CentraleSupélec

université PARIS-SACLAY

Rapport Pôle Projet P15

Modélisation Mathématiques des Systèmes Complexes
Sujet No. 2
Non Stationary Point Process

Réalisé dans le cadre du pôle P15
Louis Rouxel
Amélie Bellazi
Arnaud Van de Velde
Gavriil Kharkhordine
Dillan Reggane

24 septembre 2025

Table des matières

Chapitre 1

Définitions et contexte

Les documents [3] [4] ont permis de définir les différents objets et concepts mathématiques utiles.

1.1 Définition d'un processus

Définition 1.1.1 (Processus ponctuel). *Soit $(\Omega, \mathcal{F}, P)$ un espace de probabilité. Soit $(t_i)_{i\in\mathbb{N}^*}$ une suite de variables aléatoires non négatives telles que $\forall i \in \mathbb{N}^*$, $t_i < t_{i+1}$. On appelle $(t_i)_{i\in\mathbb{N}^*}$ un processus ponctuel sur $\mathbb{R}^+$.*

En particulier, les variables t_i peuvent représenter les instants d'occurrence de transactions financières, d'arrivées d'ordres dans un carnet d'ordres, etc. Nous commençons à compter les événements avec l'indice 1. Si nécessaire, nous supposons que $t_0 = 0$.

Définition 1.1.2 (Processus de comptage). *Un processus aléatoire $\{N_t; t \geq 0\}$ à valeurs entières est un processus de comptage si*

i) $N_0 = 0$;
ii) $\forall s \leq t, \quad N_s \leq N_t$.

On définit les tops dans un processus de comptage comme les instants où un événement se produit, c'est-à-dire les moments où le processus N_t augmente d'une unité. On peut alors interpréter la différence $N(b) - N(a)$ comme le nombre de tops se produisant dans l'intervalle $]a, b]$

Définition 1.1.3 (Processus de comptage associé). *Soit $(t_i)_{i\in\mathbb{N}^*}$ un processus ponctuel. Le processus càd-làg défini par*

$$N(t) = \sum_{i\in\mathbb{N}^*} \mathbb{1}_{t_i \leq t} \tag{1.1}$$

est appelé le processus de comptage associé à $(t_i)_{i\in\mathbb{N}^*}$. Ce dernier compte le nombre d'occurences avant l'instant t.

Définition 1.1.4 (Durée inter-événements). *Le processus $(\delta t_i)_{i\in\mathbb{N}^*}$ défini par*

$$\forall i \in \mathbb{N}^*, \quad \delta t_i = t_i - t_{i-1} \tag{1.2}$$

est appelé le processus de durées associé à $(t_i)_{i\in\mathbb{N}^*}$.

Définition 1.1.5 (Processus d'intensité). *Soit N un processus ponctuel adapté à une filtration $\mathcal{F}_t$. Le processus d'intensité est défini par*

$$\lambda(t|\mathcal{F}_t) = \lim_{h\to 0} \mathbb{E}\left[\frac{N(t+h) - N(t)}{h}\middle|\mathcal{F}_t\right], \tag{1.3}$$

ou de manière équivalente,

$$\lambda(t|\mathcal{F}_t) = \lim_{h\to 0} \frac{1}{h}\mathbb{P}\left[N(t+h) - N(t) > 0|\mathcal{F}_t\right]. \tag{1.4}$$

L'intensité désigne donc la fréquence instantanée d'occurrence des événements dans un processus ponctuel N. En fait λ joue un rôle différent en fonction de ce qu'on étudie. Dans le cas d'une loi de Poisson, λ est le nombre moyen d'événements dans un intervalle donné. Par exemple, si une station-service reçoit en moyenne λ=5 clients par heure, alors le nombre de clients observés en une heure suit une loi de Poisson de paramètre λ=5. Dans le cas d'un processus stochastique de Poisson, c'est différent. Ce dernier compte le nombre d'événements survenus jusqu'à l'instant t. Si le processus est homogène, le paramètre λ est une intensité constante, et il représente le taux moyen d'événements par unité de temps :

$$P(N(t+h) - N(t) = 1) = \lambda h + o(h).$$

— λ *est* ***le nombre moyen d'événements par unité de temps****.*
— *Par exemple, si des clients arrivent dans un magasin à un rythme de* $\lambda = 5$ ***clients par heure****, alors :*
  — *En* ***2 heures****, on attend en moyenne* ***10 clients*** *(*$\lambda \times t$*).*
  — *L'arrivée des clients est* ***aléatoire****, et le temps entre deux arrivées suit une* ***loi exponentielle*** *de paramètre* λ*.*

Toutefois, si le processus est non homogène, l'intensité $\lambda(t)$ ***dépend du temps*** *:*

$$P(N(t+h) - N(t) = 1) = \lambda(t)h + o(h). \tag{1.5}$$

— *Ici,* $\lambda(t)$ *peut* ***varier au cours du temps*** *pour modéliser des phénomènes* ***saisonniers ou cycliques****.*
— *Par exemple, en finance, le flux d'ordres dans un carnet de marché est souvent* ***plus intense à l'ouverture et à la clôture*** *des marchés, donc* $\lambda(t)$ *est plus élevé à ces moments.*

L'intensité dépend du choix de la filtration, mais on suppose ici que la filtration naturelle du processus N *est utilisée, notée* $\mathcal{F}_t^N$*. On écrira donc simplement* $\lambda(t)$ *au lieu de* $\lambda(t|\mathcal{F}_t^N)$*.*

Définition 1.1.6 (Processus de Poisson homogène)**.** *Soit* $\lambda \in \mathbb{R}_+^*$*. Un processus de Poisson de taux constant* λ *est un processus ponctuel défini par :*

$$\mathbb{P}\left[N(t+h) - N(t) = 1 | \mathcal{F}_t\right] = \lambda h + o(h), \tag{1.6}$$

$$\mathbb{P}\left[N(t+h) - N(t) > 1 | \mathcal{F}_t\right] = o(h). \tag{1.7}$$

Propriétés :
— *L'intensité ne dépend pas de l'historique du processus* N*, et la probabilité d'occurrence d'un événement dans* $(t, t+h]$ *est indépendante de* $\mathcal{F}_t$*.*
— *Les durées* $(\delta t_i)_{i \in \mathbb{N}^*}$ *d'un processus de Poisson homogène sont indépendantes et identiquement distribuées (i.i.d.) selon une loi exponentielle de paramètre* λ*.*

