Rapport de projet de Processus Stochastiques : Modélisation mathématique d'une file d'attente

Lucie Agnello   Yannick Kouloni   Guillaume Levier

Décembre 2023

Table des matières

1 Introduction

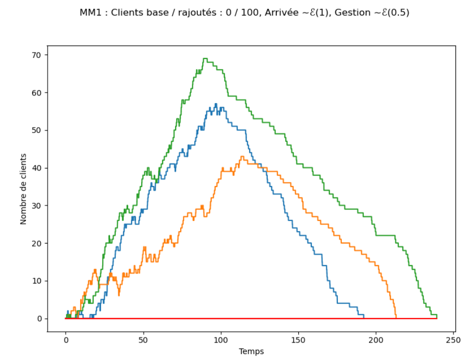
Dans ce projet, nous allons nous intéresser à la modélisation mathématique d'une file d'attente à l'aide d'outils probabilistes robustes abordés dans le cours de processus stochastiques.
Une file d'attente est un système où les clients attendent de manière organisée quelque chose. Les files d'attente font partie des modèles aléatoires les plus répandus et les plus utiles. Le cas le plus simple à décrire est sans doute de considérer ici notre flux d'évènement les clients arrivant séquentiellement réclamant un service devant un guichet appelé serveur ; puisqu'un client occupe le serveur pendant un certain temps, les autres clients doivent attendre avant d'être servis, formant ainsi une file d'attente.
Il existe différents types de systèmes de file d'attente tels que par exemple la file $M/M/1$ et la file $M/M/\infty$ où le premier M indique l'abscence de mémoire des durées entre les arrivées des clients, le second M indique l'abscence de mémoire des durées de services et le 1 final indique enfin qu'il n'y a qu'un seul serveur. Notre projet est consacré particulièrement à la file d'attente $M/M/\infty$ qui est sans doute la plus simple à étudier et où chaque client dispose d'un serveur dédié dès son arrivée qui débute son traitement immédiatement ; une particularité de ce modèle est qu'à tout instant, le nombre de clients dans la file est égal au nombre de clients en cours de service. Des situations concrètes très variées peuvent bénéficier de la modélisation par la file $M/M/\infty$.
En ingénierie, on s'intéresse à des métriques de performance des files d'attente, par exemple la taille moyenne de la file d'attente, le taux d'utilisation du serveur et le temps moyen d'attente d'un client.

2 Modélisation d'une file d'attente par une chaîne de markov à temps continu d'espace d'états $\mathbb{N}$

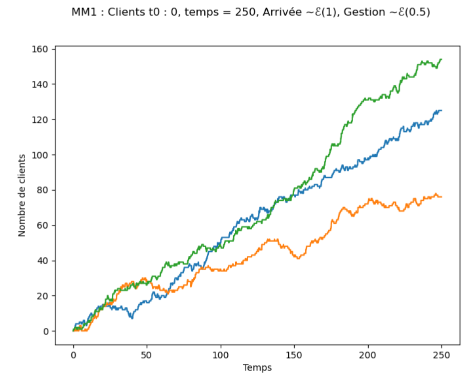
2.1 File d'attente $M/M/1$

Dans ce système de file d'attente, les clients font la queue devant un serveur. On modélise les durées qui séparent les arrivées des clients successifs par des variables aléatoires i.i.d (indépendantes identiquement distribuées) de loi exponentielle de paramètre λ , tandis que les durées de traitement des clients successifs par le serveur sont modélisées par des variables aléatoires i.i.d de loi exponentielle de paramètre μ. Le choix de la loi exponentielle est justifiable par sa propriété d'abscence de mémoire. Dans une nomenclature due à Kendall, on dit qu'il s'agit d'une file $M/M/1$ de taux λ et μ.

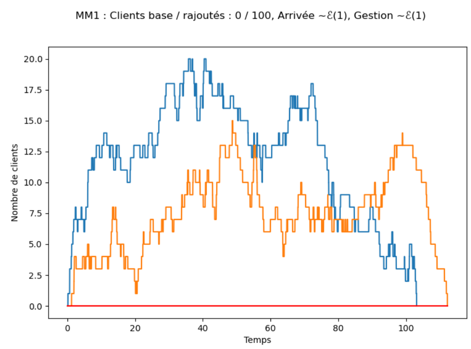
Après implémentation du modèle M/M/1 suite à plusieurs simulations, nous avons obtenu les figures suivantes :



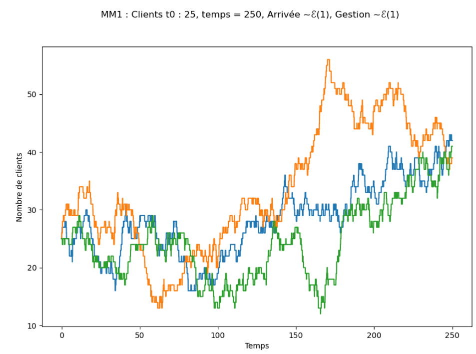
(a) clients : 0+100, $\lambda = 1$, $\mu = 0.5$



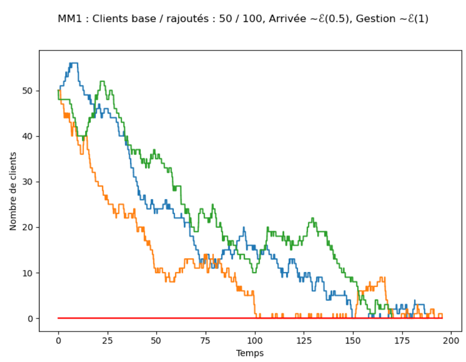
(b) clients base : 0, $\lambda = 1$, $\mu = 0.5$



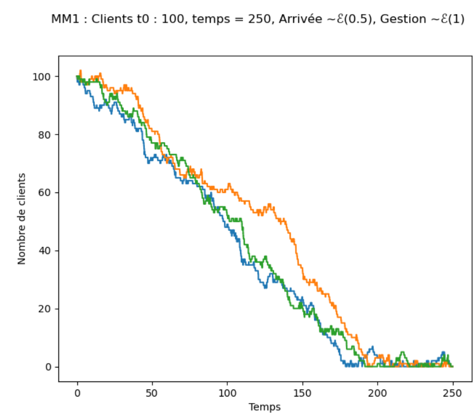
(c) clients : 0+100, $\lambda = 1$, $\mu = 1$



(d) clients base : 25, $\lambda = 1$, $\mu = 1$



(e) clients : 50+100, $\lambda = 0.5$, $\mu = 1$



(f) clients base : 100, $\lambda = 0.5$, $\mu = 1$

FIGURE 1 – M/M/1

Les paramètres généraux de ces simulations de M/M/1 sont λ, μ et le nombre de clients en temps nul. Les schémas à gauche montrent les cas d'une file d'attente dans laquelle on définit le nombre de clients qui arrivent après le temps initial et ceux à droite représentent des files d'attente où le temps maximum est défini.

