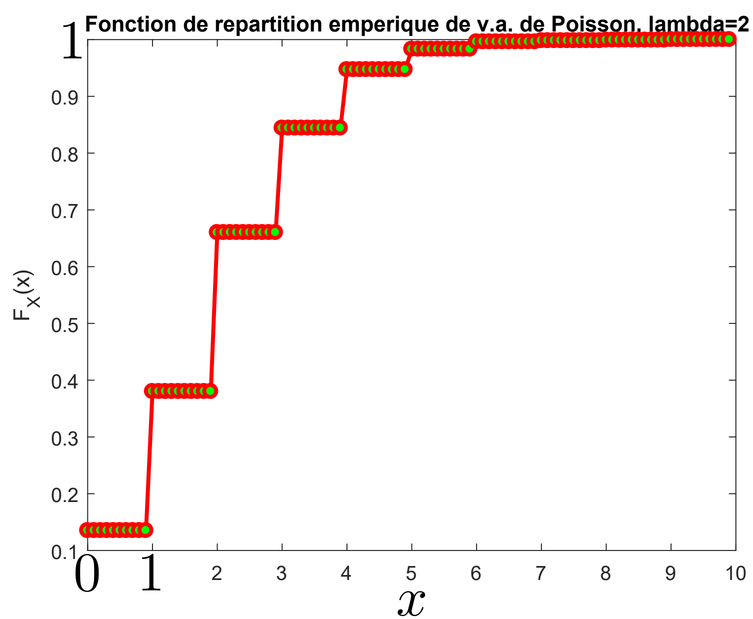
Simulation Monte Carlo

TP 1: Simulation de variables aléatoires discrètes

Irina Kortchemski, CYTECH



Théorie

- **Loi forte des grands nombres**

  Soient $X_1, X_2, ..., X_n$ des variables aléatoires de même loi. On note $\overline{X}_n = \frac{1}{n}\sum_{i=1}^{n} X_i$. Alors

$$\lim_{n\to\infty} \overline{X}_n = \mathbb{E}(X) \quad presque \quad surement$$

- **Théorème Central Limite**

  Soient $X_1, X_2, ..., X_n$ des variables aléatoires indépendantes et identiquement distribuées d'espérance μ et de variance σ^2.

  Alors la loi de

$$Y_n = \sqrt{n}\frac{\overline{X}_n - \mu}{\sigma}$$

  tend vers la loi normale centrée réduite. En d'autre termes, pour tous a et b réels,

$$\lim_{n\to\infty} \mathbb{P}[a \leq \sqrt{n}\frac{\overline{X}_n - \mu}{\sigma} \leq b] = \mathbb{P}[a \leq Z \leq b]$$

  ou Z est une variable gaussienne centrée réduite, $Z \sim \mathcal{N}(0,1)$.

Intervalle de confiance

- On choisi $a = -1.96, b = 1.96$. Donc

$$\lim_{n\to\infty} \mathbb{P}[|\sqrt{n}\frac{\overline{X}_n - \mu}{\sigma}| \leq 1.96] = 0.95$$

- Un échantillon de n valeurs de v.a. X_i est utilisé pour estimer l'espérance inconnue $\mathbb{E}(X) = \mu$.
  Pour grand n la probabilité que μ se trouve dans l'intervalle de confiance:

$$\overline{X}_n - \frac{1.96 \cdot \sigma}{\sqrt{n}} \leq \mu \leq \overline{X}_n + \frac{1.96 \cdot \sigma}{\sqrt{n}}$$

  est égale à 0.95.

- Pour grand n ce résultat reste vrai quand σ est remplacé par l'quart-type de variance empirique $\sqrt{\mathbb{V}ar_{emp}(X)} = (\frac{1}{n}\sum_{i=1}^{n}(X_i - \overline{X}_n)^2)^{\frac{1}{2}}$:

$$\overline{X}_n - \frac{1.96\sqrt{\mathbb{V}ar_{emp}(X)}}{\sqrt{n}} \leq \mu \leq \overline{X}_n + \frac{1.96\sqrt{\mathbb{V}ar_{emp}(X)}}{\sqrt{n}}$$

Simulation d'une v.a. de Bernoulli X

- V.a. de Bernoulli X modélise des succès et des échecs:

$$X = \begin{cases} 1\ (succes) & avec\ \ proba\ \ p_1 = 1/3 \\ 0\ (echec) & avec\ \ proba\ \ p_2 = 2/3 \end{cases}$$

- Simuler une v.a. X signifie simuler un échantillon des valeurs (réalisations indépendantes) de X.
- Comment simuler une chaîne de valeurs (N_{mc} valeurs) 0 et 1 telle que les fréquences d'apparition de 1 et de 0 sont respectivement 1/3 et 2/3?
- Soit une probabilité $0 \leq p \leq 1$
  IDEE PRINCIPALE $\mathbb{P}[rand() \leq p] = p$
  C'est la définition de la v.a. uniforme U par sa fonction de répartition: $[0, 1] \rightarrow [0, 1] : \quad x \rightarrow F_U(x) = x$

Simulation d'une v.a. de Bernoulli X

- Intuitivement:
  - for $n = 1 : N_{mc}$
  - U= rand()
  - if U <1/3

    $X(n) = 1$

    else $X(n) = 0$
  - end if
  - end for
- Display(X)

- L'autre possibilité
  - U= rand(N_{mc})
  - $X = (U < 1/3)$

Vérification de la simulation.