On a également la définition équivalente :

Définition 1.1.7 (Processus de Poisson simple)**.** *Un processus de comptage* $(N_t)_{t \in \mathbb{R}_+}$ *est appelé un* ***processus de Poisson simple*** *de paramètre* $\lambda \in \mathbb{R}_+^*$ *lorsque :*

1. *Le processus a des accroissements indépendants, c'est-à-dire que pour tous* $s, t \in \mathbb{R}_+^*$*, la variable aléatoire* $N_{t+s} - N_s$ *est indépendante de* $\sigma(N_u \mid u \in [0, s])$*.*
2. *Pour tous* $s, t \in \mathbb{R}_+$*, la variable aléatoire* $N_{t+s} - N_s$ *suit une loi de Poisson de paramètre* λt*.*

Définition 1.1.8 (Processus de Poisson non homogène)**.** *Soit* $\lambda : \mathbb{R}_+ \to \mathbb{R}_+$ *une fonction localement intégrable et* $\Lambda : \mathbb{R}_+ \to \mathbb{R}_+$ *la fonction définie par*

$$\Lambda(t) = \int_0^t \lambda(s) ds. \tag{1.8}$$

Un processus de comptage $(N_t)_{t \geq 0}$ *est appelé processus de Poisson d'intensité* λ *lorsque :*

1. *Le processus est à accroissements indépendants.*
2. *Pour tous* $s < t \in \mathbb{R}_+$*,* $N_t - N_s$ *suit une loi de Poisson de paramètre* $\Lambda(t) - \Lambda(s)$*.*

— On a $\mathbb{E}[N_t] = \mathrm{Var}(N_t) = \Lambda(t)$.
— Le processus de Poisson simple correspond à une intensité constante au cours du temps, c'est-à-dire $\lambda(t) = \lambda$ pour tout $t \in \mathbb{R}_+$.

La construction d'un *processus de Poisson non homogène* peut se faire à partir d'un processus de Poisson simple par une modification de l'échelle de temps. **Si** N **est un processus de Poisson simple d'intensité 1, alors le processus** $\tilde{N}$ **défini par** $\tilde{N}_t = N(\Lambda(t))$ **est un processus de Poisson non homogène d'intensité** λ**.** Notons

$$\Lambda^{-1}(t) = \inf\{x \in \mathbb{R}_+ \mid \Lambda(x) \geq t\} \tag{1.9}$$

la pseudo-inverse continue à gauche de Λ et posons $S_n = \Lambda^{-1}(T_n)$ pour $n \in \mathbb{N}^*$, où T_n est le temps du n-ième saut de N. Alors on a

$$\tilde{N}_t = \sum_{n \in \mathbb{N}^*} \mathbb{1}_{\{S_n \leq t\}}, \quad t \in \mathbb{R}_+. \tag{1.10}$$

On peut alors réaliser un test pour vérifier qu'une simulation est bien adaptée à des données qui suivraient un processus de poisson.

Théorème 4 (Théorème du changement de temps aléatoire)
Soit $\{t_1, t_2, \ldots, t_k\}$ une réalisation sur l'intervalle $[0, T]$ d'un processus ponctuel avec une fonction d'intensité conditionnelle $\lambda^*(\cdot)$.
Si $\lambda^*(\cdot)$ est strictement positive sur $[0, T]$ et que $\Lambda(T) < \infty$ presque sûrement, alors les points transformés

$$\{\Lambda(t_1), \Lambda(t_2), \ldots, \Lambda(t_k)\}$$

forment un processus de Poisson d'intensité unitaire.
Le théorème du changement de temps aléatoire est fondamental dans la procédure d'ajustement de modèle appelée *analyse des résidus* (des processus ponctuels). Si ce test initial est concluant, alors les temps inter-arrivées,

$$\{\tau_1, \tau_2, \tau_3, \ldots\} = \{t_1^*, t_2^* - t_1^*, t_3^* - t_2^*, \ldots\},$$

doivent être testés pour vérifier que $\tau_i \overset{\text{i.i.d.}}{\sim} \text{Exp}(1)$. Une approche qualitative consiste à tracer un graphique quantile–quantile (Q–Q) pour les τ_i en utilisant la distribution exponentielle.

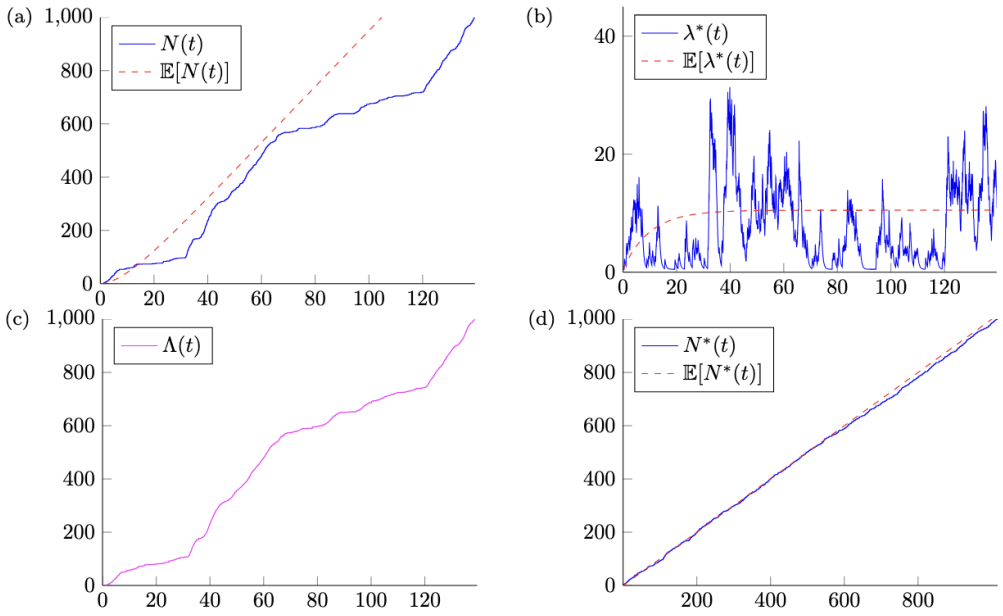

Fig. 6: An example of using the random time change theorem to transform a Hawkes process into a unit rate Poisson process. (a) A Hawkes process $N(t)$ with $(\lambda, \alpha, \beta) = (0.5, 2, 2.1)$, with the associated (b) conditional intensity function and (c) compensator. (d) The transformed process $N^*(t)$, where $t_i^* = \Lambda(t_i)$.

