TABLEAUX, TRACÉS DE COURBES
Bibliothèques Numpy et Matplotlib.pyplot

Note: Les questions des exercices ne constituent que la trame principale de ce que vous devez faire pour vous guider, rien n'interdit bien sûr que vous alliez plus loin, que vous testiez d'autres choses, bref que vous expérimentiez…

1. QUELQUES NOTIONS SUR LES TABLEAUX ET LA BIBLIOTHEQUE NUMPY

En sciences, nous sommes très souvent conduits à utiliser des tableaux de données numériques (issues de calculs ou d'expériences) et il est commode de pouvoir « visualiser » ces données sous forme de courbes (l'homme est un animal visuel).
Ainsi nous allons introduire les objets de type tableaux, `array`, afin de pouvoir stocker facilement les données, comme des couples de points, pour pouvoir ensuite tracer des courbes.

L'objet de type `array` peut être considéré comme une variante de l'objet `list` mais avec les différences et caractéristiques suivantes :

✔ Tous les éléments d'un tableau doivent être du même type, de préférence des nombres entiers (`integer`), des réels (`real`) ou des complexes (`complex`) afin d'améliorer l'efficacité du stockage des données et du calcul numérique.
✔ Le nombre d'éléments du tableau doit être connu avant de créer le tableau.
✔ Les tableaux ne font pas partie de la distribution standard de Python. Il faut donc faire appel à une bibliothèque spécifique (package en anglais) appelée **Numerical Python** souvent abrégée par **NumPy**.
✔ Avec `numpy`, une large gamme d'opérations mathématiques est directement disponible sur les tableaux, ainsi ***il n'est plus nécessaire de faire des boucles sur les éléments du tableau***, c'est « automatique » comme nous allons le voir sur des exemples. On parle de ***vectorisation***.
✔ Un tableau avec un seul indice est appelé un vecteur (`vector`). Il est bien sûr possible de manipuler des tableaux à plusieurs indices.

Un des gros avantages de stocker des nombres dans un objet de type `array` plutôt que dans un objet de type `list` est que l'utilisation des tableaux permet aux programmes de tourner plus rapidement. Cela peut-être crucial pour de nombreuses applications qui nécessitent des mathématiques avancées dans les sciences et l'industrie.

Voici quelques commandes importantes sur les tableaux :

```
>>> import numpy as np
```
Permet d'importer la bibliothèque NumPy.

```
>>> a=np.array(r)
```
Convertit la liste `r` en un tableau (assigné à la variable `a`).

```
>>> a=np.zeros(n)
```
Crée un tableau de n éléments tous égaux à zéro.

```
>>> a=np.zeros_like(c)
```
Crée un tableau d'éléments tous égaux à zéro, de type identique à celui du tableau `c` (entier, réel etc…) et de même longueur que ce dernier.

>>> a=np.linspace(p,q,n) ⇒ ***commande très utile***

Crée un tableau de n éléments uniformément distribués sur l'intervalle $[p,q]$. Un élément i du tableau est accessible par la commande $a[i]$ (comme pour les listes).

⌨ Taper le code suivant dans un script Python et exécuter le :

```
import numpy as np

def f(t):
    return t**2*np.exp(-t**2)

t = np.linspace(0, 3, 51)
y = np.zeros(len(t))
y = f(t)
```

 Quel est le type de la variable y ? Faire afficher à l'écran le contenu de y. Que fait la commande y = f(t) ? Nous n'avons pas utilisé from math import * mais nous avons la commande np.exp, expliquer.
?

2. TRACÉS DE COURBES, BIBLIOTHÈQUES MATPLOTLIB.PYLAB

Avec l'utilisation des tableaux, il est assez facile de tracer des courbes sous Python. Afin de voir les commandes essentielles, nous allons travailler sur un exemple:

⌨ Taper le code suivant dans un script Python et exécuter le :

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

def f1(t):
    return t**2*np.exp(-t**2)

def f2(t):
    return t**2*f1(t)

t = np.linspace(0, 3, 51)
y1 = f1(t)
y2 = f2(t)

plt.plot(t, y1, 'r-')
plt.plot(t, y2, 'bo')
plt.xlabel('t')
plt.ylabel('y')
plt.legend(['t^2*exp(-t^2)', 't^4*exp(-t^2)'])
plt.title('Plotting two curves in the same plot')
plt.savefig('tmp.eps')
plt.show()
```

⌨ Dans un nouveau script, remplacer la dernière partie du code précédent par le code suivant :

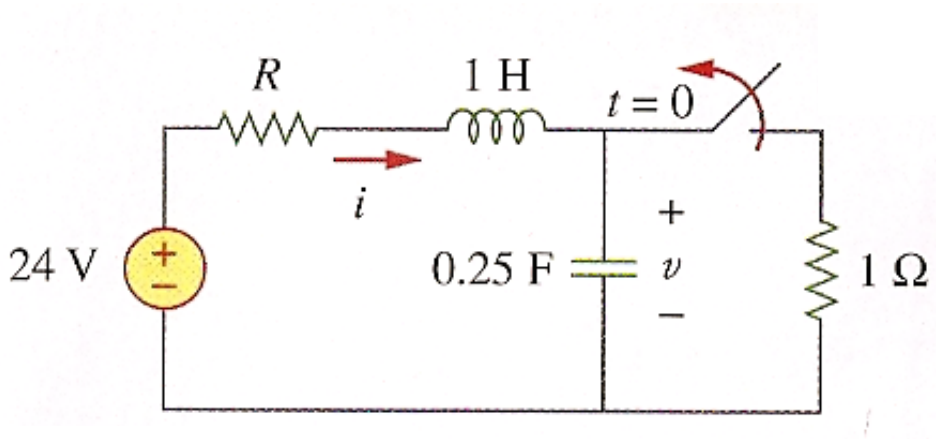
```
plt.subplot(2, 1, 1)
t = np.linspace(0, 3, 51)
y1 = f1(t)
y2 = f2(t)
plt.plot(t, y1, 'r-', t, y2, 'bo')
```

