Ghazouani Soukaina
Réthoré Sophie
IMR2

Simulations numériques

Exercice 1 : simulation de lois de probabilité

Partie 1

1. GENERATION D'UNE SEQUENCE PSEUDO-ALEATOIRE DE N NOMBRE SELON UNE LOI UNIFORME SUR [0,1]

Nous souhaitons générer une séquence de nombres de façon pseudo-aléatoire selon une distribution uniforme.

Dans ce type de séquence, seuls les premiers nombres sont définis aléatoirement et les suivants suivent un algorithme défini.

Nous allons dans notre cas nous servir d'un algorithme utilisant la formule suivante :

$x_i = a * x_i - 1 + c \ modulo \ m$

Les valeurs a, c et m ne doivent pas être choisies au hasard, les entiers c et m doivent être premiers entre eux.

La valeur a sert de « multiplicateur », la valeur « c » d'incrément et la valeur m est le « module ». La

Valeur initiale X0 va permettre de générer notre suite pseudo-aléatoire.

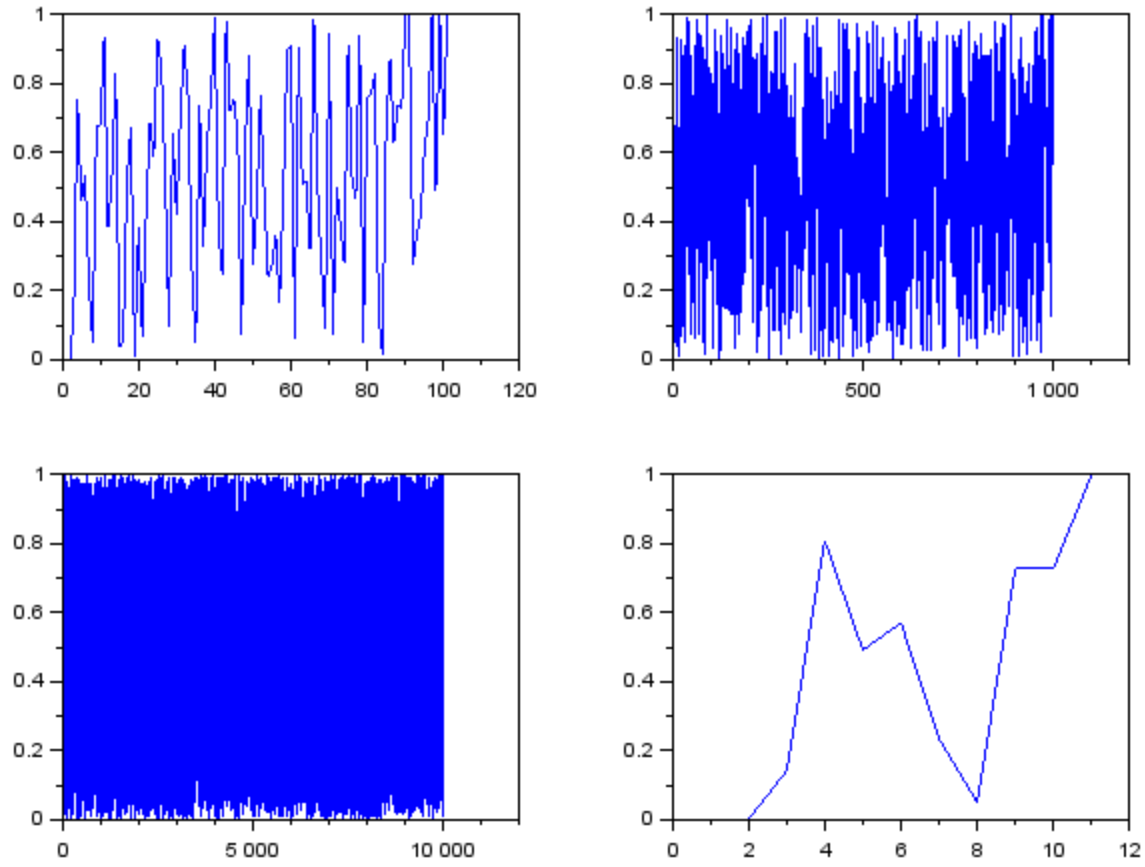
La fonction genere_nb_aleatoire(n,X0) suivante nous permet d'obtenir des séquences de nombres :

Ghazouani Soukaina
Réthoré Sophie
IMR2

```
//1. cette fonction permet de générer N nombres aléatoires de loi uniforme sur [0,1]
function X=genere_nb_aleatoire(n, x0)

   //méthode minimal standard : générateurs a congruence linéaire nonoptimisés
   a= 16807;
   m=(2^31)-1
   c=0;
   //m=10^8;
   X = zeros(n,1);
   X(1,1) = x0
   for k = 1 : n
     X(k+1,1) = modulo(a*X(k,1) + c,m) ;
     //X(k+1,1) = a*X(k,1) + c * modulo(m);
   end
   //on ramène les valeurs générées à des valeurs comprises entre 0 et 1
   for i=1:n+1
      X(i)=X(i)/max(X);
   end

   //affichage du résultat
   //plot(X);
endfunction
```

Pour obtenir une loi uniforme générant des nombres variant entre 0 et 1, nous avons cherché le plus grand des nombre générés et divisé chacune des valeurs obtenues par ce grand nombre.

