Міністерство освіти і науки України
Криворізький національний університет
Факультет інформаційних технологій
Кафедра автоматизації, комп'ютерних наук і технологій

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти – бакалавр
за освітньо-професійною програмою
«Комп'ютерні науки»
зі спеціальності
122 – Комп'ютерні науки

тема роботи:

« Розробка алгоритму та програмна реалізація розв'язання стохастичної задачі управління запасами »

Виконав студент гр. КН-22-1 _____ Агаєв І. І.

Керівник _____ Маринич І. А.

Нормоконтроль _____ Маринич І. А.

Завідувач кафедри _____ Рубан С. А.

Кривий Ріг – 2026

КРИВОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет: *інформаційних технологій*

Кафедра: *автоматизації, комп'ютерних наук і технологій*

Ступінь вищої освіти: *Бакалавр*

Спеціальність: *122 – Комп'ютерні науки*

ЗАТВЕРДЖУЮ

Зав. кафедрою: *к.т.н. Рубан С.А.*

_____

« *26* » *січня* 2026 р.

ЗАВДАННЯ

на кваліфікаційну роботу бакалавра

студентові групи КН-22-1 *Агаєву Ілгару Ілгаровичу*

1. Тема кваліфікаційної роботи: *«Розробка алгоритму та програмна реалізація розв'язання стохастичної задачі управління запасами»*

затверджено наказом по університету № *173с* від *30.03.2026 р.*

2. Термін здачі кваліфікаційної роботи: *10.06.2026 р.*

3. Склад кваліфікаційної роботи: *Пояснювальна записка, додатки, презентація у Microsoft PowerPoint в електронному та друкованому вигляді*

4. Консультанти кваліфікаційної роботи:

Розділ 1-2 доц. Маринич І. А.

Нормоконтроль доц. Маринич І. А.

5. Календарний план:

№	Етапи роботи	Термін виконання
1	*Вступ*	*01.03.26*
2	*Розділ 1*	*05.04.26*
3	*Розділ 2*	*01.05.26*
4	*Висновки*	*25.05.26*
5	*Оформлення кваліфікаційної роботи*	*28.05.26*
6	*Підготовка презентації та графічного матеріалу*	*20.05.26*
7	*Підготовка доповіді до захисту*	*05.06.26*

6. Дата видачі завдання: *28.01.2026р.*

Керівник _____ / Маринич І. А./

7. Запевнення: *Я, Агаєв Ілгар Ілгарович, запевняю, що ця кваліфікаційна робота виконана самостійно, не містить академічного плагіату, фабрикації, фальсифікації. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.*

Із чинним Положенням про академічну доброчесність Криворізького національного університету ознайомлений.

Чітко усвідомлюю, що в разі виявлення у кваліфікаційній роботі умисних порушень робота не допускається до захисту або оцінюється незадовільно.

Студент _____ /Агаєв І. І./

АНОТАЦІЯ

Агаєв І. М. Розробка алгоритму та програмна реалізація розв’язання стохастичної задачі управління запасами : кваліфікаційна робота бакалавра : 122 – Комп’ютерні науки. Кривий Ріг. Криворізький національний університет, 2026. 68 с.

Актуальність теми зумовлена необхідністю розроблення алгоритмічних і програмних засобів для розв’язання стохастичних задач управління запасами, що враховують випадковий характер попиту та дозволяють підвищити ефективність діяльності підприємств.

Метою роботи є розроблення алгоритму та його програмної реалізації для розв’язання стохастичних задач управління запасами з урахуванням випадкового характеру попиту та різних сценаріїв дефіциту.

У першому розділі було розглянуто теоретичні основи управління запасами як важливої складової логістичної діяльності підприємства. Наведено класифікацію запасів, охарактеризовано основні задачі управління запасами, а також проаналізовано детерміновані та стохастичні моделі. Окрему увагу приділено стохастичним моделям з відкладеним і втраченим попитом, їх особливостям, параметрам та практичному застосуванню.

У другому розділі було розроблено алгоритми розв’язання стохастичних задач управління запасами з урахуванням відкладеного та втраченого попиту. Представлено програмну реалізацію розроблених алгоритмів мовою програмування Python із використанням бібліотек NumPy та Matplotlib. Також виконано моделювання роботи системи управління запасами, побудовано графіки зміни рівня запасів у часі та проведено аналіз отриманих результатів.

Ключові слова:

АЛГОРИТМ, ВІДКЛАДЕНИЙ ПОПИТ, ВТРАЧЕНИЙ ПОПИТ, ДЕФІЦИТ, СТОХАСТИЧНА МОДЕЛЬ, УПРАВЛІННЯ ЗАПАСАМИ, PYTHON, MATPLOTLIB, NUMPY

ЗМІСТ

ВСТУП

У сучасних умовах цифровізації економіки та зростання конкуренції ефективне управління ресурсами підприємства набуває особливого значення. Однією з ключових складових ресурсного забезпечення є матеріальні запаси, які відіграють важливу роль у підтриманні безперервності виробничих процесів, забезпеченні стабільності постачання та задоволенні попиту споживачів. Невідповідність обсягів запасів реальним потребам підприємства може призводити як до надлишкових витрат на їх утримання, так і до втрат через дефіцит продукції.

У практиці управління запасами підприємства стикаються з невизначеністю, яка проявляється у вигляді випадкових коливань попиту, змін тривалості постачання, нестабільності ринкового середовища та інших факторів. У таких умовах використання класичних детермінованих моделей є обмеженим, оскільки вони не враховують стохастичний характер реальних процесів. Це зумовлює необхідність застосування стохастичних моделей управління запасами, які дозволяють більш адекватно описувати поведінку системи та забезпечувати прийняття обґрунтованих управлінських рішень.

Особливої актуальності набуває врахування різних сценаріїв дефіциту, зокрема ситуацій із відкладеним попитом (backorders) та втраченим попитом (lost sales). Вибір відповідної моделі безпосередньо впливає на фінансові результати діяльності підприємства, рівень обслуговування клієнтів та ефективність логістичних процесів.

З розвитком інформаційних технологій з’являються нові можливості для автоматизації процесів управління запасами. Сучасні методи математичного моделювання, імітаційного аналізу та оптимізації, реалізовані за допомогою мов програмування, зокрема Python, дозволяють створювати ефективні інструменти підтримки прийняття рішень. Це дає змогу не лише аналізувати існуючі процеси, але й прогнозувати їх поведінку в умовах невизначеності.

Таким чином, *актуальність даної роботи* зумовлена необхідністю

розроблення алгоритмічних і програмних засобів для розв’язання стохастичних задач управління запасами, що враховують випадковий характер попиту та дозволяють підвищити ефективність діяльності підприємств.

Об’єктом дослідження є процеси управління запасами підприємства в умовах невизначеності та випадкових впливів.

Предметом дослідження є математичні моделі, алгоритми та програмні засоби розв’язання стохастичних задач управління запасами.

Метою роботи є розроблення алгоритму та його програмної реалізації для розв’язання стохастичних задач управління запасами з урахуванням випадкового характеру попиту та різних сценаріїв дефіциту.

Для досягнення поставленої мети необхідно вирішити такі задачі:

- проаналізувати сучасні підходи до управління запасами та їх математичне забезпечення;
- дослідити існуючі детерміновані та стохастичні моделі управління запасами;
- виконати класифікацію моделей та визначити особливості їх застосування;
- побудувати стохастичні моделі управління запасами з урахуванням відкладеного та втраченого попиту;
- обґрунтувати вибір методу розв’язання задачі (імітаційне моделювання, оптимізаційні підходи);
- розробити алгоритм розв’язання стохастичної задачі управління запасами;
- виконати програмну реалізацію алгоритму з використанням мови Python;
- провести експериментальні дослідження та оцінити ефективність запропонованого підходу.

Методи дослідження, використані у роботі, включають:

- методи математичного моделювання;
- методи теорії ймовірностей та математичної статистики;

- імітаційне моделювання (метод Монте-Карло);
- методи оптимізації;
- методи алгоритмізації та програмування.

Розроблений алгоритм розв’язання стохастичних задач управління запасами з урахуванням різних типів дефіциту, дозволяє більш точно оцінювати витрати та підвищувати ефективність прийняття рішень.

Практичне значення роботи полягає у створенні програмного засобу, який може бути використаний для підтримки процесу управління запасами на підприємствах різних галузей. Запропонований підхід дозволяє зменшити витрати, пов’язані із зберіганням запасів, та мінімізувати втрати від дефіциту продукції.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ОСНОВИ СТОХАСТИЧНИХ МОДЕЛЕЙ УПРАВЛІННЯ ЗАПАСАМИ

1.1 Опис і класифікація управління запасами

Управління запасами є однією з ключових функцій логістики та відіграє важливу роль у забезпеченні ефективного взаємозв'язку між виробництвом, постачанням та споживанням продукції. Основною метою управління запасами є підтримання оптимального рівня запасів, що дозволяє задовольнити потреби клієнтів при мінімізації загальних витрат підприємства. В умовах сучасного конкурентного ринку, де швидкість реагування на зміни попиту та пропозиції має критичне значення, управління запасами стає ще більш актуальним, оскільки ефективне планування та контроль запасів дозволяють підприємствам зменшувати витрати, покращувати якість обслуговування та підвищувати конкурентні переваги[1].

Управління запасами забезпечує безперервність виробничих та торговельних процесів. Невідповідність обсягів запасів реальним потребам підприємства може призводити як до надлишкових витрат на зберігання та страхування запасів, так і до втрат прибутку через дефіцит продукції. Наприклад, надлишкові запаси на складі збільшують витрати на оренду приміщень, енергоресурси та обслуговування, тоді як дефіцит продукції може призвести до втрати клієнтів та негативного впливу на репутацію компанії. У зв'язку з цим сучасні підприємства прагнуть поєднувати економічну ефективність і високий рівень обслуговування споживачів, що передбачає комплексний підхід до планування запасів.

В управлінні запасами виділяють кілька основних типів запасів, кожен з яких виконує специфічні функції в логістичній ланцюговій системі[4]:

- Поточні (регулярні) запаси забезпечують безперервність виробничого процесу та задоволення поточного попиту. Вони формуються

через невідповідність термінів і обсягів поставок та споживання матеріальних ресурсів. Наприклад, на підприємствах харчової промисловості поточні запаси сировини забезпечують стабільну роботу ліній з переробки продуктів щодня.

– Страхові (буферні) запаси використовуються для компенсації випадкових коливань попиту та пропозиції. Вони дозволяють підтримувати безперервність виробництва або торговельної діяльності у разі неточності прогнозів або порушень поставок. Наприклад, промислове підприємство може створювати страхові запаси електронних компонентів, щоб уникнути простоїв у випадку затримки постачання від постачальника.

– Сезонні запаси формуються для забезпечення нормального функціонування підприємства у періоди сезонних коливань виробництва або попиту. Такі запаси часто зустрічаються у сферах сільського господарства, роздрібної торгівлі та енергетики, де попит на продукцію різко зростає або падає у певні сезони.

– Спекулятивні запаси створюються у очікуванні зміни ринкових умов, наприклад, підвищення цін на сировину або можливого дефіциту продукції. Вони дозволяють підприємству захиститися від негативних ринкових коливань та отримати додатковий прибуток від правильного прогнозування тенденцій.

До основних завдань управління запасами належать[4]:

– Визначення оптимального рівня запасів – розрахунок необхідних обсягів для кожного виду запасів з урахуванням специфіки діяльності підприємства та характеристик ринку. Оптимальний рівень дозволяє уникнути як надлишку, так і дефіциту продукції, забезпечуючи баланс між витратами та обсягом обслуговування клієнтів.

– Розробка та впровадження систем контролю – створення ефективних систем моніторингу та управління запасами для своєчасного поповнення та мінімізації витрат. Сучасні автоматизовані системи дозволяють відстежувати рух товарів у режимі реального часу, прогнозувати потреби та швидко реагувати на зміни.

– Аналіз та оптимізація процесів управління запасами – постійний аналіз даних щодо запасів з метою виявлення можливостей їх оптимізації та зниження витрат. Наприклад, застосування методів математичного моделювання та імітаційного аналізу дозволяє оцінити вплив різних стратегій управління запасами на фінансові результати підприємства.

1.2 Класифікація моделей управління запасами

Моделі управління запасами поділяють на різні типи залежно від характеру параметрів, які вони враховують, та способу застосування[3]:

– Детерміновані моделі – моделі, у яких усі параметри системи (попит, час поставки, витрати) відомі та залишаються незмінними. Вони використовуються на стабільних ринках, де коливання попиту та пропозиції мінімальні.

– Стохастичні моделі – моделі, що враховують випадкові коливання параметрів системи, наприклад, змінність попиту та часу поставки. Вони застосовуються в умовах невизначеності та дозволяють більш точно прогнозувати поведінку системи та приймати обґрунтовані управлінські рішення.

– Статичні моделі – передбачають незмінність параметрів системи з часом, прості в реалізації та використовуються для аналізу короткострокових рішень.

– Динамічні моделі – відображають зміни параметрів системи з плином часу та застосовуються для довгострокового планування, що особливо важливо для великих підприємств з комплексними ланцюгами постачання.

– Моделі з урахуванням дефіциту та без дефіциту. Моделі з урахуванням дефіциту дозволяють тимчасово допускати відсутність товару на складі та при цьому враховуються пов’язані з цим витрати. Моделі без дефіциту – це такі моделі, у яких прагнуть повністю уникнути ситуацій дефіциту товару. Перелічимо деякі з методів класифікації запасів.

Серед методів класифікації запасів широко застосовуються ABC- та

XYZ-аналізи.

ABC-аналіз базується на принципі Парето [7]: 20% об’єктів забезпечують 80% результату. Він дозволяє розподілити запаси на три групи:

– Група A – включає товари з найбільшою вартістю та значущістю для підприємства, хоча вони можуть складати невелику частину від загальної кількості товарів. Такі товари потребують ретельного контролю та частого моніторингу;

– Група B містить товари середньої значущості та вартості, які потребують помірного рівня контролю.

– Група C включає товари з найменшою вартістю, що становлять більшу частину загальної кількості і потребують мінімального контролю.

Метод ABC-аналізу [6]допомагає підприємствам концентрувати зусилля на найбільш значущих позиціях запасів, покращуючи управління та знижуючи витрати.

XYZ-аналіз доповнює ABC-аналіз, класифікуючи товари за ступенем передбачуваності попиту:

– Група X – товари з рівномірним, стабільним попитом, які легко прогнозуються. Зазвичай це матеріали або продукти, необхідні для безперервного виробництва.

