Université Paris Diderot (Paris VII)
et Pierre et Marie Curie (Paris VI)
Année 2011/2012.

Préparation à l'Agrégation externe de Mathématiques (Option Probabilités)

Feuille de TP n°1 : Introduction Probabiliste à Scilab (Correction)

1 Informations Pratiques

1. Scilab est un logiciel libre téléchargeable à l'adresse `www.scilab.org`
2. Scilab possède une aide en ligne (que l'on obtient en tapant la commande `help` ou `help rand` si l'on veut avoir des explications concernant la fonction `rand`).

2 Quelques manipulations de matrices

On présente ci-après quelques commandes de base relatives à la manipulation de matrices, de booléens et à l'affichage.

Initialisation manuelle d'une matrice :

```
M=[1 3 5 ; -2 6 4]
```

Reprendre cette commande en ajoutant un `;` final.

Cette commande crée une variable matricielle $M = \begin{pmatrix} 1 & 3 & 5 \\ -2 & 6 & 4 \end{pmatrix}$. Si l'on omet le point virgule la matrice est affichée par `Scilab`. Avec le point virgule elle est bien initialisée mais n'est plus affichée.

Extraction de lignes et colonnes :

```
l=M(1, :) ; c=M( :,3)
```

Les commandes ci-dessus permettent d'extraire respectivement la première ligne et la troisième colonne de la matrice M. On a ainsi $l = (1\ 3\ 5)$, $c = \begin{pmatrix} 5 \\ 4 \end{pmatrix}$.

Transposition de `M` :

```
N=M'
```

Initialisation automatique d'un vecteur :

```
[0 :0.1 :2]
```

La commande précédente initialise un vecteur ligne dont les composantes s'échelonnent de 0 à 2 par pas de 0.1 (il contient donc 21 éléments).

Initialisation d'une matrice de zéros :

```
P=zeros(4,5)
```

La matrice P est une matrice de dimension 4×5 initialisée avec des entrées nulles.

$\rightsquigarrow$ Que fait la commande `A=ones(M)` ?

Elle initialise la matrice A qui a même dimension que M et ne contient que des 1.

Suppression de colonne :

```
P( :,3)=[]
```

Initialisation d'une matrice par blocs `x=[1 :2] ; P=[x 2*x ; 3*x 4*x]`

La commande `x=[1 :2]` peut sembler ambigue. En effet elle correspond à une initialisation similaire à `[0 :.1 :2]` sauf que dans ce cas là on ne précise pas la valeur du pas (qui par défaut vaut 1). Ainsi `x=[1 :2]` fournit le résultat $x = (1\ 2)$ que l'on aurait également obtenu par la commande `x=[1,2]`. La matrice `P` est une matrice 2×4 qui contient $P = \begin{pmatrix} 1 & 2 & 2 & 4 \\ 3 & 6 & 4 & 8 \end{pmatrix}$.

$\rightsquigarrow$ Initialiser une matrice 3×4 qui contienne `M` en haut à gauche, `x'` en haut à droite, `x` en bas à gauche et `[0 0]` en bas à droite, en utilisant la construction par bloc précédente.

La commande `R=[M x' ;x [0,0]]` répond à la question. On observe au passage que la construction par blocs s'étend aux matrices. Ici `[M x']` forme une matrice 2×4 et `x [0,0]` est un vecteur ligne à 4 colonnes.

$\rightsquigarrow$ Que renvoie la commande `s=ones(x)'*x` ? Il faut être vigilant aux dimensions ici. En effet `ones(x')` renvoie le vecteur *colonne* $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$. La variable `s` contient donc le produit *matriciel* d'un vecteur colonne 2×1 et d'un vecteur ligne 1×2, c'est donc une matrice 2×2 et $s = \begin{pmatrix} 1 & 2 \\ 1 & 2 \end{pmatrix}$.

$\rightsquigarrow$ Si `x` et `y` sont deux vecteurs colonnes, comment calculer leur produit scalaire ? Il suffit d'en transposer un des deux pour en faire un vecteur ligne et de le multiplier avec l'autre, i.e $x = [1; 2]; y = [3; 4]$ alors $\langle x, y \rangle = 11$ s'obtient de façon équivalent par les commandes $x' * y$ ou $y' * x$.

Vecteurs booléens.

`l=M>2 ;`

Cette commande renvoie une matrice de même dimension que M à entrées booléennes (i.e. prenant les valeurs `T` (pour *True*) et `F` (pour *False*). Les entrées sont ici `T` ou `F` en fonction du résultat du test coefficient par coefficient pour la matrice `M`, i.e. on teste que chaque coefficient est ou non strictement supérieur à 2. On a $l = \begin{pmatrix} F & T & T \\ F & T & T \end{pmatrix}$ pour $M = \begin{pmatrix} 1 & 3 & 5 \\ -2 & 6 & 4 \end{pmatrix}$.

$\rightsquigarrow$ Expliquer la commande `x=[-2 :.5 :2] ; l=(x>-1 & x<1) ; x(l)=3.14 ; x`

La première commande initialise le vecteur `x` ; ses composantes varient de -2 à 2 par pas de 0.5 ainsi $x = (-2\ -1.5\ -1\ -.5\ 0\ 0.5\ 1\ 1.5\ 2)$ vecteur à 9 éléments. La deuxième commande renvoie dans `l` une matrice booléenne de même dimension que `x` dont les entrées valent `T` aux indices pour lesquels les coefficients de `x` sont dans $]-1, 1[$, le caractère `&` correspond au 'et' logique (le 'ou' s'écrit lui `|`). Ainsi $l = (F\ F\ F\ T\ T\ T\ F\ F\ F)$. La commande `x(l)=3.14` affecte aux indices de `x` pour lesquels ceux de `l` valent `T` la valeur 3.14. La dernière instruction `x` affiche donc $x = (-2\ -1.5\ -1\ 3.14\ 3.14\ 3.14\ 1\ 1.5\ 2)$.

$\rightsquigarrow$ D'autre commandes à tester.

`a=sum(x) ; a=mean(x) ; x.^2 ; sort(x) ; -sort(-x) ; M.^2 ; M^2 ;`

Les fonctions `sum` et `mean` renvoient respectivement la somme et la moyenne des coefficients contenus dans le vecteur `x`. La fonction `sort` effectue un tri par ordre décroissant des valeurs du vecteur `x`. La commande `-sort(-x)` permet d'obtenir le tri par ordre croissant. Enfin les commandes `x.^2,M.^2` effectuent une élévation au carré terme à terme là où la commande `M^2` renvoie une erreur. En effet cette commande est équivalente à l'instruction `M*M` qui requiert une matrice carrée en argument.

