# Équations différentielles : Euler

## 1 Méthode d'Euler

### 1.1 Cadre de l'étude et résultats théoriques

**Théorème de Cauchy-Lipschitz** : Sous des conditions raisonnables sur F (qui relèvent du cours de Mathématiques) il existe une unique fonction y de classe $C^1$ sur [a,b] qui prenne une valeur donnée en a et qui vérifie l'équation :

$$y'(t) = F(t, y(t)) \quad (E)$$

On ne connaît malheureusement pas toujours la solution exacte d'une équation différentielle ; une solution approchée est par contre souvent suffisante.

### 1.2 Principe de la méthode

Nous allons chercher une solution approchée de $(E)$. Nous allons nous contenter de trouver une valeur approchée de la solution en un certain nombre de points ; il sera possible éventuellement d'approcher la fonction trouvée entre deux points consécutifs par une fonction simple comme une fonction affine. Plus exactement nous allons découper l'intervalle [a,b] de résolution en sous-intervalle de même longueur, on pose :

$$t_k = a + k\frac{b-a}{n} = a + k \times h \text{ et on cherche les } y_k = y(t_k)$$

Nous avons discrétisé l'intervalle [a,b] ; h s'appelle *le pas d'intégration.*
On a alors :

$$y_{k+1} - y_k = y(t_{k+1}) - y(t_k) = \int_{t_k}^{t_{k+1}} f'(u)du = \int_{t_k}^{t_{k+1}} F(u, y(u))du$$

La méthode d'Euler consiste à approximer cette dernière intégrale en assimilant la fonction F à une fonction constante sur l'intervalle d'intégration :

$$y_{k+1} - y_k = \int_{t_k}^{t_{k+1}} F(u, y(u))du \approx \int_{t_k}^{t_{k+1}} F(t_k, y(t_k))du = hF(t_k; y_k)$$

La méthode d'Euler s'écrit alors :

$$y_{k+1} = y_k + hF(t_k; y_k)$$

### 1.3 Exemples

*Exercice 1* : Résoudre l'équation différentielle y'=2y et y(0)=1 avec la méthode d'Euler ; on comparera les résultats obtenus par la méthode avec la solution exacte en t=1 (on admet que l'erreur y est la plus grande) pour n=10 et n=100

*Exercice 2* : Écrire les relations de récurrence qui permettent d'implémenter la méthode d'Euler avec les équations différentielles suivantes :

$y' = \sin y$

$y' = \sqrt{ty}$

## 1.4 Convergence de la méthode

**Convergence** : on dit que la méthode est convergente si :

$$\lim_{h \to 0} \max \{|y(t_i) - y_i|, i \in [\![1, n]\!]\} = 0$$

La méthode d'Euler est bien convergente.

On appelle **erreur de consistance** la quantité :

$$e(h) = \sum_{i=0}^{i=n-1} |y(t_{i+1}) - (y(t_i) + hF(t_i, y(t_i)))|$$

Une méthode est dite d'ordre p si il existe une constante K>0 telle que $e(h) < Kh^p$

La méthode d'Euler est d'ordre 1. Il existe des méthodes d'ordre supérieur : par exemple la méthode de Runge-Kutta est d'ordre 4.

Lorsque la solution exacte est connue, il est possible d'évaluer commise par :

$$E(h) = \max \{|y(t_i) - y_i|, i \in [\![1, n]\!]\}$$

*Exercice* : Évaluer E(h) dans l'exemple de l'équation y'=2y et y(0)=1 sur [0 ;1]

## 1.5 Choix du pas

Voici sur l'exemple de l'équation $y' = y$ et $y(0) = 1$ sur [0 ;1] un tableau des temps de calcul et des erreurs de la méthode pour différents h :

| h | $10^{-2}$ | $10^{-3}$ | $10^{-4}$ | $10^{-5}$ | $10^{-6}$ | $10^{-7}$ |
|---|---|---|---|---|---|---|
| temps(s) | $2.5\ 10^{-4}$ | $2.4\ 10^{-3}$ | $2.7\ 10^{-2}$ | 0.3 | 3.3 | 33 |
| erreur | $1.36\ 10^{-2}$ | $1.36\ 10^{-3}$ | $1.36\ 10^{-4}$ | $1.36\ 10^{-5}$ | $1.36\ 10^{-6}$ | $1.36\ 10^{-7}$ |

Le temps de calcul qui est en principe linéaire (la complexité de l'algorithme étant linéaire) augmente légèrement plus rapidement : les problèmes de gestion mémoire pour des listes ayant un grand nombre d'éléments apparaissent lorsque h devient très faible. L'erreur est divisée par 10 lorsque le pas est divisé par 10, ce qui rend bien compte de l'ordre 1 de la méthode d'Euler.

En résumé :

- Pas petit = temps de calcul élevé
- Pas grand = erreur de consistance importante

# 2 Mise en œuvre de la méthode

## 2.1 Équations scalaires d'ordre 1

*Exercice :* Quels sont les différents prototypes possibles pour implémenter la méthode d'Euler ?

