Travaux pratiques n°7

Gestion de fichiers en Python. Application à l'analyse de réponses temporelles de systèmes asservis

F. Cuvellier, J. Tanoh

M.P.S.I. 1, 2018–2019

1 Gestion de fichiers en Python

1.1 Ouvrir et fermer un fichier

Python ouvre un fichier à l'aide de la fonction `open()`. De façon précise, le fichier est indiqué par son chemin ; l'ouvrir signifie simplement que Python a accès à son contenu. La fonction `open()` associe un objet, que l'on a intérêt à désigner par une variable (sinon, le contenu peut être plus difficile à exploiter).
Par exemple, `A = open('./essai.txt')` ouvre le fichier `essai.txt` du répertoire courant (`./`) et l'associe à l'objet référencé par la variable `A`. Le chemin dépend du système d'exploitation (par exemple, la barre oblique (/) sert de délimiteur du nom de répertoire sous Linux, alors que sous Windows, c'est la barre contre-oblique (\)).
Parmi les arguments optionnels de la fonction `open()`, deux méritent une attention particulière : d'une part, le *mode* (`'r'` ou `'w'`), d'autre part, l'*encodage* (`encoding = 'utf8'`, par exemple). Le mode est le deuxième argument de la fonction `open()` ; il sert à indiquer si le fichier peut être seulement lu (`'r'`), ou, au contraire, si on peut le modifier (`'w'`). Par défaut, la valeur est `'r'`. L'encodage d'un fichier texte décrit la façon dont les caractères spéciaux seront encodés (couramment, on se sert de l'UTF8).
Par exemple, l'instruction `A = open('./essai.txt', 'w', encoding = 'iso-8859-1')` ouvre le fichier `./essai.txt`, l'associe à l'objet `A`, il est modifiable. De plus, la lecture et l'écriture se font suivant l'encodage iso-8859-1 (qui sert essentiellement aux langues latines).

Une fois que le fichier a été exploité, on prend la précaution de le fermer, à l'aide de la méthode `close()`. Ici, on exécuterait l'instruction `A.close()`.

1.2 Exploiter le contenu d'un fichier texte en mode lecture

Plusieurs méthodes permettent d'exploiter un fichier texte en mode lecture, dont voici quelqu'unes.

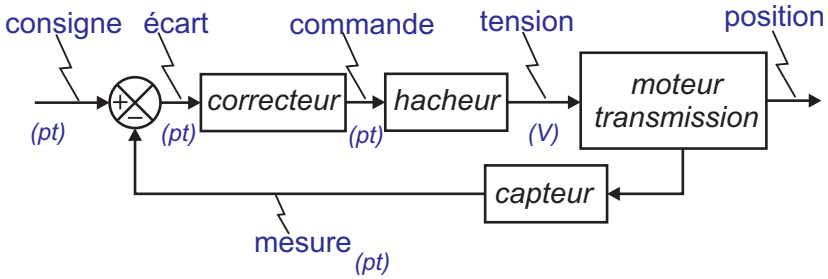
Il est essentiel de voir qu'il est perçu comme une suite de lignes, que termine toutes un caractère de fin de ligne (`\n`).
On peut le lire entièrement et en faire une chaîne de caractères à l'aide de la méthode `read()`.
Pour produire le tableau des lignes d'un fichier, on se sert de la méthode `readlines()`.
Pour lire une seule ligne du fichier (précisément, la ligne *suivante*), on se sert de la méthode `readline()`.

Un pointeur parcourt tout le fichier. En pratique, le fichier n'est lu qu'à partir de la position courante du pointeur. (Par exemple, la méthode `readline()` fait avancer le pointeur jusqu'au caractère de fin de ligne suivant.) Diverses méthodes permettent de déplacer le pointeur et de déterminer sa position. Pour les connaître, consulter la page d'aide.

