Simulation Numérique 3 :

Problème dynamique à une dimension

Le but de ce chapitre est d'étudier la résolution d'un problème dynamique à une dimension, linéaire ou non, conduisant à la résolution approchée d'une équation différentielle ordinaire par la méthode d'Euler.

En d'autre terme, on veux trouver une solution à une équation différentielle de la forme

$$\begin{cases} y'(t) = F(t, y(t)) \\ y(t_0) = y_0 \end{cases} \tag{1}$$

avec condition initiale.

Rappelons que la plupart de nombreux problèmes de sciences physiques, de chimie, d'optimisation conduisent à des résolutions numériques [1] d'équations différentielles pas nécessairement linéaires.
L'algorithme au programme est la méthode d'Euler.

I MÉTHODE D'EULER

1 Motivation

Un théorème mathématique [2] assure que sous certaines conditions, l'équation (1) admet une unique solution.
Mais la théorie mathématique ne permet pas de trouver cette solution en général.

Par exemple, pour l'équation d'un pendule de la forme

$$\ddot{\theta} = -k\dot{\theta} - \alpha \sin\theta,$$

1. Par exemple parce qu'on ne sait pas résoudre de manière exacte !
2. Théorème de Cauchy-Lipschitz

on doit faire une hypothèse d'*angle relativement petit* pour écrire

$$\sin\theta \approx \theta$$

puis

$$\ddot{\theta} = -k\dot{\theta} - \alpha\theta$$

équation différentielle linéaire à cœfficients constants dont on sait déterminer les solutions exactes.

Mais les hypothèses de linéarisation peuvent être trop réductrices voire inexistantes, d'où la nécessité de pouvoir résoudre numériquement les équations différentielles.

2 Principe

On cherche à calculer numériquement la valeur approchée d'une solution du problème (1) sur un segment $[a,b]$ avec $a = t_0$.

On crée alors une **subdivision**

$$t_0 = a < t_1 < t_2 < \cdots < t_n = b$$

du segment $[a,b]$ que l'on peut supposer **régulière** pour simplifier, c'est-à-dire telle que le pas $t_{k+1} - t_k = h$ soit indépendant de k. Alors

$$h = \frac{b-a}{n}, \quad t_k = a + k\frac{b-a}{n} \quad \text{et} \quad \boxed{t_{k+1} = t_k + h}$$

On va calculer une valeur approchée de proche en proche $y_k \approx y(t_k)$ partant de la condition initiale $\boxed{y_0 = y(a)}$ (valeur exacte ici) puis on tire y_{k+1} à partir de y_k grâce à la relation

$$y'(t_k) = F(t_k, y(t_k)) \approx F(t_k, y_k)$$

avec [1] $y'(t_k) \approx \dfrac{y_{k+1} - y_k}{t_{k+1} - t_k} = \dfrac{y_{k+1} - y_k}{h}$, ce qui donne

