Lycée Carnot ECS1B – 2018/2019

Travaux Pratiques d'Informatique

Deuxième partie

Matthias Gorny

Table des matières

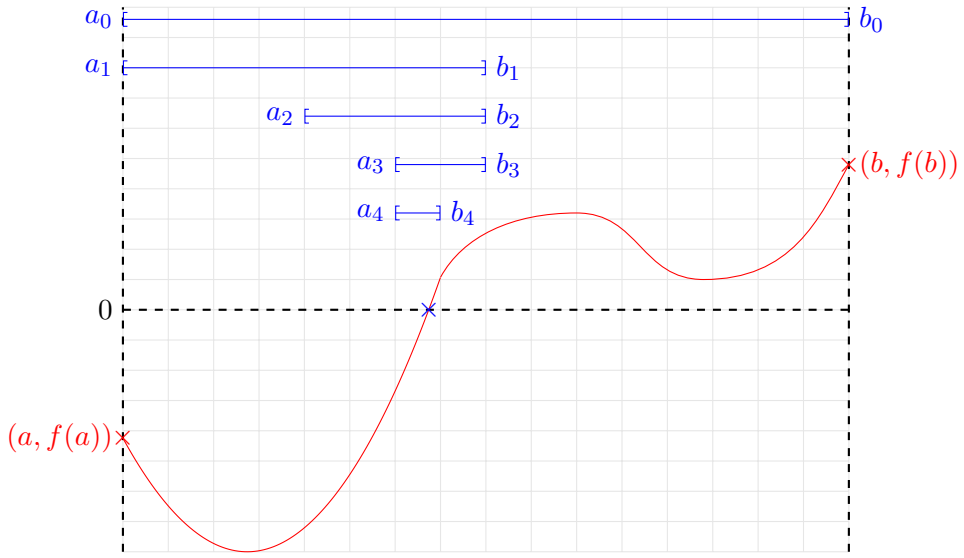
Feuille de TP n° 9

Résolution de manière approchée de l'équation $f(x) = 0$

I Programmation en Scilab de l'algorithme de dichotomie

Soit $(a, b) \in I^2$ tel que $a < b$. Soit f une fonction continue sur $[a, b]$ à valeurs réelles telle que 0 est compris entre $f(a)$ et $f(b)$. En cours, pour prouver le théorème des valeurs intermédiaires, nous avons utilisé la méthode dite de dichotomie : on introduit les deux suites $(a_n)_{n\in\mathbb{N}}$ et $(b_n)_{n\in\mathbb{N}}$ définies par $a_0 = a$, $b_0 = b$ et, pour tout $n \in \mathbb{N}$,

$$(a_{n+1}, b_{n+1}) = \begin{cases} \left(a_n, \dfrac{a_n + b_n}{2}\right) & \text{si} \quad f(a_n)\, f\left(\dfrac{a_n + b_n}{2}\right) < 0, \\ \left(\dfrac{a_n + b_n}{2}, b_n\right) & \text{si} \quad f(a_n)\, f\left(\dfrac{a_n + b_n}{2}\right) \geqslant 0. \end{cases}$$



On a montré que les suites $(a_n)_{n\in\mathbb{N}}$ et $(b_n)_{n\in\mathbb{N}}$ sont adjacentes et convergent toutes les deux vers un réel c vérifiant $f(c) = 0$. De plus on a

$$\forall n \in \mathbb{N}, \qquad a_n \leqslant a_{n+1} \leqslant c \leqslant b_{n+1} \leqslant b_n.$$

Soit $\varepsilon > 0$. Il existe $n_0 \in \mathbb{N}$ tel que $0 < b_{n_0} - a_{n_0} \leqslant \varepsilon$. Nous en déduisons que :

- $0 < b_{n_0} - c \leqslant \varepsilon$ donc b_{n_0} est une valeur approchée par excès de c.
- $0 < c - a_{n_0} \leqslant \varepsilon$ donc a_{n_0} est une valeur approchée par défaut de c.

La construction des suites $(a_n)_{n\in\mathbb{N}}$ et $(b_n)_{n\in\mathbb{N}}$ se fait de manière algorithmique. On se propose de la programmer en Scilab.

Exercice 1. (★★) Supposons que l'on dispose d'une fonction f continue sur un intervalle I de $\mathbb{R}$ que l'on code en Scilab dans un script nommé `f.sci`. Le programme (incomplet) suivant est composé de deux parties :

- Une première consacrée à l'implémentation d'une fonction f continue sur un intervalle I de $\mathbb{R}$.
- Une seconde consacrée à l'implémentation de l'algorithme de dichotomie : le programme demande à l'utilisateur trois réels a, b, ε vérifiant $(a, b) \in I^2$, $a < b$ et $\varepsilon > 0$ et affiche une valeur approchée par défaut et par excès, avec une précision ε, d'une solution de l'équation $f(x) = 0$ d'inconnue $x \in [a, b]$.

```
//Définition de la fonction f
function y=f(x)
      y=
endfunction;

//Algorithme de dichotomie
a=input('a=? ')
b=input('b=? ')
eps=input('précision : eps=? ')
if  f(a)*f(b)>0   then
      disp('erreur : f(a)*f(b)>0')//Message d'erreur
elseif   a>=b   then
      disp('erreur : a>=b')//Message d'erreur
else
      I='['+string(a)+','+string(b)+']';//Intervalle I=[a,b]
      while
            if  f((a+b)/2)==0   then

            elseif   f(a)*f((a+b)/2)<0   then

            else

            end
      end
      disp('L''équation f(x)=0 possède une solution c sur '+I+' qui vérifie '))
      disp(string(a)+' <= c <= '+string(b)+'.')
end
```

1) Compléter ce programme et implémenter-le dans un script nommé `dichotomie.sce`.
2) Tester le programme avec $f : x \longmapsto \cos(x/2)$, $a = 0$, $b = 4$ et $\varepsilon = 10^{-5}$.
3) Modifier le script afin de représenter graphiquement cette méthode (tracer la courbe représentative de f et représenter les intervalles $[a_n, b_n]$ successifs).

Exercice 2. (★) Créer un script appelé `sqrt2_Dicho.sce`. Copier/coller le contenu du script de `dichotomie.sce` et modifier-le de telle sorte qu'il affiche en plus le nombre d'itérations nécessaires pour déterminer une valeur approchée de $\sqrt{2} \in [a, b]$ à ε-près.

Tester avec $a = 1$, $b = 2$ et $\varepsilon = 10^{-6}$. Noter ici le nombre d'itérations : $\square$

Exercice 3. (★★★) La fonction $\mathrm{Arccos} : [-1, 1] \longrightarrow [0, \pi]$ est la fonction réciproque de $\cos : [0, \pi] \longrightarrow [-1, 1]$ (cf. feuille d'exercice 13).

1) A l'aide de l'algorithme de dichotomie, écrire une fonction Scilab nommée `Arccos.sci` qui prend en argument un réel $x \in [-1, 1]$ et qui calcule (une valeur approchée à 10^{-10} près de) $\mathrm{Arccos}(x)$. On pourra ajouter un message d'erreur si on donne un réel $x \notin [-1, 1]$ en argument.
2) Représenter sur un même graphique la fonction Arccos en utilisant la fonction `Arccos.sci` et la fonction prédéfinie `acos`.

II Méthode du point fixe

Supposons que résoudre $f(x) = 0$ soit équivalent à résoudre l'équation $g(x) = x$, pour une certaine fonction g (il en existe forcément : par exemple $g : x \mapsto f(x) + x$), et donc équivalent à déterminer un point fixe de g. Soit $(x_n)_{n\in\mathbb{N}}$ la suite définie par $x_0 \in D_g$ (bien choisi) et, pour tout $n \in \mathbb{N}$, $x_{n+1} = g(x_n)$. On sait que, si $(x_n)_{n\in\mathbb{N}}$ converge vers un réel ℓ et si g est continue au voisinage de ℓ, alors ℓ est un point fixe de g.

Pour que la suite $(x_n)_{n\in\mathbb{N}}$ converge, il faut bien choisir la fonction g. Nous avons vu dans le chapitre *Dérivation* que c'est le cas si g est définie sur un intervalle fermé I vérifiant :

- $g(I) \subset I$,
- g est dérivable sur I,
- g' est bornée par un réel $M \in]0,1[$ sur I,
- g admet un point fixe sur I.

