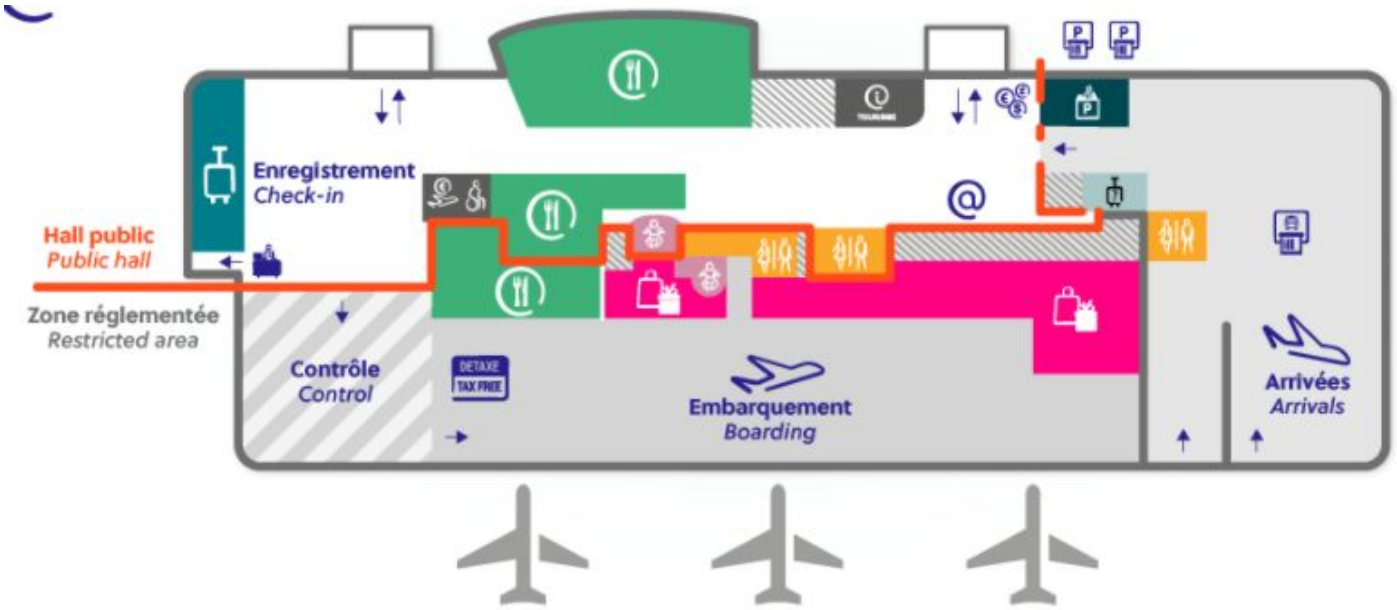
Simulation

Gestion Parking Taxi Aéroport



Version n°3.3

Tables des matières

Introduction

Dans le cadre de notre cursus à 3iL, nous avons suivi le cours de Modélisation & Analyse des Systèmes. Ce cours nous a permis d'utiliser et exploiter les méthodes à dispositions pour permettre de simuler des phénomènes d'attentes afin de gérer la productivité d'une tâche.

Nous avons comme sujet la gestion du dimensionnement des aires d'attentes des taxis qui desservent un aéroport pour qu'il y ait le moins d'attente pour les taxis et pour les usagers. Nous allons, dans ce rapport, montrer l'apport des simulations de phénomènes d'attentes pour répondre à cette problématique.

Nous allons décrire le logiciel qui permet de simuler des cas réel par le biais de paramètres. Ces simulation doivent nous permettre de valider nos calculs théoriques. Nous étudions ici les phénomènes d'attentes, pour l'attente des taxis, et les phénomènes d'arrivées pour les passagers.

Notre application à été codé par le biais du langage PHP, un langage orienté Web, associé à du Javascript pour la partie calcul. Ces langages permettent aussi de mettre en place la simulation et les outils dont nous avons besoin pour celle-ci, c'est à dire les différents générateurs de lois de probabilités, les différents tests à mettre en place mais aussi les différents processus de simulation d'événements.

Ces différents outils seront décrits dans la partie "TP1" et "TP2" de ce rapport.

Nous présenterons, pour commencer, le sujet qui nous a été posé. Par la suite nous détaillerons l'implémentation des outils puis nous expliquerons les lois que nous avons utilisé pour traiter ce phénomène. Pour finir, nous appliquerons la théorie à des données réel par le biais du simulateur et nous testerons de la validité de nos résultat.

La théorie

Sujet choisi :

Notre sujet consiste à répondre à une problématique, comment dimensionner l'aire d'attente des taxis d'un aéroport.

Notre sujet peut traiter de deux sujet différents

→ Arrivée des avions : Il doit y avoir assez de taxis pour répondre à la demande

→ Départ des avions : les dépôts des usagers par les taxis doit répondre à la demande

Dans notre cas, nous avons choisi de traiter le premier problème, c'est à dire l'arrivée des avions et donc le nombre de taxis nécessaire pour répondre à la demande et donc l'aire minimale pour répondre à la problématique.

Fonctionnement du système "Parking Aéroport"

Nous avons plusieurs informations pour répondre à la problématique

→ L'arrivée des taxis qui viennent prendre des passagers de l'aéroport, suit une loi de Poisson de cadence égale à 2,7 taxis par minute.

→ Une place de stationnement, dite plate-forme de chargement, est réservée à la porte des « arrivées » de l'aéroport pour permettent aux taxis de s'arrêter afin de charger les passagers.

→ Le temps total de mise en place et de chargement jusqu'au démarrage, suit une loi exponentielle de moyenne 20 secondes.

→ Il y a toujours des passagers pour prendre un taxi, à ces heures là (pas de temps mort).

Nous avons donc deux lois que nous utiliserons :

$$\rightarrow\ T \sim \rho\,(\lambda)\ avec\ \lambda = 2.7\ mm^{-1}$$

$$\rightarrow\ C \sim \varepsilon\,(\mu)\ avec\ \mu = 20\ secondes\ soit\ 3\ mm^{-1}$$

Lois de traitement

Loi des arrivés

La loi des arrivées correspond ici à l'arrivée des taxis pour prendre en charge les usagers, elle suit un loi de poisson de cadence $\lambda = 2{,}7\ mm^{-1}$. Cette cadence correspond à une moyenne de taxi arrivant toutes les minutes.

Loi des services

La loi des services correspond ici à la prise en charge d'un usager par un chauffeur de taxi. Celle-ci suit une loi exponentielle de moyenne $\mu = 20$ secondes.

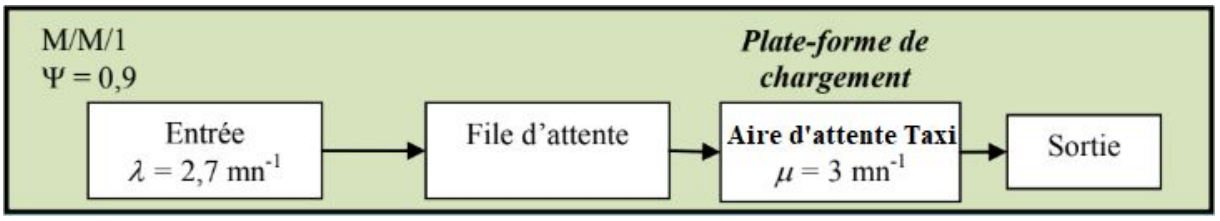
Type de système

Le système ne dispose que d'une plateforme de chargement.

C'est un système M / M /1 , les paramètres μ & λ de la loi des arrivées et de la loi des services, nous déduirons les différentes moyennes, statistiques et temps d'attentes qui nous permettront de comprendre le fonctionnement de ce système

On peut noter qu'on peut aussi calculer, à l'aide des paramètres des lois, le facteur de charge du système, ce facteur Ψ doit être inférieur à 1 pour que le système soit pérenne.

Schéma du modèle d'attente et de classification au sens de Kendall :



Exemple

En analysant cette exercice, on utilise les formules correspondant au système M/M/1, voici les différents résultat :

Combien de temps, en moyenne, attendront les taxis avant de se mettre en place pour prendre un passager ?

$$E(TAF) = \Psi^2/\lambda(1-\Psi) = \frac{0{,}9^2}{2{,}7 \times 0{,}1} = 3 \text{ minutes en moyenne}$$

Combien de temps, en moyenne, passeront les taxis entre le moment où ils arrivent à l'aéroport et celui où ils quittent la plate-forme de chargement ?

$$E(TAS) = \Psi/\lambda(1-\Psi) = E(TAF)/\Psi = \frac{3}{0{,}9} = 3{,}33 \text{ mn soit encore 3 mn 20 s en moyenne.}$$

Donner le nombre moyen de taxis en attente dans la file.

Formule de Little : $E(N_x) = \lambda E(W_x)$ soit $E(NbAF) = 3 \times 2{,}7 = 8{,}1$ taxis en moyenne.

Combien de places de stationnement d'attente doit prévoir l'aéroport pour que 99 % des taxis qui arrivent puissent rentrer immédiatement dans l'aire d'attente des taxis ?
Si on appelle n le nombre de place de stationnement nécessaire, il faut que soit inférieure à 1 % la probabilité que le nombre de taxis en attente soit supérieur à n.

$$\text{Or } \Pr(N \le n) = 1 - \Psi^{n+1} \Rightarrow \Pr(N > n) = \Psi^{n+1} < 1\%$$

$$\text{Soit ici } 0{,}9^n < 1\% \Leftrightarrow (n+1)\ln 0{,}9 < \ln 0{,}01 \Rightarrow n+1 \ge \frac{4{,}605}{0{,}105} = 43{,}7.$$

Donc il faut prévoir au moins 43 places de stationnement.

Est-ce que cette aire entrera dans le terrain dont dispose actuellement l'aéroport, un rectangle de 50 m par 20 m, sachant qu'on peut mettre un taxi sur une surface de 4 m (longueur) par 3 m (largeur), en files parallèles adjacentes (les zones pour virer d'une file à l'autre sont en plus de la taille ci-dessus du terrain) ?

