Modélisation d'une file d'attente

Zinedine Bounnah, Thibaut Desix, Maxime Grouazel

Décembre 2022

Table des matières

1 Introduction

Les files d'attentes font partie intégrante des problèmes de la vie quotidienne. On les retrouve dans de nombreuses situations et leur étude est clé pour des raisons multiples.

Prenons l'exemple de clients qui font la queue pour passer à la caisse : la problématique pour l'entreprise est de pouvoir contrôler le nombre de clients présents dans la file d'attente en ouvrant le nombre de caisses nécessaires à un trafic fluide. Cependant, un nombre de caisses plus élevé peut entraîner une perte pour l'entreprise qui paie les employés. On voit donc que ce problème nécessite une modélisation mathématique pour en étudier les paramètres.

Les files d'attente sont caractérisées par des paramètres décrits par la notation de Kendall. On note une file d'attente A/B/s/k/Z, où A et B sont les lois que suivent les instants d'arrivées et services, s est le nombre de serveurs de la file, k la capacité maximale de la file, m la population totale de clients et Z la politique de service.

Les modèles de file d'attente dépendent donc d'une politique de service qui peut être "first in first out" (premier arrivé premier servi), "last in first out" (dernier arrivé premier servi) ou encore "shortest job first" (celui dont la durée de traitement est la plus petite est pris en charge en premier) qui donne une approche différente du problème. On peut aussi prendre en compte d'autres paramètres comme l'impatience, la priorité, l'incertitude entre autres pour se rapprocher le plus d'un cas concret de la vie quotidienne.
On peut ainsi modéliser de nombreux cas concrets comme un parking de K places par la file M/M/K/K où les deux K représentent le nombre de guichets et la limite de capacité de la file, la saturation d'un trafic routier, d'un réseau de télécommunications, gestion d'un stock de production, maintenance d'un équipement informatique, mouvements de populations, prévisions météorologiques, etc.

Dans cette présentation, nous considérerons un des cas les plus élémentaires pour une file d'attente : le cas d'une file d'attente M/M/1 avec une politique de service "first in first out".

2 File d'attente M/M/1

Le but de notre travail va donc être de comparer des résultats théoriques sur la file M/M/1 avec des résultats obtenus numériquement.

2.1 Définition

Le modèle M/M/1 correspond à une unique file d'attente où les clients qui arrivent sont servis un à un, M (Markov) indiquant l'absence de mémoire des processus modélisant les arrivées et services.
Si un client arrive alors qu'aucun autre client n'est en train d'être servi, ie. la queue ne contient pas de client, le client est servi directement, sinon il doit attendre dans la queue que les clients précédents soient servis.

Cette file est caractérisée par un processus de Poisson de paramètre λ pour les arrivées (avec donc des temps entre chaque arrivée suivant une loi exponentielle de même paramètre), et par des temps de service suivant une loi exponentielle de paramètre μ. Les temps d'arrivée et de service sont indépendants.

Le processus de Poisson est un processus de comptage, autrement dit N(t) représente le nombre de réalisations d'un événement à un instant t donné en partant du temps 0. C'est un processus à temps continu et à espace d'états discret.

Un processus de Poisson vérifie:

-l'indépendance des accroissements: Pour toute suite $0 \leq t_1 \leq t_2 \leq \ldots \leq t_k$, les variables aléatoires N(t_{i+1})-N(t_i), $0 \leq i \leq k-1$, sont indépendantes.

-La stationnarité du processus: Le nombre d'arrivée dans un intervalle de longueur h suit une loi de Poisson de paramètre λ h:

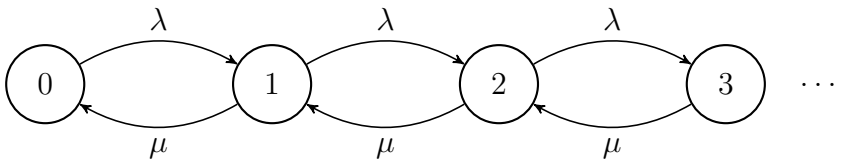
$$P[N(t+h) - N(t) = k] = e^{-\lambda h} \frac{(\lambda h)^k}{k!} \tag{1}$$

On a : $N(t+h) - N(t) \stackrel{d}{=} N(h) - N(0) = N(h)$, les accroissements ne dépendent que de la taille de l'intervalle.