- On vérifie que les probabilités des succès et des échecs dans la chaîne sont en effet 1/3 et 2/3.
- Plus N_{mc} est élevé plus les fréquences sont proches aux probabilités théoriques.
- • function[proba1, proba2]=Freq(N_{mc})
  ○ counter1=0; counter2=0
  • for $n = 1 : N_{mc}$
  • U= rand()
  • if U <1/3
  ○ $X(n) = 1$
  ○ counter1=counter1+1
  ○ else $X(n) = 0$
  ○ counter2=counter2+1
  • end if
  • end for
  • proba1=counter1/N_{mc}
  • proba2=counter2/N_{mc}
  • endfunction
- • for $N_{mc} = 1 : 200$
  • [proba1(N_{mc}),proba2(N_{mc})]= Freq(N_{mc})
  • end for
  • plot(proba1,'r')
  • plot(proba2,'g')

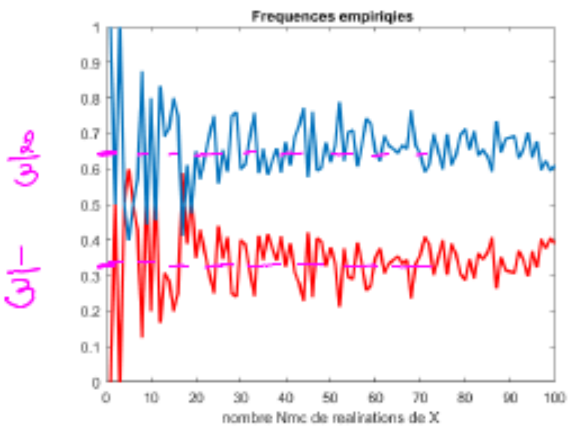
MATLAB

```
from numpy.random import rand
import matplotlib.pyplot as plt

def Frequence(Nmc):
    counter1=0.0
    counter2=0.0
    X=[]

    for n in range(0,Nmc):
        U=rand()

        if U<1/3.0:
            X.append(1)
            counter1=counter1+1
        else:
            X.append(0)
            counter2=counter2+1
    prob1=counter1/Nmc
    prob2=counter2/Nmc
    return (prob1, prob2)

Nmc_lim=1000

proba1=[]
proba2=[]

for Nmc in range(1,Nmc_lim+1):

    Y=Frequence(Nmc)
    proba2.append(Y[1])
    proba1.append(Y[0])

fig=plt.figure()

plt.plot(proba1)
plt.plot(proba2,"r")
```

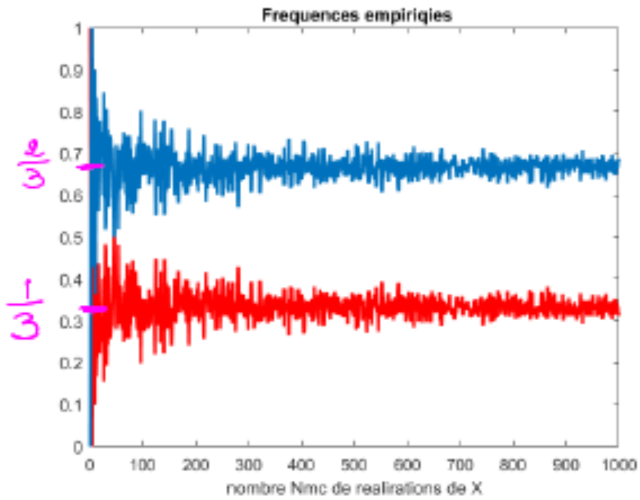
PYTHON

```
Taille=1000;
Proba1=zeros(1,Taille);
Proba2=zeros(1,Taille);


for Nmc=1:Taille
    [Proba1(Nmc), Proba2(Nmc)]=Frequence(Nmc);
end
figure;
    plot(Proba1,'r','LineWidth',2);
    hold on
    plot(Proba2,'LineWidth',2);
    xlabel 'nombre Nmc de realirations de X'
    title ' Frequences empiriqies'


function[proba1, proba2]=Frequence(Nmc)
X=zeros(1,Nmc);
counter1=0;
counter2=0;
for n=1:Nmc
    U=rand();
    if U<1/3
        X(n)=1;
        counter1=counter1+1;
        else
        X(n)=0;
        counter2=counter2+1;
    end
end
proba1=counter1/Nmc; % frequence empirique
proba2=counter2/Nmc;
end
```

Conclusion : Plus la taille de l'échantillon est élevée plus les simulations sont précises : fréquences empiriques $\longrightarrow$ probas théoriques