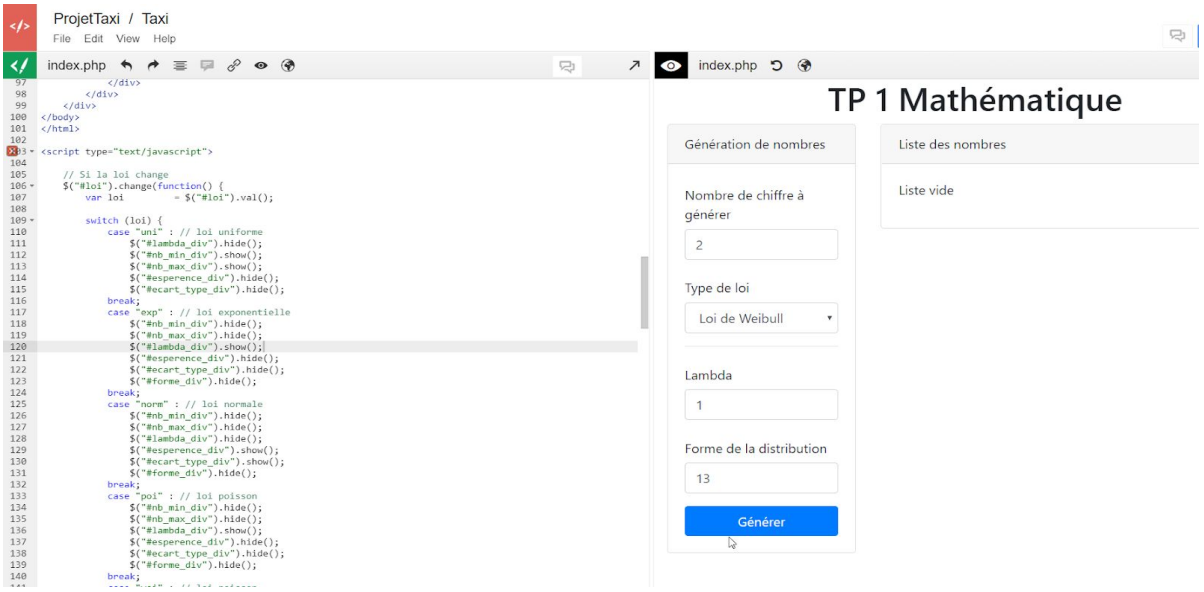
Sur 50 m on peut placer 12 voitures et sur 20 m on peut matérialiser 6 files soit au total 72 véhicules en stationnement. Le terrain suffit largement.

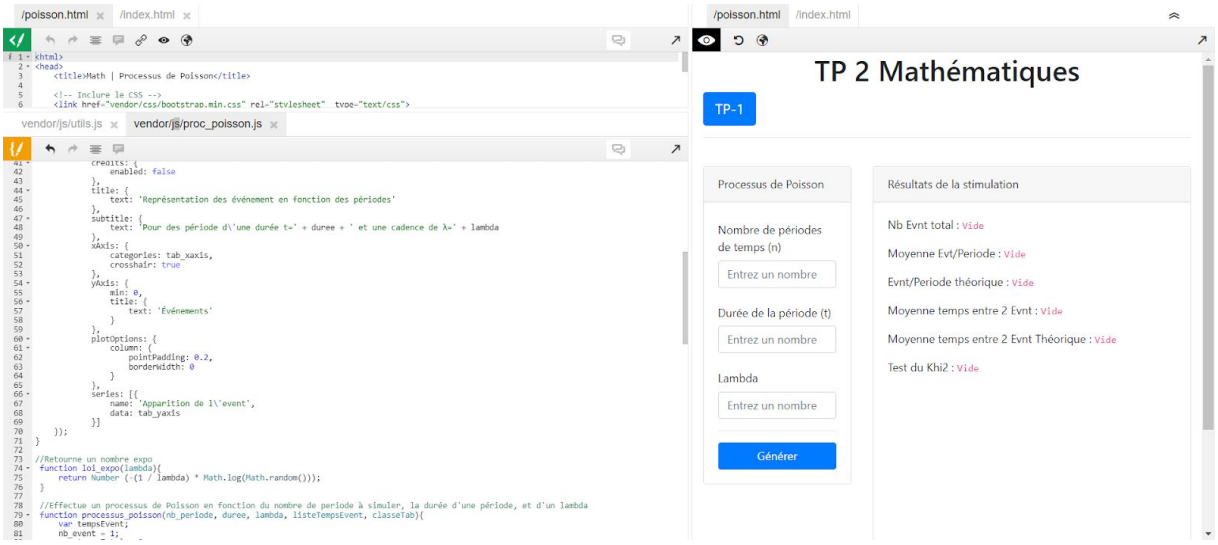
La simulation

Technologie et outils

Comme dit dans l'introduction de ce rapport, nous avons choisis les technologies PHP et Javascript pour effectuer ce projet.

Nous avons travaillé grâce aux nouveaux outils collaboratifs proposées par google.





TP - 1 Générateur des lois

Cette partie traite de la partie TP-1 de ce projet, elle correspond au développement des différents outils utilisés pour répondre à la problématique générale de ce projet.

L'objectif de cette partie et de réaliser plusieurs fonctions de génération de nombres aléatoires (pseudo-aléatoires). La particularité de ces fonctions est qu'elles suivent différentes lois. Nous testerons ici la loi Uniforme, la loi de Poisson, la loi Normale et pour finir la loi exponentielle. Pour tester la validité de nos données, nous allons appliquer sur ces dernières des tests du Khi2. Ce test est le même pour toutes les lois.

Nous commencerons par expliquer de quelle manière on testera les données puis nous détaillerons l'implémentation de chaques loi.

Le code pourra être trouvé dans la partie intitulé "CODE"

Test du Khi2

Le test du Khi2 est un test statistique permettant de contrôler l'adéquation d'une série de données à nos lois de probabilités. C'est un test d'adéquation.

Ce test nous permet de vérifier si un échantillon d'une variable aléatoire donne des résultat comparable à à ceux d'une loi de probabilité.

On supposera toujours **l'hypothèse h_0**, hypothèse nulle : **La variable aléatoire Y suit la loi de probabilité P**. Nous choisissons un risque ρ de 5%

Nous allons donc appliquer la formule du Khi2 à à nos échantillons pour comparer la distribution réelle à la distribution théorique (nous appellerons la valeur trouvée D) :

$$D = \sum_{i}^{n} \frac{(Effectif\ Obs(i) - Effectif\ théorique(i))^2}{Effectif\ théorique(i)}$$

Plus D sera proche de 0 plus l'adéquation est bonne.

Code Khi ² (voir code)

Loi Uniforme

On pose V suit une loi uniforme entre a et b (définir par l'utilisateur). Si a et b sont égale à 0 et 1, on arrive sur le comportement de la fonction "random" mis à disposition par la plupart des langages.

Pour adapter cette fonction à une loi uniforme paramétrable, on utilise la formule

x = arrondi (random * (b - a) + a)

Ici la fonction "random" est la fonction random proposée par le système (génère un nombre aléatoire entre 0 et 1)..

x est alors une valeur de la loi uniforme V.

La probabilité d'obtenir chaque résultat est de $\frac{1}{b-a}$.

Voici le code correspondant :

```
/**
*   Générer un certain nombre de nombre alléatoires suivant la loi uniforme
*   Ici on se contentera de la formule Math.random de javascript
*/
function generer_nombre_uni(nb_number, nb_min, nb_max) {
    // Tableau contenant les numéros générés
    var nombre_tab = new Array();
    var min = nb_min;
    var max = nb_max;

    // Génération des nombres
    var i = 0;
    while (i < nb_number) {
        nombre_tab[i++] = Math.random() * (max - min) + min;
    }

    return nombre_tab;
}
```

Ci dessous le rendu sur l'application et le test du Khi2 à 10 - 1 = 9 Degré De Liberté pour 100 000 valeurs testées.

Loi exponentielle

La loi exponentielle est une loi continue qui définie la durée de vie d'un phénomène sans mémoire.

On considère un λ réel positif. On pose V suit une loi exponentielle de paramètre λ tel que

$$Pr(k) = \lambda e^{-\lambda k}$$

Pour la génération de nombre on utilisera l'algorithme :

```
/**
*   Générer un certain nombre de nombre alléatoires suivant la loi exponentielle
*/
function generer_nombre_exp(nb_number, lambda) {
    // Tableau contenant les numéros générés
    var nombre_tab = new Array();

    // Génération des nombres
    var i = 0;
    while (i < nb_number) {
        var x = Math.random();
        nombre_tab[i++] = -(1 / lambda) * Math.log(1-x);
    }

    return nombre_tab;
}
```

Ci dessous le rendu sur l'application et le test du Khi2 à 10 - 1 - 1 = 8 Degré De Liberté pour 100 000 valeurs testées.

Loi de Poisson

La loi de Poisson est une loi discrète qui décrit le comportement des occurrences d'un évènement se produisant dans un intervalle de temps défini. La loi de Poisson nécessite comme paramètre l'espérance de cette évènement indépendamment du temps écoulé. On appel ce paramètre λ.

$$Pr(k) = e^{-\lambda} \frac{\lambda^k}{\lambda!}$$

Pour la génération de nombre on utilisera l'algorithme :

```
/**
*   Générer un certain nombre de nombre alléatoires suivant la loi de poisson
*   source en annexe
*/
function generer_nombre_poi(nb_number, lambda) {
    // Tableau contenant les numéros générés
    var nombre_tab = new Array();

    function genPoisson(l) {
        var l = Math.exp(-lambda);
        var k = 0;
        var p = 1;
        do{
            k++;
            x = Math.random()
            p = p * x;
        }while(p > l);
        return k-1;
    }

    // Génération des nombres
    var i = 0;
    while (i < nb_number) {
        nombre_tab[i] = genPoisson(lambda);
        i++;
    }

    return nombre_tab;
}
```

Ci dessous le rendu sur l'application et le test du Khi2 à 10 - 1 - 1 = 8 Degré De Liberté pour 100 000 valeurs testées.

