

## FILE D'ATTENTE ET SIMULATION

*Ce texte vise à étudier, à l'aide de l'outil informatique et plus particulièrement de la simulation, l'organisation des caisses de paiement à la sortie d'un commerce. Le candidat a pour objectif de décrire quelques méthodes de génération de lois usuellement utilisées en probabilités, et de les appliquer au cadre qui lui est proposé. Il veillera à équilibrer au cours de son exposé les explications relevant de la modélisation, les simulations informatiques et les raisonnements mathématiques.*

### 1. PRÉSENTATION DU MODÈLE

L'organisation des caisses de paiement à la sortie d'un magasin peut être modélisée à l'aide de trois paramètres  $A$ ,  $B$  et  $C$  :

- (1) Le paramètre  $A$  représente le flux d'arrivées des clients aux caisses de paiement. Dans tout le texte, ce flux est modélisé à l'aide d'un espace de probabilité  $(\Omega, \mathcal{A}, \mathbb{P})$  et d'une suite de v.a.  $(T_n)_{n \geq 0}$  vérifiant :

(a)  $T_0 = 0$ , (b)  $\forall n \geq 0, T_n \leq T_{n+1}$ .

De façon claire, pour tout  $n \geq 1$ ,  $T_n$  modélise l'arrivée du  $n$  ème client dans la file de paiement (celle-ci pouvant être vide ou non). Nous nous focaliserons en particulier sur le cas où la suite  $(\tau_n = T_{n+1} - T_n)_{n \geq 0}$  est une suite de v.a. indépendantes et identiquement distribuées.

- (2) Le paramètre  $B$  symbolise la durée du service de chacun des clients. Ces quantités peuvent aussi bien être déterministes qu'aléatoires. Dans le second cas, il est nécessaire de spécifier la loi de ces variables, la façon dont elles dépendent les unes des autres ainsi que la façon dont elles sont corrélées aux variables régissant le flux entrant.
- (3) Le paramètre  $C$  indique des règles de service particulières. Par exemple, il est possible que le commerce compte plusieurs guichets ou que le nombre de clients en attente de paiement soit limité.

Dans ce contexte, nous appelons file d'attente l'ensemble constitué par les personnes en attente de paiement (les clients en cours de paiement en sont donc exclus).

### 2. MODÉLISATION DES TEMPS D'ARRIVÉE

En raison des propriétés d'absence de mémoire des lois exponentielles, nous fixons un réel  $\lambda > 0$  et choisissons de modéliser la loi commune des temps d'inter-arrivées  $(\tau_n)_{n \geq 1}$  par une loi exponentielle de paramètre  $\lambda > 0$ , dont nous rappelons la densité par rapport à la mesure de Lebesgue sur  $\mathbb{R}$  :

$$p_\lambda : x \in \mathbb{R} \mapsto \lambda \exp(-\lambda x) \mathbf{1}_{]0, +\infty[}(x).$$

Dans ce contexte, le paramètre  $\lambda$  représente l'inverse du temps moyen séparant deux arrivées consécutives de clients. En supposant que la taille de la file d'attente ne soit pas limitée, le nombre de clients  $y$  étant entrés (en ne prenant pas en compte les sorties) à un instant  $t \geq 0$  quelconque est donné par la relation :

$$N_t = \sum_{n \geq 1} \mathbf{1}_{[0, t]}(T_n).$$

On rappelle que le processus  $(N_t)_{t \geq 0}$  est un processus de Poisson de paramètre  $\lambda$ . En particulier, il s'agit d'un processus continu à droite dont les accroissements  $(N_{t_{i+1}} - N_{t_i})_{0 \leq i \leq n-1}$ , pour une suite

$t_0 = 0 < t_1 < \dots < t_n$  donnée, sont indépendants et suivent des lois de Poisson de paramètres respectifs  $\lambda t_1, \lambda(t_2 - t_1), \dots, \lambda(t_n - t_{n-1})$ .

Dans les applications que nous allons traiter, nous aurons besoin de simuler des réalisations d'un tel processus. Pour cela, nous proposons une méthode de simulation de la loi exponentielle à l'aide d'un générateur aléatoire de la loi uniforme sur l'intervalle  $]0, 1[$ . Un tel générateur est implémenté dans tous les langages courants et sera désigné ici par la notation `random`. Son fonctionnement ne sera en revanche pas abordé.

