

EFFICIENT SIMULATION OF HAWKES PROCESSES

by

Mehmet Atilla Purazer

B.S., Industrial Engineering, Boğaziçi University, 1992

M.S., Industrial Engineering, Boğaziçi University, 1994

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

Graduate Program in Industrial Engineering
Boğaziçi University

2024

ACKNOWLEDGEMENTS

First of all, I am grateful to my thesis supervisor Professor Wolfgang Hörmann. I have gained a new vision from this valuable guidance and support during my thesis. Next, I want to thank Professor Aybek Korugan and Professor Kemal Dinçer Dineç for their important contributions as the members of my thesis supervisory committee during the last four years. I want to thank Professor Refik Güllü and Professor Zeynep Gökçe İşlier for being the members of my thesis defense committee. Additionally and most important, I want also to thank all the academic staff of Industrial Engineering Department of Boğaziçi University for their continuous effort to build and sustain a valuable academic environment.

Finally, I want to thank my family, my wife Banu, my daughter Duru and my son Emre for their continuous support, motivation and patience during my thesis.

ABSTRACT

EFFICIENT SIMULATION OF HAWKES PROCESSES

The aim of this study is to implement a fast and efficient algorithm to simulate Hawkes process whose intensity function depends on the events in its process history. In real life cases like seismology, disease spread, stock markets and social media, history (past events) affects future (next events). Hawkes process is suitable to model these cases with its history dependent conditional intensity function. Ogata's modified thinning algorithm is frequently used to simulate Hawkes processes. It is an inefficient algorithm especially for generating huge number of events for different purposes like statistical studies. A cluster-based (composition) algorithm simulating its sub-processes with inversion method using order statistic and time re-scaling theorem is proposed to the alternative of Ogata's modified thinning algorithm. Theoretical and empirical performance assessments are executed to compare the proposed cluster-based (composition) algorithm with the other algorithms and software in the literature. In the theoretical studies the number of operations which are required to generate Hawkes process events are identified and counted for different simulation algorithms to compare their performances. These operations are random variates, intensity function evaluations, inverse cumulative intensity function evaluations, comparison operations and sorting operations. For the empirical studies, a software of the proposed algorithm is implemented in R. This software is used to count the number of operations in the experiments. The results of the empirical studies justify theoretical studies. Furthermore, the time performance of the implemented software is compared with the software about Hawkes process simulation in the literature. The results of the theoretical and empirical studies show that the proposed algorithm is faster and more efficient compared to the other algorithms in the literature.

ÖZET

HAWKES SÜREÇLERİNİN VERİMLİ SİMÜLASYONU

Bu çalışmanın amacı, yoğunluk fonksiyonu proses geçmişindeki olaylara bağlı olan Hawkes prosesini simüle etmek için hızlı ve verimli bir algoritma uygulamaktır. Sismoloji, hastalık yayılımı, borsalar ve sosyal medya gibi gerçek hayattaki vakalarda, tarih (geçmiş olaylar) geleceği (sonraki olaylar) etkiler. Hawkes süreci, geçmişe bağlı koşullu yoğunluk fonksiyonu ile bu durumları modellemek için uygundur. Ogata'nın modifiye edilmiş inceltme algoritması, Hawkes süreçlerini simüle etmek için sıklıkla kullanılır. Özellikle istatistiksel çalışmalar gibi farklı amaçlar için çok sayıda olay oluşturmak için verimsiz bir algoritmadır. Ogata'nın modifiye inceltme algoritması alternatifine, sıra istatistiği ve zaman yeniden ölçeklendirme teoremini kullanarak alt süreçlerini ters çevirme yöntemiyle simüle eden küme tabanlı (birleştirme) bir algoritma önerilmiştir. Önerilen küme tabanlı (birleştirme) algoritmanın literatürdeki diğer algoritma ve yazılımlarla karşılaştırılması için teorik ve ampirik performans değerlendirmeleri yapılmıştır. Teorik çalışmalarda, Hawkes süreç olaylarını oluşturmak için gerekli olan işlem sayısı belirlenir ve performanslarını karşılaştırmak için farklı simülasyon algoritmaları için sayılır. Bu işlemler rassal değişkenler, yoğunluk fonksiyonu değerlendirmeleri, ters kümülatif yoğunluk fonksiyonu değerlendirmeleri, karşılaştırma işlemleri ve sıralama işlemleridir. Ampirik çalışmalar için, önerilen algoritmanın bir yazılımı R'de uygulanmaktadır. Bu yazılım, deneylerdeki işlem sayısını saymak için kullanılır. Ampirik çalışmaların sonuçları teorik çalışmaları haklı çıkarmaktadır. Ayrıca, uygulanan yazılımın zaman performansı, literatürde yer alan Hawkes süreç simülasyonu ile ilgili yazılımlarla karşılaştırılmıştır. Yapılan teorik ve ampirik çalışmaların sonuçları, önerilen algoritmanın literatürdeki diğer algoritmalara göre daha hızlı ve verimli olduğunu göstermektedir.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS	iii
ABSTRACT	iv
ÖZET	v
LIST OF FIGURES	ix
LIST OF TABLES	xi
LIST OF SYMBOLS	xiii
LIST OF ACRONYMS/ABBREVIATIONS	xvi
1. INTRODUCTION	1
2. RANDOM PROCESSES	4
2.1. The Poisson Process	9
2.2. The Non-Homogeneous Poisson Process	14
2.3. The Hawkes (Self-Exciting) Process	20
2.4. Cluster Process Representation of Hawkes Process	26
2.5. The Extensions of Hawkes Process	33
2.6. Parameter Estimation	38
3. RANDOM PROCESS GENERATION IN THE LITERATURE	50
3.1. Preliminaries for Simulation Algorithms	50
3.2. The Poisson Process Simulation Methods	53
3.3. The Non-Homogeneous Poisson Process Simulation Methods	54
3.3.1. The Inversion Method with Inter-event Times for NHPP	55
3.3.2. The Inversion Method with Order Statistics for NHPP	57
3.3.3. The Thinning Method	60
3.3.4. The Composition Method	61
3.4. Hawkes Process Simulation Algorithms	62
3.4.1. The Inversion Algorithm	62
3.4.2. Ogata's Modified Thinning Algorithm	63
3.4.3. The Cluster-based (Composition) Algorithm	65
3.4.4. The Software Packages for Hawkes Process Generation	67

4. THE PERFORMANCE OF THE GENERATION ALGORITHMS	69
4.1. The Performance of the Poisson Process Generation Methods	70
4.2. The Performance of the NHPP Generation Methods	73
4.2.1. The Inversion Methods	74
4.2.2. The Thinning Method	76
4.2.3. The Composition Method	79
4.2.4. Performance Comparison of Different NHPP Generation Methods	81
4.3. The Performance of the Hawkes Process Generation Algorithms	85
4.3.1. Ogata's Modified Thinning Algorithm	86
4.3.2. The Cluster-based (Composition) Algorithm	90
4.3.3. The Performance Comparison of the HP Generation Algorithms	98
4.4. The Pseudo Code of The Algorithm	104
5. THE IMPLEMENTATION DETAILS	110
5.1. The Set-up Steps	110
5.2. Sorting the Event Times	112
5.3. The Inputs and Outputs of the Software with Sample Cases	113
6. THE PERFORMANCE EXPERIMENTS	120
6.1. The Performance Comparison with Other Softwares	120
6.2. The Time Comparison with Ogata's Modified Thinning Algorithm . . .	125
6.3. The Comparison of the Number of Operations Required	129
6.4. The Time Performance of the Presented Algorithm	135
7. CONCLUSION	137
REFERENCES	140
APPENDIX A: OGATA'S MODIFIED THINNING ALGORITHM	147
APPENDIX B: THE R CODE OF THE SOFTWARE	150
B.1. The R Code of the Software	150
B.2. The Pseudo Code of the Presented Hawkes Process Algorithm	183
B.3. The Outputs of the Package for the Sample Cases	186
APPENDIX C: THE ADDITIONAL PERFORMANCE EXPERIMENTS . . .	194
C.1. The Number of Operations of the Proposed Algorithm	194
C.2. The Theoretical and Empirical Results Comparison	196

C.2.1. The Experiments with Constant Base Intensity	196
C.2.2. The Experiments with the Time Varying Base Intensity Function	197
C.2.3. The Experiments with Ogata's Modified Thinning Algorithm . .	198
APPENDIX D: THE SOFTWARE PACKAGES FOR HP SIMULATION . . .	199
D.1. The R Packages Experiment Output	204
APPENDIX E: APPROXIMATING A COMPLICATED FUNCTION	210
E.1. Approximating a Complicated Function	210
E.2. Numerical Integration	212
APPENDIX F: THE FUNCTIONS OF THE R PACKAGE RUNURAN . . .	222

LIST OF FIGURES

Figure 6.1.	The time performance comparison with other software with parameter setting 3	124
Figure 6.2.	"Time required vs Number of the events" experiments for Case 1 with parameter setting 1	127
Figure 6.3.	"Time required vs Number of the events" experiments for Case 2 with parameter setting 2	128
Figure 6.4.	Ogata's modified thinning algorithm: Number of the excitation function evaluations vs Number of the events generated with parameter setting 2	130
Figure 6.5.	Ogata's modified thinning algorithm, Number of excitation function evaluations for accepted events vs Number of events generated with parameter setting 2	132
Figure 6.6.	Ogata's modified thinning algorithm: Number of the excitation function evaluations for rejected events vs Number of the events with parameter setting 2	133
Figure 6.7.	Cluster-based algorithm: Total Poisson and Uniform random numbers vs Number of the events with parameter setting 2	134
Figure 6.8.	Cluster based algorithm: events sorted and unsorted with the parameter setting 4	136

Figure C.1. Cluster based algorithm - events vs number of operations with pa-	
rameter setting 5	195



LIST OF TABLES

Table 2.1.	The probability density function and its related intensity function .	35
Table 3.1.	The R-packages with simulation algorithms	68
Table 3.2.	The Python-packages with simulation algorithms	68
Table 4.1.	The total number of operations which are required by the Poisson process generation methods	73
Table 4.2.	The total number of the operations which are required by the NHPP simulation with the thinning and inversion methods	83
Table 4.3.	The total number of the operations which are required by the NHPP simulation with the composition method	84
Table 4.4.	The total number of operations which are required by the Hawkes process simulation algorithms on the unbounded time interval . . .	101
Table 4.5.	The total number of the operations which are required for the Hawkes process simulation algorithms on the bounded time interval	102
Table 5.1.	The running time of the bucket sort and the radix sort	113
Table 5.2.	The time performance of different sorting methods in micro seconds	113
Table D.1.	The speed comparison with the Hawkes process generation functions of the R-package "IHSEP"	206

Table D.2.	The speed comparison with the Hawkes process generation functions of the R-package "RHawkes"	207
Table D.3.	The speed comparison with the Hawkes process generation functions of the R-package "hawkesbow"	208
Table D.4.	The speed comparison with the Hawkes process generation functions of the R-package "hawkes"	209
Table E.1.	Experiment results with numerical integration using Simpson's rule 1/3 with cubic spline	218
Table E.2.	Experiment results with numerical integration using Gauss-Lobatto rules with cubic spline interpolation	219
Table E.3.	Experiment results with numerical integration using Boole's rule with cubic spline interpolation with tolerance limit for the area difference for iterations	220
Table E.4.	Experiment results numerical integration using Boole's rule with cubic spline interpolation with convergence criterion	221

LIST OF SYMBOLS

c	A temporal shift
E	An exponential random variate with rate one
$E[.]$	The expected value function
f	Fertile event
F	Fertile set
$f(t)$	Probability density function
$F(t)$	Cumulative distribution function
$\mathcal{F}$	Event space
G_i	The number of events in generation i
H_t	The set of the occurrence times t_i 's of the events that happened before time t
k	The number of independent non empty Hawkes simulation runs, non-empty cascades
$L(t H_t)$	A time interval of length for Ogata's modified thinning algorithm
lam0	The base intensity function of Hawkes process
M	A finite bound for thinning method
m_i	The weight of event i
m_r	r^{th} moment of a random variable
$M(t H_t)$	A local bound for Ogata's modified thinning algorithm
marks	Weight of the events
MLE	Maximum Likelihood Estimation
n	The number of events
n^*	The branching factor
$\{N(t), t \geq 0\}$	A counting process
$o(h)$	Little o notation
P	A Poisson random variate
$P(.)$	A probability function which assigns, to each event in the event space a probability

r	power of the random variable
repr	Reproduction number, mean number of offspring (A non-negative numeric value)
S_n	The occurrence time of the n^{th} event of the process
S_n^*	The occurrence time of the n^{th} event of the process with the hat intensity at the thinning method/Ogata's modified thinning algorithm
ϕ	The excitation function of Hawkes process
t	Time
T	Simulation period end time
U	A standard uniform random number
$\text{Var}(\cdot)$	Variance function
X	A random variable
α	The magnitude of influence (initial value)
β	The exponent that describes how fast an event is forgotten
λ	Rate of homogeneous Poisson process and rate of exponential distribution
$\lambda(t)$	The intensity function of NHPP
$\lambda^*(t)$	The hat intensity function at the thinning method/Ogata's modified thinning algorithm
$\lambda(t H_t)$	Conditional intensity function of Hawkes process
$\lambda_0(t)$	The base intensity function of Hawkes Process, immigrant sub-process intensity function
$\Lambda(t)$	The Cumulative intensity function of NHPP
$\Lambda(T)$	Total area under the intensity function, $\lambda(t)$ during the simulation period $(0, T)$
$\Lambda_0(t)$	The cumulative base intensity function of Hawkes process
$\Lambda_0(T)$	Total area under the base intensity function, $\lambda_0(t)$ during the simulation period $(0, T)$
$\Lambda^{-1}(t)$	The inverse cumulative intensity function of NHPP

$\Lambda_0^{-1}(t)$	The inverse cumulative base intensity function of Hawkes process
μ	mean of the random variable
ν	Total mass of the excitation function $\varphi(t)$
$\varphi(t)$	The excitation function of Hawkes Process, offspring sub-process intensity function
$\Phi(t)$	The cumulative excitation function of Hawkes process
$\Phi(T)$	Total area under the excitation function, $\lambda_0(t)$ during the simulation period $(0, T)$
$\Phi^{-1}(t)$	The inverse cumulative excitation function of Hawkes process
Ω	Sample space which is the set of all possible outcomes

LIST OF ACRONYMS/ABBREVIATIONS

CDF	The cumulative distribution function
iid	independent and identically distributed
HP	Hawkes Process
NHPP	Non-Homogenous Poisson Process
RVG	Random Variate Generation



1. INTRODUCTION

A social media cascade can be modeled as a point process, because each user who received the post in cascade has the potential to re-post to his/her followers. It can be modeled as a point process whose intensity function is history dependent. The self-exciting process, also called Hawkes process [1] is a suitable candidate to model the propagation of a post in social media [2]. A social media post has similar characteristics like earthquake propagation, disease spread, demand for a stock. In these areas, Hawkes processes are frequently used to simulate the process and investigate the future behavior of the process. The difference between social media and other fields is the effect of human behavior on the process. The human behavior affects the type of the excitation function. For seismology, the exponential decay is more suitable. However, for social media case, power law function is more frequently used.

There are two Hawkes process generation algorithms in the literature. They are intensity-based and cluster-based algorithms. Ogata's modified thinning algorithm [3] is an intensity-based algorithm which is frequently used in the studies. It is the analogue of the thinning method for the non-homogeneous Poisson process [4]. The cluster-based algorithm [5, 6] decomposes the Hawkes process into (immigrant and offspring) sub-processes which are independent non-homogeneous Poisson processes. This algorithm is based on the paper of Hawkes and Oakes "A Cluster Process Representation of a Self-Exciting Process" [7].

However, two important subjects are not studied in the literature. The first subject is the performance assessment of the Hawkes process simulation algorithms and the choice of the homogeneous and non-homogeneous Poisson process simulation method for cluster-based Hawkes process generation algorithm. The simulation method of the non-homogeneous Poisson process is important because there are many non-homogeneous Poisson processes to be generated (for the sub-processes) with small number of expected events in the cluster-based Hawkes process generation algorithm.

The second subject is to design a fast and efficient Hawkes process simulation algorithm to generate huge number of Hawkes process events for scientific studies.

To design an efficient Hawkes process implementation, it is important to count the number of main operations used for the generation to decide which is the most efficient method to simulate Hawkes processes. This is the main part of the theoretical study. The second important point of the theoretical study to design an efficient implementation is the choice of the random variate generation method. According to the results of theoretical studies, an efficient simulation algorithm for Hawkes processes with general smooth intensity function is going to be presented. The empirical part includes the comparison of the speed performance of the presented algorithm with algorithms in the literature. These packages are IHSEP [8], RHawkes [9], hawkesbow [10] and hawkes [11].

Due to the increasing popularity of Hawkes process several generalizations are suggested in the literature. We will also discuss the simulation of marked Hawkes process [2, 12] and of renewal Hawkes process [13]. The cluster-based (composition) algorithm can simulate the Galton Watson branching process which can be modeled as a Hawkes process with a single immigrant.

The structure of this study is as follows:

- Chapter 2 contains the definitions and related subjects of random processes; Poisson process, non-homogeneous Poisson process and Hawkes process.
- Chapter 3 contains the preliminaries for the simulation algorithms and the simulation algorithms for Poisson process, non-homogeneous Poisson process and Hawkes process in the literature.
- Chapter 4 focuses on the performances of the point processes (Poisson process, non-homogeneous Poisson process and Hawkes process) generation algorithms. The pseudo code of the fastest algorithm for the Hawkes process generation is stated.

- Chapter 5 contains the implementation details of the Cluster-based (composition) algorithm - simulating its sub-processes with inversion method using order statistics and time re-scaling theorem are presented.
- Chapter 6 includes the results of empirical studies.
- Chapter 7 summarizes the results of this study.



2. RANDOM PROCESSES

Chapter 2 contains the definitions and related topics of random processes. These subjects are the basis of the simulation algorithms of the (homogeneous) Poisson process, non-homogeneous Poisson process and Hawkes process.

A random process (or stochastic process) $\{X(t), t \in T\}$ is a collection of the random variables in a probability space $(\Omega, \mathcal{F}, P)$. That is, for each $t \in T$, $X(t)$ is a random variable. The index t is often interpreted as time and, as a result, $X(t)$ is the state of the process at time t [14, 15]. A probability space consists of three elements: Ω is sample space which is the set of all possible outcomes, $\mathcal{F}$ is event space - an event is a set of outcomes in the sample space and P is a probability function which assigns a probability to each event in the event space [15]. The set T is called the index set of the process. When T is a countable set, the stochastic process, $\{X_n, n = 0, 1, \dots\}$ on the non-negative integers is said to be a discrete-time process. If T is an interval of the real line, the stochastic process, $\{X(t), t \geq 0\}$ indexed by the non-negative real numbers, is said to be a continuous-time process. The state space of a stochastic process is defined as the set of all possible values that the random variables $X(t)$ can assume. [14]. Stochastic processes can be grouped into various categories, which include random walks, martingales, Markov processes, Levy processes, Gaussian processes, random fields, renewal processes, and branching processes. This study focuses on the Poisson process, non-homogeneous Poisson process and Hawkes (self-exciting) process. Before going deeper into them, the following concepts about stochastic processes are introduced: point process, counting process, Poisson random variable, exponential random variable and renewal process.

The Point Process: A point process on the line may be taken as modeling the occurrences of some phenomenon at the time epochs $\{S_n\}$, with " n " in some suitable index set [16]. The random process $\{S_n, n = 1, 2, \dots\}$ is said to be a point process if S_n represents the occurrence time of the n^{th} event. A point process S_n must satisfy:

- $S_1 \geq 0$
- $S_{n-1} < S_n$, for $n \geq 2$

For such a process, there are four equivalent descriptions of the sample paths:

- counting measures
- non-decreasing integer-valued step functions
- sequences of points, and
- sequences of intervals

Point processes are used to model mathematically the physical systems that produce a stochastic set of localized events in time or space.

The Counting Process: The random process $\{N(t), t \geq 0\}$ is said to be a counting process if $N(t)$ is the number of events that occurred on the time interval $(0, t]$. (A formal definition of $N(t)$ is given by $\{N(t) \equiv \max\{n : S_n \leq t\}\}$ where $S_0 \equiv 0$. S_n is the occurrence time of the n^{th} event [14]. A counting process $N(t)$ must satisfy:

- (i) $N(0) = 0$
- (ii) $N(t) \geq 0$
- (iii) $N(t) \in 0, 1, 2, \dots \quad \forall t \geq 0$ [$N(t)$ is integer valued.]
- (iv) $N(t+s) - N(t)$ is the number of events occurred on the time interval $(t, t+s]$
 $\forall t, s \geq 0$
- (v) $N(t+s) \geq N(t) \quad \forall t, s \geq 0$ [$N(t)$ is non-decreasing and right continuous]

The counting process is integer valued, non-decreasing and right continuous function. A counting process $\{N(t), t \geq 0\}$ is said to possess independent increments if the numbers of events that occur in disjoint time intervals are independent (the co-variance of the number of events in disjoint time intervals is zero). The increments that occur in any interval is independent of the process history [14]. In other words, the counting process is said to have independent increments if $N(t_i + s_i) - N(t_i)$ for $i = 1, 2, \dots$ are

independent random variables whenever $t_{i+1} > t_i + s_i \quad \forall i$ [17].

A counting process $\{N(t), t \geq 0\}$ is said to possess stationary increments if the distribution of the number of events that occur in any interval of time depends only on the length of the time interval. In other words, the distribution of $N(t + s) - N(t)$ does not depend on t [17]. The rate $\lambda, (\lambda > 0)$ of the process is constant during the whole process.

An analogue from daily life is that independent increments says that customer traffic in one interval of time does not affect traffic in another (disjoint) interval of time. Stationary increments says that the traffic process is not changing over time [17].

A counting process is called simple (i.e., events occur one by one, [16]) when

$$P\{N(t + h) - N(t) = 0 \text{ or } 1\} = 1 \quad \forall t, \text{ and as } h \rightarrow 0. \quad (2.1)$$

The relation between the point process and the counting process can be stated as follows:

$$S_n \leq t \Leftrightarrow N(t) \geq n. \quad (2.2)$$

The Poisson Random Variable: The discrete random variable X , taking on one of the values $0, 1, 2, \dots$ is said to have Poisson distribution with parameter $\lambda > 0$, if its probability mass function is expressed as: [14, 17, 18]

$$p(n; \lambda) = P\{X = n; \lambda\} = e^{-\lambda} \frac{\lambda^n}{n!}, \quad n = 0, 1, \dots \quad (2.3)$$

The positive real number λ in Equation 2.3 is equal to the expected value of the random variable X and to its variance, $E[X] = Var(X) = \lambda$.

It's well-known that a binomial distribution with parameters (n =number of independent trials, p =probability of success) converges to a Poisson distribution with parameter (λ =mean) when the number of independent trials, n is increased and the probability of success p is decreased at the same time in such a way so that the mean stays constant, $\lambda = np$ [17, 18]. This Poisson approximation is also called "Law of rare events". The events are rare because they occur with small probability [19]. As a rule of thumb, if $n \geq 100$ and $np \leq 10$, the Poisson distribution ($\lambda = np$) can provide a very good approximation to the binomial distribution [20].

The form of the probability mass function $P\{X = k; \lambda\}$ of Poisson random variable X with parameter λ , increases monotonically with k up to the point where k reaches the largest integer not exceeding λ , and after that point decreases monotonically with k [18].

The sum of independent Poisson random variables are also a Poisson random variable. Let X and Y be independent Poisson random variables with means λ_1 and λ_2 respectively than the random variable Z which is the sum of X and Y is also a Poisson random variable with mean $\lambda_1 + \lambda_2$ [14].

The Exponential Random Variable: The continuous random variable X , taking on $x \geq 0$ is said to have exponential distribution with rate parameter $\lambda > 0$, if its probability density function can be written as: [14, 18, 21]

$$f_X(x; \lambda) = \lambda e^{-\lambda x}, \quad (2.4)$$

or equivalently if its cumulative distribution function can be expressed as

$$F_X(X \leq x; \lambda) = P\{X \leq x\} = 1 - e^{-\lambda x}. \quad (2.5)$$

The expected value of exponential distribution with rate λ is expressed as

$$E[X; \lambda] = \frac{1}{\lambda}. \quad (2.6)$$

The variance of exponential distribution with rate λ is expressed as

$$Var(X; \lambda) = \left(\frac{1}{\lambda}\right)^2. \quad (2.7)$$

The exponential random variable X with rate λ , is memoryless because it satisfies the following equation

$$P\{X > s + t | X > t\} = \frac{P\{X > s + t\}}{P\{X > t\}} = e^{-\lambda s} = P\{X > s\} \quad \forall s, t \geq 0. \quad (2.8)$$

The form of the probability density function of exponential distribution decreases monotonically.

The Renewal Process: A counting process whose sequence of inter-event times are independent and identically distributed is called a renewal process. The formal definition of renewal process is as follows [17, 21]:

Let X_i be a sequence of independent and identically distributed non-negative random variables having distribution function F and assume that $F(0) \neq 1$ so that X_i are not identically 0, and set

- $S_0 = 0$
- $S_n = \sum_{i=1}^n X_i$
with
- $N(t) = \sup(n : S_n \leq t)$

the process $\{N(t), t \geq 0\}$ is called a renewal process where

- S_n is the event time of the n^{th} event (called renewal times)

- X_i is the time between the events S_{n-1} and S_n
- $N(t)$ is the number of events until time t

An event is also called a renewal, because if it is considered the time of the event occurrence as the new origin then (because the inter-event times X_i are independent and identically distributed) the process of future events is also a renewal process with inter-event time distribution F [17].

The Strong Law for Renewal Processes [17]: With probability 1,

$$\lim_{t \rightarrow \infty} \frac{N(t)}{t} = \frac{1}{\lambda}, \quad (2.9)$$

where $\frac{1}{\lambda}$ is the inverse of the mean inter-event time.

A renewal process is easily simulated by generating its inter-event times.

2.1. The Poisson Process

The Poisson process is a stochastic process for modeling the event times of a system. In other words, it is a model for a series of discrete events. It is a counting process $\{N(t), t \geq 0\}$ which counts the number of events on the interval $(0, t]$ and it is a point process $\{S_n, n = 1, 2, \dots\}$ which generates the occurrence times S_n of the events entered in the system. Beside the definition of the (homogeneous) Poisson process, the following properties of the (homogeneous) Poisson process are presented:

- The inter-event time distribution of the Poisson process
- Time re-scaling theorem

The definition of the Poisson process [14,19,21]: The counting process $\{N(t), t \geq 0\}$ is said to be a Poisson process with rate $\lambda > 0$, if

- (i) $N(0) = 0$
- (ii) The process has independent increments.
- (iii) The number of events in any interval of length t is Poisson distributed with mean λt . It is expressed as

$$P\{N(t+s) - N(s) = n\} = e^{-\lambda t} \frac{(\lambda t)^n}{n!} \quad \forall t, s \geq 0. \quad (2.10)$$

The rate λ of the Poisson process is a non-negative constant function (stationary increments) and it is assumed to be independent of the process history (independent increments). In the literature, the Poisson process is also called the homogeneous Poisson process or stationary Poisson process because its rate λ is constant during the process. $E[N(t)]$ = Expected number of events generated by a Poisson process with rate λ on the interval $(0, t]$ is expressed as

$$E[N(t)] = \lambda t = \int_0^t \lambda \, du. \quad (2.11)$$

Equation 2.11 indicates the area under the rate λ of a homogeneous Poisson process on the interval $(0, t]$.

An alternative definition of the Poisson Process is as follows [14,18]: The counting process $N(t)$, $t \geq 0$ is said to be a homogeneous Poisson process having rate $\lambda > 0$, if

- (i) $N(0) = 0$
- (ii) The process has stationary and independent increments.
- (iii) $P\{N(h) = 1\} = \lambda h + o(h)$
- (iv) $P\{N(h) = 0\} = 1 - \lambda h - o(h)$

" $o(h)$ " : little o notation [22]. This mathematical notation indicates that the decay (respectively, growth) rate of a function is faster (respectively, slower) than the decay (respectively, growth) rate of another function. The function $f(\cdot)$ is said to be " $o(h)$ "

if it satisfies the following equation

$$\lim_{h \rightarrow 0} \frac{f(h)}{h} = 0. \quad (2.12)$$

Equation 2.12 indicates that $f(h)$ goes to zero faster than h does [22].

The two definitions are equivalent [14]. An important point about the Poisson process is that it is the only counting process with stationary, independent increments and continuous inter-arrival times [17].

The time " t " passed between two consecutive events of the Poisson process is an exponential random variable with mean $\frac{1}{\lambda}$ and variance $(\frac{1}{\lambda})^2$ [14, 18, 21]. The distribution function F of exponential distribution with rate λ is formulated as

$$F(t; \lambda) = P\{T \leq t\} = 1 - e^{-\lambda t} \quad \forall t \geq 0. \quad (2.13)$$

To summarize: The Poisson process is

- the only counting process with stationary, independent increments and continuous inter-arrival times [17].
- a special case of renewal processes, the inter event times are exponentially distributed [17].

A counting process with rate λ is called a Poisson process with rate λ because the number events, $N(t)$ generated by the process on the interval $(0, t]$ is Poisson distributed with mean λt [17].

The Inter-event Time Distribution of the Poisson Process: The Poisson process is a special case of the renewal process. If the inter-event time distribution F of a renewal process is exponentially distributed with rate λ , then the renewal process is said to be a Poisson process with rate λ [17]. The probability density function $f(t)$ of inter-event

times between successive events of a point process is defined as [14]

$$f(t) = \lim_{dt \rightarrow 0} \frac{P\{X \in (t, t + dt)\}}{dt}. \quad (2.14)$$

The intensity function $\lambda(t)$ is defined as conditional probability density function that next event of the point process will occur after t time units since the last event occurred, not before not later [14]. It can be written as

$$P\{X \in (t, t + dt) | X > t\} = \frac{P\{X \in (t, t + dt), X > t\}}{P\{X > t\}} = \frac{P\{X \in (t, t + dt)\}}{P\{X > t\}}, \quad (2.15)$$

$$P\{X \in (t, t + dt) | X > t\} \approx \frac{f(t) dt}{1 - F(t)} = \lambda(t) dt. \quad (2.16)$$

The intensity function $\lambda(t)$ of a point process can be stated in terms of distribution of $F(t)$ and probability density function $f(t)$ of inter-event time X as

$$\lambda(t) = \frac{f(t)}{1 - F(t)}. \quad (2.17)$$

This result indicates that the intensity function $\lambda(t)$ uniquely determines the distribution function F of inter-event times [14]. The distribution function of inter event times can be reached by first integrating Equation 2.17 with the condition $F(0) = 0$ because F is a distribution function [14]

$$\int_0^t \lambda(u) du = -\log [1 - F(t)], \quad (2.18)$$

and then rearranging it in term of $F(t)$

$$F(t) = 1 - e^{-\int_0^t \lambda(u) du}. \quad (2.19)$$

If $\lambda(t)$ is a scalar as λ , Equation 2.19 turns to the exponential distribution function F with rate λ . It can be expressed as

$$F(t) = 1 - e^{-\lambda t}, \quad (2.20)$$

which satisfies the following equation

$$\lambda(t) = \frac{f(t)}{1 - F(t)} = \frac{\lambda e^{-\lambda t}}{e^{-\lambda t}} = \lambda. \quad (2.21)$$

As a summary, inter-event times of the (homogeneous) Poisson process with rate λ are exponentially distributed with rate λ [14].

Time Re-scaling Theorem: The definition of the Time Re-scaling Theorem is as follows [23]: Given a point process with conditional intensity function $\lambda(t|H_t)$ and with occurrence times $0 < s_1 < \dots < s_{N(T)} \leq T$ satisfying $0 < \lambda(t|H_t) \forall t \in (0, T]$. Define the transformation

$$\Lambda(s_j) = \int_0^{s_j} \lambda(t|H_t) dt \quad (2.22)$$

for $j = 1, \dots, N(T)$ and assume that $\Lambda(t) < \infty$ with probability one $\forall t \in (0, T]$. Then the $\Lambda(s_j)$'s are a Poisson process with unit rate. The time re-scaling theorem generates a history-dependent re-scaling of the time axis that converts a point process into a Poisson process with a unit rate [23]. The inter-event times of the Poisson process with rate one is exponentially distributed with rate one [14]. The direct consequence of this fact is that

$$\Lambda(s_j) - \Lambda(s_{j-1}) \quad (2.23)$$

is exponentially distributed with rate one for $j = 1, \dots, N(T)$ and $s_0 = 0$. Daley and Vere-Jones [16] (page 258, Theorem 7.4.I) stated that this transformation (transforming a simple point process with strictly positive intensity $\lambda(t)$ and cumulative intensity

function $\Lambda(t)$ into the homogeneous Poisson process with rate one) can be performed by the variable change (Time Re-scaling Theorem) $t \mapsto \Lambda(t)$. As a summary, time re-scaling theorem states that any simple point process with an integrable conditional intensity function may be transformed into the Poisson process with unit rate.

2.2. The Non-Homogeneous Poisson Process

The non-homogeneous Poisson process is a generalization of the Poisson process which is obtained by allowing the rate at time t to be a function of t [14, 18, 21]. In other words, the stationary increment assumption of the Poisson process is relaxed. In the literature, it is also called the non-stationary Poisson process, the in-homogeneous Poisson process. Beside the definition of the non-homogeneous Poisson process, the following properties of the non-homogeneous Poisson process are presented:

- The inter-event time distribution of the non-homogeneous Poisson process
- The distribution of the area under the intensity function between successive events
- The distribution of the number of events on the simulation interval
- Order statistics
- Transforming a non-homogeneous Poisson process into a (homogeneous) Poisson process

Definition of the Non-Homogeneous Poisson Process: The definition of the non-homogeneous Poisson process [14, 19, 21] is as follows:

The counting process $\{N(t), t \geq 0\}$ is said to be a non-homogeneous Poisson process with the intensity function $\lambda(t)$, if

- $N(0) = 0$
- The process has independent increments.
- $P\{N(t+h) - N(t) = 1\} = \lambda(t)h + o(h)$
- $P\{N(t+h) - N(t) = 0\} = 1 - \lambda(t)h - o(h)$

The intensity function $\lambda(t)$ is a non-negative time varying function and assumed to be independent of the process history (independent increments). The number of events n ($n = 0, 1 \dots$) in any interval $(s, s + t]$ is Poisson distributed with parameter $\int_s^{s+t} \lambda(u) du \quad \forall s, t \geq 0$ [16, 21] and it is expressed as

$$P\{N(s, s + t] = n\} = P\{N(s + t] - N(s) = n\} = e^{-\int_s^{s+t} \lambda(u) du} \frac{(\int_s^{s+t} \lambda(u) du)^n}{n!}. \quad (2.24)$$

$E[N(t)]$ = Expected number of events generated by a non-homogeneous Poisson process with intensity function $\lambda(t)$ on the interval $(0, t]$ can be formulated as [14]

$$E[N(t)] = \int_0^t \lambda(u) du. \quad (2.25)$$

Equation 2.25 indicates the area under the intensity function $\lambda(t)$ of a non-homogeneous Poisson process on the interval $(0, t]$. It is also called the mean value function of the non-homogeneous Poisson process.

The Inter-event Time Distribution of the Non-Homogeneous Poisson Process: The inter-event times of the Poisson process with rate λ are exponentially distributed with mean $\frac{1}{\lambda}$ on the interval $(0, t]$ because the rate λ of the Poisson process is constant, doesn't change on the interval $(0, t]$ (stationary increments of the Poisson process). However, the non-homogeneous Poisson process doesn't have stationary increments i.e. its intensity function $\lambda(t)$ is a function of time t . The inter-event time distribution of the non-homogeneous Poisson process can be stated using the relationship between the intensity function of the non-homogeneous Poisson process and the probability density function of inter-event times,

$$\lambda(t) = \frac{f(t)}{1 - F(t)} = \frac{\frac{dF(t)}{dt}}{1 - F(t)} = -\frac{d}{dt} \log [1 - F(t)]. \quad (2.26)$$

Suppose S_{last} is the last known event time and S_{next} will be next event time then the inter-event time between these events $t = S_{next} - S_{last}$. The cumulative distribution function of t (inter event-time distribution of the non-homogeneous Poisson process

given the last event time S_{last} , $F(t; S_{last})$ can be derived by integrating Equation 2.26 on the interval $(S_{last}, S_{last} + t)$ with following intermediate steps

$$\int_{S_{last}}^{S_{last}+t} \lambda(u) du = \int_{S_{last}}^{S_{last}+t} -\frac{d}{dt} \log[1 - F(t)] dt, \quad (2.27)$$

$$\int_{S_{last}}^{S_{last}+t} \lambda(u) du = -\log[1 - F(t; S_{last})]_0^t, \quad (2.28)$$

$$\int_{S_{last}}^{S_{last}+t} \lambda(u) du = -\log[1 - F(t; S_{last})] + \log[1 - F(0; S_{last})], \quad (2.29)$$

where $F(0; S_{last}) = 0$ because F is a distribution function

$$\int_{S_{last}}^{S_{last}+t} \lambda(u) du = -\log[1 - F(t; S_{last})] \quad (2.30)$$

After rearranging Equation 2.30: The distribution function of the inter event time t given the last event time S_{last} is expressed as

$$F(t; S_{last}) = 1 - \exp\left(-\int_{S_{last}}^{S_{last}+t} \lambda(u) du\right). \quad (2.31)$$

As a summary, the inter-event time distribution of a non-homogeneous Poisson process with the intensity function $\lambda(t)$ depends on the last event time S_{last} in the process history. The inter-event time distribution of the non-homogeneous Poisson process doesn't have memoryless property because it doesn't satisfy the following statement for the memoryless property,

$$P\{X > s + t | X > t\} = P\{X > s\} \quad \forall s, t \geq 0. \quad (2.32)$$

Note: If we set $\lambda(t) = \lambda$ then

$$F(t; S_{last}) = 1 - \exp\left(-\int_{S_{last}}^{S_{last}+t} \lambda(u) du\right) \quad (2.33)$$

reduces to

$$F(t; S_{last}) = 1 - \exp(-\lambda t). \quad (2.34)$$

Equation 2.34 is the distribution function of inter-event times of a Poisson process with rate λ , independent of the last event occurred in process and satisfy the memoryless property condition.

The Distribution of the Area under the Intensity Function between Successive Events: The interpretation of E_i (the exponential random variate with mean one) is the area under the intensity function between the successive events. To show this, it is required to rearrange the distribution function of the inter-event times. The distribution function of t (the inter event time between S_{last} and S_{next}) is expressed as

$$F_{S_{next}}(t; S_{last}) = 1 - \exp\left(-\int_{S_{last}}^{S_{last}+t} \lambda(u) du\right) \quad \text{where } S_{next} = S_{last} + t. \quad (2.35)$$

F is a distribution function therefore it is uniformly distributed on $(0, 1]$. After rearranging Equation 2.35, it is reached that the area under the intensity function between the successive events is exponentially distributed with rate one. This is formulated as

$$\int_{S_{last}}^{S_{last}+t} \lambda(u) du = -\log(1 - U) = E. \quad (2.36)$$

The Distribution of the Number of Events on the Simulation Interval: If the counting process $N(t)$, $t \geq 0$ is a Poisson process with rate λ , then $N(t = T)$ is

Poisson distributed with mean λT , $N(t = T) \sim \text{Poisson}(\lambda T)$ [17, 24]. Furthermore, the number of events n occurred in the time interval $(0, T]$ can be expressed as

$$n = \sup\{n : \sum_{i=1}^n E_i \leq \lambda T\}. \quad (2.37)$$

" E_i " in Equation 2.37 is an exponential random variate with mean one. If the counting process $N(t)$, $t \geq 0$ is a non-homogeneous Poisson process with intensity function $\lambda(t)$, then $N(t = T)$ is Poisson distributed with mean $\int_0^T \lambda(u) du$, in other words $N(t = T) \sim \text{Poisson}(\int_0^T \lambda(u) du)$ [24]. The mean of the Poisson random variable is equal to the area under the intensity function of NHPP on the simulation interval $(0, T)$. Furthermore, the number of events n occurred in the time interval $(0, T]$ can be expressed

$$n = \sup\{n : \sum_{i=1}^n E_i \leq \int_0^T \lambda(u) du\}. \quad (2.38)$$

" E_i " in Equation 2.38 is an exponential random variate with mean one.

Order Statistics: Order statistic method is based on the theorem below: [25]:

Let $S_1, S_2, \dots$ be random variables representing the event times of a non-homogeneous Poisson process with intensity function $\lambda(t)$ and continuous cumulative intensity function $\Lambda(t)$ on the interval $(0, T)$, and let $N(t)$ represent the total number of events occurring before time t in the process. Then, conditional on the number of events $N(T) = n$, the event times $S_1, S_2, \dots, S_n$ are distributed as order statistics from a sample with the distribution function F ,

$$F(t) = \frac{\Lambda(t)}{\Lambda(T)} \quad \forall t \in (0, T]. \quad (2.39)$$

Note: The non-homogeneous Poisson process with the intensity function $\lambda(t)$ turns to the homogeneous Poisson process with rate λ if $\lambda(t) = \lambda$. Then the distribution

function F of the event times given the number of events turns to

$$F(t) = \frac{\Lambda(t)}{\Lambda(T)} = \frac{\lambda t}{\lambda T} = \frac{t}{T} \quad \forall t \in (0, T]. \quad (2.40)$$

Transforming a Non-homogeneous Poisson Process into a (Homogeneous) Poisson Process: Any non-homogeneous Poisson Process is a simple point process therefore it may be transformed into a homogeneous Poisson process with a unit rate [26]. A non-homogeneous Poisson process as a simple point process with intensity function, $\lambda(t)$ can be transformed into the homogeneous Poisson process with rate one by the variable change $t \mapsto \Lambda(t)$ where $\Lambda(t)$ is the cumulative intensity function of the non-homogeneous Poisson process ([16] page 258, Theorem 7.4.I). The distribution function of the inter-event times of the non-homogeneous Poisson process with intensity function $\lambda(t)$ can be written as

$$F(X = t; S_n) = 1 - \exp\left\{-\int_{S_n}^{S_n+t} \lambda(u) du\right\}. \quad (2.41)$$

Using the variable change,

$$t \mapsto \Lambda(t) \quad (2.42)$$

with $\Lambda(t) = \int_{S_n}^{S_n+t} \lambda(u) du$ and

- $du \mapsto \lambda(u) du$
- $S_n < S_n + u < S_n + t$
 $\mapsto S_n < \Lambda(S_n + u) < S_n + t$
 $\mapsto \Lambda^{-1}(S_n) < S_n + u < \Lambda^{-1}(S_n + t)$

Equation 2.41 turns to

$$F(\Lambda(t); S_n) = 1 - \exp\left\{-\int_{\Lambda^{-1}(S_n)}^{\Lambda^{-1}(S_n+t)} \lambda(\Lambda(t)) \lambda(t) dt\right\} \quad (2.43)$$

and

$$F(\Lambda(t); S_n) = 1 - \exp\left\{- \int_{\Lambda^{-1}(S_n)}^{\Lambda^{-1}(S_n+t)} \lambda(\Lambda(t)) \lambda(t) dt \right\}. \quad (2.44)$$

The anti-derivative of $[\lambda(\Lambda(t))\lambda(t)]$ is $\Lambda(\Lambda(t))$. After evaluating the integral, Equation 2.44 can be written as

$$F(\Lambda(t); S_n) = 1 - \exp\left\{- \Lambda(\Lambda(t)) \Big|_{\Lambda^{-1}(S_n)}^{\Lambda^{-1}(S_n+t)}\right\}, \quad (2.45)$$

$$F(\Lambda(t); S_n) = 1 - \exp\{-\Lambda(S_n + t) + \Lambda(S_n)\}, \quad (2.46)$$

because $\Lambda(t) = \int_{S_n}^{S_n+t} \lambda(u) du$ and

$$F(\Lambda(t); S_n) = 1 - \exp\{-\Lambda(t)\}. \quad (2.47)$$

As a summary, the random variable $\Lambda(X)$ is exponentially distributed with unit rate and time re-scaling theorem indicates that any non-homogeneous Poisson process (which is also a simple point process) with an integrable intensity function may be transformed into a Poisson process with unit rate. The variables represent:

- t represents the time between two consecutive events (inter-event time) of the non-homogeneous Poisson process with intensity function $\lambda(t)$.
- $\Lambda(t)$ represents the area under the intensity function $\lambda(t)$ between two consecutive event times of the non-homogeneous Poisson process.

