**BE n°2 – Probabilités/Statistique avec MATLAB**
Variables continues

# 1 Génération de variables continues

## 1.1 Loi normale multi-dimensionnelle

Soit $X = (X_1, X_2, \ldots, X_n)^T$ une variable multi-dimensionnelle de loi $\mathcal{N}(\mathbf{0}, \mathbf{I})$. On cherche une matrice $\mathbf{A}$ et un vecteur $\mathbf{b}$ tels que le vecteur $Y = (Y_1, Y_2, \ldots, Y_n)^T = \mathbf{A}X + \mathbf{b}$ suive une loi $\mathcal{N}(\mathbf{m}, \mathbf{\Sigma})$, avec $\mathbf{m} = (m_1, m_2, \ldots, m_n)^T$ et $\mathbf{\Sigma}$ une matrice symétrique définie positive. On montre que $\mathbf{A}$ et $\mathbf{b}$ vérifient les relations suivantes :

$$\begin{aligned} \mathbf{b} &= \mathbf{m} \\ \mathbf{A}\mathbf{A}^T &= \mathbf{\Sigma} \end{aligned}$$

1. A l'aide de la fonction `randn`, créer une fonction `normmult.m` qui renvoie une suite de vecteurs $\left(Y^i\right)_{1 \leq i \leq N}$, chacun distribué suivant la loi $\mathcal{N}(\mathbf{m}, \mathbf{\Sigma})$ (utiliser la fonction `chol` qui permet d'obtenir $\mathbf{A}$ à partir de $\mathbf{\Sigma}$). Tester cette fonction avec $\mathbf{m} = \begin{pmatrix} 1 \\ 0.5 \end{pmatrix}$ et $\mathbf{\Sigma} = \begin{pmatrix} 2 & 0.5 \\ 0.5 & 1 \end{pmatrix}$
2. Vérifier sur l'histogramme la normalité des variables $Y_1$ et $Y_2$.
3. Estimer la moyenne et la matrice de covariance de $Y$ à l'aide des commandes `mean` et `cov` et comparer avec les résultats théoriques.

## 1.2 Loi exponentielle

On rappelle que si $X$ est une variable aléatoire continue de fonction de répartition $F(x)$ strictement croissante, alors la variable aléatoire $Y = F(X)$ est uniformément distribuée sur $[0, 1]$. Par conséquent, pour générer la variable aléatoire $X$, il suffit de générer la variable uniforme $Y$, et de calculer $X = F^{-1}(Y)$ (dans le cas où $F^{-1}$ a une forme simple).

1. Appliquer cette propriété pour générer une variable de loi exponentielle de paramètre $\lambda > 0$.
2. Vérifier le résultat en calculant moyenne et variance, et en visualisant l'histogramme.
   *Rappel* : la densité de probabilité d'une variable exponentielle $X$ est donnée par :

$$f(x) = \begin{cases} \lambda \exp(-\lambda x) & \text{si } x \geq 0 \\ 0 & \text{sinon} \end{cases}$$

   On a alors $E[X] = \frac{1}{\lambda}$ et $var(X) = \frac{1}{\lambda^2}$.

# 2 Probabilité d'erreur de bits

On émet des symboles binaires $x$ tels que :

$$\begin{aligned} \text{Classe } C_0 : &\quad x = 0 \\ \text{Classe } C_1 : &\quad x = 1 \end{aligned}$$

Lorsque ce signal est émis sur le canal de transmission, il est perturbé par un bruit $b$, de telle sorte que le signal reçu $z$ s'écrit :

$$z = x + b$$

Le problème consiste alors à retouver le symbole $x$ émis à partir de la donnée observée $z$. On suppose ici que le bruit $b$ est une variable aléatoire gaussienne centrée et de variance $\sigma^2$, ce qui correspond au modèle le plus couramment

utilisé. On suppose de plus que le symbole $x$ prend les valeurs 0 et 1 avec les mêmes probabilités. A la réception du symbole, on décide que le signal émis $x$ vaut 0 si $z < 1/2$, et qu'il vaut 1 si $z > 1/2$. La probabilité d'erreur de cette décision s'écrit alors :

$$P_e = P\left[z < 1/2 \text{ et } x = 1\right] + P\left[z > 1/2 \text{ et } x = 0\right]$$

On veut estimer cette probabilité par *fréquence relative*, et la comparer à la valeur théorique :

$$P_e = \frac{1}{2}\left(1 + \Phi\left(\frac{-1}{2\sigma}\right) - \Phi\left(\frac{1}{2\sigma}\right)\right) = \Phi\left(\frac{-1}{2\sigma}\right)$$

où $\Phi(x)$ est la fonction de répartition de la loi normale $\mathcal{N}(0,1)$.