$$\boxed{y_{k+1} = y_k + hF(t_k, y_k)}$$

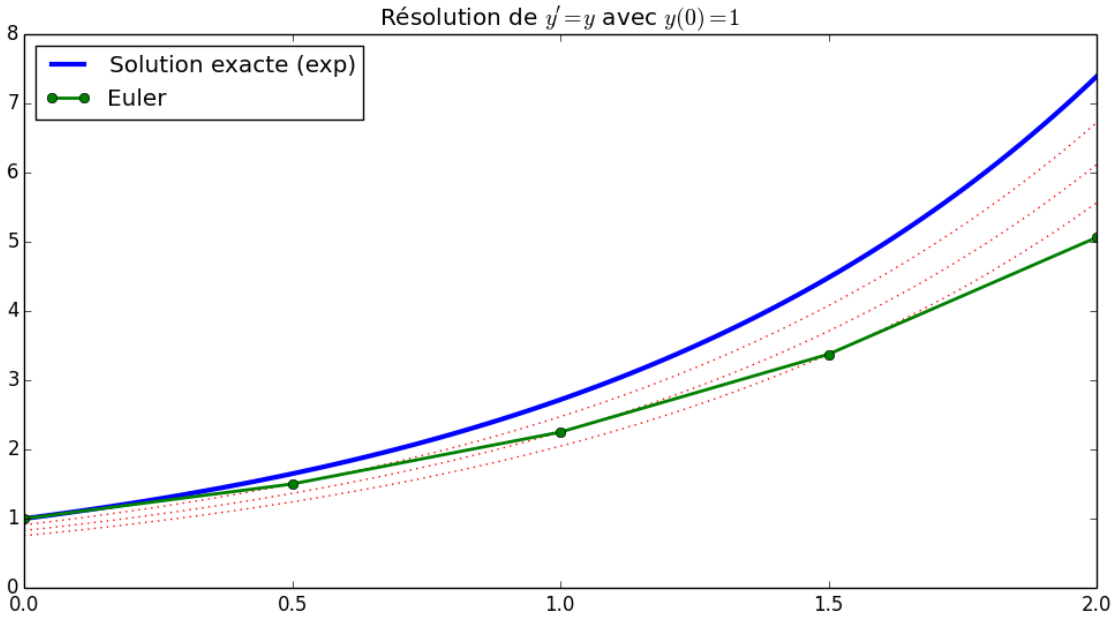
Ce schéma numérique est appelé **méthode d'Euler explicite**.

Autre approche possible :

$$y(t_{k+1}) - y(t_k) = \int_{t_k}^{t_{k+1}} y'(t)\,\mathrm{d}t = \int_{t_k}^{t_{k+1}} F(t, y(t))\,\mathrm{d}t \approx hF(t_k, y(t_k))$$

(rectangles à gauche).

1. On remplace localement y par sa tangente.



Les pointillés représentent les solutions exactes issues des nouvelles valeurs initiales.

3 Algorithme

Version tableau :

Version liste :

Euler (F, a, b, y_0, h)

- $y \leftarrow y_0$
- $Y \leftarrow [y_0]$
- $t \leftarrow a$
- $T \leftarrow [a]$
- **Tant que** $t + h \leqslant b$ **Faire**
  - $y \leftarrow y + hF(t, y)$
  - $t \leftarrow t + h$
  - Ajout de y au bout de Y
  - Ajout de t au bout de T
- **FinTq**
- **Retourner** (T, Y)

Invariant : À la fin de la k^{e} étape, t contient t_k et y contient y_k.

On pourra écrira d'autres versions de l'algorithme, avec un tableau et/ou en prenant n en argument.

DONNER UN EXEMPLE EN DIMENSION 1.

4 Équations différentielles d'ordre supérieur

Pour résoudre des équations différentielles d'ordre $p \geqslant 2$, l'idée est de se ramener à une équation vectorielle à l'ordre 1 en posant $Y = \begin{pmatrix} y \\ y' \\ \vdots \\ y^{(p-1)} \end{pmatrix}$ et en écrivant $X' = F(t, X)$.

Par exemple, pour l'équation du pendule $\ddot{\theta} = -k\dot{\theta} - \alpha \sin\theta$, en posant $X = \begin{pmatrix} \theta \\ \dot{\theta} \end{pmatrix}$, on a $\dot{Y} = F(t, Y)$, avec $F\left(t, \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}\right) = \begin{pmatrix} x_2 \\ -kx_2 - \alpha \sin x_1 \end{pmatrix}$.

A DETAILLER PLUS

La méthode d'Euler s'écrit alors exactement de la même manière mais...

⚠ Si on stocke les vecteurs sous forme de liste en Python, la somme `X=X+h*F(t,X)` est une concaténation (le `h*` aussi !) Utiliser des tableaux NumPy permet de résoudre ce petit problème d'écriture.

5 Stabilité, consistance, convergence

Les valeurs approchées trouvées avec la méthode d'Euler dépendent évidemment du pas $h = \dfrac{b-a}{n} \underset{n\to+\infty}{\longrightarrow} 0$.

