I – Variables aléatoires

Introduction

Dans ce premier TP, nous allons faire nos premiers pas dans la manipulation d'expériences aléatoires avec MATLAB, notamment en ce qui a trait à la génération (**simulation**) informatique d'un jeu de données, et plus généralement de valeurs d'une variable aléatoire de distribution donnée.

Sur ce point, un logiciel comme MATLAB contient déjà plusieurs commandes « haut niveau » permettant d'obtenir aisément le résultat voulu, mais nous allons aller voir un peu comment celles-ci s'y prennent.

Nous vous invitons (comme au premier semestre) à travailler en petits groupes et à tenir un « **journal de bord** » numérique (document ou dossier partagé) comportant

- vos expérimentations et résultats (code, figures, ...) ;
- vos réponses aux questions, observations, conclusions ;
- toute autre information jugée digne d'intérêt (références externes, ...).

Bref tout ce qui pourra vous être utile pour générer un compte-rendu plus officiel ou lors d'une épreuve de travaux pratiques comportant des exercices semblables à ceux traités lors des séances.

Au-delà de l'évaluation du module, cela a pour but d'encourager de bonnes pratiques de documentation et d'archivage qui facilitent grandement la vie de tout ingénieur ou scientifique au sens large : reproductibilité des résultats, rédaction de rapport, ... ou simple consultation personnelle subséquente.

Mais assez parlé, plongeons dès maintenant dans le vif du sujet.

Simulation

Les logiciels de calcul comme MATLAB sont habituellement capables de générer des nombres (pseudo) aléatoires suivant la plupart des distributions de probabilité courantes (ici via la boîte à outils *Statistics and Machine Learning*). Par exemple, on génére facilement 2000 valeurs tirées selon une loi exponentielle d'*espérance* $\mu = 3$ (donc de *paramètre* $\lambda = \frac{1}{3}$)

```
x = exprnd(3,1,2000);          % à quoi sert le 1 ?
hist(x)
xlabel("x")
ylabel("effectif")
```

Profitons-en pour rappeler la distinction importante qui existe entre mode, moyenne et médiane :

```
mode(x), mean(x), median(x)   % les résultats sont-ils cohérents avec le TD ?
```

Il arrive ceci dit parfois que l'on doive se débrouiller avec seulement des nombres tirés d'une loi uniforme $\mathcal{U}([0,1])$ – bien souvent approximativement obtenue par renormalisation d'une $\mathcal{U}(\llbracket 0, 2^n \llbracket)$... Ou encore on se demande comment fait MATLAB lorsqu'on lui demande des nombres aléatoires comme ci-haut.

Voici une façon générale de procéder, appelée **méthode de la fonction de répartition inverse**[1]. Supposons donc que l'on a accès à une variable aléatoire $U \sim \mathcal{U}([0,1])$, et que l'on se donne une fonction

$$F : \mathbb{R} \longrightarrow [0,1]$$

représentant la fonction de répartion de la variable à simuler, que l'on supposera ici *strictement* croissante pour simplifier – en plus bien sûr de demander que

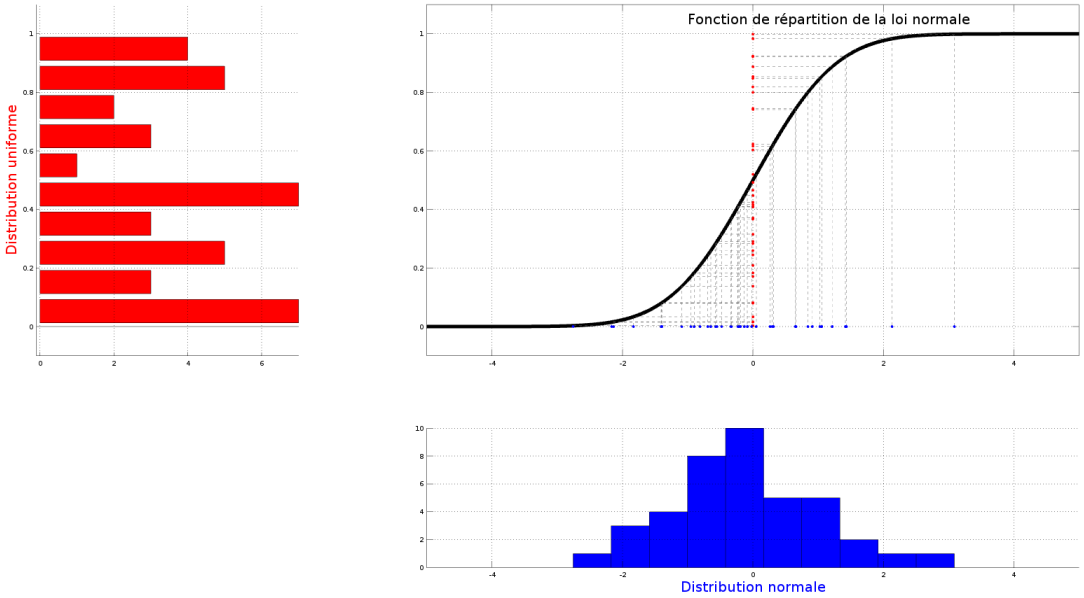
$$\lim_{x \to -\infty} F(x) = 0, \qquad \lim_{x \to +\infty} F(x) = 1.$$

