ECMM450 Stochastic Processes
Simulation of Non-Homogeneous Poisson Processes

70054986
Department of Computer Science, University of Exeter
(Dated: April 7, 2023)

This project is about Non-Homogeneous Poisson processes and how to simulate them. We will be reviewing the thinning algorithm of Lewis and Shedler (1979) for simulating NHPP.

I. INTRODUCTION

This report will be loosely structured according to the following points:

1. Explanataion of what is meant by a Non-Homogeneous Poisson Process (NHPP) accompanied by clear mathematical definition.
2. Review of the thinning algorithm of Lewis and Shedler (1979) for simulating NHPP. Short description of the algorithm explaining briefly why it works, and its main benefits compared to other approaches.
3. Simulation of occurrence of 1000 successive events $t_1, t_2, ..., t_{1000}$ for a homogeneous Poisson process having a rate of 8 events per year. Visual representation of the same.
4. Considering a NHPP that has a rate function that increases smoothly from 1 event per year at $t = 0$ by 1% per year, i.e. $(t) = (1.01)^t$. Determining whether the thinning algorithm can be used to simulate this process from the previous homogeneous Poisson process data.
5. Code to perform the thinning algorithm and use it to find occurence times for a NHPP having the rate function $\lambda(t) = (1.01)^t$.
6. Make a figure showing $N(t)$ versus t for your NHPP simulation and compare it to what was shown in the figure for the homogeneous Poisson process. By integrating the rate function, add a line to your figure showing the expectation $E[N(t)]$ versus t.

II. DESCRIPTION AND MATHEMATICAL DEFINITION

The following section details what is meant by a Non-Homogeneous Poisson Process (NHPP) giving a clear mathematical definition.

A non-homogeneous Poisson process can be thought of as a generalization of the homogeneous Poisson process, in that, as opposed to its homogeneous counterpart where rate of occurrence of events is constant (denoted by λ), here the rate is a function of time, denoted by $\lambda(t)$. Thus, the number of occrrences in the interval $(0, T]$ follows Poisson distribution $Pois(\int_0^T \lambda(t)dt)$. More formally, allowing the rate parameter to vary with time results in the following definition.

Definition II.1 (Ross, 2009, p.339, Definition 5.4)**.** The counting process $N(t), t \geq 0$ is said to be a nonhomogeneous Poisson process with intensity function $\lambda(t) \geq 0, t \geq 0$, if

1. $N(0) = 0$.
2. The process has independent increments.
3. $PN(t + h) - N(h) = 1 = \lambda(t)h + o(h)$.
4. $PN(t + h) - N(h) \geq 2 = o(h)$.

where $o(h)$ denotes higher order terms of h

Time sampling an ordinary Poisson process (with constant rate λ) results in a Non-Homogeneous Poisson process. Given $\{N(t), t \geq 0\}$, a Poisson process wih rate λ, if the event occurring at time t is counted with probability $p(t)$, then $\{N_c(t), t \geq 0\}$ is a Non-Homogeneous Poisson process. ¡Insert reference¿

III. REVIEW OF THE THINNING ALGORITHM

1. Some other other algorithms

There exist other algorithms for simulating Poisson processes, some of which we will discuss now.

1. Time-scale transformation of a homogeneous Poisson process via inverse of the integrated rate function $\Lambda(x)$
2. Generate intervals between the points individually
3. Order statistics from Poisson variates
4. Log-linear rate function

2. The thinning algorithm

To construct a Non-Homogeneous Poisson process $\{N(t), t \geq 0\}$, with rate parameter $\lambda(t)$, over the interval $(0, T]$, the algorithm starts with a Non-Homogeneous

Poisson process $\{N^*(t), t \geq 0\}$, with rate parameter $\bar{\lambda}(t)$ that dominates the set $\lambda(t)$ for all $t \in (0, T]$, that is

$$\bar{\lambda}(t) \geq \lambda(t) \forall t \in (0, T]$$
$$\bar{\lambda}(t) = sup_{t \in (0,T]} \lambda(t)$$

Then, for all t, the point from the dominating NHPP is retained with probability $\lambda(t)/\bar{\lambda}$. The remaining points form a NHPP with rate parameter $\lambda(t)$. It is noted that since points are deleted independently, the number of points in $\{N(x) : x \geq 0\}$ in any set of non-overlapping intervals are mutually independent.

