Gestion de flux dans le réseau
TD n ° 6
Modélisation mathématique
Q4

Sibylle Roux Juliette Arazo Nicolas Le Gallo
Tanguy Thomas

13 décembre 2017

Table des matières

1 Etude de la file M/M/1

1.1 Conception d'une représentation informatique

Pour stocké les valeurs simulés on utilisera un tableau de la forme :

instant t	$q(t)$	incrément

Cette représentation est idéale : elle réunit toutes les informations utiles du fonctionnement des serveurs :

- l'instant où se passe l'evenement
- le type d'evenement (entrée ou sortie d'un client dans le système)
- le nombres de clients présent dans le système

Cette dernière valeur nous sera utile pour avoir la taille de la fille d'attente :

$$taille\_file = max(q(t) - 1, 0) \tag{1}$$

1.2 Conception et développement d'un algorithme de simulation en scilab

Nous avons à notre disposition une fonction SciLab $insere(q, ta, ts)$ avec :

- **q** : Matrice de notre représentation informatique de la file
- **ta** : Temps actuel
- **ts** : Temps de service

```
function newq = insere(q, ta, ts)
    if q($, 1) < ta then // aucune requête dans le système
        q($+1,:) = [ta, 1, 1]; // ajout de la requête en fin de liste
    else // inscription de la requête en file d'attente
        ind = sum(q(:, 1) < ta); // recherche du point d'insertion
        q(ind+2:$+1, :) = q(ind+1:$, :); // création d'un trou pour insérer
        q(ind+1,:) = [ta, q(ind,2), 1]; // insertion au bon endroit
        q(ind+1:$,2) = q(ind+1:$,2) + 1; // correction de la taille de
                                         // la file pour les lignes suivantes
    end
    // Inscription du départ de la requête après son temps de
    // service ts
    s = q($, 1) + ts // calcul de la date de sortie
    q($+1, :) = [s, q($, 2) - 1, -1]; // inscription de la date de sortie à la fin
    newq = q // On renvoie la file ainsi modifiée
endfunction
```

Nous avons aussi la fonction $randExp(n, lambda)$ qui genere un vecteur de taille n de valeurs aléatoires suivant la loi exponentielle de paramètre $\lambda = lambda$

```
function t = randExp(n, lambda)
    t = -log(1 - rand(n,1)) / lambda
endfunction
```

C'est à l'aide de ces deux fonctions citées plus haut que l'on peut définir la fonction : $queue(Tmax, lambda, mu)$ où :

- **Tmax** : Instant maximal de la représentation de la file
- **lambda** : λ correspondant aux temps inter-arrivées
- **mu** : λ correspondant aux temps de service

```
function a=queue(Tmax, lambda, mu)
    Q = [0, 0, 0]; // Initialisation de la file
    t = 0; // temps courant
    while (t < Tmax)
        t_ia=randExp(1,lambda); // tirage de l'intervalle inter-arrivée
        t=t+t_ia;// mise à jour de l'instannt courant :
                 // instant d'arrivée de la prochaine requête
        ts=randExp(1,mu); // tirage de son temps de service
        Q=insere(Q,t,ts);// insertion dans la file des évènements
    end
    a = Q (Q(:,1)<Tmax,:) // on renvoie l'état de la file à la date Tmax
endfunction
```

1.3 Simulation de trajectoires

1.3.1 Simulation de l'évolution d'un file d'attente

Pour simuler les différentes trajectoires, on va utiliser une fonction Scilab $traj(n, lambda, mu)$ où :

- **n** : correspondant à l'instant maximal de la représentation de la file
- **lambda** : λ correspondant aux temps inter-arrivées
- **mu** : λ correspondant aux temps de service

```
function traj(n,lambda,mu)
    for i=1:50 // 50 trajectoires
        Q = queue(n, lambda, mu); // lambda est 5 fois plus grand que mu
        plot2d2(Q(:,1), max(Q(:,2) - 1, 0), style=2) // trace la courbe
    end
endfunction
```

