Concours Agro-Véto 2023

Correction du sujet 'Modélisation mathématique et Informatique'

Partie 1 : Marche aléatoire additive

Sous-partie 1.1 : Déplacement libre

1. $X_3(\Omega) = \{-3h, -h, h, 3h\}$.
   On notera D_k l'événement : 'le k-ème saut se fait vers la droite' ;
   ainsi que G_k l'événement : 'le k-ème saut se fait vers la gauche'.
   On a alors : $[X_3 = 3h] = D_1 \cap D_2 \cap D_3$.
   Par indépendance des événements,

$$P(X_3 = 3h) = P(D_1) \times P(D_2) \times P(D_3) = \frac{1}{2^3}.$$

   De même $[X_3 = -3h] = G_1 \cap G_2 \cap G_3$ puis

$$P(X_3 = -3h) = \frac{1}{2^3}.$$

   Ensuite, $[X_3 = h] = \big(D_1 \cap D_2 \cap G_3\big) \cup \big(D_1 \cap G_2 \cap D_3\big) \cup \big(G_1 \cap D_2 \cap D_3\big)$.
   Les événements de cette réunion sont disjoints, donc :

$$P(X_3 = h) = \frac{1}{2^3} + \frac{1}{2^3} + \frac{1}{2^3}.$$

   De même $[X_3 = -h] = \big(G_1 \cap G_2 \cap D_3\big) \cup \big(G_1 \cap D_2 \cap G_3\big) \cup \big(D_1 \cap G_2 \cap G_3\big)$ puis :

$$P(X_3 = -h) = \frac{3}{2^3}.$$

   En conclusion :

$$P(X_3 = 3h) = P(X_3 = -3h) = \frac{1}{8} \quad ; \quad P(X_3 = h) = P(X_3 = -h) = \frac{3}{8}.$$

   Remarque 1 : on vérifie les calculs en notant que $\displaystyle\sum_{d \in X_3(\Omega)} P(X_3 = d) = 1$.

   Remarque 2 : cette question pouvait aussi être traitée en utilisant la variable Y_3 (définie en question 3) et les résultats sur la loi binomiale.

2. Puisque chaque saut a la même probabilité d'être effectué vers la droite ou vers la gauche, la variable aléatoire X_k est centrée :

$$E(X_k) = 0.$$

3. Chaque saut a une probabilité $\frac{1}{2}$ d'être effectué vers la droite, les k sauts sont indépendants, donc

$$Y_k \text{ suit la loi binomiale } \mathcal{B}(k, \tfrac{1}{2}).$$

4. Après k sauts, la particule a effectué Y_k sauts vers la droite et donc $(k - Y_k)$ sauts vers la gauche. Ainsi

$$X_k = Y_k \times h + (k - Y_k) \times (-h) = (2Y_k - k)h.$$

5. (a) On sait que $Y_k(\Omega) = \{0, 1, \ldots, k\}$ et que $X_k = (2Y_k - k)h$.
Donc $X_k(\Omega) = \{-kh, (-k+2)h, , \ldots, (k-2)h, kh\} = \{(2j-k)h, j \in \{0, \ldots, k\}\}$.
Pour $j \in \{0, \ldots, k\}$:

$$P(X_k = (2j-k)h) = P(Y_k = j) = \frac{\binom{k}{j}}{2^k}.$$

(b) On sait que $X_k = (2Y_k - k)h$ donc

$$V(X_k) = 2^2 h^2 V(Y_k) = 4 \times h^2 \times k \times \frac{1}{2} \times \frac{1}{2} = kh^2.$$

Sous-partie 1.2 : Déplacement contraint

6. Soit $k \in \mathbb{N}$. Soit $i \in \{1, \ldots, n\}$. On applique la formule des probabilités totales à l'événement $[X_{k+1} = i]$ avec le système complet d'événements $([X_k = j])_{1 \leq j \leq n}$:

$$P(X_{k+1} = i) = \sum_{j=1}^{n} P(X_k = j) \times P_{[X_k = j]}(X_{k+1} = i).$$