Algorithm 1: (Lewis and Shedler, 1979, p.7, Algorithm 1) Simulation of an Inhomogeneous Poisson Process with Bounded Intensity Function $\lambda(t)$, on $[0, T]$

Input: λ, T
Initialize $n = m = 0$, $t_0 = s_0 = 0$,
$\bar{\lambda}(t) = sup_{t \in (0,T]} \lambda(t)$;
while $s_m < T$ **do**
  Generate $u \sim$`uniform(0,1)`;
  Let $w = -ln(u)/\bar{\lambda}$;
  Set $s_{m+1} = s_m + w$;
  Generate $D \sim$ `uniform(0,1)`;
  if $D < \lambda(s_{m+1})/\bar{\lambda}$ **then**
    $t_{n+1} = s_{m+1}$;
    $n = n + 1$;
  end
  $m = m + 1$
end
if $t_n \leq T$ **then**
  return $\{t_k\}_{k=1,2,\ldots,n}$
else
  return $\{t_k\}_{k=1,2,\ldots,n-1}$
end

3. *Benefits over other algorithms*

The paper discusses a few other methods with which to simulate NHPP. However, each method entails drawbacks with respect to computational efficiency. in its simplest implementation (¡insert reference¿), the thinning method obviates the need for numerical integration of the rate function, for ordering of points, and for generation of Poisson variates.

IV. SIMULATION OF A HOMOGENEOUS POISSON PROCESS

Uniform random numbers are used to generate Poisson variates by using the following algorithm, that can be found in ¡Insert reference¿. Full implementation in Python can be found in the Appendix. Figure 1 shows the result of simulating a homogeneous Poisson process.

Algorithm 2: Simulation of a Homogeneous Poisson Process with Rate λ, on $[0, T]$ ¡Insert reference to Yuanda Chen¿

Input: λ, N
Initialize $n_0 = 0$, $t_0 = 0$;
while $True$ **do**
  Generate $u \sim$`uniform(0,1)`;
  Let $w = -ln(u)/\bar{\lambda}$;
  Set $t_{n+1} = t_n + w$;
  if $n + 1 > N$ **then**
    return $\{t_k\}_{k=1,2,\ldots,n}$
  else
    Set $n = n + 1$;
  end
end

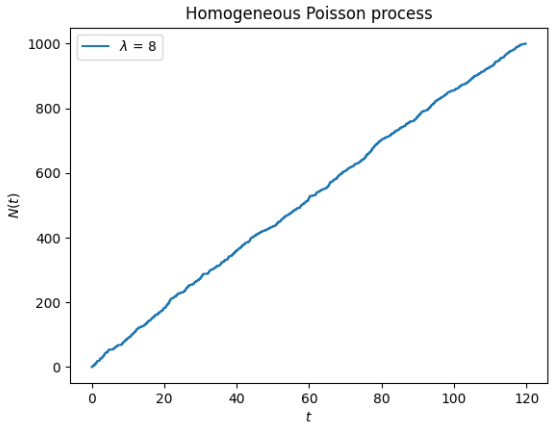

FIG. 1. Simulation of the time of occurrence of 1000 successive events for a homogeneous Poisson process having a rate of 8 events per year. $N(t)$ denotes the number of events occurring in the time period $[0, t]$.

The observation is made that the graph appears relatively linear. This would be expected as the rate of occurrence of events remains constant over any interval of time (a, b) and hence the slope corresponding to the number of events over time (i.e. rate) also remains the same.

V. NHPP WITH A SMOOTHLY INCREASING RATE FUNCTION

A NHPP is considered with a rate function that increases smoothly from 1 event per year at $t = 0$ by 1% per year, i.e. $\lambda(t) = (1.01)^t$. Looking at the previous homogeneous Poisson process from IV , it can be seen that the maximum time taken to accumulate 1000 events is $\tilde{1}40$. Figure 2 shows this. Therefore, the maximum rate for the NHPP would be $(1.01)^{140} = 4.027$. Recall that the rate for the previous homogeneous Poisson process was 8. Hence, this is within the bounds of the original process. Thinning can be applied to simulate the NHPP from the

