

**МИНОБРНАУКИ РОССИИ**

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«МИРЭА – Российский технологический университет»**

|  |
| --- |
| **РТУ МИРЭА** |
|  |
| **Институт кибербезопасности и цифровых технологий (ИКБ)** |
|  |
| КБ-2 «Информационно-аналитические системы кибербезопасности» |

**ОТЧЕТ О ВЫПОЛНЕНИИ КОНТРОЛЬНОГО ДОМАШНЕГО ЗАДАНИЯ №2**

**ПО ДИСЦИПЛИНЫ «ФОРМАЛИЗОВАННЫЕ МОДЕЛИ И МЕТОДЫ ИНФОРМАЦИОННО-АНАЛИТИЧЕСКОГО МОНИТОРИНГА»**

Выполнил:

Студент 4-ого курса

Учебной группы БИСО-02-22

Зубарев В.С.

Оглавление

[**Цели работы** 3](#_Toc210844240)

[**Решение игры** 4](#_Toc210844241)

[**Имитационная модель №1** 6](#_Toc210844242)

[**Имитационная модель №2** 7](#_Toc210844243)

[**Имитационная модель №3** 8](#_Toc210844244)

[**Имитационная модель №4** 9](#_Toc210844245)

[**Вывод** 11](#_Toc210844246)

[**Приложение 1** 12](#_Toc210844247)

# **Цели работы**

В информационной системе (ИС) можно использовать два протокола обмена информацией между сотрудниками. Известно, что злоумышленники могут использовать 5 типов DDoS-атак для перегрузки системы передачи данных и замедления ее работы. Время передачи одного сообщения в системе при использовании штатных средств защиты информации указано в таблице 1. Требуется определить оптимальную стратегию использования протоколов передачи данных, гарантирующую минимальное среднее время обработки одной операции. Представить задачу в виде матричной игры и решить ее графоаналитическим методом.

Имитационная модель с осциллограммой

Игроки выбрали оптимальные значения и среднее по двум осциллограммам совпало

Защита выбрала оптимальное значение злоумышленник меняет значения в пределах активных стратегий

Защита выбрала оптимальное значение злоумышленник меняет значения в пределах всех стратегий

Сравнить с решением по вальду для первого игрока



Таблица 1- Вариант исходных данных №7

# **Решение игры**

Для решения игры необходимо применить смешанные стратегии. Ведем обозначения и , где , где – множества смешанных стратегий игроков, выполняется . Любая матричная игра имеет цену в смешанных стратегиях .Введем и получим следующие уравнения.

Программное решение задачи представлено на рисунке 1.

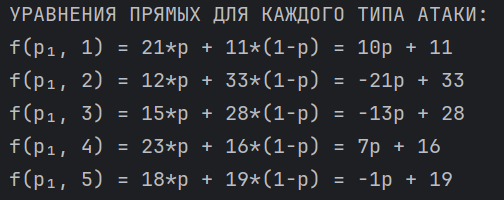


Рисунок 1 - Уравнения платежных функций

Графическое решение задачи . представлено на рисунке 2.

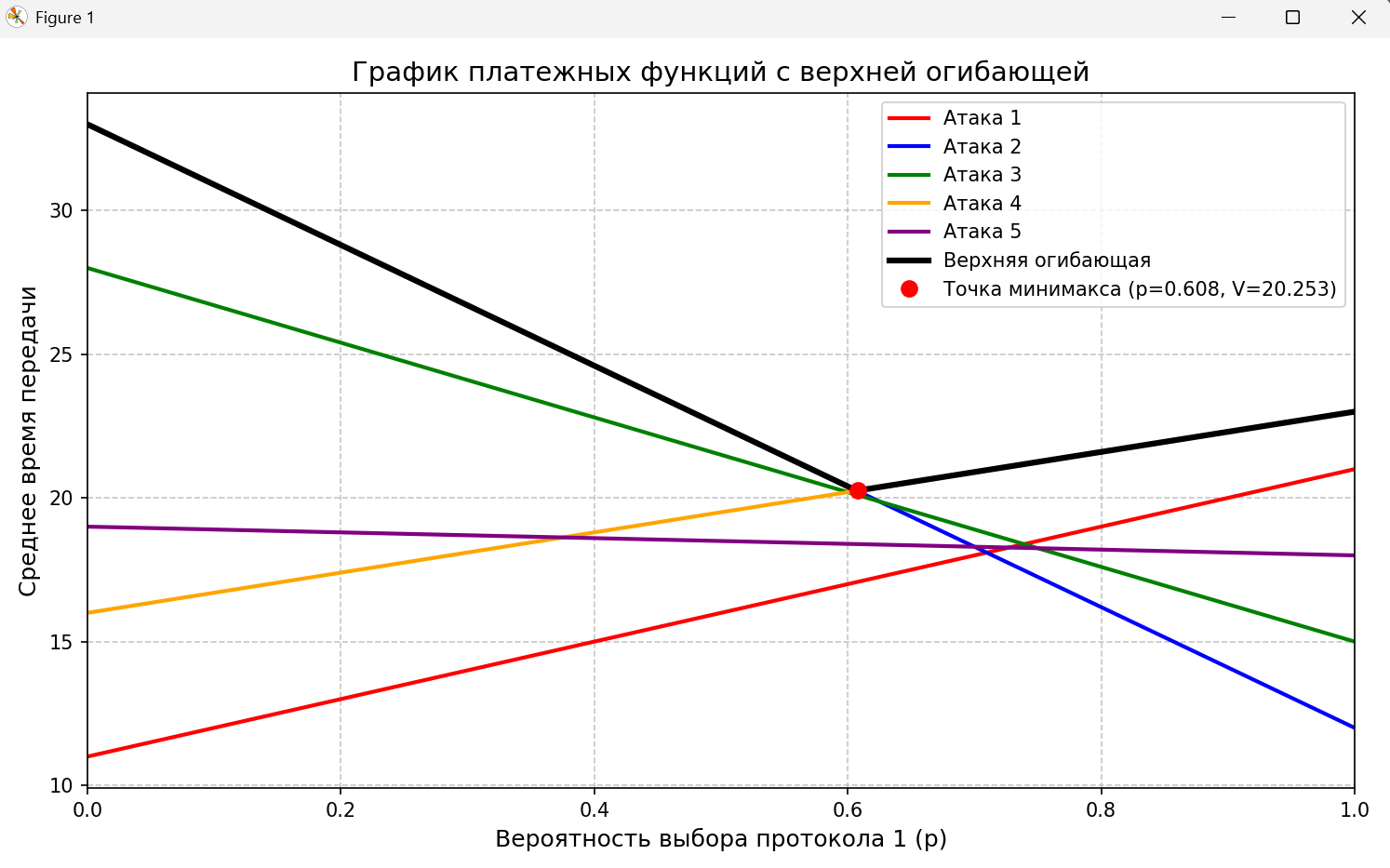


Рисунок 2 - Графическое решение

Данный рисунок иллюстрирует, что при смешанной стратеги игрока А, с распределением вероятностей выбора протоколов время передачи пакета не превысит 20.253 условных единиц независимо от действий игрока B. Так же видно, что для достижения оптимального результата кол-во активных стратегий игрока B = 2 (Атака типа 2 и 4). Найдем значения вероятностей q для игрока B решив одно из уравнений вида , где i – протокол защиты, а j и k – варианты атак такие что . Для значений задачи

# **Имитационная модель №1**

Для создания имитационной модели 1000 экспериментов, где в первом случае игрок A выбирает свою стратегию, на основании вероятности p, а затем игрок B выбреет свою стратегию из активных, независимо от игрока A на основании вероятности q. Результат проведения эксперимента показан на рисунке 3 и 4.

