teaching sciences for innovation

Proposition de corrigé

Concours : Banque PT

Année : 2023

Filière : PT

Épreuve : Informatique et Modélisation

Ceci est une proposition de corrigé des concours de CPGE, réalisée bénévolement par des enseignants de Sciences Industrielles de l'Ingénieur et d'Informatique, membres de l'UPSTI (Union des Professeurs de Sciences et Techniques Industrielles), et publiée sur le site de l'association :

https://www.upsti.fr/espace-etudiants/annales-de-concours

A l'attention des étudiants

Ce document vous apportera des éléments de corrections pour le sujet traité, mais n'est ni un corrigé officiel du concours, ni un corrigé détaillé ou exhaustif de l'épreuve en question.

L'UPSTI ne répondra pas directement aux questions que peuvent soulever ces corrigés : nous vous invitons à vous rapprocher de vos enseignants si vous souhaitez des compléments d'information, et à vous adresser à eux pour nous faire remonter vos éventuelles remarques.

Licence et Copyright

En cas de doute, n'hésitez pas à nous contacter à : corrigesconcours@upsti.fr.

Informez-vous !

Retrouvez plus d'information sur les Sciences de l'Ingénieur, l'orientation, les Grandes Ecoles ainsi que sur les Olympiades de Sciences de l'Ingénieur et sur les Sciences de l'Ingénieur au Féminin sur notre site : www.upsti.fr

L'équipe UPSTI

Elasticité d'un brin d'ADN

Corrigé UPSTI

Partie Informatique

Fonctions de base

Question 18 Pour quel intervalle de déplacement de la bille D_b la tension U est-elle une loi affine du déplacement ? Donner alors a et b tels que : $U = aD_b + b$.

Sur l'intervalle de déplacement $D_b \in [1,5;2,1]\mu$m, la tension U est proche de la droite $U = aD_b + b$ avec $a = \frac{6-(-1)}{1,6-2,0} = -17,5\,\text{mV}/\mu\text{m}$ et $b = 36\,\text{mV}$.

Question 19 Quel est l'ordre de grandeur de la résolution de la mesure de D_b permise par cette mesure ?

U est compris approximativement entre -4 et $9\,$mV, soit une plage de l'ordre de $16\,\text{mV} = 2^4$.

Un codage sur 16 bits permet de représenter 2^{16} valeurs différentes.

La résolution de la mesure de U sera donc de $\frac{2^4}{2^{16}} = 2^{-12}$ en mV.

La résolution de la mesure de D_b sera donc de $\frac{2^{-12}}{a} = \frac{1}{4096\times17,5} \approx \frac{1}{7\times10^5} \approx 10^{-6}\mu$m.

Question 20 Écrire une fonction `somme(a)` qui prend pour argument une liste et retourne la somme de ses éléments.

```
1  def somme(a):
2          s = 0 # Valeur de la somme partielle
3          for i in a:
4                  s += i
5          return s
```

Question 21 Écrire une fonction `moyenne(a)` qui prend pour argument une liste et retourne la moyenne de ses éléments.

```
6  def moyenne(a):
7          # On ignore le cas de la liste a vide
8          return somme(a)/len(a)
9
10 # Rien n'interdit la fonction len(), mais si besoin
11 def len(a):
12         nb = 0 # Compteur du nombre d'éléments de a
13         for i in a:
14                 nb += 1
15         return nb
```

Question 22 Écrire une fonction `variance(a)` qui prend pour argument une liste et retourne la variance de ses éléments.

```
16 def variance(a):
17         s = 0 # Valeur de la somme partielle des carrés
```

```
        for i in a:
                s += i**2

        return s/n - moyenne(a)**2
```

Méthode 1 : oscillations forcées de la bille dans un liquide

Question 23 Montrer que :

$$a_1 = \frac{\sum_{i=0}^{n-1} x_i \sum_{i=0}^{n-1} y_i - n \sum_{i=0}^{n-1} x_i y_i}{\left(\sum_{i=0}^{n-1} x_i\right)^2 - n \sum_{i=0}^{n-1} x_i^2}$$

et

$$a_0 = \overline{Y} - a_1 \overline{X}$$

On souhaite trouver a_0 et a_1 qui minimisent la fonction $e = \sum_{i=0}^{n-1} (y_i - a_0 - a_1 x_i)^2$.

