Національний технічний університет України «КПІ імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації та управління

**КУРСОВА РОБОТА**

з дисципліни «Моделювання систем»

на тему: Визначення найбільш вигідної роздрібної ціни продажу товару в магазині з урахуванням плати на кредит

Виконав студент 4 курсу Перевірив керівник

Група ІС-72 Новікова П.А.

Цилюрик Дмитро Сергійович (прізвище, ім'я, по батькові)

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

\_\_\_\_\_\_\_\_\_\_\_ підпис Робота захищена з оцінкою

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Номер залікової книжки Члени комісії по захисту:

ІС-7227 \_\_\_\_\_\_\_\_\_\_ підпис

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище, ім'я, по батькові)

\_\_\_\_\_\_\_\_\_\_ підпис

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище, ім'я, по батькові)

**Київ 2020**

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

Кафедра автоматизованих систем обробки інформації та управління

Дисципліна «Моделювання систем»

Спеціальність Інформаційні системи та технології

Курс 4 Група ІС-72 Семестр 7

**ЗАВДАННЯ**

на курсову роботу студента

Цилюрика Дмитра Сергійовича

(прізвище, ім'я, по батькові)

1.Тема роботи: Визначення найбільш вигідної роздрібної ціни продажу товару в магазині з урахуванням плати на кредит

2. Термін здачі студентом закінченої роботи "26" грудня 2020 р.

3. Вихідні дані до проекту

Варіант №21-2

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 немає.

6. Дата видачі завдання "1" вересня 2020 р.

# КАЛЕНДАРНИЙ ПЛАН

|  |  |  |  |
| --- | --- | --- | --- |
| № | Назва етапів курсового проекту (роботи) | Термін виконання етапів роботи | Примітка |
| 1 | Отримання завдання | 13.09.2020 |  |
|  | Формулювання теми курсової роботи | 20.09.2020 |  |
| 2 | Аналіз можливих методів вирішення поставленого завдання | 04.10.2020 |  |
| 3 | Розробка концептуальної моделі | 04.10.2020 |  |
| 4 | Перший контроль за процесом виконання курсового проекту (роботи), консультація у викладача | 06.10.2020 |  |
| 5 | Опис імітаційної моделі | 09.10.2020 |  |
| 6 | Опис програмної реалізації імітаційної моделі | 20.11.2020 |  |
| 7 | Другий контроль за процесом виконання курсового проекту (роботи), консультація у викладача | 25.11.2020 |  |
| 8 | Аналіз та оцінка результатів | 07.12.2020 |  |
| 9 | Оформлення пояснювальної записки | 20.12.2020 |  |
| 10 | Здача пояснювальної записки | 20.12.2020 |  |
| 11 | Захист курсового проекту (роботи) | 24.12.2020 |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Студент\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис)

Керівник \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

«\_\_» \_\_\_\_\_\_\_\_\_\_ 2020 р.

# РЕФЕРАТ

Курсова робота: 48 с., 11 рис., 5 табл., 1 додаток, 4 джерела літератури.

Об'єкт дослідження – магазин продажу товарів.

Мета роботи – вибір і обґрунтування економічно ефективної роздрібної ціни для продажу товару.

Метод дослідження – імітаційне моделювання роботи магазину.

Проведено дослідження і розроблена програмна реалізація імітаційної моделі магазину. Розроблено план і проведені експерименти з імітаційною моделлю. Результати моделювання використані для визначення ефективної роздрібної ціни продажу товару.

ІМІТАЦІЙНА МОДЕЛЬ, МОВА МОДЕЛЮВАННЯ, ПЛАНУВАННЯ ЕКСПЕРИМЕНТІВ, МАЗАГИН

# ЗМІСТ

[КАЛЕНДАРНИЙ ПЛАН 3](#_Toc59401824)

[РЕФЕРАТ 4](#_Toc59401825)

[ЗМІСТ 5](#_Toc59401826)

[ВСТУП 6](#_Toc59401827)

[ПОСТАНОВКА ЗАДАЧІ 7](#_Toc59401828)

[1 АНАЛІЗ МОЖЛИВИХ МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ПОСТАВЛЕНОГО ЗАВДАННЯ 8](#_Toc59401829)

[1.1. Методи моделювання 8](#_Toc59401830)

[1.2. Засоби автоматизації 10](#_Toc59401831)

[2 РОЗРОБКА КОНЦЕПТУАЛЬНОЇ МОДЕЛІ 11](#_Toc59401832)

[2.1. Ціль моделювання 11](#_Toc59401833)

[2.2. Вхідні та вихідні дані, параметри моделі 11](#_Toc59401834)

[2.3. Обмеження 12](#_Toc59401835)

[2.4. Цільова функція 12](#_Toc59401836)

[2.5. Схема моделі 12](#_Toc59401837)

[2.6. Визначення структури моделі 13](#_Toc59401838)

[3 РОЗРОБКА ІМІТАЦІЙНОЇ МОДЕЛІ ТА ОПИСУ ЇЇ ФУНКЦІОНУВАННЯ 15](#_Toc59401839)

[3.1. Опис імітаційної моделі 15](#_Toc59401840)

[3.2. Опис програмної реалізації імітаційної моделі 18](#_Toc59401841)

[Вибір засобу програмної реалізації 18](#_Toc59401842)

[Опис блоків програмного коду 19](#_Toc59401843)

[4 ОЦІНКА АДЕКВАТНОСТІ МОДЕЛІ 20](#_Toc59401844)

[5 ОРГАНІЗАЦІЯ ЕКСПЕРИМЕНТІВ З МОДЕЛЛЮ 21](#_Toc59401845)

[5.1. План експериментів 21](#_Toc59401846)

[5.2. Аналіз і оцінка результатів 21](#_Toc59401847)

[ВИСНОВКИ 25](#_Toc59401848)

[ПЕРЕЛІК ПОСИЛАНЬ 26](#_Toc59401849)

[ДОДАТКИ 27](#_Toc59401850)

# ВСТУП

Метою даної роботи виступає визначення оптимального значення роздрібної ціни в магазині з продажу товарів. Клієнти, які надходять до магазину, з певною ймовірністю можуть купити товар, або покинути даних магазин.

Для вирішення поставленої задачі є доцільним використання методу конструювання моделей, а саме імітаційне моделювання. При цьому, основним завданням буде побудова моделі, структура якої схожа до структури досліджуваного об’єкта. Таким чином будуть застосовані засоби відтворення поведінки системи на моделі, відображення властивостей середовища, в якому функціонує досліджувана система. При цьому, даний підхід дозволяє проводити ряд експериментів на побудованій моделі.

Імітаційне моделювання може використовуватись як універсальний підхід для прийняття рішення в умовах невизначеності для врахування в моделях факторів, що досить складно формалізуються, а також на практиці використовувати основні принципи системного підходу для вирішення практичних задач.

# ПОСТАНОВКА ЗАДАЧІ

Приватний магазин купує партію з N одиниць товару за оптовою ціною s одиниць вартості. Гроші на придбання товару власник магазину бере у кредит. Процентна ставка за кредит становить k відсотків від суми непогашеного кредиту в день (табл. 3.23). Потік покупців в магазин - пуассонівський з параметром λ (1/хв). Імовірність того, що покупець не буде купувати товар і відразу піде з магазину залежить від довжини черги і роздрібної ціни товару: , де - імовірність того, що довжина черги "влаштовує" покупця, - імовірність того, що він купить товар за цією роздрібною ціною. Час обслуговування покупця в магазині - рівномірно розподілена в інтервалі А±В випадкова величина.

***Визначити:*** найбільш вигідну роздрібну ціну продажу товару в магазині з урахуванням плати за кредит.

***Примітка:*** вважати, що після продажу одиниці товару власник магазину відразу ж “гасить” борг (якщо він ще є), це приводить до того, що величина кредиту і виплат процентів по ньому зменшується.

Таблиця 1 – Вхідні дані задачі відповідно до варіанту

|  |  |
| --- | --- |
| Параметри | Значення |
|  | 2 |
|  | 500 |
|  | 1000 |
|  | 0.033 |
|  | 0.25 |
|  | 0.25 |
|  | 0.2 |
|  | 0.2 |
|  | 0.1 |
|  | 0.3 |
|  | 0.3 |
|  | 0.2 |
|  | 0.1 |
|  | 0.06 |
|  | 0.04 |
|  | 30 |
|  | 5 |

# АНАЛІЗ МОЖЛИВИХ МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ПОСТАВЛЕНОГО ЗАВДАННЯ

## Методи моделювання

Методи моделювання, які вирішують задачу ідентифікації математичної моделі, можна класифікувати на три основні групи:

1. Аналітичні;
2. Чисельні;
3. Імітаційні.

***Аналітичні методи*** дозволяють отримати характеристики системи як деякі функції параметрів її функціонування. Таким чином, аналітична модель являє собою систему рівнянь, при вирішенні якої отримують параметри, необхідні для оцінки системи (час відповіді, пропускну здатність і т.д.).

Переваги аналітичних методів:

* висока точність обчислень.

