Série de Travaux Dirigés MODSIM
Simulation

Dans les exercices ci-dessous, « programmer » signifie :

- Soit écrire le programme si vous utilisez un langage universel ,
- Soit trouver la commande du langage spécifique que vous utilisez ,

Les exercices cochés d'un astérix sont facultatifs (en en mode TP libre).

Exercice 1*: *Méthodes des carrés moyens.* C'est le procédé algorithmique le plus ancien et dû à *Von Neumann*.

(i) choisir le terme initial de la suite U_0 .

(ii) calculer $a=U_0^2$.

(iii) Poser U_1 =suite des chiffres situés au milieu de la partie décimale de a.

(iv) Poser $a=U_1^2$; retour en (iii)

Solution. $U_0 = 0.2481$; $a=U_0^2=0.06155361 \Rightarrow U_1 = 0.1553;$

$a=U_1^2=0.02411809 \Rightarrow U_2 = 0.4118;$ etc...

Validation : On a, $\overline{U}=0.570055$ qui est bien voisin de $1/2$. Test khi2.

Exercice 2 *: (i) Soit un générateur congruentiel avec $\lambda=13$, $m=2^6$. Déterminer la période maximale pour $X_0=1$ puis 3. Générer une période de chacune des suites pour $X_0=1,2,3,4$ et comparer.

(ii) Trouver la période maximale pour le générateur congruentiel avec $\lambda=7^5$, $m=2^{31}-1, c=0$. Générer 3 nombres aléatoires lorsque la racine est $X_0=123457$.

Exercice 3*: Soit X une variable aléatoire de loi exponentielle de paramètre $\lambda=2$.

(i) Générer une suite de $n=5$ observations artificielles de cette loi (en utilisant la suite donnée dans la table 6). Evaluer la moyenne arithmétique de ces 5 observations. Comparer avec l'espérance mathématique de X . Qu'en déduisez-vous ?

(ii) (TP N°2) Rédiger un programme (en C ? JAVA ?,) permettant de recommencer l'expérience pour $n=10,20,30,50$. Tracer sur un même graphique l'histogramme expérimental et la courbe théorique.

Exercice 4 : Ecrire le programme générateur de la loi géométrique de trois manières :

(i) Utiliser le programme usuel de la méthode d'inversion. Donner une interprétation.

(ii) Résoudre explicitement l'équation en X, et remarquer qu'on peut utiliser le générateur de la loi exponentielle.

(iii) Utiliser un algorithme similaire à celui de la loi binomiale.

Comparer les performances des trois algorithmes.

Sol : Pour la loi partant de 1 **:** $p_i = pq^i, i \geq 1$ et $F(j-1) == P(X \leq j-1) = 1-q^{j-1}$.

(i) L'algorithme direct d'inversion donnerait

Générer U

$X = j$ si $1-q^{j-1} \leq U < 1-q^j$.

(ii) (Cette inégalité peut être écrite $q^j < 1-U \leq q^{j-1}$. On a

$X = \min\{j : q^j < 1-U\}$. Puisque Log est monotone croissante,

$X = \min\{j : jLog(q) < Log(1-U)\}$ ou

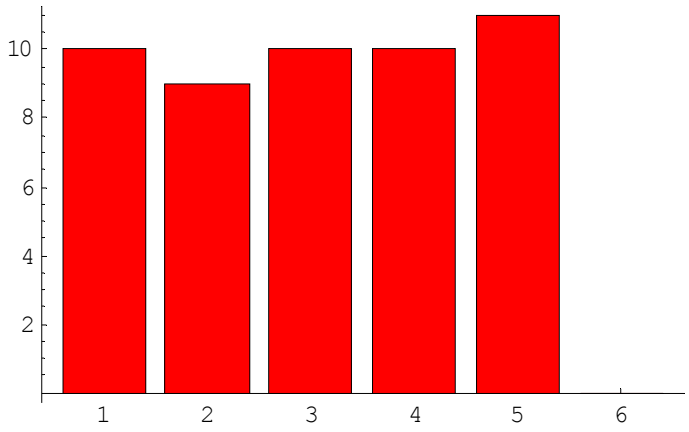
$X = \min\{j : j > Log(1-U)/Log(q)\}$. D'où $X = Int[(Log(1-U))/Log(q)]+1$. Par conséquent $X = Int[Log(U)/Log(q)]+1$ suit également une loi géométrique de paramètre p partant de 1.

