Université Paris Cité
Mathématiques et Informatique

Master MMA M2, MPPM

Simulation de processus de Poisson

Barthélémy Metzlé et Olivier Bouët-Willaumez

Projet de fin de semestre

27 janvier 2024
Paris

Table des matières

1 Introduction

Nous allons présenter dans ce rapport plusieurs algorithmes permettant de simuler numériquement des processus de Poisson homogènes et non-homogènes. Ces algorithmes ont été introduit en 1979 par P. W. A. Lewis et G. S. Schedler dans un articles intitulé "Simulation of Nonhomogeneous Poisson Processes by Thinning". Nous commençons par faire quelques rappels sur les processus de poisson homogène et non-homogène.

Définition 1. *Un processus de Poisson T_n est un processus stochastique ponctuel dont la fonction de comptage vérifie*

- *Indépendance des accroissements : $N_t - N_s$ est indépendant de $\sigma(N_u; \forall\ u \leq\ s)$ pour $0 \leq s \leq t$*
- *Stationnarité : $\forall\ 0 \leq s \leq t$, $N_t - N_s$ suit la même loi que N_{t-s}*

Nous rappellons que la fonction de comptage d'un processus de Poisson est définie par

$$N_t = sup\{n \geq 0; T_n \leq t\}$$

Plus simplement cela signifie que le processus a une "absence de mémoire" c'est à dire que les événements présent de dépendent pas de ceux du passé. Et que pour deux intervalles de même longueur, la loi du nombre de points appartenant à ces intervalles et la même.

On a le théorème suivant qui nous permet de caractériser les processus de Poisson.

Théorème 1. *Si T_n est un processus de poisson alors $\exists\ \lambda > 0$ tel que $\forall\ t\ \geq 0 \Rightarrow X \sim Poisson(\lambda t)$. Ainsi $\mathbf{E}[N_t] = \mathbf{V}ar[N_t] = \lambda t$.*

On dit alors que λ est l'intensité du processus. Une simple observation nous permet de retrouver que $\lambda = \frac{\mathbf{E}[N_t]}{t}$. Ainsi cette quantité peut être interprété comme la "densité moyenne" des points que l'on cherche à estimer à un temps fixé t. Ce qui nous permet de définir deux types de processus de Poisson.

Définition 2. *- On dit qu'un Processus de Poisson est homogène si λ est constante en fonction du temps.*
- On dit qu'un processus de Poisson est non-homogène si $\lambda = \lambda(t)$ est variable en fonction du temps.

Dans ce rapport nous allons présenter trois algorithmes permettant de simuler ces processus.

Tout d'abord nous présenterons un algorithme capable de simuler des processus de Poisson non-homogènes unidimensionnels et ensuite nous présenterons deux algorithmes permettant de simuler des processus de Poisson homogènes bidimensionnels sur un rectangle et sur un cercle.

2 Simulation de processus de Poisson non-homogène en dimension 1.

Nous avons rappelé en introduction qu'un processus de Poisson non-homogène est un processus de Poisson dont l'intensité est une fonction du temps. C'est à dire que la quantité moyenne d'évenements dans un intervalle varie en fonction du temps.

Définition 3. *Pour un processus de Poisson homogène on a* $\mathbf{E}[N_t] = \int_0^t \lambda dx = \lambda t$. *Soit* T_n *un processus de poisson non-homogène de fonction d'intensité* $\lambda(t)$. *Nous définissons alors sa fonction moyenne :*

$$m(t) = \int_0^t \lambda(x)dx \ = \Lambda(t) - \Lambda(0).$$

Exemple 1. *Un bon exemple peut être la simulation du nombre de clients dans un supermarché où le nombre de clients moyen varie en fonction de l'heure.*

2.1 Théorème indispensable.

Dans cette partie nous allons introduire un théorème sur lequel se repose l'algorithme permettant de simuler un processus de Poisson non-homogène unidimensionnel.

