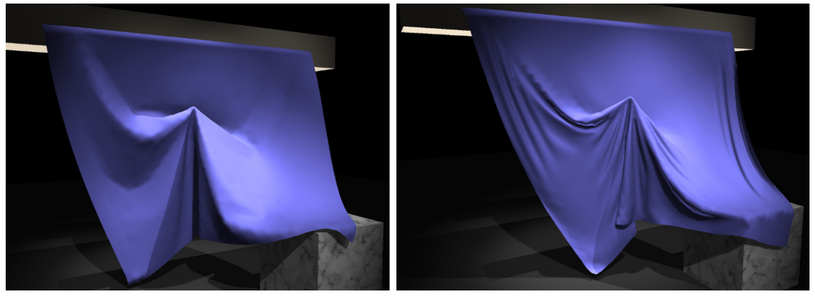
(-) Computationally costly, difficult to parameterized, lack of control

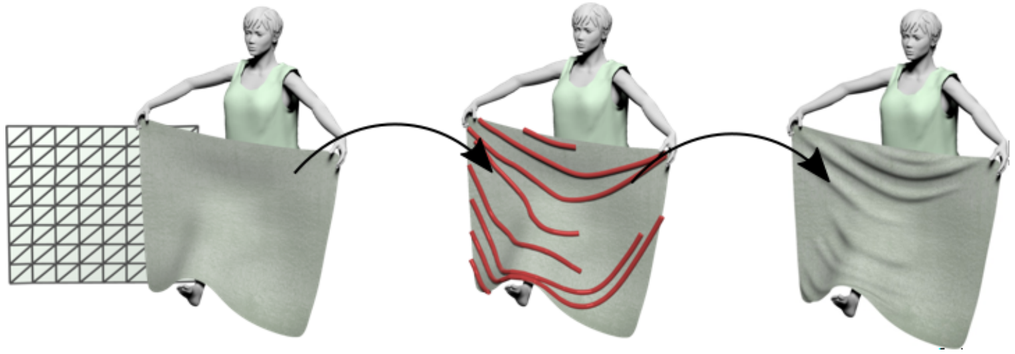
Idea: Separate low resolution cloth simulation from local wrinkles appearance

- Cloth is a developable material (cannot compress/stretch w/r 2D pattern)
- Detect compression on numerical surface $\rightarrow$ add wrinkles

Model wrinkles using

- Texture deformation [Hadap et al., IEEE Proc. on Viz. 1999]
- Subdivision mesh with PBD length constraints [Muller and Chentanez, SCA 2010]
- Example data and ML [Wang et al. ACM SIGGRAPH 2010]
- Strain to generate geometrical wrinkles [Rohmer et al., ACM SIGGRAPH Asia 2011]

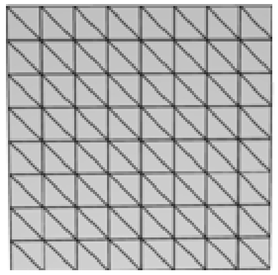


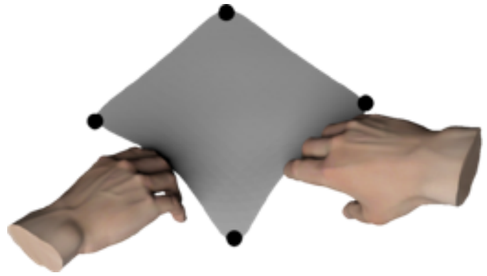


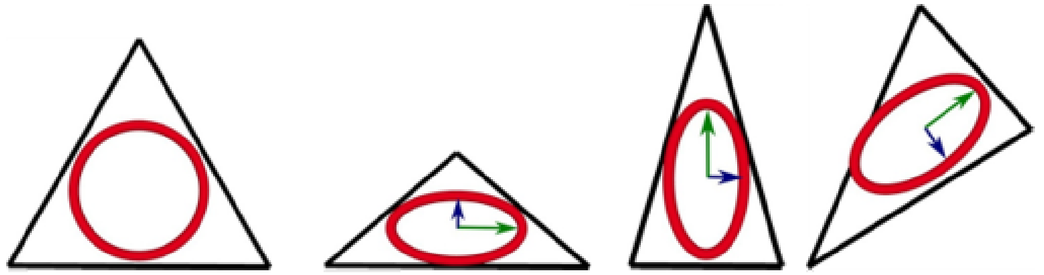
Geometrical wrinkles from strain

Compute strain on low resolution mesh

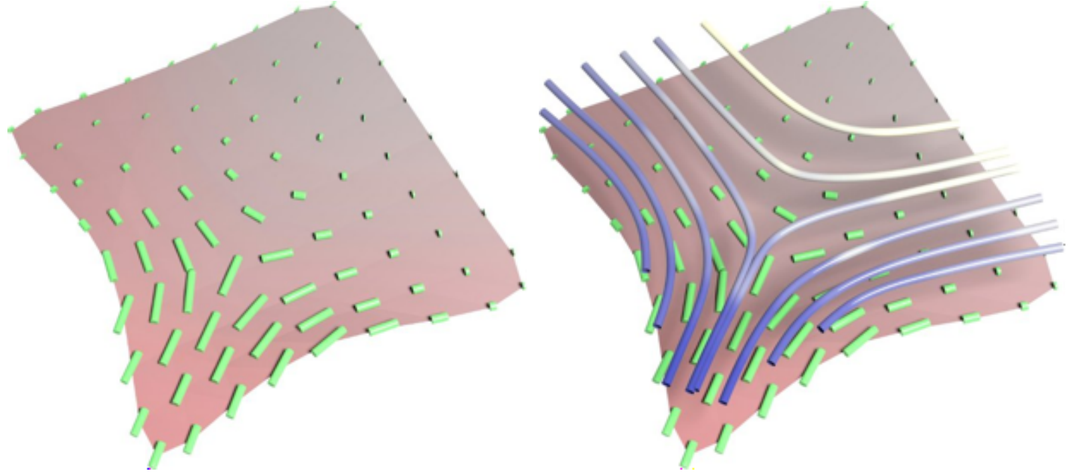
$$\mathbf{F} = [e_1' - e_1 \;\; e_2' - e_2], \quad \sigma = \mathbf{F}\,\mathbf{F}^T, \quad (\lambda_1, \lambda_2) = \text{Spectrum}(\sigma)$$







