CHAPITRE 8

RÉSOLUTION D'ÉQUATIONS DIFFÉRENTIELLES

Sommaire

I Équation différentielle d'ordre 1 1
I.1 Principe 1
I.2 Méthode d'Euler Explicite 2
I.3 Exemples d'utilisation 3
I.4 Limite de la méthode 7
I.5 Méthode d'Euler implicite 8
II Équations différentielles du second ordre 9
II.1 Exemple : tension aux bornes du condensateur sur un circuit RLC 9
II.2 Équation du pendule 11
III Utilisation de odeint 12
IV Fonctions Euler explicite sur Python 13
IV.1 Pour les équations d'ordre 2 14

Le but de ce cours est de construire des méthodes numériques pour résoudre des équations différentielles. Les équations différentielles interviennent dans différentes filières en CPGE (maths, physique, chimie ou encore SI) et ne sont pas toujours résolubles (problèmes des trois corps par exemple). Cependant, il existe plusieurs méthodes numériques permettant d'en approcher les solutions. Nous allons, ici en regarder quelques unes.

I Équation différentielle d'ordre 1

Dans un premier temps, nous allons nous intéresser aux équations différentielles d'ordre 1 mis en problème de Cauchy : On cherche à trouver y dérivable sur $[t_0, T]$ telle que :

$$\begin{cases} y'(t) = F(t, y(t)) \\ y(t_0) = y_0 \end{cases} \tag{8.1}$$

sur $[t_0, T]$ avec F une fonction continue de $[t_0, T] \times \mathbb{R}$. Cette écriture regroupe à la fois les équations dites "linéaires", en prenant $F : (t, y) \to a(t)y$ ce qui donne :

$$y'(t) = a(t)y(t)$$

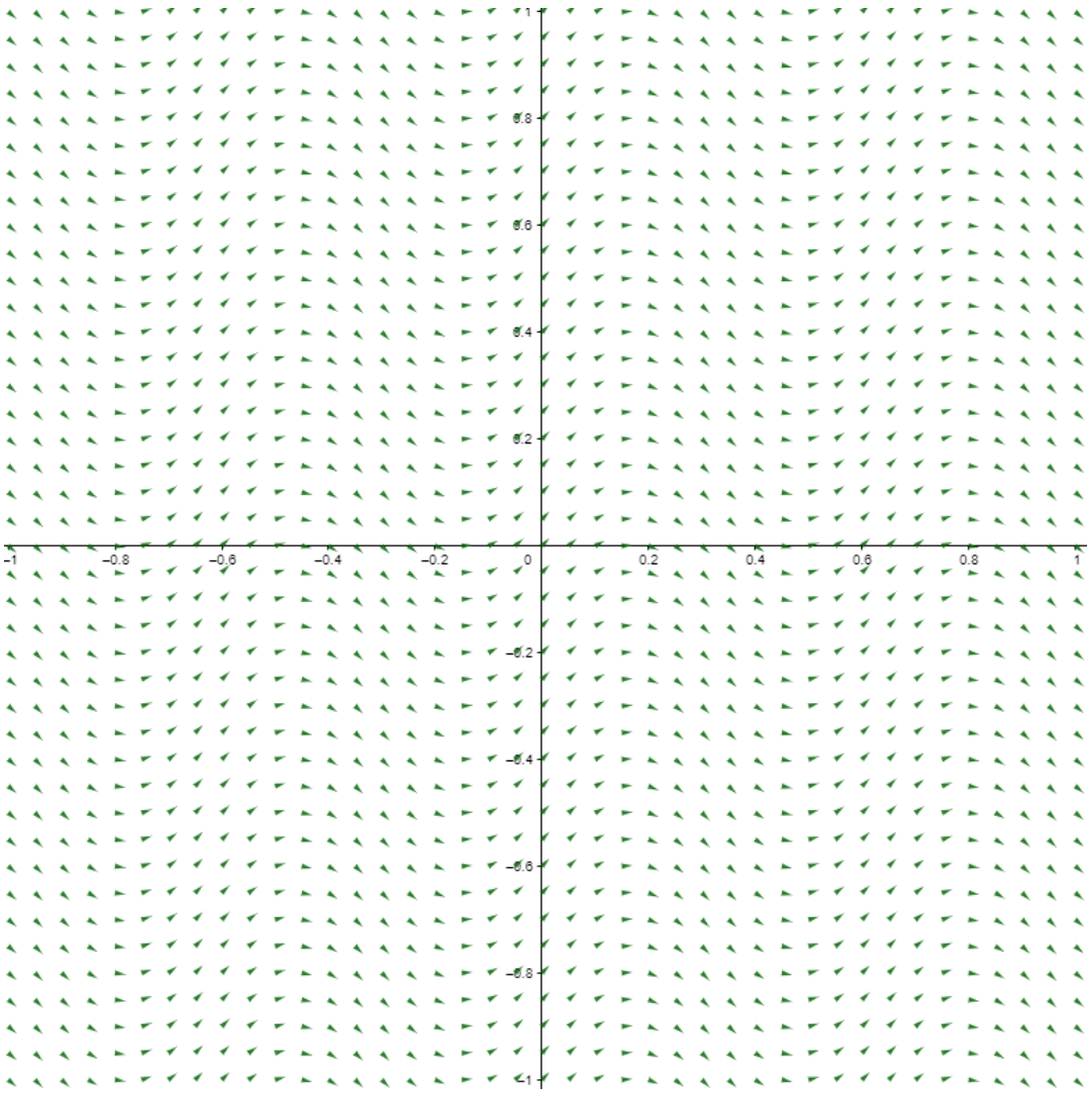
mais également des équations plus difficiles à résoudre et non linéaires, en prenant, par exemple $F(t, y) \to \cos(ty)$ ce qui donne :

$$y'(t) = \cos(ty(t)).$$

I.1 Principe

Exercice 8.1: On s'intéresse à l'équation $y'(t) = \cos(10t)$ sur le segment $[-1, 1]$, sur le graphe suivant, chaque vecteur a pour direction la même direction que la tangente à la courbe de la fonction solution de l'équation précédente et passant par ce point. Sans la calculer, dessiner en ligne brisée la fonction solution valant 0 en 0 (on placera un point tout les 0.1 unité)

