3iL

TP2 – Processus de Poisson

Modélisation et Analyse de Systèmes

FOCK-CHOW-THO Sandrine – PEREZ Emmanuel

03/05/2018

Table des matières

I. INTRODUCTION

L'objet de cette étude est la réalisation de programmes de simulation du processus de Poisson. Pour cela, nous travaillerons d'abord sur la définition d'un processus de Poisson et de ses caractéristiques. Nous indiquerons les différents domaines d'application, puis nous réaliserons une IHM qui simulera le problème. Cela nous permettra principalement de comparer les résultats et les analyser.

II. PROCESSUS DE POISSON

1. Présentation de la loi de Poisson

En théorie des probabilités et en statistiques, la loi de Poisson est une loi de probabilité discrète qui décrit le comportement du nombre d'évènements se produisant dans un laps de temps fixé, si ces évènements se produisent avec une fréquence moyenne connue et indépendamment du temps écoulé depuis l'évènement précédent. La loi de Poisson est pertinente pour décrire le nombre d'événements dans d'autres types d'intervalles, spatiaux plutôt que temporels, comme des segments, surfaces ou volumes.

Si le nombre moyen d'occurrences dans un intervalle de temps fixé est λ, alors la probabilité qu'il existe exactement *k* occurrences (*k* étant un entier naturel, k = 0, 1, 2, …) est :

$$p(k) = P(X = k) = \frac{\lambda^k}{k!}\, e^{-\lambda}$$

On dit ici que *X* suit la loi de Poisson de paramètre *λ*.

2. Qu'est-ce qu'un processus ?

D'après le cours, un processus aléatoire est un processus dont l'état varie de manière aléatoire en fonction d'une variable qui est souvent le temps. Son état est caractérisé par l'un de ses paramètres (position, vitesse, couleur, nombre d'éléments, taille, etc.).

3. Processus de Poisson

3.1. Définition

Un processus de Poisson est :

- Un processus stochastique continu : le paramètre *t* représente le temps,
- Un processus ponctuel : on peut attribuer une date à chaque événement et sa réalisation est instantanée,
- Un processus sans mémoire : il s'agit d'un processus dit markovien (propriété de Markov), c'est-à-dire que chaque probabilité ne dépend pas des valeurs précédentes,
- Dont la cadence λ est constante : nombre d'arrivées d'évènements par intervalle de temps (en s^{-1} ou h^{-1} ou mn^{-1}).

Soit $\{Z(t)_{t\in R+}\}$ un processus de Poisson de taux d'arrivée λ :

On a : $P_n(t) = e^{-\lambda t}.\frac{(\lambda t)^n}{n!}$, $\forall n \in N$ c'est-à-dire que le nombre d'arrivées dans [0,*t*[est une variable aléatoire qui suit une loi de Poisson de paramètre λt.

3.2. Domaine d'application

Un processus de Poisson est un processus de comptage classiquement utilisé dans les modélisations de file d'attente (guichets, péages, etc.), de gestions de stocks ou certains processus physiques ou biologiques.

C'est un modèle simple mais avec des hypothèses simplificatrices pas toujours respectées. Il faut donc bien les avoir en tête pour décider si ce processus est un modèle convenable.

3.3. Restrictions

- Comme vu précédemment, un évènement à venir ne dépend pas des résultats passés mais uniquement du présent.
- Un processus est stationnaire, c'est-à-dire que le nombre d'évènements entre deux instants t_1 et t_2 ne dépend que de t_1 – t_2.
- Le processus ne prend pas en compte deux évènements simultanés.

III. METHODES DE REALISATION

1. Outils théoriques

Dans sa forme la plus simple, on s'intéresse aux processus homogènes. $\{N_t : t \geq 0\}$ est un processus de Poisson homogène d'intensité λ > 0. L'algorithme naturel pour simuler les dates d'arrivées T_1, T_2, …, T_n est alors :

1- Poser T_0 = 0
2- Pour i = 1, 2, …, n, on génère E suivant une loi exponentielle de paramètre λ, puis on pose : $T_i = T_i + E$

Les choses se compliquent lorsque le processus de Poisson n'est plus homogène.

