Résolution approchée d'équations différentielles Méthode d'Euler

Anis SAIED
Institut Préparatoire aux Etudes d'Ingénieurs de Nabeul (IPEIN),
Université de Carthage,
Tunisie
anis_saied@hotmail.com

Mise à jour la plus récente : 9 février 2017

Résumé

Après les équations numériques, les équations différentielles. Dans ce cours, nous étudions les équations différentielles linéaires du premier ordre (EDL1). Les équations différentielles ont un lien évident avec l'intégration numérique. C'est donc sans surprise que l'on va utiliser dans cette section des méthodes proches de celles de la précédente : découpage de l'intervalle de résolution en `n` morceaux, puis approximation de la solution sur chacun de ces intervalles. C'est le principe de base de la méthode d'Euler, qui est la principale méthode de résolution numérique d'équations différentielles, sur laquelle nous allons concentrer tous nos efforts. Dans la suite, nous allons étudier la méthode d'Euler, méthode numérique permettant d'obtenir une approximation de la solution d'un tel problème de Cauchy. Et on termine par des exemples d'utilisation de la fonction de résolution approchée d'équations différentielles `odeint`.

Table des matières

Introduction

Ce TP s'intéresse à la résolution numérique d'équations différentielles, l'objectif est d'utiliser la méthode d'Euler pour intégrer quelques équations différentielles simples.

1 Équations différentielles linéaires du premier ordre (EDL1O)

Définition 1. Une **équation différentielle** est une équation qui relie une fonction f à une ou plusieurs de ses dérivées. Exemple : $f(x) = \alpha f'(x) + \beta f''(x)$.

Remarque 1.
Pour simplifier l'écriture des équations différentielles, on remplace couramment $f(x)$ par y et $f'(x)$ par y'.
Exemple : $f(x) = \alpha f'(x) + \beta$ devient $y = \alpha y' + \beta$.

Définition 2. Une **équation différentielle** est une équation dont l'inconnue est une fonction y (réelle ou complexe, même si nous traiterons surtout le cas réel dans ce chapitre), et faisant intervenir les dérivées successives $y', y'', ..., y^{(n)}$ de la fonction y. L'équation est dité **d'ordre** `n` si la dérivée d'ordre le plus élevé de y apparaissant dans l'équation est $y^{(n)}$.

Définition 3. Une équation différentielle est **linéaire** si elle s'écrit sous la forme :

$$a_n y^{(n)} + a_{n-1} y^{(n-1)} + ... + a_1 y' + a_0 y = b$$

où $a_0, a_1, ..., a_n$ et b sont des fonctions.

Définition 4. La fonction b est appelé le **second membre**. Si b est *nulle*, alors l'équation est dite **homogène** ou **sans second membre**. Si $a_0, a_1, ..., a_n$ sont des *constantes*, on parle d'équation différentielle linéaire à **coefficients constants**.

Définition 5. Une courbe intégrale associée à une équation différentielle est une courbe représentative de la fonction solution y de l'équation différentielle.

Remarque 2.
Les équations différentielles considérées doivent être écrites sous la forme : $y' = F(y,t)$
où F est un fonction à deux variables y et t.
Par exemple, pour l'équation différentielle $y' + 2ty = 1$ on définit $F(y,t) = 1 - 2ty$ ce qui s'écrit avec *Python* :

```
def F(y, t):
        return 1-2*t*y
```

Remarque 3.
Même si, comme c'est parfois le cas, on écrit l'équation sous la forme : $y'(t) + 2ty(t) = 1$
il ne faut surtout pas définir la fonction F en écrivant :

```
def F(y, t):
        return 1-2*t*y(t) #INCORRECT
```

Remarque 4.
En mathématiques et en physique, l'habitude est plutôt de considérer les équations différentielles sous la forme $y' = F(t,y)$ et non $y' = F(y,t)$. Mais en Python, la fonction `odeint`, que nous verrons en fin de chapitre, utilise l'écriture $y' = F(y,t)$. Par souci de simplicité, nous nous conformerons à cette écriture dans ce cours.