3 Générer de l'Alea

La fonction la plus simple s'appelle `rand`. Elle permet de simuler des nombres pseudo-aléatoires distribués selon la loi uniforme sur $[0, 1]$. La fonction `grand` permet de simuler des échantillons suivant toutes les lois de probabilité classiques (on renvoie à l'aide en ligne pour plus de précisions).

$\rightsquigarrow$ Tester la commande `grand (1,1) ;`

Cette commande retourne en effet comme suggéré dans l'énoncé 0.2113249. En fait dans la mesure où cette fonction ne fait qu'itérer les valeurs d'une suite "pseudo-aléatoire" (de la forme $x_{n+1} = ax_n + b[1]$ si l'on pense à un générateur de type congruenciel) la valeur obtenue pour le paramètre d'initialisation par défaut x_0 est toujours la même. Pour explicitement initialiser ce paramètre on utilise la commande `rand('seed',n)` où `n` est un réel. **Attention :** on a indiqué ici l'exemple du générateur congrenciel pour donner une idée de ce qu'est une suite pseudo-aléatoire, le générateur de `Scilab` est malgré tout beaucoup plus évolué.

$\rightsquigarrow$ Tester les commandes `rand(2,4) ; grand(1,3,'exp',2) ; grand(3,1,'nor',1,2) ;`

La première commande renvoie une matrice de taille 2×4 contenant des réalisations uniformes "indépendantes" sur $[0, 1]$. La deuxième renvoie une matrice 1×3 de réalisations exponentielles indépendantes de moyenne 2. **Attention :** pour la loi exponentielle c'est la moyenne qui apparaît dans la fonction `grand` et non le paramètre. Ici on simule donc des lois exponentielles de paramètre $1/2$. La dernière commande renvoie une matrice 3×1 de réalisations normales indépendantes de paramètres $m = 0, \sigma = 2$. C'est l'écart-type qui est en argument pour la simulation de lois normales avec `grand`, i.e. la variance est ici $\sigma^2 = 4$.

$\rightsquigarrow$ Que font les commandes suivantes :

1. `plot2d(cumsum(rand(1,1000))./[1 :1000]);`
   Cette commande est une écriture compacte de la suite d'instructions suivante :

```
U=rand(1,1000); // On tire un vecteur ligne de taille 1000 de réalisations uniformes
X=cumsum(U);    // On stocke dans X les sommes cumulées desdites réalisations
X=X./[1:1000];  // On divise l'entrée X(i) par i (Division terme à terme)
plot(X);        // On affiche le résultat
```

   Cette opération conduit à ce que $X_i := \frac{1}{i}\sum_{j=1}^{i} U_j$ où $(U_j)_{j\geq 1}$ est une suite i.i.d. de $\mathcal{U}_{[0,1]}$. Ainsi on affiche la moyenne empirique en fonction du nombre de réalisations.

2. `plot2d([1 :1000],cumsum(rand(1,1000))./[1 :1000],4)`
   C'est la même commande que précédemment sauf que l'on utilise le `plot2d` dans sa version `plot(x,y)` où `x` représente le vecteurs des abscisses et `y` celui des ordonnées. Par défaut, si l'on omet l'argument `x`, celui-ci est échelonné de 1 en 1 entre 1 et la taille de `y`. Ici l'abscisse par défaut et celle explicitement indiquée en question 2. coincident.

3. `xbasc() ; plot2d(cumsum(-log(rand(1,1000)))./[1 :1000])`
   La commande `xbasc() ;` permet d'effacer les graphiques de la fenêtre de `Scilab` prévue à cet effet. On peut aussi dans les versions plus récentes utiliser la commande `clf ;` Il faut également remarquer que la commande `-log(rand(1,1000))` fournit un 1000-échantillon i.i.d. de variables exponentielles de paramètre 1. En effet, un procédé de simulation des lois exponentielles est directement fourni par la méthode d'inversion de la fonction de répartition que l'on peut formuler de la façon suivante (cf. Bercu et Chafai [BC07] ou Ouvrard [Ouv00]).

Proposition 3.1 (Simulation par inversion de la fonction de répartition) *Soit X une variable aléatoire réelle de fonction de répartition F. Pour $u \in [0,1]$, on désigne par $F^-(u) := \inf\{x \in \mathbb{R} : F(x) \geq u\}$ l'inverse généralisée de la fonction de répartition F. Si $U \sim \mathcal{U}_{[0,1]}$ alors $F^-(U)$ a pour fonction de répartition F.*

On observe que lorsque F est bijective alors F^- coincide avec la fonction inverse usuelle F^{-1}. C'est bien le cas par exemple pour la loi exponentielle de paramètre $\lambda > 0$ pour laquelle on a pour tout $x \in \mathbb{R}^+$, $F_\lambda(x) := \mathbb{P}[\mathcal{E}(\lambda) \leq x] = \int_0^x \lambda \exp(-\lambda y)dy = 1 - \exp(-\lambda x)$. Il vient ainsi pour tout $u \in [0,1]$, $F^{-1}(u) = -\frac{1}{\lambda}\log(1-u)$.

D'après la proposition, si $U \sim \mathcal{U}_{[0,1]}$ alors $X := -\frac{1}{\lambda}\log(1-U) \sim \mathcal{E}(\lambda)$. Comme par ailleurs $U \overset{\text{(loi)}}{=} 1-U$ on a encore que $-\frac{1}{\lambda}\log(U) \sim \mathcal{E}(\lambda)$. La commande représente toujours la moyenne empirique d'échantillons de lois exponentielles de paramètre 1 en fonction du nombre de réalisations dans l'échantillon.

4. `plot2d([1 :1000]',cumsum(grand(1000,10,'nor',0,1),'r')./([1 :1000]'*ones(1,10)));`
   On trace ainsi les moyennes empiriques associées au nombre de réalisations pour 10 différents 1000-échantillons de variables normales centrées réduites i.i.d. On observe que l'abscisse est un vecteur colonne et que la matrice des réalisations a autant de lignes que la taille de l'échantillon.

5. `histplot(100,grand(1,1000,'exp',2)) ;`
   Cette commande permet de réaliser l'histogramme associé à la répartition de la réalisation d'un échantillon de taille 1000 de lois exponentielles de paramètre 1/2.