Théorème 2 (P.A. Lewis p.405)**.** *soit* T_n^* *un processus de Poisson non-homogène de fonction d'intensité* $\lambda^*(t)$ *dont la fonction de comptage* N_t *suit une loi de Poisson de paramètre* $m^*(t) = \int_0^{x_0} \lambda^*(t)dt \ = \Lambda^*(t) - \Lambda^*(0)$
Soit $\{T_1^*, ..., T_n^*\}$ *des réalisations de ce processus sur l'intervalle* $[0, t]$ *et* $\forall\ 0 \leq s \leq t,\ \lambda(s) < \lambda^*(s)$ *. Nous construisons maintenant un nouveau processus* T_n *de fonction d'intensité* $\lambda(t)$ *à partir de* T_n^*. *Le but est de conserver les valeurs de* T_n^* *qui nous conviennent. Nous le construisons tel que* $\forall\ i \in [\![1; n]\!]$ *nous rejettons* T_i^* *avec une probabilité* $p = 1 - \frac{\lambda(T_i^*)}{\lambda^*(T_i^*)}$ *et nous l'acceptons avec une probabilité* $\frac{\lambda(T_i^*)}{\lambda^*(T_i^*)}$ *. Le nouveau processus ainsi construit est un processus de Poisson de fonction d'intensité* $\lambda(t)$ *sur* $[0; t]$.

Ce théorème nous permet de définir un algorithme d'aminscissement permettant de simuler des processus de Poisson non-homogènes grâce à d'autres processus de Poisson que nous savons déjà simuler et dont la fonction d'intensité est connue.

2.2 Algorithme de simulation de processus de Poisson non-homogènes.

Grâce au théorème précédent nous savons que si nous générons un premier processus de Poisson dont la fonction d'intensité est telle que $\forall t \lambda(t) \leq \lambda^*(t)$. Alors, si nous simulons une variable aléatoire U suivant une loi de bernoulli avec $\mathbf{P}(U = 1) = \frac{\lambda}{\lambda^*}$, et telle que si $U = 1$ nous

conservons T_i^*, et si $U = 0$ nous rejettons celui-ci. Il ne reste qu'à concaténer la liste des T_i^* restant.

Algorithm 1: Processus de Poisson non-homogène unidimensionnel

Data: x_0, λ, λ^*
Result: $x_{\text{list}}, y_{\text{list}}$
$x = 0$,
$x_{\text{list}} = [0]$
$y_{\text{list}} = [0]$
$X^*_{\text{list}} = [0]$
$y^*_{\text{list}} = [0]$
i = 1 **while** $X^* < x_0$ **do**
    $D = \text{np.random.uniform}(0, 1)$
    $U = \text{np.random.uniform}(0, 1)$
    $e = -\text{np.log}(U)/(\lambda \times X^*[i])$
    $X^*.\text{append}(X^{star}[i-1] + e)$
    $Y^*.\text{append}(D * \lambda^*(X^*[i]))$
    if $D \leq \frac{\lambda(X^*[i])}{\lambda^*(X^*[i])}$ **then**
        $x_{\text{list}}.\text{append}(X^*[i])$
        $y_{\text{list}}.\text{append}(Y^*[i])$
    $i = i + 1$
$n = \text{len}(x_{\text{list}})$

Pour cet algorithme on utilise un algorithme de simulation de variable aléatoire qui nous permet de simuler une variable aléatoire exponentielle. Cette variable aléatoire qui suit une loi exponentielle de paramètre $\lambda^*(t)$. Il faut commencer par genérer les loi exponentielles de manière cumulative de façon à ce que la liste soit naturellement ordonnée. Pour cela nous utilisons un algorithme de simulation de variables aléatoires classique :

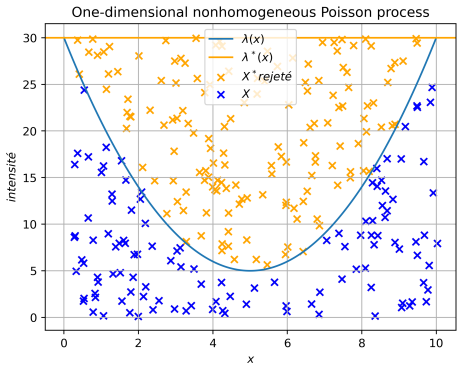
— Nous simulons une variable aléatoire uniforme U sur [0,1]
— Nous calculons $F^{-1}(U)$

Dans notre cas F est la fonction de répartition d'une loi exponentielle.

