Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Кафедра АСОІУ
Спеціальність 126 «Інформаційні системи та технології»

КУРСОВА РОБОТА

з дисципліни

“Моделювання систем”

Задача №11

Виконав:
студент групи ІС-72
Кривохижа Р.А.
№ зал. кн. ІС-7217

Прийняли:
Проф.
СТЕЦЕНКО І.В.

Ст. вик.
НОВІКОВА П.А.

_____ _____ _____

(підпис) (оцінка) (підпис)

(підпис)

Київ - 2020

Національний технічний університет України «КПІ ім. І.Сікорського»

Кафедра автоматизованих систем обробки інформації та управління

Дисципліна «Моделювання систем»

Спеціальність Інформаційні управляючи системи та технології

Курс 4 Група ІС - 72 Семестр 7

ЗАВДАННЯ №11

на курсову роботу студента

Кривохижи Романа Андрійовича

1.Тема роботи створення імітаційної моделі, її програмної реалізації проведення експериментів для задачі №11

2. Термін здачі студентом закінченої роботи "05" грудня 2020 р.

3. Вихідні дані до проекту лістинг програми, рисунки формату png

4. Зміст розрахунково-пояснювальної записки (перелік питань, що розробляються)

1. Аналіз існуючих методів вирішення завдання 2. Розробка концептуальної моделі 3. Вибір засобів моделювання 4. Розробка структурної схеми імітаційної моделі та опис її функціонування 4.1 Опис імітаційної моделі 4.2 Опис програмної реалізації імітаційної моделі 4.3 Оцінка адекватності моделі 5. Результати експериментів на моделі 5.1. План експериментів 5.2. Аналіз і оцінка результатів. Висновки.

5. Перелік графічного матеріалу (з точним позначенням обов'язкових креслень)

Графічного матеріалу не має.

6. Дата видачі завдання "05" жовтня 2020 р.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів курсового проекту (роботи)	Термін виконання етапів роботи	Примітка
1	Отримання завдання Формулювання теми курсової роботи	05.10.2020р	
2	Аналіз можливих методів вирішення поставленого завдання	15.10.2020р	
3	Розробка концептуальної моделі	26.10.2020р	
4	Перший контроль за процесом виконання курсового проекту (роботи), консультація у викладача	04.11.2020р	
5	Опис імітаційної моделі	10.11.2020р	
6	Опис програмної реалізації імітаційної моделі	20.11.2020р	
7	Другий контроль за процесом виконання курсового проекту (роботи), консультація у викладача	05.11.2020р	
8	Аналіз та оцінка результатів	10.11.2020р	
9	Оформлення пояснювальної записки	15.12.2020р	
10	Здача пояснювальної записки	24.12.2020р	
11	Захист курсового проекту (роботи)	24.12.2020р	

Студент_____

(підпис)

Керівник _____ _____

(підпис) (прізвище, ім'я, по батькові)

«05» жовтня 2020 р.

РЕФЕРАТ

Курсова робота: 44- с., 22- рис., 1- табл., 1- додатки, 4- джерел літератури.

Об'єкт дослідження – магістральний канал передачі даних.

Мета роботи – вибір і обґрунтування найкращого режиму роботи системи з точки зору максимізації прибутку.

Метод дослідження – імітаційне моделювання роботи магістрального каналу передачі даних.

Проведено дослідження різних режимів роботи магістрального каналу і розроблена програмна реалізація імітаційної моделі системи. Виконана верифікація, проведений дисперсійний аналіз по заданим параметрам. Результати використані для прийняття рішення про режим роботи системи.

ІМІТАЦІЙНА МОДЕЛЬ, МОВА МОДЕЛЮВАННЯ, ВЕРИФІКАЦІЯ ПЛАНУВАННЯ ЕКСПЕРИМЕНТІВ, МАГІСТРАЛЬНИЙ КАНАЛ ПЕРЕДАЧІ ДАНИЗ , МЕРЕЖА ПЕТРІ, ДИСПЕРСІЙНИЙ АНАЛІЗ

ЗМІСТ

ВСТУП 6
1 ПОСТАНОВКА ЗАДАЧІ 7
2 АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ ВИРІШЕННЯ ЗАДАЧІ 8
3 РОЗРОБКА КОНЦЕПТУАЛЬНОЇ МОДЕЛІ 9
3.1 Структурна схема моделі 10
3.2 Параметри, вхідні ти вихідні змінні 10
3.3 Цільова функція (критерій якості) системи 11
4 РОЗРОБКА СТРУКТУРНОЇ СХЕМИ ІМІТАЦІЙНОЇ МОДЕЛІ ТА ОПИС ЇЇ ФУНКЦІОНУВАННЯ 12
4.1 Вибір засобу реалізації 12
4.2 Опис програмної реалізації імітаційної моделі 13
5 ОЦІНКА АДЕКВАТНОСТІ МОДЕЛІ 15
5.1 Тестовий прогін 15
5.2 Аналітична оцінка 16
5.3 Дослідження параметру k 17
5.4 Перевірка результатів моделювання на стаціонарність 20
5.4.1 Перевірка на відтворюваність результату (за контрольним експериментом) 20
5.4.2 Перевірка на відтворюваність результату (перевірка рівності середніх значень) 21
6 РЕЗУЛЬТАТИ ЕКСПЕРИМЕНТІВ НА МОДЕЛІ 22
6.1 План експериментів 22
6.2 Аналіз і оцінка результатів 23
6.2.1 Дисперсійний аналіз 23
6.2.2 Закон розподілу часу перебування повідомлення в системі 24
6.2.3 Визначення завантаження резервного каналу та частоти переривань 27
7 ВИСНОВКИ 30
8 ПЕРЕЛІК ПОСИЛАНЬ 31
Додаток. Лістинг програми 32
Додаток 1. 32

ВСТУП

У даній роботі досліджується система магістрального каналу передачі повідомлень. Метою нашого дослідження є вибір найкращого режиму роботи даної системи.

Для вирішення нашої задачі скористаємось таким методом як імітаційне моделювання. На відміну від інших видів моделювання імітаційне моделювання враховує зміну властивостей об’єктів у часі. Імітаційне моделювання може використовуватись як універсальний підхід для прийняття рішень в умовах невизначеності для врахування в моделях факторів, що досить складно формалізуються.

1 ПОСТАНОВКА ЗАДАЧІ

Магістральний канал передачі даних складається із загального накопичувача та двох каналів - основного і резервного. Повідомлення поступають в систему через **18±10** секунд і чекають в накопичувачі початку передачі. У нормальному режимі роботи повідомлення передаються по основному каналу за **20±8** секунд. В основному каналі через інтервали часу **400±50** секунд трапляються збої. Якщо збій трапляється під час передачі повідомлення, то відбувається його переривання. При цьому за час **5** секунд запускається резервний канал, який передає перерване повідомлення з самого початку. Відновлення основного каналу займає **100±25** секунд. До відновлення основного каналу повідомлення передаються по резервному каналу. Після відновлення резервний канал відключається і основний канал продовжує роботу із чергового повідомлення.

Прибуток від передачі одного повідомлення через основний канал складає **60** одиниць вартості, а при передачі через резервний канал – **30** одиниць вартості. Є можливість підвищити надійність роботи основного каналу. При збільшенні середнього часу напрацювання на відмову на **k** секунд прибуток з кожного повідомлення зменшується на **k × 0,04** одиниць вартості.

Визначить найкращий режим роботи системи, відповідні завантаження резервного каналу, частоту переривання повідомлень і функцію розподілу часу передачі повідомлень по магістралі

2 АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ ВИРІШЕННЯ ЗАДАЧІ

Існує безліч методів моделювання, виділимо такі методи: аналітичне моделювання, математичне моделювання та імітаційне моделювання.

Аналітичне моделювання полягає у тому, що вхідні та вихідні дані перебувають у певній залежності один від одного. Дана залежність представлена у вигляді відомих аналітичних функцій. Перевагою даного методу є можливість застосування методів класичного математичного аналізу до залежності даних. Тому якщо є можливість використати аналітичну модель системи, то завжди віддають перевагу цьому методу моделювання.

Деякі системи можуть бути настільки складні, що знаходження залежності даних у явному вигляді є неможливим. У таких випадках застосовують математичне моделювання. Метод полягає у відшуканні розв’язку задачі за допомогою чисельних методів або спеціального програмного забезпечення.

Також існують такі системи, опис який не піддається опису аналітичним функціях. У такому разі процес функціонування системи може бути описаний алгоритмом імітації. Імітацію зазвичай виконують за допомогою комп’ютерної програми, яка відтворює процес функціонування складної системи у часі. Після багатократних прогонів імітаційної моделі можна робити певні висновки стосовно адекватності моделі та її властивостей. Такий метод моделювання називають імітаційним моделюванням.

Оскільки система магістрального каналу передачі повідомлень є доволі складною та не описується аналітичними функціями, то для вирішення поставленої задачі використаємо метод імітаційного моделювання.

3 РОЗРОБКА КОНЦЕПТУАЛЬНОЇ МОДЕЛІ

За допомогою концептуальної моделі ми ідентифікуємо причинно-наслідкові зв’язки, які властиві системі магістрального каналу передачі повідомлень для визначення найкращого режиму роботи даної системи.