Dans cette perspective, nous rappelons le lemme suivant :

**Lemme 2.1** : *Étant donnée une variable aléatoire réelle  $X$ , de fonction de répartition donnée  $F$ , nous définissons :*

$$\forall x \in ]0, 1[, F^{-1}(x) = \inf\{t \in \mathbb{R}, F(t) \geq x\}.$$

*Alors,  $F^{-1}$  est borélienne, et pour une variable aléatoire  $U$  de loi uniforme sur  $]0, 1[$ ,  $F^{-1}(U)$  suit la loi de  $X$ .*

En cherchant à appliquer ce résultat à la loi exponentielle de paramètre  $\lambda > 0$ , un calcul rapide montre que :

**Proposition 2.2** : *Pour tout  $\lambda > 0$  et pour toute variable aléatoire  $U$  de loi uniforme sur  $]0, 1[$ , la variable  $X$  définie par  $X = -\ln(U)/\lambda$  suit une loi exponentielle de paramètre  $\lambda > 0$ .*

En appliquant la proposition 2.2 et en s'appuyant sur la définition du processus de Poisson, nous en déduisons :

**Algorithme 2.3** : *Pour deux entiers  $p, n$  non nuls arbitrairement fixés, l'algorithme suivant permet de simuler une réalisation du vecteur  $(N(1/n), N(2/n), \dots, N(p/n))$  :*

```

j=1
T(1) = - ln(random)/λ
Tant que (T(j)<p/n) faire
    T(j+1) = T(j) - ln(random)/λ
    j=j+1
Fin de boucle
Pour 0 ≤ j ≤ p, N(j) = Nombre de T(i) entre 0 et j/n
Renvoyer le vecteur N

```

Nous avons implémenté cet algorithme en Scilab et représenté en annexe (FIGURE 1) une réalisation du vecteur  $(N(1), N(2), N(3), \dots, N(1000))$ .

### 3. MODÉLISATION DU NOMBRE D'ACHATS D'UN CLIENT EN COURS DE PAIEMENT

Nous cherchons maintenant à modéliser le nombre d'articles achetés par un client en paiement. Dans ce contexte, nous supposons que le magasin offre un large choix de produits, de sorte que nous désignons par  $N$  le nombre total d'articles différents en magasin. Dans un souci de simplification, nous admettons que chaque client n'achète qu'un seul exemplaire du même produit. Dans ces conditions, le nombre de biens que le client envisage acquérir est :

$$(ACH) \quad A = \sum_{j=1}^N Z_j,$$

où, pour tout  $1 \leq j \leq N$ ,  $Z_j = 1$  si le client envisage acheter le  $j$ ème produit et  $Z_j = 0$  dans le cas contraire. De fait, le client se dirige vers la file d'attente si et seulement si  $A \geq 1$ . En particulier, le nombre d'articles présentés par un client à la caisse suit la loi de  $A$  conditionnée à prendre des valeurs supérieures à 1, c'est-à-dire la loi de  $A$  sachant  $\{A \geq 1\}$ .

Afin de déterminer la loi de  $A$ , il n'est pas déraisonnable de supposer que les variables aléatoires  $(Z_j)_{1 \leq j \leq N}$  sont indépendantes et équidistribuées. On désigne alors par  $p$  leur espérance commune. De la relation (ACH), il vient  $\mathbb{E}(A) = pN$ .

Le nombre moyen,  $\mathbb{E}(A)$ , d'articles achetés par un client entrant dans le magasin peut être assimilé à un réel compris entre 0 et 10. De fait, il est très vraisemblable que la loi de la variable  $A$  puisse être modélisée par une loi de Poisson de paramètre  $c = \mathbb{E}(A)$ .

Nous en déduisons finalement que la loi du nombre d'articles présentés à la caisse par un client suit une loi de Poisson de paramètre  $c$  conditionnée à prendre des valeurs supérieures à 1 :

$$\forall j \geq 1, \mathbb{P}\{A = j | A \geq 1\} = \frac{c^j}{j!} (\exp(c) - 1)^{-1}.$$

Nous désignons par  $\nu(c)$  cette loi. Son espérance est donnée par  $\frac{c}{1 - \exp(-c)}$ .

