TP 0

Initiation à Matlab

Ce qui est dit dans ce chapitre concerne Matlab 6[1], mais reste en grande partie valable pour Matlab 5 et les logiciels Octave 2.1 et Scilab 2.6. De façon générale, chaque bout de code « Matlab » figurant dans ce document fonctionne avec au moins l'un de ces logiciels, la plupart du temps avec Matlab, et parfois avec tous.

Matlab est un logiciel commercial, et je vous encourage à jeter un œil au logiciel libre Octave. Scilab est disponible le jour de l'oral. Pour savoir ce qu'est exactement un logiciel libre, lisez la page w3 suivante :

`http://www.fsf.org/philosophy/free-sw.fr.html`

Il serait temps que les responsables de l'organisation de l'épreuve orale fassent en sorte que Octave (`http://www.octave.org/`) soit disponible le jour de l'épreuve. C'est un véritable logiciel libre, et donc en parfaite adéquation avec l'idée que l'on peut se faire de l'école publique et de l'éducation nationale. En tant que logiciel libre, il est gratuit, ce qui permet une certaine égalité d'accès à cet outil. Scilab est un beau projet mais ressemble moins à Matlab que Octave, et sa licence d'utilisation n'est pas aussi libre que celle d'Octave.

Matlab est un logiciel commercial de calcul matriciel développé par la société MathWorks (`http://www.mathworks.com/`). Son nom est la contraction de « Matrix Laboratory ». Il consiste essentiellement en un *interpréteur de commandes*, écrites dans un langage de programmation spécifique appelé langage Matlab. Les commandes Matlab sont saisies et interprétées ligne à ligne dans une fenêtre (console). Comme nous le verrons plus loin, elles peuvent également être regroupées dans un fichier dont le nom se termine par `.m`. Les versions récentes de Matlab comportent un éditeur de tels fichiers, qui permet d'exécuter les commandes pas à pas (débogage).

Les variables Matlab sont toutes des *tableaux*, « arrays » en anglais, définies au moment de leur affectation. Il n'y a donc pas besoin de les déclarer. Les matrices et tenseurs sont des tableaux particuliers. Un nombre complexe (ou réel ou entier) est un tableau de taille 1×1. Le langage Matlab a été conçu pour faciliter les opérations sur les matrices.

Le langage Matlab permet de manipuler des données de différents types dont certains sont imbriqués : entiers, nombres réels, nombres complexes, caractères, booléens. Ces données peuvent êtres assemblées en tableaux de différents types imbriqués : vecteurs, matrices, tenseurs (matrices à plus de 2 dimensions), tableaux cellulaires, tableaux structurés. Les chaînes de caractères sont des vecteurs de caractères.

Les tableaux cellulaires et structurés ne sont pas abordés dans cette introduction. Le lecteur curieux pourra faire `help lists`, `help cell` et `help struct` pour obtenir de l'aide sur ces aspects.

Ce chapitre a été écrit dans le soucis d'être à la fois intuitif et peu rébarbatif. Il peut donc manquer de rigueur pour un lecteur familier avec les présentations rationnalisées des langages

[1] Testé avec Matlab 6.1.0.450 Release 12.1 sous Irix sur la machine `ondine.cict.fr` du Centre Interuniversitaire de Calcul de Toulouse.

de programmation.

0.1 M-a-Ma... Matlab

En Matlab, les nombres réels sont représentés en virgule flottante avec 52 chiffres significatifs en base 2, soit un peu moins de 16 en base 10 (cf. `eps`). Les opérations élémentaires sur les nombres réels sont :

`+` `-` `*` `/` `\` `^`

qui représentent respectivement l'addition, la soustraction, la multiplication, la division à gauche et la division à droite, et enfin l'élévation à une puissance. Ces opérations s'appliquent également aux matrices, et dans ce cas, \ et `/` sont différentes.

Comme nous allons le voir, Matlab différentie minuscules et majuscules dans les noms de variables. L'affectation est notée `=`. La table 0.1 donne quelques variables spéciales de Matlab, qui sont toujours définies.

`[]`	matrice vide
`ans`	dernière évaluation effectuée par Matlab
`computer`	type de machine (cf. aussi `ispc` et `isunix`)
`cputime`	temps CPU depuis le lancement de Matlab
`eps`	précision des flottants
`i` ou `j`	$\exp(i\pi/2) = \sqrt{-1}$
`Inf`	infini positif
`NaN`	pas-un-nombre (Not a Number)
`pi`	valeur de π
`realmax`	plus grand nombre réel flottant positif
`realmin`	plus petit nombre réel flottant positif

TAB. 1 – Quelques variables spéciales de Matlab.

Matlab effectue ses calculs dans l'ensemble des tableaux à plusieurs dimensions, et donc en particulier les matrices carrées ou non, et les vecteurs ligne et colonne. Les opérations sur ces tableaux ne sont faites que lorsqu'elles ont un sens bien entendu. En général, Matlab donne un sens assez intuitif aux opérations entre matrices. Cela dit, son langage contient de nombreuses spécificités que nous allons apprendre à utiliser.

Voici un exemple de code Matlab, à saisir ligne par ligne dans la fenêtre de commandes. Vous pouvez vous dispenser de saisir les lignes précédées du caractère `%` car ce sont des commentaires, et sont donc ignorées par l'interpréteur de Matlab.

```
% Cette ligne est un commentaire car elle est préfixée par le caractère %
% On crée une variable réelle nommée a, initialisée à 0
% Matlab affiche sa valeur une fois que l'on a appuyé sur la touche entrée
a=0
% En suffixant par un point-virgule, on évite l'affichage de la valeur de a
a=0;
% Pour connaître le contenu d'une variable, il suffit d'invoquer son nom
a
% Matlab différencie majuscules et minuscules.
% Ainsi, on peut créer la variables A, différente de a
A=1
% Vous pouvez rappeler les commandes précédement executées au moyen
% des touches fléchées de votre clavier (haut et bas).
%
```

```
% Voici un calcul compliqué à base des variables a et A précédentes.
A*a+cos(a)/(1+sqrt(1+A^2)) % sqrt est la racine carrée (« square root »)
% La variable spéciale ans contient la dernière réponse de Matlab
% qui n'a pas été affectée à une variable par le symbole =
ans
% On peut lister les variables actuellement définies avec la commande whos
% On voit que pour Matlab, les variables a et A sont des matrices 1 x 1
whos
% Pour une liste plus succincte, on peut utilier la commande who
who
% On peut détruire une variable au moyen de la commande clear
clear a          % destruction de la variable a
% Vérifions que la variable a n'existe plus
who
% On peut aussi détruire toutes les variables comme suit
clear
% Vérification
who
% Pour obtenir de l'aide (en anglais !) sur une commande, on peut
% utiliser la commande help
help who
% La commande helpwin permet d'obtenir une liste
% des commandes Matlab classée par thème.
helpwin
% Valeurs particulières que l'on est amené à rencontrer
1/0   % Infini positif Inf
-1/0  % Infini négatif -Inf
0/0   % Not a Number (NaN)
0*Inf % Not a Number (NaN)
1/Inf % Donne bien zéro
pi       % Donne bien le nombre pi
help pi % Donne sa définition pour Matlab
i        % Racine carrée (complexe) de -1.
help i  % Sa définition pour Matlab.
        % On peut quand même utiliser i comme variable...
```

Remarque 0.1.1 *(À l'aide !).* La commande `help` permet d'obtenir de l'aide sur les commandes Matlab, `help help` donne de l'aide sur l'aide ! Enfin, la commande `lookfor` permet de lister les commandes Matlab par mots clés.

Remarque 0.1.2 *(Affichages).* Si `X` est un tableau :

- la commande `disp(X)` affiche son contenu sans afficher son nom. S'il est vide, rien n'est affiché. La variable `X` vide est symbolisée par `[]`, on peut tester si `X` est vide grâce à la commande `isempty(X)`.
- la commande `display(X)` affichera le nom de la variable (`X` ici) ainsi que son contenu, même s'il est vide. Cette commande est utilisée automatiquement par Matlab lorsque qu'une expression ne se termine pas par un point-virgule.

```
% Une matrice à deux lignes et trois colonnes en spécifiant ses entrées
% Les lignes sont séparées par des points virgules
% Les entrées d'une ligne sont séparées par des virgules
A=[2,4,8;3,9,27]
% On peut accéder à une entrée particulière de la matrice A comme suit
A(2,3)
% b est un vecteur ligne contenant la deuxième ligne de A
b=A(2,:)
% c est un vecteur colonne contenant la troisième colonne de A
```

```
c=A(:,3)
% B est la sous-matrice 2x2 principale de A
B=A(1:2,1:2)
% Pour afficher à nouveau le contenu de la matrice A
A
% U est un vecteur ligne contenant les entiers de 1 à 20
U=[1:20]
% V est un vecteur ligne contenant les entiers de 1 à 20 par incrément de 2
V=[1:2:20]
% Devrait donner un vecteur ligne nul
V-U(1:2:20)
% W est un vecteur contenant les réels de Pi/2 à Pi par incrément de 0.1
W=[pi/2:.1:pi]
```

Les commandes comme `sqrt` qui prennent un ou plusieurs arguments (ou paramètres) entre parenthèses constituent des *fonctions*. Nous verrons plus loin comment en créer de nouvelles.

Exercice 0.1.3. À quoi peuvent servir les fonctions `fix`, `round`, `floor`, `ceil`, `mod` et `rem`? Idem avec les commandes `imag`, `conj`, `angle` et `abs`?

0.2 Calculs matriciels élémentaires

```
A=[3,4;1,0] % Matrice 2x2
B=(A>0)     % B = matrice tq Bij=1 si Aij>0 et 0 sinon
V=[5;5]   % Vecteur colonne de dimension 2, s'écrit aussi [5,5]'
length(V) % Renvoie la longueur du vecteur V
size(A)   % La fonction size renvoie la taille
%
%% Opérations matricielles classiques
%
B=A' % B = transposée de A
A*B  % Multiplication matricielle de A et de B
A+B  % Addition matricielle de A et de B
A^99 % Puissance matricielle
expm(A)  % Exponentielle matricielle
inv(A)   % Matrice inverse
sqrtm(A) % Racine carrée matricielle cf. help sqrtm
logm(A)  % Logarithme matriciel. cf. help logm
A*V  % Image du vecteur colonne V par la matrice carrée A
V'*A  % Devinez...
A\V  % Solution du système linéaire AX=V (par pivot, et pas par inv(A))
help slash  % Aide explicative sur la division matricielle précédente
%
%% Opérations entrée par entrée
%
C=A.*B    % C(i,j)=A(i,j)*B(i,j)
C=A./B    % C(i,j)=A(i,j)/B(i,j)
C=A.^3    % C(i,j)=A(i,j)^3
C=3+A     % C(i,j)=3+A(i,j)
C=3*A     % C(i,j)=3*A(i,j)
C=A./3    % C(i,j)=A(i,j)/3
C=cos(A)  % C(i,j)=cos(A(i,j))
C=log(M)  % C(i,j)=log(A(i,j)).  Ne pas confondre avec logm(A) !
C=sqrt(M) % C(i,j)=sqrt(A(i,j)). Ne pas confondre avec sqrtm(A) !
C=exp(A)  % C(i,j)=exp(A(i,j)).  Ne pas confondre avec expm(A) !
C=abs(A)  % C(i,j)=|A(i,j)|
```

© 2001 Dj. Chafaï. GFDL copyleft. Version djc-amops-germinal du 2001-12-05, compilée par LaTeX 2ε le 5 décembre 2001 à 19:56.

```
% Notez qu'à chaque fois, la variable C est redéfinie,
% sans que cela pose problème.
% Listons les variables actuellement définies
whos
% Consultons l'aide sur les opérations élémentaires
help ops
% et sur les opérations élémentaires matricielles
help elmat
%
% Pour finir...
%
n=4;
A=ones(n,n)  % Matrice de taille 4x4 dont tous les éléments valent 1
B=zeros(n,n) % Matrice nulle de taille 4x4
C=eye(n,n)   % Matrice identité de taille 4x4. Que donne eye(3,7) ?
J=[A,B,C]    % J = juxtaposition horizontale des matrices A, B et C
K=[A;B;C]    % K = juxtaposition verticale des matrices A, B et C
N=0*J        % Astuce pour obtenir une matrice N nulle de même taille que J
             % sans connaitre la taille de J.
% Création d'une matrice M 4x3 « creuse » à partir d'un vecteur de données
% [0.5,0.1,0.2], et de vecteurs de position des données [1,2,4] et [2,1,3]
% Donc M(i,j)=0 sauf M(1,2)=0.5, M(2,1)=0.1 et M(4,3)=0.2
M = sparse( [1,2,4], [2,1,3], [0.5,0.1,0.2], 4, 3)
```

On a donc les opérations supplémentaire suivantes sur les matrices :

`.*` `./` `.\` `.^`

qui opèrent entrée par entrée. Vous pouvez faire `help arith` pour de l'aide sur les opérations arithmétiques et `help @` pour de l'aide sur les opérateurs en général et les caractères spéciaux. Comme on vient de le voir, les sous matrices s'obtiennent en spécifiant des intervalles d'indices. Si i, j, k sont des entiers relatifs, alors :

1. `i:j` est identique à $i, i+1, \ldots, j$
2. `i:j:k` est identique à $i, i+k, i+2k, , \ldots, j$

L'intervalle vide est représenté par la matrice vide `[]`. Une ligne ou une colonne entière peut être obtenue en utilisant le caractère `:` seul. Enfin, une matrice d'entiers `E` peut servir à spécifier les indices d'une matrice `A`, en écrivant `A(E)`. Il ne faut pas confondre les expressions de la forme `[... ]`, qui permettent de fabriquer des matrices, avec celles de la forme `M(... )`, qui permettent de considérer une sous-matrice de la matrice `M`.

Exercice 0.2.1. À quoi peuvent servir les fonctions `fliplr`, `flipud` et `sort` ? Dites-vous bien que `lr`=« left,right » et que `ud`=« up,down ». Vous connaissez à présent `ones`, `zeros` et `eye`, à quoi correspondent `vander`, `toeplitz`, `pascal`, `magic`, `kron`, `hadamard` et `hilbert` ?

0.3 Calculs matriciels plus élaborés

```
M=ones(3,3)+eye(3,3)
V=[1:10]
det(M)    % Déterminant de M
rank(M)   % Rang de la matrice
trace(M)  % Trace de M
D=eig(M)  % Renvoie le vecteur colonne des valeurs propres de M
[P,D]=eig(M) % Diagonalise M = P*D*P^-1 avec D diagonale
null(M)  % Renvoie une BON du noyau (« null space » en anglais)
tril(M)     % Partie triangulaire inférieure (lower) de M
```

```
tril(M,-1) % Idem sans la diagonale, cf. help tril
triu(M)    % Partie triangulaire supérieure (upper) de M
triu(M,+1) % Idem sans la diagonale, cf. help triu
diag(M)    % Diagonale de M, dans un vecteur colonne
diag(V)    % Matrice diagonale dont la diagonale est le vecteur V
blkdiag(eye(3,3),M) % Matrice block diagonale de blocks eye(3,3) et M
%
sum(V)   % Somme des éléments du vecteur V
sum(M)   % Somme des colonnes de la matrice M (renvoie un vecteur ligne)
sum(M,2) % Somme les éléments de la matrice M selon la dimension 2
sum(sum(M)) % Somme totale des éléments de la matrice M
cumsum(V)   % Sommes cumulatives des entrées du vecteur V
cumsum(M)   % Matrice des sommes cumulatives des colonnes de M
cumprod(V)  % Produits cumulatifs des entrées du vecteur V
cumprod(M)  % Matrice des produits cumulatifs des colonnes de M
max(M)      % Renvoie un vect. ligne = au max sur chaque col. de M
max(max(M)) % Maximum des entrée de la matrice M. Que donne min ?
range(V)    % Renvoie l'étendue de V, c'est-à-dire max(V)-min(V)
%% Réplication de matrice par blocs
repmat(M,3,4) % Matrice par block 3x4 où chaque block = M
              % M n'a pas besoin d'être carrée.
%% Redimentionnement de matrice, colonne par colonne
reshape([1,2,3,4],2,2) % Donne la matrice 2x2 [1,3;2,4]
reshape([1,2,3;4,5,6],3,2) % Donne la matrice 3x2 [1,5;4,3;2,6]
%% Aide sur les opérations matricielles élémentaires
%% et sur les fonctions matricielles
helpwin elmat
helpwin matfun
```

Exercice 0.3.1. À quoi peuvent servir les fonctions `svd`, `norm` et `chol` ?

Remarque 0.3.2. La fonction Matlab `funm` permet d'évaluer une fonction sur une matrice. Vous pouvez l'ignorer en première lecture (et même en seconde). Elle nécessite la connaissance de la notion de descripteur de fonction (*function handle* en anglais). Cf. `help functions`, `help function_handle`, `help str2func` et `help func2str`.

Exercice 0.3.3. À quoi peuvent servir les fonctions `expm1`, `expm2`, `expm3` ?