6. `xtitle('Histogramme expo')`
   Cette commande permet simplement de rajouter un titre au graphique.

4 Loi forte des grands nombres

On peut rappeler le résultat.

Théorème 4.1 (Loi des grands nombres) *Soit $(X_i)_{i\in\mathbb{N}^*}$ une suite de variable aléatoires réelles i.i.d de loi X définies sur un espace de probabilité $(\Omega, \mathcal{F}, \mathbb{P})$.*

- *Si $X \in L^1(\mathbb{P})$, i.e. $\mathbb{E}[|X|] < +\infty$, alors $\bar{X}_n := \frac{1}{n}\sum_{i=1}^n X_i \underset{\mathbb{P}-p.s.,n}{\longrightarrow} \mathbb{E}[X]$.*

- *Réciproquement, si $\bar{X}_n$ converge presque sûrement vers un réel a, alors X est intégrable et $\mathbb{E}[X] = a$.*

Nous avons illustré à la question précédente que les moyennes empiriques des lois uniformes, exponentielles et normale convergeaient bien vers leur moyenne. Il est intéressant de remarquer que lorsque la condition d'intégrabilité n'est pas vérifiée, ce qui est par exemple le cas pour la loi de Cauchy, i.e. lorsque X admet la densité $f_C(x) = \frac{1}{\pi(1+x^2)}$, on observe numériquement un phénomène de "non-convergence" et non de divergence. Précisément les moyennes empiriques ne stabilisent pas. On peut simuler des lois de Cauchy en utilisant le

procédé décrit à la Proposition 3.1. On a pour tout $x \in \mathbb{R}$, $F_C(x) := \int_{-\infty}^{x} \frac{dy}{\pi(1+y^2)} = \frac{1}{\pi}\arctan(y)|_{-\infty}^{x} = \frac{\arctan(x)}{\pi} + \frac{1}{2}$ soit encore pour $u \in [0,1]$, $F_C^{-1}(u) = \tan(\pi(u-1/2))$. On propose le script `Scilab` suivant pour simuler la loi de Cauchy.

```
// Cette fonction renvoie une matrice de taille l\times c
// contenant des realisations de loi de Cauchy
function res=Cauchy(l,c)
  r=rand(l,c); // on simule des lois uniformes
  res=tan(%pi*(r-1/2));
endfunction
```

Un résultat d'exécution sous `Scilab` est le suivant

```
-->res=Cauchy(1,100000);
 -->mean(res)
 ans  =    3.0596384
 -->mean(Cauchy(1,100000))
 ans  =    0.987704
 -->mean(Cauchy(1,100000))
 ans  =  - 8.9034353
-->mean(Cauchy(1,1000000))
 ans  =    4.5953611
```

5 Théorème central limite

La loi des grands nombres donne la convergence des moyennes empiriques vers la moyenne (si la variable sous-jacente X est intégrable, i.e. $X \in L^1(\mathbb{P})$). Lorsque l'on a des moments d'ordre 2, i.e. $X \in L^2(\mathbb{P})$, le théorème central limite va donner un résultat asymptotique concernant la distribution renormalisée de la différence entre moyenne empirique et moyenne. Précisément on a le résultat suivant.

Théorème 5.1 (Théorème central limite) *Soit $(X_i)_{i\in\mathbb{N}^*}$ une suite de variable aléatoires réelles i.i.d de loi X définies sur un espace de probabilité $(\Omega, \mathcal{F}, \mathbb{P})$. Si $X \in L^2(\mathbb{P})$ en notant $m = \mathbb{E}[X]$, $\sigma = \mathbb{E}[(X - \mathbb{E}[X])^2]^{1/2}$ on a*

$$\frac{\sqrt{n}}{\sigma}(\bar{X}_n - m) \xrightarrow[n]{\text{(loi)}} \mathcal{N}(0,1), \ \bar{X}_n := \frac{1}{n}\sum_{i=1}^{n} X_i.$$

Remarque 5.1 *Ce théorème fondamental permet notamment de donner des **intervalles de confiance** pour les estimateurs des moyennes. En effet si l'on pose $Z^{(n)} := \frac{\sqrt{n}}{\sigma}(\bar{X}_n - m)$, la convergence en loi donne que pour $-\infty < a < b < +\infty$, $\mathbb{P}[Z^{(n)} \in [a,b]] \underset{n}{\longrightarrow} \mathbb{P}[\mathcal{N}(0,1) \in [a,b]]$. On exploite ensuite ce résultat pour construire un intervalle de confiance au seuil α, $\alpha \in [0,1)$. On va pour cela lire $C(\alpha) := \inf\{x \in \mathbb{R}^+ : \mathbb{P}[\mathcal{N}(0,1) \in [-x,x]] \geq \alpha\}$ à l'aide d'une table de loi normale, ou plus directement en `Scilab` à l'aide de la fonction `cdfnor` qui évalue la fonction de répartition de la loi normale. Cette fonction a 4 ou 5 arguments, un appel possible est le suivant :* `[P,Q]=cdfnor("PQ",X,Mean,Std)` *qui renvoie la valeur* `P` $= \frac{1}{(2\pi)^{1/2}\texttt{Std}} \int_{-\infty}^{\texttt{X}} \exp(-\frac{(y-\texttt{Mean})^2}{2\texttt{Std}^2})dy$, `Q` $= 1 -$ `P`. *On peut ensuite, pour un α donné, trouver le $C(\alpha)$ en procédant par dichotomie. Notons que la valeur la plus communément employée correspond au seuil $\alpha = 0.95$ pour lequel $C(\alpha) = 1.96$.*

La convergence en loi précédente donne ensuite que pour n "grand", $\mathbb{P}[Z^{(n)} \in [-C(\alpha), C(\alpha)]] \simeq \mathbb{P}[\mathcal{N}(0,1) \in [-C(\alpha), C(\alpha)]] = \alpha$ ce que l'on peut encore réécrire comme $\mathbb{P}[m \in [\bar{X}_n - \frac{C(\alpha)\sigma}{\sqrt{n}}, \bar{X}_n + \frac{C(\alpha)\sigma}{\sqrt{n}}]] \simeq \alpha$. Ce contrôle permet ainsi d'estimer la probabilité que le paramètre que l'on cherche à estimer, i.e. le m, se trouve dans un intervalle centré autour de la moyenne empirique $\bar{X}_n$ que l'on a calculé sur l'ordinateur et d'amplitude $2C(\alpha)\sigma/\sqrt{n}$. On dit alors que pour n "grand" avec une probabilité "proche" de α, $m \in [\bar{X}_n - \frac{C(\alpha)\sigma}{\sqrt{n}}, \bar{X}_n + \frac{C(\alpha)\sigma}{\sqrt{n}}] := I_C(\alpha, n, \sigma)$. C'est l'intervalle $I_C(\alpha, n, \sigma)$ que l'on appelle intervalle de confiance.

