Devoir - Bilan 1er semestre

I - Étude de la dispersion des valeurs de résistance d'un stock de résistors

1 Introduction

2 Densité de probabilité

Q1. Pour les 4 propositions suivantes, précisez sur votre copie celles qui sont vraies et celles qui sont fausses.

Prop 1 : Dans le cas de la FIGURE 2, il y a quasiment 0% de chance qu'un résistor ait **exactement** une valeur de 150 Ω.

Prop 2 : Dans le cas de la FIGURE 2, il y a 50% de chance qu'un résistor ait une valeur inférieure ou égale à 100 Ω.

Prop 3 : On peut dire que la probabilité qu'un résistor ait une valeur supérieure à 95 Ω est de $1 - \mathcal{P}_r(95)$.

Prop 4 : Dans le cas de la FIGURE 2, la probabilité qu'un résistor ait **exactement** une valeur de 120 Ω est d'environ 5%.

Prop 1 Vrai.

Prop 2 Vrai.

Prop 3 Vrai.

Prop 4 Faux, elle est de $0,5\%$.

Q2. Définir une fonction `dens(R:float,mu:float,sigma:float)->float` permettant de calculer $dens(R)$ pour des valeurs d'espérance `mu` et d'écart-type `sigma` données. Ainsi, `dens(95,100,10)` renverra 0,0352.

Pour cette question, il suffit de transcrire la définition mathématique de la densité de probabilité *dens*.

```
1 import math as m
2 def dens(R,mu,sigma):
3     '''Fonction permettant de calculer la probabilité de valeur de résistance.
4     Entrée : R la valeur de la résistance testée, mu et sigma les paramètres de
          la loi normale.
5     Sortie : Evaluation de la densité de probabilité pour R.'''
6     return 1/(sigma*m.sqrt(2*m.pi))*m.exp(-1/2*((R-mu)/sigma)**2)
```

Q3. Créez une liste de 101 éléments allant de 50 à 150 inclus. Cette liste sera nommée `abscisse` et sera la liste utilisée comme abscisse pour tracer le graphe de la densité de probabilité.

On va créer cette liste par compréhension. Une difficulté de cette question est de déterminer le pas

---

entre deux éléments de la liste. On demande à ce qu'il y ait 101 éléments, il y a donc 100 espacements entre les éléments. De ce fait, le pas est :

$$pas = \frac{150 - 50}{100} = 1$$

On construit alors la liste `abscisse` par :

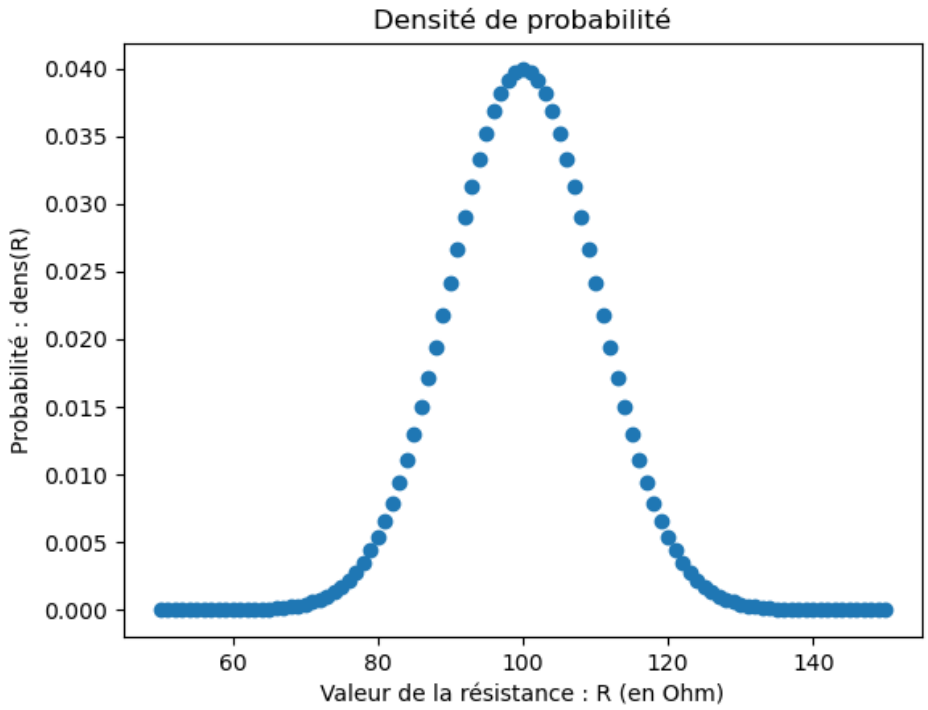
```
1 | abscisse=[50+1*i for i in range(101)]
```

Q4. Créez maintenant la liste `ordonnee` qui va contenir les 101 évaluations de la fonction *dens* de la liste `abscisse`. (On rappelle que `mu`=100 et `sigma`=10, pour créer cette figure.)

