Cours 12 – Méthode numérique d'Euler d'ordre 2

A / Transformation d'une équation différentielle d'ordre 2 en un système de deux équations différentielles d'ordre 1 : exemple du pendule simple non linéaire

On considère un pendule simple de masse m, de longueur ℓ qui va osciller d'arrière en avant à cause du champ de gravité de la Terre. La seconde loi Newton, le théorème du moment cinétique ou le théorème de l'énergie mécanique donne l'équation différentielle suivante qui gouverne l'évolution de l'angle $\theta(t)$ par rapport à la verticale (il faudra savoir établir cette équation par les trois méthodes !):

$$\frac{d^2\theta(t)}{dt^2}+\alpha\sin\theta(t)=0$$

Il s'agit d'une équation différentielle d'ordre 2 où $\alpha \equiv g/\ell$. La fonction inconnue à déterminer $\theta(t)$ nous donne accès à la position de la masse, à sa vitesse, à son accélération et à la tension de la corde. L'objectif de ce cours est de résoudre numériquement, par la méthode d'Euler, cette équation quand $\sin\theta=\sin\theta$!!, nous savons bien (j'espère) résoudre analytiquement cette équation quand $\theta \ll 1$, on retrouve alors l'équation différentielle d'un oscillateur harmonique. Ici **l'équation différentielle est non linéaire** et on ne peut pas la résoudre analytiquement, il faut donc adopter une approche numérique.

Pour ce faire, **une méthode importante est de transformer cette équation différentielle d'ordre 2 en deux équations différentielles d'ordre 1** afin de pouvoir utiliser simplement la méthode d'Euler pour les équations différentielles d'ordre 1 (cf. le cours d'informatique). Le prix à payer est que l'on se retrouve avec deux fonctions inconnues. En effet, en posant $\omega \equiv d\theta/dt$, la vitesse angulaire du pendule, on obtient le système d'équations différentielles avec deux fonctions inconnues suivant :

Exemple du pendule simple non linéaire

$$\underbrace{\frac{d^2\theta}{dt^2}+\alpha\sin\theta=0}_{\substack{\text{équation différentielle d'ordre 2}\\ \text{fonction inconnue } \theta(t)}} \Rightarrow \underbrace{\begin{cases}\dfrac{d\theta}{dt}=\omega\\ \dfrac{d\omega}{dt}=-\alpha\sin\theta\end{cases}}_{\substack{\text{2 équations différentielles d'ordre 1}\\ \text{2 fonctions inconnues } \theta(t) \text{ et } \omega(t)}}$$

Pour résoudre ce système, nous devons connaître les deux conditions initiales $\theta(t_0)=\theta_{\text{init}}$ et $\omega(t_0)=\omega_{\text{init}}$. Comme dans le cours précédent, on remplace **la dérivée à droite** par un taux d'accroissement fini, on pourra écrire avec $t_{k+1}=t_k+\delta t$: $\frac{d\theta}{dt}(t_k)\approx\frac{\theta_{k+1}-\theta_k}{\delta t}=\omega_k$ et $\frac{d\omega}{dt}(t_k)\approx\frac{\omega_{k+1}-\omega_k}{\delta t}=-\alpha\sin\theta_k$. Cela donne :

<u>Résolution numérique par Euler d'une équation différentielle d'ordre 2 : approche système</u>

<u>Initialisation</u> :

$\theta_0=\theta_{\text{init}}$, $\omega_0=\omega_{\text{init}}$

<u>Récurrence</u> (simple fois 2, car 2 équations différentielles d'ordre 1) :

A partir du pas δt, calculer les point (t_{k+1},θ_{k+1}) et (t_{k+1},ω_{k+1}) à partir des points précédents (t_k,ω_k) et (t_k,θ_k) en suivant la procédure suivante :

$$t_{k+1}=t_k+\delta t \text{ et } \begin{cases}\theta_{k+1}=\theta_k+\omega_k\,\delta t\\ \omega_{k+1}=\omega_k+(-\alpha\sin\theta_k)\delta t\end{cases}$$

