Animation basée sur la physique

Basée sur Physically Based Modeling cours Siggraph 2001 par Andrew Witkin et David Baraff
Merci à Marie-Paule Cani, François Faure, Nicolas Holzschuch et Estelle Duveau.

Objectif

Reproduire le comportement d'**objets physique** soumis à des **forces** dans un environnements

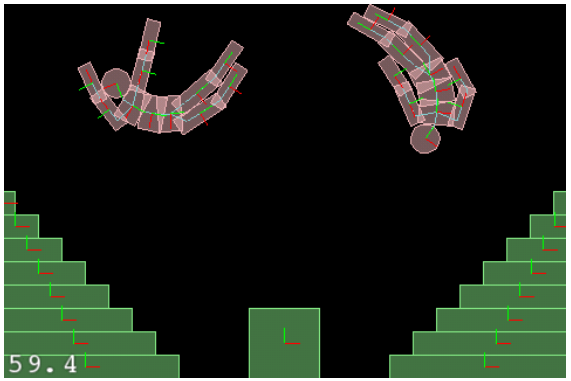
Générer **automatiquement** des animations plausibles

- rehausser le réalisme dans les applications interactives
- diversifier les animations
- animer rapidement et de façon réaliste des scènes qui seraient normalement trop complexes à animer manuellement (explosions, écroulements, fluides…)

Exemples

Objets rigides et chaînes articulées

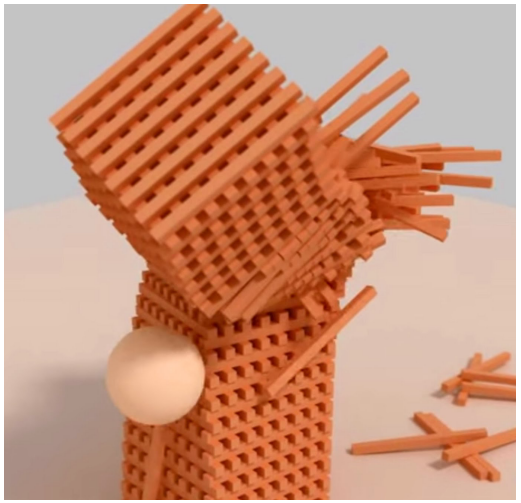
Box2D



Cocos2D,
Box2D Ragdoll example

2K Games, *Mafia II*

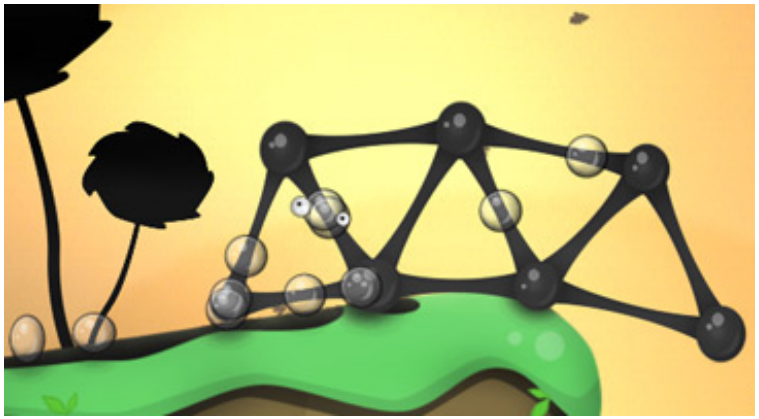
BULLET PHYSICS LIBRARY



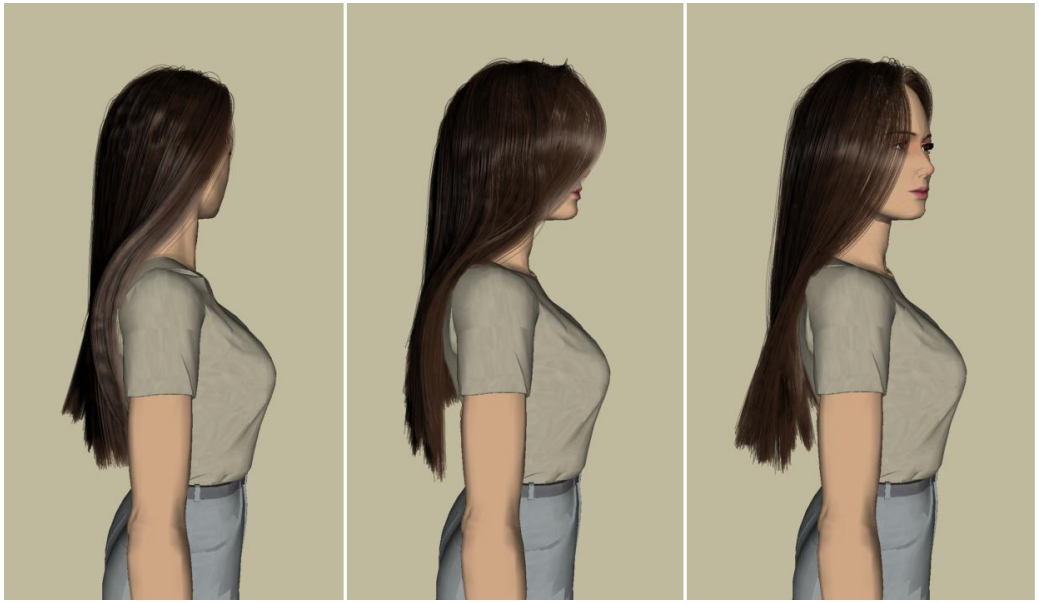
Moby Motion,
Blender - Bullet Physics

Exemples

Objets déformables



2D Boys, *World of Goo*

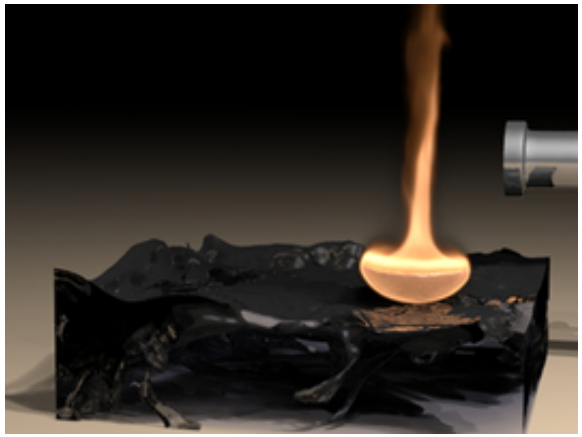


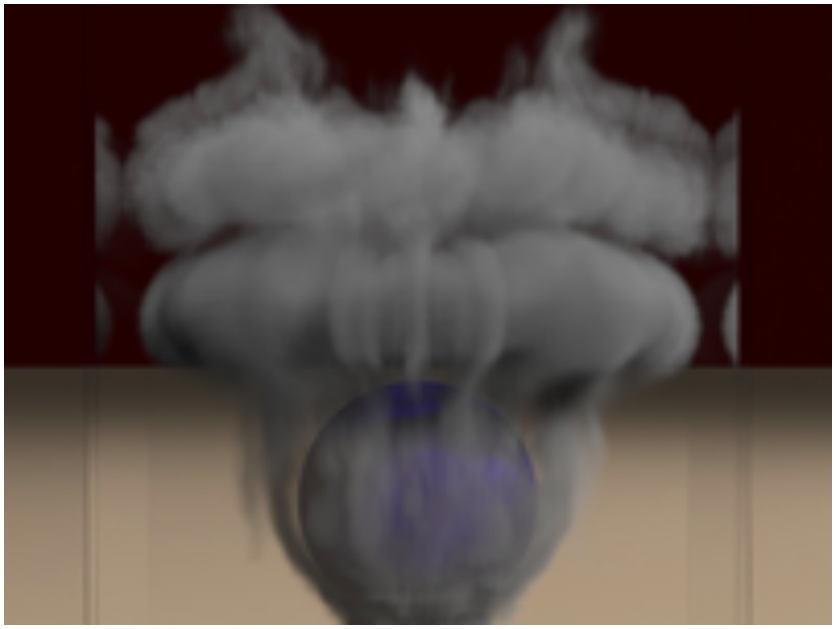
[Bertails et al., 2006]

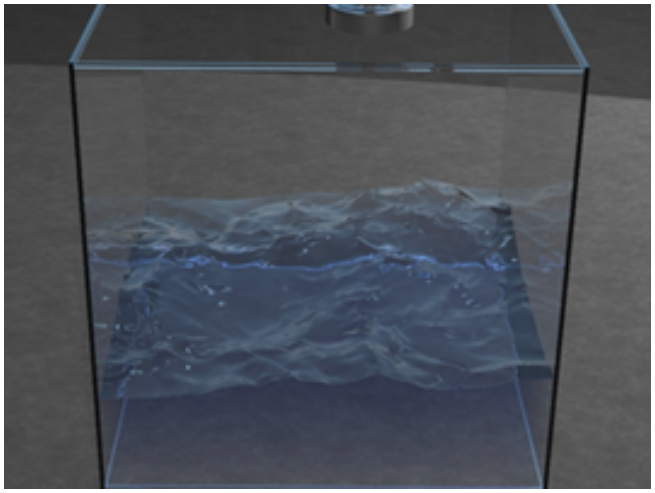
[Baraff et al., 2003]

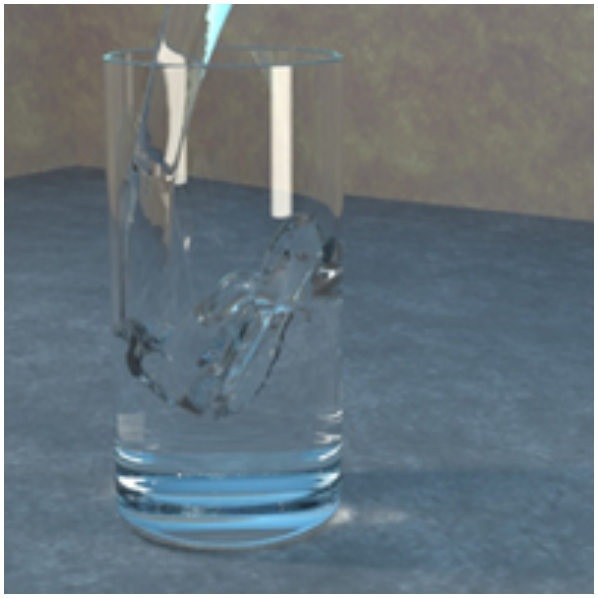
Exemples

Fluides









http://physbam.stanford.edu

Simulation physique

Animation basée sur la physique

$\Longrightarrow$ **simulation physique**

Challenges

- stabilité numérique
- contrôle artistique

Lois du mouvement de Newton

1. En l'absence de toute force, un corps matériel, s'il est au repos, reste au repos ; s'il est en mouvement, conserve un mouvement rectiligne et uniforme.
2. L'application d'une force $\mathbf{f}$ à un corps de masse m se traduit par la variation de sa vitesse (accélération).
3. Entre deux corps, il ne peut y avoir d'action que mutuelle. Il y a toujours lieu ainsi d'apparier deux forces : de même direction, égales en valeur absolues, et de sens opposés