1. Générer les vecteurs $z$ et $x$ sur $N = 1000$ points, avec $\sigma^2 = 0.5$.
2. Estimer $P\left[z < 1/2 \text{ et } x = 1\right]$ par fréquence relative à partir de $x$ et de $z$, en sélectionnant les indices $k$ tels que $x(k) = 1$ et $z(k) < 1/2$ (on pourra utiliser au choix les fonctions "`find`", "`length`", "`mean`", ou tout autre fonction si besoin est, ainsi que les opérateurs logiques).
3. De la même façon, estimer $P\left[z > 1/2 \text{ et } x = 0\right]$, puis écrire une fonction `prob_erreur_est` qui renvoie une estimation de $P_e$ en fonction de $N$ et $\sigma^2$.
4. Estimer ainsi $P_e$ pour $\sigma^2$ variant de 0.1 à 10 avec un pas de 0.1 (on pourra faire une boucle utilisant la fonction "`prob_erreur_est`", à moins bien sûr de maîtriser la commande "`kron`"). Tracer le résultat ainsi obtenu en fonction de $\sigma^2$ (commande "`plot(x,y)`").
5. Créer une fonction "`p=prob_erreur_th(sigma2)`" qui renvoie la valeur théorique de $P_e$ pour les mêmes valeurs de $\sigma^2$ (utiliser la commande "`normcdf`"), et superposer les deux courbes.

# 3 Ruptures et recuit simulé

Dans beaucoup d'applications pratiques (en images, télécoms, radar, astronomie,...), on peut être amené à chercher des ruptures dans un signal, ces ruptures pouvant concerner un ou plusieurs paramètres caractérisant ce signal. Une des situations les plus classiques est la détection de ruptures d'amplitudes. Plus précisément, le signal observé s'écrit

$$y(n) = x(n) + b(n)\ , \quad n = 1, \ldots, N \tag{1}$$

où les variables $b(n)$ sont indépendantes, de loi $\mathcal{N}\left(0, \sigma^2\right)$, et le signal $(x(n))_{n=1,\ldots,N}$ est constants par morceaux. Dans cet exercice, on se contentera d'un signal $x(n)$ à 3 niveaux, c'est-à-dire que le signal $x(n)$ peut s'écrire sous la forme :

$$\begin{aligned} x(n) &= A_1 \quad \text{pour } n \in \{1, 2, \ldots, t_1\} \\ x(n) &= A_2 \quad \text{pour } n \in \{t_1 + 1, \ldots, t_2\} \\ x(n) &= A_3 \quad \text{pour } n \in \{t_2 + 1, \ldots, N\} \end{aligned}$$

avec $0 = t_0 < t_1 < t_2 < t_3 = N$. Le problème est alors d'estimer, à l'aide des observations $(y(n))_{n=1,\ldots,N}$, les instants de rupture $\mathbf{t} = (t_1, t_2)$, ainsi que les amplitudes $\mathbf{A} = (A_1, A_2, A_3)$. On peut montrer que l'estimateur du maximum de vraisemblance de $\widehat{\mathbf{t}}_{MV}$ est le vecteur $\mathbf{t} = (t_1, t_2)$ qui minimise la fonction

$$f(\mathbf{t}) = \sum_{k=1}^{3} \sum_{n=t_{k-1}+1}^{t_k} \left(y(n) - \widehat{A}_k(\mathbf{t})\right)^2 \tag{2}$$

où

$$\widehat{A}_k(\mathbf{t}) = \frac{1}{t_k - t_{k-1}} \sum_{n=t_{k-1}+1}^{t_k} y(n)\ . \tag{3}$$

L'estimateur du maximum de vraisemblance de $A$ est alors

$$\widehat{A}_{MV} = \widehat{A}(\widehat{\mathbf{t}}_{MV})$$

Le problème est donc de minimiser la fonction $f$ par rapport à $\mathbf{t}$, lorsque $\mathbf{t}$ parcourt l'ensemble $\mathcal{D} = \{(t_1, t_2) \in \{1, \ldots, N-1\}^2 \,|\, t_1 < t_2\}$. Une méthode exacte est de calculer toutes les valeurs prises par $f$ sur l'ensemble $\mathcal{D}$. On aurait ainsi $\frac{N(N-1)}{2}$ valeurs à calculer, et le couple $(t_1, t_2)$ optimal serait celui qui donne la valeur de $f$ minimale. On voit donc que cette méthode est coûteuse en nombre de calculs si $N$ devient (très) grand (et ceci serait d'autant plus vrai si on cherchait à déterminer plus de deux ruptures).