Loi Normale

La génération de nombres suivant une loi normale peut se faire via plusieurs méthode.

Nous avons choisi d'utiliser la méthode Box-Muller car très simple à mettre en oeuvre algorithmiquement.

Une loi normale est caractérisée par sa moyenne m et son écart-type θ.

⟶ On pose U et V, variables aléatoires uniformes sur]0,1]

⟶ On pose X et Y, variables indépendantes suivant une loi normale centré réduite

⟶ X = cos(2 π V) et Y = sin(2 π V)

Pour la génération de nombre on utilisera l'algorithme :

```
/**
*   Générer un certain nombre de nombre alléatoires suivant la loi normale
*   (Box Muller)
*/
function generer_nombre_norm(nb_number, esperance, ecart_type) {

    // Transformation de Box-Muller -> retourne un nombre gaussien suivant une loi centrée réduite
    function gauss_by_BM() {
        var u = 0, v = 0;
        while(u === 0) u = Math.random(); // Nombre différent de 0
        while(v === 0) v = Math.random();
        return Math.sqrt( -2.0 * Math.log( u ) ) * Math.cos( 2.0 * Math.PI * v );
    }

    // Tableau contenant les numéros générés
    var nombre_tab = new Array();

    // Génération des nombres
    var i = 0;
    while (i < nb_number) {
        var x = gauss_by_BM();
        nombre_tab[i++] = Number(esperance) + Number(x) * Number(ecart_type);
    }

    return nombre_tab;
}
```

Ci dessous le rendu sur l'application et le test du Khi2 à 10 - 1 - 2 = 7 Degré De Liberté pour 100 000 valeurs testées.

Nous constatons bien que notre outil permet de générer des échantillons de valeurs suivant des lois de probabilités par le biais de nombres pseudo-aléatoires. On peut l'affirmer par les différentes conclusions des tests du Khi2.

TP - 2 Processus de Poisson

L'objectif du TP-2 de cours de modélisation et analyse des systèmes consiste à modéliser par le biais d'une application, un processus de Poisson paramétrable.

Un processus de Poisson, se basant sur la loi du même nom, est un processus stochastique permettant de représenter l'apparition d'un événement pendant une période donnée, par exemple l'apparition des taxis au parking d'un aéroport. Cela permet donc de gérer son dimensionnement.

Notre IHM a été réalisée en PHP comme précédemment et pour les mêmes raisons. (Connaissance du code et facilité du travail collaboratif)

Processus de Poisson

Notre objectif pour simuler ce processus consiste donc à simuler des période de temps (t) et de simuler l'apparition d'un événement dans ce laps de temps à partir d'une fréquence d'apparition moyenne appelée λ.

Nous avons donc utilisé les différents générateurs de lois provenant du TP-1. Plus précisément, nous avons utilisés la loi de Poisson et la loi exponentielle sur laquelle elle se base.

Loi exponentielle

Nous avons re-utilisé l'expression suivante:

$$x = -\frac{1}{\lambda} \cdot \ln(random)$$

Et voici son algorithme associé:

```
//Retourne un nombre expo
 function loi_expo(lambda){
     return Number (-(1 / lambda) * Math.log(Math.random()));
 }
```

Loi de poisson

La loi de Poisson est une loi discrète qui décrit le comportement et le nombre d'événements se produisant dans un intervalle de temps connu. Elle se génère à partir de la loi exponentielle. (voir ci-dessus)

Notre simulateur doit pouvoir permettre de simuler un événement aléatoire dont l'occurrence sur une période dépend d'un paramètre lambda. Le tout sur plusieurs période

Nous avons donc plusieurs paramètre à prendre en compte :

→ Le nombre de période à simuler (nb_periode)

→ La durée t d'une période (duree)

→ Cadence λ (lambda)

on peut noter que la cadence correspond à la fréquence d'apparition moyenne de l'événement dans un intervalle t.

Voici l'algorithme que nous avons utilisé pour la simulation du processus de Poisson

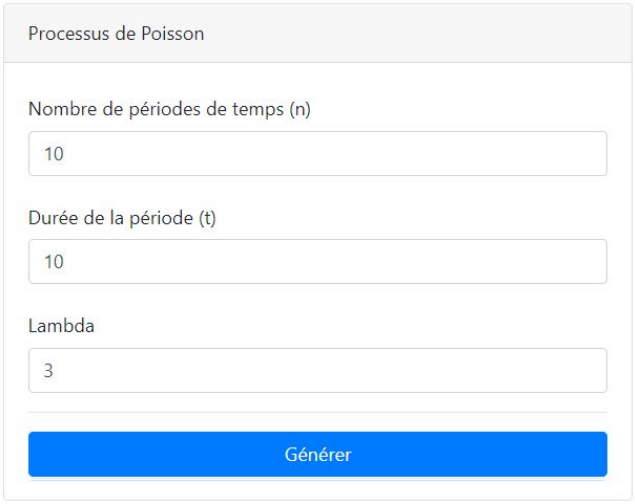
```
//Effectue un processus de Poisson en fonction du nombre de periode à simuler, la durée d'une période, et d'un lambda
function processus_poisson(nb_periode, duree, lambda, listeTempsEvent, classeTab){
    var tempsEvent;
    nb_event = 1;
    var tempsTotal = 0;

   //On simule un event
    tempsEvent = loi_expo(lambda);
    //tant que le moment d'apparition de l'event est inférieur a la durée actuel
    while ((tempsTotal + tempsEvent) < (nb_periode * duree)) {
    //On enregistre le temps
    tempsTotal += tempsEvent;
       listeTempsEvent.push(tempsTotal);
       //On incrémente le nombre d'event
       nb_event += 1;
       //On simule un autre event
       tempsEvent = loi_expo(lambda);
    }
    //Creation du tableau de classes
    createTabClasse(listeTempsEvent, duree, classeTab);
    return nb_event;
}
```

Cet algorithme nous permet de récupérer un tableau contenant le temps d'apparition de chaque événement, le nombre d'événements total. Par la suite nous générons, grâce à ces informations un tableau correspondant aux classes d'occurrences des événements.

Nous avons ensuite déterminer certaines statistiques pour analyser notre simulateur. Sur la simulation nous aurons le nombre d'événements, la moyenne d'événement par période et la moyenne de temps entre deux événements. Nous aurons aussi leurs valeurs théoriques et la valeurs D^2 du test du Khi² d'ajustement.

L'application

Dans notre application voilà le premier paramètre que nous avons simulé :



Ce qui nous a donné comme résultats :

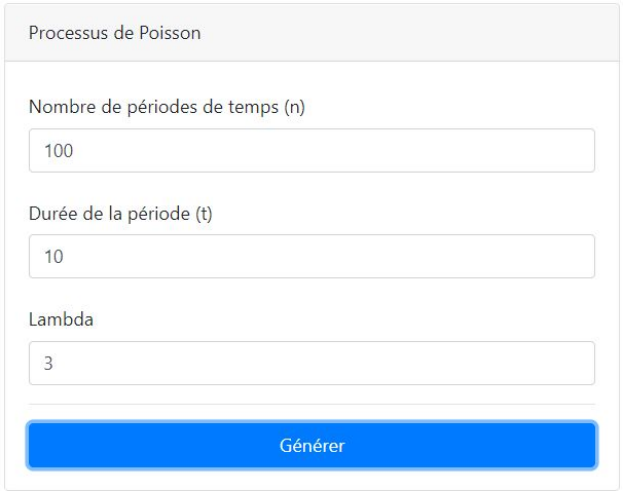
On peut voir que le test du Khi² est valide la justesse de nos données à une précision de 95% avec 8 degré de liberté.

En revanche, la moyenne d'événements par périodes et la moyenne de temps entre 2 événements sont proches de leurs moyennes théoriques mais pas assez pour pouvoir faire un réel constat.

Nous avons donc fait une seconde simulation pour faire un constat sur les grands nombres.

On pourra noter que l'augmentation du nombre de périodes influe sur le nombre de classes et donc le degré de liberté.

Paramétrage :



Résultat :

On constate sur cette simulation que les moyennes sont plus ajustées aux moyennes théoriques.

On pourra noter que le temps d'exécution grandit avec le nombre de période mais surtout avec le paramètre lambda.

Ce TP nous a permis de mettre en place une simulation d'un processus aléatoire par le biais d'un ordinateur mais aussi de comparer les valeurs théoriques aux valeurs pratiques.

Pour cela nous avons pris en compte plusieurs paramètres entrés par l'utilisateur; la cadence qui fait varier la fréquence d'apparition de l'événement, le nombre de périodes et aussi leurs durées. Pour vérifier la validité de nos résultats, nous avons appliqué un test d'ajustement, plus précisément le test du Khi². De plus nous avons effectué un test sur les grands nombres en augmentant le nombre de périodes.