2.3. The Hawkes (Self-Exciting) Process

Hawkes introduced a new class of point process which he calls self-exciting process [1]. Later, it is called Hawkes Process. Hawkes Process is a doubly stochastic process

whose conditional intensity function is stochastic and depends on the history of the process. A doubly stochastic process models an observed random variable in two steps. In first step, the distribution of the random variable which represents the events of the process is in a standard way using one or more parameters. In the second step, some of these parameters are themselves random variables. [27–29].

Definition of Hawkes (Self-Exciting) Process [1, 2, 24]: The conditional intensity function of Hawkes process $\lambda(t|H_t)$ takes the form as follows

$$\lambda(t|H_t) = \lambda_0 + \int_0^t \varphi(u) dN(u) = \lambda_0 + \sum_{i:t > t_i} \varphi(t - t_i). \quad (2.48)$$

$\lambda_0 : R \rightarrow R^+$ is the base intensity function.

$\varphi(.) : R \rightarrow R^+$ is called the (memory kernel) excitation function.

$t_i : t_i < t$, is the event time having occurred before current time t , and which contributes to the event intensity at time t .

H_t : the set of the occurrence times t_i 's of the events that happened before time t

The simple counting process $\{N(t), t \geq 0\}$ is said to be a Hawkes self-exciting (Hawkes) process with the conditional intensity function $\lambda(t|H_t)$, $t \geq 0$, if

- $N(0) = 0$
- $P\{N(t+h) - N(t) = 1\} = \lambda(t|H_t) h + o(h)$
- $P\{N(t+h) - N(t) \geq 2\} = o(h)$
- $P\{N(t+h) - N(t) = 0\} = 1 - \lambda(t|H_t) h - o(h)$

The conditional intensity function $\lambda(t|H_t)$ depends on the history encoded by H_t . In this case, the function $\lambda(t|H_t)$ is itself a stochastic process because it depends on the event times $\{S_1, S_2, \dots, S_{N(t)}\}$, which are themselves random. Hawkes process doesn't have stationary increments because its conditional intensity function is a time varying function. Hawkes process doesn't have independent increments because its conditional intensity function depends on the events in the process history.

The Base Intensity Function: The function $\lambda_0(t) \geq 0$ is the base (or background) intensity function of Hawkes process, describing the arrival of events triggered by external sources. These events are also known as exogenous or immigrant events, and their arrival is independent of the previous events within the process. In the original form of Hawkes process [1], base intensity function is a constant rate (a scalar) during the simulation interval. It is assumed that the rate of immigrants to a new area is constant over the time. The inter-arrival time of immigrants is exponentially distributed. The immigrants are modeled by a homogeneous Poisson process. The constant function can be written as

$$\lambda_0(t) = \lambda_0. \quad (2.49)$$

The base intensity function can be a time varying function. In this case, the immigrants are modeled by a non-homogeneous Poisson process and the inter arrival times of the immigrants aren't exponentially distributed. This relaxation of constant base intensity function brings the flexibility to Hawkes process to model real life cases for example the social media case. This subject is going to be explained in Section 2.5 in the part "renewal Hawkes process". The base intensity function can be a periodic function. An example for the periodic function is the sine function. It models the rate of immigrants which is increasing during the hot seasons and is decreasing during the cold seasons. The sine function with positive values can be written as

$$\lambda_0(t) = \sin(t) + 1. \quad (2.50)$$

The +1 in Equation 2.50 assures that the intensity function greater than or equal zero.

The Excitation Function: The function $\varphi(t) \geq 0$ is the excitation function of Hawkes process, describing the arrival of events triggered by internal sources, by the events in the Hawkes process history, H_t . The excitation function is also called infectivity function, fertility function, or memory kernel in the literature. Each event joined to the Hawkes process history H_t (as an immigrant which is generated by immigrant

sub-process or as an offspring which is generated by offspring sub-process) triggers an offspring sub-process with the excitation function $\varphi(t)$ to generate its direct offspring. The self-exciting feature of Hawkes process arises through the summation term of its conditional intensity function, where the excitation function $\varphi(t-t_i)$ modulates the change that an event at time t_i has on the intensity function at time t [2]. Typically, the excitation function $\varphi(t)$ is taken to be monotonically decreasing so that more recent events have higher influence on the current event intensity, compared to the events having occurred further away in time. However, the excitation function $\varphi(t)$ doesn't have to be monotonically decreasing because there are cases in which human behavior affects the situation like the interest of users to a post in the social media. The interest of the users for a post first increases and then decreases as time passes. There are several reasons for the behavior change like the daily routine of the user, the time zone of the user etc. As excitation function for this case, the Weibull distribution with shape parameter $k = 1.5$ and scale parameter $\lambda = 1$ can be an example.

The frequently used excitation functions for Hawkes process in the literature are exponential decay and power law function because the influence of an event will naturally decrease over time. Furthermore these excitation functions $\varphi(t)$ have the closed form of their cumulative intensity functions $\phi(t)$ and their inverse cumulative intensity functions $\phi^{-1}(t)$ which are useful to simulate the processes. The excitation functions, exponential decay and power law with their cumulative and inverse cumulative functions are presented below.

Exponential decay function: In many natural phenomena, quantities grow or decay at a rate proportional to their size. The most popular excitation function for Hawkes process is the exponential decay function in the literature. The exponential decay function can be written as

$$\varphi(t) = \alpha e^{-\beta t} \quad (2.51)$$

where $t \geq 0$ $\alpha, \beta > 0$ and $\beta > \alpha$. The condition $\beta > \alpha$ assures that the exponential

function is a monotonically decreasing function and the total area under the exponential decay excitation function on the time interval $(0, \infty)$ is less than one, which is mathematically formulated as

$$\int_0^{\infty} \alpha e^{-\beta t} dt < 1, \quad (2.52)$$

α : accounts for the magnitude of influence (initial value),

$e^{-\beta t}$: models the memory over time and

β : the exponent that describes how fast an event is forgotten.

Each event in the process instantaneously increases the intensity by α , then over time this event's influence decays at rate β .

The cumulative exponential decay function is expressed as

$$\Phi(t) = \int_0^t \alpha e^{-\beta u} du = \frac{\alpha}{\beta} (1 - e^{-\beta t}) \quad (2.53)$$

and the inverse of the cumulative exponential decay function (Quantile function) is expressed as

$$\Phi^{-1}(t) = -\frac{\log(1 - t \frac{\beta}{\alpha})}{\beta} \quad (2.54)$$

The popularity of the exponential decay function comes from its modeling structure of the memory over time because the rate of decay, β is constant for all t according to the law of exponential change that is expressed as [30].

$$\frac{dy}{dt} = ky \quad (2.55)$$

where $k = -\beta$ and y is the present value. Equation 2.55 is the mathematical formulation of the sentence "quantities decay at a rate proportional to their size."

The summation part of the Hawkes process conditional function consists of the contribution (excitation) of each event in process history H_t . In other words, $\varphi(t - t_i)$ has to be computed separately for each event t_i in the process history H_t and summed up. This is computationally time-consuming operation as the number of events in process history increases. However, if the excitation function is exponential decay then the value of the summation part of the conditional intensity function at the last event time in process history is updated by multiplying it with $e^{-\beta t}$ where t is time between the new event generated and the last event in process history. This feature is the advantage of exponential decay function to reduce the number of operations to calculate the summation part of the Hawkes process conditional intensity function.

Power law function: Another popular excitation function for Hawkes Process is the power law excitation function because power law function indicates that extreme values are more frequent than they are expected. Power law is a functional relationship between two quantities where one quantity varies as a power of another quantity. The power law decay function is expressed as

$$\varphi(t) = \alpha(t + c)^{-(\beta+1)} \quad (2.56)$$

where $t \geq 0$, $\alpha, \beta, c > 0$, and $\alpha < \beta c^\beta$. The condition $\alpha < \beta c^\beta$ assures that power law function is monotonically decreasing function and the total area under the power law decay excitation function on the time interval $(0, \infty)$ is less than one which is mathematically formulated as

$$\int_0^\infty \alpha(t + c)^{-(\beta+1)} dt < 1 \quad (2.57)$$

α : accounts for the magnitude of influence (initial value),

c : a temporal shift term to keep $\varphi(t)$ bounded when " t " is closed to 0,

$(t + c)^{-(\beta+1)}$: models the memory over time and

$1 + \beta$: the power-law exponent that describes how fast an event is forgotten.

Unlike to the exponential decay function, its rate of decay depends on time, in this case, $\frac{1+\beta}{t+c}$.

The cumulative power law function is expressed as

$$\phi(t) = \int_0^t \alpha(y+c)^{-(\beta+1)} dy = \frac{\alpha}{\beta} (c^{-\beta} - (t+c)^{-\beta}) \quad (2.58)$$

and the inverse of the cumulative power law function (Quantile function) is expressed as

$$\Phi^{-1}(t) = \left(\frac{\alpha c^\beta}{\alpha - t\beta c^\theta} \right)^{\frac{1}{\beta}} - c. \quad (2.59)$$

The additional important points about the excitation function:

- The area under the excitation function determines the expected number of offspring for each parent event.
- The excitation function models the inter event time distribution between parent event and its offspring events.

2.4. Cluster Process Representation of Hawkes Process

The self-exciting (Hawkes) process can be represented as cluster process [7]. The general cluster model models the immigrants and its direct offspring. In other words, there are several clusters with immigrant events as cluster centers and with immigrants direct offspring, first generation events. The Galton Watson branching process models a single immigrant and its direct and indirect offspring. In other words there is a single cluster with an immigrant event as cluster center, but this cluster contains all of the descendants of the immigrant. Each event in the cluster has the feature to generate further offspring (self-exciting feature). Hawkes process can be seen as a combination of these two processes. Hawkes process can contain multiple clusters and all the events in the clusters can generate further offspring (self-exciting feature). The features of

Galton Watson branching process are useful to calculate the expected number of events of Hawkes process.

The Cluster Point Process: The events in a point process can be classified as immigrants and offspring if each immigrant can generate its offspring. A cluster point process is generated by first constructing a point process of immigrant events, each of which give rise to a random number of offspring events. The process of offspring events is the cluster process [31]. If the number of offspring events follows a Poisson distribution, the process is called a Cox Poisson Cluster Process. The general cluster model satisfies the following two conditions [32]:

- (i) The event times of the immigrants are S_n , $n = 1, 2, \dots$
- (ii) The event times of the offspring events which are triggered by the n^{th} immigrant are $S_{nj} = S_n + \sum_{k=1}^j X_k$ where X_k 's are inter-event times between offspring.

The cluster mechanism is also a natural way to describe the locations of individuals from consecutive generations of a branching process. Hawkes processes which can be represented as clusters can be used in social media and disease spread which are very popular in today's world.

Galton Watson Branching Process and Branching Factor: The Galton–Watson branching process is a branching stochastic process arising from Francis Galton's statistical investigation of the extinction of family names. It is the simplest and most frequently applied branching process, a type of discrete-time Markov chain that is, collection of discrete random variables, X_n , $n = (0, \infty)$, where the time $n = 0, 1, 2, \dots$ is also discrete [33]. The random variable X_n may represent the population size at time n or generation n . The term chain implies that each of the random variables are discrete valued, their values come from the set of non-negative integers $\{0, 1, 2, \dots\}$. The Markov property means that the population size at time " $n+1$ " is only dependent on the population size at time " n " and doesn't depend on the size at earlier times. In this way, the population size X_n at time n predicts the population size in the next

generation, X_{n+1} [33]. The Three important assumptions about the Markov process $X_n, n = 0, \dots, \infty$, define a single-type Galton-Watson branching process (GWbp).

- Every individual in the population in generation "n" gives birth to Y offspring of the same type in the next generation, where Y is a discrete random variable that takes values in $\{0, 1, 2, \dots\}$. The offspring probabilities of Y are $p_j = P\{Y = j\}$, where $j = 0, 1, 2, \dots$
- Every individual in the population gives birth independently of all other individuals.
- The same offspring distribution applies to all generations.

Galton Watson branching process is a class of Markov chain [14]. The branching factor of branching process n^* is the mean number of direct offspring generated by each parent event, the out-degree. Each event can produce j new offspring with probability $P_j \geq 0, j \geq 0$, independently of the numbers produced by other events.

The branching factor n^* is a key quantity in the cluster-process representation of Hawkes processes because the number of events generated by an event in the process history (immigrant or offspring) is Poisson distributed with parameter n^* that is equal to the area under the excitation function $\varphi(t)$ on the interval $(0, \infty)$ when simulating Hawkes process by clusters is expressed as [16]

$$n^* = \int_0^\infty \varphi(u) du. \quad (2.60)$$

The events of the Hawkes process are generated generation by generation when simulating Hawkes process by clusters. Each immigrant event forms a cluster. To compute the expected number of events of each cluster: Let G_i represent the number of offspring events in generation i and $G_0 = 1$ which is the immigrant event as the

cluster center. The number of offspring events in generation 1,2,..., are expressed as

$$G_1 = n^* \quad , \quad G_2 = (n^*)^2, \dots \quad (2.61)$$

In the similar way, the number of offspring events in generation i , is expressed as

$$G_i = (n^*)^i. \quad (2.62)$$

If $n^* < 1$, expected number of the total events including the immigrant event in a cluster can be written [16]

$$E[N(t = \infty)] = \sum_{i=0}^{\infty} G_i = \sum_{i=0}^{\infty} (n^*)^i = \frac{1}{1 - n^*} \quad \text{where } n^* < 1. \quad (2.63)$$

$\Rightarrow$ Expected number of total events is bounded if $n^* < 1$.

If $n^* \geq 1$, expected number of total events is unbounded that is stated as

$$E[N(t = \infty)] = \sum_{i=0}^{\infty} G_i = \sum_{i=0}^{\infty} (n^*)^i = \infty \quad \text{where } n^* \geq 1. \quad (2.64)$$

$\Rightarrow$ Expected number of total events is unbounded if $n^* \geq 1$.

The branching factor n^* in Equation 2.60 (also called reproduction number or total load of excitation function) plays an important role about the ratio of total number of immigrant events and offspring events. Due to the self-exciting nature of Hawkes process, each immigrant event forms its cluster. The combination of all of the clusters constitutes Hawkes process event set. The expected number of events generated by an immigrant (including the immigrant and all its direct and indirect offspring) is a geometric series which is the sum of all events. It can be written as

$$\sum_{i=0}^{\infty} (n^*)^i = \frac{1}{1 - n^*}. \quad (2.65)$$

For each immigrant event which is the center of a cluster, the expected number of all of its direct and indirect offspring events (including all of its descendants) can be written as

$$\frac{1}{1 - n^*} - 1. \quad (2.66)$$

Expected number of the immigrant events is equal to the area under base intensity function $\lambda_0(t)$ is $\int_0^\infty \lambda_0(u) du$. This number is also equal to the expected number of the clusters which are foremd by Hawkes process. Expected number of the Hawkes process events is expressed as

$$\int_0^\infty \lambda_0(u) du \frac{1}{1 - n^*}. \quad (2.67)$$

Each cluster contains an immigrant event as cluster center and its descendants. The branching factor n^* (the expected number of direct offspring generated by the events in the Hawkes process history H_t) affects the ratio of total number of immigrant events and offspring events.

- If the branching factor $n^* < 0.5$ then the expected number of immigrants is greater than the expected number of offspring.
- If the branching factor $n^* = 0.5$ then the expected number of immigrants is equal to the expected number of offspring.
- If the branching factor $n^* > 0.5$ then the expected number of immigrants is less than the expected number of offspring.
- If the branching factor $n^* \geq 1$ then the expected number of offspring is infinity.
- If the branching factor $n^* < 1$ then the expected number of offspring is finite.

Therefore an efficient non-homogeneous Poisson process generation method for immigrant and offspring sub-process simulation plays an important role at the performance of the cluster-based (composition) algorithm for the Hawkes process simulation.

Hawkes Process by Clusters: One reason for Hawkes process' popularity is that Hawkes process as self-exciting process combines in a model both a cluster process representation and a simple conditional intensity function, which is linear so that Hawkes process can be decomposed into sub-processes. Hawkes and Oakes [7] introduced an equivalent view of the Hawkes process which refers to the Poisson cluster process interpretation, which separates the events of a Hawkes process into immigrant events and offspring events. The offspring events are generated by an existing event in the process history, while the immigrant events arrive independently and thus don't have an existing parent event. The offspring events are said to be structured into clusters, associated with each immigrant event [2,16]. Hawkes and Oakes proved that a Poisson cluster process exists satisfying the axioms of a self-exciting process, and then showed that there is only one finite intensity orderly process satisfying the intensity function of Hawkes Process and $0 < \int_0^\infty \varphi(t)dt < 1$ for a given non-negative excitation function $\varphi(\cdot)$ in [7]. The following lemmas are used in the proof. Lemma 1 from [7] states that if $\lambda_0 > 0$ and $\int_0^\infty \varphi(\tau)d\tau < 1$, then there exists a stationary Poisson cluster process with rate

$$\lambda = \frac{\lambda_0}{1 - \int_0^\infty \varphi(\tau)d\tau}. \quad (2.68)$$

Lemma 2 from [7] states that there exists at most one stationary orderly point process of finite rate whose complete intensity is given by

$$\lambda(t|H_t) = \lambda_0 + \int_{-\infty}^t \varphi(t-s) dN_s \quad (2.69)$$

with $\lambda_0 > 0$, $\varphi(\tau) \geq 0$ for $\tau \geq 0$, $0 < \int_0^\infty \varphi(\tau)d\tau < 1$.

As Hawkes and Oakes stated in the proof that Hawkes (self-exciting) process is the modified version of a pure birth and death process [34] by

- setting death rate $\mu=0$
- setting birth rate (for offspring) $\lambda = \varphi(t)$ (excitation function of Hawkes process)

- adding immigration at rate λ_0 (base intensity of Hawkes process)

then the intensity of the process turns to the conditional intensity function of a Hawkes process that is

$$\lambda(t|H_t) = \lambda_0 + \int_0^t \varphi(u) dN(u) = \lambda_0 + \sum_{i:t>t_i} \varphi(t-t_i). \quad (2.70)$$

This process can be seen as a cluster process where the immigrant process (as a Poisson process with the rate λ_0) generates the immigrants as the cluster centers $N_c(t)$. For each immigrant in $N_c(t)$, the clusters are built by the offspring process (as a non-homogeneous Poisson process with excitation function $\varphi(t)$) which generates all the descendants of all generations of the immigrants. These clusters are mutually independent by construction [7]. In other words, the events which are directly or indirectly connected to an immigrant (a cluster center) form the cluster of offspring associated with that immigrant.

The conditional intensity function of Hawkes process, $\lambda(t|H_t)$ is the summation of λ_0 and $\varphi(\cdot)$'s. The Hawkes process is the composition (superposition) of the subprocesses called immigrant process and offspring processes with respective intensity functions, λ_0 and $\varphi(\cdot)$ that are independent non-homogeneous Poisson processes. The theorem about the superposition of the general point processes satisfying complete independence property can be found at [16] p.38, Theorem 2.4.VI.

Each event (immigrant or offspring) in a cluster can generate offspring (self-exciting nature of Hawkes process). The events in the clusters are generated one generation after the other. The number of events in the generation " n " depends only on the number of events in previous generation " $n-1$ ". Every individual in a cluster gives offspring following a Poisson distribution with mean which is equal to the area under the excitation function $\varphi(t)$ and gives offspring independently of all other individual. This process satisfies the conditions of Galton Watson branching process.

Any event of the process which is generated at time t_i , has the potential to produce further events whose locations are those of a finite Poisson process with intensity measure $\varphi(t-t_i)$. It is assumed that $\varphi(\cdot)$ has total mass (total area under the excitation function $\varphi(\cdot)$) $\nu \equiv \int_0^\infty \varphi(\cdot) < 1$. Total mass ν of the excitation function $\varphi(t)$ is equal to the branching factor " n^* " of Galton Watson branching process. All these finite Poisson processes are mutually independent and, given the event that generates them, identically distributed and independent of the immigrant process as well. Consequently, each event of the process has the potential to produce descendants (offspring events) whose numbers in successive generations constitute a Galton-Watson branching process with Poisson offspring distribution whose mean is ν . Since $\nu < 1$, this branching process is subcritical and therefore the total size of the branching process with mean $1/(1-\nu) < \infty$ including the initial immigrant member is finite. Regard the totality of all descendants of a given immigrant event generated at time t_i as a cluster center; then the totality of all such immigrant events and their clusters constitutes a Hawkes process [16]. Daley and Vere-Jones [16] indicated that Hawkes (self-exciting) process can be used in infectivity models because of its cluster-based representation. The immigrants are infected individuals from outside of the system. The clusters which immigrants generate are the process of new infections they produce. The infections from inside of the system are offspring.

2.5. The Extensions of Hawkes Process

There are many extensions of Hawkes processes in the literature. Two of the most popular of them are

- (i) Renewal Hawkes process
- (ii) Marked Hawkes process

They are applied to model the real life situations, like disease spread, earthquakes, stock markets. The conditional intensity function of the original Hawkes process is stated in Equation 2.48. The base intensity function for immigrant sub-process is a

scalar λ_0 . This implies that the inter-event time distribution of immigrant events is exponential distribution with rate λ_0 [35]. The excitation function $\varphi(t)$ is same for all offspring sub-processes. This indicates that the expected number of offspring for all offspring sub-processes which are triggered by an event in process history H_t is same. These two conditions are relaxed by the extensions of Hawkes process.

Renewal Hawkes Process: The renewal Hawkes process is an extension of the Hawkes process introduced by Wheatley et al. [13]. In the original context of the Hawkes process, the base intensity function $\lambda_0(\cdot)$ is a scalar λ where the inter-event times of immigrant sub-process of Hawkes process is exponentially distributed with rate λ and immigrant sub-process is a Poisson process with rate λ . The conditional intensity function of renewal Hawkes process, $\lambda(t|H_t)$ takes the form that can be written as

$$\lambda(t|H_t) = \lambda_0(t) + \int_0^t \varphi(u) dN(u) = \lambda_0(t) + \sum_{i:t>t_i} \varphi(t-t_i). \quad (2.71)$$

where the base intensity function with constant rate λ_0 is replaced with a time varying intensity function, $\lambda_0(t)$. The Hawkes process is generalized by replacing the Poisson immigration process with a renewal process [13]. According to the cluster process representation of a self-exciting (Hawkes) process [7] which is stated in Section 2.4, the (immigrant and offspring) sub-processes are (non-homogeneous) Poisson processes. The relationship between the intensity function of a NHPP $\lambda(t)$ and the probability density function $f(t)$ of inter-event times of a NHPP is stated in Equation 2.17. If the probability density function $f(t)$ of inter-event times of immigrant sub-process is given, then the base intensity function $\lambda_0(t)$ of a renewal Hawkes process can be computed using that the equation that is expressed as

$$\lambda_0(t) = \frac{f(t)}{1 - F(t)} \quad (2.72)$$

where $F(\cdot)$ is the distribution function of inter-event times, integral of $f(\cdot)$ and $1 - F(\cdot)$ is survival function.

Some of the inter-event time probability density functions and their related intensity functions for the point processes are listed in Table 2.1 below:

Table 2.1. The probability density function and its related intensity function

Inter-event time		$f(t)$	$F(t)$	$\lambda(t) = \frac{f(t)}{1-F(t)}$
Exponential dist.	rate $\lambda > 0$	$\lambda e^{-\lambda t}$	$1 - e^{-\lambda t}$	scalar λ
Weibull dist.	shape $k > 0$	$\lambda k (\lambda t)^{k-1} e^{-(\lambda t)^k}$	$1 - e^{-(\lambda t)^k}$	$k \lambda^k t^{k-1}$
	with rate $\lambda > 0$			
Weibull dist.	shape $k > 0$	$k t^{k-1} e^{-t^k}$	$1 - e^{-t^k}$	$k t^{k-1}$
	with rate $\lambda=1$			
Weibull dist.	shape $k = 1$	e^{-t}	$1 - e^{-t}$	scalar 1
	with rate $\lambda=1$			
Power law dist.	exponent $\beta > 0$	$k(t+c)^{-(\beta+1)}$	$1 - \frac{k}{\beta}(t+c)^{-\beta}$	$\frac{\beta}{t+c}$
	temporal shift c	where $k=\beta c^\beta$		

Note : Weibull distribution with parameters, shape $k = 1$ and rate λ reduces to exponential distribution with rate λ .

Expected total number of events generated by a renewal Hawkes process with base function $\lambda_0(t)$ and excitation function $\varphi(\cdot)$ is as follows: Expected number of immigrants of renewal Hawkes process = Total area under base intensity function which is equal to $\int_0^\infty \lambda_0(u) du$. Expected total number of events (including the immigrant event and its direct and indirect offspring events) in a cluster triggered by an immigrant can be written as

$$\frac{1}{1 - n^*} \quad (2.73)$$

where n^* is the branching factor stated at Equation 2.60 and $0 < n^* < 1$. Expected total number events of a renewal Hawkes process can be written as

$$\frac{\int_0^\infty \lambda_0(u) du}{1 - n^*}. \quad (2.74)$$

Note: The sub-processes are independent of each other and $\lambda_0(t) > 0$, $\forall t \geq 0$ and $0 < n^* < 1$.

Marked Hawkes Process: The other extension of the Hawkes process is the marked Hawkes process [2, 12, 36]. Each event which is generated as immigrant or offspring can generate its direct offspring. The excitation function $\varphi(t)$ of offspring sub-process of the Hawkes process has two important duties which are

- assigns the (magnitude of) influence for each event in the process history and
- models the magnitude of influence over time

In the original Hawkes process definition, the (magnitude of) influence of each event and its influence model over time are same (in other words the excitation function for each offspring sub-process is same). However, in real life, like seismology, finance, social media etc., they can be different. For example, in the social media case, the propagation of a post (offspring events) depends on the number of followers of the social media user who received the post as a parent event. To generate such a self-exciting process, the mark, " m " is added to the excitation part of the Hawkes process conditional intensity function that is formulated as

$$\lambda(t|H_t) = \lambda_0 + \int_0^t m\varphi(u)dN(u) = \lambda_0 + \sum_{i:t>t_i} m \varphi(t-t_i) \quad (2.75)$$

and the mark " m " can be defined as an independent and identically distributed random number, " m_i ".

The aim to define the marks to the events is to differentiate the events in terms of the (magnitude of) influence or in terms of the model of influence over time to simulate the offspring sub-process triggered by itself to generate its direct offspring. The original Hawkes process definition can be seen as a marked Hawkes process with mark = 1 for all the events in process history.

The mark "m" which is defined as independent and identically distributed random number affects only the (magnitude of) influence of the event and doesn't affect the model of influence over time used by the offspring sub-process to generate the event's direct offspring event times. For example, for the exponential decay excitation function stated in the Equation 2.51 where α is the influence (initial value) of the event and $e^{-\beta t}$ models its influence over time. The expected number of events (the area under the excitation function) of the offspring process is changed as "m" times of $\int_0^t \varphi(u)du$. However, the influence model over time (the structure of the excitation function) remains the same.

According to the "Order Statistics" which is presented at Section, 2.2: Given the number of events "n" which are going to be generated by offspring sub-processes with excitation functions $\varphi(t)$ and $m\varphi(t)$ respectively, the event times which are generated by the sub-processes are same if the same seed is used to simulate the process. This situation can be explained using the distribution function in Order Statistics: the distribution function of the offspring sub-process with the excitation function "m $\varphi(t)$ " is expressed as

$$F(t) = \frac{m\Phi(t)}{m\Phi(T)} = \frac{\Phi(t)}{\Phi(T)} \quad (2.76)$$

that is same as the distribution function F of the offspring sub-process with the excitation function " $\varphi(t)$ " that is expressed as

$$F(t) = \frac{\Phi(t)}{\Phi(T)}. \quad (2.77)$$

The marked Hawkes process cannot be represented as Galton Watson branching process because the expected number of offspring are different for each parent event. As examples of marked Hawkes process, Rizoiu et al. [2] applied the marked Hawkes process for information diffusion in social media and Lee and Seo [12] implemented the marked Hawkes process in finance.

2.6. Parameter Estimation

The intensity function and its parameters $\Theta = \{\theta\}$ of a random process describe the random process. The estimation of them is based on the observed events of the random process. The following methods and algorithms are used to estimate the parameters of the intensity functions in the literature:

- The maximum likelihood estimation
- The expectation maximization algorithm
- The method of moments

The Maximum Likelihood Estimation: One of the popular statistical inference method is maximum likelihood estimation (MLE). MLE is a method for deriving the most likely estimators of the unknown parameters based on the observed data set [37]. The performance of MLE depends on the chosen intensity function with the parameter set θ and the number of points (observations) t_i , $i = 1...n$ which are collected. The intensity function $\lambda(t; \theta)$ of a random process is a non negative function. Therefore, the cumulative intensity function (compensator, $\Lambda(t; \theta)$) is monotonically non decreasing function. The difference between intensity function and the likelihood function is as follows:

- Intensity function $\lambda(t; \theta)$: The intensity function is known and its parameter set θ is fixed. The intensity function value at the observation t_i is computed with the intensity function. The intensity function is known.
- Likelihood function $L(\theta; t)$: Based on the events t_i , $i = 1...n$ which are observed, the likelihood of the parameters $L(\theta; t)$ for the chosen intensity function is computed. It is used to estimate the parameters of the chosen intensity function. The observations are known.

The likelihood function $L(\theta; t)$ can be stated in terms of probability density function $f(t_i; \theta)$ as

$$L(\theta; t) = \left[\prod_{i=1}^{n(T)} f(t_i) \right]. \quad (2.78)$$

The log-likelihood function $l(\theta; t)$ can be stated in terms of probability density function $f(t_i; \theta)$ as

$$l(\theta; t) = \left[\sum_{i=1}^{n(T)} \log(f(t_i; \theta)) \right]. \quad (2.79)$$

The connection between the probability density function $f(t_i; \theta)$ and the intensity function $\lambda(t_i; \theta)$ is expressed as (same as in Equation 2.72)

$$\lambda(t) = \frac{f(t)}{1 - F(t)} \quad (2.80)$$

where $F(t)$ is the cumulative distribution function and $1 - F(t)$ is the survival function.

As stated at Proposition 7.2.III in Daley and Vere-Jones [16], the likelihood function $L(\theta; t)$ of a regular point process N with the conditional intensity function $\lambda(t; \theta)$ on the time interval $(0, T < \infty]$ with all observed event times $t_1, t_2, \dots, t_{N(T)}$ is expressed as

$$L(\theta; t) = \left[\prod_{i=1}^{n(T)} \lambda(t_i; \theta) \right] \exp \left(- \int_0^T \lambda(u; \theta) du \right). \quad (2.81)$$

This is the general form of the likelihood function for a large class of point processes, including Hawkes processes. The proof can be found at Laub et al. [24]. For the proof, Equation 2.80 must be rearranged as " $f(t) = \dots$ " and be inserted in Equation 2.78 so that the likelihood function can be written in terms of $\lambda(t_i; \theta)$. To achieve this, Equation 2.80 must be integrated on the time interval (t_k, t) where (t_k) is last event in

the process history, then must be rearranged as

$$F(t) = 1 - \exp\left(-\int_{t_k}^t \lambda(u)du\right). \quad (2.82)$$

Equation 2.82 is same as Equation 2.31 in Section 2.2, the inter-event time distribution of the non-homogeneous Poisson process. The probability density function $f(t)$ is calculated by taking the derivative of Equation 2.82. The probability density function $f(t)$ is expressed as

$$f(t) = \lambda(t) \exp\left(-\int_{t_k}^t \lambda(u)du\right). \quad (2.83)$$

When substituting $f(t)$ in Equation 2.83 into Equation 2.78 and including the probability of there is no event on the time interval $(t_{N(T)}, T)$, the likelihood function $L(\theta; t)$ in terms of $\lambda(t)$ can be written as

$$L(\theta; t) = \left[\prod_{i=1}^{n(T)} \lambda(t_i) \right] \exp\left(-\int_0^T \lambda(u)du\right). \quad (2.84)$$

In Equation 2.83 the term $\exp(-\int_{t_k}^t \lambda(u)du)$ is survival function $1 - F(t; t_k)$ for the event t which will occur after the event t_k . The sum of the survival function values for all of the observed events $t_1, t_2, \dots, t_{N(T)}$ during the simulation period $(0, T)$ including the survival function value on the interval $(t_{N(T)}, T)$ (where there is no event occurred) is equal to the total area under the intensity function $\int_0^T \lambda(u)du$ on the simulation interval $(0, T)$. The form of the log-likelihood function $l(\lambda)$ in terms of intensity function is formulated as

$$l(\theta; t) = \left[\sum_{i=1}^{n(T)} \log(\lambda(t_i; \theta)) \right] - \int_0^T \lambda(u; \theta)du. \quad (2.85)$$

When the log-likelihood function in terms of probability density function in Equation 2.79 and in terms of intensity function in Equation 2.85 are compared, the difference is the total area under the intensity function $\lambda(u; \theta)$ which is equal to $\int_0^T \lambda(u; \theta)du$. This

formulated as

$$\left[\sum_{i=1}^{n(T)} \log(f(t_i; \theta)) \right] = \left[\sum_{i=1}^{n(T)} \log(\lambda(t_i; \theta)) \right] - \int_0^T \lambda(u; \theta) du. \quad (2.86)$$

The MLE $\hat{\lambda}$ of the intensity rate of homogeneous Poisson process can be computed as below: Using the Equations 2.81 and 2.85, the maximum likelihood estimator $\hat{\lambda}$ of a homogeneous Poisson process with rate λ on the time interval $(0, T < \infty]$ with all observed event times $t_1, t_2, \dots, t_{N(T)}$ can be computed as follows:

- Compute the likelihood function $L(\lambda)$

The likelihood function $L(\lambda)$ is expressed as

$$L(\lambda) = \left[\prod_{i=1}^{n(T)} \lambda \right] \exp \left(- \int_0^T \lambda du \right). \quad (2.87)$$

- Compute the log-likelihood function $l(\lambda)$

The log-likelihood function $l(\lambda) = \log L(\lambda)$ is expressed as

$$l(\lambda) = \sum_{i=1}^{n(T)} \log(\lambda) - \int_0^T \lambda du \quad (2.88)$$

and

$$l(\lambda) = n(T) \log(\lambda) - \lambda T. \quad (2.89)$$

- Take the derivative of $l(\lambda)$ with respect to λ and set it equal to zero that can be written as

$$\frac{n(T)}{\lambda} - T = 0. \quad (2.90)$$

- The maximum likelihood estimator of λ is obtained as

$$\hat{\lambda} = \frac{n(T)}{T}. \quad (2.91)$$

The log-likelihood function formula for the Hawkes processes can be derived by replacing $\lambda(t_i)$ in Equation 2.85 with the conditional intensity function of the Hawkes process $\lambda(t|H_t)$ in Equation 2.48 and calculating the total area under $\lambda(t|H_t)$ on the simulation interval $(0, T]$. The log-likelihood function of Hawkes process is expressed as [24]

$$l(\theta; t) = \sum_{i=1}^{n(T)} \log \left[\lambda_0 + \sum_{j=1}^{i-1} \varphi(t_i - t_j) \right] - (\lambda_0 T + \sum_{i=1}^{n(T)} \Phi(T - t_i)) \quad (2.92)$$

where

- $\Phi(t) = \int_0^t \varphi(u) du$ is the area under the excitation function on the interval $(0, t)$
- $(\lambda_0 T + \sum_{i=1}^{n(T)} \Phi(T - t_i))$ is the total area under the conditional intensity function $\lambda(t|H_t)$ on the interval $(0, T)$.

Equation 2.92 contains double summation. This means quadratic increase in the number of the operations. If the excitation function is set as exponential decay function $\varphi(t) = \alpha e^{-\beta t}$ and $\Phi(t) = \int_0^t \alpha e^{-\beta u} du = \frac{\alpha}{\beta} (1 - e^{-\beta t})$, this double summation can be eliminated. The log-likelihood function of the Hawkes processes with exponential decay excitation function is expressed as

$$l(\theta; t) = \sum_{i=1}^{n(T)} \log \left[\lambda_0 + \alpha \sum_{j=1}^{i-1} e^{-\beta(t_i - t_j)} \right] - (\lambda_0 T + \frac{\alpha}{\beta} \sum_{i=1}^{n(T)} [1 - e^{-\beta(T - t_i)}]). \quad (2.93)$$

To eliminate the double summation, $A(i)$ is defined as

$$A(i) = \sum_{j=1}^{i-1} e^{-\beta(t_i - t_j)} \quad (2.94)$$

where $i \in \{2, \dots, n(T)\}$ and $A(1)=0$. $A(i)$ can be written in terms of $A(i-1)$ as

$$A(i) = e^{-\beta(t_i - t_{i-1})}(1 + A(i-1)). \quad (2.95)$$

The log-likelihood function can be rewritten as

$$l(\theta; t) = \sum_{i=1}^{n(T)} \log [\lambda_0 + \alpha A(i)] - (\lambda_0 T + \frac{\alpha}{\beta} \sum_{i=1}^{n(T)} [1 - e^{-\beta(T - t_i)}]). \quad (2.96)$$

An example for the Hawkes process parameter estimation: Base intensity rate $\lambda_0 = 1$, excitation function: exponential decay $\varphi(t) = \alpha e^{-\beta t}$, simulation end time $=T$ and 5 observations, $N(T) = 5$. The log-likelihood function is expressed as

$$l(\alpha, \beta; t) = \sum_{i=1}^5 \log [1 + \alpha A(i)] - (T + \frac{\alpha}{\beta} \sum_{i=1}^5 [1 - e^{-\beta(T - t_i)}]). \quad (2.97)$$

The formulas of $A(i)$ s are stated as follows:

$$A(1) = 0 \quad (2.98)$$

$$A(2) = e^{-\beta(t_2 - t_1)}(1 + A(1)) \quad (2.99)$$

$$A(2) = e^{-\beta(t_2 - t_1)}$$

$$\begin{aligned}
A(3) &= e^{-\beta(t_3-t_2)}(1 + A(2)) \\
A(3) &= e^{-\beta(t_3-t_2)}(1 + e^{-\beta(t_2-t_1)}) \\
A(3) &= e^{-\beta(t_3-t_2)} + e^{-\beta(t_3-t_1)}
\end{aligned} \tag{2.100}$$

$$\begin{aligned}
A(4) &= e^{-\beta(t_4-t_3)}(1 + A(3)) \\
A(4) &= e^{-\beta(t_4-t_3)}(1 + e^{-\beta(t_3-t_2)} + e^{-\beta(t_3-t_1)}) \\
A(4) &= e^{-\beta(t_4-t_3)} + e^{-\beta(t_4-t_2)} + e^{-\beta(t_4-t_1)}
\end{aligned} \tag{2.101}$$

$$\begin{aligned}
A(5) &= e^{-\beta(t_5-t_4)}(1 + A(4)) \\
A(5) &= e^{-\beta(t_5-t_4)}(1 + e^{-\beta(t_4-t_3)} + e^{-\beta(t_4-t_2)} + e^{-\beta(t_4-t_1)}) \\
A(5) &= e^{-\beta(t_5-t_4)} + e^{-\beta(t_5-t_3)} + e^{-\beta(t_5-t_2)} + e^{-\beta(t_5-t_1)}
\end{aligned} \tag{2.102}$$

The derivatives of $A(i)$ s with respect to β are expressed as follows:

$$\frac{dA(1)}{d\beta} = 0, \tag{2.103}$$

$$\frac{dA(2)}{d\beta} = -(t_2 - t_1)e^{-\beta(t_2-t_1)}, \tag{2.104}$$

$$\frac{dA(3)}{d\beta} = -(t_3 - t_2)e^{-\beta(t_3-t_2)} - (t_3 - t_1)e^{-\beta(t_3-t_1)}, \tag{2.105}$$

$$\frac{dA(4)}{d\beta} = -(t_4 - t_3)e^{-\beta(t_4-t_3)} - (t_4 - t_2)e^{-\beta(t_4-t_2)} - (t_4 - t_1)e^{-\beta(t_4-t_1)}, \tag{2.106}$$

$$\begin{aligned} \frac{dA(5)}{d\beta} = & -(t_5 - t_4)e^{-\beta(t_5-t_4)} - (t_5 - t_3)e^{-\beta(t_5-t_3)} \\ & -(t_5 - t_2)e^{-\beta(t_5-t_2)} - (t_5 - t_1)e^{-\beta(t_5-t_1)}. \end{aligned} \quad (2.107)$$

The derivative of the log-likelihood function in Equation 2.97 with respect to α is expressed as

$$\frac{d l(\alpha, \beta : t)}{d \alpha} = \sum_{i=1}^5 \frac{A(i)}{1 + \alpha A(i)} - \left(\frac{1}{\beta} \sum_{i=1}^5 [1 - e^{-\beta(T-t_i)}] \right). \quad (2.108)$$

The derivative of the log-likelihood function in Equation 2.97 with respect to β is expressed as

$$\begin{aligned} \frac{d l(\alpha, \beta : t)}{d \beta} = & \sum_{i=1}^5 \frac{\alpha \frac{dA(i)}{d\beta}}{1 + \alpha A(i)} + 5 \frac{\alpha}{\beta^2} - \frac{\alpha}{\beta^2} \sum_{i=1}^5 (e^{-\beta(T-t_i)}) \\ & - \frac{\alpha}{\beta} \sum_{i=1}^5 (e^{-\beta(T-t_i)})(T - t_i). \end{aligned} \quad (2.109)$$

The data is simulated with the following input:

- base intensity rate $\lambda_0 = 1$
- excitation function, exponential decay $\varphi(t) = 1e^{-4t}$
- simulation end time, $T = 3$

The output (the event times t_i s) are as follows:

$t_1=1.467967, t_2= 1.555567, t_3=1.573065, t_4=2.232881, t_5=2.487068$

Note: The output is generated with the implemented code.

