Méthodes numériques de résolution

d'une ou d'un système d'équations différentielles

---

1. Problème de Cauchy :

Un problème de Cauchy est un problème constitué d'une équation différentielle dont on recherche une solution vérifiant une certaine condition initiale. Cette condition peut prendre plusieurs formes selon la nature de l'équation différentielle. Pour une condition initiale adaptée à la forme de l'équation différentielle, le théorème de Cauchy-Lipschitz assure l'existence et l'unicité d'une solution au problème de Cauchy.

Autrement dit, le problème de Cauchy consiste à trouver les fonctions Y de $[0,T] \to \mathbb{R}^N$, telles que :

$$\begin{cases} \dfrac{dY}{dt} = F(t,Y) \\ Y(t_0) = Y_0 \end{cases}$$

Où $t_0 \in [0,T]$et $Y_0 \in \mathbb{R}^N$ sont des données. T étant la durée de la simulation numérique.

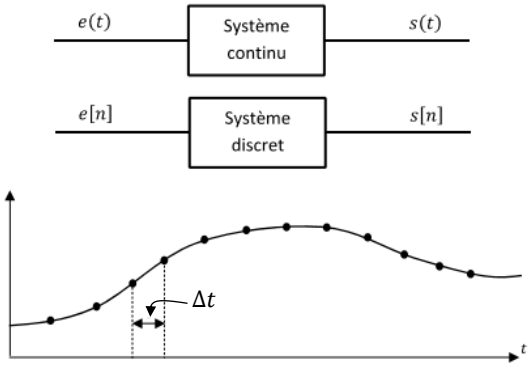
2. Résolutions numériques :

La résolution numérique consiste à retrouver la courbe d'évolution de la solution Y(t), en calculant un certain nombre de points sur un intervalle de temps donné. Cette méthode met donc en œuvre ce qu'il est commun d'appeler un échantillonnage du temps.

2.1 Discrétisation des problèmes :

Remarque : Dans bon nombre de systèmes, les variables sont échantillonnées et on ne connaît qu'une liste de valeurs à différents temps. On parle de variables discrètes.

L'objectif est d'obtenir une solution approchée au problème de Cauchy pour une discrétisation temporelle de l'intervalle donné.



Cette figure représente un signal continu en fonction du temps sur lequel ont été prises des valeurs à différents instants espacés d'un temps Δt supposé ici constant et appelé « période d'échantillonnage ou pas de temps ».

2.2. Schémas d'intégration :

D'un point de vue mathématique, l'intégration de l'équation différentielle entre deux pas de temps de l'échantillonnage t_i et t_{i+1}conduit à :

$$\int_{t_i}^{t_{i+1}} \frac{dY}{dt} = \int_{t_i}^{t_{i+1}} F(t,Y)dt \text{ qui se simplifie en } Y_{t_{i+1}} = Y_{t_i} + \int_{t_i}^{t_{i+1}} F(t,Y)dt$$

Le but est donc d'aborder le calcul général de l'intégrale :

$$I = \int_{t_i}^{t_{i+1}} F(t,Y)dt$$

L'idée principale est de trouver des méthodes numériques qui permettent de calculer rapidement une valeur approchée $\tilde{I}$ de l'intégrale I à calculer :

$$\tilde{I} \approx I$$

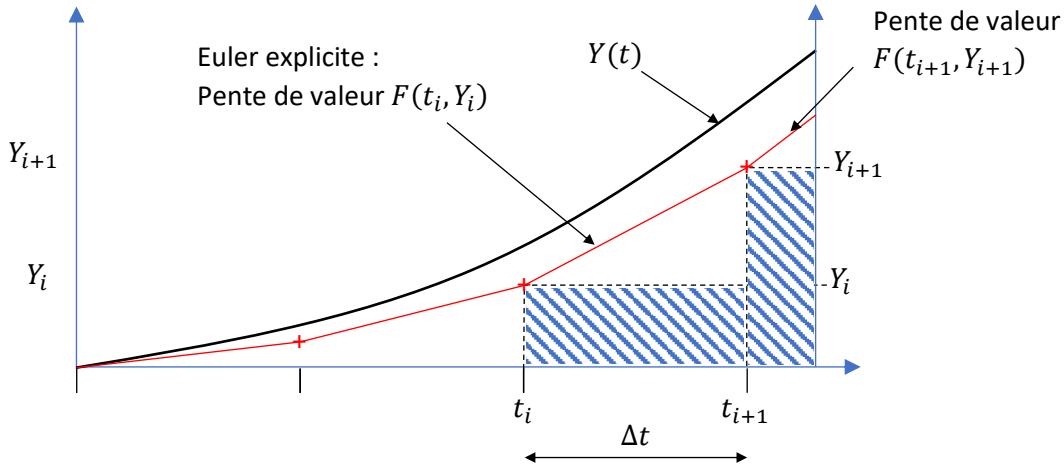
2.2.1. Méthode d'Euler explicite :

Dans ce cas, la façon d'approximer l'intégrale est de réaliser une méthode des rectangles à gauche.