0.4 Générateurs aléatoires, graphiques

```
rand  % Renvoie un nombre pseudo-aléatoire en 0 et 1 selon la loi uniforme
randn % Renvoie un nombre pseudo-aléatoire tiré selon la loi normale
% Une matrice pseudo-aléatoire 4x4 dont les entrées sont
% des réalisations pseudo-indépendantes de loi uniforme sur [0,1]
rand(4,4)
% Idem mais avec la loi normale
randn(4,4)
% 1000 réalisations pseudo-uniformes et pseudo-indépendantes
Y=randn(1000,1); % Y est un vecteur colonne ici.
mean(Y)    % Moyenne arithmétique = ( Y(1)+...+Y(1000) ) / 1000
median(Y)  % Médiane
std(Y)     % Écart type normalisé en N-1
plot(Y)        % Tracé du vecteur Y. Abscisses ? Ordonnées ?
plot(Y,'r-')   % Autre tracé avec couleur et type de ligne
X=[1:2:2*length(Y)];
plot(X,Y,'r-') % Vous comprenez...
```

© 2001 Dj. Chafaï. GFDL copyleft. Version djc-amops-germinal du 2001-12-05, compilée par LaTeX 2ε le 5 décembre 2001 à 19:56.

```
hist(Y)          % Histogramme (10 classes par défaut, cf. help hist)
[Effectifs,Classes]=hist(Y,50)  % Histogramme à 50 classes de Matlab
                                % Classes = centres des 50 classes
                                % L'histogramme n'est pas tracé.
[Effectifs,Classes]=histo(Y,50) % Histogramme à 50 classes de Stixbox
                                % Classes = 51 extrémités des classes
                                % L'histogramme est tracé.

%
% Tracé des moyennes empiriques successives, converge vers 0 d'après la LGN
title('Loi des grands nombres')                  % Titre de la figure
xlabel('Nombre de réalisations iid normales')    % Titre des abscisses
ylabel('Moyenne empirique')                      % Titre des ordonnées
legend('Legende 1')                              % Légende
plot(cumsum(Y)'./[1:length(Y)],'b-') % Tracé des moyennes empiriques
%
%
A=eye(100,100)+rand(100,100);
figure                  % Création d'une nouvelle fenêtre graphique
title('Ma matrice')     % Titre de la figure
xlabel('Dimension 1')   % Titre des abscisses
ylabel('Dimension 2')   % Titre des ordonnées
imagesc(A);             % Trace la matrice, couleurs = valeurs relatives,
hold off                % On ne va pas utiliser la même fenêtre graphique
contour(A);             % Trace les lignes de niveau
```

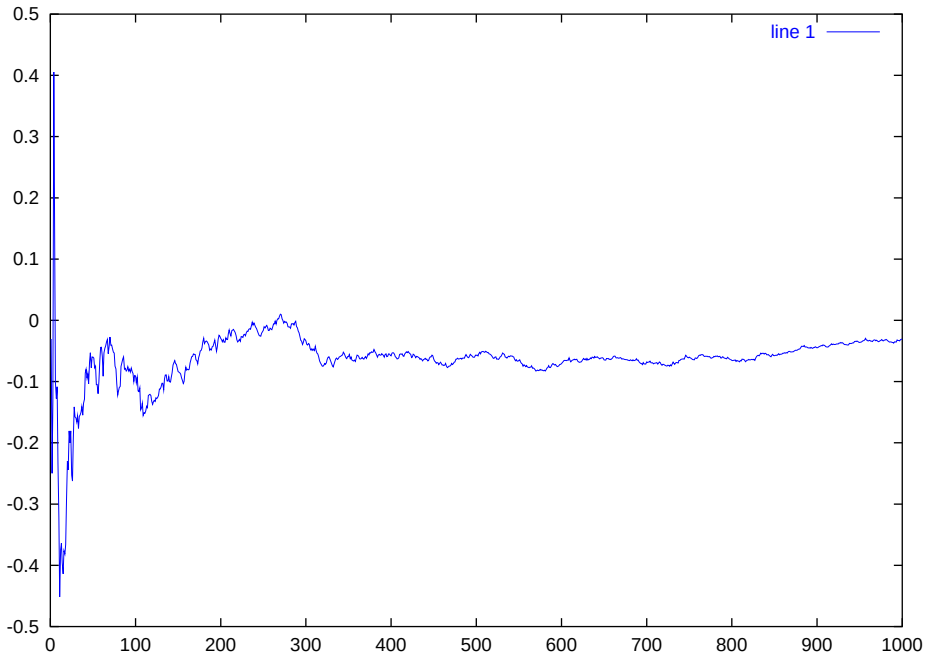

FIG. 1 – Ce que donne `plot(cumsum(randn(1,1000))./[1:1000],'b-')`.

Exercice 0.4.1. À quoi peuvent servir les fonctions `mesh`, `bar` et `stairs` ?

Exercice 0.4.2 *(Sous-graphiques).* Apprenez à vous servir de la commande `subplot` qui permet de tracer des sous-graphiques.

Exercice 0.4.3. Étudiez tous les arguments des fonctions `histo` et `hist`.

Exercice 0.4.4 *(Tracés de graphiques faciles ?).* À quoi peuvent servir les fonctions `ezplot`, `ezcontour`, `ezcontourf`, `ezmesh` `ezmshrc`, `ezplot3`, `ezpolar`, `ezsurf`, et `ezsurfc` ?

Exercice 0.4.5. Les fonctions `sprand` et `sprandn` permettent de générer des matrices creuses aléatoires selon la loi uniforme ou normale. Elles sont en quelque sorte obtenues par croisement de `sparse`, `rand` et `randn`. À quoi peuvent servir les fonctions `speye`, `spones` ?

La bibliothèque Stixbox fournit quelques fonctions supplémentaires utiles pour faire de la simulation. Nous les utiliserons au fur et à mesure de nos avancées, tout au long de l'année. Vous en trouverez la liste page 113. Nous n'utiliserons pas la boite à outils de statistique de Matlab, qui est pourtant bien plus complète, car elle n'est pas disponible le jour de l'épreuve. Vous entrouverez un descriptif 114.

0.5 Chaînes de caractères

En Matlab, les caractères constituent un type de données, au même titre que les nombres réels. Ainsi, `'1'` représente le caractère « 1 », qui n'est pas de même nature que le nombre 1. Une chaîne de caractères est un vecteur dont les entrées sont des caractères. Une présentation de la gestion des chaînes de caractère sous Matlab est donnée en faisant `help strings`. Le code Matlab :

```
1 S='Ceci est une chaîne'
```

définit une variable `S` de type chaîne de caractères, qui n'est rien d'autre qu'un vecteur de caractères. En tant que vecteur, `S(1)` vaut `'C'` et `S(4)` vaut `'i'`, tandisque `length(S)` vaut 19. La commande `ischar` permet de tester si une expression est du type chaîne de caractères. Ainsi, `ischar('1.32')` renvoie `1`, alors que `ischar(1.32)` renvoie `0`. Les chaînes de caractères étant des vecteurs, elles sont concaténables, ainsi, le code Matlab

```
2 ['Ceci est ','une chaîne']
```

donnera

```
3 'Ceci est une chaîne'
```

On peut également définir sans difficulté des matrices de caractères.

Il est possible de convertir un réel en une chaîne de caractères qui représente son écriture en base 10. Ainsi `num2str(1.32)` renvoie la chaîne de caractères `'1.32'`. L'opération inverse peut être obtenue par la commande `str2num`. La commande `dec2base` permet d'obtenir la chaîne de caractères qui représente l'écriture dans une base de numération d'un entier. Ainsi `dec2base(11,3)` renvoie la chaîne `'102'`, qui est l'écriture en base 3 du nombre 11. Les commandes `dec2bin` et `dec2hex` ont des racourcis pour les base 2 et 16 respectivement. La commande `base2dec` permet d'effectuer l'opération inverse : `dec2base('102',3)` renvoie `11`, qui n'est pas une chaîne mais un entier donc un nombre.

La commande `char` permet d'obtenir une chaîne de caractères à partir d'un vecteur de codes ASCII. Ainsi, `char([65,66,67])` renvoie `'ABC'`. Les codes supérieurs ou égaux à 127 donnerons des caractères qui dépendent de la page de code du système.

Les commandes `lower` et `upper` permettent de convertir la chaîne de caractères passée en argument en minuscules et en majuscules respectivement.

La commande `sprintf` permet de construire des chaînes de caractères à partir de variables de différents types, cf. `help sprintf`. Réciproquement, la commande `sscanf` permet de décomposer une chaîne de caractères en plusieurs variables de différents type, cf. `help sscanf`. Ces deux commandes généralisent les commandes `num2str` et `str2num` respectivement. Elles sont très pratiques. Voici un exemple explicatif :

```
%%% Illustration de l'utilisation de sprintf.
clear;
r=1.23;
n=5;
s='blabla';
chaine=sprintf('r vaut %f, n vaut %d et s vaut %s',r,n,s);
disp(chaine)
%%% Illustration de l'utilisation de sscanf.
clear;
chaine='1.2 : 5.5 : 3.8 ; 51';
X=sscanf(chaine,'%f : %f : %f ; %d ; %s');
```

0.6 Entrées et sorties

La commande **input** permet de demander à l'utilisateur de saisir des valeurs de variables. La commande **pause** permet de stopper l'exécution de Matlab pendant un temps déterminé, cf. **pause**. La forme spéciale **pause off** désactive les pauses tandisque **pause on** les réactive.

La commande **save** permet de sauvegarder le contenu des variables en cours ainsi que leur nom dans un fichier, dont le nom est par défaut matlab.mat. Ce fichier peut être lu par la commande **load**, qui restaure donc toutes les variables.

```
% La commande input permet d'afficher une question et de récupérer la
% réponse dans une variable
n=input('Donnez la valeur de n : ')
%
% La commande sprintf permet de faire un affichage formaté de chaînes
% de caractères comprenant des variables. Faire help sprintf.
%
entier=1;
reel=pi;
sprintf('Voici un entier %i et un réel %12.8f\n',entier,reel)
%
%% Sauvegarde et lecture des variables dans des fichiers
%
% La commande save permet de sauvegarder une variable dans un fichier
A=rand(1000,10);
save 'mata' A; % La matrice A est sauvegardée dans le fichier mata.dat
% La commande load permet de lire un fichier de données
clear A        % On détruit A
load 'mata'    % On lit le fichier mata.dat
whos           % A apparaît de nouveau
% Pour sauvegarder A sous forme de fichier texte
% (le nom de la variable est perdu)
save -ASCII 'mata.txt' A
% Pour lire la matrice dans un fichier texte et l'affecter à une variable
B=load('mata.txt')
% Pour sauvegarder toutes les variables en cours
save   % Sauvegarde toutes les variables dans le fichier matlab.mat
load   % Pour restaurer les variables sauvegardées
```

Les commandes **fprint** et **fscanf** permettent de faire des lectures et écritures formatées vers des fichiers de données :

```
%%% Exemple d'entrées sorties sur un fichier
```

```
clear;

% Une matrice
position=rand(1,10);
vitesse=rand(1,10);

% Ouverture du fichier en mode réécriture (w+)
% Le contenu précédent est détruit
id=fopen('donnees.txt','w+');
% Ecriture des variables dedans
fprintf(id,'Position = %f, vitesse = %f\n',position,vitesse);
% Fermeture du fichier
fclose(id);

% Jetez un coup d'oeil au fichier donnees.txt !

% Ouverture du fichier en mode lecture
id=fopen('donnees.txt','r');
% Lecture de 10x10 éléments
PV=fscanf(id,'Position = %f, vitesse = %f\n',[10 10]);
% Fermeture
fclose(id);

% Comparer PV et [position',vitesse']
```

0.7 Opérations logiques, boucles et exécutions conditionnelles

Pour Matlab, tout nombre peut être considéré comme une valeur logique, en identifiant les nombres non nuls à *vrai* (noté `1`) et le nombre zéro à *faux* (noté `0`). La commande `boolean` permet d'obtenir le booléen associé à un nombre vu comme une valeur logique. Les principales opérations sur les valeurs logiques sont :

- la négation logique `~`
- le « ou » logique `|`
- le « et » logique `&`

On peut également utiliser les respectivement les fonctions `or`, `not` et `and`. Il y en a d'autre, faites donc un `help ops`. On peut créer des valeurs logiques à partir de nombres avec les opérations de comparaisons, qui activent automatiquement la nature logique de leurs arguments :

- « `a` égal à `b` » s'écrit : `a == b`
- « `a` différent de `b` » s'écrit : `a ~= b`
- « `a` supérieur strictement à `b` » s'écrit : `a > b`
- « `a` supérieur ou égal à `b` » s'écrit : `a >= b`
- « `a` inférieur strictement à `b` » s'écrit : `a < b`
- « `a` inférieur ou égal à `b` » s'écrit : `a <= b`

On peut également utiliser les respectivement les fonctions `eq`, `ne`, `gt`, `ge`, `lt` et `leq`. Le code Matlab `R=randn(5,5); M=(R>2.1)` crée une matrice `M` de même taille que `R` qui contient des `0` là où `R` est inférieure à 2.1 et des `1` là ou `R` est supérieure ou égale à 2.1. C'est donc une matrice « booléenne ». La commande `any(V)` renvoie `1` si au moins l'un des éléments du vecteur `V` est non nul, et `0` sinon. La fonction `all(V)` renvoie `1` si tous les éléments du vecteur `V` sont non nuls, et `0` sinon. Enfin la fonction `find(V)` renvoie les indices correspondants aux éléments non nul du vecteur `V`. Ainsi, `V(find(V>2))` renvoie les valeurs supérieures à 2 du vecteur `V`.

Les commandes Matlab `isreal`, `ischar`, `issparse`, `isnan`, `isinf`, `isfinite` permettent de tester la nature des variables. Ainsi, `isreal(a)` renverra `1` si la variable `a` est un réel et `0` sinon.

```
%%% Exemples de calculs logiques
V=[1,0,-3,3.4];     % Un vecteur de nombres
boolean(V)          % Doit donner [1,0,1,1]
~V                  % Doit donner [0,1,0,0]
ischar(V)           % Doir donner 0
isreal(V)           % Doit donner 1
isfinite(V)         % Doit donner [1,1,1,1]
isinf(1./V)         % Doit donner [0,1,0,0]
isnan(cos(1./V))    % Doit donner [0,1,0,0]
V>0.5               % Doit donner [1,0,0,1]
(V>-10 & V<=3)      % Doit donner [1,1,1,0]
(V>-10 | V<=3)      % Doit donner [1,1,1,1]
find(V>0.5)         % Doit donner [1,4]
V(find(V>0.5)).^2   % Doit donner [1.0,11.56]
any(V)              % Doit donner 1
all(V)              % Doit donner 0
% Aide sur les opérations élémentaires
help ops
```

Exercice 0.7.1. Que donne `find(M)` pour une matrice `M`? Que donne alors `[I,J]=find(M)`? `[I,J,K]=find(M)`?

Le langage Matlab comprend les boucles du type `for` et `while` ainsi que les structures d'exécutions conditionnelles du type `if`. L'argument d'un `if` et d'un `while` est de type logique, ce qui n'est pas le cas de celui d'un `for`, qui est de type matriciel. La commande `break` permet de sortir de la boucle `while` où `for` la plus interne.

```
% Boucles FOR
N=10;
for n=1:N,
 for m=1:N,
  A(n,m)=1/(n+m+1);
 end
end

% Boucle WHILE (TANT QUE en français)
n=10;
r=-Inf;
while (A(n,n)>0),
 r=A(n,n);
 n=n+1;
end

% Exécution conditionnelle IF
if n==m
 A(n,m)=1;
elseif abs(n-m)==1
 A(n,m)=-1;
else
 A(n,m)=0;
end

% Exécution ESSAI/ERREUR/INTERCEPTION
% Tente de déterminer l'inverse de M
% Affiche NaN si ça n'est pas possible
% Aucune erreur n'est provoquée.
% À tester avec M=eye(2) et M=ones(1,2)
```

```
31 try, inv(M), catch, disp(NaN); end
```

0.8 Fonctions et fichiers .m

Les *commandes* Matlab peuvent toutes être considérées comme des *fonctions*, c'est-à-dire des *entitées nommées*, qui prennent des paramètres éventuels (arguments) et qui renvoient des résultats (valeurs de retour). Ce sont en quelque sorte des sous-programmes, bien que la notion de programme n'ait pas beaucoup de sens pour un interpréteur de langage comme Matlab.

Beaucoup de fonctions Matlab, comme `mean` par exemple, sont en réalité écrites en Matlab, et le code Matlab correspondant est stocké dans un fichier dont le nom se termine par .m. Pour `mean`, il s'agit de mean.m. Pour ajouter de nouvelles fonctions à Matlab, il nous suffit d'écrire de nouveaux fichiers de ce type. Vous aurez compris qu'un fichier .m ne contient qu'une seule fonction, qui a le même nom que le fichier, au suffixe .m près.

Voici un code qui définit une fonction `stat`, qui prend comme paramètre un vecteur `x` et qui renvoie sa moyenne et son écart type. Ce code devra être stocké dans le fichier nommé stat.m pour que Matlab fasse le lien avec la fonction `stat`. Le commentaire de la ligne n°2 constitue l'aide qui est affichée lorsque l'utilisateur tape `help stat`.

```
1 function [moyenne,ecartype] = stat(x)
2 %STAT Revoie la moyenne et l'écart type du vecteur passé en argument.
3 n = length(x);
4 moyenne = sum(x) / n;
5 ecartype = sqrt(sum((x - moyenne).^2)/n);
6 return % En fait, inutile en fin de fonction...
```

Le fichier stat.m devra être placé dans un répertoire que Matlab scrutera. En général, Matlab cherche automatiquement dans le répertoire en cours. Pour afficher le répertoire en cours, utilisez la fonction `pwd`, pour lister son contenu, utilisez la fonction `ls` et pour changer de répertoire courant, utilisez la fonction `cd`.