On remarque dans un premier cas que si les clients affluent plus rapidement qu'ils ne sont traités ($\lambda > \mu$), le nombre de clients dans la file d'attente a tendance à exploser. En haut à gauche, avec un nombre de clients finis, tous les clients sont arrivés au bout d'un certain temps.

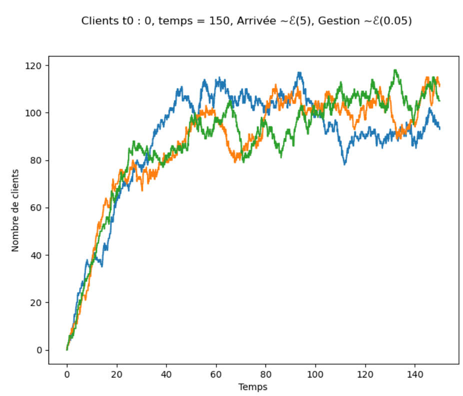
Dans un second cas, si les clients affluent au même rythme qu'ils sont traités ($\lambda = \mu$), le nombre de clients dans la file n'explose pas.
Si le nombre de clients au départ est significativement supérieur à zéro, alors on aura des simulations semblables à une marche aléatoire symétrique.

Dans un troisième cas, si les clients sont plus vite traités qu'ils n'arrivent ($\lambda < \mu$), alors la file aura tendance à être nulle en temps fini, peu importe le nombre de clients dans la file d'attente en temps initial, puis à peu s'éloigner de zéro.

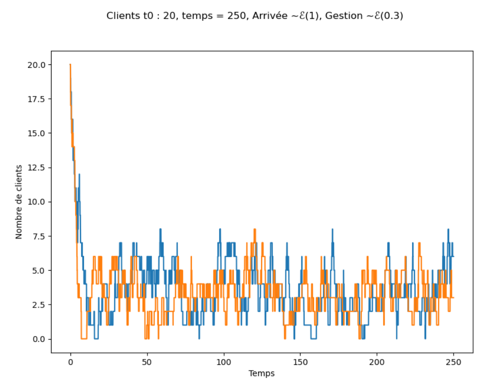
2.2 Cas de la file d'attente $M/M/\infty$

Ce système consiste un analogue à espace discret du processus d'Ornstein-Uhlenbeck. On a une infinité de serveurs identiques et le temps passé dans le système par un client est le temps de service qui est une variable aléatoire i.i.d d'espérance mathématique $\frac{1}{\mu}$ d'après la formule de Little. On montre que ce sytème admet une distribution stationnaire qui est elle une distribution de Poisson. Les durées qui séparent les arrivées des clients sont i.i.d de loi exponentielle de paramètre λ.

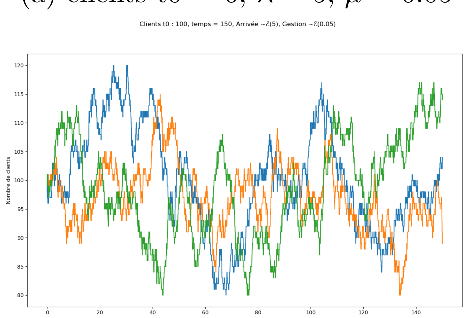
Après implémentation du modèle $M/M/\infty$, nous avons obtenu les résultats de simulations suivantes :



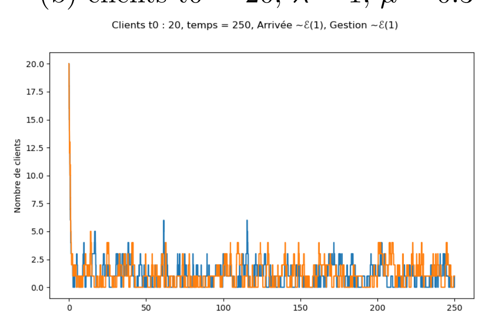
(a) clients t0 = 0, $\lambda = 5$, $\mu = 0.05$



(b) clients t0 = 20, $\lambda = 1$, $\mu = 0.3$

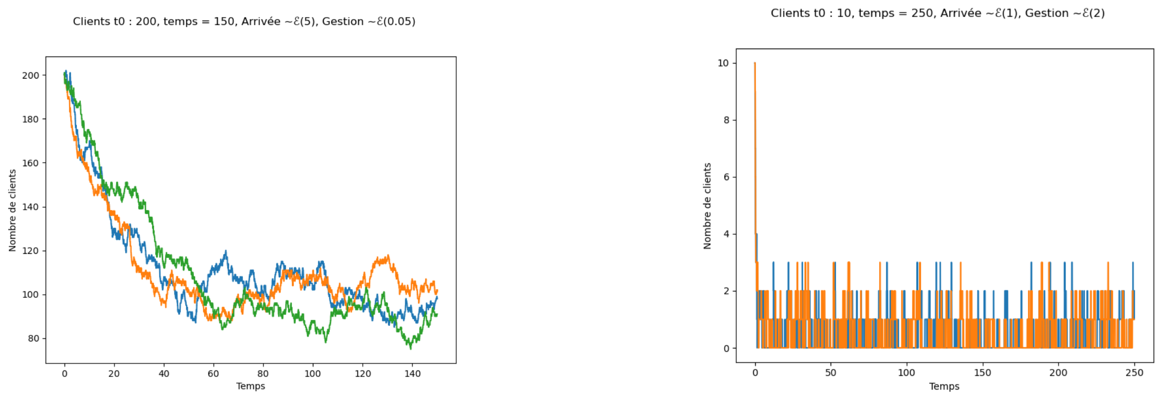


(c) clients t0 = 100, $\lambda = 5$, $\mu = 0.05$



(d) clients t0 = 20, $\lambda = 1$, $\mu = 1$

FIGURE 2 – M/M/∞ (1)