Les cas $i = 1$ et $i = n$ sont particuliers, on les traitera à part.
- Premier cas : $2 \leq i \leq n-1$.
D'après les règles de la marche aléatoire on a dans ce cas :

$$P(X_{k+1} = i) = \frac{1}{2} P(X_k = i-1) + \frac{1}{2} P(X_k = i+1).$$

- Second cas : $i = 1$. Alors

$$P(X_{k+1} = 1) = \frac{1}{2} P(X_k = 1) + \frac{1}{2} P(X_k = 2).$$

- Troisième cas : $i = n$. Alors

$$P(X_{k+1} = n) = \frac{1}{2} P(X_k = n-1) + \frac{1}{2} P(X_k = n).$$

Toutes ces relations s'écrivent matriciellement :

$$\vec{p}^{\,(k+1)} = M \vec{p}^{\,(k)}$$

où M est la matrice écrite dans l'énoncé en bas de la page 2.
On en déduit par une récurrence directe que :

$$\forall k \in \mathbb{N}, \quad \vec{p}^{\,(k)} = M^k \vec{p}^{\,(0)}.$$

7. La matrice M est symétrique et à coefficients réels donc M est diagonalisable dans $\mathcal{M}_n(\mathbb{R})$ et il existe une base orthonormée de $\mathcal{M}_{n,1}(\mathbb{R})$ formée par des vecteurs propres de M.
Matriciellement ceci signifie qu'il existe une matrice D diagonale et une matrice P inversible telles que :
$$M = PDP^{-1} \qquad \text{et} \qquad P^{-1} = P^{\mathsf{T}}.$$

8. Par opérations élementaires sur les lignes :
$$\begin{aligned}
\text{rang}(M - I_3) &= \text{rang} \begin{pmatrix} -1/2 & 1/2 & 0 \\ 1/2 & -1 & 1/2 \\ 0 & 1/2 & -1/2 \end{pmatrix} \\
&= \text{rang} \begin{pmatrix} -1/2 & 1/2 & 0 \\ 0 & -1/2 & 1/2 \\ 0 & 1/2 & -1/2 \end{pmatrix} \\
&= \text{rang} \begin{pmatrix} -1/2 & 1/2 & 0 \\ 0 & -1/2 & 1/2 \\ 0 & 0 & 0 \end{pmatrix} \\
&= 2.
\end{aligned}$$
Ainsi $1 \in Sp(M)$ et, d'après le théorème du rang, le sous-espace propre E_1 est de dimension $3 - 2 = 1$.
On peut remarquer que $M \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$
donc le vecteur $\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$ est un vecteur propre associé à la valeur propre 1.

9. (a) Pour $i \in [\![2, n]\!]$, les vecteurs $\vec{v_i}$ sont associés à des valeurs propres différentes de 1, puisqu'on a admis que $E_1 = Vect(\vec{v_1})$. De plus M est symétrique réelle donc ses sous-espaces propres sont deux à deux orthogonaux, ainsi $\vec{v_i}$ est orthogonal à $\vec{v_1}$:
$$\vec{v_1}^{\mathsf{T}} . \vec{v_i} = 0.$$
Attention aux notations de l'énoncé : ici le '.' désigne le produit matriciel. L'énoncé a décidé de noter de la même façon le produit scalaire, ce qui complique la compréhension de la question suivante notamment.
L'égalité qui vient d'être démontrée peut se réécrire :
$$\sum_{j=1}^{n} v_{i,j} = 0.$$

   (b) Ces notations sont bien compliquées, puisqu'en fait :
$$\lambda_1 = 1 \quad ; \quad \|\vec{v_1}\|_2 = 1 \quad ; \quad \text{donc} \quad N = M - \vec{v_1}.\vec{v_1}^{\mathsf{T}}$$
Ici le '.' désigne le produit matriciel... et ainsi :
$$N = M - \frac{1}{n} \begin{pmatrix} 1 & 1 & \dots & 1 \\ 1 & 1 & \dots & 1 \\ \vdots & \vdots & \dots & \vdots \\ 1 & 1 & \dots & 1 \end{pmatrix}.$$