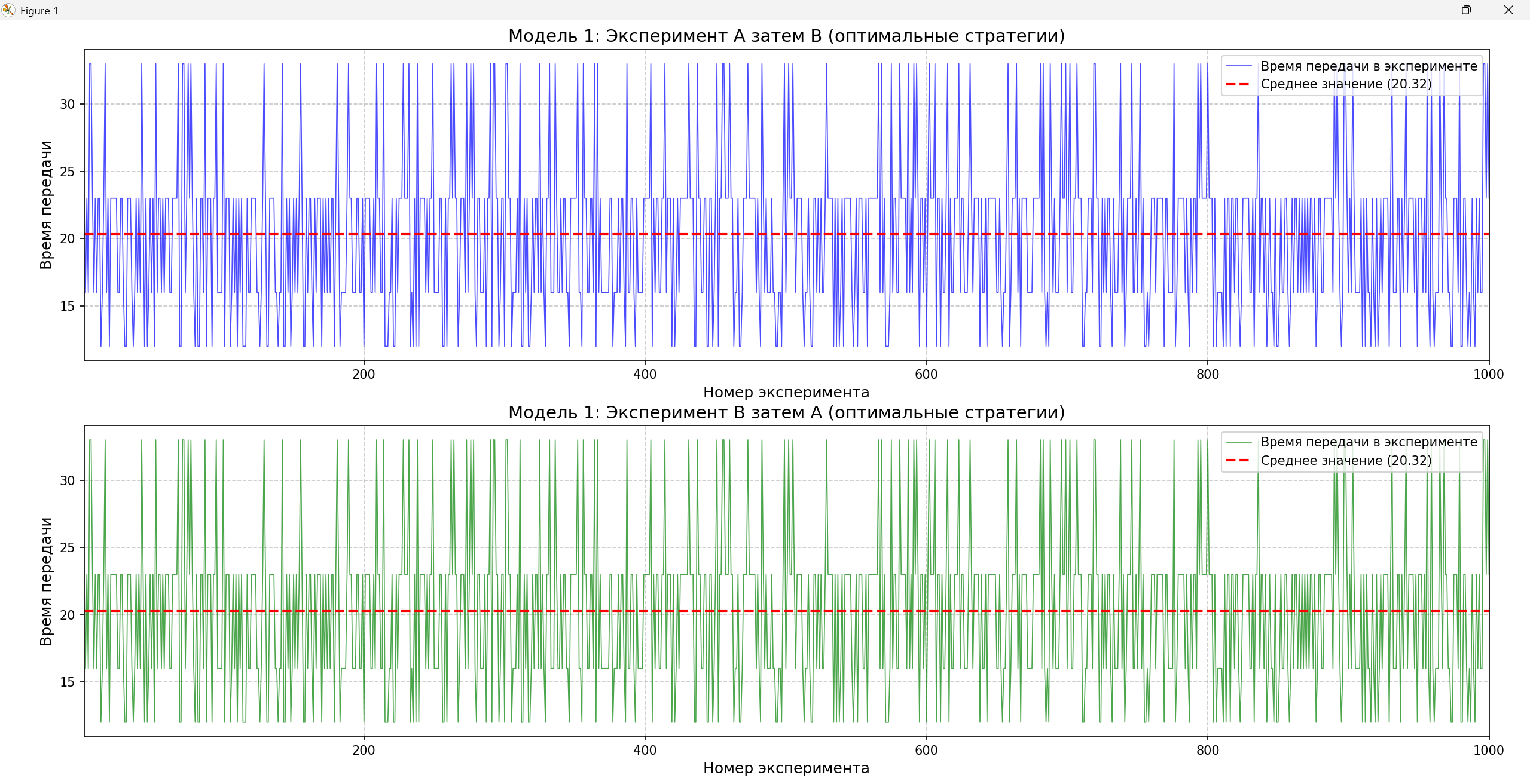


Рисунок 3 - Графики времени передачи в эксперименте

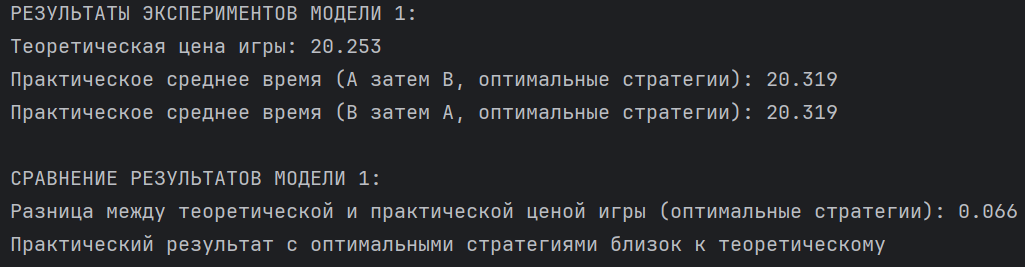


Рисунок 4 - Программная обработка результатов экспериментов

# **Имитационная модель №2**

Для создания имитационной модели 1000 экспериментов, где в первом случае игрок A выбирает свою стратегию, на основании вероятности p, а затем игрок B выбреет свою стратегию из активных, независимо от игрока A на основании вероятности q. В отличие от имитационной модели №1 злоумышленник выбирает стратегию, не учитывая вероятности попадания в седловую точку. Так как активных стратегий всего 2, то вероятность выбора любой стратегии из активных q = 0.5. Результат имитации представлен на рисунках 5 и 6.