(a) clients t0 = 200, $\lambda = 5$, $\mu = 0.05$

(b) clients t0 = 20, $\lambda = 1$, $\mu = 2$

FIGURE 3 – M/M/∞ (2)

Dans ces simulations, on peut voir à gauche des cas où les clients sont traités bien plus lentement qu'ils n'arrivent ($\lambda >> \mu$). Puisque les clients sont traités dès qu'ils arrivent dans le cas M/M/∞, on remarque pour les cas présentés $(\lambda, \mu) = (5, 0.05)$ que peu importe le nombre de clients au départ, on a en un temps fini que le nombre de clients finit par fluctuer autour de $\frac{\lambda}{\mu}$. Dans la colonne de droite, on augmente μ et on remarque toujours la convergence, qui est d'ailleurs d'autant plus rapide que μ est grand.

2.2.1 Construction rigoureuse du processus $(X_t)_{t\geq 0}$

Soit X_t le nombre de clients dans la file d'attente à l'instant t. On montre que le processus $(X_t)_{t\geq 0}$ est une chaîne de Markov à temps continu d'espace d'états $\mathbb{N}$. On considère également deux suites de variables aléatoires indépendantes :

$(T_n)_{n\in\mathbb{N}}$ correspondant au temps d'arrivée du $n^{\text{ième}}$ client, telle que la durée d'arrivée entre chaque client $T_{i+1} - T_i \sim \mathcal{E}(\lambda)$ et $(S_n)_{n\in\mathbb{N}} \sim \mathcal{E}(\mu)$ où S_n est la durée de service du $n^{\text{ième}}$ client. Un seul client est traité pour une unité de temps.

2.2.2 Cas où la file d'attente est vide au temps initial

Pour construire rigoureusement le processus, on se place tout d'abord dans le cas où la file est vide au temps initial ; soit $X_0 = 0$.
On modélise les durées séparant les arrivées des clients par une suite $(E_n)_{n\geq 0}$ de variables aléatoires i.i.d $\sim \mathcal{E}(\lambda)$, telles que $E_n = T_{n+1} - T_n$. On montre que le $n^{\text{ième}}$ client arrive donc au temps $T_n := E_1 + ... + E_n$, suit une loi $Gamma(n, \lambda)$.
Pour tout réel $t \geq 0$, le nombre de clients arrivés dans l'intervalle de temps $[0, t]$ est modélisée par :

$$N_t := \sum_{n=0}^{+\infty} \mathbb{1}_{T_n \leq t} \tag{1}$$

$(N_t)_{n\geq 0}$ est un processus de comptage.

En effet :
- $N_0 = 0$
- $N_t \in \mathbb{N}, \forall t$
- L'application $t \mapsto N_t$ est croissante .

Étant donné que $(N_t)_{t\geq 0}$ compte le nombre de clients arrivés dans des intervalles de temps indépendants avec une variation de l'intensité d'arrivée λ au cours du temps, le processus de comptage $(N_t)_{t\geq 0}$ est un processus de Poisson issu de 0 et d'intensité λ. On complète maintenant notre modèle en modèlisant la durée de service. Pour tout entier $n \geq 0$, on note S_n la durée de service du $n^{\text{ième}}$ client de sorte que ce client quitte la file au temps $T_n + S_n$. Un seul client peut être traité pour chaque unité de temps. La suite $(S_n)_{n\geq 0}$ est constituée de variables aléatoires i.i.d de la loi exponentielle de paramètre μ.Pour tout $t \in \mathbb{R}_+$, le nombre de clients dans la file au temps t vaut :

$$X_t := \sum_{n=0}^{+\infty} \mathbb{1}_{T_n \leq t < T_n + S_n} \tag{2}$$

On remarque bien que le nombre de clients au temps initial vaut 0. Pour un temps donné, on compte le nombre de clients qui sont entre le temps où ils arrivent et le temps où ils sortent de la file d'attente avec T_n temps d'arrivée du $n^{\text{ième}}$ client et S_n durée de service des clients au temps n.

2.2.3 Cas où la file n'est pas forcément vide au temps initial

Dans le cas général où X_0 n'est pas forcément nul, on introduit une suite $(S'_n)_{n\geq 0}$ de variables aléatoires i.i.d de loi exponentielle de paramètre μ, indépendante de X_0, et on modélise les durées de service de ces clients initiaux par $S'_1, ..., S'_{X_0}$.
On suppose que $X_0, (S_n)_{n\geq 0}, (S'_n)_{n\geq 1}, (E_n)_{n\geq 0}$ sont indépendantes. Pour tout réel $t \geq 0$, le nombre de clients dans la file au temps t vaut donc :

$$X_t := \sum_{n=1}^{X_0} \mathbb{1}_{S'_n > t} + \sum_{n=0}^{+\infty} \mathbb{1}_{T_n \leq t < T_n + S_n} \tag{3}$$

En effet, nous avons deux sommes. La 1ère correspond au temps de traitements des clients initiaux qui sont immédiatement pris en charge. Et en parallèle, les clients arrivants sont également traités dès qu'ils arrivent dans la file d'attente.

Ce cas-là permet de généraliser le processus de comptage que la file soit initialement vide ou pas.