En utilisant la question précédente :

$$N\vec{v_i} = M\vec{v_i} - \frac{1}{n}\begin{pmatrix} \sum_{j=1}^n v_{i,j} \\ \sum_{j=1}^n v_{i,j} \\ \vdots \\ \sum_{j=1}^n v_{i,j} \end{pmatrix} = M\vec{v_i} = \lambda_i \vec{v_i}.$$

(c) Soit $i \in [\![2, n]\!]$.
On sait que $N\vec{v_i} = \lambda_i\vec{v_i}$, donc l'inégalité (8) se réécrit : $\|\lambda_i\vec{v_i}\|_1 < \|\vec{v_i}\|_1$.
Donc : $|\lambda_i| \times \|\vec{v_i}\|_1 < \|\vec{v_i}\|_1$.
Puisque $\|\vec{v_i}\|_1 > 0$, on conclut que $|\lambda_i| < 1$.

10. (a) Pour $i \in [\![2, n]\!]$, $\lim\limits_{k\to+\infty} \lambda_i^k = 0$ puisque $|\lambda_i| < 1$. Ainsi

$$\lim_{k\to+\infty} D^k = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \vdots & & \ddots & \\ 0 & \dots & & 0 \end{pmatrix}.$$

Cette matrice limite sera appelée Δ.

(b) Par récurrence simple on prouve que $M^k = PD^kP^\mathsf{T}$.

(c) En passant à la limite (en admettant quelques résultats sur les limites de matrices un peu hors-programme de BCPST...) :

$$\lim_{k\to+\infty} M^k = P\Delta P^\mathsf{T}.$$

On sait que la première colonne de P (et donc la première ligne de P^T) est le vecteur $\vec{v_1}$. Ainsi

$$\Delta P^\mathsf{T} = \frac{1}{\sqrt{n}}\begin{pmatrix} 1 & 1 & \dots & 1 \\ 0 & 0 & \dots & 0 \\ \vdots & & \dots & \vdots \\ 0 & 0 & \dots & 0 \end{pmatrix}$$

puis :

$$\lim_{k\to+\infty} M^k = \frac{1}{n}\begin{pmatrix} 1 & * & \dots & * \\ 1 & * & \dots & * \\ \vdots & & \dots & \vdots \\ 1 & * & \dots & * \end{pmatrix}\begin{pmatrix} 1 & 1 & \dots & 1 \\ 0 & 0 & \dots & 0 \\ \vdots & & \dots & \vdots \\ 0 & 0 & \dots & 0 \end{pmatrix} = \frac{1}{n}\begin{pmatrix} 1 & 1 & \dots & 1 \\ 1 & 1 & \dots & \\ \vdots & & \dots & \vdots \\ 1 & & \dots & 1 \end{pmatrix}.$$

On note L cette limite.

11.

$$\begin{aligned}\lim_{k\to+\infty}\vec{p}^{\,(k)} &= \lim_{k\to+\infty} M^k\vec{p}^{\,(0)} = L\vec{p}^{\,(0)}\\ &= \frac{1}{n}\begin{pmatrix}\sum_{j=1}^n \vec{p}_j^{\,(0)}\\ \sum_{j=1}^n \vec{p}_j^{\,(0)}\\ \vdots\\ \sum_{j=1}^n \vec{p}_j^{\,(0)}\end{pmatrix}\\ &= \frac{1}{n}\begin{pmatrix}1\\1\\\vdots\\1\end{pmatrix}.\end{aligned}$$

Autrement dit, pour tout i fixé de $[\![1,n]\!]$: $\lim\limits_{k\to+\infty} P(X_k = i) = \dfrac{1}{n}$.

On remarque que cette limite ne dépend pas du point de départ de la marche aléatoire.