Exercice 4. (★★) Dans cette exercice, nous considérons la fonction $g : x \in \mathbb{R}_+^* \mapsto 2\left(x - \dfrac{1}{x}\right)$. Pour tout $x \in \mathbb{R}^*$, $x^2 = 2$ si et seulement si $x = g(x)$. Créer un script appelé `sqrt2_Pointfixe1.sce` qui prend en entrée un réel x_0 et qui calcule les 100 premières valeurs de la suite $(x_n)_{n\in\mathbb{N}}$ intervenant dans la méthode du point fixe avec la fonction g. Constater que la suite ne semble pas converger.

Exercice 5. (★★★) Dans cette exercice, nous considérons la fonction $g : x \in \mathbb{R}_+^* \mapsto \dfrac{2}{3}\left(x + \dfrac{1}{x}\right)$. Pour tout $x \in \mathbb{R}^*$, $x^2 = 2$ si et seulement si $x = g(x)$.

1) Soit $(x_n)_{n\in\mathbb{N}}$ la suite définie par $x_0 \in \mathbb{R}_+^*$ et, pour tout $n \in \mathbb{N}$, $x_{n+1} = g(x_n)$.
   a) Montrer que, pour tout $n \in \mathbb{N}^*$, $x_n > 1$.
   b) Montrer que g' est bornée sur $]1, +\infty[$ par un réel $M \in]0,1[$.
   c) Montrer que, pour tout $n \in \mathbb{N}^*$, $|x_{n+1} - \sqrt{2}| \leqslant M|x_n - \sqrt{2}|$.
   d) En déduire que, pour tout $n \in \mathbb{N}$, $|x_n - \sqrt{2}| \leqslant M^n|x_0 - \sqrt{2}|$ et donc que $(x_n)_{n\in\mathbb{N}}$ converge vers $\sqrt{2}$.
2) Créer un script appelé `sqrt2_Pointfixe2.sce` qui prend en entrée deux réels x_0 et $\varepsilon > 0$ et qui calcule une valeur approchée de $\sqrt{2}$ à ε-près en utilisant la méthode du point fixe (la suite étant initialisée à x_0 et construite avec la fonction g).
3) Tester avec $x_0 = 1$ et $\varepsilon = 10^{-8}$. Noter ici le nombre d'itérations : $\square$
   Comparer avec l'approximation fournie par la méthode de dichotomie.
4) Peut-on remplacer la boucle `while` par une boucle `for` (quitte à perdre en rapidité) ?

Exercice 6. (★★) Nous cherchons une valeur approchée d'une solution de l'équation $x(e^x + 1) = \ln(e^x + 1)$.

1) Montrer que, pour tout $x \in \mathbb{R}$, $x(e^x + 1) = \ln(e^x + 1)$ si et seulement si $f(x) = x$ avec $f : x \longmapsto e^{-x}\ln(1 + e^{-x})$.
2) Soit $(x_n)_{n\in\mathbb{N}}$ la suite définie par $x_0 \in \mathbb{R}$ et, pour tout $n \in \mathbb{N}$, $x_{n+1} = f(x_n)$. Nous avons montré dans l'exercice 19 de la feuille d'exercice n° 13, que f admet un unique point fixe $\ell \in]0,1[$ et que $(x_n)_{n\in\mathbb{N}}$ converge vers ℓ.

   Créer un script Scilab qui prend en entrée deux réels x_0 et $\varepsilon > 0$ et qui calcule une valeur approchée de ℓ à ε-près en utilisant la suite $(x_n)_{n\in\mathbb{N}}$.

III Méthode de Newton

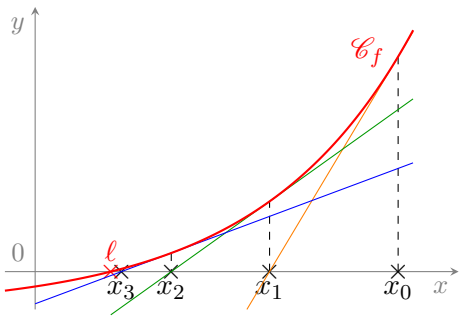
Soit $(a, b) \in \mathbb{R}^2$ tel que $a \leqslant b$. Soit f une fonction dérivable sur $[a, b]$ et à valeurs réelles vérifiant

- Il existe $x^* \in]a, b[$ tel que $f(x^*) = 0$,
- f' ne s'annule pas sur $[a, b]$.

La méthode de Newton est un cas particulier de la méthode du point fixe dans le but de trouver une valeur approchée de la racine x_0. Elle se base sur la construction d'une suite récurrente $(x_n)_{n\in\mathbb{N}}$ définie par $x_0 \in]a, b[$ (« proche » de x^*) et

$$\forall n \in \mathbb{N}, \qquad x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}.$$

Interprétation graphique. Pour tout $n \in \mathbb{N}$, x_{n+1} est la solution de l'équation $f(x_n) + f'(x_n)(x - x_n) = 0$, c'est-à-dire x_{n+1} est l'abscisse du point d'intersection de la tangente à $\mathscr{C}_f$ en x_n avec la droite des abscisses.



Convergence et algorithme. Si $(x_n)_{n\in\mathbb{N}}$ converge vers un réel $\ell \in]a,b[$ alors, par continuité de f et f' sur $[a,b]$, nous obtenons $\ell = \ell - f(\ell)/f'(\ell)$, soit $f(\ell) = 0$. Ceci nous fournit une méthode algorithmique pour déterminer une valeur approchée d'une solution de l'équation $f(x) = 0$. Dans l'exercice suivant, avec des hypothèses supplémentaires sur la fonction f, on montre que la suite $(x_n)_{n\in\mathbb{N}}$ converge vers ℓ de manière quadratique : si x_0 est suffisamment proche de ℓ, alors il existe $K > 0$ et $\lambda \in]0,1[$ tels que $|x_n - \ell| \leqslant K\,\lambda^{2^n}$ pour n assez grand. Il s'agit donc d'une convergence très rapide. Une fois fixée une précision ε, on arrête l'algorithme dès que $|x_{n+1} - x_n| \leqslant \varepsilon$.

Exercice 7 – Méthode de Newton. (★★) Supposons que f est de classe C^2 sur $[a,b]$. Le but de cet exercice est de montrer qu'il existe $\varepsilon > 0$ tel que, si $|x_0 - x^*| \leqslant \varepsilon$, alors la suite $(x_n)_{n\in\mathbb{N}}$ converge vers x^* à vitesse quadratique, c'est-à-dire

$$\exists K > 0, \quad \exists \lambda \in]0,1[, \qquad \forall n \in \mathbb{N}, \qquad |x_n - x^*| \leqslant K\,\lambda^{2^n}.$$

1) Posons $\varphi : x \in [a,b] \longmapsto x - \dfrac{f(x)}{f'(x)}$. Remarquer que x^* est un point fixe de φ.
2) Justifier l'existence de $m > 0$ et $M > 0$ tels que, pour tout $x \in [a,b]$, $|f'(x)| \geqslant m$ et $|f''(x)| \leqslant M$.
3) En utilisant la formule de Taylor-Lagrange, montrer que

$$\forall x \in [a,b], \qquad |\varphi(x) - x^*| \leqslant \frac{M}{2m}|x - x^*|^2.$$

4) Pour tout $n \in \mathbb{N}$, posons $u_n = \dfrac{M}{2m}|x_n - x^*|$.
   a) Montrer que, pour tout $n \in \mathbb{N}$, $u_n \leqslant u_0^{2^n}$.
   b) Conclure

Exercice 8. (★★)

1) Créer un script appelé `sqrt2_Newton.sce` qui prend en entrée deux réels x_0 et $\varepsilon > 0$ et qui calcule une valeur approchée de $\sqrt{2}$ à ε-près en utilisant la méthode de Newton (la suite étant initialisée à x_0) avec la fonction $f : x \longmapsto x^2 - 2$
2) Tester avec $x_0 = 1$ et $\varepsilon = 10^{-8}$. Noter ici le nombre d'itérations : $\square$
   Comparer avec les approximations fournies par la méthode de dichotomie et la méthode du point fixe.