3 Propriétés de la file d'attente markovienne

3.1 Absence de mémoire

L'abscence de mémoire pour la modélisation du temps de service $(S_n)_{n\geq 0}$ resp. de la durée d'arrivée entre chaque client consiste à stipuler que

$$\forall t > 0, \forall s > 0, \mathbb{P}(S_n > s + t \mid S_n > s) = \mathbb{P}(S_n > t) \tag{4}$$

Autrement dit, sachant que le service a démarré à l'instant s, le temps d'achèvement d'un service à un instant donné selon cette condition est le même que s'il débutait à cet instant. Nous pouvons donc relier ce modèle à un modèle markovien.

En résumé, la perte de mémoire dans le contexte des lois exponentielle et géométrique signifie que le temps d'attente futur n'est pas influencé par le temps déjà écoulé sans événement.

3.2 Loi du nombre de clients

Nous avons calculé la variable aléatoire $(X_t)_{t\geq 0}$ dans la sous_partie 2.2.3 qui estime le nombre de clients présents dans la file d'attente au temps t. Notre variable se décompose en deux indicatrices qui correspondent à deux variables aléatoires. Afin de déterminer la loi de $(X_t)_{t\geq 0}$ sachant qu'elle se décompose en somme de deux variables aléatoires et que nous pouvons déterminer leur loi respective, on peut utiliser le produit de convolution.

En effet,

Théorème 1 *Si X et Y sont deux variables aléatoires indépendantes de lois respectives $\mathbb{P}_X$ et $\mathbb{P}_Y$ alors $X+Y$ a pour loi $\mathbb{P}_{X+Y} = \mathbb{P}_X * \mathbb{P}_Y$ le produit de convolution de $\mathbb{P}_X$ et $\mathbb{P}_Y$.*

Par hypothèse, S'_n, T_n, S_n sont indépendants $\forall t \geq 0$, d'après les ressources fournies.
Calculer la loi de X_t sachant $X_0 = k$ revient à étudier X_t dans la file d'attente initialement non vide en utilisant notre équation 4.

On s'intéresse à notre première indicatrice :
$\sum_{n=1}^{X_0} \mathbb{1}_{S'_n > t}$: On a $\mathbb{P}(S'_n > t) = \int_t^{\infty} \mu e^{-\mu x}\, dx = e^{\mu t}$.

On en déduit que $\mathbb{1}_{S'_n > t} \sim \mathcal{B}(e^{-\mu t}) \Rightarrow \sum_{n=1}^{X_0} \mathbb{1}_{S'_n > t} \sim Bin(k, e^{-\mu t})$
avec les S'_n sont i.i.d et $X_0 = k$.

Pour notre deuxième indicatrice :
$Y_t := \sum_{n=0}^{+\infty} \mathbb{1}_{T_n \leq t < T_n + S_n}$:
D'après 2.2.2, nous avons $(N_t)_{t\geq 0} \sim \mathcal{B}(e^{-\mu t})$. Si on veut déterminer la distribution de N_t, le nombre total d'événements dans l'intervalle $[0, t]$, on peut exprimer cela en termes des temps entre les événements, c'est-à-dire T_n. On a :

$$N_t = \max\{n : T_n \leq t\} \tag{5}$$

Lorsque $N_t = n$, cela signifie que n clients sont arrivés dans l'intervalle $[0, t]$. La loi des instants de saut dit que ces n événements sont distribués comme un échantillon réordonné de variables aléatoires i.i.d. uniformes sur $[0, t]$. Cela signifie que les temps d'arrivée des événements sont distribués uniformément dans l'intervalle.

La probabilité qu'un client arrivé à un instant aléatoire (uniforme) dans l'intervalle $[0, t]$ soit encore présent à l'instant t est donnée par $q(t) := 1 - e^{-\mu t}$.

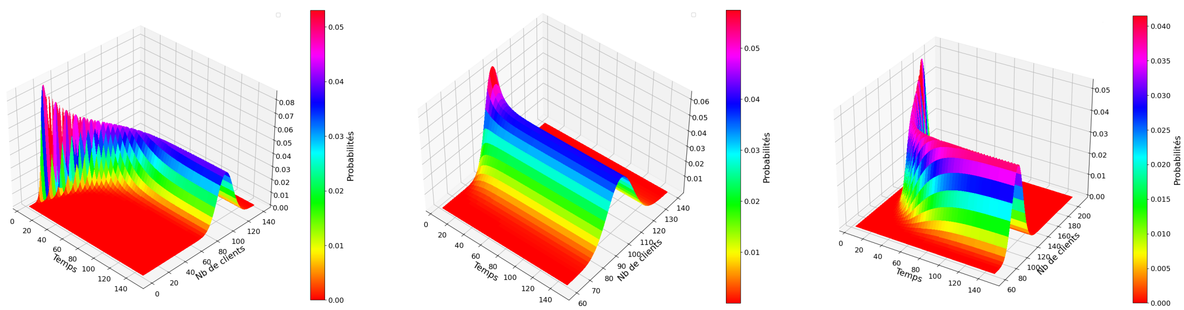
La loi conditionnelle de Y_t sachant que $N_t = n$ est ensuite calculée.

La probabilité $\mathbb{P}(Y_t = k)$ est exprimée comme une somme sur tous les n. En utilisant la loi des instants de saut et la probabilité de présence d'un client, on arrive à la conclusion que, conditionnellement à $N_t = n$, la variable aléatoire (Y_t) suit une loi de Poisson de paramètre $\frac{\lambda(1-e^{-\mu t})}{\mu}$.