$\Longrightarrow$ on en extrait plusieurs formules et concepts décrivant la **physique mécanique**

Physique d'une particule

Particule = corps infinitésimal définie par sa :

- **position** au cours du temps $\mathbf{x}(t)$
- **masse** m

On note sa **vitesse** $\mathbf{v} = \dot{\mathbf{x}} = \frac{d\mathbf{x}}{dt}$

et son **accélération** $\mathbf{a} = \ddot{\mathbf{x}} = \frac{d\mathbf{v}}{dt}$

Si n forces $\mathbf{f}_i$ sont appliquées à la particule,

alors d'après la **2nde loi de Newton** :

$$\sum_{i=1}^{n} \mathbf{f}_i = \mathbf{F} = m\mathbf{a}$$

$$\Leftrightarrow \ddot{\mathbf{x}} = \frac{\mathbf{F}}{m}$$

Physique d'une particule

Les forces peuvent **dépendre** de la position $\mathbf{x}$ et de la vitesse $\dot{\mathbf{x}}$ de la particule ainsi que du temps t :

$$\ddot{\mathbf{x}}(t) = \frac{\mathbf{F}(t, \mathbf{x}(t), \dot{\mathbf{x}}(t))}{m}$$

$\Longrightarrow$ **équation différentielle ordinaire (ODE)**
(car $\mathbf{x}$ ne dépend que de t)

Mais $\mathbf{F}$ arbitraire $\Longrightarrow$ équation non-linéaire

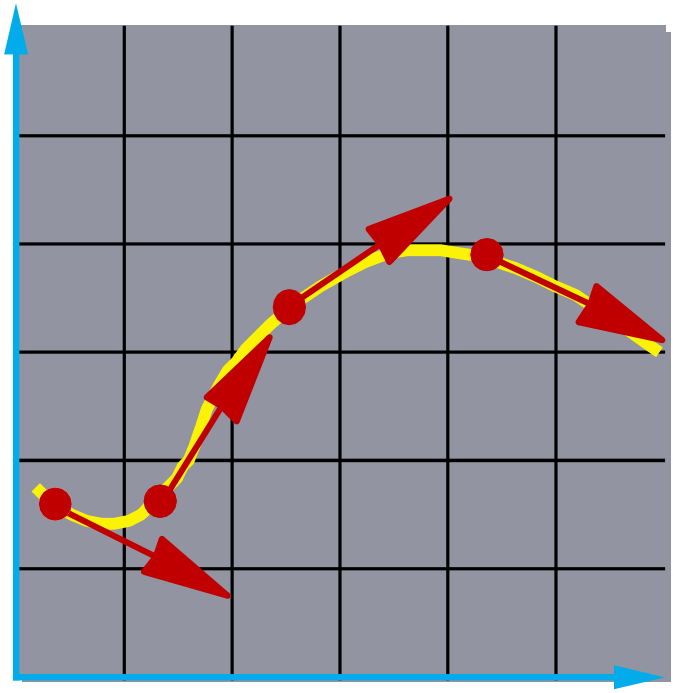
$\Longrightarrow$ pas de solution analytique

$\Longrightarrow$ **intégration numérique**

Intégration numérique

On part d'une valeur initiale $\mathbf{x}_0 = \mathbf{x}(t_0)$

On avance d'un pas de temps Δt dans la direction de la vitesse $\dot{\mathbf{x}}(t_0)$



Intégration numérique

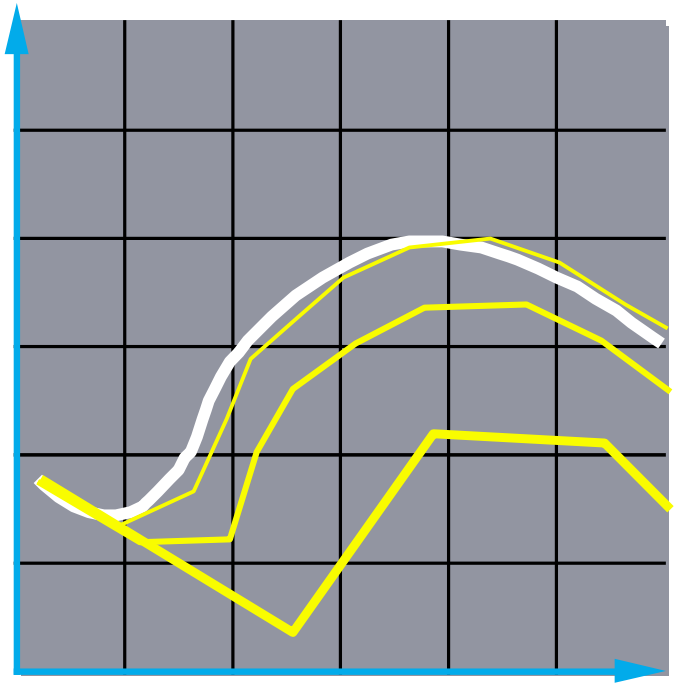
On part d'une valeur initiale $\mathbf{x}_0 = \mathbf{x}(t_0)$

On avance d'un pas de temps Δt dans la direction de la vitesse $\dot{\mathbf{x}}(t_0)$:

$$\mathbf{x}(t_0 + \Delta t) = \mathbf{x}_0 + \Delta t\, \dot{\mathbf{x}}(t_0)$$

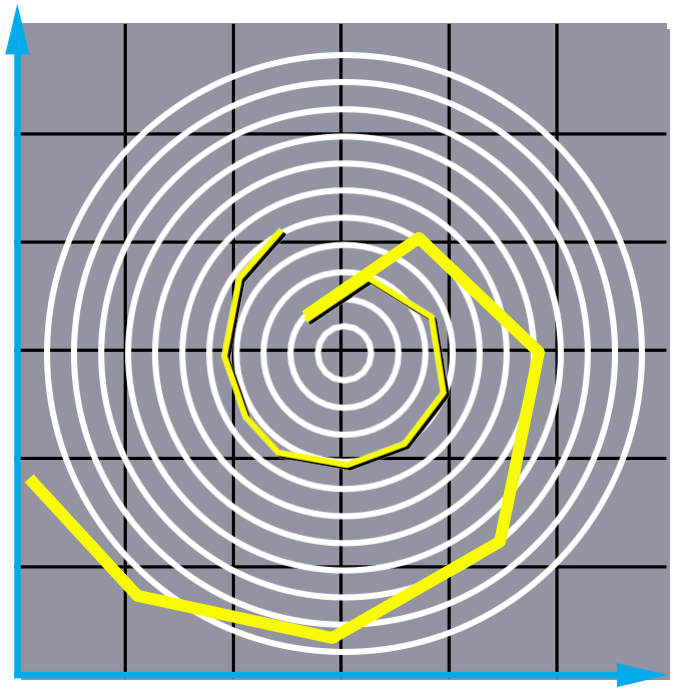
$\Longrightarrow$ **intégration d'Euler**

- simple
- Δt grand $\Longrightarrow$ erreur forte
- Δt petit $\Longrightarrow$ lent



Limitations d'Euler : peu précis

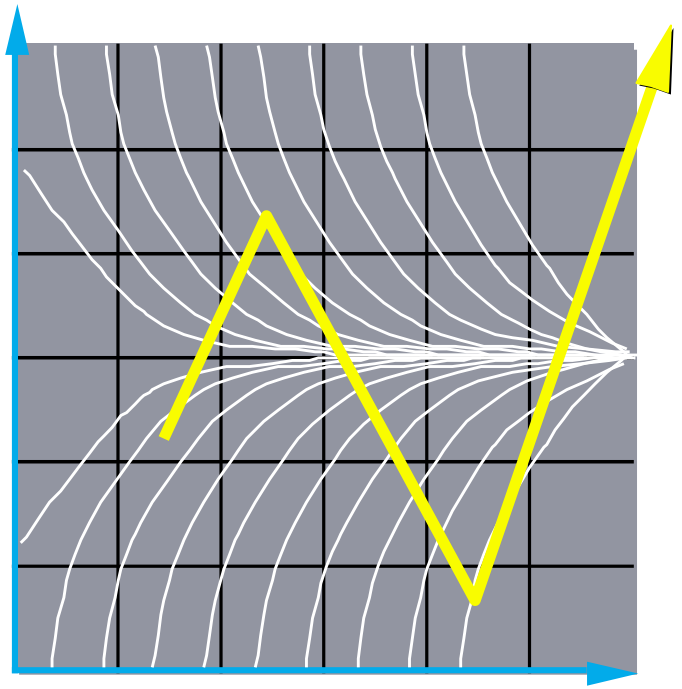
Ex : $\mathbf{x}(t)$ trajectoire circulaire transformée en spirale



$\Longrightarrow$ **approximation linéaire de** $\mathbf{x}(t)$

Limitations d'Euler : peu stable

Ex : $\mathbf{x}(t)$ diverge au lieu de converger



$\Longrightarrow$ **pas de temps** Δt **constant**

Limitations d'Euler

Si $\mathbf{x}(t)$ est n fois continue, développement de Taylor :

$$\mathbf{x}(t_0 + \Delta t) = \mathbf{x}_0 + \Delta t\, \dot{\mathbf{x}}(t_0) + \frac{\Delta t^2}{2} \ddot{\mathbf{x}}(t_0) + \ldots + \frac{\Delta t^n}{n!} \frac{\partial^n \mathbf{x}}{\partial t^n}(t_0)$$

$\Longrightarrow$ Euler correct seulement si $\mathbf{x}(t)$ est **linéaire**

$\Longrightarrow$ Erreur dominée par $(\Delta t^2/2)\ddot{\mathbf{x}}(t_0)$, c.-à-d. en $O(\Delta t^2)$

$\Longrightarrow$ Si on sait calculer $\ddot{\mathbf{x}}(t_0)$, alors erreur en $O(\Delta t^3)$

Méthode du point milieu

Développement de Taylor à l'ordre 2 en $\mathbf{x}$:

$$\mathbf{x}(t_0+\Delta t) = \mathbf{x}_0 + \Delta t\,\dot{\mathbf{x}}(t_0) + \frac{\Delta t^2}{2}\ddot{\mathbf{x}}(t_0) + O(\Delta t^3)$$

où la dérivée première $\dot{\mathbf{x}} = f(\mathbf{x}(t),t) = f(\mathbf{x}(t))$
(pour simplifier)

et donc la dérivée seconde $\ddot{\mathbf{x}} = \dfrac{\partial f}{\partial \mathbf{x}}\dot{\mathbf{x}} = f' f$

Méthode du point milieu

Développement de Taylor à l'ordre 1 sur f :

$$f(\mathbf{x}_o + \Delta\mathbf{x}) = f(\mathbf{x}_o) + \Delta\mathbf{x} f'(\mathbf{x}_o) + O(\Delta\mathbf{x}^2)$$