#### 4. SIMULATION DU NOMBRE D'ACHATS ET TEMPS DE SIMULATION

Nous cherchons désormais à simuler informatiquement la loi  $\nu(c)$ . Pour cela, nous allons appliquer le lemme 2.1. Bien-sûr, il est irréaliste de vouloir implémenter l'ensemble des valeurs de la fonction de répartition de  $\nu(c)$ . En revanche, il est possible de les calculer au fur et à mesure de l'algorithme de simulation :

**Algorithme 4.1** : *L'algorithme suivant permet de modéliser une variable aléatoire de loi  $\nu(c)$  :*

```

U = (exp(c)-1)*random
j=1
p=c
Tant que (U>p), faire
    j=j+1
    p=p+c^j/j!
Fin de boucle
Simulation=j

```

Une autre stratégie consiste à s'appuyer sur la connexion entre les lois exponentielles et les lois de Poisson à travers le processus de Poisson. Rappelons en effet que pour une suite  $(\sigma_n)_{n \geq 1}$  de variables aléatoires indépendantes et de même loi exponentielle de paramètre  $c$ , la variable :

$$S = \inf\{n \geq 1, \sigma_1 + \dots + \sigma_n > 1\} - 1,$$

suit une loi de Poisson de paramètre  $c$ . Nous en déduisons l'algorithme suivant :

**Algorithme 4.2** : *L'algorithme suivant permet de modéliser une variable aléatoire de loi de Poisson de paramètre  $c$  :*

```

V=random
j=0
Tant que (V > exp(-c)), faire
    j=j+1
    V=V*random
Fin de boucle
Simulation=j

```

L'algorithme 4.2 ne permet pas de simuler la loi  $\nu(c)$ , mais une loi de Poisson de paramètre  $c$ . De façon générale, le principe de simulation d'une loi conditionnelle s'appuie sur la proposition suivante :

**Proposition 4.3 :** Soit  $(X_n, Y_n)_{n \geq 1}$  une suite de vecteurs aléatoires à valeurs dans  $\mathbb{R}^d \times \{0, 1\}$  indépendants et identiquement distribués, tels que  $\mathbb{P}\{Y_1 = 1\} = 1 - \mathbb{P}\{Y_1 = 0\} = p$ , avec  $p \in ]0, 1[$ . Soit :

$$\tau = \inf\{n \geq 1, Y_n = 1\}, \text{ avec } \inf(\emptyset) = +\infty.$$

Alors,

- (1)  $\mathbb{P}\{\tau < \infty\} = 1$  et  $\tau$  suit une loi géométrique sur  $\mathbb{N}^*$  de paramètre  $p$ .
- (2)  $\forall A \in \mathcal{B}(\mathbb{R}^d)$ ,  $\mathbb{P}\{X_\tau \in A\} = \mathbb{P}(\{X_1 \in A\} | \{Y_1 = 1\})$ .

En appliquant la Proposition 4.3, nous en déduisons un autre algorithme de simulation de  $\nu$  :

**Algorithme 4.4 :** L'algorithme suivant permet de modéliser la loi  $\nu$  :

```

N=Poisson simulée par Algorithme 4.2
Tant que (N=0), faire
  N=Poisson simulée par Algorithme 4.2
Fin de boucle

```

Afin de comparer leurs performances dans le cas arbitraire où  $c$  vaut 5, nous avons implémenté les algorithmes 4.1 et 4.4 sur un serveur classique et sous le langage **Scilab**. En particulier, suite au calcul de 100000 simulations de la loi  $\nu(5)$ , la commande **timer**, utilisée pour déterminer le temps d'exécution d'un programme, nous a renvoyé les valeurs suivantes :

Algorithme 1. Temps de Simulation 15.61

Algorithme 2. Temps de Simulation 18.01

Dans cette situation, il serait donc préférable de s'appuyer sur l'algorithme 1.

Dans la suite, nous désignons, de façon générique, par **randach**( $c$ ) tout générateur de loi  $\nu(c)$ .

## 5. MODÉLISATION DU TEMPS DE SERVICE

Il nous reste désormais à modéliser le temps de paiement d'un client aux caisses. Dans cette perspective, pour  $n \geq 1$ , nous désignons par  $R_n$  le temps de paiement du  $n$ ème client entré dans la file d'attente. Il est raisonnable de formuler les hypothèses suivantes :

- (1) Les variables aléatoires  $(R_n)_{n \geq 1}$  sont indépendantes et identiquement distribuées.
- (2) Les suites  $(R_n)_{n \geq 1}$  et  $(\tau_n)_{n \geq 1}$  sont indépendantes.

En revanche, pour tout  $n \geq 1$ , il paraît vraisemblable que les variables aléatoires  $R_n$  et  $A_n$  soient corrélées. Dans un souci de simplification, nous admettons simplement qu'il existe une relation affine entre le temps de paiement et le nombre d'articles présentés en caisse. Plus exactement, nous formulons l'hypothèse qu'il existe deux constantes  $a, D > 0$  telles que :

$$\forall n \geq 1, R_n = aA_n + D.$$

Notons que cette hypothèse de modélisation ne contredit en rien les contraintes d'indépendance formulées en (1) et (2). En revanche, en vertu de la discussion menée dans la partie 4, nous sommes capables de simuler la loi de  $R_1$ .