Théorème 2 (Loi de X_t) *On montre que pour* $\forall k \in \mathbb{N}$ *et* $\forall t \geq 0$,

$$Loi(X_t \mid X_0 = k) = Bin(k, e^{-\mu t}) * Poi(\tfrac{\lambda}{\mu}(1 - e^{-\mu t}))$$

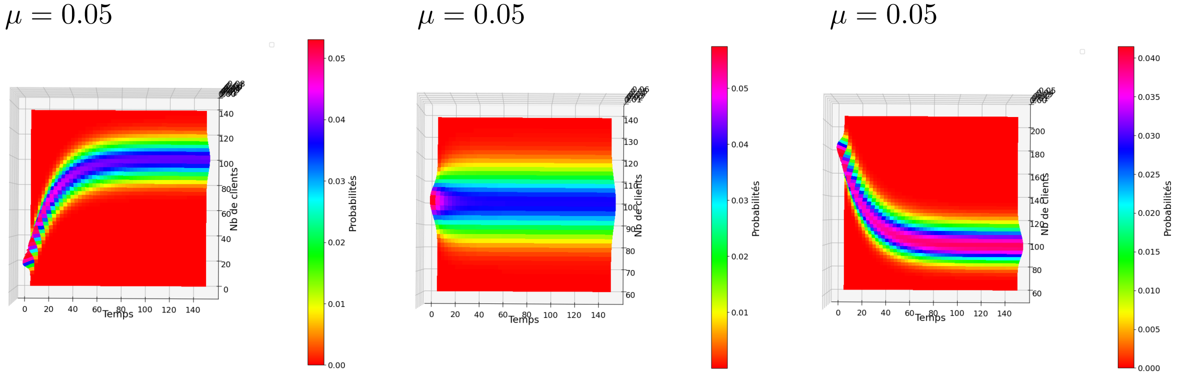
où $\mathbb{1}_{S_1' > t}, \ldots, \mathbb{1}_{S_k' > t}$ *sont des variables aléatoires i.i.d de loi de Bernoulli* $\mathcal{B}(e^{-\mu t})$ *et* $\sum_{n=0}^{+\infty} \mathbb{1}_{T_n \leq t < T_n + S_n} \sim \mathcal{P}(\frac{\lambda(1-e^{-\mu t})}{\mu})$.



(a) clients t0 $= 0$, $\lambda = 5$, $\mu = 0.05$

(b) clients t0 $= 0$, $\lambda = 5$, $\mu = 0.05$

(c) clients t0 $= 200$, $\lambda = 5$, $\mu = 0.05$



(d) clients t0 $= 0$, $\lambda = 5$, $\mu = 0.05$

(e) clients t0 $= 100$, $\lambda = 5$, $\mu = 0.05$

(f) clients t0 $= 200$, $\lambda = 5$, $\mu = 0.05$

FIGURE 4 – Évolution des probabilités d'une file d'attente M/M/∞ en fonction du nombre de clients (en ordonnée) et du temps (en abscisse)

3.3 Comportement en temps long

Étant donné que $(X_t)_{t\geq 0}$ est une chaîne de Markov à temps continu d'espace d'états dénombrable infini $\mathbb{N}$, nous pouvons remarquer que X_t agit de la même façon qu'une marche aléatoire sur $\mathbb{N}$. En effet, les transitions entre les états se produisent en raison des arrivées et des départs dûs au traitement du client. Les arrivées font augmenter de 1 tandis que les départs de 1 font diminuer le nombre X_t. Elles sont basées sur les lois d'arrivée et de sortie de la file suivant le processus de Poisson.
Donc à partir de n'importe quel état i, il existe une probabilité non nulle de revenir à cet état.
Ceci prouve l'existence et l'unicité de la loi invariante de la chaîne, avec pour loi stationnaire $\pi = \mathcal{P}(\lambda/\mu)$. Le nombre moyen de clients dans la file d'attente en régime stationnaire est donc égal à λ/μ, et d'autre part, la loi du temps de séjour dans la file est la loi exponentielle de paramètre μ (chaque client est servi immédiatement).

Le théorème de la section précédente garantit que si la variable initiale $X_0 \sim \pi \Rightarrow \forall t \geq 0, X_t \sim \pi$. Cela signifie que la distribution du processus converge vers une distribution invariante au fil du temps.

Le comportement en temps long se réfère à la manière dont certaines caractéristiques du système évoluent lorsque le temps tend vers l'infini. Ces caractéristiques peuvent inclure certaines mesures de performance, en utilisant le principe du théorème ergodique.

- **Nombre Moyen de Clients :** Le nombre moyen de clients dans la file en temps long est donné par λ/μ, conformément à la formule de la loi de Little. Lorsque le taux d'arrivée moyen λ est inférieur au taux de service moyen μ, le système atteint un état stationnaire où le nombre moyen de clients est stable.
- **Temps Moyen Passé dans la File :** Le temps moyen que chaque client passe dans la file d'attente en temps long est donné par $\frac{1}{\mu}$. Lorsque $\lambda < \mu$, la file d'attente atteint un équilibre stable, et le temps moyen qu'un client passe dans la file diminue à mesure que la différence entre le taux de service et le taux d'arrivée augmente.
- **Convergence Vers la Distribution Stationnaire :** En temps long, la distribution des probabilités d'état tend vers la distribution stationnaire π. Cela signifie que la probabilité d'observer un certain nombre de clients dans la file suit la distribution de Poisson caractéristique de la file M/M/∞, d'après le théorème ergodique.

4 Validation du modèle à l'aide des simulations informatiques

4.1 Prise en compte des paramètres de modélisation

4.1.1 Méthodologie pour la simulation d'une file d'attente

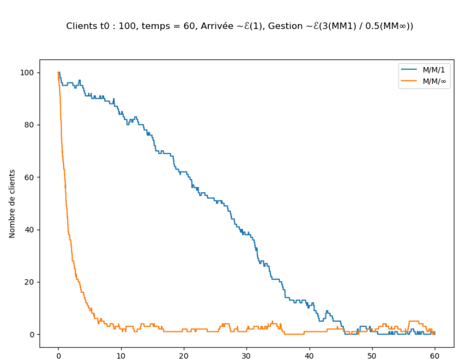
Pour la simulation d'une file d'attente, il faut tenir compte :
— du temps de service de chaque client
— les temps d'arrivée des clients.
Le modèle le plus simple est celui où :
— Le temps de service ne dépend pas des temps d'arrivée ni de la longueur de la file ;
— Les clients arrivent selon un processus de Poisson.
— Dans ce cas, on parle d'une simulation à évènements discrets.