En choisissant $\Delta\mathbf{x} = \frac{\Delta t}{2} f(\mathbf{x}_0)$ on obtient :

$$f(\mathbf{x}_o + \frac{\Delta t}{2} f(\mathbf{x}_0)) = f(\mathbf{x}_o) + \frac{\Delta t}{2} \underbrace{f(\mathbf{x}_0) f'(\mathbf{x}_o)} + O(\Delta t^2)$$

$$= f(\mathbf{x}_o) + \frac{\Delta t}{2} \ddot{\mathbf{x}}_0 + O(\Delta t^2)$$

En multipliant par Δt et en réorganisant :

$$\Delta t f(\mathbf{x}_o + \frac{\Delta t}{2} f(\mathbf{x}_0)) - \Delta t \dot{\mathbf{x}}_o = \frac{\Delta t^2}{2} \ddot{\mathbf{x}}_0 + O(\Delta t^3)$$

Méthode du point milieu

$$\Delta t\, f\left(\mathbf{x}_o + \frac{\Delta t}{2} f(\mathbf{x}_0)\right) - \Delta t\, \dot{\mathbf{x}}_o = \frac{\Delta t^2}{2} \ddot{\mathbf{x}}_0 + O(\Delta t^3)$$

En réinjectant dans :

$$\mathbf{x}(t_0 + \Delta t) = \mathbf{x}_0 + \Delta t\, \dot{\mathbf{x}}(t_0) + \frac{\Delta t^2}{2} \ddot{\mathbf{x}}(t_0) + O(\Delta t^3)$$

On obtient :

$$\mathbf{x}(t_0 + \Delta t) = \mathbf{x}_0 + \Delta t\, f\left(\mathbf{x}_0 + \frac{\Delta t}{2} f(\mathbf{x}_0)\right)$$

Méthode du point milieu

a) Calculer une itération d'Euler :

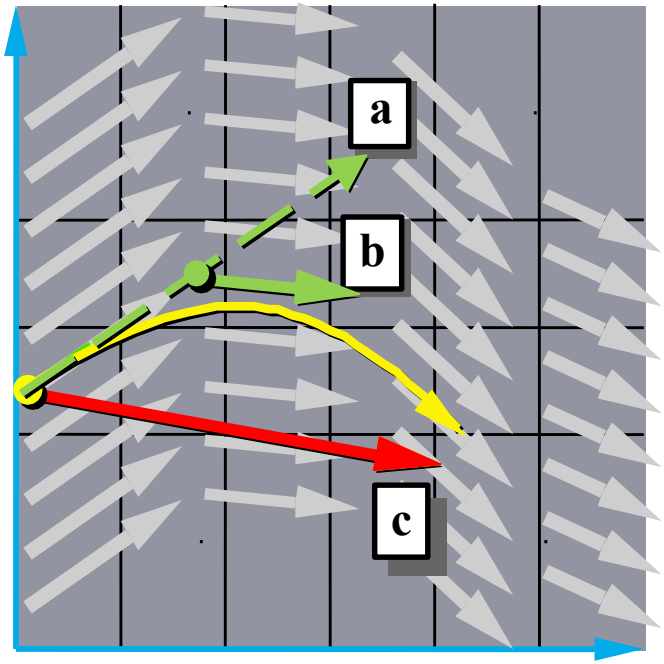
$$\Delta \mathbf{x} = \Delta t\, \mathbf{f}(\mathbf{x}, t)$$

b) Évaluer la dérivée à mi-distance :

$$\mathbf{f}_{\text{mid}} = \mathbf{f}\left(\mathbf{x} + \frac{\Delta \mathbf{x}}{2}, t + \frac{\Delta t}{2}\right)$$

c) L'utiliser pour mettre à jour $\mathbf{x}$:

$$\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \Delta t\, \mathbf{f}_{\text{mid}}$$



$\Longrightarrow$ **Meilleure approximation**

$\Longrightarrow$ **Plus coûteux** (2 évaluations de $\mathbf{f}$)

Pas de temps adaptatifs

Performance $\Longrightarrow$ Δt **le plus grand possible**

Stabilité $\Longrightarrow$ Δt **suffisamment petit**

$\Longrightarrow$ adapter Δt durant la simulation

Pour Euler :

- calculer $\mathbf{x_a}$ avec une itération d'Euler de pas Δt
- calculer $\mathbf{x_b}$ avec deux itérations d'Euler de pas $\Delta t/2$

$\mathbf{x_a}$ et $\mathbf{x_b}$ diffèrent de $O(\Delta t^2)$

$\Longrightarrow$ $e = |\, \mathbf{x_a} - \mathbf{x_b} \,|$ est une **mesure de l'erreur** commise par une itération d'Euler de pas Δt

Pas de temps adaptatifs

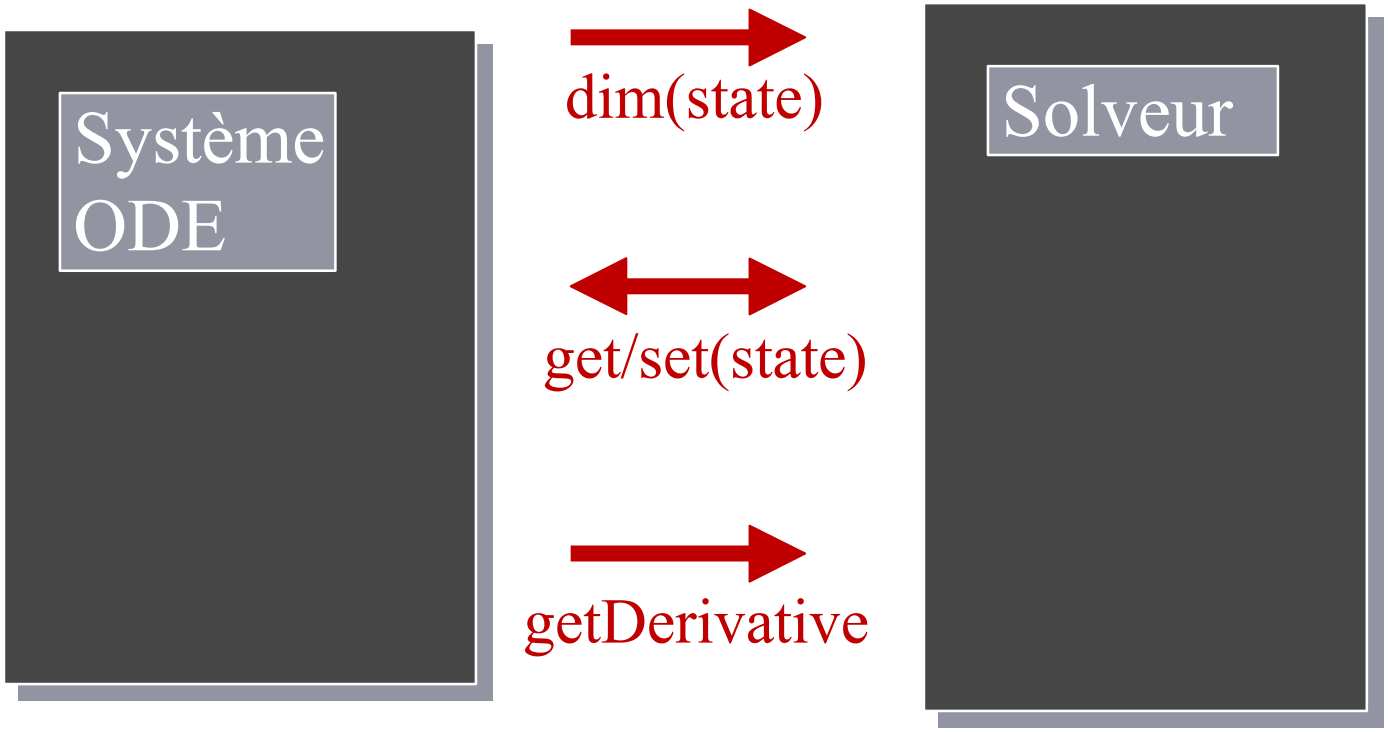
Ex : on veut une erreur $e < 10^{-4}$ par pas d'Euler.

Si l'erreur courante est de 10^{-8},
on peut **augmenter** le pas de $\left(\frac{10^{-4}}{10^{-8}}\right)^{\frac{1}{2}} \Delta t = 100\Delta t$

Si l'erreur courante est de 10^{-3},
on doit **diminuer** le pas de $\left(\frac{10^{-4}}{10^{-3}}\right)^{\frac{1}{2}} \Delta t \approx 0.316\Delta t$

Implémentation

Solveur générique



Physique d'une particule

Équation d'ordre 2 (dérivée seconde) :

$$\ddot{\mathbf{x}}(t) = \frac{\mathbf{F}\left(t, \mathbf{x}(t), \dot{\mathbf{x}}(t)\right)}{m}$$

$\Leftrightarrow$ 2 équations d'ordre 1 couplées :

$$\begin{cases} \dot{\mathbf{x}} = \mathbf{v} \\ \dot{\mathbf{v}} = \mathbf{F}/m \end{cases}$$

$\Longrightarrow$ difficile à résoudre

Espace des phases

Concaténation de $\mathbf{x}$ et $\mathbf{v}$: $\begin{bmatrix} \mathbf{x} \\ \mathbf{v} \end{bmatrix}$

= point 6D dans l'**espace des phases**

Concaténation de $\dot{\mathbf{x}}$ et $\dot{\mathbf{v}}$: $\begin{bmatrix} \dot{\mathbf{x}} \\ \dot{\mathbf{v}} \end{bmatrix}$

= vitesse 6D dans l'**espace des phases**

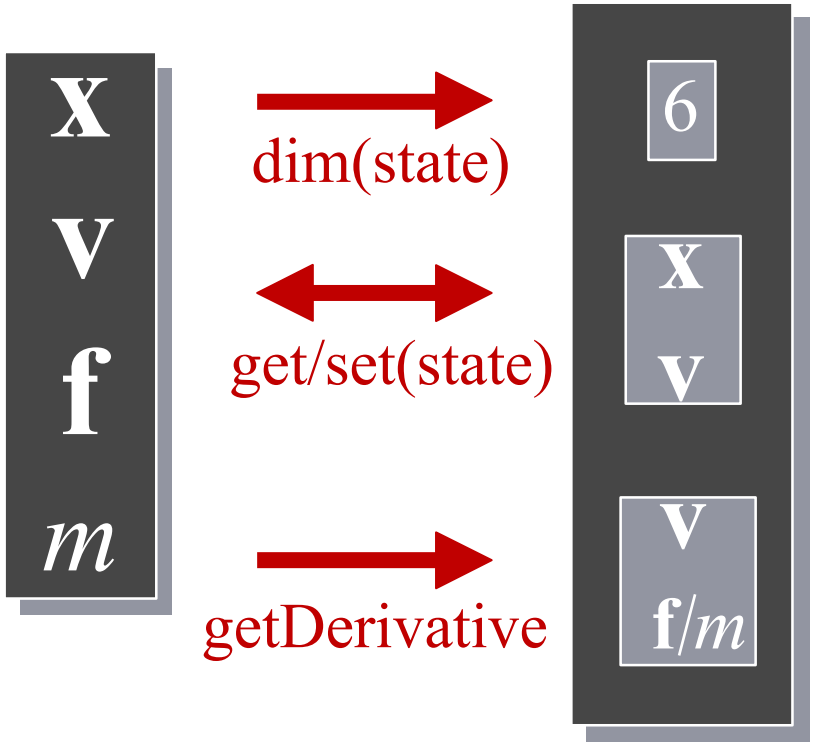
1 seule équation différentielle d'ordre 1 :

$$\begin{bmatrix} \dot{\mathbf{x}} \\ \dot{\mathbf{v}} \end{bmatrix} = \begin{bmatrix} \mathbf{v} \\ \mathbf{F}/m \end{bmatrix}$$