Calcul des dérivées partielles : $e = \sum_{i=0}^{n-1} (y_i - a_0 - a_1 x_i)^2 = \sum_{i=0}^{n-1} \left(y_i^2 - 2y_i a_0 - 2y_i x_i a_1 + 2a_0 a_1 x_i + a_0^2 + a_1^2 x_i^2\right)$

$$\begin{cases} \dfrac{\partial e}{\partial a_0} = \sum_{i=0}^{n-1} (-2y_i + 2a_1 x_i + 2a_0) \\ \dfrac{\partial e}{\partial a_1} = \sum_{i=0}^{n-1} \left(-2y_i x_i + 2a_0 x_i + 2a_1 x_i^2\right) \end{cases}$$

La valeur de e est minimale lorsque ses dérivées partielles sont nulles :

$$\begin{cases} \dfrac{\partial e}{\partial a_0} = 0 = \sum_{i=0}^{n-1} (-2y_i + 2a_1 x_i + 2a_0) \\ \dfrac{\partial e}{\partial a_1} = 0 = \sum_{i=0}^{n-1} \left(-2y_i x_i + 2a_0 x_i + 2a_1 x_i^2\right) \end{cases}$$

$$\begin{cases} 0 = \sum_{i=0}^{n-1} (-y_i + a_1 x_i + a_0) \\ 0 = \sum_{i=0}^{n-1} \left(-y_i x_i + a_0 x_i + a_1 x_i^2\right) \end{cases}$$

$$\begin{cases} \sum_{i=0}^{n-1} a_0 + \sum_{i=0}^{n-1} a_1 x_i = \sum_{i=0}^{n-1} y_i \\ \sum_{i=0}^{n-1} a_0 x_i + \sum_{i=0}^{n-1} a_1 x_i^2 = \sum_{i=0}^{n-1} y_i x_i \end{cases}$$

La première équation du système donne directement l'expression recherchée de a_0 :

$$\begin{aligned}\sum_{i=0}^{n-1} a_0 + \sum_{i=0}^{n-1} a_1 x_i &= \sum_{i=0}^{n-1} y_i \\ \sum_{i=0}^{n-1} a_0 &= \sum_{i=0}^{n-1} y_i - \sum_{i=0}^{n-1} a_1 x_i \\ n \times a_0 &= n \times \overline{Y} - a_1 \times n \times \overline{X} \\ a_0 &= \overline{Y} - a_1 \times \overline{X}\end{aligned}$$

En injectant cette expression dans la deuxième équation :

$$\begin{cases}\displaystyle\sum_{i=0}^{n-1} a_0 x_i + \sum_{i=0}^{n-1} a_1 x_i^2 = \sum_{i=0}^{n-1} y_i x_i \\ \displaystyle a_0 \sum_{i=0}^{n-1} x_i + a_1 \sum_{i=0}^{n-1} x_i^2 = \sum_{i=0}^{n-1} y_i x_i \\ \displaystyle a_1 \sum_{i=0}^{n-1} x_i^2 = \sum_{i=0}^{n-1} y_i x_i - a_0 \sum_{i=0}^{n-1} x_i \\ \displaystyle a_1 \left(\sum_{i=0}^{n-1} x_i^2 - \overline{X} \sum_{i=0}^{n-1} x_i \right) = \sum_{i=0}^{n-1} y_i x_i - \overline{Y} \sum_{i=0}^{n-1} x_i \\ \displaystyle a_1 \left(\sum_{i=0}^{n-1} x_i^2 - \frac{1}{n} \left(\sum_{i=0}^{n-1} x_i \right)^2 \right) = \sum_{i=0}^{n-1} y_i x_i - \overline{Y} \sum_{i=0}^{n-1} x_i \\ \displaystyle a_1 \left(\sum_{i=0}^{n-1} x_i^2 - \frac{1}{n} \left(\sum_{i=0}^{n-1} x_i \right)^2 \right) = \sum_{i=0}^{n-1} y_i x_i - \frac{1}{n} \sum_{i=0}^{n-1} y_i \sum_{i=0}^{n-1} x_i \\ \displaystyle a_1 \left(n \sum_{i=0}^{n-1} x_i^2 - \left(\sum_{i=0}^{n-1} x_i \right)^2 \right) = n \sum_{i=0}^{n-1} y_i x_i - \sum_{i=0}^{n-1} y_i \sum_{i=0}^{n-1} x_i \\ \displaystyle a_1 = \frac{\sum_{i=0}^{n-1} y_i \sum_{i=0}^{n-1} x_i - n \sum_{i=0}^{n-1} y_i x_i}{\left(\sum_{i=0}^{n-1} x_i \right)^2 - n \sum_{i=0}^{n-1} x_i^2}\end{cases}$$