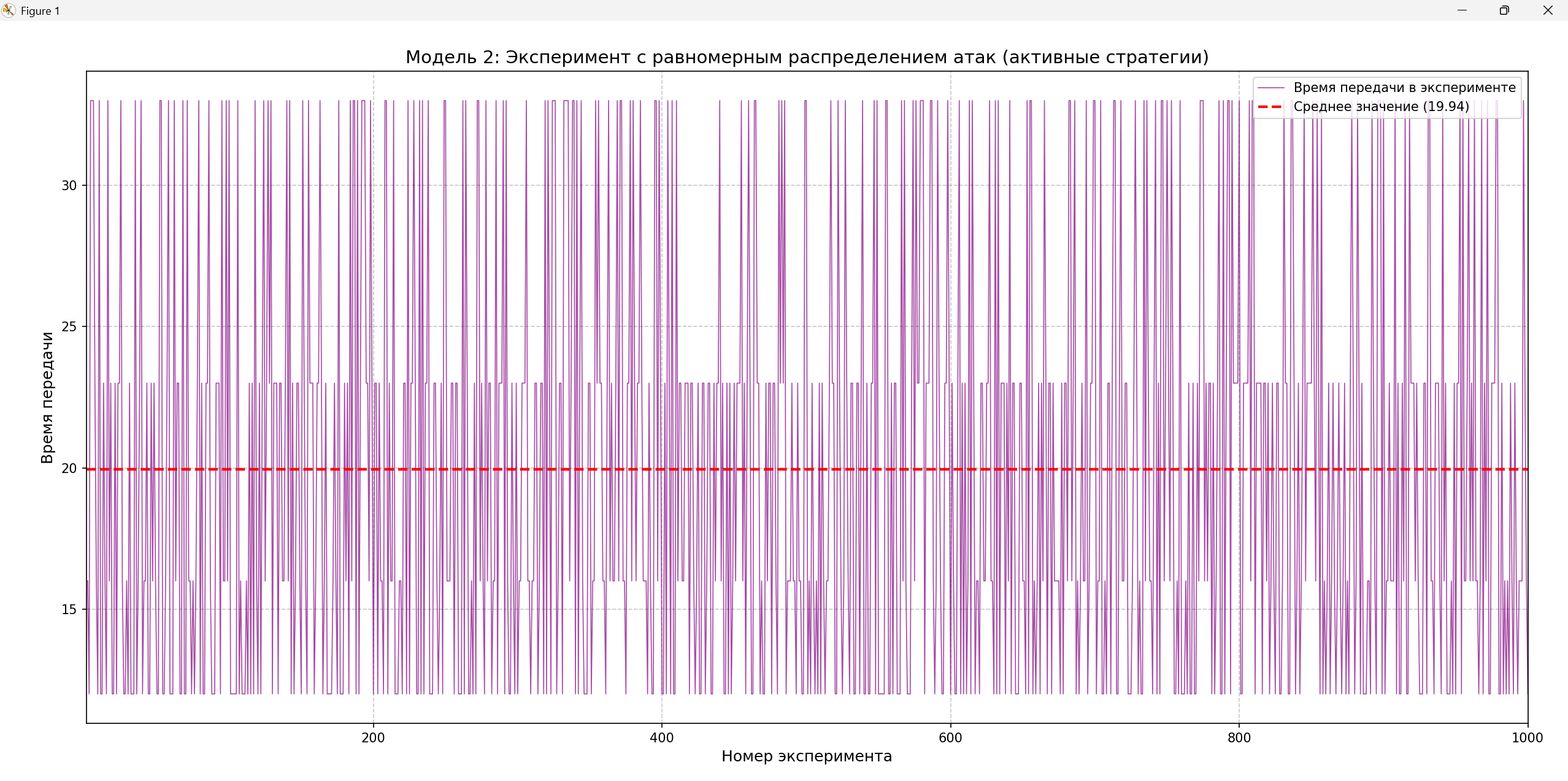


Рисунок 5 - График эксперимента

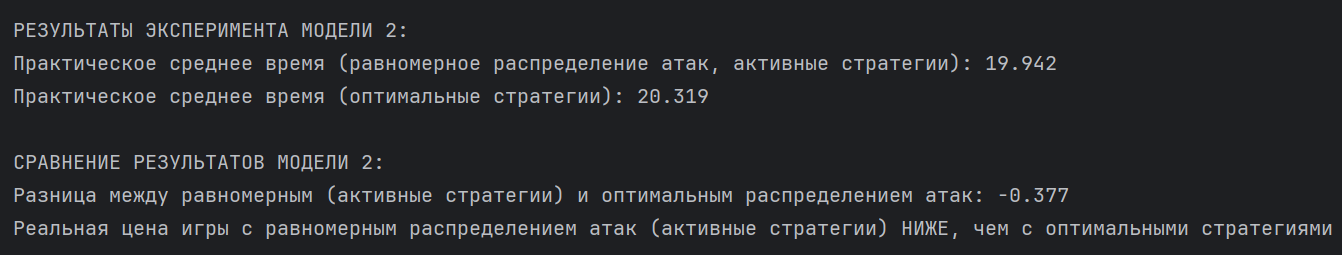


Рисунок 6 - Программная обработка результатов эксперимента

# **Имитационная модель №3**

Для создания имитационной модели 1000 экспериментов, где в первом случае игрок A выбирает свою стратегию, на основании вероятности p, а затем игрок B выбреет свою стратегию из всех возможных, независимо от игрока A на основании вероятности q. В отличие от имитационной модели №1 злоумышленник выбирает стратегию, не учитывая вероятности попадания в седловую точку. Так как всех стратегий у игрока B - 5, то вероятность выбора любой стратегии из активных q = 0.2. Результат имитации представлен на рисунках 7 и 8.

