

RÉSOLUTION APPROCHÉE D'ÉQUATIONS DIFFÉRENTIELLES

On cherche à approcher numériquement une solution f sur $[a, b]$ d'un problème de Cauchy de la forme :

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

Pour cela, on découpe $[a, b]$ en n intervalles de longueur $h = \frac{b-a}{n}$; on pose

$$t_0 = a, t_1 = a + h, \dots, t_i = a + i \times h, \dots, t_n = a + n \times h = b$$

On calcule ensuite des valeurs approchées de $f(t_k)$ pour tout $k \in \llbracket 0, n \rrbracket$.

I. Méthode d'Euler

1) Principe

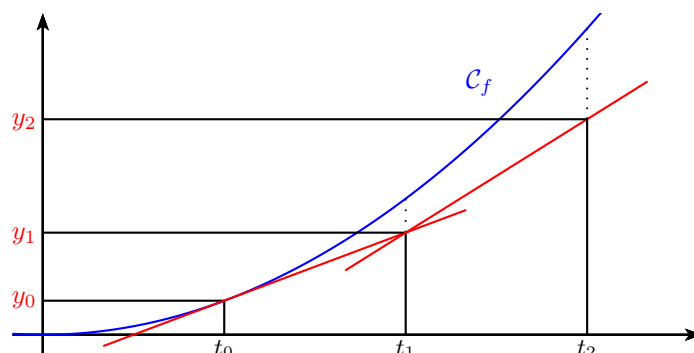
Posons $y_0 = f(t_0)$.

Comme $f(t_0 + h) \underset{h \rightarrow 0}{=} f(t_0) + hf'(t_0) + o(h)$, on a $f(t_1) \approx f(t_0) + hf'(t_0)$, i.e $f(t_1) \approx f(t_0) + hF(t_0, f(t_0))$.

On pose donc $y_1 = y_0 + hF(t_0, y_0)$.

De même, $f(t_2) \approx f(t_1) + hf'(t_1)$, soit encore $f(t_2) \approx f(t_1) + hF(t_1, f(t_1))$.

Comme $f(t_1) \approx y_1$, on pose $y_2 = y_1 + hF(t_1, y_1)$



On définit ainsi $y_0, \dots, y_n$ tels que $y_k \approx f(t_k)$, en posant :

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

2) Implémentation

```
def euler(F,a,b,y0,n):
    h = (b-a)/n # le pas
    y = y0 # la valeur courante de y
    t = a # la valeur courante de t
    les_yk = [y0]
    les_tk = [a]
    for i in range(n):
        y = y + h*F(t,y)
        t = t + h
        les_yk.append(y)
        les_tk.append(t)
    return les_tk, les_yk
```

3) Exemple

On veut résoudre numériquement sur $[0, 1]$:

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

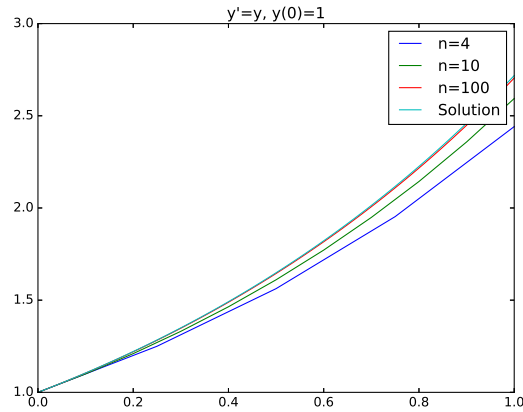
Cette équation différentielle est de la forme $y' = F(t, y(t))$ avec $F : (t, z) \mapsto z$.

Pour utiliser la fonction `euler`, on définit donc :

```
def F1(t,z):
    return z
```

L'appel `euler(F1, 0, 1, 1, 4)` permet d'obtenir :

`([0, 0.25, 0.5, 0.75, 1.0], [1, 1.25, 1.5625, 1.953125, 2.44140625])`



II. Equation différentielle d'ordre 2

On souhaite approcher numériquement une solution f sur $[a, b]$ d'un problème de Cauchy de la forme :

$$\begin{cases} y'' = F(t, y(t), y'(t)) \\ y(a) = y_0 \\ y'(a) = y'_0 \end{cases}$$

1) Une premier approche

On procède comme pour l'ordre 1 : pour tout $k \in \llbracket 0, n-1 \rrbracket$, on pose

$$y_{k+1} = y_k + h y'_k \text{ et } y'_{k+1} = y'_k + h F(t_k, y_k, y'_k)$$

```
def euler_ordre2(F,a,b,y0,yp0,n):
    h = (b-a)/n # le pas
    y = y0 # la valeur courante de y
    yp = yp0 # La valeur courante de y'
    t = a # la valeur courante de t
    les_yk = [y0]
    les_ypk = [yp0]
    les_tk = [a]
    for i in range(n):
        y, yp = y + h*yp, yp + h*F(t,y,yp)
        t = t + h
        les_yk.append(y)
        les_ypk.append(yp)
        les_tk.append(t)
    return les_tk, les_yk, les_ypk
```