Remarque 0.8.1 *(Sous-fonctions).* On peut définir des fonctions dans le corps d'une fonction. Elles ne seront visibles que pour la fonction principale, et l'on parle alors de sous-fonctions, cf. `help function`. *Vous pouvez vous en passer en première lecture.*

0.8.1 Nombre variable d'arguments et de valeurs de retour

Il est parfois commode d'écrire des fonctions qui prennent un nombre variable d'arguments. Ceci peut être fait en Matlab grâce à la commande `nargin` qui renvoie le nombre d'arguments passés à la fonction lors de son appel. Voici un exemple :

```
1  function fd = firstdigit(x,b,convert)
2  %fd = firstdigit(x,b)
3  % Renvoie le premier chiffre non nul dans l'écriture en base b
4  % de x, en partant de la gauche.
5  % * x doit être un vecteur de réels >0 pas trop grands.
6  % * b doit être un entier entre 2 et 36
7  %     36 = 10 + 26, histoire de pouvoir écrire avec l'alphabet...
8  %     ce paramètre est optionnel et vaut par défaut 10.
9  % * convert est un paramètre optionnel valant par défaut 0.
10 %   si convert n'est pas nul, la sortie fd est un entier correspondant
11 %   à la valeur en base 10 du premier chiffre, alors que si convert est
12 %   nul (par défaut), la sortie fd est un caractère alphanumérique.
13
14 %%% base 10 par défaut
```

```
if (nargin==1), b=10; end
%%% si dépassement, on sort avec une erreur
if ~(ceil(b)==b & b>1 & b<37)
 error('b n''est pas un entier entre 2 et 36');
end
%%% si x n'est pas positif, on prend sa valeur absolue
if ~(x>0)
 warning('x n''est pas positif, il est remplace par -x');
 x=-x;
end
x=reshape(x,[1,length(x)]);
%%% décalage de x en base b tel que y>=1 en base b
y=x.*b.^ceil(-log(x)/log(b));
%%% si dépassement, on sort avec une erreur
if isinf(y) | isnan(y) | ~isreal(y),
 error('x trop petit ou trop grand par rapport à b');
end
c=floor(y);                    % c = premier chiffre en base 10
s=dec2base(c,b);               % conversion en base b
if nargin==3,
  if convert,
       fd=base2dec(s(:,1),b);  % valeur en base 10 du premier caractère
  end
else
  fd=s(:,1);                   % Premier caractère en base b.
end
return;
```

De la même manière, on peut définir grâce à **nargout** des fonctions qui renvoient un nombre variable de valeurs de retour, un peu comme le fait la fonction **eig** que nous avons déjà vue. Voici un exemple de fonction Matlab, qui renvoie une approximation constante par morceaux de la fonction de répartition empirique d'un échantillon aléatoire, en utilisant la fonction Matlab **hist** :

```
function  [ecdf,mi,ma,dt,rg] = empcdf(e,n)
%EMPCDF renvoie une approximation constante par morceaux de la
%fonction de répartition empirique d'un échantillon.
%[ecdf,min,max] = empicdf(e,n)
% * e est un échantillon (vecteur de réalisations i.i.d.)
% * n est un entier optionnel valant par défaut 10.
% * L'intervalle [min(e),max(e)] est subdivisé en n sous-intervalles
%   de même longueur. ecdf est un vecteur de longueur n.
%   ecdf(i) représente la valeur de la fonction de répartition
%   empirique à l'extrémité droite du i-ème sous-intervalle (classe).
% * Les valeurs de retour mi,ma,rg et dt sont optionnelles
%   et correspondent respectivement à l'étalement de l'échantillon et au
%   pas de discrétisation associé à n.
% * Les sorties peuvent être exploitées pour un tracé d'une approximation
%   affine par morceaux de la fonction de répartition empirique :
%    plot([mi+dt:dt:ma],ecdf)
%EMPCDF fait appel à HIST.
if (nargin==1), n=10; end;
[frequences,classes]=hist(e,n,1); %histogramme normalisé en n classes.
ecdf=cumsum(frequences);
if (nargout >= 2), mi=min(e); end
if (nargout >= 3), ma=max(e); end
if (nargout >= 4), dt=(ma-mi)/n; end
```

```
24 if (nargout >= 5), rg=ma-mi; end  %i.e. rg=range(e)
25 return;
```

0.8.2 Laïus sur les fichiers .m et Matlab

La commande Matlab **type** permet de lister le contenu du fichier .m d'une fonction. Ainsi, **type rbinom** va vous montrer le code source Matlab de la fonction Stixbox **rbinom**. Un certain nombre de fonctions Matlab ne correspondent à aucun fichier .m, elles sont « internes » à Matlab pour une plus grande efficacité et l'on parle de fonctions « built-in » en anglais. C'est par exemple le cas de la fonction **type** elle-même! Cette séparation entre fonctions internes et externes se retrouve dans la plupart des interpréteurs de langages. La commande **exist** permet de connaître le type d'une commande où d'une variable, cf. **help exist**.

Sur la plupart des systèmes informatiques, les fichiers sont organisés en une structure arborescente de répertoires qui contiennent des fichiers et des sous-répertoires. La commande Matlab **which** donne le chemin du fichier .m associé à une fonction Matlab, lorsqu'il existe. Par exemple, **which mean -ALL** affichera tous les fichiers mean.m avec leurs chemins. Seul le premier sera utilisé par Matlab! La commande **what** liste les fichiers .m du répertoire en cours. Faites **help what** et **help which** pour plus d'information.

Matlab a besoin de savoir où chercher les fichiers .m dans l'arborescence des fichiers du système. Il recherchera automatiquement dans une liste de réperoires appelée « PATH ». La commande **path** vous liste les répertoires du « PATH ». Les commandes **addpath** et **rmpath** permettent d'ajouter et d'enlever des répertoires de la liste « PATH ». Ces modifications ne sont pas permanentes et sont perdues lorsque vous quittez Matlab.

Par défaut, la liste « PATH » contient les répertoires de Matlab, qui contiennent eux-même les fichiers .m qui constituent Matlab. Si Stixbox a été correctement installée, le répertoire qui contient les fichiers .m qui la constituent a été ajouté au « PATH » par l'administrateur système. Afin d'accélérer la lecture des fichiers .m, Matlab maintient en permanence une base de données des fichiers .m avec le répertoire associé qui les contient. La commande **rehash** permet de mettre à jour cette base de données en relisant la liste « PATH ».

Il faut enfin savoir que Matlab cherche d'abord les fichiers .m des fonctions dans le répertoire courant (« working directory »). Répétons-le, la commande **pwd** affiche le répertoire courant (« print working directory »), la commande **cd** change de répertoire courant (« change directory »), et les commande **ls** ou **dir** listent le contenu du répertoire courant.

En général, vous n'aurez pas à modifier la liste « PATH ». Sa modificaton permanente dépend du système utilisé (Unix, MS-Windows,...).

0.8.3 Récursivité

Le langage Matlab autorise la récursivité, comme le montre l'exemple archi-classique suivant :

```
1 function fn=fact(n);
2 %FACT calcule de façon récursive et inefficace la factorielle de n
3 %fn=fact(n);
4 if (n<=0), fn=1; else fn=n*fact(n-1); end;
5 return;
```

0.8.4 Fonctions en ligne

Pour des fonctions très légères, on peut procéder autrement que par la création d'un fichier .m : la directive **inline** permet de créer des fonction nouvelles à la volée. Voici un exemple :

```
% On définit une nouvelle fonction trisin qui prend 3 paramètres a,b et c
trisin=inline('sin(a)*sin(b)*sin(c)');
% On teste notre nouvelle fonction
trisin(pi/2,pi/4,pi/2)
```

Matlab détermine automatiquement le nombre de paramètres de la fonction. Bien entendu, une fois que vous quitterez Matlab, cette fonction sera perdue car elle n'est pas sauvegardée dans un fichier.

Exercice 0.8.2. Apprenez à utiliser la fonction `fplot` pour tracer les graphes de fonctions définies avec `inline`.

0.9 Débogage & optimisation

Diverses commandes permettent de déboguer les programmes Matlab. Citons par exemple `dbstack`, `dbstep`, `dbstop`, `dbcont`, `dbclear`, `dbtype`, `dbup`, `dbdown`, `dbstatus` et `dbquit`. Leur aide vous renseignera sur leur usage.

La commande `keyboard` permet à l'utilisateur de reprendre le contrôle en interrompant momentanément l'interpretteur Matlab, afin par exemple d'examiner le contenu des variables, cf. `help keyboard`.

Dans une fonction, la commande `error` permet d'afficher un message d'erreur et provoque la sortie immédiate de la fonction, tandis que la commande `warning` permet d'afficher un message d'avertissement, sans pour autant interrompre l'exécution de la fonction. Les messages d'avertissement peuvent être désactivés, cf. `help warning`. Les commandes `lasterr` et `lastwarn` permettent d'afficher respectivement le dernier message d'erreur et le dernier message d'avertissement. La commande `mfilename` permet de connaître le fichier `.m` qui contient la fonction en cours d'exécution.

Les commandes `tic` et `toc` permettent de déterminer le temps d'execution d'une commande, ce qui est très utile pour l'optimisation et la comparaison lors du test de différentes méthodes. Voici un exemple dont la sortie graphique est donnée par la figure 2 :

```
% Comparaison de la rapidité de plusieurs méthodes
% de calcul de la factorielle grace a tic & toc.

clear; nmin=2; nmax=150; N=[nmin:nmax];
for n=N,
 m=n-nmin+1;
  tic; fact(n); T(m,1)=toc;
  tic; f=nmin; for i=nmin+1:n; f=f*i; end; T(m,2)=toc;
  tic; cumprod(nmin:n); T(m,3)=toc;
  tic; gamma(n-1); T(m,4)=toc;
end
clf; hold on; title('Calcul de la factorielle');
plot(N,T(:,1),N,T(:,2),N,T(:,3),N,T(:,4));
xlabel('n'); ylabel('Temps');
legend('Recursif','Boucle','cumprod','gamma built-in',2);
```

La commande `cputime` permet d'obtenir le *temps machine* utilisé par Matlab depuis son lancement, cf. `help cputime`.

Exercice 0.9.1. Pourquoi les tracés sont linéaires dans l'exemple précédent ?

Exercice 0.9.2. Utiliser `cmutime` en lieu et place de `tic` et `toc` dans l'exemple précédent.

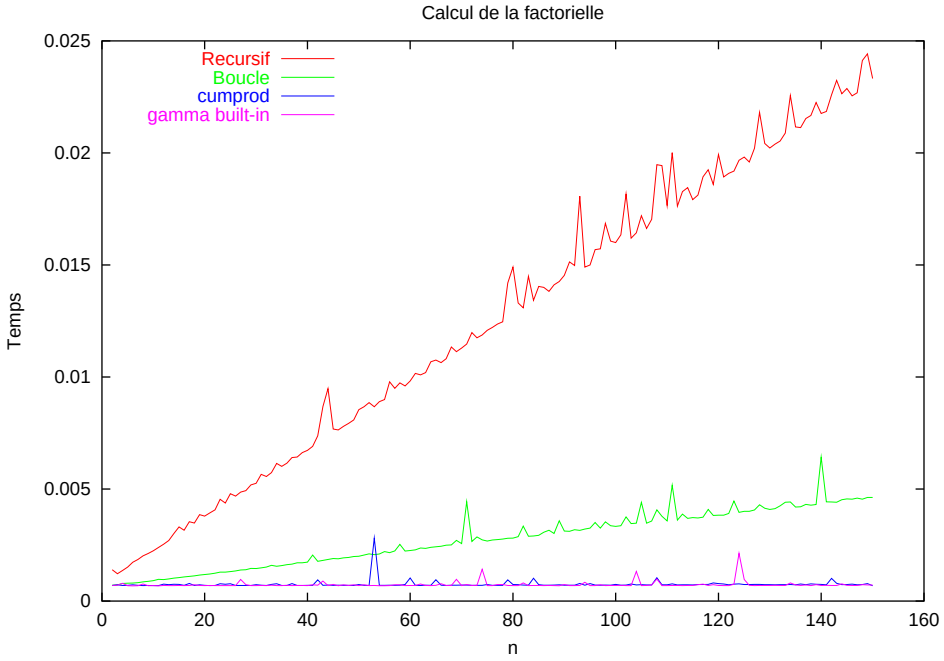

FIG. 2 – Comparaison des temps de calcul de la fonction factorielle (Octave 2.1).

0.10 Derniers conseils

Nous verrons, tout au long des travaux pratiques, plusieurs autres fonctions Matlab dont il n'a pas été question dans ce chapitre. Voici quelques petits conseils avant de vous lancer seul :

1. Expérimentez vous-même, en vous servant de l'aide au moyen des fonctions `help` et `lookfor`.
2. La combinaison de touches `Ctrl-c` (appuyer sur la touche `Ctrl` et sur la touche `c` en même temps) permet d'interrompre l'exécution de la commande en cours, ce qui peut être très utile pour sortir d'une longue boucle... La touche `tab` permet de compléter la commande que l'on a commencé à saisir.
3. N'oubliez pas de désactiver l'affichage d'une commande en la suffixant par un point-virgule, surtout dans une boucle ou pour une grosse matrice.
4. Sauvegardez régulièrement votre code Matlab dans un fichier pour éviter de le perdre en cas de problème.
5. Dites-vous bien que vous vous trompez bien plus souvent que Matlab !
6. Le langage Matlab étant par défaut interprété, il est souvent plus efficace d'adopter une formulation matricielle en lieu et place de boucles pour certains calculs. Cependant, les boucles sont parfois plus lisibles, à vous de voir.
7. Si vous disposez d'un micro-ordinateur PC chez vous, je vous conseille d'installer Scilab ou Octave, qui sont plus ou moins libres et ressemblent à Matlab. La licence de ce dernier est chère et je ne peux pas vous encourager à le pirater...
8. Exercez-vous tout au long de l'année pour vous familiariser et réflechir à des développements, n'attendez pas le lendemain des écrits pour vous y mettre !
9. Dites-vous bien que le jury n'attend pas des « petits génies » de l'informatique, mais plutôt des candidats qui l'épatent avec des mathématiques appliquées...
10. Ne prenez pas cette liste pour les dix commandements.

Un dernier petit exercice avant de clore ce chapitre introductif :

Exercice 0.10.1. Comprenez-vous le code suivant ?

```
clear;
n=100; a=1; b=2;
A=repmat([-n:n].^2,2*n+1,1);
B=(sqrt(a*A+b*A')<n-1)+(sqrt(a*A+b*A')>n+1);
figure(1); clf; imagesc(B);
```

Chapitre 11

Quelques lois classiques

Dans toute la suite, on note $\mathcal{L}(X)$ la loi de la v.a. X et $*$ la convolution des fonctions ou des lois de probabilité :

$$(f*g)(x) := \int f(x-y)g(y)\,dy = \int f(z)g(x-z)\,dz = (g*f)(x),$$

et

$$\mathbf{E}_{\mu*\nu}(\varphi) := \int \varphi(x+y)\,d(\mu\otimes\nu)(x,y) = \int \varphi(x+y)\,d(\nu\otimes\mu)(x,y) = \mathbf{E}_{\nu*\mu}(\varphi)\,.$$

Si μ et ν ont pour densités respectives f et g par rapport à la mesure de Lebesgue, alors $\mu*\nu$ a pour densité $f*g$. On rappelle que la convolée $\mu*\nu$ de deux mesures de probabilité représente la loi de la somme de deux v.a. indépendantes X, Y de loi respectives μ et ν :

$$X, Y \text{ indépendantes} \Rightarrow \mathcal{L}(X+Y) = \mathcal{L}(X) * \mathcal{L}(Y).$$

La réciproque est fausse en général. On note μ^{*n} la loi obtenue en convolant n fois μ :

$$\mu^{*n} := \underbrace{\mu * \cdots * \mu}_{n \text{ fois.}}.$$

D'autre part :

$$X, Y \text{ indépendantes} \Leftrightarrow \mathcal{L}((X,Y)) = \mathcal{L}(X)\otimes\mathcal{L}(Y).$$

On note δ_a la masse de Dirac en a, c'est à dire la mesure de probabilité qui affecte la masse 1 aux ensembles mesurables contenant le point a et 0 aux autres. C'est la loi d'une v.a. constante de valeur a. Sa moyenne vaut a et sa variance est nulle. Pour tout μ et ν, on a $\mu*\nu = \nu*\mu$ et $\mu*\delta_0 = \delta_0*\mu = \mu$ et $\delta_a*\delta_b = \delta_{a+b}$. La masse de Dirac en 0 est donc l'élément neutre de la convolution.

Heuristique 11.0.1 *(Convolution = lissage).* Convoler une fonction f par une fonction g revient à prendre en x une moyenne des valeurs de f pondérées par la translation de g en x. Il est donc naturel que la masse de Dirac en 0 soit l'élément neutre de la convolution. Si g est lisse, on fait donc du lissage de f. Pour ce faire, on prend souvent pour g une fonction $\mathcal{C}^\infty$ paire et à décroissance rapide voire même à support compact de façon à remplacer $f(x)$ par une « g-moyenne » de f au voisinage de x. Penser à la technique de régularisation par convolution utilisée en analyse, en particulier pour montrer la densité de $\mathcal{C}_c^\infty$ dans les espaces $\mathbf{L}^p$ de Lebesgue.

