Travaux pratiques | [D] TP n°2

Résolution numérique d'une ED

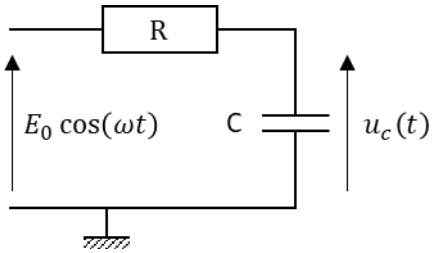
L'objectif de ce TP est d'apprendre à résoudre numériquement une équation différentielle d'ordre 1. La résolution d'une ED d'ordre quelconque sera vue plus tard dans le cours de mécanique.

Savoir résoudre numériquement une ED est une compétence indispensable en physique puisque cela nous permet d'obtenir la solution d'un problème, même si l'ED n'admet pas de solution analytique.

I - Exemple corrigé : le circuit RC en RSF

I.1 - Problème physique

On considère le circuit RC ci-contre en régime sinusoïdal forcé.



La loi des mailles donne :

$$\frac{du_c}{dt} + \frac{u_c(t)}{RC} = \frac{E_0}{RC}\cos(\omega t)$$

La solution de cette ED est :

$$u_c(t) = \underbrace{A\exp\left(-\frac{t}{RC}\right)}_{\text{SEH}} + \underbrace{U_m(\omega)\cos\big(\omega t + \phi(\omega)\big)}_{\text{SP}}$$

Avec :

$$A = u_c(0) - U_m(\omega)\cos\big(\phi(\omega)\big) \qquad U_m(\omega) = \frac{E_0}{\sqrt{1+(\omega RC)^2}} \qquad \phi(\omega) = \arctan(-\omega RC)$$

Nous allons chercher à résoudre numériquement l'ED et comparer la solution ainsi trouvée avec la solution analytique ci-dessus.

I.2 - Méthode d'Euler

On rappelle que la variation d'une fonction $f(t)$ entre deux instants très proches vaut :

$$\mathrm{d}f \simeq f(t+dt) - f(t)$$

Ce résultat est mathématiquement exact lorsque $dt \to 0$. En pratique, nous allons choisir un pas dt petit mais non nul. Plus cette valeur sera petite, plus la solution numérique sera proche de la solution réelle de l'ED, mais plus le temps de calcul pour aboutir à cette solution sera long. Il faut donc trouver un compromis raisonnable : on prendra en pratique $dt \simeq \tau/10^{3\text{ ou }4}$, où τ est la durée typique du régime transitoire.

L'ED devient :

$$\frac{u_c(t+dt)-u_c(t)}{dt} + \frac{u_c(t)}{RC} = \frac{E_0}{RC}\cos(\omega t) \quad \Rightarrow \quad \boxed{u_c(t+dt) = u_c(t) + \left(\frac{E_0}{RC}\cos(\omega t) - \frac{u_c(t)}{RC}\right)dt}$$

Ainsi, connaissant t et $u_c(t)$, on peut en déduire $u_c(t+dt)$. On détermine ainsi, de proche en proche, la valeur de $u_c(t)$ pour tout temps, à partir de la valeur initiale $u_c(0)$.

En discrétisant l'expression précédente, on obtient :

$$\boxed{u_c[i+1] = u_c[i] + \left(\frac{E_0}{RC}\cos(\omega \cdot t[i]) - \frac{u_c[i]}{RC}\right)dt} \quad \text{avec :} \quad t[i] = i \cdot dt$$

I.3 - Fonction « odeint »

Python possède une fonction « odeint » qui retourne la solution d'une ED.

$sol = odeint(deriv, CI, temps)$

Paramètres :

sol	Array	Solution $y(t)$ de l'ED, de même taille que l'array : $temps$
$deriv$	Fonction	Fonction qui prend en argument une fonction y et le temps t (obligatoire, même y' ne dépend pas explicitement du temps), et qui retourne sa dérivée y', **dont l'expression est donnée par l'ED**. $def\ deriv(y,t)$: $dydt$ = # à compléter avec l'ED. Dans notre exemple : $dydt = \frac{E_0}{RC}\cos(\omega t) - \frac{y}{RC}$ $return\ dydt$
CI	Float ou Liste	Condition initiale : $y(0)$
$temps$	Array	Temps auxquels est évaluée $y(t)$

II - Mise en application : chute d'une bille dans du glycérol

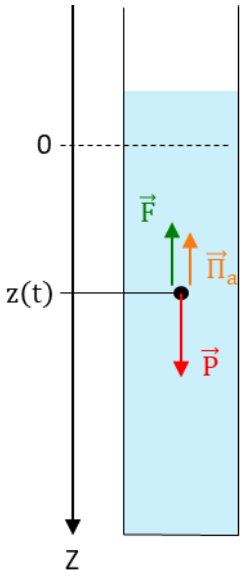
La force de frottement fluide est la force qu'exerce un fluide (gaz, liquide) sur un objet en mouvement. Cette force macroscopique est le résultat d'une multitude de chocs à l'échelle microscopique entre les particules de fluide et l'objet en mouvement.

En effet, lorsque l'objet se déplace à travers le fluide, il entre en contact avec les particules de fluide. Chaque choc donne lieu à un transfert d'une petite partie de la quantité de mouvement de l'objet vers le fluide, ralentissant ainsi légèrement la vitesse de l'objet. De par leur nombre extrêmement élevé, ces interactions microscopiques sont difficiles à modéliser et il n'est pas simple d'obtenir une expression théorique de la force macroscopique.

L'objectif de cette partie est d'étudier cette force.

II.1 - Éléments théoriques

On étudie la chute d'une bille en acier de volume V et de masse volumique ρ_b (donc de masse $m = \rho_b V$) dans une éprouvette cylindrique remplie de glycérol de masse volumique ρ_f.