Концептуальна схема моделі, згідно з постановкою задачі, має вигляд, зображений на рисунку 3.1.

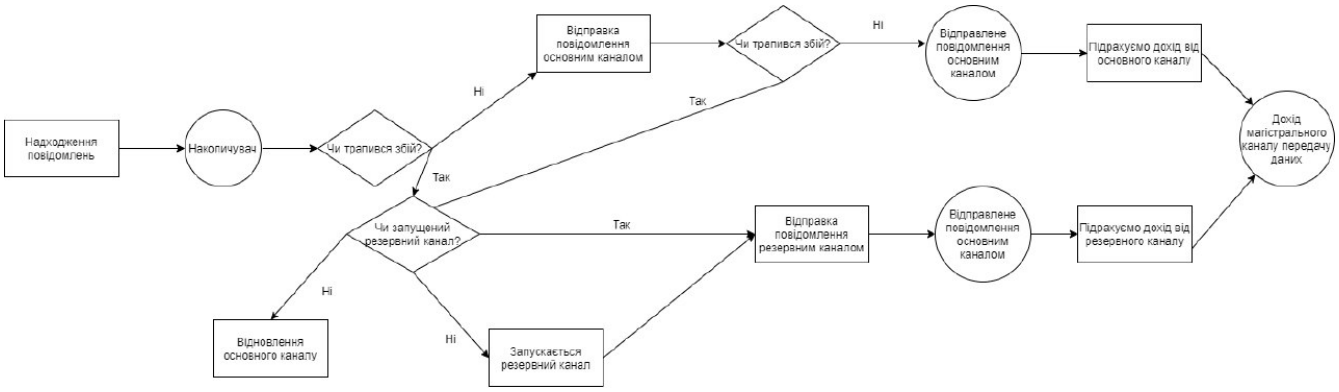

Рисунок 3.1 – Концептуальна схема моделі передачі повідомлень

3.1 Структурна схема моделі

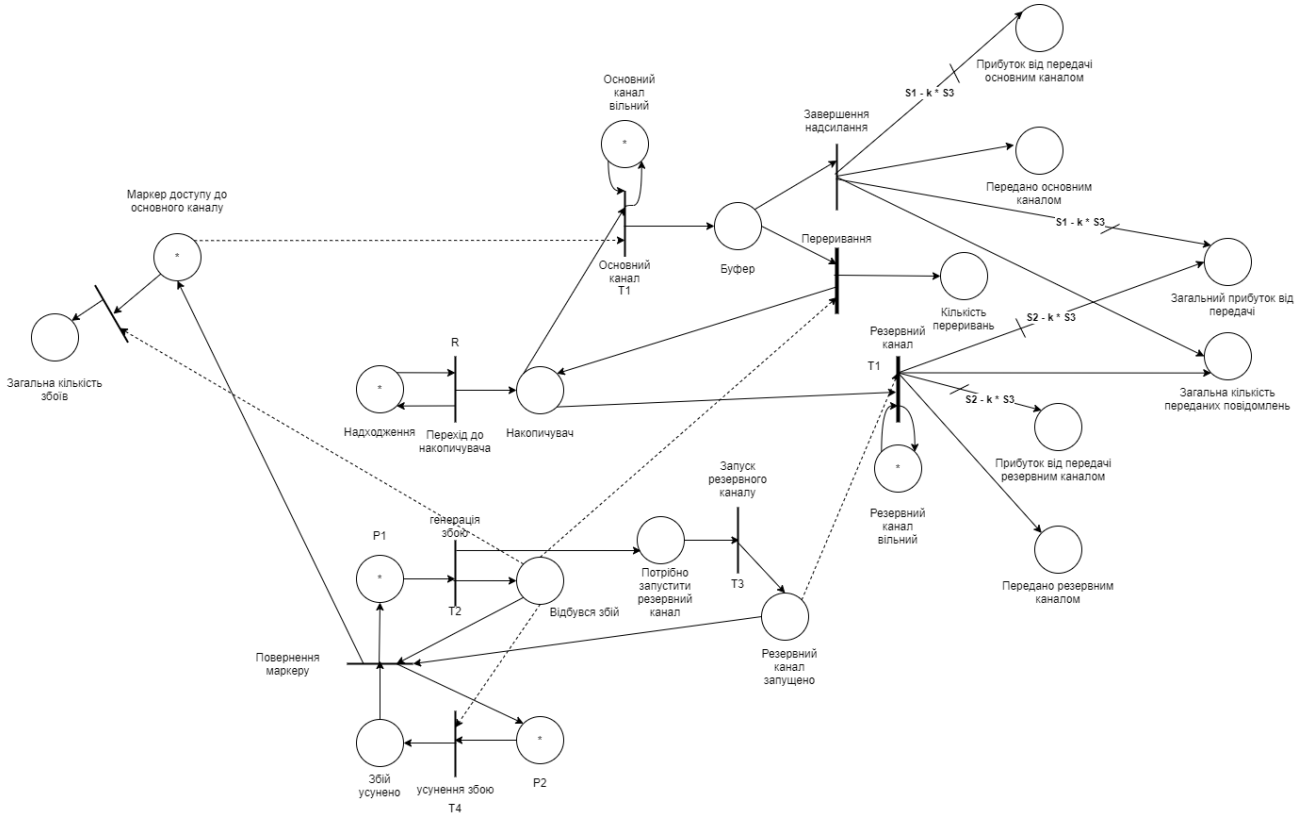

Рисунок 3.1.1 – Структурна схема моделі

3.2 Параметри, вхідні ти вихідні змінні

Відповідно до постановки задачі маємо наступні вхідні параметри:

- Q – час моделювання;
- k – параметр збільшення середнього часу напрацювання на відмову;

Параметри моделі:

- T1 – час передачі повідомлення по основному каналу - 20±8 одиниць часу;
- T2 – час напрацювання на відмову - 400±50 одиниць часу ;
- T3 – час запуску резервного каналу – 5 одиниць часу;
- T4 – час відновлення основного каналу - 100±25 одиниць часу;
- R – час надходження повідомлень - 18±10 одиниць часу;
- S1 – прибуток від передачі одного повідомлення через основний канал – 60 одиниць вартості;

- S2 – прибуток від передачі одного повідомлення через резервний канал – 30 одиниць вартості;
- S3 – коефіцієнт зменшення прибутку – 0.04 одиниць вартості;

Особливості роботи системи:

- після відновлення роботи основного каналу, повідомлення, яке передається на даний момент по резервному каналу, не переходить на передачу до основного каналу;
- після відмови роботи основного каналу, повідомлення, яке передається в даний момент по основному каналу, переходить до резервного каналу;

Допоміжні змінні:

- Кількість повідомлень, переданих по основному каналу – X1;
- Кількість повідомлень, переданих по резервному каналу – X2;

Обмеження змінних:

- $0 \leq k < \infty$

Вихідні параметри:

- Частота переривання;
- Завантаження резервного каналу;
- Загальний прибуток;

3.3 Цільова функція (критерій якості) системи

Метою вирішення даного завдання є оптимізація роботи каналу передачі для збільшення прибутку. Оскільки при збільшені середнього часу напрацювання на відмову на k секунд прибуток з передачі одного повідомлення зменшується на $k \times S_3$, то цільова функція:

$$Z(k) = (S_1 - k \times S_3) \times X_1 + (S_2 - k \times S_3) \times X_2 \rightarrow max$$

Підставимо відомі значення з умови і отримаємо:

$$Z(k) = (60 - k \times 0.04) \times X_1 + (30 - k \times 0.04) \times X_2 \rightarrow max$$

4 РОЗРОБКА СТРУКТУРНОЇ СХЕМИ ІМІТАЦІЙНОЇ МОДЕЛІ ТА ОПИС ЇЇ ФУНКЦІОНУВАННЯ

4.1 Вибір засобу реалізації

Для моделювання роботи системи необхідно мати середовище для моделювання. До даного середовища висуваються наступні вимоги:

- досвідчене керівництво або детальні інструкції для користувача;
- забезпечення достатньої діагностики помилок та простоти їх виправлення;
- природний засіб моделювання;

Для програмної реалізація імітаційної моделі було обрано мову програмування Python. Python — інтерпретована об'єктно-орієнтована мова програмування високого рівня зі строгою динамічною типізацією. Розроблена в 1990 році Гвідо ван Россумом. Структури даних високого рівня разом із динамічною семантикою та динамічним зв'язуванням роблять її ефективною для швидкої розробки програм, а також як засіб поєднання наявних компонентів. Python підтримує модулі та пакети модулів, що сприяє модульності та повторному використанню коду. Інтерпретатор Python та стандартні бібліотеки доступні як у скомпільованій, так і у вихідній формі на всіх основних платформах. В мові програмування Python підтримується кілька парадигм програмування, зокрема: об'єктно-орієнтована, процедурна, функціональна та аспектно-орієнтована. [4]

Також, для даної мови програмування написано багато різних статистичних та математичних пакетів, які працюють доволі ефективно та мають

зручний інтерфейс. Враховуючи, що базовий код для створення мережі Петрі вже був реалізований мною в одній з лабораторних робіт, Python було обрано в якості основної мови для створення програмної реалізації імітаційної моделі.