2 Analyse de réponses temporelles de systèmes asservis

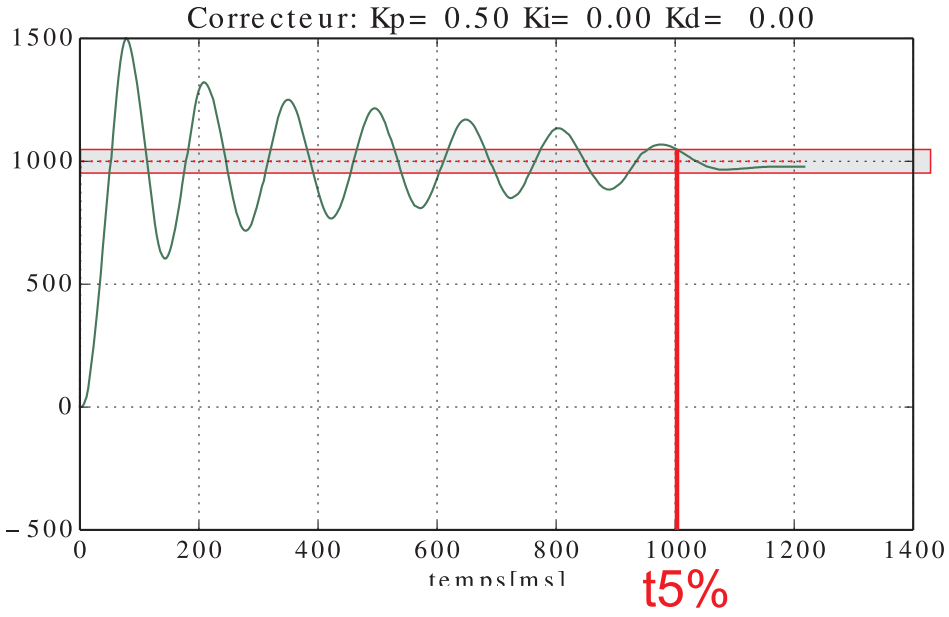
2.1 Présentation du problème

Des essais d'un système asservi en position ont été réalisés pour différents réglages d'un correcteur et des mesures ont été faites qui permettent de valider certains réglages selon les exigences du cahier des charges.



Les courbes construites à partir de huit essais (parmi de très nombreux) sont données en annexe. Il est possible de déterminer deux grandeurs intéressantes :

- L'erreur statique de position : c'est la valeur vers laquelle tend l'écart entre la consigne et la mesure si la position se stabilise ; c'est le cas pour les essais à étudier.
- Le temps de réponse à 5% qui correspond à la date à partir de laquelle le signal de mesure reste à l'intérieur d'une bande de $\pm 5\%$ par rapport à l'asymptote du signal de mesure.



Le système de commande du processus est numérique, la commande est élaborée avec une période très proche de 3 ms, les mesures sont effectuées au même rythme.
Les résultats sont fournis sous la forme de fichiers `csv` (cette extension indique simplement comment sont séparées les données) avec un en-tête dans lequel la première ligne sera indiquée sur le tracés car elle contient des données de réglages ; suivent trois autres lignes, non utiles. Le fichier continue par les données selon quatre colonnes :

- Le temps en milliseconde,
- La consigne en point,
- La mesure en point,
- La commande en point.

2.2 Exemple de début de fichier à traiter

Voici les quinze premières lignes du fichier `essai4.csv`.
La première donne le titre ; la troisième décrit les colonnes. La première des lignes de mesures est la cinquième.

```
Correcteur: Kp= 0.10 Ki= 0.00 Kd= 0.00
Consigne [pt] : 1000.00
date;Consigne;Mesure;Commande
[ms];[pt];[pt];[pt]
0; 0; 0; 0
1; 1000; 0; 100
4; 1000; 1; 99
7; 1000; 4; 99
10; 1000; 13; 98
13; 1000; 22; 97
16; 1000; 35; 96
19; 1000; 51; 94
22; 1000; 70; 93
25; 1000; 93; 90
28; 1000; 122; 87
```

Les fichiers à traiter correspondent à des réponses d'un système avec réglage assurant la stabilité et on est assuré que les vingt dernières lignes de données correspondent à l'état stabilisé.