FIGURE 1.1 – Un exemple d'utilisation du théorème du changement de temps aléatoire pour transformer un processus de Hawkes en un processus de Poisson d'intensité unitaire, d'après Patrick J. Laub [1].

On peut imaginer que l'intensité des arrivées à un guichet subit une modification brusque au cours du temps : $\lambda(t) = \lambda_1$ si $t \leq t_0$ et $\lambda(t) = \lambda_2$ si $t > t_0$.

En fiabilité, lorsque un matériel subit une succession de pannes et lorsque chaque panne est réparée instantanément, on utilise souvent un *processus de Weibull*, c'est-à-dire un processus de Poisson non homogène d'intensité :

$$\lambda(s) = \frac{\beta}{\alpha}\left(\frac{s}{\alpha}\right)^{\beta-1}, \quad s \in \mathbb{R}_+. \tag{1.11}$$

On peut alors montrer que le premier instant de saut $S_1 = \Lambda^{-1}(T_1)$ a pour loi la loi de Weibull de paramètres α et β dont la densité est donnée par

$$f(t) = \frac{\beta}{\alpha}\left(\frac{t}{\alpha}\right)^{\beta-1} \exp\left(-\left(\frac{t}{\alpha}\right)^{\beta}\right) \mathbb{1}_{t\geq 0}. \tag{1.12}$$

De plus, si $\beta = 1$ (taux de panne constant dans le temps), on retrouve la loi exponentielle de paramètre $\lambda = \frac{1}{\alpha}$ et donc le processus de Poisson. Si $\beta > 1$, le taux de défaillance augmente avec le temps.

Définition 1.1.9 (Processus de Hawkes). Un processus de Hawkes est un processus de comptage $(N_t)_{t\geq 0}$ dont l'intensité conditionnelle $\lambda(t)$ dépend du passé du processus. Il est défini par :

$$\lambda(t) = \mu + \sum_{s<t} \varphi(t-s) dN_s$$

où :
— $\mu > 0$ est le taux de base du processus, représentant l'intensité spontanée d'occurrence d'événements.
— $\varphi : \mathbb{R}_+ \to \mathbb{R}_+$ est une fonction de mémoire, aussi appelée *noyau d'influence*, qui modélise l'influence des événements passés sur l'intensité actuelle.

Simulation de processus de Poisson

Afin de simuler notre processus de Poisson on se basera principalement sur cette définition :

$$P(N(t+h) - N(t) = 1) = \lambda(t)h + o(h). \tag{1.13}$$

1. Processus de Poisson homogène

Deux implémentations sont proposées :

— **Version directe avec NumPy :**
Cette version génère directement N échantillons de la variable aléatoire $\mathcal{P}(\lambda T)$:

```
def Poisson_sim(T, N, lam):
    return np.random.poisson(lam * T, N)
```

— **Version par simulation d'inter-arrivées :**
Cette version utilise la propriété que les temps entre événements suivent une loi exponentielle de paramètre λ. On les accumule jusqu'à atteindre la durée totale T.

```
def Poisson_sim2(T,N,lam):
  Table= []
  eps=T/N
  for i in range(N):
    if rd.random()<lam*eps:
      Table.append(i*eps)
  return Table
```

2. Processus de Poisson non homogène

Un processus de Poisson non homogène permet de modéliser un taux d'arrivée d'événements variable au cours du temps : $\lambda = \lambda(t)$.

Définition du taux $\lambda(t)$ par interpolation

Dans notre cas, $\lambda(t)$ est estimé à partir de données empiriques (par exemple, une moyenne d'événements observés) via interpolation linéaire :

```
def ft(t, T=x_smooth , N=y_smooth):
    f = interp.interp1d(T, N, kind='linear',
                        bounds_error=False,
                        fill_value=(0, N[-1]))
    return f(t)

def lam(t):
    return ft(t)
```

Simulation du processus

On utilise ici une méthode d'inter-arrivées, similaire à la version homogène, mais en évaluant dynamiquement le taux $\lambda(t)$:

```
def Poisson_non_homo(T,N,lam):
    eps=T/N
    Table=[]
    for i in range(N):
        if rd.random()<lam(i*eps)*eps:
            Table.append(i*eps)
    return Table
```

Ce modèle permet de mieux coller à la réalité observée lorsque les événements ne se produisent pas à un rythme constant, mais varient en fonction de l'heure ou d'autres facteurs externes.

Exemple d'utilisation

Voici un exemple illustrant la simulation de plusieurs trajectoires et l'interpolation du taux $\lambda(t)$:

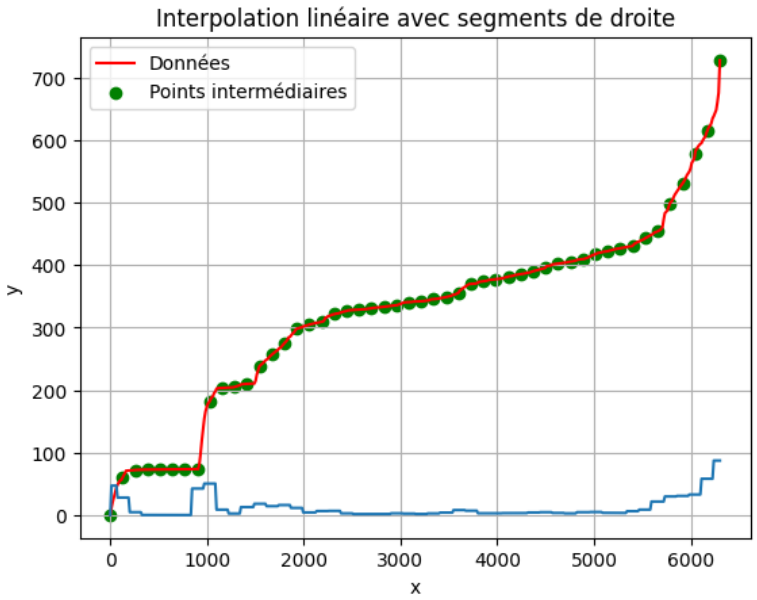

FIGURE 1.2 – Simulation de 5 courbes à partir d'un processus de Poisson non homogène

La figure ci-dessus montre cinq réalisations d'un processus de Poisson non homogène simulé à l'aide d'un taux variable $\lambda(t)$. On observe que l'accumulation des événements n'est pas régulière, reflétant la nature dynamique du taux d'arrivée.