```
plt.xlabel('t')
plt.ylabel('y')
plt.axis([t[0], t[-1], min(y2)-0.05, max(y2)+0.5])
plt.legend(['t^2*exp(-t^2)', 't^4*exp(-t^2)'])
plt.title('Top figure')
plt.subplot(2, 1, 2)
t3 = t[::4]
y3 = f2(t3)
plt.plot(t, y1, 'b-', t3, y3, 'ys')
plt.xlabel('t')
plt.ylabel('y')
plt.axis([0, 4, -0.2, 0.6])
plt.legend(['t^2*exp(-t^2)', 't^4*exp(-t^2)'])
plt.title('Bottom figure')
plt.savefig('tmp2.eps')
plt.show()
```

 Remarquer la commande `ptl.` devant chaque instruction. Assurez-vous de bien comprendre ce que fait chaque ligne du code.

APPLICATION 1 : CIRCUIT D'ORDRE 2 EN REGIME TRANSITOIRE

On considère le circuit ci-dessous. A $t>0$, on ouvre l'interrupteur. On obtient par le calcul (je vous invite à le faire pour vous entraîner !) les résultats suivants :



$\Rightarrow$ Si $R = 5\ \Omega$:

$$\begin{cases} v(t) = 24 + \frac{4}{3}\left(-16e^{-t} + e^{-4t}\right)\ \text{V} \\ i(t) = \frac{4}{3}\left(4e^{-t} + e^{-4t}\right)\ \text{A} \end{cases}$$

$\Rightarrow$ Si $R = 4\ \Omega$:

$$\begin{cases} v(t) = 24 - 19.2\left(1+t\right)e^{-2t}\ \text{V} \\ i(t) = \left(4{,}8 + 9.6t\right)e^{-2t}\ \text{A} \end{cases}$$

$\Rightarrow$ Si $R = 1\ \Omega$:

$$\begin{cases} v(t) = 24 + \left(21.694\sin\left(1.936t\right) - 12\cos\left(1.936t\right)\right)e^{-0{,}5t}\ \text{V} \\ i(t) = \left(3{,}1\sin\left(1.936t\right) + 12\cos\left(1.936t\right)\right)e^{-0{,}5t} \end{cases}$$

Ecrire un code qui permette de tracer deux graphes. Sur le premier, on tracera les courbes de la tension aux bornes du condensateur $v = f(t)$ et sur le deuxième, on tracera les courbes relatives à l'intensité $i = f(t)$. Les axes seront libellés, il y aura une légende etc…

Pour chacun des régimes précédents, tracer le portrait de phase $\frac{dv}{dt} = f(v)$.


Analyser physiquement les courbes. Quel type de régime obtient-on ?

APPLICATION 2 : ETUDE DE LA CINETIQUE DE DUPLICATION DE BACTERIES

On place 100 bactéries, à l'instant initial et à 37°C, dans un récipient riche en nutriments pour permettre aux bactéries de se dupliquer. On enregistre l'évolution suivante :

Temps (min)	***Nombre de bactéries***
0	100
15	200
30	400
45	800
60	1600


Quel est l'ordre de la vitesse de duplication des bactéries. Combien de bactéries seront présentes après deux heures ? Quelle est la constante de vitesse de ce processus ? Pensez à sortir votre cours sur la cinétique chimique !

Pour répondre à ces questions, il faudra écrire un code qui, entre autres, permet de tracer un graphique montrant, en fonction du temps, l'évolution de $[\text{bactéries}]$, de $\ln[\text{bactéries}]$ et de $1/[\text{bactéries}]$.

APPLICATION 3 : TRAJECTOIRE DANS LE CHAMP DE PESANTEUR TERRESTRE

Ecrire un code qui permette de tracer la trajectoire d'un ballon $y = f(x)$ (on prend y pour la verticale) dans le champ de pesanteur terrestre $\vec{g} = -g\vec{j}$ (on néglige les frottements de l'air et autres effets). Vous prendrez comme vitesse initiale $v_0 = 20$ m.s^1.
Il faudra nommer les axes, mettre des légendes, des titres etc.
Vous pourrez tracer différentes courbes (sur le même graphe) pour différents angles initiaux de tir et vérifier que la portée est maximale pour un angle de 45 °.

Rappels de cinématique :

$$\vec{a} : \begin{cases} a_x = 0 \\ a_y = -g \end{cases} \qquad \vec{v} : \begin{cases} v_x(t) = v_0 \cos\theta_0 \\ v_y(t) = -gt + v_0 \sin\theta_0 \end{cases} \qquad \overrightarrow{OP} : \begin{cases} x(t) = (v_0 \cos\theta_0)t \\ y(t) = -\frac{1}{2}gt^2 + (v_0 \sin\theta_0)t \end{cases}$$

$$y(x) = -\left(\frac{g}{2v_0^2\left(\cos\theta_0\right)^2}\right)x^2 + \left(\tan\theta_0\right)x$$