4.1.2 Méthodologie pour la simulation d'un processus de Poisson

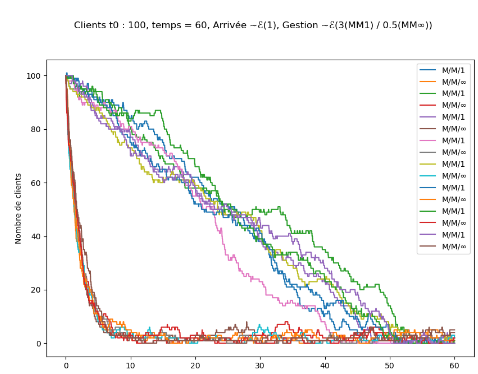
Première méthode :
— On calcule les temps entre les arrivées ;
— Pour cela, il suffit de simuler des variables exponentielles ;
Deuxième méthode :
— On simule d'abord une variable de Poisson sur un intervalle $[a, b]$;
— Puis on simule des variables uniformes autant de fois que la valeur obtenue pour la variable de Poisson ;
— Finalement, on trie en ordre croissant les temps d'arrivée ainsi générés.

4.2 Comparaison des trajectoires des files $M/M/1$ et $M/M/\infty$

Après implémentation des modèles $M/M/1$ et $M/M/\infty$, nous avons obtenu les résultats de simulations sur un même graphique en vue d'une comparaison :



(a) Fig 11.1 du projet



(b) Simulation de 8 trajectoires

Nous observons un début de trajectoire de file d'attente $M/M/1$ de paramètres $\lambda = 1$ et $\mu = 3$, et de file d'attente $M/M/\infty$ de paramètres $\lambda = 1$ et $\mu = 0.5$. Les deux trajectoires sont issues de 100. Le premier processus a pour loi invariante la loi géométrique sur $\mathbb{N}$ de paramètre $\frac{1}{3}$, et le second la loi de Poisson de paramètre $\frac{1}{0.5} = 2$. Dans les deux cas, loi invariante a pour moyenne 2. On voit que la première trajectoire a un comportement linéaire et la seconde un comportement exponentiel.

5 Conclusion

En définitive, nous pouvons noter que, le processus de Poisson est le processus de comptage le plus simple utilisé pour la modélisation mathématique d'une file d'attente ; il s'agit d'ailleurs d'un processus de Markov du fait de sa propriété d'absence de mémoire. Nous constatons que lorsqu'on simule l'arrivée des clients dans la file , il faut évidemment un modèle approprié comme le processus de Poisson d'intensité λ indiquant le nombre moyen de clients arrivant dans la file par unité de temps, qui donne des résultats de simulation très réalistes. En d'autres termes, la probabilité qu'un client arrive à un moment donné ne dépend pas de l'arrivée des autres clients, la probabilité qu'un client arrive à un moment donné ne dépend pas du temps en question et les clients arrivant un seul à la fois. De plus, il faut bien noter que notre modèle de file d'attente $M/M/\infty$ admet une distribution stationnaire qui est une distribution de Poisson ; plus précisement comme loi invariante la loi de Poisson de paramètre $\frac{\lambda}{\mu}$.

Une généralisation naturelle d'un processus de Poisson est appelé processus de renouvellement qui présente une particularité telle que les temps d'arrivée ne soient pas indépendants et permettant d'avoir une distribution autre que la distribution exponentielle pour les temps entre les arrivées.

```python
import numpy as np
from numpy.random import rand
import matplotlib.pyplot as plt
from matplotlib import cm
import scipy.stats as sp

#%% Simulation d'une file d'attente M/M/1 à nombre de clients fixe

def queueMM1_clients_fixes(lbd = 1, mu = 1, nbc = 100, ct0 = 0) :
    #Simulation de la loi exponentielle pour les durées d'arrivée avec cumul pour les temps d'arrivée
    explbdraw=-np.log(rand(nbc))/lbd # durée séparant les arrivées de chaque client
    explbd = np.append(np.zeros(ct0),explbdraw) # les clients initiaux arrivent au temps initial
    cumlbd = np.cumsum(explbd) # cumul pour obtenir les temps d'arrivée

    #Simulation de la loi exponentielle pour les durées de service pour chaque client
    expmu = -np.log(rand(nbc+ct0))/mu #durée de traitement de chaque client

    #Temps auquel chaque client est traité (temps de sortie de la file d'attente)
    traite=np.array([i for i in expmu]) # Le temps de sortie est au moins égal à la durée du service de chaque client
    traite[0]+=cumlbd[0] #Le client initial n'attend pas
    for i in range(1,nbc+ct0) :
        if cumlbd[i] < traite[i-1] : #Si un client est en cours de traitement,
            #le nouveau client sort de la file quand le client devant lui est traité + sa durée de service
            traite[i]+=traite[i-1]
        else : #Si aucun client n'est en cours de traitement, celui qui arrive sort de la file quand il arrive + service
            traite[i]+=cumlbd[i]

    prec=round(traite[-1]*200) # ajuste la précision en fonction du temps de traitement du dernier client
    temps = np.linspace(0,traite[-1]*prec/(prec-1), prec+1)
    #En faisant ça, on a le temps de traitement du dernier client en avant-dernier terme et la file est vidée à la fin

    clients = np.zeros(len(temps))
    clients[0]=ct0 #Clients de départ
    for t in range(1,len(temps)) :
        clients[t]=sum((cumlbd<temps[t]).astype(int))-sum((traite<temps[t]).astype(int))
        #Le nombre de clients présents dans la file est la différence des clients arrivés et de ceux sortis de celle-ci.
    return temps, clients