$N_{mc} = 1:100$

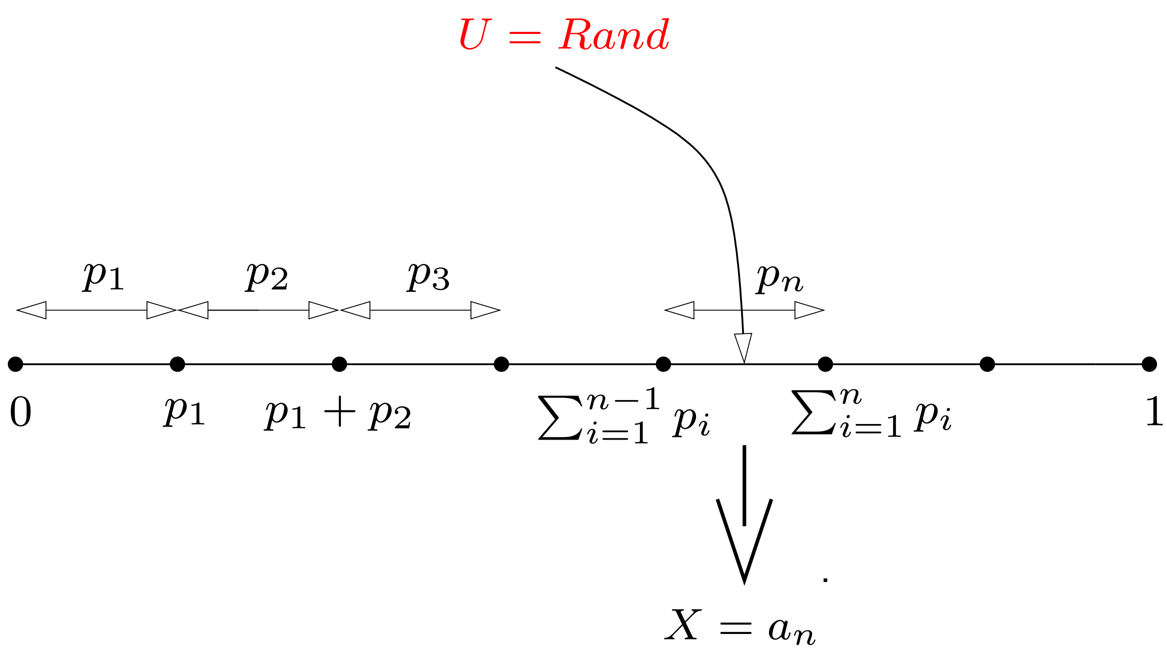


$N_{mc} = 1:1000$

Simulation d'une v.a. discrete X

- Variable aléatoire X est définie par un tableau:

$\mathbb{P}$	$p_1 = 0.25$	$p_2 = 0.3$	$p_3 = 0.1$	$p_4 = 0.18$	$p_5 = 0.05$	$p_6 = 0.08$	$p_7 = 0.02$	$p_8 = 0.02$
X	$a_1 = 1$	$a_2 = 2$	$a_3 = 3$	$a_4 = 4$	$a_5 = 5$	$a_6 = 6$	$a_7 = 7$	$a_8 = 8$



- Idée de simulation: on cherche dans quel intervalle tombe une v.a. uniforme U.

Inversion de la fonction de répartition.

- Justification à partir de la théorie:

$$\sum_{i=1}^{n-1} p_i < \color{red}{Rand(\,)} \leq \sum_{i=1}^{\color{blue}{n}} p_i \quad \Rightarrow X = a_{\color{blue}{n}}$$

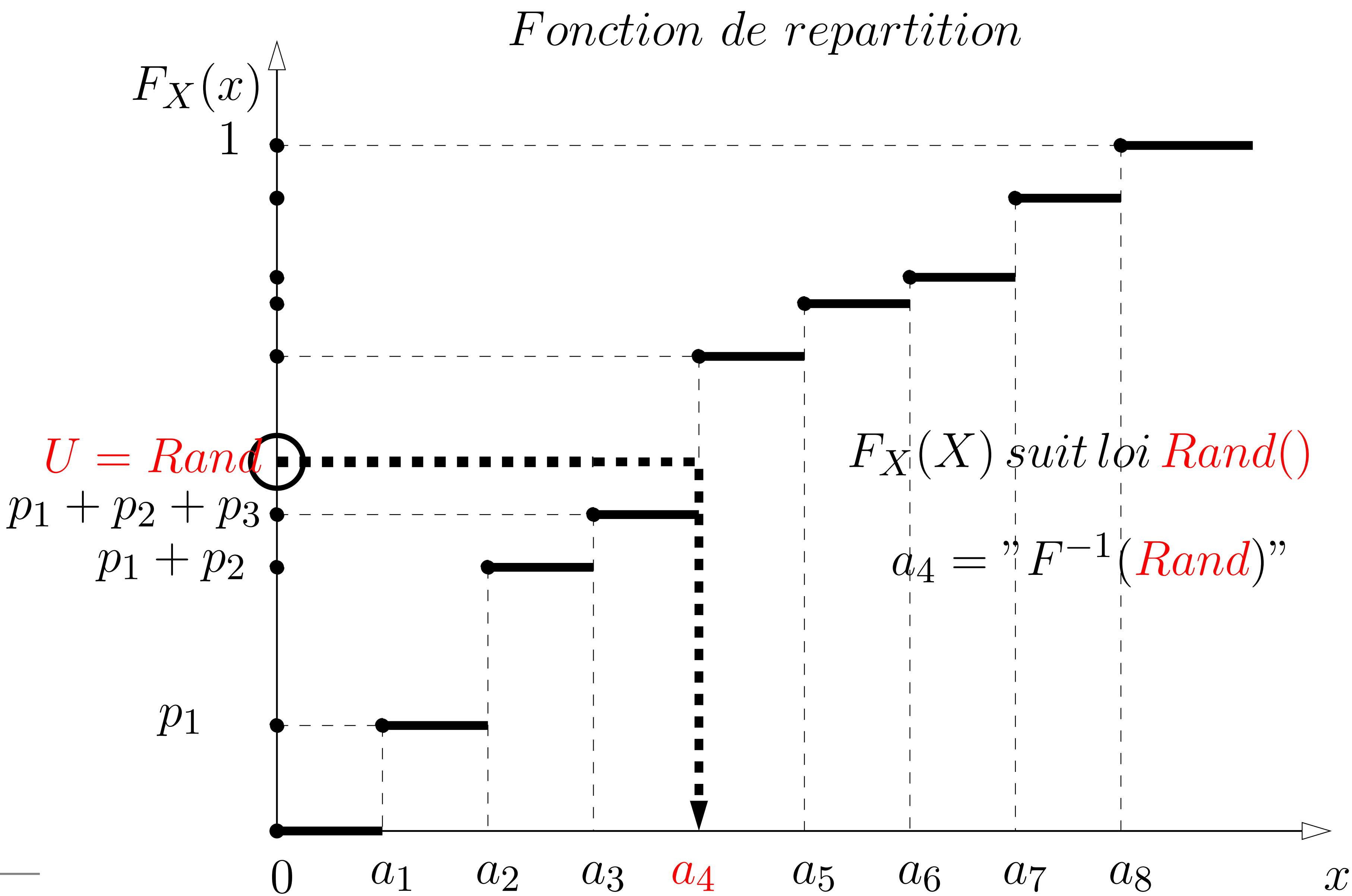
- Fonction de répartition apparaît:

$$F_X(a_{n-1}) < \color{red}{Rand(\,)} \leq F_X(a_{\color{blue}{n}}) \quad \Rightarrow X = a_{\color{blue}{n}}$$