3 Modélisation de la file M/M/1

3.1 Chaîne de Markov continue

Le processus $X(t)$ qui compte le nombre de personne dans la file d'attente au temps t peut être vu, par définition, comme une chaîne de markov continue et plus précisément comme un cas de processus naissance-mort.
Le diagramme de taux de transition de $X(t)$ est le suivant :



et sa matrice d'intensité la suivante :

$$Q = \begin{pmatrix} -\lambda & \lambda & & \\ \mu & -(\lambda+\mu) & \lambda & \\ & \mu & -(\lambda+\mu) & \lambda \\ & & \ddots & \end{pmatrix}$$

3.2 Temps d'attente avant d'observer un mouvement dans la file

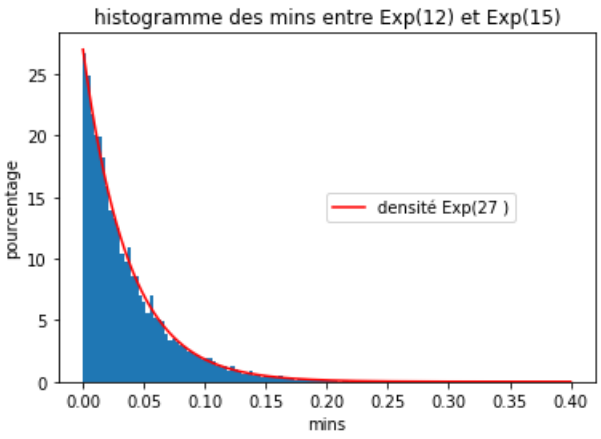
S'il n'y a personne dans la file d'attente la chaîne va directement vers l'état 1 et le temps d'attente sera alors déterminé par une variable aléatoire suivant une loi exponentielle de paramètre λ.
Sinon, il est déterminé par le minimum entre le temps d'attente d'une arrivée et celui d'un départ.

Comme ce minimum est celui de deux variables aléatoires suivant une loi exponentielle de paramètre respectif λ et μ, il suit lui aussi une loi exponentielle mais cette fois-ci de paramètre $\lambda + \mu$ (On voit donc bien que c'est une chaine de markov continue).

Preuve. Soient $X \sim Exp(\lambda), Y \sim Exp(\mu)$ et $Z = min(X, Y)$ avec X, Y indépendants.
$\mathbb{P}(Z < t) = \mathbb{P}(\{X < t\} \cup \{Y < t\}) = 1 - \mathbb{P}(X > t)\mathbb{P}(Y > t) = 1 - e^{\lambda+\mu}$ □

Numériquement :

-On simule X un vecteur de réalisation de $X \sim Exp(\lambda)$ et Y, un de $Y \sim Exp(\mu)$. Ces deux vecteurs sont de même taille n avec n grand.
-On extrait les minimums entre chacune des réalisations de X et de Y.
-On trace l'histogramme des minimums et on ajoute la densité d'une loi exponentielle de paramètre $\lambda + \mu$ à l'histogramme.
Pour $\lambda = 12, \mu = 15$ on obtient :



3.3 Probabilités de transition

S'il n'y a personne dans la file la probabilité d'aller en 1 vaut 1 : $p_{0,1} = 1$. Sinon, on peut voir la probabilité d'avoir une personne de plus dans la file comme la probabilité que le temps d'attente d'une arrivée, T_a soit plus petit que le temps d'attente d'un départ, T_d. Pour $i \in \mathbb{N}_*$ avec i l'état de la chaîne,

Preuve.

$$\begin{aligned}
\mathbb{P}(X < Y) &= \int_0^\infty \int_0^y \lambda e^{-\lambda x} \mu e^{-\mu y} dx dy \\
&= \int_0^\infty \mu e^{-\mu y}(1 - e^{-\lambda y}) dy \\
&= 1 - \frac{\mu}{\lambda + \mu} \int_0^\infty (\lambda + \mu) e^{-(\lambda+\mu)y} dy \qquad (1) \\
&= 1 - \frac{\mu}{\lambda + \mu} \\
&= \frac{\lambda}{\lambda + \mu}
\end{aligned}$$

□

Numériquement :

-On simule X un vecteur de réalisations $X \sim Exp(\lambda)$ et Y un vecteur de réalisation de $Y \sim Exp(\mu)$. Ces deux vecteurs sont de même taille n avec n grand.
-On regarde ensuite la proportion de fois où les réalisations de X ont été plus petites que celles de Y.