– Група Y – товари з коливним попитом, що часто мають сезонні або трендові зміни. Прогнозування попиту для таких товарів потребує додаткових зусиль.

– Група Z – товари з епізодичним і непередбачуваним попитом, прогнозування якого утруднене. Зазвичай це товари з нерегулярним використанням.

XYZ-аналіз допомагає підприємствам краще розуміти структуру попиту на свої товари та розробляти більш ефективні стратегії управління запасами.

Застосування ABC- та XYZ-аналізів дозволяє підприємствам ефективніше концентрувати ресурси на критично важливих позиціях та

розробляти стратегії управління запасами, що відповідають реальним потребам ринку.

1.3 Оцінка та застосування стохастичних моделей управління запасами

Стохастичні моделі управління запасами дозволяють підприємствам враховувати випадковість попиту, часу постачання та інших факторів, що роблять класичні детерміновані моделі недостатньо точними. Для того, щоб ці моделі приносили практичну користь, необхідно оцінювати їх ефективність, аналізувати вплив основних параметрів на результати та вивчати конкретні приклади застосування у різних сферах бізнесу.

Оцінка стохастичних моделей включає визначення ключових показників ефективності, таких як рівень обслуговування клієнтів, тривалість періодів дефіциту та середні витрати на зберігання і дефіцит[2]. Аналіз параметрів моделі дозволяє зрозуміти, як зміни попиту, часу поставки та витрат впливають на оптимальний обсяг замовлення та частоту дефіциту. Практичні приклади застосування демонструють реальні вигоди від використання стохастичних моделей у рітейлі, виробництві, логістиці та маркетингу, допомагаючи підприємствам зменшувати витрати, підвищувати рівень обслуговування та мінімізувати ризики.

У наступних підпунктах детально розглядаються:

- методи оцінки ефективності стохастичних моделей,
- вплив ключових параметрів на управління запасами,
- практичні приклади застосування у різних сферах діяльності.

1.3.1 Методи оцінки ефективності стохастичних моделей

Для того, щоб стохастичні моделі управління запасами були корисними на практиці, необхідно оцінювати їх ефективність за певними критеріями.

Оцінка дозволяє порівняти різні моделі, вибрати оптимальні стратегії управління запасами та зменшити ризики дефіциту або надлишку.

Основні критерії оцінки[7]:

– Рівень обслуговування клієнтів (Service Level) – ймовірність того, що попит буде задоволений у повному обсязі протягом певного періоду часу. Чим вищий рівень обслуговування, тим менше втрачених продажів, але, як правило, зростають витрати на зберігання.

– Середній час дефіциту – середня тривалість періодів, коли товар відсутній на складі, але попит залишається. Цей показник важливий для оцінки можливих втрат доходів і впливу на лояльність клієнтів.

– Середні витрати на зберігання та дефіцит – включають витрати на утримання запасів (оренда, страхування, амортизація) і витрати від дефіциту (втрата прибутку, додаткові логістичні витрати, штрафи за несвоєчасну доставку).

Симуляційне моделювання

Одним із ефективних методів оцінки є імітаційне моделювання, зокрема метод Монте-Карло. Він дозволяє змоделювати тисячі сценаріїв змін попиту та часу постачання, оцінюючи, як змінюються рівень запасу, витрати та періоди дефіциту. Симуляції дають змогу:

- визначити ймовірність виникнення дефіциту;
- оцінити вплив зміни параметрів моделі (λ, τ, h, p або b);
- обрати найбільш надійну стратегію замовлення та поповнення запасів.

Порівняння моделей

Для прийняття управлінських рішень корисно порівнювати декілька стохастичних моделей за вищезгаданими критеріями. Наприклад, можна оцінити:

- модель з відкладеним попитом проти моделі з втраченим попитом;
- вплив різних рівнів страхових запасів на ймовірність дефіциту;

– чутливість оптимального замовлення до зміни інтенсивності попиту або витрат на зберігання.

Такі оцінки дозволяють підприємствам вибирати моделі, які забезпечують оптимальний баланс між витратами та якістю обслуговування, та підвищують точність управління запасами в умовах невизначеності.

1.3.2 Вплив параметрів моделі на управління запасами

Реальна ефективність стохастичних моделей управління запасами значною мірою залежить від ключових параметрів моделі. Зміни цих параметрів впливають на обсяг замовлень, частоту дефіциту, витрати на зберігання та рівень обслуговування клієнтів. Аналіз чутливості параметрів дозволяє підприємствам адаптувати моделі до реальних умов ринку та приймати обґрунтовані управлінські рішення.

Основні параметри та їх вплив[7]:

– Інтенсивність попиту (d або λ). Інтенсивність попиту визначає середній обсяг товару, що необхідно продати або поставити за одиницю часу. Зростання попиту збільшує ризик дефіциту, якщо обсяг замовлення не змінюється, і водночас підвищує витрати на поповнення запасів. Зменшення попиту навпаки може призвести до надлишку запасів та зайвих витрат на зберігання.

– Час затримки поставки (τ). Чим довший час доставки товару від постачальника, тим більший необхідний страховий запас для запобігання дефіциту. Збільшення τ підвищує загальні витрати на зберігання, тоді як скорочення часу доставки дозволяє зменшити страхові запаси та оптимізувати обсяг замовлень.

– Витрати на зберігання (h). Питомі витрати на утримання запасів включають оренду складу, страхування, витрати на персонал і втрати через псування чи застарівання товарів. Високі витрати на зберігання змушують оптимізувати розміри замовлення і зменшувати обсяги запасів, що може підвищити ризик дефіциту.

– Витрати від дефіциту (p або b). Витрати, пов'язані з невиконанням попиту, включають упущений прибуток, додаткові логістичні витрати та потенційну втрату клієнтів. Зростання цих витрат стимулює збільшення страхових запасів і частоту замовлень для мінімізації ризику дефіциту.

Аналіз чутливості

Аналіз чутливості дозволяє дослідити, як зміни параметрів впливають на:

– оптимальний розмір замовлення;
– тривалість періодів дефіциту;
– середні витрати на зберігання та дефіцит;
– ймовірність обслуговування попиту.

Наприклад, у моделі з відкладеним попитом збільшення інтенсивності попиту (d) без коригування обсягу замовлення може призвести до тривалого дефіциту, тоді як збільшення витрат на дефіцит (p) стимулює підприємство збільшити страхові запаси для збереження рівня обслуговування клієнтів.

Практичне значення

Розуміння впливу параметрів дозволяє підприємствам:

– підбирати найбільш підходящі моделі для конкретних умов;
– оцінювати ризики дефіциту та втрат;
– розробляти гнучкі стратегії управління запасами, які реагують на зміни попиту, часу доставки та витрат;
– підвищувати ефективність логістики та мінімізувати загальні витрати.

Таким чином, аналіз впливу параметрів моделі є невід'ємною частиною процесу оптимізації запасів та дозволяє підприємству досягти оптимального балансу між витратами та рівнем обслуговування клієнтів.

1.3.3 Практичні приклади застосування стохастичних моделей

Стохастичні моделі управління запасами мають широке практичне застосування у різних сферах бізнесу та виробництва. Їх використання

дозволяє підприємствам враховувати невизначеність попиту, час доставки та інші випадкові фактори, що безпосередньо впливають на фінансові результати та рівень обслуговування клієнтів.

Приклади застосування[9]:

– Рітейл. У роздрібній торгівлі моделі з відкладеним або втраченим попитом допомагають прогнозувати попит на швидкозростаючі або сезонні товари, наприклад, одяг у святковий сезон чи популярні гаджети. Завдяки моделюванню компанії можуть визначити оптимальні обсяги замовлення та страховий запас, щоб зменшити дефіцит і втрати від незадоволеного попиту.

– Виробництво. На виробничих підприємствах стохастичні моделі використовуються для управління запасами сировини та компонентів. Наприклад, у виробництві електронної техніки запаси мікросхем і деталей можуть бути обмежені, а терміни поставок від постачальників варіюються. Моделі з урахуванням випадкового попиту дозволяють забезпечити безперервність виробничого процесу і уникнути простоїв через дефіцит ключових компонентів.

– Логістика та дистрибуція. У логістиці стохастичні моделі застосовуються для планування роботи складів та маршрутизування поставок. Вони дозволяють оптимізувати обсяги запасів на різних точках зберігання, враховуючи змінний попит та затримки доставки. Це зменшує витрати на транспортування і зберігання, а також підвищує точність виконання замовлень.

– Маркетинг та управління запасами для нових продуктів. При введенні нових продуктів на ринок попит важко передбачити. Стохастичні моделі дозволяють оцінити можливий розкид попиту і визначити оптимальні обсяги первинного замовлення та страхові запаси, щоб зменшити ризик втрати продажів і одночасно уникнути надлишку продукції.

Використання стохастичних моделей дозволяє підприємствам:

- адаптувати управління запасами до реальних умов ринку;
- зменшити витрати на зберігання та дефіцит;
- підвищити точність прогнозування попиту;

- забезпечити високий рівень обслуговування клієнтів.

Таким чином, стохастичні моделі стають незамінним інструментом для будь-якого підприємства, що прагне оптимізувати логістичні процеси та забезпечити ефективне управління запасами в умовах невизначеності.

1.4 Стохастичні моделі управління запасами

У роботі детально розглядаються стохастичні моделі управління запасами. Такі моделі базуються на тому, що ключові параметри систем управління запасами є випадковими величинами. Розподіл цих параметрів, як правило, підпорядковується нормальному закону Гаусса або експоненційному закону.

До основних видів стохастичних моделей управління запасами відносять:

- моделі з урахуванням втрат через дефіцит;
- моделі з втраченим попитом.

1.4.1 Моделі з урахуванням втрат через дефіцит

Моделі економічного розміру замовлення з урахуванням дефіциту (EOQ with backorders) застосовуються для підприємств, які можуть тимчасово допустити дефіцит товару, відкладаючи незадоволений попит на майбутнє[10]. Такі моделі допомагають мінімізувати сумарні витрати, включаючи витрати на розміщення замовлення, зберігання та дефіцит.

Основні припущення моделей включають:

- фіксована затримка поповнення – доставка товару здійснюється із заздалегідь визначеною затримкою;
- витрати на упущений попит – включають втрату прибутку та додаткові витрати на залучення нових клієнтів.

У моделях з відкладеним попитом враховується дефіцит, який виникає під час поставки, при визначенні розміру партії. Це дозволяє задовольнити як поточний попит, так і попит, що виник раніше, але залишився незадоволеним.

Основні аспекти моделей з відкладеним попитом[11]:

– Попит – інтерес до продукту може виникнути пізніше, коли споживачі приймуть рішення про покупку у майбутньому. Запит залежить від часу, ціни товару, сезонності та інших факторів.

– Час затримки – період, який потрібен для перетворення відкладеного запиту у реальний попит на товар. Він може залежати від різних факторів, таких як час доставки, виробничі цикли та інші обмеження.

– Витрати на зберігання запасів – витрати, пов'язані з утриманням товарів у очікуванні майбутнього попиту, що включають витрати на зберігання, страхування, втрати від псування або застарівання продукції.

– Управління запасами – визначення оптимального рівня запасів та стратегій їх контролю в умовах відкладеного попиту. Це включає прийняття рішень про те, скільки товарів закупити зараз і скільки залишити на майбутнє, щоб одночасно знизити витрати на зберігання та задовольнити попит.

Основний підхід до вирішення проблем з накопиченим попитом полягає у створенні математичних моделей та алгоритмів оптимізації, які дозволяють визначити найефективніші стратегії управління запасами в умовах відкладеного попиту та інших обмежень. Такі моделі стають незамінним інструментом у багатьох сферах, включаючи рітейл, виробничі процеси, логістику та інші галузі.

Математична формулювання моделі з урахуванням втрат через дефіцит[10]:

$$C(Q) = \frac{K \cdot d}{Q} + \frac{h \cdot Q}{2} \cdot \left(1 - \frac{d}{Q}\right) + \frac{p \cdot d}{Q} \cdot \left(1 - \frac{Q}{d}\right) \quad (1.1)$$

де K – фіксовані витрати на замовлення;

d – інтенсивність попиту;

Q – розмір замовлення;

h – питомі витрати на зберігання; p – витрати від дефіциту.

З формули (1.1) отримуємо оптимальний розмір замовлення:

$$Q^* = \sqrt{\frac{2Kd}{h \cdot \left(1 - \frac{d}{Q}\right) + p}} \tag{1.2}$$

де Q^* – розмір виробничого замовлення, який мінімізує загальні витрати за одиницю часу;

K – фіксовані витрати на замовлення;

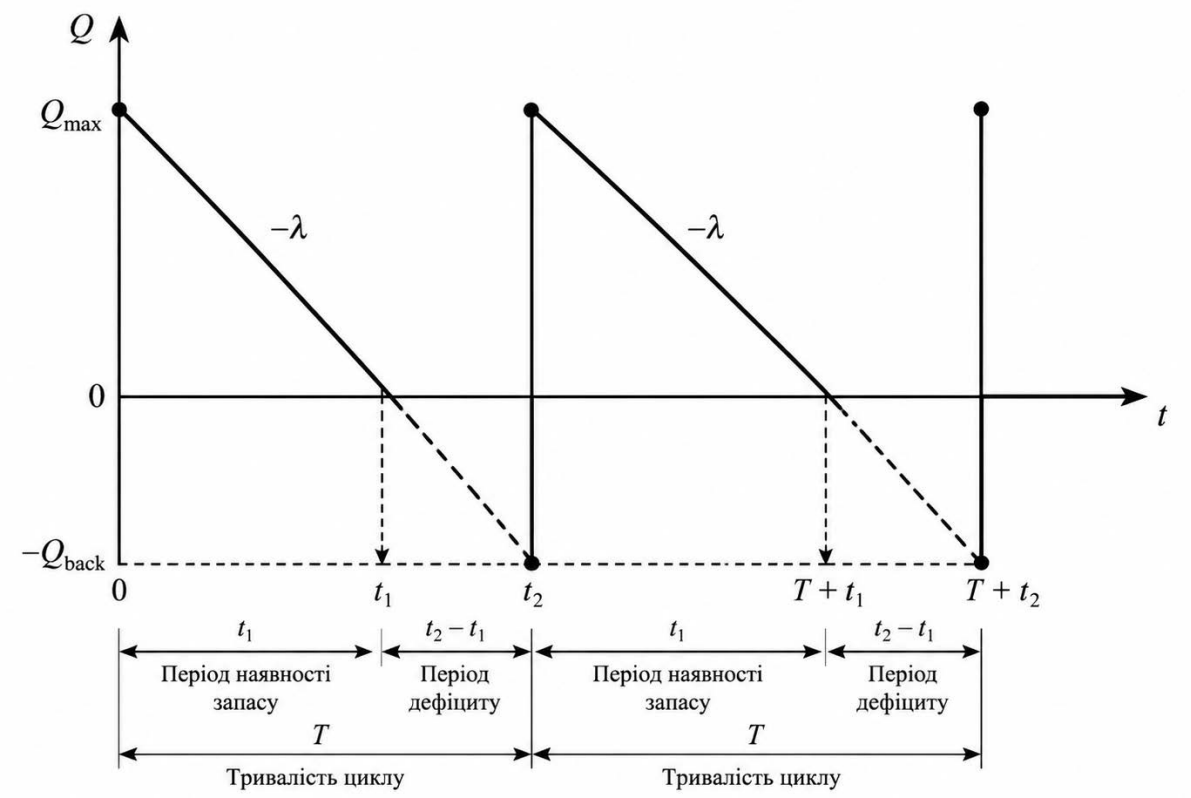
d – інтенсивність попиту;

Q – розмір замовлення;

h – питомі витрати на зберігання;

p – витрати від дефіциту.