- Théorème:
  - $F_X(X) \quad suit \quad la \quad loi \quad \color{red}{Uniforme}$
  - $F_X^{-1}(\color{red}{Rand(\,)}) = X$

- Définition de la fonction de répartition inverse $F_X^{-1}(u) = \{\inf x : F_X(x) \geq u\}$.

Simulation d'une v.a. discrete X



Simulation d'une v.a. discrete X

- Algorithme 1A: Simulation d'une réalisation de variable aléatoire discrete

  ◦ Définir le vecteur de probabilité $p(n)$

  ◦ Définir le vecteur a des réalisations $a(n)$ de la variable aléatoire X.

  - function[X]=V_A_Discrete(p, a)

  ◦ Set $n = 1, \quad F = p(1)$

  ◦ Générer U % U=rand()

  ◦ While $U > F$

  ◦ Set $F = F + p(n + 1), \quad n = n + 1$

  ◦ end while

  ◦ Set $X = a(n)$

  - endfunction

- On applique l'Algorithme 1A N_{mc} fois pour générer N_{mc} réalisations de X

MATLAB

```
clear;

clc;

p=[0.25 0.3 0.1 0.18 0.05 0.08 0.02 0.02];
a=[1 2 3 4 5 6 7 8];
Nmc=100;
disp(sum(p));
V_A_Discrete(p,a);
X=Chaine_valeurs_V_A(p,a,Nmc);
disp(X)

function[X]=V_A_Discrete(p,a)
   n=1;
   F=p(1);
   U=rand();
   while U>F
      F=F+p(n+1);
      n=n+1;
   end
   X=a(n);
end


function[X]=Chaine_valeurs_V_A(p,a,Nmc)
  esp=0;
  var=0;
   for n=1:Nmc
      X(n)=V_A_Discrete(p,a);
      esp=esp+X(n);
      var=var+X(n)^2;
   end
   esperance=esp/Nmc;
   variance=var/Nmc-(esperance)^2;
   disp('esperance empirique');
   disp(esperance);
   disp('variance empirique');
   disp(variance);

   esperance_th=p*a';
   %variance_th=(a-esperance_th).^2*p';
   variance_th=a.^2*p-esperance_th^2;
   disp('esperance theorique');
   disp(esperance_th);
   disp('variance empirique');
   disp(variance_th);

end
```

$a.\hat{\ }2 \cdot p' = (1\ 4\ 9\ 16\ \ ) \cdot \begin{pmatrix} p_1 \\ p_2 \\ \vdots \\ p_8 \end{pmatrix} = \sum_{i=1}^{8} a_i^2 p_i = \mathbb{E}[X^2]$

$\frac{1}{N_{mc}} \sum_{n=1}^{N_{mc}} X(n)$

espérance empirique

$Var[X] = \mathbb{E}[X^2] - (\mathbb{E}[X])^2$

$\frac{1}{N_{mc}} \sum_{n=1}^{N_{mc}} X(n)^2$

$(\ldots\ldots)\begin{pmatrix} \vdots \end{pmatrix} = \sum^{8} p_i a_i$

espérance théorique

comment

a' – vecteur transposé , $a.\hat{\ }2 = (1\ 4\ 9\ \ldots\ )$

$\mathbb{E}[(X - \mathbb{E}[X])^2]$

Algo 1B: Simulation d'un échantillon

- On répète l'algorithme 1A N_{mc} fois pour simuler une chaîne de N_{mc} valeurs.
- Algorithme 1B de simulation d'une chaîne de valeurs de v.a. discrete.

  • function [X]= Chaine_valeurs_V_A_Discrete (p, a)
  ◦ for $n = 1 : N_{mc}$
  ◦ $X(n)$=V_A_Discrete(p, a)
  ◦ end
  • endfunction
- Simuler et visualiser une chaîne de réalisations de X.

Verification de simulation.

- On vérifie que la loi simulée est bien la loi demandée. Il existe trois tests de validation.
- Test1 On compare l'espérance d'un échantillon simulé avec l'espérance théorique.
- Test2 On compare la variance d'un échantillon simulé avec la variance théorique.
- Test3 On construit la fonction de repartition et la fonction de densité d'un échantillon simulé (les fonctions de repartition et de densité empiriques) et on les compare avec les fonctions de repartition et de densité analytiques da la loi demandée.

Verification de simulation.