2 Cadre du problème de Cauchy

Dans le cadre de notre cours on cherche à résoudre un problème numérique simple dit problème de *Cauchy* du type :

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

où :

- y est l'inconnue ;
- y est une fonction dérivable définie sur un intervalle I de $\mathbb{R}$;
- $t_0 \in I$;

On possède donc une équation différentielle et une condition initiale. Le théorème de Cauchy Lipschitz assure l'existence et l'unicité d'une solution à ce problème.

3 Méthode d'Euler

La méthode d'*Euler* est une procédure numérique pour résoudre par approximation des équations différentielles du premier ordre avec une condition initiale. C'est la plus simple des méthodes de résolution numérique des équations différentielles.

On considère une équation différentielle d'ordre 1 avec condition initiale (problème de Cauchy) :

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

On cherche à résoudre le problème de Cauchy sur un intervalle $I = [a,b]$. Soit $n \in \mathbb{N}^*$, On subdivise l'intervalle $[a,b]$ en n petits segments de même longueur $h = \frac{b-a}{n}$.
On pose alors pour $0 \leq k \leq n$:

$$t_0 = a \qquad t_1 = t_0 + h \qquad t_2 = t_0 + 2h \qquad \ldots \qquad t_k = a + kh \qquad \ldots \qquad t_n = t_0 + nh = b$$

On part de t_0 . On approche le petit morceau de courbe entre t_0 et t_1 par la tangente à la courbe au point d'abscisse t_0 .

La méthode d'Euler sert à déterminer approximativement les valeurs prises par y aux instants $t_0, t_1, \ldots, t_{n-1}$.
Nous utiliserons les notations suivantes :

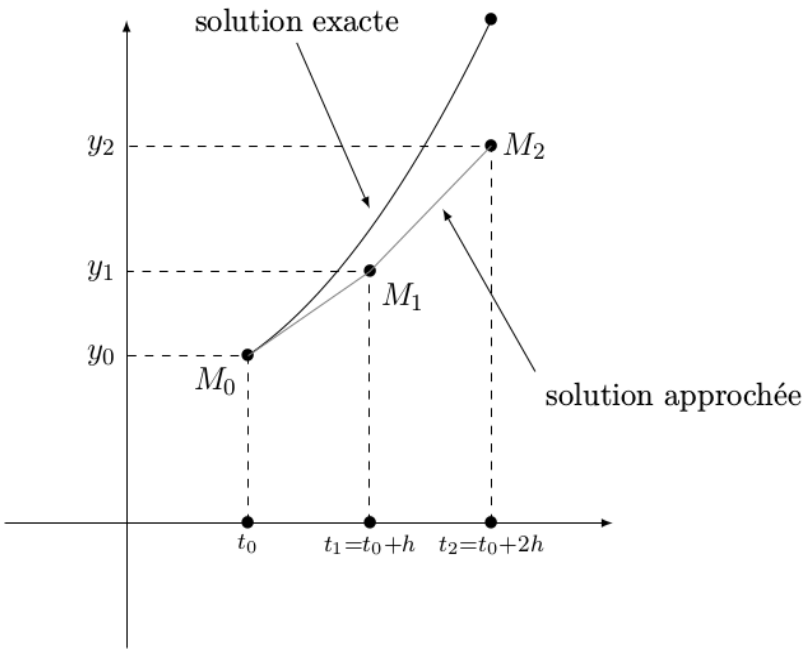
- $y(t_0), y(t_1), \ldots, y(t_{n-1})$ sont les valeurs exactes (on connaît seulement $y(t_0)$) ;
- $y_0, y_1, \ldots, y_{n-1}$ sont les valeurs approchées fournies par la méthode d'Euler.