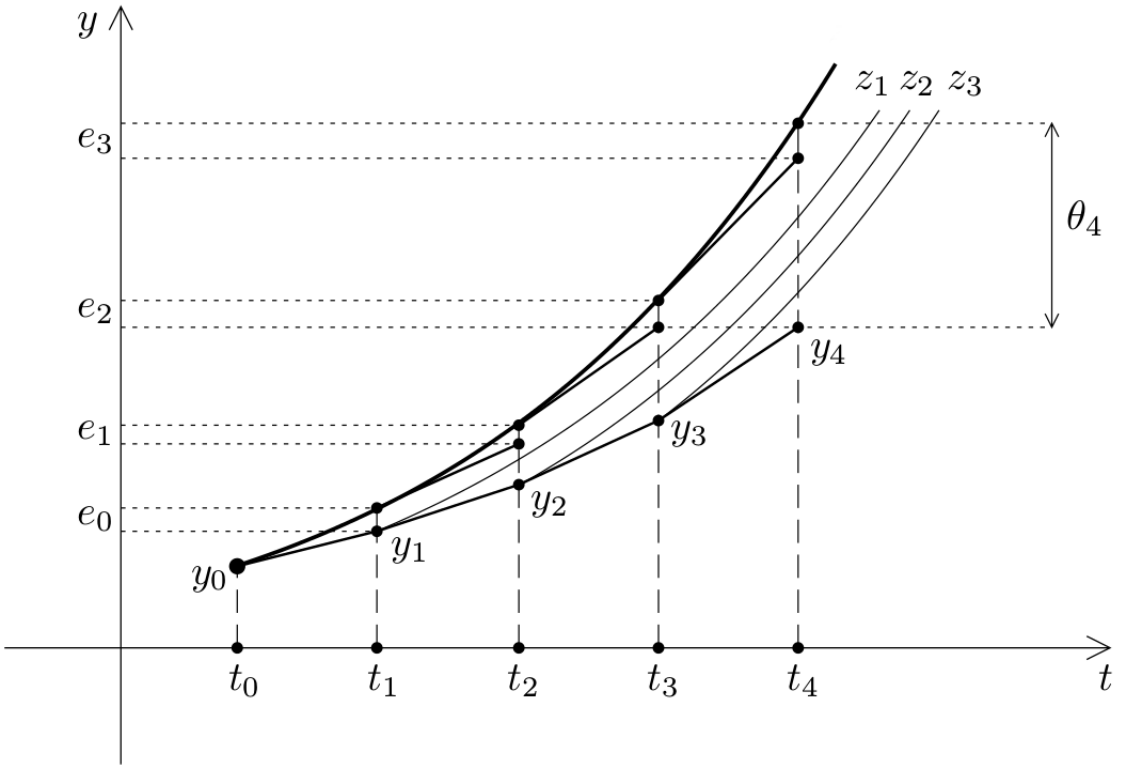
Ce que l'on attend naturellement de notre schéma numérique, c'est qu'il soit **convergent**, c'est-à-dire que :

$$\theta_n = \theta(h) = \max_k \left|y(t_k) - y_k\right| \underset{h\to 0}{\longrightarrow} 0$$

où $\max_k \left|y(t_k) - y_k\right|$ est appelée **erreur globale**.

Une étude mathématique montre que la convergence sera vérifiée lorsque le schéma est :

- **stable** : une petite erreur initiale et des petites erreurs d'arrondi provoquent une erreur finale contrôlable.
- **consistant** : l'erreur de consistance relative pour passer de t_k à t_{k+1} est l'écart e_k entre $y(t_{k+1})$ et $\tilde{y}_{k+1}$ où y est la solution exacte et $\tilde{y}_{k+1}$ la valeur obtenue en appliquant une fois la méthode à partir de $(t_k, y(t_k))$. Autrement dit, $e_k = y(t_{k+1}) - \big(y(t_k) + hF(t_k, y(t_k))\big)$.
  L'erreur de consistance globale est $E(h) = \sum_k |e_k|$.
  Le schéma est consistant si $E(h) = \sum_k |e_k| \underset{h\to 0}{\longrightarrow} 0$.