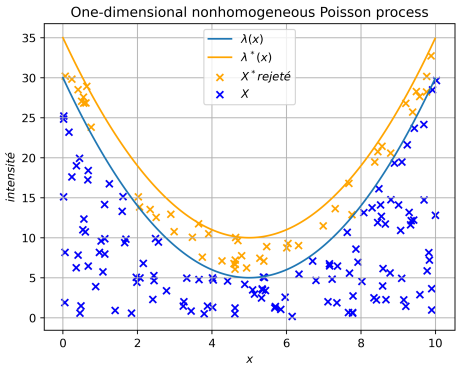
Ensuite nous génèrons une deuxième variable aléatoire uniforme sur $[0; 1]$. Selon le Théorème 1 nous savons que si nous effectuons une selection, selon une loi de bernoulli, des variables du processus de Poisson que nous avons simulé juste avant, le processus qui en resultera sera toujours un processus de Poisson. C'est ce que nous faisons au niveau de la condition $D \leq \frac{\lambda(X^*[i])}{\lambda^*(X^*[i])}$. Ainsi ce resultat nous assure bien que le processus en sortie de l'algorithme est toujours un processus de Poisson et qu'il est de fonction d'intensité $\lambda(t)$.

La fonction $\lambda^*(t)$ la plus simple que nous pouvons utiliser pour mettre en place cet algorithme est la fonction constante $\lambda^*(t) = max(\lambda(t); t \in [0, T])$, c'est à dire que T_n^* est homogène. Il est important de noter que la probabilité de rejet est plus faible si les deux fonctions d'intensités on des valeurs proches en tous points.

Exemple 2. *Voici un exemple de simulations de processus de Poisson de fonction d'intensité* $\lambda(t) = 5 + (t-5)^2$.



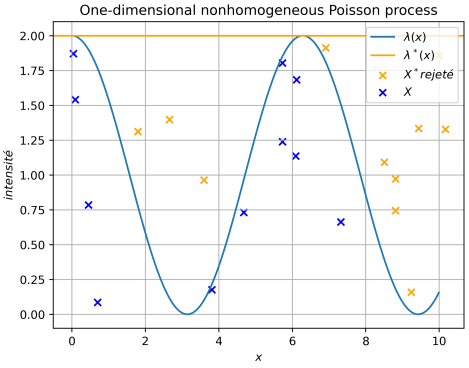
(a) $\lambda = 5 + (t-5)^2$, $\lambda^* = max(\lambda(t)) = 30$



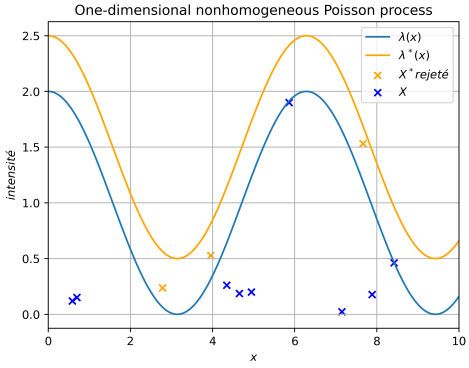
(b) $\lambda = 5 + (t-5)^2$, $\lambda^* = 10 + (t-5)^2$

FIGURE 1 – Algorithme 1

Exemple 3. *Voici un autre exemple en utilisant cette fois-ci une fonction trigonométrique ici* $\lambda(t) = 1 + cos(t)$.