Недоліки аналітичних методів:

* Розроблені тільки для вузького класу моделей, завдань;
* При виведенні формул, на яких вони ґрунтуються і які використовуються для розрахунку цікавлять параметрів, необхідно прийняти певні припущення і спростити модель.

Дані методи не підходять для вирішення даної задачі оскільки немає точної математичної постановки задачі (відсутня очікувана роздрібна ціна товару та бажаний прибуток).

Також дані методи не можуть враховувати черги під час моделювання, що є важливою частиною нашої задачі.

При цьому, для аналітичного моделювання є характерним запис процесів функціонування елементів системи у вигляді деяких функціональних залежностей або логічних умов.

Виходячи з контексту поставленої задачі, можна навести наступну формулу для обчислення загальної суми відсотків за кредитом:

,

де – сума кредиту, – відсоткова ставка, – кількість періодів (днів).

Але ми не можемо розв`язати задачу з допомогою цієї формули, оскільки нам невідомо, скільки днів необхідно для того, щоб продати весь товар в магазині.

***Чисельні методи моделювання***. Математична модель також представляє собою систему лінійних, нелінійних рівнянь, але вирішується вона методами обчислювальної математики. Всі чисельні методи передбачають ітераційне рішення задачі. На нульовій ітерації задається початкове рішення (наближення) і оцінюється його точність, на наступних ітераціях початкове наближення послідовно уточняється. Ітераційний процес продовжується до тих пір, поки не буде досягнута задана точність рішення, що не буде виконаний заданий критерій зупинки.

Клас задач, які вирішуються чисельними методами, значно ширший. Чисельні методи не дають точних рішень, але дозволяють задати будь-яку ступінь точності рішення. Використання чисельних методів виправдано, коли аналітичні методи не існують, наприклад, математична модель описується занадто складним нелінійним рівнянням або системою нелінійних рівнянь.

Даний тип методів також не підходить для вирішення поставленої задачі, оскільки ми не можемо задати початкове (наближене) вирішення задачі, а також систему лінійних рівнянь.

***Імітаційні методи моделювання***. З розвитком обчислювальної техніки широке застосування отримали імітаційні методи моделювання для аналізу систем, переважаючими в яких є стохастичні впливу.

Суть імітаційного моделювання полягає в імітації процесу функціонування системи в часі, з дотриманням таких же співвідношень тривалості операцій як в системі оригіналі. При цьому імітуються елементарні явища, що становлять процес; зберігається їх логічна структура, послідовність протікання в часі. Результатом імітаційного моделювання є отримання оцінок характеристик системи.

Імітаційне моделювання являє емпіричний підхід до дослідження складних систем, який дозволяє, при проведенні багатоваріантних розрахунків, виконувати деякі узагальнення щодо якісних закономірностей властивих системі. Імітаційне моделювання відображає тимчасовий, просторовий і логічний акцент досліджуваного об'єкта.

Методи імітаційного моделювання широко застосовуються при дослідженні систем масового обслуговування. Під системою масового обслуговування (СМО) розуміють динамічну систему, призначену для ефективного обслуговування потоку заявок (вимог на обслуговування) при обмеженнях на ресурси системи. Характерним для наведеної задачі і для інших подібних завдань з різних предметних областей є:

* Умова подвійний випадковості: випадковий момент часу надходження заявки на обслуговування та випадкова тривалість часу обслуговування;
* Наявність черг.

Застосування імітаційних методів моделювання доцільно при виконанні певних умов:

1. Завдання не піддається повній математичній формалізації. До цієї категорії відносяться багато моделей масового обслуговування, пов'язані з розглядом черг.
2. Відсутні аналогічні рішення або оцінки занадто глибокі. Тобто, математичні процедури настільки складні і трудомісткі, що імітаційне моделювання дає більш простий спосіб вирішення завдання.
3. Потрібно детально представляти динаміку функціонування системи.
4. Потрібно провести цілу низку експериментів з моделлю, тобто проведення досліджень з набором варіантів.
5. Крім оцінки певних характеристик системи, необхідно здійснити спостереження за ходом процесу протягом певного періоду на імітаційні моделі.

Використання імітаційного моделювання зручне також тому, що воно має низку переваг:

1. Легкість повтору експериментів на комп'ютері.
2. Можливість управління експериментом, включаючи його переривання та поновлення.
3. Легкість зміни умов проведення експерименту.

Отже, оптимальним для вирішення даної задачі є метод імітаційного моделювання

Імітаційну модель можливо описати мережею Петрі. Мережі Петрі є засобом формального опису процесів функціонування дискретних систем. Процес функціонування дискретної системи – це упорядкована в часі послідовність подій.

Дану предметну область можна чітко представити за допомогою мережі Перті, оскільки є чітко виділені переходи (надходження покупців, обслуговування покупця, дія процентної ставки), позиції (борг, кількість товару) та зв’язки між ними.

## Засоби автоматизації

Засобом автоматизації була обрана мова програмування Python. Python - інтерпретована об'єктно-орієнтована мова програмування високого рівня зі строгою динамічною типізацією. Python підтримує модулі та пакети модулів, що сприяє модульності та повторному використанню коду. Також, в мові програмування Python підтримується кілька парадигм програмування, зокрема: об'єктно-орієнтована, процедурна, функціональна.

В той же час, Python надає широкий спектр статистичних бібліотек, серед яких:

* Numpy
* Pandas
* Scipy
* Matplotlib

Для розробки моделі буде використано середовище JupyterLab, яке дозволяє описувати код в зручному для подальшого читання форматі.

Для тестування та аналізу результатів моделі будуть використані бібліотеки Pandas та matplotlib.

# РОЗРОБКА КОНЦЕПТУАЛЬНОЇ МОДЕЛІ

## Ціль моделювання

Приватний магазин купує партію з N одиниць товару за оптовою ціною s одиниць вартості. Гроші на придбання товару власник магазину бере у кредит. Процентна ставка за кредит становить k відсотків від суми непогашеного кредиту в день (табл. 3.23). Потік покупців в магазин - пуассонівський з параметром λ (1/хв). Імовірність того, що покупець не буде купувати товар і відразу піде з магазину залежить від довжини черги і роздрібної ціни товару: , де - імовірність того, що довжина черги "влаштовує" покупця, - імовірність того, що він купить товар за цією роздрібною ціною. Час обслуговування покупця в магазині - рівномірно розподілена в інтервалі А±В випадкова величина.

Головною ціллю моделювання є визначення найбільш вигідної роздрібної ціни продажу товару в магазині з урахуванням плати за кредит

## Вхідні та вихідні дані, параметри моделі

В табличному вигляді представлено параметри та їх значення, які необхідні для функціонування системи.

Таблиця 2.1 – Параметри моделі

|  |  |  |
| --- | --- | --- |
| **Параметри** | **Значення** | **Опис** |
|  | 2 | Процентна ставка за кредит від суми непогашеного кредиту в день |
|  | 500 | Оптова ціна одиниці товару |
|  | 1000 | Кількість закупленого товару |
|  | 0.033 | Параметр пуасонівського потоку покупців в магазин |
|  | 0.25 | Складова ймовірності відходу клієнта з магазину при черзі до 5 осіб |
|  | 0.25 | Складова ймовірності відходу клієнта з магазину при черзі 6-10 осіб |
|  | 0.2 | Складова ймовірності відходу клієнта з магазину при черзі 11-20 осіб |
|  | 0.1 | Складова ймовірності відходу клієнта з магазину при черзі 21-50 осіб |
|  | 0.3 | Складова ймовірності відходу клієнта з магазину при роздрібній ціні до 1.5s |
|  | 0.3 | Складова ймовірності відходу клієнта з магазину при роздрібній ціні 1.5s - 2s |
|  | 0.2 | Складова ймовірності відходу клієнта з магазину при роздрібній ціні 2s - 3s |
|  | 0.1 | Складова ймовірності відходу клієнта з магазину при роздрібній ціні 3s - 4s |
|  | 0.06 | Складова ймовірності відходу клієнта з магазину при роздрібній ціні 4s - 6s |
|  | 0.04 | Складова ймовірності відходу клієнта з магазину при роздрібній ціні 6s - 10s |
|  | 30 | Початковий параметр рівномірно розподіленої величини часу обслуговування покупця в магазині |
|  | 5 | Кінцевий параметр рівномірно розподіленої величини часу обслуговування покупця в магазині |
|  | 1000 | Показник точності роботи з фінансами |

В табличному вигляді представлено вхідні та вихідні дані, які необхідні для функціонування системи.

Таблиця 2.2 – Вхідні дані

|  |  |
| --- | --- |
| **Змінна** | **Опис** |
|  | Коефіцієнт зміни оптової ціни продукту ( – роздрібна ціна) |

Таблиця 2.3 – Вихідні дані

|  |  |
| --- | --- |
| **Змінна** | **Опис** |
| D | Борг магазину (dept) |
| P | Прибуток магазину (profit) |

## Обмеження

Система що моделюється не має обмежень.