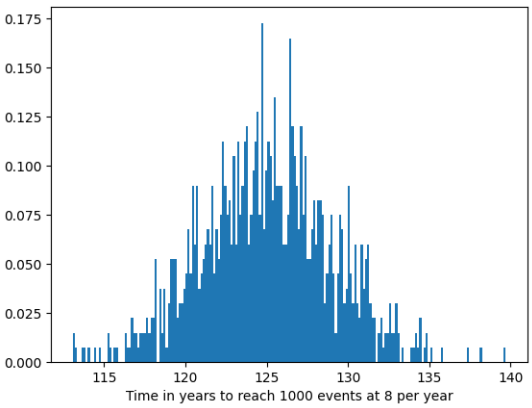

FIG. 2. Time in years to accumulate 1000 events at 8 per year

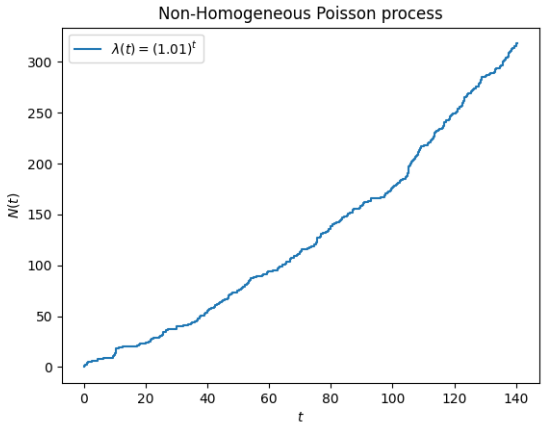

FIG. 3. Accumulated events following a NHPP with rate parameter $\lambda(t) = 1.01^t$

previous HPP data.

VI. PERFORM THE THINNING ALGORITHM WITH DATA FROM HOMOGENEOUS POISSON PROCESS

Using the thinning algorithm, the data points from the previous homogeneous process are time-sampled to construct a Non-Homogeneous process.

VII. GRAPHICAL COMPARISON OF HPP AND NHPP

Figure 3 showcases the cumulative events $N(t)$ against time t. It is observed that, as opposed to homogeneous Poisson process, the line is not quite linear - the slope increases as time progresses. This can be expected as the rate parameter $\lambda(t) = (1.01)^t$ also increases with time. Due to deletion of points from the original homogeneous process, it is also noted that the total number of points has decreased from 1000 to just over 300.

By integrating the rate function, we get

$$\int_0^t \lambda(t)dt = \int_0^t 1.01^t dt$$
$$= 1.01^t/ln(1.01) + C$$

Noting the boundary condition that at $t = 0$, $E[N(t)] = 0$, we have $C = \frac{1.01^0}{ln(1.01)} = 100.499$. Thus, $E[N(t)] = \frac{1.01^0}{ln(1.01)} - 100.499$. Plotting this onto the earlier graph, Figure 4 is obtained. It can be seen that the orange line closely follows the blue stepped graph.

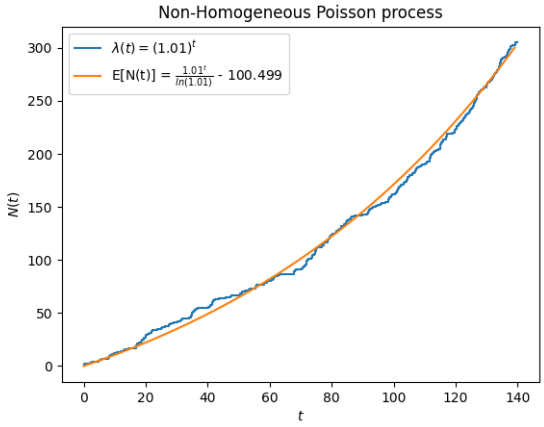

FIG. 4. $E[N(t)]$, denoted by $\frac{1.01^0}{ln(1.01)} - 100.499$ superimposed on the NHPP

ACKNOWLEDGMENTS

We wish to acknowledge the support of the author community in using REVTEX, offering suggestions and encouragement, testing new versions, ...