Quelques exemples d'exécution du code A.1.2 de l'annexe :

```
probainf(12,15)
paramètres lambd= 12 et mu= 15
vraie proba : 0.4444444444444444
simulation :  0.4448
erreur relative : 0.00079999998

probainf(10, 20)
paramètres lambd= 10 et mu= 20
vraie proba : 0.3333333333333333
simulation :  0.3329
```

```
erreur relative : 0.0013
```

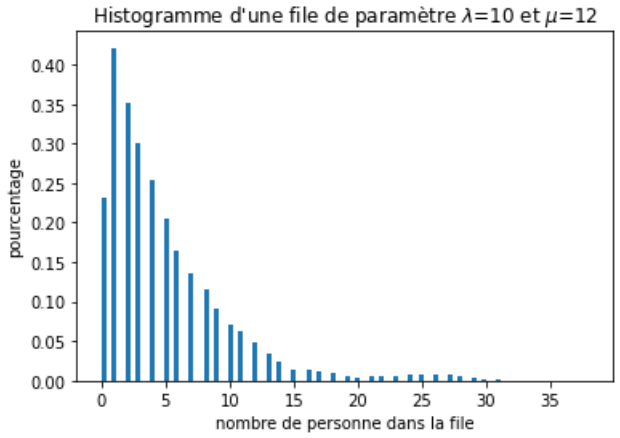
3.4 Simulation de la file

On connait désormais les probabilités de transition ainsi que la distribution des temps d'attente. On peut alors simuler la file numériquement.

Principe de la simulation : A.1.3

-On se fixe un temps auquel on arrête la chaine, T
-Au temps t=0, on simule une réalisation de $X \sim E(\lambda)$
-Tant que t < T, on vérifie si l'état de la chaîne est 0.
-Si on est en 0, le temps d'attente est une réalisation de $X \sim E(\lambda)$ et on fait +1. Sinon, c'est une réalisation de $X \sim E(\lambda + \mu)$ et la chaîne fait +1 avec probabilité $\frac{\lambda}{\lambda+\mu}$ et -1 avec probabilité $\frac{\mu}{\lambda+\mu}$.
-t=t+temps d'attente

Pour une chaine de paramètres $\lambda = 10,\ \mu = 12$ une réalisation de la chaine nous donnerait l'histogramme du nombre de personne dans la file suivant :



3.5 Stabilité du système

Il paraît évident que si le nombre d'arrivées par unité de temps est plus grand que le nombre de départs par unité de temps, i.e $\lambda > \mu$, le système ne sera pas stable.

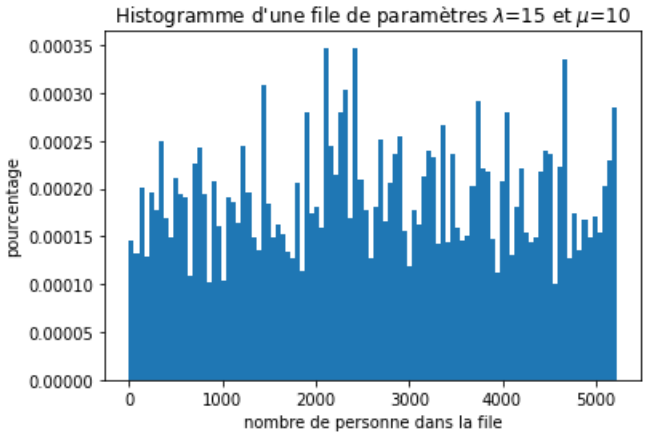
Supposons le système stable et cherchons les probabilités invariantes.

Il faut résoudre : $\pi Q = 0$, avec $Q = \begin{pmatrix} -\lambda & \lambda & & \\ \mu & -(\lambda+\mu) & \lambda & \\ & \mu & -(\lambda+\mu) & \lambda \\ & & \ddots & \end{pmatrix}$ la matrice d'intensité de la chaîne de markov.

