МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное
учреждение высшего образования
**«Рязанский государственный радиотехнический университет
Имени В. Ф. Уткина»**

Факультет вычислительной техники
Кафедра вычислительной и прикладной математики

Отчёт по практической работе №7

по дисциплине:
“Моделирование”

по теме:

“ Моделирование случайных блужданий”


Выполнил: студент гр. 242

Фокин А.М.

Проверил: Анастасьев А. А.


Рязань 2025

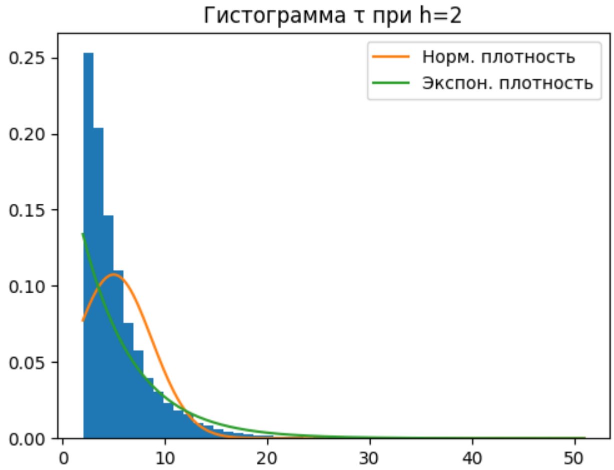
**Цель работы:**

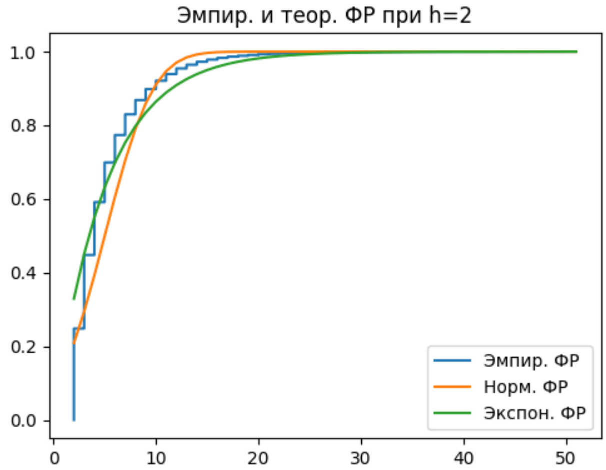
В результате проведения определенного количества экспериментов требуется построить статистическое распределение исследуемого параметра 20 (гистограмму и эмпирическую функцию распределения) и определить целесообразность аппроксимации полученного распределения одним из известных законов (нормальным, экспоненциальным, логарифмически-нормальным и др.).
6(19 вар.). Модель падения дождевой капли. При воздействии случайных порывов легкого ветра падение дождевой капли можно моделировать случайным блужданием на квадратной решетке. Движение начинается с узла, расположенного на расстоянии h над горизонтальной линией (поверхностью земли). Вероятность p↓ шага «вниз» больше вероятности p↑ шага «вверх». Вероятности скачков целесообразно выбирать равными p↓=0,5; p↑=0,1; p←=p→=0,2. Определите время τ, за которое капля достигает горизонтальной прямой, и функциональную зависимость τ от h (4..6 значений).
**Программа реализует следующие операции:**
– **Моделирование случайного блуждания дождевой капли:** `r = np.random.rand()` → выбор шага из p_down, p_up, p_left, p_right в функции one_walk(h)
– **Подсчёт времени достижения поверхности земли:**
при каждом шаге координата y уменьшается, и счётчик шагов steps увеличивается до тех пор, пока y > 0
– **Формирование статистического распределения времени падения:**
после N экспериментов собираются значения τ в массив times, вычисляются среднее и стандартное отклонение:
`mean_t = times.mean(), std_t = times.std()`
– **Построение гистограммы и наложение теоретических плотностей:**
```
plt.hist(times, bins=50, density=True)
plt.plot(xs, norm_pdf, ...), plt.plot(xs, exp_pdf, ...)
```
– **Построение эмпирической функции распределения и сравнение с теоретическими:**
`s, y = empirical_cdf(times)`
plt.step(s, y, ...) — эмпирическая ФР, далее накладываются нормальная и экспоненциальная.
– **Оценка соответствия распределений критерием Колмогорова (KS):**
```
ks_norm = ks_statistic(times, lambda x: normal_cdf(x, mean_t, std_t))
ks_exp = ks_statistic(times, lambda x: 1 - math.exp(-lam_exp *
```