Ainsi on utilise l'approximation :

$$I = \int_{t_i}^{t_{i+1}} F(t,Y)dt \approx \Delta t \cdot F(t_i, Y_i) = \tilde{I}$$

Cette approximation est d'autant meilleure que la largeur Δt des rectangles tend vers 0,

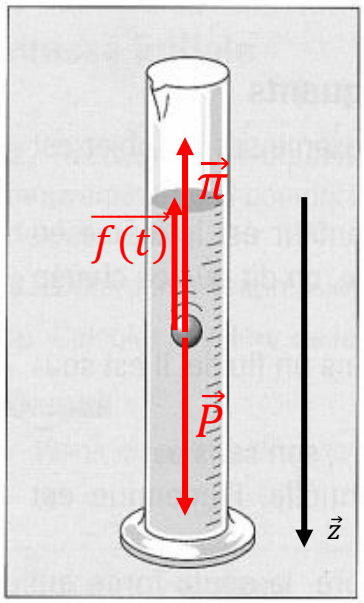


La relation de récurrence générale à utiliser dans ce cas est :

$$Y_{t_{i+1}} = Y_{t_i} + \Delta t \cdot F(t_i, Y_i)$$

Le problème se ramène alors à un calcul itératif, facile à automatiser à l'aide d'un programme informatique.

2.2.2 Exemple de résolution numérique : Chute d'une bille d'acier dans une éprouvette remplie d'huile.



Le principe fondamental appliqué à la bille s'écrit :

$$\vec{P} + \vec{\pi} + \overrightarrow{f(t)} = m \cdot \frac{d\overrightarrow{v(t)}}{dt} \quad (1)$$

Avec :

- $\vec{P} = m \cdot \vec{g}$
- $\vec{\pi} = \rho_{huile} \cdot Vol_{bille} \cdot \vec{g}$
- $\overrightarrow{f(t)} = -\eta \cdot \overrightarrow{v(t)}$

Données :

- Masse de la bille : $m \approx 4 \cdot 10^{-3}\ kg$
- Volume de la bille : $Vol_{bille} = 5{,}2 \cdot 10^{-7}\ m^3$
- Masse volumique de l'huile (20°C) : $\rho_{huile} = 920\ kg \cdot m^{-3}$
- Viscosité de l'huile (20°C) : $\eta = 0.39\ Pa.s$

En projetant l'équation (1) sur $\vec{z}$ il vient :

$$m \cdot g - \rho_{huile} \cdot Vol_{bille} \cdot g - \eta \cdot v(t) = m \cdot \frac{dv(t)}{dt}$$

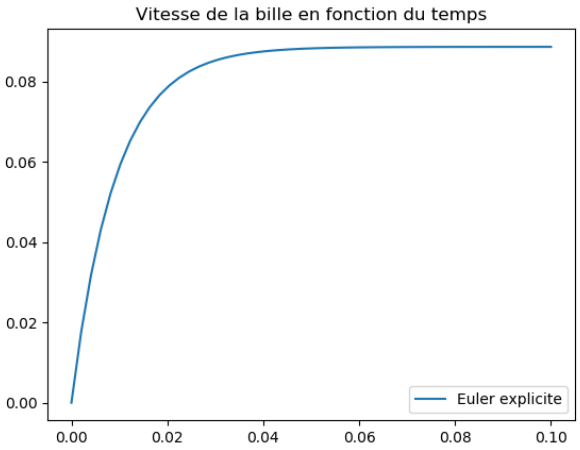
La bille est lancée sans vitesse initiale : $v(t_0) = v_0 = 0$

Mise en forme contextuelle du problème de Cauchy :

$$\begin{cases} \frac{dY}{dt} = F(t,Y) \\ Y(t_0) = Y_0 \end{cases} \Leftrightarrow \begin{cases} \frac{dV}{dt} = F(t,V) = \frac{dV(t)}{dt} = -\frac{\eta}{m} \cdot V(t) + g \cdot \left(1 - \frac{\rho_{huile} \cdot Vol_{bille}}{m}\right) \\ V(t_0) = V_0 = 0 \end{cases}$$