1.3.2 Distribution statistiques de la taille de la file d'attente

Pour avoir la distribution statistique de des trajectoires, on va utiliser une fonction Scilab $distrib(n, lambda, mu)$ où :

- **n** : correspondant à l'instant maximal de la représentation de la file
- **lambda** : λ correspondant aux temps inter-arrivées
- **mu** : λ correspondant aux temps de service

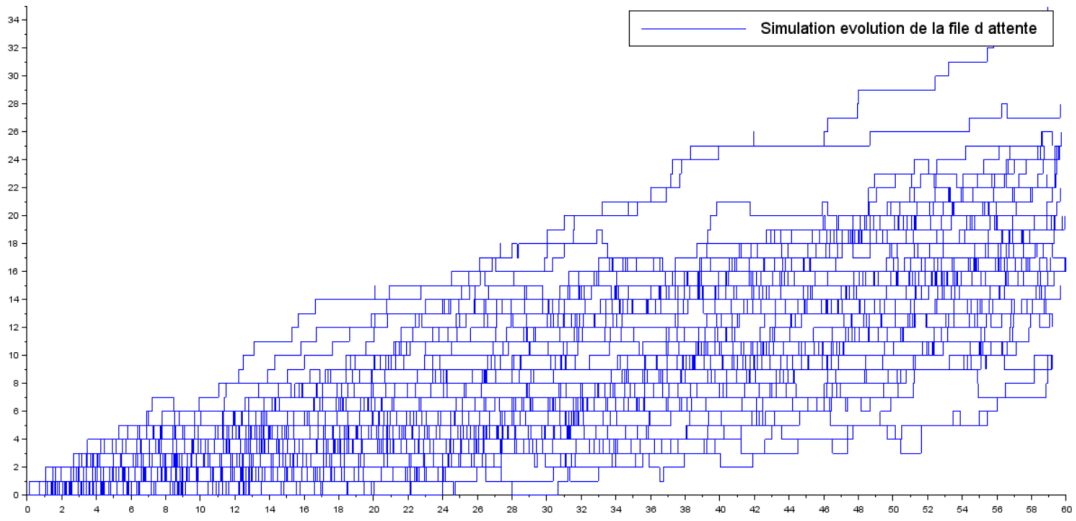
```
function Qi=distrib(n,lambda,mu)
    Qi = zeros(500, 1);
    for i=1:500
        Q = queue(n, lambda, mu);
        Qi(i) = Q($, 2);
    end
    distr = tabul(Qi,'i')
    bar(distr(:,1),distr(:,2)/500)
    legend("Distribution de Q"+string(n))
endfunction
```

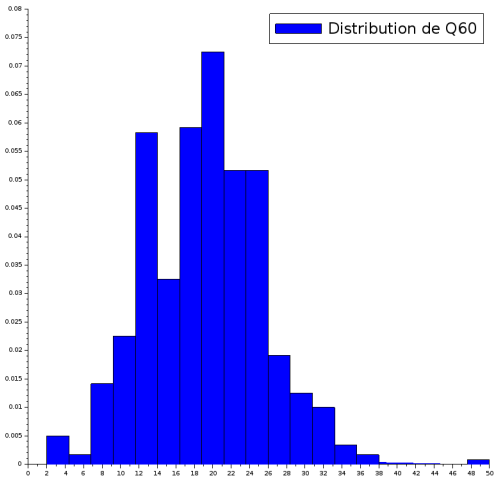
Lorsque l'amplitude sera très grande, on optera plutôt pour une autre version de cette fonction $distribv2(n, lambda, mu)$:

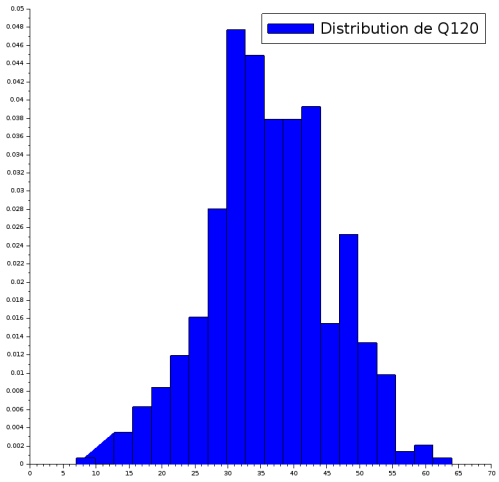
```
function Qi=distribv2(n,lambda,mu)
    Qi = zeros(500, 1);
    for i=1:500
        Q = queue(n, lambda, mu);
        Qi(i) = Q($, 2);
    end
    histplot(20,Qi)
    legend("Distribution de Q"+string(n))
endfunction
```

1.3.3 Temps de service supérieur en moyenne aux temps inter-arrivées

Paramètre : $n = 60$; $lambda = 0.5$; $mu = 0.2$



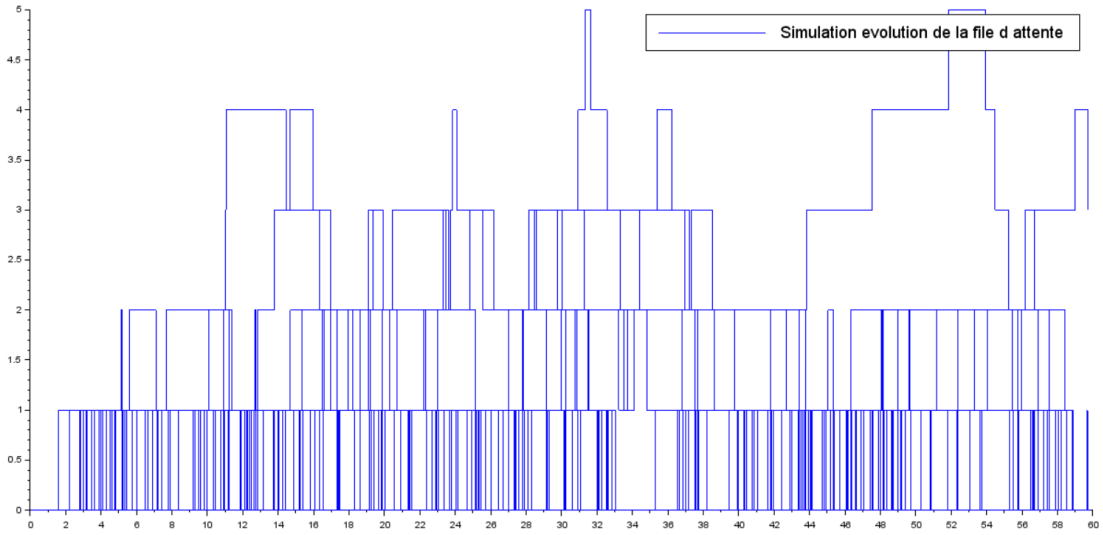


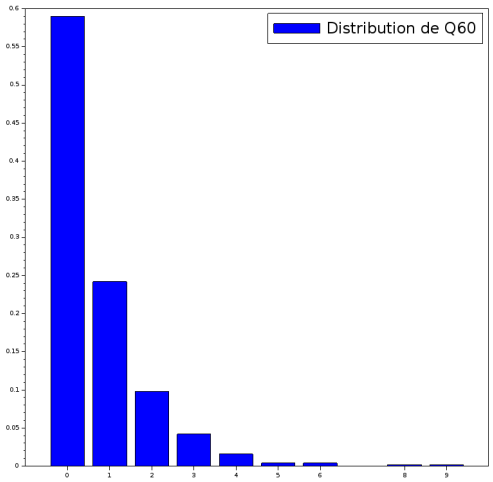


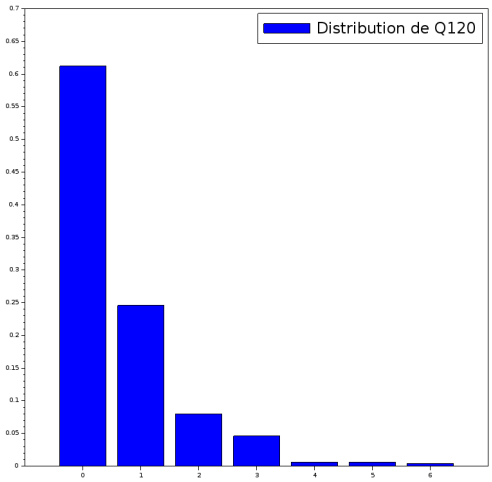
	Q60	Q120
mean()	18.54	36.806
variance()	40.212826	81.158681