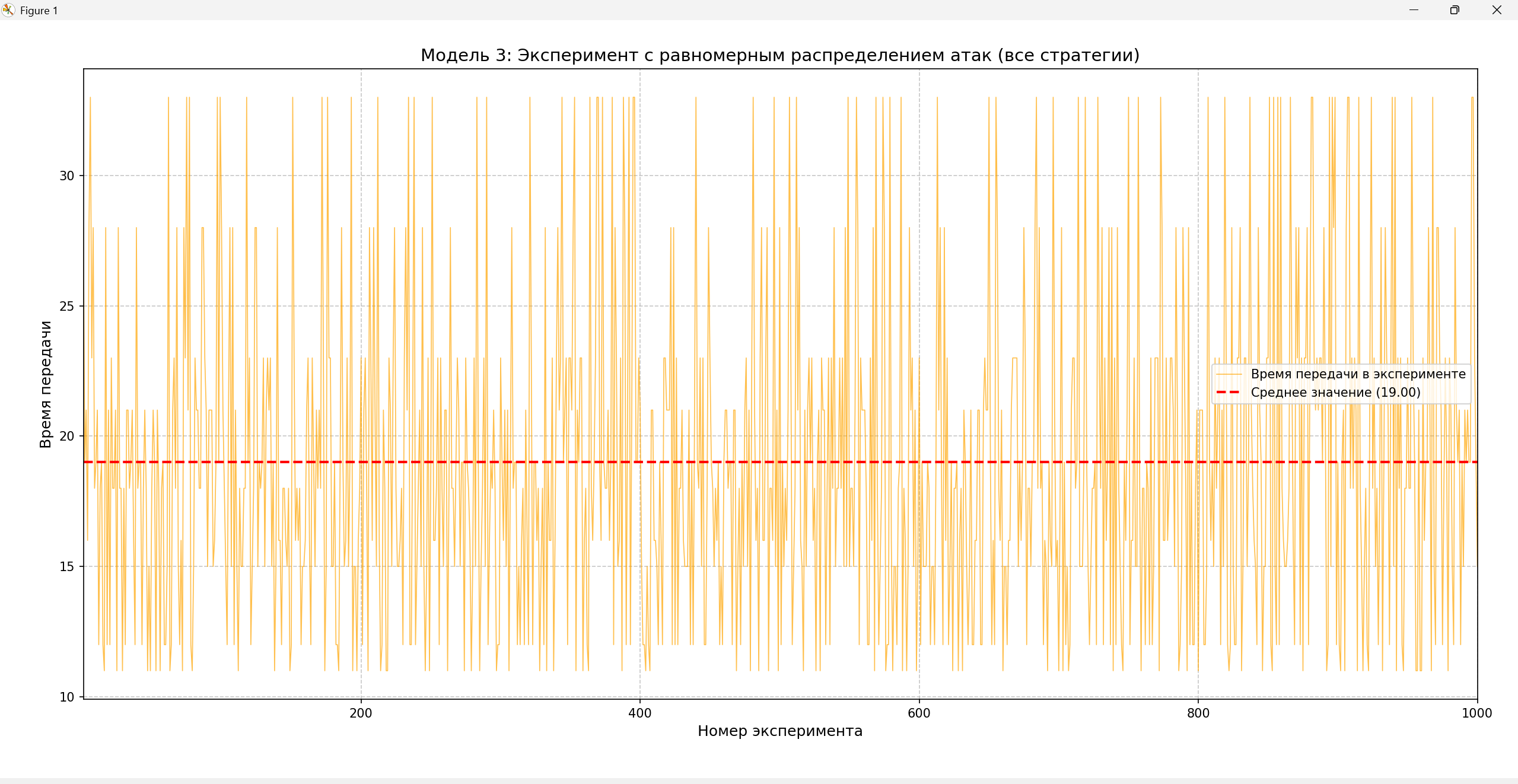


Рисунок 7 - График эксперимента

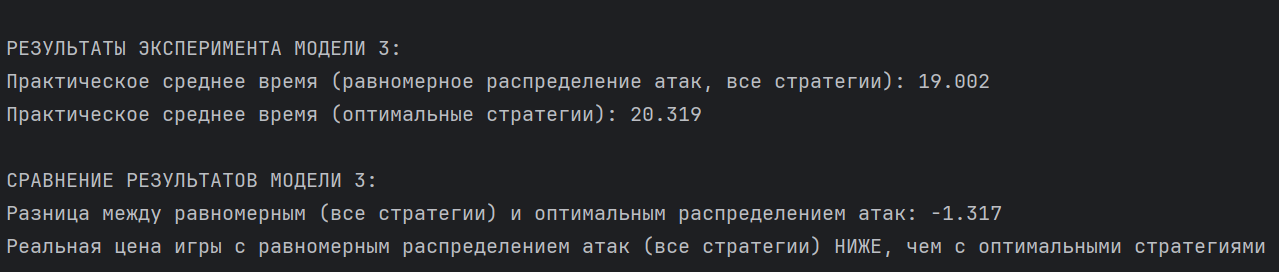


Рисунок 8 - Программная обработка результатов эксперимента

# **Имитационная модель №4**

Для создания имитационной модели 1000 экспериментов, где в первом случае игрок A выбирает свою стратегию, на основании критерия Вальда, а затем игрок B выбреет свою стратегию из активных, независимо от игрока A на основании вероятности q. Значения матрицы, для выявления оптимального по Вальду протокола показаны в таблице 1.



Из таблицы видно, что оптимальным по критерию Вальда будет протокол 1. Проведем эксперимент с учетом того, что игрок A всегда выбирает протокол 1.

Результат проведения эксперимента показан на рисунке 9 и 10.

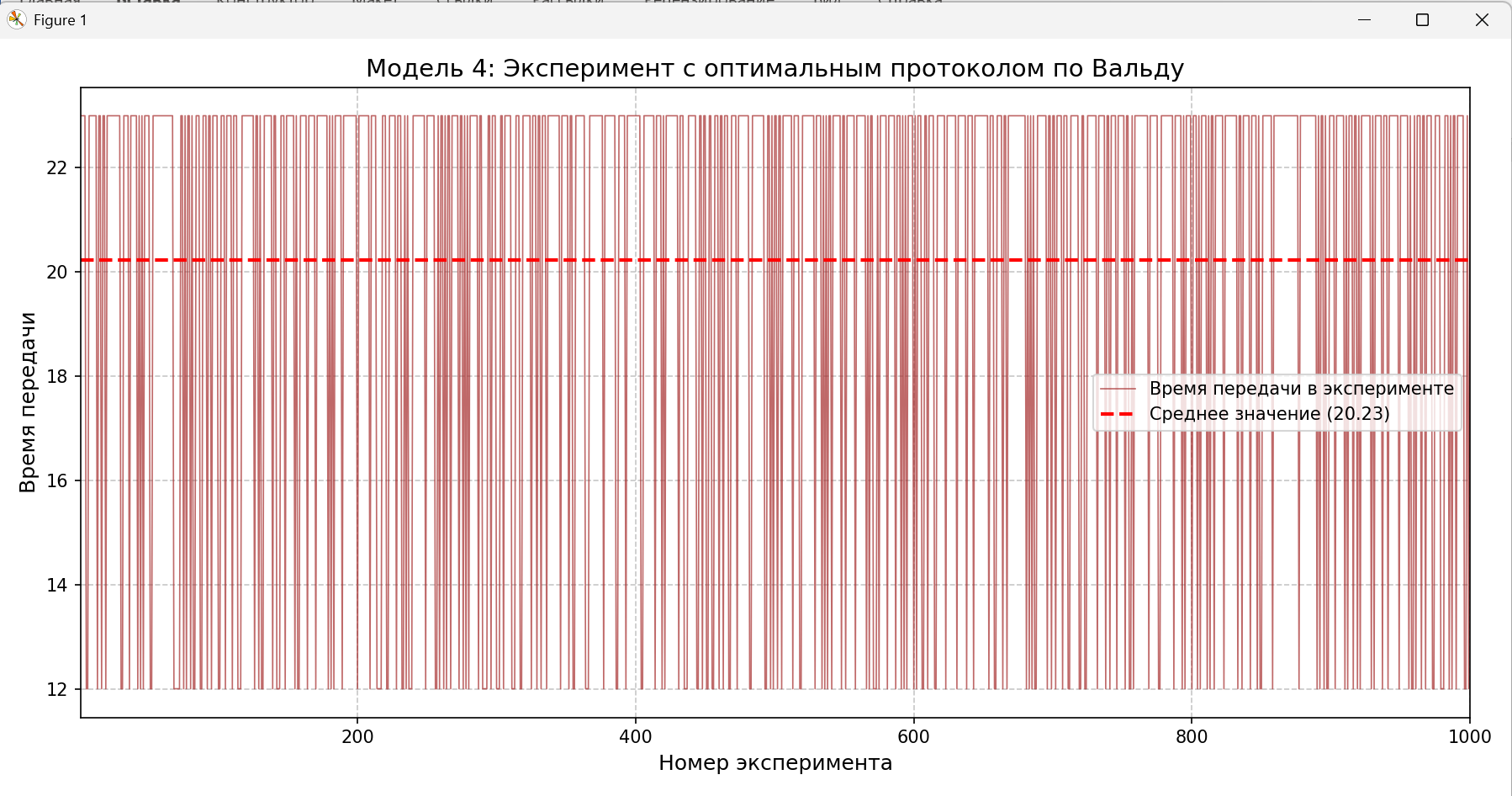


Рисунок 9 - График эксперимента

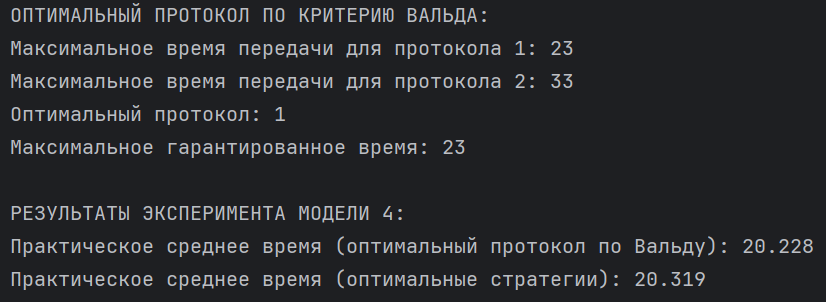


Рисунок 10 - Программная обработка результатов эксперимента

# **Вывод**

Из проведенных экспериментов видно, что любое отклонение игроков A и B от стратегий связанных с седловой точкой ухудшает положение того игрока, который отклонился от данной стратегии, что подтверждает выполнение неравенств . Где ,для текущих условий задачи. Программный анализ результатов эксперимента показан на рисунке 11.