Appendix A: Code

Below is the code for section IV. It simulates a homogeneous Poisson process with rate parameter $\lambda = 8$ per year. Time to 1000 events is simulated and the graph generated from this code is Figure 1.

```
# Imports
import numpy as np
import matplotlib.pyplot as plt

# define hpp function
```

```
def gen_hpp(lmbda ,  N):
    '''
    param lmbda: rate parameter
    N: Number of events
    '''
    # inits
    t = [0]

    # begin loop
    while True:
        # generate uniform r.v. ~ Unif[0,1]
        u = np.random.uniform(0,1)
        # generate w ~ Exponential(lambda)
        w = - np.log(u)/lmbda
        t.append(t[-1] + w)
        # exit condition
        if len(t) > N:
           # get time to event & count the
           # number of events
           return t , np.arange(len(t))

if __name__ == '__main__': # main namespace
    l , N = 8 , 1000
    # generate the time(s) to event(s)
    # AND count of events
    hpp_event_times , events = \
    gen_hpp(lmbda = l , N = N)
    print(events , hpp_event_times) # debug

    # Make plots
    fig , ax = plt.subplots()
    # step graph
    ax.step(hpp_event_times , events ,
     label = f"$\lambda$ = {l}" , lw = 0.5)
    ax.set_xlabel(r'$t$')
    ax.set_ylabel(r'$N(t)$')
    ax.set_title('Homogeneous Poisson process')
    ax.legend(loc='best')
    plt.show()
```

Below is the code for section VI. It simulates a Non-Homogeneous Poisson process with rate parameter $\lambda(t) = 1.01^t$ that smoothly increases from 1 event per year. The time boundary is kept at 140 years as this is roughly the longest time it took the HPP to reach 1000 events. The graphs generated from this code are Figure 3 and Figure 4. In the latter, $E[N(t)]$ is shown along with the realizations from the NHPP to show that both the lines adhere to each other.

```
# Imports
import numpy as np
import matplotlib.pyplot as plt

# define hpp function
def gen_nhpp(lmbda_bar = 8 ,  T = 140):
    '''
    param lmbda_bar: rate parameter that
    dominates the rate param of the NHPP
    T : maximum time
    '''
    # inits
    s = [0]
    t = [0]

    # begin loop
    while s[-1] < T:
        # generate uniform r.v. ~ Unif[0,1]
        u = np.random.uniform(0,1)
        # generate w ~ Exponential(lambda)
        w = - np.log(u)/lmbda_bar
        s.append(s[-1] + w)
        # geenrate D ~ Unif[0,1]
        D = np.random.uniform(0,1)
        # acceptance criterion
        if D < (1.01)**s[-1] / lmbda_bar:
            t.append(s[-1])

        if t[-1] > T:
            num_events = np.arange(len(t[:-1]))
            print(f'Breakpoint 1: the number of events \
            is {num_events[-1]}, and the time taken to \
            reach them is {t[:-1][-1]}')
            # get time to event & count the
            # number of events
            return t[:-1] , num_events
    else:
        num_events = np.arange(len(t))
        print(f'Breakpoint 2: the number of events is \
        {num_events[-1]}, and the time taken to reach \
        them is {t[-1]}')
        # get time to event & count the number
        # of events
        return t , num_events

# main namespace
if __name__ == '__main__':
    # generate the time(s) to event(s)
    # AND count of events
    nhpp_event_times , events = gen_nhpp()
    # Make plots
    fig , ax = plt.subplots()
    # step graph
    ax.step(nhpp_event_times , events ,
            label = f'$\lambda(t) = (1.01)^t$')
    # Integrating the rate function to get E[N(t)]
    x = np.arange(140)
    y = 1.01**x/np.log(1.01) - \
    (1.01)**0/np.log(1.01)
    ax.plot(x , y ,
     label = r'E[N(t)] = $\frac{1.01^t}\
            {ln(1.01)}$ - 100.499')
    # Auxiliaries
    ax.set_xlabel(r'$t$')
    ax.set_ylabel(r'$N(t)$')
    ax.set_title('Non-Homogeneous \
                 Poisson process')
```

```
    ax.legend(loc='best')
```

```
plt.show()
```

---

[1] S. M. Ross, *Introduction to Probability Models* (Academic Press, 1997).

[2] P. A. Lewis and G. S. Shedler, Simulation of nonhomogeneous poisson processes by thinning, Naval Research Logistics Quarterly **26**, 403–413 (1979).

[3] Y. Chen, Thinning algorithms for simulating point processes.