Principe d'animation

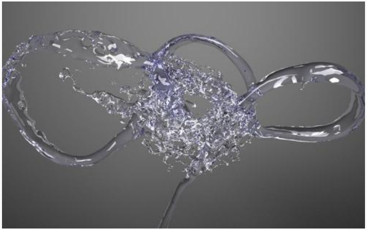
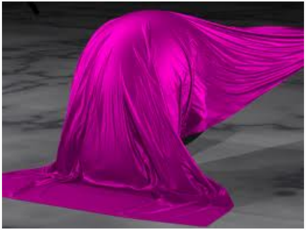
Systèmes de particules

Christian Gentil

PARV : systèmes de particules 2020-2021

Master 2 IIA - Université de Bourgogne



Systèmes d'équations différentielles ordinaires (EDO)

Système de particules

Système masse-ressorts

Systèmes d'équations différentielles ordinaires (EDO)

Formulation du problème initiale

Le principe est de décrire un système à partir d'une "loi" d'évolution.

$$\dot{x}(t) = f(x(t), t)$$

- $x(t)$ est la variable considérée dépendant du temps (ce peut être un vecteur : position x, y, z d'une particule),
- $\dot{x}(t)$ est la dérivée de cette variable en fonction du temps : **son évolution en fonction du temps**,
- $f(x(t), t)$ la loi d'évolution de $x(t)$.

problème
Connaissant une condition initiale x_0 pour t_0
on cherche la solution $x(t)$ $t \in]-\infty, +\infty[$**, telle que** $x(t_0) = x_0$.

Exemple

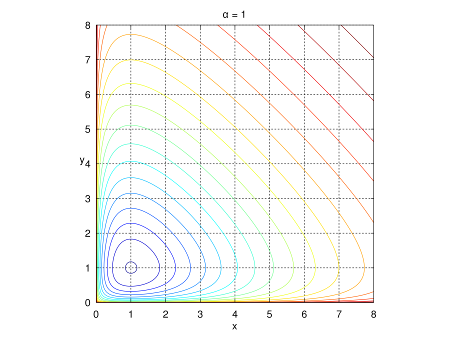
Système de deux espèces, proies et prédateurs, équations de Lotka-Volterra :

$$\begin{cases} \dot{x}(t) & = x(t)(A - B \times y(t)) \\ \dot{y}(t) & = y(t)(C \times x(t) - D) \end{cases}$$

- $A =$ taux de reproduction intrinsèque des proies (constant, indépendant du nombre de prédateurs),
- $B =$ taux de mortalité des proies dû aux prédateurs rencontrés,
- $C =$ taux de reproduction des prédateurs en fonction des proies rencontrées et mangées,
- $D =$ taux de mortalité intrinsèque des prédateurs (constant, indépendant du nombre de proies),

Exemple

La solution des équations de Lotka-Volterra



Exemple

Mécanique du point, loi de Newton : $m\vec{a} = \vec{f}$

$$\begin{cases} \dot{x}(t) & = v(t) \\ \dot{v}(t) & = \frac{\vec{f}}{m} \end{cases}$$

Calcul de la solution

Solution explicite

- exemple $\dot{x}(t) = -kx(t)$
- $x(t) = e^{-kt}$

Solution numérique

- On discrétise le temps : Δt,
- en partant de la condition initiale $x_0 = x(t_0)$, on utilise la fonction dérivée f pour calculer une approximation de $x(t + \Delta t)$

Méthode d 'Euler

la plus simple

$$x(t+h) = x + h\,\dot{x}(t)$$

Exemple : particule soumise à la gravité

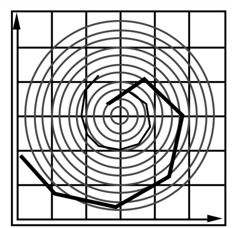
- $m\vec{a} = m\vec{g} \Rightarrow f(x(t), t) = \vec{g}$
- $x(t_0) = x_0,\ \dot{x}(t) = \vec{v}_0$
- $x(t + \triangle t) = x(t) + \vec{g} \triangle t$

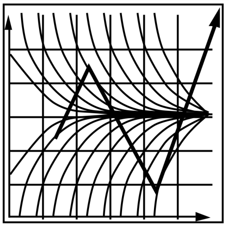
Ajout d'une force de résistance à l'avancement :

$f(x(t), t) = \vec{g} - k\dot{x}(t)$

Méthode d 'Euler

Mais elle introduit des problème de "dérive" ou d'instabilité numérique.