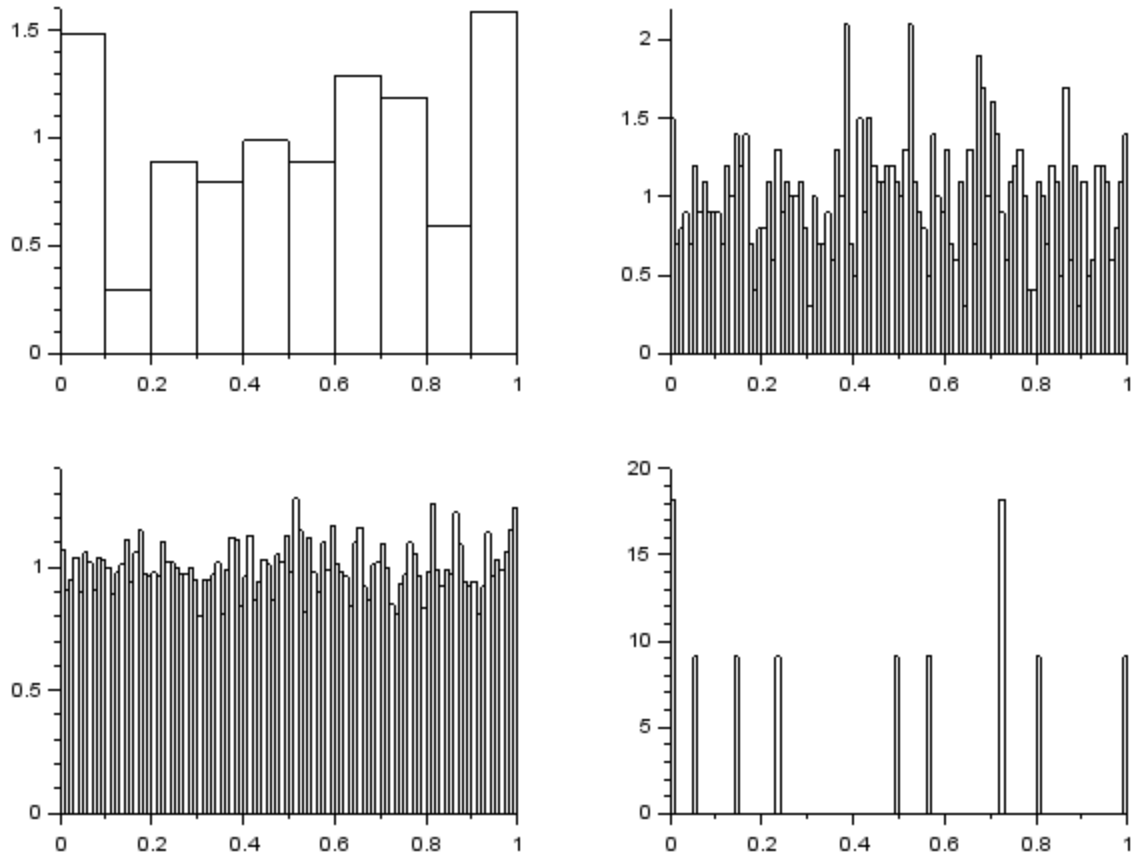
En faisant varier le paramètre N, nous obtenons les résultats suivants, pour respectivement N=100, N=1000, N=10000 et N=10.
Nous pouvons observer le fait qu'avec notre algorithme, plus la valeur N est grande et plus les valeurs générées se stabilisent.



2. HISTOGRAMMES GENERES POUR PLUSIEURS VALEURS DE N

Nous avons créé une fonction principale appelée « main » qui fait appel à la fonction de génération de nombres pseudo-aléatoires en faisant varier la valeur de N.

Une fois un tableau contenant les valeurs précédemment générées crée, nous pouvons passer ces mêmes valeurs à la fonction « histplot ». Celle-ci permet de générer les histogrammes suivants (nous gardons les mêmes valeurs de N que précédemment) :



Nous pouvons constater que plus la valeur de N est élevée et plus les valeurs générées sont réparties uniformément entre 0 et 1.

L'algorithme de génération utilisé est fonctionnel mais pas totalement satisfaisant car il n'est efficace qu'avec de grandes valeurs de N.

3. TEST DU CHI2

Nous souhaitons caractériser les séquences de nombres générées à l'aide du test du chi2.

Pour cela, nous avons défini la fonction suivante :

Ghazouani Soukaina
Réthoré Sophie
IMR2

```
function test_chi2(X)
   n= size(X,1)-1;

   //on récupère un tableau contenant les valeurs générées aléatoirement
   // on parcours ce tableau et on classifie les valeurs selon plusieurs classes.
   for i=1:1:n
      //disp (X(i));
      if X(i)>=0 & X(i) <0.1
         //classe1(i)=X(i);
         classe1=classe1+1;
      elseif X(i)>=0.1 & X(i) <0.2
         //classe2(i)=X(i);
         classe2=classe2+1;
      elseif X(i)>=0.2 & X(i) <0.3
         //classe3(i)=X(i);
         classe3=classe3+1;
      elseif X(i)>=0.3 & X(i) <0.4
         //classe4(i)=X(i);
         classe4=classe4+1;
      elseif X(i)>=0.4 & X(i) <0.5
         //classe5(i)=X(i);
         classe5=classe5+1;
      elseif X(i)>=0.5 & X(i) <0.6
         //classe6(i)=X(i);
         classe6=classe6+1;
      elseif X(i)>=0.6 & X(i) <0.7
         //classe7(i)=X(i);
         classe7=classe7+1;
      elseif X(i)>=0.7 & X(i) <0.8
         //classe8(i)=X(i);
         classe8=classe8+1;
      elseif X(i)>=0.8 & X(i) <0.9
         //classe9(i)=X(i);
         classe9=classe9+1;
      elseif X(i)>=0.9 & X(i) <=1
         //classe10(i)=X(i);
         classe10=classe10+1;
      end
   end

   p=1/10;

   //calcul de D^2
   d2= (((classe1-n*p)^2)/(n*p)) + (((classe2-n*p)^2)/(n*p)) + (((classe3-n*p)^2)/(n*p)) + (((classe4-
n*p)^2)/(n*p)) + (((classe5-n*p)^2)/(n*p)) + (((classe6-n*p)^2)/(n*p)) + (((classe7-n*p)^2)/(n*p)) +
(((classe8-n*p)^2)/(n*p)) + (((classe9-n*p)^2)/(n*p)) + (((classe10-n*p)^2)/(n*p));

   // pour un seuil de 5%
   // selon le tableau le résultat seuil est 16,919

   if d2 > 16.919 then
      disp('le phénomène observé navait que 5% de chance de se produire donc on rejette H0 et la
séquence observée ne suit pas une loi uniforme sur [0,1]');
   else
     disp('la séquence observée suit une loi uniforme selon le test du chi2') ;
   end

endfunction
```

Posons l'hypothèse H0 : « la séquence donnée en paramètres suit la loi Uniforme sur [0,1] ».

On a la probabilité p = 1/n selon la loi uniforme discrète.

Pour simplifier notre test, nous répartissons les valeurs données par le générateur en différentes classes. Nous cherchons à obtenir le nombre de valeurs appartenant à chaque classe, comme dans le tableau suivant par exemple (pour N=10) :

Valeurs	De 0,0 à 0,1	De 0,1 à 0,2	De 0,2 à 0,3	De 0,3 à 0,4	De 0,4 à 0,5	De 0,5 à 0,6	De 0,6 à 0,7	De 0,7 à 0,8	De 0,8 à 0,9	De 0,9 à 1,0
Nombre de valeurs dans chaque « classe »	2	2	1	0	1	2	0	0	1	1

Nous pouvons ensuite calculer D^2 qui est la distance entre la loi observée de X (variable aléatoire des valeurs générées) et la loi théorique de X.