Une variante : la méthode de la sécante (ou de Lagrange). La méthode de Newton nécessite que la fonction f considérée soit dérivable. La méthode de la sécante consiste à remplacer, pour tout $n \in \mathbb{N}^*$,

$$f'(x_n) \qquad \text{par} \qquad \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}.$$

On construit donc la suite $(x_n)_{n\in\mathbb{N}}$ définie par $x_0 \in \mathbb{R}$, $x_1 \in \mathbb{R}$ et, pour tout $n \in \mathbb{N}$,

$$x_{n+2} = x_{n+1} - \frac{(x_{n+1} - x_n)f(x_{n+1})}{f(x_{n+1}) - f(x_n)}.$$

Exercice 9. (★★★) Créer un script appelé `sqrt2_Secante.sce` qui prend en entrée trois réels x_0, x_1 et $\varepsilon > 0$ et qui calcule une valeur approchée de $\sqrt{2}$ à ε-près en utilisant la méthode de la sécante (la suite étant initialisée à x_0 et x_1).

Tester avec $x_0 = 1$, $x_1 = 2$ et $\varepsilon = 10^{-8}$. Noter ici le nombre d'itérations : $\square$
Comparer avec les approximations fournies par la méthode de dichotomie, la méthode du point fixe et la méthode de Newton.

Feuille de TP n° 10

Simulation et représentation de variables aléatoires réelles discrètes

Ce TP accompagne le chapitre 6 (Informatique et Algorithmique) : **Probabilités avec Scilab**.

I Entrainement

1) Simulation de variables aléatoires réelles discrètes

Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin.
John von Neumann

a) La fonction `rand`

• La fonction `rand` de Scilab simule un nombre tiré aléatoirement entre 0 et 1. Exécuter trois fois consécutivement la commande `rand()` dans la console. Constater que vous avez tous les trois mêmes valeurs...

• Fermer Scilab et exécuter trois fois consécutivement la commande `rand()` dans la console. Constater que vous avez tous les trois mêmes valeurs... `0.2113249`, `0.7560439` et `0.0002211`. Pas très aléatoire tout ça... les raisons sont expliquées en détail dans le cours. On parle de nombres pseudo-aléatoires.

• L'algorithme utilisé par Scilab pour simuler des nombres pseudo-aléatoires peut être modifié. Il suffit de changer la graine (ou le germe) de l'algorithme, c'est-à-dire sa terme initial, en lui affectant la date (en seconde) de la machine. Cela permet d'intégrer une petite dose de hasard dans le générateur. Exécuter `n=getdate('s'); rand('seed',n);` puis exécuter trois fois consécutivement la commande `rand()` dans la console. Comparer avec vos camarades. Qu'en pensez-vous ?

• Exécuter les commandes suivantes dans la console :

```
-->rand(1,10);
-->rand(5,1);
-->rand(7,6);
```

• Exécuter la commande `rand('seed',0);` puis `rand()` à nouveau. Que s'est-il passé ?

b) Simulations avec `rand`

• Soient `a` et `b` des entiers tels que `a` $<$ `b`. Justifier brièvement que le nombre renvoyé par la commande `X=a+floor((b-a+1)*rand())` peut être assimilé à la réalisation d'une variable aléatoire de loi uniforme sur $[\![\texttt{a}, \texttt{b}]\!]$. Exécuter les commandes suivantes dans la console :

```
-->a=-6; b=10; X=a+floor((b-a+1)*rand())
-->X=a+floor((b-a+1)*rand(9,5))
```

Écrire une commande qui simule le résultat de 20 lancers successifs d'un dé équilibré à six faces. Tester-la plusieurs fois dans la console.

• Soit `p` un réel de $]0, 1[$. Justifier brièvement que le nombre renvoyé par la commande `rand()<p` peut être assimilé à la réalisation d'une variable aléatoire de loi de Bernoulli de paramètre `p`. Exécuter les commandes suivantes dans la console :

```
-->X=(rand()<0.1)
-->X=(rand(4,7)<0.8)
```

• Soient p un réel de $]0,1[$ et n un entier naturel non nul. Justifier brièvement que le nombre renvoyé par la commande `sum(rand(1,n)<p)` peut être assimilé à la réalisation d'une variable aléatoire de loi binomiale de paramètres n et p. Exécuter les commandes suivantes dans la console :

```
-->X=sum(rand(1,20)<0.9)
-->X=sum(rand(1,20)<0.2)
```

Écrire une commande qui simule le nombre de 6 obtenus en lançant 100 fois un dés équilibré à six faces. Tester-la plusieurs fois. Commenter les résultats obtenus.

• Soient p un réel de $]0,1[$. Justifier brièvement que le nombre renvoyé par les instructions `X=1; while (rand()>=p); X=X+1; end; X` peut être assimilé à la réalisation d'une variable aléatoire de loi géométrique de paramètre p. Exécuter les commandes suivantes dans la console :

```
-->X=1; while (rand()>=0.9); X=X+1; end; X
-->X=1; while (rand()>=0.1); X=X+1; end; X
```

Écrire une commande qui simule le nombre de lancers de dés nécessaires pour obtenir le premier 6. Tester-la plusieurs fois.

c) Simulations avec `grand`

• Exécuter les commandes suivantes dans la console :

```
-->U=grand(5,7,'uin',-1,2)
-->X=grand(9,1,'bin',1,0.6)
-->Y=grand(1,12,'bin',50,0.6)
```

```
-->Z=grand(4,4,'geom',0.3)
-->T=grand(6,2,'poi',5)
```

• Simuler une variable aléatoire de loi de Poisson de paramètre 9.

II Représentation graphique

1) Représentation graphique de lois théoriques

a) Diagrammes en bâtons

• Écrire un script contentant le programme suivant :

```
n=input('Entrer un entier naturel n strictement positif : ')
p=input('Entrer un réel p compris entre 0 et 1 : ')
u=(1-p)^n; U=[u];
for  k=1:n
     u=u*(n-k+1)/k;
     u=u*p/(1-p);
     U=[U,u];
end
```

Enregistrer ce script sous le nom `LoiBin.sce` et exécuter-le dans la console (avec un `clf();` avant).

• Tester `LoiBin.sce` avec différentes valeurs de n et p. Pour chaque essai, exécuter la commande `bar(0:n,U,'r')` ou `plot2d3(0:n,U,style=5)`.

• Construire le diagramme en bâtons représentant la loi du résultat d'un lancer de dé équilibré. Même question pour la loi de la somme du résultats de deux dés.

b) Fonctions de répartition

• Exécuter les commandes suivantes dans la console :

```
-->X=[-1,2,3,4,2,2,3,2,3,5,6,-1,3,4,2];
-->tabul(X)
-->T=tabul(X,'i')
```

```
-->T(:,2)=T(:,2)/sum(T(:,2))
-->T(:,2)=cumsum(T(:,2));
```

Que font les commandes `tabul` et `cumsum` ?

• Exécuter la commande `plot2d2(T(:,1),T(:,2))`.

• Construire la fonction de répartition d'une variable aléatoire de loi binomiale (pour différents paramètres).

• Construire la fonction de répartition de la loi du résultat d'un lancer de dé équilibré. Même question pour la loi de la somme du résultats de deux dés.

2) Représentation graphique de phénomènes aléatoires

• Exécuter les commandes suivantes dans la console :

```
-->p=0.7; n=10;
-->X=grand(1,1000,'bin',n,p);
-->T=tabul(X,'i');
```

```
-->T(:,2)=T(:,2)/sum(T(:,2));
-->clf(); plot2d3(T(:,1),T(:,2),style=-1)
```

Exécuter ensuite la commande `LoiBin.sce` afin de superposer le diagramme théorique de la loi $\mathcal{B}(10, 7/10)$ au diagramme empirique obtenu. Commenter. On ajoutera une légende.

• Exécuter les commandes suivantes dans la console :

```
-->T(:,2)=cumsum(T(:,2));
-->clf(); plot2d2(T(:,1),T(:,2))
```

```
-->exec('LoiBin.sce',-1)
//Prendre n=10 et p=0.7
-->plot2d2(0:n,cumsum(U),style=5)
```

• Faites la même chose pour 1000 lancers de dés successifs (diagramme en bâtons, fonction de répartition empirique, superposition avec les courbes théoriques).