(a) $\lambda = 1 + cos(t)$, $\lambda^* = max(\lambda(t)) = 2$



(b) $\lambda = 1 + cos(t)$, $\lambda^* = 1.5 + cos(t)$

FIGURE 2 – Algorithme 1

3 Simulation de processus de Poisson homogène en dimension 2.

Dans cette section, nous définissons d'abord ce qu'est le processus de Poisson homogène en dimension 2, puis, après avoir utilisé quelques théorèmes qui caractérisent la restriction à des surfaces R telles que des rectangles ou encore des cercles de ce dernier, nous proposons des implémentations via Python de ces méthodes.

3.1 Le processus de Poisson homogène en dimension 2.

Définition 4. *Le processus de Poisson non-homogène $\mathcal{P}$ en dimension 2 (d'intensité $\lambda > 0$) est défini par les deux propriétés suivantes : - Premièrement, si l'on se munit d'une certaine région $\mathcal{R}$ du plan qui possède une aire donnée, que l'on note par exemple $\mathcal{A}$, alors le nombre de points du processus dans cette région $\mathcal{R}$ suit une loi de Poisson de paramètre $\lambda\mathcal{A}$. - Deuxièmement, étant donné deux régions du plan $\mathcal{R}_1$ et $\mathcal{R}_2$ disjointes, alors le nombre de points du processus de Poisson dans ces régions sont indépendants. Cette deuxième propriété est également généralisable à un ensemble fini de régions du plan disjointes.*

Définition 5. *L'intensité d'un processus de Poisson est un paramètre qui mesure la fréquence moyenne à laquelle des événements se produisent dans une région donnée de l'espace $\mathcal{R}$. On le note souvent $\lambda > 0$.*

Exemple 4. *Les applications de ce processus sont multiples, on pourraît penser par exemple à des modélisations d'apparition d'arbres dans un jardin de surface rectangulaire, ou encore, d'apparition de séismes (si on groupe un séisme avec ses répliques pour garder l'indépendance bien-sûr) sur une surface relativement circulaire comme une ville.*

Selon les auteurs de l'article que nous étudions dans ce projet, il est relativement facile d'étudier la projection du processus de Poisson en dimension 2 sur des surfaces simples telles que des rectangles ou des cercles et d'en faire des simulations.

Nous définissons maintenant quelques théorèmes qui permettent de générer des processus de Poisson dans une surface rectangulaire et circulaire.

3.2 Le processus de Poisson homogène dans un rectangle.

Le premier théorème caractérise les projections du processus de Poisson homogène en dimension 2 sur chacun des axes d'une surface rectangulaire (qui sont alors toujours des processus de Poisson mais de dimension 1). Il est important de constater que les rectangles considérés par la suite ont le coin inférieur gauche aux coordonnées $(0, 0)$.

Théorème 3 (P. A. W. Lewis p408.). *Considerons un processus de Poisson homogène en dimension 2 d'intensité λ. D'après la Définition 1, le nombre de points dans une surface rectangulaire $\mathcal{R} = \{(x, y) : 0 < x \leq x_0, 0 < y \leq y_0\}$ suit une loi de Poisson de paramètre $\lambda \times x_0 \times y_0$. Si l'on note $(X_1, Y_1), \ldots, (X_N, Y_N)$ les coordonnées du processus dans $\mathcal{R}$, que l'on a réordonné de façon à ce que $X_1 < X_2 < \cdots < X_N$, alors $X_1, X_2, \ldots, X_N$ forment un processus de Poisson homogène de dimension 1 sur $(0 < x \leq x_0)$ d'intensité $\lambda \times y_0$. Si maintenant les points sont renommés $(X'_1, Y'_1), \ldots, (X'_N, Y'_N)$ avec cette fois-ci $Y'_1 < Y'_2 < \cdots < Y'_N$, alors $Y'_1, Y'_2, \ldots, Y'_N$ forment un processus de Poisson homogène en dimension 1 sur $(0 < y \leq y_0)$ d'intensité $\lambda \times x_0$.*

Le second théorème permet d'établir les propriétés d'indépendance conditionnelle du processus de Poisson. C'est celui qui nous sera utile pour réaliser les algorithmes de simulations dans la prochaine sous partie.