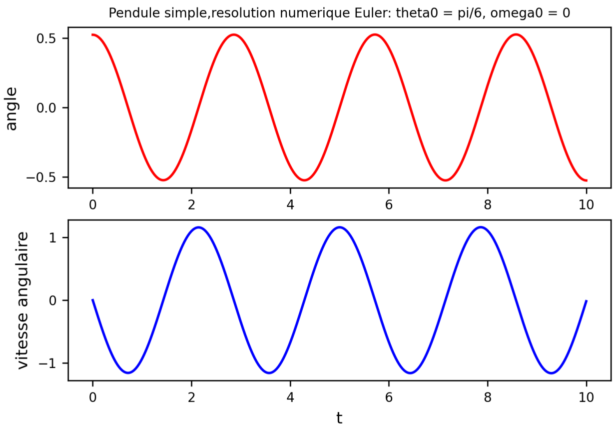
Voici un exemple de code pour résoudre numériquement le système d'équations différentielles précédent.

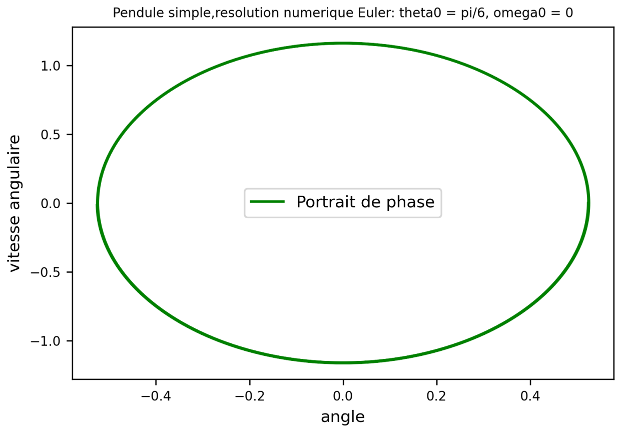
```
from math import *
import numpy as np
import matplotlib.pyplot as plt

#résolution par la méthode d'Euler
#T: durée de l'intervalle de résolution
#n: nombre de points de calcul
#theta0: valeur initiale de la vitesse angulaire
#omega0: valeur initiale de l'angle
#alpha: paramètre physique, ici g/l
def pendule(T,n,theta0,omega0,alpha):
        dt=T/(n-1)
        t=np.linspace(0,T,n)
        omega=np.zeros(n)
        theta=np.zeros(n)
        omega[0]=omega0
        theta[0]=theta0
        for k in range (n-1):
                theta[k+1]=theta[k]+(omega[k])*dt
                omega[k+1]=omega[k]+(-alpha*np.sin(theta[k]))*dt
        return theta, omega, t

#valeur des paramètres
n=50000
T=10
omega0=0
theta0=10*pi/180
alpha=5

#appel de la fonction euler
theta, omega, t = pendule(T,n,theta0,omega0,alpha)

#tracé des graphes
plt.subplot(2,1,1)
plt.xlabel('t',size=10)
plt.ylabel('angle',size=10)
plt.plot(t,theta, 'r-',markersize=3)
plt.title('Pendule simple,resolution numerique Euler: theta0=10°, omega0=0', size=8)
plt.xticks(size=8)
plt.yticks(size=8)

plt.subplot(2,1,2)
plt.xlabel('t',size=10)
plt.ylabel('vitesse angulaire',size=10)
plt.plot(t,omega, 'b-',markersize=3)
plt.xticks(size=8)
plt.yticks(size=8)
plt.show()

plt.plot(theta,omega, 'g-',markersize=3)
plt.xlabel('angle',size=10)
plt.ylabel('vitesse angulaire',size=10)
plt.title('Pendule simple,resolution numerique Euler: theta0=10°, omega0=0',size=8)
plt.legend(['Portrait de phase'],fontsize=10)
plt.xticks(size=8)
plt.yticks(size=8)
plt.show()
```

En plus de tracer l'évolution au cours du temps de θ et de ω, il est très intéressant et riche de tracer, dans l'étude des systèmes dynamiques, **le portrait de phase**, c'est à dire ω en fonction de θ. Voici les résultats que l'on obtient pour diverses conditions initiales.