$\Longrightarrow$ plus simple à résoudre

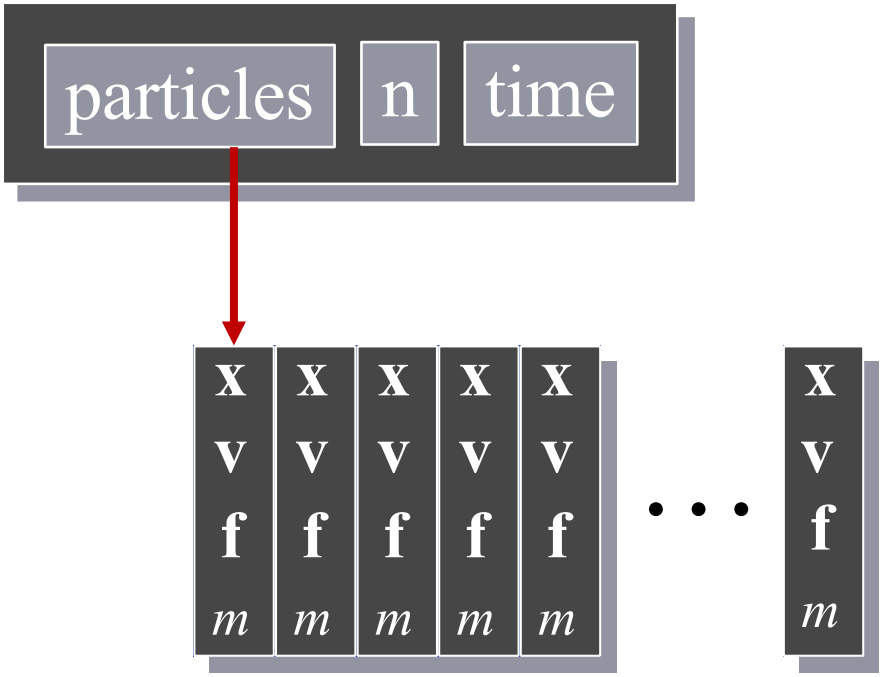
Implémentation

Une particule



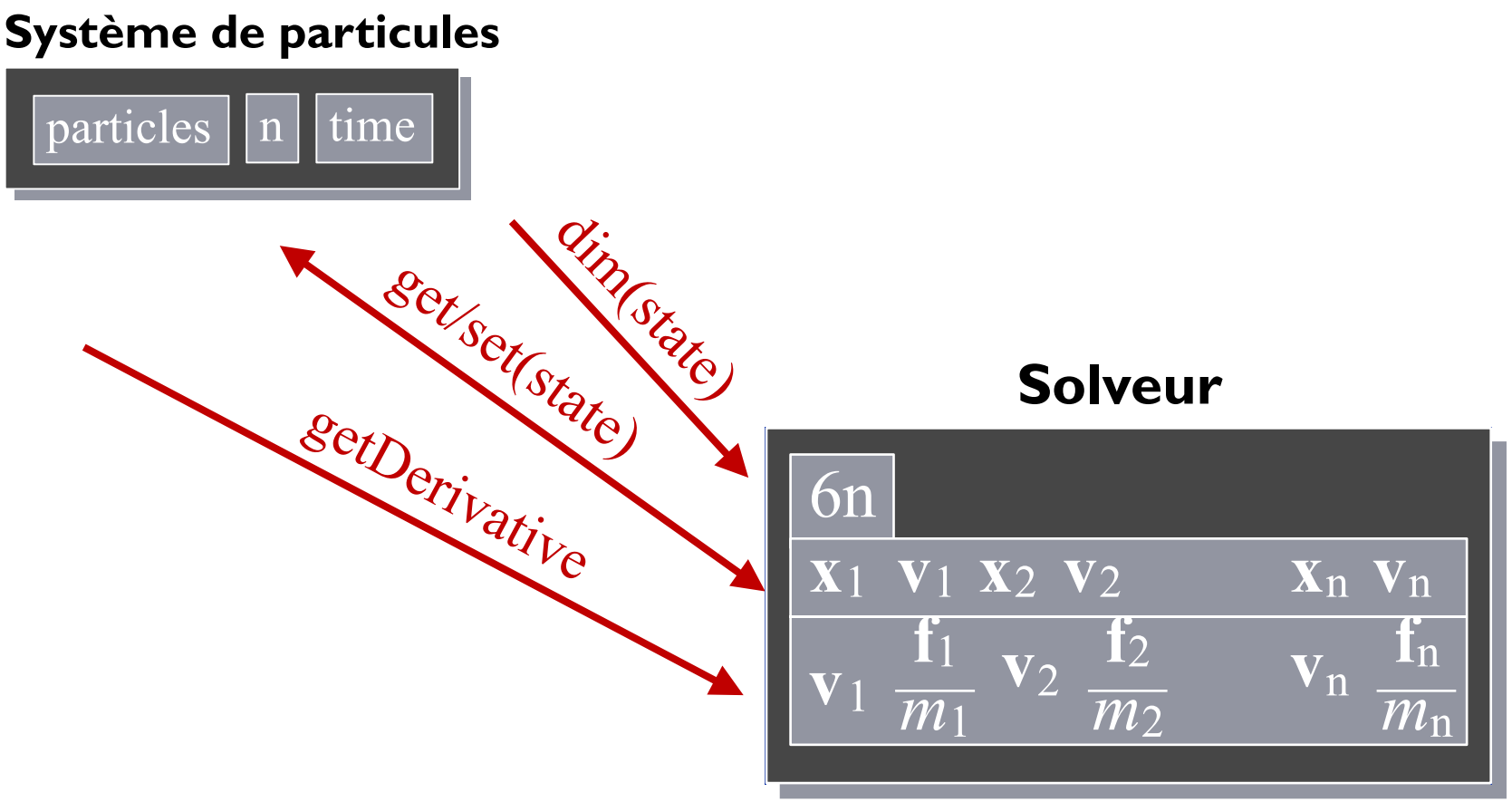
Implémentation

Système de particules



Implémentation

Interface avec le solveur



Forces

- **Unaires**
- **N-aires**
- **D’interaction spatial**

Forces unaires

Gravité de la terre

force **sans contact,** proportionnelle à la masse m et dirigée par le vecteur constant de **gravité** $\mathbf{g}$:

$$\mathbf{f} = m\mathbf{g}$$

Augmente l'énergie du système

Forces unaires

Amortissement visqueux

résistance au mouvement, opposée à la vitesse $\mathbf{v}(t)$
et pondéré par le **coefficient de viscosité** ν :

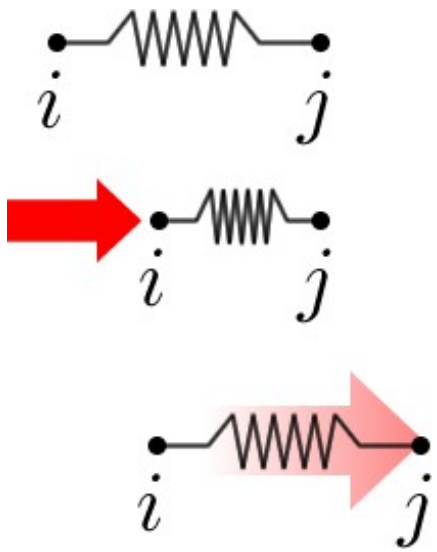
$$\mathbf{f} = -\nu\mathbf{v}$$

Diminue l'énergie du système $\Longrightarrow$ stabilise le système

Forces n-aires

Système masse – ressort

ressort reliant 2 masses i et j



Lorsqu'on applique une force à une des masses,, le ressort est compressé et compensera en appliquant une force opposée à l'autre masse pour se décompresser.

$$\mathbf{f}_{i,j} = -k_s \left(||\mathbf{l}|| - r \right) \frac{\mathbf{l}}{||\mathbf{l}||}$$

$$\mathbf{f}_{j,i} = -\mathbf{f}_{i,j}$$

avec k_s la **constante de rigidité** du ressort

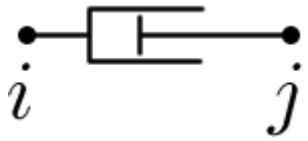
r la **longueur** du ressort au **repos**

$\mathbf{l} = \mathbf{x}_i(t) - \mathbf{x}_j(t)$

Forces n-aires

Système masse – amortisseur

amortisseur reliant 2 masses i et j



Force inverse à la vitesse appliquée pour compresser l'amortisseur :

$$\mathbf{f}_{i,j} = -k_d \frac{(\mathbf{v}_i(t) - \mathbf{v}_j(t)) \cdot \mathbf{l}}{||\mathbf{l}||} \frac{\mathbf{l}}{||\mathbf{l}||}$$

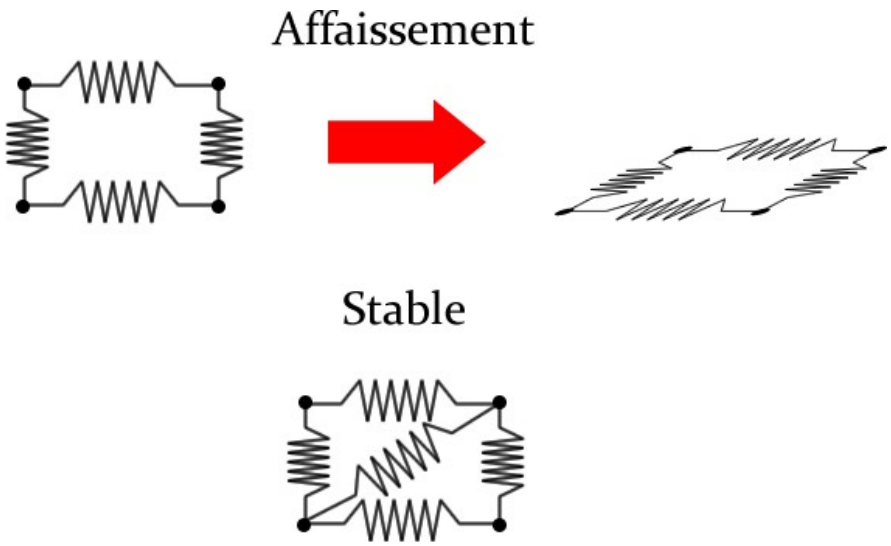
avec k_d la **constante d'amortissement**

Forces n-aires

Réseaux de masse – ressort

permet de simuler divers types de matériaux, en fonction des constantes données au ressort.