Espérance et variance

- Calcul théorique pour l'espérance et la variance.
  - Calculer l'espérance théorique : $\mathbb{E}(X) = \sum_{i=1}^{8} p_i a_i$
  - Calculer la variance théorique $\mathbb{V}ar(X) = \sum_{i=1}^{8} p_i a_i^2 - \mathbb{E}(X)^2$
- Travail à faire. Calcul empirique. Soit $N_{mc} = 100$
  - Calculer l'espérance empirique : $\mathbb{E}_{emp}(X) = \frac{1}{N_{mc}} \sum_{i=1}^{N_{mc}} X_i$
  - Calculer la variance empirique :

  $\mathbb{V}ar_{emp}(X) = \frac{1}{N_{mc}} \sum_{i=1}^{N_{mc}} (X_i)^2 - (\frac{1}{N_{mc}} \sum_{i=1}^{N_{mc}} X_i)^2$
- Est-ce que le résultat est coherent avec la loi des Grands Nombres?

$$\lim_{N_{mc}\to\infty} \frac{1}{N_{mc}} \sum_{i=1}^{N_{mc}} X_i = \mathbb{E}(X)?$$

$$\lim_{N_{mc}\to\infty} \frac{1}{N_{mc}} \sum_{i=1}^{N_{mc}} (X_i)^2 - (\frac{1}{N_{mc}} \sum_{i=1}^{N_{mc}} X_i)^2 = \mathbb{V}ar(X)?$$

- Faites les mêmes calculs pour $N_{mc} = 1000$, $N_{mc} = 100000$. Conclure.

Simulation de v.a. de Poisson,1

- Soit X est v.a. de Poisson $\mathbb{P}(X = n) = \frac{e^{-\lambda}\lambda^n}{n!}$.
- Montrer $\mathbb{P}(X = n + 1) = \frac{\lambda}{(n+1)} \cdot \mathbb{P}(X = n)$
- Algorithme 2A de simulation de v.a. de Poisson
  - • function[X]=V_A_Poisson(λ)
  - ◦ Set $n = 0, \quad proba = e^{-\lambda}, \quad F = proba;$
  - ◦ Générer U
  - ◦ While $U > F$
  - ◦ Set $proba = \frac{\lambda}{(n+1)} \cdot proba, \quad F = F + proba, \quad n = n + 1$
  - ◦ end
  - ◦ Set $X = n$
  - ◦ end
  - • endfunction

Simulation de v.a. de Poisson,2

- Algorithme 2B de simulation de N_{mc} valeurs de v.a. de Poisson
  - function [X]= Chaine_valeurs_V_A_Poisson (λ)

  ○ for $n = 1 : N_{mc}$

  ○ $X(n)$=V_A_Poisson(λ)

  ○ end
  - endfunction
- Visualiser un échantillon simulé: $N_{mc} = 100$ terms, $\lambda = 2$
- Calcul théorique.
  - L'espérance théorique : $\mathbb{E}(X) = \sum_{n=1}^{\infty} n \cdot \frac{e^{-\lambda}\lambda^n}{n!} = \lambda$
  - La variance théorique $\mathbb{V}ar(X) = \sum_{n=1}^{\infty} n^2 \cdot \frac{e^{-\lambda}\lambda^n}{n!} = \lambda$
- Travail à faire. Calcul empirique. Soit $N_{mc} = 10000, \lambda = 2$
  - Calculer l'espérance empirique : $\mathbb{E}_{empirique}(X) = \frac{1}{N_{mc}} \sum_{i=1}^{N_{mc}} X_i$ et comparer avec λ
  - Calculer la variance empirique :

  $\mathbb{V}ar_{empirique}(X) = \frac{1}{N_{mc}} \sum_{i=1}^{N_{mc}} (X_i)^2 - (\frac{1}{N_{mc}} \sum_{i=1}^{N_{mc}} X_i)^2$ et comparer avec λ.
- Conclure.

```
import numpy as np
from numpy.random import rand
from math import *
import matplotlib.pyplot as plt
#import matplotlib.colors as colors

lambd=10.0
Nmc=1000
a=0.0
delta=0.2
print delta

def V_A_Poisson(lambd):
    n=0
    proba=exp(-lambd)
    F=proba
    U=rand()
    while U>F:
        proba=proba*lambd/(n+1)
        F=F+proba
        n=n+1
    return n



def Chaine_Valeurs_Poisson(lambd,Nmc):
    X=[]
    for n in range(0,Nmc):
        X.append(V_A_Poisson(lambd))
    return X

Y=Chaine_Valeurs_Poisson(lambd,Nmc)


print(Y)
print('Esperance empirique')
print(np.mean(Y))
print('Esperance théorique')
```

Verification de simulation.

Fonction de repartition.

- Il y a un échantillon de N_{mc} v.a. de Poisson. Nous utilisons la définition de la fonction de répartition $F_X(x) = \mathbb{P}[X \leq x], \quad x \in R$ est une variable classique,
- On choisit un intervalle $[a, b]$ sur lequel on veut tracer la fonction de repartition, on discrétise cet intervalle en N_x parties :

$$\Delta x = \frac{b-a}{N_x}, \quad x_i = a + \Delta x \cdot i, \quad 0 \leq i \leq N_x$$

On définit $F_X(x)$ en chaque point discrete x_i:

$$F_X(x_i) = \mathbb{P}[X \leq x_i].$$

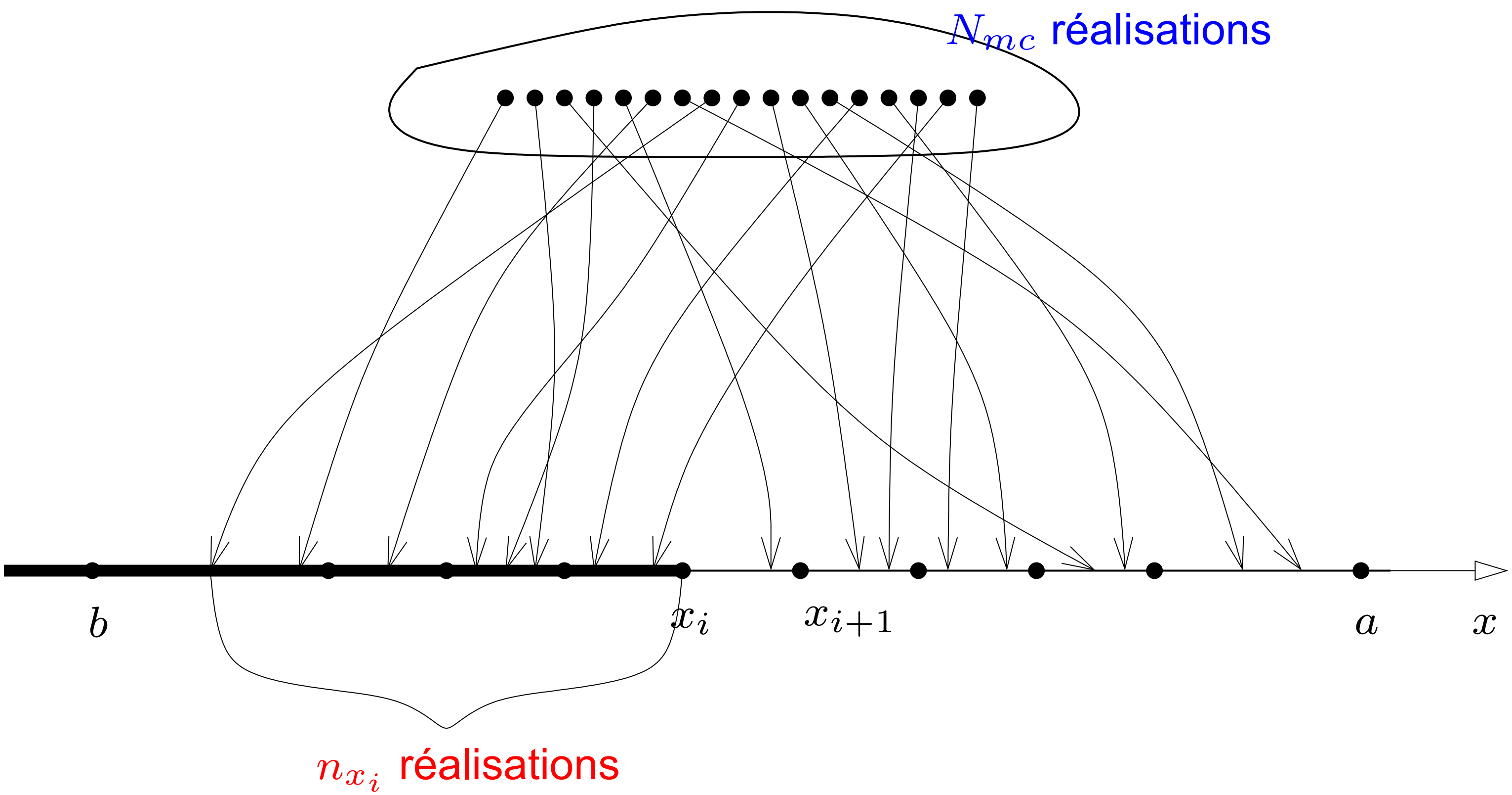
- On définit la fonction empirique de répartition $F_X(x)$ en chaque point discrete x_i:

$$F_X(x_i) = \mathbb{P}[X \leq x_i] = \mathbb{E}[1_{X \leq x_i}] = \frac{1}{N_{mc}} \sum_{n=1}^{N_{mc}} 1_{X_n \leq x_i}$$

Verification de simulation.

Fonctions de repartition.

- On calcule le nombre de réalisation n_{x_i} de v.a. X qui tombe dans l'intervalle $]-\infty, x_i]$, puis on calcule la probabilité: $\mathbb{P}[X \leq x_i] = \frac{n_{x_i}}{N_{mc}}$.
- On répète cette procedure pour chaque intervalle $]-\infty, x_i], \quad i = 1 : N_x$.



Verification de simulation.

Algorithme de simulation de la fonction de repartition

● Simuler un échantillon de N_{mc} réalisations de v.a. X et utiliser le vecteur X comme l'entrée.

● function [ ]= Repartition_Poisson (X)

○ Définir a, Δx et N_x

○ for $i = 1 : N_x + 1$ % Par une boucle définir les points x_i d'un vecteur x

○ $x(i) = a + \Delta x \cdot (i - 1)$

○ counter =0 % Initialisation du Compteur

○ for $n = 1 : N_{mc}$

○ if $X_n \leq x_i$

○ counter = counter +1

○ end if

○ end for

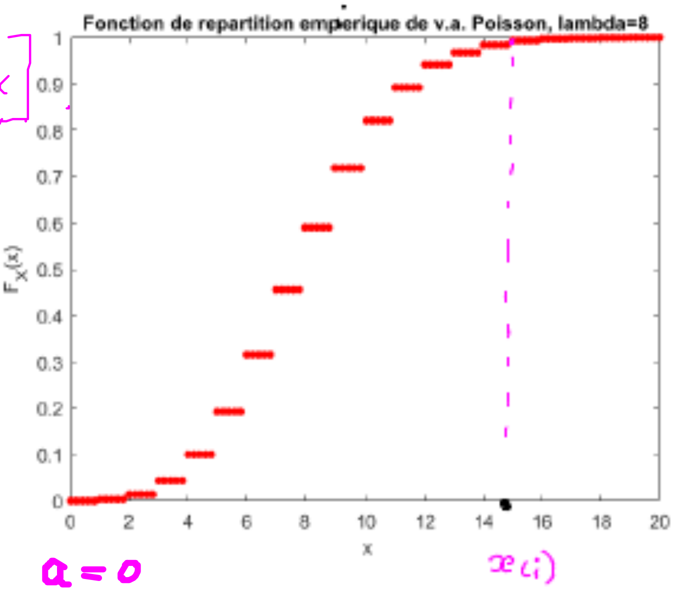
Proba(i)= $\frac{Counter}{N_{mc}}$ % Calcul la probabilité empirique