Pour la loi partant de 0, $X = Int[Log(U)/Log(q)]$

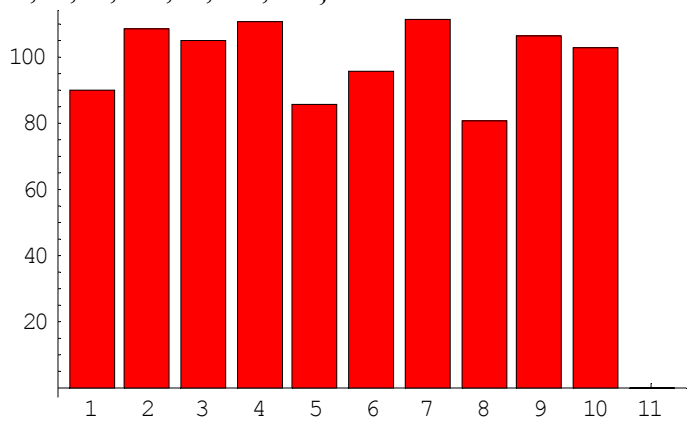
Exercice 5: Ecrire un programme permettant de générer 50 nombres aléatoires, les représenter dans le plan, puis tester leur uniformité à l'aide du test 3 d'uniformité. Confirmer ou infirmer la visualisation graphique à l'aide d'un test de khi-deux.

Solution : Ce petit programme en Mathematica reprend l'exemple 2 . On applique le test 3 en partageant l'intervalle [0,1] en 5 intervalles. L'histogramme représente la distribution des fréquences des intervalles.

```
b=Table[Random[],{50}]
ListPlot[b]
C=RangeCount[b,{0.2,0.4,0.6,0.8,1}]
BarChart[c]
c={10,9,10,10,11}
```



Ce second exemple montre la distribution de 1000 nombres aléatoires sur 10 intervalles.
c={90,109,105,111,86,96,112,81,107,103}



Reprendre l'expérience à l'aide d'un programme de votre choix.

Exercice 6: Générer une variable uniforme discrète sans passer par la technique de tri.

Sol: Puisque $P(X=j)=\frac{1}{m},\ j=1,2,...,m$, l'algorithme serait $X=j\ \ si\ \ \frac{j-1}{m}\le U<\frac{j}{m}$, ou bien $X=j\ \ si\ \ j-1\le mU<j$ ou encore $X=\lfloor mU\rfloor+1$, ou $\lfloor x\rfloor$ est la partie entière de x (le plus grand entier $\le x$.

Exercice 7. Cet exercice reprend l'exemple du début de ce chapitre pour illustrer la simulation par la méthode de Monte Carlo du calcul d'une intégrale $I=\int_0^1 0.5e^{0.5}dx$

(i) Reprendre les expérimentations de calcul de précision et de comparaison.
(ii) Adapter le programme au calcul d'autres intégrales.

Sol : Le petit programme suivant calcule une valeur approchée de $I=\int_0^1 0.5e^{0.5}dx$ par la méthode de Monte Carlo avec une séquence de $N=100$ nombres aléatoires générés selon une *méthode congruentielle*.

```
J=NIntegrate[0.50Exp[0.5x],{x,0,1}
A=Table[Random[],{100}] ;
C=Table[0.5Exp[0.5a[[i]],{i,100}];
U=N[Sum[c[[i]],{i,1,100}]/100]
F=Abs[J-U]
```

Pour comparer, la première ligne fournit la valeur approchée par une méthode numérique de quadrature classique (de type Trapèze) : $J=0.648721$. Les résultats sont donnés dans la table. Le tracé de la courbe est réalisé par le programme suivant

```
<<Graphics`FilledPlot`
FilledPlot[0.5Exp[0.5x],{x,0,1],PlotRange→{0,1},Frame→True]
```

Exercice 8: Simuler manuellement une observation de la loi d'Erlang de paramètre k=3, λ=0.1. Ecrire un programme pour la simulation de cette loi . Quelle méthode préconiser-vous ?