Notons que les paramètres $C(\alpha)$ et $1/\sqrt{n}$ sont intrinsèques à l'objet limite (la loi normale) et que le σ dépend lui de la loi dont on cherche à estimer la moyenne.

Deux points méritent d'être précisés dans le procédé ci-dessus.

1. Notons que l'on cherche à estimer numériquement la moyenne $m = \mathbb{E}[X]$ d'une loi X et que l'on donne l'intervalle de confiance en terme de l'écart-type qui fait intervenir le moment d'ordre 2 de X que l'on connaît *a priori* encore moins !

   On peut aisément contourner ce problème en calculant numériquement (pour les mêmes réalisations que celles utilisées pour l'estimation de la moyenne empirique) un estimateur $\hat{\sigma}_n^2$ qui va converger presque sûrement vers σ^2 et pour lequel on aura toujours

$$\bar{Z}^{(n)} = \frac{\sqrt{n}}{\hat{\sigma}_n}(\bar{X}_n - m) \xrightarrow[n]{\text{(loi)}} \mathcal{N}(0,1),\ \bar{X}_n := \frac{1}{n}\sum_{i=1}^{n} X_i,\ \hat{\sigma}_n \underset{\mathbb{P}-p.s.,\ n}{\longrightarrow} \sigma. \tag{5.1}$$

   Pour établir (5.1) on va utiliser le Lemme de Slutsky que l'on rappelle ci-dessous.

Lemme 5.1 (Lemme de Slutsky) *Soit $(A_n)_{n\geq 1}, (B_n)_{n\geq 1}$ deux suites de variables aléatoires réelles définies sur le même espace de probabilité $(\Omega, \mathcal{F}, \mathbb{P})$. Si*

$$A_n \xrightarrow[n]{\text{(loi)}} A,\ B_n \xrightarrow[n]{\mathbb{P}} b,$$

où b est une constante alors le ***couple*** $(A_n, B_n) \xrightarrow[n]{\text{(loi)}} (A, b)$.

   Nous renvoyons pour une preuve de ce lemme à Ouvrard [Ouv00]. Ecrivons maintenant $\bar{Z}^{(n)} = Z^{(n)} \times \frac{\sigma}{\hat{\sigma}_n}$. On a bien d'après le Théorème 5.1 que $Z^{(n)} \xrightarrow[n]{\text{(loi)}} \mathcal{N}(0,1)$ et pour n'importe quel estimateur consistant $\hat{\sigma}_n$ de σ, i.e. tel que $\hat{\sigma}_n \underset{\mathbb{P}-p.s.,n}{\longrightarrow} \sigma$, on aura bien $\frac{\sigma}{\hat{\sigma}_n} \underset{\mathbb{P}-p.s,\ n}{\longrightarrow} 1$, donc en particulier en probabilité. Le lemme donne ainsi $(Z^{(n)}, \frac{\sigma}{\hat{\sigma}_n}) \xrightarrow[n]{\text{(loi)}} (\mathcal{N}(0,1), 1)$. Comme la convergence en loi est conservée par image continue, on a en considérant $h : \mathbb{R}^2 \to \mathbb{R},\ (x,y) \mapsto h(x,y) = xy$, que $h(Z^{(n)}, \frac{\sigma}{\hat{\sigma}_n}) := \bar{Z}^{(n)} \xrightarrow[n]{\text{(loi)}} \mathcal{N}(0,1)$, soit (5.1). En général on prendra pour $\hat{\sigma}_n^2$ l'estimateur sans biais de la variance, i.e. tel que $\mathbb{E}[\hat{\sigma}_n^2] = \sigma^2$ soit $\hat{\sigma}_n^2 = \frac{1}{n-1}\sum_{i=1}^{n} X_i^2 - \frac{n}{n-1}\bar{X}_n^2$.

2. Le deuxième point qui fait question dans la remarque 5.1 est que l'on a considéré n "grand". Ce résultat est asymptotique et peu précis tel quel. Sans autres hypothèses sur les moments de la variable aléatoire X on ne peut hélas pas dire beaucoup plus. En revanche, dès lors que $X \in L^3(\mathbb{P})$ il est possible de contrôler en termes **non asymptotiques**, i.e. valables pour tout $n \in \mathbb{N}^*$, les écarts entre les fonctions de répartition des variables $Z^{(n)}$ et de l'objet limite $\mathcal{N}(0,1)$. Précisément on le théorème suivant :

Théorème 5.2 (Berry-Esseen) *Soit $(X_i)_{i\in\mathbb{N}^*}$ une suite de variable aléatoires réelles i.i.d de loi X définies sur un espace de probabilité $(\Omega, \mathcal{F}, \mathbb{P})$. Si $X \in L^3(\mathbb{P})$ en notant $m = \mathbb{E}[X],\ \sigma = \mathbb{E}[(X - \mathbb{E}[X])^2]^{1/2}$, si $Z^{(n)} := \frac{\sqrt{n}}{\sigma}(\bar{X}_n - m)$ et pour tout $x \in \mathbb{R},\ F_{Z^{(n)}}(x) := \mathbb{P}[Z^{(n)} \leq x],\ \Phi(x) := \mathbb{P}[\mathcal{N}(0,1) \leq x]$, alors il existe $C < 0.4784$ telle que*

$$\sup_{x\in\mathbb{R}, n\in\mathbb{N}^*} |F_{Z^{(n)}}(x) - \Phi(x)| \leq C\frac{\mathbb{E}[|X|^3]}{\sigma\sqrt{n}}.$$

   Nous renvoyons à l'ouvrage de Shiryaev [Shi96] pour une preuve. La constante indiquée a été obtenue récemment.