Théorème 4 (P. A. W. Lewis p408.). *Considérons qu'un processus de Poisson homogène en dimension 2 d'intensité λ soit observé dans un rectangle $\mathcal{R} = \{(x, y) : 0 \leq x \leq x_0, 0 < y \leq y_0\}$. On sait que le nombre de points dans $\mathcal{R}$ que l'on note alors $N(\mathcal{R})$ suit une loi de Poisson de paramètre $\lambda \times x_0 \times y_0$. Si l'on suppose que $N(\mathcal{R}) = n > 0$ et que l'on note $(X_1, Y_1), \ldots, (X_n, Y_n)$ les points du processus, réordonnés comme d'habitude de telle manière à ce que $X_1 < X_2 < \cdots < X_n$, alors, conditionnellement au fait d'avoir $N(\mathcal{R}) = n$, $X_1, \ldots, X_n$ sont des statistiques d'ordres uniformes sur l'axe des abscisses et les $Y_1, \ldots, Y_n$ sont indépendant et distribués également uniformément sur l'axe des ordonnées (indépendamment des X_i).*

Nous définissons la statistique d'ordre d'un échantillon aléatoire pour donner sens au théorème précédent.

Définition 6. *Soit $X_1, \ldots, X_n$ un échantillon aléatoire provenant d'une certaine distribution de probabilité de densité $f(x)$. Les statistiques d'ordres de cet échantillon sont alors les valeurs de cet échantillon ordonnées.*

Ces deux théorèmes nous permettent alors d'implémenter facilement un algorithme de simulation du processus homogène de Poisson en dimension 2 dans un rectangle. Nous en détaillons l'idée dans la partie suivante.

3.3 Algorithme pour la génération du processus dans une surface rectangulaire.

Dans cet algorithme, on se donne tout d'abord la largeur et la longueur du rectangle (x_0, y_0) dans lequel on veut simuler, et l'intensité du processus, λ.

On génère ensuite les points $X_1, \ldots, X_n$ en cumulant des lois exponentielles de paramètre $\lambda \times y_0$ (en utilisant le Théorème 3) tant qu'on ne dépasse pas la coordonnée x_0 en abscisse. Pour cela, on utilise la méthode de simulation de la loi exponentielle passant par l'inverse de la fonction de répartition de cette dernière et utilisant une variable U uniforme sur $[0, 1]$.

Algorithm 2: Processus de Poisson homogène en dimension 2 dans un rectangle

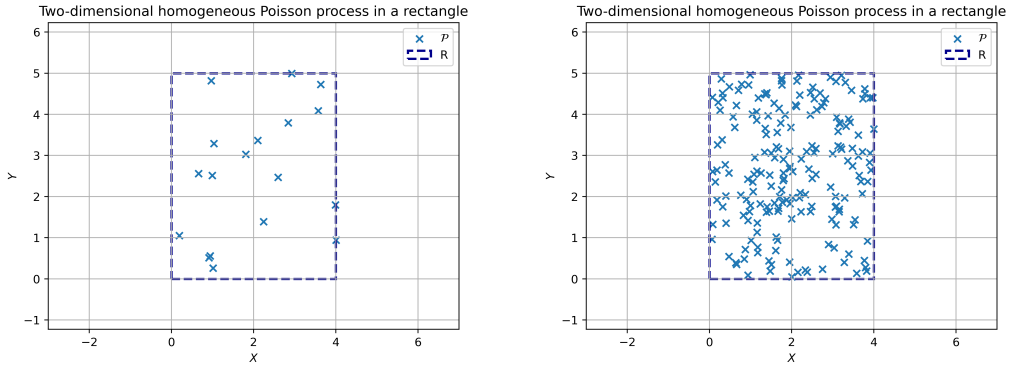
Data: x_0, y_0, λ
Result: $x_{\text{list}}, y_{\text{list}}$

$x = 0$,
x_{list}=[ ]
while $x < x_0$ **do**
    $U = \text{np.random.uniform}(0, 1)$
    $e = -\text{np.log}(U)/(\lambda \times y_0)$
    $x = x + e$
    $x_{\text{list}}.\text{append}(x)$
$x_{\text{list}}.\text{sort}()$
$n = \text{len}(x_{\text{list}})$
$y_{\text{list}} = \text{np.random.uniform}(0, y_0, n)$

Ensuite, on réordonne les points $X_1, \ldots, X_n$, puis on récupère leur nombre n car il est aléatoire et dépend des étapes précédentes, il suit une loi de Poisson.