III Exercices

Exercice 1. (★) Écrire une fonction Scilab qui prend en entrée deux entiers naturels n et p strictement positifs tels que $p \leqslant n$, qui simule p tirages successifs et avec remise dans une urne contenant n boules numérotées de 1 à n et qui renvoie un vecteur contenant les p numéros tirés (dans l'ordre du tirage).

Exercice 2. (★★) Recommencer l'exercice précédent avec des tirages sans remise.
Indication : on pourra commencer par se donner le vecteur `B=1:`n représentant les numéros des n boules, ainsi qu'un vecteur vide `T=[]` représentant les numéros piochés. Ensuite on pourra

- *tirer uniformément un nombre `k` dans `1:length(B)`,*
- *ajouter `B(k)` à `T` et supprimer `B(k)` de `B`,*

puis recommencer ces deux étapes à partir des vecteurs `B` et `T` ainsi modifiés.

Exercice 3. (★) Recommencer l'exercice précédent avec des tirages simultanés.
Indication : on utilisera la suite de commandes `T=tabul(T,'i'); T=T(:,1)`.

Exercice 4 – Modélisation de l'exercice 17 du TD n° 10. (★★) Écrire une fonction qui prend en entrée un entier naturel `n` non nul, qui simule l'expérience aléatoire consistant à effectuer des tirages successifs avec remise dans une urne contenant n boules numérotées de 1 à n jusqu'à ce qu'on obtienne un numéro supérieur ou égal au numéro précédent. La fonction renverra le nombre de tirages effectués.

Exercice 5 – Modélisation de l'exercice 1 du concours blanc. (★★) Écrire une fonction qui prend en entrée un entier naturel `n` non nul, qui simule l'expérience aléatoire consistant à effectuer des tirages successifs avec remise dans une urne contenant n boules numérotées de 1 à n jusqu'à ce que la somme cumulée des numéros des boules obtenues soit supérieure ou égale à `n`. La fonction renverra le nombre de tirages effectués.

Exercice 6. (★★) Écrire une fonction Scilab qui prend en entrée deux réels p et q de $]0, 1[$ tels que $p + q < 1$ et qui simule la réalisation d'une variable aléatoire X telle que

$$X(\Omega) = \{0, 1, 2\}, \qquad \mathbb{P}(X = 0) = p, \qquad \mathbb{P}(X = 1) = q \qquad \text{et} \qquad \mathbb{P}(X = 2) = 1 - p - q.$$

Indication : on s'inspirera de la suite de commandes permettant de simuler une variable aléatoire de loi de Bernoulli à l'aide la fonction `rand`.

Exercice 7 – Modélisation de l'exercice 19 du TD n° 9. (★★) Une mouche entre dans un studio de deux pièces (une chambre et une salle de bain). Elle se trouve initialement dans la salle de bain. On relève sa position dans le studio toutes les minutes. Si elle est dans la salle de bain à la $n^{\text{ième}}$ minute, elle y reste avec probabilité $1/3$ ou elle va dans la chambre avec probabilité $2/3$. Si elle est dans la chambre à la $n^{\text{ième}}$ minute, elle y reste avec probabilité $1/2$, elle va dans la salle de bain avec probabilité $1/4$ ou elle sort part la fenêtre avec probabilité $1/4$.

A l'aide l'exercice précédent, écrire une fonction Scilab qui prend en entrée un entier naturel `n` non nul, qui simule cette expérience aléatoire et qui renvoie la position de la mouche à l'issue de la $\texttt{n}^{\text{ième}}$ étape (on codera la salle de bain par 0, la chambre par 1 et l'extérieur par 2).

Exercice 8 – Modélisation de l'exercice 3 du DS n° 3. (★★★) Soient $n \in \mathbb{N}^*$ et $p \in]0,1[$. On dispose de

- deux urnes U et V initialement vides et pouvant contenir au plus n boules chacune.
- une pièce de monnaie telle que la probabilité de tomber sur Pile est p.

On lance successivement la pièce au plus $2n-1$ fois. Lorsqu'elle tombe sur Pile, on ajoute une boule dans l'urne U. Lorsqu'elle tombe sur Face, on ajoute une boule dans l'urne V. On s'arrête lorsque l'une des deux urnes est pleine. On note X_n la variable aléatoire réelle égale au nombre de boules contenues dans l'urne qui n'est pas pleine à l'issue de l'expérience.

1) Écrire une fonction Scilab qui prend en entrée p et n et qui renvoie une réalisation de X_n.
   Indication : on introduira deux variables `u` et `v`, initialement nulles, qui comptent le nombre de boules contenues dans les urnes U et V respectivement. L'expérience s'arrête dès que `u`=n ou `v`=n.
2) On réalise $N = 10000$ fois cette expérience dans le cas où $p = 1/2$. Calculer la valeur moyenne obtenue.
3) On a montré dans le DS n° 3 que, si $p = 1/2$, alors $\mathbb{E}(X_n) = n - \prod_{k=1}^{n-1}\left(1+\frac{1}{2k}\right)$. Écrire une fonction qui prend en entrée n et qui renvoie une valeur de $\mathbb{E}(X_n)$ calculée à l'aide de cette formule. Comparer avec le résultat de la question précédente, pour différentes valeurs de n.

Exercice 9. (★★) On dispose d'un stock suffisant de boules rouges et d'une urne qui contient $r \geqslant 2$ boules rouges et $b \geqslant 2$ boules bleues. On tire une boule au hasard dans l'urne. Si elle est rouge, on la remet dans l'urne. Si elle est bleue, on la remplace par une rouge et on la remet dans l'urne. On recommence cette opération indéfiniment.

1) Écrire une fonction commençant par `function R=tirage(r,b,n)` qui simule `n` tirages successifs avec cette règle et renvoie le vecteur `R` dont la valeur est `[r1,...,rn]` où, pour tout $k \in [\![1, \texttt{n}]\!]$, $\texttt{r}_k$ est le nombre de boules rouges à l'issue du $k^{\text{ième}}$ tirage.
   Indication : sachant qu'il y a $\texttt{r}_k$ boules rouges à l'issue du $k^{\text{ième}}$ tirage, la probabilité de tirer une boule rouge au $(k+1)^{\text{ième}}$ tirage est $\frac{\texttt{r}_k}{\texttt{r+b}}$.
2) Écrire un programme qui demande à un entier `N` et les nombres `r,b,n`, qui simule `N` fois cette expérience et qui renvoie un vecteur `[m1,...,mn]` où, pour tout $k \in [\![1, \texttt{n}]\!]$, $\texttt{m}_k$ est la moyenne des valeurs prises par $\texttt{r}_k$ au cours de ces `N` expériences.

Exercice 10 – Marche aléatoire symétrique. (★★) On lance une infinité de fois une pièce équilibrée de façon indépendance. La suite de variables aléatoires $(S_n)_{n\in\mathbb{N}}$ définie par $S_0 = 0$ et

$$\forall n \in \mathbb{N}, \qquad S_{n+1} - S_n = \begin{cases} 1 & \text{si le } n^{\text{ième}} \text{ lancer est tombé sur Pile} \\ -1 & \text{si le } n^{\text{ième}} \text{ lancer est tombé sur Face} \end{cases}$$

est appelée marche aléatoire simple symétrique sur $\mathbb{Z}$.

1) Écrire une fonction `marchealeatoire` qui prend en entrée un entier naturel n et qui construit un vecteur $[S_0, S_1, \ldots, S_n]$ correspondant aux n premiers pas d'une marche aléatoire.
2) Modifier le programme pour qu'il représente graphiquement les n premiers pas.
3) Écrire un programme qui renvoie l'instant en lequel la marche aléatoire revient en 0 pour la première fois (i.e. le plus petit entier $k \in \mathbb{N}^*$ tel que $S_k = 0$).
   On peut montrer que la marche revient en 0 presque sûrement.

Exercice 11 – Loi binomiale à paramètre Poisson. (★)

1) Écrire un programme qui prend en entrée deux réels a et $p \in]0,1[$, qui simule une réalisation N d'une variable aléatoire de loi $\mathcal{P}(a)$ et qui renvoie une simulation d'une variable aléatoire de loi $\mathcal{B}(N,p)$.
2) On réalise $N = 10000$ fois cette expérience. Calculer la valeur moyenne obtenue et la comparer avec ap.