Justification = série de Taylor avec erreur en $O(h^2)$

$$x(t+h) = x + h\,(\vec{v} + \vec{g}\,t) + O(h^2)$$

(voir cours du SIGGRAPH)

Méthode du "Midpoint"

Pour avoir une erreur en $O(h^3)$ on doit considérer $\ddot{x}(t)$

Midpoint

$$x(t+h) = x + h\,\dot{x}(t) + \frac{h^2}{2}\,\ddot{x}(t)$$

Pour simplifier l'écriture de l'idée de la démonstration, on peut supposer que f dépend de t uniquement par le biais de $x(t)$ [1] : $\dot{x}(t) = f(x(t))$

$$\ddot{x}(t) = \frac{\partial f}{\partial x}\,\dot{x}(t) = f'f$$

[1] sans cette hypothèse le résultat est le identique

Méthode du "Midpoint"

Le calcul de f' est souvent compliqué et coûteux.

$\Rightarrow$ on calcule un approximation à partir d'une série de Taylor à l'ordre 2 :

$$f(x + \Delta x) = x + \Delta x \; f'(x) + O(h^2)^2$$

En choisissant $\Delta x = \frac{h}{2} f(x)$, on introduit $\ddot{x}$:

$$\begin{aligned} f(x + \frac{h}{2} f(x)) &= f(x) + \frac{h}{2} f(x) f'(x) + O(h^2) \\ &= f(x) + \frac{h}{2} \ddot{x}(x) + O(h^2) \end{aligned}$$

[2] Pour le cas général, ici il faudrait utiliser un développement de Taylor pour une fonction à deux variables

Méthode du "Midpoint"

Rappel:

$$f(x + \frac{h}{2}f(x)) = f(x) + \frac{h}{2}\ddot{x}(x) + O(h^2)$$

en arrangeant et multipliant pas h

$$h(f(x + \frac{h}{2}f(x)) - f(x)) = \frac{h^2}{2}\ddot{x}(x) + O(h^3)$$

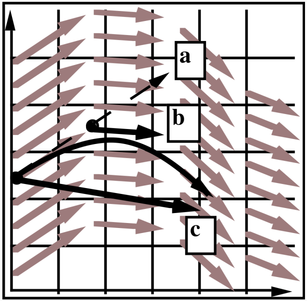
On replace dans notre série de Taylor initiale (à l'ordre 3) :

et c'est fini

$$x(t + h) = x + h \times f[x + \frac{h}{2}f(x)]$$

Méthode du "Midpoint"

Interprétation des calculs :$x(t+h) = x + h \times f[x + \frac{h}{2} f(x)]$

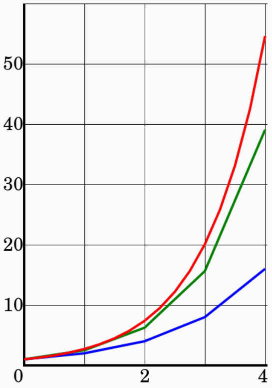


- a On calcule l'étape d'Euler :
  $\Delta x = \Delta t \; f(x,t)$
- b On évalue f au point milieu :
  $f_{mid} = f(x + \frac{\Delta x}{2}, t + \frac{\Delta t}{2})$
- c On avance d'un pas en utilisant la valeur du point milieu :
  $x(t+\Delta t) = x(t) + \Delta f_{mid}$

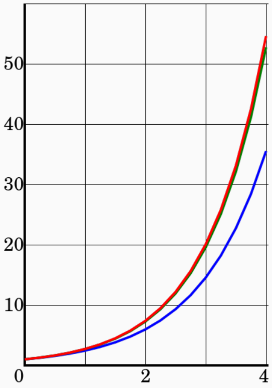
Erreur en O(h3) mais nécessite deux évaluations de f.

Méthode du "Midpoint"

Illustration $\dot{x}(t) = x(t)$: en rouge la solution $x(t) = e^t$, en bleu méthode d'Euler, en vert methode du "midpoint".