## 6. ORGANISATION DE PAIEMENT À GUICHET UNIQUE

Nous visons maintenant à étudier la taille de la file d'attente à un temps donné. En particulier, nous prenons désormais en compte les sorties survenues après la réalisation des services. Dans ce modèle, nous supposons que le serveur ne compte qu'un guichet, que la file peut contenir un nombre illimité de clients et que les clients sont servis dans l'ordre d'arrivée.

On désigne par  $(U_n)_{n \geq 1}$  la suite des instants d'arrivée des clients à la caisse : pour tout  $n \geq 1$ , la variable aléatoire  $U_n$  désigne le temps d'arrivée au guichet du  $n$  ème client entré dans la file. En s'appuyant sur la description du temps de service réalisée dans la partie 5, nous établissons la relation de récurrence :

$$\begin{aligned} (REC) \quad & U_1 = T_1, \\ & \forall n \geq 2, U_n = \max(U_{n-1} + R_{n-1}, T_n). \end{aligned}$$

De fait, il est possible de compter les clients ayant accédé à la caisse selon le processus  $(S(t))_{t \geq 0}$  défini par :

$$\forall t \geq 0, S(t) = \sum_{n \geq 1} \mathbf{1}_{[0,t]}(U_n).$$

On en déduit finalement que la taille de la file d'attente est modélisée par le processus  $(F(t))_{t \geq 0}$  défini par :

$$(FILE) \quad \forall t \geq 0, F(t) = N(t) - S(t).$$

Cette définition permet de proposer un algorithme donnant la taille de la file d'attente jusqu'à un instant donné :

**Algorithme 6.1** : Pour un entier  $p$  non nul arbitrairement fixé, l'algorithme suivant permet de simuler une réalisation du vecteur  $(F(1), F(2), \dots, F(p))$  :

```

T(1) = - ln(random)/λ
U(1) = T(1)
n=1
Tant que (T(n)<p) faire
    T(n+1) = T(n) - ln(random)/λ
    U(n+1) = max(U(n)+ a*randach(c) + D, T(n+1))
    n=n+1
Fin de boucle
Pour 0 ≤ j ≤ p, N(j) = Nombre de T(i) entre 0 et j
Pour 0 ≤ j ≤ p, S(j) = Nombre de U(i) entre 0 et j.
Pour 0 ≤ j ≤ p, F(j) = N(j)-S(j).

```

Nous avons implémenté cet algorithme en Scilab et représenté en annexe (FIGURES 2, 3 ET 4) une réalisation du vecteur  $(F(1), F(2), \dots, F(1000))$  dans les trois cas suivants :

- (1)  $\lambda = 0.15, a = 0.8, c = 5, D = 1,$
- (2)  $\lambda^{-1} = 0.8 \times 5 / (1 - \exp(-5)) + 1 \sim 5, a = 0.8, c = 5, D = 1,$
- (3)  $\lambda = 0.25, a = 0.8, c = 5, D = 1.$

Ces trois graphiques font apparaître les phénomènes suivants :

- (1) Si l'espérance des temps d'inter-arrivées est strictement plus grande que l'espérance du temps de paiement, la file d'attente passe fréquemment par 0. Ceci peut encore s'exprimer de la façon suivante : le temps d'attente s'annule avec une fréquence assez élevée.
- (2) Si l'espérance des temps d'inter-arrivées est égale à l'espérance du temps de paiement, la file d'attente passe par 0, mais la fréquence de libération du guichet semble beaucoup plus faible. En particulier, il existe des longues périodes au cours desquelles la file compte toujours au moins un client.
- (3) Si l'espérance des temps d'inter-arrivées est strictement inférieure à l'espérance du temps de paiement, la taille de la file d'attente explose avec le temps. Le serveur est manifestement saturé.

## 7. CAS D'UN NOMBRE FINI DE GUICHETS

On suppose désormais que l'organe de service dispose de  $k$  guichets, pour un entier  $k \geq 1$  fixé. En revanche, nous conservons les hypothèses suivantes :

- (1) Les clients arrivent selon un processus de Poisson de paramètre  $\lambda > 0$ .
- (2) Les clients en attente forment une unique file d'attente et sont servis dans l'ordre d'arrivée.
- (3) Les temps de paiement sont régis par une relation affine avec le nombre d'achats.

Ajoutons que les guichets sont numérotés de 1 à  $k$  : si deux guichets sont libres, le client choisi celui de plus petit numéro.