## Цільова функція

Необхідно знайти таке значення , при якому чистий прибуток магазину буде найбільшим (максимізувати прибуток магазину):

## Схема моделі

Представлення концептуальної моделі функціонування магазину:

Diagram

Description automatically generated Рисунок 2.1 – Схема мережі Петрі, що відповідає заданій моделі

## Визначення структури моделі

Diagram

Description automatically generated

Рисунок 2.2 – Структура моделі «дія процентної ставки»

Diagram

Description automatically generatedDiagram

Description automatically generated

Рисунок 2.3 – Структура моделі «взаємодія покупця з магазином»

# РОЗРОБКА ІМІТАЦІЙНОЇ МОДЕЛІ ТА ОПИСУ ЇЇ ФУНКЦІОНУВАННЯ

## Опис імітаційної моделі

Дана система складається з позицій та переходів, при цьому не виходячи за рівень можливостей базової мережі Петрі з часовими затримками.

Одиницею модельного часу є 1 хвилина.

Надалі розглянемо більше детально сегменти моделі.

Маркери, що відповідають клієнтам, поступають за пуасонівським потоком. Після чого, відповідно до заданих параметрів моделі, визначається ймовірність того, що покупець залишиться в магазині та того, що покупець залишить магазин.

У випадку залишення магазину, оновлюється кількість маркерів в позиції «Кількість покупців що покинули магазин».

Якщо ж покупець залишається в магазині, збільшується кількість маркерів в позиції «Черга покупців» на один. Після чого як звільнився ресурс (в даному випадку продавець), відбувається обслуговування покупця відповідно до рівномірного закону розподілу.

Після процесу обслуговування збільшується кількість покупців що купили товар. Разом з тим оновлюється значення маркерів в позиції «Прибуток».

В той же чай, в системі конфігурує елемент нарахування процентної ставки на борг. Оновлення боргу відбувається один раз на добу відповідно до формули:

У випадку, коли присутні маркери в позиції «Борг» та «Прибуток», спрацьовує перехід «Гасіння боргу», що дозволить зменшити борг та дію процентної ставки в майбутньому.

Система припиняє моделювання, як тільки закінчуються припаси товару.

В результаті моделювання, маємо:

* Кількість клієнтів, що покинули магазин;
* Кількість клієнтів, що купили товар (має бути таким же як і початкові припаси продукту);
* Борг магазину;
* Прибуток магазину;
* Погашений борг магазину.

Даний перелік показників є достатнім для подальшого аналізу системи.

Diagram

Description automatically generated

Рисунок 3.1 – Схема мережі Петрі, що відповідає заданій моделі

Таблиця 3.1 – Основні об’єкти

|  |  |  |
| --- | --- | --- |
| **Тип об’єкту** | **Назва об’єкту** | **Опис** |
| Клас | Distribution | Описує необхідні для подальшого використання функції розподілу. |
| Клас | RandGenerator | Надає функціонал генерування випадкового числа відповідно до функції розподілу. |
| Клас | Position | Є еквівалентом позиції в мережі Петрі. Надає функціонал зміни кількості маркерів. |
| Клас | Transition | Є еквівалентом переходу в мережі Перті. Описує формат виконання переходу маркерів з одного елементу мережі в інший. При цьому надає функціонал перевірки виконання даних переходів. Зберігає інформацію про час здійснення переходу. При цьому містить інформацію про пріоритет, часові затримки. |
| Клас | TransitionWithProbabilityFunc | Є дочірнім класом від Transition. При цьому надає функціонал роботи з функцією для визначення ймовірності переходу. |
| Клас | Arc | Є еквівалентом зв’язку в мережі Петрі. Тобто, пов’язує позиції та переходи. Надає функціонал перевірки можливості переходу. Виконує переміщення маркерів з позиції в перехід та з переходу в позицію. |
| Клас | ArcAll | Є дочірнім класом від Arc. При цьому надає функціонал роботи у випадку, коли є необхідним перемістити всі маркери між елементами мережі Перті. |
| Клас | ArcAllWithMoveToFunc | Є дочірнім класом від ArcAll. При цьому надає можливість задати функцію для підрахунку кількості маркерів які необхідно перемістити. |
| Клас | Model | Відповідає моделі системи. Містить головний функціонал для запуску та імітації поведінки системи. При цьому надає можливість для підрахунку статистик та вирішення конфліктних переходів. |
| Клас | StoreSystem | Описує елементи та поведінку системи моделювання. |
| Функція | generate | Генерує випадкове число відповідно до вхідних параметрів функції розподілу. |
| Функція | add\_markers | Додавання маркерів в позицію |
| Функція | remove\_markers | Видалення маркерів з позиції |
| Функція | \_update\_markers\_release\_times | Оновлення інформацію про час надходження маркерів до переходу. |
| Функція | get\_delay | Генерує запізнення для наступного переходу. |
| Функція | check\_can\_perform\_move\_from\_transition | Перевірка на можливість здійснення виходу маркерів з елементу. |
| Функція | perform\_move\_from\_transition | Виводить маркети з елементу до переходу. |
| Функція | check\_can\_perform\_move\_to\_transition | Перевірка на можливість здійснення переміщення маркерів до елементу. |
| Функція | perform\_move\_to\_transition | Виводить маркети з переходу до елементу. |
| Функція | \_set\_arc\_ends | Оновлення даних про зв’язки між елементами. |
| Функція | check\_move\_possibility | Перевірка можливості здійснення переходу. |
| Функція | move\_from | Переводить маркерів з початку зв’язку. |
| Функція | move\_to | Переміщення маркерів в кінець зв’язку. |
| Функція | run | Запуск процесу моделювання. |
| Функція | \_get\_transitions\_with\_resolved\_conflicts | Надає список всіх переходів з вирішеними конфліктами. |
| Функція | get\_stay\_in\_store\_probability | Визначає ймовірність того, що покупець залишиться в магазині. |
| Функція | get\_leave\_probability | Визначає ймовірність того, що покупець покине магазин. |
| Функція | calculate\_debt\_rate\_effect | Визначає кількість боргу після діє процентної ставки. |
| Функція | \_stop\_condition | Перевіряє необхідність зупинку моделювання системи. |
| Функція | update\_statistics | Оновлення статистики. |
| Функція | simulate | Запускає симуляцію описаної системи. |

## Опис програмної реалізації імітаційної моделі

### Вибір засобу програмної реалізації

Засобом автоматизації була обрана мова програмування Python. Python - інтерпретована об'єктно-орієнтована мова програмування високого рівня зі строгою динамічною типізацією. Python підтримує модулі та пакети модулів, що сприяє модульності та повторному використанню коду. Також, в мові програмування Python підтримується кілька парадигм програмування, зокрема: об'єктно-орієнтована, процедурна, функціональна.

В той же час, Python надає широкий спектр статистичних бібліотек, серед яких:

* Numpy
* Pandas
* Scipy
* Matplotlib

Для розробки моделі буде використано середовище JupyterLab, яке дозволяє описувати код в зручному для подальшого читання форматі.

Для тестування та аналізу результатів моделі будуть використані бібліотеки Pandas та matplotlib.

### Опис блоків програмного коду

Diagram

Description automatically generated

Рисунок 3.2 – Структура схеми імітації моделі

# ОЦІНКА АДЕКВАТНОСТІ МОДЕЛІ

Для того, щоб переконатися що модель є адекватною, проведемо пробний експеримент. В якості вхідного параметру використаємо значення 1.7 (тобто, роздрібна ціна рівна ). Порівняємо значення результатів моделювання з очікуваними.

Отримали наступний результат:

Table

Description automatically generated

Рисунок 4.1 – Результати пробного експерименту

Як можна помітити, кожна позиція мережі була задіяна в процесі моделювання. Максимальний розмір черги при чому склав 1. Що є цілком можливим, оскільки проміжок надходження покупців в магазин в середньому складає 30 хвилин, а середній час обслуговування ті ж самі 30 хвилин. Результуюча кількість маркерів в позиції «Погашений борг» рівна 850000000 (еквівалент грошових одиниць), що є очікуваним результатом.

Оцінимо середню ймовірність покупки товару: кількість клієнтів що покинули магазин рівна 12242; кількість обслугованих покупців – 1000 (що відповідає кількості запасу товару). В сумі отримаємо . Отже, маємо:

При цьому, враховуючи початкові параметри та середню кількість покупці у черзі, маємо: , . Отримаємо:

Тому, можемо зробити висновок що теоретична та експериментальна частота покупок є майже однаковою.

Розрахуємо результуючий середній час надходження клієнтів: всього клієнтів 13242, час моделювання 411539. Маємо:

Бачимо, що теоретичний та експериментальний час надходження клієнтів є майже однаковим.

Базуючись на попередніх міркуваннях, можна зробити висновок, що отримані дані є вірними та робота моделі є адекватною.

# ОРГАНІЗАЦІЯ ЕКСПЕРИМЕНТІВ З МОДЕЛЛЮ

## План експериментів

Основною ціллю моделювання даної системи є визначення найбільш вигідної роздрібної ціни з урахуванням процентної ставки. Для цього підраховується дохід - різниця між значенням прибутком та боргу в кінці моделювання.