The MLE of the parameters of exponential decay excitation function are going to be estimated using Newton's method: The initial guesses for the parameters of exponential decay excitation function are $\alpha = 2$ and $\beta = 3$. The values of the derivative of the log-likelihood functions in Equation 2.108 and in Equation 2.109 are -0.368 and

0.233 respectively. After several iterations, the estimates reach to $\alpha = 1.3$ and $\beta = 4.1$ which are close to the parameters $\alpha = 1$ and $\beta = 4$. The values of the derivative of the log-likelihood functions are 0.116 and 0.010 respectively.

The Expectation Maximization Algorithm: The log-likelihood function in Equation 2.92 doesn't have analytic form for the maximization. It contains double summation which makes the numerical maximization computationally inefficient because of the quadratic increase of the function evaluations as the number of the events are increasing. Therefore, MLE is not attractive for the parameter estimation of Hawkes processes. The only exception are Hawkes processes with exponential decay excitation function with small number of observed event times because the double summation part can be reduced to single summation in the log-likelihood function formula as mentioned with an example.

The main difficulty to interpret the Hawkes process event data is that the branching structure is unknown. If the branching structure is known, the expectation-maximization algorithm can be applied to estimate the parameters of the Hawkes processes. Therefore, it is required to define a random variable Z which represents the branching structure of the Hawkes process. The random variable Z is defined as $Z = \{z_{ij}, i = 1 \dots n(T), j = 1 \dots i - 1\}$. [24].

- For the offspring events ($i = 1 \dots n(T), j = 1 \dots i - 1$)

The $z_{ij} = 1$ if the event t_i is an offspring event and its parent is the event t_j , otherwise $z_{ij} = 0$

- For the immigrant events ($i = 1 \dots n(T), j = 0$)

The $z_{i0} = 1$ if the event t_i is an immigrant event, otherwise $z_{i0} = 0$

By definition, the immigrant events are triggered by the outside sources and they don't have parent events which are generated by the process.

The most recent direct offspring event time $c_j(t)$ of the event t_j before time t can be defined as

$$c_j(t) = \max_{t_i < t} z_{ij} t_i \quad (2.110)$$

and $c_j(t) = 0$ if there aren't any offspring on the interval (t_j, t) .

Given the branching structure of the Hawkes process events (the $Z = \{z_{ij}\}$ values), the conditional probability density of the Hawkes process $f(t_i|Z)$ can be constructed. The likelihood $L(\theta, Z)$ and log likelihood $l(\theta, Z)$ functions conditional on the branching structure $Z = \{z_{ij}\}$ (using $c_j(t)$) can be derived and their closed forms can be found in Laub et al. in Subsection 6.1 "EM Algorithm for Hawkes Processes" [24]. The log likelihood function after the simplifications (the term $c_j(t)$ disappears in the likelihood function) in Laub et al. [24] is expressed as

$$l(\theta, Z) = -\lambda_0 T - \sum_{i=1}^{n(T)} \int_{t_i}^T \varphi(t - t_i) dt + \log \lambda_0 \sum_{i=1}^{n(T)} z_{i0} + \sum_{i=1}^{n(T)} \sum_{0 < j < i} z_{ij} \log \varphi(t_i - t_j). \quad (2.111)$$

The log-likelihood function $l(\theta, Z)$ contains the parameters Θ of Hawkes process conditional intensity function $\lambda(t|H_t)$ and the random variable Z which stores the branching structure of the Hawkes process. Instead of directly maximizing this log-likelihood, the alternative is to compute the expected values of the random variable $Z = \{z_{ij}\}$ which contains the branching structure of the Hawkes process.

The Expectation-Maximization algorithm can be summarized as follows:

- (i) Given an initial guess of the parameters $\Theta = \{\theta\}$ of the base intensity function λ_0 and the excitation function $\varphi(t)$ of the conditional intensity function of Hawkes process $\lambda(t|H_t)$.

- (ii) The Expectation Step: The expected values of the random variable $Z = \{z_{ij}\}$ are computed using the values of the parameters $\Theta = \{\theta\}$ which are given/computed. $Z = \{z_{ij}\}$ are either 1 or 0 and each of them are conditionally independent Bernoulli trial.

Therefore, $E[Z_{ij}|\theta, t_i] = P\{Z_{ij} = 1|\theta, t_i\}$ [24].

- For offspring events: $0 < j < i$

$$z_{ij} = P\{Z_{ij} = 1|\theta, t_i\} = \frac{\varphi(t_i - t_j)}{\lambda_0 + \sum_{t_k < t_i} \varphi(t_i - t_k)}$$

- For immigrant events: $j = 0$

$$z_{i0} = P\{Z_{i0} = 1|\theta, t_i\} = \frac{\lambda_0}{\lambda_0 + \sum_{t_k < t_i} \varphi(t_i - t_k)}$$

- (iii) The Maximization Step: Using the expected values of the random variable Z_{ij} which stores the branching structure, the expected log likelihood is maximized to compute the parameters $\Theta = \{\theta\}$ of the conditional intensity function of Hawkes process $\lambda(t|H_t)$.
- (iv) Go to (ii), the expectation step until the parameter values $\Theta = \{\theta\}$ converge to the MLE.

This is an iterative algorithm. At each iteration, $Z = \{z_{ij}\}$ and $\Theta = \{\theta\}$ are improved. The algorithm stops until a (local) maximum is reached. Furthermore, the algorithm uses numerical methods for the maximization of the parameters, because the closed forms for the parameters $\Theta = \{\theta\}$ don't exist for most of the excitation functions of Hawkes process [24].

The Method of Moments: If the maximum likelihood function is complex or if it is difficult to formulate, the method of moments is an alternative method to estimate the parameters. The moments are defined as powers r of the random variables X with mean μ . The first raw moment is the mean of the random variable of X , the second central moment is the variance of the random variable X . The four different types of moments are as follows [38]:

- raw moments (about origin), $m_r = E[X^r]$

The first raw moment is equal to the the mean of X .

- central moments (about mean of the random variable), $E[(X - \mu)^r]$

The first central moment of X is equal to 0(zero).

The second central moment of X is equal to the variance of X .

- absolute moments, $E[|X|^r]$
- generalized moments, $E[(X - a)^r]$

where a is a real number

The main idea beyond the moment matching method is to calculate the estimates of the parameters by setting sample moments to be as close as possible to their population counterparts. One of the important steps of the moment matching method is to select the function $g(t, \theta)$ which can represent the sample with the observations t_i , $i = 1 \dots n$ which are collected. The value of the parameters θ can be computed using the fact that the first central moment is equal to zero that is expressed as

$$E[g_i(\theta)] = 0 \quad (2.112)$$

where $g_i(\theta) = g(t_i, \theta)$.

The moment matching method is simple and easy to implement. If the number of the observed events is small, the estimation of the parameters is not efficient. Furthermore, the parameter estimates can be inconsistent if the selected function (whose parameters are estimated) doesn't fit the observed events.

3. RANDOM PROCESS GENERATION IN THE LITERATURE

In Chapter 3, first the preliminaries for the simulation algorithms are presented. Then the simulation algorithms for the Poisson process, the non-homogeneous Poisson process and the Hawkes process generation are listed.

3.1. Preliminaries for Simulation Algorithms

Before going into the details of the point process generation algorithms, there are several concepts to be mentioned which are related with the algorithms. These concepts are:

- The Inversion method
- The Acceptance Rejection Method
- The Exponential and Poisson Random Variate Generation

The Inversion Method: The inversion method is based on the following theorem: [39, 40]. Let U be a uniform $(0, 1)$ random variable. For any continuous (cumulative) distribution function F the random variable X is defined as

$$X = F^{-1}(U) \tag{3.1}$$

where F^{-1} is defined to be that value of x such that $F(x) = u$. For any discrete (cumulative) distribution function F the random variable X is defined as

$$X = F^{-1}(U) = \inf\{x : F(x) \geq U\}. \tag{3.2}$$

If the cumulative intensity function cannot be inverted (the closed form does not exist), numeric inversion methods can be used to compute inverse cumulative intensity function value for x . Polynomial interpolation is a numerical method for estimating the inverse cumulative intensity function. It requires the nodes of cumulative intensity function to estimate inverse cumulative intensity function. The selection of the nodes affects the performance of the estimation. There are more information about polynomial interpolation presented in appendix E.

The Acceptance Rejection Method: Acceptance rejection method is also called rejection sampling in the literature. The motivation beyond this method is that the majority of the (cumulative) distributions cannot be inverted. This method generates a sample from a distribution X with the density function f using a sample from distribution Y with density function g [41]. The method is as follows:

- (i) Generate a sample y from distribution Y and generate a standard uniform random number u on the interval $(0, 1)$
- (ii) If $u < \frac{f(y)}{Mg(y)}$
 - then accept y as a sample from f
 - else reject y
- (iii) Go to step (i)

M is such a number that assures $Mg(.) \geq f(.)$ for all the values in domain of f and g .

The connection between the acceptance rejection method for the random variate generation and the thinning method for the non-homogeneous Poisson process simulation [4], which is going to be explained in details in Subsection 3.3.3, is as follows: the thinning method is the process analogue of the acceptance rejection method for random variate generation. Acceptance rejection method generates independent and identically distributed random samples. The method starts from the beginning after each sample generation. Therefore, the generated samples are in random order. However thinning method generates new events on the simulation interval $(0, T)$ which depend on the

occurrence time of the last event in the process history. It starts at the time point "0" and ends after the time point "T". The generated event times are monotonically increasing on the simulation interval $(0, T)$.

The Exponential and Poisson Random Variate Generation: The exponential random variate X with rate λ can be generated by inversion method [39, 40]. The exponential random variate is generated by the formula

$$X = F^{-1}(U) = \frac{-\log(1 - U)}{\lambda}. \quad (3.3)$$

U is uniform random number on the interval $(0, 1)$. The Poisson random variate with mean λ can be generated by the inverse transform method [40]. The random variable X which is Poisson distributed with mean λ has the probability mass function that is expressed as

$$p(i) = P\{X = i\} = e^{-\lambda} \frac{\lambda^i}{i!} \quad i = 0, 1, 2, \dots \quad (3.4)$$

The method for a Poisson random variate generation with mean λ is as follows: [40] The key to using the inverse transform method to generate such a random variable is the following identity stated in Equation 3.5

$$p(i + 1) = \frac{\lambda}{i + 1} p(i) \quad (3.5)$$

The steps of the method are as follows [40]:

- (i) Generate a random number U
- (ii) $i = 0$, $p = e^{-\lambda}$, $F = p$ where $e^{-\lambda} = p(i = 0)$
- (iii) If $U < F$, set $X = i$ and stop
- (iv) $p = \frac{\lambda p}{i + 1}$, $F = F + p$, $i = i + 1$
- (v) Go to Step (iii)

where

- i is the value presently under consideration
- $p = p(i)$ is the probability that $X = i$
- $F = F(i)$ is the probability that $X \leq i$

3.2. The Poisson Process Simulation Methods

A summary of the Poisson process generation algorithms is presented at [42]. The event times $\{S_n, n = 1, 2, \dots\}$ of a homogeneous Poisson Process with rate λ on the time interval $(0, T]$ can be simulated with

- (i) Inter-event time with exponential random variates with rate one [21].
- (ii) Order statistics with a Poisson random variate with mean λT and uniform random numbers on the interval $(0, 1)$ [16, 21, 25].

The Inversion Method with Inter-event Times for the Poisson Process: This method uses exponential random variates with rate one to generate the inter-event times. It is based on the following statement:

The area between two consecutive events of a Poisson process is exponentially distributed with rate $\lambda = 1$, "E" which is shown in Equation 2.37. The area under the rate function of the Poisson process between two consecutive events is $\lambda t = E$. The inter-event time t between these two events is E/λ . The steps of the method are as follows [35]:

- (i) Calculate the total area under the rate λ on the time interval $(0, T] = \lambda T$
- (ii) $S_0 = 0$
- (iii) $i = 0$
- (iv) $i = i + 1$
- (v) Generate an exponential random variate with rate one, E_i .
- (vi) If $\sum_i E_i > \lambda T$ then deliver S_i 's and stop.

- (vii) $S_i = S_{i-1} + E_i/\lambda$
- (viii) Go to step (iv)

The Inversion Method with Order Statistics for the Poisson Process: This method uses a Poisson random variate with mean λT to determine the number of events on the interval $(0, T]$ as stated in Subsection 2.2 and uses uniform random numbers to calculate the event times on the interval $(0, T]$. [25,35]. If the number of events $N(T) = n$ occurred on the time interval $(0, T]$ is given, then the event times are uniformly distributed on the time interval $(0, T]$. The steps of the method are as follows [42]:

- (i) Calculate the total area under the rate λ on the time interval $(0, T] = \lambda T$
- (ii) Generate a Poisson random variate, " P " with mean λT .
- (iii) If $P = 0$, then Stop.
- (iv) Generate " P " standard uniform random numbers, U_i on the interval $(0, 1)$ where $i = 1, \dots, P$
- (v) The event times are computed as $S_i = T * U_i$ where $i = 1, \dots, P$
- (vi) Deliver S_i 's and stop.

The event times S_i 's are unsorted and required to be sorted if simulation application forces ordered set of event times.

3.3. The Non-Homogeneous Poisson Process Simulation Methods

A summary of the non-homogeneous process generation methods is presented at [43]. The event times $\{S_n, n = 1, 2, \dots\}$ of a non-homogeneous Poisson Process with intensity function $\lambda(t)$ can be simulated on the time interval $(0, T]$ with

- (i) Inversion methods
- (ii) Composition method
- (iii) Thinning method [4]

3.3.1. The Inversion Method with Inter-event Times for NHPP

The first inversion method uses exponential random variates with rate one to simulate the inter-event times of the non-homogeneous Poisson process [21] which is based on "the time re-scaling theorem" in Section 2.1. The exponential random variates represent the inter-event times of the homogeneous Poisson process with rate one. The inverse cumulative intensity function of the non-homogeneous Poisson process converts these inter-events times to the inter-event times of the non-homogeneous Poisson process with intensity function $\lambda(t)$. The details are as follows: For any non-homogeneous Poisson process with

- the intensity function $\lambda(t)$ and
- the cumulative intensity function $\Lambda(t)$,

the cumulative distribution function, " $F_{s_{n+1}}(T = t; S_n)$ " for the inter-event time, t to the next event, S_{n+1} (known an event just occurred at time S_n) can be written as

$$F_{s_{n+1}}(T = t; S_n) = 1 - \exp\left(- \int_{s_n}^{s_{n+1}} \lambda(u) du\right), \quad (3.6)$$

$$F_{s_{n+1}}(T = t; S_n) = 1 - \exp\left(- \int_{s_n}^{S_n+t} \lambda(u) du\right). \quad (3.7)$$

Because " F " is a distribution function, it is uniformly distributed on the interval (0,1) and is expressed as

$$F_{s_{n+1}}(T = t; S_n) \sim U(0, 1), \quad (3.8)$$

$$U = 1 - \exp\left(- \int_{s_n}^{S_n+t} \lambda(u) du\right). \quad (3.9)$$

After rearranging Equation 3.9, the area under the intensity function between successive events is exponentially distributed with rate one and is expressed as

$$-\log (1 - U) = \int_{s_n}^{S_n+t} \lambda(u) du \quad (3.10)$$

where $-\log (1 - U) = E$ (E is exponential random variate with rate 1 and U is uniform random number on $(0, 1]$). Equation 3.10 can be rewritten as

$$E = \Lambda(S_n + t) - \Lambda(S_n) \quad (3.11)$$

where $\Lambda(t)$ is the cumulative intensity function. After arranging Equation 3.11,

- the inter event time t can be written as

$$t = \Lambda^{-1}(E + \Lambda(S_n)) - S_n \quad (3.12)$$

where Λ^{-1} is the inverse cumulative intensity function.

- the next event time, S_{n+1} can be expressed as

$$S_{n+1} = S_n + t = \Lambda^{-1}(E + \Lambda(S_n)). \quad (3.13)$$

To summarize the results,

- The first event time S_1 is expressed as

$$S_1 = \Lambda^{-1}(E_1 + \Lambda(S_0)) = \Lambda^{-1}(E_1) \quad (3.14)$$

where $S_0 = 0$ and $\Lambda(S_0) = 0$.

- The second event time S_2 is expressed as

$$S_2 = \Lambda^{-1}(E_2 + \Lambda(S_1)) = \Lambda^{-1}(E_2 + E_1) = \Lambda^{-1}\left(\sum_{i=1}^2 E_i\right) \quad (3.15)$$

where $S_1 = \Lambda^{-1}(E_1)$.

- The n^{th} event time S_n is expressed as

$$S_n = \Lambda^{-1}\left(\sum_{i=1}^n E_i\right) \quad (3.16)$$

where E_i is an exponential random variate with rate one.

The steps of the method are as follows [21, 35]:

- (i) Calculate the total area under the intensity function for the time interval $(0, T] = \int_0^T \lambda(u) du$
- (ii) $S_0 = 0$
- (iii) $i = 0$
- (iv) $i = i + 1$
- (v) Generate an exponential random variate with rate one, E_i .
- (vi) If $\sum_i E_i > \int_0^T \lambda(u) du$ then deliver S_i 's and stop.
- (vii) $S_i = \Lambda^{-1}(\sum_i E_i)$
- (viii) Go to step (iv)

3.3.2. The Inversion Method with Order Statistics for NHPP

The second inversion method uses a Poisson random variate P and " P " uniform random numbers to simulate the non-homogeneous Poisson process. The algorithm is based on "Order Statistic" [25] which is presented in Section 2.2. As presented in Section 2.2, the number of events " P " which are generated by a non-homogeneous Poisson process with intensity function $\lambda(t)$ on the interval $(0, T)$ is Poisson distributed

with mean $= \Lambda(T)$ [14].

Given the number of events " P " of a non-homogeneous Poisson process, the events of the non-homogeneous Poisson process can be simulated as follows:

- Simulate the event times t_i , $i = 1, \dots, P$ of the homogeneous Poisson process with rate one on the interval $(0, \Lambda(T)]$ using order statistics. They are uniformly distributed on the interval $(0, \Lambda(T)]$.
- Convert the event times t_i , $i = 1, \dots, P$ of the homogeneous Poisson process with rate one on the interval $(0, \Lambda(T)]$ to the event times of the non-homogeneous Poisson process with intensity function $\lambda(t)$ on the interval $(0, T]$ using the inverse cumulative intensity function $\Lambda^{-1}(\cdot)$ to evaluate $\Lambda^{-1}(t_i)$ $i = 1, \dots, P$.

Devroye [35] and Pasupathy [42] mentioned this method. The steps of the method are in detail as follows:

- (i) Calculate the total area under the intensity function $\lambda(t)$ for the time interval $(0, T]$, $\int_0^T \lambda(u) du = \Lambda(T)$
- (ii) Generate a Poisson random variate, " P " with mean $\Lambda(T)$.
- (iii) If $P = 0$, then Stop.
- (iv) Generate " P " standard uniform random numbers, U_i on the interval $(0, 1)$ where $i = 1, \dots, P$
- (v) The events times are $S_i = \Lambda^{-1}[\Lambda(T) U_i]$ where $i = 1, \dots, P$

Note: The simulation period $(0, T)$ of the non-homogeneous Poisson process is re-scaled by the variable change, $t \rightarrow \Lambda(t)$, to the simulation period $(0, \Lambda(T))$ of the homogeneous Poisson process.

$[\Lambda(T) U_i]$ with $i = 1, \dots, P$ are the event times of the homogeneous Poisson process with rate one on the interval $(0, \Lambda(T))$. These event times are transformed to the event times of non-homogeneous Poisson process with intensity function $\lambda(t)$ by the execution of the operation $\Lambda^{-1}(\cdot)$.

- (vi) Deliver S_i 's and stop.

The event times S_i 's are unsorted and required to be sorted if the simulation application forces ordered set of event times.

The simulation method can generate empty output sets. If the simulation application forces non-empty output sets, this can be achieved as follows: The Poisson random variable is a discrete random variable. It has the parameter $\lambda > 0$ which is expected number of events in a defined interval. It takes one of the values $0, 1, 2, \dots$. The probability mass function of the Poisson random variable with the parameter λ is stated in Equation 2.3. The cumulative distribution function of the Poisson random variable X with parameter λ is expressed as

$$P(X \leq i) = p(0) + p(1) + \dots + p(i) = \sum_{i=0}^i p(i). \quad (3.17)$$

The Poisson random variable X returns 0 with the probability $P\{X \leq 0\} = p(0)$. This is the percentage of simulated cascades which are empty (they don't contain any events) because the immigrant sub-process doesn't generate any immigrants.

To simulate a NHPP with non-empty output set, it is required to generate a Poisson random variate greater than zero for the process. This aim can be achieved as follows: X is a Poisson random variable with parameter $\Lambda(T)$ and $P\{X \leq i\}$ is its cumulative distribution function. $P\{X \leq i\}$ is uniformly distributed on the interval $(0, 1]$ because it is a cumulative distribution function. The interval $(P\{X \leq 0\}, 1]$ contains the probabilities of the Poisson random variates greater than zero with mean $\Lambda(t)$. If a uniform random number on the interval $(P\{X \leq 0\}, 1]$ is generated, (which is the probability of a Poisson random variate that is greater than zero) then a Poisson random variate with mean $\Lambda(T)$ greater than zero can be computed using quantile function (inverse cumulative distribution function) of the Poisson random variable. The following operations are required for this case.

- Calculate the probability $P\{X \leq 0\}$ of the Poisson random variable with mean $\Lambda(T)$

- Generate a uniform random variate "u" on the interval $(P\{X \leq 0\}, 1]$
- Generate a Poisson random variate, "p" by evaluating the inverse cumulative distribution function (quantile function) of the Poisson random variable with mean $\Lambda(T)$ at "u"

The steps (ii) and (iii) of the method are required to be replaced the above operations.

3.3.3. The Thinning Method

The third method for the non-homogeneous Poisson Process simulation is the thinning method by Lewis and Shedler [4]. Thinning method is based on the following theorem by Lewis and Shedler [4] p. 5-6, Theorem 1. Consider a one-dimensional non-homogeneous Poisson Process, $N^*(t) : t \geq 0$ with hat (rate) function, $\lambda^*(t)$, so that the number of points, $N^*(t_0)$ in a time interval $(0, T]$ has a Poisson distribution with parameter $\mu^* = \Lambda^*(t_0) - \Lambda^*(0)$. Let $S_1^*, S_2^*, \dots, S_{N^*(T)}^*$ be the points of the process in the time interval $(0, T]$. Suppose that for $0 \leq t \leq T$, $\lambda(t) \leq \lambda^*(t)$. For $i = 1, 2, \dots, n$, delete the point S_i^* with probability $1 - \frac{\lambda(S_i^*)}{\lambda^*(S_i^*)}$, then the remaining points form a non-homogeneous Poisson Process, $N(t) : t \geq 0$ with rate function $\lambda(t)$ in the time interval $(0, T]$.

According to the theorem, an event S_i^* of the non-homogeneous Poisson Process with intensity function, $\lambda^*(t)$ is accepted as an event of the non-homogeneous Poisson Process with intensity function $\lambda(t)$ with the probability $\frac{\lambda(S_i^*)}{\lambda^*(S_i^*)}$ and rejected with probability $1 - \frac{\lambda(S_i^*)}{\lambda^*(S_i^*)}$.

It is also possible to generate the events from a homogeneous Poisson process with constant rate λ^* instead of to generate events from a non-homogeneous Poisson process with time varying intensity function $\lambda^*(t)$ with satisfying $\lambda^* \geq \lambda(t) \forall t \in (0, T]$. This version of the thinning method is used during the studies. The steps of the thinning method with the constant rate function λ^* are as follows [4]:

- (i) Generate the events of the homogeneous Poisson process with intensity function λ^* on the time interval $(0, T]$.
- (ii) If the number of generated events, $n^* = 0$ then, stop, there aren't any events in the process.
- (iii) Denote the (ordered) event times by $S_1^*, S_2^*, \dots, S_{n^*}^*$
- (iv) $k \leftarrow 0$ and $i \leftarrow 1$
- (v) Generate U_i (uniform random variable on the interval $(0,1)$)
- (vi) If $U_i \leq \frac{\lambda(S_i^*)}{\lambda^*}$ then $k \leftarrow k + 1$ and $S_k = S_i^*$
- (vii) $i \leftarrow i + 1$.
- (viii) If $i \leq n^*$ go to step (v)
- (ix) Deliver S_k 's and stop.

3.3.4. The Composition Method

The next method for non-homogeneous Poisson Process simulation is the composition method. It can be used when there is a natural decomposition dictated by the analytic form of the intensity function $\lambda(t) = \lambda_1(t) + \lambda_2(t) + \dots + \lambda_n(t)$. Each individual process with the intensity function $\lambda_i(t)$ creates its events that build a cluster. The union of these clusters constructs the events of the total process. The total process is the superposition of the individual processes. Devroye [35] stated that because the basic operation in merging (i.e., superposition of) the processes is to take the minimal value from the "n" processes, it could be advantageous for large "n" to store the times in a heap containing "n" elements. It is sufficient to consider the minimum of "n" random variables $X_1, \dots, X_n$ with the individual intensity rates λ_n . The steps of the composition method are as follows [35]:

- (i) Generate $S_{1,1}, \dots, S_{n,1}$ for the "n" Poisson processes with intensity rates λ_i , and store these values together with the indices of the corresponding processes in a table.
- (ii) $T \leftarrow 0$ (T is the running time)
- (iii) $k \leftarrow 0$ (k is a counter)

- (iv) Repeat
 - (a) Find the minimal element (say, S_{i_j}) in the table and delete it.
 - (b) $k \leftarrow k + 1$
 - (c) $S_k \leftarrow S_{i_j}$
 - (d) Generate the value $S_{i_{j+1}}$ and insert it into the table.
- (v) Until False

Usually, the composition method is slow because the method must deal with all the individual intensity rates [35]. In other words, instead of generating a single process, "n" processes must be simulated.

3.4. Hawkes Process Simulation Algorithms

The event times $\{S_n, n = 1, 2, \dots\}$ of a self-exciting (Hawkes) process with conditional intensity function $\lambda(t|H_t) = \lambda_0 + \sum_{i:t>t_i} \varphi(t - t_i)$ can be simulated on the time interval $(0, T]$ with

- (i) Inversion algorithm
- (ii) Ogata's modified thinning algorithm [3]
- (iii) Cluster-based (Composition) algorithm based on the paper of [7]

The (immigrant and offspring) sub-processes of cluster-based (Composition) algorithm are independent non-homogeneous Poisson processes and can be simulated with one of the non-homogeneous Poisson process generation methods presented in section 3.3.

3.4.1. The Inversion Algorithm

The inversion algorithm [21] for the non-homogeneous Poisson process (inversion method to generate the inter-event time with the exponential random variates with rate one) can theoretically be used to simulate Hawkes processes. However, in practice it is an inefficient algorithm because the structure of the conditional intensity func-

tion, which is a sum with an increasing number of summands, changes after each new event joining to the process history, because the conditional intensity function itself is a stochastic process. This is the main difficulty of this algorithm, namely iterative numerical computation of the inverse of the cumulative (integrated) conditional intensity function after each new event joining to the process history. In other words, it requires repeated numerical solution of the equation defining the inverse cumulative conditional intensity function.

3.4.2. Ogata's Modified Thinning Algorithm

Ogata's modified thinning algorithm for self-exciting (Hawkes) process simulation is the modified version of Lewis and Shedler's thinning method for the non-homogeneous Poisson process simulation. The intensity-based thinning method is initially introduced by Lewis and Shedler [4] in the general context of the non-homogeneous Poisson processes. Ogata [3] modified this method to simulate Hawkes processes. Like the thinning method, Ogata's modified thinning algorithm generates accepted and rejected event times during the process simulation period. The main idea of the algorithm can be summarized as follows: Given a fixed process history " H_t " with the last event occurred at time " t ", Hawkes process can be seen as a non-homogeneous Poisson process with an intensity function whose structure is defined at the last event time in process history and is valid until the process generates a new event. Until the occurrence of the new (accepted) event, the algorithm may generate rejected events. The new (accepted) event generation causes a change on the structure of conditional intensity function of Hawkes process and a new non-homogeneous Poisson process with an updated conditional intensity function starts and it finishes at the time when the process generates a new accepted event. The process continues in the same way until the end of the simulation period or a certain number of events which are generated by the algorithm. In other words, the simulation period is randomly decomposed into sub-intervals which are simulated with a non-homogeneous Poisson processes with different conditional intensity functions to generate a single accepted event for each sub-interval.

The Hawkes process has a stochastic conditional intensity function $\lambda(t|H_t)$. This means the intensity function value at a certain time " t " depends on the events in the process history H_t that occurred before the time " t ". Such a process with conditional intensity function $\lambda(t|H_t)$ can be simulated using the thinning method by Lewis and Shedler [4] if there exist a finite bound " M " such that [16]:

- $\lambda(t|H_t) \leq M$ for all possible past histories and
- the process is simulated over a finite interval $(0, T)$ given some initial history H_0

The thinning algorithm works fine in its original context with deterministic intensity function because it is possible to find a finite bound " M " over the simulation interval $(0, T)$ for all possible past histories. However, its main difficulty is to find a finite bound " M " over the simulation interval $(0, T)$ for all possible past histories if the process to be simulated has random (stochastic) conditional intensity function. It is difficult or even not possible.

Ogata [3] proposed an algorithm to overcome this difficulty which requires only local boundedness condition on the conditional intensity function. Later it is called "Ogata's Modified Thinning Algorithm" in the literature. The steps of Ogata's modified thinning algorithm (for the case the conditional intensity function is monotonically decreasing if no events occurs) are as follows [16, 24]:

- (i) Set $t = 0$, $i = 0$.
- (ii) Stop if the termination condition is met; otherwise, $M = \lambda(t^+|H_t)$
- (iii) Generate an exponential random variate, E with mean one and a uniform random variable on $(0, 1)$.
- (iv) Compute the thinning step time, replace t by $t + E/M$
- (v) If $\frac{\lambda(t|H_t)}{M} > U$, go to step (ii).
- (vi) Otherwise,
 - (a) advance $i \leftarrow i + 1$,
 - (b) set $t_i = t$,

- (c) update H to $H \cup t_i$, and
- (d) return to step (ii)
- (vii) The output is the list $\{i; t_1, \dots, t_i\}$

The termination condition can be simulation end time T or the number of Hawkes process events which are simulated by the algorithm. The complexity of Ogata's modified thinning algorithm is $O(N^2)$ which makes the algorithm inefficient [44]. An important point about Ogata's modified thinning algorithm is that it works with monotone intensity functions (if no more events occur) [3]. It is especially simple for exponential decay function $\alpha e^{-\beta t}$ (which is also a monotone function) whose rate of change is constant (independent of time). It is sufficient to update the total excitation produced by the events in the process history at previous thinning step by multiplying with $e^{-\beta t}$ (which models the memory over time) instead of calculating the excitation which are caused by each event in the process history one by one and summing up all the excitation after every new accepted event. This feature of exponential decay function saves a lot of operations. This is one of the reasons why the exponential decay excitation function, $\alpha e^{-\beta t}$ is frequently used as the excitation function for the Hawkes process simulation with Ogata's modified thinning algorithm in the literature. The detailed version of Ogata's modified algorithm is Appendix A.

3.4.3. The Cluster-based (Composition) Algorithm

The cluster-based (combination) algorithm is based on the cluster process representation of a self-exciting process [7]. The structure of the conditional intensity function of Hawkes Process which is linear provides the opportunity to decompose the self-exciting (Hawkes) process into sub-processes called immigrant process and offspring processes. The base intensity function of Hawkes Process λ_0 is the intensity function of the immigrant process. The excitation function of Hawkes Process $\varphi(t)$ is the intensity function of the offspring processes so that the events of the Hawkes process can be classified as immigrant events and offspring events. The sub-processes are independent non-homogeneous Poisson processes [7] which can be generated by one of the

non-homogeneous Poisson process generation methods presented in Section 3.3. Each event in process history H_t (immigrant or offspring) triggers an offspring sub-process with excitation function $\varphi(t)$ which can generate its direct offspring. Therefore, the cluster-based (composition) algorithm is a recursive algorithm. There are an immigrant sub-process and random number of sub-processes whose total number is equal to the total number of events in the process history H_t . The starting time of each offspring sub-process is one of the event times t_i in process history H_t . The composition of the events of the sub-processes constitutes the event set of Hawkes process. Therefore, the cluster-based approach is a composition algorithm ([35], p.255-256, the composition method).

However, Devroye [35] indicated that composition algorithm is slow as stated in the section 3.3.4 but the decomposition of hawkes process reduces a double stochastic Hawkes process into many stochastic non-homogeneous Poisson processes. This property affects the number of operations to simulate Hawkes process and is going to be investigated in details in Section 4.3. Researchers have described the composition algorithms for Hawkes process generation. They used mainly constant function for base intensity and exponential decay for excitation functions as in the original Hawkes process definition. Laub et al. [24,45] present a composition algorithm for Hawkes process simulation with constant base intensity and exponentially distributed excitation function. Moller and Rasmussen [5] used cluster-based algorithm to research the edge effects of the process. Bacry et al. [46] provide a similar extension of the immigration birth simulation method. Another method to generate Hawkes processes is provided by Dassios and Zhao [47] with exponential decay excitation function.

The steps of cluster-based (composition) algorithm are follows:

The inputs: the immigrant process intensity function, the offspring process intensity function, the simulation interval $(0, T)$

- (i) Simulate the immigrant sub-process to generate "immigrant events" with an non-homogeneous Poisson process generation method in the literature

- (ii) Add "these immigrant events" in the fertile set " F "
- (iii) Stop if the fertile set is an empty set, " $F = \emptyset$ ", there is no event in the process
- (iv) Add "the immigrant events" in the process history " H "
- (v) Repeat
 - (a) Take an event " f " from the fertile set " F " and delete this event " f " from the fertile set " F "
 - (b) Simulate the offspring sub-process for the event " f " to generate its "offspring events" with a non-homogeneous Poisson process generation method in the literature
 - (c) Remove its offspring events which are beyond the simulation end time " T " (if necessary)
 - (d) Add the remaining "offspring events" in the fertile set " F "
 - (e) Add the remaining "offspring event" in the process history " H "
- (vi) Until the fertile set $F = \emptyset$
- (vii) Sort the event times in the process history " H " if simulation application forces ordered event times
- (viii) Deliver the event times in the process history " H " as output

3.4.4. The Software Packages for Hawkes Process Generation

In this section, the software about Hawkes process generation are summarized with indicating their simulation algorithms. The packages which are related to the Hawkes process generation in R and their simulation algorithms are listed in Table 3.1:

Table 3.1. The R-packages with simulation algorithms

Package Name	Function name	Algorithm	Base intensity	Excitation function
emhawkes	hsim	Ogata's	scalar	exponential kernel
hawkes	simulateHawkes	Ogata's	scalar	exponential kernel
	hawkes	cluster	function	function
IHSEP	simHawkes0	cluster	function	function
	simHawkes1	cluster	function	function
	simHawkes1a	cluster	function	function
RHawkes	simRHawkes	cluster	function	function
	simRHawkes1	cluster	function	function
MRHawkes	simMRHawkes	cluster	function	function
	simNSMHP	cluster	function	function
ptprocess	simulate	Ogata's	function	
stelfi	sim_hawkes	Ogata's	scalar	exponential kernel
stpphawkes	simulate_temporal	Ogata's	scalar	exponential kernel
surveillance	twinSIR_simulation	Ogata's		exponential kernel

More information about the software about the software in R are presented in Appendix D. The packages related to Hawkes process generation in Python and their simulation algorithms are listed in Table 3.2. More information about the software in Python are presented in Appendix D.

Table 3.2. The Python-packages with simulation algorithms

Package Name	Simulation algorithm
HawkesPyLib	Ogata's modified thinning algorithm
hawkes	Ogata's modified thinning algorithm
py-hawkes	Ogata's modified thinning algorithm
tick.hawkes	Ogata's modified thinning algorithm
hawkeslib	Ogata's modified thinning algorithm
Hawkes	Ogata's modified thinning algorithm

4. THE PERFORMANCE OF THE GENERATION ALGORITHMS

In Chapter 4, the performances of the point processes (the Poisson process, the non-homogeneous Poisson process and Hawkes process) generation algorithms are investigated. Hawkes process generation algorithms (intensity-based and cluster-based) are well known but there is no discussion about their performance in the literature. Therefore, the performance of Hawkes process generation algorithms are compared in this chapter. The performance of the homogeneous and the non-homogeneous Poisson processes generation methods are also investigated because the sub-processes of Hawkes process are non-homogeneous Poisson processes. This discussion is also important to design the fastest possible algorithms because designing a fast process implementation requires fast methods of process generation and random variate generation.

The target of this chapter is to decide the fastest generation methods and algorithms for the point processes: the Poisson process, the non-homogeneous Poisson process and Hawkes process. At this point, counting the number of operations of the algorithms plays an important role. Counting has important applications in computer science and mathematics. Some of them are determining complexity of algorithms and computing discrete probability of some events. Therefore, to decide the fastest algorithm, the number of operations by the point process simulation algorithms is going to be counted and compared how the number of operations (required to simulate the point processes) changes as the number of events " n " generated by the algorithms increases. The following main operations are counted:

- the random variates
- the intensity function evaluations
- the inverse cumulative intensity function evaluations

The point process generation algorithms need the random variates to generate the individual event times and to determine the total number of events which are generated by the process. The intensity function evaluations are used for the "comparison operations" to select some of the event times of a process as event times of another (desired) process at the methods and algorithms which use thinning operation. The inverse cumulative intensity function evaluations convert the event times of a process to the event times of another (desired) process at the methods and algorithms which employ inversion method. As additional operations, the requirements of

- comparison operations and
- sorting operations

for the algorithms are specified and counted.

The order of the processes to be studied are as follows:

- Poisson process
- Non-homogeneous Poisson process
- Hawkes process

Hawkes process turns to the non-homogeneous Poisson process if its excitation $\varphi(t) = 0$ (Hawkes process doesn't have independent increments). The non-homogeneous Poisson process turns to the Poisson process if its intensity function is a constant function (scalar), $\lambda(t) = \lambda$ (The non-homogeneous Poisson process doesn't have stationary increments).

4.1. The Performance of the Poisson Process Generation Methods

The Poisson process generation methods in the literature are listed in Section 3.2. These methods are based on:

- (i) Inter-event times
- (ii) Order statistics

The first method (The Poisson process generation method with inter-event times) requires

- an exponential random variate with rate one

per event generation of Poisson process with rate λ on the time interval $(0, T)$. The exponential random variate is needed to compute the area under the rate function, λ between the successive events. Total number of random variates required to generate " n " (homogeneous) Poisson process event times are

- " $n + 1$ " exponential random variates with rate one

The " n " exponential random variates are used to compute the event times of " n " events. The additional "+1" exponential random variate is required to assure that the cumulative sum of exponential random variates (which are used to generate the event times except the last, "+1" exponential variate) exceeds the area under the rate function of the homogeneous Poisson process which is equal to λT so that the process terminates.

This method requires a "stopping condition" for the process. It is achieved by comparing the cumulative sum of exponential random variates with the area under the rate function on the simulation period $(0, T)$ after each generation of an exponential random variate. The process generates events until the cumulative sum of exponential exceeds the area under the intensity function where the stopping condition is satisfied. This comparison after each generation of an exponential random variate is named as "Comparison type 1" in this study. Total number of comparison operations are

- " $n + 1$ " "Comparison type - 1" operations

The event times which are generated by this method are in increasing order.

The second method (The Poisson process generation with order statistics) to simulate the Poisson process with rate λ requires

- a standard uniform random number

for each event generation on the simulation interval $(0, T)$. The standard uniform random numbers are used to compute the event times of the (homogeneous) Poisson process with rate λ . The total number of random variates required to generate " n " (homogeneous) Poisson process event times are

- " n " standard uniform random numbers
- a Poisson random variate with mean $= \lambda T$ which is the area under the rate function,

A Poisson random variate with mean $= \lambda T$ is required to determine the total number of events " n " which are going to be generated by the (homogeneous) Poisson process with rate λ .

The events times are not ordered. If the simulation application forces ordered event times, then the uniform random numbers (or the generated event times) are required to be sorted to generate the event times in increasing order ("sorting operation"). The number of operations for sorting depends on the sorting method. There are sorting methods whose number of operations are increasing linearly as the number of events increases. A linear sorting method from the literature is going to be presented in Section 5.2.

The total number of main and additional operations which are required to generate " n " events by the Poisson process generation methods are listed in Table 4.1.

Table 4.1. The total number of operations which are required by the Poisson process generation methods

		METHODS	
		Inter-event times	Order statistics
MAIN OPERATIONS			
Random variates	Exp.	n+1	
	Uniform		n
	Poisson		1
Additional operations			
Comparison type-1		n+1	
Sorting			required

To summarize Table 4.1, both methods require same number of random variates. There is no clear difference between these two methods if simulation application forces ordered set of events. Otherwise, the method with order statistics requires fewer number of operation than the other method.

4.2. The Performance of the NHPP Generation Methods

The number of operations required and their behavior (as the number of generated events increases) to simulate the non-homogeneous Poisson process with different methods are investigated for the performance assessment. The number of operations of both of the inversion methods for the non-homogeneous Poisson process generation in the next Section 4.2.1 follows the number of operations of both of the Poisson process generation methods (in the previous Section 4.1). The only difference is an additional inverse cumulative intensity function $\Lambda^{-1}(\cdot)$ evaluation per event generated by the non-homogeneous Poisson process with intensity function $\lambda(t)$. The reason for the requirement of inverse cumulative intensity function evaluation is that the area under the rate function of the Poisson process between successive events (which exponentially distributed with mean one) is a rectangle. Dividing this area by the rate λ

returns the inter-event time. Although the area under the intensity function $\lambda(t)$ of non-homogeneous Poisson process between successive events is exponentially distributed with mean one, its shape is not a rectangle. Therefore, inverse cumulative intensity function $\Lambda^{-1}(\cdot)$ is required to compute the inter event times. The details are presented in Section 4.2.1.