Heuristique 11.0.2 *(Convolution des densités = addition d'un bruit indépendant sur la v.a.).* Soit X et Y des v.a. de densités respectives f et g. Dire que g est *lisse* signifie que la v.a. Y est purement *continue*, c'est à dire que les valeurs prises sont bien réparties et sans sauts. Ainsi, si X est indépendante de Y, la v.a. $X+Y$ aura en général des valeurs mieux réparties que celles de X lorsque g est lisse. La densité de $X+Y$ est la convolée de la densité f de la v.a. X par la densité

g de la v.a. Y. Si g est lisse, il en sera de même pour $f * g$. Ainsi, lisser une densité se traduit sur les v.a. par l'addition d'une v.a. indépendante à densité lisse (penser par exemple au lissage d'une v.a. discrète X par addition d'une v.a. indépendante Y de loi uniforme où gaussienne, centrée et de faible variance par rapport à l'écart entre les valeurs discrètes prises pas X).

Remarque 11.0.3 *(Fonctions de répartition et densité).* Considérons une loi de probabilité μ sur $\mathbb{R}$, de fonction de répartition F. En tant que fonction de répartition, la fonction F est croissante, continue à droite, et $F(\mathbb{R}) = [0,1]$. On montre alors qu'elle ne peut avoir qu'un nombre dénombrable de points de discontinuité, notés $\{s_n, n \in \mathbb{N}\} \subset \mathbb{R}$, cf. [BL98, prop. III.2.2 page 48]. On montre de plus, [BL98, III, pages 49-50], qu'il existe trois réels positifs ou nuls α, β, γ vérifiant $\alpha + \beta + \gamma = 1$, tels que F possède la décomposition suivante :

$$F = \alpha F_{\text{disc}} + \beta F_{\text{sing}} + \gamma F_{\text{cont}},$$

où les fonctions $F_\bullet$, qui vérifient les propriétés suivantes, n'existent que lorsque le coefficient associé est non nul :

1. F_{disc} est constante par morceaux, et c'est la fonction de répartition de la loi de probabilité discrète $\sum_{n=0}^{+\infty} \delta_{s_n}$
2. F_{cont} est absolument continue et c'est la fonction de répartition d'une loi de probabilité à densité par rapport à la mesure de Lebesgue sur $\mathbb{R}$
3. F_{sing} est continue et c'est la fonction de répartition d'une mesure de probabilité singulière par rapport à la mesure de Lebesgue sur $\mathbb{R}$

F est continue si et seulement si $\alpha = 0$ et μ admet une densité si et seulement si $\alpha = \beta = 0$. Il ne suffit donc pas que F soit continue pour que μ admette une densité par rapport à la mesure de Lebesgue. En d'autre termes, la composante étrangère à la mesure de Lebegue de μ dans sa décomposition de Lebesgue-Radon-Nikodym (cf. [Rud75, thm 6.9, page 117]) ne se réduit pas aux sauts de la fonction F. Un exemple simple est fourni par les lois fractales de la section 1.5.1 page 24. Bien entendu, si F est C^1, alors μ admet une densité qui n'est rien d'autre que F'.

Remarque 11.0.4 *(Densité au sens de la théorie des distributions).* Une loi de probabilité est, en tant que mesure de Radon, une distribution. Elle s'identifie à sa densité lorsque cette dernière existe. La fonction de répartition F_X d'une v.a.r. X existe toujours. Sa dérivée F'_X au sens des distributions a donc toujours un sens. Lorsque la loi $\mathcal{L}(X)$ admet une densité f, cette dernière n'est rien d'autre que F'_X. Les points de discontinuité de F_X se traduisent sur F'_X par des masses de Dirac au sens des distributions, tandisque les « plats » du graphe de F_X correspondent à des intervalles qui ne sont par portées par $\mathcal{L}(X)$. Il ne suffit pas que F_X soit continue pour que $\mathcal{L}(X)$ admette une densité, i.e. que F'_X soit une fonction en tant que distribution. Les F_X pour lesquelles c'est le cas sont dites « absolument continues » et s'écrivent comme l'intégrale d'une fonction $\mathbf{L}^1$ (qui est alors la densité associée). Une condition suffisante d'absolue continuité est que F_X soit $\mathcal{C}^1$.

Remarque 11.0.5. Les générateurs aléatoires de la bibliothèque Stixbox pour Matlab sont listés dans la table 1.1 page 43. Le tableau 11.1 page 107 recense quelques lois usuelles. Pour les démonstrations et des extensions, on renvoie principalement à [DCD82a] et [Bou86].

11.1 Fonctions génératrices – la transformée du pauvre

Si X est une variable aléatoire à valeurs dans $\bar{\mathbb{N}}$, on définit sa fonction génératrice par :

$$\begin{aligned} G_X : [0,1[&\to \mathbb{R} \\ s &\mapsto G_X(s) := \mathbf{E}\left(s^X\right) = \sum_{k=0}^{+\infty} s^k \, \mathbb{P}(X = k). \end{aligned}$$

Elle ne dépend que de la loi de X, qu'elle caractérise. En d'autre termes, si X et Y sont deux v.a. entières, alors $\mathcal{L}(X) = \mathcal{L}(Y)$ si et seulement si $G_X = G_Y$. De plus, $G_{X+Y} = G_X G_Y$ lorsque X et Y sont indépendantes (la réciproque est fausse). On peut définir la fonction génératrice du couple de v.a. entières (X, Y) en $(s_1, s_2) \in [0,1[^2$ par : $G_{(X,Y)}(s_1, s_2) := \mathbf{E}\left(s_1^X s_2^Y\right)$. Les v.a. X et Y sont alors indépendantes si et seulement si $G_{(X,Y)}(s_1, s_2) = G_X(s_1) G_Y(s_2)$ pour tout $(s_1, s_2) \in [0,1[^2$.

La fonction génératrice permet également de calculer les moments puisque $\mathbb{P}(X = n) = \frac{1}{n!} G_X^n(0)$ et $G'_X(1) = \mathbf{E}(X)$ et plus généralement : $G_X^{(n)}(1) = \mathbf{E}(X(X-1)\cdots(X-n+1))$.

11.2 Lois discrètes à support sur entiers naturels

On rappelle que si X et Y sont deux v.a. entières (i.e. à valeurs dans $\bar{\mathbb{N}} := \mathbb{N} \cup (+\infty)$), indépendantes, de lois respectives μ et ν, alors :

$$\mathbb{P}(X + Y = n) = \sum_{k_1 + k_2 = n} \mathbb{P}(X = k_1)\mathbb{P}(Y = k_2),$$

en d'autre termes :

$$(\mu * \nu) = \sum_{k_1 + k_2 = n} \mu(\{k_1\})\, \nu(\{k_2\})\, \delta_n.$$

11.2.1 Loi de Bernoulli

C'est la loi $q\delta_a + p\delta_b$ où $p \in [0,1]$, $q := 1 - p$ et $a \neq b$. Le plus souvent, on utilise $q\delta_{-1} + p\delta_1$ ou $q\delta_0 + p\delta_1$.

La loi $q\delta_0 + p\delta_1$ est de moyenne p et de variance pq. Elle permet de modéliser le résultat d'un jet de pièce dans un jeu de pile ou face, avec probabilité p de gagner (codé 1) et probabilité $q := 1 - p$ de perdre (codé 0). Elle a pour fonction génératrice $q + ps$.

11.2.2 Loi uniforme discrète

C'est la loi donnée par :

$$\frac{1}{n} \sum_{i=1}^{n} \delta_{a_i},$$

où les a_i sont tous différents. Elle a pour moyenne $(a_1 + \cdots + a_n)/n$ et pour variance $(n\|a\|_2^2/n - (a_1 + \cdots + a_n)^2)/n^2$. On l'utilise pour modéliser un phénomène pouvant prendre n valeurs différentes $a_1, \ldots, a_n$ avec la même probabilité $1/n$. Elle a pour propriété remarquable de maximiser, à n fixé, l'entropie de Shannon $\mathbf{H}(p_1, \ldots, p_n)$ définie pour une loi de probabilité $p_1\delta_{a_1} + \cdots + p_n\delta_{a_n}$ par :

$$\mathbf{H}(p_1, \ldots, p_n) := -\sum_{k=1}^{n} p_i \log p_i.$$

11.2.3 Schéma de Bernoulli

Le schéma de Bernoulli correspond à un jeu de pile ou face infini, avec probabilité de succès à chaque lancé de $p \in]0,1[$ (les cas extrêmes sont triviaux). On le modélise par la donnée d'une suite $(X_n, n \in \mathbb{N}^*)$ de v.a. i.i.d. de loi de Bernoulli de paramètre p sur $\{0,1\}$: à chaque lancé, on code 0 un échec (probabilité $q := 1 - p$) et 1 un succès (probabilité p). Le théorème de Caratheodory permet de définir une tribu et une loi de probabilité $\mathcal{L}$ sur l'ensemble $\{0,1\}^{\mathbb{N}^*}$ des suites infinies de 0 et de 1 telle que $(X_1, X_2, \ldots)$ soit de loi $\mathcal{L}$. Bien entendu, pour toute suite s dans cet espace, on a $\mathcal{L}(\{s\}) = 0$. En revanche, on a pour tout $n \in \mathbb{N}^*$ et $a \in \{0,1\}^n$,

par indépendance des $X_1, \ldots, X_n$: $\mathbb{P}(X_1 = a_1, \ldots, X_n = a_n) = \mathbb{P}(X_1 = a_1) \cdots \mathbb{P}(X_n = a_n)$. La plupart des questions que l'on peut se poser sur le jeu de pile ou face reviennent à étudier certaines propriétés de la loi $\mathcal{L}$.

11.2.4 Loi binomiale

Notée $\mathcal{B}(n,p)$, c'est la loi :

$$\sum_{k=0}^{n} C_n^k \, p^k (1-p)^{n-k} \, \delta_k = (q\delta_0 + p\delta_1)^{*n}.$$

Elle permet de modéliser le nombre de gains pour n lancés au jeu de pile ou face avec probabilité de gain de p. Si $X_1, \ldots, X_n$ sont i.i.d. de loi de Bernoulli $q\delta_0 + p\delta_1$, alors $X_1 + \cdots + X_n$ suit la loi $\mathcal{B}(n,p)$. La probabilité de gagner k fois en n lancés vaut $C_n^k p^k (1-p)^{n-k}$. Sa moyenne est np et sa variance npq. Elle a pour fonction génératrice :

$$G_{\mathcal{B}(n,p)}(s) := \mathbf{E}_{\mathcal{B}(n,p)}(s^\bullet) = (q+ps)^n.$$

11.2.5 Loi géométrique

La loi géométrique $\mathcal{G}(p)$ de paramètre $p \in [0,1]$, ou loi de Pascal, correspond à la loi du premier succès à un jeu de pile ou face avec probabilité de gain p. Si $(X_n, n \in \mathbb{N}^*)$ est une suite de v.a. i.i.d. de loi de Bernoulli $q\delta_0 + p\delta_1$, la loi géométrique de paramètre p est la loi de la v.a. « premier succès » définie par :

$$T(p) := \inf\{n \in \mathbb{N}^*, X_n = 1\}.$$

Elle est donnée par :

$$\mathcal{G}(p) := p^{-1} q^{-1} \sum_{k=1}^{+\infty} p^k \delta_k.$$

Sa moyenne vaut $1/p$, sa variance q/p^2 et sa fonction génératrice est $sp/(1-sq)$.

On préfère parfois numéroter les lancés de pile ou face à partir de 0, on a alors la loi géométrique de base 0 :

$$p^{-1} \sum_{k=0}^{+\infty} p^k \delta_k,$$

qui a pour moyenne q/p et pour variance q/p^2.

11.2.6 Loi négative-binomiale

C'est la loi du nombre de lancés nécessaires à un jeu de pile ou face de probabilité de gain p pour obtenir exactement m succès. C'est donc la loi de la v.a. $T_m(p)$ où $T_1(p) = T(p)$ (de loi géométrique) et $T_i(p)$ $(i \geqslant 1)$ est défini par récurrence par :

$$T_i(p) := \inf\{n > T_{i-1}, X_n = 1\}.$$

Il est facile de voir que $T_m(p)$ à la même loi que la somme de m v.a. i.i.d. de loi géométrique de paramètre p. On a donc :

$$\mathcal{G}_m(p) = \sum_{k=m}^{+\infty} C_{n-1}^{m-1} \, p^m q^{k-m} \, \delta_k.$$

On retrouve bien la loi géométrique pour $m = 1$: $\mathcal{G}(p) = \mathcal{G}_1(p)$. La loi $\mathcal{G}_m(p)$ a pour moyenne m/p, pour variance mq/p^2 et pour fonction génératrice $(ps/(1-sq))^m$. Elle tire son nom du fait que les coefficients de sa fonction génératrice proviennent d'une sorte de « formule du binôme inversé ».

11.2.7 Loi de Poisson

La loi de Poisson $\mathcal{P}(\lambda)$ de paramètre $\lambda > 0$ est définie par :

$$e^{-\lambda} \sum_{n=0}^{+\infty} \frac{\lambda^n}{n!} \delta_n,$$

Elle a pour moyenne et variance λ et pour fonction génératrice $e^{\lambda(s-1)}$.

11.2.8 Loi multinomiale

La loi multinomiale $\mathcal{M}(n, k, p_1, \ldots, p_k)$ de taille $n \in \mathbb{N}^*$, de dimension $k \in \mathbb{N}^*$ et de paramètre $(p_1, \ldots, p_k) \in \mathbb{R}_+$ avec $p_1 + \cdots + p_k = 1$ est la loi sur $\mathbb{N}^k$ donnée par :

$$\mathcal{M}(n, k, p_1, \ldots, p_k)(\{n_1, \ldots, n_k\}) := \frac{n!}{n_1! \cdots n_k!} p_1^{n_1} \cdots p_k^{n_k}.$$

Elle tire son nom de la formule du multinôme :

$$(a_1 + \cdots + a_k)^n = \sum_{\substack{n_1, \ldots, n_k \\ n_1 + \cdots + n_k = n}} \frac{n!}{n_1! \cdots n_k!} a_1^{n_1} \cdots a_k^{n_k}.$$

On retrouve la loi binomiale pour $k = 1$: $\mathcal{M}(n, k, p) = \mathcal{B}(n, p)$. Si $(X_1, \ldots, X_n)$ sont des v.a. i.i.d. de loi $(p_1, \ldots, p_k)$ sur $\{1, \ldots, k\}$, et si $N_i := \#\{j = 1, \ldots, n, \text{ tq. } X_j = i\}$ alors $(N_1, \ldots, N_k)$ suit la loi $\mathcal{M}(n, k, p_1, \ldots, p_k)$. Remarquons que pâr définition, $N_1 + \cdots + N_k = n$. La loi multinomiale permet par exemple de modéliser les résultats de n lancés indépendants d'un dé à k faces, N_i représentant alors le nombre de fois où la face numéro i a été obtenue. On a enfin la convergence étroite suivante vers une loi normale :

$$\left(\frac{N_1 - np_1}{\sqrt{np_1}}, \ldots, \frac{N_k - np_k}{\sqrt{np_k}} \right) \xrightarrow[n \to +\infty]{\text{étr.}} \mathcal{N}(0, \Sigma(p_1, \ldots, p_k)),$$

où

$$\Sigma(p_1, \ldots, p_k) := \mathbf{I}_k - (\sqrt{p_1}, \ldots, \sqrt{p_k})^\top (\sqrt{p_1}, \ldots, \sqrt{p_k}).$$

La loi normale $\mathcal{N}(0, \Sigma(p_1, \ldots, p_k))$ est la loi de la projection d'un vecteur aléatoire de loi $\mathcal{N}(0, \mathbf{I}_k)$ sur l'hyperplan orthogonal au vecteur $\left(\sqrt{p_1}, \ldots, \sqrt{p_k}\right)^\top$. Elle est dégénérée (i.e. sa matrice de covariance est singulière).

11.2.9 Loi hypergéométrique

La loi hypergéométrique de paramètres $(N, N_1, n) \in \mathbb{N}^* \times \mathbb{N}^* \times \mathbb{N}^*$ avec $N_1 \leqslant N$ et $n \leqslant N$ est la loi sur $\{0, \ldots, n\}$ donnée par :

$$\sum_{k=0}^{n} \frac{C_{N_1}^k C_{N-N_1}^{n-k}}{C_N^n} \delta_k.$$

Si l'on considère une population de N individus dont N_1 possèdent un caractère particulier, par exemple une maladie, alors la loi géométrique de paramètre (N, N_1, n) représente la loi du nombre d'individus possédant ce caractère dans un échantillon de taille n de la population. C'est donc la loi de $X_1 + \cdots + X_n$ où les v.a. X_i sont i.i.d. et ont pour loi $((N - N_1)/N)\delta_0 + (N_1/N)\delta_1$.

Elle a pour moyenne nN_1/N et pour variance $n(N-n)N_1(N-N_1)(N-1)^{-1}N^{-2}$. De la même façon que nous avions généralisé la loi binomiale en une loi multinomiale, on pourrait définir une loi hypergéométrique multiple lorsque l'on s'intéresse à un nombre $m \geqslant 1$ de caractères dans la population.