$h = 1$



$h = 0.25$

Méthode de Runge-Kutta

- Il est possible de poursuivre le principe du "midpoint" en évaluant plusieurs fois f pour éliminer des erreurs de plus haut degré.
- La méthode la plus populaire est celle de "Runge-Kutta" d'odre 4 donnant une erreur en $O(h^5)$.
- La méthode du "midpoint" pourrait être nommée "Runge-Kutta" d'odre 2.

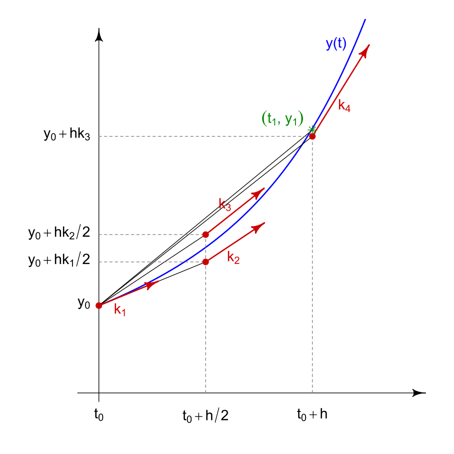
Méthode de Runge-Kutta

Directement au résultat :

$$
\begin{aligned}
k_1 &= h\ f(x, t) \\
k_2 &= h\ f(x + \frac{k_1}{2}, t + \frac{h}{2}) \\
k_3 &= h\ f(x + \frac{k_2}{2}, t + \frac{h}{2}) \\
k_4 &= h\ f(x + \frac{k_3}{2}, t + \frac{h}{2}) \\
x(t+h) &= x + \frac{1}{6}k_1 + \frac{1}{3}k_2 + \frac{1}{3}k_3 + \frac{1}{6}k_4
\end{aligned}
$$

Méthode de Runge-Kutta

Illustration de la méthode :



Pas adaptatif (fortement recommandé)

Quelle que soit la méthode :

- un pas élevé sera préférable car moins de calculs,
- mais si trop élevé :
  - il y aura une erreur d'approximation importante,
  - ou pire, introduira de l'instabilité

Principe du pas adaptatif :

- On estime l'erreur
- Si l'estimation est trop importante on réduit le pas pour réduire l'erreur,
- la réduction du pas tient compte de l'ordre de l'erreur.

Pas adaptatif : estimation de l'erreur

Exemple du cas d'une erreur en $O(h^2)$ (ex méthode d'Euler).
On calcule $x(t+h)$

- en utilisant un pas $= h \longrightarrow x_a$ première approximation
- en utilisant deux pas de $= \frac{h}{2} \longrightarrow x_b$ deuxième approximation

Les deux calculs donnant une erreur d'approximation en $O(h^2)$, l'écart entre ces deux approximation est en $O(h^2)$.

$e = |x_a - x_b|$

- est une **ESTIMATION** de l'erreur courante,
- facile à calculer,
- peut être prise en défaut,
- mais raisonnablement efficace
- $\Rightarrow$ **peut être utilisée pour adapter le pas**.

Pas adaptatif

Exemple :
Si on souhaite une erreur de 10^{-4} et que $e = 10^{-4}$, l'erreur étant en $O(h^2)$, on peut augmenté le pas de

$$\left(\frac{10^{-4}}{10^{-8}}\right)^{\frac{1}{2}} h = 100h$$

Si $e = 10^{-3}$ il faudra diminuer le pas de

$$\left(\frac{10^{-4}}{10^{-3}}\right)^{\frac{1}{2}} h = 0.316h$$

Remarque sur l'implémentation

Les méthodes de résolution EDO sont indépendantes de l'application.

Il est possible d'écrire un code générique quelle que soit l'application. Du point de vue du solveur le système sur lequel il opère est une boite noire $f(x(t), t)$ qu'il doit pouvoir évaluer.