Nous savons que $D^2 = \sum_{k=1}^{10} \frac{(Nk - n\,(p_k))2}{n(p_k)}$

Avec

- Nk : nombre de classes concernées par un cas
- n : nombre de nombres générés
- p : probabilité de réalisation de l'évènement (ici 1/N car c'est une loi uniforme).

Nous pouvons en déduire que nous avons affaire à une loi de $chi2_{k-1}$, soit $chi2_9$.

En définissant un seuil à 5%, nous pouvons observer dans le tableau de la loi du chi2, la valeur correspondante. Celle-ci est de 16,919.

Nous intégrons cette valeur à ne pas dépasser directement dans notre code.

En réalisant le test précédent avec des valeurs de N différentes, nous obtenons les résultats suivants :

N = 100
D2 = 13.4
La séquence observée suit une loi uniforme selon le test du chi2

N = 1000
D2= 4.5
La séquence observée suit une loi uniforme selon le test du chi2

N=10000
D2 = 7.608
La séquence observée suit une loi uniforme selon le test du chi2

N=10
D2 = 8.
La séquence observée suit une loi uniforme selon le test du chi2

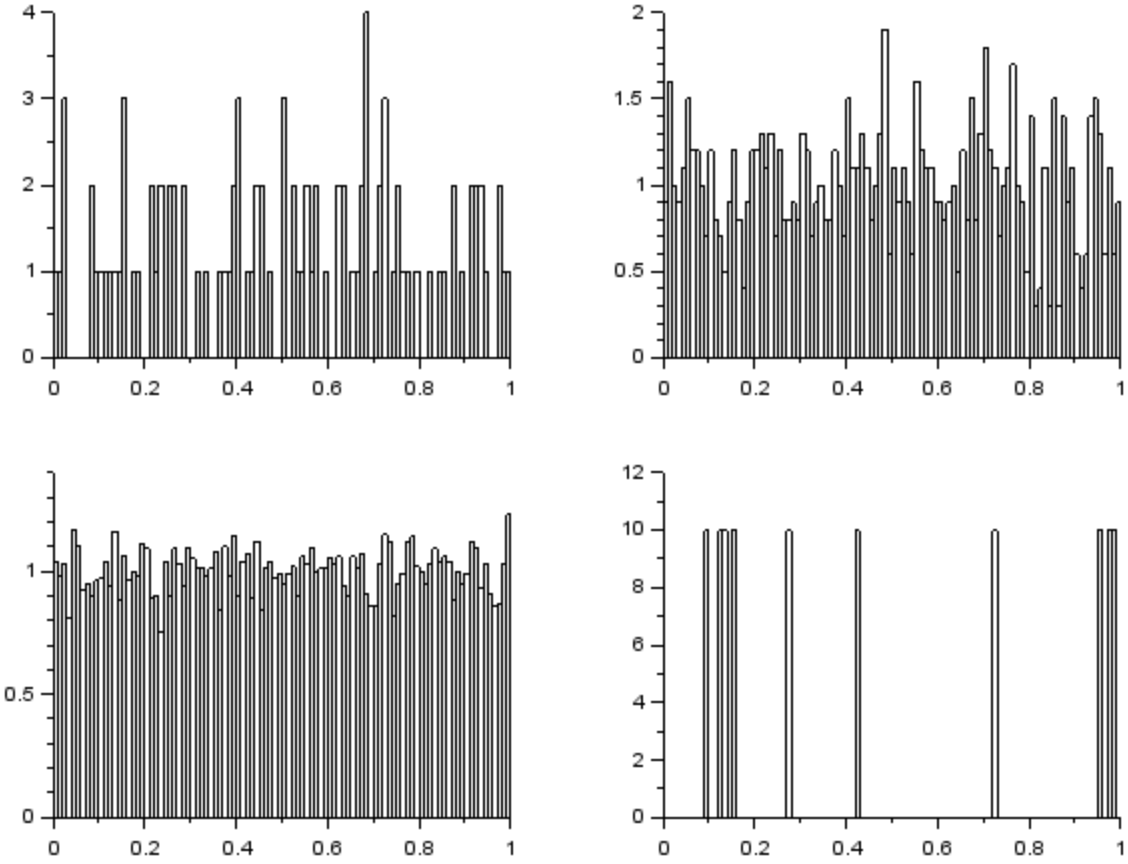
Comme les valeurs de D^2 observées ne dépassent pas 16,919, nous ne faisons pas fasse à un phénomène rare et pouvons accepter l'hypothèse H0 avec un risque de 5%.

Nous pouvons donc en conclure que les valeurs générées pseudo-aléatoirement par notre générateur « minimal standard » suit bien une loi uniforme sur [0,1].

4. ETUDE DE LA FONCTION « RAND » DE SCILAB

Nous allons à présent réaliser la même étude que précédemment mais cette fois-ci avec la fonction « rand » déjà existante sous Scilab.

Nous obtenons les histogrammes suivants pour les mêmes valeurs de N que précédemment (respectivement N=100, N=1000, N=10000, N=10).



Nous obtenons ensuite les résultats suivants au test du chi2, nous pouvons donc en conclure que la fonction « rand (N,1,"uniform") » de scilab suit bien une loi uniforme sur [0,1].
Nous observons aussi le fait qu'en générant plusieurs fois d'affilées des valeurs avec la fonction « rand », celles-ci sont différentes (car aléatoires) mais satisfont quand même les critères imposés au test du chi2 (5%).

N = 100
D2 = 4.9393939
La séquence observée suit une loi uniforme selon le test du chi2