Exercice 12 – Temps d'attente du $k^{\text{ième}}$ Pile. (★★)

1) Écrire un programme qui prend en entrée un réel p de $]0,1[$ et un entier $k \in \mathbb{N}^*$ et qui simule l'expérience aléatoire dont le résultat est le temps d'attente du $k^{\text{ième}}$ pile lors de lancers successifs d'une pièce truquée de sorte que Pile apparaît avec la probabilité p.
2) On réalise 10000 fois cette expérience. Calculer une valeur approchée du nombre moyen de lancers réalisés à chaque expérience.

Exercice 13 – Simulation et représentation de v.a. de loi géométrique. (★★★)

1) Écrire une fonction `simgeom.sci` qui prend en entrée un réel `p` de $]0,1[$ et un entier naturel `N` strictement positif et qui renvoie un vecteur contenant `N` réalisations indépendantes de variables aléatoires de loi géométrique de paramètre `p`. On n'utilisera pas la fonction `grand`.
2) En utilisant les fonctions `tabul` et `plotd2d3`, représenter les diagrammes en bâtons de
   a) `simgeom(0.2,50)`, `simgeom(0.5,50)` et `simgeom(0.8,50)`.
   b) `simgeom(0.2,1000)`, `simgeom(0.5,1000)` et `simgeom(0.8,1000)`.
   On pourra utiliser `subplot(2,3,#)`.
   c) Comparer avec les diagrammes en bâtons théoriques.
   La commande `p*((1-p)*ones(1,K)).^(0:(K-1))` renvoie un vecteur content $\mathbb{P}(X=k)$ pour tout $k \in [\![1,K]\!]$ lorsque X est une v.a. de loi $\mathcal{G}(p)$. Comprenez-vous pourquoi ?
3) En utilisant les fonctions `tabul`, `cumsum` et `plotd2d2`, représenter les fonctions de répartition empiriques de
   a) `simgeom(0.2,50)`, `simgeom(0.5,50)` et `simgeom(0.8,50)`.
   b) `simgeom(0.2,1000)`, `simgeom(0.5,1000)` et `simgeom(0.8,1000)`.
   On pourra utiliser `subplot(2,3,#)`.
   c) Comparer avec les fonctions de répartitions théoriques.
   La commande `plot2d2(1:K,1-(p*ones(1:K)).^(1:K))` trace la courbe représentative de la fonction de répartition d'une v.a. de loi $\mathcal{G}(p)$ sur l'intervalle $[1,K]$. Comprenez-vous pourquoi ?

Exercice 14 – Simulation et représentation de v.a. de loi de Poisson. (★★★)

Reprendre l'exercice précédent avec la loi de Poisson. On utilisera cette fois la fonction `grand` et on pourra tester les paramètres 1, 5 et 20.
On se limitera aux valeurs significatives des probabilités.

Feuille de TP n° 11

Calcul approché d'intégrales

I Méthode des rectangles

1) Rappels et compléments sur les sommes de Riemann

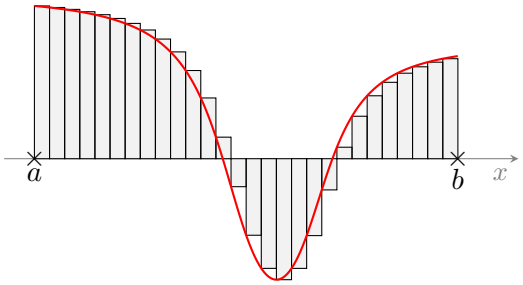
a) Convergence des sommes de Riemann

Soit f une fonction continue sur $[a,b]$ avec a et b deux réels tels que $a<b$. Fixons $n\in\mathbb{N}^*$ et notons

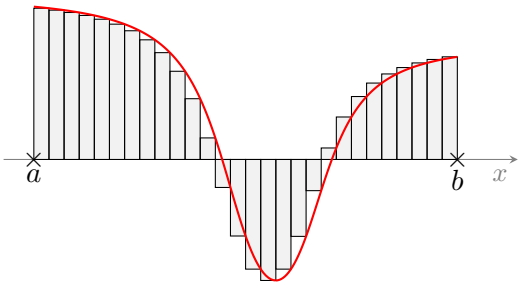
$$S_n(f)=\frac{b-a}{n}\sum_{k=0}^{n-1} f\left(a+k\frac{b-a}{n}\right),$$

$$T_n(f)=\frac{b-a}{n}\sum_{k=0}^{n-1} f\left(a+(k+1)\frac{b-a}{n}\right),$$

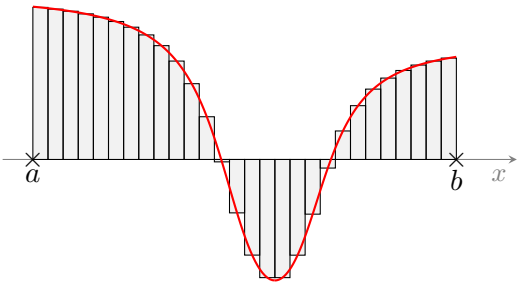
$$M_n(f)=\frac{b-a}{n}\sum_{k=0}^{n-1} f\left(a+\left(k+\frac{1}{2}\right)\frac{b-a}{n}\right).$$



En rouge la courbe représentative de f. Colorée en gris, la somme de Riemann $S_n(f)$.



Colorée en gris, la somme de Riemann $T_n(f)$.



Colorée en gris, la somme de Riemann $M_n(f)$.

Les suites $(S_n(f))_{n\geqslant 1}$, $(T_n(f))_{n\geqslant 1}$ et $(M_n(f))_{n\geqslant 1}$ convergent vers $\displaystyle\int_a^b f(t)\,dt$. Cela fournit une méthode numérique permettant d'approcher des intégrales par ces sommes. Cette méthode d'approximation est appelée la méthode des rectangles (à gauche si l'on considère $(S_n(f))_{n\geqslant 1}$, à droite si l'on considère $(T_n(f))_{n\geqslant 1}$, au milieu si l'on considère $(M_n(f))_{n\geqslant 1}$).

b) Cas des fonctions continues et croissantes

Supposons que f est continue et croissante sur $[a,b]$. Pour tout $n\in\mathbb{N}^*$,

$$\forall k\in[\![0,n-1]\!],\qquad \forall t\in\left[\frac{k}{n},\frac{k+1}{n}\right],\qquad f\left(a+k\frac{b-a}{n}\right)\leqslant f\left(a+t(b-a)\right)\leqslant f\left(a+(k+1)\frac{b-a}{n}\right)$$

donc, en intégrant,

$$\forall k\in[\![0,n-1]\!],\qquad \frac{1}{n}f\left(a+k\frac{b-a}{n}\right)\leqslant\int_{k/n}^{(k+1)/n} f\left(a+t(b-a)\right)\,dt\leqslant\frac{1}{n}f\left(a+(k+1)\frac{b-a}{n}\right).$$

On somme et on utilise la relation de Chasles : $\forall k\in[\![0,n-1]\!],\qquad \dfrac{S_n(f)}{b-a}\leqslant\displaystyle\int_0^1 f\left(a+t(b-a)\right)\,dt\leqslant\frac{T_n(f)}{b-a}$.