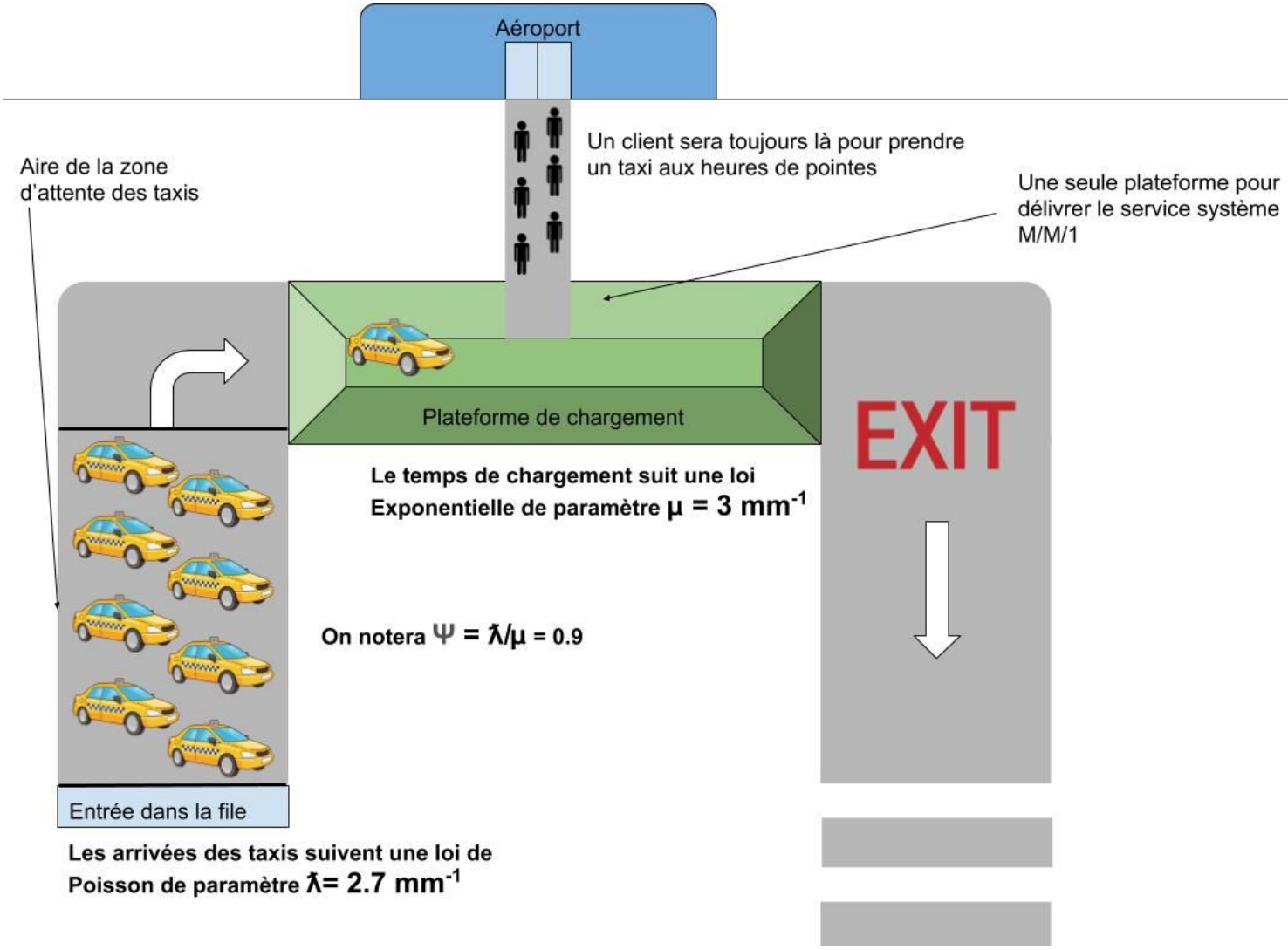
Du côté algorithmique, nous avons pu consolider nos connaissances en javascript et technologies du WEB.

TPF - Gestion parking taxi (V3)

Théorie

Ce TP consiste en l'implémentation décrit plus haut dans la partie "La théorie". On cherchera à simuler le processus et comparer les résultats avec les résultats théoriques tout en appliquant les tests habituelles (grand nombres et Khi²). On pourra noter que les premières valeurs choisies pour la simulation proviennent du sujet de partiel n°TF1305 du livret de partiels du cours de mathématiques et analyse des systèmes décrit plus haut.

Voici un schéma explicatif du processus :



L'application

Pour effectuer cette simulation nous nous somme basé sur ce qui a été établis lors du TP2. En effet notre problème correspond à un processus de Poisson auquel on a rajouté les différents éléments relatifs à la file d'attente et le temps de traitement.

Paramétrage de l'application :

→ Le nombre de période à simuler (nb_periode)

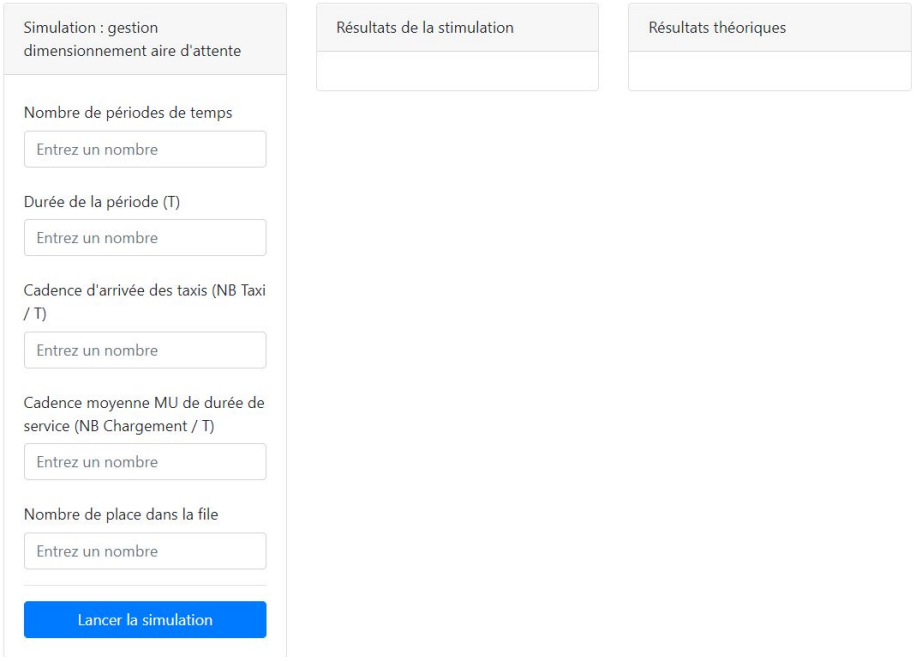
→ La durée t d'une période (duree)

→ Cadence λ d'arrivée des taxis par unité de temps (lambda)

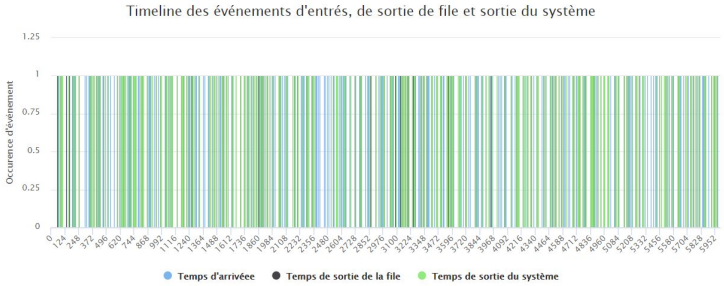
→ Cadence μ de temps de chargement des usagers dans le taxi par unité de temps (mu)

→ Nombre de places dans la file (nbPlaceFile)

Voici l'ihm de base de notre simulateur :



Nous disposons aussi d'une timeline zoomable représentant l'entrée, la sortie de la file et la sortie du système en fonction du temps.



Nous aurons aussi comme statistiques :

E(NbS) : nombre de taxis moyens dans le système

E(TAS) : temps d'attente moyen dans le système

E(NbF) : nombre de taxis moyens dans la file

E(TAF) : temps d'attente moyen dans la file

E(TC) : temps de chargement moyen

Maintenant le rendu de la simulation :

On observa à gauche les paramètre de la simulation, au centre les résultats de la simulation et à droite les paramètre théoriques.

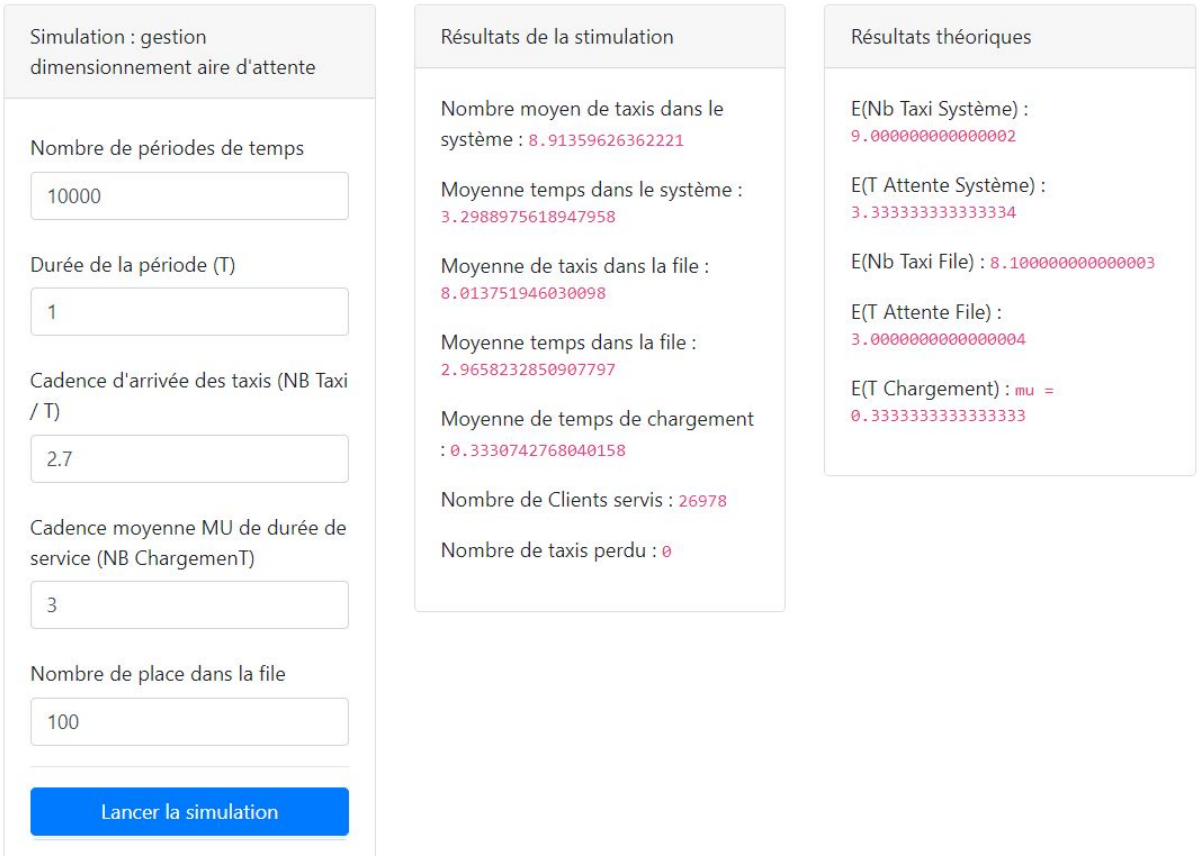
En utilisant les paramètres de l'exercice TF1305 ($\lambda = 2.7\ mm^{-1}\ et\ \mu = 3\ mm^{-1}$) :

On prend une taille égale à 10 pour commencer.