Enfin, on génère les $Y_1, \ldots, Y_n$, uniformes sur $[0, y_0]$, d'après le Théorème 4, et on retourne le processus $(X_1, Y_1), \ldots, (X_n, Y_n)$ généré dans le rectangle défini au début.

Exemple 5. *Voici des exemples d'éxécutions de cet algorithme pour différentes intensités λ du processus, on remarque comme prévu que plus λ est grand, plus le nombre de points généré est important.*



(a) $\lambda = 1$, $(x_0, y_0) = (4, 5)$ (b) $\lambda = 10$, $(x_0, y_0) = (4, 5)$

FIGURE 3 – Algorithme 2

3.4 Le processus de Poisson homogène dans un cercle.

Nous définissons désormais deux théorèmes qui nous permettent de caractériser les projections du processus de Poisson homogène en dimension 2 dans une surface circulaire de rayon r_0 sur chacune des coordonnées.

Théorème 5 (P. A. W. Lewis p410.)**.** *Considérons un processus de Poisson homogène en dimension 2 d'intensité λ. Le nombre $N(\mathcal{C})$ de points dans un cercle $\mathcal{C}$ de rayon r_0 et d'aire πr_0^2 suit une loi de Poisson de paramètre $\lambda \times \pi \times r_0^2$. Notons $(R_1, \theta_1), \ldots, (R_N, \theta_N)$, les points de ce processus dans le cercle $\mathcal{C}$, que l'on a ordonné de façon à ce que $R_1 < \cdots < R_N$. Alors $R_1, \ldots, R_N$ est un processus de Poisson non homogène de dimension 1 sur $0 \leq r \leq r_0$ avec une fonction d'intensité définie par $\lambda(r) = 2\pi\lambda r$. De la même manière que pour le rectangle, si les points du processus sont renommés $(R'_1, \theta'_1), \ldots, (R'_N, \theta'_N)$ avec l'ordre $\theta'_1 < \cdots < \theta'_N$, alors $\theta'_1, \ldots, \theta'_N$ est cette fois-ci un processus de Poisson homogène de dimension 1, sur $0 < \theta \leq 2\pi$, de fonction d'intensité$\lambda r_0^2/2$.*

Ici encore, un second théorème nous permet d'établir les propriétés d'indépendance conditionnelle du processus de Poisson. C'est celui qui nous sera utile pour réaliser les algorithmes de simulations dans la prochaine sous partie.

Théorème 6 (P. A. W. Lewis p410.)**.** *Considérons qu'un processus de Poisson en dimension 2 homogène d'intensité λ soit observé dans une surface circulaire $\mathcal{C}$ de rayon r_0. Le nombre $N(\mathcal{C}) = n$ de points dans le cercle $\mathcal{C}$ suit une loi de Poisson de paramètre $\lambda \times \pi \times r_0^2$. Si $N(\mathcal{C}) = n > 0$, et que l'on note encore une fois les points du processus $(R_1, \theta_1), \ldots, (R_n, \theta_n)$, avec l'ordre $R_1 < \cdots < R_n$. Alors conditionnellement au fait que l'on ai observé n points dans $\mathcal{C}$, $R_1, \ldots, R_n$ forment des statistiques d'ordre de densité $f(r) = 2r/r_0^2$ sur $0 \leq r \leq r_0$, et $\theta_1, \ldots, \theta_n$ sont indépendants et uniformes sur $0 < \theta < 2\pi$ (et indépendant des R_i).*

Ces deux théorèmes nous permettent alors d'implémenter facilement un algorithme de simulation du processus homogène de Poisson en dimension 2 dans un cercle. Nous en détaillons l'idée dans la partie suivante.

