Российский Государственный Университет нефти и газа имени И.М.Губкина

Кафедра «Прикладной математики и компьютерного моделирования»

Лабораторная работа №2:

«Моделирование и анализ процесса восстановления»

Выполнил работу:

студент гр. АМ-20-06

Коротченя И. С.

Проверил:

Кузнецов П. В.

Москва   
2023 г.

**Постановка задачи**

Требуется смоделировать K реализаций процесса восстановления длины T с заданным законом распределения F(x) интервалов между скачками и проверить выполнение:

1. ЗБЧ при t = T;
2. ЦПТ при t = T;

Ход работы

Случайный процесс — процесс (т.е. изменение во времени сосотояния системы) течение которого может быть различным в зависимости от случая. (Последовательность X={X(t), t - время})

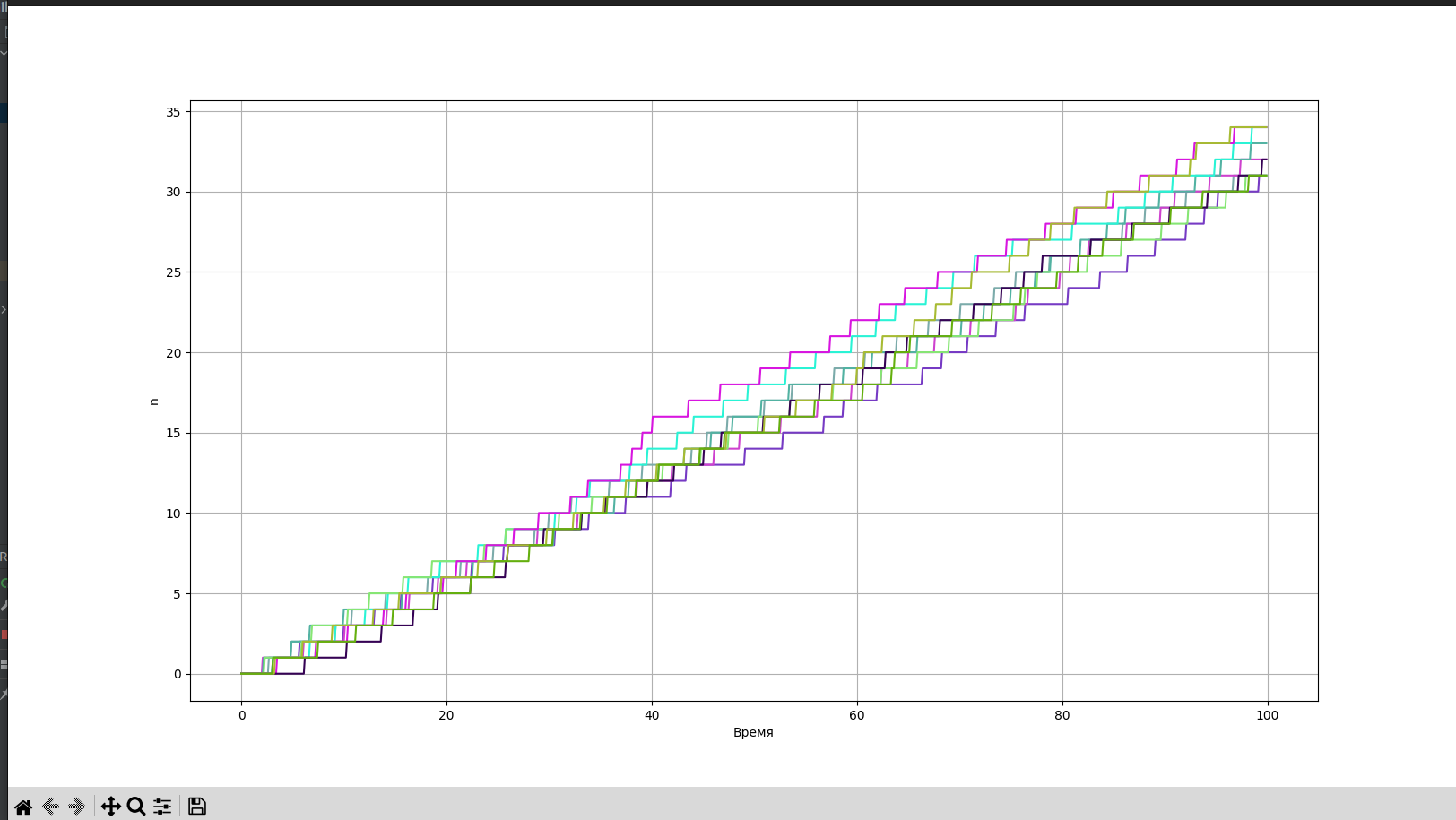
Процессом восстановления называется процесс:

Где имеют заданное нормальное распределение, t = T – ограничение по времени.

Пусть распределение случайной величины задается плотностью имеющей вид:

Тогда говорят, что имеет распределение N(3, 1)

Используя Sn, были построены k = 10 траекторий получившихся процессов восстановления.