Pour garantir un système plus stable, favoriser des formes triangulaires :



Forces d'interaction spatial

Forces s'appliquant à **toutes les pairs** (ou n-tuples) de particules, par exemple :

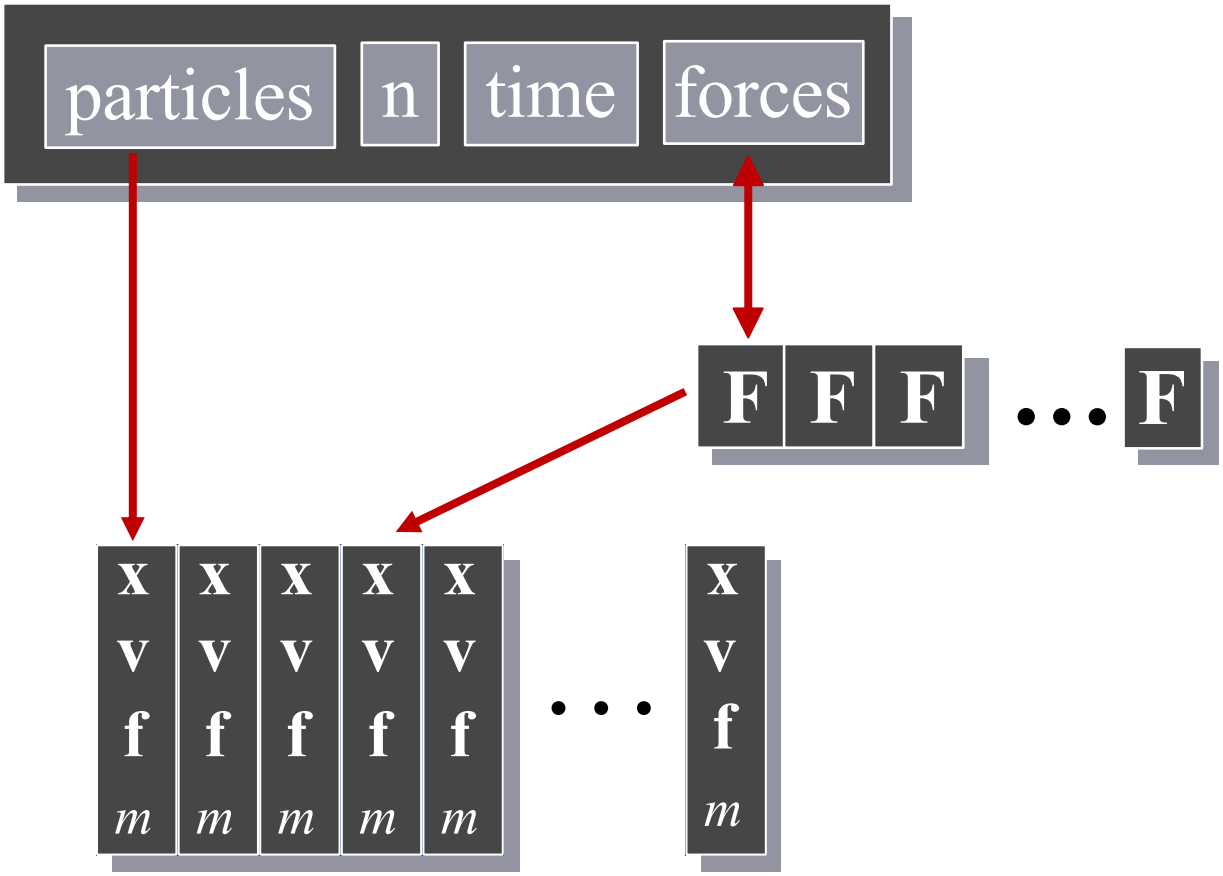
- gravitation
- électromagnétisme
- interaction atomique faible et forte.

Utilisées pour approximer le comportement des fluides

$\Longrightarrow$ complexité en $O(n^2)$

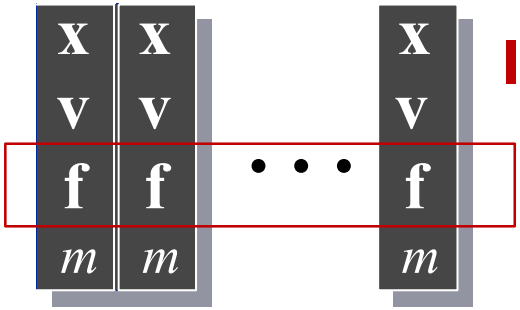
Implémentation

Système de particules

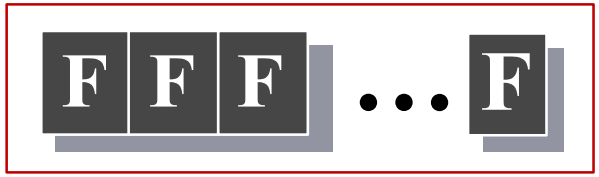


Implémentation

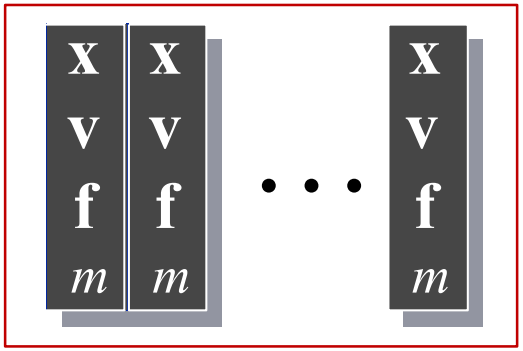
Boucle de calcul des dérivées



1. Mettre à zéro les forces



2. Appliquer les forces
(i.e., sommer pour chaque particule)



3. Retourner [v, f/*m*,…] au solveur

Collisions

Deux problèmes à résoudre :

1. **détection**
2. **réponse**

Problèmes (très) difficiles dans le cas général !

Détection

Particules infinitésimales

Probabilité d'inter-collisions nulle

$\Longrightarrow$ **collisions uniquement avec la scène**

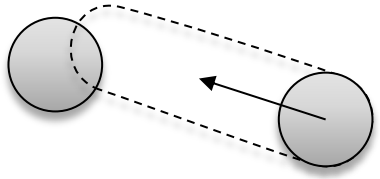
Trajectoire linéaire durant un petit pas de temps

$\Longrightarrow$ **tracé de rayon** (et ses structures d'accélérations)

Particules avec un volume

Collisions entre particules plus négligeables

1. Fixer le repère d'une des particules (statique)
2. Extruder le volume de l'autre le long de sa trajectoire
3. Tester l'intersection des volumes (*ex.* : sphère – capsule)



Réponse

Méthode dite de pénalité

force répulsive (ressort) dont l'intensité (la raideur)est proportionnelle à la distance d'interpénétration

$\Longrightarrow$ n'empêche pas complètement les interpénétrations (sauf à anticiper les collisions)

$\Longrightarrow$ instabilités numériques lorsque la raideur est très grande

$\Longrightarrow$ requière plusieurs pas d'intégration pour faire effet

Réponse

Méthode à base d'impulsion

modification **instantanée** de la **vitesse** hors de la boucle d'intégration

Basé sur la **quantité de mouvement** : $\mathbf{p}(t) = m\mathbf{v}(t)$

Au moment de la collision :

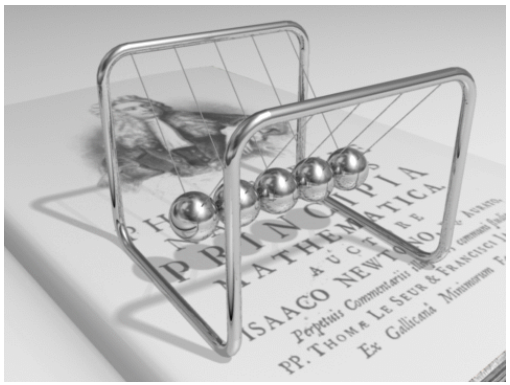
$$\Delta\mathbf{v}(t) = \frac{\Delta\mathbf{p}(t)}{m}$$ ← **impulsion**

Dans un système fermé, conservation de la quantité de mouvement :

$$\sum_i \mathbf{p}'_i(t) = \sum_i \mathbf{p}_i(t)$$

Conservation de l'énergie cinétique totale :

$$\sum_i \frac{1}{2} m_i \mathbf{v}'^2_i(t) = \sum_i \frac{1}{2} m_i \mathbf{v}_i^2(t)$$



Réponse élastique

En **1D**, pour **deux particules** de vitesse v_1 et v_2 et de masses respectives m_1 et m_2 :

$$\begin{cases} m_1 v_1 + m_2 v_2 = m_1 v'_1 + m_2 v'_2 \\ m_1 v_1^2 + m_2 v_2^2 = m_1 v'^2_1 + m_2 v'^2_2 \end{cases}$$

où « prime » indique la vitesse après la collision.

En résolvant le système, **vitesses après collision** :

$$v'_1 = \frac{m_1 v_1 + m_2 v_2 + m_2 (v_2 - v_1)}{m_1 + m_2}$$

$$v'_2 = \frac{m_1 v_1 + m_2 v_2 + m_1 (v_1 - v_2)}{m_1 + m_2}$$

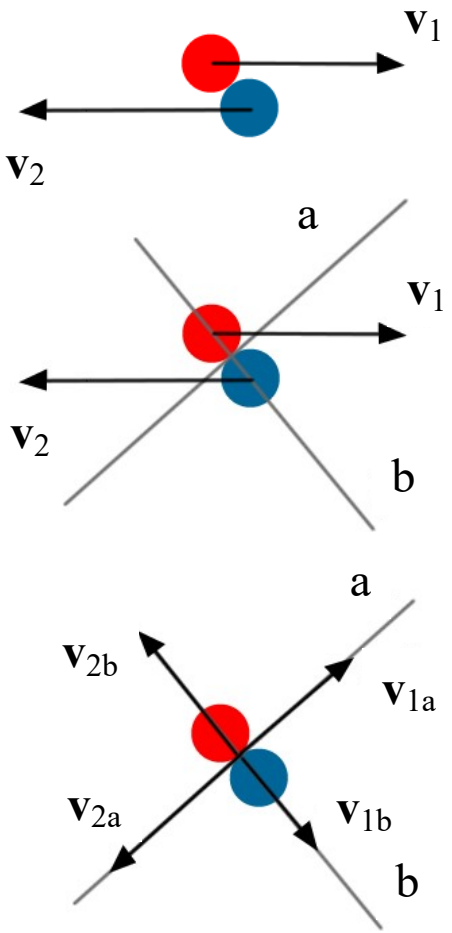
Remarque : si $m_1 = m_2$ alors $v'_1 = v_2$ et $v'_2 = v_1$

Réponse élastique

En **nD**, diviser en **n** collisions **1D**.