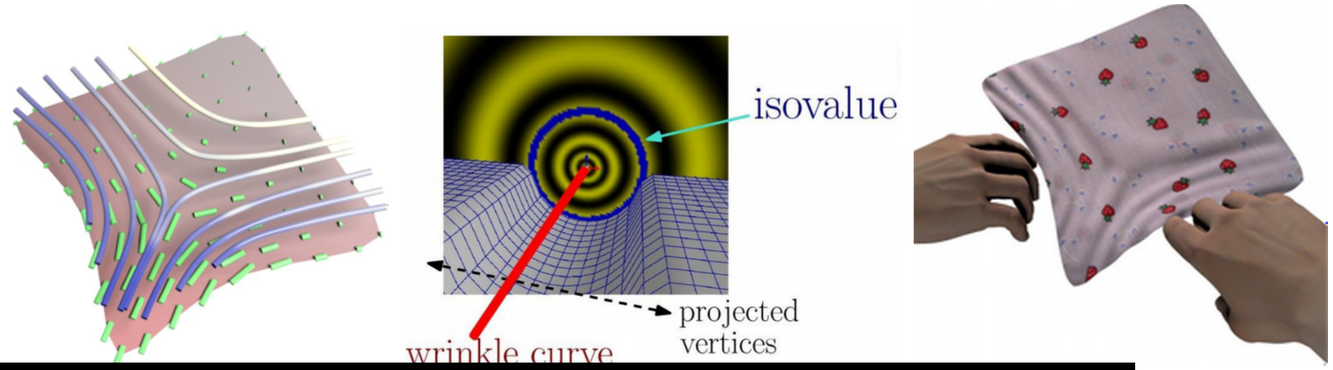
- Analyse main direction of compression $(0 < \min(\lambda_1, \lambda_2) < 1)$ as a vector field
- Generate stream lines $\rightarrow$ *wrinkle curves*

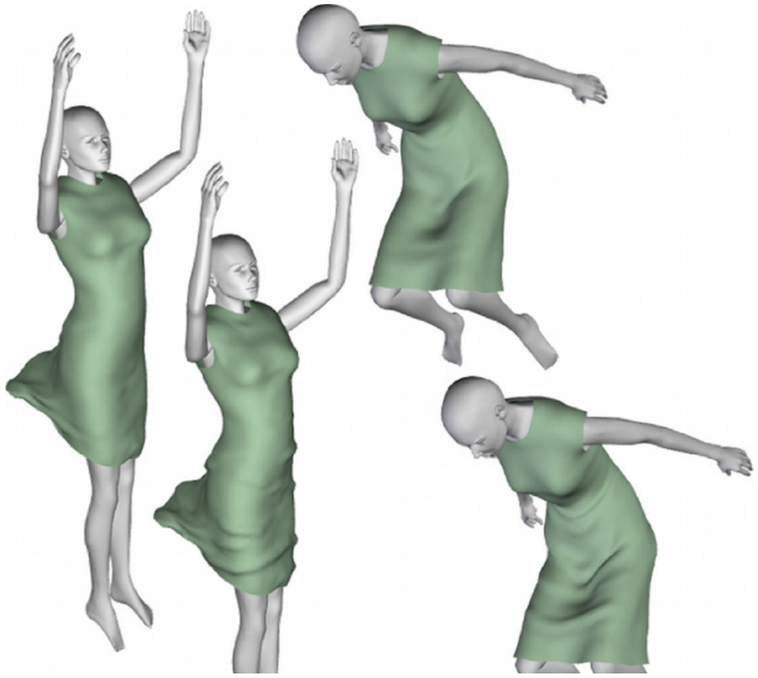


Geometrical wrinkles from strain

Create geometrical deformation using implicit surfaces

Allow smooth blending between wrinkles





[Animation Wrinkling: Augmenting Coarse Cloth Simulations with Realistic-Looking Wrinkles, ACM SIGGRAPH Asia 2011]

Animating fluids

Fluid in grid: Stable Fluids

Consider a rectangular grid filled with fluid

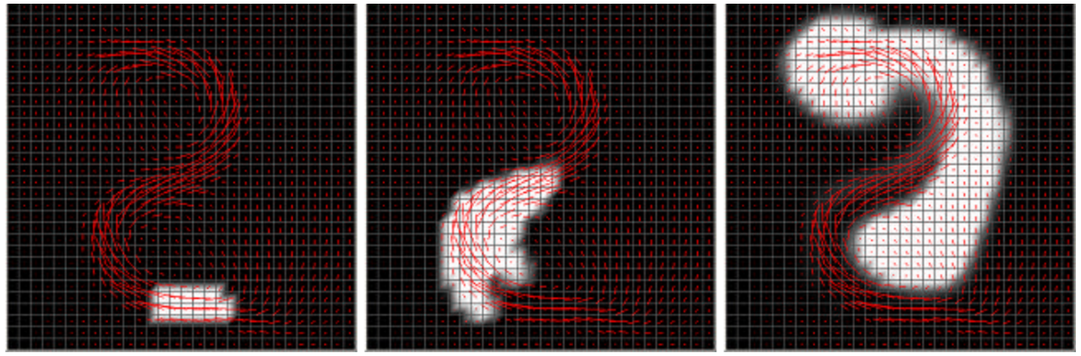
Direct approach: Solve Navier-Stokes equation using finite differences on the grid.

$$\frac{\partial u}{\partial t} = -\frac{1}{\rho}\nabla \mathrm{p} + f - u \cdot \nabla u + \nu \triangle u$$

(-) Stability hard to achieve, loose advection details on the grid. (ex. [Foster and Metaxas, 97])

Well known improvement: **Jos Stam, Stable Fluids, ACM SIGGRAPH 1999**

- First idea: get rid of explicit pression expression
  - Pression p models incompressible behavior.
  $\rightarrow$ Project velocity u into divergence free field at each step instead of tracking pression values.