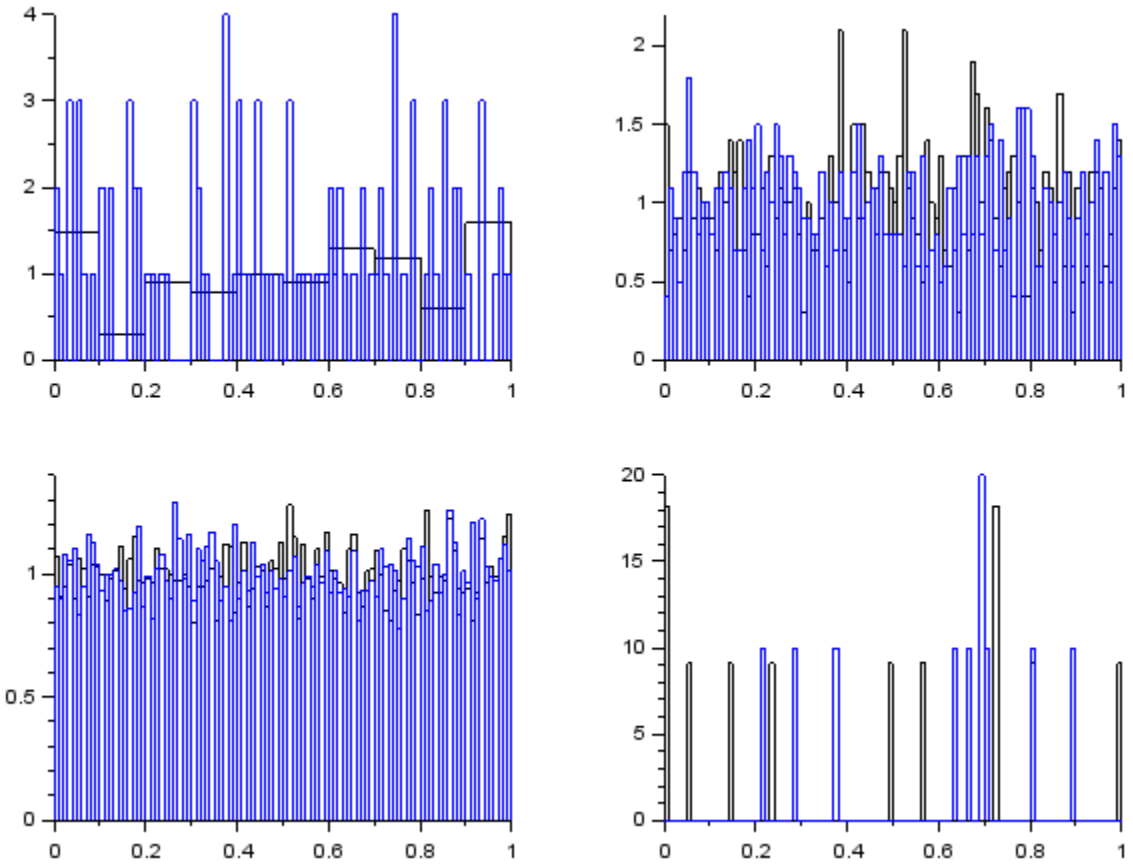
N = 1000
D2 = 10.53954
La séquence observée suit une loi uniforme selon le test du chi2

N = 10000
D2 = 2.2851285
La séquence observée suit une loi uniforme selon le test du chi2

N = 10
D2 = 14.333333
La séquence observée suit une loi uniforme selon le test du chi2

5. COMPARAISON DES 2 METHODES DE GENERATION DE NOMBRE PSEUDO-ALEATOIRES

Les histogrammes suivants permettent de comparer les nombres générés par la fonction « rand » de Scilab en bleu et par notre fonction en noir.



Nous pouvons voir que pour des plus petites valeurs de N, la fonction « rand » de Scilab nous donne un résultat plus satisfaisant. Les valeurs sont mieux réparties entre 0 et 1.

Notre générateur congruentiel répond globalement bien à ce que nous attendons c'est-à-dire qu'il génére des nombre compris entre 0 et 1 et de façon quasiment aléatoire. Cependant nous avons pu observer le fait que les résultats des D^2 observés sont ne varient que très peu.

Ghazouani Soukaina
Réthoré Sophie
IMR2

Partie 2

6. GENERATION DE NOMBRES PSEUDO-ALEATOIRES SUIVANT UNE LOI CONTINUE

A partir des nombres générés en partie 1, nous devions pouvoir construire une séquence de nombres aléatoires de loi discrète ou continue.

Pour cela, plusieurs méthodes sont disponibles :

- La méthode du rejet
- La génération par la fonction de répartition
- La génération par changement de variable

Nous avons décidé comme dans l'exemple du cours de simuler une génération de nombres selon une loi continue exponentielle et ceci avec la méthode de génération par fonction de répartition.

Cependant la fonction « loi_deux(X, lambda) » prenant en paramètres la séquence de nombres précédemment générée et un paramètre lambda (lambda variant entre 0.5 et 1.5) nous renvoie des erreurs liées à une mauvaise définition de la fonction « log2 » que nous n'avons pas réussi à résoudre.

```
function loi_deux(X, lambda)
  n=size(X,1)-1;
  x=zeros(n);

  //disp('n=',n);
  for i=1:n
    //j=1-X(i);
    //disp('X(i)=',X(i));
    if (X(i)<>0) then
      x(i) = -(log2(1-X(i))/lambda);
    end
  end
  disp ('x=',x);
  histplot([0:0.1:1] ,X) ;
endfunction
```

Pour la fonction exponentielle, nous avons la fonction de répartition suivante F(x) = p(X<=x) = $1-e^{-\lambda x}$ avec x compris entre 0 et 1.

Et F(x) = 0 pour x négatif.

Selon l'exemple du cours, nous obtenons l'équivalence

y= $1-e^{-\lambda x}$ avec x appartient à R+ ⟷ x = - $\frac{\ln(1-y)}{\lambda}$ avec y appartient [0, 1[

La fonction à créer aurait donc du être celle-ci :

x(i) = -(log2(1-**X**(i))/**lambda**);

Ghazouani Soukaina
Réthoré Sophie
IMR2

Exercice 2 : loi des grands nombres

1. L'esperance et l'ecart type de xi suivant une loi de Bernoulli

On choisit une loi de Bernoulli pour les variables Xi avec 1<i<N

La moyenne empirique est donnée par :

$$\bar{X}n = \frac{1}{N}\sum_{i=1}^{N} Xi$$

On note p le paramètre de la variable aléatoire Xi.

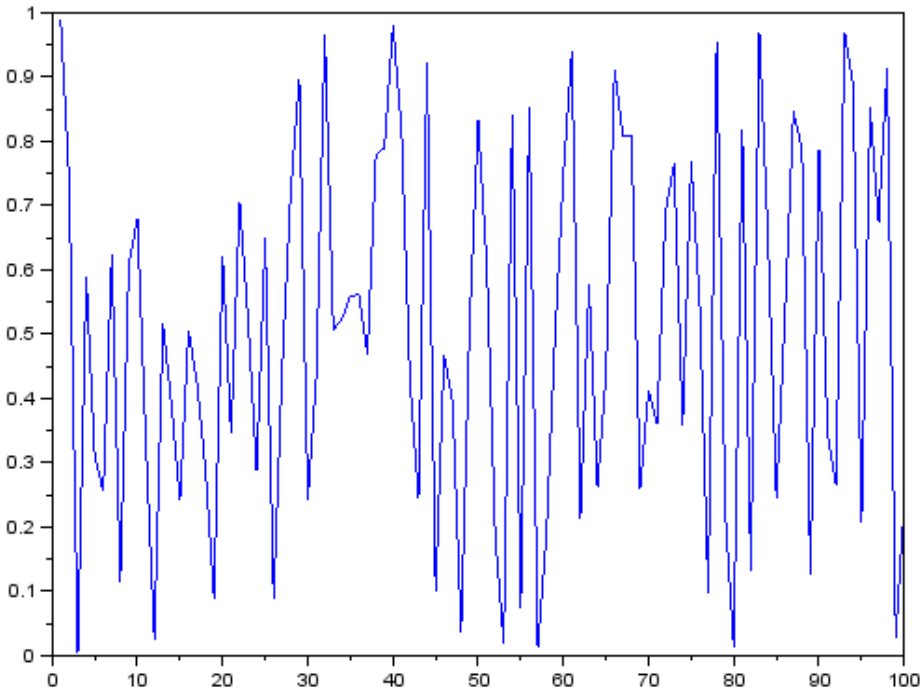
Alors :

L'espérance de xi est : E[Xi]=p

l'ecart-type = sqrt(pq) avec q= 1-p.