4.2.1. The Inversion Methods

There are two methods which use inversion method:

- The inversion method using inter-event times
- The inversion method using order statistics and time re-scaling theorem

The inversion method using inter-event times: The following operations are required for each event time generation of a non-homogeneous Poisson process with intensity function $\lambda(t)$ on the time interval $(0, T)$ with this method:

- an exponential random variate with rate one,
- an evaluation of inverse cumulative intensity function $\Lambda^{-1}(t)$

The exponential random variate determines the area under the intensity function $\lambda(t)$ between the successive events. The inverse cumulative intensity function $\Lambda^{-1}(t)$ uses this exponential random variate to compute the inter-event time between the successive events. Total number of random variates and inverse cumulative intensity function evaluations to generate " n " non-homogeneous Poisson process event times are

- " $n + 1$ " exponential random variates with rate one
- " n " evaluations of inverse cumulative intensity function $\Lambda^{-1}(t)$

The " n " exponential random variates together with " n " evaluations of inverse cumulative intensity function $\Lambda^{-1}(t)$ are used to compute " n " event times of the process.

The additional "+1" exponential random variate assures that the cumulative sum of exponential random variates (which are used to generate the event times except the last exponential number) exceeds the area under the intensity function of the non-homogeneous Poisson process which is equal to $\Lambda(T)$ so that the process stops. This method requires same comparison operation "Comparison type-1" like the Poisson process generation with inter-event times in Section 4.1 as the stopping condition of the process. Total number of comparison operations are

- " $n + 1$ " "Comparison type - 1" operations

The inversion method using order statistics and time re-scaling theorem: The following operations are required for each event time generation of a non-homogeneous Poisson process with intensity function $\lambda(t)$ on the time interval $(0, T)$ with this method:

- a standard uniform number,
- an evaluation of inverse cumulative intensity function $\Lambda^{-1}(T)$

The standard uniform random numbers are needed to compute the event times of the Poisson process with rate one. The inverse cumulative intensity function $\Lambda^{-1}(t)$ is used to convert the event times of the (homogeneous) Poisson process with rate one to the non-homogeneous Poisson process with intensity function $\lambda(t)$. Total number of random variates and inverse cumulative intensity function evaluations to generate " n " non-homogeneous Poisson process event times are

- a Poisson random variate with mean = area under the intensity function, $\Lambda(T)$
- " n " standard uniform random numbers
- " n " evaluations of inverse cumulative intensity function $\Lambda^{-1}(t)$

A Poisson random variate with mean = λT is required to determine total number of events " n " which are going to be generated by the non-homogeneous Poisson process

with intensity function $\lambda(t)$. The " n " standard uniform random numbers together with " n " evaluations of inverse cumulative intensity function $\Lambda^{-1}(t)$ are used to compute " n " event times of the process.

Like the (homogeneous) Poisson process generation method in Section 4.1, the event times are not in increasing order. If the simulation application forces ordered event times, then the uniform random numbers (or the generated event times) are required to be sorted to generate the event times in increasing order ("sorting operation").

4.2.2. The Thinning Method

The following operations as lower bound are required for each event time generation of the non-homogeneous Poisson process with intensity function $\lambda(t)$ on the time interval $(0, T)$ with the thinning method:

- (at least) an exponential random variate with rate one,
- (at least) a standard uniform number,
- (at least) an evaluation of intensity function $\lambda(t)$

The exponential random variate with rate one is needed to generate the event time " t " of the Poisson process with the hat intensity function λ^* . The standard uniform random number is required to decide whether the event time " t " of the Poisson process with hat intensity function λ^* is accepted as an event of the (simulated) non-homogeneous Poisson process with intensity function $\lambda(t)$ or is rejected. The evaluation of intensity function $\lambda(t)$ value at the thinning step time " t " is needed for the comparison with the hat intensity function λ^* value at time " t ".

Total number of random variates and intensity function evaluations required to generate " n " event times of the non-homogeneous Poisson process with intensity function $\lambda(t)$ on the time interval $(0, T)$ with this method is

- (at least) " $n + 1$ " exponential random variates with rate one
- (at least) " n " standard uniform random numbers
- (at least) " n " intensity function $\lambda(t)$ evaluations

The term "at least" is added in front of the number of operations, because they are a lower bound. The lower bounds of the operations are

- " $n + 1$ " exponential random variates with rate one
- " n " standard uniform random numbers
- " $n + 1$ " intensity function $\lambda(t)$ evaluations

if thinning method doesn't generate any rejected events at the thinning step of the method. This can be achieved if the hat intensity is equal to the intensity function of the process to be simulated.

This situation can be explained by investigating the idea on which thinning method is based. Thinning method needs a process with an intensity function $\lambda^*(t)$ to simulate a process with the intensity function $\lambda(t)$ on the simulation interval $(0, T)$. $\lambda^*(t)$ is called hat intensity because it must satisfy $\lambda^*(t) \geq \lambda(t) \forall t \in (0, T)$. The important point is that simulation of the process with an intensity function $\lambda^*(t)$ is easier than the process with the intensity function $\lambda(t)$ on the simulation interval $(0, T)$. The easiest and most frequently used hat intensity function is a constant function $\lambda^*(t) = \lambda^*$. The thinning method decomposes the process with hat intensity function in two processes with intensity function $\lambda(t)$ and with intensity function of $\lambda^*(t) - \lambda(t)$. Thinning method classifies the generated events of the process with an intensity function $\lambda^*(t)$ as

- accepted events

The accepted events are the events of the (aimed) process with the intensity function $\lambda(t)$ on the simulation interval $(0, T)$. They are also the events of the process with the hat intensity function λ^* .

- rejected events

The rejected events are only the events of the process with the hat intensity function λ^* on the simulation interval $(0, T)$. They are not the events of the process with intensity function $\lambda(t)$.

The classification of the events are based on the theorem by Lewis and Shedler [4] presented in Section 3.3.3. The number of rejected events depends on the gap between λ and $\lambda(t)$. As a natural consequence, if $\lambda^*(t) = \lambda(t)$ then there aren't any rejected events.

The "+1" exponential random variate assures that the event time of the Poisson process with the hat intensity λ^* exceeds the simulation end time T and the algorithm stops. In addition to this, "+1" indicates the number of processes to be simulated i.e. one stopping condition per process. The "+1" intensity function $\lambda(t)$ evaluation is the the intensity function evaluation at time 0 (which is the starting time of the NHPP simulation).

This method requires the following "comparison operations"

- Comparison type 1 - to check the stopping condition of the process.

The events times generated by the process with hat intensity $\lambda^*(t)$ is compared to the simulation end time " T ". These event times are also the thinning step times. The process stops if the thinning step time is greater than the simulation end time T .

- Comparison type 2 - to decide an event generated by the process with hat intensity $\lambda^*(t)$ as an (accepted) event of the non-homogeneous Poisson process with (aimed) intensity function $\lambda(t)$ or as a rejected event by comparing the standard uniform random number " u " with the ratio $\frac{\lambda(t)}{\lambda^*(t)}$ at the thinning step time " t ".

Total number of comparison operations are

- (at least) " $n + 1$ " "Comparison type - 1" operations
- (at least) " n " "Comparison type - 2" operations

The lower bounds for these operations are: " $n + 1$ " for "Comparison type - 1" operations and " n " for "Comparison type - 2" operations under the condition that the method doesn't generate any rejected events.

The event times generated by this method are in the increasing order.

4.2.3. The Composition Method

As indicated in Section 3.3.4, decomposition can be used when there is a natural decomposition dictated by the analytic form of the intensity function $\lambda(t) = \lambda_1(t) + \lambda_2(t) + \dots + \lambda_m(t)$. In other words, the process is decomposed into sub-processes with respective intensity functions. Each sub-process can be simulated with one of the methods stated in Section 4.2.1 and in Section 4.2.2. If it is forced to decompose the process into sub-processes, then the number of processes to be simulated increases from "one" to " m " after decomposition where " m " is the number of sub-processes after decomposition. Generation of the process with composition method with simulating its sub-processes with one the methods stated, increases total number random variates by " $m - 1$ ". In the case simulating with

- inversion method using inter-event times, the exponential random variates increases from " $n + 1$ " to " $n + m$ "
- inversion method using order statistics and time re-scaling theorem, the Poisson random variates increases from " $+1$ " to " m "
- thinning method, the exponential random variates increases from (at least) " $n+1$ " to (at least) " $n + m$ "

The number " n " in the thinning method case represents the number of accepted

events. There are also rejected events. The method requires same number of random variates to generate each rejected event as each accepted event. Therefore, the number of random numbers is stated as "at least". If n^* is defined as total number of accepted and the rejected event, then the statement will be formulated as "the exponential random variates increases from $n^* + 1$ to $n^* + m$ ".

On the other hand, there aren't any rejected events in other methods therefore the numbers are exact.

To explain the reason of this increase in the number of random variates, there are two ways, a short way and a detailed way. The short way is as follows:

At the both inversion methods: Suppose total number of events which are generated by the process is " n " and the number of events which are generated by each of the sub-processes are " n_i " where $i = 1, \dots, m$. The sum of the events generated by these sub-processes is equal to the number of events which are generated by the process, $n_1 + \dots + n_m = n$. For each sub-process i , $n_i + 1$ random numbers are required. Total number of random numbers required to generate " m " sub-processes are $n_1 + \dots + n_m + m = n + m$ whereas total number of random numbers to generate the process without decomposing the process is " $n + 1$ ". The increase in the random numbers at the thinning can be explained in the similar way.

The detailed way is as follows: The meaning of "+1" random number in the non-homogeneous Poisson process generation methods must be explained.

- In the inversion method using inter-event times, the "+1" (last) exponential random variate in the cumulative sum of exponential random variates assures that the cumulative sum of exponential random variates exceeds the area under the intensity function so that the generation process terminates. The "+1" (last) exponential random variate indirectly determines the number events generated by the process which the total number exponential random variates in the cumulative sum minus one.

- In the inversion method using order statistics and time re-scaling theorem, the "+1" Poisson random variate directly determines the number events generated by the non-homogeneous Poisson process.
- In the thinning method, the process with the hat intensity function generates the candidate event times at the thinning step for the non-homogeneous Poisson process. Like in the inversion method using inter-event times, the "+1" (last) exponential number assures that the cumulative sum of exponential random variates exceeds the area under the hat intensity function so that the generation process terminates. This "+1" (last) exponential number indirectly determines the number of (accepted) events generated by the process.

The operations in methods which uses the "+1" random variates happen once during the whole process generation. The "+1" random number can be interpreted as it (directly or indirectly) determines the number of events generated by the process. If a process is decomposed into " m " sub-processes, then the number of processes to be generated increases from one to " m ". This result indicates that the meaning of "+1" and " m " are the numbers of the processes to be simulated separately in the non-homogeneous Poisson process generation. This is the disadvantage of the composition method in the non-homogeneous Poisson process generation. However, composition method has an advantage in the non-homogeneous Poisson process generation. It classifies the events generated by the non-homogeneous Poisson process according to the sub-processes.

4.2.4. Performance Comparison of Different NHPP Generation Methods

The number of operations (random variates, intensity function and inverse cumulative intensity evaluations) of the non-homogeneous Poisson process generation methods (thinning and inversion methods) are counted. The additional operations (comparing and sorting) are identified and counted for each method. The non-homogeneous Poisson process generation methods are as follows:

- The thinning method
- The inversion method
 - with using inter-event times
 - with order statistics and time re-scaling theorem
- The composition method (the sub-processes are simulated with one of the above stated methods)

An important note is that there is a (one) process to be simulated by thinning method and inversion methods. However, there are " m " (sub)processes to be simulated by the composition method because the process is decomposed into " m " processes.

The total number of main and additional operations which are required by the NHPP generation with "thinning and inversion methods" on the simulation interval $(0, T)$ are listed in Table 4.2.

Table 4.2. The total number of the operations which are required by the NHPP simulation with the thinning and inversion methods

		METHODS		
		Thinning	Inversion	Inversion with
MAIN OPERATIONS				Order Statistic
				Time Re-scaling
Random variates	Exp.	(at least) $n+1$	$n+1$	
	Uniform	(at least) n		n
	Poisson			1
Intensity Function		(at least) $n+1$		
Inv. Cum. Intensity			n	n
Additional operations				
Comparison type-1		(at least) $n+1$	$n+1$	
Comparison type-2		(at least) n		
Sorting				required

where

- n = total number of events generated by the non-homogeneous Poisson process
- Comparison type-1 = comparison operation for the stopping condition of the process
- Comparison type-2 = comparison operation for deciding the events as accepted or rejected events by the thinning method.

The total number of main operations and additional operations which are required by the NHPP generation with the "composition method" on the simulation interval $(0, T)$ is listed in Table 4.3:

Table 4.3. The total number of the operations which are required by the NHPP simulation with the composition method

		METHOD	COMPOSITION	
		sub-processes	simulated	with
		Thinning	Inversion	Inversion with
MAIN OPERATIONS				Order Statistic
				Time Re-scaling
Random variates	Exp.	(at least) $n+m$	$n+m$	
	Uniform	(at least) n		n
	Poisson			m
Intensity Function		(at least) $n+m$		
Inv. Cum. Intensity			n	n
Additional operations				
Comparison type-1		(at least) $n+m$	$n+m$	
Comparison type-2		(at least) n		
Sorting		required	required	required

where

- n = total number of events generated by the non-homogeneous Poisson process
- m = number of sub-processes
- Comparison type-1 = comparison operation for the stopping condition of the process
- Comparison type-2 = comparison operation for deciding the events as accepted or rejected events by the thinning method.

The main result of Table 4.2 and Table 4.3 is that, the composition methods need more operations compared to the thinning and inversion methods because the number of processes which are simulated are increasing at the composition method. This result is coherent with the comment of Devroye [35] in Section 3.3.4 "Usually, the composition method is slow because the method must deal with all the individual

intensity rates". The results in Table 4.2 indicate that both of the inversion methods perform better than thinning method. Both inversion methods require same number of random variates. There is no clear difference between these two methods if simulation application forces ordered set of events. If sorting of the events are not required by the simulation application, the inversion method with order statistics and time re-scaling theorem requires fewer number of operation than the other inversion method. Consequently, the inversion method with order statistics and time re-scaling theorem performs better. For small number of event generation, the thinning method can be taken into the consideration because both inversion methods need inverse cumulative intensity function evaluations. If the closed form of the inverse cumulative intensity function is not available or computationally time consuming, numerical methods are required to compute it. This increases the set-up time of the method. As the number of events "n" increases, the negative effect of the set-up time of inversion methods decreases and is going to be negligible.

4.3. The Performance of the Hawkes Process Generation Algorithms

The number of operations required and the behavior (as the number of generated events increases) to simulate the self-exciting (Hawkes) process with different algorithms are investigated for the performance assessment. Before going into the details about the number of operations which are required by the algorithms, the following important issue must be stated. This issue is sorting operation.

- All cluster-based (composition) algorithms (independent of the simulation method of the sub-processes) require "sorting operation" because the generated events by the algorithms are unsorted. One important point about the sorting is when to make it. Sorting operation must be performed after combining all the events of all the sub-processes to constitute Hawkes process event set to prevent unnecessary sorting operations. However, the sorting requirement depends on the the simulation application. There are some simulation applications for which it is sufficient to generate an unsorted set. In such cases sorting is not necessary.

Both cluster-based (composition) algorithms simulating its sub-processes with inversion method can simulate Hawkes process for unbounded simulation end time ($T = \infty$) if the area under the base intensity function is bounded and the area under the excitation function is less than one so that all the events of Hawkes process can be generated (the full cascade). The inversion method provides this advantage. The algorithms which use thinning logic can simulate Hawkes processes only for bounded simulation end time T if constant upper bound for thinning is used. They need a stopping condition for the process. The commonly used stopping condition is a bounded simulation end time T . However cluster-based (composition) algorithms with simulating its sub-processes with inversion method generate also rejected events like the algorithms which use thinning method logic for bounded simulation time ($T < \infty$), because their offspring sub-processes generate events beyond the simulation end time which must be rejected. The number of rejected events depends on the branching factor of Hawkes process. Therefore, the number of operations required for " n " Hawkes process events are a lower bound like the algorithms using thinning method logic. However, if the simulation end time " T " is unbounded, the number of operations required for " n " Hawkes process events are exact. First, the performance of all Hawkes process simulation algorithms are assessed on the bounded time interval $(0, T < \infty)$. Next, the performance of Hawkes process simulation algorithms (which simulate their sub-processes with one of the inversion methods) on the unbounded time interval $(0, T = \infty)$ are investigated.

4.3.1. Ogata's Modified Thinning Algorithm

The Hawkes process which has the conditional intensity function $\lambda(t|H_t)$ is simulated with Ogata's modified thinning algorithm on the bounded time interval $(0, T)$. The following operations are required for each event:

- (at least) an exponential random variate with rate one
- (at least) a standard uniform random number
- (at least) a base intensity function λ_0 evaluation
- (at least) " n_t " excitation function $\varphi(\cdot)$ evaluations, where " n_t " is the number of

events in the process history H_t at the thinning step execution time, " t "

An exponential random variate is used to compute inter-event time between the thinning steps. In other words, it is used to compute the candidate event time for Ogata's modified thinning algorithm (event times of the process with the hat intensity function $\lambda^*(t)$). A base intensity function λ_0 evaluation and " n_t " times excitation function $\varphi(t - t_i)$ evaluations are performed to calculate the conditional intensity function value $\lambda(t|H_t)$ at the thinning step time " t ". The standard uniform random number u is used to decide the candidate event as an accepted event for the Hawkes process or as a rejected event by comparing u with $\frac{\lambda(t|H_t)}{\lambda^*(t)}$ at the thinning step time " t " ($\lambda^*(t)$ is the local hat function). The total number of random variates and the intensity function evaluations required to generate " n " Hawkes process event times on the bounded time interval $(0, T)$ are

- (at least) " $n + 1$ " exponential random variates with rate one
- (at least) " n " standard uniform random numbers
- (at least) " $n + 1$ " base intensity function λ_0 evaluations
- (at least) " $\frac{n^2 - n}{2} + 1$ " excitation function $\varphi(\cdot)$ evaluations

The term "at least" is added in front of the number of operations, because they are a lower bound. The explanation is same as the thinning method in Section 4.2.2. The additional "+1" exponential random variate assures that the last event time of the Poisson process with the hat intensity function λ^* exceeds the simulation end time T and the algorithm stops. The additional "+1" base intensity function evaluation is the evaluation of the conditional intensity function evaluation at time 0 (the starting point of the simulation interval).

The lower bound for the total number of excitation function evaluations, "Total Ex" during the whole simulation period for " n " event times (generated by Hawkes

process and Hawkes process is terminated) is expressed as

$$Total\ Ex = \sum_{i=1}^n i - 1 = \frac{(n-1)n}{2} = \frac{n^2 - n}{2} + 1 \quad (4.1)$$

because before the " n^{th} " event time of the Hawkes process, there are " $n - 1$ " events in Hawkes process history which contribute excitation to the conditional intensity function value through the excitation function. The additional "+1" excitation function evaluation in Equation 4.1 is the value of excitation function at time 0, $\varphi(0)$. This is the increase in the conditional intensity function value which is caused by the new accepted event. This conditional intensity function value (the hat intensity rate of the dominating (hat) function) is used to compute the next thinning step. About the rejected event times: each rejected event requires same number of operations like an accepted event in Ogata's modified thinning algorithm. The number of operations for these rejected events increases the total number of operations but they don't change the nature of polynomial increase with order two (i.e quadratic increase) of the number of operations.

This algorithm requires same "comparison operations" like the thinning method for the NHPP simulation presented in Section 4.2.2. Total number of comparison operations on the bounded time interval $(0, T)$ are

- (at least) " $n + 1$ " "Comparison type - 1" operations
- (at least) " n " "Comparison type - 2" operations

Beside "+1" "Comparison type - 1" operation together with "+1" exponential random variate with rate one is used for the stopping condition Hawkes process.

The event times generated by this algorithm are in increasing order. As a summary, the total number of operations increases quadratically (in other words polynomial with order two) by Hawkes process generation with Ogata's modified thinning algorithm. This is coherent with the running time of the Ogata's thinning algorithm

$O(N^2)$ [44].

In the literature, it is stated that Ogata's modified thinning algorithm cannot classify the events as immigrants or offspring [44, 48]. However, Ogata's modified thinning algorithm can classify the events as immigrant events and offspring events by adding the the following steps to the algorithm:

- At the thinning step at time "t" if the $u \leq \frac{\lambda(t|H_t)}{\lambda^*(t)}$ is satisfied, then the event at time "t" is an Hawkes (accepted) event generated by the algorithm.
- For a Hawkes (accepted) event at time "t" if $u \leq \frac{\lambda_0(t)}{\lambda^*(t)}$ is satisfied than the Hawkes (accepted) event is an immigrant event otherwise an offspring event.
(where the conditional intensity function $\lambda(t|H_t)$ is replaced with the base intensity function $\lambda_0(t)$ and the same uniform random number is used for both comparisons)

Before the first event of a Hawkes process it is clear that the conditional intensity function $\lambda(t|H_t)$ is equal to the base intensity function $\lambda_0(t)$ because there aren't any events in the process history H_t . Thus, it is for sure that the first Hawkes (accepted) event must be an immigrant. Like the cluster-based (composition) algorithm, Ogata's modified thinning algorithm can classify the events as immigrants and offspring with identifying their ascendants and descendants. One could check between which of the two shifted excitation function values of the sum in the conditional intensity function the standard uniform random number lies to form also the clusters, but it is slow compared the cluster-based (composition) algorithm. As mentioned in Section 2.3, the exponential (decay) function has a special feature. Its rate of decay is constant for all t according to the law of exponential change in Equation 2.55 [30]. If the exponential decay function is selected as excitation function of Hawkes process, the total number of the operations which are required to simulate the Hawkes process are presented as follows:

- (at least) " $n + 1$ " exponential random variates with rate one
- (at least) " n " standard uniform random numbers
- (at least) " $n + 1$ " base intensity function λ_0 evaluations
- (at least) " $n + 1$ " excitation function $\varphi(\cdot)$ evaluations
- (at least) " $n + 1$ " "Comparison type - 1" operations
- (at least) " n " "Comparison type - 2" operations

As a result, the number of the excitation function evaluations decreases from (at least) " $\frac{n^2-n}{2} + 1$ " to (at least) " $n + 1$ " because the excitation which are caused by the events in the process history don't need to be computed one by one at every thinning step. Instead of this computationally time consuming operation, the sum of the excitation is updated by multiplying the sum of excitation at the previous thinning step with $e^{-\beta t}$ (which models the memory over time) in Equation 2.52 in Section 2.3 where " t " the time between successive thinning steps. This operation must be executed at every thinning step and its total number of the executions is (at least) " n ". The "1" excitation function evaluation is the value of the excitation function at time 0, $\varphi(0)$. This is the increase in the conditional intensity function value which is caused by the new accepted event. (Note: $e^{-\beta t}$ is an exponential distribution with α , the magnitude of influence is one)

4.3.2. The Cluster-based (Composition) Algorithm

The cluster-based (composition) algorithms compose a Hawkes process by combining " $n + 1$ " small sub-processes, an (" $+1$ ") immigrant sub-process which is triggered external sources and " n " offspring sub-processes which are triggered by internal sources, namely by each of the " n " events in the process history. The number of processes which are generated by the algorithm increases from "one" to " $n + 1$ " after decomposition. The sub-processes can be simulated with

- thinning method
- inversion method using inter-event times
- inversion method using order statistics and time re-scaling theorem

Cluster-based (Composition) Algorithm - Simulating its Sub-processes with Thinning method: The sub-processes of Hawkes process are generated by the thinning method. For each event generation on the bounded time interval $(0, T)$, the following operations (as lower bound) are required:

- (at least) an exponential random variate,
- (at least) a standard uniform random number,
- (at least) an evaluation of sub-process intensity functions (base intensity function λ_0 or excitation function $\varphi(\cdot)$)

The explanations of the operations are same as stated in Section 4.2.2 because the sub-processes are generated with thinning method. Total number of random variates and intensity function evaluations (as lower bound) which are required to generate " n " Hawkes process event times on the bounded time interval $(0, T)$ are

- (at least) " $2n + 1$ " exponential random variates with rate one
- (at least) " n " standard uniform random numbers
- (at least) " n " evaluations of sub-process intensity functions (base intensity function λ_0 or excitation function $\varphi(\cdot)$)

The explanation of (at least) " $2n + 1$ " exponential random variates is as follows: (at least) " n " exponential random variates are used to generate the event times of the sub-processes which are simulated with thinning method. The similar details are presented in Section 4.2.2 for the thinning method. The additional " $n + 1$ " exponential random variates are used for the stopping condition of " $n + 1$ " sub-processes which are simulated by the algorithm. The difference between the thinning method in Section 4.2.2 and this algorithm is that the number of processes to be generated are increasing

from "one" to " $n + 1$ " after decomposition.

This algorithm requires same "comparison operations" for its sub-process generations like the thinning method for the non-homogeneous Poisson process simulation which is presented in Section 4.2.2. Total number of comparison operations on the bounded time interval $(0, T)$ are

- (at least) " $2n + 1$ " "Comparison type - 1" operations
- (at least) " n " "Comparison type - 2" operations

Each "Comparison type - 1" operation uses an exponential random variate and each "Comparison type - 2" operation uses a uniform random number. Therefore, the number of exponential random variates and the number of "Comparison type - 1" operations are equal. The same explanation is valid for uniform random numbers and "Comparison type - 2" operations. The term "at least" is added in front of the number of operations, because they are lower bounds. The explanation is same as the thinning method in Section 4.2.2. An important note about this algorithm is that the offspring sub-processes which are generated by the thinning method don't generate any Hawkes process event times which are beyond the simulation interval " $(0, T)$ " because the thinning method simulates the sub-processes on the simulation interval " $(0, T - t_{parent})$ " where t_{parent} is the parent event time (in the Hawkes process history) and the start time of the sub-process. The parent event triggers the offspring sub-process.

Simulating a (immigrant and offspring) sub-process with thinning method generates rejected events at the thinning step. The number of operations for these rejected events increases total number of operations but they don't change the nature of the linear increase of the number of operations as the number of event times increases. To clarify this statement:

- The sub-processes can only be generated on the bounded interval $(0, T < \infty)$ with the thinning method otherwise the sub-process doesn't stop.
- The process with the hat intensity function λ^* can generate a bounded number of events " n_{hat} " on the interval $(0, T < \infty)$.
- The same number of operations are required for each of the " n_{hat} " events generated by the process with the hat intensity function λ^*
- The thinning operation classifies these " n_{hat} " events as
 - The " n " events of the sub-process with the intensity function (base intensity function λ_0 or the excitation function $\varphi(.)$). These " n " events are accepted events for the process at the thinning operation.
 - " $n_{rejected}$ " rejected events.
 - $n_{hat} = n + n_{rejected}$
- The composition of the sub-processes constitutes Hawkes process.
- The upper bound of the total number of operations for " n " events of Hawkes process is $(0 < n_{rejected} < \infty)$
 - " $2n_{hat} + 1$ " exponential random variates with rate one
 - " n_{hat} " standard uniform random numbers
 - " n_{hat} " evaluations of the sub-process intensity functions (base intensity function λ_0 or excitation function $\varphi(.)$)
 - " $2n_{hat} + 1$ " "Comparison type - 1" operations
 - " n_{hat} " "Comparison type - 2" operations
- The lower bound of the total number of operations for " n " events of Hawkes process is ($n_{rejected} = 0$ this leads to $n_{hat} = n$)
 - " $2n + 1$ " exponential random variates with rate one
 - " n " standard uniform random numbers
 - " n " evaluations of the sub-process intensity functions (base intensity function λ_0 or excitation function $\varphi(.)$)
 - " $2n + 1$ " "Comparison type - 1" operations
 - " n " "Comparison type - 2" operations

The real number of rejected events depends on the excitation function $\varphi(.)$ of Hawkes process and can be only assessed by simulation.

Like all the cluster-based (composition) algorithms, this algorithm requires "sorting operation" for the event times because the event times are unsorted if the simulation application requires ordered event times as output.

Cluster-based (Composition) Algorithm - Simulating its Sub-processes with Inversion Method: The sub-processes of the Hawkes process are generated by the inversion method. For each event generation on the bounded time interval $(0, T)$, the following operations are required:

- (at least) an exponential random variate with rate one,
- (at least) an evaluation of the inverse cumulative base intensity function $\Lambda_0^{-1}(t)$ or inverse cumulative of excitation function $\Phi^{-1}(t)$ depending on the sub-process generated, immigrant or offspring sub-process respectively

The exponential random variate with rate one determines the area under the intensity function $\lambda(t)$ between the successive events. The inverse cumulative intensity function $\Lambda^{-1}(t)$ uses this exponential random variate to compute the inter-event time between the successive events.

Total number of random variates and inverse cumulative intensity function evaluations to generate "n" Hawkes process event times on the bounded time interval $(0, T)$ are

- (at least) " $2n + 1$ " exponential random variates with rate one
- (at least) " n " evaluations of the inverse cumulative base intensity function $\Lambda_0^{-1}(t)$ or inverse cumulative of excitation function $\Phi^{-1}(t)$

The explanation of " $2n + 1$ " exponential random variates is as follows: " n " exponential random variates are used to generate the event times of the sub-processes. The additional " $n + 1$ " exponential random variates are used for the stopping conditions of " $n + 1$ " sub-processes simulated by the algorithm as stated in Section 4.2.1. The difference is that the number of processes which are generated are increasing from "one" to " $n + 1$ " after decomposition.

This method requires same "comparison operation" like the Poisson process generation with inter-event times in Section 4.1. Total number of comparison operations on the bounded time interval $(0, T)$ are

- (at least) " $2n + 1$ " "Comparison type - 1" operations

Each "Comparison type - 1" operation uses an exponential random variate.

The term "at least" is added in front of the number of operations, because offspring sub-processes can generate the Hawkes process event times beyond the bounded time interval " $(0, T < \infty)$ " which must be rejected. The detailed explanation is as follows:

- The area under the excitation function of the offspring sub-process is computed once for the simulation period $(0, T)$ at the beginning of the simulation as the set-up of the algorithm because it is a time consuming operation if the closed form of inverse cumulative intensity function is not available or is difficult to compute. Numerical methods are required to compute inverse cumulative intensity function.
- This area is the total load of the excitation function of the offspring sub-process on the interval $(0, T)$. The method uses a random amount (which is equal to an exponential random variate with mean one) from this total load, each time to generate an offspring event until the cumulative sum of exponential random variates exceeds the total load of the excitation function and the offspring sub-process stops.
- The event times of the offspring sub-process on the interval $(0, T)$ are computed

with the inverse cumulative intensity function using the cumulative sum of exponential random variates with mean one as input.

- The event times of the Hawkes process are computed by adding the event times of the offspring process (which are inter-event times between the parent event and its offspring) to their parent event time which triggers the sub-process.
- This sum can be greater than the simulation end T if T is bounded. As a result, the offspring sub-processes can generate event times beyond the simulation interval end time " T ". These events must be rejected.
- The ratio of the rejected events to the events of Hawkes process is increasing if the simulation period is small i.e. T approaches to zero. If the simulation period increases (simulation end time T increases), the ratio decreases.

An important point which must be noticed is that the number of operations required to simulate Hawkes process becomes exact in the limit case where the simulation interval T is unbounded ($T = \infty$), because there aren't any rejected events. Like all the cluster-based (composition) algorithms, this algorithm requires "sorting operation" for the generated event times because the event times are unsorted if the simulation application requires ordered event times as output.

Cluster-based (Composition) Algorithm - Simulating its Sub-processes with Inversion Method using Order Statistics and Time Re-scaling Theorem: The sub-processes of Hawkes process are generated by inversion method using order statistics and time re-scaling theorem. For each event generation on the bounded time interval, the following operations are required:

- (at least) a Poisson random variate with mean equals to the area under the excitation function of the offspring sub-process,
- (at least) a standard uniform number,
- (at least) an evaluation of inverse cumulative base intensity function $\Lambda_0^{-1}(t)$ or an evaluation of inverse cumulative excitation function $\Phi^{-1}(t)$

The Poisson random variate with mean equals to the area under intensity function of the sub-process determines the number of events of the sub-process. The standard uniform random number and the evaluation of inverse cumulative base intensity function $\Lambda_0^{-1}(t)$ (or of excitation function $\Phi^{-1}(t)$) are used to compute the event time of the immigrant (or offspring) sub-process respectively. Total number of random variates and inverse cumulative intensity function evaluations to generate " n " Hawkes process event times on the bounded time interval are

- " $n+1$ " Poisson random variates (" n " Poisson random variates with mean equal to area under the intensity function of the offspring sub-process and a $(+1)$ Poisson random variate with mean, equal to area under the intensity function of the immigrant sub-process)
- (at least) " n " standard uniform random numbers
- (at least) " n " evaluations of inverse cumulative base intensity function $\Lambda_0^{-1}(t)$ or inverse cumulative of excitation function $\Phi^{-1}(t)$

The " $n+1$ " Poisson random variates are required to determine the number of events for " $n+1$ " sub-processes (an $(+1)$ immigrant process and " n " offspring sub-processes). This is similar to the non-homogeneous Poisson process generation with inversion, order statistics and time re-scaling theorem in Section 4.2.1. The difference is that the number of processes to be generated increases from "one" to " $n+1$ " after decomposition. This method doesn't require "comparison operation" like the Poisson process generation with inversion, order statistic and time re-scaling theorem in Section 4.2.1. The term "at least" is added in front of the number of operations, because offspring sub-processes can generate Hawkes process event times beyond the bounded time interval " $(0, T < \infty)$ " which must be rejected. The detailed explanation is as follows:

- The area under the excitation function of the offspring sub-process is computed once for the simulation period $(0, T)$ at the beginning of the simulation as set-up of the process because it is a time consuming operation if the closed form of the inverse cumulative intensity function is not available or is difficult to compute.

Numerical methods are required to compute the inverse cumulative intensity function.

- This area is the mean of the Poisson random variable which determines the number of events of the offspring sub-process on the simulation period $(0, T)$.
- The event times of the offspring sub-process on the interval $(0, T)$ are computed using order statistic and inverse cumulative intensity function.
- The event times of the Hawkes process are computed by adding the event times of the offspring process (which are inter-event times between the parent event and its offspring) to their parent event time which triggers the sub-process.
- This sum can be greater than the simulation end T if T is bounded. As a result, the offspring sub-processes can generate event times beyond the simulation interval end time " T ". These events must be rejected.
- The ratio of rejected events to the events of Hawkes process is increasing if the simulation period is small i.e. T approaches to zero. If the simulation period increases (simulation end time T increases), the ratio decreases.

An important point which must be noticed is that the number of operations required to simulate Hawkes process becomes exact in the limit case where the simulation interval T is unbounded ($T = \infty$), because there aren't any rejected events.

Like all the cluster-based (composition) algorithms, this algorithm requires "sorting operation" for the event times generated because the event times are unsorted if the simulation application requires ordered event times as output.

4.3.3. The Performance Comparison of the HP Generation Algorithms

Before comparing the performances of different Hawkes process generation algorithms, it would be helpful to summarize again some of the properties of algorithms. Hawkes process is a doubly stochastic process because the conditional intensity function of Hawkes process is itself a stochastic process. Decomposing the stochastic conditional intensity function into the deterministic base intensity function λ_0 and the determin-

istic excitation function $\varphi(t)$ reduces doubly stochastic Hawkes process into stochastic immigrant and offspring sub-processes which are independent non-homogeneous Poisson processes. The sub-processes can be simulated with a smooth general intensity function with thinning and inversion methods [42]. This is an important advantage of cluster-based (composition) algorithm. To simulate the self-exciting (Hawkes) process with Ogata's modified thinning algorithm, the algorithm requires monotone function base intensity $\lambda_0(t)$ and excitation functions $\varphi(t)$. A drawback of the decomposition is that the number of processes which must be simulated are increasing. Instead of a process generation with many events, a lot of processes with small number of events are simulated. For the bounded simulation end time, composition methods generate events beyond the simulation end time T which must be rejected. For the unbounded simulation end time, composition methods doesn't generate any events which must be rejected.

For the performance comparison of the algorithms, the number of operations (random variates, intensity function and inverse cumulative intensity evaluations) of Hawkes generation algorithms are counted. Additional operations (sorting and comparing) are identified and counted for each algorithm. Hawkes process generation algorithms are as follows:

- Ogata's modified thinning algorithm
- Hawkes process generation by clusters whose sub-processes are generated with
 - thinning method
 - inversion method
 - inversion method using order statistics and time re-scaling theorem

Firstly, the performance comparisons of the Hawkes process generation algorithms are summarized for the unbounded time interval to differentiate the performances of the algorithms. The results are listed in Table 4.4. Secondly, the performance assessments of the Hawkes process generation algorithms are summarized for the bounded time interval. The results are listed in Table 4.5. However, the results for the bounded

interval are lower bounds because the algorithms generate rejected events, but type of the rejected events are different. The difference between different types of rejected events can be explained as below:

- For Ogata's modified thinning algorithm and cluster-based (composition) algorithm simulating its sub-processes with thinning method, the cause of the rejected events is the thinning step of the algorithm. The cause of the rejected events doesn't depend on the simulation end time " T ". If " T " increases, the rejected events don't disappear. Therefore, the number of operations required for these algorithms remains as "lower bounds" as the simulation end time " T " increases and goes to infinity.
- However, for the other cluster-based (composition) algorithms (which simulates its sub-processes with one of the inversion methods), the cause of the rejected events depends on the simulation end time " T ". Only the events which are generated beyond " T " are rejected. If " T " is unbounded (" $T \rightarrow \infty$ "), there aren't any events beyond the simulation end time " T ". The number of operations which are required by these algorithms becomes exact as presented in Table 4.4.

As a conclusion, the performances of the algorithms cannot be clearly differentiated by counting the number of operations if the Hawkes process is simulated only on the bounded interval.

For the first case: The total number of operations which are required by the Hawkes process generation algorithms on the unbounded time interval " $(0, T = \infty)$ " are listed in Table 4.4.

Table 4.4. The total number of operations which are required by the Hawkes process simulation algorithms on the unbounded time interval

		ALGORITHM	
		Cluster	
		Subprocesses	simulated with
		Inversion	Inversion with
MAIN OPERATIONS			Order Statistic
			Time Re-scaling
Random variate	Exp.	2n+1	
	Uniform		n
	Poisson		n+1
Inv.Cum. Intensity		n	n
Additional operations			
Comparison type-1		2n+1	
Sorting		required	required

where

- cluster = Hawkes simulation algorithm with clusters
- n= total number of events of Hawkes process (Number of Hawkes process events in the process history H_t , all generations together)
- Comparison type-1 = comparison operation for the stopping condition of the process

The number of operations in Table 4.4 are exact, because there aren't any rejected events if the simulation interval is unbounded. The cluster-based (composition) algorithm inversion method using order statistics and time re-scaling theorem requires the smallest number of operations on the unbounded simulation interval.

For the second case: The total number of operations which are required by the Hawkes process generation algorithms on the bounded time interval " $(0, T < \infty)$ " are listed in Table 4.5.

Table 4.5. The total number of the operations which are required for the Hawkes process simulation algorithms on the bounded time interval

		ALGORITHM				
		Ogata's	Ogata's	Cluster		
		general	exponential	Subprocesses	simulated	with
		excitation func.	decay	Thinning	Inversion	Inversion with
MAIN OPERATIONS						Order Statistic
						Time Re-scaling
Random variate	Exp.	(at least) $n+1$	(at least) $n+1$	(at least) $2n+1$	(at least) $2n+1$	
	Uniform	(at least) n	(at least) n	(at least) n		(at least) n
	Poisson					$n+1$
Intensity Function	Base int.	(at least) $n+1$	(at least) $n+1$	(at least) n_0		
	Excitation	(at least) $\frac{n^2-n}{2} + 1$	(at least) $n+1$	(at least) $n - n_0$		
Inv.Cum. Intensity					(at least) n	(at least) n
Additional operations						
Comparison type-1		(at least) $n+1$	(at least) $n+1$	(at least) $2n+1$	(at least) $2n+1$	
Comparison type-2		(at least) n	(at least) n	(at least) n		
Sorting				required	required	required

where

- Ogata's = Ogata's modified thinning algorithm
- cluster = Hawkes simulation algorithm with clusters
- n = total number of events of the Hawkes process (the number of Hawkes process events in the process history H_t , all the generations together)
- n_0 = number of immigrant events of Hawkes process (generation 0)
- Comparison type-1 = comparison operation for the stopping condition of the process
- Comparison type-2 = comparison operation for deciding the events as accepted or rejected events by the thinning method.

The results based on the bounded simulation interval are summarized in Table 4.5. Although the cluster-based (composition) algorithm (which simulates the sub-processes with inversion method using order statistics and time re-scaling theorem) requires the smallest "lower bounds" for the number of operations in Table 4.5, these results don't indicate explicitly to decide the fastest algorithm, because the numbers are lower

bounds in the table.

To reach an explicit conclusion, it is necessary to combine the results of the unbounded simulation interval in Table 4.4 and the results of the bounded simulation interval in Table 4.5. This combination explicitly indicates that the cluster-based (composition) algorithm inversion method using order statistics and time re-scaling theorem requires the smallest number of operations among the algorithms. It is theoretically the fastest algorithm.

The summary of this chapter which is based on the results in Table 4.4 and Table 4.5 is as follows:

- The cluster-based (composition) algorithm - simulating its sub-processes with inversion method using order statistics and time re-scaling theorem requires the smallest number of operations among all the Hawkes process generation algorithms including the special case of Ogata's modified thinning algorithm with exponential decay excitation function. Exponential decay function is the popular excitation function of the Hawkes process in the literature because the quadratic order of complexity of Ogata's modified thinning algorithm decreases to linear order of complexity.
- Decomposing Hawkes process into sub-processes decreases the quadratic (polynomial with order two) increase of the number of operations in Ogata's modified thinning algorithm [3] to linear increase of the number of operations in cluster-based algorithms as the number of events increases.

The number of rejected events goes to zero as the simulation end time " T " increases and goes to infinity by the cluster-based (composition) algorithms which simulates the sub-processes with one of the inversion methods.

4.4. The Pseudo Code of The Algorithm

According to the performance assessments of the Hawkes process generation algorithms in Chapter 4, the proposed cluster-based (Composition) algorithm - simulating its sub-processes with inversion method using order statistics and time re-scaling theorem requires less number of operations compared to the other algorithms. This algorithm is strongly recommended and therefore all the details of its implementation is given. This algorithm can also simulate the extension of Hawkes process, renewal Hawkes process and marked Hawkes process. Furthermore, the algorithm can simulate Galton Watson branching process.