On a alors :

$$\begin{aligned} \frac{y(t_1) - y(t_0)}{h} &\approx y'(t_0) \\ &\approx F(y(t_0), t_0 \\ y(t_1) &\approx y_0 + hF(y(t_0), t_0) \end{aligned}$$

On pose : $y_1 = y_0 + hF(y_0, t_0)$. Le nombre y_1 est une approximation de la valeur exacte $y(t_1)$.
On recommence le procédé à partir du point $M_1(t_1, y_1)$...
De manière générale, on pose :

$$\forall k \in 0, \ldots, n-1, y_{k+1} = y_k + hF(y_k, t_k)$$



Ce qui amène à construire successivement les points $M_k(t_k, y_k)$. La ligne polygonale reliant ces points est alors une approximation de la courbe représentative de la solution.
Cette méthode de résolution approchée, la seule à notre programme, s'appelle la méthode d'*Euler*.
Il existe de nombreuses autres méthodes qui peuvent être proposées dans le cadre d'un problème.

3.1 Comment programmer la méthode d'Euler ?

Dans ce programme, on donne la valeur du pas h et on vérifie à chaque itération que $a + t_k h \leq b$.

```
def euler(F, a, b, y0, h=1e-2):
        """Fournit une solution approchée à l'équation y'=F(y,t), y(a)=y0
        sur l'intervalle [a,b] par la méthode d'Euler
        le pas utilisé est h, par défaut 0.01
        La solution renvoyée est une liste de valeurs pour les_t,
        et une liste de valeurs pour les_y"""
        y = y0
        t = a
        les_y = [y0] # la liste des valeurs renvoyées
        les_t = [a]
        while t+h <= b:
                #les_y = [y0,...,yk], les_t = [t0,...,tk]
                y += h * F(y, t)
                les_y.append(y)
                t += h
                les_t.append(t)
                # à la fin de la boucle, t contient a+nh≈b
        return les_t, les_y
```

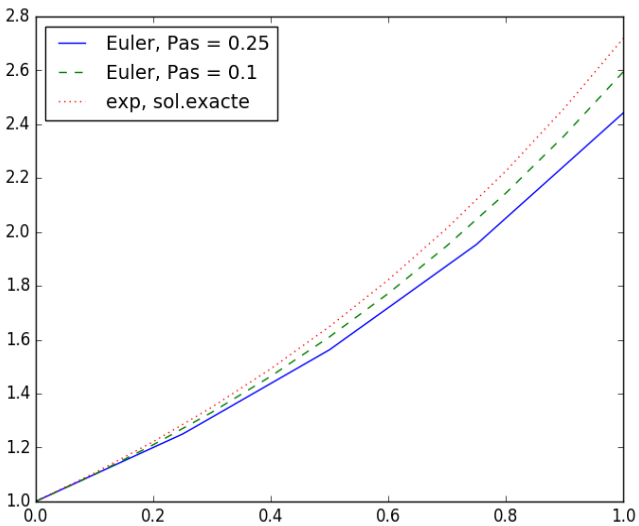
Remarque 5.
Si le nombre de subdivisions n nous est donné à la place de h, on pourra préférer écrire la fonction avec une boucle `while`, en utilisant la fonction `linspace` du module `numpy`.

3.2 Exemple Python

On considère l'équation $y' = y$ avec la condition initiale $y(0) = 1$.
La solution est la fonction exponentielle, ce qui permet de comparer les résultats obtenus.
On va travailler sur l'intervalle $[0,1]$ en considérant tout d'abord un pas de 0.25 puis un pas de 0.1.
La fonction exponentielle est représentée en pointillés pour comparer.
F doit être définie comme une fonction de deux variables même si ici elle ne dépend que de y.

```
import numpy as np
import matplotlib.pyplot as plt
def F(y,t):
        return y

t,y = euler(F,0,1,1,0.25)
plt.plot(t,y,'b-',label='Euler, Pas = 0.25') #ligne bleue continue
print(t)

t,y = euler(F,0,1,1,0.1)
plt.plot(t,y,'g--',label='Euler, Pas = 0.1')  #ligne verte discontinue
print(t)

t = np.linspace(0,1,11)
plt.plot(t,np.exp(t),'r:',label='exp, sol.exacte ') #ligne rouge pointillée
print(t)

plt.legend(loc='upper left')
plt.show()
```