Sol : $U_1 = 0.09656 \Rightarrow Y_1 = -10Log(0.08656) = 23.38$;

$Y_2 = -10Log(0.96657) = 0.34$; $Y_3 = -10Log(0.64842) = 4.33$. D'où

$X_1 = Y_1 + Y_2 + Y_3 = 23.38 + 0.34 + 4.33 = 28.05$

Exercice 9: Indiquer deux méthodes de simulation de la loi binomiale (n petit et n grand). Application numérique n=2, p=1/4.

Sol :(i) n petit, méthode directe : Si $U \in (0, p_0) \Rightarrow X = 0$, $U \in (p_0, p_0 + p_1) \Rightarrow X = 1,$ $U \in (p_0 + p_1, 1) \Rightarrow X = 2$, où $p_1 = C_2^1 \times 0.25 \times 0.75 = 0.375,$ $p_2 = C_2^2 (0.75)^2 = 0.56250$. Ainsi avec les cinq nombres aléatoires $U_0 = 0.12224; U_1 = 0.45637; U_2 = 0.66678, U_3 = 0.88793, U_4 = 0.32416$, on obtient la suite de variables binomiales : 1,2,2,2,1 . (ii) pour n grand utiliser l'un des algorithmes d'approximation normale.

Exercice 10 : Simuler la densité $f(x) = 20x(1-x)^3,\ 0 < x < 1$ à l'aide de la méthode du rejet avec une densité auxiliaire uniforme $g(x) = 1,\ 0 < x < 1$. Dans ce cas, $h(x) = \frac{f(x)}{g(x)} = 20x(1-x)^3$. La dérivée s'annule pour x=1 ou x=1/4, le maximum étant obtenu pour cette dernière valeur. On a donc $h(x) \le 20 \times \frac{1}{4} \times \left(\frac{1}{'4}\right)^3 = \frac{135}{64} = B$.

L'algorithme de génération est donc le suivant :

1. Générer U_1 *et* $U_2 \in U[0,1]$ (U_1 est donc de densité g) ;
2. Si $U_2 \le \frac{256}{27} U_1 (1 - U_1)^3$, $X = U_1$

   Sinon aller en 1.

Le nombre moyen d'itérations est $135/64 \approx 2.109375$.

Simulation de vecteurs.

Exercice 11: Générer des observations du vecteur $W = (X, Y)$ de densité $f(x, y) = \lambda x e^{-(\lambda + y)x}$ par la méthode du conditionnement puis par la méthode du rejet.

Sol: $f_X(x) = \int_0^\infty \lambda x e^{-(\lambda+y)x} dy = \lambda e^{-\lambda x} \in Exp(\lambda)$; $f_{Y/X}(y/x) = \frac{f(x,y)}{f_X(x)} = x e^{-xy} \in Exp(x)$,

d'où l'algorithme :

1. Tirer $U_1, U_2 \in U[0,1]$;

2. $X = -\frac{1}{\lambda} Log\, U_1$;

3. $Y = -\frac{1}{X} Log\, U_2$

4. $W = (X, Y)$.

Si par exemple $\lambda = 2$ et avec la suite aléatoire $U_1 = 0.058962, U_2 = 0.673284, U_3 = 0.479909,$ $U_4 = 0.948578,$ Etape 1 : $X_1 = -0.5Log[0.058962] = 1.41543,$ $X_2 = -\frac{1}{1.41543} Log[0.673284] = 0.279483,$ par suite $X = (X_1, X_2) = (1.41543, 0.279483)$; Etape 2 : $X_1 = -0.5Log[0.0479909] = 0.367079,$ $X_2 = -\frac{1}{0.367079} Log[0.948578] = 0.143814,$ par suite $X = (X_1, X_2) = (0.367079, 0.143814)$. Etc….

Simulation de processus aléatoires

Exercice 12 : Un moniteur matériel (capteurs/sondes) mesure le trafic des paquets arrivant à un nœud d'un réseau. On suppose que le nombre de paquets suit un processus de Poisson . Les temps d'inter-arrivées entre les paquets successifs est en moyenne de 20 ms.

Pour évaluer les performances du réseau, on souhaite remplacer ce moniteur matériel par un moniteur logiciel qui générerait automatiquement le flux de paquets.

