PTSI / Informatique / DS N° 2 - Correction

Analyse harmonique

1 Diagramme de Bode

Question 1. Quelles sont les fonctions de transfert dont le diagramme de Bode est représenté sur la figure 1 ?

```
num = [1]
for i in range(5):
    den = [1,0.1+i/5,1]
```

Les fonctions de transfert sont donc :
$H_1(p) = \frac{1}{p^2+0.1p+1}$; $H_2(p) = \frac{1}{p^2+0.3p+1}$; $H_3(p) = \frac{1}{p^2+0.5p+1}$;
$H_4(p) = \frac{1}{p^2+0.7p+1}$;$H_5(p) = \frac{1}{p^2+0.9p+1}$;

Question 2. A une étape de l'algorithme proposé,
1. combien de données contiennent les listes `w`, `gain` et `phase` ?
2. en admettant que chacune de ces données est de type `float` codé en double précision, quelle quantité de mémoire est nécessaire pour le stockage de ces trois listes ?

1. `np.arange(0.1, 10, 0.02)`
   nombre de données : $\frac{9.98-0.1}{0.02}+1 = 495$
2. en admettant que chacune de ces données est de type `float` codé en double précision, quelle quantité de mémoire est nécessaire pour le stockage de ces trois listes ?
   Le codage en double précision se fait sur 64 bits.
   La quantité de mémoire nécessaire est donc : $\frac{3\times495\times64}{8\times1000} = 11.88$ ko.

2 Propriétés caractéristiques du système

2.1 Asymptote infinie de la courbe en gain

Question 3. Proposer un script permettant de donner la valeur de l'asymptote lorsque $\omega \to \infty$ de la courbe de gain en $dB/decade$ (décibels par décade).

```
print((gain[-1]-gain[-2])*(np.log(10))/(np.log(w[-1])-np.log(w[-2])))
```

ou

```
(gain[494]-gain[493])*(np.log(10))/(np.log(w[494])-np.log(w[493]))
```

2.2 Résonance

Question 4. Dans le contexte de la figure 1, écrire la fonction `picResonance(w,gain,phase)` retournant pour une courbe un triplet de valeurs respectives la pulsation de résonance wr, le gain maximal gr et la phase correspondante pr :`(wr,gr,pr)`. Dans le cas où ce pic n'existerait pas, elle retourne un triplet vide.

```
def picResonance(w,gain,phase):
    n=len(w)
    gr=gain[0]
    i=1
    while i<n and gr<=gain[i]:
        gr=gain[i]
        i+=1
    if i==1:
        return ()
    else:
        return (w[i-1],gr,phase[i-1])
```

Question 5. Dans le cas où le diagramme en gain serait multi-pics, écrire la fonction `picsResonance(w,gain,phase)` retournant une liste `L` de triplets de valeurs respectives une pulsation de résonance *wr*, le gain *gr* et la phase correspondante *pr* :`(wr,gr,pr)`. La liste `L` peut présenter l'allure suivante : `[(wr1,gr1,pr1),(wr2,gr2,pr2),...,(wrk,grk,prk)]`. Dans le cas où il n'y aurait aucun pics, la fonction retourne une liste vide.

```
def picResonance(w,gain,phase):
    n=len(w)
    gr=gain[0]
    L=[]
    i=1
    while i<n:
        if gain[i-1]<=gain[i]:
            i+=1
            while i<n and gain[i-1]<=gain[i]:
                i+=1
            if i<n:
                L.append((w[i-1],gain[i-1],phase[i-1]))
        i+=1
    return L
```

2.3 Bande passante

Question 6. Écrire la fonction `pulsationCoupure(w,gain)` retournant la valeur de la pulsation de coupure `wc` en utilisant une méthode par balayage de la liste `gain`. On s'assurera de la terminaison de l'algorithme. Si la pulsation de coupure n'existe pas, on retourne -1.

```
def pulsationCoupure(w,gain):
    n=len(w)
    i=0
    while i<n and gain[i]+3>0:
        i+=1
    if i==n:
        return -1
    else :
        return w[i-1]
```

Question 7. Proposer une fonction `pulsationCoupure(w,gain)` retournant la valeur de la pulsation de coupure `wc` en utilisant une méthode de dichotomie sur la liste `gain`. Si la pulsation de coupure n'existe pas, on retourne -1.

```
1  def pulsationCoupure(w,gain):
2      a=0
3      b=len(w)
4      while b-a>1:
5          m=(a+b)//2
6          if (gain[a]+3)*(gain[m]+3)<0:
7              b=m
8          else:
9              a=m
10     if a==len(w)-1:
11         return -1
12     else :
13         return w[a]
```

Question 8.

