TP 1 - Intro. à Scilab, Illustration de convergence p.s. - Corrigé succinct

Exercice 1. **1.** Voici une fonction (enregistrée dans un fichier `TP1Echauffement.sci`) qui prend en entrée un vecteur $vect\_x$, un réel m et un réel strictement positif s et qui retourne un vecteur de même taille que $vect\_x$ et contenant les valeurs de la densité de la loi normale $\mathcal{N}(m, s^2)$ aux points d'abscisses $vect\_x$. Si le paramètre s est négatif, la fonction retourne la chaîne de caractère "erreur !".

```
function f_vect_x=densite_gaussienne(vect_x,m,s)
    if (s<=0) then
        disp('erreur!')
    else f_vect_x=exp(-(vect_x-m).^2/(2*s^2))/(sqrt(2*%pi)*s);
    end
endfunction
```

2. Le script suivant permet d'utiliser cette fonction.

```
clear
exec('TP1_Echauffement.sci')

vect_x=linspace(-6,6,100);
f_1=densite_gaussienne(vect_x,0,1)
f_2=densite_gaussienne(vect_x,0,2)
f_3=densite_gaussienne(vect_x,0,sqrt(0.2))


scf(1)
clf
plot2d(vect_x',[f_1',f_2',f_3'],style=[1,2,3],leg='s2=1@s2=4@s2=0.2')
```

La Figure 1 est obtenue.

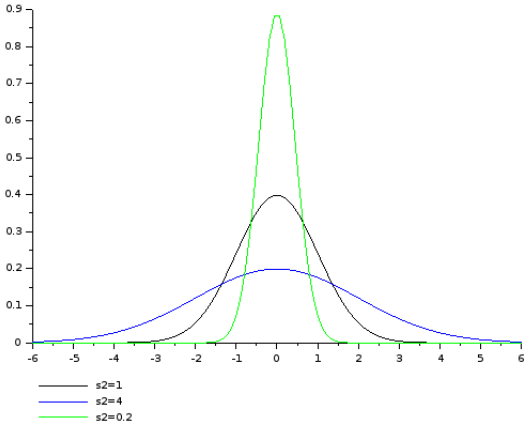


Figure 1 – Exercice 1. Tracé des densités de lois normales centrées pour 3 écarts-types s différents.

Exercice 2. **1.** Les commandes suivantes permettent de générer un vecteur colonne de n réalisations indépendantes d'une variable X de loi normale de moyenne 3 et d'écart-type 1.

```
n=100;
Xsimu=grand(n,1,'nor',m,s);
```

2. En notant $X_{simu} = [x_1, \dots, x_n]$, la moyenne empirique $\bar{X}$ et la variance empirique s^2 de X_{simu} sont respectivement

$$\bar{X} = \frac{1}{n}\sum_{i=1}^{n} x_i, \quad s^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{X})^2.$$

La moyenne empirique $\bar{X}$ est un estimateur (=fonction mesurable des données x_i) sans biais de l'espérance théorique $\mathbb{E}[X] = 3$, ce qui signifie que $\mathbb{E}[\bar{X}] = 3$. Elle est obtenue par la commande `mean(Xsimu)`.

La variance empirique est un estimateur biaisé de la variance théorique $\mathrm{Var}(X) = 1$: on a $\mathbb{E}[s^2] = (n-1)/n\mathrm{Var}(X)$. La commande `variance(Xsimu)` calcule automatiquement la version "débiaisée" de l'estimateur, à savoir $s^2_{bis} = n/(n-1)s^2 = \sum_{i=1}^{n}(x_i - \bar{X})^2/(n-1)$ (on a alors bien $\mathbb{E}[s^2_{bis}] = \mathrm{Var}(X)$).

Exercice 3. **1.** Pour calculer $k!$, on utilise `factorial(k)` ou `prod(1:k)`. Pour calculer la liste $1!, 2!, \dots, k!$, on utilise `factorial(1:k)` ou `cumprod(1:k)`. Scilab calcule 170! mais pas 171!, qui dépasse le plus grand nombre représenté en machine.
2 -3 -4 . Voici les fonctions demandées, enregistrées dans un fichier `TP1Ex3Coefficients_Binomiaux.sci`. Attention aux initialisations : $\binom{n}{0} = 1$...

```
//fonctions de calculs des coefficients binomiaux:
//chacune retourne le vecteur des C_n^k pour k=0,1,...,n

function Cnk=CoeffBin1(n)
    //utilisation de la formule Cnk=n(n-1)...(n-k+1)/k! et boucle
    Cnk=ones(1,n+1)
    for k=1:n
        Cnk(k+1)=prod(n:-1:(n-k+1))/prod(1:k);
    end
endfunction

function Cnk=CoeffBin2(n)
    //utilisation de la formule Cnk=n(n-1)...(n-k+1)/k! sans boucle
    Cnk=[1,cumprod(n:-1:1)./cumprod(1:n)];
endfunction

function Cnk=CoeffBin3(n)
    //utilisation de la FDR de la loi B(n,1/2)
    Cnk=ones(1,n+1);
    for k=1:n
        Cnk(k+1)=2^n*(cdfbin('PQ',k,n,0.5,0.5)-cdfbin('PQ',k-1,n,0.5,0.5));
    end
endfunction

function Cnk=CoeffBin4(n)
    //utilisation du triangle de Pascal
    Cnk=1;
    for k=1:n
        Cnk=[Cnk 0]+[0 Cnk]
    end
endfunction
```