11.3 Transformées de Fourier & de Laplace

Pour les lois continues, la fonction génératrice est remplacée par la transformée de Fourier ou de Laplace, qui sont plus puissantes et gardent leur sens pour les lois discrètes.

Soit μ une mesure de probabilité sur $(\Omega, \mathcal{A})$ où Ω est un espace vectoriel topologique et $\mathcal{A}$ sa tribu borélienne. On note Ω' l'espace dual topologique de Ω, constitué des formes linéaires continues sur Ω. La transformée de Fourier de μ est définie par :

$$\begin{aligned} \mathcal{F}ourier(\mu) : \Omega' &\to \mathbb{R} \\ f &\mapsto \mathbf{E}_\mu(\exp(if)) \end{aligned}$$

Cette fonction est bien définie puisque pour tout $f \in \Omega'$, la fonction $\exp(if) : \Omega \to \mathbb{C}$ est continue et bornée. Lorsque $\Omega \subset \mathbb{R}^n$, on identifie Ω' à $\mathbb{R}^n$ et on pose :

$$\begin{aligned} \mathcal{F}ourier(\mu) : \mathbb{R}^n &\to \mathbb{C} \\ u &\mapsto \int_\Omega \exp(i < u, x >) \, d\mu(x). \end{aligned}$$

La transformée de Laplace, quand à elle, est définie (lorsque cela a un sens) par :

$$\begin{aligned} \mathcal{L}aplace(\mu) : \mathbb{R}^n &\to \mathbb{R} \\ s &\mapsto \int_\Omega \exp(- < s, x >) \, d\mu(x). \end{aligned}$$

Formellement, il suffit de prendre $u = is$ dans la définition de la transformée de Fourier pour obtenir la transformée de Laplace, lorsqu'elle existe.

La transformée de Fourier (et de Laplace) caractérise la loi et l'indépendance. Ainsi, deux v.a. ont la même loi si et seulement si elles ont même transformée et X et Y sont indépendantes si et seulement si la transformée du couple (X, Y) en (s_1, s_2) est égale au produit des transformées respectives en s_1 et s_2. bien entendu, la transformée en s de la somme de deux v.a. indépendantes est égale au produit des transformées respectives en s (la réciproque est fausse).

Si X a pour loi μ, on dit que μ est symétrique si $-X$ a même loi que X. La transformée de Fourier d'une loi symétrique est réelle.

La transformée de Fourier de δ_a avec $a \in \mathbb{R}^n$ est donnée par $e^{i<s,a>}$, elle permet de calculer les transformées de Fourier des lois discrètes (i.e. à support au plus dénombrable sans point d'accumulation). Ainsi, les lois binomiales $\mathcal{B}(n,p)$ et de Poisson $\mathcal{P}(\lambda)$ ont pour transformée de Fourier respectives :

$$(pe^{is} + q)^n \quad \text{et} \quad e^{\lambda(e^{is}-1)}.$$

Quant à la loi géométrique $\mathcal{G}(p)$, sur $\mathbb{N}^*$, et à la loi négative-binomiale $\mathcal{G}_m(p)$), elles ont pour transformée de Fourier respectives :

$$\frac{pe^{is}}{1 - qe^{is}} \quad \text{et} \quad \frac{p^m e^{mis}}{(1 - qe^{is})^m}.$$

Heuristique 11.3.1 *(Transformation de Fourier = analyse en fréquences).* La transformation de Fourier d'une fonction f exprime les composantes en fréquences (on dit aussi spectre), indexées par s, de son argument f. La fonction $x \mapsto e^{isx}$ représente une fonction périodique élémentaire, dite *harmonique de fréquence* $2\pi/s$. Il faut beaucoup de spectre pour reconstituer une fonction f constante ou plus généralement « peu périodique » à partir des harmoniques. La transformation de Fourier change la convolution en un produit. Pour tronquer le spectre d'un signal f, il suffit de le convoler avec la transformée de Fourier inverse de la fonction indicatrice de troncature.

11.4 Lois continues

11.4.1 Loi uniforme

La loi uniforme $\mathcal{U}(a,b)$ sur l'intervalle compact non vide $[a,b] \subset \mathbb{R}^n$ a pour densité par rapport à la mesure de Lebesgue dx sur $\mathbb{R}$:

$$\frac{1}{b-a}\,\mathrm{I}_{[a,b]}(x).$$

Elle a pour moyenne $(a+b)/2$ et pour variance $(b-a)^2/12$. Sa transformée de Fourier est :

$$e^{is(a+b)/2}\,\frac{\sin(s(b-a)/2)}{s(b-a)/2}.$$

Tout ceci se généralise sans difficultés à un pavé $[a_1,b_1]\times\cdots\times[a_n,b_n]$ de $\mathbb{R}^n$.

11.4.2 Loi triangulaire

C'est la loi « uniforme » sur le triangle du plan de base $[-a,a]$ et de sommet de coordonnées $(0,1)$. Elle a pour densité par rapport à la mesure de Lebesgue dx sur $\mathbb{R}$:

$$\frac{1}{a}\left(1-\frac{|x|}{a}\right)\mathrm{I}_{[-a,a]}(x),$$

pour moyenne 0, pour variance $a^2/6$ et pour transformée de Fourier :

$$\frac{2(1-\cos(as))}{a^2s^2}.$$

11.4.3 Loi gaussienne (ou normale)

On pourrait définir les lois gaussiennes sur des espaces très généraux (de Banach par exemple), mais cela nous mènerait bien trop loi. La loi gaussienne $\mathcal{N}(m,\Gamma)$, sur $\mathbb{R}^n$, de moyenne $m\in\mathbb{R}^n$ et de matrice de covariance $\Sigma\in \mathrm{SO}(\mathbb{R}^n)$, a pour densité par rapport à la mesure de Lebesgue $dx_1\cdots dx_n$ sur $\mathbb{R}^n$:

$$(2\pi)^{-n/2}|\Sigma|^{-1/2}\,\exp\left(-\frac{1}{2}\,(x-m)^\top\Sigma\,(x-m)\right).$$

Elle a pour transformée de Fourier :

$$\exp\left(i<s,m>-\frac{1}{2}\,<\Sigma\,s,s>\right).$$

Elle a pour propriété de maximiser l'entropie de Shannon définie sur les lois des vecteurs aléatoires à densité f de covariance fixée Σ par :

$$\mathbf{H}(f) := -\int_{\mathbb{R}^n} f(x)\log f(x)\,dx.$$

11.4.4 Loi exponentielle et loi gamma

La loi gamma $\Gamma(a,\lambda)$ de paramètres $(a,\lambda)\in\mathbb{R}_+^*\times\mathbb{R}_+^*$ a pour densité sur $\mathbb{R}$ par rapport à la mesure de Lebesgue dx sur $\mathbb{R}$:

$$\frac{\lambda^a}{\Gamma(a)}\,x^{a-1}\,e^{-\lambda x}\,\mathrm{I}_{\mathbb{R}^+}(x),$$

où $\Gamma(a)$ est la fonction Gamma d'Euler définie par :

$$\Gamma(a) := \int_0^{+\infty} x^{a-1}\, e^{-x}\, dx.$$

On a $\Gamma(n) = (n-1)!$. Parfois, on choisit un paramétrage différent de la loi gamma, en posant $b = 1/\lambda$. On note alors $G(a,b) = \Gamma(a, 1/b)$. Pour $a = 1$, on obtient la loi exponentielle $\mathcal{E}(\lambda)$. La loi gamma $\Gamma(a,\lambda)$ a pour moyenne a/λ, pour variance a/λ^2 et pour transformée de Fourier $(1 - is\lambda^{-1})^{-a}$. La loi $\Gamma(n,\lambda)$ est parfois appelée loi de Erlang. Les lois gamma sont très importantes puisqu'elles apparaissent lorsque l'on fait la somme de v.a.r. i.i.d. de loi exponentielle. Elles constituent donc en quelque sorte l'analogue continu de la loi négative-binomiale, et donnent également la loi du temps de depassement d'un seuil pour un processus de Poisson.

11.4.5 Loi double exponentielle

Également appelée loi de Laplace. Elle a pour densité sur $\mathbb{R}$ par rapport à la mesure de Lebesgue dx sur $\mathbb{R}$:

$$\frac{\lambda}{2}\, e^{-\lambda|x|},$$

où λ est un paramètre dans $\mathbb{R}_+$. Sa moyenne vaut 0, sa variance $2/\lambda^2$ et sa transformée de Fourier est $1/(1 + s^2/\lambda^2)$.

11.4.6 Loi de Dirichlet

La loi de Dirichlet de taille $n \in \mathbb{N}^*$ et de paramètre $\alpha \in (\mathbb{R}_+^*)^n$ est la loi du vecteur aléatoire $(X_1, \ldots, X_n)/(X_1 + \cdots + X_n)$ lorsque les variables aléatoires $X_1, \ldots, X_n$ sont i.i.d. et suivent des lois gamma de paramètres respectifs $(1,\alpha_1), \ldots, (1,\alpha_n)$, cf. [DCD82a, exercice 8.2.15, page 192]. Elle est portée par le simplexe $\mathbb{S}(n,1) := \{x \in \mathbb{R}^n,\ x_i \geqslant 0,\ x_1 + \cdots + x_n = 1\}$. La densité de $(X_1, \cdots, X_{n-1})$ est donnée par :

$$\frac{\Gamma(\alpha_1 + \cdots + \alpha_n)}{\Gamma(\alpha_1) \cdots \Gamma(\alpha_n)} \left(\prod_{i=1}^{n-1} x_i^{\alpha_i - 1} \right) \left(1 - \sum_{i=1}^{n-1} x_i \right)^{\alpha_k - 1}.$$

La loi de Dirichlet de taille n et de paramètre $\alpha = (1, \ldots, 1)$ n'est rien d'autre que la loi uniforme sur le simplexe $\mathbb{S}(n,1)$, cf. section 1.6.1 page 30.

11.4.7 Lois du chi-deux

La loi du $\chi^2(n)$ à n degrès de liberté n'est rien d'autre que la loi de la somme des carrés de n v.a.r. i.i.d. de loi normale centrées réduites sur $\mathbb{R}$. C'est aussi la loi gamma $G(n/2, 2)$ de paramètres $(a,b) := (n/2, 2)$, i.e. la loi gamma $\Gamma(n/2, 1/2)$ de paramètres $(a,\lambda) := (n/2, 1/2)$. La loi du $\chi^2(n)$ intervient dans le test du même nom (cf. 4.2 page 69).

11.4.8 Loi exponentielle et loi de Weibull

La loi de Weibull $W(\alpha,\lambda)$ de paramètres $(\alpha,\lambda) \in \mathbb{R}_+^* \times \mathbb{R}_+^*$ est la loi sur $\mathbb{R}$ dont la densité par rapport à la mesure de Lebesgue dx sur $\mathbb{R}$ est :

$$\alpha\lambda\, x^{\alpha-1}\, \exp\left(-\lambda x^\alpha\right) \mathrm{I}_{\mathbb{R}_+}(x).$$

Elle est plus communément donnée par sa fonction de répartition :

$$F_{\alpha,\lambda}(u) = \left(1 - \exp\left(-\lambda x^\alpha\right)\right) \mathrm{I}_{\mathbb{R}_+}.$$

Sa moyenne vaut $\lambda^{-1/\alpha}\Gamma(1+\alpha^{-1})$ et sa variance $\lambda^{-2/\alpha}\Big(\Gamma(1+2\alpha^{-1}) - \big(\Gamma(1+\alpha^{-1})\big)^2\Big)$. Cette loi intervient en fiabilité des systèmes. Notons que l'on retrouve la loi exponentielle de paramètre λ lorsque $\alpha = 1$:

$$W(1,\lambda) = \mathcal{E}(\lambda).$$

Pour $\alpha = 2$, on parle parfois de loi de Rayleigh.

11.4.9 Loi de Cauchy

C'est la loi sur $\mathbb{R}$ de densité par rapport à la mesure de Lebesgue dx sur $\mathbb{R}$:

$$\frac{c}{\pi}\,\frac{1}{c^2+x^2},$$

où c est un paramètre positif. Elle n'a pas de moyenne et donc pas de variance. Sa transformée de Fourier est $\exp\Big(-c|s|^2\Big)$.

11.4.10 Loi Beta

La loi bêta de paramètres $(a,b) \in \mathbb{R}_+^* \times \mathbb{R}_+^*$ est la loi sur $\mathbb{R}$ de densité par rapport à la mesure de Lebesgue dx sur $\mathbb{R}$:

$$\frac{1}{B(a,b)}\,x^{a-1}(1-x)^{b-1}\,\mathrm{I}_{\{0<x<1\}}(x),$$

où $B(a,b)$ est la fonction bêta d'Euler définie par :

$$B(a,b) := \int_0^1 x^{a-1}(1-x)^{b-1}\,dx = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}.$$

La loi bêta de paramètres (a,b) a pour moyenne $a/(a+b)$ et pour variance $ab(a+b)^{-2}(a+b+1)^{-1}$.

11.4.11 Loi de Pareto

Si $\mathcal{L}(X) = \mathcal{E}(\lambda)$ et si $r \in \mathbb{R}$, la v.a. $r\exp(X)$ suit par définition une loi de Pareto de paramètres λ et r. Sa densité par rapport à la mesure de Lebesgue sur $\mathbb{R}$ vaut :

$$\frac{\lambda\, r^\lambda}{x^{1+\lambda}}\,\mathrm{I}_{\{x>r\}}(x).$$

Elle a pour moyenne $\alpha r/(\alpha-1)$ si $\alpha > 1$ et n'a pas moyenne sinon. Si $\alpha > 2$, elle a pour variance $\alpha r^2(\alpha-1)^{-2}(\alpha-2)^{-1}$, et elle n'a pas de moment d'ordre deux sinon.

11.4.12 Loi log-normale

C'est la loi de l'exponentielle d'une variable aléaroite de loi gaussienne $\mathcal{N}(m,\sigma^2)$. Elle a pour densité par rapport à la mesure de Lebesgue dx sur $\mathbb{R}$:

$$(2\pi\sigma^2)^{-1/2}\,x^{-1}\,\exp\left(-\frac{(\log(x)-m)^2}{2\sigma^2}\right)\mathrm{I}_{\{x>0\}}(x),$$

pour moyenne $\exp\big(m+\sigma^2/2\big)$ et pour variance $\exp\big(2m+\sigma^2\big)\Big(e^{\sigma^2}-1\Big)$.

11.4.13 Loi de Ficher-Snedecor

La loi de Fisher-Snedecor $\mathcal{F}(n,m)$ à n et m degrés de liberté est la loi de

$$\frac{X/n}{Y/m},$$

où X et Y sont indépendantes et ont pour lois respectives $\chi^2(n)$ et $\chi^2(m)$. Elle a pour densité par rapport à la mesure de Lebesgue dx sur $\mathbb{R}$:

$$\frac{\Gamma((n+m)/2)n^{n/2}m^{n/2}}{\Gamma(n/2)\Gamma(m/2)}\,\frac{x^{n/2-1}}{(m+nx)^{(n+m)/2}}.$$

Pour $m>4$, sa moyenne vaut $m/(m-2)$ et sa variance $2m^2(m+n-2)n^{-1}(m-4)^{-1}(m-2)^{-2}$.

11.4.14 Loi de Student

La loi de Student $\mathcal{T}_n$ à n degrés de liberté est la loi de

$$\frac{\sqrt{n}\,X}{\sqrt{Y}},$$

où X et et Y sont indépendantes et ont pour lois respectives $\mathcal{N}(0,1)$ et $\chi^2(n)$. Elle a pour densité par rapport à la mesure de Lebesgue dx sur $\mathbb{R}$:

$$\frac{\Gamma((n+1)/2)}{\Gamma(n/2)\sqrt{n\pi}}\left(1+\frac{x^2}{n}\right)^{-(n+1)/2}.$$

Elle est centrée et sa variance vaut $n/(n-2)$ pour $n>2$.

11.4.15 Loi de Kolmogorov-Smirnov

C'est la loi sur $\mathbb{R}$ de fonction de répartition :

$$\mathbf{F}(u) := \begin{cases} 0 & \text{si } u \leqslant 0, \\ 1+2\displaystyle\sum_{k=1}^{+\infty}(-1)^k e^{-2k^2u^2} & \text{si } u>0. \end{cases}$$

Elle intervient dans le test d'adéquation de Kolmogorov-Smirnov (cf. 4.3 page 72). C'est aussi la loi sur supremum de la valeur absolue d'un pont brownien sur $[0,1]$ (cf. section 2.4 page 52).

11.4.16 Statitique d'ordre, quantiles, médiane

Soit μ une loi de probabilité sur $\mathbb{R}$, de fonction de répartition F. On note $F(x^-)$ sa limite à gauche en x. On dit qu'un réel m est une *médiane* pour μ lorsque $F(m^-)\leqslant 1/2\leqslant F(m)$. Plus généralement, pour tout $\alpha\in]0,1[$, on dit qu'un réel q_α est un *quantile* d'ordre α pour μ lorsque $F(q_\alpha^-)\leqslant\alpha\leqslant F(q_\alpha)$. Pour tout ordre α, un quantile d'ordre α existe toujours, car la fonction de répartition F est croissante, et a pour limite 0 en $-\infty$ et 1 en $+\infty$. La croissance de F entraîne également que l'ensemble des quantiles d'ordre α est un intervalle. Si q est un quantile, l'intervalle des quantiles du même ordre est réduit à $\{q\}$ si et seulement si $\mu(\{q_\alpha\})=0$, et alors F est continue en α (on rappelle qu'une fonction de répartition est toujours continue à droite). Souvent, on appelle encore quantile d'ordre α le milieu de l'intervalle des quantiles d'ordre α, ce qui permet de parler *du* quantile d'ordre α, même quand il y en a plusieurs. Enfin, lorsque la loi μ possède une densité, sa fonction de répartition est continue et les intervalles quantiles sont tous des singletons.