Afin de construire un algorithme équivalent à celui proposé dans la partie 6, nous rappelons qu'il est possible de définir sur  $\mathbb{R}^k$  l'application de réordonnement, notée  $\Sigma$ , telle que :

$$(x_1, \dots, x_k) \mapsto \Sigma(x_1, \dots, x_k) = (x_{(1)}, \dots, x_{(k)}), \begin{cases} \exists \sigma \in \mathcal{S}_k, (x_{(1)}, \dots, x_{(k)}) = (x_{\sigma(1)}, \dots, x_{\sigma(k)}), \\ x_{(1)} \leq x_{(2)} \leq \dots \leq x_{(k)}. \end{cases}$$

Rappelons que  $\Sigma$  est borélienne.

En suivant le raisonnement effectué dans la partie 6, nous affirmons qu'il existe une suite  $(V^n = (V_1^n, \dots, V_k^n))_{n \geq k}$  à valeurs dans  $\mathbb{R}^k$ , telle que :

$$\begin{aligned} & \forall 1 \leq i \leq k, U_i = T_i, \\ & V^k = \Sigma(U_1 + R_1, \dots, U_k + R_k), \\ (\text{REC}(k)) \quad & \forall i \geq k + 1, \begin{cases} U_i = \max(V_1^{i-1}, T_i), \\ V^i = \Sigma(U_i + R_i, V_2^{i-1}, \dots, V_k^{i-1}), \end{cases} \end{aligned}$$

$(U_n)_{n \geq 1}$  désignant maintenant la suite des instants d'arrivée des clients aux différentes caisses. Voici finalement l'algorithme que nous proposons pour simuler la taille de la file d'attente jusqu'à un instant donné :

**Algorithme 7.1** : Pour un entier  $p$  non nul arbitrairement fixé, l'algorithme suivant permet de simuler une réalisation du vecteur  $(F(1), F(2), \dots, F(p))$  :

```

T(1) = -ln(random)/λ
U(1) = T(1)
n=1
Tant que (T(n)<p et n < k) faire
    T(n+1) = T(n) - ln(random)/λ
    U(n+1) = T(n+1)
    n=n+1
fin de boucle
V=tri par ordre croissant de (U(1)+a*randach(c)+D,...,U(k)+a*randach(c)+D)
Tant que (T(n)<p) faire
    T(n+1) = T(n) - ln(random)/λ
    U(n+1) = max(V(1),T(n+1))
    V=tri par ordre croissant de (U(n+1)+a*randach(c)+D,V(2),...,V(k))
    n=n+1
Fin de boucle
Pour 0 ≤ j ≤ p, N(j) = Nombre de T(i) entre 0 et j
Pour 0 ≤ j ≤ p, S(j) = Nombre de U(i) entre 0 et j
Pour 0 ≤ j ≤ p, F(j) = N(j)-S(j)

```

Cet algorithme permettrait d'obtenir une étude graphique similaire à celle proposée dans la partie 6.

## 8. PROBLÈME D'OPTIMISATION

Dans ce paragraphe, nous visons à calibrer le nombre de caisses du magasin de façon à ce que la taille de la file d'attente demeure raisonnable au cours d'une journée d'ouverture.

Dans cette perspective, nous cherchons à estimer, pour un réel  $t > 0$  fixé, la probabilité que la taille de la file d'attente franchisse entre les instants 0 et  $t$  un seuil critique  $S$  donné,  $S$  étant un entier non nul. Plus exactement, nous cherchons à déterminer la valeur de :

$$p = \mathbb{P}\left\{\max_{[0,t]} F > S\right\},$$

pour différentes valeurs de  $k$ . Nous choisirons alors le plus petit entier  $k$  rendant cette quantité inférieure à 0.05.

Le calcul théorique de  $p$  est bien-sûr très délicat. Aussi, l'utilisation de la simulation peut-elle se révéler particulièrement efficace : le calcul d'une valeur approchée de  $p$  revient dans ce cas à un calcul numérique d'une espérance à l'aide de la loi des grands nombres.

Dans ce contexte, l'algorithme 7.1 n'est pas le plus pertinent : il permet simplement de calculer des valeurs ponctuelles de la taille de la file, alors que nous souhaitons déterminer son maximum sur  $[0, t]$ . En revanche, l'algorithme 7.1 permet d'obtenir les réalisations des suites  $(T_n)_{n \geq 1}$  et  $(U_n)_{n \geq 1}$  jusqu'à un instant donné. Comme le montre la proposition suivante, le maximum de la file d'attente sur  $[0, t]$  s'obtient par un simple comptage :

