TD informatique : Lecture dans un fichier - Représentations graphiques

1 Commandes de base

L'objectif de cette séance est d'apprendre à lire des données dans un fichier, à les manipuler et à produire des représentations graphiques de ces données.

1.1 Lire un fichier

On rappelle ici les commandes `Python` utiles. Celles-ci sont rappelées dans le Mémento.

```
##ouvrir un fichier
f=open("mon_petit_fichier.txt","r")
#"r" pour read
#"w" pour write
```

La commande crée une variable `f` qui correspond à l'accès au fichier (pas à son contenu),il faut donc aller lire le contenu de `f`. Il y a plusieurs méthodes pour cela :

```
L1=f.read() #l'ensemble du contenu est intégré
            #dans une chaîne de caractères

L1=f.read(15) #seuls les 15 premiers caractères
        #sont lus et insérés dans L1

L1=f.readline() #seule la première ligne est
        #lue et insérée dans L1

L2=[]
for ligne in f :
        L2.append(ligne) #chaque ligne du fichier
                #est insérée dans L2
```

On peut également se souvenir de la méthode suivante qui permet de créer une liste de sous-chaînes de caractères à partir d'une chaîne contenant le caractère ";" :

```
ligne.split(";")
#Testez sur :
ligne="vert;bleu;orange"
```

Enfin, il est impératif de refermer l'accès au fichier sous peine de l'endommager :

```
f.close() #très important : à ne pas oublier!
```

1.2 Représentation graphique de données

On rappelle également les commandes `Python` permettant les représentations graphiques.

```
import matplotlib.pyplot as plt

les_x=[x for x in range(50)] #création abscisses
les_y=[x**2 for x in les_x] #création ordonnées

plt.plot(les_x,les_y) #le plus simple
plt.plot(les_x,les_y,'+') #pour ne pas relier les points
plt.plot(les_x,les_y,'green') #pour tracer en vert

help(plt.plot) #pour plus d'options

plt.plot(les_x,les_y,label="$x^2$") #pour légender
plt.legend() #pour afficher la légende

plt.xlabel("les x") #axe des abscisses
plt.ylabel("les y=x2") #axe des ordonnées
plt.title("le titre : important") #tout est dans le nom

plt.show() #pour afficher
```

2 Échauffement

Les données issues d'une simulation ont été enregistrées dans le fichier `premiers_pas.txt`. Il s'agit des valeurs correspondant à l'évolution d'une fonction exponentielle.

1. S'assurer que le fichier à lire est dans le même répertoire que le fichier `.py` qui est utilisé pour la lecture.
2. Écrire une fonction `lecture` qui prend comme argument une chaîne de caractère `"nom_fichier.txt"` correspondant au nom du fichier à lire, et qui renvoie autant de listes qu'il y a de colonnes dans le fichier.
3. Comment faut-il modifier la fonction pour lire un fichier qui n'est pas dans le répertoire de travail ?
4. Utiliser la fonction `lecture` pour définir deux listes `les_x`, `les_y` correspondant chacune à une colonne du fichier `premiers_pas.txt`.
5. Représenter `les_y` en fonction de `les_x`.

3 Étude fréquentielle d'un filtre

Le problème est le suivant : on dispose d'un filtre dont on ne connait pas les caractéristiques. Pour l'étudier, on l'alimente donc avec différentes tensions sinusoïdales et on analyse son comportement en fonction de la fréquence du signal d'entrée.

60 mesures ont été réalisées (avec 60 tensions d'entrée différentes) et les résultats (entrée et sortie en fonction du temps) ont été stockés dans des fichiers texte. Chaque mesure est stockée dans un fichier distinct et le but de l'exercice est d'automatiser la lecture et l'exploitation de ces fichiers.

3.1 Prise en main

1. À l'aide de la fonction `lecture` définie préalablement, ouvrir et lire l'un des fichiers de mesure.
2. Représenter sur un même graphique la tension d'entrée et la tension de sortie pour une des mesures.
3. Vérifier sur quelques fichiers que les différentes mesures proposées permettent une qualification du filtre.

3.2 Traitement d'une mesure

4. Écrire une fonction `pulsation` qui permet de déterminer la pulsation ω du signal traité dans le fichier `mesureX.txt`. On rappelle que, pour déterminer la période d'un signal sinusoïdal, il faut repérer deux passages successifs à une même valeur (avec le même sens de variation). Par ailleurs il est plus précis de repérer le passage du signal là où la pente est la plus grande.
5. Écrire une fonction `valeur_moyenne` qui renvoie la valeur moyenne du signal représenté dans une liste `L`.
6. Écrire une fonction `valeur_efficace` qui renvoie la valeur efficace du signal sinusoïdal représenté dans une liste `L`.
7. Écrire une fonction `amplitude` qui renvoie l'amplitude du signal sinusoïdal représenté dans une liste `L`.
8. Écrire une fonction `gain` qui renvoie le gain entre le signal de sortie et le signal d'entrée enregistrés dans le fichier `mesureX.txt`.
9. Écrire une fonction `déphasage` qui renvoie le déphasage entre le signal de sortie et le signal d'entrée enregistrés dans le fichier `mesureX.txt`. Attention au signe et à l'intervalle de variation.
10. Modifier les fonctions `pulsation`, `gain` et `déphasage` pour n'avoir à lire qu'une seule fois le fichier `mesureX.txt`.

3.3 Diagramme de Bode

11. À partir des fonctions précédentes, écrire un programme permettant de traiter les 60 fichiers texte disponibles afin de produire trois listes : `les_w`, `les_GdB`, `les_phi` contenant respectivement les pulsations, les gains et les déphasages de chacune des mesures réalisées. Attention à ranger ces valeurs dans le même ordre que la numérotation des fichiers auxquels elles se rapportent.
12. Tracer le diagramme de Bode en gain (en dB) et en déphasage (en °). Attention aux légendes.
13. Déterminer les caractéristiques du filtre.