○ end for

○ plot(x, Proba) % Tracer le graphe de la fonction de repartition

● endfunction

```
clear;
Nmc=1000;
lambda=2; 8
a=0;
delta=0.1; 0.2
X=Chaine_valeurs_Poisson(lambda,Nmc);
repartition_graphe(a,delta,X);

function[X]=V_A_Poisson(lambda)
   n=0;
   proba=exp(-lambda);
   F=proba;
   U=rand();
   while U>F
      proba=proba*lambda/(n+1);
      F=F+proba;
      n=n+1;
   end
   X=n;
end

function[X]=Chaine_valeurs_Poisson(lambda,Nmc)

   for n=1:Nmc
      X(n)=V_A_Poisson(lambda);
   end
end


function[P,x]=fonction_repartition(X,a,delta)
N_x=100;
for i =1:N_x+1
x(i)=a+delta*(i-1);   x(i)∈[a,delta*N_x]
cont=0;
  for n=1:length(X)
if X(n)<=x(i)
   cont=cont+1;
end
  end
P(i)=cont/(length(X));
end
end
```




```
function[]=repartition_graphe(a,delta,X)
tic;

[P,x]=fonction_repartition(X,a,delta);
figure;
plot(x,P,'ro','MarkerSize',4,'MarkerFaceColor', 'r' );
xlabel 'x'
ylabel 'F_X(x)'
title 'Fonction de repartition empirique de v.a. Poisson, lambda=8'

disp(toc);
% 'LineWidth',1,
end
```

Verification de simulation.

Fonction de la densité.

- Nous utilisons la définition de la fonction de densité $f_X(x) = \frac{\mathbb{P}[x<X\leq x+\Delta x]}{\Delta x}$
- On choisit l'intervalle $[a, b]$, on discrétise cet intervalle en N_x parties :

$$\Delta x = \frac{b-a}{N_x}, \quad x_i = a + \Delta x \cdot i, \quad 0 \leq i \leq N_x$$

On définit $f_X(x)$ en chaque point discrete x_i:

$$f_X(x_i) = \frac{\mathbb{P}[x_i < X \leq x_i + \Delta x]}{\Delta x}.$$

On utilise l'échantillon de v.a. X, on fixe x_i, on calcule le nombre de réalisation $n_{x_i,x_i+\Delta x}$ qui tombe dans l'intervalle $[x_i, x_i + \Delta x]$, puis on calcule la probabilité:

$$\mathbb{P}[x_i < X \leq x_i{+}\Delta x] = \mathbb{E}[1_{x_i<X\leq x_i+\Delta x}] = \frac{1}{N_{mc}} \sum_{n=1}^{N_{mc}} 1_{x_i<X_n\leq x_i+\Delta x} = \frac{n_{x_i,x_i+\Delta x}}{N_{mc}}.$$

Cette procedure on répète pour chaque intervalle $[x_i, x_i + \Delta x], \quad i = 1 : N_x.$

Verification de simulation.

- Algorithme de simulation de fonction de densité

Pour un paramètre d'entré on utilise un échantillon de N_{mc} réalisations de v.a. X.

• function[Densite]= Densite_empirique(X)

∘ for $i = 1 : N_x$ % Par une boucle définir les points x_i

∘ $x(i) = a + \Delta x \cdot (i - 1)$

∘ counter=0 %Initialisation du Compteur

∘ for $n = 1 : N_{mc}$

∘ if $X(n) \in [x(i), x(i) + \Delta x]$

∘ counter = counter +1

∘ end if

∘ end for

∘ Proba(i)= $\frac{counter}{N_{mc}}$ % probabilité empirique

∘ Densité(i)=Proba(i)/Δx % Δx=1 si la v. a. est discrete

∘ end for

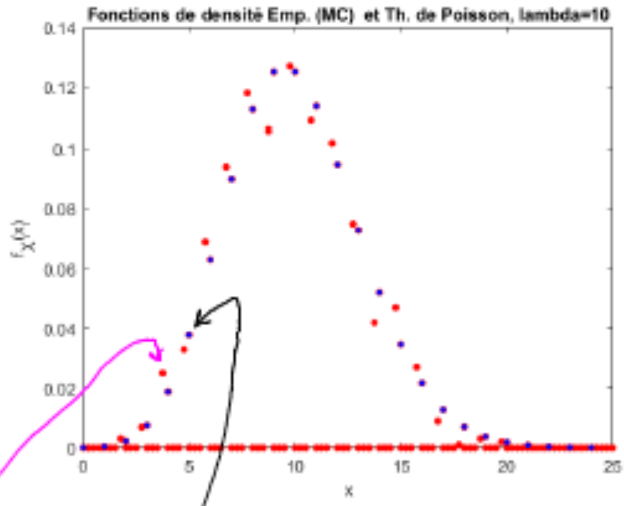
∘ plot(x, Densité) % Tracer le graphe de la fonction de densité