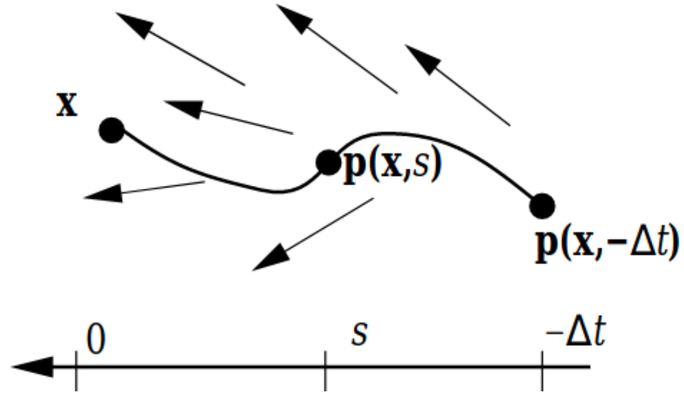
$$\Rightarrow \frac{\partial u}{\partial t} = \mathrm{P}(f - u \cdot \nabla u + \nu \triangle u)$$

P projection operator such that $\mathrm{div}(u) = 0$

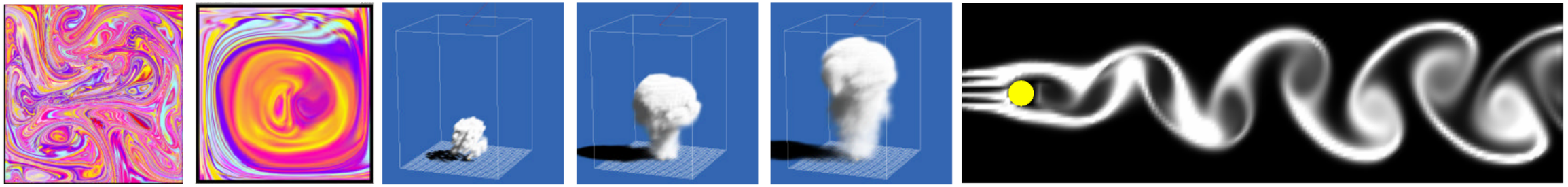
Stable fluids method

General approach: split each component

$$u^k \underbrace{\rightarrow}_{\text{add forces}} u_1^k \underbrace{\rightarrow}_{\text{advect}} u_2^k \underbrace{\rightarrow}_{\text{diffuse}} u_3^k \underbrace{\rightarrow}_{\text{project}} u^{k+1}$$



- Advection: Method of characteristics (use explicit position, trace back their trajectory in time)
- Diffusion: Use implicit finite differences
- Project: Poisson solver



Jos Stam, Real-Timer Fluid Dynamics for Games, 2003

Stable fluids example

?

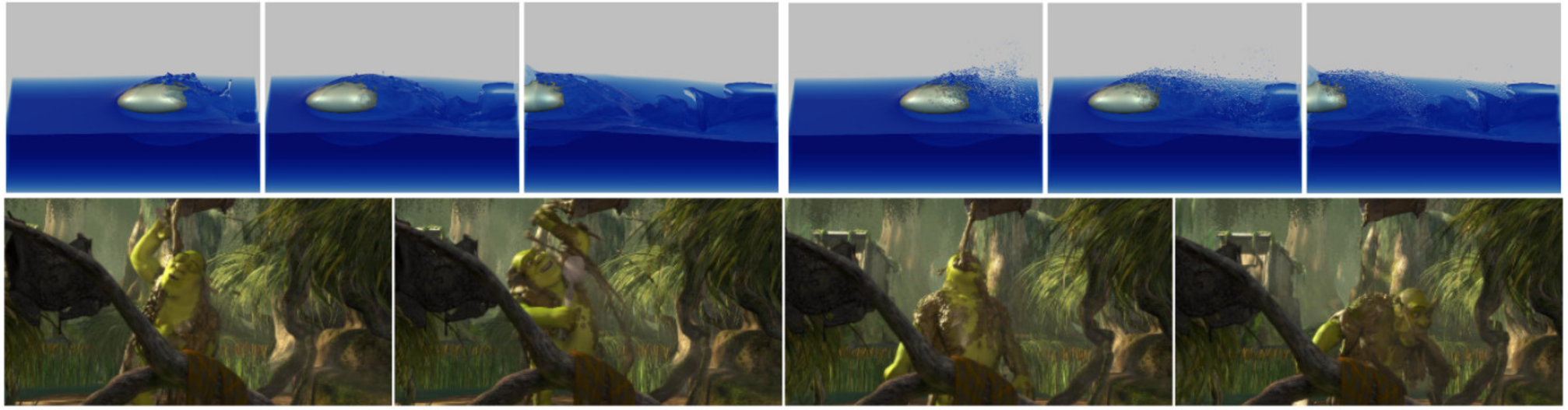
Level Set methods introduction

Eulerian models (ex. stable fluids) do not handle natively free boundaries such as fluid/air.