**Proposition 8.1 :** *Étant données les suites  $(T_n)_{n \geq 1}$  et  $(U_n)_{n \geq 1}$  préalablement définies, et :*

$$\tau = \max\{n \geq 1, T_n \leq t\},$$

*on définit pour  $\tau \geq 1$  :*

$$\forall 1 \leq i \leq \tau, m_i = \sum_{k=1}^i \mathbf{1}_{\{T_i < U_k\}}.$$

*Alors,*

$$\max_{[0,t]} F = \max_{i=1, \dots, \tau} (m_i), \text{ avec } \max(\emptyset) = 0.$$

Nous en déduisons l'algorithme suivant :

**Algorithme 8.2 :** *Étant donné un réel  $t > 0$  fixé, l'algorithme suivant permet de simuler une réalisation de la variable  $\max_{[0,t]} F$  :*

```

Simuler comme dans Algorithme 7.1 une réalisation de  $(T(1), \dots, T(\tau), U(1), \dots, U(\tau))$ 
Pour  $i=1$  jusqu'à  $\tau$ 
     $m(i) = [\text{Nombre de } U(j) > T(i) \text{ pour } j \in \{1, \dots, i\}]$ .
Fin de boucle
Renvoyer  $\max(m)$ 

```

Afin d'appliquer cet algorithme, nous supposons que :

- (1) L'unité de temps est la minute.
- (2) Le magasin est ouvert 10 heures par jour.
- (3) La constante  $a$  est fixée à trente secondes par article, et la constante  $D$  à une minute et trente secondes.
- (4) Le temps moyen d'inter-arrivée est de une minute et vingt secondes.

- (5) Le nombre moyen d'articles achetés par un client entrant dans le magasin est de 3.
- (6) Le seuil critique est fixé à 20 personnes.

En utilisant ces données numériques, nous avons simulé, pour des valeurs de  $k$  allant de 1 à 4, une réalisation de la loi binomiale de paramètres 100 et  $p$ . Nous avons consigné les résultats obtenus dans le tableau ci-dessous en prenant soin de faire apparaître la moyenne empirique associée :

```

Pour k=1: Realisation de la somme: 100. Moyenne empirique: 1
Pour k=2: Realisation de la somme: 100. Moyenne empirique: 1
Pour k=3: Realisation de la somme: 2. Moyenne empirique: 0.02
Pour k=4: Realisation de la somme: 0. Moyenne empirique: 0

```

Afin de préciser la validité de ces estimations, nous accompagnons ces résultats de la donnée d'intervalles de confiance. Pour cela, nous rappelons que pour une variable aléatoire  $S_n$  de loi binomiale de paramètres  $n \geq 1$  et  $p$  :

$$\forall \varepsilon > 0, \mathbb{P}\left\{p - \frac{S_n}{n} > \varepsilon\right\} \leq \exp(-2n\varepsilon^2).$$

En choisissant  $\varepsilon = \sqrt{-\ln(0.05)/200}$ , nous en déduisons les intervalles de confiance suivants au niveau 0.95 :

- (1) Si  $k = 1$ ,  $p \in ]0.87, 1]$ ,
- (2) Si  $k = 2$ ,  $p \in ]0.87, 1]$ ,
- (3) Si  $k = 3$ ,  $p \in [0, 0.15]$ ,
- (4) Si  $k = 4$ ,  $p \in [0, 0.13]$ .

Les valeurs  $k = 1$  et  $k = 2$  sont d'emblée éliminées. Des précisions semblent en revanche nécessaire pour la valeur  $k = 3$ . Dans cette perspective, un calcul similaire réalisé sur un échantillon de taille 1000 nous a permis d'obtenir, pour  $k = 3$ , une réalisation de la moyenne empirique égale à 0.007 et un intervalle de confiance au niveau 0.95 égal à  $[0, 0.046]$ . De fait, la valeur  $k = 3$  apparaît satisfaisante.