1. Pour l'algorithme précédent, définir un variant de boucle et effectuer une preuve de terminaison,
2. Définir un invariant de boucle et effectuer une preuve de correction.
3. Quel intérêt présente la méthode de dichotomie par rapport à une méthode par balayage ?

1. On définit comme variant de boucle la fonction $f : a,b \to b-a$. Cette fonction est strictement décroissante à chaque itération puisque l'intervalle $[a,b]$ est divisé par deux à chaque tour de boucle : il devient soit $[a,m]$, soit $[m,b]$.
   Au bout de k itérations telles que $len(w) \leq 2^k$ (soit $log_2(len(w))+1 > k \geq log_2(len(w))$), $f(a,b)$ sera égal à 1. On sort alors de la boucle `while`.
2. On définit comme invariant de boucle la propriété suivante :
   (P_i) : A chaque itération i, la pulsation de coupure si elle existe, appartient à l'intervalle $[w[a];w[b]]$.
   Cette propriété est vérifiée à l'initialisation puisque a et b sont les limites inférieures et supérieures des indices des listes `w` et `gain`. (P_0) est donc vraie.
   Supposons vraie la propriété (P_i). A l'itération suivante, si `(gain[a]+3)*(gain[m]+3)<0` alors c'est que la pulsation de coupure appartient à l'intervalle $[w[a];w[m]]$. On remplace donc b par m. Si ce n'est pas le cas, c'est qu'elle appartient à l'intervalle $[w[m];w[b]]$. On remplace donc a par m. La propriété (P_{i+1}) sera alors aussi vérifiée.
   A la sortie de la boucle `while`, si `a==len(w)-1`, c'est que ω_c n'appartient pas à $[w[a];w[b]]$. Sinon, c'est que ω_c est $w[a]$] ou $w[b]$. Le résultat est donc donné à 0.02 rad/s près.
3. La méthode par dichotomie nécessite moins d'itérations (9 au maximum pour 495 données numériques) que par un balayage de la liste.

3 Stockage des données dans un fichier texte

Question 9.
1. Quelle sera la taille approximative du fichier texte `"bode.txt"` sachant que le nombre de données est identique à celui de la question 2 ?
2. Quel serait le gain de taille en % si on se limitait à 3 chiffres significatifs pour chacune des valeurs numériques ?

Chaque ligne comporte environ 36 caractères (fin de ligne inclus). Sans compter la première ligne, cela fait $495 * 36 = 17820$ caractères. Chacun est codé en ASCII sur un octet. Cela fait donc environ 18 ko.

Question 10. Écrire un script utilisant les listes `w`, `gain` et `phase`, et permettant de créer le fichier "bode.txt". A la fin de l'écriture, on assurera sa fermeture "propre".

```
f=open("bode.txt","w")
f.write("pulsation"+";"+"gain"+";"+"phase"+"\n")
for i in range(len(w)):
    f.write(str(w[i])+";"+str(gain[i])+";"+str(phase[i])+"\n")
f.close()
```

Question 11. Proposer une fonction `retourneListes(nomFichier)` prenant en argument une chaîne de caractère `nomFichier` et retournant les listes `w` (pulsations) et `gain` (gains). A la fin de la lecture, on assurera la fermeture "propre" du fichier.

```
def retourneListes(nomFichier):
    f=open(nomFichier,"r")
    f.readline()
    w=[]
    gain=[]
    for x in f:
        L=x.split(";")
        w.append(float(L[0]))
        gain.append(float(L[1]))
    f.close()
    return w,gain
```