Méthodes numériques

Utilisation de odeint our les équations différentielles du 2$^{\text{nd}}$ ordre

Objectif : résoudre une équation différentielle d'ordre 2 à l'aide de la fonction `odeint` de Python.

Précédemment, nous avons vu la méthode d'Euler explicite pour résoudre une équation différentielle d'ordre 1. Cette méthode n'est cependant pas toujours fiable. Nous allons ici utiliser la fonction `odeint`, de la librairie `scipy.integrate`. Elle permet de résoudre des équations différentielles de manière fiable et rapide.

De plus, nous nous intéressons à une équation **d'ordre 2**, comme celle du pendule simple :

$$\ddot{\theta} + \omega_0^2 \sin\theta = 0, \quad \text{soit aussi} \quad \boxed{\ddot{\theta} = -\omega_0^2 \sin\theta.} \tag{1}$$

a/ Reformulation de l'équation en une équation d'ordre 1

La fonction `odeint` a une syntaxe particulière, qui oblige à transformer l'équation d'ordre 2 sur θ, en une équation d'ordre 1 sur le couple $y = \begin{pmatrix} \theta \\ \dot{\theta} \end{pmatrix}$.

On parle parfois de vecteur à deux composantes pour désigner y :

– sa première composante est $y_0 = \theta$,

– sa seconde composante est $y_1 = \dot{\theta}$.

$\rightsquigarrow_1$ Dériver le couple y pour trouver l'équation différentielle qu'il suit.

$$\begin{aligned}
\frac{\mathrm{d}y}{\mathrm{d}t} &= \frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} \theta \\ \dot{\theta} \end{pmatrix} \\
&= \begin{pmatrix} \dot{\theta} \\ \ddot{\theta} \end{pmatrix} \\
&= \begin{pmatrix} \dot{\theta} \\ -\omega_0^2 \sin\theta \end{pmatrix} \\
&= \begin{pmatrix} y_1 \\ -\omega_0^2 \sin y_0 \end{pmatrix}
\end{aligned}$$

On arrive donc à une équation du premier ordre sur y :

$$\boxed{\frac{\mathrm{d}y}{\mathrm{d}t} = F(y) \quad \text{avec} \quad F(y) = \begin{pmatrix} y_1 \\ -\omega_0^2 \sin y_0 \end{pmatrix}.} \tag{2}$$

Bilan : on est passé de l'équation d'ordre 2 sur θ à l'équation ci-dessus, d'ordre 1 sur le couple $y = \begin{pmatrix} \theta \\ \dot\theta \end{pmatrix}$.

Autres exemples pour bien comprendre

(à faire sur feuille à part).

$\leadsto_2$ On considère l'équation du pendule, mais avec des frottements :

$$\ddot\theta = -\omega_0^2 \sin\theta - \alpha\dot\theta.$$

Avec la même démarche que ci-dessus, transformer cette équation d'ordre 2 sur θ en une équation d'ordre 1 sur le couple $y = \begin{pmatrix} \theta \\ \dot\theta \end{pmatrix}$ (donc en une équation du type $\frac{\mathrm{d}y}{\mathrm{d}t} = F(y)$).

$$\begin{aligned} \frac{\mathrm{d}y}{\mathrm{d}t} &= \frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} \theta \\ \dot\theta \end{pmatrix} \\ &= \begin{pmatrix} \dot\theta \\ \ddot\theta \end{pmatrix} \\ &= \begin{pmatrix} \dot\theta \\ -\omega_0^2 \sin\theta - \alpha\dot\theta \end{pmatrix} \\ &= \begin{pmatrix} y_1 \\ -\omega_0^2 \sin y_0 - \alpha\, y_1 \end{pmatrix} \end{aligned}$$

Donc on a $\boxed{\frac{\mathrm{d}y}{\mathrm{d}t} = F(y) \text{ avec } F(y) = \begin{pmatrix} y_1 \\ -\omega_0^2 \sin y_0 - \alpha\, y_1 \end{pmatrix}.}$