Remarque 6.
La comparaison graphique de la solution générée par la méthode d'Euler avec la solution exacte nous permet de remarquer qu'on peut augmenter la précision de cette méthode d'Euler en diminuant la taille de h, mais un h petit implique une augmentation de nombre d'intervalles $[t_i, t_{i+1}]$, donc on doit évaluer la fonction un grand nombre de fois ce qui est coûteux en temps d'exécution et peut provoquer un cumul des erreurs numériques commises par la machine.



4 La fonction odeint du package scipy.integrate

La bibliothèque **scipy** fournit une fonction de résolution approchée et automatique des équations différentielles ordinaires. Il s'agit de la fonction **odeint**, acronyme de *ordinary differential equations integrating*.
Pour pouvoir l'utiliser, il faut d'abord l'importer :

```
from scipy.integrate import odeint
```

On appelle **odeint** sous la forme **odeint(F,y0,t)** où F est la fonction qui intervient dans l'équation différentielle, y_0 est la valeur initiale et t est le vecteur contenant les valeurs $[t_0 = a, t_1, ..., t_n = b]$.

Après avoir défini les objets de l'équation que l'on veut résoudre, on demande la résolution de l'équation différentielle au moyen de la commande suivante :

```
y = odeint (F , y0 , t )
```

Le résultat y est un tableau des approximations de $y(t)$ correspondant aux différents instants du tableau t.

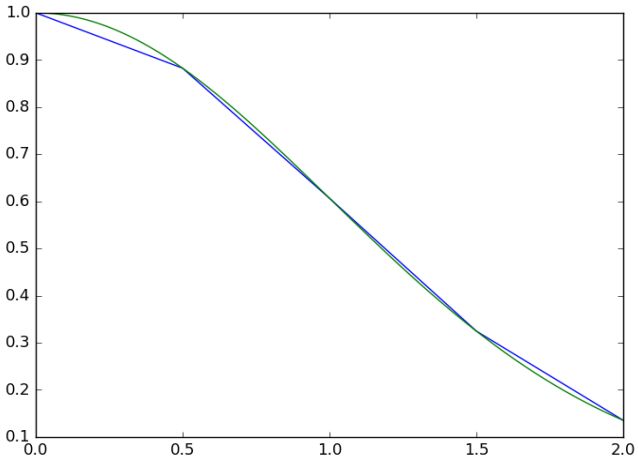
Exemple :
On considère l'équation différentielle d'ordre 1 avec condition initiale :

$$\begin{cases} \frac{dy}{dt} = -ty \\ y(0) = 1 \end{cases}$$

On veut calculer des valeurs approchées de y entre 0 et 2 pour 5 valeurs régulièrement espacées.
Remarque. Le tracé est superposé avec celui de la fonction $t \longmapsto exp(-t^2/2)$.

```
#ED
def F(y,t):
        return -t*y
t=np.linspace(0,2,5)
y=odeint(F,1,t)
print (t,y)
```

```
#exp
exp_t = np.linspace(0,t[4],1000)
exp_y = np.exp(-exp_t**2/2)
plt.plot(t,y,exp_t,exp_y)
plt.show()
```



5 Exemples de résolution approchée d'ED1O.

```
from math import *
from scipy.integrate import odeint
import numpy as np
import matplotlib.pyplot as plt
```

5.1 En mathématiques

Exemple. On veut résoudre sur l'intervalle [0, 5] le problème de Cauchy :

$$\begin{cases} y' + 2ty = 1 \\ y(0) = 1 \end{cases}$$

- La fonction F associée à l'équation différentielle :

```
def F(y, t):
        return 1-2*t*y
```