Premier cas : $\theta_{\text{init}} = \dfrac{\pi}{6}$ et $\omega_{\text{init}} = 0$

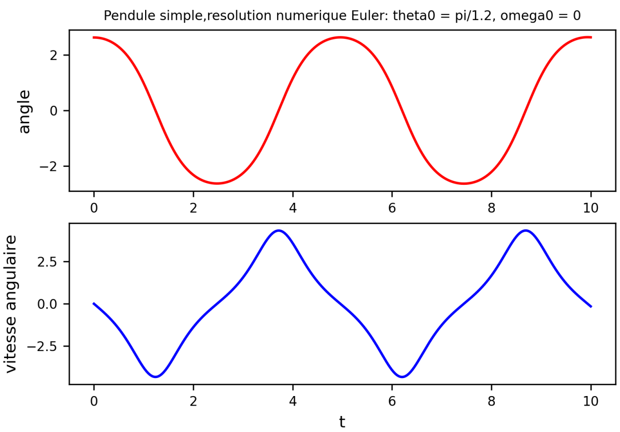
Dans ce cas là, on a, avec une bonne approximation, $\sin\theta \approx \theta$, les angles restent faibles et on retrouve le mouvement d'un oscillateur harmonique.

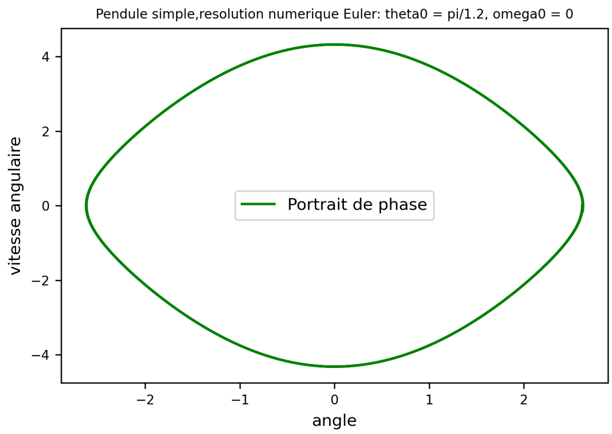




Deuxième cas : $\theta_{\text{init}} = \dfrac{\pi}{1.2}$ et $\omega_{\text{init}} = 0$

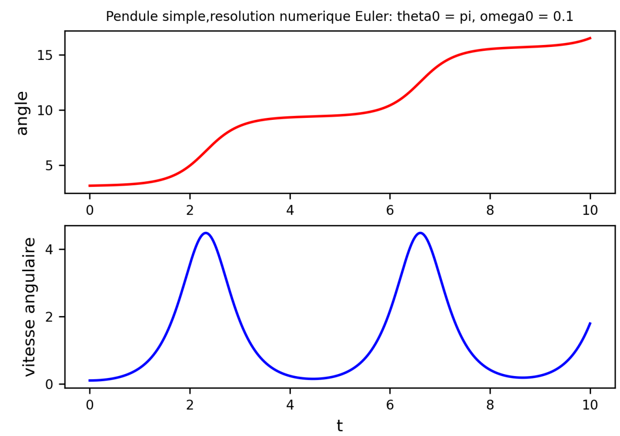
Cette fois, le pendule oscille de façon périodique avec de grands angles mais l'oscillation n'est plus harmonique, $\sin\theta \neq \theta$.