E



Principe 8.2: Approximation avec la donnée de la dérivée

Soit y une fonction dérivable sur un intervalle $[t_0, T]$, il existe une fonction ϵ de limite nulle en 0 tel que :

$$y(t_0 + h) = y(t_0) + hy'(t_0) + \epsilon(h)$$

ainsi, si h est "petit" on peut dire :

$$y(t_0 + h) \approx y(t_0) + hy'(t_0)$$

et plus h est petit, plus l'approximation devient précise.

I.2 Méthode d'Euler Explicite

Si y est une solution du problème de Cauchy 8.1, l'idée est de construire point par point une approximation de la solution en partant du seul point qu'on connait de la courbe représentative de y, c'est-à-dire (t_0, y_0). On détermine la valeur de $y'(t_0)$ grâce à l'équation différentielle et on approxime la valeur de la solution en $t_0 + h$ avec la méthode précédente :

$$\boxed{y(t_0+h) \approx y(t_0) + hF(t_0, y(t_0))}$$

En effet, on rappelle que $y'(t_0) = F(t_0, y(t_0))$. On note alors $y_0 = y(t_0)$ et y_1 la valeur approchée de $y(t_0+h)$ c'est-à-dire :

$$y_1 = y_0 + hF(t_0, y_0)$$

en connaissant, y_1 une valeur approchée de $y(t_0+h)$ on connait également une valeur approchée de la dérivée car $y'(t_0+h) = F(t_0+h, y(t_0+h))$ on alors $y'(t_0+h) \approx F(t_0+h, y_1)$. On peut alors construire y_2 une approximation de $y(t_0+2h)$ de la manière suivante :

$$\boxed{y_2 = y_1 + hF(t_0+h, y_1)}$$

et ainsi de suite. Pour construire une solution sur l'intervalle $[t_0, T]$ de notre équation, on choisit h, qui est le pas temporelle entre deux points du graphe de notre solution, de manière équirépartie sur tout l'intervalle : si on veut construire N points de notre solution on prend le pas $h = \dfrac{T-t_0}{N}$. La méthode d'Euler est une méthode itérative : à partir des valeurs t_0, T, $y(t_0)$, N et de la fonction F, on construit deux suites (t_n) et (y_n) comme suit :

$$\begin{cases} t_{n+1} = t_n + \dfrac{T-t_0}{N} = t_n + h \\ y_{n+1} = y_n + hF(t_n, y_n) \end{cases}$$

La valeur de h est alors appelé le pas de discrétisation en temps.

Définition 8.3: Méthode d'Euler explicite avec un pas régulier

Soit y une solution de l'équation :

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

sur $[t_0, T]$ avec F une fonction continue de $[t_0, T] \times \mathbb{R}$. Soit $N \in \mathbb{N}^*$ Si on note le pas de temps $h = \dfrac{T-t_0}{N}$, la **méthode d'Euler explicite** permet de construire un $(N+1)$-uplet $(y_k)_{k\in[\![0,N]\!]}$ approchant les valeurs du $(N+1)$-uplet $(y(t_0+kh))_{k\in[\![0,N]\!]}$. La suite est définie par récurrence comme suit :

$$\begin{cases} t_{n+1} = t_n + h \\ y_{n+1} = y_n + hF(t_n, y_n) \end{cases}$$

I.3 Exemples d'utilisation

Approcher l'exponentielle

Commençons par un problème simple où nous approchons la fonction exponentielle sur $[0, 1]$ via l'équation différentielle pour laquelle elle est solution :

$$\begin{cases} y'(t) = y(t) \\ y(0) = 1 \end{cases} \tag{8.2}$$

Exercice 8.4: On note $h = 1/N$ où N est le nombre de points crée par la méthode d'Euler explicite. Quelle la suite $(y_k)_{k\in[\![0,N]\!]}$ approchant les valeurs du $(N+1)$-uplet $(\exp(h/N))_{k\in[\![0,N]\!]}$ ainsi crée ? Spécifier surtout y_N approchant le nombre de Neper.

