Cours 14 : Calcul scientifique. Résolution approchée d'équations différentielles

MPSI-PCSI - Lycée Thiers

Equation différentielle du 2^{nd} ordre

Soit ϕ une application réelle définie sur une partie de $\mathbb{R}^3$. Résoudre, ou intégrer, une équation différentielle d'ordre 2 de la forme :

$$y'' = \phi(x, y, y') \qquad (E)$$

C'est déterminer toutes les application y définies et deux fois dérivables sur une réunion d'intervalles ouverts I de $\mathbb{R}$ et vérifiant :

$$\forall\, x \in I, \quad y''(x) = \phi(x, y(x), y'(x))$$

• Comment les résoudre ? : Nous allons adapter la méthode d'Euler vue pour la résolution d'un E.D.O. du premier ordre au second ordre. Il y a deux approches :

1. Généraliser la méthode d'Euler à un système d'E.D.O. du 1er ordre. Voir comment une E.D.O. du 2nd ordre est équivalente à un système de deux E.D.O. du 1er ordre...
2. La méthode d'Euler revient à appliquer le schéma d'approximation de la dérivée :
$$y'(x) \approx \frac{y(x+dx) - y(x)}{dx}$$
Obtenir de même un schéma d'approximation de la dérivée seconde...

Système de deux E.D.O. du 1er ordre

Considérons un système de deux équations différentielles :

$$(E) \qquad \begin{cases} y' = \phi_1(y,z,t) \\ z' = \phi_2(y,z,t) \end{cases} \qquad \text{avec conditions initiales } y(t_0) \text{ et } z(t_0)$$

où y, z sont des applications réelles dérivables et $\phi_1, \phi_2 : \mathbb{R}^3 \longrightarrow \mathbb{R}$.

Comment le résoudre ?

• Première approche : Méthode d'Euler appliquée à un système

On considère une subdivision régulière $(t_0, t_1, \ldots, t_n)$ de l'intervalle d'intégration $[a,b]$ de pas $dt = \dfrac{b-a}{n}$. On note aussi $y_k = y(t_k)$ et $z_k = z(t_k)$.

En appliquant le schéma d'Euler 2 fois :

$$\forall\, t \in [a,b[\; : \; \begin{cases} \dfrac{y(t+dt)-y(t)}{dt} = \phi_1(y,z,t) \\ \dfrac{z(t+dt)-z(t)}{dt} = \phi_2(y,z,t) \end{cases}$$

$$\Longrightarrow \boxed{\forall\, k \in [[0, n-1]] : \begin{cases} y_{k+1} = y_k + dt \cdot \phi_1(y_k, z_k, t_k) \\ z_{k+1} = z_k + dt \cdot \phi_2(y_k, z_k, t_k) \end{cases}}$$

Système de deux E.D.O. du 1er ordre

Cela nous donne le principe de résolution (code à compléter) :

```
phi1 = lambda y, z, t : ... # à compléter
phi2 = lambda y, z, t : ... # à compléter
a, b, n = ..., ..., ... # à compléter
y0, z0 = ..., ... # à compléter
# Constitution des tableaux T, Y, Z de (n+1) éléments :
T, Y, Z = [0]*(n+1), [0]*(n+1), [0]*(n+1)    # ou np.empty(n+1)
T[0], Y[0], Z[0] = a, y0, z0    # Conditions initiales
# Remplissage :
dt = (b-a)/n
for k in range(n):
    Y[k+1] = Y[k] + dt * phi1(Y[k],Z[k],T[k])
    Z[k+1] = Z[k] + dt * phi2(Y[k],Z[k],T[k])
    T[k+1] = T[k] + dt
```

À la fin du script les tableaux `T`, `Y` et `Z` contiennent la subdivision régulière et les valeurs qu'y prennent y et z.

2ème approche : Euler appliqué à une E.D.O vectorielle.

