TP 4 - Fichiers et dictionnaires

L'objectif du TP est de dépouiller de manière automatique les fichiers textes contenant les résultats d'essais [1] expérimentaux réalisés sur un moteur à courant continu. À partir de ces essais, on pourra déterminer le couple résistant exercé sur le moteur, puis évaluer l'influence de la vitesse de rotation du moteur sur le couple résistant.

I Exercice 1 : post-traitement d'un essai

L'emplacement d'un fichier est déterminé par un chemin (*absolu*), qui pourra prendre une forme différente suivant le système d'exploitation de la machine. Les chemins sont représentés par une chaîne de caractères, on pourra rencontrer par exemple :

- `"C:\Users\nom_utilisateur\Mes Documents\TP info\essais\essais1.txt"` sur un système d'exploitation de type Windows ;
- `"/home/nom_utilisateur/Documents/TP info/essais/essais1.txt"` sur un système d'exploitation de type Unix (Linux ou MacOS X par exemple).

Dans les deux cas, le chemin commence par une suite de répertoires (séparés par `\` sous Windows et `/` sous Unix) et se termine par le nom de fichier proprement dit (ici, `essai1.txt`).

Il est également possible de repérer les fichiers par rapport à un *répertoire de travail courant.* L'emplacement de chaque fichier est alors repéré en partant de ce répertoire de travail : on parle alors de *chemin relatif.*

Par exemple, si le chemin relatif d'un fichier est `"essais1.txt"` et que le répertoire de travail est `"C:\Users\nom_utilisateur\Mes Documents\TP info\essais"` le chemin absolu du fichier sera : `"C:\Users\nom_utilisateur\Mes Documents\TP info\essais\essais1.txt"`

Attention ! En Python, le caractère contre-oblique `\` joue un rôle spécial dans l'interprétation des chaînes de caractères : il sert à y introduire des caractères spéciaux comme le saut à la ligne `'\n'` ou la tabulation `'\t'`.

Ceci pose donc problème lorsqu'on doit saisir des chemins de Windows. Pour travailler avec le chemin `C:\Users\nom_utilisateur`, on peut :

- redoubler chaque barre oblique pour indiquer à Python qu'elle ne sert pas à introduire de caractère spécial : `"C:\\Users\\nom_utilisateur"` ;
- utiliser la barre oblique `/` au lieu de la contre-oblique `\`. Les fonctions de gestion de fichiers de Python l'interpréteront comme un séparateur de répertoires même si l'on travaille sous Windows : `"C:/Users/nom_utilisateur"`

Q1. Importer les fonctions `getcwd` et `chdir` du module `os` (comme operating system), puis déterminer le chemin du répertoire courant à l'aide de la fonction `getcwd()` (fonction qui ne prend pas de paramètre).

Modifier le répertoire courant afin de vous placer dans le dossier contenant les fichiers des différents essais en utilisant la fonction `chdir`, puis vérifier que le répertoire courant a été modifié.

1. Les essais ont été réalisés sur le système de trancanage de la salle de TP de S2I.

II Analyse d'un essai

Extraction des mesures d'un essai

Le répertoire courant étant celui qui contient l'ensemble des fichiers résultats, on travaillera par la suite avec les chemins relatifs, par exemple `"essai1.txt"`.

Q2. Écrire un script python permettant d'afficher les 20 premières lignes du premier essai `"essai1.txt"` (ne pas oublier de fermer le fichier).

Chaque fichier est construit de la manière suivante :

— Les 12 premières lignes contiennent les informations relatives à l'essai. Pour le premier essai, on peut lire que la fréquence d'acquisition est de 1000 Hz, que la consigne de vitesse est de 50 tr/min selon une loi en trapèze avec une durée d'accélération de 0.3 s, une phase à vitesse constante de 0.5 s puis une durée de décélération de 0.3 s, ainsi que d'autres informations.

— La 13e ligne donne les grandeurs mesurées : temps(s), Consigne (tr/min)...

— Les lignes suivantes donnent les valeurs mesurées à chaque instant (avec des nombres à virgules, contrairement à Python qui utilise des points).

— Les différentes informations d'une ligne sont séparées par une tabulation.

