Fiche TP 06 :
Simuler l'aléatoire

Licence 1 MASS semestre 2, 2007/08

Exercice 1 : Opérations de base

a - Tester les lignes suivantes :

```
s := readlib(randomize)();
rand();
rand();
randomize(s);
rand();
rand();
```

Quel est le role de "randomize()" ? Expliquer le résultat des lignes précèdentes.

b - Affecter dans une variable *maxValue* la valeur maximale que peut retourner la fonction rand.

c - Ecrire un algorithme **uniformAB** qui retourne un nombre réel pseudo-alétoire entre $[a, b[$.

d - Ecrire un algorithme **random** qui retourne un nombre entier pseudo-alétoire entre $[0, M[$.

e - Ecrire un algorithme qui simule le tirage d'une pièce équilibrée.

f - Ecrire un algorithme qui simule le tirage d'un dé à 6 faces.

Exercice 3 : Test

a - Modifier le programme ci-dessous pour qu'il renvoie le graphique de n points dont les coordonées sont dans $[a, a[$.

```
points := proc(n)
# n : entier, nombres de points
# sortie : plot des points

  local P, i, graph;

  P := array(0..n-1):
  for i from 0 to n-1 do
    P[i] := [ uniformAB(-10,10), uniformAB(-10,10) ]:
  od:

  graph := plot(P, x=-10..10, y=-10..10, style=point, symbol=circle):

  RETURN(graph);
end;

g := points(1000):
```

```
with(plots);
display(g);
```

b - Ecrire un programme qui calcule la distance entre deux points dont les coordonées sont données en paramètres.

c - Ecrire un algorithme qui compte le nombre de points (dont les coordonnées sont aléatoirement choisies dans $[a, a[$) qui sont situés dans un cercle C de centre $(0, 0)$ et de rayon r.

d - Tester votre programme sur 1000 ou 10000 points, est-ce le nombre de points attendu dans le cercle et de l'ordre de grandeur de celui que vous obtenez par la simulation ?

Exercice 4 : Suites stochatiques

a -
```
suite := proc(u0, a, b, n)
# u0 : reel, premier terme de la suite
# a,b : reel, raison de la suite
# n : nombre de termes
# sortie : plot des termes

  local P, i, u, graph;

  u := u0;
  P := array(0..n-1):
  for i from 0 to n-1 do
    P[i] := [ i, u ]:
    u := a * u + b:
  od:

  graph := plot(P, x=0..n, y=u0..u, style=line, symbol=circle):

  RETURN(graph);
end;
```

Que produit l'exécution du programme ci-dessus ? Tester-le en faisant varier ses paramètres, en particulier vous testerez pour les valeurs de $u_0 = 10$, $a = 1.01$, $b = 0$ et $n = 100$.

b - Ecrire un programme *suiteAddStoch* en modifiant le programme ci-dessus pour qu'il représente les points de la suite stochatique de terme général : $u_{n+1} = au_n + b + r$ où r est un nombre réel issu de la réalisation d'une variable aléatoire de loi uniforme $U(-\epsilon_1, \epsilon_1)$ (ce qui signifie que r est compris entre $-\epsilon_1$ et ϵ_1).

c - Comparer graphiquement les deux suites précèdentes. Vous pourrez utiliser par exemple $\epsilon_1 = 1$.

d - Ecrire un programme *suiteMultStoch* en modifiant le programme du (a) pour qu'il représente les points de la suite stochatique de terme général : $u_{n+1} = (a + r)u_n + b$ où r est un nombre réel issu de la réalisation d'une variable aléatoire de loi uniforme $U(-\epsilon_2, \epsilon_2)$ (ce qui signifie que r est compris entre $-\epsilon_2$ et ϵ_2).

e - Comparer graphiquement les trois suites obtenues. Vous pourrez utiliser par exemple $\epsilon_2 = 0.05$.

Exercice 2 : Tirage sans remise

Une urne contient 5 boules noires et n boules rouges.

a - Ecrire un programme qui affiche le résultat du tirage de p boules de l'urne.

b - Ecrire un algorithme qui retourne, lors d'une simulation de tirages successifs de boules sans remise, le nombre de tirages nécessaire pour l'apparition de toutes les boules noires.