****

Проверка ЗБЧ

Необходимо проверить следующее:

Для этого была зафиксирована последняя реализация процесса и получены и результат для каждой траектории:

Результаты:

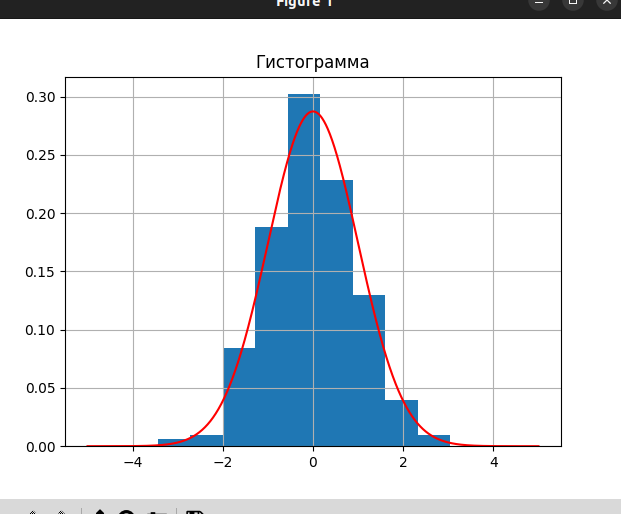
1. 3.070965495449249
2. 2.917037512063864
3. 3.18961621620471
4. 2.9190722348044633
5. 3.125563879318157
6. 3.019830248251529
7. 3.290930232427767
8. 3.0004965423528227
9. 2.9697447803092456
10. 3.0623002368409287

Как видим, значения близки к мат ожиданию, следовательно, закон выполняется.

Проверка ЦПТ

Необходимо проверить следующее:

Для проверки построим k=500 и проверим распределение по критерию хи-квадрат (Кроме того, построим гистограмму).

****

|  |  |  |  |
| --- | --- | --- | --- |
| Данные | Значение хи-квадрат | Квантиль хи-квадрат распределения | а |
| проницаемости | 22.84 | 30.14 | 0.95 |

Так как значение хи-квадрат меньше, чем квантиль, можно сделать вывод о неотвержение нулевой гипотезу о равенстве распределений

Приложение 1. Код программы

import math  
  
import numpy as np  
from scipy import stats  
import matplotlib.pyplot as plt  
  
  
def find\_sn(t, X, is\_last, return\_sn=False):  
 S\_n = 0  
 n = 0  
 while S\_n <= t:  
 S\_n += X[n]  
 n += 1  
 n -= 1  
 S\_n -= X[n]  
 if is\_last:  
 print(S\_n / n)  
 if return\_sn:  
 return S\_n, n  
 return n  
  
  
def gen\_proc(mu, std, t, ax):  
 n = []  
 X = np.random.normal(mu, std, size=100000)  
 for t\_now in t:  
 n.append(find\_sn(t\_now, X, is\_last=(t\_now == t[-1])))  
 draw(n, t, ax)  
  
  
def draw(n, t, ax):  
 col = (np.random.random(), np.random.random(), np.random.random())  
 ax.plot(t, n, c=col)  
  
  
def gen\_all\_tract(n, mu, std):  
 fig = plt.figure()  
 ax = fig.add\_subplot()  
  
 for \_ in range(n):  
 gen\_proc(mu, std, np.arange(0, 100, 0.1), ax)  
  
 ax.set\_xlabel('Время')  
 ax.set\_ylabel('n')  
 ax.grid()  
 plt.show()  
  
  
def give\_S\_n\_and\_n(count, mu, std):  
 S\_n = []  
 n = []  
 for \_ in range(count):  
 X = np.random.normal(mu, std, size=100000)  
 S\_n\_now, n\_now = find\_sn(99.9, X, is\_last=False, return\_sn=True)  
 S\_n.append(S\_n\_now)  
 n.append(n\_now)  
 return S\_n, n  
  
  
def draw\_hist(sample):  
 fig = plt.figure()  
 ax = fig.add\_subplot(111)  
  
 n\_bins = 10  
 intervals = np.linspace(min(sample), max(sample), n\_bins) # начало каждого интервала  
 frequency = np.array([len([value for value in sample if intervals[i] <= value < intervals[i + 1]]) for i in range(n\_bins-1)])  
 frequency = frequency / len(sample) # частоты  
 labels = [(intervals[i] + intervals[i + 1]) / 2 for i in range(n\_bins-1)] # середины столбцов  
 width = labels[1] - labels[0] # ширина одного столбца  
  
 ax.bar(labels, frequency, width)  
  
 x = np.linspace(-5, 5, num=1000) # создаем массив значений x  
 y = stats.norm.pdf(x) \* width # вычисляем плотность вероятности стандартного нормального распределения  
 plt.plot(x, y, c='r') # строим график  
  
 ax.set\_title('Гистограмма')  
 plt.grid()  
 plt.show()  
  
  
def calculate\_chi\_2(sample):  
 n\_bins = 20  
 n =len(sample)  
 bin\_edges = np.linspace(np.min(sample), np.max(sample), n\_bins)  
 p\_i = np.array([stats.norm.cdf(bin\_edges[i]) - stats.norm.cdf(bin\_edges[i-1]) for i in range(1, n\_bins)])  
 n\_p\_i = n\*p\_i  
  
 h\_n\_i = np.array(  
 [len([value for value in sample if bin\_edges[i- 1] <= value < bin\_edges[i]]) for i in range(1, n\_bins)])  
 h\_n\_i = h\_n\_i  
  
 chi\_2 = sum((h\_n\_i - n\_p\_i) \*\* 2 / n\_p\_i)  
  
 df = n\_bins - 1 # степени свободы  
 alpha = 0.95 # уровень значимости  
 c = stats.chi2.ppf(alpha, df)  
 print(chi\_2, c)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 mu = 3  
 std = 1  
 # gen\_all\_tract(n=10, mu=mu, std=std)  
  
 S\_n, n = give\_S\_n\_and\_n(count=500, mu=mu, std=std)  
 S\_n = [(S\_n[i] - mu \* n[i]) / (std \* math.sqrt(n[i])) for i in range(len(S\_n))]  
 draw\_hist(S\_n)  
 calculate\_chi\_2(S\_n)