12. *Interprétation :*

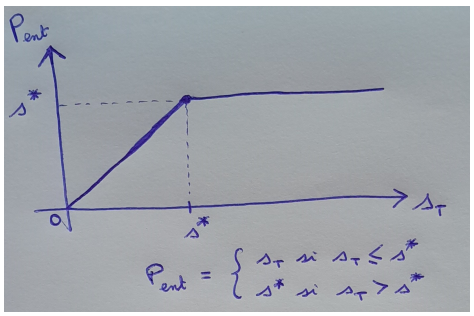
On peut finalement conclure que la suite de variables aléatoires (X_k) converge en probabilité vers une variable de loi uniforme $\mathcal{U}([\![1,n]\!])$.

Physiquement, lorsqu'on libère des molécules d'un gaz dans une boîte, alors sous certaines conditions ces molécules se répartiront à long terme de façon uniforme dans la boîte.

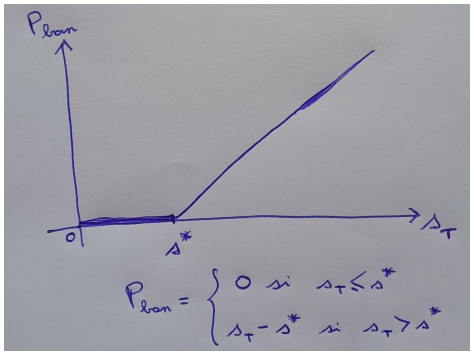
Partie 2 : Marche aléatoire multiplicative

Sous-partie 2.1 : Contrat à terme

13. Plus la durée du contrat est longue, plus le prix du blé a le temps d'évoluer, ce qui augmente sa volatilité et donc le risque pour la banque. Ainsi la banque aura intérêt à faire payer plus cher c_T lorsque T augmente.
    Donc le prix du call c_T dépend de T, la durée du contrat.
14. Voici la courbe demandée (on ne prend pas en compte le prix du call) :



15. Voici la courbe de P_{ban} en fonction de s_T :



Sous-partie 2.2 : Le modèle de Cox-Ross-Rubinstein

Remarque : la remise à 0 du compteur de questions est étrange en plein milieu d'une partie.

1. (a) On peut, comme le suggère l'énoncé, représenter la situation par un arbre probabiliste.
    On peut aussi refaire un raisonnement analogue à celui fait pour étudier la variable X_k en début de partie 1, ce qui permet de répondre rapidement aux questions (a), (b) et (c).
    Considérons Z_n, une variable aléatoire suivant la même marche aléatoire que X_n dans la partie 1, sauf que le pas est ± 1 et que la probabilité de sauter vers la droite est p (dans ce cas $Z_{n+1} = Z_n + 1$), alors que la probabilité d'aller vers la gauche est $1 - p$ (dans ce cas $Z_{n+1} = Z_n - 1$).
    On a alors $S_n = u^{Z_n} S_0$.
    Puisque $Z_3(\Omega) = \{-3, -1, 1, 3\}$ (déjà expliqué en sous-partie 1.1),

$$\mathcal{D}_3 = S_3(\Omega) = \{u^3 s_0, u s_0, u^{-1} s_0, u^{-3} s_0\}.$$

Si on note U_n le nombre de mouvements 'up' lors des n premiers instants, la variable Z_n dépend de U_n.

Plus précisément : $Z_n = U_n - (n - U_n) = 2U_n - n$.

Puisque U_n suit la loi binomiale $\mathcal{B}(n,p)$, il semble cohérent de parler de 'modèle recombinant'.

(b) $P(S_3 = u^3 s_0) = P(U_3 = 3) = p^3$.

$P(S_3 = us_0) = P(U_3 = 1) = \binom{3}{1} p^1 (1-p)^{3-1} = 3p(1-p)^2$.