Question 24 Écrire une fonction `regression_lineaire(x,y)` qui prend pour arguments deux listes d'abscisses x et d'ordonnées y et qui retourne les coefficients a_0 et a_1 tels que définis ci-dessus.

```
22 def regression_lineaire(x,y):
23         mX = moyenne(x)
24         mY = moyenne(y)
25 
26         numerateur = 0
27         denominateur = 0
28         for i in range(len(x)):
29                 numerateur += (x[i] - mX)*(y[i] - mY)
30                 denominateur += (x[i] - mX)**2
31 
32         a1 = numerateur/denominateur
33         a0 = mY - a1 * mX
34         return a0, a1
35 
```

Question 25 Évaluer l'ordre de la complexité d'un appel à la fonction `regression_lineaire` en fonction de la taille des données n.

Les deux calculs de moyenne sont en $\mathcal{O}(n)$ opérations élémentaires. La boucle `for` est aussi en $\mathcal{O}(n)$ opérations élémentaires.

La complexité de cette fonction en fonction de la taille des données n est donc linéaire (en $\mathcal{O}(n)$ opérations élémentaires).

Remarque

Si le calcul de la moyenne était réalisé à chaque itération de la boucle, alors la complexité serait en $\mathcal{O}(n^2)$ opérations élémentaires.

Question 26 Écrire un script qui permet de tracer la figure ci-dessus à partir de la liste des tensions U et des fréquences f correspondantes, les barres d'erreur ne sont pas à représenter.

```
a1, a2 = regression_lineaire(f, U)
plt.plot(f, U, 'ks', linestyle = 'none') # Trace avec des carrés noirs (k = black), sans relier les
    points

# On récupère les coordonnées min/max des fréquences, pour tracer la régression linéaire entre ces deux
    points
U0 = f[0] * a_1 + a_0
Um = f[-1] * a_1 + a_0
plt.plot([f[0], f[-1]], [U0, Um], linestyle = 'dashed') # Droite de régression en traits interrompus

plt.xlabel("Fréquence (Hz)")
plt.ylabel("Tension U (microV)")

plt.show()
```

Remarque

L'aide fournie en fin de sujet ne couvre pas toute la syntaxe nécessaire pour tracer la figure du sujet.

Question 27 Écrire une fonction `traitement(a)` qui prend pour argument une liste de liste de dimension $n \times l$ et retourne une liste de liste de même dimension calculant la moyenne et la variance pour tout élément de a. On aura donc `len(traitement(a)[i])=2`.

```
def traitement(a):
        res = []
        for i in a:
                m = moyenne(i)
                v = variance(i)
                res.append([m,v])
        return res
```

Question 28 Écrire une fonction `regression_lineaire_gen(x,yv,listf)` qui prend pour argument une liste d'abscisses x, une liste de liste d'ordonnées et leur variances yv, une liste de fonction `listf` et qui retourne les coefficients a de la régression linéaire comme défini ci-dessus. On rappelle qu'une documentation partielle est disponible en annexe.

Les arguments sont des listes Python, mais la documentation porte sur des array (tableaux) numpy : il faut donc les transtyper.

```python
def regression_lineaire_gen(x,yv,listf):
        n = len(x)
        m = len(listf)
        # Construction de y et V
        y = np.empty(n, float)
        V = np.zeros((n, n), float)
        for i in range(n):
            y[i] = yv[0]
            V[i,i] = 1/yv[1]

        # Construction de J
        J = np.zeros((n, m), float)
        for i in range(n):
            for j in range(m):
                J[i,j] = listf[j](x[i])

        Jt = np.transpose(J)
        # Calcul de a
        a = np.dot(Jt, V)
        a = np.dot(a, J)
        a = np.linalg.inv(a)
        a = np.dot(a, Jt)
        a = np.dot(a, V)
        a = np.dot(a, y)
```

Question 29 Évaluer l'ordre de la complexité d'un appel à la fonction `regression_lineaire_gen()` pour m et n quelconques. Comparer à la question 25 pour $m = 2$.