Ce qui donne le script python suivant :

```
# Créé par admin.prepa, le 19/09/2022 en Python 3.7
import matplotlib.pyplot as plt
import numpy as np

#------Initialisation------#
V0=0
N=50 #nombre de points de calcul
t0,tf=0,0.1 #temps initial et temps final

#---Déclaration de la fonction différentielle à intégrer---#
def F(V) :
    #constantes de l'expérience
    g=9.81 #accélération de la pesanteur [m/s2]
    n=0.39 #viscosité dynamique [Pa/s]
    m=4e-3 #masse de la bille [kg]
    ro=920 #masse volumique de la bille [kg/m3]
    Vol=5.2e-7 #volume de la bille [m3]
    #équation différentielle
    dv=-(n/m)*V+g*(1-(ro*Vol)/m)
    return dv

#---Mise en oeuvre d'Euler explicite---#
#---Déclaration de la fonction Euler---#
def euler(F,t0,tf,N):
    valeurs_V=[V0] # initialisation de la liste des vitesses
    valeurs_t=[t0] #initialisation de la liste temporelle
    V=V0 #initialisation de l'incrémentation de la liste des vitesses
    t=t0 #initialisation de l'incrémentation dela liste temporelle
    dt=(tf-t0)/(N-1) # pas temporel d'intégration
    for i in range(N-1) :
        t=t+dt
        V=V+dt*F(V)
        valeurs_t.append(t)
        valeurs_V.append(V)
    return valeurs_t, valeurs_V

#---Appel de la fonction Euler---#
valeurs_t,valeurs_V =euler(F,t0,tf,N)

#------Affichage------#
plt.plot(valeurs_t,valeurs_V,label="Euler explicite")
plt.legend(loc='lower right')
plt.title("Vitesse de la bille en fonction du temps")
plt.show()
```



Intégration de

$$\begin{cases} \frac{dV}{dt} = F(t,V) = \frac{dV(t)}{dt} = -\frac{\eta}{m} \cdot V(t) + g \cdot \left(1 - \frac{\rho_{huile} \cdot Vol_{bille}}{m}\right) \\ v(t_0) = v_0 = 0 \end{cases}$$

Schéma Euler explicite

$$\Delta t = 2e-3\ s\ , T = t_{max} = 0{,}1\ s$$

2.2.3 Autres schémas d'intégration :

La méthode d'Euler explicite ne convient pas à tous les types d'équations différentielles. Parfois le schéma numérique donne des résultats très éloignés des solutions réelles. D'autres méthodes sont en pratiques utilisées. En termes de schémas explicites, la méthode de Runge-Kutta à l'ordre 4 (RK4) est souvent utilisée.

Concernant le choix du pas de temps, il obéit à un compromis entre le temps de calcul, la stabilité et le stockage en mémoire. Le choix de ce pas de temps dépendra essentiellement de la dynamique du système à représenter : si on veut représenter une dynamique à 10 kHz, il faudrait prendre un pas de temps de l'ordre de $10^{-5}s$ (10 points par période). Le choix du pas de temps est souvent fait de manière empirique par expérience de l'ingénieur.

Remarque 1 :

La résolution numérique des équations différentielles est implantée dans python. Il s'agit entre autres des fonctions ***odeint***, ***solve_ivp***, du module ***integrate*** de la librairie ***Scipy***. Il convient donc de la charger.

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

ou

```
from scipy.integrate import solve_ivp
```

Remarque 2 : Attention, il faut bien respecter la syntaxe pour la fonction F() qui décrit l'équation différentielle. Dans le cas de l'utilisation de la méthode ***odeint,*** il est nécessaire d'inverser les arguments par rapport à l'écriture mathématique habituelle. C'est-à-dire :

Dans le script, il faut écrire def F(Y,t) plutôt que def F(t,Y)

Comparaison des scripts Euler explicite VS odeint :