(c) $U_n(\Omega) = [\![0, n]\!]$ et $S_n = u^{2U_n - n} s_0$, donc :

$$\mathcal{D}_n = S_n(\Omega) = \{u^{2k-n} s_0, k \in [\![0, n]\!]\}.$$

2. (a) Avant de commencer, rappelons que le prix du blé à la date n est $S_n = u^{2k-n} s_0$, avec k désignant la valeur de U_n, c'est à dire le nombre de 'up' lors des k premiers instants.

Pour l'exécution ou non du call, il est important de savoir si ce prix du blé dépasse s^*. Or on a les équivalences suivantes :

$$S_n > s^* \Leftrightarrow u^{2k-n} > \frac{s^*}{s_0} \Leftrightarrow 2k > n + \frac{\ln\left(\frac{s^*}{s_0}\right)}{\ln(u)} \Leftrightarrow k \geq 1 + \frac{n}{2} + \frac{\ln\left(\frac{s^*}{s_0}\right)}{2\ln(u)}.$$

On vient ainsi de comprendre la définition de a : a est le plus petit nombre de 'up' possibles pour que le prix du blé dépasse la valeur s^*.

On peut maintenant démarrer les questions.

A la date n, deux cas sont possibles :

- Cas 1 : le prix du blé S_n est inférieur ou égal à s^* :

dans ce cas la banque ne verse rien à la meunerie, et ainsi $G_n = 0$.

- Cas 2 : $S_n > s^*$:

dans ce cas $S_n = u^{2k-n} s_0$ avec k, le nombre de 'up', qui est supérieur ou égal à a, donc k est compris entre a et n. La banque prend alors à sa charge la différence entre le cours du blé et le prix d'exercice s^*, autrement dit la banque verse à l'entreprise la somme $G_n = S_n - s^* = u^{2k-n} s_0 - s^*$.

En conclusion :

$$\mathcal{D}_{G_n} = \{0\} \cup \{u^{2k-n} s_0 - s^*, k \in \{a, \ldots, n\}\}.$$

(b) On l'a plus ou moins déjà dit : a est le plus petit nombre de 'up' qui déclenche une dépense pour la banque.

(c) Ici $n = 3$, $u = 2$, $s_0 = 1$; $s^* = 4$ et $p = \frac{1}{3}$.

La loi de S_3 est donnée par :

$$P(S_3 = 8) = p^3 = \frac{1}{27}.$$

$$P(S_3 = 2) = 3p^2(1-p) = \frac{2}{9}.$$

$$P(S_3 = \frac{1}{2}) = 3p(1-p)^2 = \frac{4}{9}.$$

$$P(S_3 = \frac{1}{8}) = (1-p)^3 = \frac{8}{27}.$$

On peut calculer a :

$$1 + \frac{n}{2} + \frac{\ln\left(\frac{s^*}{s_0}\right)}{2\ln(u)} = 1 + \frac{3}{2} + \frac{\ln(4)}{2\ln(2)} = \frac{7}{2}$$

donc $a = 3$.

Ainsi $\mathcal{D}_{G_3} = \{0, 2^3 - 4\} = \{0, 4\}$.

On aurait aussi pu déterminer cet ensemble sans calculer a. En effet, le prix d'exercice s^* vaut 4 et n'est dépassé que si $S_3 = 8$. Voici enfin la loi de G_3 :

$$P(G_3 = 4) = P(S_3 = 8) = \frac{1}{27}.$$

$$P(G_3 = 0) = 1 - P(G_3 = 4) = \frac{26}{27}.$$

3. (a) La variable aléatoire G_n désigne la somme que la banque versera à l'entreprise à la date n. Ainsi l'espérance de G_n désigne le coût moyen, pour la banque, de cette opération financière.

   Le prix juste du call de terme n doit donc être égal à cette espérance $E(G_n)$.

   (b) En $(2c)$: $E(G_3) = 4P(G_3 = 4) = \frac{4}{27}$.

   Ici le prix juste du call est : $c_3 = \frac{4}{27}$.