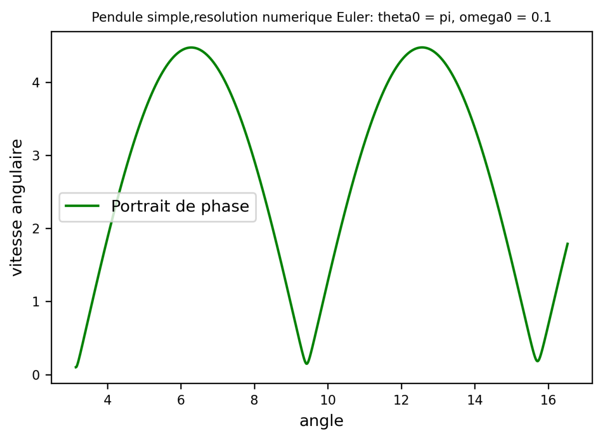




Troisième cas : $\theta_{\text{init}} = \pi$ et $\omega_{\text{init}} = 0.1$

On place le pendule à la verticale $\theta = \pi$ et on le pousse avec une petite vitesse angulaire. Le pendule va faire des tours !





Remarques: Il faut ici un δt très petit pour que la méthode d'Euler soit précise, elle est gourmande en temps de calcul. C'est une méthode simple mais pas très performante.

B / Résolution directe de l'équation différentielle d'ordre 2

Il est possible évidemment de résoudre numériquement directement l'équation différentielle d'ordre 2 en approximant la dérivée seconde comme cela a état vu dans le cours sur la dérivation numérique. Dans une approche dynamique (physique) des problèmes, il est plus intéressant de passer par un système d'équations différentielles d'ordre 1 mais dans d'autre cas, la résolution directe de l'équation différentielle d'ordre 2 est plus pertinente, en particulier dans un approche d'ingénierie.

On part de nouveau de l'exemple précédent sur la pendule simple non linéaire :

$$\frac{d^2\theta(t)}{dt^2}+\alpha\sin\theta(t)=0 \text{ avec les conditions initiales } \begin{cases}\theta(0)=\theta_{\text{init}}\\ \dfrac{d\theta}{dt}(0)\equiv\omega(0)=\omega_{\text{init}}\end{cases}$$

<u>Résolution numérique par Euler d'une équation différentielle d'ordre 2 : approche directe</u>

<u>Initialisation</u> :

$\theta_0=\theta_{\text{init}}$ connu et par la dérivée à droite $\omega_0=\omega_{\text{init}}\approx\dfrac{\theta_1-\theta_0}{\delta t}$ donc $\theta_1=\theta_0+\omega_0\delta t$ connu également.

<u>Récurrence</u> (double fois 1, car une équation différentielle d'ordre 2) :

$$\frac{d^2\theta}{dt^2}(t_k)\approx\frac{\theta_{k+2}-2\theta_{k+1}+\theta_k}{\delta t^2}=-\alpha\sin\theta_k$$

On isole alors θ_{k+2} pour avoir l'équation :

$$t_{k+1}=t_k+\delta t \text{ et } \theta_{k+2}\approx 2\theta_{k+1}-\theta_k+(-\alpha\sin\theta_k)\,\delta t^2$$

Voici un exemple de code pour résoudre numériquement directement l'équation différentielle d'ordre 2. Comparez bien avec le précédent code pour voir ce qui est différent et ce qui est commun. Par exemple, ici, on ne peut pas récupérer le tableau des vitesses angulaires ce qui est dommagable pour une approche dynamique du pendule.

```
def pendule(T,n,theta0,omega0,alpha):
    dt = T/(n-1)              # pas de temps = (tf-ti)/nb_intervalles
    t = np.linspace(0,T,n)   # on veut n points en sortie
    theta = np.zeros(n)      # déclaration du futur vecteur en sortie
    theta[0] = theta0        # CI en position
    theta[1] = theta0 + omega0*dt  # CI en vitesse
    for k in range(n-2):  # après 2 CI, il reste n-2 pts à calculer
        theta[k+2]=2*theta[k+1]-theta[k]+(-alpha*np.sin(theta[k]))*(dt**2)
    return theta, t
```