On parle de *déciles* pour les quantiles d'ordre 1/10, de *quartiles* pour ceux d'ordre 1/4, etc. On parle aussi parfois de *percentiles* pour les quantiles. Une médiane est un quantile d'ordre 1/2. La fonction Matlab `median` permet d'obtenir la médiane empirique d'un échantillon (i.e. la médiane de la fonction de répartition empirique), alors que la fonction Stixbox `quantile` permet d'obtenir un quantile empirique d'ordre quelconque d'un échantillon.

Si $(x_1, \dots, x_n)$ est un vecteur de $\mathbb{R}^n$, on note $(x_{(1)}, \dots, x_{(n)})$ le vecteur de $\mathbb{R}^n$ obtenu en réordonnant de façon croissante les composantes de x. Si $X := (X_1, \dots, X_n)$ est un vecteur aléatoire de $\mathbb{R}^n$, on appelle statistique d'ordre de X le vecteur aléatoire $\tilde{X} := (X_{(1)}, \dots, X_{(n)})$ obtenu en réordonnant les coordonnées de X à ω fixé. La fonction Matlab `sort` permet de trier de façon croissante un vecteur.

Si $X := (X_1, \dots, X_n)$ est un échantillon d'une loi μ sur $\mathbb{R}$, alors la médiane empirique n'est rien d'autre que $X_{((n+1)/2)}$ si n est impair et $(X_{((n/2)} + X_{((1+n/2)})/2$ si n est pair. De manière générale, si α n'est pas un multiple de $1/n$, le quantile d'ordre α empirique vaut $(X_{([n\alpha])} + X_{([n\alpha]+1)})/2$. Enfin, on peut montrer que si μ a une densité f par rapport à la mesure de Lebesgue, alors la loi de $X_{(i)}$ a pour densité :

$$f_i(x) = \frac{n!}{(i-1)!(n-i)!} F^{i-1}(x)(1-F(x))^{n-i} f(x),$$

où F est la fonction de répartition de μ. On retrouve bien f pour $n = 1$.

Si $x := (x_1, \dots, x_n) \in \mathbb{R}^n$, on appelle *rang* de x_i dans la suite x la quantité

$$r_i(x) := \sum_{j=1}^{n} \mathrm{I}_{x_j \leqslant x_i},$$

qui représente le nombre de composantes de x qui sont inférieures ou égales à x_i. Il peut arriver que $r_i = r_j$, mais on a toujours $x_{r_i(x)} = x_{(i)}$. Lorsque les composantes de x sont toutes différentes, l'application $r : i \in \{1, \dots, n\} \mapsto r_i(x)\{1, \dots, n\}$ est une permutation. Le vecteur $r(x) := (r_1(x), \dots, r_n(x))$ est appelé *vecteur rang* de x. On définit naturellement le vecteur aléatoire rang $R(X)$ d'un vecteur aléatoire $X := (X_1, \dots, X_n)$ par $R(X)(\omega) := r(X(\omega))$. On montre, cf. [DCD82a, thm 4.4.29 page 107] que pour un échantillon $X := (X_1, \dots, X_n)$ d'une loi μ sur $\mathbb{R}$, continue :

- le vecteur rang $R(X)$ est presque-sûrement une permutation
- le vecteur rang $R(X)$ suit la loi uniforme sur l'ensemble des permutations de $\{1, \dots, n\}$
- la statistique d'ordre $\tilde{X}$ de X et son rang $R(X)$ sont indépendants
- la statistique d'ordre $\tilde{X}$ a pour loi la loi trace de $n!\mu \otimes \cdots \otimes \mu$ sur $\{x \in \mathbb{R}^n,\ x_1 \leqslant \cdots \leqslant x_n\}$.

11.5 Convolutions & convergences

On a les propriétés suivantes :

1. $\mathrm{Bern}(p, \{0,1\})^{*n} = \mathcal{B}(n,p)$.
2. $\mathcal{G}(p)^{*m} = \mathcal{G}_m(p)$.
3. $\mathcal{G}_{m_1}(p) * \cdots * \mathcal{G}_{m_n}(p) = \mathcal{G}_{m_1+\cdots+m_n}(p)$.
4. $\mathcal{E}(\lambda)^{*n} = \Gamma(n,\lambda)$.
5. $\mathcal{N}(m_1,\Sigma_1) * \cdots * \mathcal{N}(m_n,\Sigma_n) = \mathcal{N}(m_1+\cdots+m_n, \Sigma_1+\cdots+\Sigma_n)$.
6. $\mathcal{P}(\lambda_1) * \cdots * \mathcal{P}(\lambda_n) = \mathcal{P}(\lambda_1+\cdots+\lambda_n)$.
7. Si $\mathcal{L}(X) = \mathcal{N}(0, \mathbf{I}_n)$ alors $\mathcal{L}(X_1^2+\cdots+X_n^2) = G(n/2,2) = \chi^2(n)$.
8. Si $\mathcal{L}((X,Y)) = \Gamma(a,1) \otimes \Gamma(b,1)$ alors $\mathcal{L}(X/(X+Y)) = \mathrm{Beta}(a,b)$.
9. Si $\mathcal{L}(X) = \Gamma(1,\lambda) = \mathcal{E}(\lambda)$ alors $\mathcal{L}(X^{-\alpha}) = W(\alpha,\lambda)$.

10. Si $\mathcal{L}((X,Y)) = \chi^2(n) \otimes \chi^2(m)$, alors $\mathcal{L}((mX)/(nY)) = \mathcal{F}(n,m)$.
11. Si $\mathcal{L}((X,Y)) = \mathcal{N}(0,1) \otimes \chi^2(n)$, alors $\mathcal{L}(\sqrt{n}X/\sqrt{Y}) = \mathcal{T}(n)$.
12. Si $\mathcal{L}((X_n, n \in \mathbb{N}^*)) = \mathcal{E}(\lambda)^{\mathbb{N}^*}$ et si $X_0 := 0$ et

$$N_t := \sum_{n=0}^{+\infty} \mathrm{I}_{[X_0, X_0+\cdots+X_n[}(t) = \inf\{n \in \mathbb{N},\ X_0 + \cdots + X_n > t\},$$

alors pour tout $t \in \mathbb{R}_+$, $\mathcal{L}(N_t) = \mathcal{P}(\lambda t)$ (processus de Poisson simple d'intensité λ).

13. $\mathcal{B}(n, \lambda/n) \xrightarrow[n\to+\infty]{\text{étr.}} \mathcal{P}(\lambda)$

Remarque 11.5.1. Les lois géométrique et de Bernoulli sont les analogues discrètes des lois exponentielle et gaussienne respectivement.

1. $X_n \xrightarrow[n\to+\infty]{\mathbb{P}} X$
   (a) ssi $(\forall \varepsilon > 0)\ \mathbb{P}(\{|X_n - X| \geqslant \varepsilon\}) \xrightarrow[n\to+\infty]{} 0$
   (b) ssi $\mathbf{E}(|X_n - X| \wedge 1) \xrightarrow[n\to+\infty]{} 0$
   (c) ssi $\mathbf{E}(|X_n - X|/(1 + |X_n - X|)) \xrightarrow[n\to+\infty]{} 0$
   (d) ssi de toute sous suite $(X_{n_k}, k \in \mathbb{N}^*)$, on peut extraire une sous-suite qui converge p.s. vers X
2. $X_n \xrightarrow[n\to+\infty]{\mathcal{L}} \mu$ ssi $\mathcal{L}(X_n) \xrightarrow[n\to+\infty]{\text{étr.}} \mu$.
3. $\mu_n \xrightarrow[n\to+\infty]{\text{étr.}} \mu$ (avec μ_n et μ probas)
   (a) ssi $(\forall f \in \mathcal{C}_b)\ \mathbf{E}_{\mu_n}(f) \xrightarrow[n\to+\infty]{} \mathbf{E}_\mu(f)$
   (b) ssi $(\forall s \in \mathbb{R})\ \mathbf{E}_{\mu_n}\left(e^{is\bullet}\right) \xrightarrow[n\to+\infty]{} \mathbf{E}_\mu\left(e^{is\bullet}\right)$
   (c) ssi $(\forall t \in \mathrm{Cont}(\mathrm{F}_\mu))\ F_{\mu_n}(t) \xrightarrow[n\to+\infty]{} F_\mu(t)$
4. La famille de v.a. $(X_i, i \in I) \subset \mathbf{L}^1$ est uniformément intégrable (UI)
   (a) ssi $\sup_{i\in I} \mathbf{E}\left(|X_i|\mathrm{I}_{\{|X_i|>c\}}\right) \xrightarrow[c\to+\infty]{} 0$
   (b) ssi[1] $\sup_{i\in I} \mathbf{E}(|X_i|) < +\infty$ et $(\forall \varepsilon > 0)(\exists \delta > 0)(\forall A)(\mathbb{P}(A) \leqslant \delta \Rightarrow \sup_{i\in I} \mathbf{E}(|X_i|\mathrm{I}_A) \leqslant \varepsilon)$
   (c) ssi[2] $\exists \Psi : \mathbb{R}_+ \to \mathbb{R}_+$ telle que $\sup_{i\in I} \mathbf{E}(\Psi(|X_i|)) < +\infty$ et $\Psi(x)/x \xrightarrow[x\to+\infty]{} +\infty$.
5. Une famille bornée dans $\mathbf{L}^1$ est UI.
6. Soit $(X_n, n \in \mathbb{N}^*) \subset \mathbf{L}^1$, il y a équivalence entre :
   (a) $X_n \xrightarrow[n\to+\infty]{\mathbf{L}^1} X$ avec $X \in \mathbf{L}^1$
   (b) $(X_n, n \in \mathbb{N}^*)$ est UI et $X_n \xrightarrow[n\to+\infty]{\mathbb{P}} X$
7. Si $X_n \xrightarrow[n\to+\infty]{\mathbf{L}^p} X$ avec $p \geqslant 1$, alors de toute sous-suite $(X_{n_k}, k \in \mathbb{N}^*)$, on peut extraire une sous-suite qui converge p.s. vers X
8. Si $X_n \xrightarrow[n\to+\infty]{\mathcal{L}} \mu$ alors[3] $s \mapsto \mathbf{E}\left(e^{isX_n}\right)$ converge vers $s \mapsto \mathbf{E}_\mu\left(e^{is\bullet}\right)$ uniformément sur tout compact

[1] Critère « epsilon-delta » ou encore d'équi-intégrabilité.
[2] Critère de Lavallée-Poussin. cf. [Bor95, p. 10]
[3] Théorème de continuité de Paul Lévy, cf. [Bor95, p. 32].

9. Soit $(\mu_n, n \in \mathbb{N}^*)$ des probas. Si $s \mapsto \mathbf{E}_{\mu_n}\left(e^{is\bullet}\right)$ converge simplement vers $s \mapsto G(s)$ continue en 0 alors[4] G est la transformée de Fourier d'une loi μ et $\mu_n \xrightarrow[n\to+\infty]{\text{étr.}} \mu$.
10. Soit $(X_n, n \in \mathbb{N}^*)$ v.a.r. indép. et $S_n := X_1 + \cdots + X_n$. Il y a équivalence entre[5] :
    (a) $(S_n, n \in \mathbb{N}^*)$ converge p.s.
    (b) $(S_n, n \in \mathbb{N}^*)$ converge en $\mathbb{P}$
    (c) $(S_n, n \in \mathbb{N}^*)$ converge en $\mathcal{L}$

    De plus[6], si $c > 0$ et $Y_n := X_n \mathrm{I}_{\{|X_n| \leqslant c\}}$ alors $(S_n, n \in \mathbb{N}^*)$ converge p.s. ssi on a à la fois :
    (a) $\sum_{n=1}^{+\infty} \mathbb{P}(\{X_n \neq Y_n\}) < +\infty$
    (b) $\sum_{n=1}^{+\infty} \mathbf{E}(Y_n) < +\infty$
    (c) $\sum_{n=1}^{+\infty} \mathbf{Var}(Y_n) < +\infty$
11. Si $X_n \xrightarrow[n\to+\infty]{\text{p.s}} X$ et $Y_n \xrightarrow[n\to+\infty]{\text{p.s}} Y$ et $(a, b) \in \mathbb{R}^2$ alors $aX_n + bY_n \xrightarrow[n\to+\infty]{\text{p.s}} aX + bY$ et $X_nY_n \xrightarrow[n\to+\infty]{\text{p.s}} XY$
12. L'espace $\mathbf{L}^0$ des fonctions mesurables muni de la topologie de la CV en $\mathbb{P}$ associée à la distance $d(X, Y) := \mathbf{E}(|X - Y|/(1 + |X - Y|))$ est un espace de Banach[7].
13. Si $X_n \xrightarrow[n\to+\infty]{\mathbb{P}} X$ et $Y_n \xrightarrow[n\to+\infty]{\mathbb{P}} Y$ alors $X_nY_n \xrightarrow[n\to+\infty]{\mathbb{P}} XY$
14. Si $X_n \xrightarrow[n\to+\infty]{\mathbf{L}^1} X$ et $Y_n \xrightarrow[n\to+\infty]{\mathbb{P}} Y$ et $(Y_n, n \in \mathbb{N}^*)$ unif. bornée alors $X_nY_n \xrightarrow[n\to+\infty]{\mathbf{L}^1} XY$
15. Si $(X_n, Y_n) \xrightarrow[n\to+\infty]{\mathcal{L}} (X, Y)$ alors $X_n + Y_n \xrightarrow[n\to+\infty]{\mathcal{L}} X + Y$ et $X_n \xrightarrow[n\to+\infty]{\mathcal{L}} X$ et $Y_n \xrightarrow[n\to+\infty]{\mathcal{L}} Y$
16. Si $X_n \xrightarrow[n\to+\infty]{\mathcal{L}} X$ et $Y_n \xrightarrow[n\to+\infty]{\mathcal{L}} Y$ et $(X_n, Y_n, n \in \mathbb{N}^*)$ indép. alors X et Y indép. et $X_n + Y_n \xrightarrow[n\to+\infty]{\mathcal{L}} X + Y$

$$
\begin{array}{ccccc}
 & & \text{CV } \mathbf{L}^{p>1} & & \\
 & & \Downarrow & & \\
 & & \text{CV } \mathbf{L}^1 & & \\
 & & \Downarrow & & \\
\text{CV p.s.} & \Rightarrow & \text{CV en } \mathbb{P} & \Rightarrow & \text{CV en } \mathcal{L}
\end{array}
$$

$$
\text{CV } \mathbf{L}^1 \underset{\text{sous suites}}{\Longrightarrow} \text{CV p.s.}
$$

$$
\text{CV } \mathbb{P} \underset{\text{sous suites}}{\Longrightarrow} \text{CV p.s.}
$$

$$
\text{CV } \mathbb{P} \underset{\text{si UI}}{\Longrightarrow} \text{CV } \mathbf{L}^1
$$

FIG. 11.1 – Relations entre les différents types de convergences de v.a.

[4] Théorème de continuité de Paul Lévy, cf. [Bor95, p. 32].
[5] Théorème de Paul Lévy, cf. [Bor95, p. 83]
[6] Théorème des trois séries de Kolmogorov, cf. [Bor95, p. 85]
[7] Donc un espace vectoriel normé, d'où l'additivité des suites convergentes en $\mathbb{P}$.

Loi	Moyenne	Variance	Transformée de Fourier en $s \in \mathbb{C}$
δ_a	a	0	e^{ias}
Ber. $q\delta_0 + p\delta_1$	p	pq	$1 + pe^{is}$
Ber. $q\delta_{-1} + p\delta_1$	$q - p$	pq	$qe^{-is} + pe^{is}$
Unif. sur $\{a_1, \cdots, a_n\}$	$(a_1 + \cdot \cdots + a_n)/n$	$(\sum_k a_k^2)/n - (\sum_k a_k)^2/n^2$	$\sum_k e^{ia_k s}/n$
$\mathcal{B}(n,p)$	np	npq	$(1 + pe^{is})^n$
Géom(p) (sur $\mathbb{N}$)	q/p	q/p^2	
$\mathcal{G}(p)$ (sur $\mathbb{N}^*$)	$1/p$	q/p^2	$pe^{is}\big(1 - qe^{is}\big)^{-1}$
$\mathcal{G}_m(p)$	m/p	q/p^2	$p^m e^{mis}\big(1 - qe^{is}\big)^{-m}$
$\mathcal{P}(\lambda)$	λ	λ	$\exp\big(\lambda(e^{is} - 1)\big)$
Unif. $\mathcal{U}(a,b)$	$(a+b)/2$	$(b-a)^2/12$	$2s^{-1}(b-a)^{-1}\exp\left(is(a+b)/2\right)\sin(s(b-a)/2)$
$\mathcal{E}(\lambda)$	$1/\lambda$	$1/\lambda^2$	$\big(1 - is\lambda^{-1}\big)^{-1}$
$\Gamma(a,\lambda)$	a/λ	a/λ^2	$\big(1 - is\lambda^{-1}\big)^{-a}$
$G(a,b)$	ab	ab^2	$(1 - ibs)^{-a}$
$\mathcal{N}(m,\Sigma)$	m	Σ	$\exp\left(i < m, s > - < \Sigma s, s > /2\right)$
$\chi^2(n)$	n	$2n$	$(1 - 2is)^{-n/2}$
$W(\alpha,\lambda)$	$\lambda^{-1/\alpha}\Gamma(1 + \alpha - 1)$	$\lambda^{-2/\alpha}\Big(\Gamma(1 + 2\alpha^{-1}) - \big(\Gamma(1 + \alpha^{-1})\big)^2\Big)$	
$\mathcal{F}(n,m)$	$m/(m-2)$ si $m > 4$	$2m^2(m+n-2)n^{-1}(m-4)^{-1}(m-2)^{-2}$	
$\mathcal{T}(n)$	0	$n/(n-2)$ si $n > 2$	

TAB. 11.1 – Tableau récapitulatif de quelques lois usuelles

© 2001 Dj. CHAFAÏ. GFDL copyleft. Version DJC-AMOPS-GERMINAL du 2001-12-05, compilée par LaTeX2e le 5 décembre 2001 à 19:56.