The steps of the cluster-based (composition) algorithm is presented in Section 3.4.3 which is a recursive algorithm with a loop. Each execution of this loop generates the offspring of a sub-process which is triggered by an event in the process history H_t . The total number of executions is equal to the number of events " n " in the process history. The alternative simulation is generation by generation simulation instead of sub-process by sub-process simulation. This alternative is going to reduce the number of executions of the loop from the number of events in the process history to the number of generations of the process. This is based on the vectorization of the code which may improve computational efficiency by performing basic operations on entire arrays rather than on individual array elements inside nested loops (Lebanon and El-Geish 2018).

The events of the Hawkes process are simulated "generation by generation" by this algorithm. The structure of the generation-by-generation algorithm is as follows:

- (i) Generate the events of the immigrant sub-process as generation 0.
- (ii) Generate the direct offspring of the events in generation " $i-1$ " using the offspring sub-process as generation $i = 1, 2, 3, \dots, n$ recursively until there is no event in generation " n ".
- (iii) Combine the generations $0, 1, 2, 3, \dots, n$ to constitute the Hawkes process events.
- (iv) Sort the Hawkes process events because they are unsorted if simulation applica-

tion forces ordered event times.

The sub-processes at step (i) and (ii) are non-homogeneous Poisson processes. They are simulated with a non-homogeneous Poisson process generation method that combines inversion method, order statistics and time re-scaling theorem presented in Section 3.3.2 (Inversion Algorithm with Order Statistics for the Non-homogeneous Poisson Process) because this method for the non-homogeneous Poisson process generation requires smallest number of operations compared to the other methods stated in Section 3.3 (Non-Homogeneous Poisson Process Simulation Algorithms) according to the results in Section 4.2.4 (Comparing the Performances of the different non-homogeneous Poisson Process Generation Methods) and to the results in Section 4.3.3 (Comparing the Performances of different Hawkes Process Generation Algorithms). The event times which are generated by this non-homogeneous Poisson process generation method is not sorted. It would be computationally time saving to sort the event times after generating all the event times of all the sub-processes and combining them to constitute the Hawkes process event times. The steps of the cluster-based (composition) algorithm (generation by generation):

The inputs of the algorithm

- immigrant process intensity function (base intensity function), $\lambda_0(t)$
- offspring intensity function (excitation function), $\varphi(t)$
- simulation interval $(0, T)$

The set-up of the algorithm

- Compute $\Lambda_0(T)$
 - ... the area under the immigrant process intensity function (the base intensity function) on the simulation interval $(0, T)$
 - ... the branching factor n^* which is stated in Equation 2.60
- Compute $\Phi(T)$

... the area under the offspring process intensity function (the excitation function) on the simulation interval $(0, T)$

- Compute the inverse cumulative base intensity function, $\Lambda_0^{-1}(t)$
- Compute the inverse cumulative excitation function, $\Phi^{-1}(t)$

The steps of the algorithm

- (i) Simulate the immigrant sub-process to generate the immigrant events " i " on the simulation interval $(0, T)$
 - (a) Generate a Poisson random variate " p_i " with mean $\Lambda_0(T)$
 - (b) Stop if " p_i "=0
 - (c) Generate the " p_i " uniform random numbers " u_j " where $j = 1 \dots p_i$ on the interval $(0, \Lambda_0(T))$
 - (d) Convert the uniform random numbers " u_j " on the interval $(0, \Lambda_0(T))$ to the event times of the immigrant sub-process "*immigrant*" on the interval " $(0, T)$ " by evaluation of the inverse cumulative excitation function $\Lambda_0^{-1}(u_j)$.
- (ii) $g \leftarrow 0$

... " g " is a counter for the generation number
- (iii) Add the immigrant events "*immigrant*" in the set " $Generation(g)$ "

... " $Generation(g)$ " is the event set of the g^{th} generation which can generate an offspring
- (iv) Stop if " $Generation(g) = \emptyset$ "

... There aren't any events in the immigrant generation
- (v) Repeat

... This (repeat-until) loop generates the offspring generations

 - (a) Generate a Poisson random variate " p_o " with mean $\Phi(t)$ for each event of the set " $Generation(g)$ ".
 - (b) Sum up these Poisson random variates " p_o " to get the total number of offspring, " $Offspring(g)$ " which are generated by the events of " $generation(g)$ ".

... ($Offspring(g)$ is the total number of children of the events in the set " $Generation(g)$)

- (c) Break if $\text{Offspring}(g)=0$
 - ... Termination condition of the (repeat-until) loop
 - ... There aren't any events in the offspring generation
- (d) Generate the " $\text{Offspring}(g)$ " uniform random numbers " u_o " on the interval $(0, \Phi(T))$
- (e) Convert the " $\text{Offspring}(g)$ " uniform random numbers " u_o " on the interval $(0, \Phi(T))$ to the event times of offspring sub-process on the interval " $(0, T)$ " by evaluation of inverse cumulative excitation function $\Phi^{-1}(u_o)$.
- (f) $g \leftarrow g + 1$ (increase for generation number g by one)
- (g) Compute the event times of " $\text{Generation}(g)$ " by summing up the event times of offspring sub-process with the related parent event times.
- (h) Remove the event times from the set " $\text{Generation}(g)$ " which are beyond the simulation end time " T ".
- (vi) Until
- (vii) Combine all the " $\text{Generation}(g)$ " sets to constitute Hawkes process event set " H "
- (viii) Sort the event times in Hawkes process event set " H "
 - ... if simulation application forces ordered event times as output

The output of the algorithm

- Deliver Hawkes process event set " H " as output

This algorithm can simulate

- the Hawkes process for the unbounded simulation period if the $\Lambda_0(T)$ is bounded and $\Phi(T) < 1$. In this case the step "(v).(h)" can be removed because it is obvious that all the events which are generated by algorithm are in the unbounded simulation interval.
- the renewal Hawkes process. As input, the algorithm accepts time varying base intensity function $\lambda_0(t)$, beside a scalar base intensity function λ_0 .

- the marked Hawkes process with independent identically distributed marks " m ". The mean of the Poisson random variate " p_0 " at Step (v.a) is replaced with " $m * \Phi(T)$ " instead of " $\Phi(T)$ ". The marks " m " are independent and identically random numbers.

The advantages of this algorithm are follows:

- It requires the smallest number of random variates compared to the other algorithms in the literature.
It needs less operations compared to the other algorithms.
- It is applicable to the Hawkes process generation with general and smooth intensity functions which are not required to be monotone functions as by the Ogata's modified thinning algorithm.
This is achieved by decomposing the conditional intensity function of the Hawkes process.
- It can simulate Hawkes process for the both bounded simulation interval $(0, T < \infty)$ and the unbounded simulation interval $(0, T = \infty)$.
For the unbounded case, the whole cascade (all the events) can be generated if the area under the base intensity is bounded, and the branching factor (the area under the excitation function) is less than one otherwise the number of events are unbounded. This is achieved by using the inversion method instead of the thinning method.
- It can simulate the renewal Hawkes process beside the original Hawkes process.
In the original Hawkes process, the base intensity function is a constant function λ_0 which corresponds the inter-arrival time of the immigrant events are exponentially distributed with rate λ_0 . In the renewal process, the inter arrival distribution is not restricted as to be exponential distribution. If the inter arrival distribution $F(\cdot)$ of the immigrant events known, the base intensity function of the renewal Hawkes process is computed as follows: $\lambda_0(t) = \frac{f(t)}{1-F(t)}$.
- Beside classifying the events as immigrant and offspring, the algorithm determines the ascendants and descendants of each event (with adding additional steps to

the code). The algorithm with the additional steps to the code is presented in Appendix B.2.

This is achieved by generation-by-generation event simulation.



5. THE IMPLEMENTATION DETAILS

In Chapter 5, the implementation details of the Cluster-based (composition) algorithm - simulating its sub-processes with inversion method using order statistics and time re-scaling theorem are presented.

5.1. The Set-up Steps

The cluster-based (composition) algorithm - simulating its sub-processes with the inversion method using order statistics and time re-scaling theorem requires the following set-ups:

- (i) Compute the inverse of cumulative base intensity function, $\text{Lam0Inv}(t)$
- (ii) Compute the inverse of cumulative excitation function, $\text{PhiInv}(t)$
- (iii) Compute the total area $\text{Lam0}(T)$ under the base intensity function, $\text{lam0}(t)$ during the simulation period $(0, T)$
- (iv) Compute the total area $\text{Phi}(T)$ under the excitation function, $\text{phi}(t)$ during the simulation period $(0, T)$

There are two crucial points for the set-up steps of the algorithm:

- Calculating the areas under the base intensity and excitation functions
- Computing the inverse cumulative base intensity function and inverse cumulative excitation function

As stated in Section 2.2, the number of events which are generated by a NHPP is Poisson distributed with mean that is equal to the area under the intensity function on the simulation interval. Therefore, calculating the areas under the base intensity and excitation functions are necessary. According to "order statistic" stated in Section 2.2, the event times of a NHPP are distributed as order statistics from a sample with

distribution function in Equation 2.39. Inversion method requires the inverse cumulative base intensity and excitation functions to compute the event times of NHPP. If the closed forms of the cumulative intensity function and its inverse function are not available as input of the algorithm, they must be computed with an alternative method in the algorithm. The proposed alternative method is numerical integration and approximating the inverse cumulative intensity function. The steps are as follows:

- Integrate numerically the intensity function, $\lambda(t)$ to approximate the cumulative intensity function, $\Lambda_a(t)$ (the possible methods are closed Newton-Cotes formulas or Gauss-Lobatto rules)
- Computation of the approximate inverse cumulative intensity function, $\Lambda_a^{-1}(t)$ using $\Lambda_a(t)$ with one of the interpolation methods (a good choice is cubic spline interpolation)

There are many methods to approximate $\Lambda^{-1}(t)$ at a desired precision level if $\lambda(t)$ is a smooth function (infinitely differentiable function), and the domain of integration is bounded ($|\lambda(t)| \leq M$ where M is a real number). The points for approximating the inverse cumulative intensity function $\Lambda_a^{-1}(t)$ to be used in Newton's divided-difference interpolation are obtained from the points of the approximated cumulative intensity function $\Lambda_a(t)$ by changing the places of coordinates from (x, y) to (y, x) like $(\Lambda_a(t_0), t_0)$, $(\Lambda_a(t_1), t_1), \dots, (\Lambda_a(t_{n-1}), t_{n-1}), (\Lambda_a(t_n), t_n)$. The inverse cumulative intensity function, $\Lambda_a^{-1}(t)$ receives the area under the intensity function, $\lambda(t)$ that is cumulative sum of exponential random variates with rate one as input and returns the event times of the process (cumulative inter event times) as output.

As a result, the non-homogeneous Poisson process with the intensity function $\lambda(t)$ on the simulation period $(0, T)$ is similar to the Random Variate Generation (RVG). The difference is that the cumulative intensity function $\Lambda(t)$ is a multiple times of a cumulative distribution function, $F(\cdot)$. In the implemented code, the functions from R-package "Runuran" [49] are employed. Further information about approximating a complicated function and numerical integration and the software for these points are

stated in Appendix E. The functions of R package "Runuran" which are used in the code are presented in Appendix F.

5.2. Sorting the Event Times

The event times which are generated by the cluster-based algorithm are unsorted as stated in Section 4.3. The event times must be sorted if the simulation application forces ordered set of event times as output. Only one sorting operation is sufficient and time saving after all the events of all sub-processes are generated (However some software in the literature sorts the event times after each sub-process simulation). Horsmalahti [50] indicated that bucket sort and radix sort are two well-known integer sorting algorithms. The connection between the radix sort and bucket sort can be summarized as radix sort makes several passes of bucket sort, one for each "digit" or significant part of each element.

Radix sort is a non-comparative sorting algorithm. There are buckets whose total number for each iteration is equal to the base of the elements to be sorted (For decimal number elements, there are 10 buckets: 0,1,2,3,4,5,6,7,8,9). Total number of iterations for the algorithm is equal to the maximum digit length of the elements. The missing digits are replaced with zero. For example: 23, 9, 125 will become 023, 009,125). Either starting from the left most digit or from the right most digit: the bucketing process assigns all the unsorted elements into the buckets according to the value of the digit. This bucketing process is repeated for each digit while preserving the ordering of the previous step, until it is applied to all digits of the elements to be sorted. Radix sort can be applied to data that can be sorted lexicographically [22]. Horsmalahti researched and compared the speed performances and memory consumption of the algorithms empirically and theoretically in different use cases and indicated that bucket sort was faster than radix sort, but that bucket sort uses more memory in most cases [50] . Given " n " d -digit numbers in which each digit can take on up to " k " possible values, the worst case and average case/expected running time of bucket sort and radix sort are listed in Table 5.1 [22]:

Table 5.1. The running time of the bucket sort and the radix sort

Algorithm	worst case running time	average case/expected running time
Bucket sort	$\Theta(n^2)$	$\Theta(n)$ (average case)
Radix sort	$\Theta(d(n + k))$	$\Theta(d(n + k))$

According to the results stated in Table 5.1, radix sort performs better at the worst case running time. Additionally, time performance experiments for the following sorting methods are executed using the R package "microbenchmark" [51].

- Radix sort
- Shell sort
- Quick sort

Uniform and exponential random numbers are sorted. The number of random numbers which are sorted in the experiments are 10,000;100,000 and 1,000,000. The experiments results in Table 5.2 are in micro seconds.

Table 5.2. The time performance of different sorting methods in micro seconds

	Uniform	Random	Numbers		Exponential	Random	Numbers
	10,000	100,000	1,000,000		10,000	100,000	1,000,000
Algorithm							
Radix sort	703.869	5,553.092	85,326.247		599.838	4,638.325	81,015.042
Shell sort	1,043.57	10,635.432	131,316.317		992.454	10,644.783	133,955.967
Quick sort	750.580	8,203.684	83,647.457		717.204	7,405.382	84,117.477

According to the experiment results in Table 5.2, the radix sort performance is better than the other sorting methods.

5.3. The Inputs and Outputs of the Software with Sample Cases

The software package "fastHawkes" for Hawkes process simulation is implemented in R project statistical computing [52]. It is based on the algorithm which is presented

in Section 4.4. The code of the software is presented in Appendix B.1. The main function of the R package "fastHawkes" is `rHawkes()`. It can generate Hawkes process with cluster-based algorithm and with Ogata's modified thinning algorithm. Furthermore, it can generate NHPP process with inversion method using order statistics and time re-scaling theorem and thinning method. It can also simulate multiple simulation runs at the same execution. The function "`rHawkes()`" requires the following inputs to simulate Hawkes process:

- `n` : The number of independent non empty Hawkes simulation runs, non-empty cascades
- `T` : Simulation duration, the end point of the simulation period
- `allimmigrants` : The immigrant events supplied by the user instead of simulating them
- `repr` : Mean number of offspring (A non-negative numeric value)
- `marks` : Weight of the event
- `lam0` : The base intensity function of Hawkes process
- `phi` : The excitation function of Hawkes process
- `method` : Simulation method for Hawkes process generation
- `sortevents` : The choice to sort the event times or not

The default values of inputs are

- `k=1`
- `T=infinity`
- `allimmigrants=c(0)`
- `repr=NULL`
- `marks=1`
- `lam0` : any smooth continuous function
- `phi` : any smooth continuous function
- `method`: cluster (The proposed cluster based simulation algorithm)
- `sortevents`: TRUE (events times are sorted)

The outputs are:

- The event times of the Hawkes process for multiple simulation replications
- The simulation replication numbers of the event times

The function "rhawkes()" of the package "fasthawkes" needs the following set-ups to simulate Hawkes process with the conditional intensity function on the interval $(0, T)$:

- The areas under the base intensity function and excitation function which are mean number of events generated by the immigrant and offspring sub-processes, $\Lambda_0(T) = \int_0^T \lambda(t)dt$ and $\Phi(T) = \int_0^T \varphi(t)dt$ respectively.
- The inverse cumulative base intensity and excitation functions, $\Lambda_0^{-1}(t)$ and $\Phi^{-1}(t)$ respectively.

If the closed forms of the cumulative intensity and the inverse cumulative intensity functions of the sub-processes don't exist or cannot be supplied by the user, the function can compute them using numerical integration. For the numerical integration and random variate generation, the implemented package "fastHawkes" uses the R package "Runuran" which computes the inverse cumulative function and the area under the function at the desired level of accuracy for a smooth continuous function. The R package "Runuran" (R Interface to the 'UNU.RAN' Random Variate Generators) was developed by [49]. The details about the functions (PINV and uq) of R package "Runuran" used are in the appendix "The Functions of R Package Runuran". Some of the sample cases which can be simulated with the software are presented below. Their outputs of these cases are given in Appendix B.3.

Simulating non-empty Hawkes process cascades: If the number of events which are generated by the immigrant sub-process is zero, then the cascade which is generated by Hawkes process simulation is empty. The immigrant sub-process of Hawkes process is an independent non-homogeneous Poisson process with base intensity function, $\lambda_0(t)$. The number of events, X which are generated by this process on the interval $(0, T]$

is a Poisson random variable with mean $\Lambda_0(T)$ which is equal to the area under the base intensity function, $X \sim \text{Pois}(\Lambda_0(T))$. The implemented software code doesn't generate empty immigrant sub-process event set. The detail are presented in Section 3.3.2. The R command is for this case is `qpois (runif (1, ppois(0,  $\Lambda_0(T)$ ), 1),  $\Lambda_0(T)$ ))`, where

- `qpois` is the quantile (inverse cumulative distribution) function of the Poisson distribution
- `ppois` is the cumulative distribution function of the Poisson distribution
- `runif` is the uniform random number
- $\Lambda_0(T)$ is the mean of the Poisson distribution

Generating multiple simulation runs: The input n (the number of independent Hawkes simulation runs) allows the user to simulate " n " cascades together at the same execution of the function "`rhawkes`". This feature of the function "`rHawkes`" allows to generate a huge number of data at the same time for statistical studies like estimating parameters.

Simulating a social media post: A social media post can be simulated using the following inputs. The initial social media post which is posted by a social media user is the immigrant event of the Hawkes process. This is defined as the input "`allimmigrants=c(t)`" where " t " is event time of the initial social media post, immigrant event time. The set up of a social media post simulation is the default input values with an excitation function, `phi` and initial immigrant event time = 0.

- `k=1`
- `T=infinity` (Area under the excitation function must be less than one)
- `allimmigrants=c(0)`
- `repr=NULL` or any scalar
- `marks=1` or scalar or random variate like exponential random variable with mean one.

- lam0 : NULL
- phi : any smooth continuous function
- method: cluster (The proposed cluster based simulation algorithm)
- sortevents: TRUE (events times are sorted)

The initial post doesn't need to be a single event generated by a single user. The same social media post can be posted by different social media users at the different times. In this case, there are multiple initial events, immigrants for example "allimmigrants=c(t_1, t_2, t_3)"

Simulating Hawkes process with reproduction number: The input "repr" is the production number which is the mean number of offspring which a parent events can generate. There are two possible options:

- The default case: The reproduction number repr=NULL
In this case the mean number of offspring of the sub-process is the area under the intensity function during the simulation interval which is computed by the function "rHawkes".
- The reproduction number repr=sclar (The reproduction number is supplied as an input by the user)

In this case, first the intensity function of offspring sub-process is standardized so that the area under the intensity function of offspring sub-process becomes one and then multiplied by the reproduction number so that the area under the intensity function of offspring sub-process is equal to "repr" the reproduction number, the mean number of offspring.

Given an intensity function $\lambda(t)$ and the simulation period $(0, T)$: The area under the intensity function on the simulation period $(0, T)$ is expressed as

$$\int_0^T \lambda(u) du. \quad (5.1)$$

The standardized intensity function is expressed as

$$\lambda_{standardized}(t) = \frac{\lambda(t)}{\int_0^T \lambda(u) du}. \quad (5.2)$$

The modified intensity function based on "repr" can be written as

$$\lambda_{modified}(t) = \lambda_{standardized}(t) \text{ repr} = \frac{\lambda(t)}{\int_0^T \lambda(u) du} \text{ repr}. \quad (5.3)$$

Simulating a non-homogeneous Poisson process: The sub-processes of Hawkes process are independent non-homogeneous Poisson processes. Hawkes process reduces to non-homogeneous Poisson process if the excitation function is NULL. The set up of the non-homogeneous Poisson process generation is as follows:

- k=1
- T=finite or infinity
- allimmigrants=NULL
- repr=NULL
- marks=1
- lam0 : any smooth continuous function
- phi : NULL
- method: cluster or thinning
- sortevents: TRUE (events times are sorted)

To simulate bounded number of events, the simulation end time T must be bounded or the area under the intensity function on the simulation interval $(0, T)$ must be bounded.

Simulating Hawkes process with different algorithms: The input "method" allows the user to simulate Hawkes process with the following algorithms:

- Cluster based algorithm
- Ogata's modified thinning algorithm

Finally, the software can simulate Hawkes process for bounded $(0, T < \infty)$ and unbounded $(0, T = \infty)$ simulation intervals.



6. THE PERFORMANCE EXPERIMENTS

Chapter 6 includes the empirical studies. The theoretical results of Chapter 4 are compared with empirical results in the following sections.

The following experiments are performed:

- (i) The time performance comparison between other software and the implemented software which is based on the presented cluster-based algorithm in Section 6.1
- (ii) The time performance comparison between Ogata's modified thinning algorithm and the presented cluster-based algorithm in Section 6.2
- (iii) The number of operations comparison between Ogata's modified thinning algorithm and presented cluster-based algorithm in Section 6.3
- (iv) The time performance comparison between sorted and unsorted events versions of the presented cluster-based algorithm in Section 6.4

Polynomial regression up to degree five is employed to quantify the relationship between the variables. As the degree of polynomial regression for the model increases polynomial regression can fit nonlinear relationships better, but in practice it is rarely chosen the degree to be greater than 3 or 4. Beyond this point, the model becomes too flexible and over-fits the data. The code of the software for the presented cluster-based algorithm which is used in these experiments is given in Appendix B.1.

6.1. The Performance Comparison with Other Softwares

The time experiments in this section are performed with the bounded simulation duration, $T < \infty$ instead of $T = \infty$ as an input because the packages in the literature work only with the bounded simulation duration. The speed performances of the implemented software based on the presented cluster-based Hawkes process generation algorithm is compared with the existing software in the literature. These packages are

as follows:

- R Package – IHSEP [8]
- R Package – RHawkes [9]
- R Package – hawkesbow [10]
- R Package – hawkes [11]

Then additional experiments are going to be performed with the last two packages, "hawkesbow" and "hawkes" because of their special features.

The first three packages (IHSEP, RHawkes, hawkesbow) simulates Hawkes process with the clusters which is based on the cluster process representation of Hawkes process. The last one ("hawkes") simulates Hawkes process with Ogata's modified thinning algorithm. It simulates Hawkes process only with "scalar" base intensity function and "exponential decay" excitation function. In this setting, the number of excitation function evaluations increases linearly (not quadratic) as the number of events increases because of the special property of the exponential decay function (which is stated in Section 3.4.2), the decay rate is constant over the time (independent of time). It is not required to calculate the contribution of each event in the process history to the conditional intensity function of Hawkes one by one. It is sufficient to multiply the value of the conditional intensity by $e^{-\lambda t}$ (which models the memory over time) where t is time between the last and the current thinning step points. The time experiments are performed with the R command "system.time".

Time experiments with IHSEP: The experiment results are as follows:

- IHSEP - simHawkes0: 47.076 events in 3,81 system time
- IHSEP - simHawkes1: 47.047 events in 3,79 system time
- IHSEP - simHawkes2: 47.231 events in 1,03 system time
- fastHawkes - rHawkes: 47.107 events in 0,14 system time

As a result, "rhawkes" performs better. The experiment outputs (the number of events and their duration) are in appendix D.1 in Table D.1.

Time experiments with RHawkes: The experiment results are as follows:

- RHawkes - simRHawkes1: 1.298.961 events in 14,25 system time
- fastHawkes - rHawkes: 1.292.519 events in 0,32 system time

As a result, "rhawkes" performs better. The experiment outputs (the number of events and their duration) are in appendix D.1 in Table D.2.

Time experiments with "hawkesbow": The experiment results are as follows:

- hawkesbow - hawkes: 2.000.751 events in 10,88 system time
- fastHawkes - rHawkes: 1.999.488 events in 0,61 system time

As a result, "rhawkes" performs better. The experiment outputs (the number of events and their duration) are in appendix D.1 in Table D.3

Time experiments with "hawkes": The experiment results are as follows:

- hawkes - simulateHawkes: 1.999.611 events in 2,20 system time
- fastHawkes - rHawkes: 1.998.854 events in 0,51 system time

As a result, "rhawkes" performs better. The experiment outputs (the number of generated events and their duration) are in appendix D.1 in Table D.4

Additional Experiments with the Packages "hawkesbow" and "hawkes": The purpose of the experiments in this section is compare the performance the proposed cluster-based (composition) with the R packages "hawkesbow" [10] and "hawkes" [11]. These R functions contain some special features to simulate Hawkes processes. The

special features are as follows:

- The quantile function (the closed form of inverse cumulative intensity function):
The R package "hawkesbow" is based on the cluster-based representation of Hawkes process. It simulates the sub-processes of Hawkes process using the closed form of the inverse cumulative excitation function, the quantile function. Numeric inversion isn't required.
- The advantage of exponential decay function as excitation function which is stated in Section 3.4.2

The R package "hawkes" is the software which uses Ogata's modified thinning algorithm and works only with exponential decay excitation function to benefit the special feature of exponential decay function so that the quadratic complexity of Ogata's modified thinning algorithm decreases to linear complexity.

The time experiments of the R Package "hawkes", the R package "hawkesbow" and the presented algorithm are performed using the following input parameters which satisfies stationary conditions of Hawkes process:

- Base intensity, $\lambda_0 = 2$
- Excitation function, $\varphi(t) = 0.975 * 4 * e^{-4t}$
- Simulation duration $T = 10.000, 15.000, 20.000, 25.000, 30.000$
- Number of independent replications, cascades, $k = 1$

The output data of the experiments with the following variables are presented in Figure 6.1:

- system time
- number of events

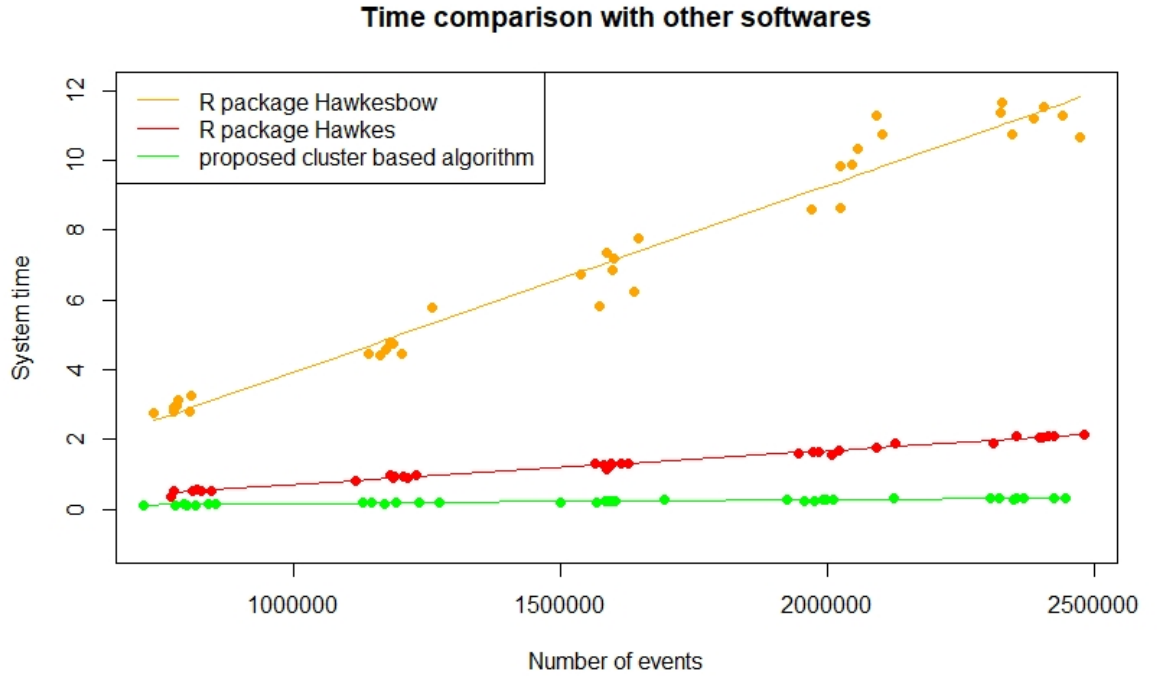


Figure 6.1. The time performance comparison with other software with parameter setting 3

The trend lines which are drawn in Figure 6.1 are as follows:

For the package "Hawkesbow", it is expressed as

$$Time = 5.339 * 10^{-06} \text{ events} - 1.396. \quad (6.1)$$

For the package "Hawkes", it is expressed as

$$Time = 9.706 * 10^{-07} \text{ events} - 2.594 * 10^{-01}. \quad (6.2)$$

For the proposed cluster-based algorithm, it is expressed as

$$Time = 1.119 * 10^{-07} \text{ events} + 5.319 * 10^{-02} \quad (6.3)$$

The trend line of the presented cluster-based algorithm has larger intercept compared to the other software. The reason for this result is that the presented cluster-based algorithm has more time-consuming operations (the computation of the inverse cumulative intensity function and the area under the base intensity function and the excitation function using numerical inversion) which are executed once at the set-up of the algorithm. The experiment results which are drawn in Figure 6.1 support that

- The presented algorithm is faster than Ogata's modified thinning algorithm even the excitation function is exponential function.
- The presented algorithm is faster than cluster-based simulation with sub-processes simulated with thinning method or using closed form of the quantile functions. The software in the literature uses the quantile functions of the programming languages.
- The simulation results are coherent with the number of operations which are required by the algorithms. The proposed algorithm which requires fewer number of operations generates the same number of events in a shorter time.

As a summary, the experiment results indicate that the implemented software based on the presented cluster-based Hawkes process algorithm is faster than the software for the special cases with

- the exponential decay function as excitation function and
- the closed form of the inverse cumulative excitation function.

6.2. The Time Comparison with Ogata's Modified Thinning Algorithm

In the next three sub-sections, the simulation duration versus total number of events generated by Ogata's modified thinning algorithm and by the proposed cluster-based Hawkes generation algorithm are compared. The experiments are performed with

- (i) unbounded simulation duration, $T = \infty$, the area under the intensity functions < 1 and multiple simulation runs (cascades) $n > 1$
- (ii) bounded simulation duration, $T < \infty$, the area under the intensity functions > 1 and single simulation run (cascade) $n = 1$

The Time Performance Comparison with Unbounded Simulation Duration: The experiments for the 1st case are performed with the following input parameters (Parameter setting 1):

- Base intensity, $\lambda_0(t) = 3.9 * e^{-4t}$
- Excitation function, $\varphi(t) = 3.9 * e^{-4t}$
- Simulation duration, $T = \text{infinity}$
- Number of independent replications, cascades,
k = 5,10,20,50,100,200,300,400,500,700,900 and 1.000

Note: An additional stopping condition is added to the experiment with Ogata's modified thinning algorithm otherwise Ogata's modified thinning cannot terminate for unbounded simulation duration. This stopping condition is a minimum value for the conditional intensity function. Experiment results of case 1 are presented in Figure 6.2:

The trend lines which are drawn in Figure 6.2 are as follows:
For Ogata's modified thinning algorithm, it is expressed as

$$Time = 4.175 * 10^{-09} \text{ events}^2 + 2.703 * 10^{-04} \text{ events}. \quad (6.4)$$

For the proposed cluster-based algorithm, it is expressed as

$$Time = 6.849 * 10^{-07} \text{ events}. \quad (6.5)$$

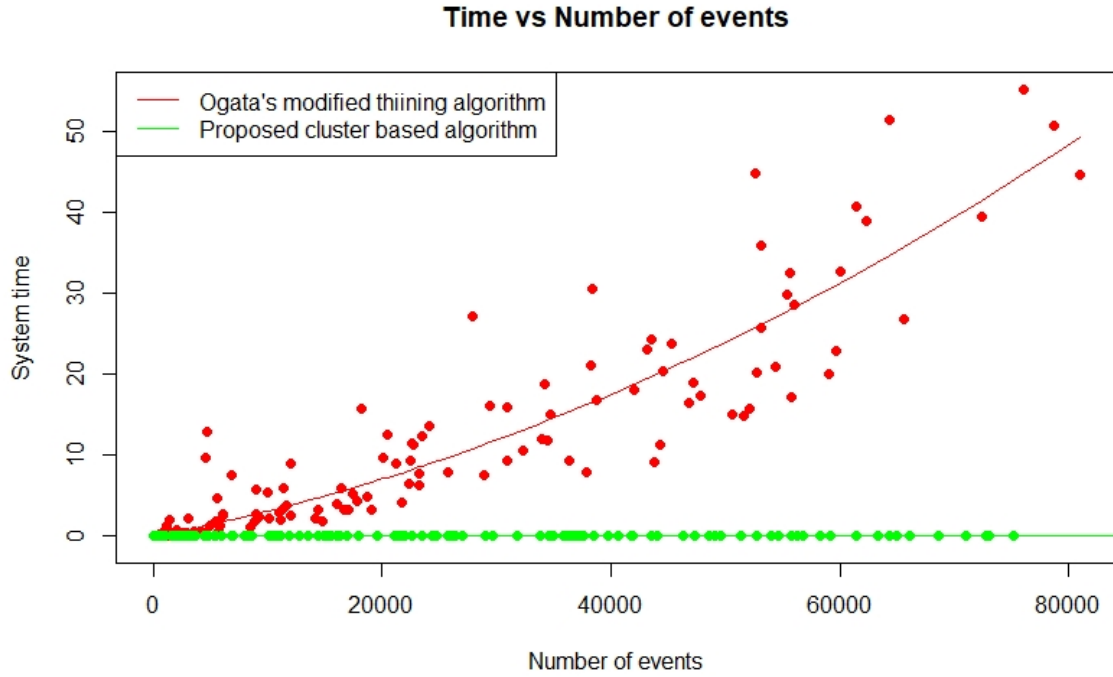


Figure 6.2. "Time required vs Number of the events" experiments for Case 1 with parameter setting 1

The trend line 6.4 of Ogata's modified thinning algorithm indicates that the time is increasing polynomial order two (quadratic) as the number of events increases whereas the trend line 6.5 of the proposed algorithm is linear.

The Time Performance Comparison with the Bounded Simulation Duration: The experiments for the 2nd case are performed with the following input parameters (Parameter setting 2):

- Base intensity, $\lambda_0(t) = 4.7 * e^{-4t}$
- Excitation function, $\varphi(t) = 4,7 * e^{-4t}$
- Simulation duration, $T = 10$
- Number of independent replications, cascades, $k = 1$

The experiment results of case 2 are presented in Figure 6.3:

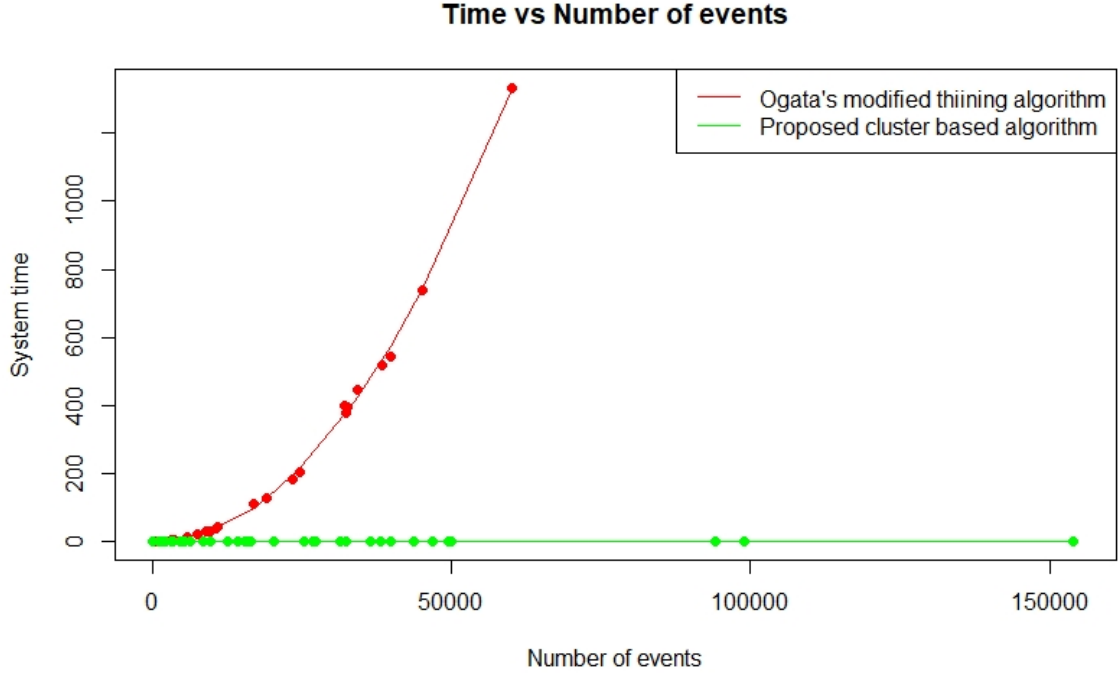


Figure 6.3. "Time required vs Number of the events" experiments for Case 2 with parameter setting 2

The trend lines which are drawn in Figure 6.3 are as follows:

For Ogata's modified thinning algorithm, it is expressed as

$$Time = 3.720 * 10^{-07} events^2 - 3.500 * 10^{-04} events. \quad (6.6)$$

For the proposed cluster-based algorithm, it is expressed as

$$Time = 3.419 * 10^{-07} events + 1.456 * 10^{-02}. \quad (6.7)$$

The experiment results indicate that there is a polynomial relationship with degree two (quadratic) between the variables the Hawkes process simulation time and the number of events generated (which is stated in Equation 6.6) if Ogata's modified thinning algorithm is used. However, there is a linear relationship between these variables

(which is stated in Equation 6.7) if the presented cluster-based algorithm is preferred. There is an important point which the experiment results indicate: When investigating the trend lines and the raw experiment data, for small number event generation Ogata's modified thinning algorithm performs better than the presented cluster-based algorithm because of the set-up of the cluster-based algorithm.

6.3. The Comparison of the Number of Operations Required

The number of operations which are generated by Ogata's modified thinning algorithm and by the presented cluster-based (composition) algorithms are investigated to find the cause of these results in Section 6.2. Parameter setting 2 is used in following experiments. Beside the number of events and the time required for the generation of the events, the following operations are counted in these experiments:

For Ogata's modified thinning algorithm

- exponential random variates
- uniform random numbers
- excitation function evaluations

For the presented cluster based algorithm

- Poisson random variates
- uniform random numbers
- inverse CDF (of intensity function) evaluations

While investigating the experiment outputs of Ogata's modified thinning algorithm, it is detected that the relationship between the following variables are linear which is coherent with theoretical studies:

- exponential random variates - event numbers
- uniform random numbers - event numbers

However the relationship between the variables

- the total number of excitation function evaluations at the thinning step for the all candidate events (i.e. the excitation function evaluations at the thinning step are counted for both accepted and rejected candidate events)
- total number of events

is not linear.

The output data of the experiments with these variables are presented in Figure 6.4:

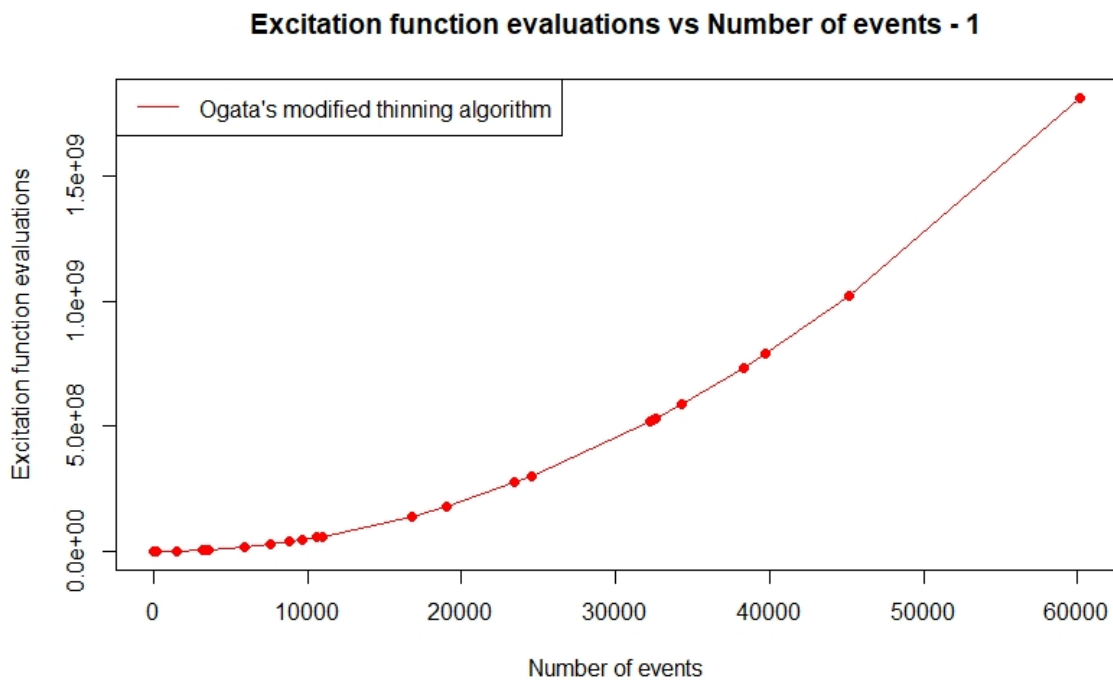


Figure 6.4. Ogata's modified thinning algorithm: Number of the excitation function evaluations vs Number of the events generated with parameter setting 2

The trend line which is drawn in Figure 6.4 it is expressed as

$$Excitation = 5.000 * 10^{-01} \text{events}^2 + 4.363 * 10^{+00} \text{events}. \quad (6.8)$$

The relationship between these variables is polynomial with degree two (quadratic) as stated in Equation 6.8. At the thinning step of the Ogata's modified thinning algorithm, there are some candidate events which are rejected. For the rejected candidate events, there are also excitation function evaluations. The number of excitation function evaluations for the events which are generated by Ogata's modified thinning algorithm (excluding rejected candidate events) are counted in the next step. The investigation is between the variables

- total number of excitation function evaluations at thinning step for the accepted candidate events (i.e. the excitation function evaluations at the thinning step are counted only for accepted candidate events)
- total number of events

The trend line which is drawn in Figure 6.5 is expressed as

$$Excitation = 5.000 * 10^{-01} \text{events}^2 - 5.000 * 10^{-01} \text{events}. \quad (6.9)$$

The relationship between these variables is polynomial with degree two as stated in Equation 6.9. This result is coherent with the theoretical studies in Equation 4.1. The next investigation is the relationship between the variables

- total number of excitation function evaluations at thinning step for the rejected candidate events (i.e. the excitation function evaluations at the thinning step are counted only for rejected candidate events)
- total number of events

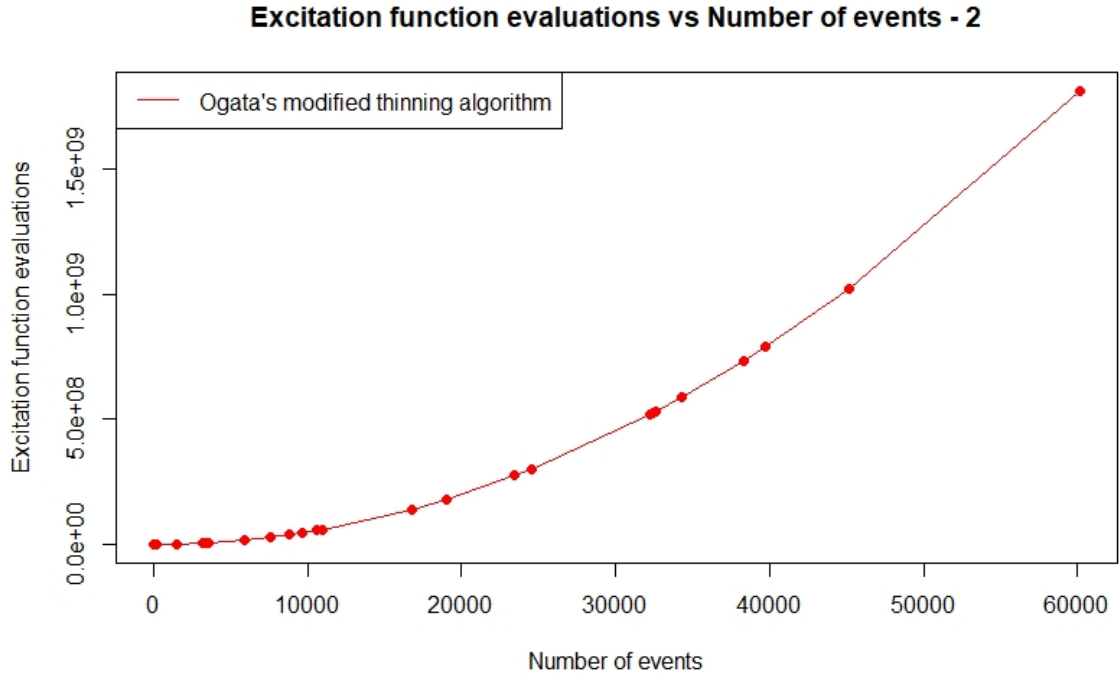


Figure 6.5. Ogata's modified thinning algorithm, Number of excitation function evaluations for accepted events vs Number of events generated with parameter setting

2

The trend line which is drawn in Figure 6.6 is it is expressed as

$$Excitation = 5.7690 \text{ events.} \quad (6.10)$$

While investigating the experiment output of the presented cluster-based algorithm, it is detected that the relationships between the variables below and the number of events are linear.