Idea: Track surface boundary using implicit surfaces

- Encode fluid volume by $\Omega = \{p \in \mathbb{R}^3, \varphi(p) = 0\}$, φ stored within a 3D grid.
- Solve Navier-Stokes equation within Ω
- Deform fluid volume using implicit surface deformation $\frac{\partial \varphi}{\partial t} + u \cdot \nabla \varphi = 0$ (advection)

Introduced by Ronald Fedkiw, Nick Foster, Stanley Osher (ex. [Practical Animation of Liquids, ACM SIGGRAPH 2001])

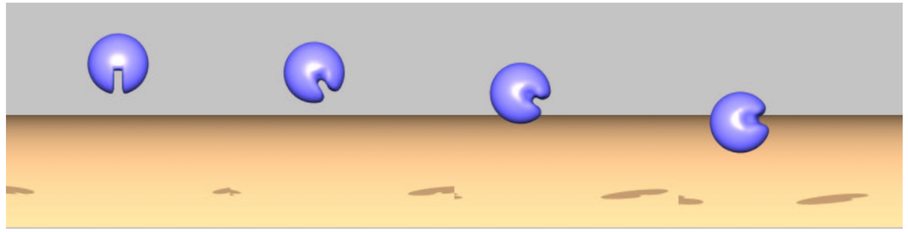


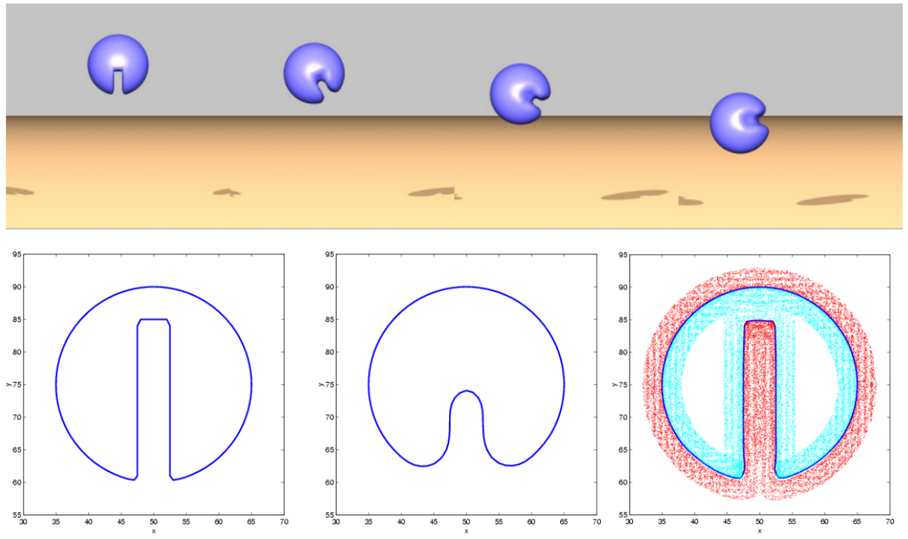
Particles Level Set

Limitation of level sets: smoothing and loss of volume from grid interpolation

$\Rightarrow$ Use of **Particle Level Set Method**

- [D. Enright et al. Hybrid Particle Level Set Method for Improved Interface Capturing. J. Comp. Physics 2001]
- [D. Enright et al. Animation and Rendering of Complex Water Surfaces, ACM SIGGRAPH 2002]



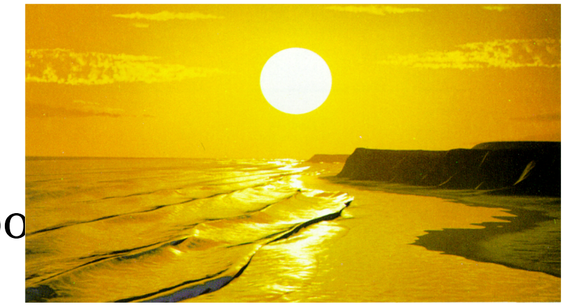


0:00 / 0:21

0:00 / 0:13

D. Enright et al. *(8 min / frames)*

Free surface water



Specific lightweight models for free surface of ocean-like water

Often based on **Shallow water equation** (neglect depth velocity compo

[A. Fournier, W. T. Reeves. A simple Model of Ocean Waves, ACM SIGGRAPH 1986]

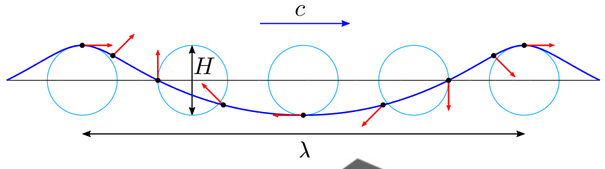
[D. Hinsiger et al. Interactive Animation of Ocean Waves. SCA 2002]

[Jerry Tessendorf, Simulating Ocean Water, ACM SIGGRAPH Course Notes 2004]

ex. *Trochoid models*

Particles following circular trajectories, waves propagates

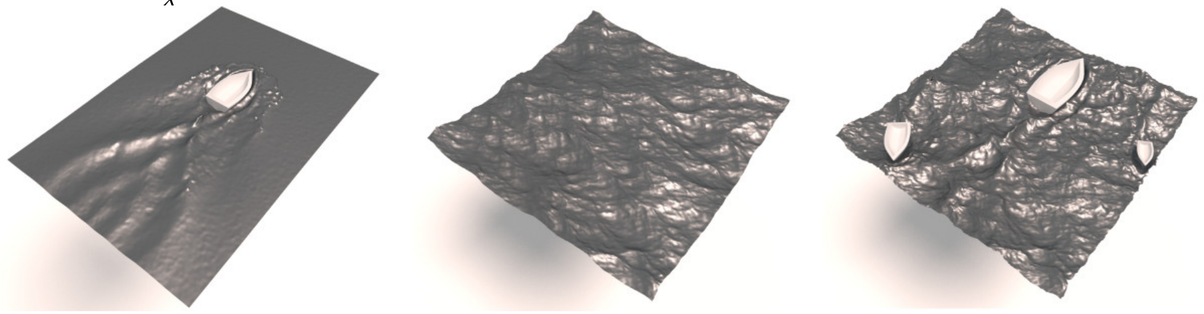
$$x(u,v,t) = u + e^{k\,v}\ \sin(k(u+c\,t))/k$$
$$x(u,v,t) = v - e^{k\,v}\ \cos(k(u+c\,t))/k$$
$$k = 2\pi\,\lambda$$



Used for procedural ocean modeling

(+) Simple and scalable

(-) Interaction with other objects



[Manteaux et al. Space-time sculpting of liquid animation, MIG 2016]

Free surface water - Example

Link Three.js

Animating fluids

SPH

General SPH idea

Pure Lagrangian: particles can follow surface deformation precisely

But how to evaluate values (density, speed, etc) between particles ?

