CN6 Résolution numérique d'une équation différentielle du 2d ordre non linéaire, effet des termes non linéaires

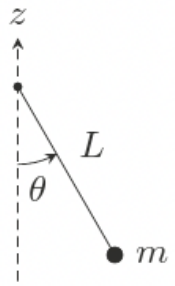
Compétence attendue par le programme 2021 : « à l'aide d'un langage de programmation résoudre numériquement une équation différentielle du deuxième ordre non-linéaire et faire apparaître l'effet des termes non-linéaires »

Capacités numériques travaillées avec Python :

- Savoir résoudre numériquement une équation en mettant en œuvre odeint ;
- Savoir tracer le résultat d'une résolution numérique
- Savoir superposer des courbes

But : nous allons résoudre l'équation différentielle du mouvement d'un pendule simple dans le cas général et dans le cas particulier des petites oscillations.

Présentation du système : Soit un peson ponctuel M de masse m suspendu par un fil de longueur L dans le champ de pesanteur terrestre g. la position du peson est repéré par rapport à l'angle θ que forme le fil par rapport à la verticale. On travaille dans le référentiel terrestre supposé galiléen.



Comme nous l'avons vu, l'application d'un théorème énergétique (TPC ou TPM), de la RFD ou du TMC/TSMC au peson ponctuel M de masse m permet d'obtenir l'équation du mouvement suivante :

$$\ddot{\theta} + \frac{g}{L}\sin\theta = 0 \qquad \text{équation E1}$$

A. Résolution numérique dans le cas général : tracé du graphe de θ=f(t)

Comme nous l'avons vu dans la fiche CN4, pour résoudre numériquement, de manière approchée, une équation différentielle du 1er ordre on peut utiliser : soit la méthode d'Euler (qui s'avère avoir tendance à diverger) soit mettre en œuvre la fonction odeint (qui utilise la méthode de Runge-Kutta, qui converge plus efficacement que la méthode d'Euler). **Nous allons donc utiliser la fonction odeint de module integrate de skipy.**

Pour cela, il faut transformer E1 qui est une équation différentielle du second ordre en un système de deux ED du 1er ordre.

$$\begin{cases} \frac{d\theta}{dt} = \dot{\theta} \\ \frac{d\dot{\theta}}{dt} = -\frac{g}{L}\sin\theta \end{cases} \qquad \text{système S1}$$

Les deux variables dépendant du temps sont donc $\dot{\theta}$ et θ. Les deux termes de droite de cette équation différentielle peuvent donc être réunies dans une liste $\left[\dot{\theta}, -\frac{g}{L}\sin\theta\right]$.

Fiche de Capacités numériques n°6

Pour pouvoir réaliser une résolution numérique, il faut connaitre les conditions initiales et fixer une durée.

Prenons par exemple les conditions initiales suivantes : peson lâché avec un angle de $\pi/4$ sans vitesse initiale.
Ainsi $\theta(0)=\pi/4$ rad et $\dot{\theta}(0) = 0$.
Fixons une durée de résolution de $\Delta t=5s$: nous aurons donc le mouvement du peson sur les 5 premières secondes de son mouvement.

Nous allons résoudre de manière approchée l'équation E1 à l'aide de **odeint** :

Pour cela nous allons définir un tableau où figurent les deux équations différentielles : dans la 1ere colonne, celle vérifiée par y[0] qui représente θ et dans la seconde, celle vérifiée par y[1] qui représente dθ/dt.

Le système d'équations s'écrit alors :

$$\begin{cases} \dfrac{dy[0]}{dt} = y[1] \\ \dfrac{dy[1]}{dt} = -\dfrac{g}{L}\sin(y[0]) \end{cases}$$

Code python

On commence par importer les bibliotheques nécessaires, à définir les constantes (données de l'énoncé, durée, échantillonnage…).

```
# importation des bibliothèques et définition des constantes
import numpy as np
from scipy.integrate import odeint
import matplotlib.pyplot as plt          # pour le tracé
N=100                                    # 100 points de discrétisation pour le tracé de la courbe
g=9.81                                   # intensité de pesanteur en m.s-2
L = 0.3                                  # longueur du fil en mètres
t0=0                                     # instant initial
tf=5                                     # instant final donc étude sur 5s
t=np.linspace(t0,tf,N)                   # intervalles de temps
```

Puis on créé une fonction afin de définir les deux termes « de droite » des systèmes d'équations à résoudre.

```
# création de la fonction
def E1(y,t) :
    theta = y[0]
    thetapoint = y[1]
    return np.array([thetapoint,-g/L*np.sin(theta)])
```