2. Representation graphique d'une realisation de Xn en fonction de N

Ci-dessous le graphe obtenu pour 1 réalisation de Xn avec N=100



Sur l'axe des abscisses, on a les 100 expériences réalisées pour Xn. Pour chacun de ces points, on a une valeur comprise entre 0 et 1 qui correspond à la probabilité de succès.

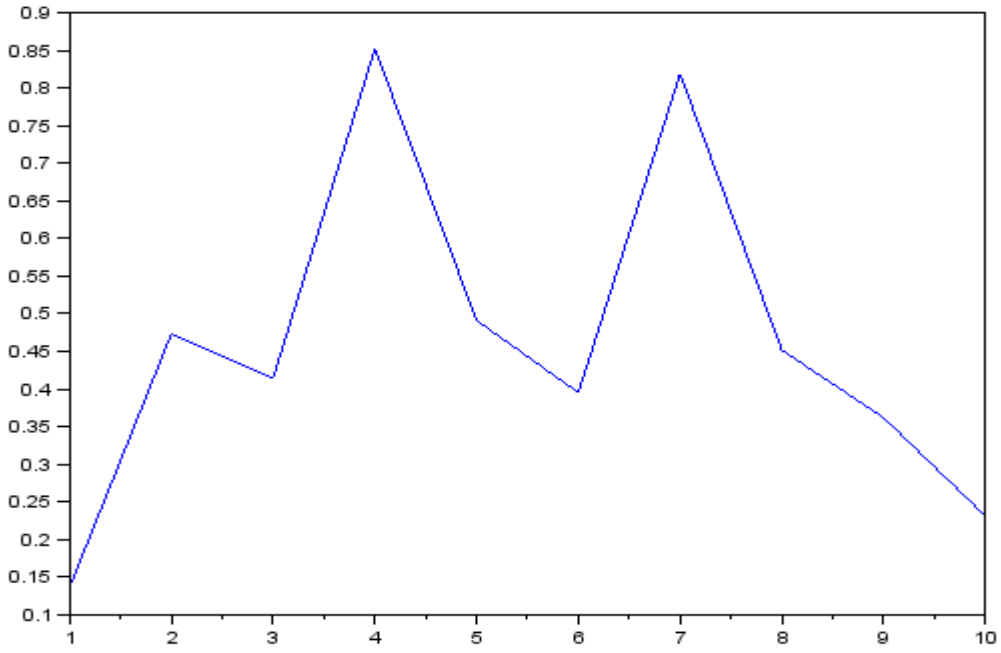
3. CALCUL NUMERIQUE DE LA MOYENNE POUR PLUSIEURS VALEURS DE N

Nous cherchons ici à trouver la moyenne numérique pour plusieurs valeurs de N :

- Pour N=10 :

```
-->repres(10)

    0.3204056

 calcul de la moyenne :

    0.0238733

 calcul de l écart type:
 ans  =


          column 1 to 7

    0.2153754    0.2838643    0.2595268    0.6314125    0.2984818    0.3809566    0.435286

          column  8 to 10

    0.1094232    0.1554041    0.4343251
```

Représentation graphique des 10 expériences réalisées :

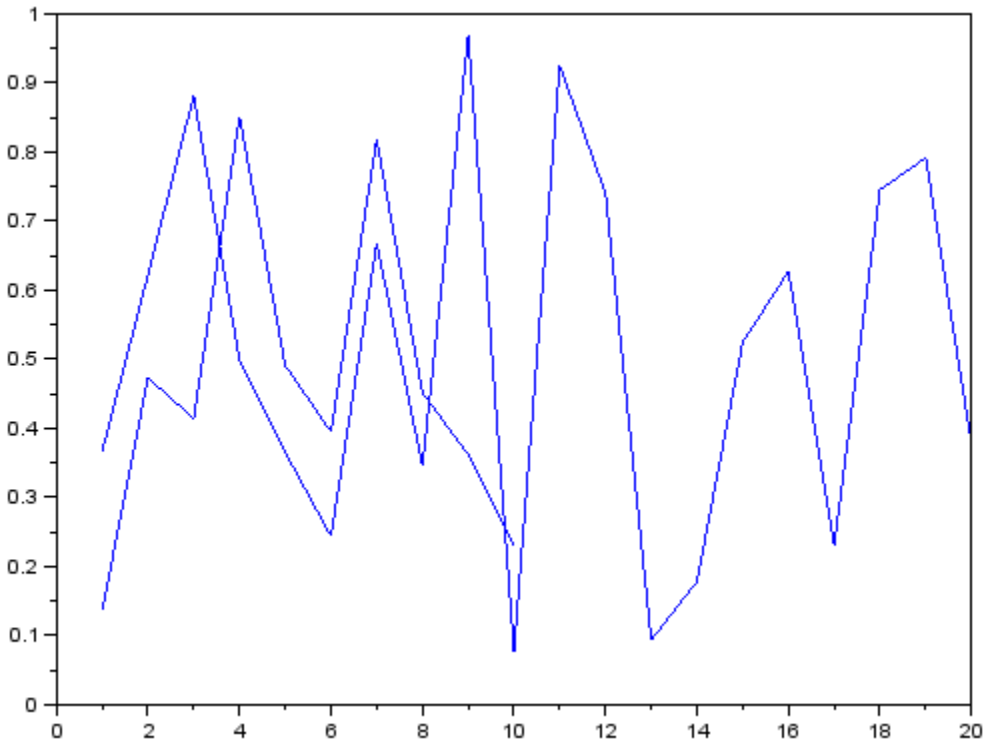


Nous obtenons une moyenne empirique d'une valeur de 0,039

- Pour N= 20

```
-->repres(20)

    0.5697298

 calcul de la moyenne :

    0.0571874

 calcul de l écart type:
 ans  =


         column 1 to 7

    0.4073455    0.7088422    0.8718486    0.2669421    0.3669498    0.6290400    0.3493621

         column  8 to 14

    0.9183151    0.1504278    0.3923354    0.6889904    0.8186653    0.8985910    0.7568131

         column 15 to 20

    0.7173781    0.6427609    0.3152925    0.5000510    0.7311326    0.2635126
```