Le changement de variable $x=a+t(b-a)$ donne enfin : $\boxed{S_n(f)\leqslant\displaystyle\int_a^b f(x)\,dx\leqslant T_n(f).}$

Combien d'itérations sont-elles nécessaires pour que $S_n(f)$ et $T_n(f)$ soient des approximations de $\displaystyle\int_a^b f(t)\,dt$ à ε-près ? On remarque que, pour tout $n \in \mathbb{N}^*$,

$$0 \leqslant T_n(f) - S_n(f) \leqslant \frac{b-a}{n}\sum_{k=0}^{n-1}\left(f\left(a+(k+1)\frac{b-a}{n}\right) - f\left(a+k\frac{b-a}{n}\right)\right) = \frac{(b-a)(f(b)-f(a))}{n}$$

car on reconnait une somme télescopique. Si $n_0 \in \mathbb{N}^*$ est tel que $\dfrac{(b-a)(f(b)-f(a))}{n_0} \leqslant \varepsilon$, par exemple $\boxed{n_0 = \left\lfloor \dfrac{(b-a)(f(b)-f(a))}{\varepsilon} \right\rfloor + 1,}$ alors $S_{n_0}(f)$ (resp. $T_{n_0}(f)$) est une approximation de $\displaystyle\int_a^b f(t)\,dt$ à ε-près par défaut (resp. excès).

c) Cas des fonctions continues et décroissantes

Supposons que f est continue et décroissante sur $[a,b]$. On se ramène au cas précédent en considérant $-f$: Pour tout $n \in \mathbb{N}^*$, $\boxed{T_n(f) \leqslant \displaystyle\int_a^b f(x)\,dx \leqslant S_n(f).}$ Si $n_0 \in \mathbb{N}^*$ est tel que $\boxed{n_0 = \left\lfloor \dfrac{(b-a)(f(a)-f(b))}{\varepsilon} \right\rfloor + 1,}$ alors $S_{n_0}(f)$ (resp. $T_{n_0}(f)$) est une approximation de $\displaystyle\int_a^b f(t)\,dt$ à ε-près par excès (resp. défaut).

d) Cas des fonctions de classe C^1

Si f est de classe C^1 sur $[a,b]$, alors on a des informations sur la vitesse de convergence de cette méthode (cf. chapitre 14) :

$$\forall n \in \mathbb{N}^*, \qquad \left|S_n(f) - \int_a^b f(t)\,dt\right| \leqslant \frac{(b-a)^2}{2n}\max_{[a,b]}|f'|.$$

Cette inégalité permet de contrôler le terme d'erreur dans l'approximation de $\displaystyle\int_a^b f(t)\,dt$ par $S_n(f)$. Plus précisément si $n_0 \in \mathbb{N}^*$ est tel que $\dfrac{(b-a)^2}{2n_0}\max\limits_{[a,b]}|f'| \leqslant \varepsilon$, par exemple $\boxed{n_0 = \left\lfloor \dfrac{(b-a)^2}{2\varepsilon}\max\limits_{[a,b]}|f'| \right\rfloor + 1,}$ alors S_{n_0} est une approximation de $\displaystyle\int_a^b f(t)\,dt$ à ε-près.

2) Mise en oeuvre avec Scilab

Exercice 1. (★★) Supposons que l'on dispose de réels a et b tels que $a < b$ et d'une fonction continue sur $[a,b]$.

1) Recopier le programme et enregistrer-le dans un script nommé `MethodeRectangle1.sce`.

```
//Choix de a et b
a=0; b=1;

//Définition de la fonction f
function y=f(x)
      y=1./(1+x.*x);
endfunction;

//Méthode des rectangles
eps=input('précision : eps=? ')
n=floor((b-a)*(f(a)-f(b))/eps)+1;
```

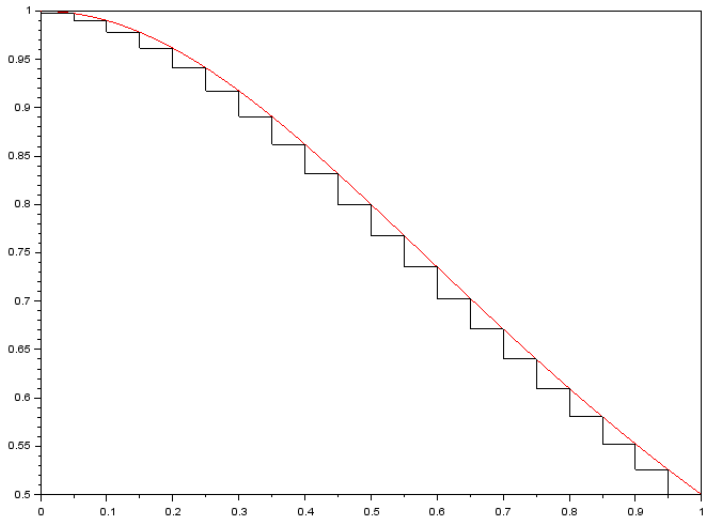
Vous vous en doutez, on s'apprête à calculer une approximation de $\displaystyle\int_0^1 \frac{dx}{1+x^2} = \frac{\pi}{4}$. Le choix de n dans l'algorithme nous est dicté par le fait que la fonction f est décroissante sur $[0,1]$.

2) En utilisant une boucle `for`, compléter ce programme pour qu'il renvoie une approximation par défaut (utilisant $T_n(f)$ dans ce cas) et par excès (utilisant $S_n(f)$ dans ce cas) de $\dfrac{\pi}{4}$.

3) En utilisant les commandes `linspace` et `sum`, modifier ce programme pour qu'il n'utilise plus de boucle `for`.

4) Modifier ce programme en changeant la valeur de `n` en exploitant le fait que $f : x \longmapsto \dfrac{1}{1+x^2}$ est de classe C^1 sur $[0,1]$. Est-ce intéressant dans ce cas ?

5) Modifier le programme afin qu'il renvoie le graphique suivant :



Exercice 2. (★★) A l'aide de boucles `while`, tester sur l'exemple de $\displaystyle\int_0^1 \frac{dx}{1+x^2}$ la méthode d'approximation qui est la plus rapide parmi la méthode des rectangles à gauche, à droite et la méthode du point milieu (celle utilisant la suite $(M_n)_{n\in\mathbb{N}}$).

Exercice 3. (★★) Calculer des approximations (à 10^{-4} près) de

1) $\ln(2) = \displaystyle\int_0^1 \frac{dx}{1+x}$, 2) $\pi = \displaystyle\int_0^1 4\sqrt{1-x^2}\,dx$ 3) $\dfrac{\pi^2}{6} = \displaystyle\int_0^1 \frac{\ln(x)}{x-1}\,dx$,

II Méthode des trapèzes

La méthode des trapèzes consiste à remplacer les rectangles des sommes de Riemann par des trapèzes rectangles de même base. On approche alors $\displaystyle\int_a^b f(t)\,dt$ par les sommes

$$\frac{b-a}{2n}\sum_{k=0}^{n-1}\left(f\left(a+\frac{k}{n}(b-a)\right)+f\left(a+\frac{k+1}{n}(b-a)\right)\right) = \frac{1}{2}\left(S_n(f)+T_n(f)\right), \qquad n\in\mathbb{N}^*.$$

On peut montrer que, si f est de classe C^2 sur $[a,b]$, alors

$$\forall n\in\mathbb{N}^*, \qquad \left|\frac{1}{2}\left(S_n(f)+T_n(f)\right)-\int_a^b f(t)\,dt\right| \leqslant \frac{(b-a)^3}{12n^2}\max_{[a,b]}|f''|.$$

Autrement dit cette quantité converge vers l'intégrale avec une vitesse $\dfrac{1}{n^2}$ (contre $\dfrac{1}{n}$ pour la méthode des rectangles).

Exercice 4. (★★)

1) Écrire un programme qui calcule une approximation de $\displaystyle\int_0^1 \frac{dx}{1+x^2}$ avec la méthode des trapèzes.

2) A l'aide d'une boucle `while`, vérifier sur cet exemple que cette méthode est bien plus rapide que la méthode des rectangles et la méthode du point milieu.

Feuille de TP n° 12

Simulation et représentation de variables aléatoires à densité

Ce TP accompagne le chapitre 6 (Informatique et Algorithmique) : **Probabilités avec Scilab**.

I Simulation de variables aléatoires réelles à densité

1) Simulations avec `rand`

a) Loi uniforme

Soient `a` et `b` des entiers tels que `a` $<$ `b`. Justifier brièvement que le nombre renvoyé par la commande `X=a+(b-a)*rand()` peut être assimilé à la réalisation d'une variable aléatoire de loi uniforme sur $[\![\texttt{a}, \texttt{b}]\!]$. Exécuter les commandes suivantes dans la console :

```
-->a=-3; b=5; X=a+(b-a)*rand();
-->X=a+(b-a)*rand(4,7);
```

Exercice 1 – Modélisation de l'exercice 10 du TD n° 27. (★) On dispose d'un bâton d'un mètre. On le casse en deux morceaux en choisissant le point de cassure au hasard.