4.2 Опис програмної реалізації імітаційної моделі

Програмна реалізація складається з 4 основних класів: Position, Transition, Arc, Model.

Таблиця 4.2.1

№	Назва класу	Параметри конструктора	Коментарі
1	Position	o num_of_markers –початкова кількість маркерів в позиції o description – короткий текстовий опис	Клас позицій
2	Transition	o delay – часова затримка o delay_distribution – закон розподілу часової затримки o delay_distribution_params – параметри закону розподілу часової затримки o priority – пріоритет позиції o probability – ймовірність переходу o description – короткий текстовий опис o save_time – індикатор необхідності збереження часу надходження	Клас переходів

		○ compute_time – індикатор необхідності збереження часу виходу	
3	Arc	○ start – початковий елемент дуги ○ end – кінцевий елемент дуги ○ multiplicity – кратність дуги ○ informational – індикатор інформаційної дуги	Клас дуг
4	Model	○ transitions – множина всіх переходів ○ positions – множина всіх позицій ○ arcs – множина всіх дуг ○ modeling_period – час моделювання ○ should_print_intermidiate_result – індикатор необхідності виводу покрокової інформації	Клас імітаційної моделі

Код відповідних класів наведено в Додатку 1.

5 ОЦІНКА АДЕКВАТНОСТІ МОДЕЛІ

5.1 Тестовий прогін

Для початку, виконаємо тестовий запуск нашої моделі з часом моделювання Q = 10000 та поглянемо на отримані результати. Зафіксуємо значення в позиції «Загальний прибуток від передачі» для подальшого порівняння з аналітичним значенням «ідеальної системи».

	description	avg_markers	max_markers	min_markers	result_markers
0	Надходження	0.0524	1.0	0	0.0
1	Накопичувач (черга)	25.9353	55.0	0	55.0
2	Основний канал вільний	0.2212	1.0	0	0.0
3	Резервний канал вільний	0.7968	1.0	0	1.0
4	Буфер	0.0387	1.0	0	0.0
5	Кількість переривань	9.6046	20.0	0	20.0
6	Прибуток від передачі основним каналом	11142.1080	22020.0	0	22020.0
7	Передано основним каналом	185.7018	367.0	0	367.0
8	Загальний прибуток від передачі	12602.7660	25050.0	0	25050.0
9	Загальна кількість переданих повідомлень	234.3904	468.0	0	468.0
10	Прибуток від передачі резервним каналом	1460.6580	3030.0	0	3030.0
11	Передано резервним каналом	48.6886	101.0	0	101.0
12	Маркер доступу до основного каналу	0.7994	1.0	0	1.0
13	Загальна кількість збоїв	9.6276	20.0	0	20.0
14	P2	0.8000	1.0	0	1.0
15	Збій усунено	0.0021	1.0	0	0.0
16	P1	0.0020	1.0	0	0.0
17	Відбувся збій	0.2033	1.0	0	0.0
18	Потрібно запустити резервний канал	0.0020	1.0	0	0.0
19	Резервний канал запущено	0.1913	1.0	0	0.0

Рис. 5.1.1 – Тестовий прогін моделі

На перший погляд, ніяких порушень функціонування моделі не виявлено. Проведемо більш детальне дослідження та знайдемо аналітичну оцінку цільової функції «ідеальної моделі».

5.2 Аналітична оцінка

Аналітична оцінка характеристик роботи системи проводиться для «ідеальної» системи, в якісь відсутня випадковість. Основним чинником випадковості в нашій системі є затримки переходів. Зафіксуємо їх, задавши їм значення, рівне середньому значенню затримки в переході.

Можемо записати наступні аналітичні вирази для «ідеальної системи»:

1. $\frac{X_2}{X_1} = \frac{100/20}{(400+k)/20} = \frac{100}{400+k}$ – аналітичне співвідношення виводилось з наступних міркувань:
   - резервний канал буде працювати поки відновлюється основний канал (100 одиниць часу)
   - основний канал буде працювати до виникнення збою (400), яке залежить від параметру k
   - час обробки 1 повідомлення складає 20 одиниць часу
2. $X_1 + X_2 = \frac{T}{18}$ – аналітичне співвідношення виводилось з наступних міркувань:
   - час моделювання системи дорівнює T
   - час надходження заявка складає 18 одиниць часу
   - ми виконуємо всі заявки, які нам надійшли

Виразимо значення X_1 та X_2 з попередніх рівнянь, отримаємо:

- $X_1 = \frac{T}{18} - \frac{100T}{18\times(500+k)}$
- $X_2 = \frac{100T}{18\times(500+k)}$

Підставимо отримані значення в цільову функцію:

$$Z = 60 \times \left(\frac{T}{18} - \frac{100T}{18 \times (500+k)}\right) + 30 \times \frac{100T}{18 \times (500+k)} - 0.04 \times k \times \frac{T}{18}$$

Зобразимо отриману цільову функцію на графіку:

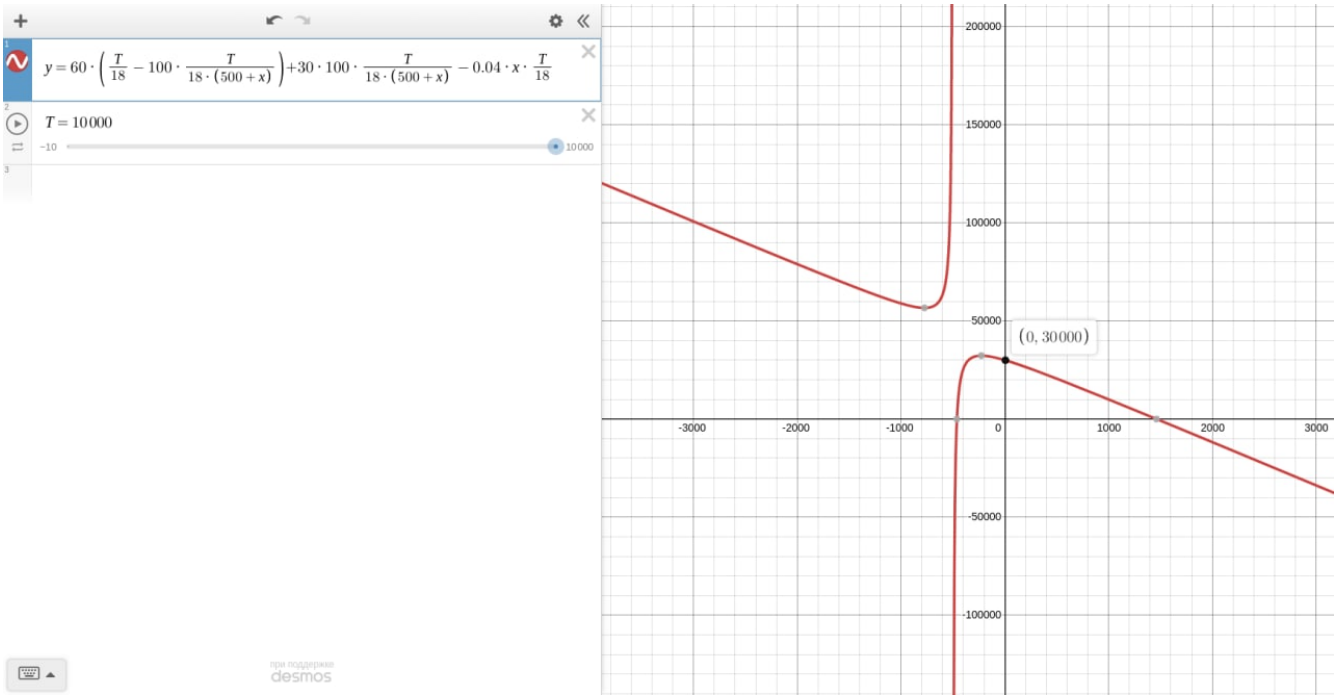

Рис. 5.2.1 – аналітичний графік цільової функції ідеальної системи

Якщо порівнювати отримане в пункті 5.1 значення загального прибутку з аналітичним значенням загального прибутку «ідеальної» системи, то побачимо, що значення ідеальної системи є більшими, ніж фактичні, хоч і схожі. Так і мало б бути, оскільки в ідеальній системі відсутня стохастичність та аналітичні формули є досить спрощеними.

Як ми знаємо, параметр k є обмеженим і не може бути меншим за 0, тому розглянемо лише праву частину графіку. Бачимо, що максимум функції досягається при k=0. Зі збільшенням k, значення прибутку спадає. Схожої поведінки ми очікуємо і від нашої системи.

5.3 Дослідження параметру k

Виконаємо тестовий запуск нашої моделі з часом моделювання Q = 10000 для різних значень параметру k та поглянемо на отримані результати. Нехай, вхідний параметр моделі k може набувати наступних значень від 0 до 50.