*Fin du texte.*

## 9. PROPOSITIONS DE DÉVELOPPEMENTS

- (1) Dans la partie 2, on pourra expliquer le choix du Processus de Poisson.
- (2) Le candidat pourra détailler les démonstrations du lemme 2.1 et de la proposition 2.2 et expliquer le fonctionnement de l'algorithme 2.3.
- (3) Le candidat pourra justifier la modélisation de la distribution de la variable  $A$  par une loi de Poisson.
- (4) Le candidat pourra expliquer l'implémentation des algorithmes de la partie 4 et réfléchir à la possibilité d'augmenter la vitesse d'exécution.
- (5) Le candidat pourra démontrer la proposition 4.3.
- (6) Dans la partie 7, le candidat pourra expliquer l'implémentation de l'algorithme 7.1 et étudier, sur le modèle de la partie 6, les différents comportements possibles de la file.
- (7) Dans la partie 8, le candidat pourra démontrer la proposition 8.1 ainsi que l'algorithme 8.2.
- (8) En remarquant que les graphiques proposés en annexe font apparaître une interpolation des valeurs de la taille de la file d'attente aux instants 1, 2, ..., 1000, le candidat pourra réfléchir à la possibilité de représenter exactement la réalisation de la file. Il pourra s'inspirer de l'algorithme 8.2.
- (9) Le candidat pourra réfléchir, sur le modèle de la partie 8, à l'application des algorithmes 7.1 et 8.2 à des problèmes d'optimisation.
- (10) Le candidat pourra éventuellement proposer un raffinement des algorithmes 7.1 et 8.2. Au lieu de calculer dans un premier temps les temps d'entrée et d'arrivée aux caisses des clients et de les réordonner par la suite, il pourrait être en effet plus judicieux de proposer un algorithme ordonnant au fur et à mesure ces quantités. Il serait alors pertinent de comparer les vitesses d'exécution des différents algorithmes.

## 10. ANNEXE

Les graphiques ci-dessous font apparaître en abscisse le temps et en ordonnée le nombre de personnes.

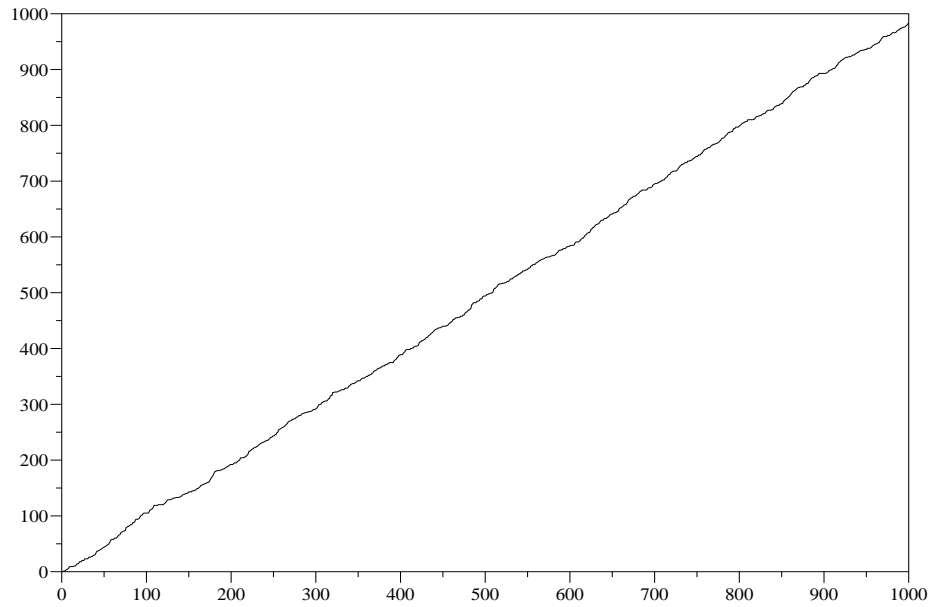


FIGURE 1. NOMBRE DE CLIENTS ENTRÉS DANS LA FILE,  $\lambda = 1$

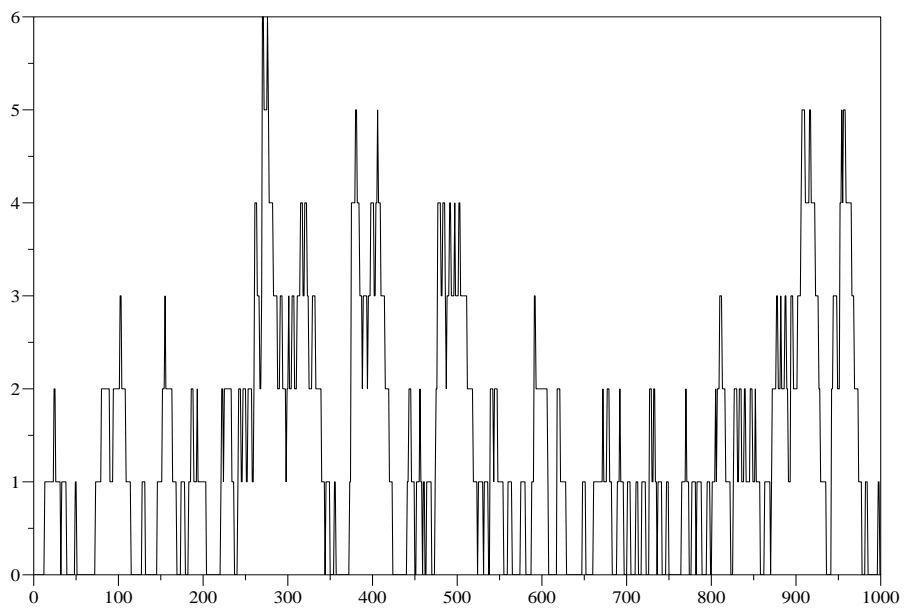


FIGURE 2. TAILLE DE LA FILE À UN GUICHET,  $\lambda = 0.15$ ,  $a = 0.8$ ,  $c = 5$ ,  $D = 1$