On constate immédiatement que le le nombre de taxis perdus est trop élevé.

On pourra aussi noter que la petite taille de la file ne fait pas tant augmenter le temps passé à l'intérieur.

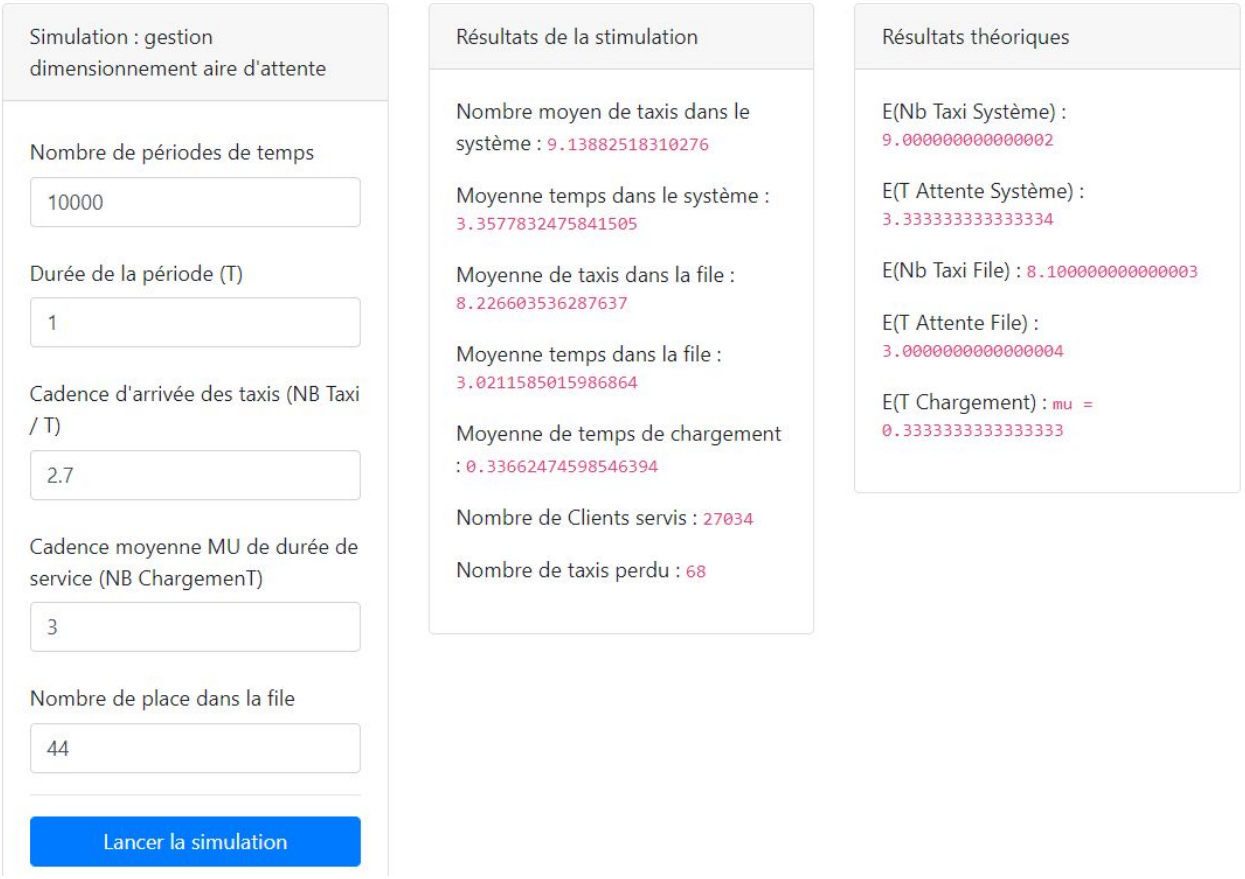
Pour la deuxième simulation nous testons cette fois avec une file d'attente de taille 100.



Logiquement, avec une file de taille plus grande, nous nous rapprochons des valeurs théorique, on constatera qu'il n'y a eu aucun taxi perdu donc cette taille est bien plus grande que la taille nécessaire pour le fonctionnement idéal du système.

D'après la théorie il faut 43.7 soit 44 places dans la file, cela implique que la probabilité qu'un taxi ne puisse pas rentrer soit inférieur à 1% d'après la formule de Little.

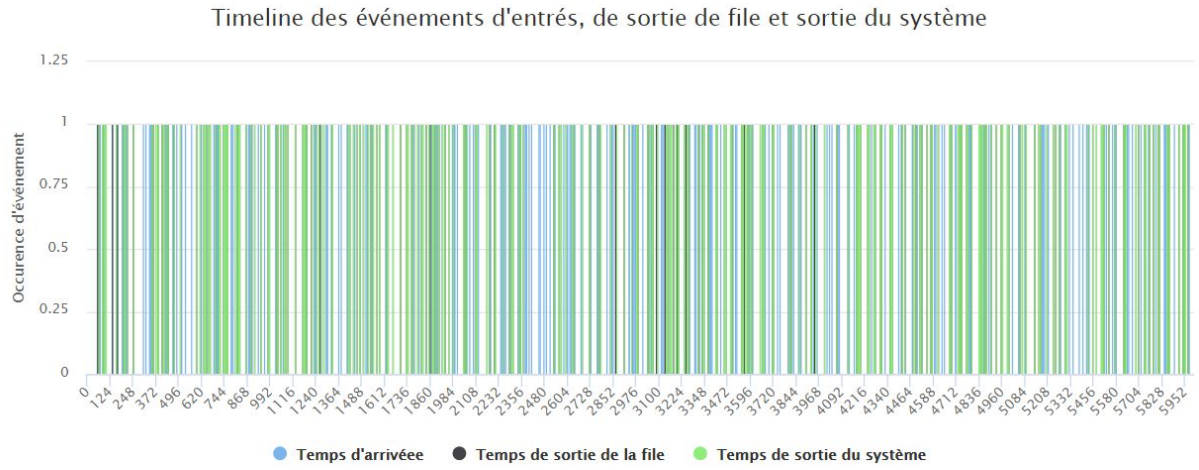
Avec les bon paramètres :



Ici on constate bien que la taille de la file permet de se rapprocher au maximum des valeurs théoriques bien que 68 taxis sont perdus par manque de place dans la file, ce manque de place reste acceptable.

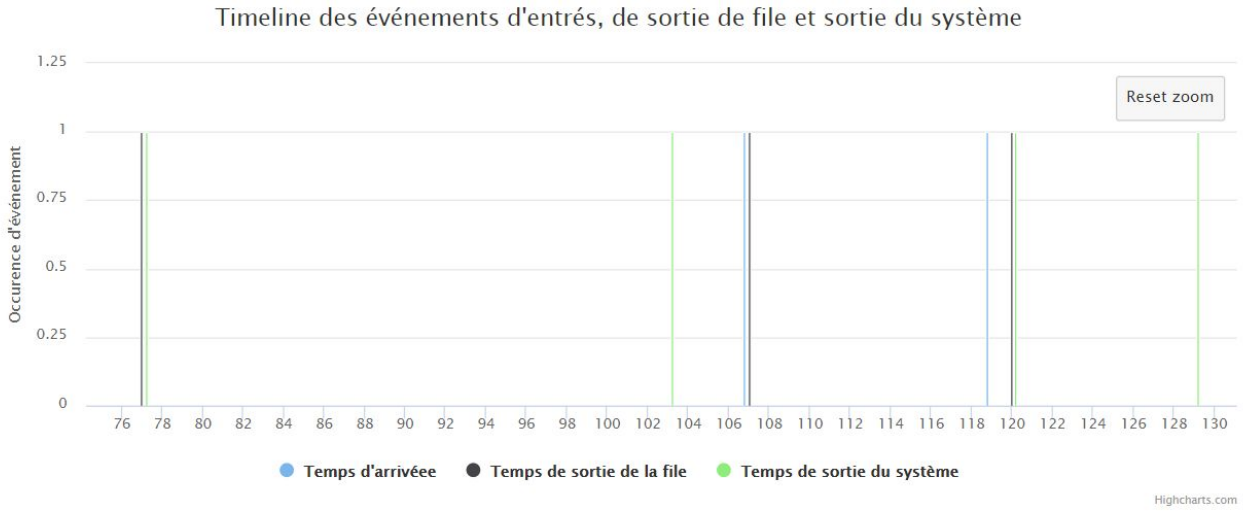
On constatera que l'aéroport dispose de 72 de places d'attentes ce qui nous fait revenir dans la situation ou la file est plus grande que le nécessaire ce qui fait qu'aucun taxi n'est perdu.

Le rendu graphique ressemble, de prime abord, à une timeline de toutes la simulation ce sui pose des difficultés d'analyse :



On notera que l'axe des abscisses correspond au temps et que le pas est un paramètre entré par l'utilisateur, par défaut nous avons mis un pas à 60 pour toutes les secondes de la minute.