	T1	T1_RANGE	T2	T2_RANGE	T3	T4	T4_RANGE	k	s1	s2	s3	modeling_time	Загальний прибуток від передачі	Передано резервним каналом	Частота переривань (N_interupt/Time)	Доля перерваних повідомлень
0	20	8	400	50	5	100	25	0	60.00	30.00	0.04	10000.0	25020.00	106	0.0023	0.04894
1	20	8	401	50	5	100	25	1	59.96	29.96	0.04	10000.0	25181.28	96	0.0020	0.04274
2	20	8	402	50	5	100	25	2	59.92	29.92	0.04	10000.0	24952.72	99	0.0019	0.04077
3	20	8	403	50	5	100	25	3	59.88	29.88	0.04	10000.0	25533.12	95	0.0019	0.04008
4	20	8	404	50	5	100	25	4	59.84	29.84	0.04	10000.0	25064.64	104	0.0020	0.04246
5	20	8	405	50	5	100	25	5	59.80	29.80	0.04	10000.0	25226.00	96	0.0020	0.04255
6	20	8	406	50	5	100	25	6	59.76	29.76	0.04	10000.0	25117.44	97	0.0020	0.04264
7	20	8	407	50	5	100	25	7	59.72	29.72	0.04	10000.0	25247.84	98	0.0019	0.04025
8	20	8	408	50	5	100	25	8	59.68	29.68	0.04	10000.0	24393.76	96	0.0019	0.04158
9	20	8	409	50	5	100	25	9	59.64	29.64	0.04	10000.0	24791.88	102	0.0019	0.04069

Рис. 5.3.1 – Перші 10 ітерацій зміни параметру k (стохастичний варіант)

	T1	T1_RANGE	T2	T2_RANGE	T3	T4	T4_RANGE	k	s1	s2	s3	modeling_time	Загальний прибуток від передачі	Передано резервним каналом	Частота переривань (N_interupt/Time)	Доля перерваних повідомлень
3	20	8	403	50	5	100	25	3	59.88	29.88	0.04	10000.0	25533.12	95	0.0019	0.04008
14	20	8	414	50	5	100	25	14	59.44	29.44	0.04	10000.0	25353.44	98	0.0019	0.03992
22	20	8	422	50	5	100	25	22	59.12	29.12	0.04	10000.0	25348.48	99	0.0019	0.03967
24	20	8	424	50	5	100	25	24	59.04	29.04	0.04	10000.0	25252.08	97	0.0021	0.04403
7	20	8	407	50	5	100	25	7	59.72	29.72	0.04	10000.0	25247.84	98	0.0019	0.04025
5	20	8	405	50	5	100	25	5	59.80	29.80	0.04	10000.0	25226.00	96	0.0020	0.04255
32	20	8	432	50	5	100	25	32	58.72	28.72	0.04	10000.0	25219.44	93	0.0020	0.04193
31	20	8	431	50	5	100	25	31	58.76	28.76	0.04	10000.0	25207.28	96	0.0020	0.04184
1	20	8	401	50	5	100	25	1	59.96	29.96	0.04	10000.0	25181.28	96	0.0020	0.04274
36	20	8	436	50	5	100	25	36	58.56	28.56	0.04	10000.0	25147.44	87	0.0018	0.03797

Рис. 5.3.2 – 10 найкращих значень параметру k за загальним прибутком (стохастичний варіант)

Досить складно аналізувати табличні дані в такому вигляді, тому відобразимо залежність параметру k від прибутку на графіку.

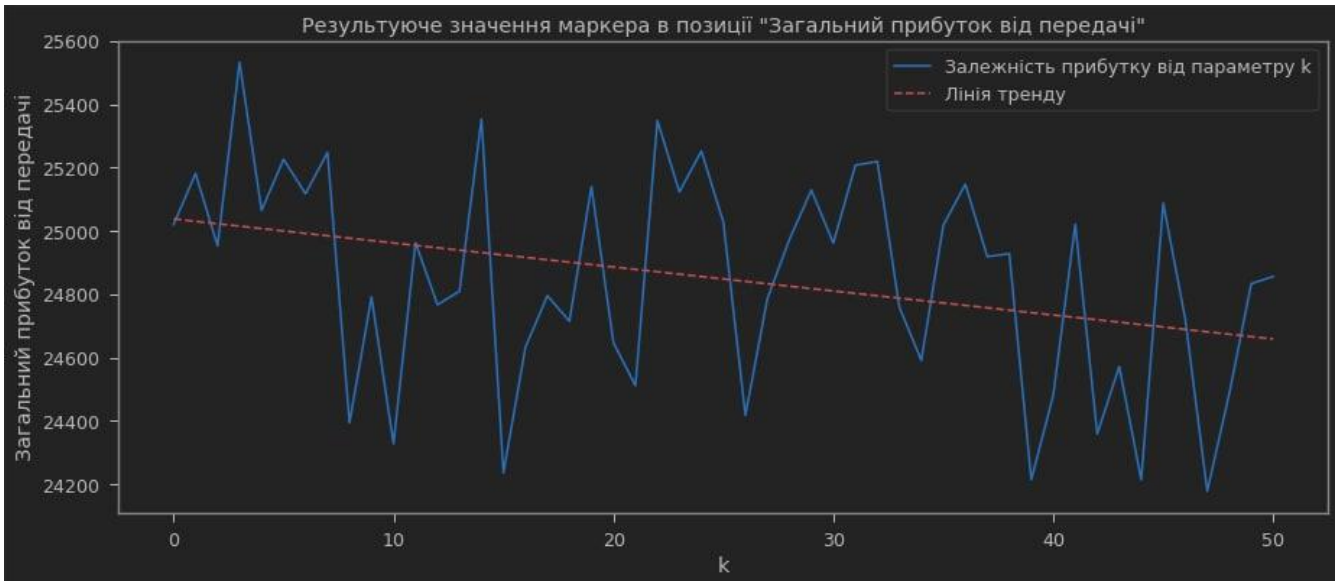

Рис. 5.3.3 – графік залежності загального прибутку від параметру k (стохастичний варіант)

Графік 5.3.3 ілюструє, що зі збільшенням k спостерігається зменшення загального прибутку. Але ми не можемо точно стверджувати даний факт, оскільки наша мережа є стохастичною та має параметри з досить широким діапазоном можливих значень (див. можливі значення параметрів T1, T2, T4, R).

Щоб зменшити стохастичність моделі, пропонується присвоїти кожній затримці в переході її середнє можливе значення, тобто фактично зробити її константою.

	T1	T1_RANGE	T2	T2_RANGE	T3	T4	T4_RANGE	k	s1	s2	s3	modeling_time	Загальний прибуток від передачі	Передано резервним каналом	Частота переривань (N_interupt/Time)	Доля перерваних повідомлень
0	20	0	400	0	5	100	0	0	60.00	30.00	0.04	10000.0	25620.00	96	0.0020	0.04211
1	20	0	401	0	5	100	0	1	59.96	29.96	0.04	10000.0	25571.04	95	0.0019	0.04008
2	20	0	402	0	5	100	0	2	59.92	29.92	0.04	10000.0	25492.16	95	0.0019	0.04017
3	20	0	403	0	5	100	0	3	59.88	29.88	0.04	10000.0	25413.36	95	0.0019	0.04025
4	20	0	404	0	5	100	0	4	59.84	29.84	0.04	10000.0	25394.48	95	0.0019	0.04025
5	20	0	405	0	5	100	0	5	59.80	29.80	0.04	10000.0	25315.80	95	0.0019	0.04034
6	20	0	406	0	5	100	0	6	59.76	29.76	0.04	10000.0	25237.20	95	0.0019	0.04043
7	20	0	407	0	5	100	0	7	59.72	29.72	0.04	10000.0	25158.68	95	0.0019	0.04051
8	20	0	408	0	5	100	0	8	59.68	29.68	0.04	10000.0	25080.24	95	0.0019	0.04060
9	20	0	409	0	5	100	0	9	59.64	29.64	0.04	10000.0	25001.88	95	0.0019	0.04069

Рис. 5.3.4 – Перші 10 ітерацій зміни параметру k

	T1	T1_RANGE	T2	T2_RANGE	T3	T4	T4_RANGE	k	s1	s2	s3	modeling_time	Загальний прибуток від передачі	Передано резервним каналом	Частота переривань (N_interupt/Time)	Доля перерваних повідомлень
0	20	0	400	0	5	100	0	0	60.00	30.00	0.04	10000.0	25620.00	96	0.0020	0.04211
1	20	0	401	0	5	100	0	1	59.96	29.96	0.04	10000.0	25571.04	95	0.0019	0.04008
2	20	0	402	0	5	100	0	2	59.92	29.92	0.04	10000.0	25492.16	95	0.0019	0.04017
3	20	0	403	0	5	100	0	3	59.88	29.88	0.04	10000.0	25413.36	95	0.0019	0.04025
4	20	0	404	0	5	100	0	4	59.84	29.84	0.04	10000.0	25394.48	95	0.0019	0.04025
20	20	0	420	0	5	100	0	20	59.20	29.20	0.04	10000.0	25329.20	95	0.0019	0.03992
5	20	0	405	0	5	100	0	5	59.80	29.80	0.04	10000.0	25315.80	95	0.0019	0.04034
21	20	0	421	0	5	100	0	21	59.16	29.16	0.04	10000.0	25251.00	95	0.0019	0.04000
6	20	0	406	0	5	100	0	6	59.76	29.76	0.04	10000.0	25237.20	95	0.0019	0.04043
22	20	0	422	0	5	100	0	22	59.12	29.12	0.04	10000.0	25172.88	95	0.0019	0.04008

Рис. 5.3.5 – 10 найкращих значень параметру k за загальним прибутком

Аналогічно до попередніх результатів, відобразимо залежність прибутку від параметру k на графіку.