Q3. Écrire une fonction python qui renvoie une nouvelle chaîne de caractères obtenue en remplaçant les "," par des "." (on fabriquera la nouvelle chaîne caractère par caractère, car le type `str` n'est pas mutable). Pour extraire les données dans le script suivant, on pourra utiliser :

— L'instruction `fichier.readline()` qui permet d'obtenir une ligne d'un fichier puis de passer à la ligne suivante, ce qui va permettre de passer les 13 premières lignes.

— L'instruction `texte.split('\t')` qui renvoie la liste des chaînes de caractères obtenue en découpant la chaîne de caractères `texte` à chaque apparition de la chaîne de caractères `'\t'`(soit à chaque tabulation).

— L'instruction `float("3.14")` qui permet d'obtenir le flottant `3.14` correspondant à la chaîne de caractère `"3.14"`. Par contre, `float("3,14")` renvoie un message d'erreur, car Python ne gère pas les flottants avec des virgules.

Q4. Écrire la fonction `extractionResultatsEssai(cheminFichier)` qui permet d'extraire les grandeurs physiques de temps, de consigne, de vitesse du moteur et de courant d'un chemin d'essai donné en entrée, soit les données présentes dans la première, la deuxième, la troisième et la cinquième colonne du fichier à partir de la $14e$ ligne.

La sortie de la fonction est un dictionnaire ayant pour clefs `"temps"`, `"consigne"`, `"vitesse"` et `"courant"` qui contient des listes de flottants correspondants respectivement aux grandeurs mesurées.

Q5. Extraire les résultats de l'essai 1, puis afficher la liste de temps afin de vérifier que la liste de temps va de 0 s a environ $1,1$ s.

Q6. Exécuter la fonction `affichage` présente dans le script Python, puis l'utiliser afin d'afficher les courbes d'un essai. Le résultat obtenu pour l'essai 15 est visible sur la figure 1.

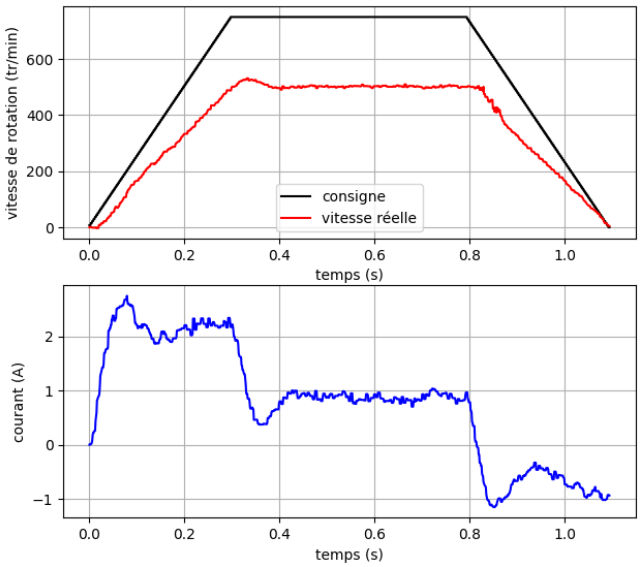


FIGURE 1 – Courbes obtenues pour l'essai 15 : "essai15.txt"

Détermination du couple résistant et de la vitesse de rotation

En régime permanent (vitesse de rotation constante, pas d'accélération), le couple moteur $c_m(t)$ sert uniquement à vaincre le couple résistant $c_r(t)$. De plus pour un moteur à courant continu, le couple moteur est proportionnel au courant $i(t)$ selon la constante de couple K, le couple résistant peut donc se déterminer par la relation suivante(en régime permanent) :

$$c_r(t) = c_m(t) = K \times i(t) \text{ (si la vitesse de rotation est constante)}$$

Avec $K = 0,13\ Nm/A$ (Newton mètre par ampère)

Q7. Écrire la fonction `moyenneSurIntervalle(L,deb,fin)` qui permet de déterminer la valeur moyenne des nombres compris entre les indices `deb` (inclus) et `fin` (exclu) de la liste `L`. Puis tester la fonction.

Q8. Écrire un script qui permet de déterminer la valeur du couple résistant de l'essai 15 en tenant compte uniquement du courant entre $0,5$ et $0,7\ s$. On rappelle que la fréquence d'acquisition est de $1000\ Hz$ soit une mesure toutes les millisecondes.