2.3 Objectif

Élaborer une fonction en Python qui réalise :

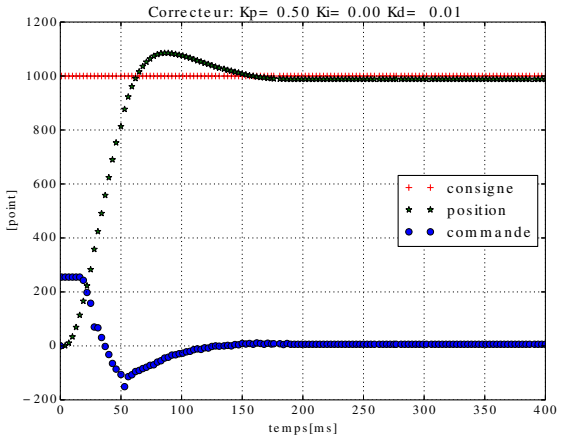
- La lecture du fichier avec extraction du texte de la première ligne,
- La détermination de la valeur moyenne des 20 dernières lignes de la position et l'erreur statique de position en valeur relative,
- La détermination du temps de réponse à 5%,
- Le tracé de la courbe de mesure en fonction du temps avec :
  - En titre, la première ligne du fichier,
  - L'asymptote de la mesure,
  - Les deux horizontales délimitant la bande de 5%,
  - Le tracé de la consigne en pointillé.

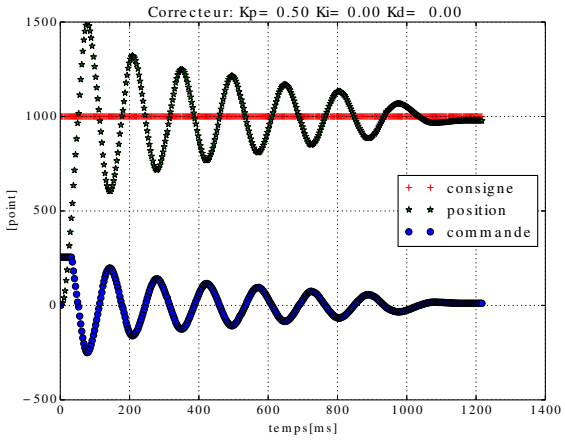
2.4 Données

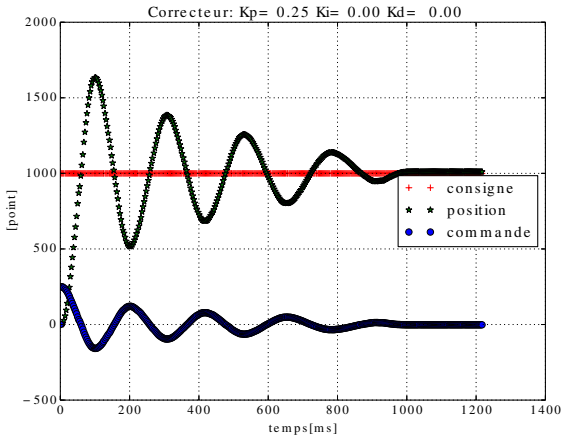
Dans un répertoire `X21.data` se trouvent un fichier `X21.py`, ébauche du travail, et huit fichiers de données à traiter.

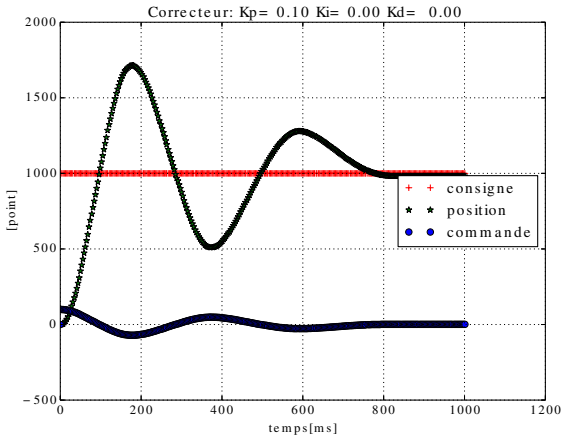
2.5 Réponses temporelles du système selon le réglage