```

```python
#Affichage du graphe pour des paramètres donnés ci-dessous
lbd = .5 ; mu = 1 ; nb_clients = 100 ; clients_init = 50

t1,c1 = queueMM1_clients_fixes(lbd, mu, nb_clients, clients_init)
plt.figure(1)
plt.plot(t1,c1)
plt.plot(np.array([t1[0],t1[-1]]),np.array([0,0]),'r')
plt.xlabel("Temps")
plt.ylabel("Nombre de clients")
plt.suptitle("MM1 : Clients base / rajoutés : "+str(clients_init)+" / "+str(nb_clients)+
             ", Arrivée ~ℰ("+str(lbd)+"), Gestion ~ℰ("+str(mu)+")")
plt.show()

#%% Simulation d'une file d'attente M/M/1 à temps fixe

def queueMM1_temps_fixe(lbd = 1, mu = 1, tps = 100, ct0 = 0) :
    #Simulation de la loi exponentielle pour les durées d'arrivée tant que le temps max n'est pas atteint
    if ct0>0 :
        cumlbd = np.zeros(ct0)
        #Les clients présents au départ arrivent en temps nul
    else :
        cumlbd = np.array([-np.log(rand())/lbd])
        #S'il n'y a pas de client, on laisse le premier client arriver
    while cumlbd[-1]<tps :
        cumlbd = np.append(cumlbd,cumlbd[-1]-np.log(rand())/lbd)
        #Les clients affluent tant que le temps max n'est pas atteint

    prec=round(tps*200) # ajuste la précision en fonction du temps défini
    temps = np.linspace(0, tps, prec)

    #Temps de traitement du premier client : il n'attend pas
    traite=np.array([cumlbd[0]-np.log(rand())/mu])

    #De la même façon, tant que le temps max n'est pas atteint, on traite des clients
    i=1
    while traite[-1] < tps :
        if cumlbd[i] < traite[i-1] :
            traite = np.append(traite, traite[i-1]-np.log(rand())/mu)
        else :
            traite = np.append(traite, cumlbd[i]-np.log(rand())/mu)
        i=i+1

    clients = np.zeros(prec)
    clients[0] = ct0 #Clients de départ
    for t in range(1,prec) :
        clients[t]=sum((cumlbd<temps[t]).astype(int))-sum((traite<temps[t]).astype(int))
        #Le nombre de clients présents dans la file est la différence des clients arrivés et de ceux sortis.

    return temps, clients