On suppose donc que le processus de Poisson est d'intensité λ(t). La première idée pour simuler un tel processus est de supposer qu'il existe $\bar{\lambda}$ tel que λ(t) ≤ $\bar{\lambda}$ pour tout *t*. On note T_1^*, T_2^*, T_3^*, … des arrivées successives d'un processus de Poisson de paramètre $\bar{\lambda}$. On met alors en place une stratégie acceptation/rejet, telle que l'on garde T_i^* avec probabilité $\frac{\lambda(T_i^*)}{\bar{\lambda}}$. La suite T_1, T_2, …, T_n ainsi construite suit un processus de Poisson non homogène d'intensité λ(t). D'où l'algorithme suivant :

1- On pose T_0 = 0 et $T^* = 0$
2- On génère une variable exponentielle E d'intensité $\bar{\lambda}$
3- On pose alors $T^* = T^* + E$
4- On génère une variable U uniforme sur [0,1]

- Si U > $\frac{\lambda(T^*)}{\bar{\lambda}}$, on rejette (et on retourne à l'étape 2).
- Sinon, on pose $T_i = T^*$.

Une autre méthode consiste à noter que l'incrément $N_t - N_s$ sachant que $0 < s < t$ suit une loi de Poisson d'intensité $\bar{\lambda} = \int_s^t \lambda(u)du$. Aussi, la fonction de répartition F_s du temps d'attente W_s est $F_s(t) = 1 - e^{-\int_s^{s+t} \lambda(u)du} = 1 - e^{-\int_0^t \lambda(s+v)dv}$. On peut utiliser l'algorithme suivant pour simuler le processus de Poisson :

1- Poser $T_0 = 0$
2- Générer U suivant une loi uniforme sur [0,1]
3- Poser $T_i = T_{i-1} + F_s^{-1}(U)$

2. Application

Afin de réaliser cette simulation informatiquement, nous utiliserons du langage web. Le gros avantage de ces langages est la facilité d'implémentation d'une interface graphique.

- Le HTML (Hyper Text Markup Language) est un langage de balise qui structure le contenu d'une page web,
- Le CSS (Cascading Style Sheets) complète le HTML en gérant l'apparence de la page web (agencements, positionnements, décorations, couleurs, taille du texte, etc.),
- Le JavaScript est un langage de programmation de scripts principalement employés dans les pages web interactives. C'est le langage que nous utiliserons pour coder les fonctions gérant la simulation du processus de Poisson.

3. Application : Interface Homme-Machine

Afin de réaliser une interface graphique ergonomique et épurée, nous avons principalement utilisé le framework Bootstrap qui est une collection d'outils utilisé pour la création d'un design (graphisme, animation et interactions avec la page dans le navigateur, etc.) de sites et d'applications web.
Nous avons aussi employé l'interface de programmation applicative (API) « HighCharts », qui est une librairie de fonctions permettant de produire facilement des graphiques à partir de nos données.

Figure 1 : Page d'accueil

La page d'accueil de notre site, concise, propose après une brève introduction de cliquer sur le bouton commencer. Celui-ci nous fait défiler la page jusqu'à la prochaine section, celle du processus de Poisson.

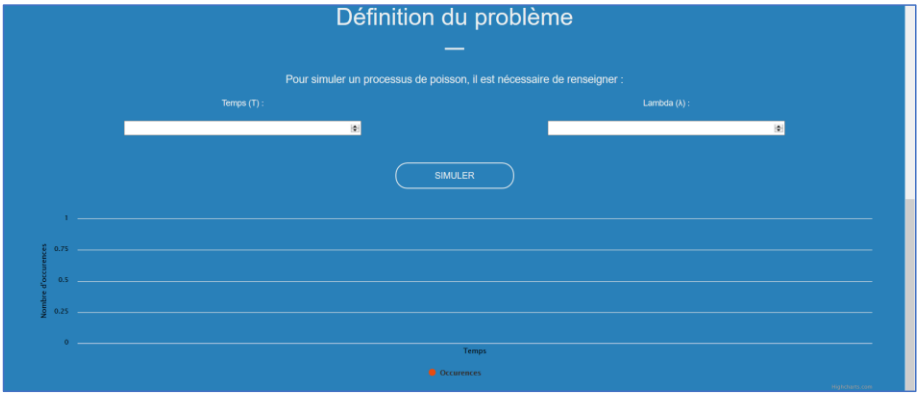

Figure 2 : Page de simulation

Sur cette page, il suffit de remplir la période de temps (T) et la cadence d'apparition des occurrences (λ) pour remplir les conditions nécessaires à la réalisation d'un processus de Poisson. Une simple pression sur la touche « Entrée » ou un clic sur le bouton « SIMULER » lance la simulation.