On peut créer très facilement cette liste également par compréhension.

```
1 | ordonnee=[dens(R,100,10) for R in abscisse]
```

Q5. D'après la documentation Matplotlib fournie en annexe, donnez le code permettant de tracer la courbe ci-dessous. Vous serez attentifs à fournir le titre du graphe, ainsi que celui des axes des abscisses et des ordonnées.



```
1 | import matplotlib.pyplot as plt
2 | plt.plot(abscisse,ordonnee,'o')
3 | plt.xlabel("Valeur de la résistance : R (en Ohm)")
4 | plt.ylabel("Probabilité : dens(R)")
5 | plt.title("Densité de probabilité")
6 | plt.show()
```

---

3 Méthodes d'intégration numérique d'une fonction

Q6. Exprimez *pas* en fonction des bornes **a** et **b** et du nombre n de rectangles/trapèzes.

Il suffit de réaliser l'opération suivante :

$$pas = \frac{\mathbf{b} - \mathbf{a}}{n}$$

Q7. Exprimez **g** et **d** en fonction de i, **a** et de *pas*. i correspond au numéro du rectangle considéré.

Dans l'annexe, on constate que i commence à 0. Ce qui signifie que pour la première subdivision, celle-ci est indexée par la valeur 0. De ce fait, la première valeur prise par **g** et **d** sont respectivement **a** et $\mathbf{a} + pas$. La relation permettant d'exprimer **g** et **d** en fonction de i, **a** et de *pas* est alors :

$$\mathbf{g} = \mathbf{a} + i \times pas \qquad \text{et} \qquad \mathbf{d} = \mathbf{a} + (i+1) \times pas$$

Q8. D'après l'annexe 1, pour la méthode dite des rectangles à gauche, donnez l'expression mathématique de l'aire $\mathcal{A}_i$ du rectangle i en fonction de **g**, *pas* et f.

$\mathcal{A}_i = f(\mathbf{g}) \times pas$

Q9. Écrivez une fonction `integration_rectangle_gauche(f:function,a:float,b:float,n:int)->float` permettant de calculer l'intégrale d'une fonction f à valeur $f(x)$ pour $x \in [a, b]$ avec n rectangles et s'appuyant sur la méthode dite des rectangles à gauche.

L'aire d'un rectangle est définie par $pas \times f(\mathbf{g})$ avec **g** qui varie selon le rectangle considéré et la réponse fournie précédemment. Le calcul d'intégrale va donc se faire en réalisant la somme de ces aires sur les n rectangles.

```
1  def integration_rectangle_gauche(f,a:float,b:float,n:int)->float:
2      '''Fonction permettant d'approximer le calcul de l'intégrale sur l'
           intervalle [a,b] d'une fonction f continue et intégrable à l'aide de la
           méthode des rectangles à gauche.
3      Entrées : f : la fonction étudiée, a : borne inférieure de l'intervalle d'é
           tude, b : borne supérieure de l'intervalle d'étude, n : nombre de
           rectangles utilisés dans la méthode.
4      Sortie : Somme des aires des rectangles approximant la valeur de l'intégrale
           .'''
5      pas=(b-a)/n
6      somme_aire=0
7      for i in range(n):
8          g=a+pas*i
9          d=g+pas # ou encore d=a+pas*(i+1)
10         somme_aire+=f(g)*pas # f(g)*pas correspond à l'aire du rectangle i
               considéré
11     return somme_aire
```

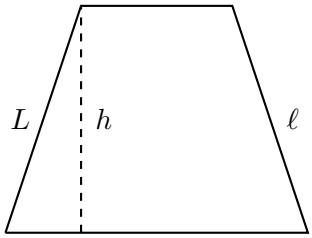
Q10. Quelle(s) modification(s) faut-il faire pour réaliser l'intégration avec la méthode des rectangles à droite et des trapèzes ? Donner les lignes de code correspondantes pour réaliser ces modifications.