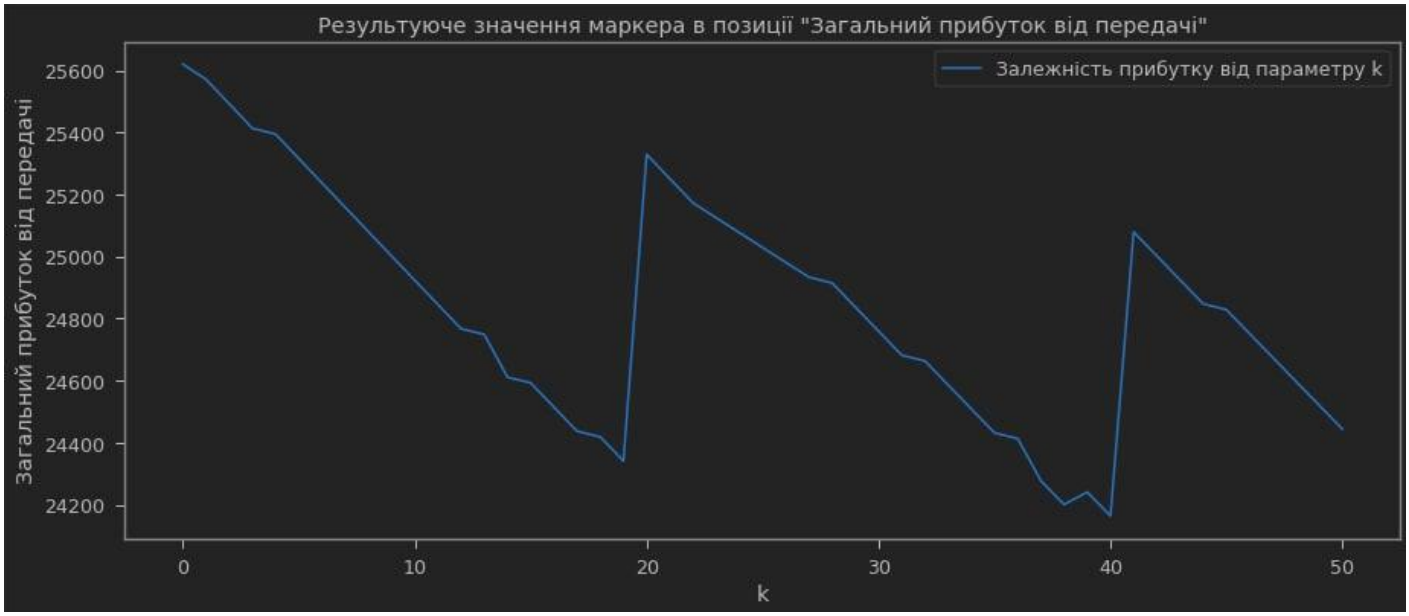

Рис. 5.3.6 – графік залежності загального прибутку від параметру k

На даному графіку чітко видно, що зі збільшенням параметру k, загальний прибуток від передачі повідомлень спадає, що нагадує поведінку «ідеальної системи». Також на даному графіку спостерігаються періодичні піки.

5.4 Перевірка результатів моделювання на стаціонарність

5.4.1 Перевірка на відтворюваність результату (за контрольним експериментом)

Для початку, проведемо 50 експериментів та знайдемо по ним середнє значення відгуку μ та стандартне відхилення σ заданої моделі.

$\mu = 25183.8$

$\sigma = 273.76$

Потім виконаємо один контрольний експеримент та подивимось чи значення його відгуку схоже на середнє значення відгуку, отриманого по 50 експериментам. Схожість будемо перевіряти по факту потрапляння в діапазон $\mu \pm \sigma$.

$\beta = 25080$ – отримане значення прибутку за контрольним експериментом

Дане значення знаходиться в діапазоні μ±σ, тому можемо сказати що не в даному експерименті не було виявлено неадекватної поведінки системи.

5.4.2 Перевірка на відтворюваність результату (перевірка рівності середніх значень)

Для початку, проведемо 50 прогонів та знайдемо для кожного прогону значення відгуку моделі. Потім, виконаємо ще 3 прогони та знайдемо для кожного з них значення відгуку моделі.

Перевіримо гіпотезу про рівність середніх значень відгуків в першому експерименті та в другому, використовуючи T-test з двобічною альтернативою. Маємо наступні значення статистики та p-value:

- stat_value = -0.07165
- p-value = 0.94316

  Отже, можемо сказати що на рівні значущості 0.05 ми можемо прийняти гіпотезу про рівність середніх значнь.

6 РЕЗУЛЬТАТИ ЕКСПЕРИМЕНТІВ НА МОДЕЛІ

6.1 План експериментів

Проведемо дисперсійний аналіз впливу факторів на відгук моделі, знайдемо функцію розподілу часу перебування повідомлення в системі та поглянемо на статистичні значення, які потрібно було знайти за умовою задачі.

Для проведення дисперсійного аналізу нам потрібно встановити кількість прогонів моделі, яку необхідно здійснити. Встановити це можна, використовуючи одну з двох способів:

- нерівність Чебишева:

$$p_1 = \frac{\sigma^2}{\varepsilon^2(1-\beta)} + 1$$

- центральна гранична теорема:

$$p_2 = \frac{\sigma^2 \times t_\varphi^2}{\varepsilon^2} + 1$$

Проведемо 50 прогонів моделі з часом моделювання Q = 10000, запам'ятавши значення відгуку на кожному прогоні. Тобі оцінимо стандартне відхилення та середнє значення даної величини. Задамо точність рівною 0,01 і отримаємо:

$p_1 = 4.4 \approx 5$

$p_2 = 18.6 \approx 19$

Надалі, будемо використовувати значення p_2 при виконанні факторного аналізу.

Для визначення часу моделювання, потрібно знайти точку, коли наша модель переходить в стаціонарний стан. Пошук переходу до стаціонарного стану будемо визначати, використовуючи середню кількість маркерів в позиції «Резервний канал вільний». Проведемо 2 тестові прогони та відобразимо їх на графіку.

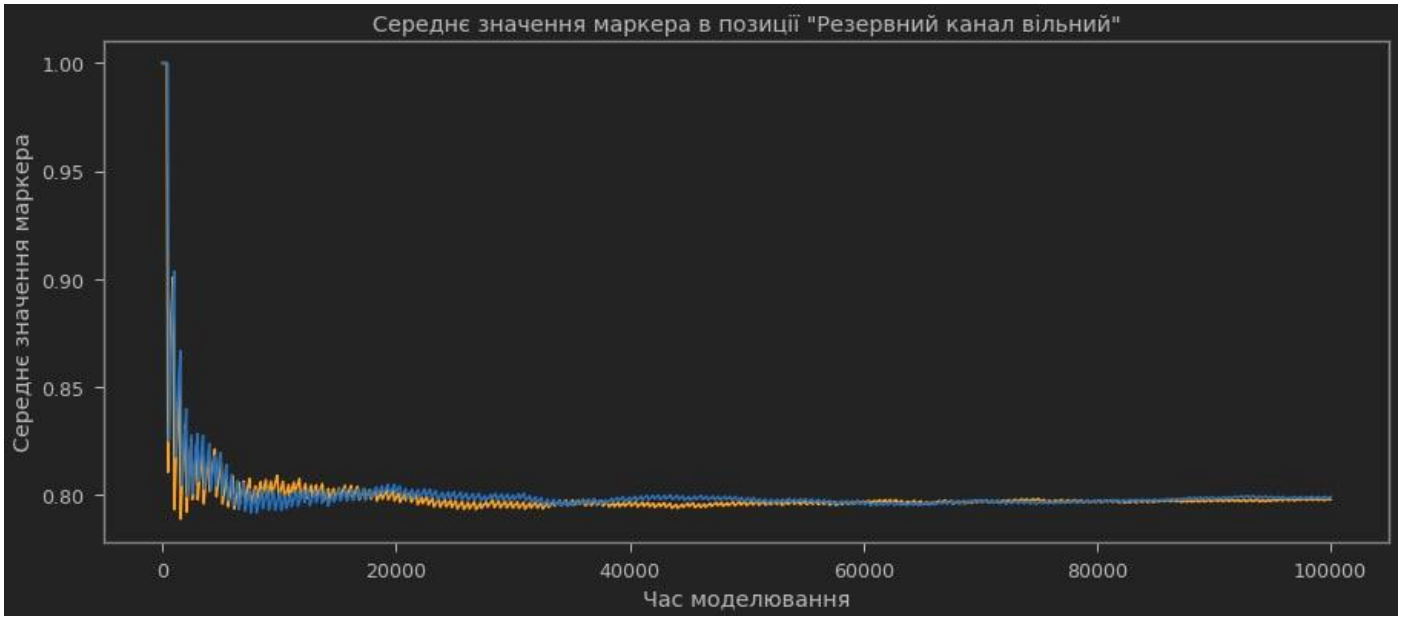

Рис. 6.1.1 – середнє значення маркера в позиції «Резервний канал вільний»

За графіком було обрано час моделювання Q = 50000.