Système de particules

Cas d'une seule particule

Deuxième loi de Newton : $m\vec{a} = \sum \vec{f} = \vec{F}$

ODE de degré 2 (de dimension 3) :

$$\vec{a} = \begin{pmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{pmatrix} = \frac{1}{m}\vec{F} \quad \text{avec } \sum \vec{f} = \vec{F}$$

Cas d'une seule particule

En posant : $X(t) = \begin{pmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{pmatrix}$, $\frac{\partial X(t)}{\partial t} = \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{pmatrix} = \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \frac{1}{m}F_x \\ \frac{1}{m}F_y \\ \frac{1}{m}F_z \end{pmatrix}$

ODE de degré 1 de dimension 6 :

$$\frac{\partial X(t)}{\partial t} = f(X(t))$$

On peut appliquer les méthodes d'Euler, du midpoint ou de Runge-Kutta.

Système de n particules

Il suffit de définir : $X(t) = \begin{pmatrix} x_0 \\ y_0 \\ z_0 \\ \dot{x}_0 \\ \dot{y}_0 \\ \dot{z}_0 \\ \vdots \\ x_{n-1} \\ y_{n-1} \\ z_{n-1} \\ \dot{x}_{n-1} \\ \dot{y}_{n-1} \\ \dot{z}_{n-1} \end{pmatrix}$, on a $\frac{\partial X(t)}{\partial t} = \begin{pmatrix} \dot{x}_0 \\ \dot{y}_0 \\ \dot{z}_0 \\ \frac{1}{m} F_x^0 \\ \frac{1}{m} F_y^0 \\ \frac{1}{m} F_z^0 \\ \vdots \\ \dot{x}_{n-1} \\ \dot{y}_{n-1} \\ \dot{z}_{n-1} \\ \frac{1}{m} F_x^{n-1} \\ \frac{1}{m} F_y^{n-1} \\ \frac{1}{m} F_z^{n-1} \end{pmatrix}$

avec F^i = somme des forces appliquées à la i^{eme} particule. et appliquer l'une des méthodes :Euler, midpoint ou de Runge-Kutta.

Système de n particules

Simulation = comme expliqué diapos précédentes

Exemple Euler :

$$X(t+\triangle t) = X(t) + \dot{X}(t) \times \triangle t$$

- On choisit un pas $\triangle t$ (il peut être adapté)
- On calcule $\dot{X}(t)$
  - pour les vitesses : elles sont dans déjà dans $X(t) \Rightarrow$ on les recopie,
  - les accélérations : on évalue les forces en fonction des position et vitesset
- Il faut partir d'une condition initiale : $t = t_0$
  - i.e. choix de la configuration de départ,
  - = choix des positions et vitesse initiale de chaque particule.

Implémentation

Structure de donnée :

```
typedef struct{
        float m;        //mass
        vec3 x;         // position vector
        vec3 v;         //velocity vector
        vec3 f; //force accumulator
} *Particle;
```

```
typedef struct{
Particle *p;    // array of pointers to particles
int n;          // number of particles
float t;        // simulation clock
} *ParticleSystem;
```

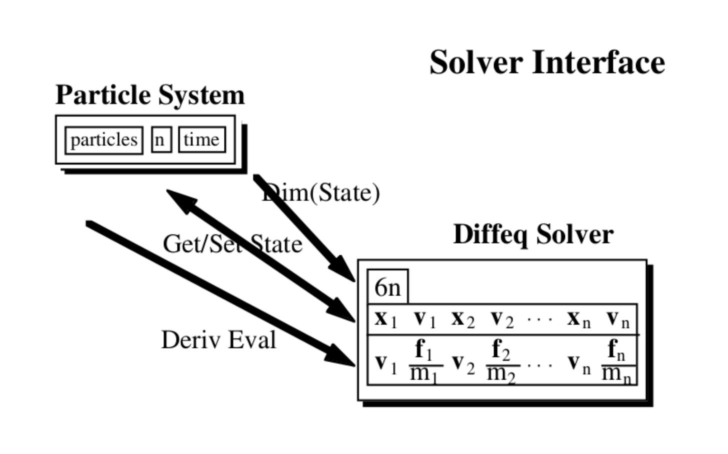
Implémentation

Structure de donnée pour le solveur:

```
typedef struct{
  vec3 x;          // position  vector
  vec3 v;          //velocity  vector
} *Xt ;
```

```
typedef struct{
  vec3 xDot;       // position derivative vector
  vec3 vDot;       //velocity derivative vector
} *XtDot ;
```