Représentation graphique des 20 expériences réalisées :

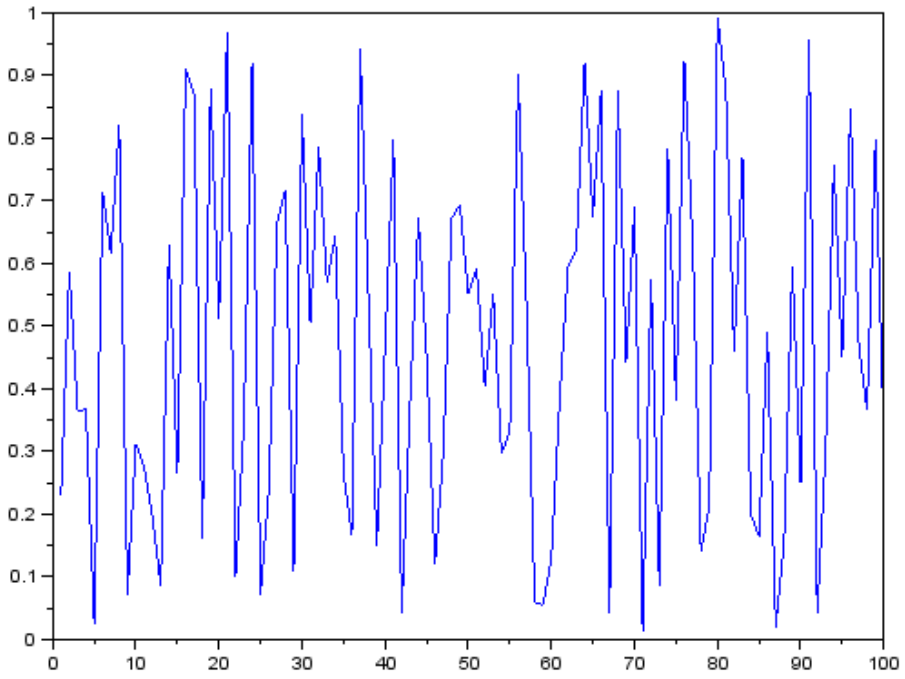


Lorsque l'on réalise 20 expériences, nous obtenons une moyenne empirique de 0,057

- Pour N= 100

```
-->repres(100)

    0.4762431

 calcul de la moyenne :

    0.0843149

 calcul de l écart type:
 ans  =


          column 1 to 7

    0.2299218    0.5881678    0.3657124    0.3669910    0.0233949    0.7157733    0.6166617

          column  8 to 14

    0.8231646    0.0721666    0.3128446    0.2761314    0.1957585    0.0865941    0.6317711

          column 15 to 21

    0.2655498    0.9106507    0.8707769    0.1619037    0.8795673    0.5119673    0.9708724

          column 22 to 28

    0.1006239    0.3435521    0.9212376    0.0714581    0.2365225    0.6660066    0.7180338

          column 29 to 35

    0.1086907    0.8384379    0.5046565    0.7858450    0.5690123    0.6441036    0.2649619
```

Représentation graphique des 100 expériences réalisées :



Lorsque l'on réalise 100 expériences, nous obtenons une moyenne empirique de 0,084

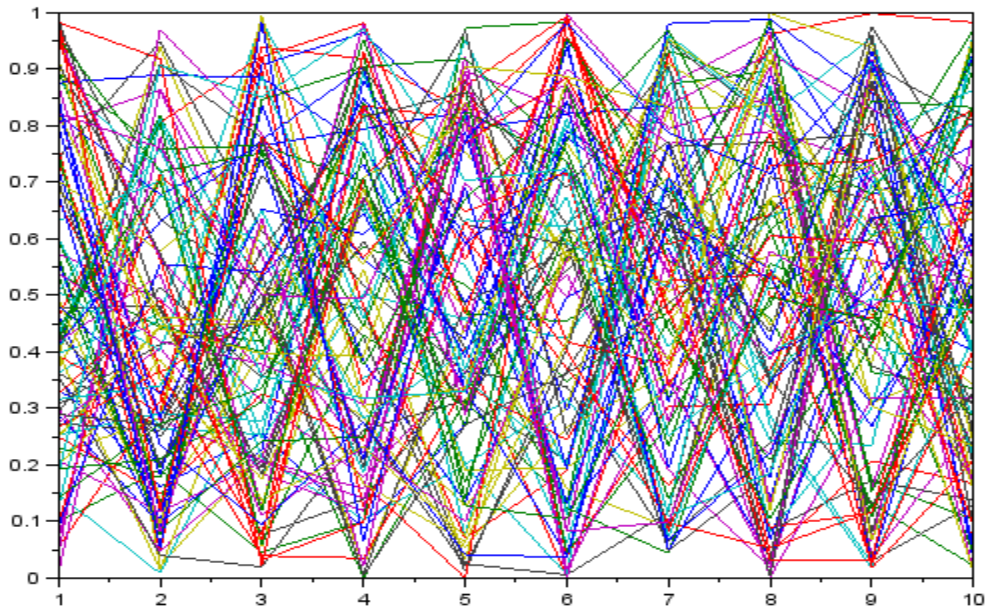
On constate alors que lorsque le nombre N est grand, la valeur moyenne calculée par le programme tend vers la valeur théorique.

4. REPRESENTATION GRAPHIQUE DE K REALISATIONS DE XN EN FONCTION DE N

L'idée de cette partie est de représenter graphiquement plusieurs réalisations en fonction de N

Voici le résultat trouvé pour 10 réalisations et pour N=100 :

```
-->repres4(100)

    0.5000671

calcul de la moyenne :

    0.0842739

calcul de l écart type:
ans  =


         column 1 to 7

    0.0441903    0.2293008    0.2706805    0.3623268    0.9926624    0.2981085    0.6993418
    0.2841374    0.3002163    0.4621830    0.6828014    0.9561965    0.3794726    0.4068444
    0.7478221    0.3253413    0.1431145    0.7308872    0.7839966    0.5614697    0.2745692
    0.9814446    0.2611944    0.1621217    0.5410475    0.8623644    0.9381308    0.4687920
    0.4803915    0.8767535    0.1060825    0.1090942    0.9269526    0.0987267    0.9582316
    0.3995500    0.9044342    0.1463016    0.3710793    0.9300042    0.5494711    0.7723589
    0.1891019    0.0624963    0.8253459    0.4966914    0.0956909    0.5147551    0.0740479
    0.9859605    0.8515070    0.4135403    0.3317022    0.8594969    0.671969     0.2728199
    0.6594892    0.0519573    0.8581838    0.4347770    0.9646158    0.7878396    0.6161082
    0.9631401    0.4462789    0.7861885    0.2431572    0.6445985    0.3123078    0.6236014
```



Ghazouani Soukaina
Réthoré Sophie
IMR2

Exercice 3 : Théorème central limite

On considère N v a Xi indépendantes toutes de meme loi. Soit la va :

$$\mathrm{Yn} = \frac{\sum_{i=1}^{N} Xi - nE(Xi)}{\sqrt{N} * \sigma}$$

L'objectif de cet exercice est de tracer des histogrammes de Yn selon une loi et en faisant varier N et les paramètres de la loi.