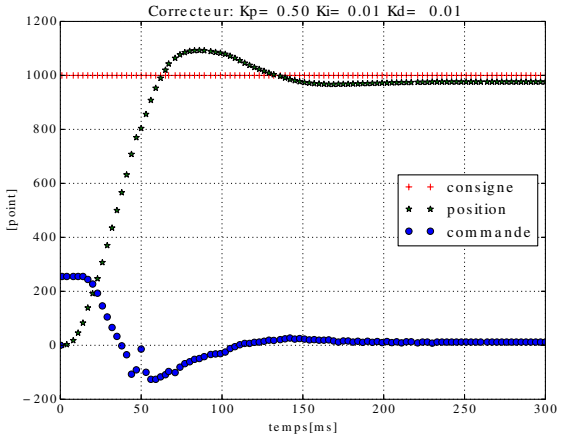
Essais du 17 mars 2016

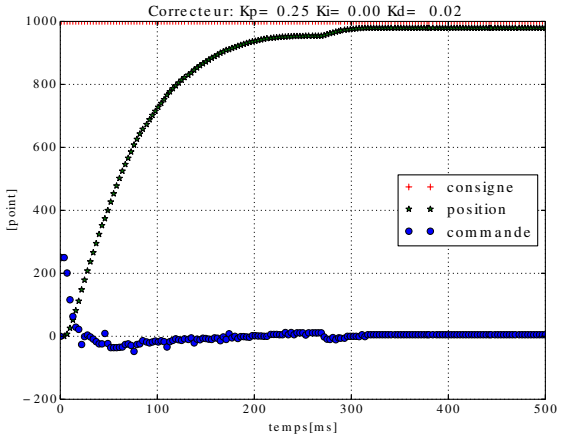


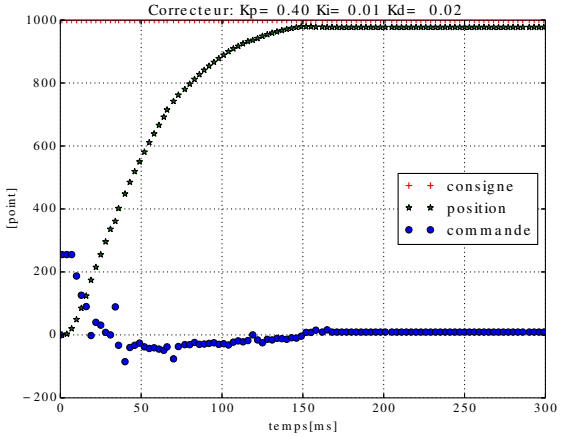


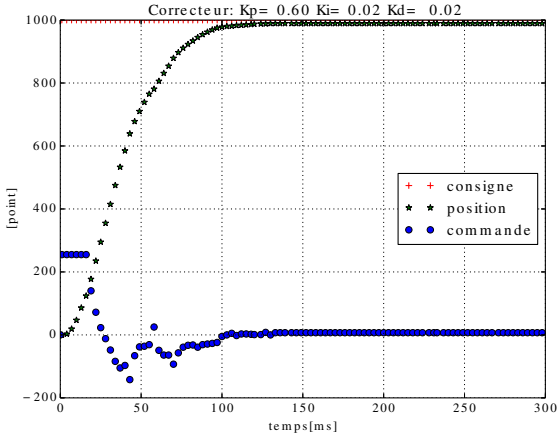









Correcteur: Kp= 0.40 Ki= 0.01 Kd= 0.02
[point]
temps[ms]
consigne
position
commande


Correcteur: Kp= 0.60 Ki= 0.02 Kd= 0.02
[point]
temps[ms]
consigne
position
commande