DS04

Algorithmique et programmation

Sources :

Proposition de corrigé

Exercice 1 : Analyse harmonique

Q 1 :

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

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

Q 2 :

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.

Q 3 :

```
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]))
```

Q 4 :

- **Proposition d'invariant :** On peut proposer un invariant d'entrée de boucle classique pour la recherche de maximum.
  `gr` *est le maximum de la liste* `gain` *pour les indices allant de* 0 *à* $i-1$ *et correspond à* `gain[i-1]`.
- **Preuve de correction :**
  On peut montrer que cet invariant est vrai par récurrence.
  - A l'initialisation, $i=1$ et `gr=gain[0]` qui ne contient bien qu'un seul terme et donc `gr` correspond bien au maximum et correspond bien à `gain[i-1]`.
  - On suppose l'invariant vrai à l'entrée de boucle. Montrons qu'il est vrai à la sortie de la boucle. Pour entrer dans la boucle while, la condition `gr<=gain[i]` est vrai et on affecte à `gr` la valeur `gain[i]`, et `i` est

itéré de 1, ainsi `gr` *est le maximum de la liste* `gain` *pour les indices allant de* 0 *à* $i-1$ *et il s'agit bien de* `gain[i-1]`.
  - A la sortie de la boucle, la condition `gr<=gain[i]` ou `i<n` est fausse.
    dans le premier cas le maximum est bien `gr=gain[i-1]`, il faut donc bien renvoyer `w[i-1]`, `gr` et `phase[i-1]`.
    dans le deuxième cas `i=n` et donc on a atteint le dernier terme.
    ainsi si `gr<=gain[i]` est vrai on revient au cas précédent et si `gr<=gain[i]` est faux, alors le maximum est bien `gain[i-1]`.
- **Proposition de variant et preuve de terminaison**
  La condition `while` possède deux conditions reliés par le "et" logique. On peut donc utiliser deux variants de boucle mais un seul suffit pour montrer la terminaison.
  on peut utiliser la suite `i-n` comme variant qui est une suite d'entiers strictement croissante donc elle devient positive au bout d'un moment.

Q 5 :

```
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
```

Q 6 :

```
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]
```

Q 7 :

```
def pulsationCoupure(w,gain):
    a=0
    b=len(w)
    while b-a>1:
        m=(a+b)//2
        if (gain[a]+3)*(gain[m]+3)<0:
            b=m
        else:
            a=m
    if a==len(w)-1:
        return -1
    else :
        return w[a]
```

Q 8 :

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.

Q 9 :

```
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()
```

Q 10 :

```
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
```