(Les $t \mapsto z_k(t)$ sont les solutions exactes issues des y_k.)
C'est bien le cas pour la méthode d'Euler.

6 Ordre

Une méthode numérique à un pas est dite d'ordre au moins p lorsque une erreur de consistance globale E est telle que $|E(h)| \leqslant Ch^p$.

On peut alors démontrer que l'erreur globale vérifie

$$\max_k \left|y(t_k) - y_k\right| \leqslant K\left|y_0 - y(t_0)\right| + K'h^p$$

avec, en général, $\left|y_0 - y(t_0)\right|$ négligeable (voire nul).

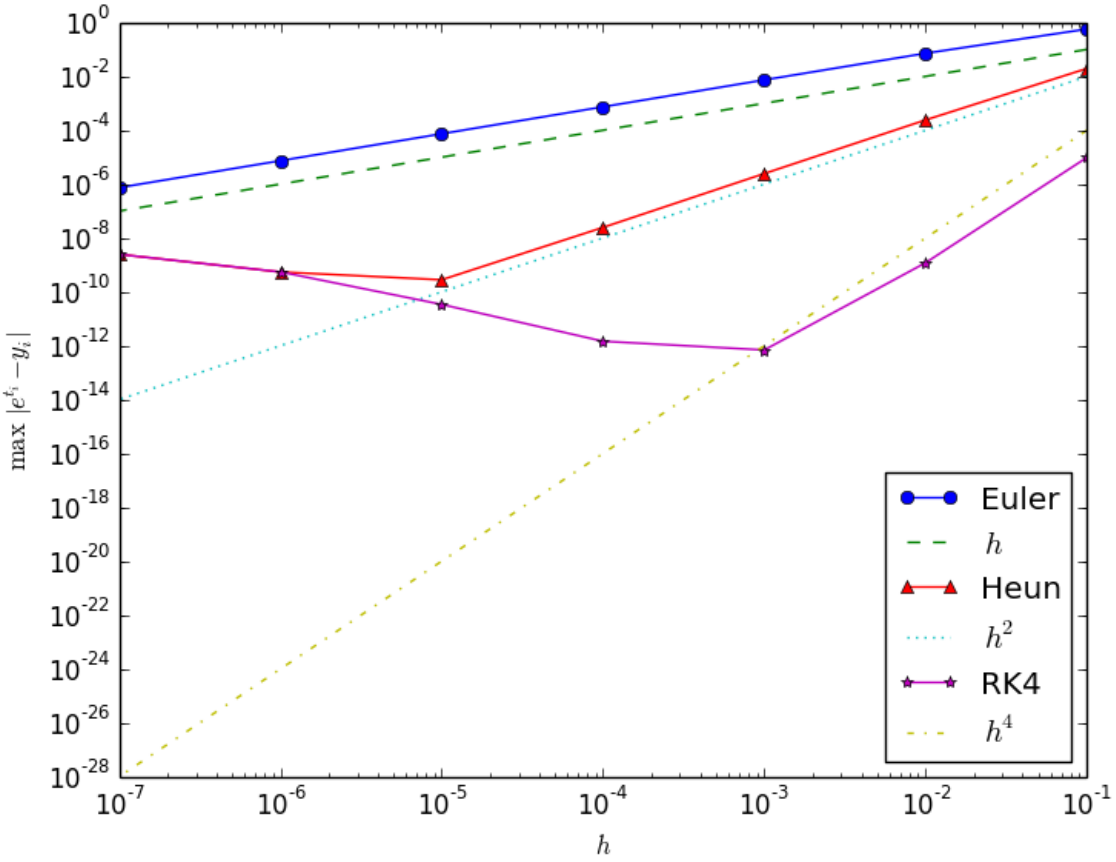
La méthode d'Euler est d'ordre 1.

On retiendra que l'erreur globale est en h donc en $\frac{1}{n}$.

Dans le graphe suivant, on représente en échelle logarithmique l'erreur globale en fonction de h pour la méthode d'Euler ainsi que pour deux autres méthodes classiques : Heun (ordre 2) et Runge-Kutta 4 (ordre ...).