En dessinant un carré sur le graphique cela permet de zoomer sur la zone choisi (la démonstration réel sera plus parlante) :

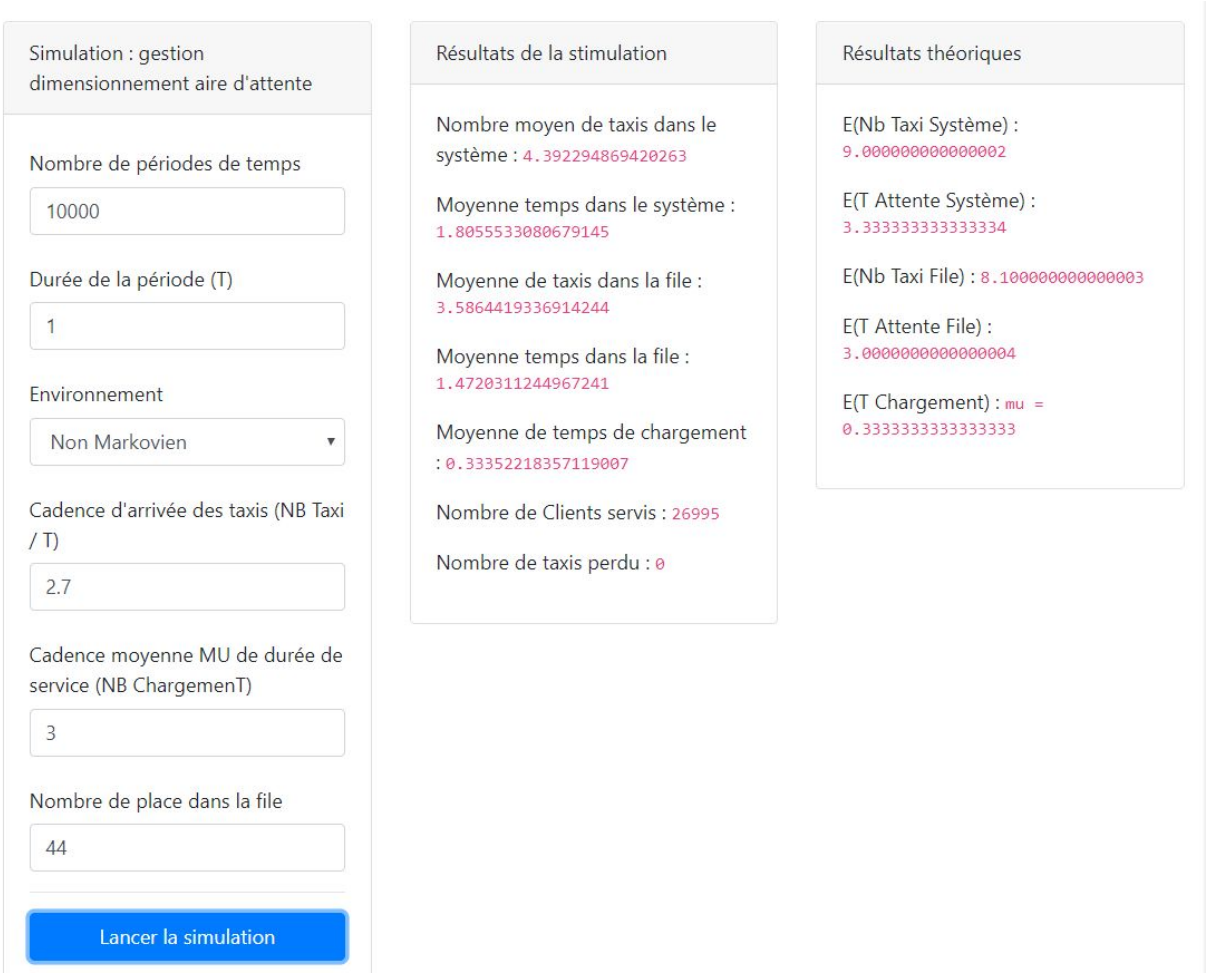


Environnement non Markovien

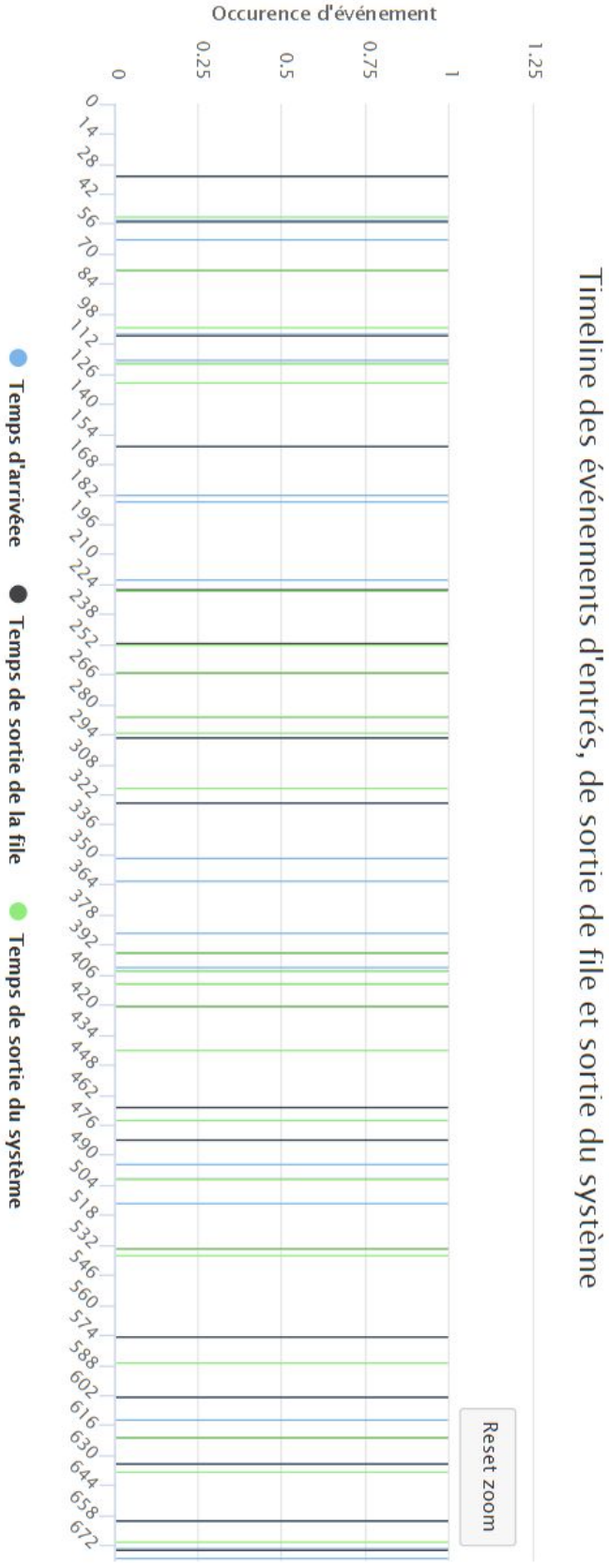
On considère la simulation comme étant une simulation d'un processus de Markov, mais à titre de comparaison nous avons implémenté une simulation similaire en environnement non Markovien.

En effet au lieu d'utiliser un processus de Poisson / Loi exponentielle, nous utilisons une loi normale ou une loi uniforme. Cela permet de comparer l'efficacité d'une simulation d'événement réels. Dans ce cas la on n'est plus dans un système M/M/1 mais dans un système G/M/1, si on remplace la loi des services on se retrouve dans un système G/G/1.

Voici le rendu sur l'application pour la loi uniforme (version améliorées de l'existant) :



Voici le rendu visuel du processus en environnement non Markovien :



Voici donc le principal algorithme utilisés pour répondre à la problématique :

```
function processus_taxi(nb_periode, duree, lambda, mu, nbPlaceFile){
    var temps_sortie = 0; // Fin de tout les chargements
    var tab_taxi = new Array();
    var tabProcPoisson;
    var nb_taxi_perdu = 0;
    var moy_taille_file = 0;
    var moy_taille_syst = 0;

    // boucler jusqu'a la fin de la simulation
    for(var i = 0; i < nb_periode; i++) {
        // nombre de taxi arrivant dans la période
        tabProcPoisson = new Array();
        var proc_poisson = processus_poisson(1, duree, lambda, tabProcPoisson, new Array);
        // traiter chaque taxis
        for(var j = 0; j < tabProcPoisson.length; j++) {
            // arrivée
            var temps_arrivee = (i * duree) + tabProcPoisson[j];
            // Pour le calcul de la moyenne taille de fille (en nb taxis)
            moy_taille_file += taille_file(tab_taxi, temps_arrivee);
            // Pour le calcul de la moyenne taille du système (en nb de taxis)
            moy_taille_syst += taille_syst(tab_taxi, temps_arrivee);
            // SI il reste de la place dans la file
            if (taille_file(tab_taxi, temps_arrivee) < nbPlaceFile) {
                // On génère un temps de chargement
                var tps_chargement = loi_expo(mu);
                // calcul du temps de sortie (début au plus tot + temps de chargement)
                temps_sortie = Math.max(temps_sortie, temps_arrivee) + tps_chargement;
                // ajout du taxi dans la liste
                tab_taxi.push(new Array(
                    temps_arrivee,
                    temps_sortie,
                    (tps_chargement),
                    taille_file(tab_taxi, temps_arrivee)
                ));
            } else {
                nb_taxi_perdu++;
            }
        }
    }
```

En ce qui concerne les différents tests :

Khi2 : Le Khi2 peut être utilisé pour vérifier le bon fonctionnement du simulateur, les différents test sur les lois et le processus de Poisson sont disponibles dans la partie TP-1 et TP-2.

Loi des grands nombre : le test sur les grands nombres a été appliqués au moment du choix du nombre de périodes dans la simulation, en effet, plus le nombre de périodes sera grand, plus les données seront précises. Ce test peut être effectué par l'utilisateur.