Модель з відкладеним попитом представлено на рисунку 1.1



Q - рівень запасу (кількість товару на складі); Q_{max} - максимальний рівень запасу після поповнення; -Q_{back} - максимальний рівень дефіциту (від'ємний запас); λ- інтенсивність попиту (швидкість витрачання запасу); t_1 час, протягом якого запас зменшується до нуля; t_2 - час, протягом якого виникає дефіцит; T - тривалість одного циклу управління запасами

Рисунок 1.1 – Модель з відкладеним попитом

На рисунку (1.1) показано зміну кількості продукту на складі протягом сумарного часу t_1 та t_2.

Кожен цикл складається з двох етапів – t_1 та t_2. У момент часу t_1 відбувається витрачання поточного замовлення зі швидкістю λ, а t_2 – це період дефіциту, коли товар відсутній на складі, але потік заявок не зменшується та продовжує надходити з тією ж інтенсивністю λ. При отриманні нового замовлення для поповнення запасів частина його миттєво задовольняє заявки з дефіцитного періоду, а залишок додається до поточного запасу, також миттєво надходячи на склад.

У моделях з дефіцитом замовлення слід розміщувати на складі у момент, коли рівень дефіциту (від'ємний запас) досягає максимального значення[12]. Точка замовлення може бути задана у від'ємних значеннях.

1.4.2 Моделі з втраченим попитом

Моделі з втраченим попитом (Lost Sales) [13] актуальні для ситуацій, коли незадоволений попит не переноситься на майбутнє, а втрачається назавжди. Це критично важливо для ринків з високою конкуренцією, де втрата продажів може суттєво вплинути на прибутковість.

Основні припущення моделей:

- Фіксована затримка поповнення – постачання товару відбувається з визначеною затримкою;
- Витрати на втрачений попит – включають втрачений прибуток та витрати на залучення нових клієнтів.

У моделях оптимального розміру замовлення з втраченим попитом дефіцит розглядається як неможливість задовольнити заявки на відвантаження товарів. Клієнтам відмовляють, а відновлення запасу відбувається у попередніх обсягах.

Основні аспекти моделей з втраченим попитом:

– Попит – кількість товару або послуги, яку споживачі готові купити або замовити за певний період. Попит може залежати від ціни, реклами, сезонності та інших факторів.

– Втрачені продажі – кількість попиту, яку неможливо задовольнити через недостатнє постачання або неможливість виконати замовлення вчасно, що може призвести до упущених можливостей і втрати доходу.

– Наслідки втраченого попиту – втрата клієнтів, зниження доходу, шкода репутації бренду. Це підкреслює важливість розробки стратегій управління запасами та виробничими потужностями для мінімізації ризику втрати попиту.

– Управління ризиком втраченого попиту – розробка стратегій і тактик, спрямованих на зниження ризику втрати попиту, таких як збільшення запасів, оптимізація виробничих і логістичних процесів, залучення альтернативних постачальників або використання нових технологій.

Моделі з втраченим попитом широко застосовуються в управлінні запасами, логістиці, маркетингу та інших сферах бізнесу. Ефективне управління ризиком втраченого попиту допомагає підприємствам мінімізувати втрати та підвищувати рівень обслуговування клієнтів.

Математична формулювання моделі з втраченим попитом[12]:

$$C(Q) = \frac{K \cdot d}{Q} + \frac{h \cdot Q}{2} + \frac{b \cdot d}{Q} \cdot \left(1 - \frac{Q}{d}\right) \tag{1.3}$$

де K – фіксовані витрати на замовлення;

d – інтенсивність попиту;

Q – розмір замовлення;

b – витрати на втрачений попит.

З формули (1.3) отримуємо оптимальний розмір замовлення:

$$Q^* = \sqrt{\frac{2Kd}{h + b}} \tag{1.4}$$

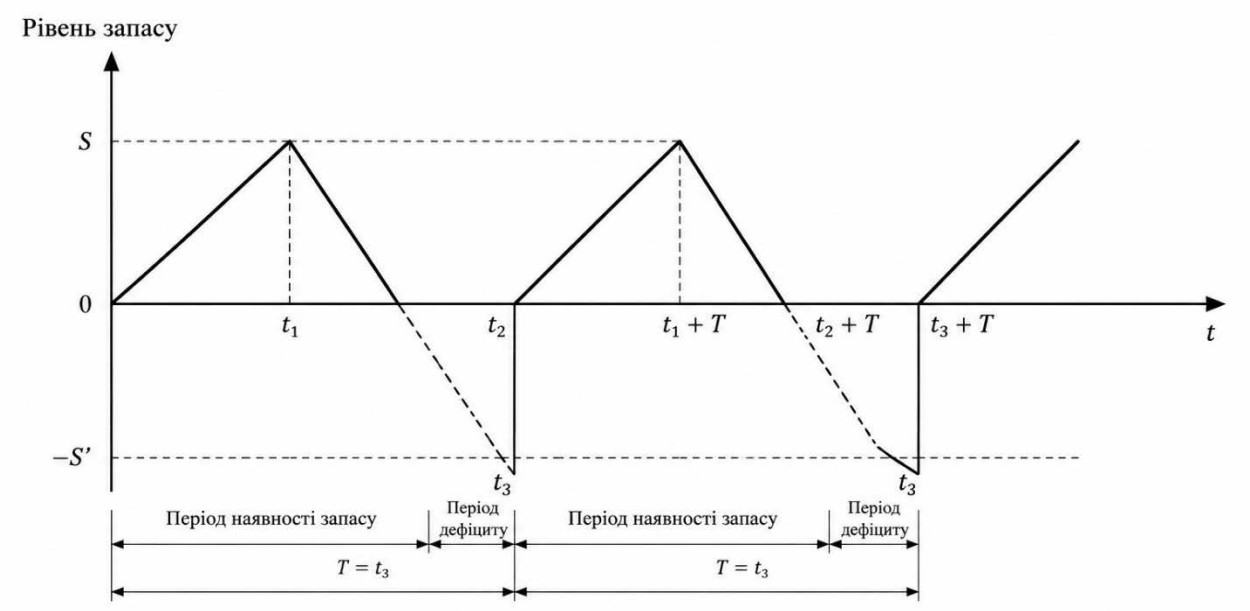
де $Q*$ – розмір виробничого замовлення, який мінімізує загальні витрати за одиницю часу, що дорівнюють сумі витрат на зберігання та витрат на відновлення виробництва;

K – фіксовані витрати на замовлення;

d – інтенсивність попиту;

b – витрати на втрачений попит.

Модель з втраченим попитом представлено на рисунку 1.2.



S - максимальний рівень запасу на складі; S' - максимальний рівень дефіциту (втраченого попиту); t – час; T - тривалість одного циклу управління запасами

Рисунок 1.2 – Модель з втраченим попитом

На рисунку 1.2 зображена модель з втраченим попитом за умови поступового поповнення. Тут S означає максимальний рівень запасу на складі, а S' – максимальний рівень дефіциту. У моделях з втраченим попитом рівень запасу не може бути від'ємним.

Моделі управління запасами, які враховують втрати через дефіцит та втрачені продажі, мають критичне значення для ефективного управління запасами, особливо в умовах високої невизначеності попиту. Ці моделі дозволяють компаніям максимально точно оцінювати та мінімізувати ризики, пов'язані з незадоволеним попитом, що в свою чергу допомагає зберегти

конкурентоспроможність, утримувати клієнтів та покращувати фінансові результати.

Застосування цих моделей вимагає від компаній не тільки глибокого розуміння своїх внутрішніх процесів, але й обізнаності про ринкові тенденції. Аналітичні дані та передові технології у сфері прогнозування та аналізу даних можуть значно покращити точність прогнозів попиту, що дозволяє більш ефективно планувати запаси та знижувати витрати.

У сучасній економіці, де швидкість реакції та адаптивність відіграють ключову роль, моделі управління запасами, орієнтовані на мінімізацію втрат від дефіциту та втраченого попиту, є невід'ємною частиною стратегічного планування будь-якого підприємства, що прагне до ефективності та сталого розвитку.

1.5 Постановка задачі дослідження

На основі розглянутих теоретичних положень, класифікації моделей управління запасами та особливостей стохастичних моделей можна сформулювати задачу подальшого дослідження. Основна увага в роботі приділяється розв’язанню стохастичних задач управління запасами з урахуванням двох важливих сценаріїв: відкладеного попиту та втраченого попиту.

Постановка задачі полягає у розробленні алгоритмічного підходу, який дозволяє моделювати процес зміни рівня запасів в умовах випадкового попиту, оцінювати можливі витрати та визначати раціональні параметри управління запасами. Для цього необхідно врахувати інтенсивність попиту, розмір замовлення, витрати на зберігання, витрати від дефіциту або втрати попиту, а також особливості поповнення запасів.

У наступному розділі буде наведено алгоритм розв’язання стохастичних задач управління запасами для моделей з відкладеним попитом та з втраченим попитом. Також буде представлено програмну реалізацію розроблених

алгоритмів мовою програмування Python із використанням бібліотек NumPy та Matplotlib. Застосування цих інструментів дозволить виконати обчислення, провести імітаційне моделювання та побудувати графічне відображення зміни рівня запасів у часі.

Таким чином, подальший етап дослідження спрямований на практичну реалізацію теоретичних моделей, їх алгоритмізацію та перевірку працездатності за допомогою програмного моделювання. Отримані результати дадуть змогу оцінити ефективність запропонованого підходу та визначити можливості його використання для підтримки прийняття рішень у сфері управління запасами.

Висновки до розділу:

У першому розділі було розглянуто теоретичні основи управління запасами та визначено його роль у забезпеченні стабільної роботи підприємства. Встановлено, що ефективне управління запасами є важливим елементом логістичної діяльності, оскільки дозволяє забезпечити безперервність виробничих і торговельних процесів, мінімізувати витрати на зберігання та зменшити ризики, пов’язані з дефіцитом продукції.

Було наведено класифікацію запасів, зокрема розглянуто поточні, страхові, сезонні та спекулятивні запаси. Також охарактеризовано основні задачі управління запасами, серед яких визначення оптимального рівня запасів, розробка систем контролю, аналіз та оптимізація процесів управління. Окрему увагу приділено методам класифікації запасів, зокрема ABC- та XYZ-аналізу, які дозволяють підприємствам визначати пріоритетні групи товарів і будувати більш ефективну систему контролю.

У розділі було проаналізовано основні види моделей управління запасами. Розглянуто детерміновані та стохастичні моделі, статичні та динамічні підходи, а також моделі з урахуванням дефіциту і без нього. Встановлено, що в умовах невизначеності найбільш доцільним є використання стохастичних моделей, оскільки вони дозволяють враховувати випадковий

характер попиту, зміну часу постачання та інші фактори, які впливають на рівень запасів.

Особливу увагу було приділено стохастичним моделям управління запасами з відкладеним попитом та втраченим попитом. Було визначено їх особливості, основні припущення, сферу застосування та математичні залежності, які використовуються для оцінювання витрат і визначення оптимального розміру замовлення. Також було розглянуто методи оцінки ефективності таких моделей, вплив основних параметрів на управління запасами та практичні приклади їх застосування у різних сферах діяльності.

Отже, у першому розділі сформовано теоретичну основу для подальшого дослідження. Отримані положення дозволяють перейти до наступного етапу роботи - розроблення алгоритмів розв’язання стохастичних задач управління запасами з відкладеним та втраченим попитом, а також їх програмної реалізації мовою Python із використанням відповідних бібліотек для обчислень і візуалізації результатів.

РОЗДІЛ 2
МАТЕМАТИЧНИЙ ОПИС ТА ПРОГРАМНА РЕАЛІЗАЦІЯ СТОХАСТИЧНОЇ ЗАДАЧІ КЕРУВАННЯ ЗАПАСАМИ

2.1 Математичний опис задачі з відкладеним попитом

Сформулюємо задачу з відкладеним попитом. Компанія займається виробництвом деталей із системи транспортування. Нехай щомісяця використовується *k* гнучких тросів у виробничому процесі. Ціна кожного троса, який буде поставлений на підприємство, становить *c* тисяч гривень. Вартість зберігання одного троса - *h* тисяч гривень. Питомі втрати прибутку від дефіциту тросів оцінюватимуться в *b* за штуку. Під час поставки попит на троси буде випадковою величиною, яка буде розподілена від 0 до *p* штук. Потрібно знайти оптимальну стратегію управління запасами.

Побудуємо математичну модель задачі з відкладеним попитом[17]. Введемо позначення:

- m - число гнучких тросів, що використовуються в партії у виробничому процесі;
- e (гр. од.) - витрати підприємства на замовлення поставки партії тросів;
- c (гр. од.) - вартість зберігання одного троса на місяць;
- п / b (гр. од.) - питомі втрати прибутку від дефіциту тросів за штуку;

Для розв'язання даної задачі потрібно враховувати такі фактори:

- d (од.) - відображає інтенсивність попиту;
- K (од.) - відповідає обсягу замовлення;
- h (од.) - визначає вартість зберігання одиниці товару за одиницю часу;
- C (од.) - позначає втрати від відсутності одиниці товару;
- μ (од.) - являє собою очікуваний попит за весь період поставки;
- r - точка відновлення замовлення;

– p - постійна інтенсивність (темп).