Partie 1

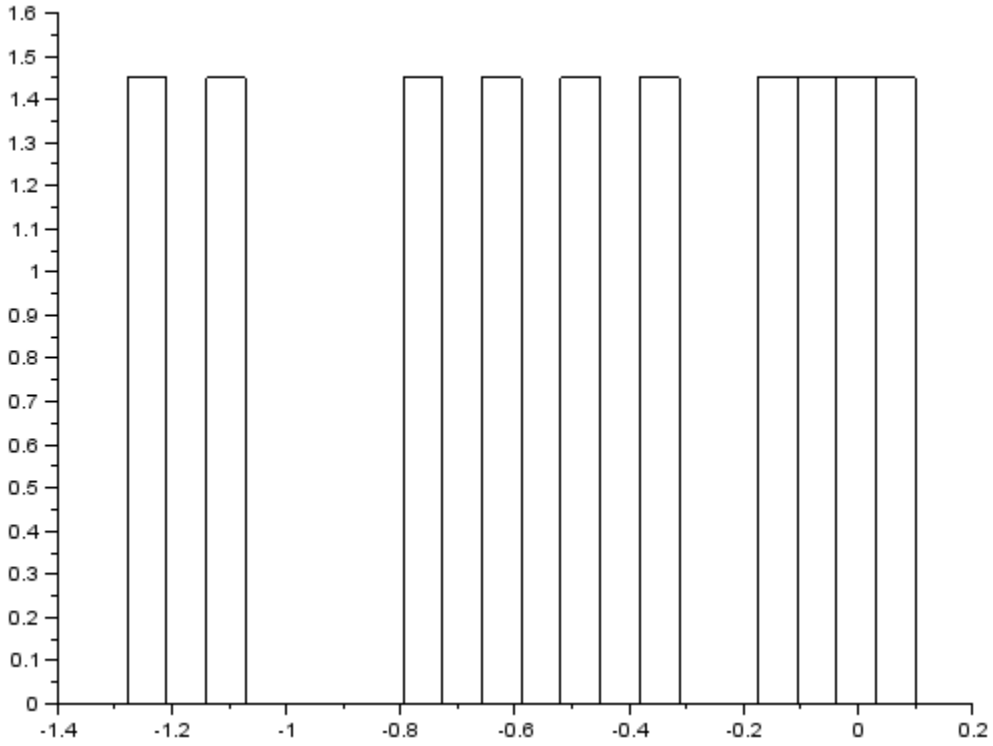
1. Histogramme de Yn suivant la loi uniforme

Pour la réalisation de cette partie nous avons créé la fonction ci-dessous, permettant de représenter l'histogramme de Yn suivant une loi uniforme :

```
function Yn=hist_loi_uniforme()
   // choisir la taille de N
   n=input('Taille de l''echantillon, N=');
   //choisir le nombre de réalisations
   k=input('Nombre de réalisations, k=');

   x=rand(k,n);
   // la moyenne d'une loi uniforme
   E=.5;
   // l'ecart type est donné par :
   sigma=1/sqrt(12);

   // initialisation de la matrice Yn
   Yn=zeros(1,k);

   for i=1:k
      Yn(i)=(sum(x(i,:))-n*E)/(sqrt(n)*sigma);
   end
   clf();
   histplot(50,Yn);
endfunction
```

- Pour N= 100 et K= 10

```
-->hist_loi_uniforme
Taille de l'echantillon, N=100
Nombre de réalisations, k=10
 ans  =


         column 1 to 7

  - 0.5138755    0.1004307  - 0.5890222  - 0.3752036  - 1.2780441  - 1.1299258  - 0.1690326

         column  8 to 10

  - 0.7753297  - 0.0748811  - 0.0306560
```



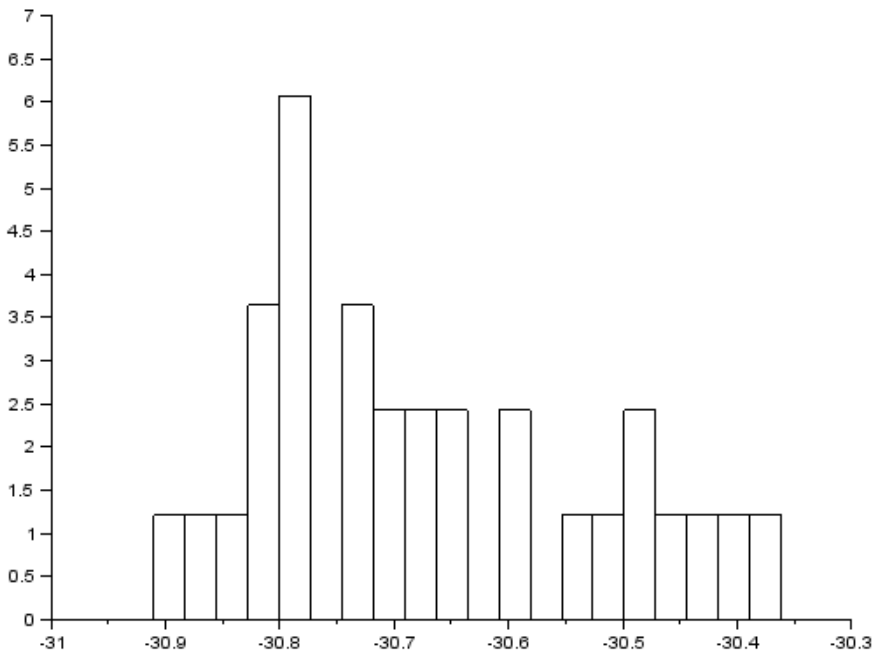
2. HISTOGRAMME DE YN SUIVANT UNE LOI CONTINUE