Pure Eulerian: Values are known at fixed point in space

But how to follow deformation of surface boundaries ?

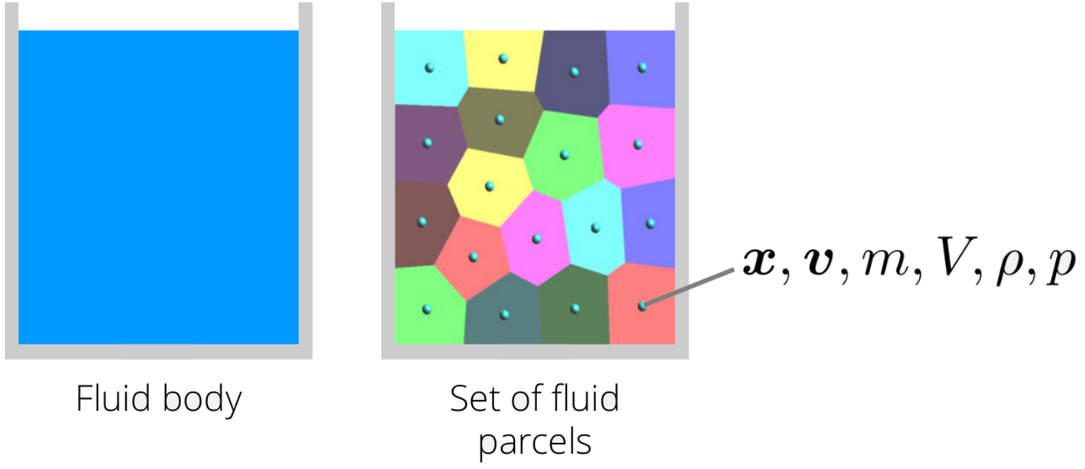
$\Rightarrow$ SPH Idea:

- Reconstruct surface from arbitrary samples in space using convolution
- Allow samples to be advected with the fluid

(+) Particle based model: can interact with other shapes

(+) Handle naturaly arbitrary boundaries

(+) Scalable



Initial used in Astronomy

[L. Lucy, A numerical approach to the testing of the fission hypothesis. The Astronomical Journal, 1977.]

Sampling and density

Arbitrary quantity A at position p can be expressed as a convolution

$$A(p) = (A \star \delta)(p) = \int_\Omega A(q)\,\delta(p-q)\,\mathrm{d}q$$

First approximation: $\delta \to W_h$

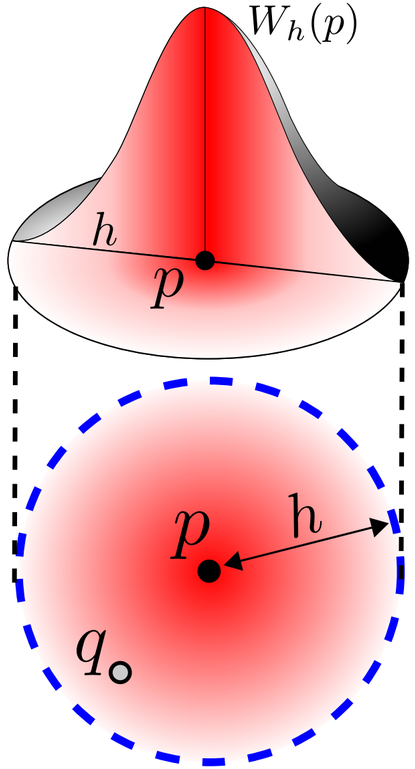
$$A(p) \simeq \int_\Omega A(q)\,W_h(p-q)\,\mathrm{d}q$$

W_h smooth kernel with limited support (/test functions)

W_h converge to δ when $h \to 0$

$$\Rightarrow \forall h, \int_\Omega W_h(p)\mathrm{d}p = 1$$

Classical *low pass* reconstruction (/average) for functions around p



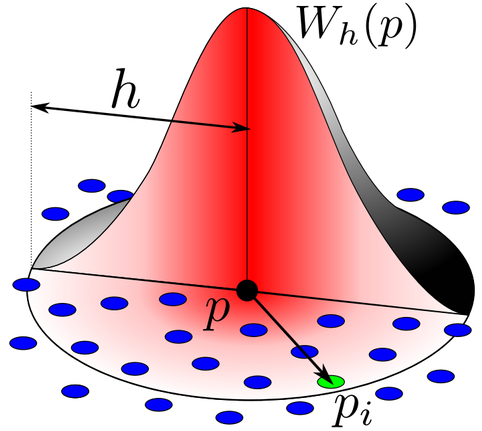
Model

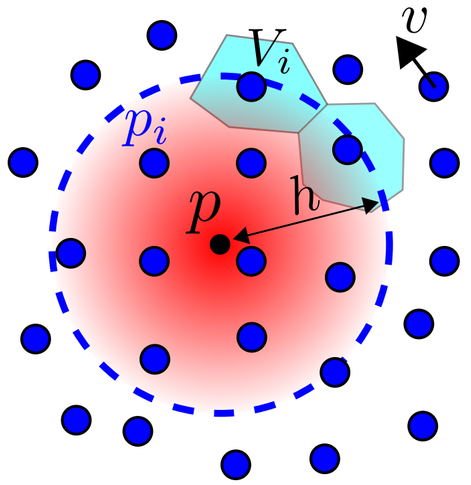
Second approximation : discrete sampling of A at position p_i.

$$A(p) = \int_\Omega A(q)\, W_h(p-q)\, \mathrm{d}q$$

$$A(p) = \sum_i A(p_i)\, W_h(p-p_i)\, V_i$$

V_i is the small volume (/area in 2D) associated to the ith samples.