   Revenons manitenant sur les commandes de la Section 3. Dans la mesure où les variables pour lesquelles on a tracé les moyennes empiriques possèdent toutes des moments d'ordre 2, on va leur associer un intervalle de confiance au seuil 95%. On propose le code `Scilab` s suivant (qui évite la boucle `for` que l'on avait utilisée en cours) :

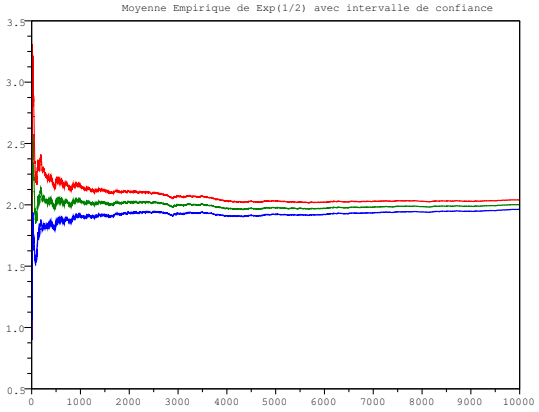
```
function res=IC(M)
  Alea=grand(M,1,'exp',2); // On tire l'aléa
  X=cumsum(Alea);          // Somme cumulée de l'aléa
  V=cumsum(Alea.^2);       // Somme cumulée des carrés de l'aléa
  Abscisse=[1:M]';
  Moy=X./Abscisse;         // Moyenne empirique
  Factor=Abscisse./(max(Abscisse-1,1)); // Facteur de normalisation pour la
                                        // variance empirique débiasée
  V=V./Abscisse-Factor.*Moy.^2;         // Variance empirique débiaisée
```

```
  Dev=1.96*sqrt(V)./sqrt(Abscisse); // Demi amplitude de l'intervalle de confiance a 5%
  res=[Moy-Dev,Moy,Moy+Dev ];
endfunction
```

On peut ensuite écrire dans `Scilab` les commandes :

```
res=IC(10000); plot([1:10000]',res);
xtitle 'Moyenne Empirique de Exp(1/2) avec intervalle de confiance'
```

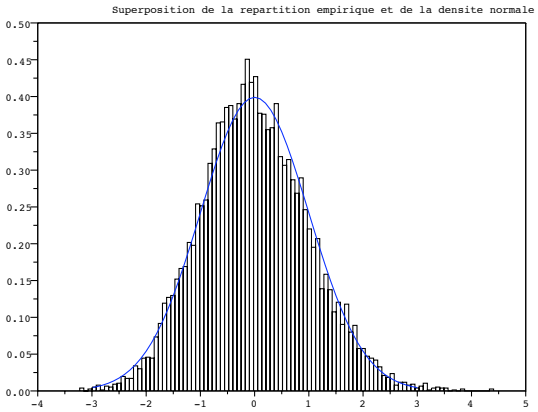
On note que pour afficher simultanément les trois courbes il faut que le vecteur d'abscisse en premier argument et la matrice en deuxième argument aient le même nombre de lignes. Le résultat produit est le suivant



On revient maintenant à la vérification du TCL. La fonction ci-après permet d'obtenir `M` réalisations indépendantes de la variable aléatoire $Z^{(n)} := \frac{\sqrt{n}}{2}(\bar{X}_n - 2)$ qui correspond bien à la normalisation du Théorème 5.1 lorsque $X \sim \mathcal{E}(1/2)$.

```
function R=VERIF_TCL(N,M)
  Alea=grand(M,N,'exp',2);
  Prov=ones(N,1);
  R=sqrt(N)*(Alea*Prov/N-2)/2;
endfunction
```

On présente les résultats obtenus pour $n = 100$ et `M`=100000. Notons pour cela qu'il est nécessaire d'augmenter les ressources mémoire de `Scilab`. Cette opération s'effectue par la commande `stacksize('max');`.



Testez ce qu'il se passe lorsque maintenant la variable aléatoire X sous-jacente n'est pas dans $L^2(\mathbb{P})$. On pourra par exemple considérer une variable aléatoire X de loi $\mu_X(dx) = \frac{dx}{1+|x|^\beta} C_\beta,\ \beta \in]2,3[,\ C_\beta := \left(\int_{\mathbb{R}} \frac{dy}{1+|y|^\beta}\right)^{-1}$.

6 Méthode de Monte Carlo

Nous avons déjà mis en oeuvre dans les sections précédentes des méthodes de Monte Carlo (en donnant **également** les **intervalles de confiance** associés). Le propos de cette section est plutôt de reformuler des intégrales en terme d'espérance pour pouvoir les estimer à l'aide d'une méthode de Monte-Carlo. On considère les trois exemples suivant :

1. $\int_0^1 4\sqrt{1-x^2}dx := I_1$ (Aire du disque unité).
2. $\int_{[-1,1]^2} \mathbb{I}_{x^2+y^2\leq 1} dxdy := I_2$ (Aire du disque unité).
3. $\int_{[-1,1]^3} \mathbb{I}_{x^2+y^2+z^2\leq 1} dxdydz := I_3$ (Volume de la boule unité).

On peut simplement réécrire en termes probabilistes :

1. $I_1 = 4\mathbb{E}[(1-U^2)^{1/2}],\ U \sim \mathcal{U}_{[0,1]}$.
2. $I_2 = 4\mathbb{E}[\mathbb{I}_{U_1^2+U_2^2\leq 1}] = 4\mathbb{P}[U_1^2+U_2^2 \leq 1]$ où U_1, U_2 sont des variables aléatoires indépendantes uniformes sur $[-1,1]$.
3. $I_3 = 8\mathbb{E}[\mathbb{I}_{U_1^2+U_2^2+U_3^2\leq 1}] = 8\mathbb{P}[U_1^2+U_2^2+U_3^2 \leq 1]$ où U_1, U_2, U_3 sont des variables aléatoires indépendantes uniformes sur $[-1,1]$.