IV. RESULTATS

Nous avons réalisé un test de simulation avec les paramètres suivants :

- T = 10
- λ = 5

1. Résultats attendus

Nous nous attendons à constater une cadence moyenne proche de notre lambda, ici égal à cinq.

2. Résultats constatés

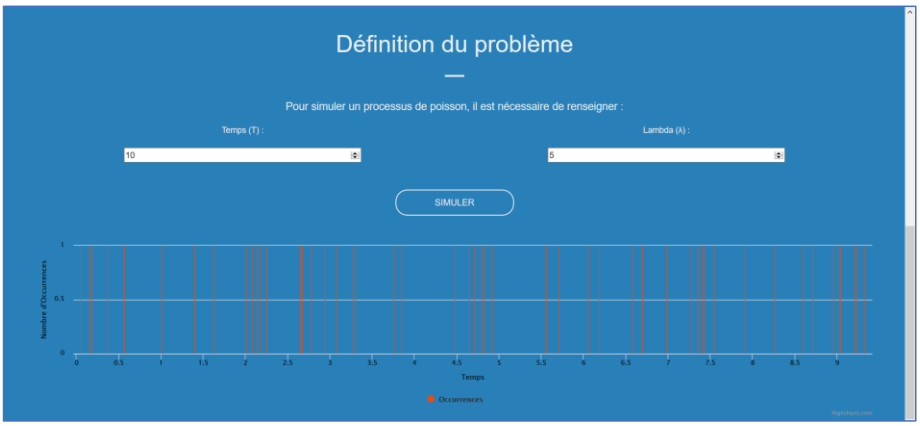

Figure 3 : Essai de simulation

Ci-dessus se trouve la représentation de l'apparition des occurrences au fil du temps. Celles-ci sont instantanées et datées.

Ci-dessous, nous avons la sortie console à la suite de la réalisation de la simulation. Pour chaque intervalle, en plus d'afficher sous forme de trait l'apparition d'une occurrence, le programme les compte. Ce qui nous permet par la suite de calculer la cadence réelle constatée sur la simulation, et de pouvoir la comparée à notre lambda de départ.

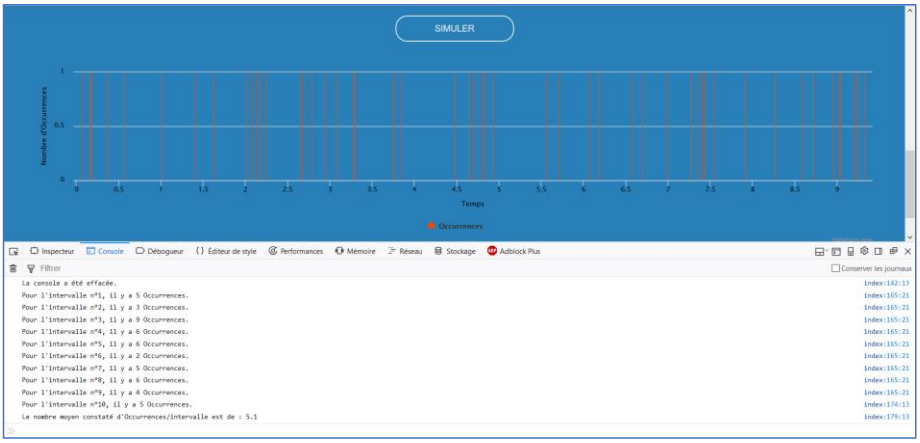

Figure 4 : Sortie console

Dans le cas d'un T supérieur à 10, nous occultons la sortie graphique qui ne fait que ralentir inutilement le traitement des opérations. De plus, son affichage ne rimerait à rien sans devoir gérer une remise à l'échelle pour obtenir un résultat lisible.

3. Comparaison

Les résultats obtenus par le programme informatique et ceux que nous avions censés être obtenus théoriquement, sont relativement proches, même sur un temps de simulation réduit. Cela montre que notre application est capable de simuler un processus de Poisson. Il pourra donc être utilisé sur des cas plus concrets, notamment afin de réaliser un remplissage aléatoire d'une file d'attente en fonction des paramètres imposés.