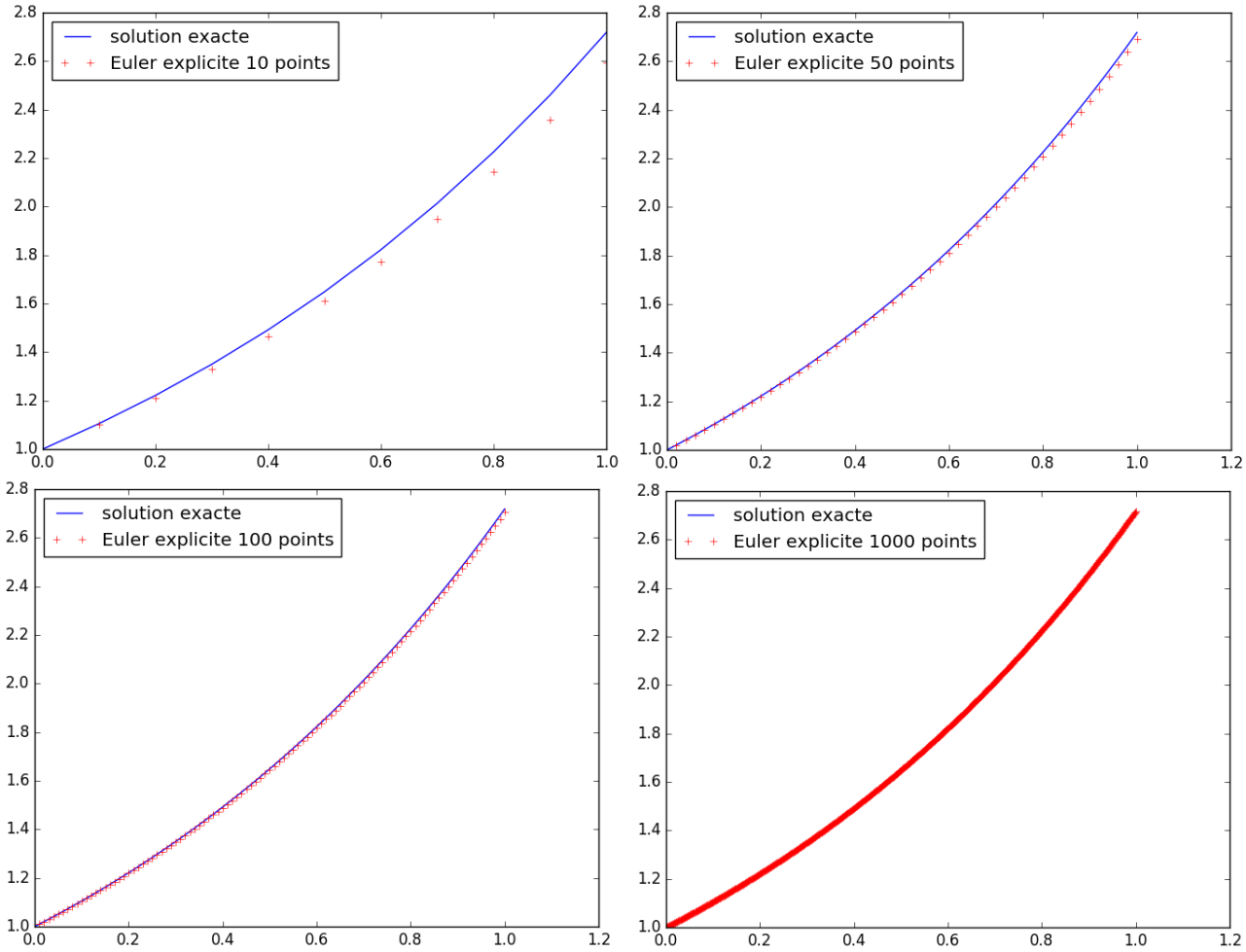
E

E

En voici le code :

méthode d'Euler explicite

```
import numpy as np
import matplotlib.pyplot as plt
t_0=0
y_0=1
T=1
N=1000
h=(T-t_0)/N
list_ye=[y_0]
list_t=[t_0]

y=y_0
t=t_0
ye=y_0
for k in range(N):
    ye=ye+h*ye
    t=t+h
    list_ye.append(ye)
    list_t.append(t)

sol=np.exp(list_t)

b,=plt.plot(list_t,sol,'-b',label='solution exacte')
c,=plt.plot(list_t,list_ye,'+r',label='Euler explicite '+str(N)+' points')

plt.legend(handles=[b,c],loc=2)
plt.show()
```



Approcher la charge du circuit RC

Le second problème est d'approcher la charge d'un circuit RC sur $[0,1]$ via l'équation différentielle pour laquelle elle est solution :

$$\begin{cases} y'(t) = \dfrac{1}{RC}\left(E - y(t)\right) \\ y(0) = 0 \end{cases}$$

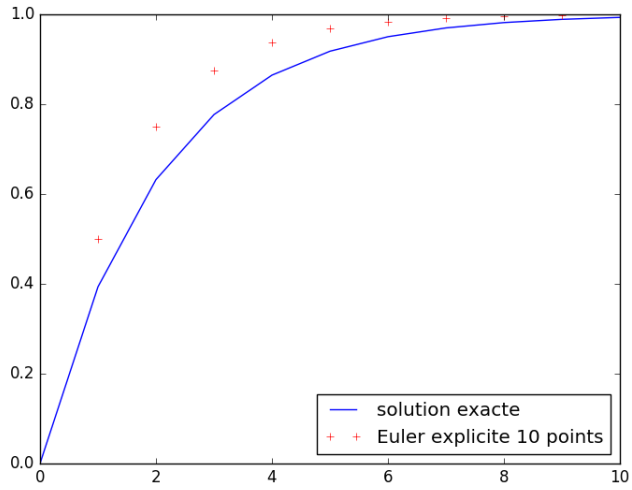
Le pas de temps est alors $h = 1/N$ où N est le nombre de points crée par la méthode. La méthode d'Euler explicite est alors $y_0 = 0$ et $t_0 = 0$ puis :