1) Écrire une commande qui simule cette expérience et renvoie la longueur du plus grand morceau.
2) Recommencer $N = 10000$ fois cette expérience et calculer la longueur moyenne du plus grand morceau.

b) Loi exponentielle

Soient $a \in \mathbb{R}_+^*$ et U est une variable aléatoire de loi uniforme sur $]0, 1[$. On a alors $1 - U > 0$ presque sûrement et $X = -\dfrac{1}{a}\ln(1-U)$ est une variable aléatoire de loi $\mathcal{E}(a)$ (cf. chapitre 6 d'Info).

Exécuter les commandes suivantes dans la console :

```
-->a=3; X=-log(1-rand())/a;
-->a=1/2; X=-log(1-rand(5,3))/a;
```

Exercice 2. (★) Écrire un programme qui simule la durée de vie (en année) de 10 ampoules dont la durée de vie moyenne est de 4 ans et qui renvoie la durée de vie maximale et la durée de vie cumulée.

Exercice 3 – Modélisation de l'exercice 2 du TD n° 27. (★★) Soit $X_\lambda \hookrightarrow \mathcal{E}(\lambda)$ avec $\lambda > 0$ et $Y_\lambda = \lfloor X_\lambda \rfloor + 1$.

1) Écrire une fonction qui prend en entrée $\lambda > 0$ et $N \in \mathbb{N}^*$ et qui renvoie un vecteur contenant N réalisations indépendantes de variables aléatoires de même loi que Y_λ.
2) Tracer l'histogramme renormalisé des réalisations lorsque $N = 10000$ pour différentes valeurs de λ.
3) Superposer à l'histogramme obtenu le diagramme en bâton d'une loi géométrique bien choisie.

2) Simulations avec `grand`

Exécuter les commandes suivantes dans la console :

```
-->U=grand(7,2,'unf',0,1)
-->V=grand(1,9,'unf',-3.2,5.4)
-->X=grand(3,10,'exp',1/6.5)
```

```
-->Y=grand(5,5,'nor',0,1)
-->Z=grand(6,2,'nor',-4,2)
-->T=grand(6,2,'nor',3,5)
```

Simuler une variable aléatoire de loi Normale de moyenne $m = 2$ et de variance $\sigma^2 = 1/2$.

II Représentation graphique de lois à densité

1) Représentation graphique de lois théoriques

Exercice 4. (★★) Écrire une fonction appelée `FRepNorm.sci` qui prend en entrée un réel x et qui renvoie une valeur approchée de $\Phi(x) = \displaystyle\int_{-\infty}^{x} e^{-t^2/2}\frac{dt}{\sqrt{2\pi}}$ à l'aide de la méthode des rectangles.

On pourra utiliser le fait que $\Phi(x) = \dfrac{1}{2} + \mathrm{sg}(x)\displaystyle\int_0^{|x|} e^{-t^2/2}\frac{dt}{\sqrt{2\pi}}$ *avec* $\mathrm{sg}(x) = \begin{cases} -1 & \textit{si} \quad x < 0 \\ 1 & \textit{si} \quad x > 0 \\ 0 & \textit{si} \quad x = 0 \end{cases}$.

Pour tracer la fonction de répartition de la loi $\mathcal{N}(\mathtt{m}, \mathtt{s}^2)$, on utilise le programme de l'exercice 4 et les propriétés qui la relie avec Φ. On peut aussi utiliser la commande prédéfinie

```
cdfnor("PQ",x,m*ones(x),s*ones(x));
```

avec `x` un vecteur représentant un intervalle discrétisé (cf. l'aide Scilab pour plus de précisions que `cdfnor`).

Exercice 5. (★★)

1) Écrire une programme qui trace dans la même fenêtre (une ligne par loi), côte-à-côte une représentation graphique d'une densité et de la fonction de répartition des variables de lois $\mathcal{U}([0,1])$, $\mathcal{E}(1)$ et $\mathcal{N}(0,1)$.
2) Modifier ce programme pour qu'il prenne en entrée des paramètres a, b, λ, m, σ^2 et pour qu'il trace une représentation graphique d'une densité et de la fonction de répartition des variables de lois $\mathcal{U}([a,b])$, $\mathcal{E}(\lambda)$ et $\mathcal{N}(m,\sigma^2)$.

2) Représentation graphique de phénomènes aléatoires continus

Exercice 6. (★★★) Écrire une fonction Scilab qui :

- prend en entrée un réel `a` strictement positif et un entier naturel `n`,
- simule `n` réalisations d'une variable aléatoire de loi $\mathcal{E}(\mathtt{a})$ et les stocke dans un vecteur X.
- ouvre une fenêtre graphique contenant deux zones,
- dans la première zone :
  - — construit l'histogramme renormalisé des données de `X`,
  - — lui superpose la courbe de la densité d'une variable aléatoire de loi $\mathcal{E}(\mathtt{a})$.
- dans la deuxième zone :
  - — construit la fonction de répartition empirique des données de `X`,
    On utilisera les commandes `tabul` *et* `cumsum`.
  - — lui superpose la courbe de la fonction de répartition d'une variable aléatoire de loi $\mathcal{E}(\mathtt{a})$.

Tester cette fonction avec $\mathtt{a} \in \{1/2, 1, 2\}$ et $\mathtt{n} \in \{10, 100, 1000, 10000\}$

Exercice 7. (★★★) Même exercice avec les lois $\mathcal{U}([a,b])$ et $\mathcal{N}(m,\sigma^2)$.

Feuille de TP n° 13

Convergences et approximations de variables aléatoires

Ce TP accompagne le chapitre 6 (Informatique et Algorithmique) : **Probabilités avec Scilab**.

Exercice 1 – Simulation d'une loi de Poisson. (★★)

1) Ouvrir un script et recopier le code suivant :

```
function   U=PoissonTheo(a,m)
u=exp(-a); U=[u];
for   k=1:m
      u=u*a/k;
      U=[U,u];
end
endfunction
```

Enregistrer et exécuter cette fonction. Que fait-elle ?

2) a) Écrire un programme qui demande à l'utilisateur un réel `a` strictement positif et un entier naturel non nul `N`, qui simule `N` réalisations indépendantes de variables aléatoires de loi de Poisson de paramètre `a` (et les stocke dans un vecteur `X`) et qui trace le diagramme en bâtons à partir de ces réalisations.
On utilisera les commandes `grand`, `tabul`, `plot2d3(,style=5)` et `clf()`.

b) Modifier ce programme en ajoutant les commandes

```
m=max(X); Y=PoissonTheo(a,m); plot2d3(0:m,Y,style=-1);
```

c) Tester ce programme pour différentes valeurs de `a` et de `N`. Commenter.

3) a) Ouvrir un nouveau script et copier/coller le programme de la question précédente dans son intégralité.

b) Modifier-le de telle sorte qu'on n'utilise plus la fonction `grand`. On utilisera cette fois le fait que l'on peut approcher une loi $\mathcal{P}(\mathtt{a})$ par une loi $\mathcal{B}(\mathtt{n}, \mathtt{a/n})$ pour `n` grand.

c) Tester ce programme pour différentes valeurs de `a` et de `N`. Commenter.

Exercice 2 – Illustration de la loi des grands nombres. (★★)

1) Si $x = (x_1, \dots, x_n) \in \mathbb{R}^n$, on appelle vecteur des moyennes cumulées de x le vecteur

$$\left(x_1, \frac{x_1+x_2}{2}, \frac{x_1+x_2+x_3}{3}, \dots, \frac{x_1+\dots,x_n}{n}\right).$$

Écrire une fonction appelée `cummoy.sci` qui prend en entrée un vecteur `v`, et qui renvoie un vecteur contenant les moyennes cumulées des coordonnées de `v`.

2) a) Écrire un programme qui demande à l'utilisateur un réel $\mathtt{p} \in]0,1[$ et un entier naturel `n` non nul, qui simule n réalisations $x_1, \dots, x_n$ de variables aléatoires de loi de Bernoulli de paramètre `p` et qui représente graphiquement $\dfrac{x_1+\dots+x_k}{k}$ en fonction de k pour k variant de 1 à `n`.

b) Tester ce programme pour différentes valeurs de `p` et de `n`. Commenter.

3) Recommencer la question précédente en remplaçant la loi binomiale par une loi géométrique, une loi de Poisson, une loi exponentielle puis une loi Normale. Commenter.