(1) Ecrire un algorithme générant ce flux de paquets.
(2) A cause de problèmes de congestion, une proportion de 10% des paquets sont perdus. Compléter l'algorithme précédent en demandant au générateur de trafic de préciser la nature des paquets acceptés ou perdus.
(3) Simuler une réalisation du flux de paquets (acceptés et perdus) à l'aide d'une suite de nombres aléatoires.

Sol : Imaginons la suite aléatoire U_1=0.9234, U_2=0.6547, U_3=0.4532, U_4=0.6754, U_5=0.3214, U_6=0.8238, U_7=0.9748, U_8=0.6562, U_9=0.8782, U_{10}=0.6214.

Algorithme 1. Générer des variables exponentielles jusqu'à épuisement du temps de simulation. Générer à chaque fois une variable de Bernoulli pour savoir si le paquet est accepté ou perdu. Dénombrer le nombre de paquets par simple addition. Exemple : $X_1 = -20Log(0.9234) = 1.59386$; $t_1 = 0 + 1.59386$; puisque $U_2 = 0.6547 > 0.1$, le paquet est accepté ; $X_2 = -20Log(0.4532) = 15.8284$; $t_2 = t_1 + 15.8284 = 17.4223$ etc ….

Algorithme 2. Générer $n = N(T)$ une variable de Poisson. Générer n variables uniformes, les ordonner par ordre croissant , former ensuite $t_1 = TU_{(1)}$, $t_2 = TU_{(2)}$,...

Exercice 13 : Soit la chaîne de Markov à temps discret donnée par la matrice

$$P = \begin{pmatrix} 1/4 & 1/2 & 1/4 \\ 1/3 & 1/3 & 1/3 \\ 1/2 & 1/4 & 1/4 \end{pmatrix} \text{ et de loi initiale } \quad \pi_0 = (1,0,0).$$

(i) Donner un algorithme de simulation d'une telle chaîne.

(ii) Indiquer une méthode alternative permettant de générer plusieurs fois (et successivement un même état). Donner l'algorithme correspondant.

(iii) Illustrer ces méthodes en générant une réalisation (trajectoire) de 3 états à l'aide de la suite uniforme : $U_0 = 0.45, U_1 = 0.87, U_2 = 0.78, U_3 = 0.65, U_4 = 0.86, U_5 = 0.18$.

Exercice 14: Donner l'algorithme de simulation de la chaîne à temps continu de matrice des taux de transition :

$$\Lambda = \begin{pmatrix} -2 & 1 & 1 \\ 1 & -3 & 2 \\ 2 & 2 & -4 \end{pmatrix} \text{ et de loi initiale } \quad \pi_0 = (1,0,0).$$