On va ici utiliser une technique moins coûteuse en calculs, appelée *algorithme du recuit simulé*, mais qui présente l'inconvénient de ne pas toujours donner la solution exacte. Pour le présent problème, cet algorithme, dans une version simplifiée, peut être décrit de la façon suivante (en notant $\mathbf{t}_{opt}$ le vecteur $\mathbf{t}$ trouvé par l'algorithme, et $f_{opt}$ la valeur de $f$ correspondante) :

INITIALISATION : génération de $\mathbf{t}^0 = \left(t_1^0, t_2^0\right)$ tiré uniformément sur $\mathcal{D}$.
on pose : $f_0 \leftarrow f(\mathbf{t}^0)$, $f_{opt} \leftarrow f_0$, $\mathbf{t}_{opt} \leftarrow \mathbf{t}^0$.
on se fixe un paramètre $\alpha$ proche mais inférieur à 1.
on initalise une suite $(T_n)_{n\in\mathbb{N}}$, c'est-à-dire qu'on fixe $T_0$.

ITERATION $n \geq 1$ : génération de $\mathbf{t} = (t_1, t_2)$ tiré uniformément sur $\mathcal{D}$.
on pose : $\Delta f = f(\mathbf{t}) - f(\mathbf{t}^{n-1})$
si $\Delta f < 0$ alors :
$\mathbf{t}^n \leftarrow \mathbf{t}$, $f_n \leftarrow f(\mathbf{t}^n)$
si $f(\mathbf{t}) < f_{opt}$ alors
$f_{opt} \leftarrow f(\mathbf{t})$, $\mathbf{t}_{opt} \leftarrow \mathbf{t}$
sinon : génération de $U$ tiré uniformément sur $[0;1]$.
si $U < \exp\left(-\frac{\Delta f}{T_n}\right)$ alors
$\mathbf{t}^n \leftarrow \mathbf{t}$, $f_n \leftarrow f(\mathbf{t}^n)$
sinon
$\mathbf{t}^n \leftarrow \mathbf{t}^{n-1}$, $f_n \leftarrow f_{n-1}$

$T_n \leftarrow \alpha T_{n-1}$

Il existe plusieurs possibilités pour arrêter l'algorithme. Ici, on se contentera de s'arrêter après un nombre $N_{iter}$ d'itérations fixé à l'avance.
Créer un fichier que l'on appellera `exo2.m`. Répondre aux questions suivantes en complétant ce fichier au fur et à mesure.

1. Pour des valeurs données du vecteur `t` (qui contient $t_1$ et $t_2$), du vecteur `A` (qui contient $A_1$, $A_2$, $A_3$), et de la variance du bruit `sigma2`, générer le signal $(y(n))_{n=1,\dots,N}$ donné dans (1).
2. Créer une fonction `[f,A_est]=critere(y,t)`, qui renvoie la valeur de la fonction $f$ donnée par (2) en fonction du signal observé `y` et du vecteur d'instants `t`, ainsi que le vecteur `A_est` doné par (3).
3. Compléter alors le fichier `exo2.m` en écrivant l'algorithme donné ci-dessus. (**Note** : pour générer un couple `t` uniformément sur $\mathcal{D}$, on utilisera successivement les commandes : `t=randperm(N-1) ; t= [0 sort(t(1 :2)) N] ;)`
4. Tester le programme avec les valeurs suivantes : $N = 100$, $RSB = 10$, $(t_1, t_2) = (33, 66)$, $(A_1, A_2, A_3) = (10, 5, 8)$, $N_{iter} = 1000$, $T_0 = 10^5$, $\alpha = 0.95$.
5. Superposer alors sur une figure le signal $(y(n))_{n=1,\dots,N}$ et le signal reconstruit $(\widehat{x}(n))_{n=1,\dots,N}$ défini par

$$\begin{aligned}
\widehat{x}(n) &= A_{opt}(1) \quad \text{pour } n \in \{1, 2, \dots, t_{opt}(1)\} \\
\widehat{x}(n) &= A_{opt}(2) \quad \text{pour } n \in \{t_{opt}(1)+1, \dots, t_{opt}(2)\} \\
\widehat{x}(n) &= A_{opt}(3) \quad \text{pour } n \in \{t_{opt}(2)+1, \dots, N\}
\end{aligned}$$

6. Refaire d'autres simulations en changeant, comme vous le souhaitez, les valeurs de $(t_1, t_2)$ et de $(A_1, A_2, A_3)$. Que peut-on en conclure ?