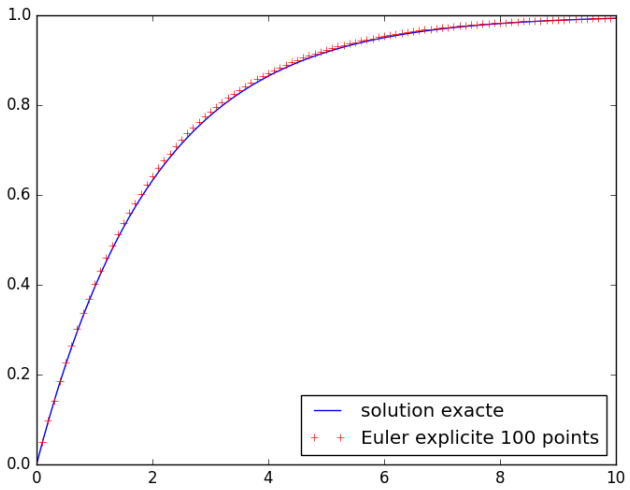
$$\begin{cases} t_{n+1} = t_n + \dfrac{1}{N} = t_n + h \\ y_{n+1} = y_n + \dfrac{h}{RC}\left(E - y_n\right) \end{cases}$$

En voici le code :

méthode d'Euler explicite

```
import numpy as np
import matplotlib.pyplot as plt
E=1;RC=2;t_0=0;y_0=0;T=10;N=10
h=T/N
list_y=[y_0]
list_t=[t_0]

y=y_0
t=t_0
for k in range(N):
    y=y+h*(E-y)/RC
    t=t+h
    list_y.append(y)
    list_t.append(t)

sol=E*(1-np.exp(-np.array(list_t)/RC))

b,=plt.plot(list_t,sol,'-b',label='solution exacte')
c,=plt.plot(list_t,list_y,'+r',label='Euler explicite '+str(N)+' points')

plt.legend(handles=[b,c],loc=4)
plt.show()
```





Approcher une solution qu'on ne connait pas formellement

Si on connait des méthodes de calculs qui nous permettent de connaître la solution formelle des deux dernières équations, ce n'est pas le cas de la suivante :

$$\begin{cases} y'(t) = \cos(ty(t)) \\ y(0) = 0 \end{cases}$$

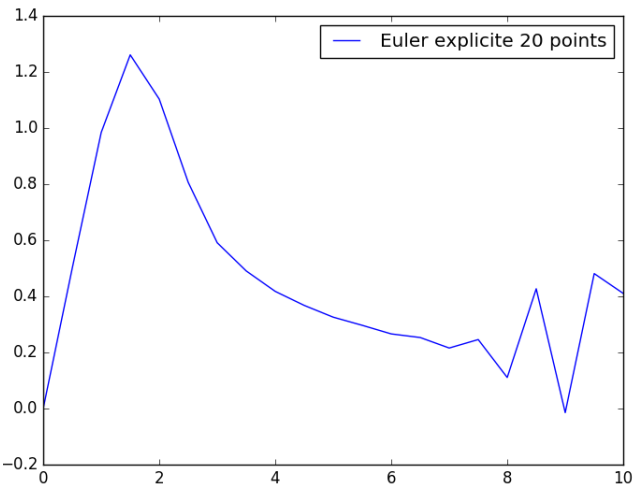
On cherche à résoudre cette équation sur $[0, 10]$. Le pas de temps est alors $h = 10/N$ où N est le nombre de points crée par la méthode. La méthode d'Euler explicite est alors $y_0 = 0$ et $t_0 = 0$ puis :

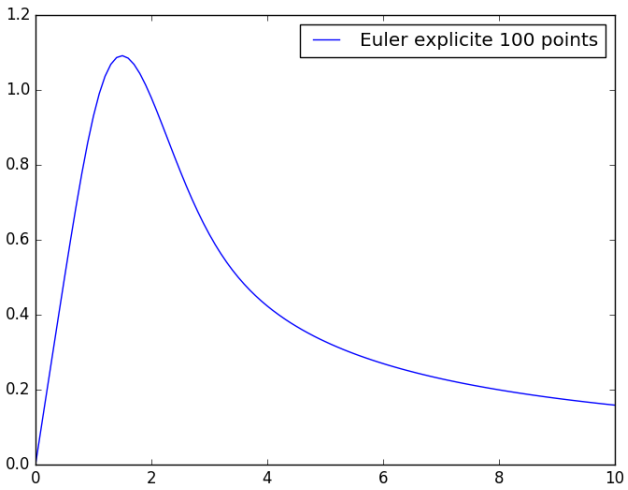
$$\begin{cases} t_{n+1} = t_n + \dfrac{10}{N} = t_n + h \\ y_{n+1} = y_n + h\cos(t_n y_n) \end{cases}$$

dont voici le code :

méthode d'Euler explicite

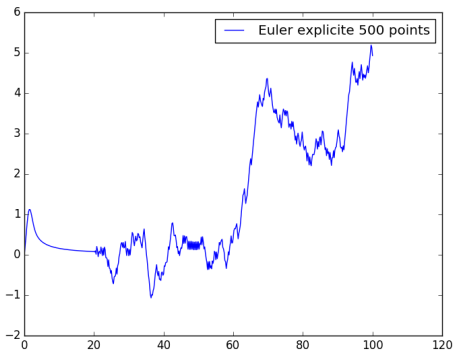
```
from math import cos
import numpy as np
import matplotlib.pyplot as plt
t_0=0;y_0=0;T=10;N=20;h=T/N
list_y=[y_0]
list_t=[t_0]

y=y_0
t=t_0
for k in range(N):
    y=y+h*cos(t*y)
    t=t+h
    list_y.append(y)
    list_t.append(t)



c,=plt.plot(list_t,list_y,'-',label='Euler explicite '+str(N)+' points')

plt.legend(handles=[c],loc=1)
plt.show()
```