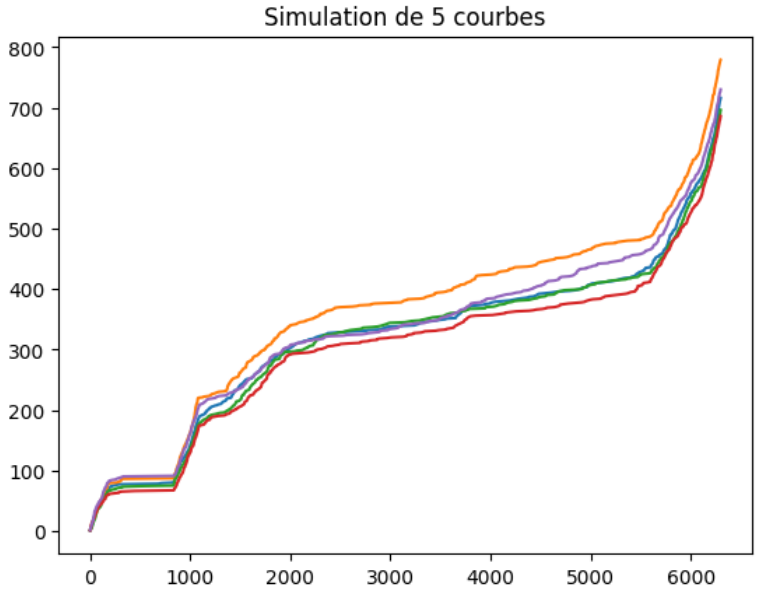

FIGURE 1.3 – Interpolation linéaire du taux $\lambda(t)$ à partir de données empiriques

La seconde figure illustre comment $\lambda(t)$ est construit à partir de données discrètes par interpolation linéaire. Les segments rouges relient les points d'observation, tandis que les points verts représentent des valeurs interpolées utilisées pour la simulation.

Définition 1.1.9 (Processus de Hawkes)**.** *Un* ***processus de Hawkes*** *est un processus de comptage $(N_t)_{t\geq 0}$ dont l'intensité conditionnelle $\lambda(t)$ dépend du passé du processus. Il est défini par :*

$$\lambda(t) = \mu + \int_0^t \phi(t-s)dN_s,$$

où :

- *$\mu > 0$ est le taux de base du processus, représentant l'intensité spontanée d'occurrence d'événements.*
- *$\phi : \mathbb{R}_+ \to \mathbb{R}_+$ est une fonction de mémoire, aussi appelée **noyau d'influence**, qui modélise l'influence des événements passés sur l'intensité actuelle.*

Un processus de Hawkes est dit :

- ***Excitateur*** *si $\phi \geq 0$, c'est-à-dire que chaque événement augmente temporairement l'intensité du processus.*
- ***Inhibiteur*** *si $\phi \leq 0$, signifiant qu'un événement réduit temporairement la probabilité d'apparition de nouveaux événements.*

Si l'on note $t_1, \ldots, t_k$ les réalisations du processus $N(\cdot)$ sur l'intervalle $[0, T]$, alors la fonction de vraisemblance L de $N(\cdot)$ est donnée par :

$$L = \prod_{i=1}^{k} \lambda^*(t_i) \exp\left(-\int_0^T \lambda^*(u)du\right)$$

1.2 Librairies Python pour l'estimation des processus de Hawkes

L'estimation des paramètres d'un processus de Hawkes repose principalement sur des techniques de *Maximum de Vraisemblance* et des approches basées sur l'*Expectation-Maximization (EM)*. Plusieurs librairies Python permettent de modéliser et d'estimer ces processus.

1.2.1 Tick

`Tick` est une bibliothèque optimisée pour la modélisation des processus de Hawkes. Elle propose des outils pour :

- Générer des réalisations de processus de Hawkes.
- Estimer les paramètres à l'aide du maximum de vraisemblance.
- Travailler avec différents noyaux $\phi(t)$: exponentiels, power-law, et plus complexes.

Installation :

```
pip install tick
```

Exemple d'utilisation :

```
from tick.hawkes import HawkesExpKern
hawkes = HawkesExpKern(decay=1.5, baseline=[0.3], adjacency=[[0.5]])
hawkes.fit(event_times)
```

1.2.2 HawkesLib

`HawkesLib` est une autre bibliothèque permettant l'estimation des processus de Hawkes avec des méthodes avancées comme l'optimisation par descente de gradient.

Plan du travail en cours / à faire (18/03/2025) :
- Afficher les data réelles sur Python : fait
- Générer des réalisations de processus de Hawkes de 3 façons différentes à l'aide de la librairie Python Hawkes : estimations des paramètres μ, α et β (en supposant que $f(t) = \alpha \cdot \beta \exp(-t\beta)$ (hypothèse bibliothèque Hawkes)) de 3 façons différentes :
  - Modèle constant par morceaux (μ constant sur chaque sous-intervalle du temps (1 seconde ici)).
  - Modèle linéaire : μ est linéaire par morceaux.
  - Modèle log-linéaire : suit un modèle exponentiel : $\log(\mu(t)) = a + bt$ (permet de mieux capturer les tendances de croissance rapide car suppose que l'intensité croît ou décroît de manière exponentielle).
- Déterminer la meilleure méthode en regardant laquelle minimise le critère AIC tout en maximisant la log-vraisemblance.
- Autre piste possible : Travailler avec différents noyaux f(t) : somme d'exponentielles, power-law, et plus complexes (regarder les autres expressions possibles pour f

Questions :
- Est ce que le chemin suivi est le bon ?
- Comment déterminer f nous même ?
- OBJ à terme : trouver un modèle qui reproduit au mieux les courbes d'ouverture et de fermetures que l'on vérifiera sur des données test.
-

1.2.3 Statsmodels et Scipy

Bien que `statsmodels` et `scipy` ne contiennent pas directement des implémentations des processus de Hawkes, ils fournissent des outils d'optimisation et d'estimation qui peuvent être utilisés pour ajuster les modèles sur des données réelles.