On propose les codes `Scilab` suivants :

```
// Methode d'estimation de Monte Carlo pour I_1
// avec conservation des intervalles de confiance pour toutes
// les valeurs entre 1 et M
// Cette fonction a vocation a etre utilisee pour visualiser
// graphiquement la stabilsation vers la moyenne
function res=IC_I_1(M)
  Alea=4*sqrt(1-rand(M,1).^2);
  X=cumsum(Alea);
  V=cumsum(Alea.^2);
  Abscisse=[1:M]';
  Moy=X./Abscisse;
  Factor=Abscisse./(max(Abscisse-1,1));
    V=V./Abscisse-Factor.*Moy.^2;
  Dev=1.96*sqrt(V)./sqrt(Abscisse);
  res=[Moy-Dev,Moy,Moy+Dev ];
endfunction

// On fait la meme en version breve, i.e. estimation de la moyenne empirique
// et de l'intervalle à 95%
// associé pour la valeur M en argument
function res=IC_I_1_Short(M)
  Alea=4*sqrt(1-rand(M,1).^2);
  X=sum(Alea);
  V=sum(Alea.^2);
  Moy=X/M;
  Factor=M/(M-1);
    V=V/M-Factor*Moy^2;
  Dev=1.96*sqrt(V)/sqrt(M);
  res=[Moy-Dev,Moy,Moy+Dev ];
endfunction

// Version brève pour l'estimation de I_2
function res=IC_I_2_Short(M)
```

```
  Alea=((-1+2*rand(M,2)).^2*ones(2,1)<=1);
  X=sum(Alea);
  V=sum(Alea); // Comme le retour aleatoire est 0 ou 1 on ne change pas en passant au carre
  Moy=X/M;
  Factor=M/(M-1);
    V=V/M-Factor*Moy^2;
  Dev=1.96*sqrt(V)/sqrt(M);
  res=4*[Moy-Dev,Moy,Moy+Dev ];
endfunction


// Version brève pour l'estimation de I_3
function res=IC_I_3_Short(M)
  Alea=((-1+2*rand(M,3)).^2*ones(3,1)<=1);
  X=sum(Alea);
  V=sum(Alea); // Comme le retour aleatoire est 0 ou 1 on ne change pas en passant au carre
  Moy=X/M;
  Factor=M/(M-1);
    V=V/M-Factor*Moy^2;
  Dev=1.96*sqrt(V)/sqrt(M);
  res=8*[Moy-Dev,Moy,Moy+Dev ];
endfunction
```

On obtient pour les sorties associées :

```
res1=IC_I_1_Short(1000000)
 res1  =   3.1411574    3.1429055    3.1446535
res2=IC_I_2_Short(1000000)
 res2  =   3.1390182    3.142236    3.1454538
res3= res3=IC_I_3_Short(1000000)
 res3  =   4.1810007    4.188832    4.1966633
```

où l'on rappelle que $4/3\pi = 4.1887902$.

On note que la simulation de 10^6 réalisations est instantanée mais nécessite l'appel à la commande `stacksize('max') ;` pour l'allocation mémoire dans `Scilab`.

7 Méthode du rejet

Nous l'avons illustrée en cours en rappelant comment simuler uniformément sur $A \subset D$, où A, D sont des boréliens de $\mathbb{R}^d$ dès lors que l'on sait simuler uniformément sur D. Précisément on a la proposition suivante.

Proposition 7.1 (Rejet uniforme) *Soit $A, D \in \mathcal{B}(\mathbb{R}^d)$ tels que $A \subset D$. Soit $(X_i)_{i\in\mathbb{N}^*}$ une suite i.i.d. de variables aléatoires définies sur un espace de probabilité $(\Omega, \mathcal{F}, \mathbb{P})$ et uniformes sur D. Introduisons $\tau := \inf\{i \in \mathbb{N}^* : X_i \in A\}$, alors X_τ est uniformément distribuée sur A, i.e. pour $B \in \mathcal{B}(\mathbb{R}^d),\ B \subset A,\ \mathbb{P}[X_\tau \in B] = \frac{|B|}{|A|}$. Le nombre de tirages avant l'obtention d'une réalisation de la loi uniforme sur A par ce procédé suit une loi géométrique $\mathcal{G}(p),\ p = \frac{|A|}{|D|}$. En particulier le nombre moyen de tirages nécessaires est $\frac{|D|}{|A|} \geq 1$.*

Preuve. Ecrivons :

$$\begin{aligned}
\mathbb{P}[X_\tau \in B] &= \mathbb{P}[X_\tau \in B, \bigcup_{i\in\mathbb{N}^*} \tau = i] = \sum_{i\in\mathbb{N}^*} \mathbb{P}[X_\tau \in B, \tau = i] = \sum_{i\in\mathbb{N}^*} \mathbb{P}[X_i \in B, \cap_{j=1}^{i-1} X_j \notin A] \\
&= \sum_{i\in\mathbb{N}^*} \mathbb{P}[X_i \in B] \prod_{j=1}^{i-1} \mathbb{P}[X_j \notin A] = \sum_{i\in\mathbb{N}^*} \frac{|B|}{|D|} \left(1 - \frac{|A|}{|D|}\right)^{i-1} = \frac{|B|}{|D|} \frac{1}{1 - \left(1 - \frac{|A|}{|D|}\right)} \\
&= \frac{|B|}{|A|},
\end{aligned}$$

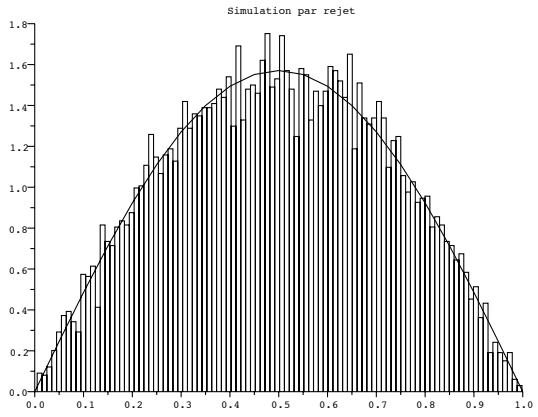
ce qui prouve la proposition. Notons que l'on a utilisé la convention $\cap_{j=1}^{0} X_j \notin A = \Omega$.

Cette méthode s'étend de façon naturelle comme précisé dans le document. Soit X une variable aléatoire réelle de loi μ_X qui possède une densité continue f_X à support dans $[a,b] \subset \mathbb{R}$ et telle que f_X soit incluse dans $[a,b] \times [0,M]$, $M \in \mathbb{R}^{+*}$. On considère une suite $(X_i)_{i\in\mathbb{N}^*}$ de variable aléatoires uniformes sur $[a,b] \times [0,M]$. On va tirer uniformément sur le carré $[a,b] \times [0,M]$ jusqu'à ce que l'une des réalisations se trouve sous le graphe de la densité, on renvoie ensuite l'abscisse associée à cette réalisation. Mathématiquement ce procédé se traduit de la façon suivante. Soit $\tau := \inf\{i \in \mathbb{N}^* : f(X_i^1) > X_i^2\}$, alors X_τ^1 a pour loi μ_X.