6.2 Аналіз і оцінка результатів

6.2.1 Дисперсійний аналіз

Визначивши кількість прогонів та час моделювання, ми можемо провести факторний аналіз впливу на відгук моделі. Параметром, від якого залежить результат моделювання, будемо вважати k, а відгуком моделі значення в позиції «Загальний прибуток від передачі».

З плану експериментів ми дізнались, що кількість прогонів становить $p = 19$, а час моделювання Q = 50000 одиниць часу. Параметр k будемо змінювати в діапазоні від 0 до 50 з кроком 5.

За результатами проведення дисперсійного аналізу, отримали наступний результат:

	sum_sq	df	F	PR(>F)
C(k)	2.372201e+08	10.0	55.053331	8.730890e-52

Рис. 6.2.1.1 – результат роботи дисперсійного аналізу

Для заданого набору маємо $F_{\text{кр}} = 1{,}8787$, тоді:

$$F_{\text{кр}} < F$$

Маємо, що розрахований критерій Фішера є більшим за критичне значення, отже можна зробити висновок, що влив фактора є значущим.

6.2.2 Закон розподілу часу перебування повідомлення в системі

Для визначення закону розподілу часу перебування повідомлення в системі, візьмемо час моделювання Q = 100000 та позбудемось випадковості.

Побудуємо гістограму для отриманого результату:

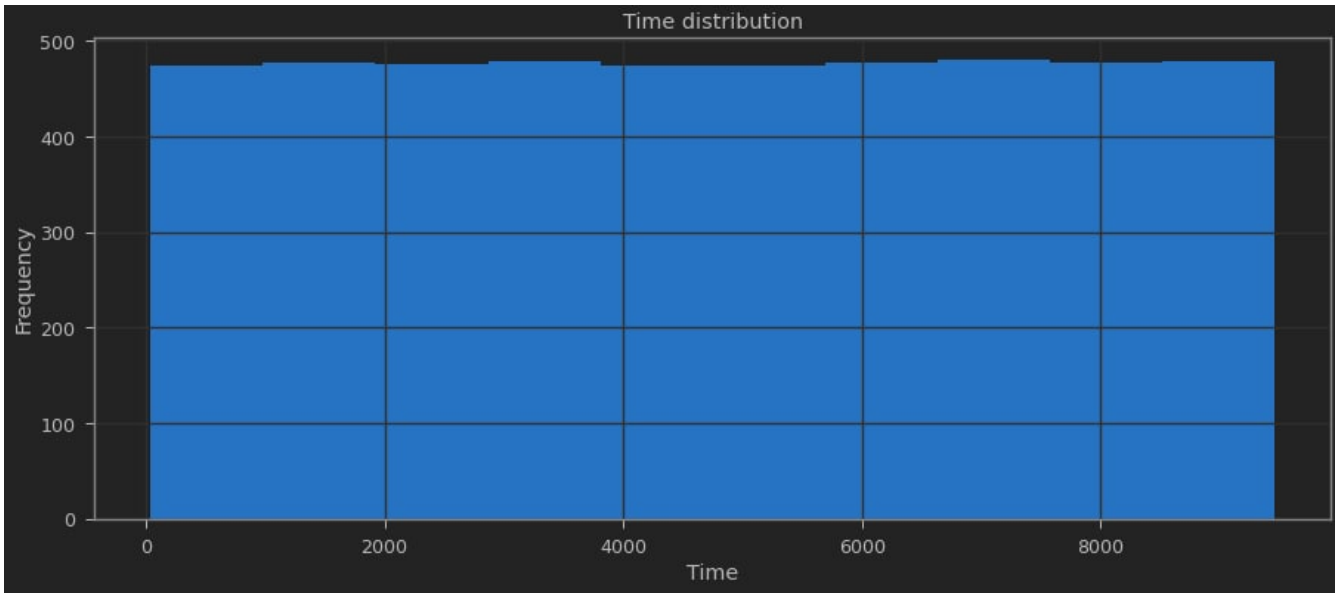

Рис. 6.2.2.1 – гістограма часу перебування в системі

Видно, що розподіл схожий на рівномірний. Давайте перевіримо це, використовуючи критерій χ^2 та тест Колмогорова-Смірнова. Висуваємо гіпотезу:

- H_0: дані мають рівномірний розподіл з заданим параметром
- H_1: H_0 не виконується

```
Можемо прийняти нульову гіпотезу про розподіл данних з заданим параметром.
Значення статистики:
        - stat_val = 0.17075
        - p-value = 0.99999
```

Рис. 6.2.2.2 – результати критерію X^2

```
Можемо прийняти нульову гіпотезу про розподіл данних з заданим параметром.
Найкращий розподіл: uniform
Значення параметрів:
        - parameters = (40.0, 9419.0)
Значення статистики:
        - p-value = 1.0
```

Рис. 6.2.2.3 – результати тесту Колмагорова-Смірнова

Обидва тести досить впевнено приймають нульову гіпотезу на рівні значущості 0,05.

Тепер повторимо експеримент з Q=500000, але повернемо випадковість до нашої моделі.

Побудуємо гістрограму для отриманого результату:

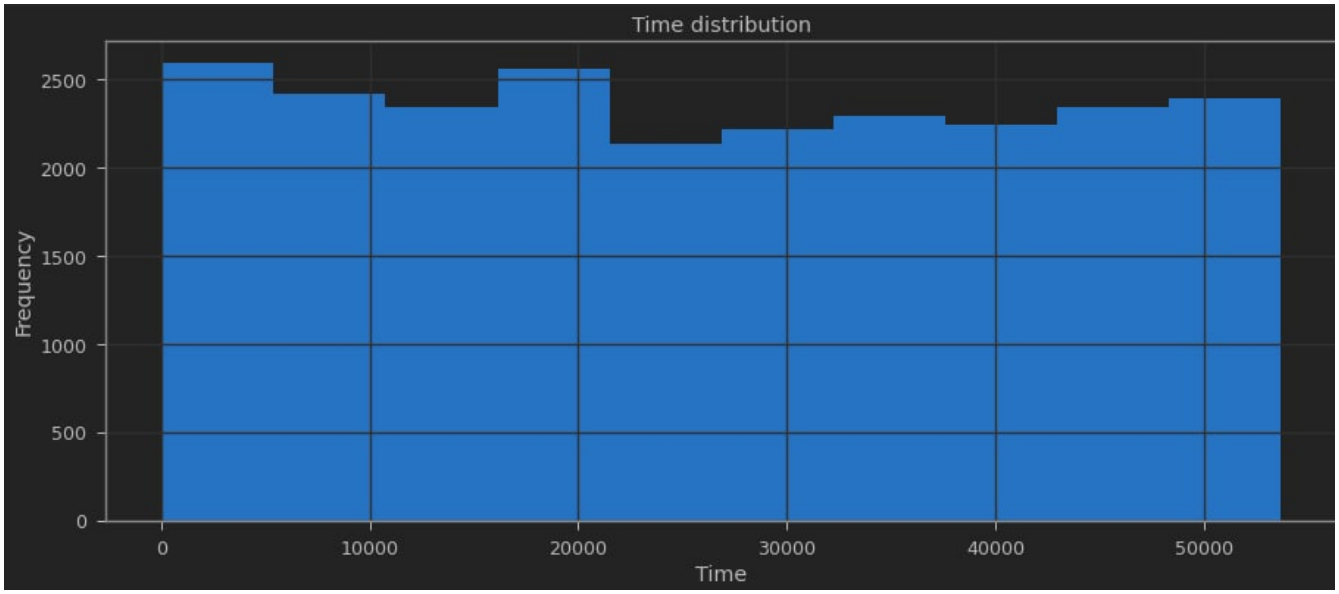

Рис. 6.2.2.4 – гістрограма часу перебування в системі (стохастичний варіант)

Як бачимо, розподіл нагадує рівномірний. Давайте перевіримо це, використовуючи критерій χ^2 та тест Колмогорова-Смірнова. Висуваємо гіпотезу:

- H_0: дані мають рівномірний розподіл з заданим параметром
- H_1: H_0 не виконується

```
Не можемо прийняти нульову гіпотезу на рівні значемості alpha=0.05.
Значення статистики:
        - stat_val = 81.15502
        - p-value = 0.0
```

Рис. 6.2.2.5 – результати критерію X^2

```
Не можемо прийняти нульову гіпотезу на рівні значемості alpha=0.05.
Значення статистики:
        - p-value = 4.770493024825132e-11
```

Рис. 6.2.2.6 – результати тесту Колмагорова-Смірнова

Перевіримо час перебування в системі на відповідність іншим розподілам, використовуючи тест Колмагорова-Смірнова.

```
- p value for dweibull = 1.046689006330884e-54
- p value for expon = 0.0
- p value for norm = 1.5973504372520088e-75
- p value for uniform = 4.770493024825132e-11

Best fitting distribution: uniform
Best p value: 4.770493024825132e-11
Parameters for the best fit: (38.0, 53660.0)
```

Рис. 6.2.2.7 – результати тесту Колмагорова-Смірнова для інших законів розподілу

Як бачимо, з перевірених законів розподілу найбільше підходить рівномірний, але ми не можемо прийняти гіпотезу про його відповідність на рівні значущості 0,05.