3.5 Algorithme pour la génération du processus dans une surface circulaire.

L'idée de cet algorithme est entièrement basée sur le Théorème 6 comme nous allons le voir ci-dessous :

1. Dans cet algorithme, on génère tout d'abord un nombre n qui suit une loi de Poisson de paramètre $\lambda \times \pi \times r_0^2$.

2.3. Ensuite, nous créons n variables indépendantes $R_1, \ldots, R_n$ ordonnées de densité $f(r) = 2r/r_0^2$ sur $[0, r_0]$. Pour cela, nous procèdons par la méthode de l'inverse de la fonction de répartition sur une variable U uniforme sur $[0, 1]$. (Avec ici, $F^{-1}(U) = \sqrt{U} r_0$)

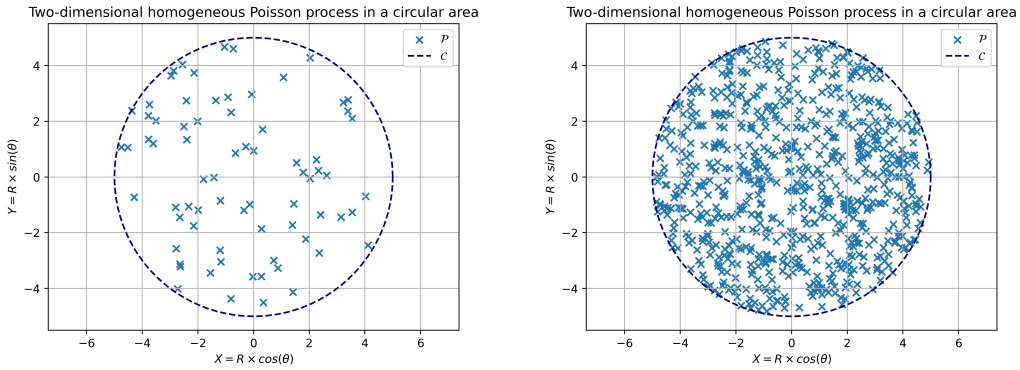
Algorithm 3: Processus de Poisson homogène en dimension 2 dans un cercle

Data: r_0, λ
Result: X, Y
$n = \text{np.random.poisson}(\lambda \pi r_0^2)$
$U = \text{np.random.rand}(n)$
$R = \sqrt{U} r_0$, np.sort(R)
$\theta = \text{np.random.uniform}(0, 2\pi, n)$
$X = R\cos(\theta)$, $Y = R\sin(\theta)$

4. On génère après cela $\theta_1, \ldots, \theta_n$ indépendants et uniformes sur $[0, 2\pi]$.

5. Pour finir, nous passons les points en coordonnées cartésiennes pour faciliter l'affichage.

Exemple 6. *Voici des exemples d'exécution de cet algorithme pour différentes intensités λ du processus, on remarque comme prévu que plus λ est grand, plus le nombre de points générés est important.*



(a) $\lambda = 1$, $r_0 = 5$ (b) $\lambda = 10$, $r_0 = 5$

FIGURE 4 – Algorithme 3

3.6 Génération du processus de Poisson homogène dans des surfaces atypiques.

La génération du processus de Poisson homogène en dimension 2 dans des surfaces non rectangulaires ou non circulaires n'est pas forcément évidente.

En effet, la méthode consistant à projeter les points du processus sur les différents axes afin d'obtenir des processus de Poisson en dimension 1 est efficace dans nos deux exemples car les processus en dimension 1 obtenus ont des fonctions d'intensités relativement simples.

Cependant, pour des régions du plan avec des géométries bien plus complexes, les fonctions d'intensité obtenues ne seront pas aussi simples, et bien souvent difficiles à définir et/ou à simuler.

Pour palier à ce problème, on pourrait par exemple décider de recouvrir la région atypique dans laquelle on veut simuler avec un cercle ou un rectangle, puis simuler un processus de Poisson dans ces derniers. Et ensuite, supprimer les points qui ne tombent pas dans la zone voulue.