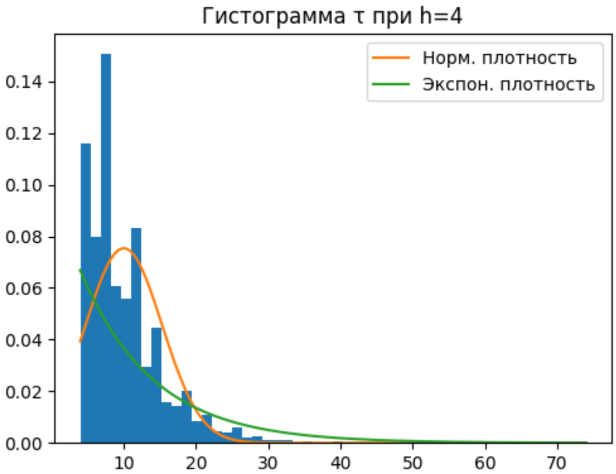
`x))`— выбирается распределение с меньшим значением статистики KS как более подходящее.

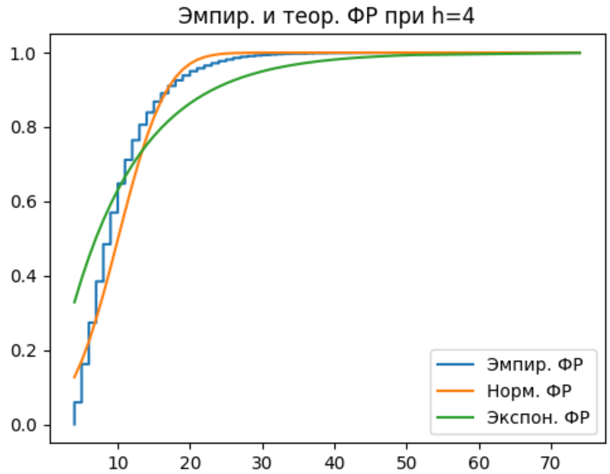
*полный код программы приведен в приложении 1*

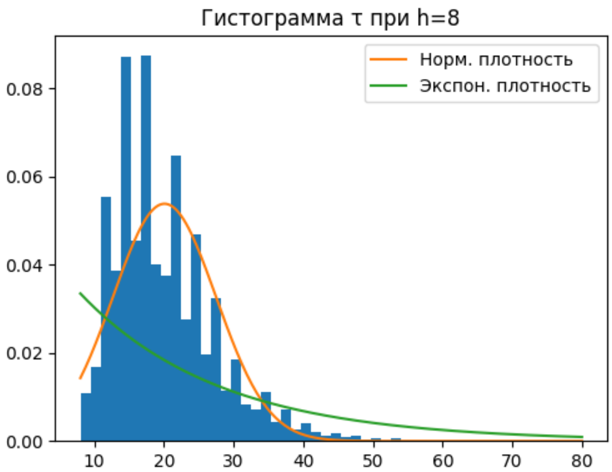
## Результат работы программы:

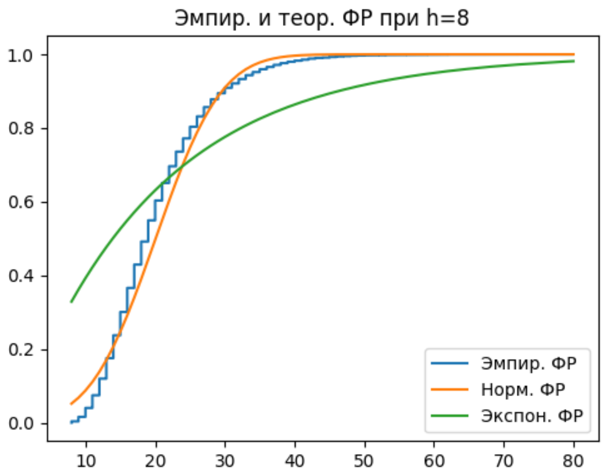


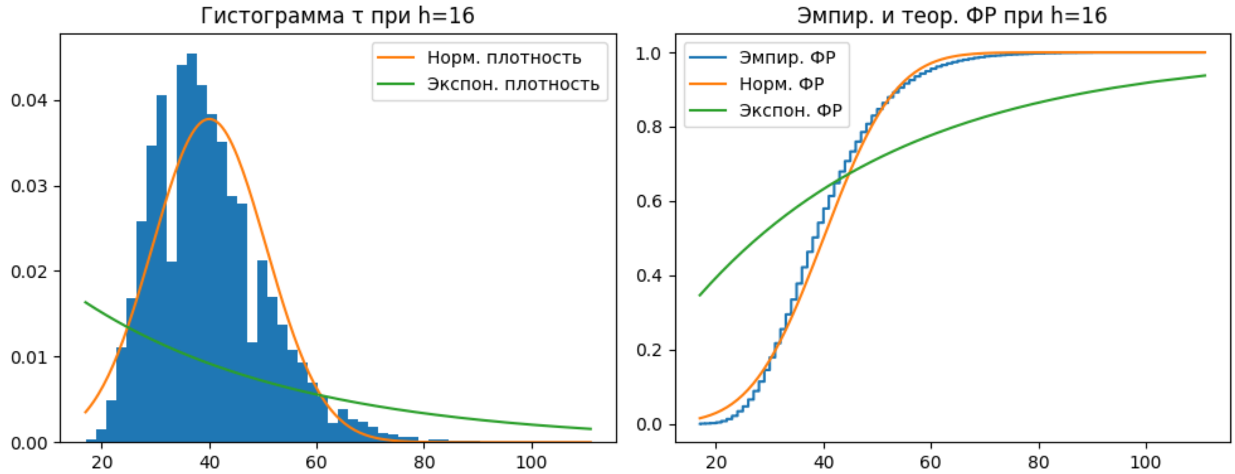



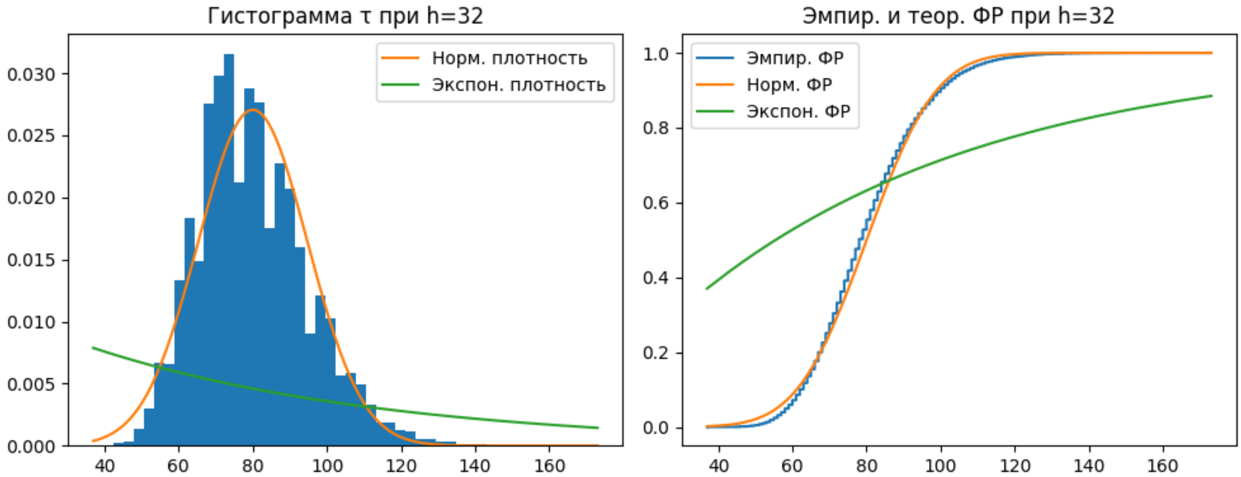





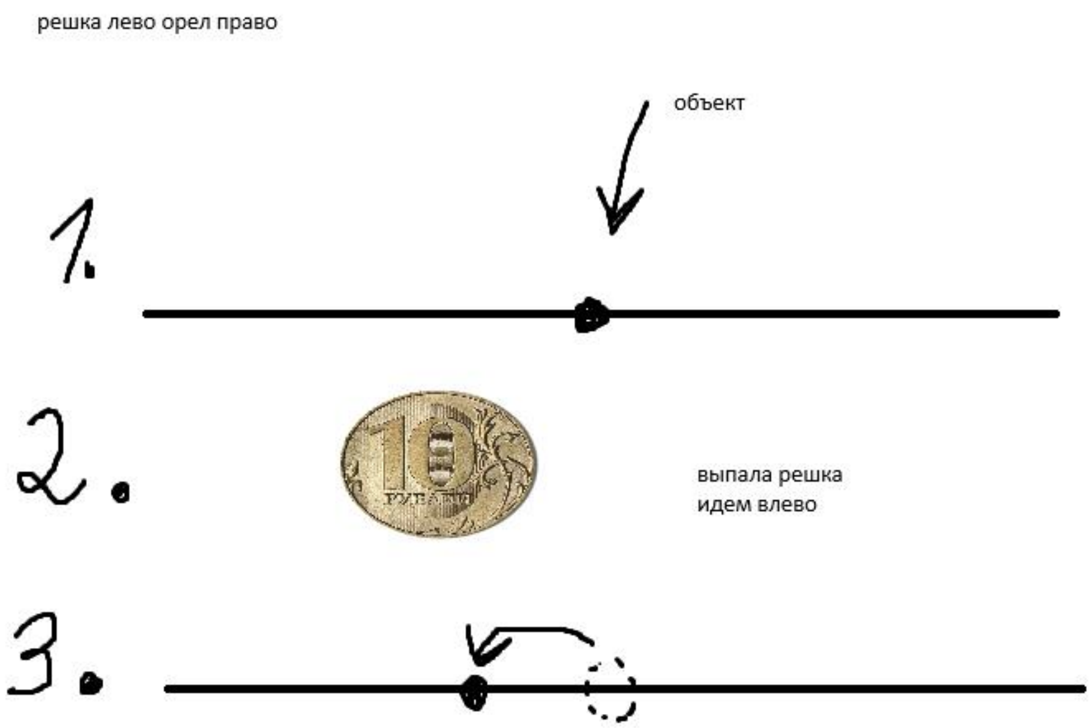




```
=========== RESTART: D:\мусорка\учеба\моделирования\lab7-Fokin-242.py ==========
h = 2
Среднее τ = 5.0073, σ = 3.7114
KS нормального = 0.2088, KS экспоненциального = 0.3292
Более целесообразная аппроксимация: нормальная
h = 4
Среднее τ = 10.0378, σ = 5.2950
KS нормального = 0.1508, KS экспоненциального = 0.3331
Более целесообразная аппроксимация: нормальная
h = 8
Среднее τ = 20.0717, σ = 7.4149
KS нормального = 0.1070, KS экспоненциального = 0.3819
Более целесообразная аппроксимация: нормальная
h = 16
Среднее τ = 40.0231, σ = 10.5679
KS нормального = 0.0803, KS экспоненциального = 0.4311
Более целесообразная аппроксимация: нормальная
h = 32
Среднее τ = 80.0368, σ = 14.7476
KS нормального = 0.0573, KS экспоненциального = 0.4787
Более целесообразная аппроксимация: нормальная
```

**Ответ на вопрос**

1. Каким образом осуществляется моделирование одномерного случайного блуждания?