III Analyse de l'ensemble des essais

On souhaite déterminer le couple résistant pour l'ensemble des essais, mais avant ça, vérifions que chaque essai est réalisé avec les mêmes conditions.

Extraction des informations d'un essai

Q9. Écrire la fonction `extractionInfoEssai(cheminFichier)` qui permet d'extraire les informations contenues dans les 12 premières lignes d'un essai. La sortie de la fonction est un dictionnaire ayant pour clefs le contenu du premier élément de la ligne et pour valeur le contenu du deuxième élément de la ligne. Les valeurs seront laissées sous forme de chaînes de caractères.

L'extraction des informations de l'essai 1 donne par exemple le dictionnaire suivant :

```
{'Date Essai': '09/03/2022', 'Heure Essai': '14:31',\
 'Frequence (Hz)': '1000,0', 'Fonctionnement': 'Uhing', \
```

```
3   'Type Pilotage': 'Trapeze', 'Nb cycles': '1', \
4   'Duree acceleration (s)': '0,30', 'Vitesse (tr/min)': '50,0',\
5    'Temps maintien (s)': '0,50', ' Duree deceleration (s)': '0,30',\
6     'Nb points de consigne': '365', 'dt consigne': '0,003'}
```

Influence de la vitesse sur le couple résistant

Q10. Écrire un script qui permet de vérifier que chacun des 20 essais est réalisé dans les mêmes conditions que l'essai 1 pour les informations suivantes :

- un pilotage de type trapèze
- une durée d'accélération de $0,3$ s
- une durée de maintien à vitesse constante de $0,5$ s
- une durée décélération de $0,3$ s

Si l'un des essais ne respecte pas l'une de ces conditions, afficher le chemin de l'essai qui n'est pas conforme.

Remarque : les essais étant tous cohérents, afin de vérifier le script précédent dans les deux cas de figure, on pourra aller modifier une valeur dans l'un des fichiers textes.

Q11. Pour chaque essai présent dans le dossier, déterminer le couple résistant et la vitesse de rotation sur l'intervalle compris entre $0,5$ et $0,7$ s (on pourra mémoriser les résultats dans deux listes).

Q12. Enregistrer les données obtenues dans le script précédent dans un fichier nommé `"resultat.txt"` contenant sur la première ligne les noms des grandeurs physiques `Vitesse du moteur (tr/min)` et `Couple resistant (Nm)` séparés par une tabulation, puis sur les lignes suivantes les valeurs déterminées toujours séparées par une tabulation.

Q13. Tracer l'allure du couple résistant en fonction de la vitesse de rotation du moteur.
Proposer une modélisation du couple résistant Cr fonction de la vitesse du moteur N.

Remarque : afin de déterminer les coefficients du modèle, la fonction `polyfit` du module `numpy` peut être utilisée. Toutes les informations relative à son utilisation sont présentes dans l'aide de la fonction.

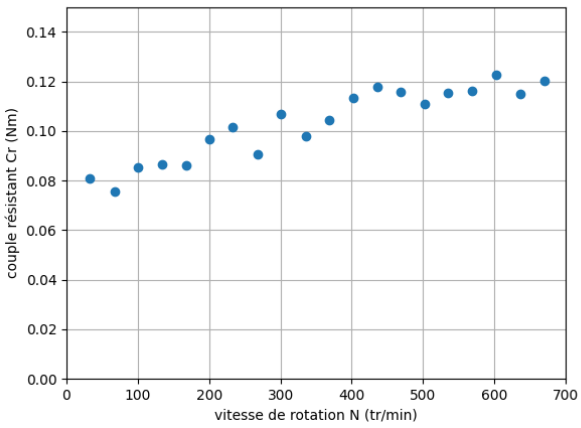
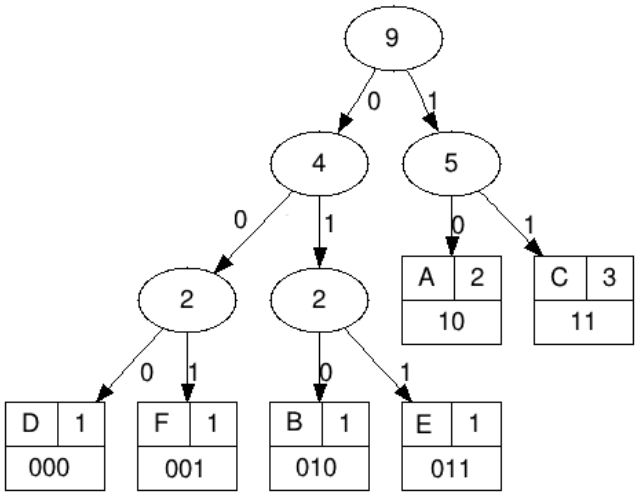