Bien noter le décrochage lorsque le pas devient trop petit : les erreurs d'arrondi perturbent les calculs.

Dans la pratique, on est confronté au choix du pas h ie du nombre d'étapes n :



- Un pas trop petit implique un temps de calcul élevé, et un risque de propagation d'erreurs d'arrondi plus grand,
- Un pas trop grand donne un résultat trop imprécis.

Voici un tableau avec un ordre de grandeur du temps d'exécution (en secondes) et de l'erreur globale commise en fonction du nombre n d'étapes pour ces 3 méthodes obtenu sur le même exemple que précédemment :

n	10^2	10^3	10^4	10^5	10^6	10^7
temps (Euler)	$5\cdot10^{-4}$	$4\cdot10^{-3}$	10^{-2}	10^{-1}	1	11
erreur (Euler)	$7\cdot10^{-2}$	$7\cdot10^{-3}$	$7\cdot10^{-4}$	$7\cdot10^{-5}$	$7\cdot10^{-6}$	$7\cdot10^{-7}$
temps (Heun)	$8\cdot10^{-4}$	$2\cdot10^{-3}$	$2\cdot10^{-2}$	0,2	2	50
erreur (Heun)	$2\cdot10^{-4}$	$2\cdot10^{-6}$	$2\cdot10^{-8}$	$2\cdot10^{-10}$	$5\cdot10^{-10}$	$2\cdot10^{-9}$
temps (RK4)	10^{-3}	$4\cdot10^{-3}$	$4\cdot10^{-2}$	0,4	4	40
erreur (RK4)	10^{-9}	$7\cdot10^{-13}$	10^{-12}	$3\cdot10^{-11}$	$5\cdot10^{-10}$	$2\cdot10^{-9}$

II D'AUTRES SCHÉMAS NUMÉRIQUES À UN PAS (HP)

Revenons quelques instants à la méthode d'Euler. Pour résoudre l'équation $y' = F(t, y)$, on a pu écrire

$$y_{k+1} - y_k \approx y(t_{k+1}) - y(t_k) = \int_{t_k}^{t_{k+1}} y'(t)\,\mathrm{d}t = \int_{t_k}^{t_{k+1}} F(t, y(t))\,\mathrm{d}t \approx hF(t_k, y(t_k))$$

en utilisant une approximation de l'intégrale par le rectangle à gauche.

On peut alors adapter les différentes formules de quadrature pour former de nouveaux schémas numériques :

- Point milieu :
$$y_{k+1} = y_k + hF\left(t_k + \frac{h}{2}, y_k + \frac{h}{2}F(t_k, y_k)\right)$$

- Trapèzes : **méthode de Heun**
$$y_{k+1} = y_k + h \cdot \frac{F(t_k, y_k) + F\left(t_{k+1}, y_k + hF(t_k, y_k)\right)}{2}$$

- Mélange de rectangles à gauche, à droite, point milieu et Simpson : **Runge-Kutta 4**
$$\begin{aligned}
y_k^{(1)} &= y_k + \frac{h}{2}F(t_k, y_k)\\
y_k^{(2)} &= y_k + \frac{h}{2}F\left(t_k + \frac{h}{2}, y_k^{(1)}\right)\\
y_k^{(3)} &= y_k + hF\left(t_k + \frac{h}{2}, y_k^{(2)}\right)\\
y_{k+1} &= y_k + \frac{h}{6}\left(F(t_k, y_k) + 2F\left(t_k + \frac{h}{2}, y_k^{(1)}\right) + 2F\left(t_k + \frac{h}{2}, y_k^{(2)}\right) + F\left(t_{k+1}, y_k^{(3)}\right)\right)
\end{aligned}$$

À noter qu'il existe une autre **méthode d'Euler**, dite **implicite**. Elle est d'ordre 1 aussi mais présente plus de stabilité que la méthode d'Euler explicite. La formule devient

$$y_{k+1} = y_k + hf(t_{k+1}, y_{k+1})$$

c'est une équation en y_{k+1} qu'il faut résoudre numériquement (avec la méthode de Newton, par exemple.)