Alors, la variable aléatoire

$$X := F^{-1}(U)$$

admet F comme fonction de répartition, comme on peut facilement s'en convaincre :

$$\mathbb{P}\big[X \leqslant x\big] = \mathbb{P}\big[F^{-1}(U) \leqslant x\big] = \mathbb{P}\big[U \leqslant F(x)\big] = \int_{-\infty}^{F(x)} f_U(t)\,\mathrm{d}t = \int_0^{F(x)} \mathrm{d}t = F(x).$$



1) Simulation d'une loi normale

Utilisons cette méthode pour simuler des tirages d'une loi $\mathcal{N}(\mu, \sigma^2)$.

```
n = 200                                % nombre de tirages
mu = 10                                % espérance
sigma = 5                              % écart-type
x = norminv(rand(1,n), mu, sigma);
hist(x,20)
xlabel("x")
ylabel("effectif")
```

a) Expérimentez en tirant plusieurs jeux de données et avec différentes valeurs de n, μ, σ (commentez).

b) Comparez les résultats obtenus avec la distribution attendue (à l'aide de `normpdf`, ou de la formule explicite si le cœur vous en dit).

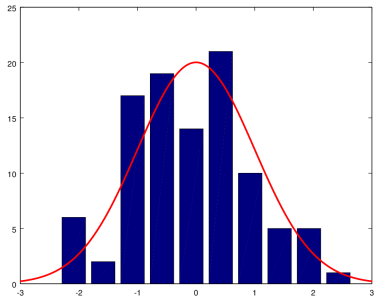
1. on voit aussi souvent « transformée inverse » même si cela n'a rien à voir avec Fourier ou Laplace

Note : Si on veut superposer les deux graphes, il faut tenir compte de la différence d'échelle sur l'axe des ordonnées : on trouve par défaut dans l'histogramme des effectifs (nombres entiers), alors que la densité est normalisée de façon à ce que l'aire totale sous la courbe soit égale à 1.

Le plus simple pour superposer les deux est sans doute de dilater verticalement la densité de façon à ce que l'aire totale sous la courbe corresponde à l'aire totale de l'histogramme ; celle-ci est donnée par

$$\frac{\text{étendue des valeurs} \times \text{nombre d'observations}}{\text{nombre de rectangles}},$$

le nombre de rectangles étant 10 par défaut.



2) Loi normale du pauvre

Cette méthode générale est bien pratique, mais elle nécessite de pouvoir évaluer aisément la réciproque F^{-1} de la fonction de répartition souhaitée – or ce n'est pas toujours le cas dans un environnement de calcul donné [2] (en plus de soulever quelques considérations numériques qui peuvent être délicates).

Voici un truc qui marche suprenamment bien pour la loi normale : on se contente d'additionner quelques nombres aléatoires entre eux !

```
m = 1                              % nombre de termes
n = 100000                         % nombre de tirages

y = zeros(1,n);
for i = 1:m
  y = y + rand(1,n);
end

hist(y,20)
```

Note : si X est une variable aléatoire de loi $\mathcal{U}([0,1])$, on peut montrer (n'est-ce pas ?) que

$$\mathbb{E}[X] = \frac{1}{2}, \qquad \mathrm{Var}(X) = \frac{1}{12}.$$

a) À partir de quelle valeur de m la distribution de Y vous semble-t-elle *raisonnablement* normale ? Vous pouvez augmenter graduellement la valeur de m et comparer soit les résultats de simulation, soit les densités elles-mêmes (que vous devriez pouvoir représenter avec les outils du 1er semestre).

b) Vérifier numériquement (à l'aide de `mean` et `var`) qu'on semble bien avoir

$$\mathbb{E}[Y] = \frac{m}{2}, \qquad \mathrm{Var}(Y) = \frac{m}{12}.$$

c) En déduire un algorithme « tout bête » (pouvant être implémenté en quelques lignes de C !) permettant de générer des nombres aléatoires approximativement distribués selon une $\mathcal{N}(0,1)$.

2. par exemple en programmation embarquée avec ressources limitées