Pour la méthode des rectangles dite à droite, l'aire d'un rectangle i est égale à $pas \times f(\mathbf{d})$. Ainsi, la structure du code précédent n'est pas à toucher. Il suffit de modifier la ligne `somme_aire+=f(g)*pas` par `somme_aire+=f(d)*pas`. Pour la méthode des trapèzes, là encore la structure du code ne sera

pas à modifier. La seule « difficulté » consiste à calculer l'aire d'un trapèze. Pour rappel, l'aire d'un trapèze est donnée par :

$$\frac{(L+\ell)\times h}{2}$$

L est la longueur de la grande base, ℓ est la longueur de la petite base et h est la hauteur du trapèze.



Dans notre cas d'étude, on trouve alors que l'aire du trapèze vaut

$$\frac{f(\mathbf{g})+f(\mathbf{d})}{2}\times pas$$

C'est-à-dire que c'est la moyenne des aires des rectangles construits avec la méthode à gauche et à droite. Dans notre code précédent, il faut donc remplacer la ligne `somme_aire+=f(g)*pas` par `somme_aire+=(f(g)+f(d))/2*pas`.

4 Intégration de la fonction de densité de probabilité

Q11. Réécrivez une fonction `Integrale(dens,a,b,n,mu,sigma)`, permettant de calculer la valeur de l'intégrale de la fonction *dens* sur l'intervalle $[a,b]$ pour des valeurs de μ et σ données.

Pour cette question, il suffit de reprendre la fonction `integration_rectangle_gauche` et de remplacer `f(g)` par `dens(g,mu,sigma`.

```
1  def Integrale(dens,a:float,b:float,n:int,mu:float,sigma:float)->float:
2      '''Fonction permettant d'approximer le calcul de l'intégrale sur l'
           intervalle [a,b] d'une fonction dens continue et intégrable à l'aide de
           la méthode des rectangles à gauche.
3      Entrées : dens : la fonction étudiée, a : borne inférieure de l'intervalle d
           'étude, b : borne supérieure de l'intervalle d'étude, n : nombre de
           rectangles utilisés dans la méthode et mu et sigma les paramètres de la
           loi normale.
4      Sortie : Somme des aires des rectangles approximant la valeur de l'intégrale
           .'''
5      pas=(b-a)/n
6      somme_aire=0
7      for i in range(n):
8          g=a+pas*i
9          d=g+pas # ou encore d=a+pas*(i+1)
10         somme_aire+=dens(g,mu,sigma)*pas # dens(g,mu,sigma)*pas correspond à l'
               aire du rectangle i considéré
11     return somme_aire
```

Q12. Quelle ligne de commande faut-il alors écrire pour réaliser le calcul de $\mathcal{P}_r(95)$ avec 20 rectangles, pour $\mu = 100\,\Omega$ et $\sigma = 10\,\Omega$?

Il suffit d'utiliser la fonction `Integrale` créée précédemment et de l'appliquer à la fonction *dens* définie en tout début de sujet sur l'intervalle $[0,95]$.

```
1  Pr95=Integrale(dens,0,95,20,100,10)
```

Q13. Quelle est la valeur de $\int_0^{200} dens(R)\mathrm{d}R$? Justifier votre réponse.

Puisque par définition $\lim_{x\to+\infty} \mathcal{P}_r(x) = \int_{-\infty}^{+\infty} dens(R)\mathrm{d}R = 1$ et que $\int_{-\infty}^{0} dens(R)\mathrm{d}R = \int_{200}^{+\infty} dens(R)\mathrm{d}R = 0$, alors

$$\int_0^{200} dens(R)\mathrm{d}R = 1$$

Q14. Écrire un code permettant de déterminer à partir de combien de rectangles le résultat de l'intégrale de la question précédente est atteint avec une précision de 1×10^{-4}. Vous initialiserez votre recherche avec 3 rectangles. (Si vous ne connaissez pas le résultat à la question précédente, on le notera pour cette question comme étant égal à res).