Відомо, що витрати, пов'язані з відсутністю товару, пропорційні очікуваному рівню дефіциту. Перейдемо до побудови математичної моделі задачі. Попит підпорядковується нормальному закону розподілу. Нехай f(x, t) - щільність розподілу попиту за період t.

Отже, щільність безумовного розподілу попиту за час виконання замовлення обчислюватиметься за формулою (2.1).

$$D_l(x) = \int_0^{\infty} f(x,l)\, g(l)\ \ dl \tag{2.1}$$

де $D_l(x)$ – функція щільності розподілу попиту за час виконання замовлення;

l – випадкова величина з щільністю розподілу $g(l)$;

$f(x, l)$ – щільність розподілу за період t.

Оскільки час виконання одного замовлення є постійним, то вважатимемо, що:

$$D_l(x) = f(x,l) = \frac{x}{p}, x \in [0,p] \tag{2.2}$$

$$\mu = \int_0^{\infty} x\, D_l(x)\, dx = \int_0^{p} \frac{xdx}{p} \tag{2.3}$$

Формула (2.2) описує лінійний розподіл попиту на товар в інтервалі від 0 до p. Формула (2.3) обчислює середнє значення попиту μ, коли попит у період поставки вважається рівномірно розподіленим від 0 до p.

Нехай $r \leq p$, тоді:

$$D_l(x)dx = \int_p^{\infty} f(x,l)g(l)dl = \int_r^{p} \frac{dx}{p} = 1 - \frac{r}{p} \tag{2.4}$$

$$\int_r^{\propto} xD_l(x)\, dx = \int_r^{p} \frac{xdx}{p} \tag{2.5}$$

За формулою (2.4) обчислюємо ймовірність того, що попит за час поставки буде більшим або дорівнюватиме заданому рівню r. Очікуване значення попиту x для значень, що перевищують поріг r, обчислюватиметься за формулою (2.5).

Для обчислення очікуваного дефіциту скористаємося формулою (2.6)

[19].

$$\bar{B} = \int_r^{\infty} (x - r) D_l(x) dx = \int_r^p \frac{x dx}{p} - r\left(1 - \frac{r}{p}\right) \quad (2.6)$$

Оптимальний розмір замовлення з урахуванням очікуваного попиту та дефіциту обчислюватиметься за формулою (2.7). Формула (2.8) обчислює різницю між очікуваним попитом і фактичним попитом.

$$Q^* = \sqrt{m\,(e + n\,\bar{B})} \quad (2.7)$$

$$e - r^* = \frac{Q^*}{\int_0^p \frac{x dx}{p}} \quad (2.8)$$

де Q^* - оптимальне значення замовлення Q; r^* - оптимальне значення r.

Оскільки:

$$Q^*(0) = \sqrt{\frac{2d(k + b\,\mu)}{h}} \quad (2.9)$$

$$Q_1^*(0) = \frac{bd}{h} \quad (2.10)$$

За формулою (2.9) обчислюємо оптимальний розмір замовлення, який мінімізує сумарні витрати на розміщення замовлень та зберігання запасів. За формулою (2.10) обчислюємо оптимальний розмір додаткового замовлення, який мінімізує витрати на дефіцит та зберігання[17].

Застосовуємо ітераційну процедуру, припускаємо, що:

$$Q_1 = \sqrt{2Kd/h} \quad (2.11)$$

Формула (2.11) обчислює оптимальний розмір замовлення, який мінімізує загальні витрати на розміщення замовлень та зберігання запасів.

Далі обчислюємо r_1, використовуючи друге рівняння в системі, де $Q^* = Q_1$. Потім визначаємо Q_2 з першого рівняння системи, приймаючи $r^* = r_1$, r_2 з другого рівняння, взявши $Q^* = Q_2$ і так далі. Процес ітерацій триватиме доти, доки відмінності між послідовними значеннями оптимального розміру замовлення та точки відновлення замовлення r не стануть досить малими.

Розглянемо цю задачу з конкретними значеннями параметрів. Компанія

займається виробництвом деталей із системи транспортування. Нехай щомісяця використовується 2500 гнучких тросів у виробничому процесі. Ціна кожного троса, який буде поставлений на підприємство, становить 150 тисяч рублів. Вартість зберігання одного троса - 5 тисяч гривень. Питомі втрати прибутку від дефіциту тросів оцінюватимуться в 15 тисяч гривень за штуку. Під час поставки попит на троси буде випадковою величиною, яка буде розподілена від 0 до 150 штук. Потрібно знайти оптимальну стратегію управління запасами.

Використовуючи формулы (2.1) – (2.5), які представлені в математичній моделі, розв'яжемо задачу:

$$D_l(x) = f(x,l) = \frac{x}{150}, x \in [0,150]$$

$$\mu = \int_0^{\infty} D_l(x)\,dx = \int_0^{150} \frac{xdx}{150} = 75.$$

Нехай r ≤ 150, тоді:

$$D_l(x)dx = \int_r^{\infty} f(x,l)g(l)dll = \int_r^{150} \frac{dx}{150} = 1 - \frac{r}{150}$$

$$\int_r^{\propto} xD_l(x)dx = \int_r^{150} \frac{xdx}{150} = 75 - \frac{r^2}{300}$$

Очікуваний дефіцит обчислюється за формулою (2.6) і дорівнюватиме:

$$\bar{B} = \int_r^{\infty} (x-r)D_l(x)dx = 75 - \frac{r^2}{300} - r\left(1 - \frac{r}{150}\right) = 75 + \frac{r^2}{300} - r.$$

На основі формул (2.7) та (2.8) можна обчислити оптимальне значення замовлення Q^* та оптимальне значення r:

$$Q* = \sqrt{2500\,(150 + 15\,\bar{B})} = 50\sqrt{150} + 15\bar{B}$$

$$150 - r^* = \frac{Q^*}{75}$$

Використовуємо формули (2.9) та (2.10), щоб обчислити оптимальний розмір замовлення та розмір додаткового замовлення:

$$Q^*(0) = 100\sqrt{1275} = 3571$$

$$Q_1^*(0) = \frac{bd}{h} = \frac{15 * 2500}{5} = 7500$$

Далі застосовуємо ітераційну процедуру:

$$Q_1 = 100\sqrt{150} = 1224$$

$$r_1 = 100 - \frac{1224}{75} = 83.68$$

$$\bar{B} = 75 + \frac{83.68^2}{300} - 83.68 = 15$$

$$Q_2 = 100\sqrt{165} = 1284$$

$$r_2 = 100 - \frac{1284}{75} = 83.$$

Аналізуючи рішення, зробимо висновки. Оскільки r_1 та r_2 відрізняються незначно, можна припустити $r^* = r_2 \approx 84$, $Q^* = Q_2 = 1284$. У системі підтримується страховий запас $I = B + r - \mu = 24$, очікувана тривалість циклу близько 15 днів.

$$\frac{Kd}{Q} + h\left(\frac{Q}{2} + r - \mu\right) + \frac{bd}{Q}\int_r^\infty (x - r)\, D_l(x)\, dx =$$

$$= \frac{2500 * 150}{1284} + 5\left(\frac{1284}{2} + 24\right) + \frac{15 * 150}{1284} * 15$$

$$= 3648.34$$

Очікувані витрати дорівнюватимуть 3648,34 т. грн.

2.2 Математичний опис задачі з втраченим попитом

Для спрощення представлення сформулюємо задачу з втраченим попитом наступним чином.

Сформулюємо задачу з втраченим попитом[21]. У компанії є певна кількість спеціалізованих каркасів сидінь, які використовуються для поставок. Річний попит становить d штук. Собівартість кожного каркаса становить m. Придбання необхідної кількості каркасів коштує компанії k тисяч рублів.

Коефіцієнт витрат з урахуванням утримання запасів оцінюється в розмірі b. Компанія має можливість придбати дефіцитні каркаси із запасів одного з виробників за додаткові s тисяч рублів на транспортування.

Розподіл попиту в момент поставки нової партії каркасів відповідає нормальному розподілу з математичним очікуванням e та середнім квадратичним відхиленням σ каркасів. Потрібно провести аналіз для визначення оптимального обсягу замовлення та визначення точки відновлення, враховуючи можливі втрати попиту та необхідність додаткових замовлень для його задоволення.

Побудуємо математичну модель задачі з втраченим попитом. Необхідно знайти оптимальну стратегію управління запасами. Введемо позначення:

- d (шт.) - річний попит;
- m (гр. од.) - собівартість кожного каркаса;
- k (шт.) - необхідна кількість каркасів;
- b - коефіцієнт витрат;
- s (гр. od.) - додаткова сума на отримання дефіцитних каркасів;
- e - математичне очікування;
- σ - середнє квадратичне відхилення;
- r - точка відновлення замовлення.

Попит є випадковою величиною, що має стандартний нормальний розподіл. Для обчислення Z скористаємося формулою (2.12):

$$Z = \frac{D - e}{\sigma} \tag{2.12}$$

Для обчислення оптимального розміру замовлення Q^* та точки відновлення запасу r^* скористаємося формулами (2.13) та (2.14):

$$Q^* = \sqrt{(2d(K + b\,\bar{B}) / h} \tag{2.13}$$

$$P(D < r^*) \equiv \int_{r^*}^{\infty} Dl(x)\,dx = \frac{h\,Q^*}{(b\,d + h\,Q^*)} \tag{2.14}$$

де Q^* - оптимальне значення замовлення Q; $P(D < r^*)$ - ймовірність того, що попит буде меншим за r^*; r^* - оптимальне значення r.

Очікуваний дефіцит обчислимо за формулою (2.15):

$$\bar{B} = h\,\varphi(z_r) + (e - r)\,(-F(z_r)) \tag{2.15}$$

Використовуючи формулу (2.15), обчислюємо оптимальний розмір замовлення. За формулою (2.16) обчислюємо функцію розподілу нормальної випадкової величини:

$$Q^* = \sqrt{\frac{2d(K + b\,\bar{B})}{h}} \tag{2.16}$$

$$F(z_r *) = \frac{h\,Q^*}{(b\,d + h\,Q^*)} \tag{2.17}$$

де z_r^* - критичне значення нормальної випадкової величини; $F(z_r^*)$ - функція розподілу нормальної випадкової величини. Функція $F(z_r^*)$ визначає ймовірність того, що попит не перевищить рівень запасу z_r.

Застосовуємо ітераційний алгоритм:

$$Q_1 = \sqrt{\frac{2Kd}{h}} \tag{2.18}$$

$$F(z_r *) = \frac{h\,Q_1}{(b\,d + h\,Q_1)} = \frac{h\sqrt{\frac{2Kd}{h}}}{bd + h\sqrt{\frac{2Kd}{h}}} \tag{2.19}$$

де Q_1 - оптимальний розмір замовлення, який мінімізує загальні витрати на розміщення замовлень та зберігання запасів. Далі знаходимо значення нормальної випадкової величини z_{r1} та точку відновлення замовлення r_1 відповідно. Після цього обчислимо очікуваний дефіцит. Ітерації триватимуть доти, доки різниці між двома послідовними значеннями не стануть досить малими.

Оскільки рівень запасу не може бути від'ємним, середній рівень запасів приблизно дорівнює:

$$(\bar{I} + Q + \bar{I}) / 2 = Q / 2 + r - \mu + B \tag{2.20}$$

Для розрахунку очікуваних річних витрат використовуватиметься така формула:

$$C = \frac{k\,d}{Q} + h\left(\frac{Q}{2} + r - \mu\right) + \left(h + \frac{bd}{Q}\right)\int_r^{\infty}(x - r)\,Dl(x)\,dx \tag{2.21}$$

Тепер розглянемо цю задачу з конкретними значеннями параметрів. У компанії є певна кількість спеціалізованих каркасів сидінь, які використовуються для поставок. Річний попит становить близько 2800 штук. Собівартість кожного каркаса становить 50 тисяч гривень. Придбання необхідної кількості каркасів коштує компанії 600 тисяч гривень. Враховуючи коефіцієнт витрат на утримання запасів, оцінений у розмірі 0,4, компанія має можливість придбати дефіцитні каркаси із запасів одного з виробників за додаткові 400 тисяч гривень на транспортування.

Розподіл попиту в момент поставки нової партії каркасів відповідає нормальному розподілу з математичним очікуванням у 900 каркасів і середнім квадратичним відхиленням 50 каркасів. Потрібно провести аналіз для визначення оптимального обсягу замовлення та визначення точки відновлення, враховуючи можливі втрати попиту та необхідність додаткових замовлень для його задоволення.

Обчислюємо річні витрати зберігання за одиницю:

$$h = b \times m = 0{,}4 \times 50000 = 20000 \text{ грн}$$

Випадкова величина попиту має стандартний нормальний розподіл, отже:

$$\int_r^{\infty} D_l(x)dx = 1 - P(D < r) = 1 - P(z < z_r) = 1 - F(z_r)$$

де $D_l(x)$ – функція щільності ймовірності попиту x за період часу l;

$z_r = \frac{r-900}{50}$ – випадкова величина рівня запасу r;

$F(z_r)$ - функція розподілу нормальної випадкової величини.

$$F(z) = \int_{-\infty}^{z} \varphi(\varepsilon)d\varepsilon = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{z} exp = \left(\frac{\varepsilon^2}{2}\right)d\varepsilon$$

де $\varphi(\varepsilon)$ –щільність ймовірності стандартного нормального розподілу.