1.3.4 Temps de service inférieur en moyenne aux temps inter-arrivées

Paramètre : $n = 60$; $lambda = 0.2$; $mu = 0.5$



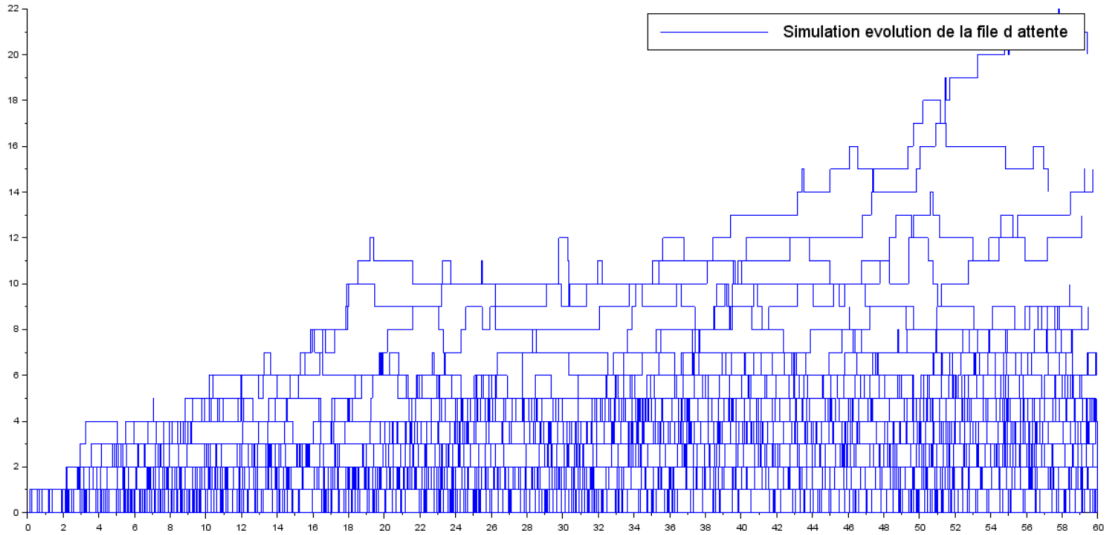


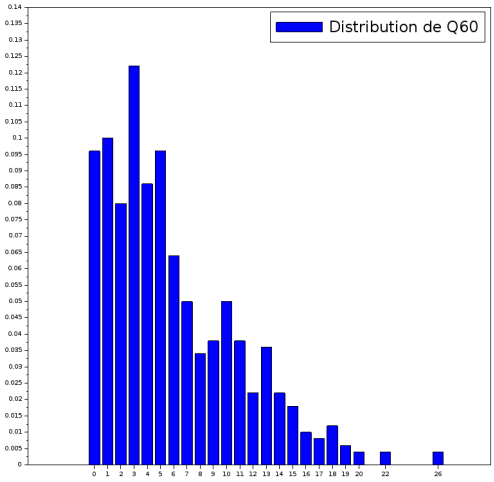


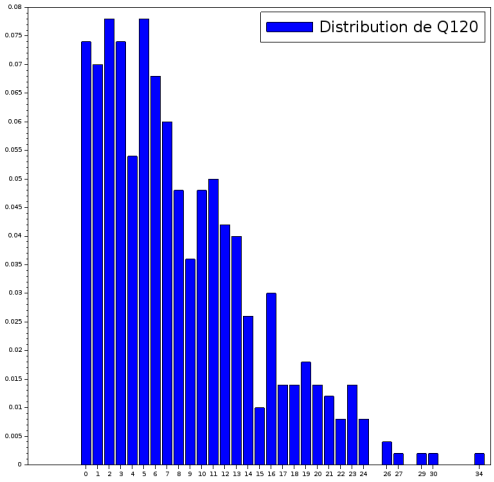
	Q60	Q120
mean()	0.582	0.704
variance()	0.9611984	1.1787415

1.3.5 Temps de service égaux en moyenne aux temps inter-arrivées

Paramètre : $n = 60$ **;** $lambda = 0.5$ **;** $mu = 0.5$







	Q60	Q120
mean()	5.726	8.476
variance()	24.792509	46.350124

2 Etude de la file à 3 serveurs

2.1 Simulation de stratégie circulaire

Pour simuler la stratégie circulaire, on va utiliser la fonction $circul(Tmax, lambda, mu)$ où :