V. CONCLUSION

Le développement de cette application nous a permis d'obtenir un simulateur du processus de Poisson fonctionnant correctement. Cette étude va nous être très utile pour la suite du travail, qui porte sur les files d'attente.

La théorie des files d'attente étudie les solutions optimales de gestion des files d'attente. Elle peut s'appliquer à diverses situations : gestion des avions au décollage ou à l'atterrissage, attente des clients

et des administrés aux guichets, ou bien encore stockage des programmes informatiques avant leur traitement.

Pour notre part, nous travaillerons sur les files d'attente au niveau des remontées mécaniques et le processus de Poisson nous permettra de simuler l'arrivée des skieurs dans la file.

VI. ANNEXES

Code JavaScript

```
<script type="text/javascript">

/*
 * Cette fonction permet de générer un nombre aléatoire
 * Par la loi de Poisson
 * @param lambda paramètre de la loi de Poisson
 * @return un nombre aléatoire
 */
loiExponentielle = function(lambda)
{
   return -(Math.log(1.0 - Math.random()) / lambda);
};

// Fonction d'affichage d'un graphique initial sans données
$(function()
{
   $('#graph_container').highcharts({
      chart: {
         type: 'column'
      },
      title: {
         text: null
      },
      yAxis: {
         title: {
            text: "Nombre d'Occurrences"
         },
         max: 1.0
      },
      xAxis: {
         title: {
            text: "Temps"
         }
      },
      plotOptions: {
         column: {
            pointPadding: 0.5,
            borderWidth: 0
         }
      },
```

Commenté [SF1]:

```
        series: [{
            name: 'Occurrences',
            data: [ ],
            color: '#e84c0fff'
        }]
    });

    // Action qui suit le clic du bouton "SIMULER"
    $('#simulation_button').on('click', function()
    {
        // Initialisation & récupération des variables
        let chart  = $('#graph_container').highcharts();
        let lambda = parseFloat($('#lambda_input').val());
        let t   = parseInt($('#t_input').val(), 10);
        let cpt = 0;
        let i = 1;
        let moy = 0;

        // On initialise directement le timer à une valeur aléatoire et non à 0
        let timer = loiExponentielle(lambda)

        // On vide la console
        console.clear();

        // On vide le graphe
        chart.series[0].remove(true);
        chart.addSeries({
            name: 'Occurrences',
            data: [ ],
            color: '#e84c0fff'
        });


        for (timer; timer < t; timer += loiExponentielle(lambda))
        {
            if (t <= 10) // On ne réalise la partie graphique que si T <= 10
            {
                chart.series[0].addPoint(
                // timer = abscisse & 1 = ordonnée
                [ timer, 1 ], true
                );
            }

            if (timer > i) // A chaque dépassement d'intervalles, on affiche le nombre d'Occurrences
du précédent, on remet les compteurs à 0
            {
                console.log("Pour l'intervalle n°" + i + ", il y a " + cpt + " Occurrences.");
                moy = moy + cpt;
                cpt = 0;
                i++;
            }
```

```
            cpt++;

        }
        console.log("Pour l'intervalle n°" + i + ", il y a " + cpt + " Occurrences.");
        moy = moy + cpt;

        // Calcul et affichage du nombre moyen constaté d'Occurrences
        moy = moy/i;
        console.log("Le nombre moyen constaté d'Occurrences/intervalle est de : " + moy);
    });

    /// Permet la gestion de pression sur la touche "Entrée" pour valider les saisies utilisateur
    // Champ d'entrée
    var lambda_input = document.getElementById("lambda_input");
    var t_input = document.getElementById("t_input");

    // Réalise un clic sur le bouton "SIMULER" si l'utilisateur appuie sur entrée
    lambda_input.addEventListener("keyup", function(event)
    {
        // Annule l'action par défaut au cas où...
        event.preventDefault();
        // 13 correspond à la touche entrée
        if (event.keyCode === 13) {
            // Déclenche le bouton "SIMULER"
            document.getElementById("simulation_button").click();
        }
    });
    // Même chose pour l'autre champ
    t_input.addEventListener("keyup", function(event)
    {
        event.preventDefault();
        if (event.keyCode === 13) {
            document.getElementById("simulation_button").click();
        }
    });

  });

</script>
```