Les algorithmes annexes

```
//Taille de la file à un moment donné
//Taxi dont le temps d'arrivée est inférieur au temps
//et dont le temps de sortie < temps
function taille_syst(liste, temps) {
    var taille = 0;
    $.each(liste, function(id, val){
        if(val[0] < temps && val[1] > temps) {
            taille++;
        }
    });
    return taille;
}

//Nombre moyen de clients dans le système
function ENBS(lambda, mu){
    var psi = Number (lambda/(mu));
    return psi/(1-psi);
}


//temps moyen de clients dans le système
function ETAS(lambda, mu){
    var psi = Number (lambda/(mu));
    return psi/(lambda*(1-psi));
}

//Nombre moyen de clients dans la file
function ENBF(lambda, mu){
    var psi = Number (lambda/(mu));
    return (psi*psi)/(1-psi);
}

//temps moyen des clients dans la file
function ETAF(lambda, mu){
    var psi = Number (lambda/(mu));
    return (psi*psi)/(lambda*(1-psi));
}

//Fraction d'inocupation de la station de chargement
function ENBSI(lambda, mu){
    var psi = Number (lambda, (mu));
    return 1-psi;
}
```

Conclusion

Durant ce projet, nous avons simulé plusieurs expériences portant sur des phénomènes aléatoires. Ces phénomènes ont été simulés grâce à des différents processus se basant sur la génération de nombres pseudo-aléatoires.

Les différents outils que nous avons développés et que nous avions à dispositions nous ont permis de faire du calcul décisionnel pour répondre à une problématique réelle. Ici l'exemple du dimensionnement d'un parking d'un aéroport pour limiter le temps d'attente des taxis.
Cela nous a permis de mieux comprendre les phénomènes de files d'attentes, d'arrivées et de traitements et plus généralement les phénomènes de naissances et de morts; Cela associé à l'utilisation des différentes lois. (Markovienne, Gaussienne/Généraliste, Déterministe...)
Notre outil permet aussi de tester la validité de nos données par le biais du test du X^2 et donne la possibilité à l'utilisateur de tester sur des grands nombres.
Une ouverture de ce sujet serait d'étendre la portée de notre système pour en accueillir d'autres paramètres comme le nombre de stations.

L'analyse des systèmes et la modélisation permettent donc de répondre à des problématique réelle par le biais de calculs complexes. Les solutions proposées sont optimales ce qui prouve l'efficacité de cette discipline.

Code

TP1

Voici les différentes fonctions pour avoir les probabilités théoriques.

```
//Loi UNIFORME
function uni() {
    return 0.1;
}

//Loi de POISSON
function poisson(min,max, lambda){
    var poisson = 0;
    for (i = Math.ceil(min); i<=Math.floor(max); i++){
        poisson +=  Math.pow(lambda, i)/fact(i)*Math.exp(-lambda);
    }
    return poisson;
}

//Loi EXPONENTIELLE
function exp(min,max, lambda){
    var infMin = 0;
    var infMax = 0;

    //proba inférieur à min
    infMin =  1 - Math.exp(-lambda*min);
    //proba supérieur a min
    infMax =  1 - Math.exp(-lambda*max);
    //on retourne proba max - proba min pour avoir l'intervalle
    return infMax-infMin;
}

//Loi NORMALE
 function norm(min, max, esperance, ecart_type){
    var borneMin = Number((min-esperance)/ecart_type);
    var borneMax = Number((max-esperance)/ecart_type);
    return Pi(borneMax) - Pi(borneMin);
}

//Loi WEIBULLL
 function weibull(min, max, lambda, forme){
    var borneMin = 1 - Math.exp(-(Math.pow(min/lambda, forme)))
    var borneMax = 1 - Math.exp(-(Math.pow(max/lambda, forme)))
    return borneMax - borneMin;
}
```

Code du khi2

```
//************ Khi2
/**
 *  Test du Khi2
 */
function khi2(loi, nombre_tab) {
    // format: {borne_min, borne_max, nb_nombre, proba)
    var classe_tab = new Array();
    var d = 0;

    // loi discrete
    if (loi == "poi") {
        var nb_classe = maxTab(nombre_tab) - minTab(nombre_tab);
    } else {
        var nb_classe = 10;
    }
    //Tableau des classes contenant les informations de chaque classe au format {borne_min, borne_max, nb_nombre, proba)
    for (var i = 0; i < nb_classe; i++) {
        classe_tab[i] = new Array();
        classe_tab[i]["borne_min"] = getBorneMin(loi, i, nombre_tab);
        classe_tab[i]["borne_max"] = getBorneMax(loi, i, nombre_tab);
        classe_tab[i]["nb_nombre"] = getProportion(nombre_tab, classe_tab[i]["borne_min"], classe_tab[i]["borne_max"]);
        classe_tab[i]["proba"] = get_proba(loi, classe_tab[i]["borne_min"], classe_tab[i]["borne_max"]);
        classe_tab[i]["effObs"] = getEffectif(nombre_tab, classe_tab[i]["borne_min"], classe_tab[i]["borne_max"]);
        classe_tab[i]["effTheo"] = getEffectifTheorique(nombre_tab, classe_tab[i]["borne_min"], classe_tab[i]["borne_max"]);
    }
    //CALCUL DU KHI2
    for (var i = 0; i < nb_classe; i++){
        d = d + Math.pow(classe_tab[i]["effObs"]-classe_tab[i]["effTheo"],2)/classe_tab[i]["effTheo"];
    }
    //Affichage HTML
    $("#d").html(precisionRound(d, 3));
    return classe_tab
}
```

TP2

IHM et gestion du comportement des boutons

```
function lancer_processus_poisson(){
     var listeTempsEvent = new Array();
     var classeTab = new Array();
     var nb_event = processus_poisson(nb_periode, duree, lambda, listeTempsEvent, classeTab);
     var ddl = nb_periode - 2;
     $("#evnt_tot").html(nb_event);
     $("#mEvntPeriode").html(mEvntPeriode(classeTab));
     $("#evntPeriodeTheo").html(evntPeriodeTheo(duree, lambda));
     $("#mTempsInter").html(tempsInterTheo(lambda));
     $("#tempsInterTheo").html(mTempsInter(listeTempsEvent, nb_event));
     $("#khi2").html(khi2Poisson(classeTab, lambda, duree));
}
```

Les algorithmes annexes

Création de classes :

```
//Crée le tableau de classe (trie par periode)
function createTabClasse(listeTempsEvent, duree, classeTab){
    var indicePeriode = 0;
    for (i=0; i<listeTempsEvent.length; i++){
        if (listeTempsEvent[i] <= duree * (indicePeriode + 1)){
            if (isNaN(classeTab[indicePeriode])){
                classeTab[indicePeriode] = 1;
            }else{
                classeTab[indicePeriode] = classeTab[indicePeriode] + 1;
            }
        }else{
           indicePeriode+=1;
        }
    }

}
```

Moyenne des événements par période :

```
//Retourne la moyenne d'événements par période
function mEvntPeriode(classTab){
    var nb = 0;
    var s = 0;
    for (i=0; i<classTab.length; i++){
        s += classTab[i];
    }
    nb = s / classTab.length +1;
    return nb;
}
```

Événements par période théorique :

```
//retourne le nombre d'événements par période théorique
 function evntPeriodeTheo(duree, lambda){
    return duree*lambda;
}
```

Moyenne de temps entre événements :

```
//Retourn la moyenne d'intervalle de temps entre deux événements de notre simulation
function mTempsInter(listeTempsEvent, nb_event){
    var s = 0;
    for (i = 0; i < listeTempsEvent.length - 1; i++){
        s += listeTempsEvent[i+1] - listeTempsEvent[i];
    }
    return s/nb_event;
}
```

Moyenne de temps entre événements théorique :

```
//Retourne le temps théorique entre deux événements
function tempsInterTheo(lambda){
    if (lambda>0){
        return 1/lambda;
    }else{
        return 0;
    }
}
```

Khi 2 simplifié :

```
function khi2Poisson(classeTab, lambda, duree){
    var s = 0;
    var classeInf5 = 0;
    var theo = evntPeriodeTheo(duree, lambda)
    for (i=0; i<classeTab.length; i++){
        if (classeTab[i] >= 5){
            s+= Math.pow(((classeInf5+classeTab[i])-theo), 2)/theo
            classeInf5 = 0;
        }else{
            classeInf5 = classeTab[i];
        }
    }
    return s;
}
```