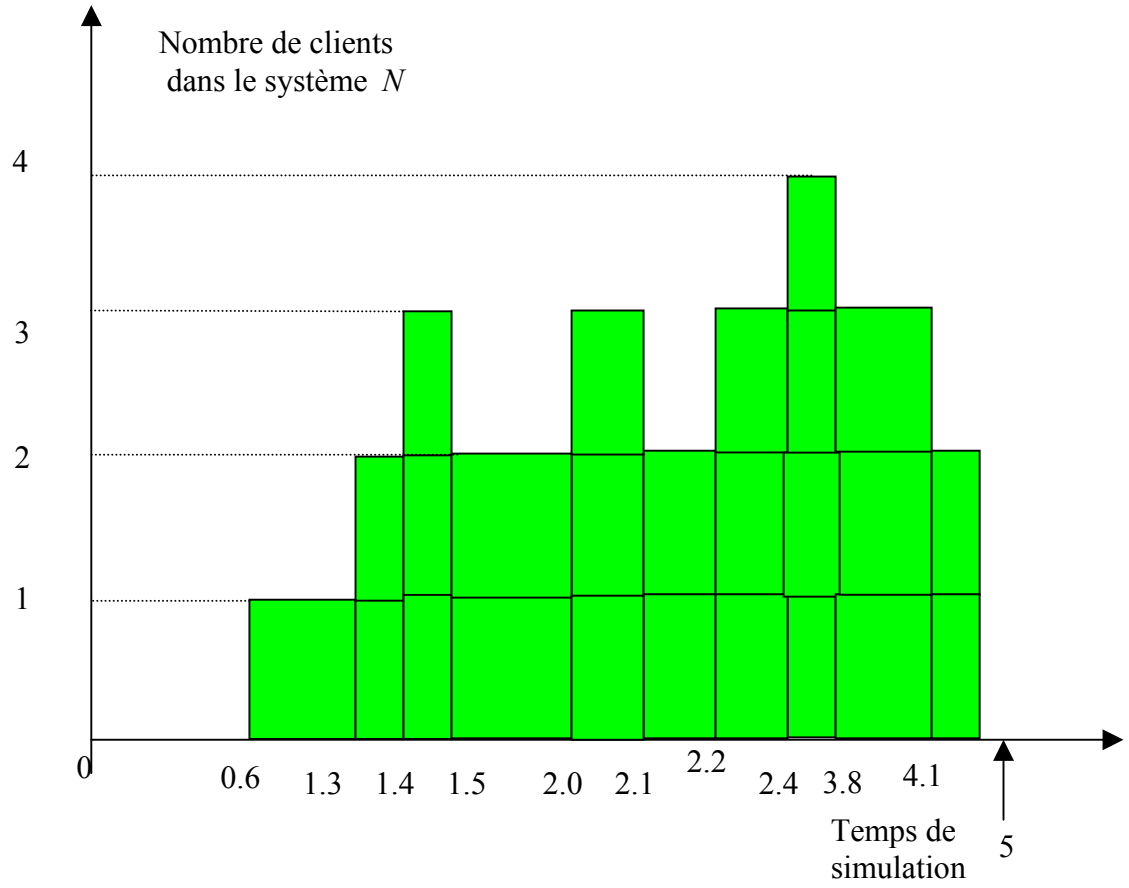
Utiliser la suite précédente pour l'application numérique .

Sol : (i) Si $X_n = 1$, pour $U < 0.25 \Rightarrow X_{n+1} = 1$, $0.25 < U < 0.75 \Rightarrow X_{n+1} = 2$, $0.25 < U < 0.75 \Rightarrow X_{n+1} = 2$ $0.75 < U < 1 \Rightarrow X_{n+1} = 3$ etc… La trajectoire avec la suite : 1,2,3, 3,3, 3,1 .

(i) Générer temps de séjour à un état selon loi géométrique.

Exercice 15. Simuler le comportement des 5 premiers clients au niveau d'une file d'attente FIFO : le processus des arrivées est Poissonien de paramètre 1 arrivée/seconde et le temps de service obéit à une loi d'Exponentielle de paramètre μ=1.1 services/seconde. On fera pour s'exercer la simulation manuellement. Ecrire ensuite un programme qui réalise cette simulation automatiquement.

Sol :



t=horloge de simulation.

A l'aide de Random[], on génère la suite suivante :

U_1=0.526599 ; U_1=0.502976 ; U_1=0.526599 ; U_1=0.526599 ; U_1=0.526599 ; U_1=0.526599 ; U_1=0.526599 ; U_1=0.526599 ; U_1=0.526599 .....

- A l'instant initial l'horloge est mise à zéro
- t=0
- Le compteur du nombre de clients dans le système est aussi initialisé
- N=0
- Générer l'instant d'arrivée du prochain client N°1. Pour cela, il suffit de générer le temps d'inter arrivée ξ_1=-Log U_1=-Log(0.526599)=0.641316 à l'aide du premier nombre aléatoire.
- Actualiser l'horloge t=0+0.641316=0.641316 et le compteur N=0+1=1.
- Générer l'instant de la prochaine arrivée N°2

t :=t+ξ_2=t-Log(U_2)=0.641316-Log(0.502976)=0.641316+0.687213=1.328529
et l'instant de la prochaine fin de service (du client N°1):
t :=t+S_1=t-Log(U_3)=0.641316-Log(0.6611167)=0.641316+0.858731=1.500647