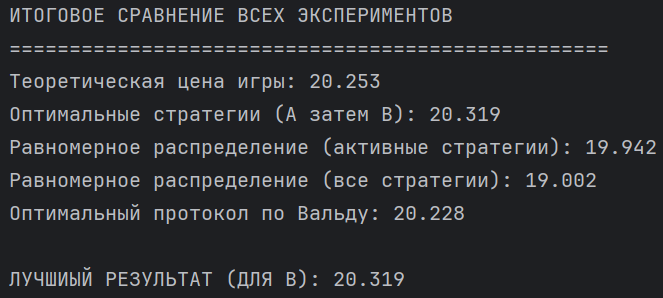


Рисунок 11 - Программный анализ результатов

# **Приложение 1**

В приложении 1 представлен листинг кода программы

from tabulate import tabulate  
import matplotlib.pyplot as plt  
import numpy as np  
import random  
  
  
# =============================================================================  
# ЗАДАНИЕ 1: Анализ матричной игры  
# =============================================================================  
  
def check\_saddle\_point(matrix):  
 *"""  
 Проверяет наличие седловой точки в матричной игре  
 """* row\_minima = [min(row) for row in matrix]  
 alpha = max(row\_minima)  
  
 cols = len(matrix[0])  
 col\_maxima = [max(matrix[i][j] for i in range(len(matrix))) for j in range(cols)]  
 beta = min(col\_maxima)  
  
 has\_saddle = alpha == beta  
  
 return has\_saddle, alpha, beta  
  
  
def calculate\_payment\_functions(payment\_matrix):  
 *"""  
 Вычисляет коэффициенты платежных функций  
 """* num\_attacks = len(payment\_matrix[0])  
 functions = []  
  
 for j in range(num\_attacks):  
 a1j = payment\_matrix[0][j]  
 a2j = payment\_matrix[1][j]  
  
 k = a1j - a2j  
 b = a2j  
  
 function\_info = {  
 'attack\_num': j + 1,  
 'a1j': a1j,  
 'a2j': a2j,  
 'k': k,  
 'b': b,  
 'equation': f"f(p₁, {j + 1}) = {a1j}\*p + {a2j}\*(1-p) = {k}p + {b}"  
 }  
 functions.append(function\_info)  
  
 return functions  
  
  
def plot\_payment\_functions(functions):  
 *"""  
 Строит график платежных функций с верхней огибающей и точкой минимакса  
 """* p = np.linspace(0, 1, 1000)  
  
 plt.figure(figsize=(10, 6))  
  
 colors = ['red', 'blue', 'green', 'orange', 'purple']  
  
 all\_y = []  
  
 for i, func in enumerate(functions):  
 k = func['k']  
 b = func['b']  
 y = k \* p + b  
 all\_y.append(y)  
 plt.plot(p, y, color=colors[i % len(colors)], linewidth=2, label=f"Атака {func['attack\_num']}")  
  
 all\_y\_array = np.array(all\_y)  
 # Ищем верхнюю огибающую (максимум для каждого p)  
 upper\_envelope = np.max(all\_y\_array, axis=0)  
  
 # Строим верхнюю огибающую  
 plt.plot(p, upper\_envelope, 'k-', linewidth=3, label='Верхняя огибающая')  
  
 # Находим точку минимакса (минимум верхней огибающей)  
 min\_index = np.argmin(upper\_envelope)  
 p\_optimal = p[min\_index]  
 value\_optimal = upper\_envelope[min\_index]  
  
 # Отмечаем точку минимакса на графике  
 plt.plot(p\_optimal, value\_optimal, 'ro', markersize=8,  
 label=f'Точка минимакса (p={p\_optimal:.3f}, V={value\_optimal:.3f})')  
  
 plt.title("График платежных функций с верхней огибающей", fontsize=14)  
 plt.xlabel("Вероятность выбора протокола 1 (p)", fontsize=12)  
 plt.ylabel("Среднее время передачи", fontsize=12)  
 plt.grid(True, linestyle='--', alpha=0.7)  
 plt.legend()  
 plt.xlim(0, 1)  
  
 plt.tight\_layout()  
 plt.show()  
  
 return p\_optimal, value\_optimal  
  
  
def find\_optimal\_strategies(matrix, p\_optimal, value\_optimal):  
 *"""  
 Находит оптимальные вероятности для игрока B по указанному методу  
 """* active\_strategies = []  
 tolerance = 0.01  
  
 for j in range(len(matrix[0])):  
 a1j = matrix[0][j]  
 a2j = matrix[1][j]  
 value\_at\_p = a1j \* p\_optimal + a2j \* (1 - p\_optimal)  
  
 if abs(value\_at\_p - value\_optimal) < tolerance:  
 active\_strategies.append(j + 1)  
  
 if len(active\_strategies) < 2:  
 differences = []  
 for j in range(len(matrix[0])):  
 a1j = matrix[0][j]  
 a2j = matrix[1][j]  
 value\_at\_p = a1j \* p\_optimal + a2j \* (1 - p\_optimal)  
 differences.append((j + 1, abs(value\_at\_p - value\_optimal)))  
  
 differences.sort(key=lambda x: x[1])  
 active\_strategies = [diff[0] for diff in differences[:2]]  
  
 q\_probabilities = [0] \* len(matrix[0])  
  
 if len(active\_strategies) == 2:  
 i, j = active\_strategies[0] - 1, active\_strategies[1] - 1  
  
 a1i, a1j = matrix[0][i], matrix[0][j]  
 a2i, a2j = matrix[1][i], matrix[1][j]  
  
 # Решаем уравнения для каждого протокола и находим q1  
 # Для протокола 1: q1 \* a1i + (1-q1) \* a1j = V  
 if (a1i - a1j) != 0:  
 q1\_from\_protocol1 = (value\_optimal - a1j) / (a1i - a1j)  
 else:  
 q1\_from\_protocol1 = 0.5  
  
 # Для протокола 2: q1 \* a2i + (1-q1) \* a2j = V  
 if (a2i - a2j) != 0:  
 q1\_from\_protocol2 = (value\_optimal - a2j) / (a2i - a2j)  
 else:  
 q1\_from\_protocol2 = 0.5  
  
 # Находим среднее значение q1  
 q1\_avg = (q1\_from\_protocol1 + q1\_from\_protocol2) / 2  
  
 # Округляем до 0.001  
 q1 = round(q1\_avg, 3)  
 q2 = round(1 - q1, 3)  
  
 # Корректируем сумму до 1.000  
 if q1 + q2 != 1.0:  
 q1 = 1.0 - q2  
  
 q\_probabilities[i] = q1  
 q\_probabilities[j] = q2  
  
 return q\_probabilities, active\_strategies  
  
  
# =============================================================================  
# ЗАДАНИЕ 2: Моделирование экспериментов  
# =============================================================================  
  
def generate\_player\_A\_choices(p\_optimal, num\_experiments=1000):  
 *"""  
 Генерирует массив выборов игрока A на основе вероятности p\_optimal  
 """* return [1 if random.random() < p\_optimal else 2 for \_ in range(num\_experiments)]  
  
  
def generate\_player\_B\_choices(q\_probabilities, num\_experiments=1000):  
 *"""  
 Генерирует массив выборов игрока B на основе вероятностей q\_probabilities  
 """* choices = []  
 for \_ in range(num\_experiments):  
 rand\_val = random.random()  
 cumulative\_prob = 0  
 attack\_type = 1  
  
 for j, prob in enumerate(q\_probabilities):  
 cumulative\_prob += prob  
 if rand\_val <= cumulative\_prob:  
 attack\_type = j + 1  
 break  
  
 choices.append(attack\_type)  
  