   (c) Soit $n \geq 1$. D'après (13) :

$$E(G_n) = \sum_{k=a}^{n} \left(u^{2k-n} s_0 - s^*\right) P(U_n = k)$$

où la variable U_n a été définie en $(2a)$. Puisque U_n suit la loi binomiale $\mathcal{B}(n, p)$:

$$E(G_n) = \sum_{k=a}^{n} \left(u^{2k-n} s_0 - s^*\right) \times \binom{n}{k} p^k (1-p)^{n-k}.$$

Sous-partie 2.3 : Calcul effectif du prix du call par Python

4. (a) *Remarque : on pourrait dire qu'il y a ici une erreur d'énoncé, puisque si $k > n$ alors $C(n, k)$ ne renvoie pas 0, ce qui aurait dû être le cas. Par exemple $C(1, 2) = 1/2$...*

   L'énoncé devrait préciser qu'ici $n \geq k$.

   Les lignes $2, 3, 4$ permettent de calculer $Num = n!$.

   Les lignes $5, 6, 7$ calculent $Den = k!$.

   Puis les lignes 8 et 9 calculent $Den = k!(n-k)!$.

   Au final la fonction retourne bien $\frac{n!}{k!(n-k)!} = \binom{n}{k}$.

La première boucle nécessite n opérations. Les deux boucles suivantes nécessitent, à elles deux, n opérations.

Donc la fonction C effectue de l'ordre de $2n$ opérations.

Le problème principal est que les calculs de factorielles feront rapidement intervenir des nombres trop grands pour être gérés correctement par l'ordinateur.

(b) On peut remarquer que $\frac{n!}{k!} = (k+1)(k+2)\dots(n-1)(n)$, donc on écrit le code suivant :

```
def Copt(n,k):
    Num=1
    for i in range(k+1,n+1):
        Num=Num*i
    Den=1
    for i in range(2,n-k+1):
        Den=Den*i
    return(Num/Den)
```

5. La fonction `Feuilles` retourne $\mathcal{D}_{G_n}$, que nous avions défini dans la formule (13).

   Autrement dit la fonction renvoie la liste des sommes que la banque peut verser à la meunerie (à ceci près que la valeur 0 sera répétée plusieurs fois dans la liste).

   Par exemple, avec les valeurs de l'application numérique de la question $(2-c)$ on obtient :

```
>>> Feuilles(n,u,s0,se)
        [0, 0, 0, 4]
```

6. Dans la fonction suivante, on calcule le prix c_n du call, noté C dans la fonction, en appelant tout d'abord la fonction `Feuilles` pour créer DG, la liste des valeurs prises par G_n, puis en calculant la somme de la formule (15).

```
def call(s0,setoile,n,u,p):
    C=0
    DG=Feuilles(n,u,s0,setoile)
    q=len(DG)
    for k in range(q):
        C=C+DG[k]*Copt(n,k)*p**k*(1-p)**(n-k)
    return(C)
```

Partie 3 : Estimation des paramètres du modèle de Cox-Ross-Rubinstein

Ici $p = \dfrac{1}{2}$.

Sous-partie 3.1 : Calcul de la volatilité historique et calibrage du modèle

7. (a) D'après la sous-partie 2.2, la variable Z_n vaut $\ln(u)$ avec la probabilité p qui vaut ici $\frac{1}{2}$, et $ln(\dfrac{1}{u}) = -\ln(u)$ avec la probabilité $1-p = \frac{1}{2}$.
Ces probabilités sont indépendantes des réalisations des autres variables Z_i.

   (b) $E(Z_1) = 0$.
$V(Z_1) = E(Z_1^2) = \big(\ln(u)\big)^2$.