BDF : La bille est soumise à trois forces.

- Son poids :

$$\vec{P} = m\vec{g} = \rho_b V\ g\ \vec{e}_z$$

- La poussée d'Archimède :

$$\vec{\Pi}_a = -\rho_f V\ g\ \vec{e}_z$$

- Une force de frottement fluide $\vec{F}$. On envisage deux modélisations :

$$\begin{cases} \vec{F} = -\alpha\ \vec{v} = -\alpha\ v\ \vec{e}_z & \rightarrow \text{ modèle des faibles vitesses} \\ \vec{F} = -\beta\ v\ \vec{v} = -\beta\ v^2\ \vec{e}_z & \rightarrow \text{ modèle des fortes vitesses} \end{cases}$$

Dans la suite, nous allons analyser un enregistrement vidéo de la chute, afin de déterminer lequel de ces deux modèles est le plus fidèle à la réalité.

Notons $\hat{g} = \left(1 - \frac{\rho_f}{\rho_b}\right) g = 4{,}866\ \mathrm{m \cdot s^{-2}}$ pour l'expérience réalisée.

⌂ Appliquer le principe fondamental de la dynamique dans le référentiel terrestre supposé galiléen et montrer que la vitesse obéit à l'équation différentielle suivante :

$$\begin{cases} \dfrac{dv}{dt} + \dfrac{\alpha}{m} v = \hat{g} & \rightarrow \text{ faibles vitesses} \\ \dfrac{dv}{dt} + \dfrac{\beta}{m} v^2 = \hat{g} & \rightarrow \text{ fortes vitesses} \end{cases} \Rightarrow \boxed{\dfrac{dv}{dt} = \left(1 - \left(\dfrac{v(t)}{v_\infty}\right)^n\right)\hat{g}} \quad \text{avec : } \begin{cases} v_\infty = v(t = +\infty) \\ n = 1 \text{ ou } 2 \text{ selon le modèle} \end{cases}$$

II.2 - Données expérimentales

Par soucis de temps, l'analyse de la vidéo a été faite en amont. Les données expérimentales (vitesse de chute en fonction du temps) se trouvent dans un fichier CSV, que l'on importe sous Python.

- Copier/coller le fichier Python et le fichier CSV qui se trouvent dans le dossier « documents en consultation » sur votre ordinateur. Faire attention à mettre les deux fichiers dans le même dossier.
- Afficher graphiquement les données expérimentales : vitesse en fonction du temps (**nuage de points**).

II.3 - Résolution numérique : méthode d'Euler

- Discrétiser l'expression encadrée et montrer que :

$$v[i+1] = v[i] + \left(1 - \left(\frac{v[i]}{v_\infty}\right)^n\right)\hat{g}\, dt$$

- Déterminer la solution de l'ED par la méthode d'Euler. Afficher la solution obtenue par-dessus les données expérimentales (**trait continu**).
- Conclure : quel modèle ($n = 1$ ou 2) de la force de frottement décrit mieux l'expérience ?

II.4 - Résolution numérique : fonction « odeint »

- Déterminer la solution de l'ED à l'aide de la fonction « odeint ». Afficher la solution obtenue par-dessus les données expérimentales (**trait continu**).

III - Pour aller plus loin : fonction « odeint » fait-maison

- Écrire une fonction « my_odeint » qui reproduit le comportement de la vraie fonction « odeint ». La fonction doit prendre les 3 mêmes arguments et renvoyer la solution de l'équation différentielle, obtenue par la méthode d'Euler.
- Déterminer la solution de l'ED à l'aide de la fonction « my_odeint ». Afficher la solution obtenue par-dessus les précédentes résolutions.

IV - DM : épreuve Centrale 2023 PC Physique 1

- Faire l'épreuve Centrale 2023 PC 1. Vous serez corrigé avec le barème officiel de l'épreuve et aurez votre classement réel, comme si vous aviez passé l'épreuve.

Annexe : code de la partie I

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

# Paramètres du problème ----------------------------------------------

E0 = 10
R = 1e3
C = 50e-9
f = 1e3
ω = 2*np.pi*f
Uc0 = 0

Um = E0/np.sqrt(1+(ω*R*C)**2)
Φ = np.arctan(-ω*R*C)
A = Uc0 - Um*np.cos(Φ)

# Solution analytique -------------------------------------------------

t_max = 7/f
t1 = np.linspace(0, t_max, 10000)  # Array de 0 à t_max de taille N

Uc1 = A*np.exp(-t1/(R*C)) + Um*np.cos(ω*t1+Φ)

# Solution numérique : méthode d'Euler --------------------------------

N = 10000
t2 = np. linspace(0, t_max, N)
dt = t2[1] - t2[0]

Uc2 = np.zeros(N)                  # Array de longueur N, rempli de 0
Uc2[0] = Uc0                       # Condition initiale

for i in range(N-1):
    Uc2[i+1] = Uc2[i] + (E0/(R*C)*np.cos(ω*t2[i]) - Uc2[i]/(R*C))*dt

# Solution numérique : odeint -----------------------------------------

from scipy.integrate import odeint

def deriv(y,t):
    dydt = E0/(R*C)*np.cos(ω*t) - y/(R*C)
    return dydt

t3 = np.linspace(0, t_max, 10000)
Uc3 = odeint(deriv, Uc0, t3)

# Affichage du graphique ----------------------------------------------

fig, ax = plt.subplots()

ax.plot(t1, Uc1, 'g-', lw=4, alpha=0.5, label='Analytique')
ax.plot(t2, Uc2, 'r-', label='Euler')
ax.plot(t3, Uc3, 'k-', label='odeint')
ax.legend()
```