Значення цільової функції - максимізація доходу в кінці моделювання.

Експерименти будемо проводити послідовно, задаючи різний набір вхідних даних.

## Аналіз і оцінка результатів

Дослідження часу моделювання не є необхідним виходячи з умови завдання: система моделюється доти, доки є записи товару. Тому цю логіку було описано під час програмної реалізації.

Для визначення кількості експериментів скористаємося формулою:



де N – це кількість прогонів;

ε - точність оцінки (будемо рахувати її рівною 15% від середнього значення цільової функції);

tα=1.96;

σ – дисперсія.

Оскільки значення дисперсії невідоме, проведемо 20 дослідів з однаковим набором вхідних параметрів для її оцінки. Потім знайдемо відповідні значення та підставимо їх у формулу, щоб отримати необхідне число реалізацій.

Отримаємо наступний результат:

Table

Description automatically generated

Рисунок 5.1 20 прогорів моделі для визначення дисперсії

Отримали:

* середнє значення = -100321972.18

Тому, маємо наступну кількість експериментів:

Таким чином, достатньо провести 11 прогонів для певного параметру.

Знаючи необхідний час моделювання та кількість прогонів, можемо знайти оптимальне значення вхідний параметрів.

Для дослідження оптимальних значень параметрів скористаємось дисперсійним аналізом. В якості фактору буде виступати коефіцієнт роздрібної ціни, а в якості відгуку – значення цільовою функції (чистий прибуток).

Для обрання набору вхідних параметрів (факторів) скористаємось тим фактом, що ймовірнісна складова того, що покупець купить товар в магазині залежить від роздрібної ціни і є однаковою для різних цін з одного інтервалу. Тому є доцільним обрати наступні значення коефіцієнтів: 1.0, 1.4, 1.9, 2.9, 3.9, 5.9, 9.9.

Отримаємо результати:

Table

Description automatically generated

Рисунок 5.2 Отримані результати запуску експериментів для першої ітерації

Text

Description automatically generated

Рисунок 5.3 Результати проведення дисперсійного аналізу для першої ітерації

За результати помітно, що даний критерій (ціновий коефіцієнт) є значущим, оскільки критичне значення критерія Фішера є рівним 1.63654249, в той час є розраховане значення рівне 67.49.

Також, за рисунком 5.2 помітно, що надалі доцільно розглядати коефіцієнти в діапазоні . Тому проведемо ще одну ітерацію пошуку найкращого цінового коефіцієнту. Використаємо наступний набір параметрів: 1.4, 1.7, 1.9, 1.999.

Отримаємо наступні результати:

*Table

Description automatically generated*

Рисунок 5.4 Отримані результати запуску експериментів для другої ітерації

Text

Description automatically generated

Рисунок 5.5 Результати проведення дисперсійного аналізу для другої ітерації

Аналогічно до першої ітерації, значення фактору є значущим. При цьому найкраще значення цільова функція набуває при значенні цінового критерію рівним 1.999, що відповідає роздрібній ціні . Даний результат є цілком очікуваним, оскільки саме це значення є перехідним при розрахунку ймовірності відходу клієнтів з магазину. Значення цільової функції при даному параметру рівне -58796955.

# ВИСНОВКИ

У даній роботі було розглянуто задачу про визначення найбільш вигідної роздрібної ціни продажу товару в магазині з урахуванням плати на кредит.

Було проаналізовано можливі методи вирішення цієї задачі. Розроблена програма для вирішення даної задачі методом імітаційного моделювання. Також був проведений аналіз експериментально отриманих даних. Визначено усі поставленні завдання та найліпший коефіцієнт роздрібної ціни.

Як результат отримали значення роздрібної ціни рівним 1999 та значення цільової функції рівним -58796955.

Керуючись отриманими результатами можна дійти висновку, що починати такого роду бізнес не є доцільним, оскільки значення результуючого боргу буде значено більшим за заробіток.

При цьому слід додати, що на результат значно впливає середній час надходження покупців та розподіл ймовірнісної складовою в залежності від цінового коефіцієнту. Якщо змінити ці вхідні дані, можна значно покращити значення цільової функції та отримати чистий прибуток більший нуля.

# ПЕРЕЛІК ПОСИЛАНЬ

1. Стеценко, І.В. Моделювання систем: навч. посіб. [Електронний ресурс, текст] / І.В. Стеценко ; М-во освіти і науки України, Черкас. держ. технол. ун-т. – Черкаси : ЧДТУ, 2010. – 399 с.
2. Теорія ймовірностей, імовірнісні процеси та математична статистика. скл. Павлов О.А., Кузнецов В.М., конспект лекцій
3. Pandas – документація [Електронний ресурс] // Режим доступу: <https://pandas.pydata.org/pandas-docs/stable/reference/index.html>
4. Free Statistics Calculators – сайт [Електронний ресурс] // Режим доступу: <https://www.danielsoper.com/statcalc/calculator.aspx?id=4>

# ДОДАТКИ

Репозиторій з кодом: <https://github.com/dtsylyuryk/system-modeling-course-work>

Лістинг програми:

*#%% md****## Курсова робота на тему "Визначення найбільш вигідної роздрібної ціни продажу товару в магазині з урахуванням плати на кредит"  
## Виконав студент гр. ІС-72 Цилюрик Дмитро****#%% md****### Програмний опис імітації мережі Петрі використовуючи ООП підхід****#%% md****##### Імпорт необхідних бібліотек****#%%***import** json  
**import** math  
**import** numpy **as** np  
**import** pandas **as** pd  
**import** sys  
**from** enum **import** Enum  
**from** IPython.display **import** display  
  
*#%% md****##### Встановлення неохідних опцій для подальшого використання****#%%*pd.set\_option(**'display.max\_rows'**, 5000)  
pd.set\_option(**'display.max\_columns'**, 5000)  
pd.set\_option(**'display.width'**, 1000)  
pd.set\_option(**'mode.chained\_assignment'**, **None**)  
pd.set\_option(**'display.float\_format'**, **lambda** x: **'%.5f'** % x)  
  
*#%% md****##### Клас збереження типів розподілу для ренерації затримок****#%%***class** Distribution(Enum):  
 EXPONENTIAL = np.random.exponential  
 UNIFORM = np.random.uniform  
 NORMAL = np.random.normal  
 POISSON = np.random.poisson  
  
*#%% md****##### Клас для генерації затримок****#%%***class** RandGenerator:  
 @staticmethod  
 **def** generate(distribution\_function, \*\*args):  
 **return** distribution\_function(\*\*args)  
  
*#%% md****##### Клас для представлення Позиції в мережі Петрі****#%%***class** Position:  
 **def** \_\_init\_\_(self, num\_of\_markers=0, description=**None**, \*\*kwargs):  
 self.num\_of\_markers = num\_of\_markers  
 self.description = description  
  
 self.in\_arcs = []  
 self.out\_arcs = []  
  
 **def** add\_markers(self, num\_of\_markers):  
 self.num\_of\_markers += num\_of\_markers  
  
 **def** remove\_markers(self, num\_of\_markers):  
 self.num\_of\_markers -= num\_of\_markers  
  
 **def** \_\_repr\_\_(self):  
 **return f"Position<{**self.description**}> = {**self.num\_of\_markers**}"***#%% md****##### Клас для представлення Переходу в мережі Перті****#%%***class** Transition:  
 **def** \_\_init\_\_(  
 self,  
 delay=0,  
 delay\_distribution=**None**,  
 delay\_distribution\_params=**None**,  
 priority=1,  
 probability=**None**,  
 description=**None**,  
 \*\*kwargs,  
 ):  
 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.in\_arcs = []  
 self.out\_arcs = []  
  
 self.markers\_release\_times = {}  
  
 @property  
 **def** probability(self):  
 **return** self.\_probability  
   
 **def** \_update\_markers\_release\_times(self, release\_timestamp):  
 **if** release\_timestamp **in** self.markers\_release\_times:  
 self.markers\_release\_times[release\_timestamp] += 1  
 **else**:  
 self.markers\_release\_times[release\_timestamp] = 1  
   
 **def** get\_delay(self):  
 **return** round(  
 self.delay   
 **if** self.delay\_distribution **is None   
 else** RandGenerator.generate(  
 self.delay\_distribution.value, \*\*self.delay\_distribution\_params  
 )  
 )  
  
 **def** check\_can\_perform\_move\_from\_transition(self):  
 **return** all(arc.check\_move\_possibility() **for** arc **in** self.in\_arcs)  
  
 **def** perform\_move\_from\_transition(self, current\_time):  
 delay = self.get\_delay()  
 release\_timestamp = current\_time + delay  
 self.\_update\_markers\_release\_times(release\_timestamp)  
 **for** arc **in** self.in\_arcs:  
 arc.move\_from()  
  
 **def** check\_can\_perform\_move\_to\_transition(self, current\_time):  
 **return** current\_time **in** self.markers\_release\_times.keys()  
   
 **def** perform\_move\_to\_transition(self, current\_time):  
 **for** \_ **in** range(self.markers\_release\_times[current\_time]):  
 **for** arc **in** self.out\_arcs:  
 arc.move\_to()  
 **else**:  
 **del** self.markers\_release\_times[current\_time]  
  