Pour cette question, il suffit de calculer pour un nombre de rectangles croissant l'intégrale $\int_0^{200} dens(R)\mathrm{d}R$ et d'arrêter notre recherche du nombre de rectangles lorsque le résultat trouvé est compris entre 0.9999 et 1.0001. Cela suppose bien sûr que la précision devient meilleure avec un nombre de rectangles croissant.

```
1 n=3
2 integrale=Integrale(dens,0,200,n,100,10)
3 while integrale<0.9999 or integrale>1.0001:
4     n=n+1
5     integrale=Integrale(dens,0,200,n,100,10)
6 print(n)
```

Remarque : On trouve effectivement 15 rectangles pour atteindre la précision voulue. En traçant le résultat de l'intégrale pour plusieurs rectangles, on obtient :

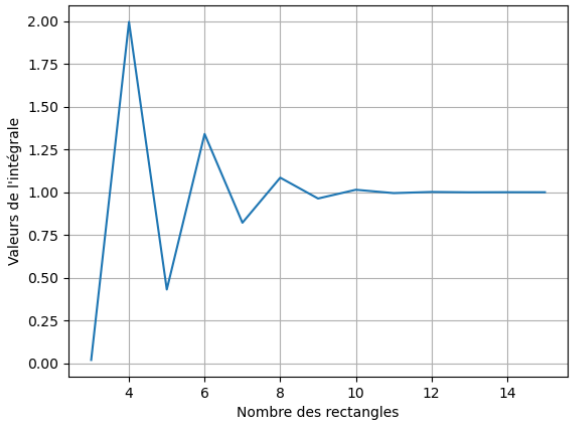


FIGURE 1 – Convergence de la méthode d'intégration pour le calcul de $\mathcal{P}_r(200)$

On constate des fluctuations de précision pour des nombres de rectangles faibles mais pas après avoir atteint la valeur de 15 rectangles. La valeur trouvée est donc cohérente.

Q15. On rappelle que $\mathcal{P}_r(95) \approx 0{,}31$. Expliquez pourquoi 15 rectangles ne sont pas suffisants pour obtenir la même précision qu'avec le calcul d'intégrale précédent.

Sur une partie de la fonction croissante, la méthode des rectangles à gauche a tendance à sous-estimer (en valeur absolue) la valeur de l'intégrale. Sur une partie décroissante de la fonction, c'est l'inverse.

Or, sur l'intervalle $[0, 200]$, la fonction est d'abord croissante puis décroissante à partir de 100. Ainsi, la sous-estimation faite par la méthode des rectangles à gauche sur l'intervalle $[0, 100]$ est compensée par la sur-estimation faite sur l'intervalle $[100, 200]$. La convergence de la méthode est donc relativement artificielle sur l'intervalle $[0, 200]$. Sur l'intervalle $[0, 95]$, la fonction *dens* est seulement croissante et nécessite donc un nombre plus important de rectangles pour arriver à convergence.

5 Recherche de l'écart-type maximal possible

Q16. Par rapport à la problématique posée, que représente en terme de probabilité $\mathcal{P}_r(103, \sigma) - \mathcal{P}_r(97, \sigma)$?

Ce calcul représente la probabilité qu'un résistor ait une valeur comprise entre 97 et 103 Ω.

Q17. Complétez sur votre copie l'expression mathématique de $\mathcal{P}_r(103, \sigma) - \mathcal{P}_r(97, \sigma)$ donnée ci-dessous. On précise que σ est ici un paramètre fixé.

$$\mathcal{P}_r(103, \sigma) - \mathcal{P}_r(97, \sigma) = \int_{\dots}^{\dots} dens(R, \sigma) \mathrm{d}R$$

$$\mathcal{P}_r(103, \sigma) - \mathcal{P}_r(97, \sigma) = \int_0^{103} dens(R, \sigma) \mathrm{d}R - \int_0^{97} dens(R, \sigma) \mathrm{d}R = \int_{97}^{103} dens(R, \sigma) \mathrm{d}R$$

Q18. Définissez une fonction `DPr(sigma:float)->float` permettant de calculer $\mathcal{DP}_r(\sigma)$ pour une valeur de σ donnée. On précise que les calculs d'intégrales nécessaires se feront avec 300 rectangles et que $\mu = 100\,\Omega$.