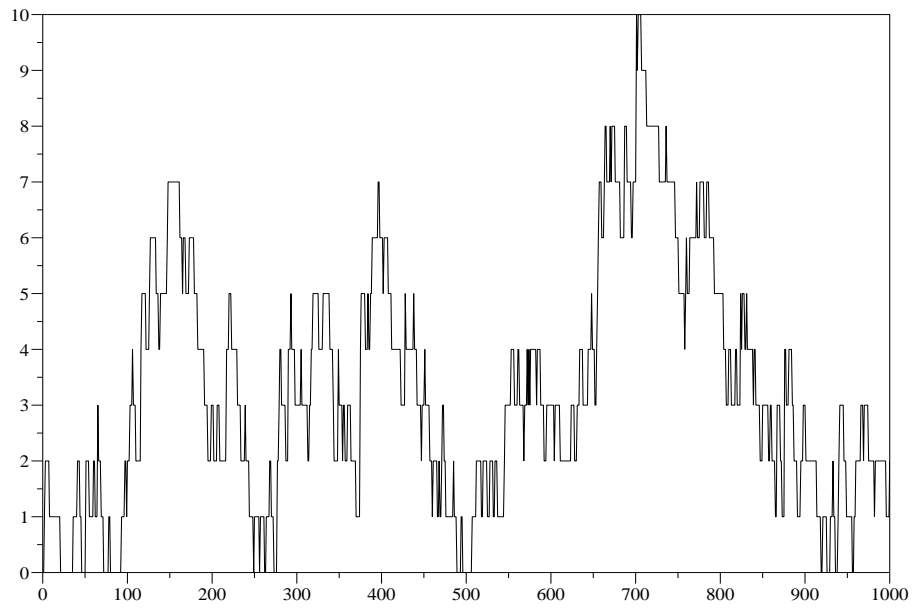


FIGURE 3. TAILLE DE LA FILE À UN GUICHET,  $\lambda \sim 0.2$ ,  $a = 0.8$ ,  $c = 5$ ,  $D = 1$

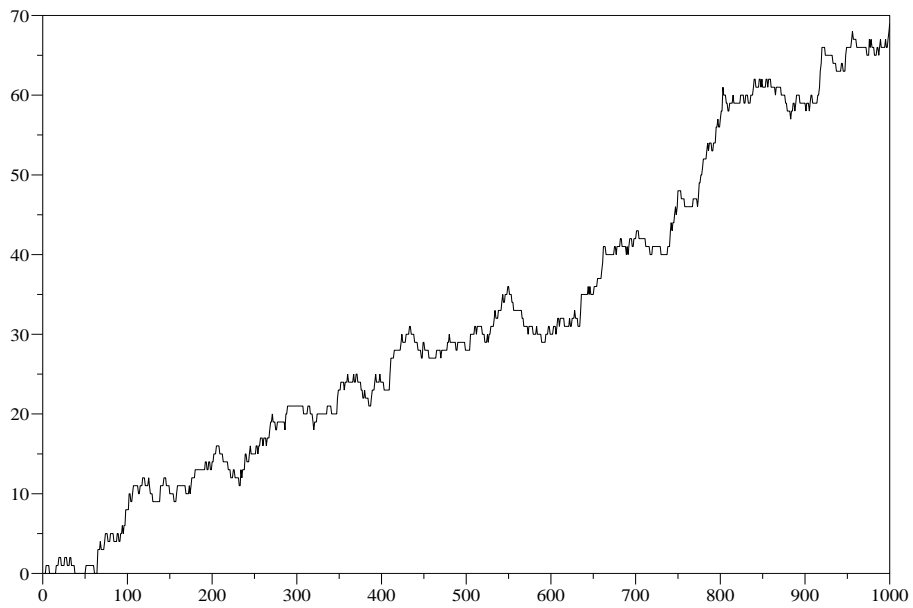


FIGURE 4. TAILLE DE LA FILE À UN GUICHET,  $\lambda = 0.25$ ,  $a = 0.8$ ,  $c = 5$ ,  $D = 1$