Le solveur


Solver Interface
Particle System
particles
n
time
Dim(State)
Get/Set State
Deriv Eval
Diffeq Solver
6n
$\mathbf{x}_1$ $\mathbf{v}_1$ $\mathbf{x}_2$ $\mathbf{v}_2$ $\cdots$ $\mathbf{x}_n$ $\mathbf{v}_n$
$\mathbf{v}_1$ $\frac{\mathbf{f}_1}{m_1}$ $\mathbf{v}_2$ $\frac{\mathbf{f}_2}{m_2}$ $\cdots$ $\mathbf{v}_n$ $\frac{\mathbf{f}_n}{m_n}$

Implémentation

- initialisation du système de particules(pos, vit.forces)
- lancer l'animation qui appelle le solveur

```
void EulerStep(ParticleSystem p, float DeltaT){
Xt = ParticleGetState(p);        // get state
XtDot = ParticleDeriv(p);        // get deriv
XtDeltat = Xt + DeltaT * XtDot // X(t+Deltat)
// MAJ particules  : pos. vitesses, forces and
    temps
ParticleSetState(p,XtDeltat,DeltaT)
}
```

Les forces

Il ne reste plus qu'a déterminer les forces agissant sur les particules.
Les principaux types de forces sont :

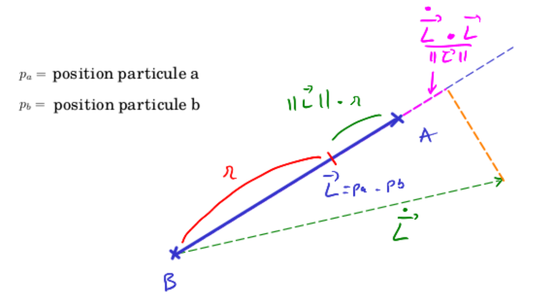
- force unaires, force constantes exemple gravité
  $f = m\vec{g}$
- forces de viscosité, s'opposant au déplacement
  $\vec{f} = -k_d \vec{v}$
- force n-aires, interaction entre particules (voir diapo suivantes).

Exemple d'interaction entre particules : loi d'élasticité de Hooke

$$f_a = -\left[k_s(||\vec{L}|| - r) + k_d \frac{\dot{\vec{L}} \bullet \vec{L}}{||\vec{L}||}\right] \frac{\vec{L}}{||\vec{L}||}$$

$\vec{L} = p_a - p_b$, $\dot{\vec{L}} = \vec{v_a} - \vec{v_b}$, et $f_b = -f_a$

k_s = constante d'élasticité, k_d = constante de viscosité.



Système masse-ressorts

Système de particules vs système masse-ressorts

Système de particules

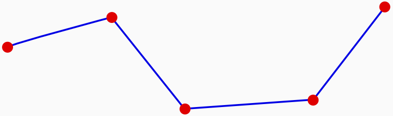
- pas de structure topologique
- chaque particule interagie avec toutes les autres
- générateur de particule
- durée de vie limitée
- simulation de poussière, fluide, fumée, étincelle, flammes,...

Système masse-ressorts

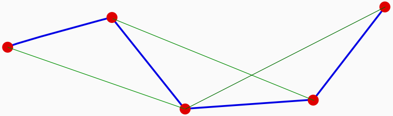
- particule = masse
- une structure topologique (réseau 2D ou 3)
- simulation de tissus, habits, peau, objets mous, semi-rigides
- problème de si objet trop rigide : divergence, temps de calculs importants

Système masse-ressorts

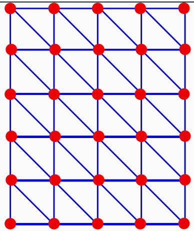
Exemple de structure topologique



Chaîne



Cheveux



Tissus

Autre que l'on ne verra pas

Géstion des collisions

- détection de la collision,
- traitement de la collision.

Modélisation énergétique

- Plus simple que modélisation masse-ressorts
- Fonction énergétique générale (s'applique à tout le modèle)
- Potentiel lié à cette énergie
- Force dérivant du potentiel

Autre que l'on ne verra pas

Contraintes

- Restriction sur la position d'un objet
- Contact, non-pénétration
- Articulations
- Limites aux articulations

Animation de solides

- mécanique du point (particules) vs mécanique du solide (solides)