Analyse du comportement des courbes

Cette section a pour objectif d'analyser la structure temporelle des courbes issues de séries d'événements, en évaluant leur similarité et leur dispersion à travers le temps.

Visualisation globale

On remarque graphiquement que les courbes des ouvertures et de fermetures d'enchere ont des comportements similaires.

Normalisation des courbes

Afin de comparer des séries ayant des échelles différentes, chaque courbe $C_i(t)$ a été normalisée par sa valeur finale :

$$\tilde{C}_i(t) = \frac{C_i(t)}{C_i(T_{\text{fin}})}$$

où :
- $C_i(t)$ est la courbe brute à l'instant t,
- $C_i(T_{\text{fin}})$ est la valeur finale de la courbe,
- $\tilde{C}_i(t)$ est la courbe normalisée.

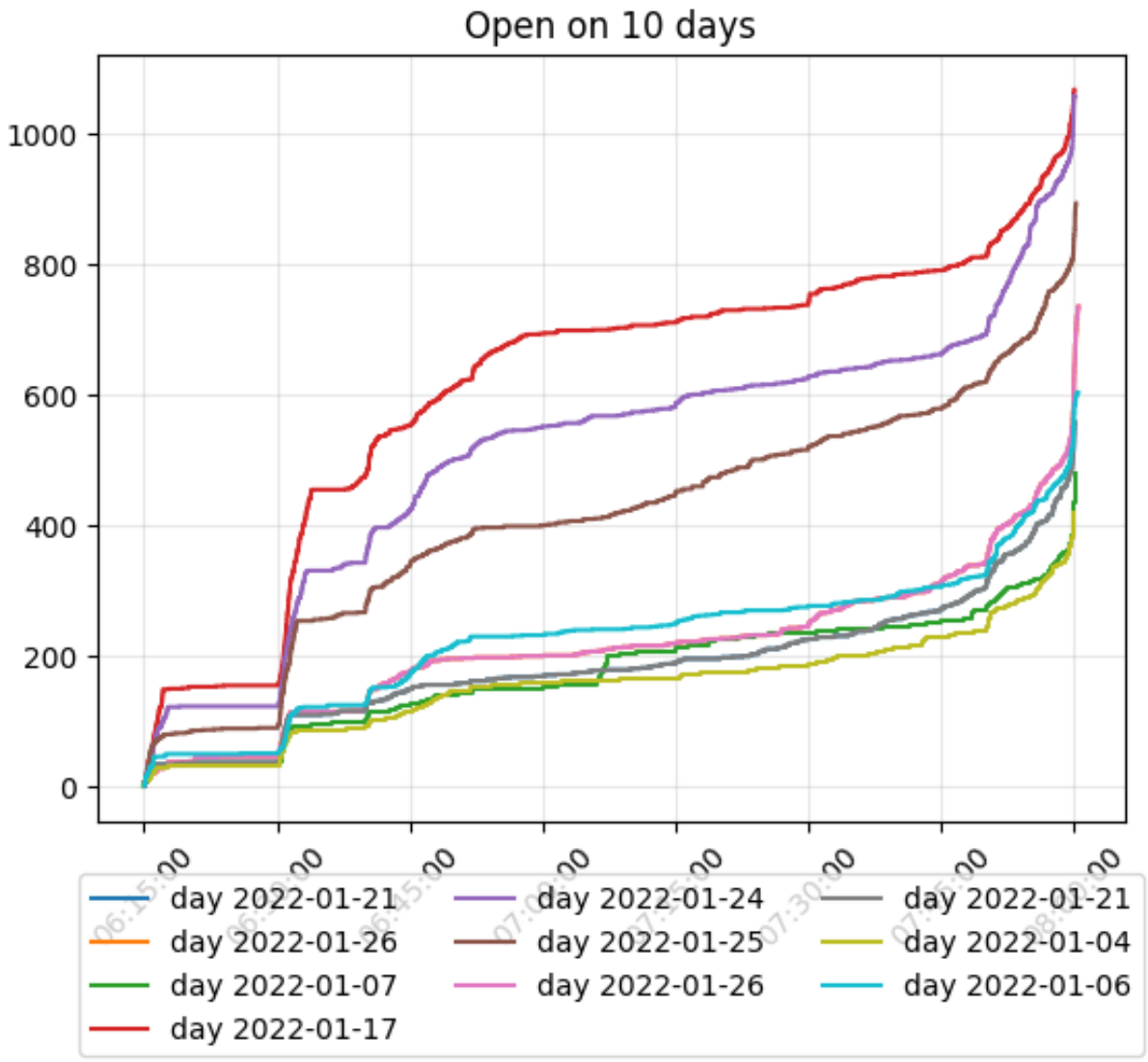

FIGURE 1.4 – Nombre d'ordre en fonction du temps pour differents jours

Cette opération permet de centrer l'analyse sur la forme des courbes, indépendamment de leur amplitude.

Code Python correspondant :

```
# Normalisation par la dernière valeur non nulle
normalized = raw_curve / (raw_curve[-1] + 1e-9)
```

Mesures statistiques

À chaque instant t, les courbes normalisées permettent de calculer les statistiques suivantes :

$$\text{Moyenne :} \quad \mu_t = \frac{1}{N}\sum_{i=1}^{N} x_{i,t}$$

$$\text{Écart-type :} \quad \sigma_t = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_{i,t} - \mu_t)^2}$$

$$\text{Coefficient de variation :} \quad \mathrm{CV}_t = \frac{\sigma_t}{\mu_t + \varepsilon}$$

où ε est une petite constante (ex. 10^{-9}) pour éviter une division par zéro.

Extrait de code :

```
mean_curve = np.mean(norm_curves, axis=0)
std_curve = np.std(norm_curves, axis=0)
coeff_variation = std_curve / (mean_curve + 1e-9)
```

Interprétation des résultats

Le graphique ci-dessous illustre :
— En haut : les courbes normalisées (en bleu), la moyenne (en rouge) et la zone d'incertitude ±1 écart-type (en rose).
— En bas : l'évolution de l'écart-type absolu au cours du temps.