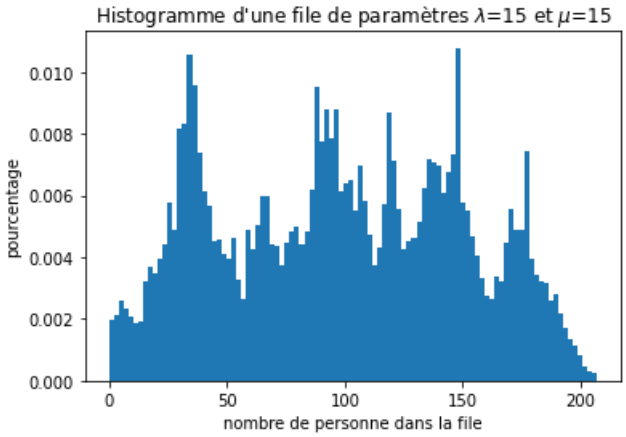
Par récurrence on obtient : $\pi(i) = (\frac{\lambda}{\mu})^i \pi(0)$

On doit de plus avoir $\sum_{i=0}^{+\infty} \pi(i) = \sum_{i=0}^{+\infty} (\frac{\lambda}{\mu})^i \pi(0) = 1$. On remarque alors que pour que cette somme soit finie, il faut que $\frac{\lambda}{\mu} < 1$, et donc que $\lambda < \mu$.

Exemple d'un histogramme pour une réalisation avec $\lambda = 15,\ \mu = 10$:



Exemple d'un histogramme pour une réalisation avec $\lambda = 15,\ \mu = 15$:



On suppose maintenant que $\lambda < \mu$,

$$\begin{aligned} \sum_{i=0}^{+\infty} \pi(i) = \sum_{i=0}^{+\infty} (\frac{\lambda}{\mu})^i \pi(0) = 1 \\ \Leftrightarrow \frac{\pi(0)}{1 - \frac{\lambda}{\mu}} = 1 \\ \Leftrightarrow \pi(0) = 1 - \frac{\lambda}{\mu} \end{aligned} \tag{1}$$

Il vient alors : $\pi(i) = (\frac{\lambda}{\mu})^i (1 - \frac{\lambda}{\mu})$

Numériquement :

A la différence des chaînes de Markov discrètes, il ne suffit pas de tracer l'histogramme des états ayant été atteints.
En effet, en prenant $\lambda = 10,\ \mu = 12$ et l'histogramme vu précédemment en 3.4 on voit que cela ne correspond pas, pour les 10 premières valeurs, au vecteur :
(Annexe A.1.4)

```
[0.16666667 0.13888889 0.11574074 0.09645062 0.08037551 0.0669796
 0.05581633 0.04651361 0.03876134 0.03230112]
```

Pour obtenir celui-ci dans notre cas, il faut regarder la proportion de temps passé dans chaque état. On obtiendrait ainsi le vecteur:
(Annexe A.1.5)

```
[0.16992577 0.1464145  0.12447591 0.10342542 0.08797772 0.0688955
 0.05795352 0.04687089 0.04137533 0.03099448]
```

3.6 Nombre de personnes moyen dans la file

On introduit L le nombre de personnes moyen dans la file.

$$\begin{aligned} \mathrm{L} &= \mathbb{E}[\pi] \\ &= \sum_{i=0}^{+\infty} \mathrm{i}\pi(i) \\ &= \sum_{i=0}^{+\infty} \mathrm{i}(\frac{\lambda}{\mu})^i \pi(0) \\ &= \frac{\frac{\lambda}{\mu}}{1-\frac{\lambda}{\mu}} \end{aligned} \tag{1}$$

Numériquement (Annexe A.1.6) :

-Dans la simulation de la file, on pose s=0 et a chaque incrémentation on ajoute le temps d'attente que l'on multiplie par l'etat actuel de la chaine (avant l'incrémentation).
-On simule une file
-On extrait s
-On divise par le temps total passé pour obtenir la moyenne des temps passés dans chaque état pondéré par les états
-On répète la procédure plusieurs fois pour une file de mêmes paramètres pour avoir une moyenne des nombres de personnes moyens.

```
Nombre moyen de personne dans la file :
simulation: 5.0397018841154955
théorique : 5.000000000000002
erreur relative : 0.00794037682
```

3.7 Temps d'attente moyen dans la file

La loi de Little est le résultat qui lie le temps moyen qu'un client passe dans la file, ici le temps d'attente moyen noté W, le nombre moyen de clients dans la file L introduit dans la partie 3.6 et le débit de la file d'attente λ.

$$W = \frac{L}{\lambda} \tag{1}$$

Numériquement (Annexe A.1.7):

Note: Le code utilisé ici a un temps d'éxécution assez long, on ne pourra donc pas faire de moyenne des temps moyens.