Intégration avec la méthode d'Euler explicite	Intégration avec la fonction *odeint* du module integrate de Scipy

```python
#------Initialisation------#
V0=0
N=50 #nombre de points de calcul
t0,tf=0,0.1 #temps initial et temps final

#---Déclaration des fonctions différentielles à intégrer---#
def F(V,t) :
    #constantes de l'expérience
    g=9.81 #accélération de la pesanteur [m/s2]
    n=0.39 #viscosité dynamique [Pa/s]
    m=4e-3 #masse de la bille [kg]
    ro=920 #masse volumique de la bille [kg/m3]
    Vol=5.2e-7 #volume de la bille [m3]

    #équation différentielle
    dv=-(n/m)*V+g*(1-(ro*Vol)/m)
    return dv

#---Mise en oeuvre d'Euler explicite---#
#---Déclaration de la fonction Euler---#
def euler(F,t0,tf,N):

    valeurs_V=[V0] # initialisation de la liste des vitesses
    valeurs_t=[t0] #initialisation de la liste temporelle
    V=V0 #initialisation de l'incrémentation de la liste des vitesses
    t=t0 #initialisation de l'incrémentation de la liste temporelle
    dt=(tf-t0)/(N) # pas temporel d'intégration

    for i in range(N-1) :# creation de deux listes de 50 valeurs ,
valeur initiale comprise
        t=t+dt
        V=V+dt*F(V,t)
        valeurs_t.append(t)
        valeurs_V.append(V)
    return valeurs_t, valeurs_V

#---Appel de la fonction Euler---#
valeurs_t,valeurs_V =euler(F,t0,tf,N)

#------Affichage------#
plt.plot(valeurs_t,valeurs_V,label="Euler explicite")
plt.legend(loc='lower right')
plt.title("Vitesse de la bille en fonction du temps")

plt.show()
```

```python
#------Initialisation------#
V0=0
N=50 #nombre de points de calcul
t0,tf=0,0.1 #temps initial et temps final

#échantillon de temps
dt=tf-t0/N

def F(V,t) :
    #constantes de l'expérience
    g=9.81 #accélération de la pesanteur [m/s2]
    n=0.39 #viscosité dynamique [Pa/s]
    m=4e-3 #masse de la bille [kg]
    ro=920 #masse volumique de la bille [kg/m3]
    Vol=5.2e-7 #volume de la bille [m3]

    #équation différentielle
    dv=-(n/m)*V+g*(1-(ro*Vol)/m)
    return dv

#---Mise en œuvre de la fonction odeint---#
t=np.linspace(t0,tf,N) # t : tableau 1D
valeurs_V=odeint(F,V0,t)

plt.plot(t,valeurs_V,label="scypi.odeint")
plt.legend(loc='lower right')
plt.title("Vitesse de la bille en fonction du temps")

plt.show()
```

2.3 Système d'équations différentielles :

Lorsque les équations différentielles sont d'ordre $n > 1$, il est possible de transformer l'équation différentielle en un système de n équations différentielles du premier ordre.

2.3.1 Méthode d'Euler vectorielle :

La méthode explicite de résolution des équations différentielles scalaires du premier ordre s'appliquent aux systèmes :

$$\frac{d\overrightarrow{Y(t)}}{dt} = \vec{F}(t, \overrightarrow{Y(t)})$$

Où $\overrightarrow{Y(t)}$ est appelé vecteur d'état qui s'exprime dans le cas d'une équation différentielle d'ordre n par :

$$\overrightarrow{Y(t)} = \begin{bmatrix} y(t) \\ \frac{dy(t)}{dt} \\ \vdots \\ \frac{d^{n-1}y(t)}{dt^{n-1}} \end{bmatrix}$$

Les composantes du vecteur d'état $\overrightarrow{Y(t)}$ caractérisent l'état du système à un instant donné.

La relation de récurrence générale à utiliser dans ce cas est :

$$\vec{Y}_{t_{i+1}} = \overrightarrow{Y_{t_i}} + \Delta t \cdot \vec{F}\left(t_i, \overrightarrow{Y_i}\right)$$

2.3.2 Exemple de résolution numérique : système masse ressort :

Un solide de masse m relié à un bâti fixe par un ressort de raideur k est écarté de sa position d'équilibre x_{libre}. Le solide en mouvement frotte sur un plan lié au bâti. Le frottement (de coefficient λ) est de type « frottement fluide » c'est-à-dire que la force de frottement exercée sur le solide par le bâti augmente proportionnellement à la vitesse de déplacement du solide.