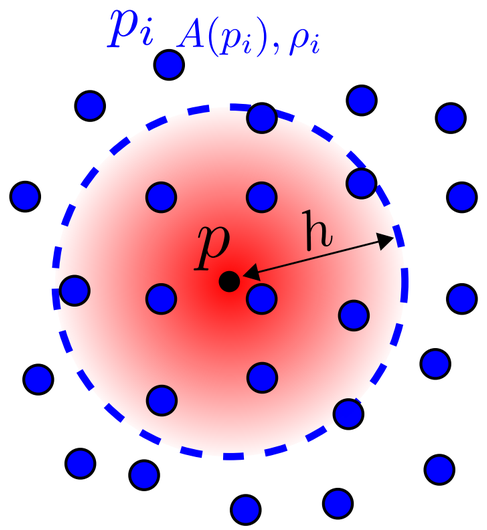




SPH Model for fluid

For a fluid: $m_i = \rho_i \, V(p_i)$

- m_i total mass associated to the volume at sample i
- ρ_i local density (supposed constant over the volume)



- General SPH formulation for a quantity A at position p

$$A(p) = \sum_i \frac{m_i}{\rho_i} A(p_i) \, W_h(p - p_i)$$

- Special case of the density

$$\rho(p) = \sum_i m_i \, W_h(p - p_i)$$

SPH Relations

- Local value at position p

$$A(p) = \sum_i \frac{m_i}{\rho_i} A(p_i)\, W_h(p - p_i)$$

- Gradient (symmetric) at position p

$$\nabla A(p) = \rho(p) \sum_i m_i \left(\frac{A(p)}{\rho^2} + \frac{A(p_i)}{\rho_i^2} \right) \nabla W_h(p - p_i)$$

- Laplacien (symmetric) at position p

$$\triangle A(p) = 2 \sum_i \frac{m_i}{\rho_i} \frac{(p - p_i) \cdot (A(p) - A(p_i))}{\|p - p_i\|^2 + \epsilon\, h^2} \nabla W_h(p - p_i), \epsilon = 10^{-2}$$

Note: derivatives are symetrized such that $\nabla A(p_i)/p_j = \nabla A(p_j)/p_i$

See for instance [Desbrun and Cani, EGCAS 96]

SPH for Navier Stokes

Continuous case

$$\frac{\mathrm{d}v}{\mathrm{d}t} = g - \frac{\nabla \mathrm{p}}{\rho} + \nu \triangle v$$

Using discrete SPH formulation at position p_i

$$v_i'(t) = g - \sum_j m_j \left(\frac{\mathrm{p}_i}{\rho_i^2} + \frac{\mathrm{p}_j}{\rho_j^2} \right) \nabla W_h(p_i - p_j) + 2\nu \sum_j \frac{m_j}{\rho_j} \frac{(p_i - p_j) \cdot (v_i - v_j)}{\|p_i - p_j\|^2 + \epsilon h^2} \nabla W_h(p_i - p_j)$$

- Pression given by the Tait equation of state

  $\mathrm{p}(p) = K \left(\frac{\rho(p)}{\rho_0} - 1 \right)^{\alpha}, \rho > \rho_0$

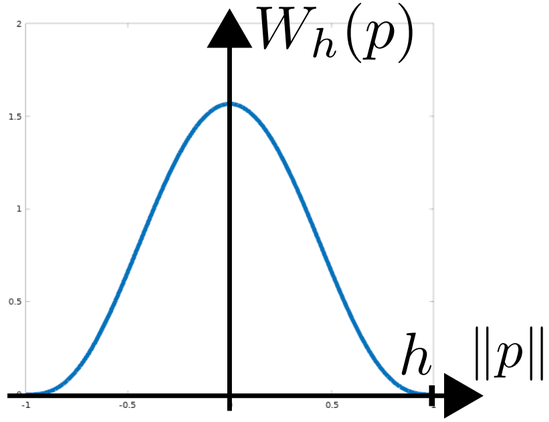
  ρ_0: rest density 1000 Kg/m^3, $\alpha \simeq 7$ for water, K stiffness constant.

- Density expressed using SPH evaluation $\rho(p_i) = \sum_j m_j W_h(p_i - p_j)$

SPH Kernel

- Smooth kernel

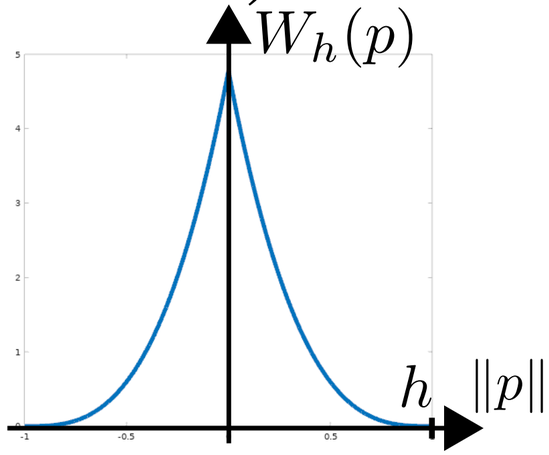
$$W_h(p) = \frac{315}{64\,\pi\,h^3}\left(1 - \left(\frac{\|p\|}{h}\right)^2\right)^3, \quad \|p\| \leq h$$



- Spiky kernel

Can give better results when used to compute pression force (avoid particles cluster)

$$W_h(p) = \frac{15}{\pi\,h^3}\left(1 - \frac{\|p\|}{h}\right)^3, \quad \|p\| \leq h$$