Далі обчислюємо очікуване значення z для значень, що перевищують поріг z_r у стандартному нормальному розподілі:

$$\int_{z_r}^{\infty} z\varphi(z)dz = -\int_{z_r}^{\infty} \varphi'(z)dz = \varphi(z_r)$$

Обчислюємо очікуване значення попиту x для $x > r$, використовуючи стандартний нормальний розподіл:

$$\int_{r}^{\infty} D_l(x)dx = \frac{1}{50}\int_{r}^{\infty} x\varphi\left(\frac{x-900}{50}\right)dx = 75 \cdot \varphi(z_r) + 900 \cdot (1 - F(z_r))$$

Очікуваний дефіцит обчислюємо за формулою (2.15).

$$\bar{B} = 50 \cdot \varphi(z_r) + (900 - r) \cdot (-F(z_r))$$

На основі формул (2.16) та (2.17) можна обчислити оптимальне значення:

$$Q^* = \sqrt{\frac{2d \cdot (K + b \cdot \bar{B})}{h}} = 40\sqrt{(105 + 70 \cdot \bar{B}}$$

$$F(z_{r*}) = \frac{h \cdot Q^*}{b \cdot d + h \cdot Q^*} = \frac{56000}{56000 + Q^*}$$

Застосовуємо ітераційний алгоритм:

$$Q_1 = \sqrt{\frac{2 \cdot 2800 \cdot 600000}{20000}} = \sqrt{168000} \approx 410 \text{ каркасів}$$

$$F(z_{r1}) = \frac{56000}{56000 + 410} = 0{,}993$$

Використовуємо таблицю стандартного нормального розподілу, де

$$F(z_{r1}) = 0{,}993;\ z_{r1} = 2{,}5$$

$$r_1 \approx 2{,}5 \cdot 50 + 900 = 1025$$

$$\bar{B} = \frac{50}{\sqrt{2\pi}}\exp\left(-\frac{2{,}5^2}{2}\right) + (900 - 1025) \cdot (1 - 0{,}993) \approx 0{,}014$$

Проведемо ще один ітераційний алгоритм:

$$Q_2 = 40 \cdot \sqrt{(105 + 70 \cdot \bar{B}} = 40 \cdot \sqrt{(105 + 70 \cdot 0{,}014} \approx 412$$

$$F(z_{r2}) = \frac{56000}{56000 + 412} = 0{,}992$$

Використовуємо таблицю стандартного нормального розподілу, де

$$F(z_{r1}) = 0{,}99;\ z_{r1} = 2{,}45$$

$$r_2 \approx 2{,}45 \cdot 50 + 900 = 1026$$

$$\bar{B} = \frac{50}{\sqrt{2\pi}} \exp\left(-\frac{2{,}45^2}{2}\right) + (900 - 1026) \cdot (1 - 0{,}992) \approx 0{,}008$$

Проведемо ще один ітераційний алгоритм:

$$Q_3 = 40 \cdot \sqrt{(105 + 70 \cdot \bar{B}} = 40 \cdot \sqrt{(105 + 70 \cdot 0{,}008} \approx 411$$

$$F(z_{r3}) = \frac{56000}{56000 + 411} = 0{,}992$$

Використовуємо таблицю стандартного нормального розподілу, де

$$F(z_{r1}) = 0{,}992;\ z_{r1} = 2{,}45; r_2 = 1026$$

$$\bar{B} = \frac{50}{\sqrt{2\pi}} \exp\left(-\frac{2{,}45^2}{2}\right) + (900 - 1026) \cdot (1 - 0{,}992) \approx 0{,}008$$

$F(z_{r3}) \approx F(z_{r3})$ тому $r_3 = r_2 = r^*$, $Q^* = Q_3$

Виходячи з перерахованих вище рішень, можна обчислити гарантійний запас:

$$r - e = 1026 - 900 = 126$$

А очікувані річні витрати розраховуватимуться за формулою (2.21) і становитимуть:

$$C = \frac{600000 \cdot 2800}{411} + 20000 \cdot \left(\frac{411}{2} + 1026 - 900\right) + (20000 + \frac{400000 \cdot 2800}{411}) \cdot 0.008 \approx 10739551{,}73$$

Таким чином, очікувані річні витрати дорівнюють 10739551,73 т. грн

2.3 Вибір мови програмування та середовища розробки

Розробка алгоритму та його програмна реалізація для розв'язання стохастичної задачі управління запасами є важливим етапом, що дозволяє

ефективно справлятися з невизначеністю попиту та інших параметрів. У цій главі буде описано процес створення алгоритмів управління запасами для двох різних моделей: модель з відкладеним попитом та модель з втраченим попитом. Для реалізації використовується мова програмування Python та редактор коду Visual Studio Code (VS Code)[22-24].

Перерахуємо основні причини вибору Python:

– Простіть та читаність синтаксису, що полегшує розробку та підтримку коду.

– Наявність багатого набору бібліотек для математичних і статистичних обчислень, таких як NumPy, SciPy, Matplotlib та Pandas.

– Спільнота та документація, які надають безліч прикладів і рішень.

Visual Studio Code (VS Code) було обрано як текстовий редактор з таких причин: безкоштовність та відкритий вихідний код, численні розширення, інтеграція з терміналом та висока продуктивність.

2.4 Розробка алгоритму

Стохастична задача управління запасами полягає в оптимізації рівня запасів з урахуванням невизначеності попиту та часу поставки. Основні параметри моделі включають: середній попит (μ) та його стандартне відхилення (σ), час поставки (L) та його стандартне відхилення, вартість зберігання (h) та вартість замовлення (K), штраф за дефіцит.

Метою є мінімізаія загальних витрат, що включають витрати на зберігання, розміщення замовлень та дефіцит. Алгоритми розв'язання задач включають такі кроки:

– Ініціалізація параметрів моделі;

– Генерація випадкового попиту на основі нормального розподілу;

– Обчислення рівня запасів та відповідних витрат;

– Оптимізація рівня замовлення для мінімізації сукупних витрат.

2.5 Модель з відкладеним попитом

Модель з відкладеним попитом передбачає, що незадоволений попит переноситься на майбутнє. Це означає, що компанія виконуватиме замовлення, навіть якщо це потребує часу. Виходячи з даних умов і значень, напишем код для розв'язання задачі управління запасами з моделлю відкладеним попиту. Цей код є реалізацією моделі відкладеного попиту з графічним інтерфейсом на Python, використовуючи бібліотеки NumPy, Matplotlib, SciPy, StatsModels та Tkinter.

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm
from statsmodels.tsa.arima.model import ARIMA
import tkinter as tk
from tkinter import ttk


class DelayedDemandModel:
    def __init__(self, d, K, h, b, mu, historical_data=None):
        self.d = d  # Інтенсивність попиту
        self.K = K  # Фіксовані витрати на замовлення
        self.h = h  # Питомі витрати на зберігання
        self.b = b  # Питомі втрати прибутку від дефіциту
        self.mu = mu  # Математичне очікування попиту за весь період поставки
        self.historical_data = historical_data
```

Рисунок 2.1 – Клас DelayedDemandModel, Конструктор _Init_

На рисунку 2.1 використовується клас DelayedDemandModel, який реалізує модель відкладеного попиту, надаючи методи для прогнозування попиту, розрахунку оптимального розміру замовлення, аналізу чутливості та візуалізації результатів.

Далі представлено Конструктор *Init*, за допомогою якого ініціалізується модель із заданими параметрами.

```
        model_fit = model.fit()
        forecast = model_fit.forecast(steps=12)
        self.forecasted_demand = forecast
    else:
        self.forecasted_demand = np.full(12, self.mu)


def calculate_optimal_order(self):
    Q0 = np.sqrt((2 * self.d * self.K) / (self.h + self.b))  # Початковий розмір замовлення без урахування дефіци
    Q1 = np.sqrt((2 * self.d * self.K) / self.h)  # Початковий розмір замовлення з урахуванням зберігання
    r1 = self.mu - Q1 / 2  # Рівень замовлення при мінімальному запасі

    B = (self.mu - r1) * (1 - (r1 / self.mu))

    self.Q_opt = Q0 + B
    self.r_opt = r1 + B

    self.C = self.K * self.d / self.Q_opt + self.h * (self.Q_opt ** 2) / 2 + self.b * B
```

Рисунок 2.2 – Методи forecast_demand і calculate_optimal_order.

На рисунку 2.2 представлено фрагмент коду, де використовується метод forecast_demand, який використовує модель ARIMA для прогнозування попиту на 12 місяців уперед, якщо наявні історичні дані. В іншому випадку використовується середнє значення попиту μ.

Метод calculate_optimal_order розраховує оптимальний розмір замовлення (Q_{opt}) та рівень дефіциту (r_{opt}), а також очікувані витрати (C), використовуючи задані параметри моделі.

```
def sensitivity_analysis(self, param_range, param_name):
    results = []
    original_value = getattr(self, param_name)

    for value in param_range:
        setattr(self, param_name, value)
        self.calculate_optimal_order()
        results.append((value, self.Q_opt, self.r_opt, self.C))

    setattr(self, param_name, original_value)
    return results


def display_results(self):
    return (
        f"Оптимальний розмір замовлення: {self.Q_opt:.2f}\n"
        f"Очікуваний рівень дефіциту: {self.r_opt:.2f}\n"
        f"Очікувані витрати: {self.C:.2f}"
    )
```

Рисунок 2.3 – Методи sensitivity_analysis и display_results.

На рисунку 2.3 представлено метод sensitivity_analysis, який виконує

аналіз чутливості для заданого параметра моделі в певному діапазоні значень, повертаючи список результатів (значення параметра, оптимальний розмір замовлення, рівень дефіциту, очікувані витрати). А також метод display_results, за допомогою якого повертається рядок із результатами оптимізації: оптимальний розмір замовлення, очікуваний рівень дефіциту та очікувані витрати.

На рисунках 2.4-2.6 представлено методи візуалізації plot_inventory_levels, plot_forecasted_demand і plot_sensitivity_analysis. Дані методи створюють графіки:

- рівня запасів із відкладеним попитом;
- прогнозованого попиту на основі історичних даних;
- аналізу чутливості для заданого параметра.

```
def plot_inventory_levels(self):
    time_period = np.linspace(0, 12, 100)
    inventory_levels = self.Q_opt * np.sin(time_period) + self.r_opt

    plt.plot(time_period, inventory_levels, label='Рівень запасів')
    plt.axhline(y=0, color='r', linestyle='--', label='Рівень дефіциту')
    plt.xlabel('Час (місяці)')
    plt.ylabel('Рівень запасів')
    plt.title('Рівень запасів з відкладеним попитом')
    plt.legend()
    plt.grid(True)
    plt.show()

def plot_forecasted_demand(self):
    if self.historical_data is not None:
        plt.plot(self.historical_data, label='Історичні дані')
        plt.plot(
            np.arange(
                len(self.historical_data),
                len(self.historical_data) + len(self.forecasted_demand)
            ),
            self.forecasted_demand,
            label='Прогнозований попит',
            linestyle='--'
        )

    plt.xlabel('Час')
    plt.ylabel('Попит')
    plt.title('Прогноз попиту')
    plt.legend()
    plt.grid(True)
    plt.show()
```

Рисунок 2.4 – Методи візуалізації plot_inventory_levels і plot_forecasted_demand

```
def plot_sensitivity_analysis(self, results, param_name):
    param_values, Q_opts, r_opts, costs = zip(*results)

    plt.figure(figsize=(10, 6))

    plt.subplot(3, 1, 1)
    plt.plot(param_values, Q_opts, marker='o')
    plt.xlabel(param_name)
    plt.ylabel('Оптимальний розмір замовлення (Q_opt)')
    plt.grid(True)

    plt.subplot(3, 1, 2)
    plt.plot(param_values, r_opts, marker='o')
    plt.xlabel(param_name)
    plt.ylabel('Очікуваний рівень дефіциту (r_opt)')
    plt.grid(True)
```

```
    plt.subplot(3, 1, 3)
    plt.plot(param_values, costs, marker='o')
    plt.xlabel(param_name)
    plt.ylabel('Очікувані витрати (C)')
    plt.grid(True)

    plt.tight_layout()
    plt.show()
```

Рисунок 2.5 – Метод візуалізації plot_sensitivity_analysis

```
class App(tk.Tk):
    def __init__(self):
        super().__init__()
        self.title("Модель відкладеного попиту")
        self.geometry("400x300")

        # Мітки та поля введення для параметрів
        self.create_widgets()

        self.model = None
```

Рисунок 2.6 – Клас App і конструктор _Init_

Далі представлено клас App, який реалізує графічний інтерфейс додатка з використанням Tkinter. Конструктор *Init* ініціалізує головне вікно додатка із заголовком і розміром, а також викликає метод для створення віджетів.

На рисунках 2.7-2.8 представлено методи create_widgets, run_model і plot_results.

Метод create_widgets створює та розміщує на вікні віджети для введення параметрів моделі та кнопку для запуску моделі. Також створюється текстове поле для відображення результатів.

Метод run_model викликається при натисканні кнопки «Запустити модель». Він зчитує значення параметрів із полів введення, створює екземпляр DelayedDemandModel, виконує прогнозування попиту та розрахунок оптимального замовлення, а потім відображає результати в текстовому полі. Також викликає метод plot_results для побудови графіків.

Метод plot_results будує графіки рівня запасів, прогнозованого попиту та аналізу чутливості для математичного очікування попиту μ.

```
def create_widgets(self):
    labels = [
        "Інтенсивність попиту (d)",
        "Фіксовані витрати на замовлення (K)",
        "Питомі витрати на зберігання (h)",
        "Питомі втрати прибутку від дефіциту (b)",
        "Математичне очікування попиту (μ)"
    ]
```

```
    self.entries = []

    for i, label in enumerate(labels):
        ttk.Label(self, text=label).grid(row=i, column=0, padx=10, pady=5, sticky='w')
        entry = ttk.Entry(self)
        entry.grid(row=i, column=1, padx=10, pady=5, sticky='ew')
        self.entries.append(entry)

    ttk.Button(
        self,
        text="Запустити модель",
        command=self.run_model
    ).grid(row=len(labels), column=0, columnspan=2, pady=10)

    self.results_text = tk.Text(self, height=10, width=50)
    self.results_text.grid(row=len(labels) + 1, column=0, columnspan=2, padx=10, pady=10)
```