Notons :

$$Y : \begin{array}{ccc} \mathbb{R} & \longrightarrow & \mathbb{R}^2 \\ t & \longmapsto & \begin{pmatrix} y(t) \\ z(t) \end{pmatrix} \end{array} ; \; Y(t_0) = \begin{pmatrix} y(t_0) \\ z(t_0) \end{pmatrix} \text{ et } \quad Y' : \begin{array}{ccc} \mathbb{R} & \longrightarrow & \mathbb{R}^2 \\ t & \longmapsto & \begin{pmatrix} y'(t) \\ z'(t) \end{pmatrix} \end{array}$$

Notons de même :

$$\Phi : \begin{array}{ccc} \mathbb{R}^2 \times \mathbb{R} & \longrightarrow & \mathbb{R}^2 \\ \begin{pmatrix} y \\ z \end{pmatrix}, t & \longmapsto & \begin{pmatrix} \phi_1(y,z,t) \\ \phi_2(y,z,t) \end{pmatrix} \end{array}$$

Alors l'équation (E) est équivalente à l'équation différentielle vectorielle :

$$Y' = \Phi(Y,t) \qquad \text{avec condition initiale } Y(t_0)$$

En appliquant le schéma d'Euler :

$$Y'(t) \approx \frac{Y(t+dt) - Y(t)}{dt} \Longrightarrow \boxed{Y(t+dt) \approx Y(t) + dt \times \Phi(Y(t),t)}$$

2ème approche : Euler appliqué à une E.D.O vectorielle.

Code : Il est préférable d'utiliser des tableaux `numpy`.

```
import numpy as np
# Remplacer phi1, phi2 par leur expression :
Phi = lambda Y, t: np.array([phi1(Y[0],Y[1],t),phi2(Y[0],Y[1],t)])
a, b, n = ..., ..., ... # à compléter
y0, z0 = ..., ... # à compléter
Y0 = np.array([y0,z0]) # Condition initiale
Y = [Y0] + [None] * n
t = a
dt = (b-a)/n
for k in range(n):
    Y[k+1] = Y[k] + dt * Phi(Y[k],t) # Euler vectoriel
    t = t + dt
y = [x[0] for x in Y] # Tableau des valeurs de la fonction y
z = [x[1] for x in Y] # Tableau des valeurs de la fonction z
```

A la fin du script le tableau `Y` contient la suite des vecteurs (y_k, z_k). Pour récupérer les valeurs des fonctions y, z on récupère les première et deuxième composantes dans deux tableaux.

`Y[0]` et `Y[1]` permettent d'accéder aux deux composantes de la variable `Y` qui

3ème approche : odeint appliqué à une E.D.O vectorielle.

Avec les mêmes notations qu'au paragraphe précédent :

```
import numpy as np
from scipy import integrate
# Remplacer phi1, phi2 par leur expression :
Phi = lambda Y, t :np.array([phi1(Y[0],Y[1],t),phi2(Y[0],Y[1],t)])
a, b, n = ..., ..., ... # à compléter
y0, z0 = ..., ... # à compléter
Y0 = np.array([y0,z0]) # Condition initiale
T = np.linspace(a,b,n+1)
Y = integrate.odeint(Phi,Y0,T)
y = Y[:,0]
z = Y[:,1]
```

La fonction odeint retourne une matrice numpy ce qui permet de permet de récupérer les tableaux de ses première et deuxième colonne par slicing.

Exemple de résolution d'EDO d'ordre 2

- Exemple :

$$y'' = \cos(t) \qquad \text{avec } y(0) = -1 \text{ et } y'(0) = 0$$

a pour solution évidente $y(t) = -\cos(t)$. sur $\mathbb{R}$.

Pour le transformer en un système de deux E.D.O. d'ordre 1 équivalent on insère la **variable auxiliaire** $\boxed{z = y'}$.

L'équation différentielle est équivalente au système (problème de Cauchy) :

$$\begin{cases} \dfrac{d}{dt}y = z \\ \dfrac{d}{dt}z = \cos(t) \end{cases} \qquad \text{avec : } y(0) = -1 \text{ et } z(0) = 0$$

qui a pour solution évidente : $z(t) = \sin(t)$ et $y(t) = -\cos(t)$.