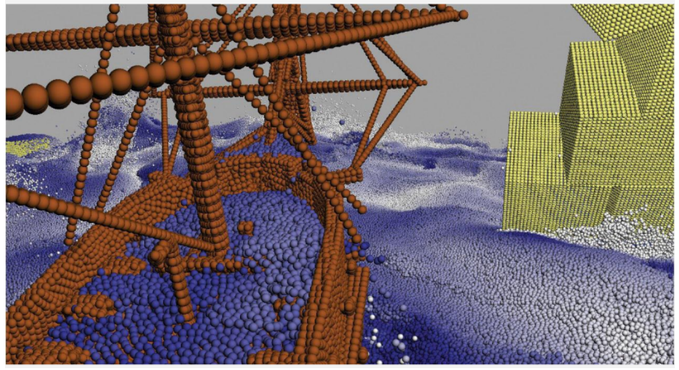


- [Desbrun and Cani, Smoothed Particles: A new paradigm for animating highly deformable bodies, EGCAS 1996]
- [M. Muller et al., Particle-Based Fluid Simulation for Interactive Applications, SCA 2003]
- [M. Ihmsen et al., SPH Fluids in Computer Graphics, EG STAR 2014]

SPH examples

Muller 2003

M. Teschner 2012 - 20M particles



SPH extensions

(+) Very versatile (interaction between any deforming shapes)

Not only fluids

(-) Not well understood accuracy

(-) Compressible

[Solenthaler et al., Predictive-Corrective Incompressible SPH, ACM SIGGRAPH 2009]
[Ihmsen et al, Implicit Incompressible SPH, IEEE TVCG 2013]

(-) Limited time step

[Macklin and Muller, Position based Fluids, ACM SIGGRAPH 2013]

(-) Boundaries are hard to handle

[Brand et al., Pressure Boundaries for Implicit Incompressible SPH, ACM TOG 2018]

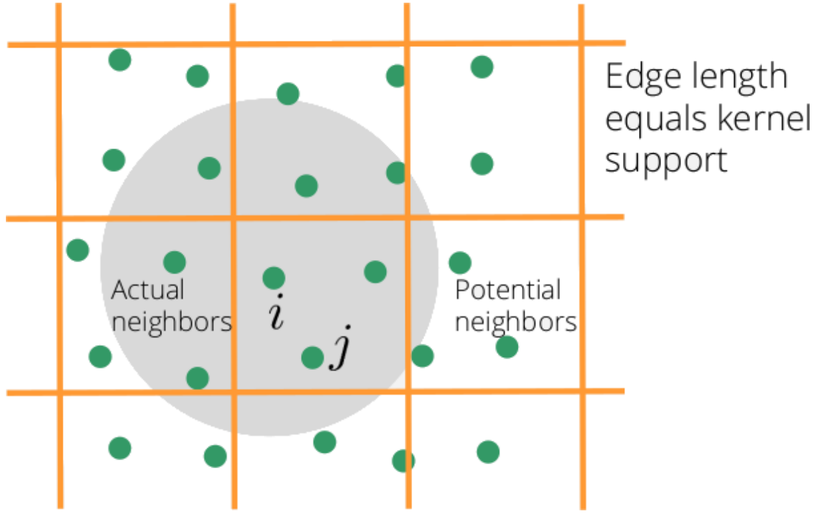
Bruno Levy

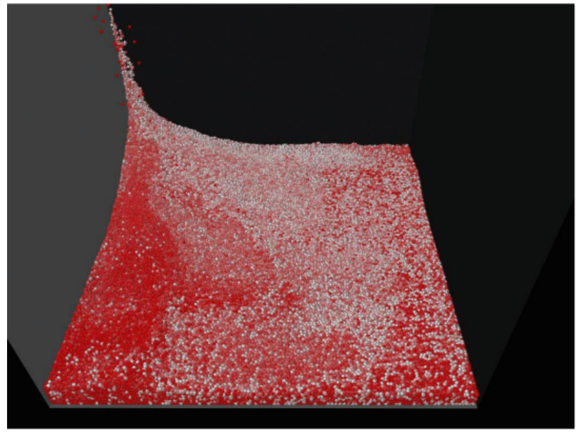
Macklin and Muller

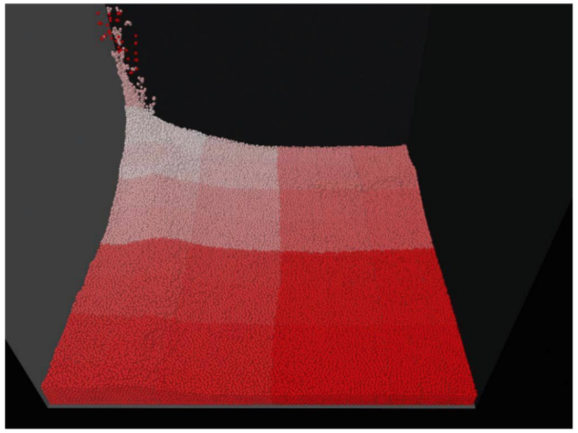
Acceleration structure

SPH based on pair-wise interation $\Rightarrow$ spatial sorting acceleration structure

- Uniform grid: simple and efficient.
- Verlet lists (wider neighborhood, updated every n steps only)
- List of vertices per cell, hash table for cell storage
- Spatial sorting for cache efficiency







M. Teschner

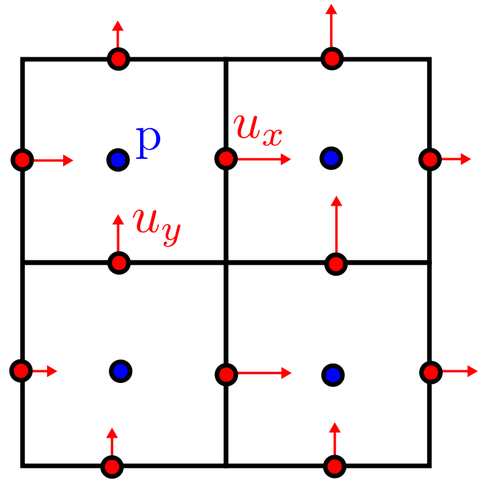
PIC/FLIP

Mix between particles and grid based approach.