 **def** \_\_repr\_\_(self):  
 **return f"Transition<{**self.description**}> = {**self.markers\_release\_times**}"***#%% md****##### Клас для представлення Переходу в мережі Перті із заданою функцієї умовірностей переходу****#%%***class** TransitionWithProbabilityFunc(Transition):  
 **def** \_\_init\_\_(self, probability\_func, \*\*kwargs):  
 super().\_\_init\_\_(\*\*kwargs)  
 self.\_queue = kwargs[**"queue"**]  
 self.\_m = kwargs[**"m"**]  
 self.\_probability\_func = probability\_func  
  
 @property  
 **def** probability(self):  
 **return** self.\_probability\_func(self.\_queue.num\_of\_markers, self.\_m)  
  
*#%% md****##### Клас для представлення Дуги в мережі Перті****#%%***class** Arc:  
 **def** \_\_init\_\_(self, start, end, multiplier=1, informational=**False**):  
 self.start = start  
 self.end = end  
 self.multiplier = multiplier  
 self.informational = informational  
  
 self.\_set\_arc\_ends()  
  
 **def** \_set\_arc\_ends(self):  
 self.start.out\_arcs.append(self)  
 self.end.in\_arcs.append(self)  
  
 **def** check\_move\_possibility(self):  
 **return** self.start.num\_of\_markers >= self.multiplier  
  
 **def** move\_from(self):  
 **if not** self.informational:  
 self.start.remove\_markers(self.multiplier)  
  
 **def** move\_to(self):  
 self.end.add\_markers(self.multiplier)  
  
 **def** \_\_repr\_\_(self):  
 **return f"{**self.start**} --> {**self.end**}"***#%% md****##### Клас для представлення Дуги в мережі Перті за умови переходу всіх мартерів із Позиції****#%%***class** ArcAll(Arc):  
 **def** \_\_init\_\_(self, move\_to\_all=**True**, \*\*kwargs):  
 super().\_\_init\_\_(\*\*kwargs)  
 self.\_move\_to\_all = move\_to\_all  
  
 **def** check\_move\_possibility(self):  
 **return** self.start.num\_of\_markers >= 0  
  
 **def** move\_from(self):  
 self.end.multiplier\_all = self.start.num\_of\_markers  
 **if not** self.informational:  
 self.start.remove\_markers(self.start.num\_of\_markers)  
  
 **def** move\_to(self):  
 **if** self.\_move\_to\_all:  
 self.end.add\_markers(self.start.multiplier\_all)  
 **else**:  
 super().move\_to()  
  
*#%% md****##### Клас для представлення Дуги в мережі Перті за умови переходу всіх мартерів із Позиції за заданої функцію визначення кількості маркерів для переходу****#%%***class** ArcAllWithMoveToFunc(ArcAll):  
 **def** \_\_init\_\_(self, move\_to\_func, \*\*kwargs):  
 super().\_\_init\_\_(\*\*kwargs)  
 self.\_move\_to\_func = move\_to\_func  
   
 **def** move\_to(self):  
 **if** self.\_move\_to\_all:  
 self.end.add\_markers(self.\_move\_to\_func(self.start.multiplier\_all))  
 **else**:  
 super().move\_to()  
  
*#%% md****##### Клас моделювання мережі Петрі****#%%***class** Model:  
 **def** \_\_init\_\_(  
 self,   
 transitions,   
 positions,   
 modeling\_time,  
 stop\_condition=**None**,  
 verbose=**False**,  
 ):  
 self.transitions = transitions  
 self.positions = positions  
 self.\_modeling\_time = modeling\_time  
 self.\_verbose = verbose  
   
 **if** stop\_condition **is None**:  
 self.\_stop\_condition = **lambda** : **True  
 else**:  
 self.\_stop\_condition = stop\_condition  
   
 self.current\_time = 0  
 self.position\_markers\_stats = {position: [] **for** position **in** positions}  
  
 **def** run(self):  
 **if** self.\_verbose:  
 self.\_display\_model\_status()  
 **while** self.current\_time < self.\_modeling\_time:  
 **while True**:   
 **if not** any(  
 transition.check\_can\_perform\_move\_from\_transition()  
 **for** transition **in** self.transitions  
 ):  
 **break** self.\_move()  
 self.update\_statistics()  
   
 **if not** any(  
 len(transition.markers\_release\_times) > 0  
 **for** transition **in** self.transitions  
 ) **or** self.\_stop\_condition():  
 **break** self.\_move()  
 self.update\_statistics()  
   
 self.current\_time += 1  
 **if** self.\_verbose:  
 self.\_display\_model\_status()  
  
 **return** self.get\_result\_statistics()  
  
 **def** \_move(self):  
 transitions\_without\_conflicts = self.\_get\_transitions\_with\_resolved\_conflicts()  
 **for** transition **in** transitions\_without\_conflicts:  
 **if** transition.check\_can\_perform\_move\_from\_transition():  
 transition.perform\_move\_from\_transition(self.current\_time)  
 **for** transition **in** self.transitions:  
 **if** transition.check\_can\_perform\_move\_to\_transition(self.current\_time):  
 transition.perform\_move\_to\_transition(self.current\_time)  
   
 **def** \_get\_transitions\_with\_resolved\_conflicts(self):  
 conflicting\_transitions = []  
 resulting\_transitions = []  
  
 **for** position **in** self.positions:  
 non\_informational\_arcs = filter(**lambda** arc: **not** arc.informational, position.out\_arcs)  
  
 valid\_transitions = [\*filter(  
 **lambda** transition: transition.check\_can\_perform\_move\_from\_transition(),  
 map(**lambda** arc: arc.end, non\_informational\_arcs)  
 )]  
  
 **if** len(valid\_transitions) > 1:  
 conflicting\_transitions.append(valid\_transitions)  
  
 **for** conflict **in** conflicting\_transitions:  
 is\_probabilistic\_conflict = conflict[0].probability **is not None  
  
 if** is\_probabilistic\_conflict:  
 probabilities = [\*map(**lambda** transition: transition.probability, conflict)]  
  
 gap = 1 - sum(probabilities)  
 **if** gap != 0:  
  
 probabilities = [p + gap / len(probabilities) **for** p **in** probabilities]  
  
 resulting\_transitions.append(np.random.choice(conflict, p=probabilities))  
 **else**:  
 conflict.sort(key=**lambda** transition: transition.priority, reverse=**True**)  
 resulting\_transitions.append(conflict[0])  
  
  
 **for** transition **in** self.transitions:  
 was\_transition\_in\_conflict = any([transition **in** conflict **for** conflict **in** conflicting\_transitions])  
  
 **if not** was\_transition\_in\_conflict:  
 resulting\_transitions.append(transition)  
  
 **return** resulting\_transitions  
  
 **def** update\_statistics(self):  
 **for** position **in** self.positions:  
 self.position\_markers\_stats[position].append(position.num\_of\_markers)  
  
 **def** get\_result\_statistics(self):  
 result\_stats = []  
  
 **for** position, stats **in** self.position\_markers\_stats.items():  
 result\_stats.append(  
 {  
 **"description"**: position.description,  
 **"avg\_markers"**: sum(stats) / len(stats),  
 **"max\_markers"**: max(stats),  
 **"min\_markers"**: min(stats),  
 **"result\_markers"**: stats[-1],  
 **"time\_modeling"**: self.current\_time,  
 }   
 )  
  
 **return** result\_stats  
  
 **def** \_display\_model\_status(self):  
 print(**f"Current time: {**self.current\_time**}"**)  
 **for** position **in** self.positions:  
 print(position)  
 **for** transition **in** self.transitions:  
 print(transition)  
 print()  
  
*#%% md*![Model](diagrams/petri\_model.png "Мережа Петрі для задачі")  
  
*#%% md****### Моделювання задачі****#%%***class** StoreSystem:  
 LAMBDA = 30  
 k = 0.02  
 s = 500  
 N = 1000  
  
 p1 = {  
 (0, 5): 0.25,  
 (6, 10): 0.25,  
 (11, 20): 0.2,  
 (21, 50): 0.2,  
 (51, sys.maxsize): 0.1,  
 }  
 p2 = {  
 (0, 1.4999): 0.3,  
 (1.5, 1.9999): 0.3,  
 (2.0, 2.9999): 0.2,  
 (3.0, 3.9999): 0.1,  
 (4.0, 5.9999): 0.06,  
 (6.0, 9.9999): 0.04,  
 (10.0, sys.maxsize): 0,  
 }  
  
 A = 30  
 B = 5  
  
 M = 1000  
 minutes\_in\_one\_day = 24 \* 60  
 modeling\_time = math.inf \* minutes\_in\_one\_day  
   