- Poisson random variates - number of events
- uniform random numbers - number of events
- number of inverse CDF evaluations - number of events

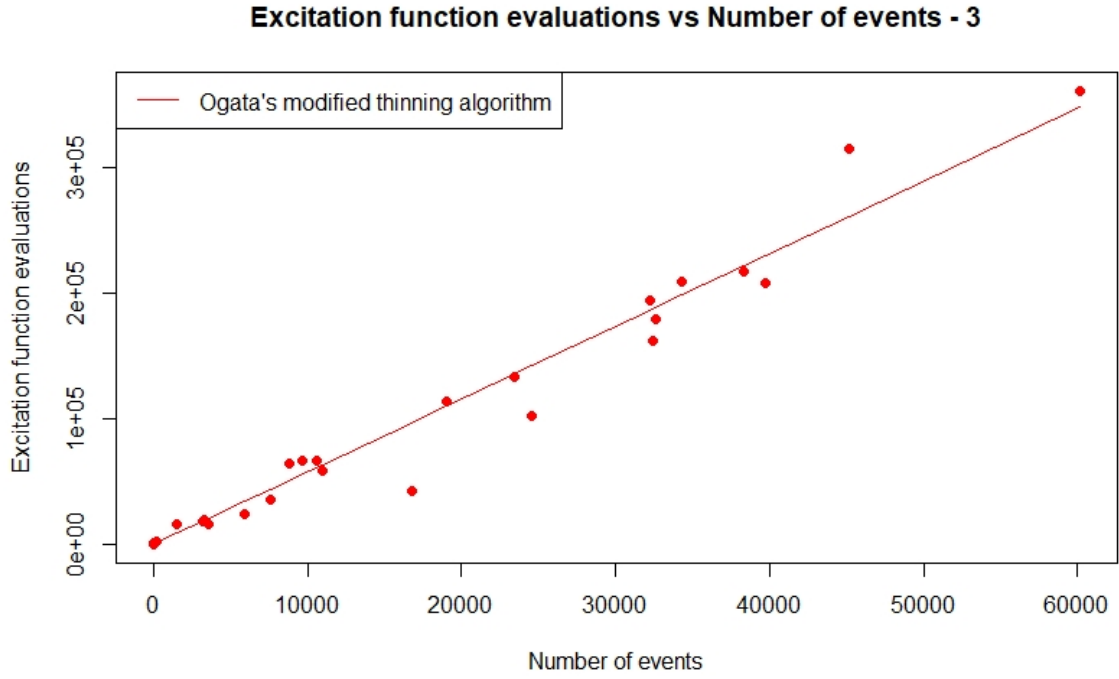


Figure 6.6. Ogata's modified thinning algorithm: Number of the excitation function evaluations for rejected events vs Number of the events with parameter setting 2

The output data of the experiments with the following variables are presented in Figure 6.7:

- Poisson random variates - number of events
- uniform random numbers - number of events

The trend lines which are drawn in Figure 6.7 are as follows:

For the Poisson random numbers vs the number of events, it is expressed as

$$\text{Poisson random numbers} = 1.000 * \text{events} + 1.000 * k. \quad (6.11)$$

For the uniform random numbers vs the number of events, it is expressed as

$$\text{Uniform random numbers} = 1.174292 * \text{events} \quad (6.12)$$

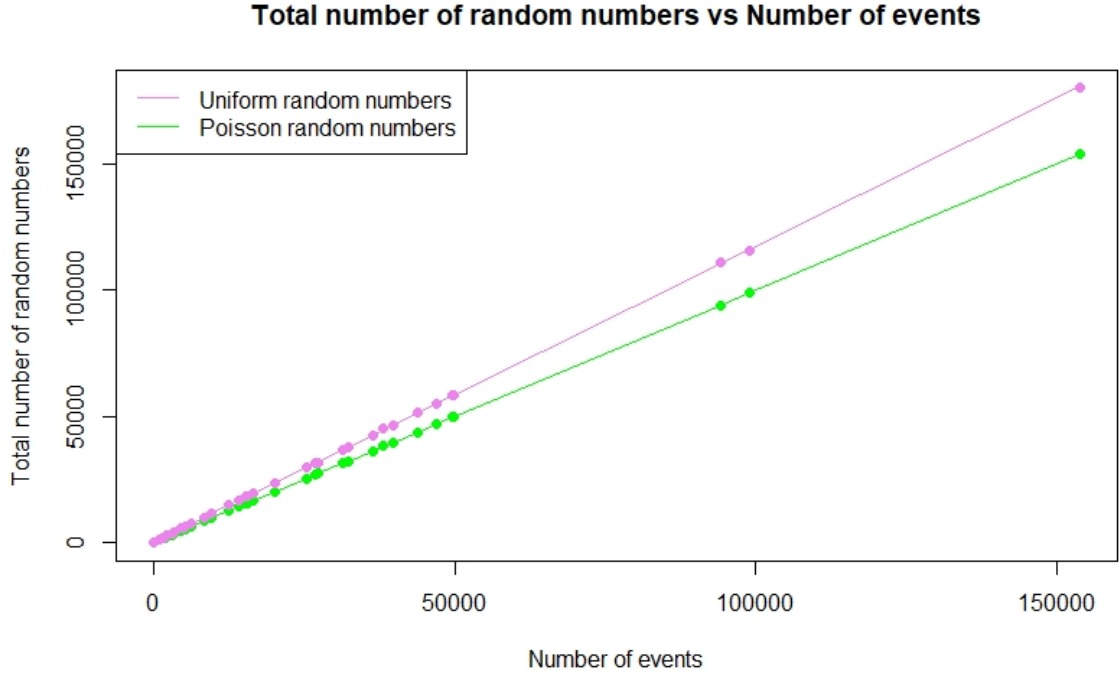


Figure 6.7. Cluster-based algorithm: Total Poisson and Uniform random numbers vs Number of the events with parameter setting 2

where " k " is the number of independent simulation replications in the parameter settings.

The trend line of the variables, the inverse cumulative intensity function evaluations and the number of events is same as the trend line of the variables uniform random numbers and the number of events. The relationship between these variables is linear. The difference between the trend lines is caused by the events that are generated beyond the simulation period, T . As $T \rightarrow \infty$, there will be no events beyond the simulation period. The trend line of the uniform random numbers is expressed as

$$\text{Uniform random numbers} = 1.000 * \text{events}. \quad (6.13)$$

The additional experiments with the unbounded simulation duration are performed in Section C.1.

The experiment results in this section indicate that the relationship between the variables, (number of operations required and number of events) is polynomial with order two if Ogata's modified thinning algorithm is used. However, it is linear if the presented cluster based algorithm is used to simulate Hawkes process.

6.4. The Time Performance of the Presented Algorithm

The purpose of this section is to compare the time performance of the presented algorithm with and without sorting the event times using the radix sort. The following time experiments are performed. The input parameters are as follows (Parameter setting 4):

- Base intensity, $\lambda_0(t) = 4.7 * e^{-4t}$
- Excitation function, $\varphi(t) = 4.7 * e^{-4t}$
- Simulation duration, $T = 20$
- Number of independent replications, cascades, $k = 1$

The experiment results of the presented cluster-based algorithm with sorting and without sorting are presented in Figure 6.8 with the following variables:

- system time
- number of events

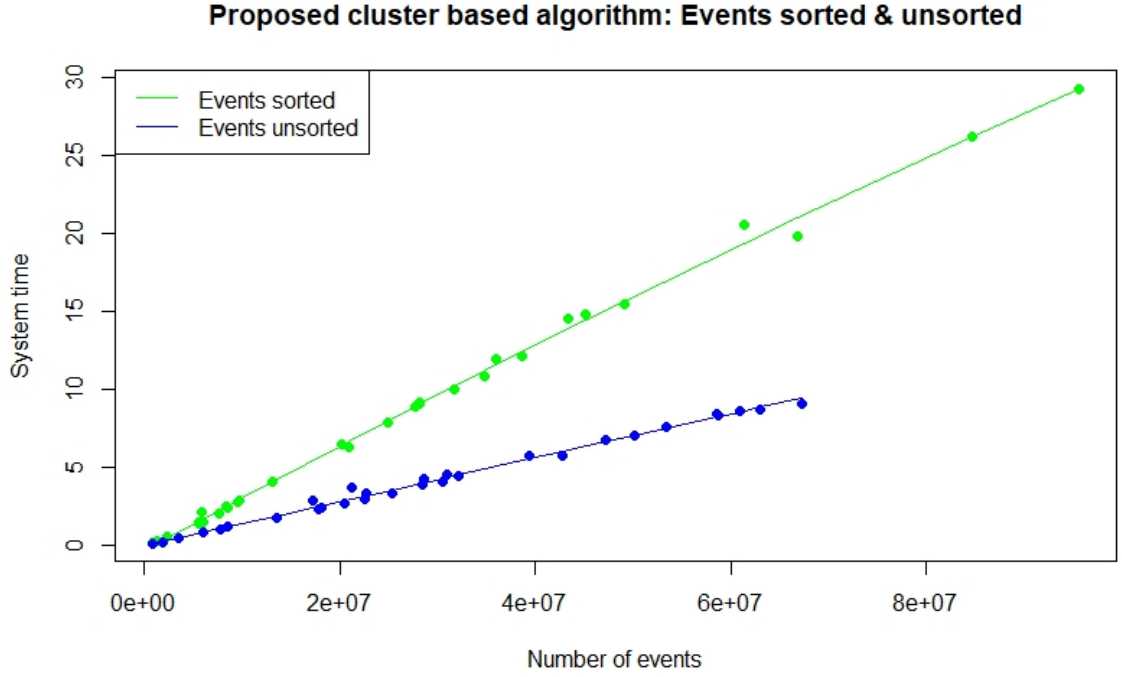


Figure 6.8. Cluster based algorithm: events sorted and unsorted with the parameter setting 4

The trend lines which are drawn in Figure 6.8 are as follows:

For the sorted case, it is expressed as

$$Time = -3.696 * 10^{-16} events^2 + 3.450 * 10^{-07} events - 3.344 * 10^{-01}. \quad (6.14)$$

For the unsorted case, it is expressed as

$$Time = 1.412 * 10^{-07} events \quad (6.15)$$

According to the experiment results, after a certain number of generated events, the time required to sort the event times of the Hawkes process takes more time than the time required to generate them. The additional performance experiments are in Appendix C.

7. CONCLUSION

The required operations to generate Hawkes processes using two well known algorithms are investigated. These algorithms are intensity based Ogata's modified thinning algorithm and cluster based (composition) algorithm. The investigation contains the determination of the operations and the enumeration of these operations. This investigation is the first research on the details of the algorithms to simulate Hawkes processes that assesses the number of the operations and the speed performances. The detailed analysis of the required number of operations for the cluster based (composition) algorithm resulted in the development of a new clearly faster version of the cluster based algorithm. It is faster than the intensity based Ogata's modified thinning algorithm and its special case with exponential decay excitation function. The detailed (step by step) pseudo of the cluster-based (composition) algorithm for Hawkes process generation is described.

The new resulted cluster-based (composition) algorithm combines the following concepts: decomposition, inversion method, order statistics and time re-scaling theorem to simulate Hawkes process.

The linear analytic structure of the Hawkes process' conditional intensity function provides the opportunity to decompose the stochastic conditional intensity function of Hawkes process into the deterministic base intensity function of the immigrant sub-process and the deterministic excitation function of the offspring sub-processes. The advantage is the linear increase in the number of operations which are required by the resulted cluster-based algorithm to simulate Hawkes process. This is especially useful for the large number of event generations.

The second advantage of the decomposition is that the composition algorithm for Hawkes process generation is suitable for the general smooth intensity (base intensity and excitation) functions which are not restricted as monotone functions by Ogata's

modified thinning algorithm if no events occur.

The next advantage of the decomposition is that the composition algorithm can easily classify the events as immigrant and offspring with indicating the ancestors and descendants of each event of the process i.e. branching structure of the process. In the literature [44, 48], it stated that Ogata's modified thinning algorithm cannot classify the events as immigrant and offspring. However, Ogata's modified thinning algorithm can classify the events as immigrant and offspring with indicating the ancestors and descendants of each event of the process in a complicating and operationally time-consuming way using the conditional intensity function compared to the proposed cluster-based (composition) algorithm.

The advantage of the inversion method over the thinning method is that it can simulate the sub-processes both on the bounded and unbounded simulation intervals so that the full event cascade of the Hawkes process can be generated if the area under the base intensity function is bounded, and the branching factor (area under the excitation function) is less than one. The intensity-based Ogata's modified thinning algorithm for Hawkes process and the thinning method for non-homogeneous Poisson process work only on the bounded time interval. They cannot generate the full event cascade.

Order statistic doesn't require comparison operations as stopping conditions of the non-homogeneous Poisson process so that the number of operations are reduced. It uses the fact that the number of events generated by a (non-homogeneous) Poisson process is Poisson distributed with mean equals to the area under the intensity function of the (non-homogeneous) Poisson process.

For the performance assessment of the presented cluster-based (composition) algorithm, the theoretical and empirical comparison studies are performed with the other algorithms in the literature. In the theoretical studies, the following main operations are counted: the random variates, the intensity function evaluations and the inverse cumulative intensity function evaluations. In addition to them, the comparison and

sorting operations requirements are identified and counted for the algorithms. According to the theoretical studies, the implemented algorithm requires smaller number of operations compared to the other algorithms. In the empirical studies, time experiments are performed. The time spent and the number of generated events are compared with different algorithms and software in the literature. The empirical studies justify the theoretical results. The same number of Hawkes process events are generated in a shorter time by the proposed algorithm compared to the other algorithms because it requires a smaller number of operations (random variates). A software for the new algorithm is also implemented which is used in the empirical studies. This software can also simulate the Hawkes process extensions, the marked Hawkes process and the renewal Hawkes process.

Furthermore, a fast implementation of NHPP generation for the Hawkes process simulation with the cluster-based (composition) algorithm is required which is investigated for the first time. The resulted NHPP method combines inversion method, order statistics and time re-scaling theorem.

REFERENCES

1. Hawkes, A., “Spectra of Some Self-exciting and Mutually Exciting Point Processes”, *Biometrika*, Vol. 58, pp. 83–90, 1971.
2. Rizoiu, M., Y. Lee, S. Mishra and L. Xie, “A Tutorial on Hawkes Processes for Events in Social Media”, S.-F. Chang (Editor), *Frontiers of Multimedia Research*, pp. 107–132, ACM, 2017.
3. Ogata, Y., “On Lewis’ Simulation Method for Point Processes”, *IEEE Transactions on Information Theory*, Vol. 27, pp. 23–31, 1981.
4. Lewis, P. and G. Shedler, “Simulation of Nonhomogeneous Poisson Processes by Thinning”, *Naval Research Logistics Quartal*, Vol. 26(1), pp. 403–413, 1979.
5. Møller, J. and J. G. Rasmussen, “Perfect Simulation of Hawkes Processes”, *Advances in Applied Probability*, Vol. 37(3), p. 629, 2005.
6. Møller, J. and J. G. Rasmussen, “Approximate Simulation of Hawkes Processes”, *Methodology and Computing in Applied Probability*, Vol. 8, pp. 53–64, 2006.
7. Hawkes, A. G. and D. Oakes, “A Cluster Process Representation of A Self-Exciting Process”, *Journal of applied probability*, Vol. 11, No. 3, pp. 493–503, 1974.
8. Chen, F., *IHSEP: Inhomogeneous Self-Exciting Process*, 2022, <https://CRAN.R-project.org/package=IHSEP>, r package 0.3.1, accessed on May 8, 2023.
9. Chen, F. and T. Stindl, *RHawkes: Renewal Hawkes Process*, 2022, <https://cran.r-project.org/package=RHawkes>, r package version 1.0, accessed on May 8, 2023.

10. Cheysson, F., *hawkesbow: Estimation of Hawkes Processes from Binned Observations*, 2021, <https://CRAN.R-project.org/package=hawkesbow>, r package version 1.0.2, accessed on April 4, 2023.
11. Zaatour, R., *hawkes: Hawkes Process Simulation and Calibration Toolkit*, 2014, <https://CRAN.R-project.org/package=hawkes>, r package version 0.0-4, accessed on April 4, 2023.
12. Lee, K. and B. K. Seo, “Marked Hawkes Process Modeling of Price Dynamics and Volatility estimation”, *Journal of Empirical Finance*, Vol. 40, pp. 174–200, 2017.
13. Wheatley, S., V. Filimonov and D. Sornette, “The Hawkes Process with Renewal Immigration & its Estimation with an EM Algorithm”, *Computational Statistics & Data Analysis*, Vol. 94, pp. 120–135, 2016.
14. Ross, S., *Introduction to Probability Models, Tenth Edition*, Academic Press, Inc., San Diego, CA, 2011.
15. Grimmett, G. R. and D. R. Stirzaker, *Probability and Random Processes 3rd Edition*, Oxford University Press Inc, New York, NY, 2001.
16. Daley, D. and D. Vere-Jones, *An Introduction to the Theory of Point Processes Volume I: Elementary Theory and Methods Second Edition*, Springer-Verlag, New York, NY, 2003.
17. Ross, S. and E. Pekoz, *A Second Course in Probability, Tenth Edition*, www.ProbabilityBookstore.com, Boston, MA, 2007.
18. Bertsekas, D. and J. Tsitsiklis, *Introduction to Probability, Second Edition*, Athena Scientific, Belmont, MA, 2011.
19. Resnick, S., *Adventures in Stochastic Processes, 4th Printing*, Academic Press, Inc., Boston, MA, 2005.

20. Prins, J., *e-Handbook of Statistical Methods*, NIST/SEMATECH, 2012, <https://doi.org/10.18434/M32189>, accessed on June 10, 2024.
21. Cinlar, E., *Introduction to Stochastic Processes*, Prentice-Hall, New Jersey, 1975.
22. Cormen, T. H., C. E. Leiserson, R. L. Rivest and C. Stein, *Introduction to Algorithms Third Edition*, The MIT Press, Cambridge, Massachusetts London, England, 2009.
23. Brown, E. N., R. Barbieri, V. Ventura, R. E. Kass and L. M. Frank, “The Time-Rescaling Theorem and Its Application to Neural Spike Train Data Analysis”, *Neural Computation*, Vol. 14, No. 2, pp. 325–346, 02 2002.
24. Laub, P. J., Y. Lee and T. Taimre, *The Elements of Hawkes Processes*, Springer, Switzerland, 2021.
25. Cox, D. R. and P. A. Lewis, *The Statistical Analysis of Series of Events*, Springer, Methuen, London, United Kingdom, 1966.
26. Howard M. Taylor, S. K., *An Introduction to Stochastic Modeling - Third edition*, Academic Press Limited, London, 2007.
27. Cox, D. R. and V. Isham, *Point Processes*, Vol. 12, CRC Press, London, 1980.
28. Clinet, S. and Y. Potiron, “Statistical Inference for the Doubly Stochastic Self-exciting Process”, *Bernoulli*, Vol. 24, No. 4B, pp. 3469 – 3493, 2018, <https://doi.org/10.3150/17-BEJ966>.
29. Tjøstheim, D., “Some Doubly Stochastic Time Series Models”, *Journal of Time Series Analysis*, Vol. 7, No. 1, pp. 51–72, 1986.
30. Stewart, J., *Calculus: Early Transcendentals, Eighth Edition*, Cengage Learning, Boston, MA, 2010.

31. Brix, A. and W. S. Kendall, “Simulation of Cluster Point Processes Without Edge Effects”, *Advances in Applied Probability*, Vol. 34(2), pp. 267–280, 2002.
32. Mikosch, T., *Non-Life Insurance Mathematics Second Edition*, Springer-Verlag, New York, NY, 2009.
33. Allen, L., *Branching Processes in Biology*, Springer-Verlag, New York, NY, 2002.
34. Kendall, D. G., “Stochastic Processes and Population Growth”, *Journal of the Royal Statistical Society Series B (Methodological)*, Vol. 11, pp. 230–282, 1949.
35. Devroye, L., *Non-uniform Random Variate Generation*, Springer, New York, NY, 1986.
36. Harte, D. S., “PtProcess: An R Package for Modelling Marked Point Processes Indexed by Time”, *Journal of Statistical Software*, Vol. 35, No. 8, pp. 1–32, 2010, <https://cran.r-project.org/package=PtProcess>, accessed on June 10, 2024.
37. Casella, G. and R. Berger, *Statistical Inference*, CRC Press, Duxbury, 2024.
38. Papoulis, A. and S. U. Pillai, *Probability, Random Variables, and Stochastic Processes*, McGraw-Hill, New York, NY, 2002.
39. Hörmann, W., J. Leydold and G. Derflinger, *Automatic Nonuniform Random Variate Generation*, Springer, New York, NY, 2004.
40. Ross, S., *Simulation, Tenth Edition*, Elsevier, Inc., San Diego, CA, 2011.
41. Neumann, J., “Various Techniques Used in Connection with Random Digits”, *Nat. Bur. Standards Appl. Math. Ser.*, Vol. 12, pp. 36—38, 1951.
42. Pasupathy, R., “Generating Nonhomogeneous Poisson Processes”, *Wiley Encyclopedia of Operations Research and Management Science*, 2010.

43. Pasupathy, R., “Generating Homogeneous Poisson Processes”, *Wiley Encyclopedia of Operations Research and Management Science*, 2010.
44. Jiang, Y. and M. D. Porter, “Simulating Fake News Dissemination on Twitter with Multivariate Hawkes Processes”, *2022 IEEE International Conference on Big Data (Big Data)*, pp. 3597–3606, Osaka, Japan, 2022.
45. Laub, P. J., T. Taimre and P. K. Pollett, “Hawkes Processes”, *arXiv preprint arXiv:1507.02822*, 2015.
46. Bacry, E., S. Delattre, M. Hoffmann and J.-F. Muzy, “Modelling Microstructure Noise with Mutually Exciting Point Processes”, *Quantitative finance*, Vol. 13, No. 1, pp. 65–77, 2013.
47. Dassios, A. and H. Zhao, “Exact Simulation of Hawkes Process with Exponentially Decaying Intensity”, *Electronic Communications in Probability*, Vol. 18(62), pp. 1–13, 2013.
48. Simon, G., “Hawkes Processes in Finance: A Review with Simulations”, *University of Oregon*, 2016.
49. Leydold, J. and W. Hörmann, *Runuran: R Interface to the 'UNU.RAN' Random Variate Generators*, 2023, <https://CRAN.R-project.org/package=Runuran>, r package version 0.38, accessed on January 18, 2023.
50. Horsmalahiti, P., “Comparison of Bucket Sort and Radix Sort”, *arXiv preprint arXiv:1206.3511*, 2012.
51. Mersmann, O., *microbenchmark: Accurate Timing Functions*, 2023, <https://CRAN.R-project.org/package=microbenchmark>, r package version 1.4.10, accessed on April 18, 2023.
52. R Core Team, *R: A Language and Environment for Statistical Com-*

- puting*, R Foundation for Statistical Computing, Vienna, Austria, 2022, <https://www.R-project.org/>.
53. Lee, K., *emhawkes: Exponential Multivariate Hawkes Model*, 2023, <https://CRAN.R-project.org/package=emhawkes>, r package version 0.9.7, accessed on April 4, 2023.
 54. Da Fonseca, J. and R. Zaatour, “Clustering and Mean Reversion in a Hawkes Microstructure Model”, *Journal of Futures Markets*, Vol. 35, No. 9, pp. 813–838, 2015.
 55. Stindl, T. and F. Chen, *MRHawkes: Multivariate Renewal Hawkes Process*, 2018, <https://CRAN.R-project.org/package=MRHawkes>, r package version 1.0, accessed on May 8, 2023.
 56. Stindl, T. and F. Chen, “Likelihood Based Inference for the Multivariate Renewal Hawkes Process”, *Computational Statistics & Data Analysis*, Vol. 123, pp. 131–145, 2018.
 57. Chen, F. and T. Stindl, “Direct Likelihood Evaluation for The Renewal Hawkes Process”, *Journal of Computational and Graphical Statistics*, Vol. 27, No. 1, pp. 119–131, 2018.
 58. Meyer, S., L. Held and M. Höhle, “Spatio-temporal analysis of epidemic phenomena using the R package surveillance”, *arXiv preprint arXiv:1411.0416*, 2014.
 59. Burden, R. and J. Faires, *Numerical Analysis, Ninth Edition*, Brooks/Cole, Cengage Learning, Boston, MA, 2011.
 60. O Searcóod, M., *Metric Spaces*, Springer Science & Business Media, London, 2007.
 61. Chapra, S. and R. Canale, *Numerical Methods for Engineers, Seventh Edition*, McGraw-Hill Education, New York, NY, 2015.

62. Derflinger, G., W. Hörmann and J. Leydold, “Random Variate Generation by Numerical Inversion When Only the Density is Known”, *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, Vol. 20, No. 4, pp. 1–25, 2010.



APPENDIX A: OGATA'S MODIFIED THINNING ALGORITHM

The Hawkes process has a stochastic conditional intensity function $\lambda(t|H_t)$. This means the intensity function value at a certain time " t " depends on the events in the process history H_t that occurred before the time " t ". Such a process with conditional intensity function $\lambda(t|H_t)$ can be simulated using the thinning method by Lewis and Shedler [4] if there exist a finite bound " M " such that [16]:

- $\lambda(t|H_t) \leq M$ for all possible past histories and
- the process is simulated over a finite interval $(0, T)$ given some initial history H_0

The thinning algorithm works fine in its original context with deterministic intensity function because it is possible to find a finite bound " M " over the simulation interval $(0, T)$ for all possible past histories. However, its main difficulty is to find a finite bound " M " over the simulation interval $(0, T)$ for all possible past histories if the process to be simulated has random (stochastic) conditional intensity function. It is difficult or even not possible.

Ogata [3] proposed an algorithm to overcome this difficulty which requires only local boundedness condition on the conditional intensity function. Later it is called "Ogata's Modified Thinning Algorithm" in the literature.

Two quantities are required for every t in $(0, T)$:

- A local bound $M(t|H_t)$
- A time interval of length $L(t|H_t)$

The intensity functions satisfy the following conditions which are stated in Equation A.1 and in Equation A.2 for every $n = 1, 2, \dots$, and all sequences $t_1, \dots, t_{n-1}$ with $0 <$

$t_1 < \dots < t_{n-1} < t$: [16]

$$\lambda_n(t + u | t_1, \dots, t_{n-1}) \leq M(t | H_t) \quad (\text{A.1})$$

$$0 \leq u < L(t | H_t) \quad (\text{A.2})$$

The bound $M(t | H_t)$ is valid until the end of the time step $L(t | H_t)$ if no new event occurs. It must be updated either a new event occurs or the time step $L(t | H_t)$ elapsed.

The steps of Ogata's modified thinning algorithm are as follows [16]:

- (i) Set $t = 0$, $i = 0$.
- (ii) Stop if the termination condition is met; otherwise, compute $M(t | H_t)$ and $L(t | H_t)$.
- (iii) Generate an exponential random variate, E with mean $1/M(t | H_t)$ and a random variable, U uniformly distributed on $(0, 1)$.
- (iv) If $E > L(t | H_t)$, set $t = t + L(t | H_t)$ and return to step (ii).
- (v) If $E \leq L(t | H_t)$ and $\frac{\lambda(t+E | H_t)}{M(t | H_t)} > U$, replace t by $t + E$ and return to step (ii).
- (vi) Otherwise,
 - (a) advance $i \leftarrow i + 1$,
 - (b) set $t_i = t + E$,
 - (c) replace t by t_i ,
 - (d) update H to $H \cup t_i$, and
 - (e) return to step (ii)
- (vii) The output is the list $\{i; t_1, \dots, t_i\}$

Ogata [3] gives extended discussion and variants of the algorithm.

The computational difficulties of computing the local bound $M(t | H_t)$ and the time period $L(t | H_t)$ for which $M(t | H_t)$ is valid (in other words greater than or equal to conditional intensity function) depends on the base intensity function $\lambda_0(t)$ and excitation

function $\varphi(t)$ of Hawkes process.

If the conditional intensity function (both base intensity function and excitation function) is monotonically not increasing, it is easy to compute the local bound $M(t|H_t)$ and the time interval $L(t|H_t)$. $M(t_i|H_t) = \lambda(t_i|H_t)$ and $L(t_i|H_t)$ is equal to remaining simulation period " (t_i, T) " where t_i is last event in process history H_t . This can lead to an inefficient algorithm if the conditional intensity function decreases very rapidly because the algorithm generates a lot of rejected events [16].



APPENDIX B: THE R CODE OF THE SOFTWARE

The R code of the software and the pseudo code of the algorithm are presented.

B.1. The R Code of the Software

The R code of the package "fastHawkes" is given below.

```
#' @title Efficient and fast Hawkes process generation algorithm
#
#' @description
#
#' Efficient and fast Hawkes process generation algorithm based on
cluster-based representation of Hawkes process
#
#' @details
#
#' This function provides to the user two alternatives to simulate
Hawkes process:
#
#' 1- Efficient and fast cluster-based simulation algorithm
#
#' Its advantage is that any smooth, general and non-negative function
can be set as base intensity and excitation functions.
#
#' It is based on cluster-based representation of Hawkes process.
Hawkes process is decomposed into immigrant and offspring processes.
#
#' The subprocesses (immigrant and offspring) that are independent
non-homogeneous Poisson processes
```

#' are generated by an algorithm that includes (numerical) inversion, order statistics and transformation (time re-scaling).

#'

#' The composition of these immigrant and offspring processes constitute the Hawkes process.

#'

#' 2- Ogata's modified thinning algorithm (for non-increasing monotone base intensity and excitation functions)

#'

#' Its disadvantage is that the excitation function must be a monotone function.

#'

#' The advantage of the cluster-based simulation algorithm is that it is faster than the Ogata's modified thinning algorithm.

#'

#' The cluster-based algorithm is efficient for the generation of Hawkes process for full cascades.

#'

#'

#'

#' The inputs are

#'

#' n: The number of independent non-empty Hawkes simulation runs, non-empty cascades

#'

#' T: Simulation duration, the end point of the simulation period

#'

#' allimmigrants : The user can supply the immigrant events instead of simulating them

#'

#' repr : Mean number of offspring (A non-negative numeric value)

```

#'
#' marks : Weight of the event
#'
#' lam0 : The base intensity function of Hawkes process
#'
#' The area under the base intensity function is equal to the
#' expected number of events generated by immigrant process
#'
#' phi : The excitation function of Hawkes process
#'
#' The area under the excitation function is equal to the expected
#' number of events generated by offspring process
#'
#'
#' method=c("cluster","thinning")
#'
#' - "cluster" Cluster-based simulation algorithm for Hawkes process
#'
#' - "thinning" Ogata's modified thinning algorithm
#'
#' sortevents=c(TRUE,FALSE) gives the user the choice to sort
#' the event times or not
#'
#' Only for the cluster-based simulation algorithm for Hawkes process,
#' the event times needs sorting.
#'
#'
#'
#' The output of the function for cluster-based simulation algorithm:
#'
#'

```

```

#' 1- Accepted event times for all n independent replications
(non empty simulation runs,cascades)
#' 2- Replication no - the replication (cascade) to which the events
belongs
#' 3- Rejected Event times
#' 4- Total number of accepted event times
#' 5- Total number of rejected event times
#' 6- Total number of Poisson random numbers
#' 7- Total number of uniform random numbers
#' 8- Total number of inverse cumulative intensity function evaluations
#'
#'
#' The output of the function for Ogata's modified thinning algorithm:
#'
#' 1- Accepted event times for all n independent replications
#' 2- Replication no - the replication (cascade) to which the events
belong
#' 3- Total number of accepted event times
#' 4- Total number of rejected event times
#' 5- Total number of exponential random numbers
#' 6- Total number of uniform random numbers
#' 7- Total number of excitation function evaluations
#' 8- Total number of excitation function evaluations at accepted
events
#'
#'
#'
#' @param n The number of non-empty simulation runs of Hawkes process
(Default value, n=1)
#' @param T Simulation duration (Default value, T=Inf)
#' @param allimmigrants The immigrant events which are supplied by

```

```

the user (Default value, allimmigrants=c(0))
#' @param repr Mean number of offspring, a non-negative numeric value
(Default value, repr=NULL)
#' @param marks Weight of the event
(Default value, marks=function(x) {1})
#' @param lam0 The base intensity function of Hawkes Process
(Default value, lam0=NULL)
#' @param phi The excitation function of Hawkes Process
(Default value, phi=NULL)
#' @param method method=c("cluster","thinning")cluster =
Cluster-based Hawkes process simulation algorithm, thinning =
Ogata's modified thinning algorithm
#' @param sortevents sortevents=c(TRUE,FALSE) event times of
Hawkes process simulation are sorted
#'
#' @importFrom Runuran "pinv.new"
#' @importFrom Runuran "uq"
#' @importFrom Runuran "unuran.details"
#' @importFrom Runuran "ur"
#' @importFrom stats "rpois"
#' @importFrom stats "ppois"
#' @importFrom stats "qpois"
#' @importFrom stats "runif"
#' @importFrom stats "rexp"
#'
#' @aliases fastHawkes
#'
#' @references
#'
#' 1. M.A.Purazer, W.Hoermann
#'

```

```

#' 2. P. J. Laub, Y. Lee, T. Taimre, The Elements of Hawkes Processes,
Springer Nature Switzerland AG 2021,ISBN 978-3-030-84638-1
#'
#' 3. Y. Ogata, On Lewis' simulation method for point processes.
IEEE Transactions on Information Theory, 27(1): 23-31. 1981.
#'
#' @return The event times of n independent replications
#' @export rHawkes
#' @examples
#'
#' The function "rHawkes" can simulate both (marked) Hawkes process
and (non)-homogeneous Poisson process
#'
#' # DEFAULT CASE: Generating a whole social media cascade until it
fades out
#' # with an initial event at time 0 and simulation duration
T=Inf (infinity).
#' # Each event in the cascade has different weight to generate
further offspring
#'
#' # Marked Hawkes process simulation with an initial event at time 0,
(random) marks and
#' # and with repr (reproduction number - mean number of offspring)
#' # -----
#' # Hawkes process simulation with n=5,
T=Inf,allimmigrants=c(0),repr=0.975,marks = function(x) {rexp(x,1)},
lam0=NULL, phi=function(x) {2000*(x+250)^(-2.33)}, method="cluster"
and sortevents=TRUE
#' rHawkes(n=500,repr=0.975,marks = function(x) {rexp(x,1)},
phi=function(x) {2000*(x+250)^(-2.33)},method="cluster",sortevents=TRUE)
#'

```

```

#'
#'
#' # ADDITIONAL CASES:
#' # Marked Hawkes process simulation with with (random) marks
#' # -----
#' # with inputs: n=2, T=Inf,allimmigrants=NULL, marks = function(x)
#' # {rexp(x,1)}, lam0=function(x) {3.9*exp(-4*x)}, phi=function(x)
#' # {3.9*exp(-4*x)}, method="cluster" and sortevents=TRUE
#' rHawkes(n=2,allimmigrants=NULL, marks = function(x) {rexp(x,1)},
#' # lam0=function(x) {3.9*exp(-4*x)}, phi=function(x) {3.9*exp(-4*x)},
#' # method="cluster",sortevents=TRUE)
#'
#' # with inputs: n=2, T=100,allimmigrants=NULL, marks = function(x)
#' # {rexp(x,1)}, lam0=1, phi=function(x) {3.9*exp(-4*x)}, method="cluster"
#' # and sortevents=TRUE
#' rHawkes(n=2,T=100,allimmigrants=NULL,marks = function(x) {rexp(x,1)},
#' # lam0=1, phi=function(x) {3.9*exp(-4*x)},method="cluster",
#' # sortevents=TRUE)
#'
#' # with inputs: n=2, T=Inf,allimmigrants=NULL, marks = function(x)
#' # {0.9}, lam0=function(x) {2000*(x+250)^(-2.33)}, phi=function(x)
#' # {2000*(x+250)^(-2.33)}}, method="cluster" and sortevents=TRUE
#' rHawkes(n=2,allimmigrants=NULL,marks = function(x) {0.9},
#' # lam0=function(x) {2000*(x+250)^(-2.33)},
#' # phi=function(x) {2000*(x+250)^(-2.33)},method="cluster",
#' # sortevents=TRUE)
#'
#'
#' # Marked Hawkes process simulation with with (random) marks
#' # and with repr (reproduction number - mean number of offspring)
#' # -----

```

```

#' # with inputs: n=5, T=Inf,allimmigrants=NULL,repr=0.975,
marks =function(x) {rexp(x,1)},lam0=function(x) {2000*(x+250)^(-2.33)},
phi=function(x) {2000*(x+250)^(-2.33)}, method="cluster" and
sortevents=TRUE

#' rHawkes(n=5,allimmigrants=NULL,repr=0.975,
marks = function(x) {rexp(x,1)},
lam0=function(x) {2000*(x+250)^(-2.33)},
phi=function(x){2000*(x+250)^(-2.33)},method="cluster",sortevents=TRUE)
#'

#' # with inputs: n=5, T=1000,allimmigrants=NULL,repr=0.9,
marks = function(x) {rexp(x,1)},lam0=function(x) {0.5},
phi=function(x) {2000*(x+250)^(-2.33)},
method="cluster" and sortevents=TRUE
#' rHawkes(n=5,T=1000,allimmigrants=NULL,repr=0.9,
marks = function(x) {rexp(x,1)},
lam0=function(x) {0.5},
phi=function(x) {2000*(x+250)^(-2.33)},
method="cluster",sortevents=TRUE)
#'

#' # with inputs: n=5, T=Inf,allimmigrants=NULL,repr=0.8,marks =
function(x) {rexp(x,1)},lam0=function(x) {3.9*exp(-4*x)},
phi=function(x) {3.9*exp(-4*x)},method="cluster" and sortevents=TRUE
#' rHawkes(n=5,allimmigrants=NULL,repr=0.8,
marks = function(x) {rexp(x,1)}, lam0=function(x) {3.9*exp(-4*x)},
  phi=function(x) {3.9*exp(-4*x)},method="cluster",sortevents=TRUE)
#'
#'
#' # Marked Hawkes process simulation with with (random) marks and
with repr (reproduction number - mean number of offspring)
#' # and the immigrant event times are supplied by the user as input
#' # -----

```

```

#' # with inputs: n=5,T=Inf,allimmigrants=c(0,1,2,5),repr=0.8,
marks = function(x) {rexp(x,1)},lam0=NULL,
phi=function(x) {3.9*exp(-4*x)}, method="cluster" and sortevents=TRUE
#' rHawkes(n=5,allimmigrants=c(0,1,2,5),repr=0.8,
marks = function(x) {rexp(x,1)},
phi=function(x) {3.9*exp(-4*x)},method="cluster",sortevents=TRUE)
#'

#' # with inputs: n=5,T=Inf,allimmigrants=c(0,1,2,5),repr=0.8,
marks = function(x) {0.9},lam0=NULL,
phi=function(x) {2000*(x+250)^(-2.33)},
method="cluster" and sortevents=TRUE
#' rHawkes(n=5,allimmigrants=c(0,1,2,5),repr=0.8,
marks = function(x) {0.9},
phi=function(x) {2000*(x+250)^(-2.33)},method="cluster",
sortevents=TRUE)
#'

#'

#' # Hawkes process simulation with repr (reproduction number -
mean number of offspring)
#' # -----
#' # with inputs: n=2, T=Inf,allimmigrants=NULL, repr=0.975, lam0=
function(x) {2000*(x+250)^(-2.33)}, phi=function(x)
{2000*(x+250)^(-2.33)}, method="cluster" and sortevents=TRUE
#' rHawkes(n=2,allimmigrants=NULL,repr=0.975,
lam0=function(x) {2000*(x+250)^(-2.33)},
phi=function(x) {2000*(x+250)^(-2.33)}
,method="cluster",sortevents=TRUE)
#'

#' # with inputs: n=2, T=100,allimmigrants=NULL, repr=0.9,
lam0=3, phi=function(x) {2000*(x+250)^(-2.33)},
method="cluster" and sortevents=TRUE