Par exemple, en **2D** :

1. On trace un axe tangent à la collision (axe a) et un axe perpendiculaire à la collision (axe b).
2. On projette $\mathbf{v}_1$ et $\mathbf{v}_2$ sur a et b et on garde leur norme : $v_{1a}, v_{1b}, v_{2a}, v_{2b}$
3. On résout les problèmes 1D sur chaque axe :
$$v'_{1a}, v'_{1b}, v'_{2a}, v'_{2b}$$
4. On les re-multiplie par les vecteurs vitesses projetés normalisés et on somme :
$$\mathbf{v}'_1 = v'_{1a}\frac{\mathbf{v}_{1a}}{||\mathbf{v}_{1a}||} + v'_{1b}\frac{\mathbf{v}_{1b}}{||\mathbf{v}_{1b}||}$$
$$\mathbf{v}'_2 = v'_{2a}\frac{\mathbf{v}_{2a}}{||\mathbf{v}_{2a}||} + v'_{2b}\frac{\mathbf{v}_{2b}}{||\mathbf{v}_{2b}||}$$



Réponse inélastique

Coefficient de restitution e supplémentaire :

$$v_1' = \frac{m_1 v_1 + m_2 v_2 + m_2 e (v_2 - v_1)}{m_1 + m_2}$$

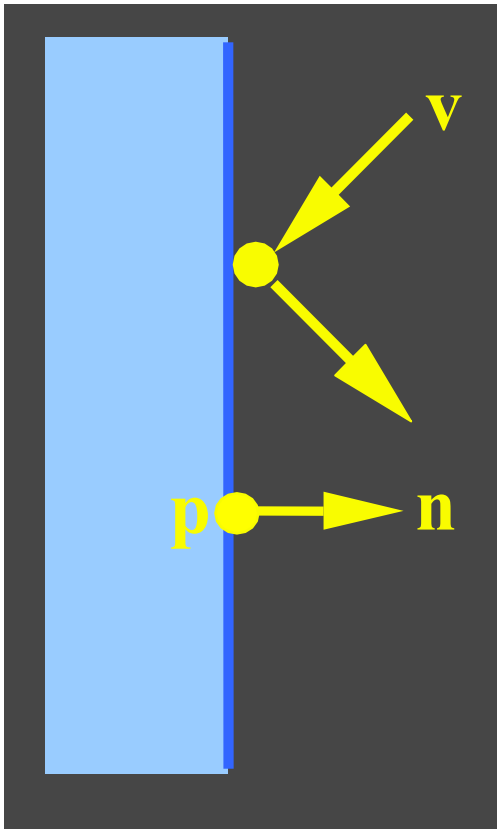
$$v_2' = \frac{m_1 v_1 + m_2 v_2 + m_1 e (v_1 - v_2)}{m_1 + m_2}$$

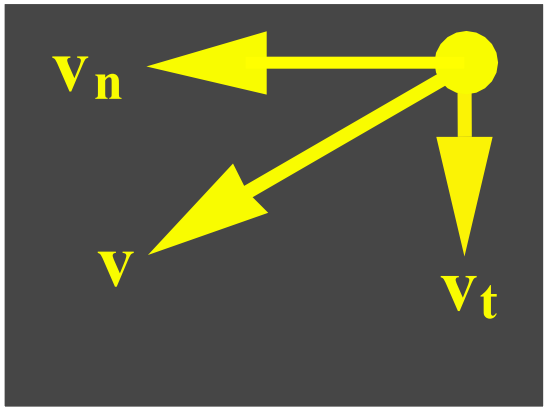
Ratio de vitesse avant et après l'impact

- si $e = 1 \Longrightarrow$ collision parfaitement élastique,
- si $e = 0 \Longrightarrow$ les 2 objets restent collés ensemble,
- mesurés empiriquement pour les matériaux connus

Collision avec un plan

Particule de vitesse $\mathbf{v}$ – plan passant par $\mathbf{p}$ de normal $\mathbf{n}$



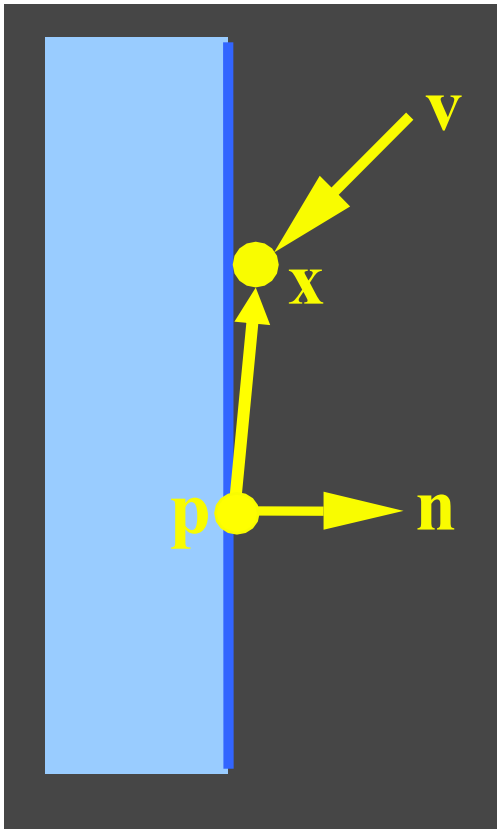


$$\mathbf{v}_n = (\mathbf{n} \cdot \mathbf{v})\mathbf{n}$$

$$\mathbf{v}_t = \mathbf{v} - \mathbf{v}_n$$

Détection

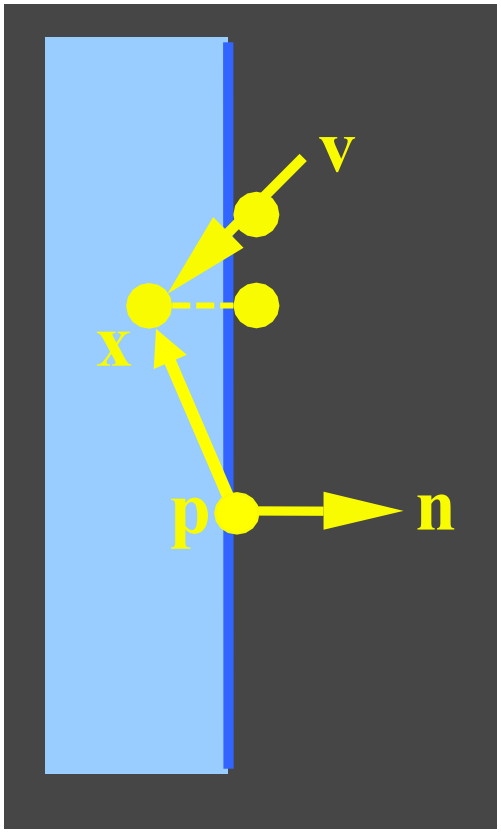
Collision si proche du mur (ε) et dirigée vers lui :



$$\begin{cases} (\mathbf{x} - \mathbf{p}) \cdot \mathbf{n} < \varepsilon \\ \mathbf{n} \cdot \mathbf{v} < 0 \end{cases}$$

Détection

Si détection après dépassement (*overshooting*),

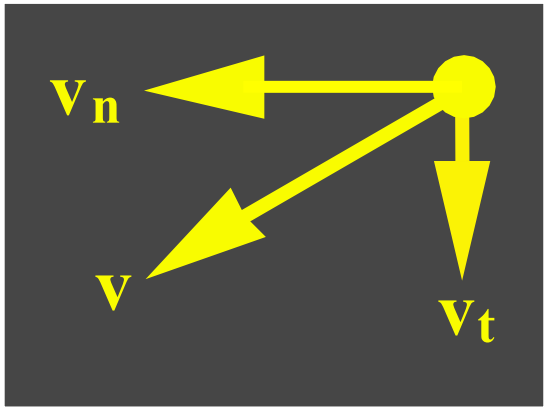


2 solutions :

1. Projection sur le point le plus proche (*fixing*)
2. Remonter le temps (*backtracking*)

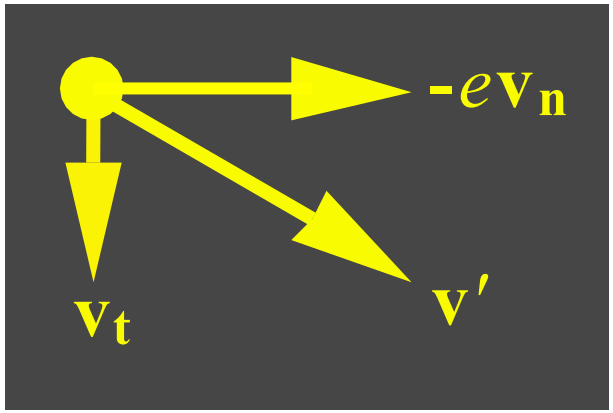
Réponse impulsionnelle

Inversion de la composante normale de la vitesse pondérée par le coefficient de restitution $e \in [0,1]$