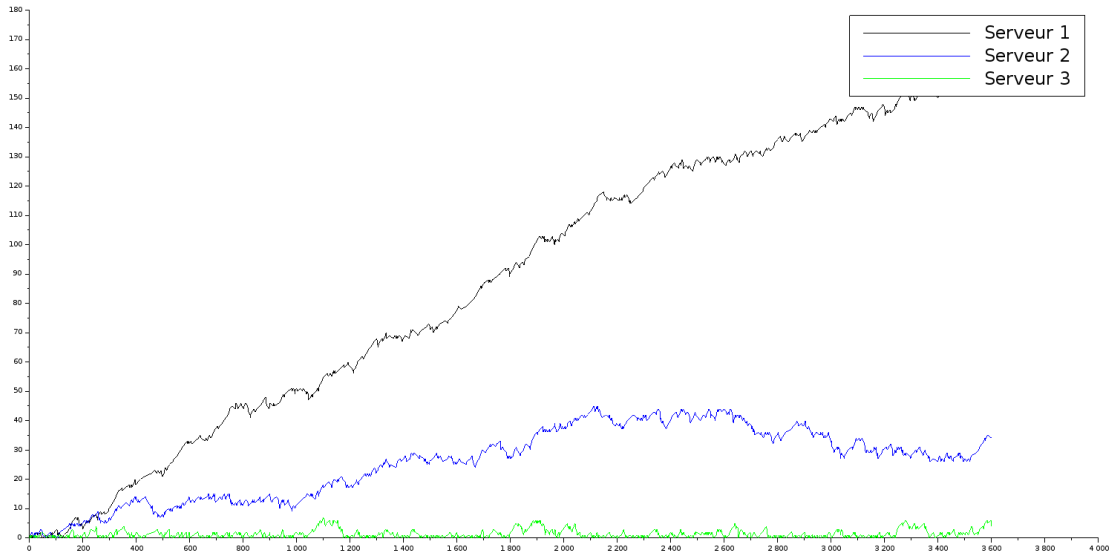
- **Tmax** : Durée en seconde de la simulation
- **lambda** : λ correspondant aux temps inter-arrivées
- **mu** : vecteur contenant les λ correspondant aux temps de service des 3 serveurs

```
function [Q1, Q2, Q3] = circul(Tmax, lambda, mu)
    Q1 = [0, 0, 0]; Q2 = Q1; Q3 = Q1;
    i = 0;
    ta = 0;
    while (ta < Tmax)
        ia = randExp(1, lambda)
        i = i+1
        ta = ta + ia
        nq = modulo(i, 3) + 1
        ts = randExp(1, mu(nq))
        select nq
        case 1.
            Q1 = insere(Q1, ta, ts)
        case 2
```