Такий результат можна пояснити наявністю в моделі великої кількості випадкових величин, які мають широкий діапазон можливих значень. Зі збільшенням кількості ітерацій, зростає і p-value для рівномірного розподілу, але за розумний час роботи імітаційного алгоритму в мене не вийшло досягнути потрібних результатів.

6.2.3 Визначення завантаження резервного каналу та частоти переривань

Визначення даних показників здійснювалось шляхом підрахунку статистичних значень всередині мережі Петрі для окремих позицій.

Частота переривань визначалась двома способами:

- Кількість перерваних повідомлень / час моделювання
- Кількість перерваних повідомлень / загальна кількість доставлених повідомлень

Останній спосіб визначення частоти фактично є долею перерваних повідомлень серед усіх.

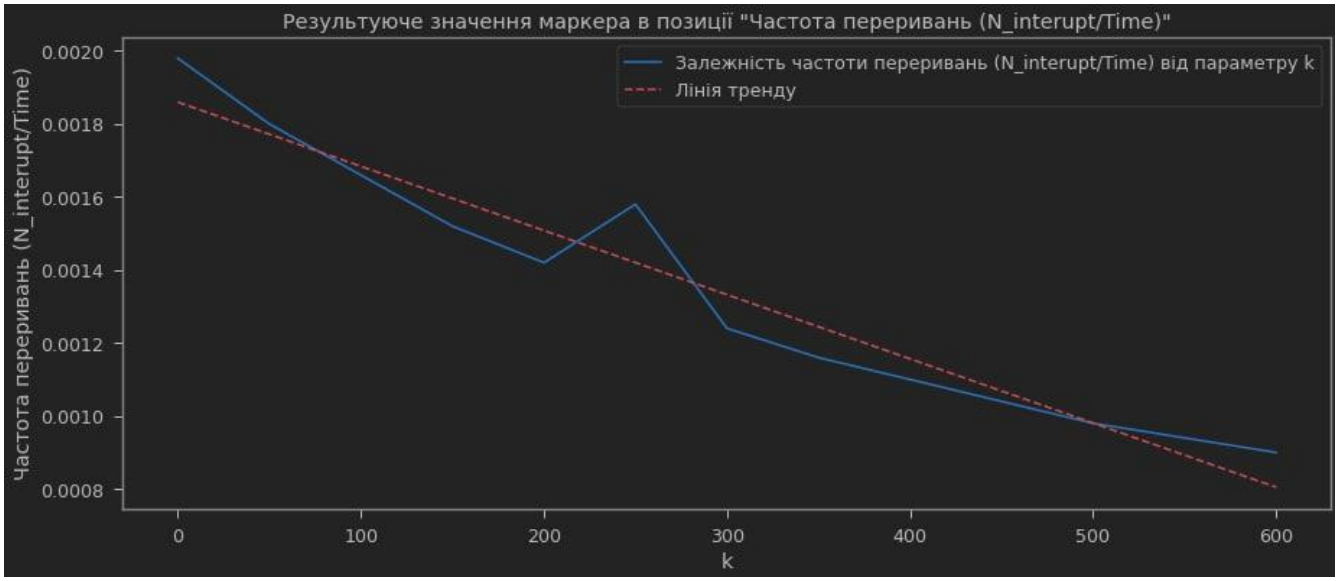

Рис. 6.2.3.1 – залежність частоти переривань (N/T) від параметру k

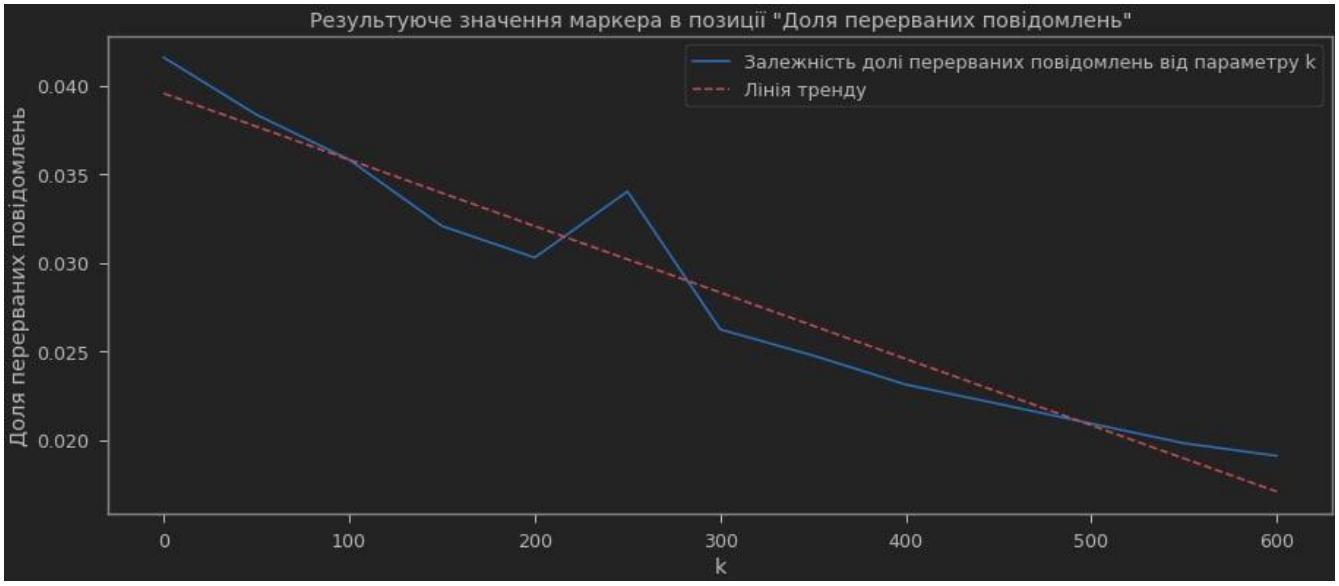

Рис. 6.2.3.2 – залежність долі перерваних повідомлень від параметру k

Судячи з графіків, дані величини поводять себе майже однаково, в залежності від зміни параметру k та сталому часі моделювання Q. Можна відмітити, що при збільшенні параметру k спадає як і доля перерваних повідомлень, та і частота переривання.

Завантаженість резервного каналу обраховується, як загальна кількість маркерів в позиції «Передано резервним каналом».

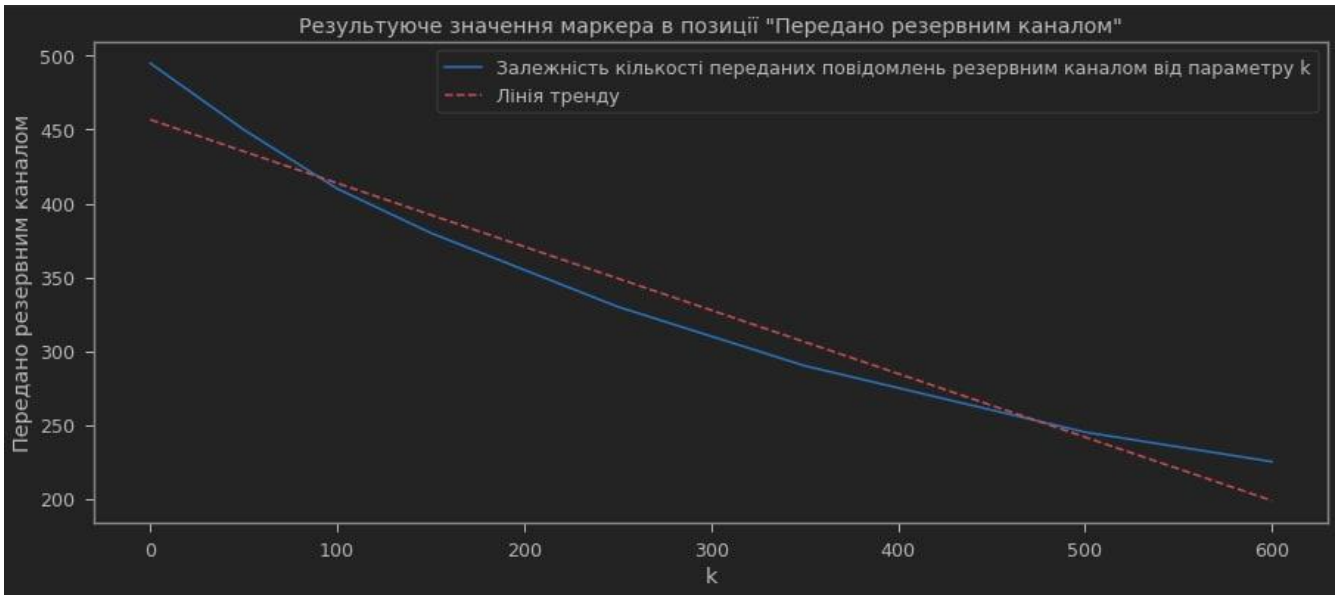

Рис. 6.2.3.3 – залежність повідомлень, переданих резервним каналом, від параметру k

Судячи з графіків, при збільшенні параметру k, спостерігається зменшення кількості повідомлень, що передає резервний канал.