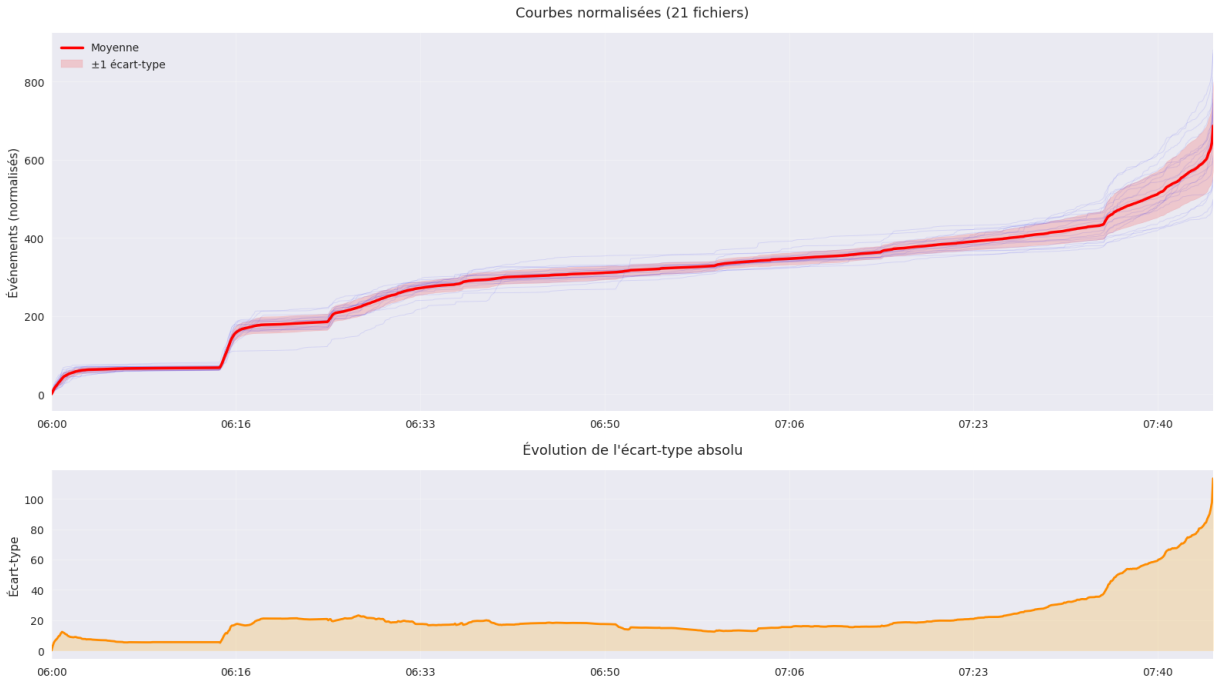

FIGURE 1.5 – Courbes normalisées avec moyenne et écart-type

Les courbes analysées présentent une dynamique temporelle similaire, comme le montre la faible dispersion autour de la moyenne. Le coefficient de variation moyen est également faible, ce qui indique que la majorité des courbes diffèrent principalement par un facteur de proportionnalité.

Cela signifie que, malgré des niveaux absolus différents, les courbes partagent une forme temporelle commune.

Conclusion : les courbes sont *similaires à un coefficient multiplicatif près*, ce qui valide l'utilisation de la normalisation pour cette analyse.

1.3 État de l'art

Cette section contient un paragraphe d'environ 10-15 lignes sur l'**état de l'art** [1] *sur la modélisation du système considéré*, en citant des références existantes dans la littérature ou même des vidéos (fournir la liste de références en faisant attention à la manière automatique pour générer les références et les citations). La façon permettant de générer des références sous Word peut paraitre compliquée, c'est une des raisons pour laquelle la plupart des chercheurs préfèrent utiliser LaTeX pour les rapports scientifiques [2].

1.4 Contribution

Dans tout projet, il est important de mettre en valeur l'**originalité** (apport/contribution) de vos approches/démarches.

1. Un conseil pour la partie état de l'art est de ne pas oublier le *fil conducteur*.
2. Plusieurs templates existent sur LaTeX, par exemple sur https ://fr.overleaf.com/latex/templates/template-for-a-project-report-or-memoire/zcgzvcmrsxsb

1.5 Approximation de la Baseline par réseau de neurones

Différentes méthodes sont evisageables pour déterminer une approximation de la baseline, censée donner la forme générale de la courbe.

— Approximation par morceaux puis minimisation de la fonction perte
— Approximation par réseau de neurones
—

Nous avons d'abord opté pour **l'approximation par réseau de neurones**. Un réseau de neurones artificiels (ANN) est un modèle d'apprentissage automatique inspiré du cerveau humain, capable de reconnaître des motifs complexes et de modéliser des relations non linéaires entre des entrées et des sorties. Il est composé de 3 couches :

— **Couche d'entrée** : Reçoit les données d'entrée (features)
— **Couches cachées** : Transforment les données via des poids, biais, et fonctions d'activation
— **Couches de sortie** : Fournit la prédiction finale (classification, régression, etc.)

Un neurone est l'unité de base d'un réseau de neurones. Il reçoit des entrées, les pondère, les additionne, applique une fonction non-linéaire, et transmet un signal. Il fonctionne de la façon suivante. Pour un vecteur d'entrée x comme suit :

$$\mathbf{x} = (x_1, x_2, \ldots, x_n)^T \in \mathbb{R}^n$$

Chaque neurone applique une transformation affine :

$$z = \mathbf{w}^T\mathbf{x} + b = \sum_{i=1}^{n} w_i x_i + b$$

où :

— $\mathbf{w} = (w_1, w_2, \ldots, w_n)^T$ est le vecteur des **poids** : les w_i $contrlentl'importancedechaqueentrex_i$. Un poids élevé (positif ou négatif) signifie que le neurone est très sensible à cette entrée.
— $b \in \mathbb{R}$ est le **biais** : il permet de *déplacer le seuil* d'activation du neurone. Il agit comme une constante qui permet à la fonction d'activation de s'activer même quand toutes les entrées sont nulles.
— z est la **pré-activation**

Le neurone applique ensuite une **fonction d'activation** non-linéaire ϕ :

$$a = \phi(z) \quad \text{où le } \textbf{résultat final } a \text{ est la sortie du neurone.}$$

La fonction d'activation est appliquée à la sortie linéaire d'un neurone pour introduire une **non-linéarité** dans le modèle. Sans elle, un réseau constitué uniquement de couches linéaires serait équivalent à une seule couche linéaire, quelle que soit sa profondeur. Ainsi, la fonction d'activation permet :

— d'augmenter la capacité de modélisation du réseau ;
— d'apprendre des relations complexes et non linéaires entre les données d'entrée et la sortie ;
— de contrôler le flux d'information (en supprimant ou en conservant certains signaux).