Le principe fondamental de la dynamique appliqué au solide donne :

$$m \cdot \frac{d^2x(t)}{dt^2} = -k \cdot (x(t) - x_{libre}) - \lambda \cdot \frac{dx(t)}{dt}$$

$$\frac{d^2x(t)}{dt^2} + \frac{\lambda}{m} \cdot \frac{dx(t)}{dt} + \frac{k}{m} \cdot (x(t) - x_{libre}) = 0$$

On pose :

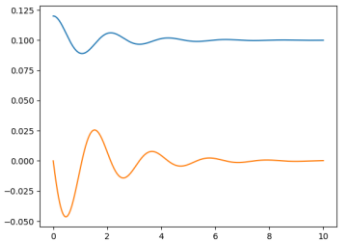
$\omega_n = \sqrt{\frac{k}{m}} = 3$ rad/s ; $a = \frac{\lambda}{2 \cdot \sqrt{k \cdot m}} = 0{,}2$; $x_{libre} = 0.1\ m$ (longueur libre du ressort)

Cette équation du *second* ordre peut se ramener au système de *deux* d'équations différentielles du premier ordre suivant :

$$\begin{cases} \dfrac{dx(t)}{dt} = v(t) \\ \dfrac{dv(t)}{dt} = -\omega_n^2 \cdot (x(t) - x_{libre}) - 2 \cdot a \cdot \omega_n \cdot v(t) \\ x_0 = x(t_0) = 0{,}12\ ;\ v_0 = v(t_0) = 0 \end{cases}$$

Ce qui donne le script python suivant :

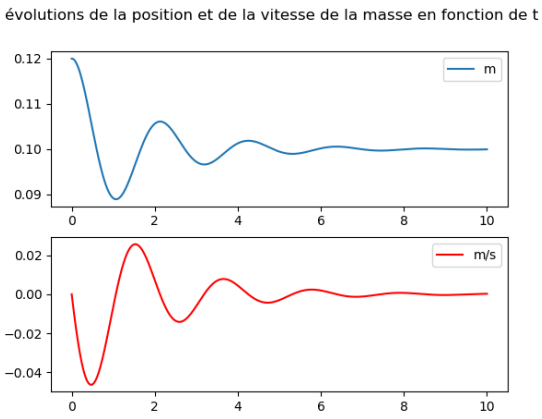
```
import numpy as np
import matplotlib.pyplot as plt

#------Initialisation------#
xl=0.1 #position de la masse au repos (longueur libre)
x0,v0=0.12,0 #affectation des conditions initiales
Y0=[x0,v0] #initialisation du vecteur d'état Y, traité sous forme de matrice
N=1000  #nombre de points de calcul
t0,tf=0,10 #temps initial et temps final

#---Déclaration des fonctions différentielles à intégrer---#
def F(Y) :
    #constantes
    wn=3
    a=0.2

    #attribution des rangs de la matrice
    x=Y[0]
    v=Y[1]
    #équations différentielles
    dx=v
    dv=-wn**2*(x-xl)-2*a*wn*v
    return np.array([dx,dv])

#---Mise en oeuvre d'Euler explicite---#
#---Déclaration de la fonction Euler explicite
def euler_vect(F,t0,tf,Y0,N) :
    valeurs_Y=[Y0]#initialisation de la liste de valeurs du vecteur d'état Y
    valeurs_t=[t0]#initialisation de la liste temporelle
    Y=Y0 #initialisation de l'incrementation du vecteur d'etat Y
    t=t0  #initialisation de l'incrementation temporelle
    dt=(tf-t0)/N # pas temporel d'intégration
    for i in range(N-1) :#constitution pas à pas des listes t et Y

        t=t+dt#incrementation temporelle
        Y=Y+dt*F(Y)#incrementation des valeurs du vecteur d'état (Euler
explicite)
        valeurs_t.append(t)#ajout de la nelle valeur dans la liste temporelle
        valeurs_Y.append(Y)#ajout des nouvelles valeurs dans la listes associée
au vecteur d'état

    return np.array(valeurs_t), np.array(valeurs_Y)

#---Appel de la fonction Euler---#
valeurs_t,valeurs_Y=euler_vect(F,t0,tf,Y0,N)

#------Affichage------#
plt.plot(valeurs_t,valeurs_Y)
plt.show()
```



Aménagement de l'affichage :

Il est possible de dissocier les deux graphiques en modifiant la partie « affichage » du script :

```
#------dissociation des affichages------#
valeurs_x=[row[0] for row in valeurs_Y]  # création
d'une liste de valeurs de positions à partir de Y
valeurs_v=[row[1] for row in valeurs_Y]  # création
d'une liste de valeurs de vitesse à partir de Y

# Affichage de deux subplot  "position" et "vitesse"
dans la fenetre d'affichage "figure"
figure,(position,vitesse)=plt.subplots(2,1) #
figure.suptitle('évolutions de la position et de la
vitesse de la masse en fonction de t')


position.plot(valeurs_t,valeurs_x,label="m")
vitesse.plot(valeurs_t,valeurs_v,
label="m/s",color="r")
position.legend()
vitesse.legend()
plt.show()
```



2.3.3) Equations différentielles couplées :

Voir TD moteur à courant continu.