Avant collision :

$$\mathbf{v} = \mathbf{v}_t + \mathbf{v}_n$$

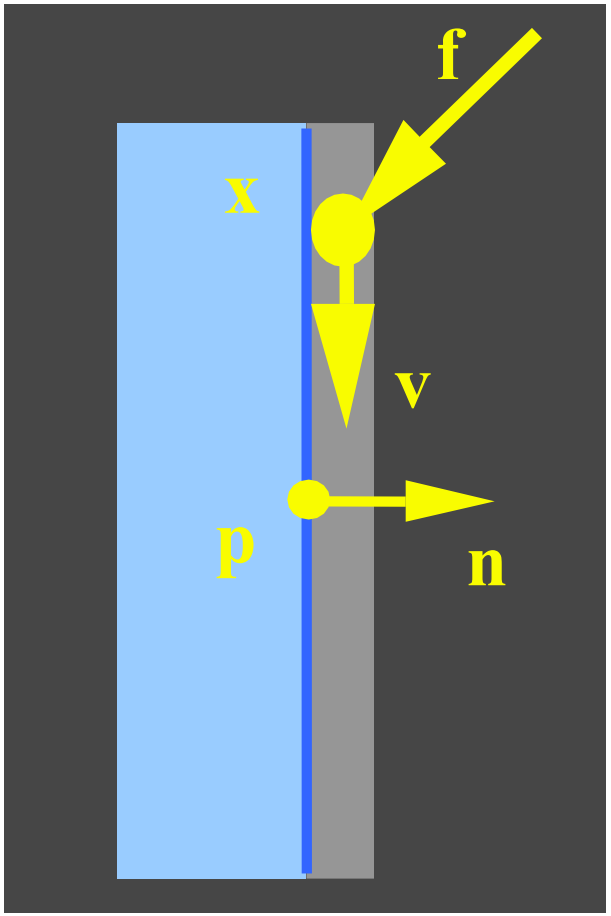


Après collision :

$$\mathbf{v}' = \mathbf{v}_t - e\mathbf{v}_n$$

Particule en contact avec le plan

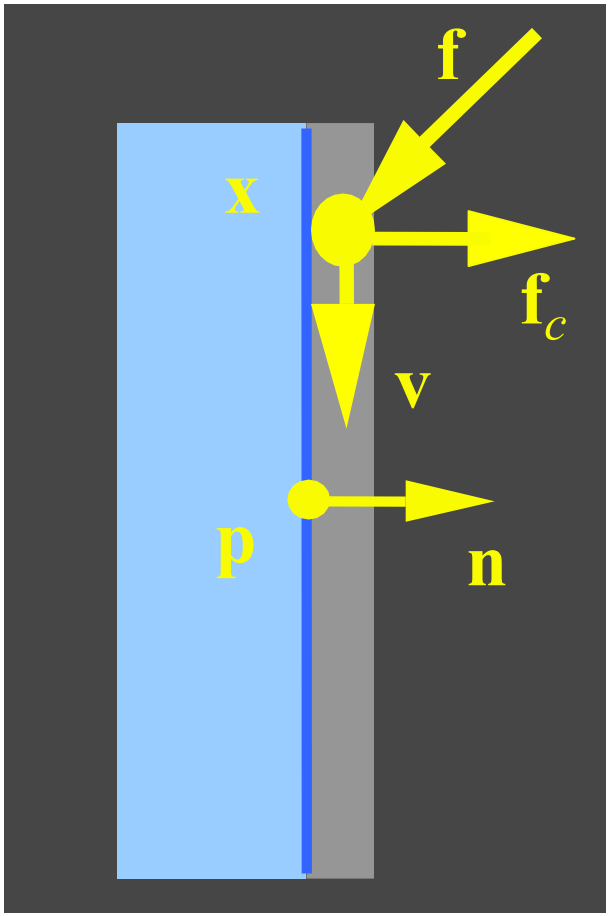
Si la composante normale de v est nulle :



$$\begin{cases} |(\mathbf{x} - \mathbf{p})\cdot\mathbf{n}| < \varepsilon \\ |\mathbf{n}\cdot\mathbf{v}| < \varepsilon \end{cases}$$

Particule en contact avec le plan

Si $\mathbf{n} \cdot \mathbf{f} < 0$, **force de contact $\mathbf{f}_c$** exercée par le plan pour annuler la composante normale de $\mathbf{f}$:



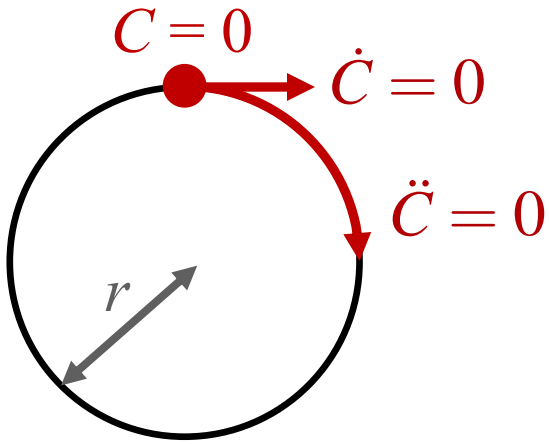
$$\mathbf{f}_c = -(\mathbf{n} \cdot \mathbf{f})\mathbf{n}$$

Contraintes

Il serait plus simple d'imposer une **contrainte**

$\Longrightarrow$ **force requise déduite automatiquement**

Exemple : particule 2D contrainte sur un cercle de centre $(0,0)$ et de rayon r



- Contrainte implicite sur sa position :

$$C(\mathbf{x}) = ||\mathbf{x}||^2 - r = 0$$

- Vitesses valides :

$$\dot{C}(\mathbf{x}) = \mathbf{x} \cdot \dot{\mathbf{x}} = 0$$

- Accélérations valides :

$$\ddot{C}(\mathbf{x}) = \ddot{\mathbf{x}} \cdot \mathbf{x} + \dot{\mathbf{x}} \cdot \dot{\mathbf{x}} = 0$$

- Or $\ddot{\mathbf{x}} = \dfrac{\mathbf{f} + \mathbf{f}_c}{m} \Rightarrow \mathbf{f}_c \cdot \mathbf{x} = -\mathbf{f} \cdot \mathbf{x} - m\dot{\mathbf{x}} \cdot \dot{\mathbf{x}}$

Contraintes

Une équation : $\mathbf{f}_c \cdot \mathbf{x} = -\mathbf{f} \cdot \mathbf{x} - m\dot{\mathbf{x}} \cdot \dot{\mathbf{x}}$

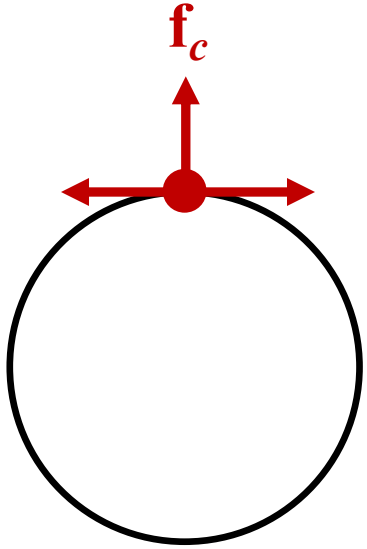
…mais deux inconnues (2 composantes de $\mathbf{f}_c$)

$\Longrightarrow$ **assurer que $\mathbf{f}_c$ est passive**, c.-à-d. n'ajoute ni n'enlève d'énergie au système :

$$\mathbf{f}_c \cdot \dot{\mathbf{x}} = 0, \ \forall \dot{\mathbf{x}} \mid \dot{C} = 0$$

$$\Leftrightarrow \ \mathbf{f}_c = \lambda \mathbf{x}$$

$$\Rightarrow \ \lambda = \frac{-\mathbf{f} \cdot \mathbf{x} - m\dot{\mathbf{x}} \cdot \dot{\mathbf{x}}}{\mathbf{x} \cdot \mathbf{x}}$$



$\Longrightarrow$ principe des *puissances virtuelles*

Système de particules contraint

$$\ddot{\mathbf{q}} = \mathbf{M}^{-1}\mathbf{Q} \qquad \text{où} \quad \mathbf{q} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_n \end{bmatrix}$$

avec m **contraintes** :

$$\mathbf{C}(\mathbf{q}) = \begin{bmatrix} C_1(\mathbf{q}) \\ C_2(\mathbf{q}) \\ \vdots \\ C_\mathrm{m}(\mathbf{q}) \end{bmatrix}$$

$$\mathbf{Q} = \begin{bmatrix} \mathbf{f}_1 & \mathbf{f}_2 & \cdots & \mathbf{f}_n \end{bmatrix}$$

$$\mathbf{M} = \begin{bmatrix} m_1 & & & & & & \\ & m_1 & & & & & \\ & & m_1 & & & & \\ & & & \cdots & & & \\ & & & & m_n & & \\ & & & & & m_n & \\ & & & & & & m_n \end{bmatrix}$$

Système de particules contraint

$$\ddot{\mathbf{q}} = \mathbf{M}^{-1}\mathbf{Q} \qquad \text{où} \quad \mathbf{q} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_n \end{bmatrix}$$

avec m **contraintes** :

$$\mathbf{C}(\mathbf{q}) = \begin{bmatrix} C_1(\mathbf{q}) \\ C_2(\mathbf{q}) \\ \vdots \\ C_\mathrm{m}(\mathbf{q}) \end{bmatrix}$$

$$\mathbf{Q} = \begin{bmatrix} \mathbf{f}_1 & \mathbf{f}_2 & \cdots & \mathbf{f}_n \end{bmatrix}$$

$$\mathbf{M} = \begin{bmatrix} m_1 & & & & & & \\ & m_1 & & & & & \\ & & m_1 & & & & \\ & & & \cdots & & & \\ & & & & m_n & & \\ & & & & & m_n & \\ & & & & & & m_n \end{bmatrix}$$

Système de particules contraint

Initialement $\mathbf{q}$ et $\dot{\mathbf{q}}$ sont **valides**, c.-à-d. $\mathbf{C} = \dot{\mathbf{C}} = \mathbf{0}$

On cherche un **ensemble de forces de contrainte** $\mathbf{Q}_c$ garantissant $\ddot{\mathbf{C}} = \mathbf{0}$

$\Longrightarrow$ calcul des dérivées de $\mathbf{C}$

$$\dot{\mathbf{C}} = \frac{\partial \mathbf{C}}{\partial \mathbf{q}} \dot{\mathbf{q}} = \mathbf{J}\dot{\mathbf{q}}$$ avec $\mathbf{J}$ la **jacobienne** de $\mathbf{C}$

$$\ddot{\mathbf{C}} = \dot{\mathbf{J}}\dot{\mathbf{q}} + \mathbf{J}\ddot{\mathbf{q}} \quad \text{où } \dot{\mathbf{J}} = \frac{\partial \dot{\mathbf{C}}}{\partial \dot{\mathbf{q}}}$$

D'où :

$$\ddot{\mathbf{C}} = \dot{\mathbf{J}}\dot{\mathbf{q}} + \mathbf{J}\mathbf{M}^{-1}(\mathbf{Q} + \mathbf{Q}_c) = \mathbf{0}$$

$$\Leftrightarrow \mathbf{J}\mathbf{M}^{-1}\mathbf{Q}_c = -\dot{\mathbf{J}}\dot{\mathbf{q}} - \mathbf{J}\mathbf{M}^{-1}\mathbf{Q}$$

Système de particules contraint

Par le principe des *puissances virtuelles* :

$$\mathbf{Q}_c = \mathbf{J}^\top \boldsymbol{\lambda} \qquad \text{où} \quad \boldsymbol{\lambda} = \begin{bmatrix} \lambda_1 & \lambda_2 & \cdots & \lambda_n \end{bmatrix}$$

$$\Rightarrow \; \mathbf{J}\mathbf{M}^{-1}\mathbf{J}^\top \boldsymbol{\lambda} = -\dot{\mathbf{J}}\dot{\mathbf{q}} - \mathbf{J}\mathbf{M}^{-1}\mathbf{Q}$$

On résout pour trouver $\boldsymbol{\lambda}$ et on obtient **l'accélération contrainte** :