Il existe différents types de fonction d'activation, que l'on choisit en fonction du problème considéré.

— **ReLU (Rectified Linear Unit)** :

$$\phi(z) = \max(0, z)$$

Utilisée par défaut dans les couches cachées. Elle permet un apprentissage rapide mais peut souffrir du "problème des neurones morts" (valeurs toujours nulles).

— **Sigmoïde (logistique)** :

$$\phi(z) = \frac{1}{1 + e^{-z}}$$

Sortie dans l'intervalle $(0, 1)$. Utile pour des probabilités, mais peut saturer (provoquant des gradients faibles).

— **Tanh (hyperbolique)** :

$$\phi(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

Sortie entre $(-1, 1)$. Centrée sur zéro, elle est souvent préférable à la sigmoïde.

— **Softmax** :

$$\phi(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

Utilisée en sortie pour des problèmes de classification multi-classes.

— **Leaky ReLU** :

$$\phi(z) = \begin{cases} z & \text{si } z > 0 \\ \alpha z & \text{si } z \leq 0 \end{cases}$$

Permet de résoudre en partie le problème des neurones morts de ReLU.

Dans notre cas, nous cherchons une fonction d'activation capable de prendre en compte les éléments suivant décrivant notre modèle : La fonction $N(t)$, représentant le **nombre cumulé d'ordres passés jusqu'à un instant** t sur une journée donnée, possède les caractéristiques suivantes :

— elle est **croissante** (strictement ou par paliers) ;
— elle est **positive** : $N(t) \geq 0$ pour tout t ;
— sa forme est souvent **sigmoïdale** (forme en "S") ou proche d'une **exponentielle aplatie** ;
— elle peut présenter des **plateaux** ou **pics soudains** (dus à des périodes de forte volatilité ou d'activité intense sur le marché).

Par conséquent, pour approximer cette fonction via un réseau de neurones, il est souhaitable d'utiliser une architecture capable :

— d'**apprendre des formes non linéaires douces** ;
— de **ne pas saturer trop rapidement**, pour éviter les gradients nuls ;
— de modéliser une **pente croissante potentiellement raide**, afin de suivre les variations rapides du nombre d'ordres.

Etant donné tous ces éléments, la fonction qui se prête le mieux à notre problème semble être la fonction **ReLU (Rectified Linear Unit)** :

$$\phi(z) = \max(0, z)$$

En effet, elle est simple, efficace, rapide et permet de bien modéliser des morceaux linéaires croissants.

Il existe différents types de réseau de neurones et c'est le Perceptron Multicouche (MLP) qui va nous intéresser car c'est le plus utilisé pour la prédiction et l'approximation de fonctions. Son fonctionnement ressemble beaucoup à ce que nous avons pu voir jusqu'à maintenant : Un **MLP (Multi-Layer Perceptron)** est un réseau de neurones supervisé constitué de plusieurs **couches entièrement connectées** :

$$\text{Entrée} \rightarrow \text{Couches cachées (avec activation)} \rightarrow \text{Sortie}$$

Chaque couche transforme l'information à l'aide de poids, de biais, et de fonctions d'activation non linéaires. Voici les étapes clés de l'algorithme :

Étape 1 – Couche d'entrée

La transformation linéaire s'écrit :

$$z^{(1)} = w^{(1)}t + b^{(1)}$$

puis on applique une fonction d'activation ϕ, par exemple ReLU :

$$a^{(1)} = \phi(z^{(1)})$$

Étape 2 – Couches cachées

Chaque couche l suivante effectue l'opération :

$$z^{(l)} = W^{(l)}a^{(l-1)} + b^{(l)}, \quad a^{(l)} = \phi(z^{(l)})$$

où :

— $W^{(l)}$ est une matrice de poids reliant la couche précédente à la couche actuelle ;
— ϕ est une fonction d'activation non linéaire comme ReLU ou Softplus.

Étape 3 – Couche de sortie

La dernière couche fournit une estimation :

$$\hat{N}(t) = W^{(L)}a^{(L-1)} + b^{(L)}$$

On peut y appliquer une fonction d'activation telle que Softplus pour s'assurer que $\hat{N}(t) \geq 0$, car le nombre d'ordres est toujours positif.

3. Pourquoi le MLP est-il idéal pour approximer une courbe $N(t)$?

Le tableau suivant résume les propriétés attendues de $N(t)$ et les capacités correspondantes d'un MLP :

Critère souhaité pour $N(t)$	**Capacité du MLP**
Fonction continue	Approxime des fonctions continues
Fonction non linéaire	Activations = non-linéarité introduite
Fonction croissante / cumulative	Peut être contraint avec des activations positives
Fonction apprise à partir de données	Aucune hypothèse de forme nécessaire
Pas besoin d'hypothèse forte	MLP = approximation universelle

Théorème d'approximation universelle

Un MLP avec **une seule couche cachée** et un nombre suffisant de neurones peut approximer **n'importe quelle fonction continue** sur un intervalle compact, avec une erreur arbitrairement faible.

Autrement dit :

> *Le MLP peut approximer n'importe quelle courbe $N(t)$ si l'on choisit correctement la profondeur du réseau, les fonctions d'activation, et les paramètres d'entraînement.*

Pour pouvoir prendre en compte au mieux les données et les pics, on va s'appuyer sur une technique de Machine Learning appelée PCA (Analyse en composantes principales

Chapitre 2

Modèle à état continu (*2 pages max*)

En début de chapitre, il est conseillé d'annoncer ce qui va être traité par la suite et dans quel but. Ensuite, il s'agit de traiter la problématique exposée en début du chapitre. Il est important de citer les différentes sources utilisées pour réaliser votre mini-projet.

Il s'agit ensuite de réexposer le problème de modélisation en s'appuyant sur un formalisme mathématique détaillé, de préciser et d'analyser les hypothèses de modélisation, etc.

Ce chapitre permettra de répondre notamment aux questions I.1, I.2 et I.3 du sujet, en précisant les méthodes théoriques utilisées.