Pour le code proposé dans l'exercice :

```
function rejet(M)
X=zeros (1,M) ;
for i =1:M;
x=rand(1,2);
while x(2)> sin(%pi*x(1)),
x=rand(1,2);
end,
X(i)=x(1) ;
end ;
xbasc();
histplot(100,X) ;
Y=[ 0 : 0.05 : 1 ] ;
plot2d (Y,%pi*sin(%pi *Y) / 2 ) ;
endfunction
```

On simule des lois uniformes sur $[0,1]^2$ et l'on teste si la réalisation obtenue se trouve sous le graphe de la fonction $x \mapsto \sin(\pi x)$. Par la méthode du rejet la quantité obtenue en sortie de boucle `while` est reliée, à constante de normalisation près, à la réalisation d'une variable aléatoire de densité $\frac{\pi}{2}\sin(\pi x)\mathbb{I}_{x\in[0,1]}$. La constante de normalisation apparaît dans l'affichage par la fonction `plot2d`.



Nous formulons ci-après l'expression de l'algorithme de rejet le plus général. La preuve est laissée en exercice.

Proposition 7.2 (Rejet général) *Soit X une variable aléatoire réelle de loi μ_X qui possède une densité continue f_X à support dans $[a,b] \subset \mathbb{R}$ et telle que qu'il existe $C \geq 1$, $\forall x \in [a,b]$, $f(x) \leq Cg(x)$, $g(x) \geq 0$, $\int g(y)dy = 1$ et où l'on sait facilement simuler des lois de densité g. Soit $(Y_i)_{i\in\mathbb{N}^*}$ une suite i.i.d. de variables aléatoires de loi de densité g et $(U_i)_{i\in\mathbb{N}^*}$ une suite i.i.d. de variables aléatoires uniformes sur $[0,1]$ indépendantes de $(Y_i)_{i\in\mathbb{N}^*}$. Soit $\tau := \inf\{i \geq 1 : q(Y_i) := \frac{f(Y_i)}{cg(Y_i)} > U_i\}$, alors Y_τ a pour loi μ_X.*

8 Lois mélanges

$\rightsquigarrow$ On demande d'interpréter en terme de mélange de lois la densité $f(x) = (\frac{1}{3}\exp(-x) + \frac{4}{3}\exp(-2x))\mathbb{I}_{x>0}$.

On reconnaît aisément que $f(x) = \frac{1}{3}f_{\mathcal{E}(1)}(x) + \frac{2}{3}f_{\mathcal{E}(2)}(x)$ où l'on note pour $\lambda > 0$, $f_{\mathcal{E}(\lambda)}(x) = \lambda\exp(-\lambda x)\mathbb{I}_{x>0}$ la densité de la loi exponentielle de paramètre λ. Ce cas est un cas particulier du cas plus général suivant.

Proposition 8.1 (Simulation de mélanges de lois) *Soit μ une mesure de probabilité discrète de la forme $\mu = \sum_{n\in\mathbb{N}^*} \delta_n p_n$ (en particulier pour tout $n \in \mathbb{N}^*$, $p_n \geq 0, \sum_{n\in\mathbb{N}^*} p_n = 1$). Soit $(f_n)_{n\in\mathbb{N}^*}$ une suite de densités sur $\mathbb{R}^d$. La fonction $x \in \mathbb{R}^d \mapsto f(x) = \sum_{n\in\mathbb{N}^*} p_n f_n(x)$ définit bien une densité sur $\mathbb{R}^d$. Si $(X_n)_{n\in\mathbb{N}^*}$ est une suite de variables aléatoires indépendantes, telle que la loi de X_n a pour densité f_n et que τ est une variable aléatoire de loi μ indépendante de la suite $(X_n)_{n\in\mathbb{N}^*}$ alors la variable aléatoire X_τ a pour densité f.*

Preuve. L'idée consiste de nouveau à utiliser une partition sur les valeurs de τ. Soit $B \in \mathcal{B}(\mathbb{R}^d)$ on écrit :

$$\begin{aligned}\mathbb{P}[X_\tau \in B] &= \mathbb{P}[X_\tau \in B, \cup_{n\in\mathbb{N}^*}\tau = n] = \sum_{n\in\mathbb{N}^*} \mathbb{P}[X_\tau \in B, \tau = n] = \sum_{n\in\mathbb{N}^*} \mathbb{P}[X_\tau \in B|\tau = n]\mathbb{P}[\tau = n] \\ &= \sum_{n\in\mathbb{N}^*} \mathbb{P}[X_n \in B]\mathbb{P}[\tau = n] = \sum_{n\in\mathbb{N}^*} \int_B f_n(x)dx p_n = \int_B \left(\sum_{n\in\mathbb{N}^*} p_n f_n(x)\right) dx := \int_B f(x)dx,\end{aligned}$$

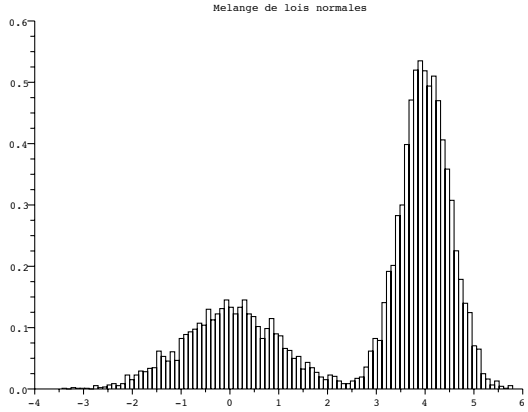
où l'on a utilisé l'indépendance de τ et $(X_n)_{n\in\mathbb{N}^*}$ pour la dernière égalité de la première ligne et le théorème de Fubini pour l'avant dernière égalité de la deuxième ligne (quantités positives).

Dans la pratique on va simuler une réalisation de τ puis simuler suivant la densité associée à la réalisation de τ obtenue. Dans le cadre de l'exemple on propose le code `Scilab` :

```
// Simulation de lois mélange a partir de lois exponentielles
function res=MELANGE_1(M)
  Alea1= rand(M,1)>1/3; // vecteur boolen associe a la realisation des tau
                        // resultat est T si le tirage est superieur a 1/3
  res=zeros(M,1);
  for i=1:M;
    res(i)=grand(1,1,'exp',1./(1+Alea1(i)));
  end
endfunction

// Simulation de lois mélange a partir de lois normales
function res=MELANGE_2(M)
  Alea1= rand(M,1)>1/3;
  res=zeros(M,1);
  for i=1:M;
    res(i)=grand(1,1,'nor',4*Alea1(i),1./(1+Alea1(i)));
  end
endfunction
```

Nous illustrons ci-après le résultat de la deuxième fonction, qui représente l'histogramme de la réalisation de 10000 réalisations de la variable aléatoire de densité $f(x) = \frac{1}{3}f_{\mathcal{N}(0,1)}(x) + \frac{2}{3}f_{\mathcal{N}(4,1/4)}(x)$, où $f_{\mathcal{N}(m,\sigma^2)}(x) = \frac{1}{(2\pi)^{1/2}\sigma}\exp\left(-\frac{|x-m|^2}{2\sigma^2}\right)$ désigne la densité de la loi normale de paramètres (m, σ^2) au point x.