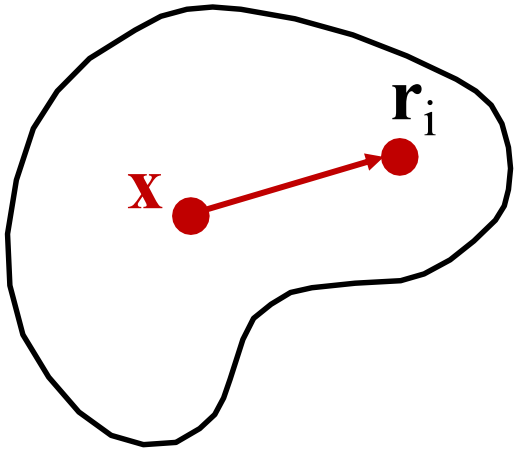
$$\ddot{\mathbf{q}} = \mathbf{M}^{-1}(\mathbf{Q} + \mathbf{J}^\top \boldsymbol{\lambda})$$

Objets rigides

Centre de masse (barycentre)

point **x** où la masse est répartie également dans n'importe quelle direction :

$$\mathbf{x} = \frac{\sum m_i \mathbf{r}_i}{\sum m_i}$$



Force appliquée en axe avec le centre de masse (provoque un déplacement vers la droite)

Force appliquée de façon désaxée à l'axe principal. (provoque un déplacement vers la droite ET une rotation)

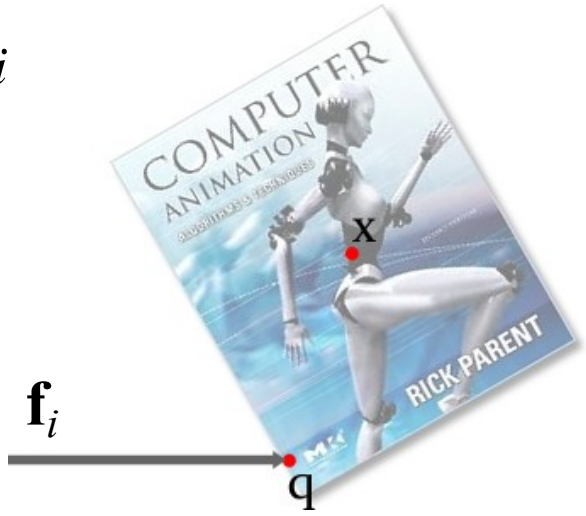
Objets rigides

La force générant le **déplacement** est appelée **force linéaire**.

La force générant la **rotation** est appelée **couple :** ("torque" en anglais)

$$\tau_i = (\mathbf{q} - \mathbf{x}) \times \mathbf{f}_i$$

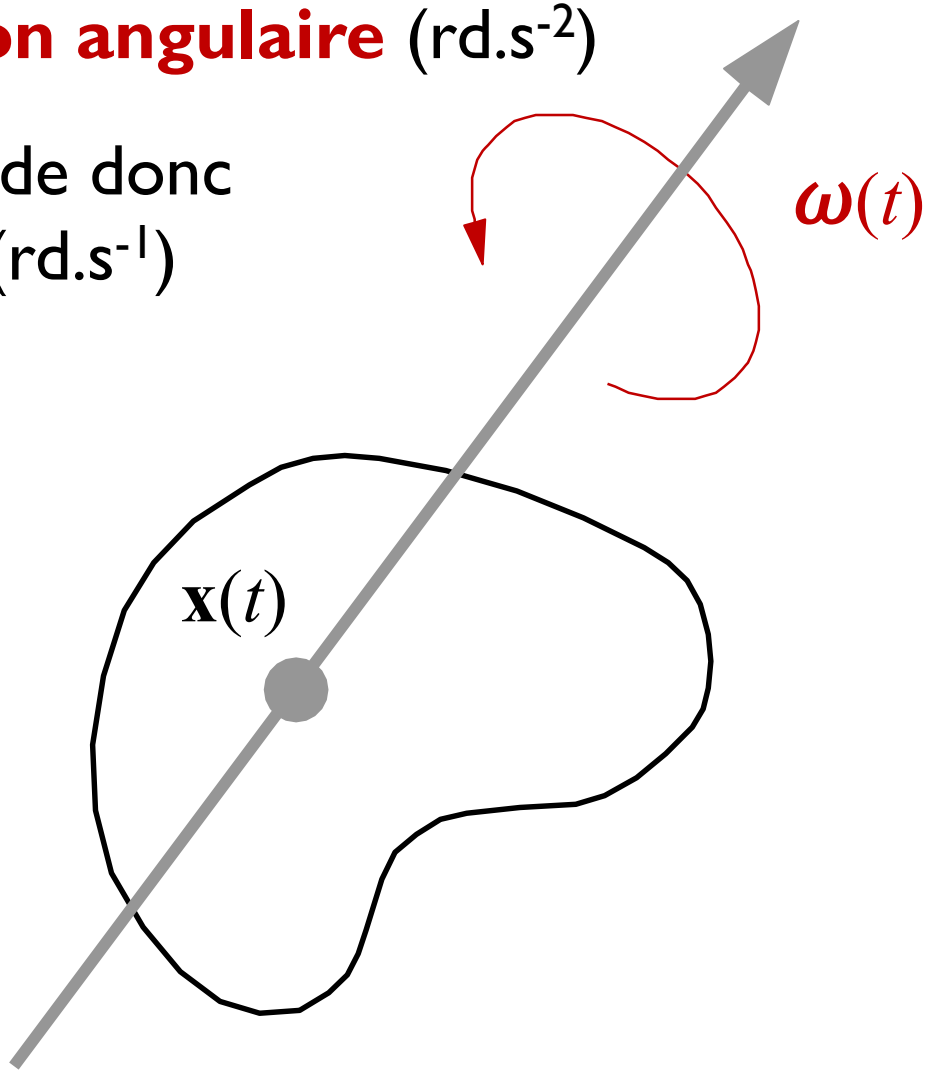
$$\tau = \sum_i \tau_i$$



Objets rigides

Comme pour une force linéaire, le couple génère une accélération : l'**accélération angulaire** (rd.s^{-2})

Un objet en rotation possède donc une **vitesse angulaire** ω (rd.s^{-1})



Quantité de mouvement

Mouvement d'un objet dont la masse est répartie dans l'espace exprimé par sa **quantité de mouvement** (*momentum*)

Linéaire : $\mathbf{P}(t) = M\mathbf{v}(t)$

$$\dot{\mathbf{P}}(t) = M\dot{\mathbf{v}}(t) = M\frac{\mathbf{F}(t)}{M} = \mathbf{F}$$

Angulaire : $\mathbf{L}(t) = I(t)\omega(t)$ avec $I(t)$ le **tenseur inertiel**

$$\dot{\mathbf{L}}(t) = \tau(t)$$

Tenseur inertiel

Matrice 3x3 symétrique décrivant la **répartition du poids** par rapport au centre de masse :

$$I = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{xy} & I_{yy} & I_{yz} \\ I_{xz} & I_{yz} & I_{zz} \end{bmatrix}$$

$$I_{xx} = \sum m_i(y_i^2 + z_i^2) \qquad I_{xy} = \sum m_i x_i y_i$$
$$I_{yy} = \sum m_i(x_i^2 + z_i^2) \qquad I_{xz} = \sum m_i x_i z_i$$
$$I_{zz} = \sum m_i(x_i^2 + y_i^2) \qquad I_{yz} = \sum m_i y_i z_i$$

Doit être orienté de la même façon que le corps qu'il décrit :

$$I_R = RIR^{-1} = RIR^{\top}$$

avec R la matrice de rotation de l'objet

Simulation

État :

$$\mathbf{X}(t) = \begin{pmatrix} \mathbf{x}(t) \\ R(t) \\ \mathbf{P}(t) \\ \mathbf{L}(t) \end{pmatrix}$$

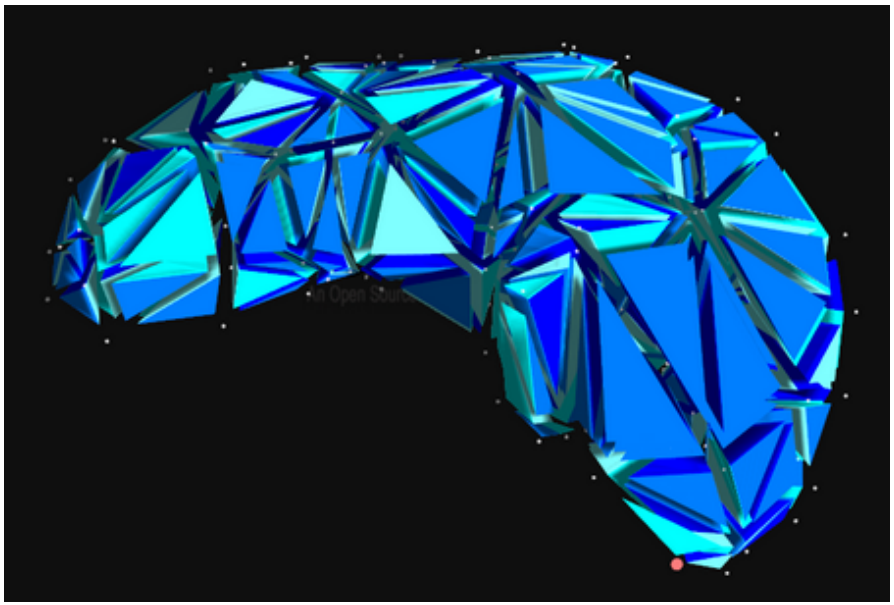
Dérivé :

$$\dot{\mathbf{X}}(t) = \begin{pmatrix} \mathbf{v}(t) \\ \omega(t) * R(t) \\ \mathbf{F}(t) \\ \tau(t) \end{pmatrix}$$

En pratique, il est préférable de représenter l'orientation par un **quaternion** $q(t)$ plutôt qu'une matrice $R(t)$

Objets déformables

- Réseaux de masses-ressorts
- Tiges : super-hélices / super-clothoïdes
- Éléments finis



[Faure et al., 2012]

[Casati et al., 2013]

Position Based Dynamics

Simulation unifiée des objets rigides, déformables et des fluides représentés par des **particules** dont les positions sont **contraintes** (égalités et inégalités).

1. **Prédiction** des positions : une itération d'Euler $\mathbf{x}_p \leftarrow \mathbf{x} + \mathbf{v}\Delta t$
2. **Résolution** des contraintes sur les positions (itérativement)
3. **Mise à jour** de l'état des particules : $\mathbf{v} \leftarrow (\mathbf{x}_p - \mathbf{x})/\Delta t$

$$\mathbf{x} \leftarrow \mathbf{x}_p$$

Bibliothèque C++ : NVIDIA FleX
https://developer.nvidia.com/flex

Code open-source:
https://github.com/InteractiveComputerGraphics/PositionBasedDynamics

Jan Bender, Matthias Müller and Miles Macklin, **A Survey on Position Based Dynamics, 2017**, In *Tutorial Proceedings of Eurographics*, 2017