8. (a) Par linéarité de l'espérance :

$$E(\overline{Z_n}) = \frac{1}{n}\sum_{i=1}^{n} E(Z_i) = 0.$$

Les variables (Z_i) sont indépendantes donc :

$$V(\overline{Z_n}) = \frac{1}{n^2}\sum_{i=1}^{n} V(Z_i) = \frac{\big(\ln(u)\big)^2}{n}.$$

Posons maintenant (on rappelle que $u > 1$) :

$$\sigma = \big|\ln(u)\big| = \ln(u) = \sigma(Z_1).$$

On peut appliquer le théorème central limite. En effet, la suite $(Z_n)_{n\geq 1}$ est une suite de variables aléatoires indépendantes de même loi, ayant une espérance $m = 0$ et une variance σ^2 non nulle.
Ainsi, d'après le théorème central limite, la suite $\Big(M_n^* = \dfrac{Z_n - 0}{\frac{\sigma}{\sqrt{n}}}\Big)_{n\geq 1}$ converge en loi vers une variable aléatoire suivant la loi normale centrée réduite.

   (b) Une volatilité élevée signifie que u est elevée, ce qui signifie que les prix sont très variables d'un instant à l'autre.

9. (a) $V(\overline{Z_n}) = \dfrac{\sigma^2}{n}$. En utilisant la linéarité de l'espérance :

$$\begin{aligned} E(V_n) &= \frac{1}{n-1}\sum_{i=1}^{n} E\big((Z_i - \overline{Z_n})^2\big) \\ &= \frac{1}{n-1}\sum_{i=1}^{n} E\big(Z_i^2 - 2Z_i\overline{Z_n} + \overline{Z_n}^2\big) \\ &= \frac{1}{n-1}\sum_{i=1}^{n} \big(E(Z_i^2) - 2E(Z_i\overline{Z_n}) + E(\overline{Z_n}^2)\big). \end{aligned}$$

D'une part :

$$E(Z_i^2) = \sigma^2.$$

D'autre part :

$$E(\overline{Z_n}^2) = \frac{\sigma^2}{n}.$$

Enfin :

$$\begin{aligned} E(Z_i\overline{Z_n}) &= \frac{1}{n}\big(E(Z_i^2) + \sum_{j\neq i} E(Z_i)E(Z_j)\big) \\ &\qquad \text{par indépendance de } Z_i \text{ et } Z_j \text{ lorsque } i \neq j \\ &= \frac{\sigma^2}{n}. \end{aligned}$$

On réinjecte tout ceci dans le calcul :

$$E(V_n) = \frac{1}{n-1}\sum_{i=1}^{n}\big(\sigma^2 - 2\frac{\sigma^2}{n} + \frac{\sigma^2}{n}\big) = \frac{1}{n-1} \times n \times \big(\sigma^2 - \frac{\sigma^2}{n}\big) = \sigma^2.$$

(b) V_n représente l'estimateur (sans biais) de la variance de Z_i, donc ici V_n est un estimateur de σ^2.

(c) On a vu que $\sigma = \ln(u)$, donc $u = \exp(\sigma)$. Or $\sqrt{V_n}$ est un estimateur de σ, donc la variable $U_n = \exp(\sqrt{V_n})$ est un estimateur de u.

Sous-partie 3.2 : Calcul de la volatilité implicite avec python

10. (a) Pour la méthode de Newton on peut écrire ceci :

```
def Newton(u0,epsilon,F,Fprime):
    u=u0
    for i in range(nmax):
        v=u-F(u)/Fprime(u)
        if abs(v-u)<epsilon:
            return(v)
        u=v
    return(v)
```

D'autres solutions sont envisageables, notamment en remplaçant le `for` par un `while`.

(b) Il faut prendre ε très petit et strictement positif, par exemple $\varepsilon = 10^{-4}$.

(c) Si $f'(u_n) = 0$ alors la fonction renvoie un message d'erreur. La question est donc étrange, mais nous allons y répondre quand même en proposant le code suivant :

```
def Newton(u0,epsilon,F,Fprime):
    u=u0
    for i in range(nmax):
        if Fprime(u)==0:
              return('Erreur, division par 0')
        else :
              v=u-F(u)/Fprime(u)
              if abs(v-u)<epsilon:
                    return(v)
        u=v
    return(v)
```