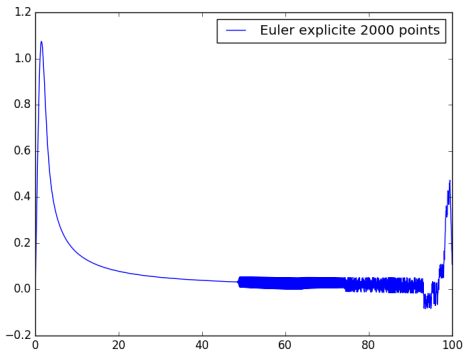




I.4 Limite de la méthode

Bien que cet algorithme semble fonctionner à merveille sur les exemples énoncés, il faut tout de même faire attention : cet algorithme ne donne pas toujours de bonne voire d'approximation (tout court) de la solution. L'étude de la convergence de tel algorithme est une part importante de la recherche internationale en Mathématiques (Analyse numérique et Équations aux dérivées partielles). Pour illustrer ces propos, on peut déjà remarqué que, sous certaine condition, l'algorithme donne des résultats "étranges" pour la dernière équation. Par exemple, dans le cas où $T = 10$ et $N = 500$ ou $N = 2000$ (voire les figures ci-dessous).





HP Voir : consistance et stabilité d'un schéma ou encore la condition Courant–Friedrichs–Lewy (CFL)

I.5 Méthode d'Euler implicite

Le principe est similaire à la méthode d'Euler explicite, sauf qu'on utilise le taux d'accroissement pour la dérivée en t_{n+1} et non t_n ce qui donne :

$$\boxed{y(t_{n+1}) \approx y(t_n) + hy'(t_{n+1})}$$

la méthode d'Euler implicite en découle : $y_0 = y(t_0)$ et $t_0 = t_0$ puis :

$$\begin{cases} t_{n+1} = t_n + \dfrac{T - t_0}{N} = t_n + h \\ y_{n+1} = y_n + hF(t_{n+1}, y_{n+1}) \end{cases}$$

l'expression de y_{n+1} n'est plus donnée directement. Un calcul supplémentaire est donc nécessaire pour trouver y_{n+1} (souvent, résoudre un système linéaire).

Définition 8.5: Méthode d'Euler implicite avec un pas régulier

Soit y une solution de l'équation :

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

sur $[t_0, T]$ avec F une fonction continue de $[t_0, T] \times \mathbb{R}$. Soit $N \in \mathbb{N}^*$ Si on note le pas de temps $h = \dfrac{T - t_0}{N}$, la **méthode d'Euler explicite** permet de construire un $(N+1)$-uplet $(y_k)_{k \in [\![0,N]\!]}$ approchant les valeurs du $(N+1)$-uplet $(y(t_0 + kh))_{k \in [\![0,N]\!]}$. La suite est définie par récurrence comme suit :

$$\begin{cases} t_{n+1} = t_n + h \\ y_{n+1} = y_n + hF(t_{n+1}, y_{n+1}) \end{cases}$$

Par exemple, dans le cas de l'exponentielle on a alors le schéma suivant : $y_0 = 1$ et $t_0 = 0$ puis :

$$\begin{cases} t_{n+1} = t_n + \dfrac{1}{N} = t_n + h \\ y_{n+1} = y_n + hy_{n+1} \end{cases}$$

ce qui revient à :

$$\begin{cases} t_{n+1} = t_n + \dfrac{1}{N} = t_n + h \\ y_{n+1} = \dfrac{y_n}{1 - h} \end{cases}$$

II Équations différentielles du second ordre

La méthode d'Euler peut être également utilisée pour résoudre des équations différentielles du second ordre. L'idée est de transformer cette équation en un système d'équations du premier ordre. On souhaite résoudre sur $[t_0, T]$ le problème dit de "Cauchy" suivant :

$$\begin{cases} y''(t) = F(t, y(t), y'(t)) \\ y(t_0) = y_0 \\ y'(t_0) = z_0 \end{cases}$$

où F est une fonction continue de $[t_0, T] \times \mathbb{R}^2$. On le transforme en un système de deux équations du premier ordre avec

$$\begin{cases} y'(t) = z(t) \\ z'(t) = F(t, y(t), z(t)) \\ y(t_0) = y_0 \\ z(t_0) = z_0 \end{cases}$$

On résout alors via la méthode d'Euler explicite vu précédemment, on contruit trois suites (t_n), (y_n) et (z_n) tel que :

$$\begin{cases} t_{n+1} = t_n + \dfrac{T - t_0}{N} = t_n + h \\ y_{n+1} = y_n + hz_n \\ z_{n+1} = z_n + hF(t_n, y_n, z_n) \end{cases}$$

II.1 Exemple : tension aux bornes du condensateur sur un circuit RLC

$$\begin{cases} y''(t) = \dfrac{E}{LC} - \dfrac{R}{L}y'(t) - \dfrac{1}{LC}y(t) \\ y(t_0) = 0 \\ y'(t_0) = 0 \end{cases}$$

avec $t_0 = 0, R = 1000\Omega, L = 0.1H, C = 10^{-9}F$ et $T = 0.001$. On rappelle que la solution analytique de ce problème est donnée par :

$$y : t \to E - Ee^{\frac{R}{2L}t}\left(\cos(\omega t) + \frac{R}{2L\omega}\sin(\omega t)\right)$$

avec $\omega_0 = \frac{1}{\sqrt{LC}}$, $Q = \frac{1}{R}\sqrt{\frac{L}{C}}$ et $\omega = \omega_0\sqrt{1 - \frac{1}{4Q^2}}$.