$\leadsto_3$ On considère l'équation suivie par la tension u aux bornes du condensateur dans un circuit RLC en régime sinusoïdal forcé :

$$\ddot u + \frac{\omega_0}{Q}\dot u + \omega_0^2\, u = \alpha\cos\omega t.$$

Avec la même démarche que ci-dessus, transformer cette équation d'ordre 2 sur u en une équation d'ordre 1 sur le couple $y = \begin{pmatrix} u \\ \dot u \end{pmatrix}$ (donc en une équation du type $\frac{\mathrm{d}y}{\mathrm{d}t} = F(y,t)$).

$$\begin{aligned}
\frac{\mathrm{d}y}{\mathrm{d}t} &= \frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} u \\ \dot{u} \end{pmatrix} \\
&= \begin{pmatrix} \dot{u} \\ \ddot{u} \end{pmatrix} \\
&= \begin{pmatrix} \dot{u} \\ -\frac{\omega_0}{Q}\dot{u} - \omega_0^2\, u + \alpha\cos\omega t \end{pmatrix} \\
&= \begin{pmatrix} y_1 \\ -\frac{\omega_0}{Q}y_1 - \omega_0^2\, y_0 + \alpha\cos\omega t \end{pmatrix}
\end{aligned}$$

Donc on a $\boxed{\frac{\mathrm{d}y}{\mathrm{d}t} = F(y,t) \text{ avec } F(y,t) = \begin{pmatrix} y_1 \\ -\frac{\omega_0}{Q}y_1 - \omega_0^2\, y_0 + \alpha\cos\omega t. \end{pmatrix}}$ (cette fois il y a dépendance explicite en t)

b/ Utilisation de odeint

On repart sur l'exemple du pendule (équations 1 et 2). on utilise la fonction `odeint` de la librairie `scipy.integrate` dont voici la syntaxe :

```
import scipy.integrate as sp
y_sol = sp.odeint(F, y_ini , t)
```

Paramètres de odeint :

- `F` est une fonction du type `F(y,t)`, qui renvoie la valeur de la dérivée de `y` à l'instant `t`.

  Pour résoudre une équation d'ordre 1, `y` est un scalaire.

  Mais si l'équation est d'ordre 2, alors `y` est de type vecteur : $y = \begin{pmatrix} y_0 \\ y_1 \end{pmatrix}$.

  Attention, même si `F` ne dépend pas de t, il faut tout de même que t apparaisse comme second argument : `F(y,t)`.

- `y_ini` : valeur initiale de `y`. C'est un couple qui a autant de composantes que `y`.

- `t` est un tableau qui contient les instants auxquels la solution sera calculée.

Par exemple, pour le problème du pendule, on peut définir `F`, `y_ini` et `t` de la façon suivante (avant l'appel à `odeint`) :

```
# Fonction F telle que d/dt(theta,theta_point) = F(theta,theta_point)
def F(y,t):
    return (y[1], -w0**2 * np.sin(y[0]))

t = np.linspace(0,5*T0,1000) # tableau des temps, de 0 à 5T0, 1000 points.

theta0 = 50 * np.pi/180.     # angle initial (rad), ici correspond à 50 degrés.
theta_prime0 = 0             # vitesse angulaire initiale (rad/s)
y_ini = (theta0,theta_prime0)   # valeur initiale de y = (theta,theta_point)
```

C'est seulement une fois `F`, `y_ini` et `t` définis comme ci-dessus qu'on peut exécuter `odeint` :

```
y_sol = sp.odeint(F, y_ini , t)
```

Objets retournés par odeint :

- Après exécution de `y_sol = sp.odeint(F, y_ini, t)`, `y_sol` est une matrice dont la première colonne contient les valeurs de $y_0(t)$ (donc dans notre exemple, de $\theta(t)$), et la seconde colonne contient les valeurs de $y_1(t)$ (donc dans notre exemple, de $\dot{\theta}(t)$).

  On peut y accéder avec la syntaxe suivante :

  ```
  theta = y_sol[:,0]          # 1ère colonne de y_sol
  theta_prime = y_sol[:,1]    # 2ème colonne de y_sol
  ```

Il suffit ensuite de tracer `theta` en fonction de `t` avec les méthodes usuelles :
`plt.plot(t,theta, label="solution de l'équation non linéaire avec odeint")`, etc.

c/ Exemple d'utilisation dans le cas du pendule

1 - Reprendre tous les éléments précédents pour écrire un algorithme qui permet de résoudre l'équation non linéaire du pendule (équation 1, que l'on a transformée en l'équation 2). On tracera la solution.