- Le prochain événement est une arrivée N°2 à l'instant t=1.328529 et N :=N+1=2
- Générer la prochaine arrivée N°3
  t :=t+ξ_3=t-Log(U_4)=1.328529-Log(0.89275)=1.328529+0.113449=1.441978
- Le prochain événement est encore une arrivée à l'instant t=1.441978 et N=N+1=3
- L'arrivée suivante N°4:
  t :=t+ξ_4=t-Log(U_5)=1.441978-Log(0.566786)=1.441978+0.567773=2.0097510
- Par conséquent le prochain événement est une fin de service (N°1) à l'instant t=1.500647 et N :=N-1=2
- La prochaine fin de service (du client N°2 qui vient d'accéder au serveur) est à
  t :=t+S_2=t-Log(U_6)=1.500647-Log(0.528431)=1.500647+0.683355=2.184002
- Le prochain événement est donc une arrivée N°4 à t=2.0097510
  Le compteur est mis à jour et N :=N+1=3
- Arrivée N°5 :
  t :=t+ξ_5=t-Log(U_7)=2.0097510-Log(0.814406)=2.0097510+0.205296=2.215046
- Le prochain événement est une fin de service à t=2.184002 ; N :=N-1=2
- Prochaine fin de service
  t :=t+S_3=t-Log(U_8)=2.184002-Log(0.837307)=2.184002+1.65081=3.834812
- Le prochain événement est une arrivée à t=2.215046
  mise à jour du compteur N :=N+1=3
- Prochaine arrivée N°6
  t :=t+ξ_6=t-Log(U_9)=2.215046-Log(0.788154)=2.215046+0.238062=2.453108
- Le prochain événement est une arrivée à t=2.453108 ; N :=N+1=4

Etc……

L'exécution de cette simulation peut être représentée dans le tableau suivant.

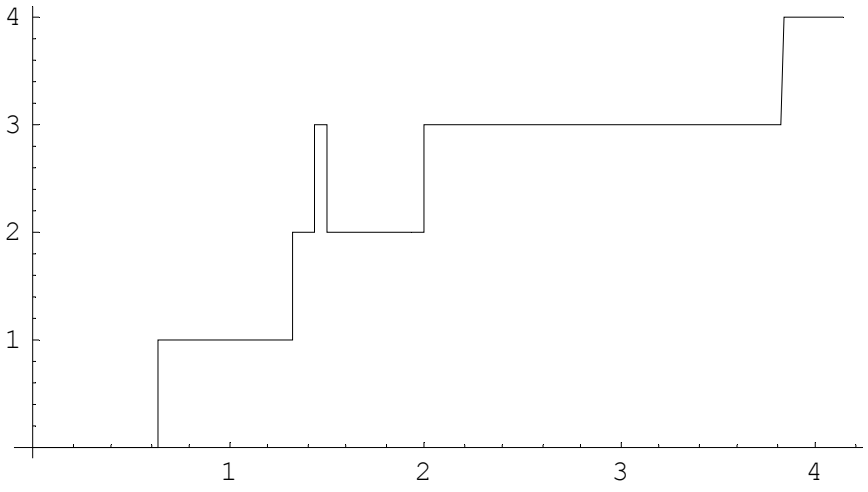
Les deux premières colonnes représentent la mise à jour de l'horloge et l'état du système (nombre de clients dans le système).
La troisième colonne est l'échéancier (event List) qui contient les évènements futurs à l'instant marqué par l'horloge t. En observation, on peut mentionner dans cet exemple, l'événement le plus proche, et enfin la dernière colonne comporte les mises à jour des statistiques des mesures de performance que nous souhaitons évaluer.

Les évènements futurs sont notés sous la forme (événement, date d'occurrence) avec les codes suivants : A pour arrivée, D pour départ et F pour fin de simulation. Pour des

raisons de commodité, nous reportons les dates à deux chiffres après la virgule. Par exemple,
(A ; 0.64) : événement arrivée à l'instant t=0.64.
(F ; 3) : fin de simulation programmée au temps t=5.0
Nous mesurons deux quantités : O : période moyenne d'occupation du serveur
N : nombre moyen de clients
En observation, nous indiquons le plus proche événement