Particles: advection, Grid: forces, pression, viscosity

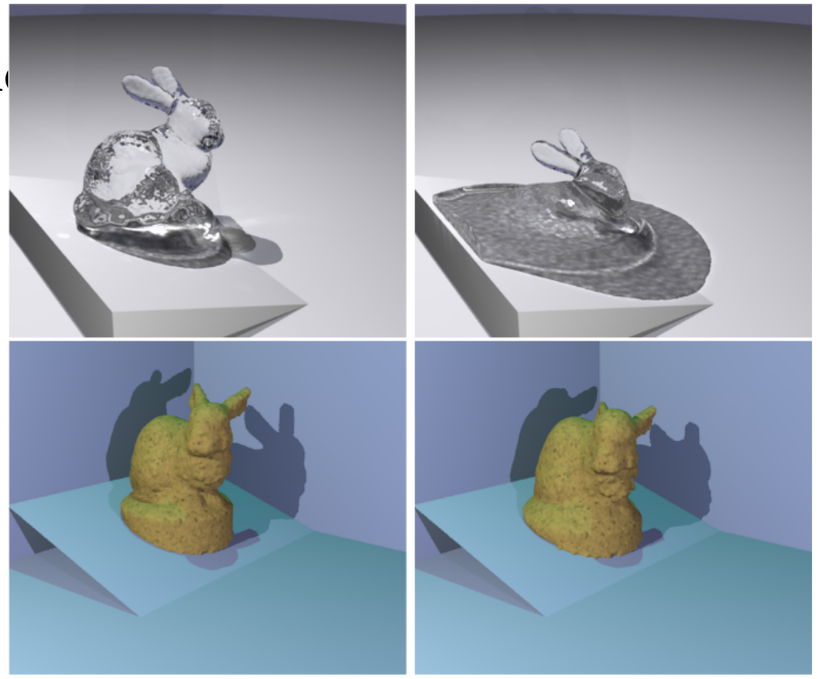
Use MAC grid (Marker-and-Cell)

Grid storing velocity on middle-edges, pressure on grid center.



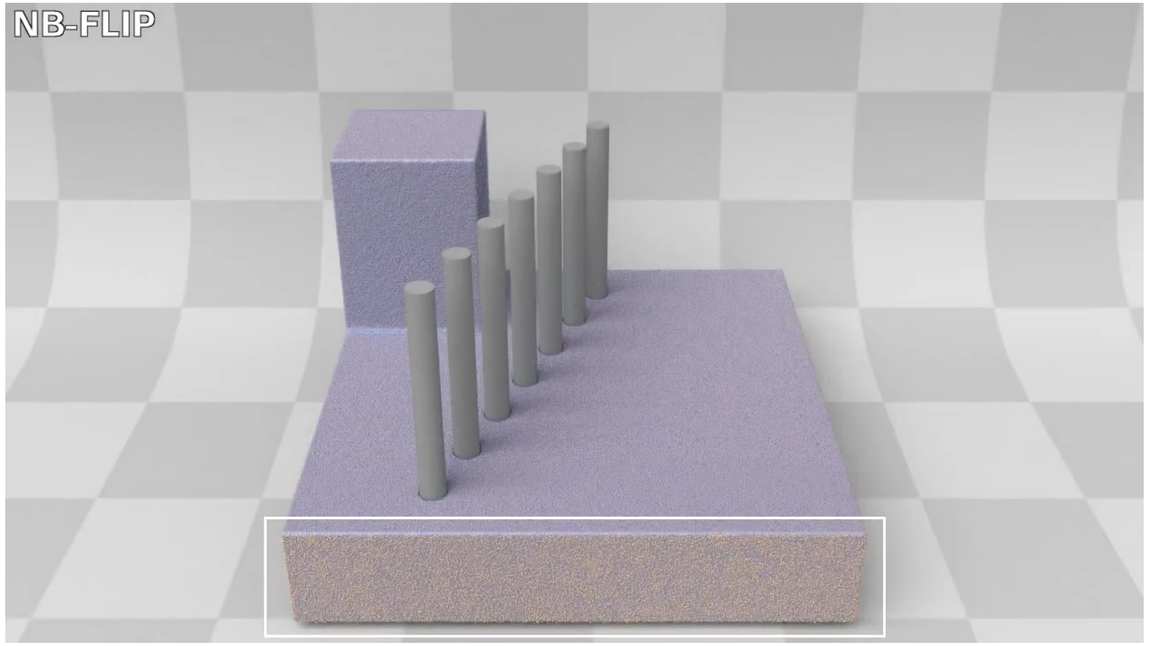
- **PIC** approach - Transfert velocity from grid to particles
- **FLIP** approach - Add velocity difference from grid to particl
- **PIC/FLIP** : blending b/w two approaches

[Y. Zhu and R. Bridson, Animating Sand as a Fluid, ACM SIGGRAPH 2005]



PIC/FLIP Method

- Transfert particle velocity to the MAC grid (Store velocity u^k on grid)
- Evolve velocity on grid (pression, forces, viscosity) excepted advection to u^{k+1}
- Add velocity difference $\Delta u = u^{k+1} - u^k$ to particles using interpolation (FLIP approach)
- Blend particle velocity with interpolated grid velocity (PIC/FLIP)
- Advect particles along grid velocity field (solve ODE)



[F. Ferstl et al., EUROGRAPHICS 2016]