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: $N_S(x)$ e $S_N(t)$. 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:

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

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:

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

Importação das bibliotecas necessárias

```
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.

```
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).

```
t1 = 15
t2 = 25
S = t2-t1
lbda = 0.2
nSim = 50000
x = 5
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:  0.0347
Probabilidade teórica: 0.0361
Tempo de simulação iterativa: 16.5610
```

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.

```python
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)
```

```
[[ 4.10626643  5.41560278  9.16047579 13.55184251 14.54916278
25.28805676
  33.18109277 33.95037859 35.82254719 40.74104953 40.96246473
41.38501519]
 [ 8.05977475  8.86703443 11.25503086 12.07106315 21.63442004
24.24535304
  24.49445898 32.19837483 32.20499691 34.69574521 46.37561937
55.41939191]
 [ 1.75129003  2.55073564  7.12202624 10.24517934 11.34775746
15.06563906
  15.15744756 18.92521977 19.96229743 21.48553064 37.32477154
49.58912236]
 [ 1.2652451   6.93865977 13.47861095 16.41059284 19.15014711
19.95904072
  26.58255393 29.20843426 30.35752216 36.25159363 39.53939798
42.97694103]
 [ 3.88778517 18.45925309 20.09683679 22.5652615  25.27509251
26.62242041
  42.79038801 47.05220921 47.53746633 50.10703368 54.72741443
55.3260577 ]]
```

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.

```python
def poissonNS_V(x, lbda, t1, t2, nSim):
    # Calcula mu (1/lambda)
    mu = 1 / lbda

    # Calcula m (número de colunas da matriz X)
    m = int(np.ceil(2 * t2 / mu))

    # Sorteia matriz X (acumulado de tempos exponenciais)
```

```python
    X = np.cumsum(np.random.exponential(mu, (nSim, m)), axis=1)

    # Calcula matriz eventos (quantos eventos ocorreram entre t1 e t2)
    eventos = np.count_nonzero((X > t1) & (X < t2), axis=1)

    # Calcula a probabilidade de ocorrer exatamente x eventos entre t1
e t2
    prob = np.count_nonzero(eventos == x) / nSim

    return prob

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 vetorial: {:.4f}'.format(fim-inic))
```

```
Probabilidade simulada:  0.1825
Probabilidade teórica: 0.1804
Tempo de simulação vetorial: 0.0824
```