Equa. diff. d'ordre supérieur

- Le système :

$$\begin{cases} \dfrac{d}{dt}y = z \\ \dfrac{d}{dt}z = \cos(t) \end{cases} \qquad \text{avec : } y(0) = -1 \text{ et } z(0) = 0$$

s'écrit comme un problème de Cauchy vectoriel :

$$\frac{d}{dt}\begin{pmatrix} y \\ z \end{pmatrix} = \begin{pmatrix} z \\ \cos(t) \end{pmatrix} \quad \text{et} \quad \begin{pmatrix} y(0) \\ z(0) \end{pmatrix} = \begin{pmatrix} -1 \\ 0 \end{pmatrix}$$

Qui a pour solution évidente :

$$Y(t) = \begin{pmatrix} -\cos(t) \\ \sin(t) \end{pmatrix}$$

Equa. diff. d'ordre supérieur - Résolution

Avec `Odeint` :

$$\frac{d}{dt}\begin{pmatrix} y \\ z \end{pmatrix} = \begin{pmatrix} z \\ \cos(t) \end{pmatrix} \quad \text{et} \quad \begin{pmatrix} y(0) \\ z(0) \end{pmatrix} = \begin{pmatrix} -1 \\ 0 \end{pmatrix}$$

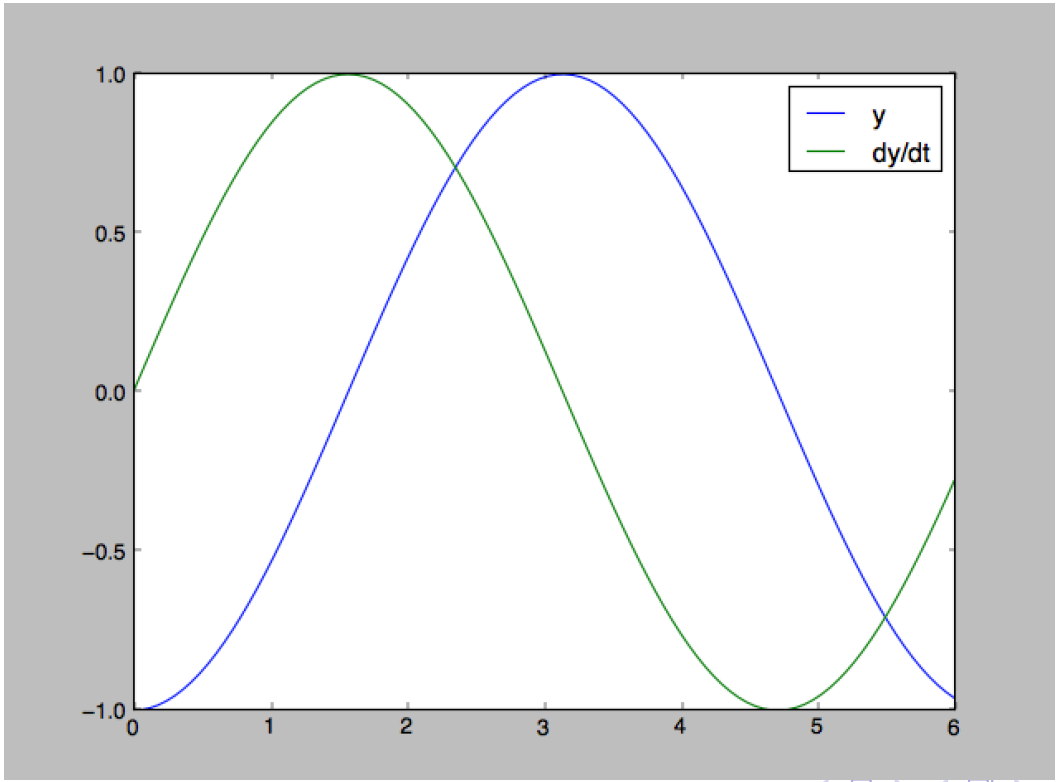
En posant : $Y = \begin{pmatrix} y \\ z \end{pmatrix} = \begin{pmatrix} Y[0] \\ Y[1] \end{pmatrix}$:

```
import numpy as np
def F(Y,t):          # array des fonctions aux seconds membres
    return np.array([Y[1],np.cos(t)])
from scipy import integrate
T = np.linspace(0,6,100)     # Subdivision régulière de [0,6]
y0 = np.array([-1,0])         # array des conditions initiales
Y = integrate.odeint(F,y0,T)     # appel de odeint
```

Equa. diff. d'ordre supérieur

Tracé du graphe :

```
import matplotlib.pyplot as plt
plt.plot(T,Y)
plt.legend(('y','dy/dt'))
plt.show()
```



Equa. diff. d'ordre supérieur - Résolution

Avec Euler vectoriel :

$$\frac{d}{dt}\begin{pmatrix} y \\ z \end{pmatrix} = \begin{pmatrix} z \\ \cos(t) \end{pmatrix} \quad \text{et} \quad \begin{pmatrix} y(0) \\ z(0) \end{pmatrix} = \begin{pmatrix} -1 \\ 0 \end{pmatrix}$$