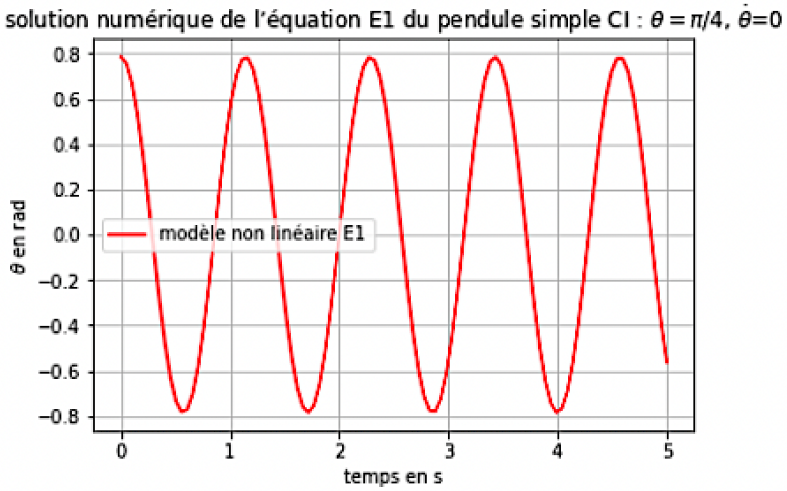
on résout pour les CI $\theta(0)=\pi/4$ rad et $\dot{\theta}(0) = 0$ et on ne récupère que les valeurs contenues dans la colonne 0 car cette colonne correspond au valeurs prises par θ (les valeurs prises par dθ/dt ne sont pas intéressantes pour tracer le graphe).

```
#résolution
y0=[np.pi/4,0]              # CI
sol_E1=odeint(E1,y0,t)      # on résout E1
Theta_E1=sol_E1[ :,0]       # on ne conserve que la 1er colonne du tableau , celle des valeurs prises par  theta.
```

On trace pour finir le graphe de la solution approchée :

```
#  tracé de la solution dans le cas général
plt.figure(1)
plt.title(r"solution numérique de l'équation E1 du pendule simple CI : $\theta=\pi/4$, $\dot{\theta}$=0")
plt.plot(t,Theta_E1,"-r", label="modèle non linéaire E1")   #tracé de l'évolution temporelle de theta pour E1
plt.grid(True)
plt.xlabel("temps en s")
plt.ylabel(r"$\theta$ en rad")
plt.legend()
plt.show()
```

remarque : les titres sont écrits de manière à faire apparaitre sur les titres la lettre grecque θ et sa dérivée $\dot{\theta}$, mais ce n'est évidemment pas une obligation.



B. Résolution numérique dans le cas particulier des petites oscillations

Si nous travaillons dans le cas particulier des petites oscillations, l'équation se linéarise ainsi :

$$\ddot{\theta} + \frac{g}{L}\theta = 0 \qquad \text{équation E2}$$

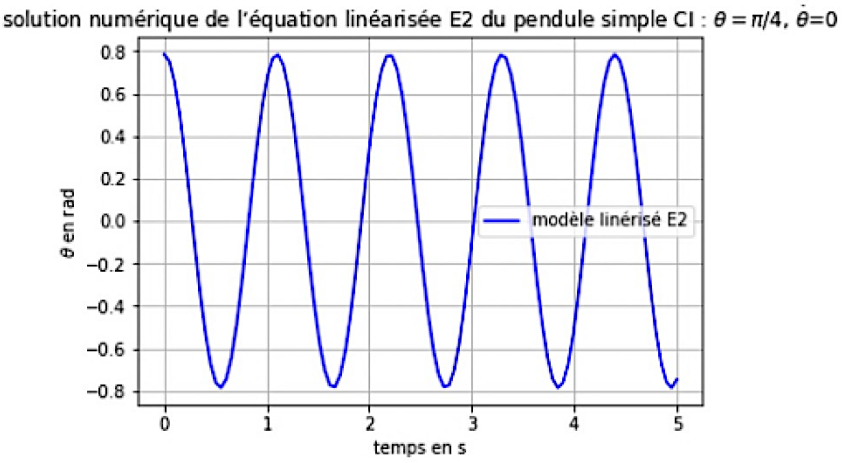
L'équation ainsi linéarisée peut là encore se décomposer en un système de 2 équations du 1er ordre :

$$\begin{cases} \frac{d\theta}{dt} = \dot{\theta} \\ \frac{d\dot{\theta}}{dt} = -\frac{g}{L}\theta \end{cases} \qquad \text{système S2}$$

Ecrivons le code python permettant de résoudre cette équation E2.

```
# importation des bibliothèques et définition des constantes
import numpy as np
from scipy.integrate import odeint
import matplotlib.pyplot as plt                 # pour le tracé
N=100                                           # 100 points de discrétisation pour le tracé de la courbe
g=9.81                                          # intensité de pesanteur en m.s-2
L = 0.3                                         # longueur du fil en mètres
t0=0                                            # instant initial
tf=5                                            # instant final donc étude sur 5s
t=np.linspace(t0,tf,N)                          # intervalles de temps
```

```
# création de la fonction
def E2(y,t) :
   theta = y[0]
   thetapoint = y[1]
   return np.array([thetapoint,-g/L*theta])

#résolution
y0=[np.pi/4,0]                  # CI
sol_E2=odeint(E2,y0,t)          # on résout E2
Theta_E2=sol_E2[ :,0]           # on ne conserve que la 1er colonne du tableau , celle des valeurs prises par  theta.


#  tracé de la solution associée à l'équation linéarisée
plt.figure(2)
plt.title(r"solution numérique de l'équation linéarisée E2 du pendule simple CI : $\theta=\pi/4$, $\dot{\theta}$=0")
plt.plot(t,Theta_E2,"-b", label="modèle linérisé E2")   #tracé de l'évolution temporelle de theta pour E2
plt.grid(True)
plt.xlabel("temps en s")
plt.ylabel(r"$\theta$ en rad")
plt.legend()
plt.show()
```