 **def** \_\_init\_\_(self, m):  
 self.m = m  
   
 *# Positions* self.customer\_generator = Position(num\_of\_markers=1, description=**"Потік покупців"**)  
 self.customer\_came = Position(num\_of\_markers=0, description=**"Покупець надійшов"**)  
 self.queue = Position(num\_of\_markers=0, description=**"Черга покупців"**)  
 self.customer\_left\_store = Position(num\_of\_markers=0, description=**"Кількість покупців, що покинули мазагин"**)  
 self.stock = Position(num\_of\_markers=self.N, description=**"Запас товару"**)  
 self.seller = Position(num\_of\_markers=1, description=**"Продавець"**)  
 self.customer\_success = Position(num\_of\_markers=0, description=**"Кількість обслугованих покупців"**)  
 self.profit = Position(num\_of\_markers=0, description=**"Прибуток"**)  
 self.debt = Position(num\_of\_markers=self.N \* self.s \* self.M, description=**"Борг"**)  
 self.repaid\_debt = Position(num\_of\_markers=0, description=**"Погашений борг"**)  
 self.delay\_generator = Position(num\_of\_markers=1, description=**"Затримка процентної ставки"**)  
 self.interest\_rate\_ready = Position(num\_of\_markers=0, description=**"Процентна ставка готова до дії"**)  
   
 *# Transitions* self.customer\_arrive = Transition(  
 delay\_distribution=Distribution.EXPONENTIAL,  
 delay\_distribution\_params={**"scale"**: self.LAMBDA},  
 description=**"Надходження покупця"** )  
 self.customer\_leave\_store = TransitionWithProbabilityFunc(  
 probability\_func=self.get\_leave\_probability,   
 queue=self.queue,  
 m=self.m,  
 description=**"Покупець покипає магазин"** )  
 self.customer\_stay\_in\_store = TransitionWithProbabilityFunc(  
 probability\_func=self.get\_stay\_in\_store\_probability,   
 queue=self.queue,  
 m=self.m,  
 description=**"Покупець стає в чергу"** )  
 self.service = Transition(  
 delay\_distribution=Distribution.UNIFORM,  
 delay\_distribution\_params={**"low"**: self.A - self.B, **"high"**: self.A + self.B},  
 description=**"Обслуговування покупця"** )  
 self.debt\_repayment = Transition(description=**"Гасіння боргу"**)  
 self.interest\_rate\_delay = Transition(delay=self.minutes\_in\_one\_day, description=**"Затримка для процентнох ставки"**)  
 self.interest\_rate\_effect = Transition(description=**"Дія процентної ставки"**, priority=2)  
   
 *# Arcs* Arc(start=self.customer\_generator, end=self.customer\_arrive)  
 Arc(start=self.customer\_arrive, end=self.customer\_generator)  
 Arc(start=self.customer\_arrive, end=self.customer\_came)  
 Arc(start=self.customer\_came, end=self.customer\_leave\_store)  
 Arc(start=self.customer\_leave\_store, end=self.customer\_left\_store)  
 Arc(start=self.customer\_came, end=self.customer\_stay\_in\_store)  
 Arc(start=self.customer\_stay\_in\_store, end=self.queue)  
 Arc(start=self.queue, end=self.service)  
 Arc(start=self.stock, end=self.service)  
 Arc(start=self.seller, end=self.service)  
 Arc(start=self.service, end=self.seller)  
 Arc(start=self.service, end=self.customer\_success)  
 Arc(start=self.service, end=self.profit, multiplier=int(self.m \* self.s \* self.M))  
 Arc(start=self.profit, end=self.debt\_repayment, multiplier=self.M)  
 Arc(start=self.debt, end=self.debt\_repayment, multiplier=self.M)  
 Arc(start=self.debt\_repayment, end=self.repaid\_debt, multiplier=self.M)  
 Arc(start=self.delay\_generator, end=self.interest\_rate\_delay)  
 Arc(start=self.interest\_rate\_delay, end=self.delay\_generator)  
 Arc(start=self.interest\_rate\_delay, end=self.interest\_rate\_ready)  
 Arc(start=self.interest\_rate\_ready, end=self.interest\_rate\_effect)  
 ArcAll(start=self.debt, end=self.interest\_rate\_effect)  
 ArcAllWithMoveToFunc(start=self.interest\_rate\_effect, end=self.debt, move\_to\_func=self.calculate\_debt\_rate\_effect)  
  
  
 @staticmethod  
 **def** get\_stay\_in\_store\_probability(num\_in\_queue, m):  
 p1\_current = next(filter(**lambda** item: item[0][0] <= num\_in\_queue <= item[0][1], StoreSystem.p1.items()))[1]  
 p2\_current = next(filter(**lambda** item: item[0][0] <= m <= item[0][1], StoreSystem.p2.items()))[1]  
 **return** p1\_current \* p2\_current  
  
 @staticmethod  
 **def** get\_leave\_probability(num\_in\_queue, m):  
 **return** 1 - StoreSystem.get\_stay\_in\_store\_probability(num\_in\_queue, m)  
  
 @staticmethod  
 **def** calculate\_debt\_rate\_effect(current\_debt):  
 **return** round(current\_debt \* (1 + StoreSystem.k))  
   
 **def** \_stop\_condition(self):  
 **return** (  
 **True   
 if** self.stock.num\_of\_markers == 0 **and** len(self.service.markers\_release\_times) == 0  
 **else False** )  
   
 **def** \_append\_target\_function\_result(self, modeling\_results):  
 modeling\_results.append(  
 {  
 **"description"**: **"Чистий прибуток"**,  
 **"avg\_markers"**: **None**,  
 **"max\_markers"**: **None**,  
 **"min\_markers"**: **None**,  
 **"result\_markers"**: (self.profit.num\_of\_markers - self.debt.num\_of\_markers) / StoreSystem.M,  
 **"time\_modeling"**: modeling\_results[0][**"time\_modeling"**],  
 }  
 )  
 **return** modeling\_results  
  
 **def** simulate(self, verbose=**False**, flag\_display\_results=**True**):  
 net = Model(  
 positions=[  
 self.customer\_generator,  
 self.customer\_came,  
 self.queue,  
 self.customer\_left\_store,  
 self.stock,  
 self.seller,  
 self.customer\_success,  
 self.profit,  
 self.debt,  
 self.repaid\_debt,  
 self.delay\_generator,  
 self.interest\_rate\_ready,  
 ],  
 transitions=[  
 self.customer\_arrive,  
 self.customer\_leave\_store,  
 self.customer\_stay\_in\_store,  
 self.service,  
 self.debt\_repayment,  
 self.interest\_rate\_delay,  
 self.interest\_rate\_effect,  
 ],  
 modeling\_time=self.modeling\_time,  
 stop\_condition=self.\_stop\_condition,  
 verbose=verbose,  
 )  
 modeling\_results = self.\_append\_target\_function\_result(net.run())  
   
 **if** flag\_display\_results:  
 display(pd.DataFrame(modeling\_results))  
   
 **return** modeling\_results  
  
  
*#%% md****### Оцінка адекватності моделі****#%%*\_system = StoreSystem(1.7)  
test\_simulation = \_system.simulate()  
  
*#%% md****#### Середній час надходження покупців****#%%*num\_of\_clients = 0  
**for** item **in** test\_simulation:  
 **if** item[**"description"**] **in** [**"Кількість покупців, що покинули мазагин"**, **"Кількість обслугованих покупців"**]:  
 num\_of\_clients += item[**"result\_markers"**]  
  
print(**f"Average arrival time: {**test\_simulation[0][**'time\_modeling'**]/num\_of\_clients**}"**)  
  
*#%% md*  
Як можна помітити, середній час надходження покупців є близьким до дійсності. За умовою клієнти надходитя в чередньому кожні 30 хлилин (пуасонівський потік із параметром λ = 1/30 ($\frac{1}{хв}$)  
  
*#%% md****#### Ймовірність купівлі товару****#%%*failed\_num\_of\_clients, succeeded\_num\_of\_clients = 0, 0  
**for** item **in** test\_simulation:  
 **if** item[**"description"**] == **"Кількість покупців, що покинули мазагин"**:  
 failed\_num\_of\_clients += item[**"result\_markers"**]  
 **elif** item[**"description"**] == **"Кількість обслугованих покупців"**:  
 succeeded\_num\_of\_clients += item[**"result\_markers"**]  
  
print(**f"Average probability of purchase: {**succeeded\_num\_of\_clients/(succeeded\_num\_of\_clients+failed\_num\_of\_clients)**}"**)  
  
*#%% md*  
При запуску тестового прогону був використаний коефіцієнт ціни товару: $m = 1.7$. Це відповідає значенню $p\_2=0.3$. Середнє значення покупців в черзі рівне 0 (див. середнє значення в позиції "Черга покупців"), тому $p\_1=0.25$. Отримаємо наступну ймовірність купівлі товару покупцем:   
\begin{equation\*}  
P = p\_1**\***p\_2 = 0.3**\***0.25 = 0.075  
\end{equation\*}  
Теоретична та експериментальна частота покупок є майже однаковими. Отже, модель є адекватною.  
  