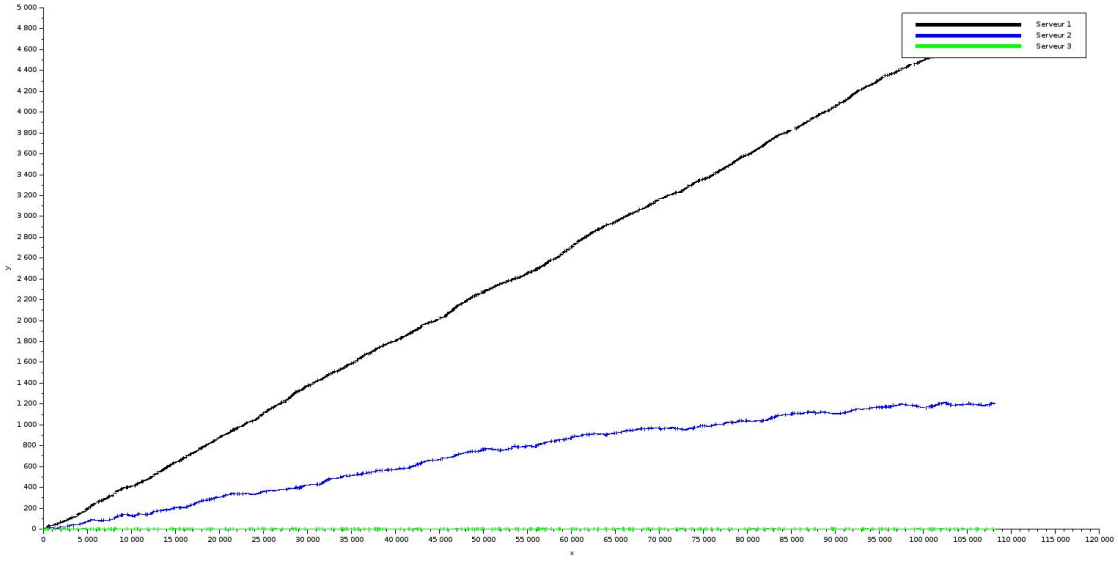
```
            Q2 = insere(Q2, ta, ts)
        else
            Q3 = insere(Q3, ta, ts)
        end
    end
    Q1 = Q1(Q1(:,1)<Tmax,:)
    Q2 = Q2(Q2(:,1)<Tmax,:)
    Q3 = Q3(Q3(:,1)<Tmax,:)
endfunction
```

Dans cette fonction, on utilise 2 autres fonctions :

- $insere(q, ta, ts)$ où :
  - **q** : File d'attente
  - **ta** : Instant d'arrivée de la requête
  - **ts** : Temps de service
- $randExp(n, lambda)$
  - **n** : taille du vecteur en sortie
  - **lambda** : λ avec lequel sera générée la valeur aléatoire qui suit une loi exponentielle



Simulation sur 1 heure



Simulation sur 30 heures

2.1.1 Etude numérique du temps de traversée du système pour une requête

2.1.2 Etude numérique du nombre de requêtes dans le système

2.1.3 Recherche d'un régime stationnaire

2.2 Simulation de la stratégie d'affection aléatoire proportionnelle

2.2.1 Simulation

2.2.2 Etude numérique

2.3 Autres stratégies, aléatoires ou/et détérministes

3 Conclusion

A Fonctions

A.1 insere(q,ta,ts)

```
function newq = insere(q, ta, ts)
    if q($, 1) < ta then
        q($+1,:) = [ta, 1, 1];
    else
        ind = sum(q(:, 1) < ta);
        q(ind+2:$+1, :) = q(ind+1:$, :);
        q(ind+1,:) = [ta, q(ind,2), 1];
        q(ind+1:$,2) = q(ind+1:$,2) + 1;
    end
    s = q($, 1) + ts
    q($+1, :) = [s, q($, 2) - 1, -1];
    newq = q
endfunction
```

A.2 randExp(n,lambda)

```
function t = randExp(n, lambda)
    t = -log(1 - rand(n,1)) / lambda
endfunction
```

A.3