Les opérations les plus coûteuses de cette fonction sont :

- La création de matrices $n \times n$ et $n \times m$, dont la complexité est en $\mathcal{O}(n^2)$ ou $\mathcal{O}(n \times m)$ opérations élémentaires, tout comme le calcul de J et sa transposition.
- Les calculs matriciels, qui sont, suivant les matrices et l'ordre dans lequel ils sont réalisés, en $\mathcal{O}(n^3)$ ou $\mathcal{O}(n \times m^2)$ ou $\mathcal{O}(n^2 \times m)$ opérations élémentaires.
- L'inversion de la matrice $J^T V J$, de dimension $(n \times n)$, qui doit être en $\mathcal{O}(n^3)$ opérations élémentaires (que ce soit fait par un Gauss ou par bloc...).

Donc la complexité d'un appel à cette fonction est de l'ordre de $\mathcal{O}(n^3 + n \times m^2)$ opérations élémentaires. Si m est petit devant n (donc par exemple si $m = 2$), alors c'est en $\mathcal{O}(n^3)$ opérations élémentaires, soit moins efficace que lalgorithme question 25 qui est en $\mathcal{O}(n)$ opérations élémentaires.

Méthode 2 : analyse du mouvement brownien d'une bille piégée

Question 30 En transformant l'équation de Langevin dans le domaine de Laplace harmonique (à condition initiale nulles) avec $p = j\omega$, exprimer le spectre de puissance du mouvement brownien de la bille $S_x(\omega) = |\widehat{x}(j\omega)^2|$ en fonction de $S_f(\omega)$, γ et k.

En transformant l'équation de Langevin dans le domaine de Laplace, avec des conditions initiales nulles(conditions de Heaviside) :

$$\gamma \cdot p \cdot x(p) + k \cdot x(p) = F_l(p) \Rightarrow x(p) = \frac{F_l(p)}{k + \gamma \cdot p}$$

Dans le domaine harmonique, son module au carré vaut :

$$S_x(\omega) = \left|\widehat{x}(j\omega)^2\right| = \frac{\left|\widehat{F_l}(j\omega)\right|^2}{k^2 + \gamma^2\omega^2} = \frac{S_f(\omega)}{k^2 + \gamma^2\omega^2}$$

Question 31 Montrer que $S_x(f) = \frac{4k_bT}{\pi^2\gamma}\frac{1}{f^2+f_c^2}$ où $f = \frac{\omega}{2\pi}$ est la fréquence du signal. Donner l'expression de f_c.

En substituant ω par $2\pi f$, et $S_f(\omega)$ par son expression :

$$S_x(f) = \frac{4\gamma k_bT}{k^2+\gamma^2 4\pi^2 f^2} = \frac{4\gamma k_bT}{\gamma^2 4\pi^2}\frac{1}{\frac{k^2}{\gamma^2 4\pi^2}+f^2} = \frac{k_bT}{\gamma\pi^2}\frac{1}{f^2+\frac{k^2}{\gamma^2 4\pi^2}}$$

Avec $f_c = \frac{k}{2\pi\gamma}$

Remarque

Il semble y avoir une erreur avec le facteur 4 au numérateur dans le sujet.

Question 32 Écrire une fonction `aire(x,y)` qui détermine l'aire sous la courbe définie par une liste d'abscisses x et d'ordonnées y de même dimension. Vous expliciterez clairement la méthode choisie.

On utilise la méthode des rectangles à gauche.

```
83 def aire(x,y):
84         # On suppose que les abscisses x sont triées par valeurs croissantes.
85         aire = 0.0
86 
87         for i in range(len(x)-1): # On n'utilisera pas le dernier point avec cette méthode
88                 aire += y[i] * (x[i+1] - x[i])
89 
90         return aire
```

Autre méthode : On utilise la méthode des trapèzes.

```
92 def aire(x,y):
93         # On suppose que les abscisses x sont triées par valeurs croissantes.
94         aire = 0.0
95 
96         for i in range(len(x)-1):
97                 aire += (y[i+1] + y[i]) * (x[i+1] - x[i])
98 
99         return aire / 2
```

Question 33 Donner les expressions analytiques de $\frac{\partial G}{\partial a_0}$ et $\frac{\partial G}{\partial a_1}$.

$$\frac{\partial G}{\partial a_0} = \frac{1}{a_1^2+f^2} \qquad \frac{\partial G}{\partial a_1} = -\frac{2a_0a_1}{\left(a_1^2+f^2\right)^2}$$

Question 34 Écrire les fonctions `G(f,a0,a1)`, `dG_da0(f,a0,a1)` et `dG_da1(f,a0,a1)` qui retournent respectivement la valeur de G, $\frac{\partial G}{\partial a_0}$ et $\frac{\partial G}{\partial a_1}$ en fonction de leurs arguments.

```
101 def G(f,a0,a1):
102         return a0 / (a1**2 + f**2)
103 
104 def dG_da0(f,a0,a1):
105         return 1 / (a1**2 + f**2)
106 
107 
108 def dG_da1(f,a0,a1):
109         return - 2*a0*a1 / ((a1**2 + f**2)**2)
```

Question 35 Donner les expressions analytiques de $\frac{\partial e}{\partial a_0}$ et $\frac{\partial e}{\partial a_1}$, en fonction de $\frac{\partial G}{\partial a_0}$ et $\frac{\partial G}{\partial a_1}$ et des données.