```

```
#Affichage du graphe pour des paramètres donnés ci-dessous
lbd = 1 ; mu = .8 ; temps = 250 ; clients_init = 60

t2,c2 = queueMM1_temps_fixe(lbd, mu, temps, clients_init)
plt.figure(2)
plt.plot(t2,c2)
plt.xlabel("Temps")
plt.ylabel("Nombre de clients")
plt.suptitle("MM1 : Clients t0 : "+str(clients_init)+", temps = "+str(temps)+
             ", Arrivée ~ℰ("+str(lbd)+"), Gestion ~ℰ("+str(mu)+")")
plt.show()

#%% Simulation d'une file d'attente M/M/∞ à temps fixe

def queueMMinf_temps_fixe(lbd=1,mu=1,tps=100,ct0=0) :
    #De la même façon : simulation de la loi exponentielle pour les durées d'arrivée avec cumul pour les temps d'arrivée
    if ct0>0 :
        cumlbd = np.zeros(ct0)
    else :
        cumlbd = np.array([-np.log(rand())/lbd])
    while cumlbd[-1]<tps :
        cumlbd = np.append(cumlbd,cumlbd[-1]-np.log(rand())/lbd)

    prec=round(tps*200) # ajuste la précision en fonction du temps défini
    temps = np.linspace(0, tps, prec)

    #Temps de traitement pour chaque client
    traite = cumlbd - np.log(rand(len(cumlbd)))/mu

    clients = np.zeros(prec)
    clients[0]=ct0 #Clients de départ
    for t in range(1,prec) :
        clients[t]=sum((cumlbd<temps[t]).astype(int))-sum((traite<temps[t]).astype(int))
        #Le nombre de clients présents dans la file est la différence des clients arrivés et de ceux traités.

    return temps, clients

#Affichage du graphe pour des paramètres donnés ci-dessous
lbd = 5 ; mu = .05 ; temps = 150 ; clients_init = 200

t3,c3 = queueMMinf_temps_fixe(lbd, mu, temps, clients_init)
plt.figure(3)
plt.plot(t3,c3)
plt.xlabel("Temps")
plt.ylabel("Nombre de clients")
plt.suptitle("Clients t0 : "+str(clients_init)+", temps = "+str(temps)+
             ", Arrivée ~ℰ("+str(lbd)+"), Gestion ~ℰ("+str(mu)+")")
plt.show()
```

```
#%% Simulation de la figure 11.1 en bas du PDF

temps = 60 ; clients_init = 100 ; lbd = 1

plt.figure(4)

mu1 = 3
t4,c4 = queueMM1_temps_fixe(lbd, mu1, temps, clients_init)
plt.plot(t4, c4, label = "M/M/1")

mu2 = .5
t5,c5 = queueMMinf_temps_fixe(lbd, mu2, temps, clients_init)
plt.plot(t5, c5, label = "M/M/∞")

plt.xlabel("Temps")
plt.ylabel("Nombre de clients")
plt.suptitle("Clients t0 : "+str(clients_init)+", temps = "+str(temps)+
             ", Arrivée ~ℰ("+str(lbd)+"), Gestion ~ℰ("+str(mu1)+"(M/M/1) / "+str(mu2)+"(M/M/∞))")
plt.legend()
plt.show()

#%% Quantité de clients arrivés à un temps T

def clients(tps = 100, lbd = 1) :
    c = 0
    s = -np.log(rand())/lbd
    while s < tps :
        #Incrémente jusqu'à atteindre le temps demandé
        s-=np.log(rand())/lbd
        c+=1
    return c

print(clients(tps = 100, lbd = .5))
print(clients(tps = 100, lbd = 1))
print(clients(tps = 100, lbd = 2))
print(clients(tps = 50, lbd = 1))
```

```python
#%% Quantité de clients dans la file d'attente à un temps T

def clinqueue_MMinf(tps = 100, lbd = 1, mu = 1, ct0 = 0) :
    if tps < 0 :
        #Les clients arrivent en temps positif
        return 0
    clitps = 0
    for i in range(ct0) :
        if -np.log(rand())/mu > tps :
            clitps+=1
        #Si les clients présents au départ ne sont pas encore traités, ils sont comptés

    #Temps d'arrivée du premier client dans la file d'attente et temps de départ de celui-ci
    cliarr = -np.log(rand())/lbd
    cliserv = cliarr - np.log(rand())/mu
    if cliarr <= tps and tps < cliserv :
        clitps+=1
        #Si au temps donné, le premier client est arrivé dans la file d'attente mais n'en est pas sorti, il est compté

    while cliarr < tps or cliserv < tps :
        #Si les clients continuent d'affluer ou d'être traités jusqu'au temps donné, on les prend en compte
        cliarr = cliarr - np.log(rand())/lbd
        cliserv = cliarr - np.log(rand())/mu
        #Temps d'arrivée et de sortie du nouveau client
        if cliarr <= tps and tps < cliserv :
            clitps+=1
            #Si le temps donné est entre le temps d'arrivée du nouveau client et son temps de sortie, il est compté
    return clitps

print(clinqueue_MMinf(tps = 100, lbd = 1, mu = .02, ct0 = 0))
print(clinqueue_MMinf(tps = 100, lbd = 1, mu = .02, ct0 = 100))
print(clinqueue_MMinf(tps = 20, lbd = 1, mu = .02, ct0 = 100))
print(clinqueue_MMinf(tps = 2, lbd = 1, mu = .5, ct0 = 100))

#%% Simulation de X_t

def binom(n,p) : #Simulation d'une loi binomiale
    return sum((rand(n)<p).astype(int))

def pois(lbd) : #Simulation d'une loi de Poisson
    s = 0
    n = -1
    while s<=1 :
        n+=1
        s-=np.log(rand())/lbd
    return n
```

```python
def X_t(lbd, mu, ct0, t) :
    #X_t est défini comme un produit de convolution entre les deux
    #Donc il est simulé comme la somme des deux lois
    return binom(ct0, np.exp(-mu * t)) + pois((lbd/mu)*(1-np.exp(-mu * t)))

print(X_t(lbd = 1, mu = .02, ct0 = 0, t = 100))
print(X_t(lbd = 1, mu = .02, ct0 = 100, t = 100))
print(X_t(lbd = 1, mu = .02, ct0 = 100, t = 20))
print(X_t(lbd = 1, mu = .5, ct0 = 100, t = 2))


#%% Probabilités exactes associées (en utilisant scipy.stats et pas de simulation)

def pbin(k, n, p) :
    return sp.binom.pmf(k, n, p)

def ppoi(k, lbd) :
    return sp.poisson.pmf(k,lbd)

def proba_X_t(lbd, mu, ct0, t, n) :
    som = 0
    if n < 0 or t < 0 :
        return 0
        #Les clients ne sont ni en nombre négatif ni n'arrivent en temps négatif
    else :
        for m in range(min(ct0,n)+1) :
            # P(X=n) = somme(m=-inf -> +inf) P(B = m) * P(P=n-m)
            # avec B la loi binomiale et P la loi de Poisson
            # m est dans Z tout entier mais la loi binomiale est définie sur {0, ... , k}
            # et la loi de Poisson est définie sur N donc n- m 0  donc m n
            # donc  m 0  et  m n  et  m k  donc m est compris entre 0 et min(n,k) (+1 car Python)
            som += pbin(m, ct0, np.exp(-mu * t)) * ppoi(n - m, lbd/mu * (1 - np.exp(-mu * t) ) )
        return som
```

```python
#%% Probas pour lbd = 1 ; mu = .02
# Plus le nombre de clients au départ est faible, plus le temps de calcul est rapide
# puisque le min entre ct0 et n est faible dans le range de m juste au-dessus.

lbd = 1 ; mu = .02 ; clients_init = 100 ; tmax = 251

T = np.array([i for i in range(tmax)])
N = np.array([i for i in range(30,105)])
P = np.array([[proba_X_t(lbd, mu, clients_init, t, n) for t in T] for n in N])
#On affiche les probas pour tous les temps et tous les nombres de clients tant qu'elle est pas quasi nulle
#(ici, la proba avec ces paramètres d'avoir moins de 30 clients est quasi nulle)

t0 = 5
#On part de t0=5 parce qu'on est p.s. au nombre de clients initiaux en temps 0
#Et les probas diminuent drastiquement après peu de temps

Tp = np.array([i for i in range(t0, tmax)])
PP = np.array([P[i,t0:] for i in range(len(P))])

fig = plt.figure(6)
ax = plt.axes(projection = '3d')

TT,NN=np.meshgrid(Tp,N)
surf = ax.plot_surface(TT, NN, PP, cmap = cm.hsv, linewidth = 0, antialiased = False)
ax.set_xlabel('Temps', fontsize = 20)
ax.set_ylabel('Nb de clients', fontsize = 20)

cbar = fig.colorbar(surf, ax = ax)
cbar.set_label(label = 'Probabilités', size = 20)
cbar.ax.tick_params(axis = 'both', which = 'major', labelsize = 20)

ax.tick_params(axis = 'both', which = 'major', labelsize = 18)
plt.legend()
plt.show()
```

```
#%% Probas pour lbd = 5 ; mu=.05 (mêmes commentaires)

lbd = 5 ; mu = .05 ; clients_init = 0 ; tmax = 151

T = np.array([i for i in range(tmax)])
N = np.array([i for i in range(0,140)])
P = np.array([[proba_X_t(lbd, mu, clients_init, t, n) for t in T] for n
     in N])

t0 = 5

Tp = np.array([i for i in range(t0, tmax)])
PP = np.array([P[i,t0:] for i in range(len(P))])

fig = plt.figure(7)
ax = plt.axes(projection ='3d')

TT,NN=np.meshgrid(Tp,N)
surf = ax.plot_surface(TT, NN, PP, cmap = cm.hsv, linewidth = 0,
    antialiased = False)
ax.set_xlabel('Temps', fontsize = 20)
ax.set_ylabel('Nb de clients', fontsize = 20)

cbar = fig.colorbar(surf, ax = ax)
cbar.set_label(label = 'Probabilités', size = 20)
cbar.ax.tick_params(axis = 'both', which = 'major', labelsize = 20)

ax.tick_params(axis = 'both', which = 'major', labelsize = 18)
plt.legend()
plt.show()
```