Pour la troisième fonction, on utilise, en notant F_n la fonction de répartition d'une variable S_n suivant une loi $\mathcal{B}(n, 1/2)$,

$$\mathbb{P}(S_n = k) = F_n(k) - F_n(k-1) = \binom{n}{k}\left(\frac{1}{2}\right)^k\left(1-\frac{1}{2}\right)^{n-k} = \binom{n}{k}\left(\frac{1}{2}\right)^n. \quad (1)$$

Pour la quatrième, $\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$: on écrit le vecteur ligne des coefficients binomiaux $\binom{n}{k-1}$ à un rang $n-1$, on le recopie en dessous avec un décalage d'un cran, et on additionne les deux.

Le script suivant (à exécuter "avec écho") permet de tester ces fonctions.

```
clear
exec('TP1Ex3Coefficients_Binomiaux.sci')

n=5;
CoeffBin1(n)
CoeffBin2(n)
CoeffBin3(n)
n=171;
C1=CoeffBin1(n); C1(1:5), C1(168:172)
C2=CoeffBin2(n); C2(1:5), C2(168:172)
C3=CoeffBin3(n); C3(1:5), C3(168:172)
C4=CoeffBin4(n); C4(1:5), C4(168:172)

n=60
C3bis=CoeffBin3(n); C3bis(1:5), C3bis(57:61)
C4bis=CoeffBin4(n); C4bis(1:5), C4bis(57:61)
```

Quand n est trop grand, on obtient, en calculant les factorielles, des valeurs dépassant le plus grand nombre représenté en machine, ce qui explique les "erreurs" des deux premières fonctions `CoeffBin1` et `CoeffBin2` (résultat de type `Inf` ou `NaN` selon que le logiciel a un numérateur infini à calculer, ou un numérateur et un dénominateur infini, dans les fonctions). La fonction `CoeffBin3` basée sur la fonction de répartition de la loi binomiale (voir (1)) renvoie 0 pour $\binom{n}{k}$ quand k est proche de n, ce qui s'explique par le fait que $F_n(k) - F_n(k-1)$ est dans ce cas très petit, et donc arrondi à 0 (valeur tombant en dessous du plus petit réel strictement positif représenté en machine).

Exercice 4. **1.** Voici une fonction renvoyant le vecteur $(B_n f(x))_{x \in vect\_x}$, pour une fonction f, un entier n et un vecteur d'abscisses $vect\_x$, fonction enregistrée dans un fichier `TP1Ex4Polynomes_Bernstein`.

```
function [Bnfx]=Polynome_Bernstein(f,n,vect_x)
    //retourne les valeurs aux abscisses vect_x du nième polynome de Bernstein de la fonction f

    Bnfx=zeros(length(vect_x),1)
    //Calcul des coeff binomiaux C_n^k pour k=0...n, par le triangle de Pascal (voir TP1, Ex3)
    CoeffBin=[1];
    for i=1:n
        CoeffBin=[CoeffBin 0]+[0 CoeffBin];
    end

    //Calcul des C_n^k*f(k/n)
    Interm=CoeffBin.*f([0:n]/n)

    //Calcul des C_n^k*f(k/n)x^k(1-x)^{n-k}
    for i=1:length(vect_x)

        Bnfx(i)=sum(Interm.*(vect_x(i).^(0:n).*(1-vect_x(i)).^(n:-1:0)));
    end

endfunction
```