Par dérivation de e :

$$\frac{\partial e}{\partial a_0} = \sum_{i=0}^{n-1} \left(-2S_{ui} + 2G(f_i, a_0, a_1) \frac{\partial G}{\partial a_0}(f_i, a_0, a_1) \right)$$

$$\frac{\partial e}{\partial a_1} = \sum_{i=0}^{n-1} \left(-2S_{ui} + 2G(f_i, a_0, a_1) \frac{\partial G}{\partial a_1}(f_i, a_0, a_1) \right)$$

Question 36 Écrire une fonction `methode_gradient(G,dG_da0,dG_da1,Su,f,h,a00,a10,err)` qui détermine et retourne les paramètres optimaux a_0 et a_1.

Remarque

On souhaite annuler le gradient, la condition d'arrêt semble donc être erronée : il manque une valeur absolue.

```
111 def de(Su, f, G, a0, a1, dG_da0,dG_da1): # Calcul des deux dérivées partielles de e
112         de_a0 = 0
113         de_a1 = 0
114         for i in range(len(f)):
115                 de_a0 += -2 * Su[i] + 2 * G(f[i], a0, a1) * dG_da0(f[i], a0, a1)
116                 de_a1 += -2 * Su[i] + 2 * G(f[i], a0, a1) * dG_da0(f[i], a0, a1)
117
118         return de_a0, de_a1
119
120 def methode_gradient(G,dG_da0,dG_da1,Su,f,h,a00,a10,err):
121         a0, a1 = a00, a10
122         gradient = 1 + err / h # Initialisation du gradient pour réaliser au moins une itération
123         de_a0, de_a1 = de(Su, f, G, a0, a1, dG_da0,dG_da1)
124
125         while abs(gradient) < err/h:
126                 a0 = a0 - h * de_a0
127                 a1 = a1 - h * de_a1
128                 de_a0, de_a1 = de(Su, f, G, a0, a1, dG_da0,dG_da1)
129                 gradient = de_a0 + de_a1
130
131         return a0, a1
```

Question 37 En considérant que i_T itérations sont nécessaires pour atteindre la condition d'arrêt de l'algorithme, donner la complexité de la méthode du gradient en fonction de la taille n des données expérimentales

La boucle de la fonction `methode_gradient` itère i_T fois, et appelle à chaque fois la fonction `de`, qui réalise n itérations.

Elle est donc de complexité en $\mathcal{O}(i_T \times n)$ opérations élémentaires.

Question 38 Écrire une fonction `force(Su,f,h,a00,a10)` qui détermine et retourne la force exercée sur la bille.

On a : $F = U\sqrt{2\pi\gamma k_b T \frac{f_c}{A}}$, avec :

- U est la tension mesurée (connue) ;
- k_b est la constante de Boltzmann (connue) ;
- T est la température (connue) ;
- A, qui est l'aide sous la courbe du spectre de puissance expérimental S_u, que l'on peut estimer par la méthode des rectangles ou des trapèzes (cf question 32).

- f_c est une contante, que l'on peut déterminer par la méthode de descente du gradient (valeur de a_1, cf question 36) ;
- γ est une constante, inconnue, que l'on peut aussi déterminer via la méthode de descente du gradient (valeur de $a_0 = \frac{4k_bT}{\pi^2\gamma}$, donc $\gamma = \frac{4k_bT}{\pi^2 a_0}$).

```
132 def force(Su, f, h, a00, a10):
133
134         # Détermination de gamma et fc
135         err = 10**-5 # Définition de l'erreur admissible
136
137         a0, fc = methode_gradient(G,dG_da0,dG_da1,Su,f,h,a00,a10,err)[1]
138         gamma = 4 * kb * T / (a0 * math.pi ** 2 )
139
140         # Détermination de A
141         A = aire(Su, f)
142
143         # Calcul de la force
144         F = U * math.sqrt(2*math.pi * gamma * kb * T * fc / A)
145
146         return F
```