 return choices  
  
  
def generate\_player\_B\_uniform\_choices(active\_strategies, num\_experiments=1000):  
 *"""  
 Генерирует массив выборов игрока B с равномерным распределением между активными стратегиями  
 """* return [random.choice(active\_strategies) for \_ in range(num\_experiments)]  
  
  
def generate\_player\_B\_all\_uniform\_choices(num\_attacks, num\_experiments=1000):  
 *"""  
 Генерирует массив выборов игрока B с равномерным распределением между всеми стратегиями  
 """* return [random.randint(1, num\_attacks) for \_ in range(num\_experiments)]  
  
  
def generate\_player\_A\_wald\_choices(optimal\_protocol, num\_experiments=1000):  
 *"""  
 Генерирует массив выборов игрока A, всегда выбирающего оптимальный протокол по Вальду  
 """* return [optimal\_protocol for \_ in range(num\_experiments)]  
  
  
def calculate\_experiment\_results(matrix, player\_A\_choices, player\_B\_choices):  
 *"""  
 Вычисляет результаты экспериментов на основе массивов выборов игроков  
 """* results = []  
 cumulative\_sum = 0  
 cumulative\_averages = []  
  
 for i in range(len(player\_A\_choices)):  
 protocol = player\_A\_choices[i]  
 attack\_type = player\_B\_choices[i]  
  
 time = matrix[protocol - 1][attack\_type - 1]  
  
 results.append(time)  
 cumulative\_sum += time  
 cumulative\_averages.append(cumulative\_sum / (i + 1))  
  
 return results, cumulative\_averages  
  
  
def plot\_experiment\_results\_task2(results\_A\_then\_B, cumulative\_averages\_A\_then\_B,  
 results\_B\_then\_A, cumulative\_averages\_B\_then\_A):  
 *"""  
 Строит графики результатов экспериментов задания 2 в одном окне  
 """* fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(12, 10))  
  
 # График для эксперимента A затем B  
 ax1.plot(range(1, len(results\_A\_then\_B) + 1), results\_A\_then\_B, 'b-', alpha=0.7, linewidth=0.8,  
 label='Время передачи в эксперименте')  
 final\_average\_A\_then\_B = cumulative\_averages\_A\_then\_B[-1]  
 ax1.axhline(y=final\_average\_A\_then\_B, color='r', linestyle='--', linewidth=2,  
 label=f'Среднее значение ({final\_average\_A\_then\_B:.2f})')  
 ax1.set\_title('Модель 1: Эксперимент A затем B (оптимальные стратегии)', fontsize=14)  
 ax1.set\_xlabel('Номер эксперимента', fontsize=12)  
 ax1.set\_ylabel('Время передачи', fontsize=12)  
 ax1.grid(True, linestyle='--', alpha=0.7)  
 ax1.legend()  
 ax1.set\_xlim(1, len(results\_A\_then\_B))  
  
 # График для эксперимента B затем A  
 ax2.plot(range(1, len(results\_B\_then\_A) + 1), results\_B\_then\_A, 'g-', alpha=0.7, linewidth=0.8,  
 label='Время передачи в эксперименте')  
 final\_average\_B\_then\_A = cumulative\_averages\_B\_then\_A[-1]  
 ax2.axhline(y=final\_average\_B\_then\_A, color='r', linestyle='--', linewidth=2,  
 label=f'Среднее значение ({final\_average\_B\_then\_A:.2f})')  
 ax2.set\_title('Модель 1: Эксперимент B затем A (оптимальные стратегии)', fontsize=14)  
 ax2.set\_xlabel('Номер эксперимента', fontsize=12)  
 ax2.set\_ylabel('Время передачи', fontsize=12)  
 ax2.grid(True, linestyle='--', alpha=0.7)  
 ax2.legend()  
 ax2.set\_xlim(1, len(results\_B\_then\_A))  
  
 plt.tight\_layout()  
 plt.show()  
  
 return final\_average\_A\_then\_B, final\_average\_B\_then\_A  
  
  
def plot\_experiment\_results\_task3(results\_uniform, cumulative\_averages\_uniform):  
 *"""  
 Строит график результатов эксперимента задания 3 в отдельном окне  
 """* plt.figure(figsize=(12, 6))  
  
 plt.plot(range(1, len(results\_uniform) + 1), results\_uniform, 'purple', alpha=0.7, linewidth=0.8,  
 label='Время передачи в эксперименте')  
  
 final\_average\_uniform = cumulative\_averages\_uniform[-1]  
 plt.axhline(y=final\_average\_uniform, color='r', linestyle='--', linewidth=2,  
 label=f'Среднее значение ({final\_average\_uniform:.2f})')  
  
 plt.title('Модель 2: Эксперимент с равномерным распределением атак (активные стратегии)', fontsize=14)  
 plt.xlabel('Номер эксперимента', fontsize=12)  
 plt.ylabel('Время передачи', fontsize=12)  
 plt.grid(True, linestyle='--', alpha=0.7)  
 plt.legend()  
 plt.xlim(1, len(results\_uniform))  
  
 plt.tight\_layout()  
 plt.show()  
  
 return final\_average\_uniform  
  
  
def plot\_experiment\_results\_task4(results\_all\_uniform, cumulative\_averages\_all\_uniform):  
 *"""  
 Строит график результатов эксперимента задания 4 в отдельном окне  
 """* plt.figure(figsize=(12, 6))  
  
 plt.plot(range(1, len(results\_all\_uniform) + 1), results\_all\_uniform, 'orange', alpha=0.7, linewidth=0.8,  
 label='Время передачи в эксперименте')  
  
 final\_average\_all\_uniform = cumulative\_averages\_all\_uniform[-1]  
 plt.axhline(y=final\_average\_all\_uniform, color='r', linestyle='--', linewidth=2,  
 label=f'Среднее значение ({final\_average\_all\_uniform:.2f})')  
  
 plt.title('Модель 3: Эксперимент с равномерным распределением атак (все стратегии)', fontsize=14)  
 plt.xlabel('Номер эксперимента', fontsize=12)  
 plt.ylabel('Время передачи', fontsize=12)  
 plt.grid(True, linestyle='--', alpha=0.7)  
 plt.legend()  
 plt.xlim(1, len(results\_all\_uniform))  
  
 plt.tight\_layout()  
 plt.show()  
  
 return final\_average\_all\_uniform  
  
  
def plot\_experiment\_results\_task5(results\_wald, cumulative\_averages\_wald):  
 *"""  
 Строит график результатов эксперимента задания 5 в отдельном окне  
 """* plt.figure(figsize=(12, 6))  
  
 plt.plot(range(1, len(results\_wald) + 1), results\_wald, 'brown', alpha=0.7, linewidth=0.8,  
 label='Время передачи в эксперименте')  
  
 final\_average\_wald = cumulative\_averages\_wald[-1]  
 plt.axhline(y=final\_average\_wald, color='r', linestyle='--', linewidth=2,  
 label=f'Среднее значение ({final\_average\_wald:.2f})')  
  
 plt.title('Модель 4: Эксперимент с оптимальным протоколом по Вальду', fontsize=14)  
 plt.xlabel('Номер эксперимента', fontsize=12)  
 plt.ylabel('Время передачи', fontsize=12)  
 plt.grid(True, linestyle='--', alpha=0.7)  
 plt.legend()  
 plt.xlim(1, len(results\_wald))  
  
 plt.tight\_layout()  
 plt.show()  
  
 return final\_average\_wald  
  
  
def find\_optimal\_protocol\_wald(matrix):  
 *"""  
 Находит оптимальный протокол по критерию Вальда  
 """* # Находим максимальное время передачи для каждого протокола  
 max\_time\_protocol1 = max(matrix[0])  
 max\_time\_protocol2 = max(matrix[1])  
  