Exercice 8.6: Construire les suites (t_n), (y_n) et (z_n) permettant d'approcher les solutions de cette équation

Démonstration. Déterminons une solution numérique grâce à la méthode d'Euler explicite : on construit alors les suites (t_n), (y_n) et (z_n) vérifiant :

E

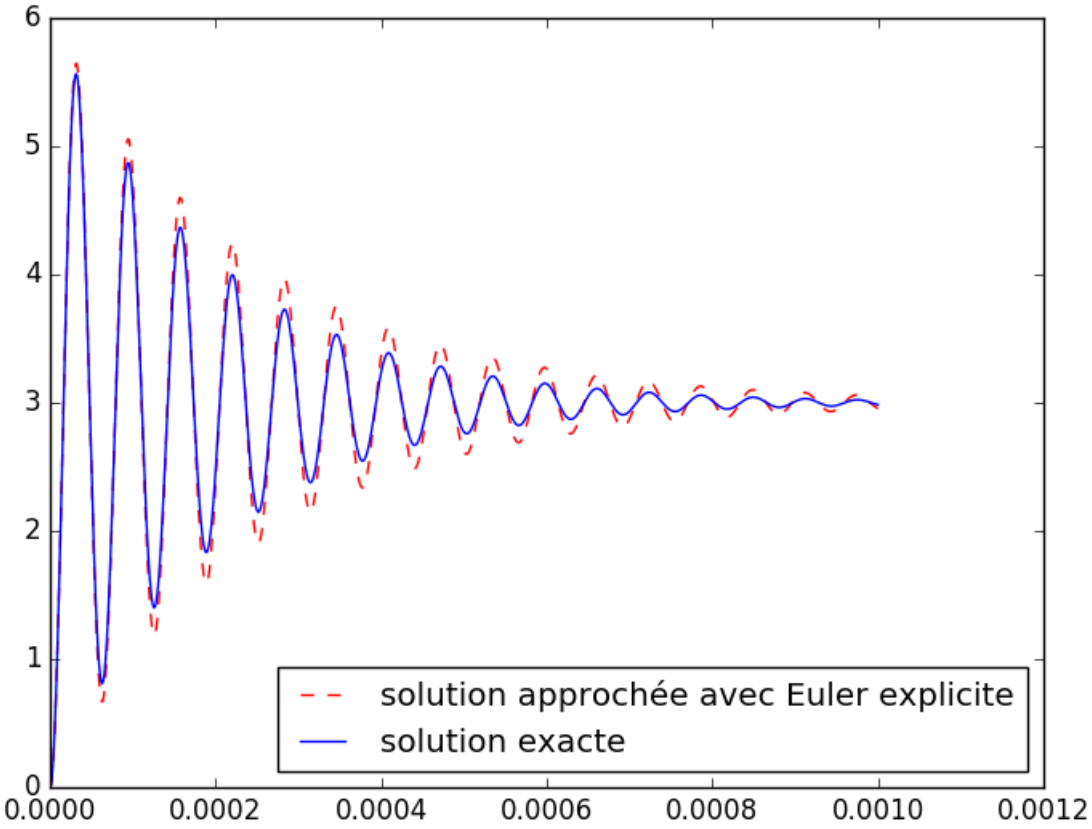
$$\begin{cases} t_{n+1} = t_n + \dfrac{T - t_0}{N} = t_n + h \\ y_{n+1} = y_n + hz_n \\ z_{n+1} = z_n + h\left(\dfrac{E}{LC} - \dfrac{R}{L}z_n - \dfrac{1}{LC}y_n\right) \end{cases}$$

□

On obtient le programme python suivant (voir page suivante)

méthode d'Euler explicite

```
import numpy as np
import matplotlib.pyplot as plt
E=3;R=1000;L=0.1;C=10**(-9)
t_0=0;y_0=0;z_0=0
T=0.001;N=5000;h=T/N

# constantes
om0=1/(np.sqrt(L*C))
Q=(1/R)*np.sqrt(L/C)
om=om0*np.sqrt(1-1/(4*Q**2))
list_y=[y_0]
list_z=[z_0]
list_t=[t_0]

y=y_0
t=t_0
z=z_0
for k in range(N):
    temp=y
    y=y+h*z
    z=z+h*((E/(L*C))-(R/L)*z-(1/(L*C))*temp)
    t=t+h
    list_z.append(z)
    list_y.append(y)
    list_t.append(t)

list_t=np.array(list_t)

sol=E-E*np.exp(-(R/(2*L))*list_t)*(np.cos(om*list_t)+(R/(2*L*om))*np.sin(om*list_t))

a,=plt.plot(list_t,list_y,'--r',label='solution approchée avec Euler explicite')
b,=plt.plot(list_t,sol,'-b',label='solution exacte')

plt.legend(handles=[a,b],loc=4)
plt.show()
```



II.2 Équation du pendule

Exercice 8.7: Expliquer la formule d'Euler explicite sur cet exemple et donner un algorithme permettant de calculer une solution approchée via cette méthode.

$$\theta'' + \frac{g}{l}\sin(\theta) = 0$$

avec $l = 50$ cm par exemple, et $\theta(0) = \pi/4$ et $\theta'(0) = 0$.