Loi μ	Fonction de répartition $F_\mu(t) := \mu([t, +\infty[)$
Discrète $\sum_{n\geqslant 0} p_n \delta_{a_n}$ $(a_n < a_{n+1})$	$\sum_{n\geqslant 0} p_n \mathrm{I}_{[a_n, a_{n+1}[}(t)$
Unif. $\mathcal{U}(a,b)$	$(b-a)^{-1}(t-a)\mathrm{I}_{[a,b]}(t) + \mathrm{I}_{]b,+\infty[}(t)$
$\mathcal{E}(\lambda)$	$(1-\exp(-\lambda t))\mathrm{I}_{\mathbb{R}_+}(t)$
$W(\alpha,\lambda)$	$(1-\exp(-\lambda t^\alpha))\mathrm{I}_{\mathbb{R}_+}(t)$
$\mathcal{N}(0,1)$	$\Phi(t) \sim_{+\infty} 1 - t^{-1}\exp\left(-t^2/2\right)$
Kolmogorov-Smirnov	$\left(1 + 2\sum_{k=1}^{+\infty}(-1)^k e^{-2k^2u^2}\right)\mathrm{I}_{\mathbb{R}_+^*}(t)$

TAB. 11.2 – Quelques fonctions de répartition

© 2001 Dj. CHAFAÏ. GFDL copyleft. Version DJC-AMOPS-GERMINAL du 2001-12-05, compilée par LaTeX 2ε le 5 décembre 2001 à 19:56.

Chapitre 12

Informations utiles (et volatiles)

12.1 Liste des leçons d'oral

200001. Loi des grands nombres ; application à l'estimation.
200002. Le théorème de la limite centrale et son utilisation.
200003. Convergence d'une chaîne de Markov vers une loi stationnaire.
200004. Décrire une méthode probabiliste ou déterministe pour le calcul approché d'une intégrale.
200005. Théorème d'arrêt pour une martingale ; applications.
200006. Martingales : convergence et applications.
200007. Vecteurs aléatoires gaussiens.
200008. Modèle linéaire gaussien en statistique.
200009. Test d'adéquation du χ^2.
200010. Tests statistiques : principes et exemples.
200011. Intervalles de confiance : principe et exemples.
200012. Écarts à la loi des grands nombres ; événements de faibles probabilité, applications.
200013. Fonction de répartition empirique : comportement asymptotique, usage en statistique.
200014. Exemples de chaînes de Markov récurrentes ou transientes à espace d'état au plus dénombrable.
200015. Évolution de la taille d'une population lorsque la loi de reproduction est homogène.
200016. Modélisation d'une duré de vie, renouvellement.
200017. Simulation de variables aléatoires; applications.
200101. Loi des grands nombres ; applications à l'estimation.
200102. Le théorème de la limite centrale ; application(s).
200103. Décrire une méthode probabiliste pour le calcul approché d'une intégrale.
200104. Espérence conditionnelle et applications (par exemple aux martingales).
200105. Exemples d'utilisation(s) de l'espérence conditionnelle en modélisation.
200106. Exemple d'utilisation(s) des martingales.
200107. Vecteur aléatoires gaussiens, utilisation.
200108. Modèle linéaire gaussien en statistique.
200109. Test du Khi-deux.
200110. Tests statistiques : principes et exemples.
200111. Intervalles de confiance : principe et exemples.

http://www.education.gouv.fr/siac/siac2/	Infos concours au Ministère
http://www.math.u-psud.fr/~agreg/	Infos en provenance du jury (Orsay)
http://smf.emath.fr/Enseignements/	Rubrique enseignements de la SMF
http://www.umpa.ens-lyon.fr/~agregmath/	Préparation de l'ENS-Lyon
http://www.maths.lth.se/matstat/stixbox/	Page officielle de Stixbox
http://www.octave.org/	Page officielle d'Octave
http://www.mathworks.com/products/matlab/	Page officielle de Matlab
http://www-rocq.inria.fr/scilab/	Page officielle de Scilab
http://www.mupad.de/	Page officielle de Mupad
http://www.math-info.univ-paris5.fr/~ycart/polys/	Polycopiés de Bernard Ycart
http://www.cmla.ens-cachan.fr/Utilisateurs/scopos/ruget.html	Page du livre « Mathématiques en situation »
http://www.cmla.ens-cachan.fr/Utilisateurs/scopos/collection.html	Page de la collection Scopos
http://www.math.u-psud.fr/~lichnew/	Page d'Alain Lichnewsky
http://www.dma.ens.fr/~ldumas/	Page de Laurent Dumas
http://www.univ-angers.fr/cufco/matlab/	Livre de Eric Schrafstetter
http://www.lsp.ups-tlse.fr/Chafai/Agreg/	Page de l'auteur pour l'agrégation
http://www.lsp.ups-tlse.fr/Bakry/	Cours de Maîtrise de Dominique Bakry
http://www.lsp.ups-tlse.fr/Fp/Letac/	Cours de DEUG de Gérard Letac
http://www.lsp.ups-tlse.fr/Besse/	Cours de modélisation statistique de Ph. Besse

TAB. 12.1 – Quelques liens sur la toile mondiale

© 2001 Dj. CHAFAÏ. GFDL copyleft. Version DJC-AMOPS-GERMINAL du 2001-12-05, compilée par LaTeX2e le 5 décembre 2001 à 19:56.

200112. Application de la transformée de Laplace (par exemple, à l'inégalité de Cramér-Chernoff, au problème de ruine du joueur, etc)

200113. Usage de la fonction de répartition empirique en statistique.

200114. Utilisation d'une loi exponentielle (par exemple, pour l'étude d'une durée de vie, etc).

200115. Simulation de variables et de vecteurs aléatoires; applications.

200116. Sommes de variables aléatoires; application (par exemple aux marches aléatoires, à la ruine du joueur etc.)

200117. Exemple de chaînes de Markov récurrentes ou transciente à espace d'états au plus dénombrable.

200118. Convergence d'une chaîne de Markov vers une loi stationnaire.

200119. Utilisation de la transformée de Laplace, de la fonction génératrice (par exemple, marches aléatoires, ruine du joueur, branchements, etc).

200120. Utilisations de la loi de Poisson en modélisation (par exemple, processus de comptage, etc).

12.2 Thèmes applicatifs 2001

- Contrôle de qualité et fiabilité.
- Géométrie effective et appliquée.
- Modèles en économie et finance.
- Propagation d'ondes.
- Télécommunications.

12.3 Textes 2001

601. Un modèle d'actifs financiers à longue mémoire

602. Le coussin financier

603. Convergence en loi des prix des actifs financiers

604. Minimisation du risque quadratique en finance

605. Instants de records

606. Aversion au risque dans les choix économiques

607. Processus ARMA et ARFIMA en finance

608. Processus à volatilité stochastique

609. Problème d'arrêt optimal et application aux options américaines.

610. Gestion optimale de portefeuilles dans un marché financier discret.

611. Modélisation ARCH

612. Modèle binomial

613. Rendement d'une ligne de transmission

614. Files d'attente

615. Indépendance conditionnelle et réseaux bayésiens

616. Fonctions de Lyapounov et stabilité de réseaux de télécommunications

617. Étude d'une liaison satellite avec erreurs aléatoires

618. Modélisation d'un central téléphonique avec répétitions d'appels

619. Réduction de dimension par analyse en composantes principales
620. Compression de données par quantification
621. Modélisation du système ALOHA
622. Codage et entropie
623. Modélisation du tracé d'un réseau de communication
624. Détection d'un signal en télécommunications
625. Indice de Pareto
626. Modèle de Cox et statistique bayésienne en fiabilité
627. Processus de Poisson et valeurs records
628. Echantillonnage préférentiel dans les calculs de fiabilité
629. Approximation exponentielle de la durée de vie d'un système fiable
630. Analyse d'un système multicomposants
631. Politique de remplacement à l'âge tau
632. Inférence statistique pour les quantiles

12.4 Information en provenance du jury

Ce qui suit est extrait du site http://www.math.u-psud.fr/~agreg/ :

12.4.1 Utilisation des ordinateurs lors de l'épreuve Modélisation

« *Après avoir tiré une feuille comportant un titre de leçon et un titre de texte, le candidat rejoint la salle de préparation. À l'accueil, il reçoit un nom de login et un mot de passe ainsi que le texte qu'il a tiré au sort. Le candidat est placé dans la salle de préparation où une machine est mise à sa disposition. Il devra indiquer au responsable son choix du système d'exploitation (environnement Linux ou Windows). Une fois le système configuré et lancé par le responsable, des icones permettront au candidat de lancer les logiciels qu'il souhaite utiliser. La documentation sera aussi disponible en ligne. Une imprimante est mise à la disposition des candidats ; il est conseillé de ne pas attendre le dernier moment pour imprimer.*

Environ un quart d'heure avant la fin de sa préparation, il sera demandé au candidat de sauvegarder ses fichiers de travail (la manipulation de sauvegarde sera expliquée). Le responsable de la salle de préparation pourra être appelé en cas de problème de fonctionnement du système ; en revanche, il ne donnera pas d'explications sur le fonctionnement des logiciels mathématiques eux-mêmes qui sont supposés connus par le candidat. »

« *La présentation du travail sur une machine devant le jury se fera à partir des fichiers sauvegardés. Cette machine aura la même configuration que celle utilisée pour la préparation et elle sera équipée d'un dispositif permettant à l'ensemble du jury d'en visionner l'écran.* »

12.4.2 Matériels et Logiciels disponibles

Matériel : PC avec processeur Pentium et clavier français.
Logiciels système : Windows 98 SE, Red Hat Linux 6.1 + fvwm95
Logiciels applicatifs :
- Maple Version V 5.1,
- Matlab Version 5.2 avec Stixbox,
- MuPAD Pro Version 1.4.2,
- Scilab Version 2.5 avec Stixbox

Autres logiciels sous Linux : GNU Emacs, vi et Netscape.
Autres logiciels sous Windows : GNU Emacs, Notepad et Internet Explorer.

12.5 Boite à outils Stixbox pour Matlab

```
%
% A rudimentary statistics toolbox.
% Version 1.10, 9-Sep-98
% Copyright (c) Anders Holtsberg.
% Comments and suggestions to andersh@maths.lth.se.
%
% Revision 01-10-98 Mathematique Universite de Paris-Sud
%
% Distribution functions.
%   dbeta     - Beta density function.
%   dbinom    - Binomial probability function.
%   dchisq    - Chisquare density function.
%   df        - F density function.
%   dgamma    - Gamma density function.
%   dhypg     - Hypergeometric probability function.
%   dnorm     - Normal density function.
%   dt        - Student t density function.
%
%   pbeta     - Beta distribution function.
%   pbinom    - Binomial cumulative probability function.
%   pchisq    - Chisquare distribution function.
%   pf        - F distribution function.
%   pgamma    - Gamma distribution function.
%   phypg     - Hypergeometric cumulative probab. function.
%   pks       - Kolmogorov Smirnov distribution function.
%   pnorm     - Normal distribution function.
%   pt        - Student t cdf.
%
%   qbeta     - Beta inverse distribution function.
%   qbinom    - Binomial inverse cdf.
%   qchisq    - Chisquare inverse distribution function.
%   qf        - F inverse distribution function.
%   qgamma    - Gamma inverse distribution function.
%   qhypg     - Hypergeometric inverse cdf.
%   qnorm     - Normal inverse distribution function.
%   qt        - Student t inverse distribution function.
%
%   rbeta     - Rand. num. , beta distribution.
%   rbinom    - Rand. num. , binomial distribution.
%   rchisq    - Rand. num. , chisquare distribution.
%   rexpweib  - Rand. num. , exp and weibull distributions.
%   rf        - Rand. num. , F distribution
%   rgamma    - Rand. num. , gamma distribution.
%   rgeom     - Rand. num. , geometric distribution.
%   rhypg     - Rand. num. , hypergeometric distribution.
%   rnorm     - Norm. rand. num. (use randn instead).
%   rpoiss    - Rand. rand. num. , poisson distribution.
%   rt        - Rand. num. , student t distribution.
%
```

```
%simulation of random variate  by  reject method
%   rjbinom.m - Rand. num. , binomial distribution.
%   rjgamma.m - Rand. numbers , gamma distribution.
%   rjpoiss.m - Rand. rand. num. , poisson distribution.
%
%Logistic regression.
%   logitfit  - Fit a logistic regression model.
%   lodds     - Log odds function.
%   loddsinv  - Inverse of log odds function.
%
% Various functions.
%   bincoef   - Binomial coefficients.
%   getdata   - Some famous multivariate data sets.
%   quantile  - Empirical quantile (percentile).
%
% Tests, confidence intervals, and model estimation.
%   cmpmod    - Compare small linear model versus large one.
%   ciquant   - Nonparametric confidence interval for quantile.
%   kstwo     - Smirnov test for two samples.
%   lsfit     - Fit a least squares model.
%   lsselect  - Select a predictor subset for regression.
%   test1n    - Tests and confid. interv., 1 norm. sample.
%   test2n    - Tests and confid. interv., 2 norm. samples.
%
% Graphics.
%   qqnorm    - Normal probability paper.
%   qqplot    - Plot empirical quantile vs empirical quantile.
%   linreg    - Linear or polynomial regression, including plot.
%   histo     - Plot a histogram (alternative to hist).
%   plotsym   - Plot with symbols.
%   plotdens  - Draw a nonparametric density estimate.
%   identify  - Identify points on a plot by mouse clicks.
%   pairs     - Pairwise scatter plots.
```