 # Выбираем протокол с минимальным максимальным временем  
 if max\_time\_protocol1 <= max\_time\_protocol2:  
 optimal\_protocol = 1  
 max\_time = max\_time\_protocol1  
 else:  
 optimal\_protocol = 2  
 max\_time = max\_time\_protocol2  
  
 return optimal\_protocol, max\_time  
  
  
# =============================================================================  
# ОСНОВНАЯ ПРОГРАММА  
# =============================================================================  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 M = [  
 [21, 12, 15, 23, 18],  
 [11, 33, 28, 16, 19]  
 ]  
  
 # Вывод исходной матрицы  
 headers = ["Протокол/Атака", "Тип 1", "Тип 2", "Тип 3", "Тип 4", "Тип 5"]  
 table\_data = [  
 ["Протокол 1"] + M[0],  
 ["Протокол 2"] + M[1]  
 ]  
  
 print("ИСХОДНАЯ МАТРИЦА ИГРЫ:")  
 print(tabulate(table\_data, headers=headers, tablefmt="grid", stralign="center"))  
  
 # Задание 1: Проверка седловой точки  
 print("\n" + "=" \* 50)  
 print("Анализ матричной игры")  
 print("=" \* 50)  
  
 has\_saddle, alpha, beta = check\_saddle\_point(M)  
  
 print("\nПРОВЕРКА СЕДЛОВОЙ ТОЧКИ:")  
 print(f"Нижняя цена игры (α): {alpha}")  
 print(f"Верхняя цена игры (β): {beta}")  
  
 if has\_saddle:  
 print("Седловая точка НАЙДЕНА - игра имеет решение в чистых стратегиях")  
 else:  
 print("Седловая точка НЕ НАЙДЕНА - необходимо решение в смешанных стратегиях")  
  
 # Уравнения платежных функций  
 functions = calculate\_payment\_functions(M)  
  
 print("\nУРАВНЕНИЯ ПЛАТЕЖНЫХ ФУНКЦИЙ:")  
 for func in functions:  
 print(func['equation'])  
  
 # Построение графика и нахождение оптимальных вероятностей для A  
 p\_optimal, value\_optimal = plot\_payment\_functions(functions)  
  
 print(f"\nОПТИМАЛЬНЫЕ ВЕРОЯТНОСТИ ДЛЯ ИГРОКА A:")  
 print(f"Вероятность выбора протокола 1 (p): {p\_optimal:.3f}")  
 print(f"Вероятность выбора протокола 2 (1-p): {1 - p\_optimal:.3f}")  
 print(f"Цена игры (V): {value\_optimal:.3f}")  
  
 # Задание 2: Нахождение оптимальных стратегий для B  
 print("\n" + "=" \* 50)  
 print("Определение оптимальных стратегий для игрока B")  
 print("=" \* 50)  
  
 q\_probabilities, active\_strategies = find\_optimal\_strategies(M, p\_optimal, value\_optimal)  
  
 print(f"\nАКТИВНЫЕ СТРАТЕГИИ ИГРОКА B: {active\_strategies}")  
  
 print(f"\nОПТИМАЛЬНЫЕ ВЕРОЯТНОСТИ ДЛЯ ИГРОКА B:")  
 for i, prob in enumerate(q\_probabilities):  
 if prob > 0:  
 print(f"Вероятность выбора атаки {i + 1} (q\_{i + 1}): {prob:.3f}")  
  
 # Проверка суммы вероятностей  
 total\_q = sum(q\_probabilities)  
 print(f"Сумма вероятностей: {total\_q:.3f}")  
  
 # Вывод уравнений для игрока B  
 print(f"\nУРАВНЕНИЯ ДЛЯ ИГРОКА B:")  
 if len(active\_strategies) == 2:  
 i, j = active\_strategies[0] - 1, active\_strategies[1] - 1  
 q\_i = q\_probabilities[i]  
 q\_j = q\_probabilities[j]  
  
 print(f"Для протокола 1: q\_{i + 1} × {M[0][i]} + q\_{j + 1} × {M[0][j]} = {value\_optimal:.3f}")  
 print(f"Для протокола 2: q\_{i + 1} × {M[1][i]} + q\_{j + 1} × {M[1][j]} = {value\_optimal:.3f}")  
 print(f"q\_{i + 1} + q\_{j + 1} = 1.000")  
  
 # Генерация массивов выборов игроков  
 # Генерируем общий массив выборов игрока A  
 player\_A\_choices = generate\_player\_A\_choices(p\_optimal, 1000)  
  
 # Генерируем массивы выборов игрока B для разных экспериментов  
 player\_B\_choices\_optimal = generate\_player\_B\_choices(q\_probabilities, 1000)  
 player\_B\_choices\_uniform = generate\_player\_B\_uniform\_choices(active\_strategies, 1000)  
 player\_B\_choices\_all\_uniform = generate\_player\_B\_all\_uniform\_choices(len(M[0]), 1000)  
 # Проведение экспериментов задания 2  
 print("\n" + "=" \* 50)  
 print("ПРОВЕДЕНИЕ ЭКСПЕРИМЕНТОВ")  
 print("=" \* 50)  
  
 # Эксперимент 1: A затем B (оптимальные стратегии)  
 results\_A\_then\_B, cumulative\_averages\_A\_then\_B = calculate\_experiment\_results(  
 M, player\_A\_choices, player\_B\_choices\_optimal  
 )  
  
 # Эксперимент 2: B затем A (оптимальные стратегии) - используем те же массивы, но в другом порядке  
 # Для этого создаем копии массивов с измененным порядком  
 player\_B\_choices\_optimal\_reordered = player\_B\_choices\_optimal.copy()  
 player\_A\_choices\_reordered = player\_A\_choices.copy()  
  
 results\_B\_then\_A, cumulative\_averages\_B\_then\_A = calculate\_experiment\_results(  
 M, player\_A\_choices\_reordered, player\_B\_choices\_optimal\_reordered  
 )  
 final\_average\_A\_then\_B, final\_average\_B\_then\_A = plot\_experiment\_results\_task2(  
 results\_A\_then\_B, cumulative\_averages\_A\_then\_B,  
 results\_B\_then\_A, cumulative\_averages\_B\_then\_A  
 )  
  
 print(f"\nРЕЗУЛЬТАТЫ ЭКСПЕРИМЕНТОВ МОДЕЛИ 1:")  
 print(f"Теоретическая цена игры: {value\_optimal:.3f}")  
 print(f"Практическое среднее время (A затем B, оптимальные стратегии): {final\_average\_A\_then\_B:.3f}")  
 print(f"Практическое среднее время (B затем A, оптимальные стратегии): {final\_average\_B\_then\_A:.3f}")  
  
 # Сравнение результатов задания 2  
 difference\_optimal = abs(final\_average\_A\_then\_B - value\_optimal)  
  
 print(f"\nСРАВНЕНИЕ РЕЗУЛЬТАТОВ МОДЕЛИ 1:")  
 print(f"Разница между теоретической и практической ценой игры (оптимальные стратегии): {difference\_optimal:.3f}")  
  
 if difference\_optimal < 0.5:  
 print("Практический результат с оптимальными стратегиями близок к теоретическому")  
 else:  
 print("Заметное расхождение между теоретическим и практическим результатами с оптимальными стратегиями")  
  