E

E

III Utilisation de odeint

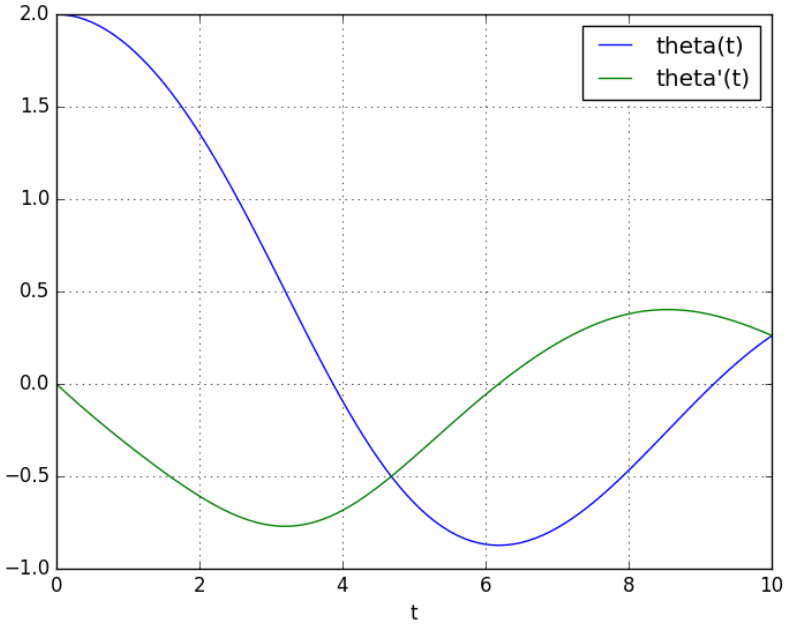
Odeint est une fonction de scypy qui permet de calculer une solution approchée d'une équation différentielle avec une méthode plus élaborée que Euler explicite ou implicite. En voici un exemple d'utilisation sur l'équation :

$$\theta''(t) + b\theta'(t) + c\sin(\theta(t)) = 0$$

avec $b = 0.25$, $c = 0.4$, $\theta(0) = 2$, $\theta'(0) = 0$, $t_0 = 0$ et $T = 10$.

```
def pend(y, t, b, c): #fonction renvoie le vecteur [theta',theta''] en fonction
        theta, thetaprim = y   #de  [theta,theta']
        dydt = [thetaprim, -b*thetaprim - c*np.sin(theta)]
        return dydt

b =0.25 #paramètres
c=0.4
y0=[2, 0.0] # listes des valeurs initiales de theta et theta'
t = np.linspace(0, 10, 101) #répartitions des points représentants la solution
# t_0=0, T=10, N+1=101
from scipy.integrate import odeint
sol = odeint(pend, y0, t, args=(b, c)) #calcule la solution

import matplotlib.pyplot as plt
plt.plot(t, sol[:, 0], 'b', label='theta(t)') #sol est un tableau
# la première colonne contient theta
```

```
plt.plot(t, sol[:, 1], 'g', label="theta'(t)") #la seconde theta'
plt.legend(loc='best')
plt.xlabel('t')
plt.grid() # permet de faire une grille
plt.show()
```



Les arguments à donner la fonction `odeint` sont les suivants (et dans l'ordre) :

- La fonction associée au problème de Cauchy `F(y,t)`. Attention, la variable y doit être avant la variable t. Il est possible de rajouter des arguments à F, il faut alors le spécifier à la fonction `odeint` comme dans l'exemple ci-dessus.
- La liste des conditions initiales `y0`
- La liste des temps pour l'extrapolation `t`

IV Fonctions Euler explicite sur Python

De manière générale, on pourra réécrire et utiliser les fonctions python suivante en TP :

La fonction suivante prend en argument la fonction F, les réels t_0, y_0, T et l'entier N et renvoie le $(N+1)$-uplet $(y_k)_{k\in[\![0,N]\!]}$ approchant les valeurs du $(N+1)$-uplet $(y(t_0+kh))_{k\in[\![0,N]\!]}$ et le $(N+1)$-uplet des temps $(t_k)_{k\in[\![0,N]\!]}$. On rappelle que la suite est définie par récurrence comme suit :

$$\begin{cases} t_{n+1} = t_n + h \\ y_{n+1} = y_n + hF(t_n, y_n) \end{cases}$$

Fonction Euler Explicite

```
def EulerExp(F, t_0, y_0, T, N):
    """
    Méthode d'Euler explicite pour résoudre une équation différentielle d'ordre 1.

    Arguments:
    - F: fonction qui prend comme arguments t et y et
            retourne la dérivée de y par rapport à t
    - t_0: temps initial
    - y_0: valeur initiale de la solution
    - T: temps final
    - N: nombre de pas

    Retourne:
    - list_t: liste des temps
    - list_y: liste des solutions
    """
    list_y = [y_0]
    list_t = [t_0]
    h = (T - t_0) / N
    y = y_0
    t = t_0
    for k in range(N):
        y = y + h * F(t, y)
        t = t + h
        list_y.append(y)  # ou list_y=list_y+[y]
        list_t.append(t)  # ou list_t=list_t+[t]
    return list_t, list_y
```

Un exemple d'utilisation lié à l'équation :

$$\begin{cases} y'(t) = \cos(ty(t)) \\ y(0) = 0 \end{cases}$$

Exemple

```
from math import cos
import numpy as np
import matplotlib.pyplot as plt
t_0=0;y_0=0;T=10;N=2000;h=T/N

def Fonc(t,y):
    return np.cos(t*y)

list_t,list_y=EulerExp(Fonc,t_0,y_0,T,N)

c,=plt.plot(list_t,list_y,'-',label='Euler explicite '+str(N)+' points')
plt.legend(handles=[c],loc=1)
plt.show()
```

Attention 8.8

La fonction donnée en argument à la fonction `EulerExp` doit avoir deux arguments `t,x` même si la fonction dépend que de x. Par exemple, pour l'équation différentielle $y' = y$, la fonction a donnée en argument est :