В одномерном случайном блуждании объект оказывается в фиксированной точке и делает шаги либо вправо, либо влево вдоль прямой линии. Каждый шаг определяется случайным выбором, например подбрасыванием монеты, где орёл может означать шаг вправо, а решка — шаг влево. Положение объекта после n шагов равно сумме этих случайных шагов.




Источник:
https://www.geeksforgeeks.org/engineering-mathematics/probabilistic-version-of-random-walk/

**Приложение 1 - код программы**

```
import numpy as np
import math
import matplotlib.pyplot as plt

p_down = 0.5
p_up = 0.1
p_left = 0.2
p_right = 0.2
probs = [p_down, p_up, p_left, p_right]
moves = [(0, -1), (0, 1), (-1, 0), (1, 0)]
h_values = [2, 4, 8, 16, 32]
N = 20000
np.random.seed(0)

def one_walk(h):
    x = 0
    y = h
    steps = 0
    while y > 0:
        r = np.random.rand()
        if r < probs[0]:
            dx, dy = moves[0]
        elif r < probs[0] + probs[1]:
            dx, dy = moves[1]
        elif r < probs[0] + probs[1] + probs[2]:
            dx, dy = moves[2]
        else:
            dx, dy = moves[3]
        x += dx
        y += dy
        steps += 1
    return steps

def empirical_cdf(data):
    s = np.sort(data)
    y = np.arange(1, len(s)+1) / len(s)
    return s, y

def normal_cdf(x, mean, std):
```

```
    return 0.5 * (1 + math.erf((x - mean) / (std *
math.sqrt(2))))

def ks_statistic(data, cdf_func):
    s = np.sort(data)
    n = len(s)
    ecdf = np.arange(1, n+1) / n
    diffs = [abs(ecdf[i] - cdf_func(s[i])) for i in range(n)]
    return max(diffs)

for h in h_values:
    times = np.empty(N, dtype=int)
    for i in range(N):
        times[i] = one_walk(h)
    mean_t = times.mean()
    std_t = times.std(ddof=0)
    lam_exp = 1 / mean_t
    s, y = empirical_cdf(times)
    ks_norm = ks_statistic(times, lambda x: normal_cdf(x,
mean_t, std_t))
    ks_exp = ks_statistic(times, lambda x: 1 - math.exp(-lam_exp
* x))
    better = "нормальная" if ks_norm < ks_exp else
"экспоненциальная"
    print(f"h = {h}")
    print(f"Среднее τ = {mean_t:.4f}, σ = {std_t:.4f}")
    print(f"KS нормального = {ks_norm:.4f}, KS экспоненциального
= {ks_exp:.4f}")
    print(f"Более целесообразная аппроксимация: {better}")
    plt.figure(figsize=(10,4))
    plt.subplot(1,2,1)
    plt.hist(times, bins=50, density=True)
    xs = np.linspace(times.min(), times.max(), 200)
    norm_pdf = (1/(std_t*math.sqrt(2*math.pi))) * np.exp(-
0.5*((xs-mean_t)/std_t)**2)
    exp_pdf = lam_exp * np.exp(-lam_exp * xs)
    plt.plot(xs, norm_pdf, label='Норм. плотность')
    plt.plot(xs, exp_pdf, label='Экспон. плотность')
    plt.title(f"Гистограмма τ при h={h}")
    plt.legend()
    plt.subplot(1,2,2)
    plt.step(s, y, where='post', label='Эмпир. ФР')
    theor_norm_cdf = [normal_cdf(x, mean_t, std_t) for x in s]
```

```
theor_exp_cdf = [1 - math.exp(-lam_exp * x) for x in s]
plt.plot(s, theor_norm_cdf, label='Норм. ФР')
plt.plot(s, theor_exp_cdf, label='Экспон. ФР')
plt.title(f"Эмпир. и теор. ФР при h={h}")
plt.legend()
plt.tight_layout()
plt.show()
```