Nous allons utiliser des listes que nous concaténerons au fur et à mesure. Dans le détail :

## 2.2 Équations scalaires d'ordre 2

L'idée pour appliquer la méthode d'Euler à une équation différentielle d'ordre 2 est de transformer cette équation différentielle (dont l'inconnue est une fonction à variables réelles) en une équation différentielle d'ordre 1 qui porte cette fois-ci sur une équation différentielle dont l'inconnue est cette fois-ci une fonction à variables vectorielles (de $\mathbb{R}$ dans $\mathbb{R}^2$).

On admet qu'une fonction à valeurs vectorielles se dérive composante par composante :

si $Y = \begin{bmatrix} f_1(t) \\ f_2(t) \end{bmatrix}$ on a $Y' = \begin{bmatrix} f_1'(t) \\ f_2'(t) \end{bmatrix}$

L'idée de la transformation est de travailler sur un vecteur Y qui est composé de la dérivée de y et de y. Sur un exemple : $y'' + ky = 0$ avec $y(0) = 1$ et $y'(0) = 0$ on pose $Y = \begin{bmatrix} y_1(t) = y'(t) \\ y_2(t) = y(t) \end{bmatrix}$.

On a alors $Y' = \begin{bmatrix} y_1'(t) = y''(t) \\ y_2'(t) = y'(t) \end{bmatrix}$ or $y'' = -ky$ donc l'équation différentielle peut aussi s'écrire :

$$Y'(t) = \begin{bmatrix} y_1'(t) \\ y_2'(t) \end{bmatrix} = \begin{bmatrix} -ky_2(t) \\ y_1(t) \end{bmatrix} = F(t, Y) \text{ avec } F\left(t, \begin{bmatrix} \alpha \\ \beta \end{bmatrix}\right) = \begin{bmatrix} -k\beta \\ \alpha \end{bmatrix}$$

La méthode d'Euler s'écrit alors : $Y_{k+1} = Y_k + hF(t_k, Y_k)$.

Pour chaque k, on va calculer un couple de valeurs $\begin{bmatrix} y_k' \\ y_k \end{bmatrix}$ dont on se servira pour l'étape k+1. On utilisera le type **array** qui est compatible avec l'addition.

*Exercice* : Écrire la méthode d'Euler pour l'équation différentielle $y''(t) - 4y'(t) + \sin y(t) = 0$

*Exercice* : Quelles sont les modification à apporter au code en Python de la méthode d'Euler adaptée à une équation différentielle scalaire d'ordre 2 ; on codera la fonction F correspondant à l'exemple précédent.

# 3 Utilisation des fonctions issues des bibliothèques

## 3.1 Présentation des libraires

Pour résoudre des équations différentielles on utilise la bibliothèque integrate du module scipy, importée par exemple avec :

import scipy.integrate as spi

La fonction utilisée est alors spi.odeint(f,y0,t)

Les arguments de cette fonction sont :

- La fonction f définie par y'=f(y,t) **Attention à l'ordre des variables ! ! !**
- y0 la condition initiale
- t un tableau de valeurs qui correspond aux $t_k$ de la méthode d'Euler (et plus généralement à la discrétisation).On peut construire ce tableau à l'aide de la commande t=np.linspace(0,T,n) de la bibliothèque numpy

Cette fonction renvoie un tableau de tableaux (une matrice donc) S=[[y_k]]. On peut récupérer les solutions à l'aide de la commande S[ :;0] qui donne les valeurs du vecteur aux différents temps.

## 3.2 Équations scalaires d'ordre 1

Reprenons l'exemple de l'équation différentielle $y' = y$ et $y(0) = 1$. La résolution à l'aide de scipy avec n=100 et tracé de la courbe s'écrit :

Pour n=4 la comparaison avec une intégration exacte et une intégration avec Euler donne :

## 3.3 Équations scalaires d'ordre 2

Les modifications à apporter sont mineures : la méthode odeint fonctionne de la même manière. Néanmoins la fonction F a désormais pour entrée un vecteur X et la variable t et pour sortie un vecteur F(X) : S=[[y_k]]=[[y'_k,y_k]]

S[ :,0] est un tableau contenant les valeurs prises par la dérivée de y aux différents temps.

S[ :1,] est un tableau contenant les valeurs prises par y aux différents temps.

Il est également possible de résoudre des problèmes à plusieurs variables d'ordre 2 (comme le problème de la sonde spatiale qui est un problème différentiel d'ordre 2 à 2 variables) par des procédés analogues dits de *vectoriallisation*. Dans le cas de la sonde spatiale on manipulera des vecteurs de dimension 4 par exemple.

## 3.4 Tracés avec matplotlib

Les tracés s'effectuent à l'aide de la commande plt.plot(T,Y) où T est le tableau des temps discrétisés et Y le tableau des valeurs de y aux temps correspondants et où on a préalablement importé la libraire à l'aide de import matplotlib.pyplot as plt ; il est également possible de tracer des portraits de phase, qui sont les courbes paramétrées définies par (y(t),y'(t)).