Annexe

Loi de Khi-deux

Le tableau donne x tel que **P(K > x) = p**

p	0,999	0,995	0,99	0,98	0,95	0,9	0,8	0,2	0,1	0,05	0,02	0,01	0,005	0,001
ddl														
1	0,0000	0,0000	0,0002	0,0006	0,0039	0,0158	0,0642	1,6424	2,7055	3,8415	5,4119	6,6349	7,8794	10,8276
2	0,0020	0,0100	0,0201	0,0404	0,1026	0,2107	0,4463	3,2189	4,6052	5,9915	7,8240	9,2103	10,5966	13,8155
3	0,0243	0,0717	0,1148	0,1848	0,3518	0,5844	1,0052	4,6416	6,2514	7,8147	9,8374	11,3449	12,8382	16,2662
4	0,0908	0,2070	0,2971	0,4294	0,7107	1,0636	1,6488	5,9886	7,7794	9,4877	11,6678	13,2767	14,8603	18,4668
5	0,2102	0,4117	0,5543	0,7519	1,1455	1,6103	2,3425	7,2893	9,2364	11,0705	13,3882	15,0863	16,7496	20,5150
6	0,3811	0,6757	0,8721	1,1344	1,6354	2,2041	3,0701	8,5581	10,6446	12,5916	15,0332	16,8119	18,5476	22,4577
7	0,5985	0,9893	1,2390	1,5643	2,1673	2,8331	3,8223	9,8032	12,0170	14,0671	16,6224	18,4753	20,2777	24,3219
8	0,8571	1,3444	1,6465	2,0325	2,7326	3,4895	4,5936	11,0301	13,3616	15,5073	18,1682	20,0902	21,9550	26,1245
9	1,1519	1,7349	2,0879	2,5324	3,3251	4,1682	5,3801	12,2421	14,6837	16,9190	19,6790	21,6660	23,5894	27,8772
10	1,4787	2,1559	2,5582	3,0591	3,9403	4,8652	6,1791	13,4420	15,9872	18,3070	21,1608	23,2093	25,1882	29,5883
11	1,8339	2,6032	3,0535	3,6087	4,5748	5,5778	6,9887	14,6314	17,2750	19,6751	22,6179	24,7250	26,7568	31,2641
12	2,2142	3,0738	3,5706	4,1783	5,2260	6,3038	7,8073	15,8120	18,5493	21,0261	24,0540	26,2170	28,2995	32,9095
13	2,6172	3,5650	4,1069	4,7654	5,8919	7,0415	8,6339	16,9848	19,8119	22,3620	25,4715	27,6882	29,8195	34,5282
14	3,0407	4,0747	4,6604	5,3682	6,5706	7,7895	9,4673	18,1508	21,0641	23,6848	26,8728	29,1412	31,3193	36,1233
15	3,4827	4,6009	5,2293	5,9849	7,2609	8,5468	10,3070	19,3107	22,3071	24,9958	28,2595	30,5779	32,8013	37,6973
16	3,9416	5,1422	5,8122	6,6142	7,9616	9,3122	11,1521	20,4651	23,5418	26,2962	29,6332	31,9999	34,2672	39,2524
17	4,4161	5,6972	6,4078	7,2550	8,6718	10,0852	12,0023	21,6146	24,7690	27,5871	30,9950	33,4087	35,7185	40,7902
18	4,9048	6,2648	7,0149	7,9062	9,3905	10,8649	12,8570	22,7595	25,9894	28,8693	32,3462	34,8053	37,1565	42,3124
19	5,4068	6,8440	7,6327	8,5670	10,1170	11,6509	13,7158	23,9004	27,2036	30,1435	33,6874	36,1909	38,5823	43,8202
20	5,9210	7,4338	8,2604	9,2367	10,8508	12,4426	14,5784	25,0375	28,4120	31,4104	35,0196	37,5662	39,9968	45,3147
21	6,4467	8,0337	8,8972	9,9146	11,5913	13,2396	15,4446	26,1711	29,6151	32,6706	36,3434	38,9322	41,4011	46,7970
22	6,9830	8,6427	9,5425	10,6000	12,3380	14,0415	16,3140	27,3015	30,8133	33,9244	37,6595	40,2894	42,7957	48,2679
23	7,5292	9,2604	10,1957	11,2926	13,0905	14,8480	17,1865	28,4288	32,0069	35,1725	38,9683	41,6384	44,1813	49,7282
24	8,0849	9,8862	10,8564	11,9918	13,8484	15,6587	18,0618	29,5533	33,1962	36,4150	40,2704	42,9798	45,5585	51,1786
25	8,6493	10,5197	11,5240	12,6973	14,6114	16,4734	18,9398	30,6752	34,3816	37,6525	41,5661	44,3141	46,9279	52,6197
26	9,2221	11,1602	12,1981	13,4086	15,3792	17,2919	19,8202	31,7946	35,5632	38,8851	42,8558	45,6417	48,2899	54,0520
27	9,8028	11,8076	12,8785	14,1254	16,1514	18,1139	20,7030	32,9117	36,7412	40,1133	44,1400	46,9629	49,6449	55,4760
28	10,3909	12,4613	13,5647	14,8475	16,9279	18,9392	21,5880	34,0266	37,9159	41,3371	45,4188	48,2782	50,9934	56,8923
29	10,9861	13,1211	14,2565	15,5745	17,7084	19,7677	22,4751	35,1394	39,0875	42,5570	46,6927	49,5879	52,3356	58,3012
30	11,5880	13,7867	14,9535	16,3062	18,4927	20,5992	23,3641	36,2502	40,2560	43,7730	47,9618	50,8922	53,6720	59,7031
40	17,9164	20,7065	22,1643	23,8376	26,5093	29,0505	32,3450	47,2685	51,8051	55,7585	60,4361	63,6907	66,7660	73,4020
50	24,6739	27,9907	29,7067	31,6639	34,7643	37,6886	41,4492	58,1638	63,1671	67,5048	72,6133	76,1539	79,4900	86,6608
60	31,7383	35,5345	37,4849	39,6994	43,1880	46,4589	50,6406	68,9721	74,3970	79,0819	84,5799	88,3794	91,9517	99,6072
70	39,0364	43,2752	45,4417	47,8934	51,7393	55,3289	59,8978	79,7146	85,5270	90,5312	96,3875	100,4252	104,2149	112,3169
80	46,5199	51,1719	53,5401	56,2128	60,3915	64,2778	69,2069	90,4053	96,5782	101,8795	108,0693	112,3288	116,3211	124,8392
90	54,1552	59,1963	61,7541	64,6347	69,1260	73,2911	78,5584	101,0537	107,5650	113,1453	119,6485	124,1163	128,2989	137,2084
100	61,9179	67,3276	70,0649	73,1422	77,9295	82,3581	87,9453	111,6667	118,4980	124,3421	131,1417	135,8067	140,1695	149,4493
120	77,7551	83,8516	86,9233	90,3667	95,7046	100,6236	106,8056	132,8063	140,2326	146,5674	153,9182	158,9502	163,6482	173,6174
140	93,9256	100,6548	104,0344	107,8149	113,6593	119,0293	125,7581	153,8537	161,8270	168,6130	176,4709	181,8403	186,8468	197,4508
160	110,3603	117,6793	121,3456	125,4400	131,7561	137,5457	144,7834	174,8283	183,3106	190,5165	198,8464	204,5301	209,8239	221,0190
180	127,0111	134,8844	138,8204	143,2096	149,9688	156,1526	163,8682	195,7434	204,7037	212,3039	221,0772	227,0561	232,6198	244,3705
200	143,8428	152,2410	156,4320	161,1003	168,2786	174,8353	183,0028	216,6088	226,0210	233,9943	243,1869	249,4451	255,2642	267,5405
250	186,5541	196,1606	200,9386	206,2490	214,3916	221,8059	231,0128	268,5986	279,0504	287,8815	298,0388	304,9396	311,3462	324,8324
300	229,9634	240,6634	245,9725	251,8637	260,8781	269,0679	279,2143	320,3971	331,7885	341,3951	352,4246	359,9064	366,8444	381,4252
400	318,2596	330,9028	337,1553	344,0781	354,6410	364,2074	376,0218	423,5895	436,6490	447,6325	460,2108	468,7245	476,6064	493,1318
500	407,9470	422,3034	429,3875	437,2194	449,1468	459,9261	473,2099	526,4014	540,9303	553,1268	567,0698	576,4928	585,2066	603,4460
600	498,6229	514,5289	522,3651	531,0191	544,1801	556,0560	570,6680	628,9433	644,8004	658,0936	673,2703	683,5156	692,9816	712,7712
700	590,0480	607,3795	615,9075	625,3175	639,6130	652,4973	668,3308	731,2805	748,3591	762,6607	778,9721	789,9735	800,1314	821,3468
800	682,0665	700,7250	709,8969	720,0107	735,3623	749,1852	766,1555	833,4557	851,6712	866,9114	884,2789	895,9843	906,7862	929,3289
900	774,5698	794,4750	804,2517	815,0267	831,3702	846,0746	864,1125	935,4987	954,7819	970,9036	989,2631	1001,6296	1013,0364	1036,8260

Sources

Algorithme de poisson : http://www.les-mathematiques.net/phorum/read.php?12,560171,560171

Loi uniforme :

Classes	0 - 1	1 - 2	2 - 3	3 - 4	4 - 5	5 - 6	6 - 7	7 - 8	8 - 9	9 - 10
Proportion	0.1	0.101	0.099	0.1	0.1	0.1	0.1	0.1	0.1	0.1
Probabilité	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
Eff Observé	99988	100622	99347	100164	100078	100092	99697	99794	99938	100278
Eff Théorique	100000	100000	100000	100000	100000	100000	100000	100000	100000	100000

Loi exponentielle

Classes	0 - 0.251	0.251 - 0.502	0.502 - 0.753	0.753 - 1.003	1.003 - 1.254	1.254 - 1.505	1.505 - 1.756	1.756 - 2.007	2.007 - 2.258	2.258 - 2.509
Proportion	0.716	0.202	0.059	0.016	0.005	0.001	0	0	0	0
Probabilité	0.715	0.204	0.058	0.017	0.005	0.001	0	0	0	0
Eff Observé	71639	20205	5869	1624	481	123	38	14	3	2
Eff Théorique	71470.449	20389.179	5816.65	1659.381	473.39	135.049	38.527	10.991	3.136	0.895

Loi de Poisson:

Classes	0 - 0	1 - 1	2 - 2	3 - 3	4 - 4	5 - 5	6 - 6	7 - 7	8 - 8	9 - 9	10 - 10	11 - 11	12 - 12	13 - 13	14 - 14	15 - 15	16 - 16	17 - 17	18 - 18
Proportion	0.007	0.034	0.084	0.14	0.176	0.175	0.146	0.105	0.065	0.036	0.018	0.008	0.003	0.001	0	0	0	0	0
Probabilité	0.007	0.034	0.084	0.14	0.175	0.175	0.146	0.104	0.065	0.036	0.018	0.008	0.003	0.001	0	0	0	0	0
Eff Observé	6848	33840	83963	140377	175695	175485	146194	104605	64953	36301	17969	8304	3467	1336	435	153	55	15	3
Eff Théorique	6737.947	33689.735	84224.337	140373.896	175467.37	175467.37	146222.808	104444.863	65278.039	36265.577	18132.789	8242.177	3434.24	1320.862	471.736	157.245	49.139	14.453	4.015

Loi normale :

Classes	1.909 - 3.687	3.687 - 5.465	5.465 - 7.243	7.243 - 9.022	9.022 - 10.8	10.8 - 12.578	12.578 - 14.356	14.356 - 16.134	16.134 - 17.912	17.912 - 19.69
Proportion	0.001	0.011	0.073	0.23	0.343	0.244	0.083	0.014	0.001	0
Probabilité	0.001	0.011	0.072	0.228	0.343	0.246	0.084	0.014	0.001	0
Eff Observé	82	1110	7276	23048	34271	24396	8318	1385	106	6
Eff Théorique	77.271	1088.703	7237.723	22828.739	34301.69	24591.679	8401.118	1362.349	104.302	3.748