11. Dans la somme écrite dans la formule (20), tous les termes de la somme sont par définition strictement positifs. Ainsi $c_n(u)$ est nul si et seulement si la somme est vide, si et seulement si $a \geq n+1$. Résolvons cette inéquation :

$$1 + \frac{n}{2} + \frac{\ln(\frac{s^*}{s_0})}{2\ln(u)} \geq n+1 \quad \Leftrightarrow \quad \ln(\frac{s^*}{s_0}) \geq n\ln(u)$$
$$\Leftrightarrow \quad \exp\big(\frac{\ln(\frac{s^*}{s_0})}{n}\big) \geq u.$$

Ainsi $u_{min} = \exp\big(\frac{\ln(\frac{s^*}{s_0})}{n}\big) = (\frac{s^*}{s_0})^{\frac{1}{n}}$.

12. Si $u < u_{min}$ la valeur du call est nulle puisque dans ce cas les variations de prix sont trop petites et il n'y a aucun risque que le prix du blé dépasse le prix d'exercice s^* avant la date n.

13. La réponse dépend de la parité de n. On suppose ici n pair, la réponse pour n impair nécessite quelques petits aménagements que nous ne détaillerons pas.

Le nombre $1 + \frac{n}{2}$ est un nombre entier, donc a est constant lorsque la quantité $\dfrac{\ln(\frac{s^*}{s_0})}{2\ln(u)}$ est comprise dans un intervalle de la forme $[k, k+1[$, avec k un entier de $[\![0, \frac{n}{2}-1]\!]$. Résolvons donc les inéquations :

$$k \leq \frac{\ln(\frac{s^*}{s_0})}{2\ln(u)} < k+1 \quad \Leftrightarrow \quad 2k\ln(u) \leq \ln(\frac{s^*}{s_0}) < 2(k+1)\ln(u)$$
$$\Leftrightarrow \quad \exp\big(\frac{\ln(\frac{s^*}{s_0})}{2(k+1)}\big) < u \leq \exp\big(\frac{\ln(\frac{s^*}{s_0})}{2k}\big).$$

Ainsi a est constant lorsque u appartient à un intervalle de la forme $I_k =]\big(\frac{s^*}{s_0}\big)^{\frac{1}{2k+1}}, (\frac{s^*}{s_0})^{\frac{1}{2k}}]$.

Lorsque $u \in I_k$, a est constant donc la fonction c_n, et par conséquent la fonction f, est dérivable. D'où l'intérêt de s'intéresser à ces intervalles I_k, qui sont des intervalles où la fonction f est dérivable, ce qui permettra d'y appliquer la méthode de Newton.

Nous pouvons maintenant donner une démarche pour rechercher un zéro de f, en évitant d'utiliser la même idée que celle qui sera donnée par l'énoncé dans deux questions.

Notre démarche pour rechercher un zéro de f serait de parcourir, à l'aide d'une boucle `for`, tous les indices k possibles dans $[\![0, \frac{n}{2}-1]\!]$, puis d'exécuter la fonction

`Newton(u0,epsilon,F,Fprime)` avec u_0 qui est le milieu de l'intervalle I_k. Les différentes valeurs candidates pour être un zéro de f pourraient ensuite être stockées dans une liste L et on déterminerait, par un algorithme de minimisation, laquelle de ces valeurs a une image par f la plus proche de 0.

14. Sur un de ces intervalles I_k, f est dérivable et, en rappelant qu'ici $p = \frac{1}{2}$:

$$f'(u) = \frac{s_0}{2^n} \sum_{k=a}^{n} (2k-n) u^{2k-n-1} \binom{n}{k}.$$

15. (a) La boucle `while` permet de déterminer, en partant de $a = n$ et en diminuant a d'une unité à chaque étape, l'intervalle I_k sur lequel la fonction F change de signe. La fonction exécute ensuite l'algorithme de Newton sur cet intervalle pour retourner la valeur de u qui correspondra à une valeur approchée d'un zéro de f.

    (b)
```
def Volatilite(F,Fprime,s0,setoile,epsilon):
    u=mystere(F,Fprime,s0,setoile,epsilon)
    return(m.log(u))
```