La loi exponentielle a été choisie, pour la simulation d'une loi continue

```
function Yn=loiexp2()

    // choisir la taille de N
  n=input('Taille de l''echantillon, n=');
 lambda=input('choisir lambda, lambda=');
  //choisir le nombre de réalisations
  k=input('Nombre de réalisations, k=');

  x = grand(n,k,"exp",1/lambda);
  //u=rand(k,n);
 // x = (lambda)*exp(-(lambda)*u);
```

```
   // la moyenne de la loi exp
      E=1/lambda;

   // l'ecart type est donné par :

      sigma=1/sqrt(lambda*lambda);

   // initialisation de la matrice Yn
   Yn=zeros(1,k);

   for i=1:k
       Yn(i)=(sum(x(i,:))-n*E)/(sqrt(n)*sigma)
   end
   clf();
  // classes = linspace(0,20,1);
   histplot(40,Yn)

endfunction
```

- Pour N= 1000, K= 30, lambda=3 nous obtenons le résultat ci-dessous :

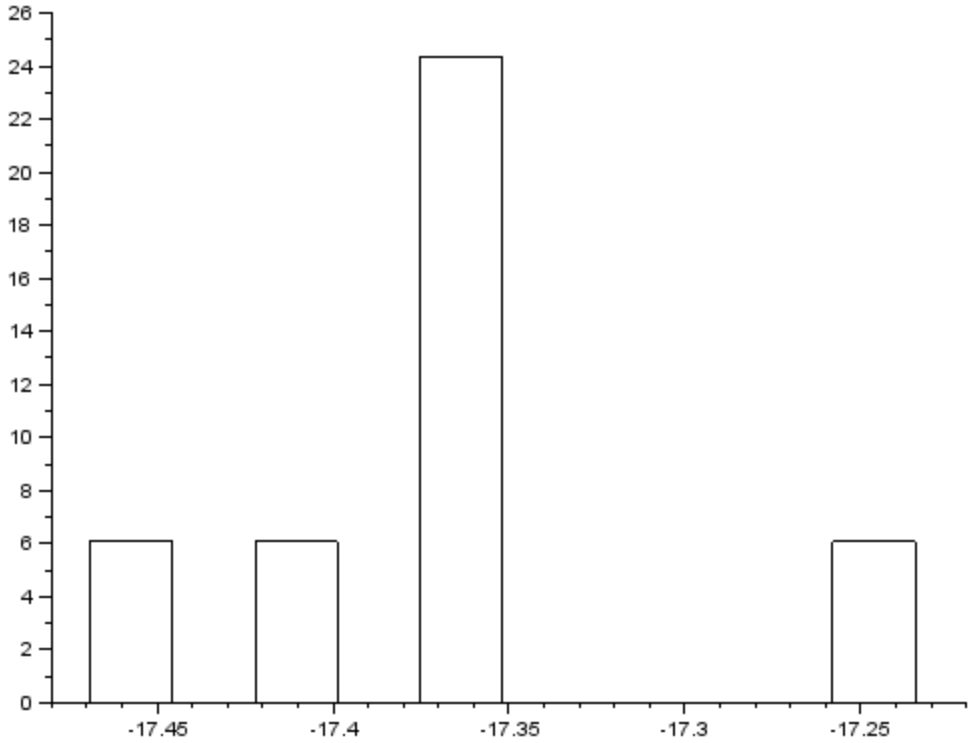


Partie 2

Dans cette partie, on choisit la loi binomial de paramètres p=0.03 et n=100. Puis p=0.5 et n=100.

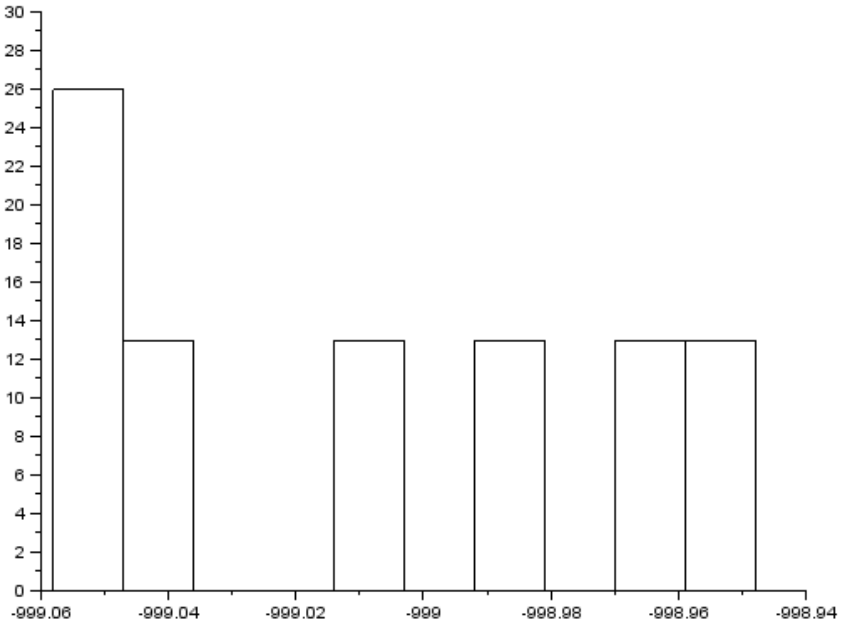
- Pour n=100 p=0.03

avec k=7, voici le résultat renvoyé par le programme



- Pour n=1000 p=0.5

avec k=7, voici le résultat renvoyé par le programme



Les résultats trouvés ne sont pas tout à fait correctes, cela est certainement lié à une mauvaise définition de la fonction hist_loi_binomiale().

Nous devions constater que la convergence est plus rapide lorsque p est voisin de 0,5.

Lorsque l'on varie les paramètres n et p, nous devions aussi remarquer que la loi de probabilité d'une variable binomiale tend vers une loi de Poisson lorsque n tend vers l'infini. Et donc dire que la loi binomiale B(n,p) converge en loi vers une loi de Poisson de paramètres np.

Conclusion

Ainsi, ce Tp nous aura permis de mettre en pratique les notions vues en cours. Au travers de l'étude d'une simulation de lois de probabilité dans un premier temps, puis par l'étude de loi des grands nombres et le théorème central limite.

Ghazouani Soukaina
Réthoré Sophie
IMR2

Sources:

http://michael.moreno.free.fr/Documents/Moreno_Tech_Sim.pdf

http://carobm.free.fr/Simulation.pdf

https://www.unilim.fr/pages_perso/jean.debord/math/random/random.htm#I

http://ces.univ-paris1.fr/membre/Jolivaldt/Docs/GENERATION.pdf