

ÉQUATIONS DIFFÉRENTIELLES. MÉTHODE D'EULER.

1. Introduction

Beaucoup de phénomènes physiques (étude d'une trajectoire, circuit RLC...) ou chimiques (évolution des concentrations lors d'une réaction...) se modélisent à l'aide d'équations différentielles.

Cependant, hormis quelques cas simples (comme les équations différentielles linéaires d'ordre 1), on sait rarement en donner les solutions exactes.

L'objet de ce chapitre est d'étudier la méthode d'Euler qui permet d'obtenir une approximation des solutions.

Remarque. On ne s'attardera pas ici sur l'existence des solutions.

Mathématiquement, si on recherche la solution soumise à condition initiale en a , le théorème de Cauchy (vu en cours de maths dans le cadre des EDL), sous des conditions raisonnables et au moins au voisinage de a , en assure l'existence et l'unicité.

Physiquement ou chimiquement, cette existence est constatée par l'expérience !

2. Présentation de la méthode

On considère une équation différentielle du 1er ordre de la forme :

$$y' = F(t, y)$$

où

- F est une application continue de $D \times [a, b] \rightarrow \mathbb{R}$.
- t est la variable (le temps).
- $y : t \mapsto y(t)$ fonction inconnue de classe $\mathcal{C}^1$

Ainsi, y est solution de $\begin{cases} y' = F(t, y) \\ y(a) = y_0 \end{cases}$ si et seulement si $\begin{cases} \forall t \in [a, b], y'(t) = F(t, y(t)) \\ y(a) = y_0 \end{cases}$

Pour trouver une expression approchée de y , on subdivise l'intervalle $[a, b]$ en $n \in \mathbb{N}^*$ intervalles de même longueur $h = \frac{b-a}{n}$ appelée le pas. On propose alors une valeur approchée y_k de chaque $y(t_k)$ où $t_k = a + kh$ en utilisant l'approximation suivante :

considérant h petit, on approche $y'(t_k)$ par $\frac{y(t_{k+1}) - y(t_k)}{h}$

ce qui permet d'écrire que $y(t_{k+1}) \simeq y(t_k) + hy'(t_k)$ soit $y(t_{k+1}) \simeq y(t_k) + hF(t_k, y(t_k))$

Illustration

On peut ainsi proposer une construction de proche en proche des y_k :

$$\begin{cases} y_0 \in \mathbb{R} \text{ (condition initiale)} \\ \forall k \in \{0, \dots, n-1\}, y_{k+1} = y_k + hF(t_k, y_k) \end{cases}$$

La fonction, affine sur chaque intervalle $[t_k, t_{k+1}]$ et de valeurs y_k et y_{k+1} aux extrémités, est une fonction approchant la solution y cherchée.

Exercice 1 Regardons sur un exemple simple l'application de cette méthode.

On cherche à approcher la solution $y : [0, 1] \rightarrow \mathbb{R}$ de l'équation différentielle $y' = y$ soumise à la condition initiale $y(0) = 1$.

1. Représenter graphiquement y . On prendra une unité de 6cm sur l'axe des abscisses et de 2cm sur les ordonnées.
2. Appliquer la méthode avec le pas $h = \frac{1}{3}$ et représenter graphiquement la fonction affine par morceaux approchant la solution.

3. Programmation

3.1 Programme Python

Pour trouver une valeur approchée de la solution de l'équation $y' = F(t, y)$, $y(a) = y_0$ sur $[a, b]$, la fonction `Euler` prend en entrée les paramètres F (fonction à 2 variables), a et b les bornes de l'intervalle, y_0 la condition initiale et h le pas.

Exercice 2

Compléter la fonction :

```
def Euler(F,a,b,y0,h):
    """ valeur approchée de la solution de y'=F(t,y)
        )
        avec y(a)=y0 """
    t=a
    y=y0
    les_t=[a]
    les_y=[y0]
    while

    return les_t, les_y
```

La fonction renvoie la liste des abscisses et des ordonnées des points à relier pour obtenir le graphe de la valeur approchée.

3.2 Choix du pas

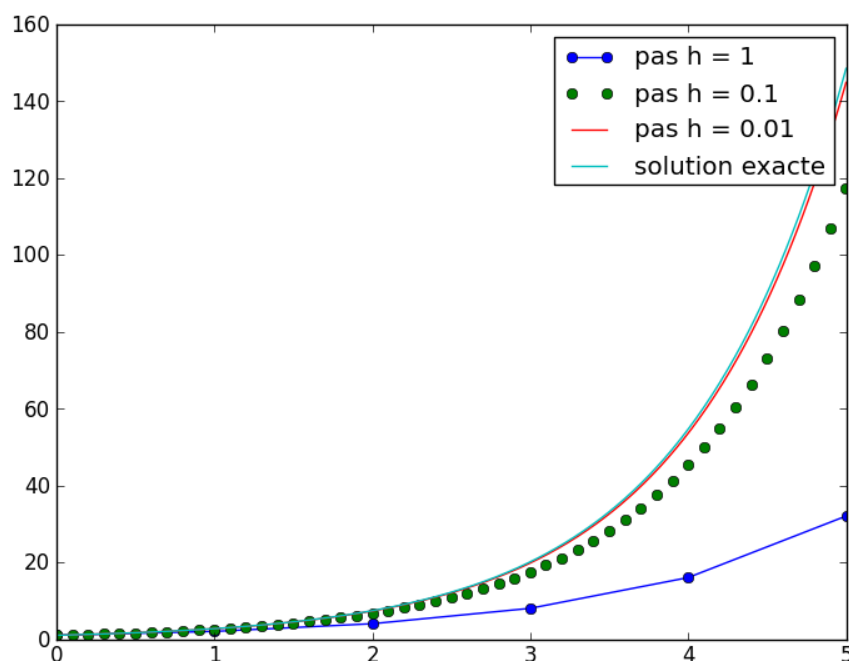
Exercice 3

Evaluer la complexité du programme `Euler` en fonction de h .

La complexité est donc en $O(\dots)$.

Ainsi, plus le pas est petit, plus le temps d'exécution est élevé.

Mais, par ailleurs, plus le pas est petit, meilleure est l'approximation : il faudra donc trouver un compromis entre les deux contraintes.



4. Utilisation de la bibliothèque `scipy`

On sait donc programmer par nous-mêmes et assez facilement la résolution approchée d'un système dynamique à une dimension.

Mais, pour gagner en fiabilité, on utilise en général des fonctions fournies dans des bibliothèques dédiées.

`odeint` est contenu dans la bibliothèque `scipy.integrate`.

L'utilisation de base est `odeint(F,y0,t)`. La valeur renvoyée est un tableau contenant une estimation de la solution de $y' = F(y(t), t)$ (condition initiale $y(a) = y0$) aux différents temps du tableau $t = [a, t_1, t_2, \dots, b]$

Pour créer le tableau des temps, on peut :

- le créer à la main par compréhension.
- utiliser `linspace`.

Enfin, en utilisant `plot` et `show`, on peut obtenir les graphes des solutions approchées. Nous mettrons cela en oeuvre en TP.