				Statistiques cumulées	
t	N	Echéancier	Observation	O	N
0	0	(A ;0.64) ; (F,5)	(A ;0.64)	0	0
0.64	1	(A ;1.32) ;(D;1.50) ; (F,5)	(A ;1.32)	0.64	1
1.32	2	(A ;1.44) ;(D;1.50) ; (F,5)	(A ;1.44)	1.32	2
1.44	3	(A ;2.00) ;(D;1.50) ; (F,5)	(D;1.50)	1.50	3
1.50	2	(A ;2.00) ;(D;2.18) ; (F,5)	(A;2.00)	2.00	2
2.00	3	(A ;2.21) ;(D;2.18) ; (F,5)	(D;2.18)	2.18	3
2.18	2	(A ;2.21) ;(D;3.83) ; (F,5)	(A ;2.21)	2.21	2
2.21	3	(A ;2.45) ;(D;3.83) ; (F,5)	(A ;2.45)	2.45	3
2.45	4	(A ;5.04) ;(D;3.83) ; (F,5)	(D;3.83)	3.83	4
3.83	3	(A ;5.04) ;(D;4.14) ; (F,5)	(D;4.14)	4.14	3
4.14	2	(A ;5.04) ;(D;6.24) ; (F,5)	(F;5)	5	2

Le graphe suivant représente la trajectoire ci-dessus générée à l'aide de Mathematica.



Estimation du nombre moyen de clients dans le système:

Rappelons que $\hat{N} = \frac{1}{T}\sum_{i=0}^{\infty} iT_i = \frac{1}{T}\int_0^T L(t)dt$. Ici , $\hat{N}=\frac{1}{5}\times$

$\{0\times(0.64-0)+1\times(1.3-0.64)+2\times(1.4-1.3)+2\times(2.0-1.5)+2\times(2.21-2.18)+2\times(5-4.14)$

$+3\times(1.5-1.4)+3\times(2.18-2.0)+3\times(2.4-2.2)+3\times(4.1-3.8)+4\times(3.8-2.4)=\frac{11.58}{5}\approx 3.044\}$

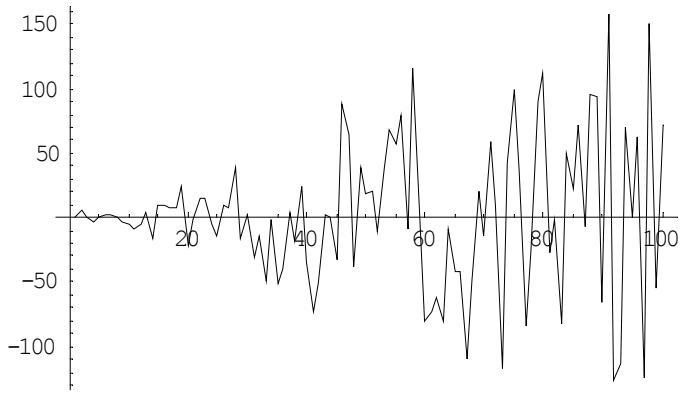
clients.

Exercice 16 Soient deux variables aléatoires U et V indépendantes. La variable U suit une loi exponentielle de paramètre $\lambda = 1$; la variable U suit une loi normale de moyenne N(0,1), de moyenne nulle et variance égale à 1.

1. Indiquer une manière de générer les valeurs du processus $X(t) = U + Vt$ pour $t > 0$ aux instants $t_1 = 1, t_2 = 2, ...$
2. Simuler le processus et étudier expérimentalement ses propriétés de stationnarité en visualisant quelques réalisations du processus et en estimant la moyenne temporelle et la covariance.

Sol :

```
q=Table[-Log[Random[],{i,1,100}]
p=Table[Random[ndist],{I,1,100}]
A=Table[q[[i]]+p[[i]]μi,{i,1,100}]
ListPlot[A,PlotJoined->True]
```



Ce processus n'est visiblement pas stationnaire (plusieurs autres visualisations le confirmerait), comme cela a été montré théoriquement au chapitre 2.

Reprendre l'expérience dans le langage de votre choix.

Exercice 17* : Montrer que certaines lois discrètes usuelles vérifient la condition $p_{i+1} = \alpha_i p_i$, $i = 0,1,...$ce qui permet d'économiser de l'espace mémoire pour l'application de l'algorithme de recherche**.**

Sol :

(i) loi binomiale: $r_i = \dfrac{p_{i+1}}{p_i} = \dfrac{(n-i)p}{(i+1)(1-p)}, i = 0,1,...,n$;

(ii) loi de Poisson : $r_i = \dfrac{\lambda}{i+1}$, $i = 0,1,....$

(iii) Loi géométrique : $r_i = 1-p$, $i = 0,1,...$