FIGURE 2 – Couple résistant en fonction de la vitesse du moteur

IV Exercice 2 : Codage de Huffman

La compression de fichier utilise un algorithme basé sur les arbres binaire appelé le codage de Huffman. Le codage de Huffman utilise un code à longueur variable pour représenter un symbole de la source (par exemple un caractère dans un fichier). Le code est déterminé à partir d'une estimation des probabilités d'apparition des symboles de source, un code court étant associé aux symboles de source les plus fréquents. La première étape du codage de Huffman consiste à créer un dictionnaire contenant la liste des caractères présents dans le texte, associé à leur fréquence dans ce texte. Exemple : `"AABCDCCEF"` donnera `'A':2,'B':1,'C':3,'D':1,'E':1,'F':1`. La deuxième étape consiste à construire un arbre de Huffman qui permet ensuite de coder chaque caractère.



Notre texte `"AABCDCCEF"` de 9 caractères ASCII (72 bits) sera ainsi codé en binaire `"10 10 010 11 000 11 11 011 001"` (22 bits). Chaque caractère constitue une des feuilles de l'arbre a laquelle on associe un poids valant son nombre d'occurrences. Puis l'arbre est créé suivant un principe simple : on associe à chaque fois les deux noeuds de plus faibles poids pour créer un noeud dont le poids équivaut à la somme des poids de ses fils jusqu'à n'en avoir plus qu'un, la racine. On associe ensuite par exemple le code 0 à la branche de gauche et le code 1 à la branche de droite. Pour obtenir le code binaire de chaque caractère, on descend de la racine jusqu'aux feuilles en ajoutant à chaque fois au code un 0 ou un 1 selon la branche suivie. Pour pouvoir être décodé par l'ordinateur, l'arbre doit aussi être transmis. Le code Python permettant de construire un arbre de Huffman et de coder chaque caractère est fourni en annexe. Les feuilles de l'arbre de Huffman (leaf) sont codées sous la forme de tuple avec comme premier élément le caractère et comme deuxième élément le poids. Pour l'arbre donné en exemple en Figure 2, on aura 6 tuples : `('A',2), ('B',1), ('C',3), ('D',1), ('E',1) et ('F',1)`.

Q14. Analyser le code du programme `TP4_Huffman.py`. Puis donner le contenu des variables `node1` et `node2` suite à l'exécution des commandes :

```
node1=make_huffman_tree(('F',1),('E',1))
node2=make_huffman_tree(('D',1),('B',1))
```

Q15. De même, donner le contenu de la variable `node3` suite à l'exécution de la commande

```
node3=make_huffman_tree(node1,node2)
```

Q16. Écrire une fonction `table_freq(txt)` qui étant donné une chaîne de caractère texte retourne un dictionnaire qui associe à chaque caractère son nombre d'occurrences dans texte. L'appel de

`table_freq("ABRACADABRA")` devra retourner un dictionnaire de la forme `'A': 5, 'R': 2, 'B': 2, 'C': 1, 'D': 1` (pas forcément dans cet ordre)

Q17. Commentez les étapes de la fonction `encode_Huffman(dico)`
Q18. Écrire une fonction `encodage(code,texte)` qui étant donné code de Huffman construit par la fonction précédente et le texte initial retourne la chaîne de bits produite par le codage de Huffman L'appel de `encodage(C,"ABRACADABRA")` devra retourner un dictionnaire de la forme `"01111001100011010111100"`

Q19. Écrire une fonction `decodage(code, texte_binaire)` qui étant donné code de Huffman un texte binaire compressé retourne le texte initial On utilisera `code_inv = dict((code[str(bits)], bits) for bits in code)` pour construire le dictionnaire inverse. L'appel de `decodage(C, "0111100110001101011110` devra retourner un dictionnaire de la forme `"ABRACADABRA"`