 # =============================================================================  
 # ЗАДАНИЕ 3: Эксперимент с равномерным распределением атак (активные стратегии)  
 # =============================================================================  
  
 print("\n" + "=" \* 50)  
 print("ЗАДАНИЕ 2: Эксперимент с равномерным распределением атак (активные стратегии)")  
 print("=" \* 50)  
  
 # Эксперимент 3: равномерное распределение атак (активные стратегии)  
 results\_uniform, cumulative\_averages\_uniform = calculate\_experiment\_results(  
 M, player\_A\_choices, player\_B\_choices\_uniform  
 )  
  
 final\_average\_uniform = plot\_experiment\_results\_task3(results\_uniform, cumulative\_averages\_uniform)  
  
 print(f"\nРЕЗУЛЬТАТЫ ЭКСПЕРИМЕНТА МОДЕЛИ 2:")  
 print(  
 f"Практическое среднее время (равномерное распределение атак, активные стратегии): {final\_average\_uniform:.3f}")  
 print(f"Практическое среднее время (оптимальные стратегии): {final\_average\_A\_then\_B:.3f}")  
  
 # Сравнение результатов задания 3  
 difference\_uniform = final\_average\_uniform - final\_average\_A\_then\_B  
  
 print(f"\nСРАВНЕНИЕ РЕЗУЛЬТАТОВ МОДЕЛИ 2:")  
 print(f"Разница между равномерным (активные стратегии) и оптимальным распределением атак: {difference\_uniform:.3f}")  
  
 if difference\_uniform < 0:  
 print(  
 "Реальная цена игры с равномерным распределением атак (активные стратегии) НИЖЕ, чем с оптимальными стратегиями")  
 else:  
 print(  
 "Реальная цена игры с равномерным распределением атак (активные стратегии) ВЫШЕ, чем с оптимальными стратегиями")  
  
 # =============================================================================  
 # ЗАДАНИЕ 4: Эксперимент с равномерным распределением атак (все стратегии)  
 # =============================================================================  
  
 print("\n" + "=" \* 50)  
 print("ЗАДАНИЕ 3: Эксперимент с равномерным распределением атак (все стратегии)")  
 print("=" \* 50)  
  
 # Эксперимент 4: равномерное распределение атак (все стратегии)  
 results\_all\_uniform, cumulative\_averages\_all\_uniform = calculate\_experiment\_results(  
 M, player\_A\_choices, player\_B\_choices\_all\_uniform  
 )  
  
 final\_average\_all\_uniform = plot\_experiment\_results\_task4(results\_all\_uniform, cumulative\_averages\_all\_uniform)  
  
 print(f"\nРЕЗУЛЬТАТЫ ЭКСПЕРИМЕНТА МОДЕЛИ 3:")  
 print(  
 f"Практическое среднее время (равномерное распределение атак, все стратегии): {final\_average\_all\_uniform:.3f}")  
 print(f"Практическое среднее время (оптимальные стратегии): {final\_average\_A\_then\_B:.3f}")  
  
 # Сравнение результатов задания 4  
 difference\_all\_uniform = final\_average\_all\_uniform - final\_average\_A\_then\_B  
  
 print(f"\nСРАВНЕНИЕ РЕЗУЛЬТАТОВ МОДЕЛИ 3:")  
 print(f"Разница между равномерным (все стратегии) и оптимальным распределением атак: {difference\_all\_uniform:.3f}")  
  
 if difference\_all\_uniform < 0:  
 print(  
 "Реальная цена игры с равномерным распределением атак (все стратегии) НИЖЕ, чем с оптимальными стратегиями")  
 else:  
 print(  
 "Реальная цена игры с равномерным распределением атак (все стратегии) ВЫШЕ, чем с оптимальными стратегиями")  
  
 # =============================================================================  
 # ЗАДАНИЕ 5: Эксперимент с оптимальным протоколом по критерию Вальда  
 # =============================================================================  
  
 print("\n" + "=" \* 50)  
 print("ЗАДАНИЕ 4: Эксперимент с оптимальным протоколом по критерию Вальда")  
 print("=" \* 50)  
  
 # Находим оптимальный протокол по критерию Вальда  
 optimal\_protocol\_wald, max\_time\_wald = find\_optimal\_protocol\_wald(M)  
  
 print(f"\nОПТИМАЛЬНЫЙ ПРОТОКОЛ ПО КРИТЕРИЮ ВАЛЬДА:")  
 print(f"Максимальное время передачи для протокола 1: {max(M[0])}")  
 print(f"Максимальное время передачи для протокола 2: {max(M[1])}")  
 print(f"Оптимальный протокол: {optimal\_protocol\_wald}")  
 print(f"Максимальное гарантированное время: {max\_time\_wald}")  
  
 # Генерируем массив выборов игрока A для критерия Вальда  
 player\_A\_choices\_wald = generate\_player\_A\_wald\_choices(optimal\_protocol\_wald, 1000)  
  
 # Эксперимент 5: оптимальный протокол по Вальду  
 results\_wald, cumulative\_averages\_wald = calculate\_experiment\_results(  
 M, player\_A\_choices\_wald, player\_B\_choices\_optimal  
 )  
  
 final\_average\_wald = plot\_experiment\_results\_task5(results\_wald, cumulative\_averages\_wald)  
  
 print(f"\nРЕЗУЛЬТАТЫ ЭКСПЕРИМЕНТА МОДЕЛИ 4:")  
 print(f"Практическое среднее время (оптимальный протокол по Вальду): {final\_average\_wald:.3f}")  
 print(f"Практическое среднее время (оптимальные стратегии): {final\_average\_A\_then\_B:.3f}")  
  
 # Сравнение результатов задания 5  
 difference\_wald = final\_average\_wald - final\_average\_A\_then\_B  
  
 print(f"\nСРАВНЕНИЕ РЕЗУЛЬТАТОВ МОДЕЛИ 4:")  
 print(f"Разница между оптимальным протоколом по Вальду и оптимальными стратегиями: {difference\_wald:.3f}")  
  
 if difference\_wald < 0:  
 print("Реальная цена игры с оптимальным протоколом по Вальду НИЖЕ, чем с оптимальными стратегиями")  
 else:  
 print("Реальная цена игры с оптимальным протоколом по Вальду ВЫШЕ, чем с оптимальными стратегиями")  
  
 # =============================================================================  
 # ИТОГОВОЕ СРАВНЕНИЕ ВСЕХ ЭКСПЕРИМЕНТОВ  
 # =============================================================================  
  
 print("\n" + "=" \* 50)  
 print("ИТОГОВОЕ СРАВНЕНИЕ ВСЕХ ЭКСПЕРИМЕНТОВ")  
 print("=" \* 50)  
  
 print(f"Теоретическая цена игры: {value\_optimal:.3f}")  
 print(f"Оптимальные стратегии (A затем B): {final\_average\_A\_then\_B:.3f}")  
 print(f"Равномерное распределение (активные стратегии): {final\_average\_uniform:.3f}")  
 print(f"Равномерное распределение (все стратегии): {final\_average\_all\_uniform:.3f}")  
 print(f"Оптимальный протокол по Вальду: {final\_average\_wald:.3f}")  
  
 print(  
 f"\nЛУЧШИЙ РЕЗУЛЬТАТ (ДЛЯ B): {max(final\_average\_A\_then\_B, final\_average\_uniform, final\_average\_all\_uniform, final\_average\_wald):.3f}")