12.6 Boite à outils statistique de Matlab

Cette boite à outils de Matlab, bien plus complète que Stixbox, n'est pas sensée être disponible le jour de l'oral malheureusement...

```
  Statistics Toolbox.
  Version 3.0 (R12.1) 1-Sep-2000

  Distributions.
   Parameter estimation.
    betafit     - Beta parameter estimation.
    binofit     - Binomial parameter estimation.
    expfit      - Exponential parameter estimation.
    gamfit      - Gamma parameter estimation.
    mle         - Maximum likelihood estimation (MLE).
    normfit     - Normal parameter estimation.
    poissfit    - Poisson parameter estimation.
```

```
 raylfit     - Rayleigh parameter estimation.
 unifit      - Uniform parameter estimation.
 weibfit     - Weibull parameter estimation.

Probability density functions (pdf).
 betapdf     - Beta density.
 binopdf     - Binomial density.
 chi2pdf     - Chi square density.
 exppdf      - Exponential density.
 fpdf        - F density.
 gampdf      - Gamma density.
 geopdf      - Geometric density.
 hygepdf     - Hypergeometric density.
 lognpdf     - Lognormal density.
 nbinpdf     - Negative binomial density.
 ncfpdf      - Noncentral F density.
 nctpdf      - Noncentral t density.
 ncx2pdf     - Noncentral Chi-square density.
 normpdf     - Normal (Gaussian) density.
 pdf         - Density func. for a specified distrib.
 poisspdf    - Poisson density.
 raylpdf     - Rayleigh density.
 tpdf        - T density.
 unidpdf     - Discrete uniform density.
 unifpdf     - Uniform density.
 weibpdf     - Weibull density.

Cumulative Distribution functions (cdf).
 betacdf     - Beta cdf.
 binocdf     - Binomial cdf.
 cdf         - Specified cumulative distrib. function.
 chi2cdf     - Chi square cdf.
 expcdf      - Exponential cdf.
 fcdf        - F cdf.
 gamcdf      - Gamma cdf.
 geocdf      - Geometric cdf.
 hygecdf     - Hypergeometric cdf.
 logncdf     - Lognormal cdf.
 nbincdf     - Negative binomial cdf.
 ncfcdf      - Noncentral F cdf.
 nctcdf      - Noncentral t cdf.
 ncx2cdf     - Noncentral Chi-square cdf.
 normcdf     - Normal (Gaussian) cdf.
 poisscdf    - Poisson cdf.
 raylcdf     - Rayleigh cdf.
 tcdf        - T cdf.
 unidcdf     - Discrete uniform cdf.
 unifcdf     - Uniform cdf.
 weibcdf     - Weibull cdf.

Critical Values of Distribution functions.
```

```
  betainv     - Beta inverse cumulative distrib. func.
  binoinv     - Binomial inverse cumulative distrib. func.
  chi2inv     - Chi square inverse cumulative distrib. func.
  expinv      - Exponential inverse cumulative distrib. func.
  finv        - F inverse cumulative distrib. func.
  gaminv      - Gamma inverse cumulative distrib. func.
  geoinv      - Geometric inverse cumulative distrib. func.
  hygeinv     - Hypergeometric inverse cumulative distrib. func.
  icdf        - Specified inverse cdf.
  logninv     - Lognormal inverse cumulative distrib. func.
  nbininv     - Negative binomial inverse distrib. func.
  ncfinv      - Noncentral F inverse cumulative distrib. func.
  nctinv      - Noncentral t inverse cumulative distrib. func.
  ncx2inv     - Noncentral Chi-square inverse distrib. func.
  norminv     - Normal (Gaussian) inv. cumulative distrib. func.
  poissinv    - Poisson inverse cumulative distrib. func.
  raylinv     - Rayleigh inverse cumulative distrib. func.
  tinv        - T inverse cumulative distrib. func.
  unidinv     - Discrete unif. inv. cumulative distrib. func.
  unifinv     - Uniform inverse cumulative distrib. func.
  weibinv     - Weibull inverse cumulative distrib. func.

Random Number Generators.
  betarnd     - Beta random numbers.
  binornd     - Binomial random numbers.
  chi2rnd     - Chi square random numbers.
  exprnd      - Exponential random numbers.
  frnd        - F random numbers.
  gamrnd      - Gamma random numbers.
  geornd      - Geometric random numbers.
  hygernd     - Hypergeometric random numbers.
  lognrnd     - Lognormal random numbers.
  mvnrnd      - Multivariate normal random numbers.
  mvtrnd      - Multivariate t random numbers.
  nbinrnd     - Negative binomial random numbers.
  ncfrnd      - Noncentral F random numbers.
  nctrnd      - Noncentral t random numbers.
  ncx2rnd     - Noncentral Chi-square random numbers.
  normrnd     - Normal (Gaussian) random numbers.
  poissrnd    - Poisson random numbers.
  random      - Random numbers from specified distribution.
  raylrnd     - Rayleigh random numbers.
  trnd        - T random numbers.
  unidrnd     - Discrete uniform random numbers.
  unifrnd     - Uniform random numbers.
  weibrnd     - Weibull random numbers.

Statistics.
  betastat    - Beta mean and variance.
  binostat    - Binomial mean and variance.
  chi2stat    - Chi square mean and variance.
```

```
  expstat     - Exponential mean and variance.
  fstat       - F mean and variance.
  gamstat     - Gamma mean and variance.
  geostat     - Geometric mean and variance.
  hygestat    - Hypergeometric mean and variance.
  lognstat    - Lognormal mean and variance.
  nbinstat    - Negative binomial mean and variance.
  ncfstat     - Noncentral F mean and variance.
  nctstat     - Noncentral t mean and variance.
  ncx2stat    - Noncentral Chi-square mean and variance.
  normstat    - Normal (Gaussian) mean and variance.
  poisstat    - Poisson mean and variance.
  raylstat    - Rayleigh mean and variance.
  tstat       - T mean and variance.
  unidstat    - Discrete uniform mean and variance.
  unifstat    - Uniform mean and variance.
  weibstat    - Weibull mean and variance.

Descriptive Statistics.
  bootstrp    - Bootstrap statistics for any function.
  corrcoef    - Correlation coefficient.
  cov         - Covariance
  crosstab    - Cross tabulation.
  geomean     - Geometric mean.
  grpstats    - Summary statistics by group.
  harmmean    - Harmonic mean.
  iqr         - Interquartile range.
  kurtosis    - Kurtosis.
  mad         - Median Absolute Deviation.
  mean        - Sample average (in matlab toolbox).
  median      - 50th percentile of a sample.
  moment      - Moments of a sample.
  nanmax      - Maximum ignoring NaNs.
  nanmean     - Mean ignoring NaNs.
  nanmedian   - Median ignoring NaNs.
  nanmin      - Minimum ignoring NaNs.
  nanstd      - Standard deviation ignoring NaNs.
  nansum      - Sum ignoring NaNs.
  prctile     - Percentiles.
  range       - Range.
  skewness    - Skewness.
  std         - Standard deviation ((in matlab toolbox).
  tabulate    - Frequency table.
  trimmean    - Trimmed mean.
  var         - Variance (in matlab toolbox).

Linear Models.
  anova1      - One-way analysis of variance.
  anova2      - Two-way analysis of variance.
  anovan      - n-way analysis of variance.
  aoctool     - Interactive tool for analysis of covariance.
```

```
  dummyvar    - Dummy-variable coding.
  friedman    - Friedman's test (nonparametric two-way anova).
  glmfit      - Generalized linear model fitting.
  kruskalwallis - Kruskal-Wallis test (nonparametric one-way anova).
  leverage    - Regression diagnostic.
  lscov       - Least-squares estimates with known cov. matrix.
  manova1     - One-way multivariate analysis of variance.
  manovacluster - Draw clusters of group means for manova1.
  multcompare - Multiple comparisons of means and other estimates.
  polyconf    - Polynomial eval. and confid. interv. estim.
  polyfit     - Least-squares polynomial fitting.
  polyval     - Predicted values for polynomial functions.
  rcoplot     - Residuals case order plot.
  regress     - Multivariate linear regression.
  regstats    - Regression diagnostics.
  ridge       - Ridge regression.
  robustfit   - Robust regression model fitting.
  rstool      - Multidimensional response surface visualization (RSM).
  stepwise    - Interactive tool for stepwise regression.
  x2fx        - Factor settings matrix (x) to design matrix (fx).

Nonlinear Models
  nlinfit     - Nonlinear least-squares data fitting (Newton).
  nlintool    - Interactive gfx tool for prediction in nonlin. models.
  nlpredci    - Confidence intervals for prediction.
  nlparci     - Confidence intervals for parameters.
  nnls        - Non-negative least-squares.

Cluster Analysis
  pdist       - Pairwise distance between observations.
  squareform  - Square matrix formatted distance.
  linkage     - Hierarchical cluster information.
  dendrogram  - Generate dendrogram plot.
  inconsistent- Inconsistent values of a cluster tree.
  cophenet    - Cophenetic coefficient.
  cluster     - Construct clusters from LINKAGE output.
  clusterdata - Construct clusters from data.

Design of Experiments (DOE)
  cordexch    - D-optimal design (coordinate exchange algorithm).
  daugment    - Augment D-optimal design.
  dcovary     - D-optimal design with fixed covariates.
  ff2n        - Two-level full-factorial design.
  fracfact    - Two-level fractional factorial design.
  fullfact    - Mixed-level full-factorial design.
  hadamard    - Hadamard matrices (orthogonal arrays).
  rowexch     - D-optimal design (row exchange algorithm).

Statistical Process Control (SPC)
  capable     - Capability indices.
  capaplot    - Capability plot.
```

```
  ewmaplot    - Exponentially weighted moving average plot.
  histfit     - Histogram with superimposed normal density.
  normspec    - Plot normal density between specification limits.
  schart      - S chart for monitoring variability.
  xbarplot    - Xbar chart for monitoring the mean.

Principal Components Analysis
  barttest    - Bartlett's test for dimensionality.
  pcacov      - Principal components from covariance matrix.
  pcares      - Residuals from principal components.
  princomp    - Principal components analysis from raw data.

Multivariate Statistics.
  classify    - Linear Discriminant Analysis.
  mahal       - Mahalanobis distance.
  manova1     - One-way multivariate analysis of variance.

Hypothesis Tests.
  ranksum     - Wilcoxon rank sum test (independent samples).
  signrank    - Wilcoxon sign rank test (paired samples).
  signtest    - Sign test (paired samples).
  ztest       - Z test.
  ttest       - One sample t test.
  ttest2      - Two sample t test.

Distribution Testing
  jbtest      - Jarque-Bera test of normality
  kstest      - Kolmogorov-Smirnov test for one sample
  kstest2     - Kolmogorov-Smirnov test for two samples
  lillietest  - Lilliefors test of normality

Nonparametric Testing
  friedman    - Friedman's test (nonparametric two-way anova).
  kruskalwallis - Kruskal-Wallis test (nonparametric one-way anova).
  ranksum     - Wilcoxon rank sum test (independent samples).
  signrank    - Wilcoxon sign rank test (paired samples).
  signtest    - Sign test (paired samples).

Statistical Plotting.
  boxplot     - Boxplots of a data matrix (one per column).
  cdfplot     - Plot of empirical cumulative distribution function.
  fsurfht     - Interactive contour plot of a function.
  gline       - Point, drag and click line drawing on figures.
  gname       - Interactive point labeling in x-y plots.
  gplotmatrix - Matrix of scatter plots grouped by a common variable.
  gscatter    - Scatter plot of two variables grouped by a third.
  lsline      - Add least-square fit line to scatter plot.
  normplot    - Normal probability plot.
  qqplot      - Quantile-Quantile plot.
  refcurve    - Reference polynomial curve.
  refline     - Reference line.
```

```
  surfht      - Interactive contour plot of a data grid.
  weibplot    - Weibull probability plot.

Statistics Demos.
  aoctool     - Interactive tool for analysis of covariance.
  disttool    - GUI tool for exploring probability distribution functions.
  glmdemo     - Generalized linear model slide show.
  polytool    - Interactive graph for prediction of fitted polynomials.
  randtool    - GUI tool for generating random numbers.
  rsmdemo     - Reaction simulation (DOE, RSM, nonlinear curve fitting).
  robustdemo  - Interactive tool to compare robust and least squares fits.

File Based I/O
  tblread     - Read in data in tabular format.
  tblwrite    - Write out data in tabular format to file.
  tdfread     - Read in text and numeric data from tab-delimitted file.
  caseread    - Read in case names.
  casewrite   - Write out case names to file.
```

Bibliographie

[ABC+00] C. Ané, S. Blachère, D. Chafaï, P. Fougères, I. Gentil, F. Malrieu, C. Roberto et G. Scheffer – *Sur les inégalités de Sobolev logarithmiques*, Panoramas et Synthèses, vol. 10, Société Mathématique de France, Paris, 2000. 20

[App96a] D. Applebaum – *Probability and information, an integrated approach*, Cambridge University Press, Cambridge, 1996. 20, 121, 142

[App96b] — , *Probability and information, an integrated approach*, Cambridge University Press, Cambridge, 1996. 49

[Bak] D. Bakry – « Cours de probabilités de maîtrise », http://www.lsp.ups-tlse.fr/Bakry/.

[Bas95] R. F. Bass – *Probabilistic techniques in analysis*, Springer-Verlag, New York, 1995. 34

[BC98] J.-C. Bertein et R. Ceschi – *Processus stochastiques et filtrage de Kalman*, Editions Hermès, Paris, 1998.

[BDY00] B. Bru, W. Doeblin et M. Yor – *Sur l'équation de Kolmogoroff, par W. Doeblin*, Éditions Elsevier, Paris, 2000, C. R. Acad. Sci. Paris Sér. I Math. **331** (2000), Special Issue. 34

[Bes] P. C. Besse – « Cours de modélisation statistique », http://www.lsp.ups-tlse.fr/Besse/.

[BL98] P. Barbe et M. Ledoux – *Probabilités*, De la licence à l'agrégation, Belin, 1998. 25, 49, 95

[Bon95] J.-L. Bon – *Fiabilité des systèmes - Méthodes mathématiques*, Masson, 1995. 86

[Bor95] V. S. Borkar – *Probability theory, an advanced course*, Springer-Verlag, New York, 1995. 46, 105, 106

[Bou86] N. Bouleau – *Probabilités de l'ingénieur, variables aléatoires et simulation*, Hermann, 1986. 18, 21, 40, 95

[CT91] T. M. Cover et J. A. Thomas – *Elements of information theory*, John Wiley & Sons Inc., New York, 1991, A Wiley-Interscience Publication. 19, 20, 121

[CT97] C. Cocozza-Thivent – *Processus stochastiques et fiabilité des systèmes*, Springer-Verlag, Berlin, 1997.

[DCD82a] D. Dacunha-Castelle et M. Duflo – *Probabilités et statistique. Tome 1*, Masson, Paris, 1982, Problèmes à temps fixe. 25, 52, 69, 95, 101, 104, 142

[DCD82b] D. Dacunha-Castelle et M. Duflo – *Exercices de probabilités et statistiques. Tome 1*, Masson, Paris, 1982, Problèmes à temps fixe.

[DCD83] — , *Probabilités et statistiques. Tome 2*, Masson, Paris, 1983, Problèmes à temps mobile.

[Dio97] E. Dion – *Invitation à la théorie de l'information*, Points sciences, Éditions du seuil, 1997. 121

[Dje01] A. Djebbar – *Une histoire de la science arabe*, Points sciences, vol. 144, Éditions du seuil, 2001, Entretien avec Jean Rosmorduc. 121

[Duf96] M. Duflo – *Algorithmes stochastiques*, Springer-Verlag, Berlin, 1996.

[Knu73] D. E. Knuth – *The art of computer programming. Volume 3*, Computer Science and Information Processing, Addison-Wesley, 1973, Sorting and searching.

[Knu75] — , *The art of computer programming*, Computer Science and Information Processing, Addison-Wesley, 1975, Volume 1: Fundamental algorithms.

[Knu76] — , *The state of the art of computer programming*, Computer Science Department, School of Humanities and Sciences, Stanford University, Stanford, Calif., 1976, Errata to *The art of computer programming, Vols. 1 and 2*, Addison-Wesley.

[Knu81] — , *The art of computer programming. Vol. 2*, Computer Science and Information Processing, Addison-Wesley, 1981, Seminumerical algorithms. 18, 21, 140

[KS91] I. Karatzas et S. E. Shreve – *Brownian motion and stochastic calculus*, Springer-Verlag, New York, 1991. 34, 35

[Lap95a] P.-S. Laplace – *Théorie analytique des probabilités. Vol. I*, Éditions Jacques Gabay, Paris, 1995, Introduction: Essai philosophique sur les probabilités. Livre I: Du calcul des fonctions génératrices., Réédition de la quatrième édition de 1819. 121

[Lap95b] — , *Théorie analytique des probabilités. Vol. II*, Éditions Jacques Gabay, Paris, 1995, Livre II: Théorie générale des probabilités. Suppléments., Réédition de la troisième édition de 1820. 121

[Law91] G. F. Lawler – *Intersections of random walks*, Birkhäuser Boston Inc., Boston, MA, 1991. 132

[Let] G. Letac – « Cours de probabilités de premier cycle », http://www.lsp.ups-tlse.fr/Fp/Letac/.

[Lév92] P. Lévy – *Processus stochastiques et mouvement brownien*, Éditions Jacques Gabay, Sceaux, 1992, Suivit d'une note de M. Loève, Réédition de la seconde édition de (1965).

[LL97] D. Lamberton et B. Lapeyre – *Introduction au calcul stochastique appliqué à la finance*, second éd., Ellipses Édition Marketing, Paris, 1997.

[Mil01] X. Milhaud – *Statistique*, De la licence à l'agrégation, Belin, 2001.

[MPB98] L. Mazliak, P. Priouret et P. Baldi – *Martingales et chaînes de Markov*, Hermann, 1998. 81

[NNGG97] E. Nagel, J. Newman, K. Gödel et J.-Y. Girard – *Le théorème de Gödel*, Points - Sciences, Éditions du Seuil, 1997. 122

[Pet95] V. V. Petrov – *Limit theorems of probability theory*, The Clarendon Press Oxford University Press, New York, 1995, Sequences of independent random variables, Oxford Science Publications. 52

[RM96] R. Rashed et R. Morelon (éds.) – *Encyclopedia of the history of Arabic science. Vol. 1–3*, Routledge, London, 1996.

[Rom92] S. Roman – *Coding and information theory*, Springer-Verlag, New York, 1992. 142

[Rom97] — , *Introduction to coding and information theory*, Springer-Verlag, New York, 1997. 142

[Rud75] W. Rudin – *Analyse réelle et complexe*, Masson, Paris, 1975. 95

[Rug00] C. Ruget (éd.) – *Mathématiques en situation – Issues de l'épreuve de modélisation de l'agrégation*, Scopos, Springer, 2000.

[Sap89] G. Saporta – *Probabilités, analyse des données et statistique*, Technip, 1989. 69, 72, 85

[Sin99] S. Singh – *Histoire des codes secrets*, Jean-Claude Latès, 1999. 122

[Spi70] F. Spitzer – *Principes des cheminements aléatoires*, Dunod, Paris, 1970. 132

[TDM93] R. Tomassone, C. Dervin et J.-P. Masson – *Biométrie, modélisation de phénomènes biologiques*, Masson, 1993. 69, 72

[TG95] A. Turing et J.-Y. Girard – *La machine de Turing*, Points - Sciences, Éditions du Seuil, 1995. 19, 122

[Tou99] P. Toulouse – *Thèmes de probabilités et statistique*, Dunod, 1999, INTERDIT À L'ORAL. 69

[VK85] K. Vo Kac – *Estimation et tests*, Ellipses, 1985. 69

[YY59] A. M. Yaglom et I. M. Yaglom – *Probabilité et Information*, Dunod, 1959, Traduit du russe par W. Mercourov. 142

© 2001 Dj. Chafaï. *GFDL copyleft. Version* djc-amops-germinal *du 2001-12-05, compilée par LaTeX2ε le 5 décembre 2001 à 19:56.*