En posant : $Y = \begin{pmatrix} y \\ z \end{pmatrix} = \begin{pmatrix} Y[0] \\ Y[1] \end{pmatrix}$:

```
import numpy as np
Phi = lambda Y,t: np.array([Y[1],np.cos(t)])
N = 100
T = np.linspace(0,6,N+1)     # Subdivision régulière de [0,6]
y0 = np.array([-1,0])         # array des conditions initiales
Y = [y0] + [None]*N
dt = 6/N
for k in range(N):
    Y[k+1] = Y[k] + dt * Phi(Y[k],T[k])     # Euler Vectoriel
```

Equa. diff. d'ordre supérieur - Résolution

Avec Euler appliqué au système :

$$\begin{cases} \dfrac{d}{dt}y = z \\ \dfrac{d}{dt}z = \cos(t) \end{cases} \qquad \text{avec : } y(0) = -1 \text{ et } z(0) = 0$$

```
import numpy as np
N = 100
T = np.linspace(0,6,N+1)
Y = np.empty(N+1)
Z = np.empty(N+1)
Y[0], Z[0] = -1, 0
dt = 6/N
for k in range(N):
    Y[k+1] = Y[k] + dt * Z[k]
    Z[k+1] = Z[k] + dt * np.cos(T[k])
```

Equa. diff. d'ordre 2 : par schéma d'Euler explicite

- Sans se ramener à un système d'équations différentielles, on peut aussi résoudre une équation différentielle à l'ordre 2 en utilisant un schéma explicite d'approximation de la dérivée seconde.
- Schéma d'Euler à l'ordre 1 se ramène à l'approximation :

$$\boxed{f'(x) \approx \frac{f(x+dx) - f(x)}{dx}}$$

pour exprimer $f(x+dx)$ en fonction de dx, $f(x)$ et $f'(x)$:

$$f(x+dx) = f(x) + dxf'(x)$$

- Schéma d'Euler à l'ordre 2 se ramène à l'approximation :

$$\boxed{f''(x) \approx \frac{f(x+dx) - 2f(x) + f(x-dx)}{dx^2}}$$

pour exprimer $f(x+dx)$ en fonction de dx, $f(x)$, $f(x-dx)$ et $f''(x)$.

- Schéma d'Euler à l'ordre 2 :

$$\boxed{f''(x) \approx \frac{f(x+dx) - 2f(x) + f(x-dx)}{dx^2}}$$

Comment s'obtient-il ?

Grâce au développement de Taylor-Young à l'ordre 3 de la fonction f :

$$f(x+dx) = f(x) + dx.f'(x) + \frac{1}{2}f''(x)dx^2 + \frac{1}{6}f'''(x)dx^3 + o(dx^3)$$
$$f(x-dx) = f(x) - dx.f'(x) + \frac{1}{2}f''(x)dx^2 - \frac{1}{6}f'''(x)dx^3 + o(dx^3)$$

$$\Longrightarrow f(x+dx) + f(x-dx) = 2f(x) + f''(x)dx^2 + o(dx^3)$$
$$\Longrightarrow f''(x) = \frac{f(x+dx) + f(x-dx) - 2f(x)}{dx^2} + o(dx)$$

Résolution par schéma d'Euler explicite

A résoudre sur $[0,6]$:

$$y'' = \cos(t) \qquad \text{avec } y(0) = -1 \text{ et } y'(0) = 0$$

On insère les schémas d'Euler à l'ordre 1 et 2 :

$$y(t+dt)+y(t-dt)-2y(t) = dt^2\cos(t) \quad \text{avec} \quad y(0) = -1 \text{ et } y(dt)-y(0) = 0\times dt$$

Soit :

$$\boxed{y(t+dt) = 2y(t) - y(t-dt) + dt^2\cos(t) \qquad \text{avec} \qquad y(0) = y(dt) = -1}$$

```
Nt = 100    choix du nombre de points
T = np.linspace(0,6,Nt+1)
Y = np.empty(Nt+1)
Y[0] = Y[1] = -1    # Conditions initiales
dt = 6 / Nt
for k in range(1, Nt):
    Y[k+1] = 2 * Y[k] - Y[k-1] + dt**2 * np.cos(T[k])
```