Des éléments de code sont déjà présents dans le fichier Capytale numéro `dac9-1273229`.

2 - Écrire une fonction `solution_petits_angles(t,theta0)` qui prend en argument un instant `t` et un angle initial `theta0`, et qui retourne la valeur `theta0 * np.cos(w0*t)` (où `w0` est la pulsation, déjà définie par ailleurs).

3 - Ajouter, sur le même graphe que celui de la question 1, la solution pour les petits angles. Faire pour cela une nouvelle ligne avec `plt.plot`.

4 - Explorer avec différents angles initiaux. Tester par exemple 5°, 45° et 140°.

À partir de quel angle voit-on clairement un désaccord entre la solution sans approximation (équation avec $\sin\theta$) et la solution avec petits angles (approximation $\sin\theta \simeq \theta$) ?

5 - À l'aide de votre simulation numérique, mesurer la période T des oscillations pour des angles initiaux de 10°, 30°, 60° et 80°. Calculer le rapport T/T_0 dans chaque cas (où $T_0 = 2\pi/\omega_0$ est la période pour les petits angles) et consigner les résultats dans un tableau sur votre feuille.

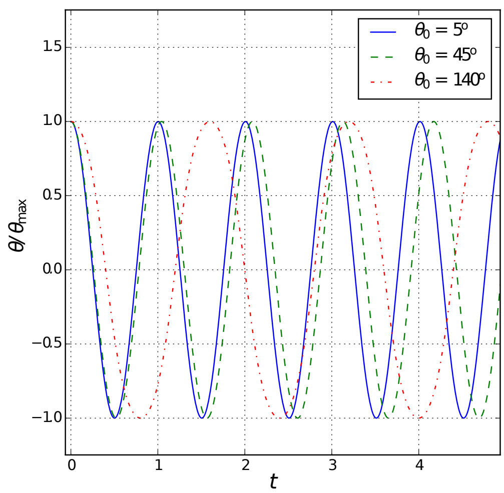
6 - On souhaite vérifier ceci **expérimentalement**, à l'aide du pendule.

Faire d'abord la mesure de T_0 en prenant un angle petit (10° par exemple).

Faire ensuite des mesures à 30°, 60° et 80°, calculer le rapport T/T_0, et comparer avec les résultats numériques : y a-t-il accord ?

Vous pouvez essayer pour des angles plus grands, mais vous verrez que le capteur pose problème et qu'il faut ruser.

7 - Si le temps le permet, recommencer la mesure de la période T pour d'autres angles initiaux θ_0, afin de tracer une courbe qui donne T en fonction de θ_0 (faire une dizaine de points). Comparer à la formule théorique de Borda (qui est une approximation) : $T = T_0\left(1 + \frac{\theta_0^2}{16} + \frac{11}{3072}\theta_0^4\right)$ en la traçant sur le même graphique.



Exemple de solutions numériques obtenues avec odeint, sans approximation, pour trois conditions initiales différentes. Cf question 1 ci-dessus.

Exemples de résultats :

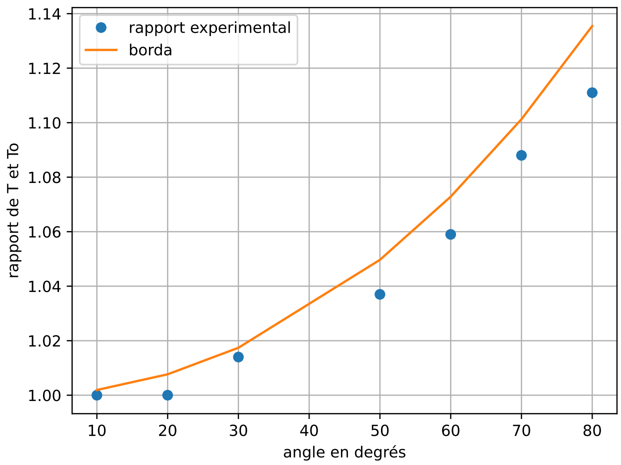
Comparaison expérience VS simulation numérique

