Processo de Poisson

Um processo de Poisson é modelado como uma sequência de variáveis aleatórias independentes $X_1, X_2, \cdots$ com distribuição de exponencial com parâmetro λ.Duas distribuições de probabilidade são importante no processos de Poisson: $N_S\left(x\right)$ e $S_N\left(t\right)$.
N_S é a variável aleatória que conta a quantidade de eventos que ocorrem no intervalo de tempo S ($S = t_2 - t_1$). Sua função de distribuição de probabilidade é dada por:

$$P\left[N_S = x\right] = e^{-\lambda S}\frac{\left(\lambda S\right)^x}{x!}$$

S_N é a variável aleatória que mede o tempo decorrido até a ocorrência do próximo evento no processo de Poisson. Sua função de distribuição acumulada é dada por:

$$P\left[S_N \leq x\right] = 1 - \sum_{j=0}^{N-1} e^{-\lambda x}\frac{\left(\lambda x\right)^j}{j!}$$

Importação das bibliotecas necessárias

In [1]:
```
import scipy.stats as st
import time
import numpy as np
```

Simulação da função de probabilidade de N_S

Simulação interativa

A função poissonNS_I simula a função de probabilidade de N_S com um algoritmo interativo. Recebe como argumento x, lambda, t1, t2 e nSim, e calcula a probabilidade de ocorrência de x eventos no intervalo entre t1 e t2. O argumento lbda é o parâmetro das exponenciais que definem o processo de Poisson. t1 e t2 definem o intervalo S. O parâmetro nSim é a quantidade de vezes que a simulação é executada.

In [2]:
```
def poissonNS_I(x, lbda, t1, t2, nsim):
    mu = 1/lbda
    deuCerto = 0
    for i in range(nSim):
        tempo = 0
        nEventos = 0
        while tempo <= t2:
            if (tempo >= t1):
                nEventos = nEventos + 1
            # sorteia variável exponencial e acumula em tempo
            tempo = tempo + st.expon.rvs(0, mu)
        # se quantidade de eventos = x, deuCerto
        if nEventos == x:
            deuCerto = deuCerto + 1
    return(deuCerto/nSim)
```

Os comandos abaixos simulam $nSim$ observações de um processos de Poisson. Imprime a proporção de vezes que o número de eventos entre t1 e t2 é igual a x (probabilidade simulada). Imprime o valor previsto pela teoria (probabilidade teórica). Imprime o tempo de simulação (em segundos).

In [3]:
```
t1 = 15
t2 = 25
S = t2-t1
lbda = 0.2
nSim = 50000
x = 3
probT = st.poisson.pmf(x, lbda*S)
inic = time.perf_counter()
probS = poissonNS_I(x, lbda, t1, t2, nSim)
fim = time.perf_counter()
print('Probabilidade simulada:  {:.4f}'.format(probS))
print('Probabilidade teórica: {:.4f}'.format(probT))
print('Tempo de simulação iterativa: {:.4f}'.format(fim-inic))
```

```
Probabilidade simulada:  0.1810
Probabilidade teórica: 0.1804
Tempo de simulação iterativa: 6.3121
```

Simulação vetorial

A função poissonNS_V simula a função de probabilidade de N_S com um algoritmo vetorial. Recebe como argumento x, lambda, t1, t2 e nSim, e calcula a probabilidade de ocorrência de x eventos no intervalo entre t1 e t2. O argumento lbda é o parâmetro das exponenciais que definem o processo de Poisson. t1 e t2 definem o intervalo S. O parâmetro nSim é a quantidade de vezes que a simulação é executada. No algoritmo vetorial precisamos sortear todas as exponenciais e colocar em uma matriz X, que terá $nSim$ linhas, cada linha contando uma observação do processo de Poisson, ou seja, os valores acumulados de tempo das variáveis aleatórias exponenciais.
A maior dificuldade é determinar a quantidade de colunas da matriz X. O que sabemos é que deveriamos acumular uma quantidade suficiente para ultrapassar t_2. Sabemos que média dos valores dos tempos sorteados será $\mu = 1/\lambda$. Precisaremos, em média, $N = ceil\left(t_2/\mu\right)$ lançamentos para chegar a t_2, onde $ceil$ é o arredondamento para cima. Um valor empírico aproximado para a quantidade de colunas necessárias é $m = ceil\left(2 \cdot t_2/\mu\right)$.
A seguir vamos testar se esse algoritmo está correto para gerar as observações necessárias

- Sortear a matriz X com n linhas e m colunas
- Acumular os valores das exponenciais em cada linha gerando os eventos

Execute os comandos a seguir para verificar se a matriz está sendo gerada de acordo.

In [4]:
```
n = 5
t1 = 10
t2 = 30
lbda = 0.2
mu = 1/lbda
m = int(np.ceil(2*t2/mu))
X = np.cumsum(np.random.exponential(mu, (n, m)), 1)
print(X)
eventos = np.count_nonzero((X>t1) & (X<t2),1)
```

```
[[13.28064999 18.27731917 20.96292172 27.19744518 27.41835505 33.51205876
  33.52669212 39.73117793 45.53651912 46.94687981 49.57188583 53.24858342
]
 [ 6.02958905 13.29494825 13.78152867 14.66956594 16.08582557 28.21169531
  29.61045326 33.92387291 38.64599161 43.07000977 51.13429972 52.16377238
]
 [10.80950878 11.99914529 16.96539147 24.2305067  30.80900698 31.48715912
  33.1923351  34.1140176  40.46105566 45.84468745 50.69635476 51.34678531
]
 [ 4.6832156   9.00135085 10.21558525 15.19703265 21.60332112 22.83862629
  33.83260849 33.89022497 39.12641373 44.14499784 59.34598398 73.90686801
]
 [ 2.72472092  6.05462579  8.71642392 14.13212385 16.46455536 24.4654313
  25.45756973 25.68249192 27.14667669 75.603018   80.99587534 88.08290394
]]
```

Para implementar a função poissonNS_V, será nenessário usar os passos anteriores mais dois passos:

- Contar quantos eventos ocorreram entre t1 e t1. Para isso você pode usar a função np.count_nonzero passando como argumento a expressão lógica que define quais instantes de tempo estão entre t1 e t2 ((X>t1) & (X<t2)). Atenção que função np_count_nozero deve contar os eventos em cada linha da matriz X.
- Contar a quantidade de linhas onde quantidade de evntos contados entre t1 e t2 é igual a x (o argumento x da função. Uasar mais uma vez a função np.count_nonzero. Não esquecer de dividir por nSim antes de retornar a probabilidade simulada.

In [6]:
```
def poissonNS_V(x, lbda, t1, t2, nSim):
    mu = 1/lbda
    m = int(np.ceil(2*t2/mu))

    X = np.cumsum(np.random.exponential(mu, (nSim, m)), axis=1)
    events = np.count_nonzero((X>t1)&(X<t2), axis=1)
    count = np.sum(events == x)
    prob = count/nSim

    return prob
```

In [7]:
```
t1 = 15
t2 = 25
S = t2-t1
lbda = 0.2
nSim = 100000
x=3
probT = st.poisson.pmf(x, lbda*S)
inic = time.perf_counter()
probS = poissonNS_V(x, lbda, t1, t2, nSim)
fim = time.perf_counter()
print('Probabilidade simulada:  {:.4f}'.format(probS))
print('Probabilidade teórica: {:.4f}'.format(probT))
print('Tempo de simulação iterativa: {:.4f}'.format(fim-inic))
```

```
Probabilidade simulada:  0.1841
Probabilidade teórica: 0.1804
Tempo de simulação iterativa: 0.0606
```