```

```

#' rHawkes(n=2,T=100,allimmigrants=NULL,repr=0.9,
lam0=3,phi=function(x) {2000*(x+250)^(-2.33)},
method="cluster",sortevents=TRUE)
#'

#'# with inputs: n=3, T=Inf,allimmigrants=NULL,repr=0.8,
lam0=function(x) {3.9*exp(-4*x)},
phi=function(x) {3.9*exp(-4*x)}, method="cluster" and sortevents=TRUE
#'# rHawkes(n=3,allimmigrants=NULL,repr=0.8,
lam0=function(x) {3.9*exp(-4*x)},
phi=function(x) {3.9*exp(-4*x)},
method="cluster",sortevents=TRUE)
#'#
#'#
#'# Hawkes process simulation with finite simulation end time
(T < infinity) and the immigrant event times are supplied by the user
as input
#'# -----
#'# with inputs: n=5, T=100,allimmigrants=c(0,1,2,5),repr=0.9,
lam0=NULL, phi=function(x) {3.9*exp(-4*x)}, method="cluster" and
sortevents=TRUE
#'# rHawkes(n=5,T=100,allimmigrants=c(0,1,2,5),repr=0.9,
phi=function(x) {3.9*exp(-4*x)},method="cluster",sortevents=TRUE)
#'#
#'# with inputs: n=5, T=100,allimmigrants=c(0,1,2,5),repr=0.8,
lam0=NULL,phi=function(x) {2000*(x+250)^(-2.33)}, method="cluster"
and sortevents=TRUE
#'# rHawkes(n=5,T=100,allimmigrants=c(0,1,2,5),repr=0.8,
phi=function(x) {2000*(x+250)^(-2.33)},method="cluster",
sortevents=TRUE)
#'#
#'#

```

```

#' # Hawkes process simulation with Ogata's thinning algorithm
#' # -----
#' # with inputs: n=5,T=1000,
lam0=function(x) {2000*(x+250)^(-2.33)},
phi=function(x) {2000*(x+250)^(-2.33)}, method="thinning"
and sortevents=TRUE
#' rHawkes(n=5,T=1000, lam0=function(x) {2000*(x+250)^(-2.33)},
phi=function(x) {2000*(x+250)^(-2.33)},method="thinning",sortevents=TRUE)
#'
#' # with inputs: n=5,T=200,lam0=function(x) {0.5},phi=function(x)
{2000*(x+250)^(-2.33)}, method="thinning" and sortevents=TRUE
#' rHawkes(n=5,T=200, lam0=function(x) {0.5}, phi=function(x)
{2000*(x+250)^(-2.33)},method="thinning",sortevents=TRUE)
#'
#'
#' # Hawkes process simulation without excitation function phi
turns to generating (non)-homogeneous Poisson process
#' # (with cluster algorithm)
#' # -----
#' # Non-homogeneous Poisson process generation -
lam0 is a time varying function
#' # with inputs: n=3,allimmigrants=NULL,
lam0=function(x) {3.9*exp(-4*x)},phi=NULL,
method="cluster" and sortevents=TRUE
#' rHawkes(n=3,allimmigrants=NULL,
lam0=function(x) {3.9*exp(-4*x)},
method="cluster",sortevents=TRUE)
#'
#' # with inputs: n=3, T=30,allimmigrants=NULL,
lam0=function(x) {sin(x)+1},phi=NULL, method="cluster" and sortevents=TRUE
#' # Base intensity lam0 is a PERIODIC function as

```

```

lam0=function(x) {sin(x)+1}
#' rHawkes(n=3,T=30,allimmigrants=NULL,
lam0=function(x) {sin(x)+1},method="cluster",sortevents=TRUE)
#'
#' # Homogeneous Poisson process generation - lam0 is scalar (constant)
#' # with inputs: n=3, T=100,allimmigrants=NULL,lam0=5,phi=NULL,
method="cluster" and sortevents=TRUE
#' rHawkes(n=3,T=100,allimmigrants=NULL,lam0=5,method="cluster",
sortevents=TRUE)
#'
#' # Hawkes process simulation without excitation function phi
turns to generating (non)-homogeneous Poisson process
#' # (with thinning algorithm)
#' # Ogata's modified thinning algorithm turns to thinning algorithm
by Lewis and Shedler for (non)-homogeneous Poisson process
#' # -----
#' # Non-homogeneous Poisson process generation -
lam0 is a time varying function
#' # with inputs: n=3,allimmigrants=NULL,
lam0=function(x) {3.9*exp(-4*x)},phi=NULL, method="thinning"
#' rHawkes(n=3,allimmigrants=NULL,
lam0=function(x) {3.9*exp(-4*x)},method="thinning")
#'
#' # Non-homogeneous Poisson process generation -
lam0 is a time varying function
#' # with inputs: n=3,allimmigrants=NULL,
lam0=function(x) {(0.975*1/(x*1*sqrt(2*pi)))} * exp(-(log(x)-0)^2/(2*1^2))),
phi=NULL, method="thinning"
#' # Base intensity and excitation function is log normal that
is not a monotone function
#' rHawkes(n=3,allimmigrants=NULL,

```

```

lam0= function(x) {(0.975*1/(x*1*sqrt(2*pi)))*exp(-(log(x)-0)^2/(2*1^2))},
method="thinning")
#'
#'
#' # Homogeneous Poisson process generation - lam0 is constant
#' # with inputs: n=3, T=100,allimmigrants=NULL,lam0=5,phi=NULL,
method="thinning"
#' rHawkes(n=3,T=100,allimmigrants=NULL,lam0=5,method="thinning")
#'
#'
#' Additional examples
#' # -----
#' # Hawkes process simulation
#' # with inputs: n=3, T=10,allimmigrants=NULL,
lam0= function(x) {(0.975*1/(x*1*sqrt(2*pi)))*exp(-(log(x)-0)^2/(2*1^2))},
phi=function(x) {(0.975*1/(x*1*sqrt(2*pi)))*exp(-(log(x)-0)^2/(2*1^2))},
method="cluster" and sortevents=TRUE
#' # Base intensity and excitation function is log normal that is
not a monotone function
#' rHawkes(n=3,T=10,allimmigrants=NULL,
lam0= function(x) {(0.975*1/(x*1*sqrt(2*pi)))*exp(-(log(x)-0)^2/(2*1^2))},
phi=function(x) {(0.975*1/(x*1*sqrt(2*pi)))*exp(-(log(x)-0)^2/(2*1^2))},
method="cluster",sortevents=TRUE)
#'
#' #Hawkes process simulation
#' # With default inputs: n=1, T=Inf,allimmigrants=NULL,
method="cluster" and sortevents=TRUE
#' # and
#' # with inputs:
#' # lam0 <- function(x) {3.9*exp(-4*x)}
#' # phi <- function(x) {2000*(x+250)^(-2.33)}

```

```

#' rHawkes(allimmigrants=NULL,lam0=lam0,phi=phi,method="cluster",
  sortevents=TRUE)
#'
rHawkes <- function (n=1,T=Inf,allimmigrants=c(0),repr=NULL,
marks=function(x){1},lam0=NULL,
phi=NULL,method=c("cluster","thinning"),sortevents=c(TRUE,FALSE))
{
  if (method == "cluster") return
    (rCluster(n,T,allimmigrants,repr,marks,lam0,phi,sortevents))
  if (method == "thinning") return
    (rThinning(n,T,lam0,phi))
#Ogata's modified thinning algorithm turns to thinning algorithm
of Lewis and Shedler
  #if phi=NULL
}

rCluster <- function (n,T,allimmigrants,repr,marks,lam0,
phi,sortevents)
{

  # computes the inverse of cumulative base intensity function,
  Lam0Inv(t)
  Using numerical inversion
  # if the the immigrants are given then the immigrant subprocess
  is not executed.
  if (length(allimmigrants) == 0)

    { if (!is.numeric(lam0))
      # Base intensity function lam0 is a time varying function
    {
      Lam0Inv <-

```

```

Runuran::pinv.new(pdf=lam0, lb=0, ub=T,uresolution=1e-010)
# computes the total area under the base intensity function
area.base <- vector()
UNU.RAN.object.Lam0Inv    <-
Runuran::unuran.details(Lam0Inv,show=FALSE, return.list=TRUE)
area.base                  <- UNU.RAN.object.Lam0Inv$area.pdf

}else {
  # Base intensity lam0 is a scalar (constant, time invariant)
  Lam0Inv <- function(x) {x/lam0}
  area.base <- lam0*T
}
}

# repr (mean number of offspring) is given by the user.
# The excitation function phi is standardized.

if(is.function(phi))
# if phi is not a function then Hawkes process turns
to non-homogeneous Poisson process
{
if (is.numeric(repr))
{
  # computes the inverse of cumulative excitation function, PhiInv(t)
  Using numerical inversion
  PhiInv    <-
Runuran::pinv.new(pdf=phi, lb=0, ub=T,uresolution=1e-010)
# computes the total area under the excitation function
area.exci <- vector()
UNU.RAN.object.PhiInv    <-

```

```

Runuran::unuran.details(PhiInv,show=FALSE, return.list=TRUE)
area.exci          <-UNU.RAN.object.PhiInv$area.pdf
#to standardize the excitation function phi by (phi/area.exci) and
#to re-scale by the reproduction number repr (phi/area.exci*repr)
phi2               <- function(x) {(phi(x)/area.exci)*repr}
PhiInv             <- Runuran::pinv.new(pdf=phi2, lb=0,
    ub=T,uresolution=1e-010)

area.exci <- vector()
area.exci <-repr

} else {

# repr (mean number of offspring) is NOT given by the user.
The code computes is repr
# which is the area under the excitation function.

# computes the inverse of cumulative excitation function, PhiInv(t)
Using numerical inversion
PhiInv          <- Runuran::pinv.new(pdf=phi, lb=0,
    ub=T,uresolution=1e-010)
# computes the total area under the excitation function
area.exci       <- vector()
UNU.RAN.object.PhiInv <-
Runuran::unuran.details(PhiInv,show=FALSE, return.list=TRUE)
area.exci       <- vector()
area.exci       <-UNU.RAN.object.PhiInv$area.pdf
}
}

```

```

# stores the event times of all n replications
HawkesGen <- list()

# stores the replication numbers of the event times of all n replications
replication_no <- list()

# stores total number of Poisson random numbers
totalPoisson <- vector()
# stores total number of uniform random numbers
totalUniform <- vector()
# stores total number of inverse CDF evaluations
totalinvCDF <- vector()

if (length(allimmigrants) != 0) {
  # i indicates the generation number
  i <- 1
  #the events of the generation 1 for n replications defined by the user
  #remove allimmigrants > T
  allimmigrants <- allimmigrants[allimmigrants<T]
  if (length(allimmigrants)==0)
  {return(print("Check the immigrant event times and the simulation
end time entered"))}

  HawkesGen[[i]] <- rep(allimmigrants,n)

  immigrant <- rep(length(allimmigrants), times=n)

  # stores the replication number of events of the generation 1
  replication_no[[i]] <-rep(which (immigrant > 0), times=immigrant)
  # stores the index (replication number) of the event times that are

```

```

inside of the simulation period (0,T) for generation i

# counting the number of Poisson random numbers for generation i
of all replications
totalPoisson[i] <- 0
# counting the number of uniform random numbers for generation i
of all replications
totalUniform[i] <- 0
# counting the number of inverse CDF evaluations for generation i
of all replications
totalinvCDF[i] <- 0
}

if (length(allimmigrants) == 0) {

# NHPP simulation for Generation 1 (using base intensity -
immigrant process)
# for n independent replications
# NHPP simulation for generation 1,
  Start-----
immigrantnumber <- vector()
# immigrantnumber stores the number of immigrants for each replication

# generates " non-empty cascades "
immigrantnumber <- stats::qpois(stats::runif(n,stats::ppois
(0,area.base),1),area.base)
# generates immigrant number > 0 for n replication
# ppois(0,area.base) returns the cumulative probability of
# less than or equal to 0 of a Poisson random variable with mean
"area base"

```

```

# runif(n,stats::ppois(0,area.base),1) generates n uniform random
numbers
# between (ppois(0,area.base)) and 1
# stats::qpois(stats::runif(n,stats::ppois(0,area.base),1),area.base)
returns
# "n" Poisson random numbers which are all greater than 0.

# generates " empty cascade "
#immigrant <- stats::rpois(n,area.base) generates Poisson random
number including zero.

# area base = total area under the base intensity function from 0
to T
# n = the total number of independent replications of the
Hawkes Process simulation

start <- 0
# i indicates the generation number
i <- 1
# generates the events of the generation 1 for n replications

if((!is.numeric(lam0)))
{
HawkesGen[[i]] <- start + Runuran::uq(Lam0Inv,
  stats::runif(sum(immigrantnumber) ,min = 0, max = area.base)/area.base)
# sum(immigrant)          indicates the total number of offspring
in the generation
# runif(sum(immigrant) ,min = 0, max = area.base)
#          generates standard uniform random numbers for each
offspring in the generation
# Runuran::uq(Lam0Inv,(runif(sum(immigrant) ,min = 0,

```

```

max = area.base)/area.base))
# generates the inter event times between immigrants and
their offspring
}else {
  HawkesGen[[i]]  <- start +
  Lam0Inv(stats::runif(sum(immigrantnumber),min = 0, max = area.base))
}

# stores the replication number of events of the generation 1
replication_no[[i]]  <-rep(which (immigrantnumber > 0),
  times=immigrantnumber[immigrantnumber>0])
# stores the index (replication number) of the event times that
are inside of the simulation period (0,T) for generation i

# counting the number of Poisson random numbers for generation i
of all replications
totalPoisson[i] <- n

# counting the number of uniform random numbers for generation i
of all replications
totalUniform[i] <- sum(immigrantnumber)
# counting the number of inverse CDF evaluations for generation i
of all replications
totalinvCDF[i]  <- sum(immigrantnumber)

}

# There is no immigrant events created that is outside the
simulation period (0,T) for generation 1.

```

```
# because the area under base intensity is calculated for the
period (0,T).
```

```
# NHPP simulation for generation 1,
```

```
End-----
```

```
reject    <- list()
```

```
rejectTime <- list()
```

```
if(is.function(phi))
```

```
# if phi is not a function then Hawkes process turns to
non-homogeneous Poisson process
```

```
{
```

```
# NHPP simulation for generations 2,3,... (using excitation
function - offspring process)
```

```
# for n independent replications
```

```
# NHPP simulation for generations 2,3,...,
```

```
Start-----
```

```
while (length(HawkesGen[[i]]) > 0 ) {
```

```
  offspring <- vector()
```

```
  offspring <- stats::rpois(length(HawkesGen[[i]]),area.exci*marks
    (length(HawkesGen[[i]])))
```

```
# area.exci = total area under the excitation function from 0 to T
```

```
# length(HawkesGen[[i]]) = the total number of living replications
```

```

that can generate further offspring
# marks(x) = A positive random number that defines the weight of
the event
#           where x denotes the number of events which have marks.

# counting the number of Poisson random numbers for generation i
of all replications
totalPoisson[i+1] <- length(HawkesGen[[i]])

if(sum(offspring) == 0) {break}
# checks for living replications

i <- i+1
# the generation number, i is updated
# rep(HawkesGen[[i-1]], times= offspring) indicates that the parent
event time of each offspring generated in this generation
# runif(sum(offspring),min = 0, max = area.exci)
#           generates standard uniform random numbers for each
offspring in the generation
# Runuran::uq(PhiInv,runif(sum(offspring),
min = 0, max = area.exci)/area.exci)
#           generates the inter event times between parents and
their offspring
# generates the events of the generation i for n replications
#HawkesGen[[i]] <- rep(HawkesGen[[i-1]], times= offspring) +
  Runuran::uq(PhiInv,stats::runif(sum(offspring),min = 0,
max = area.exci)/area.exci)

# the above can be written as one of them. The second one is
probably faster

```

```

#HawkesGen[[i]]    <- rep(HawkesGen[[i-1]], times= offspring) +
  Runuran::uq(PhiInv,stats::runif(sum(offspring)))
HawkesGen[[i]]    <- rep(HawkesGen[[i-1]], times= offspring) +
  Runuran::ur(PhiInv,n=sum(offspring))

# stores the replication number of events of the generation i
replication_no[[i]] <- rep(replication_no[[i-1]],
times=offspring)

# counting the number of uniform random numbers for generation i
of all replications
totalUniform[i] <- sum(offspring)

# counting the number of inverse CDF evaluations for generation i
of all replications
totalinvCDF[i] <- sum(offspring)

if (T!=Inf) {
  reject[[i]] <- which(HawkesGen[[i]] > T)
  rejectTime[[i]] <- HawkesGen[[i]][HawkesGen[[i]] > T]
  # stores the index of the event time that is greater than
  simulation duration, T
  if (length(reject[[i]])>0){
    HawkesGen[[i]] <- HawkesGen[[i]][-(reject[[i]])]
    replication_no[[i]] <- replication_no[[i]][-(reject[[i]])]
  }
}

}

```

```

# NHPP simulation for generations 2,3,...,
End-----

}

# sorts the event times according to the replication number(y)
and then event time(x)
dataframe_replication <- data.frame(x = unlist(HawkesGen),
y =unlist(replication_no))
output_event_replication <- list()
if (sortevents == FALSE){
  # stores the sorted event times of all n replications
  output_event_replication[[1]] <- dataframe_replication[[1]]
  # stores the replication number of sorted events of all
  n replications
  output_event_replication[[2]] <- dataframe_replication[[2]]
  # stores the total REJECTED event times
  output_event_replication[[3]] <- unlist(rejectTime)

  # stores the total number of ACCEPTED events
  output_event_replication[[4]] <- length(dataframe_replication[[1]])
  # stores the total number of REJECTED events
  output_event_replication[[5]] <- length(unlist(rejectTime))

  # stores total number of Poisson random numbers
  output_event_replication[[6]] <- sum(totalPoisson)
  # stores total number of uniform random numbers
  (i.e. total number of events generated, accepted + rejected)
  output_event_replication[[7]] <- sum(totalUniform)
  # stores total number of inverse CDF evaluations

```

```

output_event_replication[[8]] <- sum(totalinvCDF)

names(output_event_replication) <- c("Accepted event times",
  "Replication no", "Rejected Event times",
  "Total number of accepted event times",
  "Total number of rejected event times",
  "Total number of Poisson random numbers",
  "Total number of uniform random numbers",
  "Total number of inverse cumulative intensity function evaluations")

return(output_event_replication)
}

sorted_dataframe_dataframe_replication <-
  dataframe_replication[with(dataframe_replication, order(y,x)),]

# stores the sorted event times of all n replications
output_event_replication[[1]] <-
  sorted_dataframe_dataframe_replication[[1]]
# stores the replication number of sorted events of
all n replications
output_event_replication[[2]] <-
  sorted_dataframe_dataframe_replication[[2]]
# stores the total REJECTED event times
output_event_replication[[3]] <- unlist(rejectTime)

# stores the total number of ACCEPTED events
output_event_replication[[4]] <- length(dataframe_replication[[1]])
# stores the total number of REJECTED events
output_event_replication[[5]] <- length(unlist(rejectTime))

```

```

# stores total number of Poisson random numbers
output_event_replication[[6]] <- sum(totalPoisson)
# stores total number of uniform random numbers (i.e. total number
of events generated, accepted + rejected)
output_event_replication[[7]] <- sum(totalUniform)
# stores total number of inverse CDF evaluations
output_event_replication[[8]] <- sum(totalinvCDF)

names(output_event_replication) <- c("Accepted event times",
  "Replication no",
  "Rejected Event times",
  "Total number of accepted event times",
  "Total number of rejected event times",
  "Total number of Poisson random numbers",
  "Total number of uniform random numbers",
  "Total number of inverse cumulative intensity function evaluations")

return(output_event_replication)
}

#-----

#-----

rThinning <- function(n,T,lam0,phi)
  # for non-increasing monotone base intensity and excitation functions
{
  #options(warn=-1)

  if(is.null(phi))
    {phi <- function(x){0}}
  if(is.numeric(phi))

```

```

{
  phi2 <- phi
  phi <- function(x){phi2}}

# STEP 1. Compute maximum value of lam0 (t) between time 0 and T,
maxlam0

# lam0 is a non-increasing monotone function

# stores total number of exponential random numbers
totalExponential <- integer()
totalExponential <- 0
# stores total number of excitation function evaluations
totalExcinFunctEva <- integer()
totalExcinFunctEva <- 0

# stores total number of excitation function evaluations
for accepted events
totalExcinFunctEvaACCEPT <- integer()
totalExcinFunctEvaACCEPT <- 0

# stores total number of uniform random numbers for thinning algorithm
totalUniformTH <- integer()
totalUniformTH <- 0
# stores total number of accepted candidate event time
totalAccepted <- integer()
totalAccepted <- 0
# stores total number of rejected candidate event time
totalRejected <- integer()
totalRejected <- 0

if (is.numeric(lam0))

```

```

{maxlam0_initial      <- lam0}
else {maxlam0_initial      <- lam0(0)}

if (is.nan(maxlam0_initial)) {
  return (print("Check the base intensity function, it should be a
    non-increasing monotone function for Ogata's modified
    thinning algorithm"))
}

# STEP 2. Compute maximum value of phi(t) between time 0 and infinity,
maxphi
# phi is a non-increasing monotone function

maxphi_initial      <- phi(0)

if (is.nan(maxphi_initial)) {
  return (print("Check the excition function, it should be a non-
    increasing monotone function for Ogata's modified thinning algorithm"))
}

# stores the event times of all n replications
HawkesReplicationEvent <- list()
# stores the replication numbers of the event times of all
n replications
replication_no <- list()

i<-0
# counter for living cascade (i.e. having events)

repeat {
  # generates n living cascades
  # n = number of independent Hawkes Process replications

```

```

maxlam0 <- maxlam0_initial
maxphi  <- maxphi_initial
# STEP 3. lstar = maxlam0
# ... upper bound for the thinning algorithm
lstar   <- maxlam0

# STEP 4. HawkesEv = empty set
# ... Hawkes Process(HP) event set
HawkesEv <- vector()
# STEP 5. n = 0
# . number of events
# in the set HawkesEv
nn <- 0
# nn = number of events in the current simulation replication
# STEP 6. curT = 0
# . current time
curT <- 0
# STEP 7. Repeat
repeat
{
  # STEP 7.a. Generate standard Exponential, E
  # STEP 7.b t = E\lstar
  # . inter event time
  t <- stats::rexp(1)/lstar

  # counts the exponential random numbers generated
  totalExponential <- totalExponential + 1

  # STEP 7.c. curT = curT + t
  # update current time
  curT <- curT + t

```

```

# STEP 7.d. If (curT > T), {Go to Step 8}
if (curT > T) {break}

# STEP 7.e. sumphi = 0
# . sum of excitations triggered by
the events in the process history
sumphi <- 0

# STEP 7.f. If (n > 0)
if (nn > 0)
{
  # STEP 7.f.i. For i = 1 to n {sumphi = sumphi +
  phi (curT - HawkesEv[n])}
  for (j in 1:nn)
  {
    sumphi <- sumphi + phi(curT - HawkesEv[j])
    # counts excitation function evaluations
    totalExcinFuncEva <- totalExcinFuncEva + 1
    totalExcinFuncEvaACCEPT <- totalExcinFuncEvaACCEPT + 1
  }
}

# STEP 7.g. curInt = lam0(curT) + sumphi
# . current intensity

if (is.numeric(lam0))
{curInt <- lam0 + sumphi}
else {curInt <- lam0(curT) + sumphi}

# STEP 7.h. Generate standard Uniform, U
# STEP 7.i If U L.E. curInt\lstar
if (curInt > lstar) {
  return (print("Check the excition function,
  it should be a non-increasing monotone function

```

```

    for Ogata's modified thinning  algorithm"))
  }
  if (stats::runif(1, min = 0, max = 1) <= curInt / lstar)
    # STEP 7.i.i.i. then
    {
      # STEP 7.i.i.1 n = n + 1
      nn <- nn + 1
      # STEP 7.i.i.2 HawkesEv[n] = curT # Add the new event
      into the HP event set
      HawkesEv[nn] <- curT
      # STEP 7.i.i.3. Update maximum value of lam0(t) between
      time curT and T,

      if (is.numeric(lam0))
        {maxlam0      <- lam0}
      else {maxlam0      <- lam0(curT)}

      # STEP 7.i.i.4. lambda star = maxlam0 + maxphi + sumphi
      lstar <- maxlam0 + maxphi + sumphi
      #counts total number of accepted candidate events
      totalAccepted <- totalAccepted +1
    }
  # STEP 7.i.i.ii. else
  else
  {
    # STEP 7.i.i.ii.1 Update maximum value of lam0(t) between
    time curT and T, maxlam0
    if (is.numeric(lam0))
      {maxlam0      <- lam0}
    else {maxlam0      <- lam0(curT)}
    # STEP 7.i.i.ii.2. lambda star = maxlam0 + sumphi

```

```

    lstar <- maxlam0 + sumphi
    #counts total number of rejected candidate events
    totalRejected <- totalRejected +1
    # correction for total number of excitation function evaluations
    # number of excitation function evaluations are removed.
    totalExcinFunctEvaACCEPT <- totalExcinFunctEvaACCEPT - nn
  }

  # counting the number of uniform random numbers for acceptance
  rejection
  totalUniformTH <- totalUniformTH +1

  # stopping condition for simulation duration T=infinity
  if (T==Inf){if (lstar < 0.00001) {break}}
}

if(length(HawkesEv)>=0){
  i <- i+1
  # stores the event times of all n replications
  HawkesReplicationEvent[[i]] <- HawkesEv
  # stores the replication numbers of the event times of all
  n replications
  replication_no[[i]] <-
  rep(i, times=length(HawkesReplicationEvent[[i]]))
  # increase counter
  if (i == n){break}
}
}

# STEP 8. Return Hawkes Process event set and replication number
output_event_replication <- list()

```

```

# stores the event times of all n replications
output_event_replication[[1]] <- unlist(HawkesReplicationEvent)
# stores the replication number of sorted events of all n replications
output_event_replication[[2]] <- unlist(replication_no)

# stores total number of accepted candidate event time
output_event_replication[[3]] <- totalAccepted
# stores total number of rejected candidate event time
output_event_replication[[4]] <- totalRejected

# stores total number of exponential random numbers
output_event_replication[[5]] <- totalExponential
# stores total number of uniform random numbers for thinning algorithm
output_event_replication[[6]] <- totalUniformTH
# stores total number of excitation function evaluations
output_event_replication[[7]] <- totalExcinFunctEva
# stores total number of excitation function evaluations
for accepted events
output_event_replication[[8]] <- totalExcinFunctEvaACCEPT

names(output_event_replication) <- c("Accepted event times",
  "Replication no",
  "Total number of accepted event times",
  "Total number of rejected event times",
  "Total number of exponential random numbers",
  "Total number of uniform random numbers",
  "Total number of excitation function evaluations",
  "Total number of excitation function evaluations at accepted events"

return(output_event_replication)

```

}

#-----

B.2. The Pseudo Code of the Presented Hawkes Process Algorithm

The pseudo code of the presented cluster-based Hawkes process simulation algorithm for the unbounded interval $(0, T = \infty)$ is presented below (The area under the base intensity function is bounded and the branching factor is less than one).

Input:

simulation duration : $T = \text{infinity}$

base intensity function : $\text{lam0}(t)$, for the immigrant process

excitation function : $\text{phi}(t)$, for the offspring process

Begin

- (i) The set up of the algorithm
 - (a) Compute the inverse of cumulative base intensity function, $\text{Lam0Inv}(t)$
 - (b) Compute the inverse of cumulative excitation function, $\text{PhiInv}(t)$
 - (c) Compute the total area $\text{Lam0}(T)$ under the base intensity function, $\text{lam0}(t)$ during the simulation period $(0, T)$
 - (d) Compute the total area $\text{Phi}(T)$ under the excitation function, $\text{phi}(t)$ during the simulation period $(0, T)$
- (ii) Create the immigrant generation (generation 0)
 - (a) Generate the number of immigrant events n from the Poisson distribution with mean $\text{Lam0}(T)$
 - (b) Generate n uniformly distributed random numbers u_j , $j = 1, \dots, n$ on the interval $(0, \text{Lam0}(T))$,
 ... u_j s are the event times of the homogeneous Poisson Process with unit

rate on the interval $(0, \Lambda(T)]$

- (c) Create the immigrant events, $\text{imEv} = \text{Lam0Inv}(u_j)$, where $u_j, j = 1, \dots, n$
 ... The event times of $u_j, j = 1, \dots, n$ of the homogeneous Poisson process
 is converted to the event times of the immigrant sub-process with intensity
 function, $\text{lam0}(t)$ on the interval $(0, T]$

- (d) $i = 0$

... generation number for the immigrants

- (e) Generation $[i] = \text{imEv}$

- (iii) Create the offspring generations (generation 1,2,...)

Repeat

- (a) Check the number of events in generation i

... if $(\text{length}(\text{generation}[i]) == 0)$ break

... There is no event in the generation i to generate further offspring

- (b) Generate the number of offspring events of the next generation for each event
 in the current generation

- i. Generate the number of offspring m for each event in the current gen-
 eration i from the Poisson distribution with mean $\Phi(T)$ respectively

... $m = \text{Poi}(\text{length}(\text{generation}[[i]]), \Phi(T))$

... if $(\text{sum}(m) == 0)$ break

... There aren't any offspring to be generated by the parent events in
 the current generation i

- (c) Replicate the parent event times for each offspring

... Parent event time = $\text{rep}(\text{generation}[[i]], m)$

... These are the parent event times that have offspring

- (d) Generate the inter-event time between the parent event time and the off-
 spring event time

- i. Generate m uniformly distributed random numbers on $(0, \Phi(T))$,

$u_j, j = 1, \dots, m$

- ii. Inter event times = $\Phi\text{Inv}(u_j), u_j, j = 1, \dots, m$

- (e) Compute the event times of generation " $i + 1$ " (next generation)

- i. Next generation $i < i+1$

```

        ii. Generation[i] < - Parent event time + Inter event times
    Until
    (iv) Sort the Hawkes Process event times
        (a) HawkesEv = sort (unlist(Generation)) .. Hawkes Process events

End

```

Output:

Hawkes Process event times

This pseudo code is implemented for the case where the simulation period is unbounded $(0, T = \infty)$. It can be easily modified for the case with finite simulation period end time, $(0, T < \infty)$ by adding the following step (which removes the offspring event times greater than simulation end time) to the pseudo code:

- (iii)-(e)-(iii) Removing event times greater than T
 $\text{reject}[i] < - \text{which}(\text{HawkesGen}[i] > T)$
 $\text{HawkesGen}[i] < - \text{HawkesGen}[i][-(\text{reject}[i])]$

This algorithm can simulate marked Hawkes process (marks are independent and identically distributed) by modifying the following step with replacing $\Phi(T)$ with $\text{mark} * \Phi(T)$ where "mark" is an input for the algorithm.

- (iii)-(b)-(i) with independent and identically distributed "marks"
 Generate the number of offspring, m for each event in the current generation i from the Poisson distribution with mean "mark*Phi (T)" respectively.
 $\dots m = \text{Poi}(\text{length}(\text{generation}[[i]]), \text{mark} * \Phi(T))$
 $\dots \text{if } (\text{sum}(m) == 0) \text{ break}$
 $\dots$ There is no offspring to be generated by the parent events in the current generation i .

This algorithm can simulate renewal Hawkes process without any change because it accepts base intensity function as a function of time $\lambda_0(t)$.

B.3. The Outputs of the Package for the Sample Cases

The outputs of the function `rhawkes()` are presented.

Simulating non-empty Hawkes Process Cascades

The function command

```
rHawkes(n=2,
        allimmigrants=NULL,
        marks = function(x) {rexp(x,1)},
        lam0=function(x) {3.9*exp(-4*x)},
        phi=function(x) {3.9*exp(-4*x)},
        method="cluster",sortevents=TRUE)
```

The output:

\$'Event times'

```
[1] 0.4830818 0.5064280 0.5094081 0.6149907 0.6497321 0.6635942
0.6804674 0.8447767 0.9313710 1.3272573 1.3715737 0.3166842 0.5717446
```

\$'Replication no'

```
[1] 1 1 1 1 1 1 1 1 1 1 1 2 2
```

Generating Multiple Simulation Runs

The function command

```
rHawkes(n=5,
        T=3,
        allimmigrants=NULL,
        lam0=function(x) {2},
        phi=function(x) {3.9*exp(-4*x)},
        method="cluster",
        sortevents=TRUE)
```

The output:

\$'Event times'

```
[1] 0.1073102 1.7734559 2.4733248 2.4951079 2.5913834 0.9789724
0.9837567 1.0315183 1.0349784 2.4594251 2.4649887 2.7516585 2.7717737
[14] 2.7777317 2.9031240 2.9380896 2.9381281 2.9726939 2.9779821
1.2966647 1.3047193 2.8948099 0.3316476 0.4165554 0.5261560 0.7997737
[27] 1.6867091 1.9464393 2.0222701 2.1130436 2.1573465 2.2567295
2.2635507 2.2748830 2.3337279 2.3570845 2.4224563 2.4278340 2.4341448
[40] 2.6047378 2.8294871 2.9482869 2.9665670 0.6103495 0.6171197
0.8333481 1.1242172 1.2419731 1.4458465
```

\$'Replication no'

```
[1] 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 4 4 4 4 4 4 4 4 4 4
4 4 4 4 4 4 4 4 4 4 4 4 5 5 5 5 5 5
```

Simulating a Social Media Post

The function command

```
rHawkes(n=5,
        allimmigrants=c(0),
        repr=0.8,
        phi=function(x) {2000*(x+250)^(-2.33)},
        method="cluster",
        sortevents=TRUE)
```

The output:

\$'Event times'

```
[1] 0.00000 31.98126 59.42806 137.02383 287.76077 317.62337
0.00000 0.00000 88.31031 288.70278 564.42670 2292.83592
[13] 0.00000 38.82676 656.36277 659.85000 1005.22997 1899.86316
0.00000 10.91570 815.02919 927.67638 1652.11249 2202.29517
[25] 3333.69084 6177.32602 6213.02327 6303.71864 6501.64224 6716.86125
6739.83475 6742.99949 6748.83018 6748.89332 6778.70944 6782.62731
[37] 6827.54798 6936.14676 7071.06955 7071.55712 7179.77808 7183.01006
7269.84668 7289.51384 7480.74416 7501.77368 7590.81190 7900.68169
[49] 8007.91758 8141.24444 8570.37213 8806.58881
```

\$'Replication no'

```
[1] 1 1 1 1 1 1 2 3 3 3 3 3 4 4 4 4 4 4 5 5 5 5 5 5 5 5 5 5 5 5 5
5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
```

Simulating Hawkes Process with Reproduction Number

The function command

```
rHawkes(n=5,
        allimmigrants=NULL,
        repr=0.8,
        lam0=function(x) {3.9*exp(-4*x)},
        phi=function(x) {2000*(x+250)^(-2.33)},
        method="cluster",
        sortevents=TRUE)
```

The output:

\$'Event times'

```
[1] 3.389461e-02 3.082255e-01 1.513344e+02 3.647722e-01 6.649609e+01
1.498766e+02 1.765227e+02 8.081206e+02 1.070369e+03 2.054781e+03
[11] 2.240801e+03 5.054515e-01 9.354984e+01 6.588075e-01 2.454652e+02
7.855275e+02 1.525528e-01 2.510028e-01 3.276676e-01 2.298866e+02
[21] 2.626120e+02 2.804852e+02 3.034984e+02 3.626833e+02 4.960642e+02
5.004350e+02 5.053814e+02 5.519711e+02 5.585283e+02 5.842199e+02
[31] 6.487612e+02 6.686871e+02 6.890786e+02 8.051281e+02 8.778070e+02
8.941536e+02 1.052150e+03 1.085940e+03 1.154588e+03 1.226553e+03
[41] 1.281768e+03 1.397172e+03 1.408614e+03 1.477868e+03 1.794233e+03
1.865206e+03 1.876077e+03 1.879037e+03 1.920169e+03 1.941653e+03
[51] 2.037741e+03 2.041753e+03 2.052504e+03 2.096406e+03 2.130333e+03
2.256659e+03 2.543916e+03 2.575275e+03 2.890768e+03 3.236168e+03
```

\$'Replication no'

```
[1] 1 1 1 2 2 2 2 2 2 2 2 3 3 4 4 4 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
```

5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5

Simulating a Non-Homogeneous Poisson Process

The function command

```
rHawkes(n=5,
        allimmigrants=NULL,
        lam0=function(x) {3.9*exp(-4*x)},
        method="cluster",
        sortevents=TRUE)
```

The output:

\$'Event times'

```
[1] 0.208068147 0.050048442 0.229239481 0.575573435 0.030336458
0.164560873 0.649065356 0.033842245 0.007942425
```

\$'Replication no'

```
[1] 1 2 2 2 3 3 3 4 5
```

Simulating Hawkes Process with Different Algorithms

The function command (with cluster-based algorithm)

```
rHawkes(n=3,
        T=200,
        allimmigrants=NULL,
        lam0=function(x) {0.1},
        phi=function(x) {2000*(x+250)^(-2.33)},
        method="cluster",
        sortevents=TRUE)
```

The output:

\$'Event times'

```
[1] 0.1672324 8.8541950 16.8641343 23.2326442 31.2578664
36.6796042 44.3304084 46.0769248 51.0350871 56.0665788 65.5062178
[12] 68.5808367 72.1573770 76.8051972 87.4308066 96.3625610
107.6882660 118.2483803 119.6845280 119.9960764 126.5260627 146.6200654
[23] 158.1540279 164.9219716 182.4915345 194.9488647 7.5451955
11.5789660 34.2185398 39.7881504 43.6292274 49.7515946 58.6194448
[34] 61.5508256 74.5661240 77.1740956 77.8591226 82.0100484
95.0202189 17.1595837 135.7120615 142.5250839 143.6189747 148.2058825
[45] 159.4871264 168.1087086 170.3299712 184.0607791 188.2200610
0.7250604 7.3328963 18.2870156 34.0380715 39.0418555 47.2522030
[56] 47.3264852 50.6834036 62.0243458 62.8088741 68.2885639
74.2032326 83.7523122 84.7482438 107.3974390 107.4661056 112.8656549
[67] 126.5855203 143.0604689 158.1338048 169.9913540 197.0154846
```

\$'Replication no'

```

[1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3
3 3 3 3 3
[69] 3 3 3

```

The function command (Ogata's modified thinning algorithm)

```

rHawkes(n=3,
        T=200,
        allimmigrants=NULL,
        lam0=function(x) {0.1},
        phi=function(x) {2000*(x+250)^(-2.33)},
        method="thinning",
        sortevents=TRUE)

```

The output:

\$'Event times'

```

[1] 4.6734778 5.4342413 31.1284074 67.9439752 69.0869004
74.1952115 79.9812577 108.6729608 121.7760125 122.7351043 146.4839528
[12] 148.4393579 162.1019435 168.9966818 176.7096375 184.9537656
186.9429765 187.2535124 196.6706219 10.6286252 24.1374684 32.6689725
[23] 52.4595782 52.4687286 64.2087213 66.5653169 86.6634380
98.2842113 101.9384221 109.2110443 113.6659207 121.1301407 125.6653975
[34] 126.0321295 126.4888749 140.9352263 148.5698445 155.2651173
164.1352996 180.4817636 188.2825486 199.8888032 0.7405287 0.8136153
[45] 6.2892777 11.9722491 20.9315294 24.3643885 39.8122214
44.1038634 45.4371627 48.7728772 49.1753886 53.5824375 56.3793436
[56] 61.3384143 91.7950907 98.0030696 110.0292933 123.5378326

```

133.1065787 133.7162535 154.0804313 157.2902935 161.4603837 163.2487439
[67] 168.6357717 168.8610919 173.7692996 189.0429317 199.2150881

\$'Replication no'

[1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
3 3 3 3 3 3
[69] 3 3 3



APPENDIX C: THE ADDITIONAL PERFORMANCE EXPERIMENTS

C.1. The Number of Operations of the Proposed Algorithm

The experiments in this section are performed with the following input parameters (Parameter setting 5):

- Base intensity, $\lambda_0(t) = 3.9 * e^{-4t}$
- Excitation function, $\varphi(t) = 3.9 * e^{-4t}$
- Simulation duration, $T = \text{infinity}$
- Number of independent replications, cascades,
k = 1,5,10,25,50,100,250,500,1.000,5.000,10.000,25.000,50.000,
100.000,250.000,500.000,750.00, and 1.000.000

In other words, the unbounded simulation duration T , the area under the intensity functions less than one and multiple cascades. The following variables are counted:

- the events generated, n
- Poisson random numbers, p
- Uniform random numbers, u
- The inverse CDF evaluations, c

The experiment results for the presented cluster-based algorithm are presented in Figure C.1:

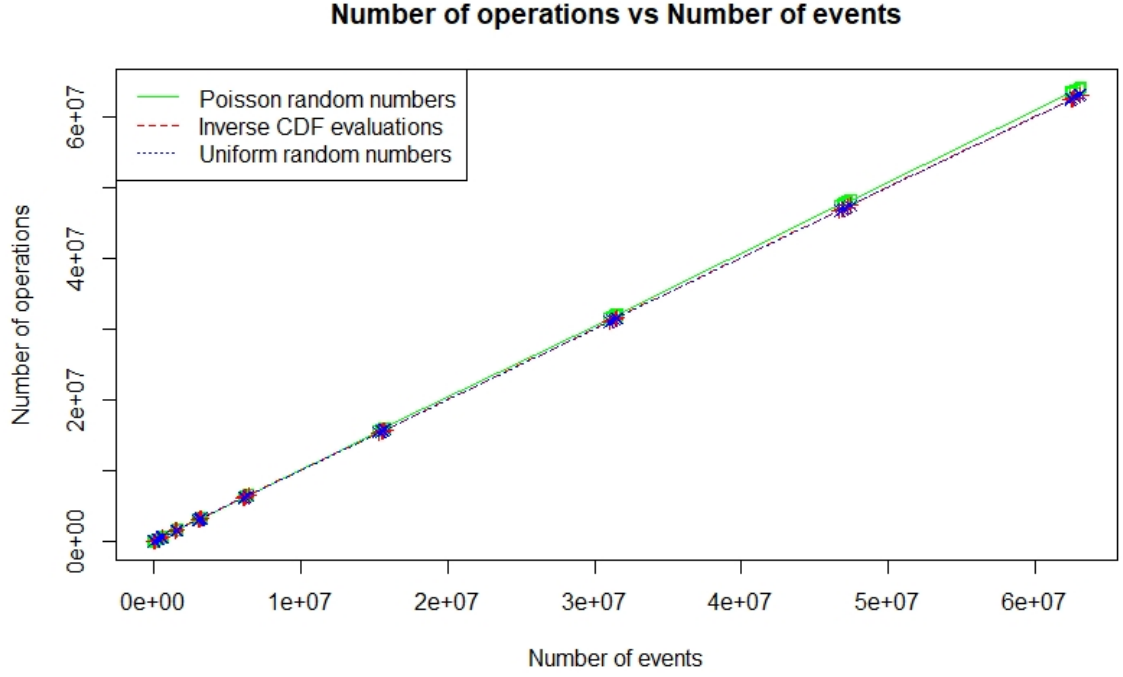


Figure C.1. Cluster based algorithm - events vs number of operations with parameter setting 5

The trend lines which are drawn in Figure C.1 are as follows:

For the Poisson random numbers p , it is expressed as

$$Poisson \text{ random numbers} = 1.000 * events + 1.000 * replication. \quad (C.1)$$

For the uniform random numbers u , it is expressed as

$$Uniform \text{ random numbers} = 1.000 * events. \quad (C.2)$$

For the inverse CDF evaluations c , it is expressed as

$$Inverse \ CDF \ evaluations = 1.000 * events. \quad (C.3)$$

The experiment results show that

- Poisson random numbers, $p = n + k$
- Uniform random numbers, $u = n$
- Inverse CDF evaluations, $c = n$

where " k " is the number of independent simulation replications and " n " is total number of generated events during k replications.