7 ВИСНОВКИ

В даній роботі була розглянута задача про роботу магістралі з передачі повідомлень двома каналами – основний та резервний. В ході виконання роботи було досліджено вплив збільшення часу напрацювання на відмову на роботу системи та загальний прибуток. Було проведено детальний аналіз та обрано оптимальні значення для роботи системи.

8 ПЕРЕЛІК ПОСИЛАНЬ

1. Стеценко І.В. Моделювання систем. Навчальний посібник
2. https://stud.com.ua/98833/informatika/imitatsiyne_modelyuvannya
3. http://web.kpi.kharkov.ua/auts/wp-content/uploads/sites/67/2017/02/MOCS_Kachanov_posobie.pdf
4. https://uk.wikipedia.org/wiki/Python

Додаток. Лістинг програми

Додаток 1.

```
class Position:
    def __init__(self, num_of_markers=0, description=None):
        self.num_of_markers = num_of_markers
        self.description = description

    def add_markers(self, amount_of_added_markers):
        self.num_of_markers += amount_of_added_markers

    def remove_markers(self, amount_of_removed_markers):
        self.num_of_markers -= amount_of_removed_markers

    def __repr__(self):
        return self.description

class Transition:
    def __init__(self, delay=0, delay_distribution=None, delay_distribution_params=None, priority=1, probability=None, description=None, save_time=False, compute_time=False):
        self.delay = delay
        self.delay_distribution = delay_distribution
        self.delay_distribution_params = delay_distribution_params
        self.priority = priority
        self.probability = probability
        self.description = description

        self.input_arcs = []
        self.output_arcs = []
```

```
    self.marker_release_timestamps = {}

    self.save_time = save_time
    self.compute_time = compute_time

def get_random_delay_period(self):
    if self.delay_distribution is Distribution.EXP:
        result = np.random.exponential(**self.delay_distribution_params)
    elif self.delay_distribution is Distribution.UNIFORM:
        result = np.random.uniform(**self.delay_distribution_params)
    elif self.delay_distribution is Distribution.NORMAL:
        result = np.random.normal(**self.delay_distribution_params)
    elif self.delay_distribution is Distribution.POISSON:
        result = np.random.poisson(**self.delay_distribution_params)

    global N_ROUND
    return round(result, N_ROUND) #!!!!!

def is_available(self):
    return all(arc.is_available() for arc in self.input_arcs)

def check_markers_with_delay(self, current_time):
    global N_ROUND
    if round(current_time, N_ROUND) in self.marker_release_timestamps.keys():
        del self.marker_release_timestamps[round(current_time, N_ROUND)]

        global queue_time, process_time

        if self.compute_time:
```

```
                process_time.append(current_time - queue_time.pop(0))


            for arc in self.output_arcs:
#                 print('Move to:', arc)
                arc.move_to()


    def make_a_transition(self, current_time):
        delay = self.delay if self.delay_distribution is None else self.get_random_delay_period()

        global queue_time, process_time
        if self.save_time:
#                 print('save:', self)
            queue_time.append(current_time)

        for arc in self.input_arcs:
#             print('Move from:', arc)
            arc.move_from(current_time, delay)


    def __repr__(self):
        return self.description


class Arc:
    def __init__(self, start, end, multiplicity=1, informational=False):
        self.start = start
        self.end = end
        self.multiplicity = multiplicity
        self.informational = informational

        if isinstance(end, Transition):
```

```
            end.input_arcs.append(self)
        elif isinstance(start, Transition):
            start.output_arcs.append(self)

    def is_available(self):
        return self.start.num_of_markers >= self.multiplicity

    def move_from(self, current_time, delay):
        if not self.informational:
            self.start.remove_markers(self.multiplicity)

        global N_ROUND
        release_timestamp = round(current_time + delay, N_ROUND)
        if release_timestamp in self.end.marker_release_timestamps:
            self.end.marker_release_timestamps[release_timestamp] += self.multiplicity
        else:
            self.end.marker_release_timestamps[release_timestamp] = self.multiplicity


    def move_to(self):
        self.end.add_markers(self.multiplicity)

    def __repr__(self):
        return f"{self.start} -> {self.end}"

class Model:
    def __init__(self, transitions, positions, arcs, modeling_period, should_print_intermediate_results=False):
        self.transitions = transitions
        self.positions = positions
```

```
    self.arcs = arcs
    self.modeling_period = modeling_period
    self.should_print_intermediate_results = should_print_intermediate_results

    global N_ROUND
    self.delta_t = round(1/10**N_ROUND, N_ROUND)
    self.time = round(0.000, N_ROUND) if N_ROUND > 1 else 0

    # Stats
    self.position_marker_stats = {position: [] for position in positions}

def run(self, return_target=False, return_avg_markers=None):
    if self.should_print_intermediate_results:
        self.print_intermediate_results()

    while self.time < self.modeling_period:
        we_have_available_transition = any(transition.is_available() or len(transition.marker_release_timestamps) > 0
                                           for transition in self.transitions)

        if not we_have_available_transition:
            break

        resolved_transitions = self.get_transitions_with_resolved_conflicts()

        for transition in resolved_transitions:
            if transition.is_available():
                transition.make_a_transition(self.time)

        for transition in self.transitions:
```

```
            transition.check_markers_with_delay(self.time)


        self.calc_stats()


        self.time += self.delta_t


        if self.should_print_intermediate_results:
            self.print_intermediate_results()
    self.result, self.result_target_value = self.print_results(return_target)


    if return_avg_markers:
        return self.position_marker_stats[[position for position in self.positions if position.description ==
return_avg_markers][0]]
    else:
        return self.result_target_value if return_target else self.result


def get_transitions_with_resolved_conflicts(self):
    resolved_list = []
    conflict_list = []


    for position in self.positions:
        output_arcs_for_position = [*filter(lambda arc: arc.start is position, self.arcs)]
        resolved = [*filter(lambda trans: trans.is_available(),
                            map(lambda arc: arc.end, output_arcs_for_position))
                    ]
        if len(resolved) > 1:
            conflict_list.append(resolved)


    conf_list_flatten = set(np.array(conflict_list).flatten().tolist())
    for position in self.positions:
```

```
            output_arcs_for_position = [*filter(lambda arc: arc.start is position, self.arcs)]
            resolved = [*filter(lambda trans: trans.is_available(),
                                map(lambda arc: arc.end, output_arcs_for_position))
                        ]

            if not conf_list_flatten.intersection(resolved):
                resolved_list += resolved
        del conf_list_flatten

        # вирішуємо конфлікти
        for conflict in conflict_list:
            # конфлікт за ймовірністю
            if conflict[0].probability is not None:
                p = [c.probability for c in conflict]
                resolved = np.random.choice(conflict, p=p)
            # конфлікт за пріоритетом
            elif len(np.unique([c.priority for c in conflict if c.priority is not None])) > 1:
                priority = [c.priority for c in conflict]
                resolved = conflict[np.argmax(priority)]
            else:
                resolved = np.random.choice(conflict)

            resolved_list += [resolved]

#         np.random.shuffle(resolved_list)
        return list(set(resolved_list))

    def calc_stats(self):
        for position in self.positions:
```

```
            self.position_marker_stats[position].append(position.num_of_markers)


    def get_result_stats(self):
        result_stats = {position: {} for position in self.positions}

        for position, stats in self.position_marker_stats.items():
            result_stats[position]["avg_markers"] = round(sum(stats) / len(stats), 5)
            result_stats[position]["max_markers"] = max(stats)
            result_stats[position]["min_markers"] = min(stats)
            result_stats[position]["result_markers"] = stats[-1]

            if position.description == 'Загальний прибуток від передачі':
                result_revenue_value = stats[-1]
            elif position.description == 'Передано резервним каналом':
                result_reserv_sent_msg = stats[-1]
            elif position.description == 'Кількість переривань':
                result_interuption_cnt = stats[-1]
            elif position.description == 'Загальна кількість переданих повідомлень':
                result_msg_cnt = stats[-1]

        return result_stats, {'Загальний прибуток від передачі': result_revenue_value,
                              'Передано резервним каналом': result_reserv_sent_msg,
                              'Частота переривань (N_interupt/Time)': round(result_interuption_cnt / self.time, 5),
                              'Доля перерваних повідомлень': round(result_interuption_cnt / result_msg_cnt, 5)}


    def print_results(self, return_target):
        result_stats, result_target_value = self.get_result_stats()
        formatted_records = []
```

```
    for position, stats in result_stats.items():
        record = stats
        record["description"] = position.description


        formatted_records.append(record)


    if not return_target:
        print(f"Modeling time spent: {self.time}")
        display(pd.DataFrame(
            formatted_records,
            columns=[
                "description",
                "avg_markers",
                "max_markers",
                "min_markers",
                "result_markers"
            ]
        ))


    return result_stats, result_target_value


def print_intermediate_results(self):
    print(f"Timestamp: {self.time}")
    print("--- Positions ---")
    for position in self.positions:
        print(position, position.num_of_markers)
    print("--- Transitions ---")
    for t in self.transitions:
        print(t, t.marker_release_timestamps)
```

print()