$\rightsquigarrow$ Pour le deuxième exemple $f(x) = \frac{1}{2}\exp(-|x|)$ on reconnaît la loi exponentielle symétrisée. Une façon de la réaliser par mélange consiste à écrire $f(x) = \frac{1}{2}(f_{\mathcal{E}(1)}(x)+f_{\mathcal{E}(1)}(-x))$. Dans la pratique il suffit de tirer le signe en suivant une loi de Bernoulli de paramètre $1/2$ et de multiplier par le résultat obtenu une réalisation de variable exponentielle de paramètre 1.

```
// Simulation de la loi exponentielle symetrique
function res=expo_sym(M)
  res= grand(1,M,'exp',1).*(-1+2*(rand(1,M)>1/2 ) )
endfunction
```

$\rightsquigarrow$ On écrit désormais une fonction qui simule la réalisation d'une loi de Poisson $N \sim \mathcal{P}(1)$ et qui rend la somme $\sum_{i=1}^{N} B_i$ où les $(B_i)_{i\in[\![1,N]\!]}$ sont de lois de Bernoulli indépendantes et indépendantes de N (avec la convention $\sum_{i=1}^{0} B_i = 0$).

```
// Simulation de la somme de N lois de Bernoulli independantes de parametre 1/2 ou N est une
// loi de Poisson de parametre 1 independante des Bernoulli
function res=Sim_Poisson_Bernoulli(M)
  AleaPoisson=grand(1,M,'poi',1);
  res=zeros(1,M)
  for i=1:M;
     res(i)=sum (rand(1,AleaPoisson(i))>1/2  );
  end
endfunction
```

Pour caractériser la loi de S on remarque que celle-ci est à valeurs dans $\mathbb{N}$. Ainsi pour $j \in \mathbb{N}$ on écrit :

$$\begin{aligned}\mathbb{P}[S=j] &= \mathbb{P}[\sum_{i=1}^{N} B_i = j] = \mathbb{P}[\sum_{i=1}^{N} B_i = j, N \geq j] = \sum_{n\geq j} \mathbb{P}[\sum_{i=1}^{N} B_i = j | N=n]\mathbb{P}[N=n] \\ &= \sum_{n\geq j} \mathbb{P}[\sum_{i=1}^{n} B_i = j]\frac{\exp(-1)}{n!},\end{aligned}$$

où l'on a de nouveau utilisé l'indépendance entre N et les $(B_i)_{i\in\mathbb{N}^*}$ pour la dernière égalité. En rappelant que la somme de n lois de Bernoulli indépendantes de paramètre p est une loi binomiale de paramètres (n,p) il vient :

$$\begin{aligned}\mathbb{P}[S=j] &= \sum_{n\geq j} C_n^j \left(\frac{1}{2}\right)^n \frac{\exp(-1)}{n!} = \exp(-1)\sum_{n\geq j}\frac{(1/2)^n}{(n-j)!j!} = \frac{\exp(-1)(1/2)^j}{j!}\sum_{n\geq j}\frac{(1/2)^{n-j}}{(n-j)!} = \frac{\exp(-1/2)}{j!}(1/2)^j \\ &= \mathbb{P}[\mathcal{P}(1/2)=j].\end{aligned}$$

Nous avons ainsi mis en évidence que $S \sim \mathcal{P}(1/2)$. Une façon d'illustrer ce résultat peut être de comparer les histogrammes empiriques associés aux réalisations de la fonction précédente et de la simulation directe de loi de Poisson de paramètre $1/2$.

9 Théorème de Wigner

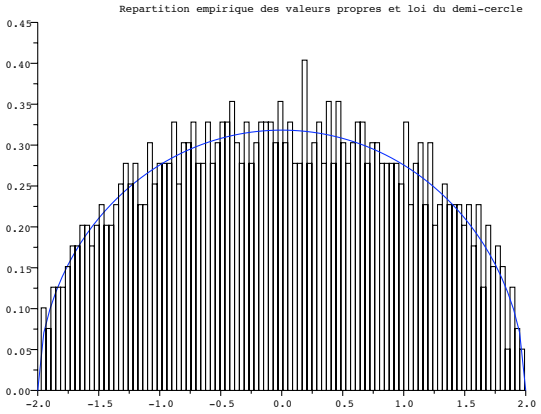
On traite cette section principalement pour donner quelques commandes relatives aux manipulations matricielles tout en illustrant numériquement un résultat profond. Le théorème indiqué dans l'énoncé est hors programme pour l'agrégation. On renvoie le lecteur intéressé à la Section 2.10 de [BC07]

Les fonctions `tril(M), tril(M,-1)` renvoient respectivement la partie sous-diagonale "large" de la matrice `M`, i.e. incluant la diagonale pour `tril(M)`, et la sous-diagonale stricte de `M`, i.e. excluant la diagonale. L'appel à la fonction `spec(M)` renvoie le spectre de la matrice `M` en argument. Pour illustrer le théorème de Wigner on propose le code `Scilab` suivant.

```
// On va simuler la realisation des valeurs propres de Matrice
// aleatoires symetriques a entrees Gaussiennes
function res=Wigner(N)
  Mat=grand(N,N,'nor',0,1);
  L=tril(Mat,-1); U=(tril(Mat))';
  G=1./sqrt(N)*(L+U);
```

```
  res=spec(G);
endfunction

res=Wigner(1000);
plot([-2:.05:2],1./(2*%pi)*sqrt(4-[-2:.05:2].^2))
xtitle('Repartition empirique des valeurs propres et loi du demi-cercle')
```



Références

[BC07] B. Bercu and D. Chafaï. *Modélisation Stochastique et Simulation.* Dunod, 2007.

[Ouv00] J.Y. Ouvrard. *Probabilités.* Cassini, 2000.

[Shi96] A.N. Shiryaev. *Probability, Second Edition.* Graduate Texts in Mathematics, 95. Springer-Verlag, New York., 1996.