-On simule une chaine.
-On extrait les temps auquel on a eu un service, les temps d'arrivées et le nombre de personne dans la file au moment d'une arrivée.
-On calcule ensuite la durée d'un service: S'il n'yavait personne quand le i-ème client arrive, la durée de service est égale au i-ème temps de service - le i-ème temps d'arrivée.
Sinon il suffit de regarder combien de temps il s'est écoulé entre le service i et le service $i+1$.
-Puis, pour chaque personne dans la file, on vérifie une première fois si le temps d'arrivée i est plus petit que tous les temps de services enregistrés puis si on a pu enregistrer le temps de service i avant l'arret de la simulation. Si ce n'est pas le cas on s'arrête.
Sinon, on fait la somme des k-1 durées de service correspondants, avec k le nombre de personne dans la file.

```
Temps moyen passé dans la file:
simulation: 0.5058573965371242
théorique : 0.5
erreur relative : 0.01171479307
```

A Annexes

A.1 Code Python

A.1.1 Preuve que le minimum entre le temps d'attente d'une arrivée et d'un départ suit une loi exponentielle de paramètre $\lambda + \mu$:

```
def densite(l,x):
    return l*np.exp(-l*x)

def histogramme(l, mu):
    n=expon.rvs(scale=1/l, size=10000)
    m=expon.rvs(scale=1/mu, size=10000)
    k=l+mu

    vect=np.minimum(n,m)

    plt.title("histogramme des mins entre Exp(%i) et Exp(%i)" %(l,mu))
    plt.ylabel('pourcentage')
    plt.xlabel('mins')
    plt.hist(vect,bins=100, density=True)
    absc=np.arange(0,0.4,0.0001)
    ordo=densite(k,absc)
    plt.plot(absc,ordo,color='r',label='densité Exp(%i )' % k )
    plt.legend(bbox_to_anchor=(0.5,0.5), loc="center left", borderaxespad=0)
    plt.show()

histogramme(10,12)
```

A.1.2 Preuve que $\mathbb{P}(X < Y) = \frac{\lambda}{(\lambda+\mu)}$:

$\mathbb{P}(X < Y) = \frac{\lambda}{(\lambda+\mu)}$ où X et Y sont des VA exp de paramètres λ et μ. Ce résultat nous est utile pour calculer les probabilités de transition.

```
def probainf(l1, l2):
    n=10000
    X=expon.rvs(scale=1/l1, size=n)
    Y=expon.rvs(scale=1/l2, size=n)
    sum=0
    for i in range(n):
        if X[i]<Y[i]:
            sum=sum+1
    print('paramètres lambd=',l1,'et mu=',l2)
    print('vraie proba :', l1/(l1+l2))
    print('simulation : ', sum/n)

probainf(12,15)
probainf(10, 20)
```