Chapitre 3

Résultats de simulation et analyse du modèle à état continu (*2 pages max*)

Le Chapitre 3 porte sur la *présentation des résultats de simulation* (ne pas oublier de préciser les valeurs numériques des paramètres considérés) et l'*analyse des résultats de simulation obtenus en utilisant le modèle à état continu*. Ce chapitre permettra de valider en simulation le modèle à état continu, complétant les réponses aux questions I.1, I.2 et I.3 du sujet.

Chapitre 4

Modèle à évènements discrets (*2 pages max*)

Le Chapitre 4 porte sur le développement d'un *modèle à évènements discrets*. **On ne demande pas de simulateur pour le modèle à évènements discrets.**

Ce chapitre permettra de répondre à la question II.1 du sujet.

Chapitre 5

Conclusion (*1 page max*)

5.1 Conclusion sur les modèles proposés

La conclusion résume les grandes lignes du rapport (principe des solutions, principaux résultats et limitations). Elle doit contenir le message essentiel que l'on souhaite faire passer et montrer l'intérêt des approches proposées dans le rapport.

5.2 Perspectives

Des perspectives seront demandées dans la plupart des projets que vous allez faire. Il est important de faire apparaître des pistes envisagées pour la continuation de votre projet.

Annexe 1 (si besoin)

Style de rédaction

Un rapport technique se rédige dans un style neutre. Il convient en particulier d'éviter un vocabulaire familier ou trop littéraire. On recherche la concision et la clarté en évitant les phrases trop longues et les termes imprécis.

L'usage des pronoms personnels (par exemple *nous*) est à limiter à de rare cas précis (par exemple pour mettre l'accent sur un choix déterminant ou pour se démarquer par rapport aux autres).

Une énumération est constituée d'objets qui se situent sur le même plan, par ailleurs les éléments de l'énumération doivent avoir la même nature grammaticale.

Il est fondamental de relire le rapport avec un regard extérieur, en particulier pour vérifier le sens des phrases, ainsi que leur clarté.

Formules mathématiques

— Toutes les notations et grandeurs utilisées doivent être définies (avec leurs dimensions dans le cas des matrices) et, dans la mesure du possible, ces notations doivent rester conventionnelles.
— Les notations doivent être cohérentes sur l'ensemble du rapport (éviter en particulier de réutiliser la même notation pour des objets différents).
— Les formules doivent en général être introduites par une phrase pour les rendre plus explicites.
— Il est conseillé de numéroter les équations, les tableaux, les figures et de les citer dans le texte. Par exemple, la variable $\mathbf{x}$ est introduite dans (5.1) de la manière suivante :

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \tag{5.1}$$

On adopte en général les conventions suivantes pour la présentation des formules :
— Les scalaires sont en italique et en minuscule (par exemple x) ;
— Les vecteurs sont en gras et en minuscule (par exemple $\mathbf{x}$) ;
— Les matrices sont en gras et en majuscule (par exemple $\mathbf{X}$).

Figures et tableaux

Dans la suite quelques consignes pour insérer les figures (ou tableaux) dans le rapport sont précisées :
— Toutes les figures doivent être numérotées et comporter une légende.
— Pour ne pas couper les paragraphes avec des figures, faire référence au numéro de la figure dans le texte (ex. "*voir* Figure 5.1" plutôt que "*on voit sur la figure ci-dessous* :").
— Afin d'optimiser la mise en page et d'éviter les grands blocs blancs, mettre les figures en haut ou en bas de page, ou encore les regrouper sur une même page.

Quelques règles pour une présentation des courbes correcte :
— Préciser la nature et les unités des axes ;
— Ajouter une légende permettant de distinguer les courbes (s'il y a plusieurs courbes) ;
— S'assurer de la lisibilité des échelles ;
— S'assurer de la lisibilité des courbes (en particulier si l'impression est en noir et blanc éviter les couleurs et préférer des symboles ou des styles de lignes différents pour identifier les courbes).

Exemple de figure. Un premier concept est illustré dans la Figure 5.1 (voir comment citer une figure dans le texte).

FIGURE 5.1 – Une jolie figure

Bibliographie

Quelques règles pour une présentation correcte des références bibliographiques sont proposées dans la suite :
— Les références bibliographiques sont regroupées à la fin du document ;
— Les références doivent être complètes : titre, auteurs, date, éditeur (revue, volume et numéro de pages pour les articles de revue) ;
— Toutes les références bibliographiques sont numérotées et doivent être citées dans le document, par exemple [2].

Divers

Il est conseillé de respecter les règles de typographie, en particulier :
— Ne mettre des majuscules qu'en début de phrase, sur les noms propres et les sigles ;
— Ne pas mettre de ' :' ou de '.' à la fin des titres ;
— Vérifier que la numérotation et le style des titres sont uniformes ;
— Définir un bas (ou haut) de page contenant le numéro de page, le nom des auteurs et le titre du document, la date ;
— Réaliser le double alignement du texte (à droite et à gauche) du texte sur l'ensemble du document.

Processus ponctuel : $\{t_1, \ldots, t_n\}$ aléatoires tq $t_1 < \ldots < t_n$

$\mathbb{P}(N(t+s) - N(t) = k) = \frac{\lambda^k}{k!} e^{-\lambda}$
$\mathbb{E}[N(t+T) - N(t)] = \lambda T$
$\Lambda(t) = \int_0^t \lambda(u)\, du$
$N(t) - N(s)$ suit une loi de poisson de paramètre $\Lambda(t) - \Lambda(s)$
$\lambda(t \mid H_t) = \mu(t) + \sum_{t_i < t} f(t - t_i)$ $f(\tau) = \alpha e^{\beta \tau}$
$t_i^* = \Lambda(t_i)$
$\lambda(t) = \mu(t) + \sum_{t_i < t} \alpha e^{-\beta(t - t_i)}$
$\log \mathcal{L}(\alpha, \beta) = \sum_{k=1}^{K} \log \lambda(t_k) - \int_0^T \lambda(t)\, dt$

Bibliographie

[1] Patrick J. Laub, Thomas Taimre, and Philip K. Pollett. Hawkes processes, 2015.

[2] M. Seidemann. Improving engineering reports and talks. *IEEE Transactions on Engineering Writing and Speech*, 1967.

[3] Ioane Muni Toke. *An Introduction to Hawkes Processes with Applications to Finance*. Polycopié Centrale-Supélec, 2011.

[4] Jérémy Bettinger Simon Viel. *Cours sur les processus de poissons*. Université de Rennes, 20.