Exemple

```
def Fonc(t,y):
    return y
```

IV.1 Pour les équations d'ordre 2

Euler explicite pour l'ordre 2

```
def EulerExp2(F, t_0, y_0, z_0, T, N):
    """
    Calcule la solution approchée d'une équation différentielle d'ordre 2
    en utilisant la méthode d'Euler explicite.

    Arguments :
    F : fonction dérivée de y'' = F(t, y, z) qui représente l'équation différentielle
    t_0 : temps initial
    y_0 : valeur initiale de y
    z_0 : valeur initiale de y' (la dérivée de y par rapport au temps)
    T : temps final jusqu'auquel la solution est calculée
    N : nombre de pas de discrétisation pour l'algorithme d'Euler

    Retourne :
    list_t : liste des temps discrétisés
    list_y : liste des valeurs de la solution y
    list_z : liste des valeurs de la dérivée de la solution z = y'

    Remarque :
    La fonction retourne trois listes : les temps, la solution y et la dérivée z.
    """
    # Initialisation des listes pour stocker les valeurs
    list_y = [y_0]
    list_z = [z_0]
    list_t = [t_0]

    # Calcul du pas de temps
    h = (T - t_0) / N

    # Initialisation des variables
    y = y_0
    t = t_0
    z = z_0

    # Boucle principale de l'algorithme d'Euler
    for k in range(N):
        # Stockage temporaire de la valeur précédente de y
        temp = y
```

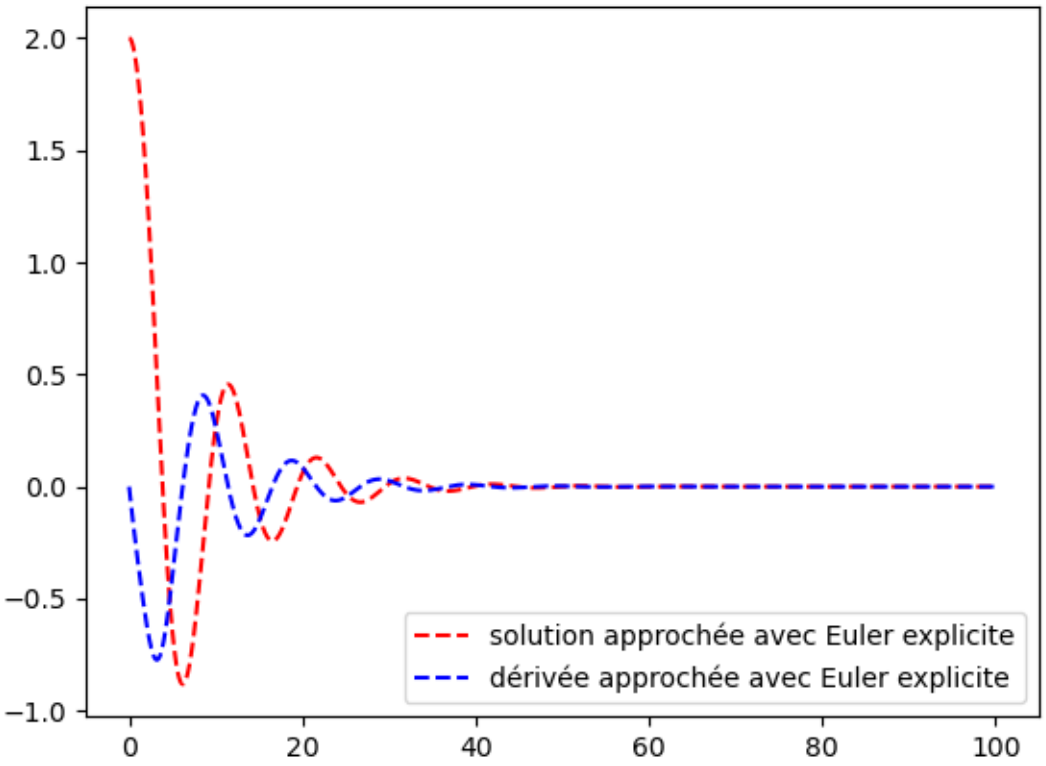
```
        # Mise à jour de y et z à l'aide de la méthode d'Euler
        y = y + h * z
        z = z + h * F(t, temp, z)

        # Mise à jour du temps
        t = t + h

        # Ajout des valeurs calculées aux listes
        list_z.append(z)
        list_y.append(y)
        list_t.append(t)

    return list_t, list_y, list_z
```

Voici un exemple d'utilisation pour

$$\theta''(t) + b\theta'(t) + c\sin(\theta(t)) = 0$$

avec $b = 0.25$, $c = 0.4$, $\theta(0) = 2$, $\theta'(0) = 0$, $t_0 = 0$ et $T = 10$.
Ici, $F(t, y, z) = -bz - c\sin(y)$ ne dépend pas de t.

```
import numpy as np
import matplotlib.pyplot as plt
b=0.25;c=0.4;y_0=2;z_0=0;t_0=0
T=100; N=10000; h=(T-t_0)/N

def Fonc(t,y,z):
    return -b*z-c*np.sin(y)

list_t,list_y,list_z=EulerExp2(Fonc,t_0,y_0,z_0,T,N)

a,=plt.plot(list_t,list_y,'--r',label='solution approchée avec Euler explicite')
b,=plt.plot(list_t,list_z,'--b',label='dérivée approchée avec Euler explicite')
plt.legend(handles=[a,b],loc=4)
plt.show()
```


2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
0
20
40
60
80
100
solution approchée avec Euler explicite
dérivée approchée avec Euler explicite