```
def run_model(self):
    try:
        d = float(self.entries[0].get())
        K = float(self.entries[1].get())
        h = float(self.entries[2].get())
        b = float(self.entries[3].get())
        μ = float(self.entries[4].get())

        historical_data = np.random.poisson(μ, 50)

        self.model = DelayedDemandModel(d, K, h, b, μ, historical_data)
        self.model.forecast_demand()
        self.model.calculate_optimal_order()
        results = self.model.display_results()

        self.results_text.delete(1.0, tk.END)
        self.results_text.insert(tk.END, results)

        self.plot_results()
```

Рисунок 2.7 – Методи create_widgets і run_model

```
    except ValueError as e:
        self.results_text.delete(1.0, tk.END)
        self.results_text.insert(tk.END, f"Помилка введення: {e}")


def plot_results(self):
    self.model.plot_inventory_levels()
    self.model.plot_forecasted_demand()

    param_range = np.linspace(50, 100, 10)
    results = self.model.sensitivity_analysis(param_range, 'μ')
    self.model.plot_sensitivity_analysis(
        results,
        'Математичне очікування попиту (μ)'
    )


if __name__ == "__main__":
    app = App()
    app.mainloop()
```

Рисунок 2.8 – Метод plot_results

Користувач може ввести значення для параметрів моделі через інтерфейс, запустити розрахунок оптимального розміру замовлення та переглянути результати, включаючи графіки прогнозування попиту, рівня запасів і аналізу чутливості.

Користувач вводить значення параметрів d, K, h, b, μ, натискає кнопку «Запустити модель», після чого отримує результати розрахунків і візуалізацію даних на графіках. На рисунку 2.9 представлено інтерфейс для введення параметрів.

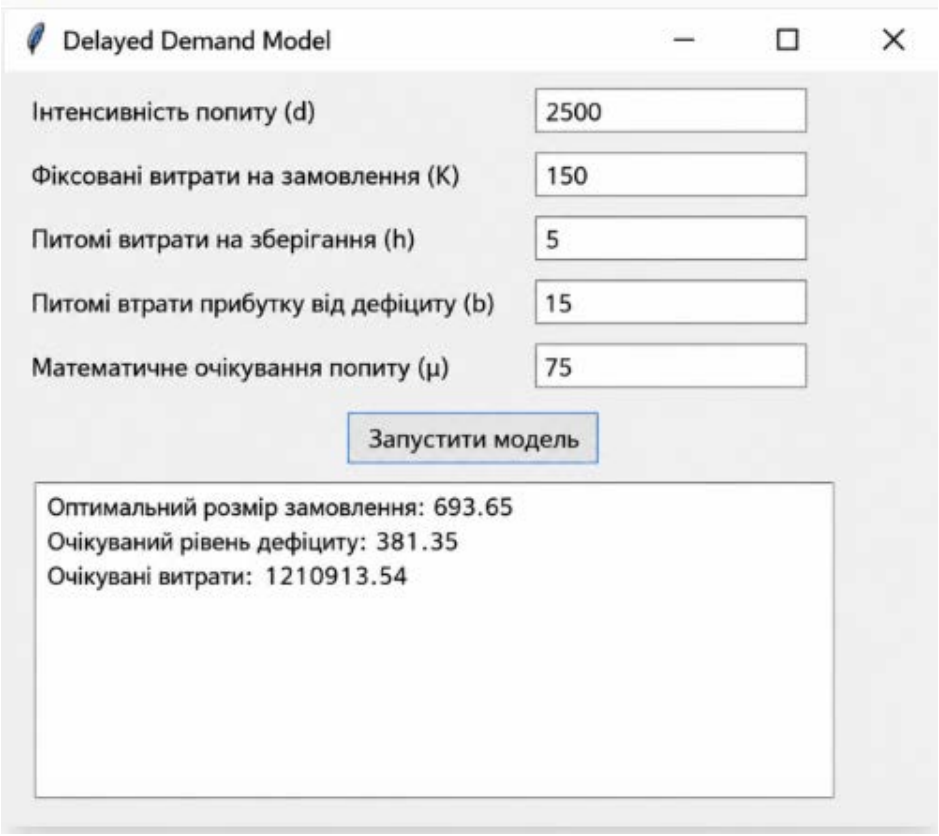


Рисунок 2.9 – Інтерфейс для вводу параметрів

На рисунках 2.10-2.13 показано результати, які виводитиме код:

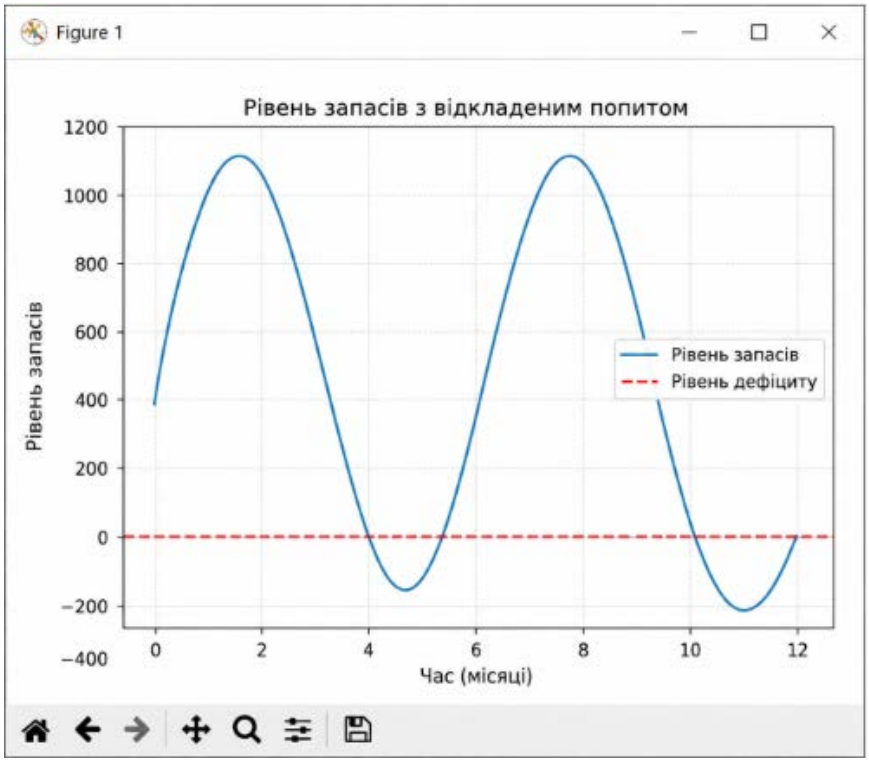


Рисунок 2.10 – Результати задачі з моделлю відкладеного попиту

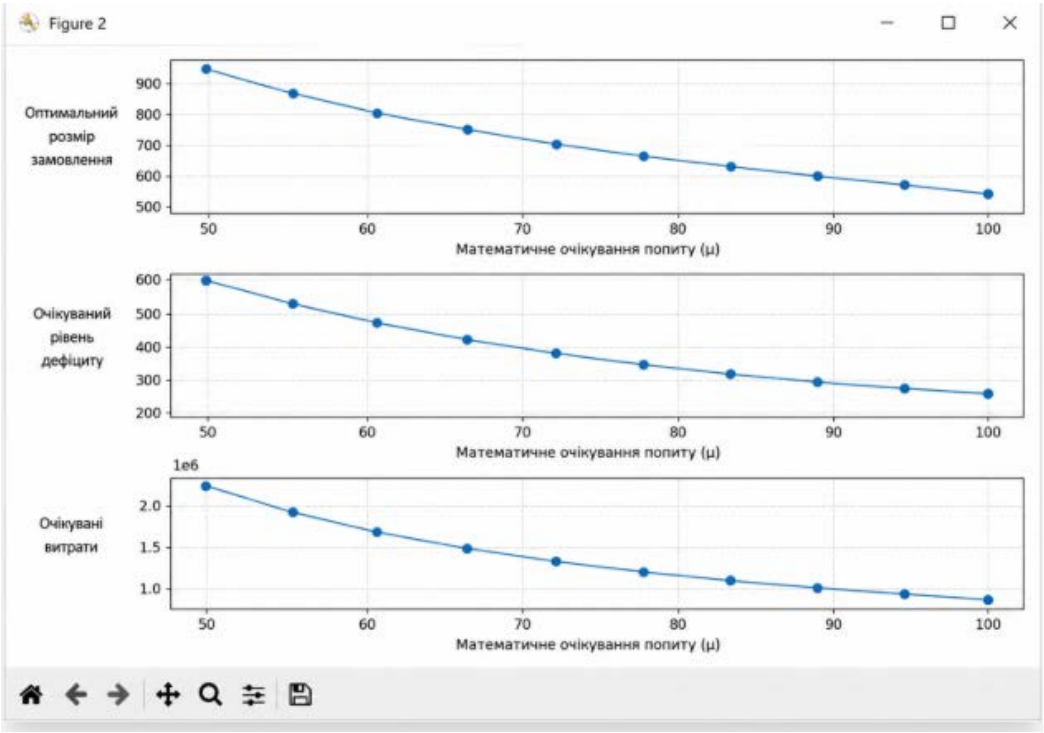


Рисунок 2.11 – Результати задачі з моделлю відкладеного попиту

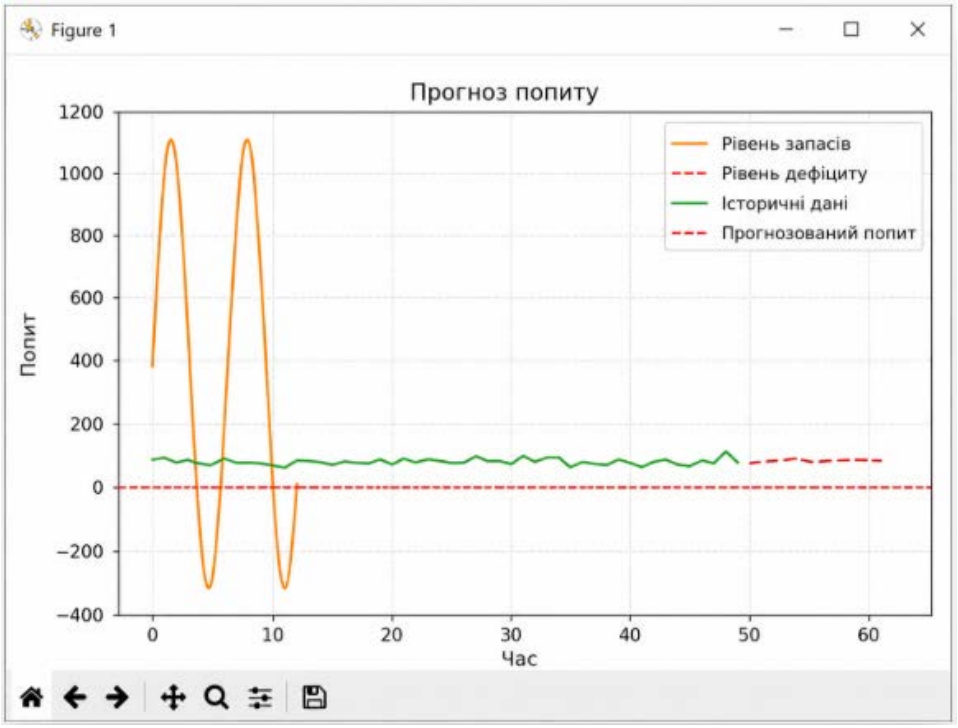


Рисунок 2.12 – Результати задачі з моделлю відкладеного попиту

На основі графіків, представлених на рисунках (2.10–2.12), можна легко виконати аналіз і прогнозування стану складських запасів.

У моделі з відкладеним попитом також спостерігаються коливання рівня

запасів, однак дефіцит не вважається остаточно втраченою реалізацією, а переноситься на наступні періоди. Прогноз попиту показує, що на основі історичних даних формується подальше очікуване значення попиту. Графіки чутливості демонструють, що зі зростанням математичного очікування попиту оптимальний розмір замовлення, очікуваний рівень дефіциту та загальні витрати поступово зменшуються. Це свідчить про більшу залежність моделі з відкладеним попитом від параметра очікуваного попиту.

2.6 Модель з втраченим попитом

Модель із втраченим попитом передбачає, що незадоволений попит не переноситься на майбутнє, а втрачається назавжди, що може призвести до упущеного прибутку та втрати клієнтів.

Виходячи з даних умов і значень, напишемо код для розв'язання задачі управління запасами з моделлю відкладеного попиту.

Цей код являє собою реалізацію моделі втраченого попиту з графічним інтерфейсом на Python, використовуючи бібліотеки NumPy, Matplotlib, SciPy та Tkinter.

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm
from scipy.optimize import minimize
import tkinter as tk
from tkinter import ttk


class LostSalesModel:
    def __init__(self, d, m, K, b, s, μ, h):
        self.d = d  # Річний попит
        self.m = m  # Собівартість кожного каркаса
        self.K = K  # Необхідні витрати на замовлення
        self.b = b  # Коефіцієнт витрат
        self.s = s  # Додаткова сума на отримання відсутніх каркасів
        self.μ = μ  # Математичне очікування попиту
        self.h = h  # Середньоквадратичне відхилення
```

Рисунок 2.13 – Клас LostSalesModel і конструктор _Init_

На рисунку 2.13 представлено фрагмент коду, де використовується клас LostSalesModel, який реалізує модель втраченого попиту, надаючи методи для розрахунку оптимального розміру замовлення, оптимізації замовлення, аналізу чутливості та візуалізації результатів.

Далі представлено Конструктор *Init*, за допомогою якого ініціалізується модель із заданими параметрами.

На рисунках 2.14-2.15 представлено методи optimize_order, sensitivity_analysis і display_results.

За допомогою методу optimize_order відбувається оптимізація розміру замовлення з використанням функції мінімізації minimize з бібліотеки SciPy. Функція мінімізує витрати, розраховуючи їх на основі поточних параметрів моделі.