θ_0	T/T_0 (simulation)	T/T_0 (exp)	T (exp)
10°	1,004	1,003	1,366 s
30°	1,024	1,018	1,386 s
60°	1,080	1,079	1,470 s
90°	1,180	1,184	1,613 s

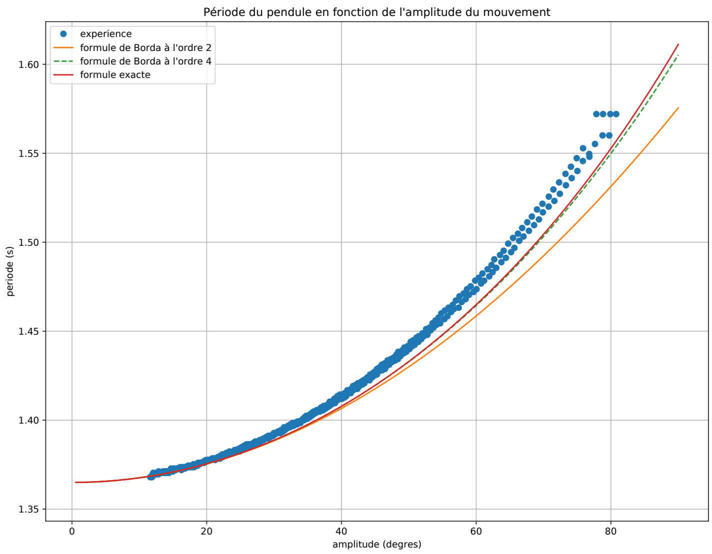
Pour l'expérience, on a mesuré $T_0 = 1{,}362\,\mathrm{s}$ pour des angles petits.

Comparaison expérience VS formule de Borda

Résultat d'un groupe en 2025 :



Résultat prof :



(mais peut-être un problème dans la façon de mesurer, automatiquement, l'amplitude, car aux grandes amplitudes elle diminue déjà d'une oscillation à l'autre...)

Code complet :

```python
# ------ Import des bibliotheques
import matplotlib.pyplot as plt  # pour les graphiques
import numpy as np               # pour les tableaux
import scipy.integrate as sp     # pour les outils numeriques comme odeint

# ------ Parametres physiques  -- on ne changera pas ces valeurs
g = 9.81          # pesanteur, m/s**2
l = 1.00          # longeur du pendule, m
w0 = (g/l)**0.5   # calcul de la pulsation, rad/s
T0 = 2.*np.pi/w0 # calcul de la periode, s, ici on obtient T0=1s

print("Periode du pendule (si petits angles) : T0 = " + str(T0) + " s")
# ------


# ------ Solution lorsqu'on fait l'approximation des petits angles (oscillateur harmonique
# t : instant t ou tableau d'instants
# theta0 : amplitude maximale
# Il est sous-entendu que la vitesse initiale est nulle.
def solution_petits_angles(t,theta0):
    return theta0 * np.cos(w0*t)


# ------ Solution de l'équation générale, sans approximation, à l'aide d'odeint

# Définition de la fonction f telle que d/dt(theta,theta_point) = F(theta,theta_point)
def F(y,t):
    return (y[1], -w0**2*np.sin(y[0]))

t = np.linspace(0,5*T0,1000) # tableau des temps, va de 0 à 5xla période, avec 1000 points

theta0 = 50 * np.pi/180.         # angle initial (rad)
theta_prime0 = 0                 # vitesse angulaire initiale (rad/s)
y_ini = (theta0,theta_prime0)    # valeur initiale de y = (theta,theta_point)
y_sol = sp.odeint(F,y_ini,t)
theta = y_sol[:,0]
theta_prime = y_sol[:,1]
# ------


# ------ Tracés
plt.figure("figure")

plt.plot(t,solution_petits_angles(t,theta0), label="solution cas des petits angles")
plt.plot(t,theta, label="solution de l'équation non linéaire avec odeint")
plt.xlabel("t (s)")
plt.ylabel("theta (degrés)")
plt.legend()
plt.grid()
plt.show()
# ------


# ------ Pour trouver automatiquement la période de la solution
L=[]
for k in range(len(t)-1):
    if theta[k]*theta[k+1]<0:
        L.append(t[k])
print(L[2]-L[0])
# ------
```