*#%% md****### Визначення кількості прогонів***  
Надалі будемо використовувати дану формулу:  
\begin{equation\*}  
N = \frac{t\_α^2σ^2}{ε^2}+1  
\end{equation\*}  
де $σ^2$ - дисперсія відгуку моделі, $ε$ - точність вимірювання, $t\_α$ - агрумент функції Лаплача  
  
*#%%*experiments = []  
**for** i **in** np.arange(20):  
 print(**f"Experiment {**i**}"**)  
 m = 1.7  
 store\_system = StoreSystem(m)  
 **for** result\_item **in** store\_system.simulate(flag\_display\_results=**False**):   
 experiments.append(  
 {  
 **"price\_coefficient"**:m,  
 \*\*result\_item  
 }  
 )  
  
*#%%*experiments\_df = pd.DataFrame(experiments)  
profit\_df = experiments\_df.loc[experiments\_df[**"description"**] == **"Чистий прибуток"**]  
  
*#%%*display(profit\_df)  
  
*#%%*needed\_number\_of\_experiments = round(  
 (1.96\*\*2\*profit\_df.std()[**"result\_markers"**]\*\*2) /   
 (profit\_df.mean()[**"result\_markers"**]\*0.15)\*\*2   
 + 1  
)  
  
*#%%*print(**f"Number of needed experiments: {**needed\_number\_of\_experiments**}"**)  
  
*#%% md****### Аналіз та оцінка результатів****#%% md*  
Для визначення впливу факторів на значення цільової функції було використано дисперсійний аналіз. Основне питання, на яке дає відповідь дисперсійний аналіз впливу фактора, формулюється так: різниця у значеннях відгуку моделі, отриманих при різних значеннях фактора обумовлена випадковістю, чи пояснюється виключно дією фактора?  
Надалі будуть використні наступні формули:  
- для розрахунку середнього значення цільової функції заданого коефіцієнта $j$  
\begin{equation\*}  
\overline{y\_j} = \frac{1}{n}\sum **\_**{i=1}^{n}y**\_**{ij},   
\end{equation\*}  
де $n$ - кількість прогонів  
- для розрахунку середнього значення цільової функції  
\begin{equation\*}  
\overline{y} = \frac{1}{m}\sum **\_**{j=1}^{m}y**\_**{j},   
\end{equation\*}  
де $m$ - кількість коефіцієнтів  
- для розрахунку значення різниць між групами коефіцієнтів  
\begin{equation\*}  
d\_{факт} = n \sum **\_**{j=1}^{m}(\overline{y**\_**{j}}-\overline{y})^2,   
\end{equation\*}  
де $n$ - кількість прогонів, $m$ - кількість коефіцієнтів  
- для розрахунку значення через групами коефіцієнтів  
\begin{equation\*}  
d\_{залиш} = \frac{\sum **\_**{i=1}^{K}\sum **\_**{j=1}^{n\_i}(y**\_**{ij}-\overline{y**\_**{i}})^2}{N-K},   
\end{equation\*}  
\begin{equation\*}  
N = m**\***n,  
\end{equation**\***}  
\begin{equation\*}  
K = m  
\end{equation\*}  
де $n$ - кількість прогонів, $m$ - кількість коефіцієнтів  
- для розрахунку значення критерію Фішера  
\begin{equation\*}  
F = \frac{d\_{факт}}{d\_{залиш}}  
\end{equation\*}  
  
*#%% md*  
Виходячи з контексту даної задачі є доцільним обрати **\*\***коефіцієнт ціни товару**\*\*** (**\*\*\_**price\_coefficient**\_\*\***) фактором, при цьому відгуком моделі буде значення цільової функції, тобто "**\*\***Чистий прибуток**\*\***" ("**\*\*\_**Прибуток**\_\*\***" - "**\*\*\_**Борг**\_\*\***")  
  
*#%% md*  
Для визначення набору параметрів звернемось до умови задачі. Маємо настуний розподіл ймовірностей в залежності від цінового коефіцієнту:  
```python  
p2 = {  
 (0, 1.4999): 0.3,  
 (1.5, 1.9999): 0.3,  
 (2.0, 2.9999): 0.2,  
 (3.0, 3.9999): 0.1,  
 (4.0, 5.9999): 0.06,  
 (6.0, 9.9999): 0.04,  
 (10.0, sys.maxsize): 0,  
}  
```  
Доцільним є вибір коефіцієнтів які знаходяться на граничному переході, тобто для першої ітерації будуть використані наступні значення коефіцієнтів:  
```python  
m in [1.0, 1.4, 1.9, 2.9, 3.9, 5.9, 9.9]  
```  
  
*#%%*results\_first\_iteration = []  
**for** m **in** [1.0, 1.4, 1.9, 2.9, 3.9, 5.9, 9.9]:  
 **for** i **in** np.arange(needed\_number\_of\_experiments):  
 print(**f"price\_coefficient: {**m**}; experiment {**i**}"**)  
 store\_system = StoreSystem(m)  
 **for** result\_item **in** store\_system.simulate(flag\_display\_results=**False**):   
 results\_first\_iteration.append(  
 {  
 **"experiment"**: i,  
 **"price\_coefficient"**:m,  
 \*\*result\_item  
 }  
 )  
  
*#%% md***\_**Для уникнення очікування виконання експериментів, результати були збережені. Для відновлення необхідно запустити наступну ділянку коду**\_***#%%*results\_first\_iteration\_path = (  
 **"experiments/price\_coefficient\_change\_11\_runs\_1\_iteration.json"**)  
results\_first\_iteration = []  
**with** open(results\_first\_iteration\_path, **"r"**) **as** f:  
 **for** line **in** f.readlines():  
 results\_first\_iteration.append(json.loads(line))  
  
*#%% md*  
Аналіз результатів та перевірка значимості фактора за критерієм Фішера.  
**\*\*\_**Помітка**\_\*\***: для визначння *```F\_critical```* був використаний сервіс [![Free Statistics Calculators](references/free\_statistics\_calculators.png "Free Statistics Calculators")](*https://www.danielsoper.com/statcalc/calculator.aspx?id=4*)  
  
*#%%*results\_first\_iteration\_df = pd.DataFrame(results\_first\_iteration)  
first\_iteration\_profit\_df = results\_first\_iteration\_df.loc[results\_first\_iteration\_df[**"description"**] == **"Чистий прибуток"**]  
first\_iteration\_profit\_df = first\_iteration\_profit\_df[  
 [  
 **"experiment"**,   
 **"price\_coefficient"**,   
 **"description"**,   
 **"result\_markers"**,   
 **"time\_modeling"**,  
 ]  
]  
first\_iteration\_profit\_df[**"result\_markers"**] = first\_iteration\_profit\_df[**"result\_markers"**].astype(**"float64"**)  
first\_iteration\_profit\_df = first\_iteration\_profit\_df.reset\_index(drop=**True**)  
first\_iteration\_profit\_df[**"global\_average"**] = first\_iteration\_profit\_df[**"result\_markers"**].mean()  
first\_iteration\_profit\_df[**"price\_coefficient\_avg\_result"**] = (  
 first\_iteration\_profit\_df  
 .groupby([**"price\_coefficient"**])[**"result\_markers"**]  
 .transform(**lambda** x: x.mean())  
)  
first\_iteration\_profit\_df[**"group\_deviation"**] = ((  
 first\_iteration\_profit\_df[**"result\_markers"**] - first\_iteration\_profit\_df[**"price\_coefficient\_avg\_result"**])\*\*2  
)  
first\_iteration\_profit\_df[**"s\_factual"**] = (  
 needed\_number\_of\_experiments \*   
 sum(  
 (  
 first\_iteration\_profit\_df.groupby([**"price\_coefficient"**])[**"result\_markers"**].mean() -   
 first\_iteration\_profit\_df[**"global\_average"**].iloc[0]  
 )\*\*2  
 )  
)  
first\_iteration\_profit\_df[**"s\_residual"**] = first\_iteration\_profit\_df[**"group\_deviation"**].sum()  
  
num\_of\_groups = len(first\_iteration\_profit\_df[**"price\_coefficient"**].unique())  
  
d\_factual = first\_iteration\_profit\_df[**"s\_factual"**].iloc[0]  
d\_residual = (  
 first\_iteration\_profit\_df[**"s\_residual"**].iloc[0] /   
 ((num\_of\_groups \* needed\_number\_of\_experiments) - num\_of\_groups)  
)  
  
F = d\_factual / d\_residual  
F\_critical = 1.63654249  
  
print(**f"Sum of Squares: {**d\_factual**}"**)  
print(**f"Degrees of Freedom: {**((num\_of\_groups \* needed\_number\_of\_experiments) - num\_of\_groups)**}"**)  
print(**f"Mean Square: {**d\_residual**}"**)  
print(**f"F-actual: {**F**}; F-critical(alpha=0.15): {**F\_critical**}"**)  
print(**f"Factor is {'not ' if** F < F\_critical **else ''}significant!"**)  
  
display(first\_iteration\_profit\_df)  
  