Метод sensitivity_analysis виконує аналіз чутливості для заданного параметра моделі в певному діапазоні значень, повертаючи список результатів (значення параметра, оптимальний розмір замовлення, рівень дефіциту, очікувані витрати). Метод display_results повертає рядок із результатами оптимізації: оптимальний розмір замовлення, очікуваний рівень дефіциту та очікувані витрати.

```
def optimize_order(self):
    def cost(Q):
        F_zr = norm.cdf(Q / (self.b * self.d + self.h * Q))
        Z_r = norm.ppf(F_zr)
        B = self.h * norm.pdf(Z_r) + self.μ - Z_r
        C = (self.K * self.d / Q + self.h * (Q ** 2) / 2 +
```

Рисунок 2.14 – Метод optimize_order

```
            self.s * norm.pdf(z_r) + self.b * self.d * norm.pdf(z_r)
        )
        return C

    result = minimize(cost, x0=self.Q1, method='BFGS')
    self.Q_opt = result.x[0]
    self.C = result.fun
    self.r_opt = norm.ppf(
        norm.cdf(self.Q_opt / (self.b * self.d + self.h * self.Q_opt))
    )


    def sensitivity_analysis(self, param_range, param_name):
        results = []
        original_value = getattr(self, param_name)

        for value in param_range:
            setattr(self, param_name, value)
            self.calculate_optimal_order()
            results.append((value, self.Q_opt, self.r_opt, self.C))

        setattr(self, param_name, original_value)
        return results


    def display_results(self):
        return (
            f"Оптимальний розмір замовлення: {self.Q_opt:.2f}\n"
            f"Очікуваний рівень дефіциту: {self.r_opt:.2f}\n"
            f"Очікувані витрати: {self.C:.2f}"
        )
```

Рисунок 2.15 – Метод sensitivity_analysis і display_results

На рисунках 2.16 - 2.17 представлено методи візуалізації plot_inventory_levels і plot_sensitivity_analysis. Дані методи створюють графіки:

- рівня запасів із відкладеним попитом;
- аналізу чутливості для заданого параметра.

```
def plot_inventory_levels(self):
    time_period = np.linspace(0, 12, 100)
    inventory_levels = self.Q_opt * np.sin(time_period) + self.r_opt

    plt.plot(time_period, inventory_levels, label='Рівень запасів')
    plt.axhline(y=0, color='r', linestyle='--', label='Рівень дефіциту')

    plt.xlabel('Час (місяці)')
    plt.ylabel('Рівень запасів')
    plt.title('Рівень запасів з втраченим попитом')

    plt.legend()
    plt.grid(True)
    plt.show()
```

Рисунок 2.16 – Метод візуалізації plot_inventory_levels

```
def plot_sensitivity_analysis(self, results, param_name):
    param_values, Q_opts, r_opts, costs = zip(*results)

    plt.figure(figsize=(10, 6))

    plt.subplot(3, 1, 1)
    plt.plot(param_values, Q_opts, marker='o')
    plt.xlabel(param_name)
    plt.ylabel('Оптимальний розмір замовлення (Q_opt)')
    plt.grid(True)

    plt.subplot(3, 1, 2)
    plt.plot(param_values, r_opts, marker='o')
    plt.xlabel(param_name)
    plt.ylabel('Очікуваний рівень дефіциту (r_opt)')
    plt.grid(True)

    plt.subplot(3, 1, 3)
    plt.plot(param_values, costs, marker='o')
    plt.xlabel(param_name)
    plt.ylabel('Очікувані витрати (C)')
    plt.grid(True)

    plt.tight_layout()
    plt.show()
```

Рисунок 2.17 – Метод візуалізації plot_sensitivity_analysis

На рисунку 2.18 представлено клас App, який реалізує графічний інтерфейс додатка з використанням Tkinter. Конструктор *Init* ініціалізує головне вікно додатка із заголовком і розміром, а також викликає метод для створення віджетів.

```
class App(tk.Tk):
    def __init__(self):
        super().__init__()
        self.title("Модель втрачених продажів")
        self.geometry("400x400")

        self.create_widgets()

        self.model = None
```

Рисунок 2.18 – Клас App і конструктор _Init_

На рисунках 2.19-2.20 представлено методи create_widgets, run_model і

plot_results. Метод create_widgets створює та розміщує на вікні віджети для введення параметрів моделі та кнопку для запуску моделі. Також створюється текстове поле для відображення результатів. Метод run_model викликається при натисканні кнопки «Запустити модель». Він зчитує значення параметрів із полів введення, створює екземпляр LostSalesModel, виконує прогнозування попиту та розрахунок оптимального замовлення, а потім відображає результати в текстовому полі. Також викликає метод plot_results для побудови графіків. Метод plot_results будує графіки рівня запасів, прогнозованого попиту та аналізу чутливості для математичного очікування попиту μ

```
def create_widgets(self):
    labels = [
        'Річний попит (d)',
        'Собівартість кожного каркаса (m)',
        'Необхідні витрати на замовлення (K)',
        'Коефіцієнт витрат (b)',
        'Додаткова сума на отримання відсутніх каркасів (s)',
        'Математичне очікування попиту (μ)',
        'Середньоквадратичне відхилення (h)'
    ]

    self.entries = []

    for i, label in enumerate(labels):
        ttk.Label(self, text=label).grid(row=i, column=0, padx=10, pady=5, sticky='w')
        entry = ttk.Entry(self)
        entry.grid(row=i, column=1, padx=10, pady=5, sticky='ew')
        self.entries.append(entry)

   ttk.Button(
       self,
       text="Запустити модель",
       command=self.run_model
   ).grid(row=len(labels), column=0, columnspan=2, pady=10)

   self.results_text = tk.Text(self, height=10, width=50)
   self.results_text.grid(row=len(labels) + 1, column=0, columnspan=2, padx=10, pady=10)
```

Рисунок 2.19 – Метод create_widgets

```
def run_model(self):
    try:
        d = float(self.entries[0].get())
        m = float(self.entries[1].get())
        K = float(self.entries[2].get())
        b = float(self.entries[3].get())
        s = float(self.entries[4].get())
        μ = float(self.entries[5].get())
        h = float(self.entries[6].get())

        self.model = LostSalesModel(d, m, K, b, s, μ, h)
        self.model.calculate_optimal_order()
        self.model.optimize_order()
        results = self.model.display_results()

        self.results_text.delete(1.0, tk.END)
        self.results_text.insert(tk.END, results)

        self.plot_results()
```

Рисунок 2.20– Метод run_model

```
    except ValueError as e:
        self.results_text.delete(1.0, tk.END)
        self.results_text.insert(tk.END, f"Помилка введення: {e}")


def plot_results(self):
    self.model.plot_inventory_levels()

    param_range = np.linspace(50, 150, 10)
    results = self.model.sensitivity_analysis(param_range, 'μ')
    self.model.plot_sensitivity_analysis(
        results,
        'Математичне очікування попиту (μ)'
    )


if __name__ == "__main__":
    app = App()
    app.mainloop()
```

Рисунок 2.21 – Метод plot_results

```
if __name__ == "__main__":
    app = App()
    app.mainloop()
```

Рисунок 2.22 – Запуск програми

На рисунку 2.22 створюється екземпляр App і запускається головний цикл Tkinter (app.mainloop()), що дозволяє взаємодіяти з графічним

інтерфейсом.

Користувач може ввести значення для параметрів моделі через інтерфейс, запустити розрахунок оптимального розміру замовлення та переглянути результати, включаючи графіки рівня запасів і аналізу чутливості.

Користувач вводить значення параметрів d, m, K, h, b, μ, s, натискає кнопку «Запустити модель», після чого отримує результати розрахунків і візуалізацію даних на графіках. На рисунку 2.23 представлено інтерфейс для введення параметрів.

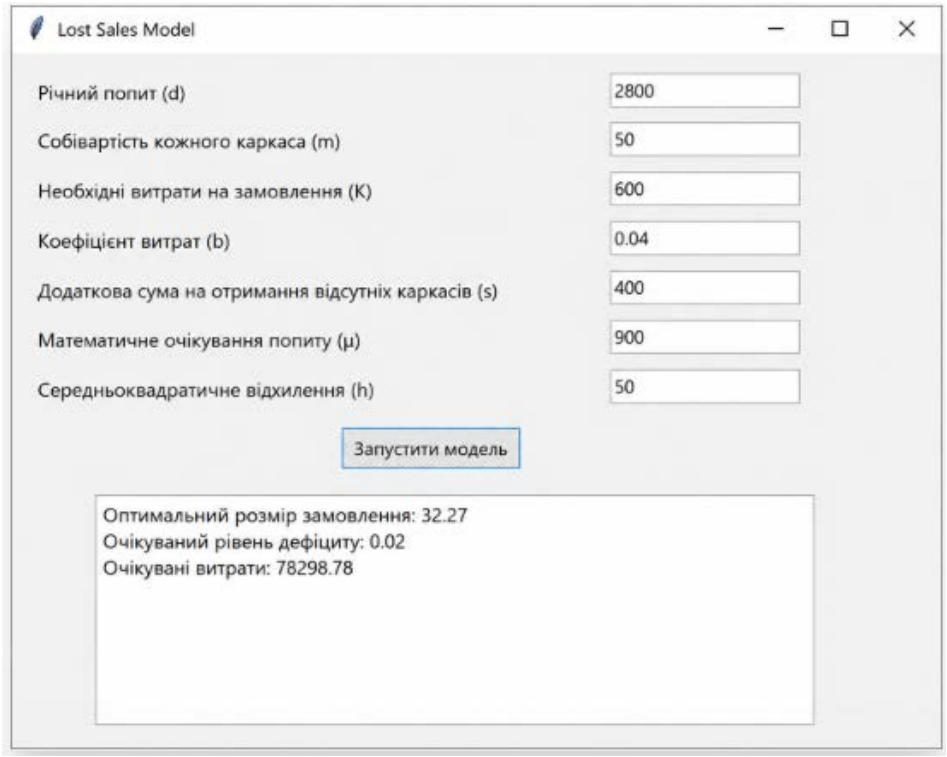


Рисунок 2.23 – Інтерфейс для вводу параметрів

На рисунках 2.24-2.26 показано результати, які виводитиме код:

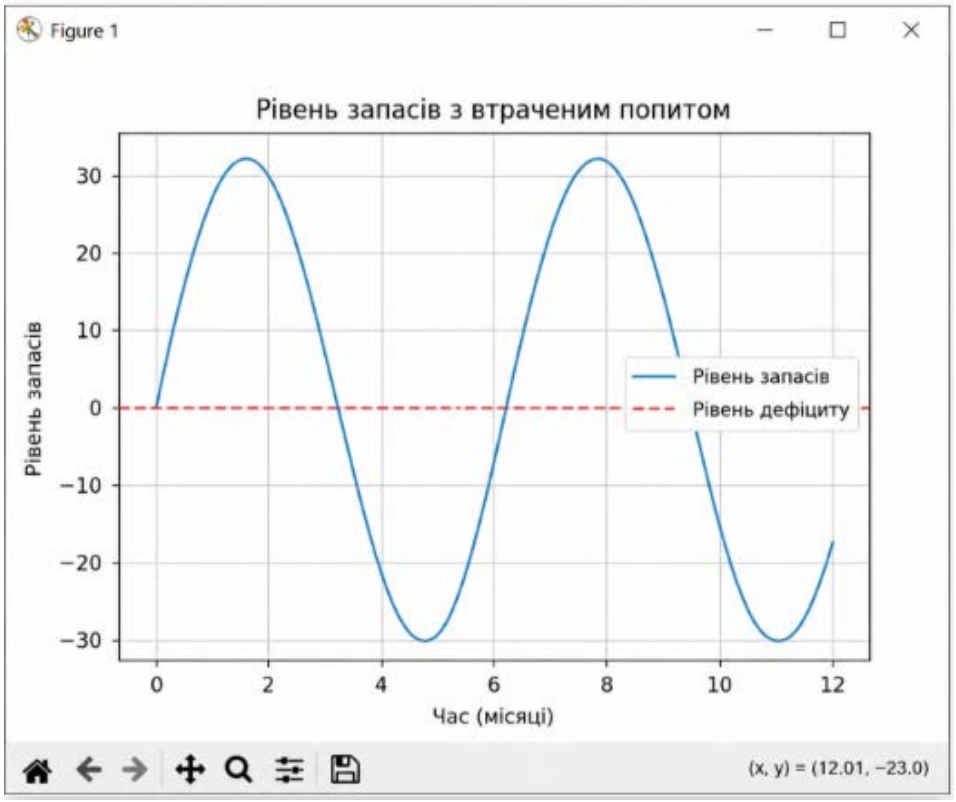


Рисунок 2.24 – Результати задачі з моделлю втраченого попиту

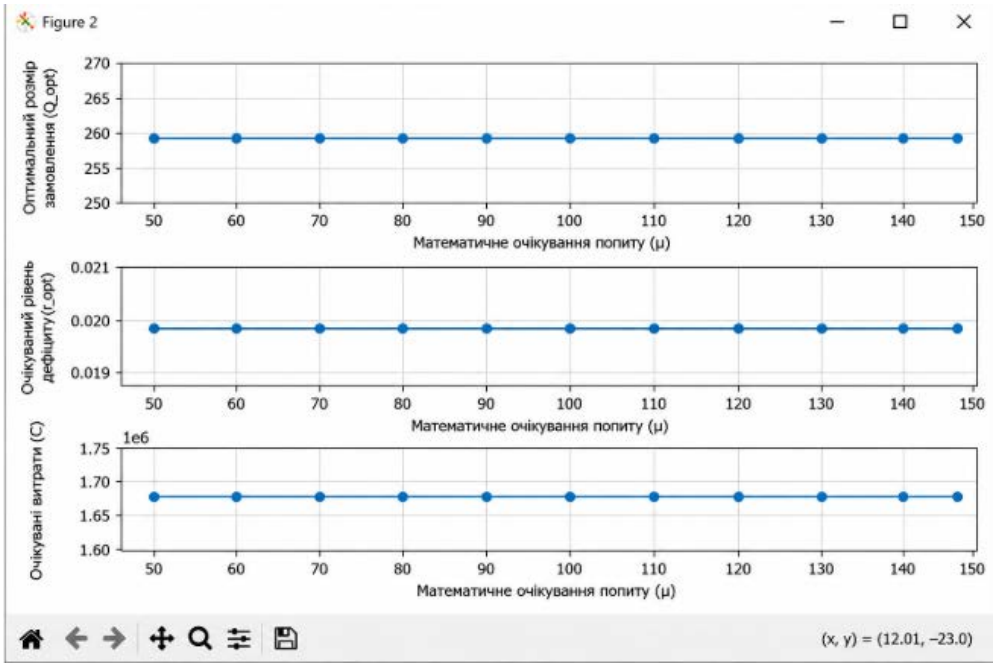


Рисунок 2.25 – Результати задачі з моделлю втраченого попиту

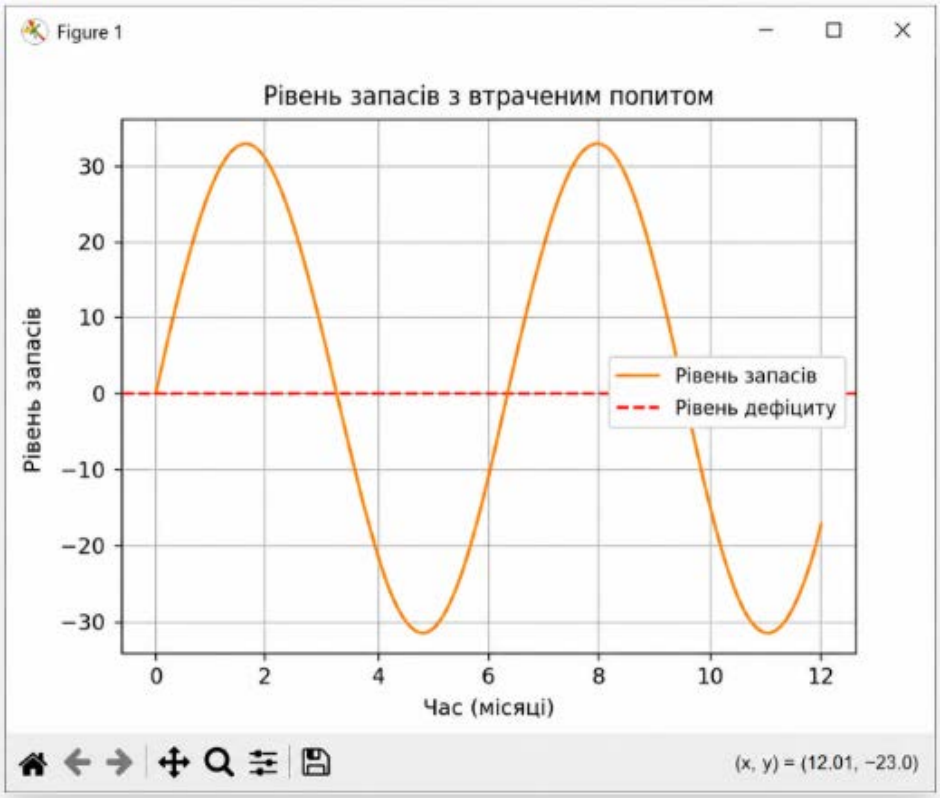


Рисунок 2.26 – Результати задачі з моделлю втраченого попиту

На основі графіків, представлених на рисунках (2.24–2.26), можна легко виконати аналіз і прогнозування стану складських запасів. За результатами графіків видно, що рівень запасів змінюється хвилеподібно: у певні періоди запас є додатним, а в окремі моменти опускається нижче нульового рівня, що свідчить про виникнення дефіциту. Оскільки модель враховує втрачений попит, незадоволений попит не переноситься на майбутні періоди, а призводить до втрат. Графік чутливості показує, що при зміні математичного очікування попиту оптимальний розмір замовлення, очікуваний рівень дефіциту та очікувані витрати майже не змінюються. Це означає, що для заданих параметрів модель є відносно стабільною до зміни цього показника.