2. Un script associé à la fonction ci-dessus est le suivant.

```
clear
exec('TP1Ex4Polynomes_Bernstein.sci')

vect_x=linspace(0,1,100)' //vecteur d'abscisses
```

```
scf(1) // cas de la fonction indiquee dans l'enonce
clf
deff('[y]=f(x)','y=sqrt(abs(2*x-1))')
plot2d(vect_x,[f(vect_x), Polynome_Bernstein(f,10,vect_x),...
   Polynome_Bernstein(f,100,vect_x), Polynome_Bernstein(f,200,vect_x)],...
   style=[1,2,3,5],leg='f@B10f@B100f@B200f')

scf(2) // cas de la fonction racine
clf
plot2d(vect_x,[sqrt(vect_x), Polynome_Bernstein(sqrt,10,vect_x),...
   Polynome_Bernstein(sqrt,100,vect_x), Polynome_Bernstein(sqrt,200,vect_x)],...
   style=[1,2,3,5],leg='f@B10f@B100f@B200f')

scf(3) // cas de la fonction exponentielle
clf
plot2d(vect_x,[exp(vect_x), Polynome_Bernstein(exp,10,vect_x),...
   Polynome_Bernstein(exp,100,vect_x), Polynome_Bernstein(exp,200,vect_x)],...
   style=[1,2,3,5],leg='f@B10f@B100f@B200f')
```

La figure obtenue est la suivante.

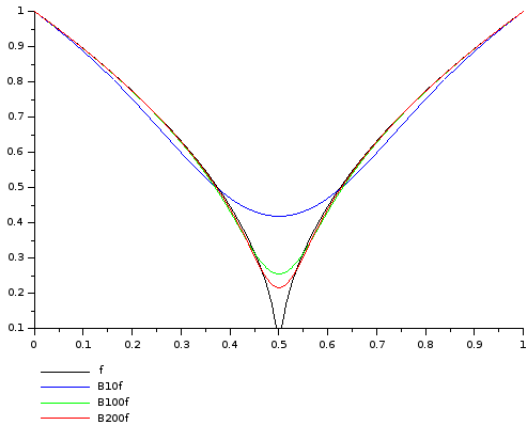


Figure 2 – Exercice 4. Tracé de 3 polynômes de Bernstein associés à la fonction $f : x \mapsto \sqrt{|2x-1|}$.

Exercice 5.

```
clear
n=5000;

scf(1)
clf
//LGN pour la loi exponentielle Exp(lambda)
lambda=1;
X_exp=grand(1,n,"exp",1/lambda); //simulation d'un n-echantillon de loi Exp(lambda)
S_exp=cumsum(X_exp)./[1:n]; // suite des moyennes empiriques
subplot(1,3,1)
plot2d([1:n],S_exp,style=1) //trace des moyennes empiriques
plot2d([1,n],[1/lambda,1/lambda],style=5) //trace de la droite correspondant à la moyenne théorique
xtitle('LGN pour la loi Exp('+string(lambda)+')','n','')
```

```
//LGN pour la loi normale N(m,sigma2)
m=0;
sigma=1;
X_norm=grand(1,n,"nor",m,sigma);
S_norm=cumsum(X_norm)./[1:n];
subplot(1,3,2)
plot2d([1:n],S_norm,style=1)
plot2d([1,n],[m,m],style=5)
xtitle('LGN pour la loi N('+string(m)+','+string(sigma)+')','n','')

//LGN pour la loi Uniforme sur les entiers de k1 à k2
k1=0;
k2=10;
esp=(k1+k2)/2;
X_unif=grand(1,n,"uin",k1,k2);
S_unif=cumsum(X_unif)./[1:n];
subplot(1,3,3)
plot2d([1:n],S_unif,style=1)
plot2d([1,n],[esp,esp],style=5)
xtitle('LGN pour la loi U('+string(k1)+',...,'+string(k2)+')','n','')

scf(2)
clf
//Cas de la loi de Cauchy C(1) //voir explications ci-dessous
NbTrajectoire=5;
X1=grand(n,NbTrajectoire,"nor",0,1)
X2=grand(n,NbTrajectoire,"nor",0,1)
X_cau=X1./X2
S_cau=cumsum(X_cau,'r')./([1:n]'*ones(1,NbTrajectoire));
plot2d([1:n]',S_cau)
xtitle('Cas de la loi de Cauchy','n','')
```

Dans le cas de la loi de Cauchy, on trace 5 (variable `NbTrajectoire`) trajectoires de la suite des moyennes empiriques : on simule un tableau à n lignes et `NbTrajectoire` colonnes dont les entrées sont des réalisation indépendantes de même loi de Cauchy, en faisant le quotient terme à terme de deux tableaux dont les entrées indépendantes suivent la loi normale centrée réduite. On obtient `X_cau` :