• endfunction

```
clear;
Nmc=10000;
lambda=10;
a=0;
delta=0.25;
X=Chaine_valeurs_Poisson(lambda,Nmc);
densite_Emp_graphe(a,delta,X);
hold on

for i =1:25
x(i)=(i-1);
densite_th(i)=densite_Th_Poisson(x(i),lambda);
end
plot(x,densite_th,'ro','MarkerSize',4,'MarkerFaceColor','b');

function[X]=V_A_Poisson(lambda)
    n=0;
    proba=exp(-lambda);
    F=proba;
    U=rand();
    while U>F
        proba=proba*lambda/(n+1);
        F=F+proba;
        n=n+1;
    end
    X=n;
end

function[X]=Chaine_valeurs_Poisson(lambda,Nmc)

    for n=1:Nmc
        X(n)=V_A_Poisson(lambda);
    end
end


function[P,x]=fonction_Emp_densite(X,a,delta)
N_x=100;
for i =1:N_x+1
x(i)=a+delta*(i-1);
cont=0;
  for n=1:length(X)
if X(n)<=x(i)+delta  &&   X(n)>x(i)
    cont=cont+1;
end
  end
P(i)=cont/(length(X));
end
end


function[]=densite_Emp_graphe(a,delta,X)

[P,x]=fonction_Emp_densite(X,a,delta);
figure;
plot(x,P,'ro','MarkerSize',4,'MarkerFaceColor', 'r' );
xlabel 'x'
ylabel 'f_X(x)'
title 'Fonctions de densité Emp. (MC)  et Th. de Poisson, lambda=10'
end

%%%%%%%%%%%%%  densité théorique %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function[f]=densite_Th_Poisson(x,lambda)
fact=1;
if x==0
    f=exp(-lambda);
else
    for k=1:x
    fact=fact*k;
    end
   f=lambda^x.*exp(-lambda)/fact;
end
end
%plot(x,densite_th,'LineWidth',4);
```

— simulés

— théoriques



$$e^{-\lambda}\frac{\lambda^x}{x!}$$

$x = 0, 1, 2, 3$

Travail à faire pour Poisson

- Soient $\lambda = 2, N_{mc} = 10000$
  - Simuler la v.a. de Poisson
  - Tracer sa fonction de repartition: $[a, b] = [0, 10], \Delta = 0.1, N_x = 100$
  - Tracer sa fonction de densité: $[a, b] = [0, 10], \Delta = 0.1, N_x = 100$
- Soient $\lambda = 4, N_{mc} = 10000$
  - Simuler la v.a. de Poisson
  - Tracer sa fonction de repartition: $[a, b] = [0, 15], \Delta = 0.15, N_x = 100$
  - Tracer sa fonction de densité: $[a, b] = [0, 15], \Delta = 1, N_x = 15$
- Soient $\lambda = 50, N_{mc} = 10000$
  - Simuler la v.a. de Poisson
  - Tracer sa fonction de repartition: $[a, b] = [0, 200], \Delta = 2, N_x = 100$
  - Tracer sa fonction de densité: $[a, b] = [0, 200], \Delta = 1, N_x = 200$

Travail à faire pour v.a. Binomial

- Soient $N = 20, p = 0.5, N_{mc} = 10000$
- On définit la v.a. Binomiale X par la somme de N v.a.indépendantes de Bernoulli Y_i:

$$X = \sum_{i=1}^{N} Y_i, \qquad Y_i = \begin{cases} 1 & avec \;\; proba \;\; p \\ 0 & avec \;\; proba \;\; 1-p \end{cases}$$

• p est la probabilité d'un succès et les valeurs de X représente le nombre de succès parmi N essais
• Simuler la v.a. de Binomiale à partir de la définition à l'aide de v.a. de Bernoulli.
• Tracer sa fonction de repartition
• Tracer sa fonction de densité

Simulation 2 de v.a. Binomiale

- La v.a. de loi binomiale $Bin(N, p)$ s'écrit comme une somme de N v. a. de Bernoulli $B(p)$ indépendantes et représente k succès parmi N essais:
  $\mathbb{P}(X = k) = C_k^N p^k (1-p)^{(N-k)}$

- Monter $\mathbb{P}(X = k+1) = \frac{p(N-k)}{(1-p)(k+1)} \cdot \mathbb{P}(X = k)$
  - Algorithme 4 de simulation d'une seule variable aléatoire Binomiale.
  - function[X]=V_A_Binomiale(p, N)
  - ◦ Set $k = 0, \quad proba = (1-p)^N, \quad F = proba;$
  - ◦ Générer U
  - ◦ While $U > F$
  - ◦ Set $proba = \frac{p(N-k)}{(1-p)(k+1)} \cdot proba, \quad F = F + proba, \quad k = k+1$
  - ◦ end
  - ◦ Set $X = k$
  - ◦ end
  - endfunction

Travail à faire pour Binomial

- Soient $N = 50, p = 0.2, N_{mc} = 10000$

  • Simuler la v.a. Binomial par l'algorithme 4

  • Calculer l'espérance et la variance empiriques, les comparer avec les valeurs théoriques
  $\mathbb{E}(X) = Np, \quad \mathbb{V}ar(X) = Np(1-p)$

  • Tracer sa fonction de repartition:
  $[a, b] = [0, 50], \Delta = 0.5, N_x = 100$

  • Tracer sa fonction de densité:
  $[a, b] = [0, 20], \Delta = 1, N_x = 20$

Simulation de v.a. Géométrique

- Soit X v.a. géométrique avec le paramètre p telle que $\mathbb{P}(X = n) = (1 - p)^{n-1} \cdot p$.
  V. a. géométrique représente le nombre d'essais avant d'arrivé d'un premiere succès. La probabilité d'un succès est p.
  - Simuler la v.a. Géométrique par l'algorithme 4 avec $p = 0.2$
  - Tracer sa fonction de densité
  - Simuler la v.a. Géométrique à l'aide de v.a. de Bernoulli.

  Simuler plusieurs fois la v.a. de Bernoulli. Compter le nombre des échecs ($X = 0$) avant d'arrivé d'un premiere succès ($X = 1$). Identifier une valeur de v.a. Géométrique à celle du compteur
  - Tracer la fonction de densité de v.a. Géométrique