2) En se ramenant à l'ordre 1

Plutôt que de réécrire la fonction `euler`, on peut traduire ce problème d'ordre 2 en un problème d'ordre 1 à deux dimensions :

$$Y' = F_2(t, Y(t))$$

où $Y(t) = (y(t), y'(t))$ et $F_2(t, u, v) = (v, F(t, u, v))$.

On réutilise ensuite la fonction `euler` définie dans la première partie.

Pour effectuer des calculs sur des couples, on utilise la structure `array` du module `numpy`.

Exemple

L'équation différentielle $y'' + y = 0$ peut s'écrire $Y'(t) = F(t, Y(t))$ avec $F(t, (z, w)) = (w, -z)$.

On définira donc :

```
def F(t,v):
    [w,z] = v
    return np.array([z,-w])
```

Pour résoudre de manière approchée cette équation sur $[0, 12]$ avec les conditions initiales $y(0) = 0$ et $y'(0) = 1$ avec 100 étapes de la méthode d'Euler, il suffit ensuite d'appeler :

```
euler(F, 0, 12, np.array([0,1]), 100)
```

III. Pour aller plus loin : d'autres méthodes

1) Méthode de Heun

Plutôt que de poser $y_{k+1} = y_k + hF(t_k, y_k)$, on pose :

$$y_{k+1} = y_k + h \times \frac{1}{2} (F(t_k, y_k) + F(t_{k+1}, y_k + hF(t_k, y_k)))$$

La quantité $y_k + hF(t_k, y_k)$ n'est autre que la valeur approchée de $f(t_{k+1})$ par la méthode d'Euler, la quantité $F(t_{k+1}, y_k + hF(t_k, y_k))$ est donc une valeur approchée pour $f'(t_{k+1})$.

2) Méthode de Runge-Kutta d'ordre 4

On pose :

$$\alpha_k = y_k + \frac{h}{2} F(t_k, y_k), \quad \beta_k = y_k + \frac{h}{2} F\left(t_k + \frac{h}{2}, \alpha_k\right), \quad \gamma_k = y_k + hF\left(t_k + \frac{h}{2}, \beta_k\right)$$

puis

$$y_{k+1} = y_k + h \times \frac{1}{6} \left(F(t_k, y_k) + 2F\left(t_k + \frac{h}{2}, \alpha_k\right) + 2F\left(t_k + \frac{h}{2}, \beta_k\right) + F(t_k + h, \gamma_k) \right)$$

3) Comparaison

On étudie le temps de calcul et l'erreur en fonction du nombre d'itérations lors du calcul d'une valeur approchée en 1 de la solution de

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

— Méthode d'Euler :

n	10^1	10^2	10^3	10^4	10^5	10^6	10^7
Temps (s)	$1,9 \cdot 10^{-5}$	$7,4 \cdot 10^{-5}$	$5,5 \cdot 10^{-4}$	$5,1 \cdot 10^{-3}$	$4,9 \cdot 10^{-2}$	$5,4 \cdot 10^{-1}$	5,6
Erreur	$1,2 \cdot 10^{-1}$	$1,3 \cdot 10^{-2}$	$1,3 \cdot 10^{-3}$	$1,4 \cdot 10^{-4}$	$1,4 \cdot 10^{-5}$	$1,4 \cdot 10^{-6}$	$1,4 \cdot 10^{-7}$

— Méthode de Heun :

n	10^1	10^2	10^3	10^4	10^5	10^6
Temps (s)	$2,6 \cdot 10^{-5}$	$1,5 \cdot 10^{-4}$	$1,0 \cdot 10^{-3}$	$1,0 \cdot 10^{-2}$	$1,0 \cdot 10^{-1}$	1,0
Erreur	$4,2 \cdot 10^{-3}$	$4,5 \cdot 10^{-5}$	$4,5 \cdot 10^{-7}$	$4,5 \cdot 10^{-9}$	$4,5 \cdot 10^{-11}$	$4,1 \cdot 10^{-13}$

— Méthode RK4 :

n	10^1	10^2	10^3
Temps (s)	$4,6 \cdot 10^{-5}$	$3,4 \cdot 10^{-4}$	$3,2 \cdot 10^{-3}$
Erreur	$2,1 \cdot 10^{-6}$	$2,2 \cdot 10^{-10}$	$2,0 \cdot 10^{-14}$