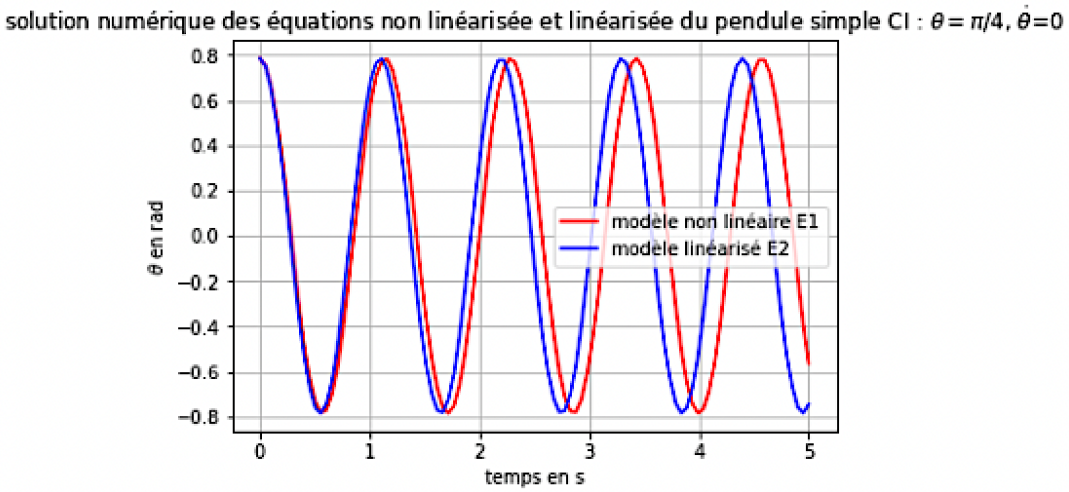


C. Comparaison des solutions numériques de E1 et E2 par superposition des graphes

```
# importation des bibliothèques et définition des constantes
import numpy as np
from scipy.integrate import odeint
import matplotlib.pyplot as plt          # pour le tracé
N=100                                    # 100 points de discrétisation pour le tracé de la courbe
g=9.81                                   # intensité de pesanteur en m.s-2
L = 0.3                                  # longueur du fil en mètres
t0=0                                     # instant initial
tf=5                                     # instant final donc étude sur 5s
t=np.linspace(t0,tf,N)                   # intervalles de temps

# création des fonctions associées à E1 et à E2
def E1(y,t) :
   theta = y[0]
   thetapoint = y[1]
   return np.array([thetapoint,-g/L*np.sin(theta)])
```

Fiche de Capacités numériques n°6

```
def E2(y,t) :
    theta = y[0]
    thetapoint = y[1]
    return np.array([thetapoint,-g/L*theta])

#résolution
y0=[np.pi/4,0]                  # CI
sol_E1=odeint(E1,y0,t)          # on résout E1
Theta_E1=sol_E1[ :,0]           # on ne conserve que la 1er colonne du tableau , celle des valeurs prises par  theta.
sol_E2=odeint(E2,y0,t)          # on résout E2
Theta_E2=sol_E2[ :,0]           # on ne conserve que la 1er colonne du tableau , celle des valeurs prises par  theta.

#  superposition des deux tracés
plt.figure(3)
plt.title(r"solution numérique des équations non linéarisée et linéarisée du pendule simple CI : $\theta=\pi/4$,
$\dot{\theta}$=0")
plt.plot(t,Theta_E1,"-r",label= "modèle non linéaire E1")   #tracé de l'évolution temporelle de theta pour E1
plt.plot(t,Theta_E2,"-b", label="modèle linéarisé E2")   #tracé de l'évolution temporelle de theta pour E2
plt.grid(True)
plt.xlabel("temps en s")
plt.ylabel(r"$\theta$ en rad")
plt.legend()
plt.show()
```



A vous de jouer :

1. Commenter la courbe ci-dessus. Quel modèle faut-il choisir pour décrire au mieux le comportement d'un pendule avec les conditions initiales $\theta(0)=\pi/4$ rad et $\dot{\theta}(0) = 0$?

2. Étudier maintenant les conditions initiales suivantes : $\theta(0)=\pi/20$ rad et $\dot{\theta}(0) = 0$. Adapter le programme faire une capture d'écran des 2 courbes superposées et commenter.

3. Rechercher par tâtonnement l'angle limite à partir duquel on ne peut raisonnablement plus parler d'isochronisme des oscillations pour le pendule simple.