Exemple 7. *Nous illustrons nos dires par un exemple. Ici, on cherche à simuler un processus dans une surface étoilée. Pour cela, on recouvre cette surface d'un cercle dans lequel on sait simuler un processus de Poisson homogène en dimension 2, puis on ne conserve que les points qui tombent dans la surface d'intérêt.*

FIGURE 5 – Processus dans la surface étoilée

4 Conclusion.

Pour conclure, nous avons vu dans la première partie un procédé permettant de simuler un processus de Poisson non homogène en dimension 1. La technique s'apparente à une méthode d'amincissement, et l'on a constaté que le choix de la fonction $\lambda^*(t)$ est important pour permettre une simulation efficace du processus souhaité et un faible nombre de points rejetés.

Par la suite, nous avons vu des techniques qui permettent de simuler des processus de Poisson homogène en dimension 2 dans des surfaces relativement simples telles que des rectangles ou des cercles.

Nous pouvons également étendre cette approche à des surfaces plus complexes, seulement en les recouvrant par des surfaces rectangulaires ou circulaires, puis en simulant le processus

dans ces dernières.

Un première ouverture intéressante serait donc d'implémenter cette simulation de processus de Poisson dans des surfaces complexes.

Par ailleurs, nous avons travaillé sur la dernière partie de l'article, qui traite de la simulation des processus de Poisson non homogènes en dimension 2. Cependant, Le choix de la fonction $\lambda^*(t)$, qui était alors cette fois-ci une fonction de deux variables, nous a posé des problèmes pour la simulation. Le code de cet algorithme est cependant en annexe.

5 Annexe.

5.1 Codes.

5.1.1 Algorithme 1.

```
def algorithme1(X_0):
    x = [0]
    y = [0]
    x_star = [0]
    y_star = [0]

    #lbd_star = 2
    i = 1
    while x_star[i-1] <X_0:

        u = np.random.uniform()
        e_intermediaire = -np.log(u)/lbd_star(x_star[i-1])
        x_star.append(x_star[i-1] + e_intermediaire)

        D = np.random.uniform()
        y_star.append(D*lbd_star(x_star[i]))

        if (D <= lbd(x_star[i])/lbd_star(x_star[i])):
            x.append(x_star[i])
            y.append(D*lbd_star(x_star[i]))
        i+=1


    return(x_star,x,y_star,y)
```

5.1.2 Algorithme 2.

```
def algorithme2(X_0,Y_0,lambd):
    x=0
    x_list=[]

    while x < X_0:
        U = np.random.uniform(0,1)
```

```
        e=-np.log(U)/(lambd*Y_0)
        x=x+e
        x_list.append(x)

    x_list.sort()
    n=len(x_list)
    y_list=np.random.uniform(0,Y_0,n)

    return x_list,y_list
```

5.1.3 Algorithme 3.

```
def algorithme3(lambd,r0):

    n = np.random.poisson(lambd*np.pi*r0**2)
    U = np.random.rand(n)
    R = np.sqrt(U)*r0
    np.sort(R)
    theta = np.random.uniform(0,2*np.pi,n)
    X=R*np.cos(theta)
    Y=R*np.sin(theta)

    return X,Y,n
```

5.1.4 Algorithme 4.

```
def lbd(X,Y):
    return np.exp(-(X+Y))

def algorithme4(x,y,X_0,Y_0):

    lbd_star = 34/4
    X_star,Y_star = algorithme2(X_0,Y_0,lbd_star)

    X = []
    Y = []

    X_star2 = []
    Y_star2 = []
    for i in range(len(X_star)):
        if (X_star[i]<x) and (Y_star[i]<y):
            X_star2.append(X_star[i])
            Y_star2.append(Y_star[i])


    i = 0
    while i < len(X_star2):
        U = np.random.uniform()
        if (U <= lbd(X_star2[i],Y_star2[i])/lbd_star):
            X.append(X_star2[i])
            Y.append(Y_star2[i])
        i += 1
```

```
    return(X,Y,X_star,Y_star)
```