A.1.3 Simulation de la file d'attente:

```
def contMC(lambd, mu, T):  ##T=temps auquel on arrete la chaine
    n=np.zeros(1) ##initialisation de la chaine
    s=0


    t1 = expon.rvs(scale=1/lambd, size=1) ##premiere arrivée
    vect_temps=np.array(t1) ##vecteur des temps d'attentes
```

```
    vect_temps_arrivee=np.array(t1)
    vect_temps_service=[]
    vect_etat_arrivee=[1]
    etat=1
    t=t1
    nb_de_passage=1

    while (t<T) :
        n=np.append(n,etat)

        ##si on n'est pas en 0 le temps d'attente suit une loi exp (lambda+mu) et
        ↪  on a une certaine proba de faire +1 ou -1 ( lambda/(lambda+mu) et
        ↪  1-lambda/(lambda+mu) )
        if (etat>0) :
            ht=expon.rvs(scale=1/(lambd+mu),size=1)
            u=np.random.rand(1)
            if (u< lambd/(lambd+mu) ) :  ##si nombre aléatoire plus petit que la
            ↪  proba de faire +1, on fait +1
                etat=etat+1
                vect_etat_arrivee=np.append(vect_etat_arrivee,etat)
                vect_temps_arrivee=np.append(vect_temps_arrivee, t+ht)

            else :
                etat=etat-1
                vect_temps_service=np.append(vect_temps_service,t+ht)


       ##sinon le T.A ~EXP(lambda) et la proba d'aller en 1 est 1
        else :
            ht=expon.rvs(scale=1/lambd, size=1)
            etat=etat+1
            vect_etat_arrivee=np.append(vect_etat_arrivee,etat)
            vect_temps_arrivee=np.append(vect_temps_arrivee, t+ht)

        s=s+ht*n[nb_de_passage] ## somme temps passé pondérée par l'état
        vect_temps=np.append(vect_temps,ht)
        t=t+ht
        nb_de_passage=nb_de_passage + 1

    return [n, vect_temps, t, s, vect_temps_arrivee, vect_temps_service,
    ↪  vect_etat_arrivee]

##réalisation d'une chaine:

MC=contMC(10,12,1000)  #arrivée 10/minutes, départ 12/minutes, on s'arrete à 1000
↪  minutes

M=MC[0] ##chaine des états
plt.title("Histogramme d'une file de paramètre $\lambda$=10 et $\mu$=12")
plt.ylabel("pourcentage")
plt.xlabel("nombre de personne dans la file")
plt.hist(M,bins=100,density=True)
```

A.1.4 Valeur théorique des probabilités invariantes:

```
def proba_inv_reelles(m,l,mu): ##m représente la valeur max à laquelle s'arreter
    vect=np.zeros(m+1)
    for i in range(m+1):
        vect[i]=(1-l/mu)*( (l/mu)**i )
```

```
    return vect
```

A.1.5 Simulation du vecteur de probabilité invariante:

```
##les états supérieurs au max sont considérés comme étant nuls
def vect_proba_inv(MC): ##MC est une réalisation de contMC
    M=MC[0]
    T=MC[1]
    t=MC[2]
    N=np.max(M).astype(int)
    vect=np.zeros(N)
    for i in range(N) :
        vect[i]=np.sum(T[np.where(M==i)])/t
    return vect
```

A.1.6 Simulation du nombre de personne moyen dans la file n fois:

```
def multipleMC(n,l,mu):
    vect_moyennep=np.zeros(n)
    for i in np.arange(n) :
        MC=contMC(l,mu,1000)
        t=MC[2] ##temps passé dans la chaine (file)
        s=MC[3] ##somme pondérée des temps d'attente
        L=s/t   ##nombre moyen de personne dans la file : moyenne des temps
        ↪  passés dans chaque état pondéré par les états.
        vect_moyennep[i]=L
    return np.mean(vect_moyennep)

moyenne=multipleMC(20,10,12)
L=moyenne
print("Nombre moyen de personne dans la file :\nsimulation:", L, "\nthéorique :",
↪  (10/12)/(1-10/12))
```

A.1.7 Simulation du temps passé dans la file:

```
def temps_file(MC):    ##MC est une chaine de markov/file d'attente
    arrivees=MC[4] ##vecteur des temps auquel on a eu une arrivée
    services=MC[5] ##vecteur des temps auquel on a eu un service
    etat=MC[6]  ## vecteur de la taille de la file a chaque arrivée
    duree=np.zeros(len(services)-1) ##durée des services

    for i in np.arange(len(services)-1):
        if (i in np.where(etat==1)[0])==True :
            duree[i]=services[i]-arrivees[i]
        else :
            duree[i]=services[i+1]-services[i]

    vectservice=[]
    for i in np.arange(len(etat)):
        if all(j < arrivees[i] for j in services)==True:
            break
        else:
            s=0

            for k in np.arange(etat[i]):
                if (i<len(duree)):
```

```
                    s=s+duree[i-k]
                else :
                    break

            vectservice=np.append(vectservice,s)

    return np.mean(vectservice)



MC=contMC(10,12,2000)
W=temps_file(MC)
print("Temps moyen passé dans la file:\nsimulation:", W, "\nthéorique :",
↪  1/(12-10))
```