*#%%*results\_first\_iteration\_df = pd.DataFrame(results\_first\_iteration)  
first\_iteration\_profit\_df = results\_first\_iteration\_df.loc[results\_first\_iteration\_df[**"description"**] == **"Чистий прибуток"**]  
first\_iteration\_profit\_df = first\_iteration\_profit\_df[  
 [  
 **"experiment"**,   
 **"price\_coefficient"**,   
 **"description"**,   
 **"result\_markers"**,   
 **"time\_modeling"**,  
 ]  
]  
first\_iteration\_profit\_df[**"result\_markers"**] = first\_iteration\_profit\_df[**"result\_markers"**].astype(**"float64"**)  
first\_iteration\_profit\_df[**"rank"**] = first\_iteration\_profit\_df[**"result\_markers"**].rank(method=**"max"**, ascending=**False**)  
first\_iteration\_profit\_df.sort\_values(by=[**'rank'**], inplace=**True**)  
first\_iteration\_profit\_df = first\_iteration\_profit\_df[first\_iteration\_profit\_df[**"rank"**] <= 10]  
first\_iteration\_profit\_df = first\_iteration\_profit\_df.reset\_index(drop=**True**)  
  
print(**"Top 10 results:"**)  
display(first\_iteration\_profit\_df)  
print(**"Grouped results based on price\_coefficient:"**)  
display(first\_iteration\_profit\_df.groupby([**"price\_coefficient"**]).size())  
  
*#%% md***\*\***Аналіз результатів**\*\***: отримали два вагомі значення коефіцієнтів, а сама *`[1.4, 1.9]`*. Тому надалі є доцільним обирати значення з цього діапазону.  
Проведемо другу ітерацію пошуку найкращого параметру на наступному наборі значень:   
```python   
m in [1.4, 1.7, 1.9, 1.999]  
```  
  
*#%%*results\_second\_iteration = []  
**for** m **in** [1.4, 1.7, 1.9, 1.999]:  
 **for** i **in** np.arange(needed\_number\_of\_experiments):  
 print(**f"price\_coefficient: {**m**}; experiment {**i**}"**)  
 store\_system = StoreSystem(m)  
 **for** result\_item **in** store\_system.simulate(flag\_display\_results=**False**):   
 results\_second\_iteration.append(  
 {  
 **"experiment"**: i,  
 **"price\_coefficient"**:m,  
 \*\*result\_item  
 }  
 )  
  
*#%% md***\_**Для уникнення очікування виконання експериментів, результати були збережені. Для відновлення необхідно запустити наступну ділянку коду**\_***#%%*results\_second\_iteration\_path = (  
 **"experiments/price\_coefficient\_change\_11\_runs\_2\_iteration.json"**)  
results\_second\_iteration = []  
**with** open(results\_second\_iteration\_path, **"r"**) **as** f:  
 **for** line **in** f.readlines():  
 results\_second\_iteration.append(json.loads(line))  
  
*#%% md*  
Аналіз результатів та перевірка значимості фактора за критерієм Фішера.  
**\*\*\_**Помітка**\_\*\***: для визначння *```F\_critical```* був використаний сервіс [![Free Statistics Calculators](references/free\_statistics\_calculators.png "Free Statistics Calculators")](*https://www.danielsoper.com/statcalc/calculator.aspx?id=4*)  
  
*#%%*results\_second\_iteration\_df = pd.DataFrame(results\_second\_iteration)  
second\_iteration\_profit\_df = results\_second\_iteration\_df.loc[results\_second\_iteration\_df[**"description"**] == **"Чистий прибуток"**]  
second\_iteration\_profit\_df = second\_iteration\_profit\_df[  
 [  
 **"experiment"**,   
 **"price\_coefficient"**,   
 **"description"**,   
 **"result\_markers"**,   
 **"time\_modeling"**,  
 ]  
]  
second\_iteration\_profit\_df[**"result\_markers"**] = second\_iteration\_profit\_df[**"result\_markers"**].astype(**"float64"**)  
second\_iteration\_profit\_df = second\_iteration\_profit\_df.reset\_index(drop=**True**)  
second\_iteration\_profit\_df[**"global\_average"**] = second\_iteration\_profit\_df[**"result\_markers"**].mean()  
second\_iteration\_profit\_df[**"price\_coefficient\_avg\_result"**] = (  
 second\_iteration\_profit\_df  
 .groupby([**"price\_coefficient"**])[**"result\_markers"**]  
 .transform(**lambda** x: x.mean())  
)  
second\_iteration\_profit\_df[**"group\_deviation"**] = ((  
 second\_iteration\_profit\_df[**"result\_markers"**] - second\_iteration\_profit\_df[**"price\_coefficient\_avg\_result"**])\*\*2  
)  
second\_iteration\_profit\_df[**"s\_factual"**] = (  
 needed\_number\_of\_experiments \*   
 sum(  
 (  
 second\_iteration\_profit\_df.groupby([**"price\_coefficient"**])[**"result\_markers"**].mean() -   
 second\_iteration\_profit\_df[**"global\_average"**].iloc[0]  
 )\*\*2  
 )  
)  
second\_iteration\_profit\_df[**"s\_residual"**] = second\_iteration\_profit\_df[**"group\_deviation"**].sum()  
  
num\_of\_groups = len(second\_iteration\_profit\_df[**"price\_coefficient"**].unique())  
  
d\_factual = second\_iteration\_profit\_df[**"s\_factual"**].iloc[0]  
d\_residual = (  
 second\_iteration\_profit\_df[**"s\_residual"**].iloc[0] /   
 ((num\_of\_groups \* needed\_number\_of\_experiments) - num\_of\_groups)  
)  
  
F = d\_factual / d\_residual  
F\_critical = 1.87109477  
  
print(**f"Sum of Squares: {**d\_factual**}"**)  
print(**f"Degrees of Freedom: {**((num\_of\_groups \* needed\_number\_of\_experiments) - num\_of\_groups)**}"**)  
print(**f"Mean Square: {**d\_residual**}"**)  
print(**f"F-actual: {**F**}; F-critical(alpha=0.15): {**F\_critical**}"**)  
print(**f"Factor is {'not ' if** F < F\_critical **else ''}significant!"**)  
  
display(second\_iteration\_profit\_df)  
  
*#%%*results\_second\_iteration\_df = pd.DataFrame(results\_second\_iteration)  
second\_iteration\_profit\_df = results\_second\_iteration\_df.loc[results\_second\_iteration\_df[**"description"**] == **"Чистий прибуток"**]  
second\_iteration\_profit\_df = second\_iteration\_profit\_df[  
 [  
 **"experiment"**,   
 **"price\_coefficient"**,   
 **"description"**,   
 **"result\_markers"**,   
 **"time\_modeling"**,  
 ]  
]  
second\_iteration\_profit\_df[**"result\_markers"**] = second\_iteration\_profit\_df[**"result\_markers"**].astype(**"float64"**)  
second\_iteration\_profit\_df[**"rank"**] = second\_iteration\_profit\_df[**"result\_markers"**].rank(method=**"max"**, ascending=**False**)  
second\_iteration\_profit\_df.sort\_values(by=[**'rank'**], inplace=**True**)  
second\_iteration\_profit\_df = second\_iteration\_profit\_df[second\_iteration\_profit\_df[**"rank"**] <= 10]  
second\_iteration\_profit\_df = second\_iteration\_profit\_df.reset\_index(drop=**True**)  
  
print(**"Top 10 results:"**)  
display(second\_iteration\_profit\_df)  
print(**"Grouped results based on price\_coefficient:"**)  
display(second\_iteration\_profit\_df.groupby([**"price\_coefficient"**]).size())  
  
*#%% md***\*\***Аналіз результатів**\*\***: значення коефіцієнта *`1.999`* займає першу позацію та зустрічається в топ-10 чотири рази, тому можна дійти до висновку, що це і є оптимальним значенням фактора, що є цілком логічним, оскільки саме це значення є граничним при переході різниць ймовірностей. Значення цільової функції в найкращому випадку: *`-58796955.06200`*. Тобто, робимо висковки, що підприємство **\*\*\_**не є прибутковим**\_\*\***.  
  
*#%% md****#### Додаткова перевірка***  
Для того, щоб переконатися, що система моделюється правильно, був проведений запуск відповідно за умов варіанту 1.   
За результати помітно, що прибуток є більшим за нуля. Це зумовлене тим, що час надходження покупців та розподіл ймовірностей відповідно до ціни товару є кращим ніж у варіанті 2.  
  
*#%%*\_system = StoreSystem(1.999)  
\_system.simulate()  
  
  
*# LAMBDA = 20  
# k = 0.01  
# s = 1000  
# N = 600  
  
# p1 = {  
# (0, 5): 0.3,  
# (6, 10): 0.3,  
# (11, 20): 0.2,  
# (21, 50): 0.1,  
# (51, sys.maxsize): 0.1,  
# }  
# p2 = {  
# (0, 1.4999): 0.45,  
# (1.5, 1.9999): 0.4,  
# (2.0, 2.9999): 0.06,  
# (3.0, 3.9999): 0.04,  
# (4.0, 5.9999): 0.03,  
# (6.0, 9.9999): 0.02,  
# (10.0, sys.maxsize): 0,  
# }  
  
# A = 30  
# B = 5  
  
# m = 1.999  
# M = 1000  
  
  
#%%*