Exercice 3 – Illustration du théorème central limite. (★★)

1) Écrire un programme qui
   - demande à l'utilisateur un réel `p` $\in]0,1[$ et un entier naturel `n` non nul,
   - forme un vecteur `T` composé de 1000 réels $t_1, \dots, t_{1000}$ où

$$\forall k \in [\![1, 1000]\!], \qquad t_k = \frac{x_k - \texttt{np}}{\sqrt{\texttt{np(1-p)}}},$$

     avec x_k une réalisation d'une variable aléatoire de loi de binomiale de paramètres `n` et `p`,
   - trace l'histogramme normalisé des valeurs de `T` et lui superpose la courbe de la densité de la loi Normale centrée réduite.
   - sur un autre graphique (mais sur une même fenêtre... on utilisera `subplot`), trace la fonction de répartition empirique (on utilisera `tabul` et `cumsum`) des valeurs de `T` et lui superpose la courbe de la fonction de répartition Φ de la densité de la loi Normale centrée réduite.

   Exécuter ce programme pour différentes valeurs de p et n. Commenter.

2) Écrire un programme qui
   - demande à l'utilisateur un réel $a > 0$ et un entier naturel `n` non nul,
   - forme un vecteur `T` composé de 1000 réels $t_1, \dots, t_{1000}$ où

$$\forall k \in [\![1, 1000]\!], \qquad t_k = \frac{x_k - \texttt{na}}{\sqrt{\texttt{na}}},$$

     avec x_k une réalisation d'une variable aléatoire de loi de Poisson de paramètre na (on utilisera `grand`),
   - trace l'histogramme normalisé des valeurs de `T` et lui superpose la courbe de la densité de la loi Normale centrée réduite.
   - sur un autre graphique (mais sur une même fenêtre... on utilisera `subplot`), trace la fonction de répartition empirique (on utilisera `tabul` et `cumsum`) des valeurs de `T` et lui superpose la courbe de la fonction de répartition Φ de la densité de la loi Normale centrée réduite.

   Exécuter ce programme pour différentes valeurs de a et n. Commenter.

Exercice 4. (★★★) A l'approche des fêtes de Noël, les élèves de S1B ont décidé de s'offrir des cadeaux selon le protocole suivant : un sac opaque contient les noms de tous les élèves (écrits chacun sur un morceau de papier). Chacun leur tour, les élèves tirent un nom au hasard parmi les noms restants au moment du tirage. Chaque élève devra offrir un cadeau à l'élève dont il a tiré le nom [1].

Pour simplifier on numérote les élèves de 1 à n et on remplace leur nom par leur numéro.

1) Écrire une fonction qui prend un entier naturel n non nul en entrée et qui renvoie une liste `L` de n numéros de telle sorte que, pour tout $k \in [\![1, n]\!]$, la $k^{\text{ième}}$ coordonnée contient le numéro de l'élève tiré par le $k^{\text{ième}}$ élève.

   On pourra introduire une liste contenant tous les numéros disponibles, choisir un numéro au hasard dans la liste, l'enlever et recommencer ainsi jusqu'à ce qu'elle soit vide.
2) Tester la fonction avec différentes valeurs de n.
3) Modifier la fonction pour qu'elle renvoie cette fois la proportion d'élèves ayant pioché leur propre nom. Tester-la avec différentes valeurs de n.
4) Si aucun élève n'a tiré son nom on dit que le tirage a réussi. Recommencer 100000 fois ce tirage avec $n = 100$ et calculer la proportion de tirages réussis. Remarquer que le nombre obtenu est proche de $1/e$.

   On a déjà rencontré une situation analogue dans l'exercice 18 du chapitre 9 (avec des parapluies).

1. Ils peuvent aussi demander à leurs professeurs de participer à ce jeu... mais ça ne change rien à l'exercice.

Exercice 5 – Marche aléatoire et loi de l'arc sin. (★★★) On lance une infinité de fois une pièce équilibrée de façon indépendance. La suite de variables aléatoires $(S_n)_{n\in\mathbb{N}}$ définie par $S_0 = 0$ et

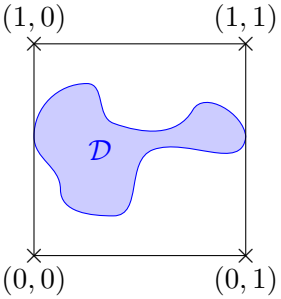
$$\forall n \in \mathbb{N}, \qquad S_{n+1} - S_n = \begin{cases} 1 & \text{si le } n^{\text{ième}} \text{ lancer est tombé sur Pile} \\ -1 & \text{si le } n^{\text{ième}} \text{ lancer est tombé sur Face} \end{cases}$$

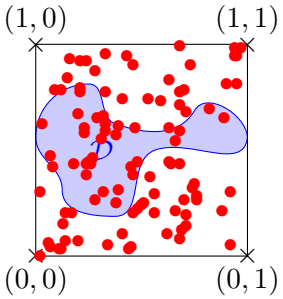
est appelée marche aléatoire simple symétrique sur $\mathbb{Z}$. On dit qu'elle visite 0 à un instant $k \in \mathbb{N}^*$ si $S_k = 0$. Pour tout $n \in \mathbb{N}^*$, on note D_n le dernier instant de visite en 0 avant le $n^{\text{ième}}$ pas (il vaut 0 si la marche n'est pas revenue en 0).

1) Écrire une fonction `marchealeatoire` qui prend en entrée un entier naturel n et qui renvoie une simulation de D_n. C'est-à-dire elle construit un vecteur $[S_0, S_1, \dots, S_n]$ correspondant aux n premiers pas d'une marche aléatoire et elle renvoie le dernier instant où la marche a visité 0 lors des n premiers pas.
2) Simuler 10000 fois la variable aléatoire $\dfrac{D_n}{n}$ avec $n = 1000$ et construire un histogramme normalisé.
3) Superposer à cet histogramme la courbe de la fonction $x \longmapsto \dfrac{1}{\pi\sqrt{x(1-x)}}$ sur l'intervalle $]0,1[$. Commenter.

Exercice 6 – Méthode de Monte-Carlo. (★★★) On cherche à approcher l'intégrale d'une fonction g définie sur $[0,1]$ et à valeurs dans $[0,1]$ (par exemple la fonction $x \mapsto \dfrac{1}{1+x^2}$ définie sur $[0,1]$ et qui admet pour intégrale $\dfrac{\pi}{4}$). Si on trace le carré de côté 1 et la courbe $\mathscr{C}_g$ de g sur $[0,1]$, l'intégrale recherchée est l'aire de la partie du carré qui se situe sous la courbe.

Plus généralement comment approcher l'aire d'une partie[1] $\mathcal{D}$ du plan (incluse dans le carré de côté 1 pour simplifier) ? On peut appliquer pour cela l'algorithme dit de Monte-Carlo : on définit X et Y deux variables aléatoires indépendantes de loi uniforme sur $[0,1]$. On déclare alors qu'une réalisation (x,y) de (X,Y) est un succès si $(x,y) \in \mathcal{D}$. On admet[2] que la probabilité de succès est égale à l'aire de $\mathcal{D}$.





La loi des grands nombres nous indique alors que si on répète un grand nombre de fois cette expérience, alors la proportion moyenne de succès se rapproche de la probabilité de succès. Par exemple voici à droite ce que l'on peut obtenir si on réalise 100 fois cette expérience. On compte 34 succès (34 points se trouvent dans la zone $\mathcal{D}$ du plan) si bien qu'une première approximation de l'aire de $\mathcal{D}$ est $0,34$. Il faudrait bien entendu réaliser davantage d'expériences pour avoir un résultat plus précis.

1) Écrire une fonction Scilab appelée `g.sci` qui prend $x \in [0,1]$ en entrée et qui renvoie $\dfrac{1}{1+x^2}$.
2) Écrire un programme Scilab qui demande à l'utilisateur un entier naturel n et qui simule n fois cette expérience (cette fois une réalisation (x,y) d'un couple de v.a (X,Y) indépendantes et de lois uniformes sur $[0,1]$ est un succès si $y \leqslant \mathrm{g}(x)$) et calcule la proportion de succès.
3) En déduire une approximation de $\dfrac{\pi}{4}$.

1. une partie « mesurable », pas trop compliquée en clair...
2. Mais c'est intuitif n'est-ce pas ? Pour le montrer rigoureusement, il nous faudrait mieux connaître les couples de variables aléatoires.