- Le vecteur t qui représente l'intervalle de temps considéré :

```
t=np.linspace(0,5,100)
```

- Le calcul du vecteur y avec la commande **odeint** :

```
y=odeint(F,1,t)
```

- Représentation graphique :

```
plt.clf()
plt.plot(t,y, 'b-')
plt.xlabel("x")
plt.ylabel("y")
plt.show()
```

5.2 En chimie

Exemple. On considère la réaction :

$$Fe^{2+} + Co^{3+} = Fe^{3+} + Co^{2+}$$

C'est une réaction d'ordre 2 et l'évolution de la concentration en ions Fe^{2+} , notée C , est décrite par l'équation différentielle :

$$\frac{dC}{dt} = -kC^2$$

On donne $k = 80.2 mol^{-1} L \cdot s^{-1}$ et $C_0 = 5.0 \cdot 10^{-4} mol \cdot L^{-1}$. On donne cicontre des mesures expérimentales de C en fonction du temps.

$t(s)$	$C(mol \cdot L^{-1})$
20	$2.78 \cdot 10^{-4}$
40	$1.92 \cdot 10^{-4}$
60	$1.47 \cdot 10^{-4}$
80	$1.19 \cdot 10^{-4}$
100	$1.0 \cdot 10^{-4}$
120	$0.86 \cdot 10^{-4}$

On donne :

```
t_mes=[20,40,60,80,100,120]
C_mes=[2.78e-4,1.92e-4,1.47e-4,1.19e-4,1.0e-4,0.86e-4]
```

- Définir les constantes de l'énoncé :

```
k=80.2
C0=5e-4
```

- La fonction F associée à l'équation différentielle :

```
def F(C,t):
        return -k*C**2
```

- Le vecteur t qui représente l'intervalle de temps considéré :

```
t=np.linspace(0,120,100)
```

- Le calcul du vecteur C avec la commande odeint :

```
C=odeint(F,C0,t)
```

- Représentation graphique :

```
plt.clf()
plt.plot(t,C,'b-')
plt.plot(t_mes,C_mes,'k+')
plt.xlabel("t (s)") ; plt.ylabel("C (mol/L)") ; plt.show()
```

5.3 En mécanique

Exemple. On alimente un moteur initialement à l'arrêt par une tension constante $U = 10V$.
La vitesse de rotation ω du moteur est solution de l'équation différentielle :

$$J\frac{d\omega}{dt} + \left(\frac{K_e K_c}{R} + f\right)\omega(t) = \frac{K_e}{R}U \qquad avec \quad \omega(0) = 0$$

Où :

- J est l'inertie du moteur, $J = 5.5 \cdot 10^{-5} kg \cdot m^2$;
- R est la résistance électrique du système, $R = 5.2\Omega$;
- K_c et K_e sont des constantes, $K_c = 0.24N \cdot m \cdot A^{-1}$ et $K_e = 0.24V \cdot rad^{-1} \cdot s$;
- f est un coefficient caractérisant les forces de frottement, $f = 0.05N \cdot m \cdot s \cdot rad^{-1}$.

Déterminer et représenter l'évolution de la vitesse sur l'intervalle de temps $[0, 10^{-2}]$ (en secondes).

- Définir les constantes de l'énoncé :

```
U=10 ; J=5.5e-5 ; R=5.2 ; Kc=0.24 ; Ke=0.24 ; f=0.05
```

- La fonction F associée à l'équation différentielle :

```
def F(w,t):
        return (Kc/R*U-(Ke*Kc/R+f)*w)/J
```

- Le vecteur t qui représente l'intervalle de temps considéré :

```
t=np.linspace(0,1e-2,100)
```

- Le calcul du vecteur ω avec la commande odeint :

```
w=odeint(F,0,t)
```

- Représentation graphique :

```
plt.clf()
plt.plot(t,w,'b-')
plt.xlabel("t (s)") ; plt.ylabel("w (rad/s)") ; plt.show()
```