Pour répondre à cette question, on va utiliser la fonction `Integrale` appliquée à la fonction *dens* entre 97 et 103 Ω.

```
1 def DPr(sigma:float) -> float:
2     return Integrale(dens, 97, 103, 300, 100, sigma) # Intégration avec 300
          rectangles à gauche sur l'intervalle [97, 103]
```

Q19. Exprimez mathématiquement en fonction de $\mathcal{DP}_r$ le problème pour lequel on va chercher le zéro.

Puisque l'on cherche la valeur de σ permettant de valider la condition que 90% des résistors aient une valeur de résistance comprise entre 97 et 103 Ω, cela revient à chercher σ tel que $\mathcal{DP}_r(\sigma) = 0.9$. Au final, le problème dont on cherche le zéro est :

$$\mathcal{DP}_r(\sigma) - 0.9 = 0$$

Q20. Mettez en place la fonction `dichotomie(f_recherche_zero, borne_g, borne_d, epsilon)` permettant de trouver la valeur de zéro de la fonction `f_recherche_zero`. Le critère d'arrêt sera atteint lorsque l'intervalle de recherche sera plus petit que `epsilon`. Les bornes de recherche initiale de la dichotomie seront `borne_g` et `borne_d`.

Le problème est posé de façon tout à fait classique. Il suffit d'écrire le code de dichotomie comme vu en cours.

```
def dichotomie(f_recherche_zero, borne_g, borne_d, epsilon):
    while borne_d - borne_g > epsilon:
        borne_m = (borne_d + borne_g) / 2
        if f_recherche_zero(borne_g) * f_recherche_zero(borne_m) < 0:
            borne_d = borne_m
        else:
            borne_g = borne_m
    return (borne_d + borne_g) / 2
```

Q21. Pour `epsilon`=10^{-4}, combien d'itérations seront nécessaires pour converger avec `borne_g`=1 et `borne_d`=10 ? (On ne demande pas d'application numérique).

Après k itérations, l'intervalle d'étude est : $\frac{10-1}{2^k}$ Ainsi, on vérifie

$$\frac{10-1}{2^k} < 10^{-4} \text{ , pour } k > \left\lceil \frac{\ln\left(\frac{9}{10^{-4}}\right)}{\ln 2} \right\rceil = \left\lceil \log_2\left(\frac{9}{10^{-4}}\right) \right\rceil$$

6 Validation ou invalidation du stock

Q22. Codez une fonction `moyenne(L:list)->float`, permettant de calculer $\bar{L}$.

Fonction classique à maîtriser :

```
def moyenne(L):
    somme = 0
    for i in L:
        somme = somme + i
    return somme / len(L)
```

Q23. Créez maintenant une fonction `estimation_ecart_type(L:list)->float` permettant de calculer l'estimation de σ.

```
def estimation_ecart_type(L):
    somme = 0
    moy = moyenne(L)
    for i in L:
        somme += (moy - i) ** 2
    return m.sqrt(somme / (len(L) - 1))
```

Q24. Quelle est la complexité de votre fonction `estimation_ecart_type` ? Justifiez clairement votre réponse.

La fonction `estimation_ecart_type` est de complexité $\mathcal{O}(n)$. En effet, l'initialisation `somme = 0` est de complexité constante, le calcul de la moyenne est de coût $\mathcal{O}(n)$. La boucle `for` effectue des opérations à coût constant. Ainsi, cette boucle est de complexité $\mathcal{O}(n)$. L'action `return` est également à coût constant. En conclusion, la complexité de la fonction `estimation_ecart_type` est bien de $\mathcal{O}(n)$.

Q25. Concluez sur le fait que l'acheteur potentiel peut acheter ou non le stock de résistors.

Puisque l'estimation de σ est inférieure à la limite autorisée qui est de 1,82, l'acheteur peut acquérir le stock de résistors.

Q26. À partir de la liste `L`, créez le dictionnaire `dico` ayant pour clés : `'valable'` et `'non valable'`, et pour valeurs respectives le nombre de résistors ayant une résistance comprise entre 97 et 103 Ω et le nombre de résistors en dehors de cet intervalle.

```
dico = {'valable': 0, 'non valable': 0}
for i in L:
    if 97 <= i <= 103:
        dico['valable'] += 1
    else:
        dico['non valable'] += 1
```

Q27. À partir de `dico`, déterminez le pourcentage de résistors valables.

```
pourcentage = dico['valable'] / (dico['valable'] + dico['non valable']) * 100
```