The experiment results show that the number of operations is increasing linearly as the number of events increases. The expected number of events for a single cascade is on the next section.

C.2. The Theoretical and Empirical Results Comparison

In this section, the experiment outputs are compared with the theoretical values. The experiments are performed with

- (i) Constant base intensity function, λ_0 in Section 6.1
- (ii) Time dependent base intensity function, $\lambda_0(t)$ in Section C.1

C.2.1. The Experiments with Constant Base Intensity

The experiment outputs based on the parameter setting 3 from Section 6.1 are compared with theoretical values.

The expected number of events of a stationary Hawkes process [7] can be written as

$$\frac{\lambda_0 * t}{1 - \int_0^\infty \varphi(s) ds}. \quad (\text{C.4})$$

The base intensity, $\lambda_0 = 2$ and the area under the excitation, $\varphi(t) = 0.975$ for the simulation periods, $T = 10.000, 15.000, 20.000, 25.000, 30.000$.

The expected number of events for the simulation period $(0, T=10.000)$ is 800.000 events. The average number of events calculated according experiment outputs is 799.985.

The expected number of events for the remaining simulation periods ($T=15.000, 20.000, 25.000, 30.000$) are 1.200.000, 1.600.000, 2.000.000 and 2.400.000 respectively.

The average number of events calculated according to the experiment results for the remaining simulation periods ($T=15.000, 20.000, 25.000, 30.000$) are 1.190.055, 1.590.238, 1.996.560 and 2.365.969 respectively.

The experiment results are coherent with the theoretical values.

C.2.2. The Experiments with the Time Varying Base Intensity Function

The experiment outputs based on the parameter setting 5 from Section C.1 are compared with the theoretical values. The expected number of generated events for the simulation period $(0, T]$ can be written as

$$\frac{\int_0^t \lambda_0(s) ds}{1 - \int_0^\infty \varphi(s) ds}. \quad (\text{C.5})$$

According to the parameter settings 5, the expected number of events per replication(cascade) is calculated as

$$n = \frac{\int_0^\infty \lambda_0(t) dt}{1 - \int_0^\infty \varphi(t) dt} = \frac{3.9/4.0}{1 - (3.9/4.0)} = \frac{0.975}{1 - 0.975} = \frac{0.975}{0.025} = 39. \quad (\text{C.6})$$

The average of the experiment result with $k=1,000.000$ replications is 39.03 per replication (cascade) including empty cascades.

C.2.3. The Experiments with Ogata's Modified Thinning Algorithm

In this section, the lower bounds (only for the accepted events) of the theoretical and experimental number of excitation function evaluations of the Ogata's modified thinning algorithm values are compared. The theoretical lower bound value is stated in Equation 4.1 as $Total\ Ex = \frac{n^2-n}{2}$. "n" is the number of generated events for Hawkes process. It doesn't include the rejected events at the thinning operation. Therefore, " $Total\ Ex$ " is a lower bound.

Based on the experiment outputs in Section 6.3 "The Comparison of the Number of the Operations with Ogata's Modified Thinning Algorithm", the trend line (drawn in Figure 6.5) of the number of excitation function evaluations of Ogata's modified thinning algorithm for the events in process history (in other words only the accepted events at the thinning operation) is in Equation 6.9 as " $Excitation = 5.000 * 10^{-01} events^2 - 5.000 * 10^{-01} events$ ". The experiment output is coherent with the theoretical value in Equation 4.1.

APPENDIX D: THE SOFTWARE PACKAGES FOR HP SIMULATION

The R packages in CRAN are listed using the following R command in R console.

```
tools::CRAN_package_db()[, c("Package", "Description", "Author")]
```

There are 20.703 R packages at the end of March 2024.

Some of the software packages in the literature are listed below:

The R Package “emhawkes”: The package “emhawkes” is created by Kyungsub Lee [53]. Kyungsub Lee described the package as follows: This package simulates a multivariate Hawkes model, introduced by Hawkes (1971), with an exponential kernel and fits the parameters from the data.

This R package has the function “`hsim(object, size = 100, lambda_component0 = NULL, N0 = NULL, ...)`” for the Hawkes Process simulation. The descriptions of the input parameters from the R package documents are as follows:

- object `hspec`-class. S4 object that specifies the parameter values (The object `hspec`-class contains the parameter values such as μ , α , β).
- Number of observations.
- Starting values of lambda component. numeric or matrix

The function “`hsim`” simulates Hawkes process with only exponential excitation function.

The R Package “hawkes”: The package “hawkes” is implemented by Riadh Zaatour [11]. Riadh Zaatour described the package as follows: The package allows to simulate Hawkes Process both in univariate and multivariate settings. It gives func-

tions to compute different moments of the number of jumps of the process on a given interval, such as mean, variance or autocorrelation of process jumps on time intervals separated by a lag [11]. Ogata's modified thinning algorithm is used to simulate the Hawkes process. The base intensity function is a scalar for in the mono-variate property. The excitation function for this simulation is exponential decay because of its Markov property. For the functions, `jumpAutocorrelation`, `jumpMean`, `jumpVariance` of the package, Riadh Zaatour referred to the paper of [54] that unified clustering behavior of Hawkes Process with mean reversion.

The R Package "hawkesbow": The package "hawkesbow" is implemented by Felix Cheyssou [10]. Some general use functions for Hawkes processes are also included: simulation of (in)homogeneous Hawkes Process, maximum likelihood estimation, residual analysis, etc. There are two functions to simulate Hawkes processes, "hawkes_ogata" and "hawkes". The former uses Ogata's modified thinning algorithm as its name implies. Cheyssou indicates that "hawkes_ogata" is less efficient than "hawkes". The latter simulates a Hawkes process using its cluster representation [10].

The R Package "IHSEP": The package "emhawkes" is implemented by Feng Chen [8]. The description of the package from its documentation in R is that "it simulates an inhomogeneous self-exciting process (IHSEP), or Hawkes process, with a given (possibly time-varying) baseline intensity and an excitation function. It calculates the likelihood of an IHSEP with given baseline intensity and excitation functions for an (increasing) sequence of event times. It also calculates the point process residuals (integral transforms of the original event times). Calculate the mean intensity process." The package has functions for Hawkes process simulation, "simHawkes0", "simHawkes1" and "simHawkes1a" and functions for non-homogeneous Poisson process simulation "simPois" and "simPois0". The functions "simHawkes0", "simHawkes1" and "simHawkes1a" are using the cluster based representation of Hawkes process to simulate Hawkes process.

The R Package “MRHawkes”: The package “MRHawkes” is implemented by Tom Stindl and Feng Chen [55]. The package “MRHawkes” simulate a (bivariate) multivariate renewal Hawkes (MRHawkes) self-exciting process, with given immigrant hazard rate functions and offspring density function. Calculate the likelihood of a MRHawkes process with given hazard rate functions and offspring density function for an (increasing) sequence of event times. Calculate the Rosenblatt residuals of the event times. Predict future event times based on observed event times up to a given time [56]. The function “simMRHawkes” simulate an (bivariate) renewal Hawkes (MRHawkes) process with given renewal immigration distribution functions λ_0 , offspring density functions φ and branching ratios η using the cascading structure of the process. This function simulates Hawkes process using cluster representation of Hawkes Process. The offspring processes are simulated using thinning method by Lewis and Shedler.

The R Package “PtProcess”: The package “PtProcess” is implemented by David Harte [36]. David Harte described the package as follows: The package fits and analyses time dependent marked point process models with an emphasis on earthquake modeling. It contains routines for the fitting of time dependent point process models, particularly marked processes with “jumps”. These models have applications to earthquake data. [36]. Thinning method [3, 4] and is used to simulate a point process with specified ground intensity function.

The R Package “RHawkes”: The package “RHawkes” is implemented by Feng Chen and Tom Stindl [9]. From the description of the package in CRAN, the renewal Hawkes (RHawkes) process is an extension to the classical Hawkes self-exciting point process widely used in the modeling of clustered event sequence data. This package provides functions to simulate the RHawkes process with a given immigrant hazard rate function and offspring birth time density function, to compute the exact likelihood of a RHawkes process using the recursive algorithm proposed by [57], to compute the Rosenblatt residuals for goodness-of-fit assessment, and to predict future event times based on observed event times up to a given time. A function implementing the linear time RHawkes process likelihood approximation algorithm proposed in [56] is also

included.

The function "simRHawkes" simulate a renewal Hawkes (RHawkes) process with given renewal immigration distribution function, offspring density function and branching ratio. This function simulates Hawkes process using cluster representation of Hawkes Process. The offspring processes are simulated using thinning method by Lewis and Shedler.

The R Package "stelfi": The package "stelfi" is implemented by Charlotte M. Jones-Todd, Alec van Helsdingen and Xiangjie Xue. The package has the function "sim_hawkes" to simulate a self-exciting Hawkes process. The function "sim_hawkes" simulate Hawkes in two different ways

- Method 1 (default) The simulation end time is specified.
- Method 2 - specifies the number of observations to simulate.

Ogata's modified thinning algorithm is used to simulate Hawkes process.

The R Package "stpphawkes": The package "stpphawkes" is implemented by J. Derek Tucker, Lyndsay Shand and John Lewis. The package has the function "simulate_temporal" to simulates a temporal Hawkes process with an exponential correlation function.

The R Package "surveillance": The package "surveillance" is implemented by Michael Hoehle, Sebastian Meyer and Michaela Paul [58]. The creators describe the package as statistical methods for the modeling and monitoring of time series of counts, proportions and categorical data, as well as for the modeling of continuous-time point processes of epidemic phenomena. The monitoring methods focus on aberration detection in count data time series from public health surveillance of communicable diseases, but applications could just as well originate from environmetrics, reliability engineering, econometrics, or social sciences. The package has the function "twinSIR_simulation"

to simulate a temporal Hawkes process with an exponential correlation function. The simulation algorithm used is Ogata's modified thinning algorithm.

The packages related to Hawkes process generation in Python and their simulation algorithms are as follows:

- HawkesPyLib by Simon Grimm
Ogata's modified thinning algorithm
<https://github.com/Simbold/HawkesPyLib>
- hawkes by Steven T. Morse
Ogata's modified thinning algorithm
<https://github.com/stmorse/hawkes>
- py-hawkes by Rastislav Kisel Ragoragino
Ogata's modified thinning algorithm
<https://github.com/ragoragino/py-hawkes>
- tick.hawkes by Emmanuel Bacry, Martin Bompaire, Stephane Gaïffas, Maryan Morel, Søren Vinther Poulsen
Ogata's modified thinning algorithm
<https://x-datainitiative.github.io/tick/modules/hawkes.html>
- hawkeslib by Caner Turkmen
Ogata's modified thinning algorithm
<https://github.com/canerturkmen/hawkeslib>
- Hawkes by Takahiro Omi
Ogata's modified thinning algorithm
<https://github.com/omitakahiro/Hawkes>

D.1. The R Packages Experiment Output

The experiment outputs of R packages are listed as follows:

- R Package – IHSEP output data are in Table D.1

The experiments for the R Package "IHSEP" (by Feng Chen) are performed with the following input parameters from the package description page:

Parameter setting for "IHSEP" experiments

- Base intensity, $\lambda_0(t) = \text{function}(x)\{200*(2+\cos(2*\pi*x))\}$
- Excitation function, $\varphi(t) = \text{dexp}$
- Simulation duration, $T = 60$
- Reproduction number, $\text{repr}=0.5$

The R package "IHSEP" has three functions for the Hawkes process simulation: "simHawkes0", "simHawkes1" and "simHawkes1a". In the experiments, the number of generated events and the generation time for these events are measured.

- R Package – RHawkes output data are in Table D.2

This software simulates renewal Hawkes processes. The experiments for the R Package "RHawkes" [57] are performed with the following input parameters (inter arrival times of the immigrant events are Weibull distributed):

Parameter setting for "RHawkes" experiments

- Base intensity, $\lambda_0(t) = \text{weibull}$, with $\text{shape} = 0.975$, $\text{scale} = 1$
- Excitation function, $\varphi(t) = \text{dexp}$
- Simulation duration, $T = 5000, 10000, 25000, 50000, 75000$
- Reproduction number, $\text{repr}=0.975$

The R package "RHawkes" has a function for the renewal Hawkes process simulation: "simRHawkes1". In the experiments, the number of generated events and the generation time for these events are measured.

- R Package – hawkesbow output data are in Table D.3

The experiments with the R Package "hawkesbow" [10] are performed with the following input parameters:

Parameter setting for "hawkesbow" experiments

- Base intensity, $\lambda_0(t) = \text{function}(x)\{1\}$,
- Excitation function, $\varphi(t) = \text{function}(x) \{2*\exp(-2*x)\}$,
- Simulation duration, $T = 1.000.000$
- Reproduction number, $\text{repr}=0.5$

The R package "hawkesbow" has a function for the Hawkes process simulation: "hawkes". In the experiments, the number of generated events and the generation time for these events are measured.

- R Package – hawkes output data are in Table D.4

The experiments with the R Package – hawkes [11] are performed with the following input parameters:

Parameter setting for "hawkes" experiments

- Base intensity, $\lambda_0(t) = 1$
- Excitation function, $\varphi(t) = \text{function}(x) \{1*\exp(-2*x)\}$,
- Simulation duration, $T = 1.000.000$

The R package "hawkes" has a function for the Hawkes process simulation: "simulateHawkes" In the experiments, the number of generated events and the generation time for these events are measured.

Table D.1. The speed comparison with the Hawkes process generation functions of the R-package "IHSEP"

simHawkes0		simHawkes1		simHawkes1a		rHawkes	
events	time	events	time	events	time	events	time
47,779	3.50	47,028	3.25	46,385	0.76	47,106	0.19
46,500	3.43	47,720	3.57	46,428	0.76	46,979	0.11
47,310	3.50	47,133	4.47	47,243	1.25	46,947	0.17
48,490	4.18	46,956	3.44	47,121	1.12	46,878	0.13
46,939	3.63	46,468	3.49	47,618	1.12	47,137	0.15
47,453	3.55	46,996	3.56	47,166	1.03	46,968	0.15
46,919	4.31	46,762	4.20	47,179	1.30	47,006	0.15
46,963	3.42	46,354	4.12	47,580	1.10	47,383	0.14
46,564	3.48	47,248	4.36	47,425	1.11	46,909	0.14
46,810	3.53	47,250	3.83	47,843	1.06	47,400	0.14
46,871	3.58	46,954	4.31	47,140	0.99	48,287	0.07
46,858	3.40	47,733	3.65	47,009	1.08	47,647	0.16
46,951	3.42	47,050	3.58	48,013	0.83	46,727	0.12
47,305	4.11	46,712	3.39	46,434	1.17	47,448	0.15
46,499	4.23	46,811	3.54	47,775	1.19	47,117	0.13
47,009	4.25	47,059	3.61	47,287	0.78	46,675	0.16
47,472	4.33	46,690	3.49	47,233	1.13	46,373	0.20
47,141	4.14	47,329	3.50	47,167	0.81	46,606	0.14
46,959	4.22	47,365	4.36	47,089	0.78	46,876	0.12
46,720	4.06	47,325	4.03	47,484	1.16	47,665	0.16
Average		Average		Average		Average	
47,076	3.81	47,047	3.79	47,231	1.03	47,107	0.14

Table D.2. The speed comparison with the Hawkes process generation functions of the R-package "RHawkes"

	simRHawkes1		rHawkes	
Simulation Time	events	time	events	time
5,000	181,611	2.02	222,413	0.09
5,000	214,558	3.13	203,352	0.11
5,000	203,185	2.92	190,045	0.10
5,000	227,518	3.22	225,337	0.11
10,000	420,244	4.32	391,820	0.16
10,000	370,039	3.86	387,292	0.10
10,000	400,585	4.38	391,396	0.15
10,000	442,833	4.44	370,859	0.12
25,000	972,501	10.3	1,016,943	0.20
25,000	976,369	10.08	1,018,341	0.28
25,000	972,296	11.55	966,032	0.20
25,000	925,912	10.17	959,316	0.29
50,000	1,892,007	19.67	1,954,536	0.44
50,000	1,915,358	23.12	1,911,490	0.50
50,000	2,007,282	22.27	1,973,547	0.61
50,000	1,975,733	21.39	1,954,449	0.42
75,000	2,874,727	30.25	2,877,165	0.64
75,000	2,956,647	31.81	2,974,729	0.64
75,000	2,997,265	32.08	2,922,497	0.64
75,000	3,052,554	34.02	2,938,818	0.66
	Average		Average	
	1,298,961	14.25	1,292,519	0.32

Table D.3. The speed comparison with the Hawkes process generation functions of the R-package "hawkesbow"

hawkes			rHawkes	
events	time		events	time
2,005,942	10.39		2,001,403	0.50
1,995,365	9.68		2,000,571	0.56
2,006,645	10.38		2,004,069	0.63
2,001,239	11.31		2,000,491	0.63
2,002,813	11.34		1,998,877	0.60
1,997,782	9.78		2,003,050	0.67
1,996,915	11.53		1,999,829	0.61
1,999,106	10.51		2,003,620	0.66
2,001,683	11.59		1,995,592	0.63
1,997,263	12.25		2,001,381	0.61
2,000,825	12.25		1,999,238	0.61
2,001,635	11.00		1,995,015	0.48
2,001,287	10.58		2,000,152	0.65
2,002,048	11.19		1,998,208	0.66
1,996,913	8.97		1,993,949	0.62
2,001,995	11.78		1,999,131	0.71
2,000,713	9.99		2,002,262	0.61
2,000,706	10.18		1,997,992	0.47
1,999,179	11.61		1,999,536	0.56
2,004,971	11.31		1,995,391	0.69
Average			Average	
2,000,751	10.88		1,999,488	0.61

Table D.4. The speed comparison with the Hawkes process generation functions of
the R-package "hawkes"

simulateHawkes		rHawkes	
events	time	events	time
1,999,570	2.00	2,000,629	0.56
1,999,068	2.58	1,995,067	0.67
2,000,529	2.59	1,998,014	0.47
1,997,938	2.12	2,003,412	0.50
1,999,308	2.00	1,998,282	0.52
2,003,735	2.01	1,996,912	0.48
2,002,814	2.56	2,001,975	0.61
2,000,568	1.92	2,000,362	0.47
1,997,758	1.97	1,996,342	0.49
2,001,158	2.50	1,997,213	0.51
1,999,363	1.98	2,001,789	0.48
2,001,473	2.02	2,001,850	0.53
1,993,150	2.02	1,999,512	0.48
1,998,339	1.97	1,995,607	0.53
2,004,175	1.92	1,999,552	0.50
1,995,298	1.91	1,994,228	0.46
2,001,685	2.46	2,000,233	0.48
1,998,975	2.51	2,001,224	0.42
2,001,821	2.42	1,999,681	0.45
1,995,494	2.53	1,995,200	0.49
Average		Average	
1,999,611	2.20	1,998,854	0.51

APPENDIX E: APPROXIMATING A COMPLICATED FUNCTION

E.1. Approximating a Complicated Function

In numerical analysis, interpolation is a type of estimation. Interpolation is a method of constructing finding new data points based on the range of a discrete set of known data points. A closely related problem is the approximation of a complicated function by a simple function. A few data points from the complicated function can be interpolated to produce a simpler function which is close to the complicated function. The resulting gain in simplicity may compensate the loss from interpolation error and give better performance in calculation process. Some interpolation methods in the literature [59] are as follows:

- Piece-wise constant interpolation

It is also called "nearest-neighbor interpolation". It selects the value of the nearest point and does not take into accounts the values of neighboring points at all. It returns a piece-wise-constant function. It is not very precise.

- Linear interpolation

It is a method of curve fitting using linear polynomials to construct new data points within the range of a discrete set of known data points. Linear interpolation uses a linear function between each pair of consecutive points in the data set. It is not very precise.

- Polynomial interpolation

It is a method of curve fitting using a polynomial to construct new data points within the points in the data set. Polynomial interpolation uses a single polynomial function of lowest possible degree that connects all the points in the data set.

However, high-degree polynomials can oscillate in a strange way. A minor fluctuation over a small portion of the interval can cause large fluctuations over the

entire range. Therefore, It is not very precise.

An alternative approach is to divide the approximation interval into several sub-intervals and construct different approximating polynomials on each sub interval. This is called piece-wise polynomial approximation.

- Spline interpolation

It is like linear interpolation, but spline interpolation employs low-degree polynomials in each of the intervals, instead of linear polynomials. It selects the polynomial pieces which fit smoothly together. The resulting function is called a spline.

[59] states that the most common piece-wise polynomial approximation uses cubic polynomials between each successive pair of points and is called "cubic spline interpolation". A general cubic polynomial involves four constants, so there is sufficient flexibility in the cubic spline procedure to ensure that the interpolant is not only continuously differentiable on the interval, but also has a continuous second derivative.

The definition of cubic spline interpolation [59]:

Given a function f defined on $[a, b]$ and a set of nodes $a = x_0 < x_1 < \dots < x_n = b$, a cubic spline interpolant S for f is a function that satisfies the following conditions:

- $S(x)$ is a cubic polynomial, denoted $S_j(x)$, on the sub-interval $[x_j, x_{j+1}]$ for each $j = 0, 1, \dots, n - 1$
- $S_j(x_j)' = f(x_j)$ and $S_j(x_{j+1}) = f(x_{j+1})$ for each $j = 0, 1, \dots, n - 1$
- $S_{j+1}(x_{j+1}) = S_j(x_{j+1})$ for each $j = 0, 1, \dots, n - 2$
- $S_{j+1}'(x_{j+1}) = S_j'(x_{j+1})$ for each $j = 0, 1, \dots, n - 2$
- $S_{j+1}''(x_{j+1}) = S_j''(x_{j+1})$ for each $j = 0, 1, \dots, n - 2$
- One of the following sets of boundary conditions is satisfied
 - $S''(x_0) = S''(x_n) = 0$ (natural boundary)
 - $S'(x_0) = f'(x_0)$ and $S'(x_n) = f'(x_n)$ (clamped boundary)

When the free boundary conditions occur, the spline is called a natural spline, and its graph approximates the shape that a long flexible rod would assume if forced to go through the data points $(x_0, f(x_0)), (x_1, f(x_1)), \dots, (x_n, f(x_n))$.

In general, clamped boundary conditions lead to more accurate approximations because they include more information about the function f namely the values of the derivative at the endpoints or their accurate value approximation.

E.2. Numerical Integration

The function to be integrated will typically be in one of the following three forms [59]:

- A simple continuous function such as a polynomial, an exponential, or a trigonometric function.
- A complicated continuous function that is difficult or impossible to differentiate or integrate directly.
- A tabulated function, where values of x and $f(x)$ are given at several discrete points, as is often the case with experimental or field data.

In numerical analysis, a quadrature rule approximates the definite integral of a function, usually stated as a weighted sum of function values at specified points within the domain of integration.

The Newton-Cotes formulas (also called the Newton–Cotes quadrature rules) are the most common numerical integration (also called the Newton–Cotes quadrature rules) schemes. They evaluate the integrand at equally spaced points. They are based on the strategy of replacing a complicated function or tabulated data with an approximating function that is easy to integrate

$$I = \int_a^b f(x)dx \cong \int_a^b f_n(x)dx \quad (\text{E.1})$$

where $f_n(x)$ is a polynomial of the form

$$f_n(x) = a_0 + a_1x + a_2x^2 + \dots + a_nx^n \quad (\text{E.2})$$

and n is the order of the polynomial.

The Newton-Cotes formulas are classified as closed and open forms. The closed Newton-Cotes formulas include the endpoints of $[a, b]$ as nodes. However, the open Newton-Cotes formulas do not include the endpoints of $[a, b]$ as nodes.

Closed Newton-Cotes Formulas. The $(n+1)$ point closed Newton-Cotes formula uses nodes

$$x_i = x_0 + ih \text{ for } i = 0, 1, \dots, n \quad (\text{E.3})$$

where $x_0 = a$, $x_n = b$ and $h = (b - a)/n$ (equal distances between nodes).

Some of the common closed Newton-Cotes formulas are below.

- $n = 1$: Trapezoidal rule

$$\int_{x_0}^{x_1} f(x)dx \cong \frac{h}{2} [f(x_0) + f(x_1)] \quad (\text{E.4})$$

$f(x)$ is approximated with a first-order Lagrange polynomial

- $n = 2$: Simpson's 1/3 rule

$$\int_{x_0}^{x_2} f(x)dx \cong \frac{h}{3} [f(x_0) + 4f(x_1) + f(x_2)] \quad (\text{E.5})$$

$f(x)$ is approximated with a second-order Lagrange polynomial.

- $n = 3$: Simpson's 3/8 rule

$$\int_{x_0}^{x_3} f(x)dx \cong \frac{3h}{8} [f(x_0) + 3f(x_1) + 3f(x_2) + f(x_3)] \quad (\text{E.6})$$

- $n = 4$: Boole's rule

$$\int_{x_0}^{x_4} f(x)dx \cong \frac{3h}{8} [7f(x_0) + 32f(x_1) + 12f(x_2) + 32f(x_3) + 7f(x_4)] \quad (\text{E.7})$$

Newton–Cotes formulas can be useful if the value of the integrand at equally spaced points is given. If it is possible to change the points at which the integrand is evaluated, then other methods such as Gaussian quadrature and Clenshaw–Curtis quadrature are probably more suitable (the points are not equally spaced).

Gauss–Lobatto rules. It is also known as Lobatto quadrature. It is like Gaussian quadrature with the following differences:

- The integration points include the end points of the integration interval.
- It is accurate for polynomials up to degree $2n - 3$, where n is the number of integration points (Quarteroni, Sacco and Saleri 2000).

Lobatto quadrature of function $f(x)$ on interval $[-1, 1]$ is expressed as

$$\int_{-1}^1 f(x) dx \cong \frac{1}{10}f(-1) + \frac{49}{90}f\left(-\sqrt{3/7}\right) + \frac{32}{45}f(0) + \frac{49}{90}f\left(\sqrt{3/7}\right) + \frac{1}{10}f(1) \quad (\text{E.8})$$

The area under the intensity function and the inputs for spline interpolation (nodes for the cumulative intensity function and inverse cumulative intensity function) are computed using the following methods:

- Simpson's 1/3 rule
- Boole's rule
- Gauss-Lobatto rule with $n=5$

The Boole's rule is selected for the area and the nodes computation because of its error function compared to Simpson's rule and its equally spaced interval compared to Gauss-Lobatto rule.

The number of sub-intervals for the numerical integration can be determined by the following ways:

- It can be defined by the user as an input
- It can be defined by tolerance value
- It can be defined using the property that the area under the intensity is convergent sequence from iteration to iteration of the numerical integration method.

The first way is simple. The number of sub intervals are defined by user an input. The stopping criteria for second way is that the area difference between successive iterations is smaller than the tolerance value.

For the third way, the area under the intensity function is a convergent sequence according to the Criterion for Convergence.

Criterion for Convergence ([60], p. 84, Theorem 6.1.2)

Suppose X is a metric space, $z \in X$ and (x_n) is a sequence in X . The following statement holds:

The distance criterion: $\text{distance } (z, x_n \mid n \in S) = 0$ for every infinite subset S of $\mathbb{N}$.

The stopping condition is the iteration where the difference of the area changes its sign. After this point, as the number of sub intervals increases, the area under the intensity function computed by numerical integration is diverging from the true value. This is

due to the intrusion of round-off error because of the great number of computations for these many segments. Thus, the level of precision is limited, and we would never reach the exact result obtained analytically ([61]).

The third way is selected to compute the area under the intensity function and the nodes for the spline interpolation.

For the spline interpolation, the R function `splinefun` from the package `stats` that performs cubic (or Hermite) spline interpolation of given data points, returning a function performing the interpolation.

The area under the intensity function and the nodes of the cumulative intensity function and of its inverse function are calculated with the following numerical integration methods and cubic spline interpolation:

- Numerical integration using Simpson's rule 1/3 with cubic spline interpolation
 - The number of sub-intervals is an input
- Numerical integration using Gauss-Lobatto rules with $n=5$ with cubic spline interpolation
 - The number of sub-intervals is an input
- Numerical integration using Boole's rule with cubic spline interpolation
 - With tolerance limit for the area difference for iterations
 - Using convergence criteria for the area

The following intensity functions are used

- Exponential decay, $\lambda_0(t) = 3.9e^{-4t}$
Simulation Duration, $T=1$
- Power law function, $\lambda_0(t) = 2000(t + 250)^{-(2.33)}$
Simulation Duration, $T=1000$

These intensity functions are selected because the closed form of cumulative and inverse cumulative function of them are available for the error calculations. The following criteria are measured:

- The ratio

$$\frac{|area\ of\ analytical\ integration - area\ of\ numerical\ integration|}{area\ of\ analytical\ integration} \quad (E.9)$$

- The maximum x-error of the inverse cumulative intensity function

$$x - error = \varepsilon_x(x_{exact}) = \max |x_{exact} - x_{approximated}| \quad (E.10)$$

- The maximum y-error of the inverse cumulative intensity function

$$y - error = \varepsilon_y(x_{exact}) = \max |y_{exact} - y_{approximated}| \quad (E.11)$$

To compute the maximum x-error and y-error, the midpoints of the sub intervals are selected as exact x values because the end points of the sub intervals are used as input to the spline interpolation. The formulas of exact and approximated x and y values are as follows:

$$x_{exact} = \text{midpoints of sub intervals}, \quad (E.12)$$

$$y_{exact} = \Lambda_0(x_{exact}), \quad (E.13)$$

$$x_{approximated} = \Lambda_0^{-1}(\Lambda_0(x_{exact})), \quad (E.14)$$

$$y_{approximated} = \Lambda_0(x_{approximated}). \quad (E.15)$$

The results are as follows:

- Numerical integration using Simpson's rule $1/3$ with cubic spline interpolation
 - The number of sub-intervals is an input

Table E.1. Experiment results with numerical integration using Simpson's rule $1/3$ with cubic spline

Exponential decay				
n	area diff/area exact		Y-error	X-error
1	0.05917057		0.3361596	0.3315913
10	8.846738e-06		0.006990225	0.0022666598
100	8.888465e-10		7.135905e-05	1.926083e-05
1000	8.896703e-14		7.137479e-07	1.839832e-07
10000	0		7.137495e-09	1.831097e-09
Power law				
n	area diff/area exact		Y-error	X-error
1	0.3392383		0.1722926	149.9064
10	0.0003914023		0.01802908	2.440999
100	4.49302e-08		0.0002199467	0.03759026
1000	4.500141e-12		2.206241e-06	0.0004211675
10000	3.880993e-16		2.206311e-08	4.258923e-06
100000	0		2.206313e-10	4.26367e-08

- Numerical integration using Gauss–Lobatto rules with $n=5$ with cubic spline interpolation
 - The number of sub-intervals is an input

Table E.2. Experiment results with numerical integration using Gauss–Lobatto rules with cubic spline interpolation

Exponential decay				
n	area diff/area exact		Y-error	X-error
1	0.817358e-05		0.3644635	0.3807723
10	4.5817443-13		0.006991726	0.002764928
100	0		7,135907e-05	1.927252e-05
1000	0		7.137479e-07	1.839844e-07
10000	0		7.137495e-09	1.831097e-09
Power law				
n	area diff/area exact		Y-error	X-error
1	0.01167027		0.312852	360.3391
10	3.134569e-08		0.01818982	1.083608
100	3.880993e-16		0.0002199505	0.03759102
1000	0		2.206241e-06	0.0004211675
10000	0		2.206311e-08	4.258923e-06
100000	0		2.206313e-10	4.26367e-08

- Numerical integration using Boole's rule with cubic spline interpolation
 - With tolerance limit for the area difference for iterations

Table E.3. Experiment results with numerical integration using Boole's rule with cubic spline interpolation with tolerance limit for the area difference for iterations

Exponential decay				
tolerance area diff	n	area diff/area exact	Y-error	X-error
1e-12	116	3.555781e-12	0.02080126	0.007742518
1e-14	216	8.537123e-14	0.01137508	0.004126361
1e-16	432	1.391922e-15	0.005747695	0.002054
1e-26	432	1.391922e-15	0.005747695	0.002054
1e-36	432	1.391922e-15	0.005747695	0.002054
Power law				
tolerance area diff	n	area diff/area exact	Y-error	X-error
1e-12	320	1.372423e-11	0.01006733	2.768313
1e-14	612	2.815014e-13	0.005346586	1.445534
1e-16	1064	1.021995e-14	0.003097837	0.8309324
1e-26	1064	1.021995e-14	0.003097837	0.8309324
1e-36	1064	1.021995e-14	0.003097837	0.8309324

- Numerical integration using Boole's rule with cubic spline interpolation
 - With convergence criterion for the area

Table E.4. Experiment results numerical integration using Boole's rule with cubic spline interpolation with convergence criterion

Exponential decay				
n	area diff/area exact		Y-error	X-error
424	1.507916e-15		0.005854979	0.002092931
Power law				
n	area diff/area exact		Y-error	X-error
1064	1.021995e-14		0.003097837	0.8309324

As a result, Boole's results performs better than Simpson's rule because the interval is divided more sub intervals. The convergence property of the area under the intensity function is in an improvement to the tolerance limit because beyond a certain tolerance limit, decreasing the tolerance does not provide any improvement. The results of the numerical integration using Boole's rule with cubic spline interpolation show this result. The explanation of this result is that this is due to the intrusion of round-off error because of the great number of computations for these many segments as stated above.

The Boole's rule and the Simpson's rule are members of the Newton–Cotes quadrature rules. Their common property is that they have equal distances between nodes for numerical integration. However, Gauss–Lobatto rules for numerical integration have unequal distances for numerical integration. They perform better numerical integration for the function in this study with fewer nodes. The next step is combining Gauss–Lobatto rules with convergence criterion.

APPENDIX F: THE FUNCTIONS OF THE R PACKAGE RUNURAN

Beside selecting a better process generation for the algorithm, a good random variate generation method improves the performance of the algorithm. General methods for random variate generation in the literature are [39]

- Inverse transform method
- Acceptance rejection method
- Composition method
- Relocate and re-scale method

The factors to be considered for the selection of random variate generation method are

- Exactness
Distribution of generated random variates has the targeted exact form
- Speed
Computing time required to generate variate.
- Space
Computer memory required for the random variate generator
- Simplicity
Algorithmic simplicity and implementation simplicity

As random variate generator, the R package "Runuran" [49] is used in the implementation of the Hawkes process simulation code. It computes the inverse cumulative function and the area under the function (which are the set ups of the algorithm) at the desired level of accuracy for a smooth continuous function. The R Package has two functions which are designed to generate Random Variates (RVG). These functions are

- “pinv.new” (Polynomial interpolation of INVerse CDF) and
- “uq” (computes quantile of the function).

The approximate inverse cumulative distribution function, F_a^{-1} is computed by calling the function “pinv.new”. The function “pinv.new” computes:

- first the cumulative distribution function(CDF) using Gauss-Lobatto integration method and
- then F_a^{-1} using the Newton’s divided-difference interpolation between carefully selected nodes.

The quantiles of the approximate inverse CDF are computed for a given (vector of) probabilities with a given resolution value by calling the function “uq”.

The NHPP can be simulated using these functions because $\lambda(t) > 0$ and $\Lambda(t)$ is a multiple of cumulative distribution function.

F_a^{-1} is computed using the inputs:

- the intensity function of NHPP, $\lambda(t)$
- the NHPP simulation period $(0, t_{sim})$ as lower and upper bound of the $\lambda(t)$ ’s domain, and
- the u-resolution, the maximum acceptable error for the approximate inverse CDF.

Given the intensity function of NHPP, $\lambda(t) > 0$, $\forall t$, total area under the intensity function is calculated by the formula

$$Total \ Area = A_{0,t_{sim}} = \int_0^{t_{sim}} \lambda(t) dt > 0. \quad (F.1)$$

We get the density $f(t)$ of the first event of the NHPP easily rescaling the intensity function such that it has area one:

$f(t) = \frac{1}{A_{0,t_{sim}}} \lambda(t)$, for $t < t_{sim}$ because

- $\frac{1}{A_{0,t_{sim}}} \lambda(t) > 0, \forall t$
- and
- $\int_0^{t_{sim}} \frac{1}{A_{0,t_{sim}}} \lambda(t) dt = 1$

The cumulative distribution function (CDF) is expressed as

$$F(t) = \int_0^t \frac{1}{A_{0,t_{sim}}} \lambda(x) dx. \quad (F.2)$$

Thus $F_a^{-1}(\cdot)$ is the inverse CDF of the event times.

The mathematical details are as follows: The area under the intensity function between two consecutive events is exponentially distributed with mean one in NHPP can be written as

$$\int_{s_n}^{s_n+T} \lambda(x) dx = e. \quad (F.3)$$

The n^{th} event time satisfies the following condition

$$\int_0^{S_n} \lambda(x) dx = \sum_{i=1}^n e_i. \quad (F.4)$$

$\lambda(t)$ is standardized by re-scaling $\frac{1}{A_{0,t_{sim}}}$. Equation F.4 turns to

$$\int_0^{S_n} \frac{1}{A_{0,t_{sim}}} \lambda(x) dx = \sum_{i=1}^n \frac{1}{A_{0,t_{sim}}} e_i. \quad (F.5)$$

$f(t) = \frac{1}{A_{0,t_{sim}}} \lambda(t)$ is the probability density function obtained by re-scaling the intensity function of the NHPP.

$F(t) = \int_0^t \frac{1}{A_{0,t_{sim}}} \lambda(x) dx$ is the cumulative distribution function (CDF).

$F_a^{-1}(\cdot)$ is the approximate inverse cumulative distribution function. It is calculated

by calling the function, “pinv.new”. The quantiles of the approximate inverse CDF returns the n^{th} event time using the inputs, F_a^{-1} and $\sum_{i=1}^n \frac{1}{A_{0,t_{sim}}} e_i$. The occurrence time of the n^{th} event is expressed as

$$F_a^{-1}\left(\sum_{i=1}^n \frac{1}{A_{0,t_{sim}}} e_i\right) \quad (\text{F.6})$$

The approximation error of F_a^{-1} is controlled by means of “uresolution” that is maximal acceptable u-error. The u-error is $|U - F(F_a^{-1}(U))|$. Selecting the “uresolution” smaller results in a more precise approximation F_a^{-1} . In addition the number of subintervals for the Newton’s interpolation increases. As a result, the setup time of the function “pinv.new” increases. Therefore, the “uresolution” value should be selected carefully.

The experiments are performed about the speed of these two methods. The spline interpolation is faster. But the UNU.RAN Random Variate Generators is more precise. This is due to the number sub intervals.

The Errors. As the numerical inversion method, the R package, Runuran (R Interface to the ‘UNU.RAN’ Random Variate Generators) [49] is selected because its u-error is less than 2.3×10^{-10} that is machine precision (by [62]).

The u-error is defined as follows: The random variate, X can be generated using F_a^{-1} if F^{-1} does not exist or is computationally difficult and/or inefficient. The results are approximated. Therefore, the amount of error should be investigated to see the performance of the F_a^{-1} .

There are two types of error,

- x-error and
- u-error.

Given the distribution function, F and U (uniform $U(0,1)$), the x-error between F^{-1} and F_a^{-1} is in Equation F.7 :

$$x - error = \varepsilon_x(u) = | F^{-1}(u) - F_a^{-1}(u) | \quad (F.7)$$

A better alternative is the u-error. The concept “u-error” that is well suited for the inversion method is expressed as

$$u - error = \varepsilon_u(u) = | u - F(F_a^{-1}(u)) | \quad (F.8)$$

The Functions PINV and uq of the Runuran are used in implemented package fastHawkes. The function PINV (pinv.new), polynomial interpolation-based inversion of CDF. PINV is a variant of numerical inversion, where the inverse CDF is approximated using Newton’s interpolating formula. As inputs, it requests the pdf, its lower and upper bound and the requested error level. The function pdf must be positive but need not be normalized (i.e., it can be any multiple of a density function). This function creates an unuran object based on ‘PINV’ (Polynomial interpolation of IN-Verse CDF). It can be used to draw samples of a continuous random variate with given probability density function pdf by means of ur. It also allows to compute quantiles by means of uq. The unuran object contains the area under the function pdf.

The set up of the proposed algorithm needs the followings:

- Total area under the intensity function for the interval $(0, T)$
- The inverse cumulative intensity function to compute the quantiles.

The function ”pinv.new” computes the required values for the simulation stated above. It is a relatively time consuming but precise (its u-error is than machine precision) function because of its set up time. But this function is executed only once

during the simulation period. The set-up time is independent of the number of events "n" generated. An example for the inverse cumulative intensity function computation command is as follows:

```
Lam0Inv <- Runuran::pinv.new(pdf=lam0, lb=0, ub=T,uresolution=1e-010)
```

An example for the area under the intensity function computation command is as follows:

```
UNU.RAN.object.Lam0Inv <- Runuran::unuran.details(Lam0Inv,show=FALSE, return.list=TRUE)
```

```
area.base <- UNU.RAN.object.Lam0Inv$area.pdf
```

The sub-processes of the algorithm need the inter-event times between parent event and its offspring. The function "uq" computes the inter-event times between parent event and its offspring events for the sub-processes. An example for the inter-event time computation command is as follows:

```
Runuran::uq(Lam0Inv, runif(1 ,min = 0, max = 1))
```

The upper bound of the uniform random number is 1 instead of $\Lambda(T)$ because the function pdf=lam0 is normalized to compute inverse cumulative density function.