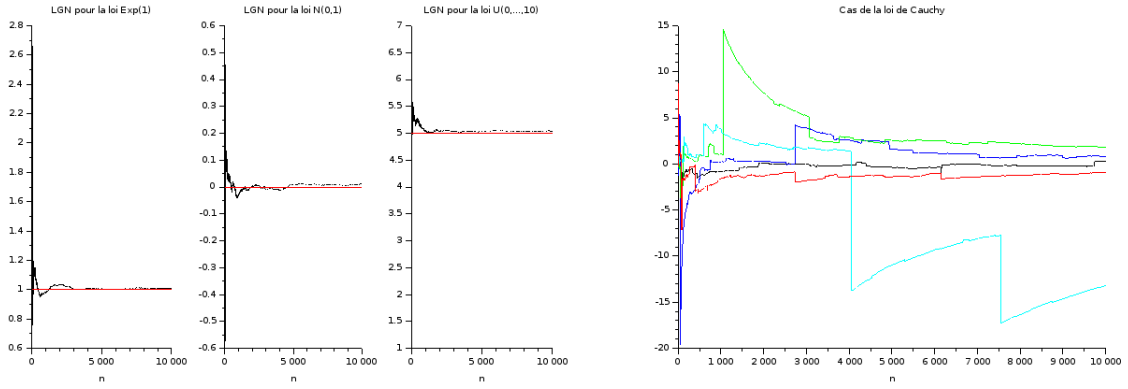
$$\texttt{X\_cau} = \begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,\texttt{NbTrajectoire}} \\ x_{2,1} & x_{2,2} & & \\ \vdots & & \ddots & \vdots \\ x_{n,1} & & \cdots & x_{n,\texttt{NbTrajectoire}} \end{pmatrix}$$

On somme de manière cumulée selon les lignes (paramètre `'r'` de la commande `cumsum`), ce qui donne :

$$\texttt{cumsum(X\_cau)} = \begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & \cdots & x_{1,\texttt{NbTrajectoire}} \\ x_{1,1}+x_{2,1} & x_{2,1}+x_{2,2} & & & x_{1,\texttt{NbTrajectoire}}+x_{2,\texttt{NbTrajectoire}} \\ \vdots & \vdots & \ddots & & \vdots \\ \vdots & & & \ddots & \vdots \\ x_{1,1}+\cdots+x_{1,n} & \cdots & \cdots & & x_{1,\texttt{NbTrajectoire}}+x_{n,\texttt{NbTrajectoire}} \end{pmatrix}$$

et on divise terme à terme par `[1:n]'*ones(1,NbTrajectoire)` qui est une matrice à n lignes et `NbTrajectoire` colonnes, chaque colonne étant égale au vecteur colonne `[1:n]`. On obtient la matrice `S_cau`, dont la colonne l ($l = 1, \ldots,$ `NbTrajectoire`) correspond au vecteur colonne $[x_{1,l}/1, (x_{1,l} +$

$x_{2,l})/2 \ldots, (x_{1,l} + x_{2,l} + \cdots + x_{n,l})/n$. On représente `NbTrajectoire` courbes correspondant aux colonnes de cette matrice (pour les ordonnées, les abscisses étant données par le vecteur `[1:n]'`. On obtient les deux graphes de la Figure 3.



(a) (b)

Figure 3 – Exercice 5. (a) Illustration de la loi des grands nombres (convergence de la suite des moyennes empiriques vers l'espérance. (b) Quelques trajectoires de la suite des moyennes empiriques dans le cas de la loi de Cauchy.

Exercice 6. **1.** Pour presque tout ω, la suite $(M_n(\omega))_n$ est croissante, majorée par 1, donc convergente.
2. Voici un script illustrant la convergence de la suite.

```
clear
scf(1)
clf
n=1000; //taille echantillon
NbRep=6;  // nb de repetitions de la simulation
MatM=zeros(n,NbRep); //chq colonne va contenir un echantillon M1, M2, ..., Mn
X=rand(n,NbRep)
for j=1:NbRep
    M=[X(1,j);zeros(n-1,1)] //vecteur contenant les M1, M2, ..., Mn
    for i=2:n
        M(i)=max(M(i-1),X(i,j))
    end
    MatM(:,j)=M;
end
plot2d([1:n]',MatM)
```

Comme $(M_n)_n$ converge presque-sûrement, elle converge également en probabilité et donc en loi. Il suffit donc de chercher la limite en loi. On passe par la caractérisation de la convergence en loi par la convergence simple des fonctions de répartition (en tout point de continuité de la répartition de la loi limite). Pour tout $x \in \mathbb{R}$, par indépendance et équidistribution des X_i,

$$\mathbb{P}(M_n \leq x) = (\mathbb{P}(X_1 \leq x))^n = \begin{cases} 0 \text{ si } x < 0 \\ x^n \text{ si } 0 \leq x < 1 \\ 0 \text{ si } x \geq 1 \end{cases} .$$

Par conséquent, $\lim_{n\to\infty} \mathbb{P}(M_n \leq x) = \mathbf{1}_{[1;\infty[}(x)$. Donc $(M_n)_n$ converge presque sûrement vers la variable aléatoire p.s. constante égale à 1.