Висновки до розділу:

У другому розділі було виконано математичний опис та програмну реалізацію стохастичної задачі управління запасами з урахуванням різних сценаріїв виникнення дефіциту. Основну увагу приділено двом моделям:

моделі з відкладеним попитом та моделі з втраченим попитом, які дозволяють дослідити поведінку системи запасів за умов випадкового характеру попиту.

Було наведено математичний опис задачі з відкладеним попитом. У межах цієї моделі дефіцит розглядається як незадоволений попит, що не втрачається, а переноситься на наступні періоди. Також було сформульовано модель з втраченим попитом, у якій незадоволений попит не компенсується в майбутньому, а призводить до втрати частини замовлень. Для обох моделей було визначено основні параметри, витрати на оформлення замовлення, зберігання запасів і дефіцит, а також сформульовано залежності для пошуку оптимального розміру замовлення.

Окремо було обґрунтовано вибір мови програмування Python та середовища розробки для реалізації поставленої задачі. Python було обрано завдяки простоті синтаксису, наявності бібліотек для математичних обчислень, роботи з масивами даних і побудови графіків. Це дало змогу реалізувати алгоритми розрахунку параметрів моделей, виконати числове моделювання та представити результати у вигляді графіків.

У розділі було розроблено алгоритм розв’язання стохастичної задачі управління запасами. Алгоритм передбачає введення початкових параметрів, розрахунок очікуваних витрат, визначення оптимального розміру замовлення, оцінювання рівня дефіциту та аналіз результатів для двох варіантів моделі. На основі програмної реалізації було побудовано моделі з відкладеним і втраченим попитом, що дозволило порівняти їх поведінку та оцінити вплив дефіциту на загальні витрати системи.

Розроблений алгоритм і його програмна реалізація мовою Python дозволяють досліджувати стохастичні задачі управління запасами, визначати оптимальні параметри замовлення, аналізувати очікуваний рівень дефіциту та прогнозувати витрати залежно від обраної моделі. Отримані результати можуть бути використані для подальшого аналізу ефективності моделей управління запасами та обґрунтування вибору оптимального підходу для підприємства.

ВИСНОВКИ

У випускній кваліфікаційній роботі було розглянуто стохастичні моделі управління запасами з відкладеним та втраченим попитом. Актуальність роботи зумовлена тим, що в реальних умовах діяльності підприємства попит на продукцію має випадковий характер, а тому використання лише детермінованих моделей не завжди дозволяє отримати точні результати для планування запасів.

У першому розділі було досліджено теоретичні основи управління запасами. Розглянуто сутність запасів, їх класифікацію, основні задачі управління та методи аналізу запасів. Також було проаналізовано детерміновані та стохастичні моделі, визначено їх особливості, переваги та обмеження. Окрему увагу приділено моделям з урахуванням дефіциту, зокрема моделям з відкладеним і втраченим попитом.

У другому розділі було виконано математичний опис стохастичних задач управління запасами. Для моделей з відкладеним та втраченим попитом було визначено основні параметри, функції витрат, залежності для розрахунку оптимального розміру замовлення, очікуваного рівня дефіциту та загальних витрат. На основі розглянутих моделей було розроблено алгоритми знаходження оптимальних значень параметрів системи управління запасами.

Програмна реалізація розроблених алгоритмів виконана мовою Python. Створена програма дозволяє здійснювати розрахунок оптимального розміру замовлення, прогнозування попиту, визначення рівня дефіциту, очікуваних витрат, а також виконувати аналіз чутливості для заданих значень параметрів. Для підвищення наочності результатів у програмі реалізовано побудову графіків і діаграм, що дає змогу аналізувати зміну попиту, рівня запасів, дефіциту та витрат залежно від параметрів моделі.

У роботі також було розроблено зручний інтерфейс користувача, який дозволяє вводити початкові параметри моделі, запускати розрахунок оптимального розміру замовлення та переглядати отримані результати.

Програма була протестована на реальних значеннях параметрів, наданих підприємством. За результатами тестування помилок і збоїв у роботі програми не виявлено, що підтверджує коректність реалізації алгоритмів.

Отримані результати мають практичне значення, оскільки можуть бути використані підприємствами для підвищення ефективності процесів управління запасами. Застосування розроблених алгоритмів дозволяє зменшити витрати на зберігання продукції, скоротити ризик виникнення дефіциту, точніше прогнозувати попит і покращити планування закупівель та виробничих процесів.

Таким чином, мету роботи було досягнуто. У роботі розроблено алгоритм та програмну реалізацію для розв’язання стохастичних задач управління запасами з урахуванням випадкового характеру попиту та різних сценаріїв дефіциту. Запропонований підхід може бути використаний як практичний інструмент підтримки прийняття рішень у сфері логістики, складського обліку та управління запасами.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Winston W.L. Introduction to Mathematical Programming: Applications and Algorithms. Boston (Mass.): PWS–KENT Publ., 1991. 794 p.

2. Progress in inventory research. //Proc. of the 4-th Internat. Symp Budapest: Akad. Kiado, 1989. 446 p.

3. Winston W.L. Operations Research: Applications and Algorithms Boston (Mass.): PWS–KENT Publ., 1990. 1434p.

4. Benati, S. A. Mixed integer linear programming formulation of the optimal of the optimal mean / *European Journal of Operational Research*. 2007. Vol. 176, № 1. pp.423–434

5. Jaworski M., Ziadé T. Expert Python Programming Packt Publishing. 2021. 630 p.

6. Scarf H. The Optimality of (S, s) Policies in the Dynamic Inventory Problem. *Mathematical Methods in the Social Sciences*. Stanford : Stanford University Press, 1960.

7. Silver E. A., Pyke D. F., Peterson R. *Inventory Management and Production Planning and Scheduling*. 3rd ed. New York : John Wiley & Sons, 1998. Porteus E. L. *Foundations of Stochastic Inventory Theory*. Stanford : Stanford University Press, 2002. 299 p.

8. Цеслів О. В., Гришко М. П. Стохастичні моделі управління запасами поставок товару. *Економічний вісник НТУУ «КПІ»*. 2019. № 16.

9. Старух М. В. Математичне моделювання управління запасами на підприємствах України. 2011. С. 120–121.

10. Харченко Ю. А., Михайленко А. С. Економіко-математичне моделювання рівня запасів товарів торговельного підприємства. *Економічний простір*. 2018. № 136. С. 202–221.

11. Вареник В. М., Рєзцова М. І. Управління запасами підприємства: теоретичні та практичні аспекти. *Європейський вектор економічного розвитку*. 2018. № 1(24). С. 5–16. DOI: 10.32342/2074-5362-2018-24-1.

12. Тюріна Н. М., Гой І. В., Бабій І. В. Логістика: навчальний посібник. Київ : Центр учбової літератури, 2015. 392 с.

13. Nahmias S., Olsen T. L. *Production and Operations Analysis*. 7th ed. Long Grove : Waveland Press, 2015.

14. Law A. M. *Simulation Modeling and Analysis*. 5th ed. New York : McGraw-Hill Education, 2015.

15. Застосування методів ABC- та XYZ-аналізу в системі бюджетних закупівель лікарських засобів / SciUp URL: https://sciup.org/140104743 (дата звернення 16.04.2026).

16. SciPy documentation. URL: https://docs.scipy.org/doc/scipy/ (дата звернення 10.05.2026).

17. Glasserman P. *Monte Carlo Methods in Financial Engineering*. New York : Springer, 2004.

18. Python Software Foundation. *Python Documentation*. URL: https://docs.python.org/3/ (дата звернення: 12.05.2026).

19. NumPy Developers. *NumPy Documentation*. URL: https://numpy.org/doc/ (дата звернення: 16.04.2026).

20. Matplotlib Developers. *matplotlib.pyplot - Matplotlib Documentation*. URL: https://matplotlib.org/stable/api/pyplot_summary.html (дата звернення: 24.04.2026).

21. ДСТУ 3008:2015. Інформація та документація. Звіти у сфері науки і техніки. Структура і правила оформлення. Київ, ДП «УкрННЦ», 2015. 26с. (Інформація та документація).

22. ДСТУ 8302:2015. Бібліографічне посилання. Загальні вимоги та правила складання Київ, ДП «УкрННЦ», 2016. 16 с. (Інформація та документація).

23. ДСТУ 3582:2013. Бібліографічний опис. Скорочення слів і словосполучень в українській мові. Загальні вимоги та правила. Київ, ДП «УкрННЦ», 2013. 23 с. (Інформація та документація).

ДОДАТОК А

Лістинг коду моделі з відкладеним попитом

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm
from statsmodels.tsa.arima.model import ARIMA
import tkinter as tk
from tkinter import ttk, messagebox
class DelayedDemandModel:
    def __init__(self, d, K, h, b, mu, historical_data=None):
        self.d = d
        self.K = K
        self.h = h
        self.b = b
        self.mu = mu
        self.historical_data = historical_data if historical_data is not None else []
    def forecast_demand(self):
        if self.historical_data is not None and len(self.historical_data) > 5:
            try:
                model = ARIMA(self.historical_data, order=(1, 1, 1))
                model_fit = model.fit()
                forecast = model_fit.forecast(steps=12)
                self.forecasted_demand = np.array(forecast)
            except Exception:
                self.forecasted_demand = np.full(12, self.mu)
        else:
            self.forecasted_demand = np.full(12, self.mu)
    def calculate_optimal_order(self):
        if self.h <= 0 or (self.h + self.b) <= 0:
            raise ValueError("Некоректні параметри витрат")
        Q = np.sqrt((2 * self.d * self.K) / (self.h + self.b))
        r = self.mu - Q / 2
        B = 0
        if self.mu != 0:
            B = (self.mu - r) * (1 - (r / self.mu))
        C = (self.K * self.d / Q) + (self.h * Q / 2) + (self.b * B)
        self.Q_opt = Q
        self.B = B
        self.r_opt = r
        self.C = C
```

```
def display_results(self):
    if not hasattr(self, "Q_opt"):
        return "Спочатку виконайте розрахунок"
    return (
        f"Оптимальний розмір замовлення (Q): {self.Q_opt:.2f}\n"
        f"Очікуваний рівень дефіциту (B): {self.B:.2f}\n"
        f"Очікувані витрати (C): {self.C:.2f}"
    )
def plot_inventory_levels(self):
    if not hasattr(self, "Q_opt"):
        return
    t = np.linspace(0, 10, 100)
    inventory = self.r_opt + self.Q_opt * (1 - (t % 1))
    plt.figure()
    plt.plot(t, inventory)
    plt.title("Рівень запасів")
    plt.xlabel("Час")
    plt.ylabel("Запаси")
    plt.grid()
    plt.show()
def plot_forecasted_demand(self):
    if not hasattr(self, "forecasted_demand"):
        return
    plt.figure()
    if self.historical_data is not None and len(self.historical_data) > 0:
        plt.plot(self.historical_data, label="Історичні дані")
    hist_len = len(self.historical_data) if self.historical_data is not None else 0
    future_x = np.arange(hist_len, hist_len + len(self.forecasted_demand))
    plt.plot(future_x, self.forecasted_demand, label="Прогноз")
    plt.title("Прогноз попиту")
    plt.xlabel("Час")
    plt.ylabel("Попит")
    plt.legend()
    plt.grid()
    plt.show()
def sensitivity_analysis(self, param_range, param_name):
    if not hasattr(self, param_name):
        raise ValueError("Невірний параметр")
    results = []
    original = getattr(self, param_name)
    try:
```

```
            for val in param_range:
                setattr(self, param_name, val)
                self.calculate_optimal_order()
                results.append((val, self.Q_opt, self.r_opt, self.C))
        finally:
            setattr(self, param_name, original)
        return results
    def plot_sensitivity_analysis(self, results, param_name):
        if not results:
            return
        vals, Qs, rs, Cs = map(np.array, zip(*results))
        fig, axes = plt.subplots(3, 1, figsize=(8, 6))
        axes[0].plot(vals, Qs)
        axes[0].set_ylabel("Q")
        axes[1].plot(vals, rs)
        axes[1].set_ylabel("r")
        axes[2].plot(vals, Cs)
        axes[2].set_ylabel("C")
        axes[2].set_xlabel(param_name)
        for ax in axes:
            ax.grid()
        plt.tight_layout()
        plt.show()
class App(tk.Tk):
    def __init__(self):
        super().__init__()
        self.title("Delayed Demand Model")
        self.geometry("420x400")
        self.create_widgets()
        self.model = None
    def create_widgets(self):
        labels = [
            "Інтенсивність попиту (d)",
            "Витрати на замовлення (K)",
            "Витрати зберігання (h)",
            "Втрати від дефіциту (b)",
            "Середній попит (μ)"
        ]
        self.entries = []
        for i, label in enumerate(labels):
```

```
                ttk.Label(self, text=label).grid(row=i, column=0, padx=10, pady=5,
sticky="w")
                entry = ttk.Entry(self, width=25)
                entry.grid(row=i, column=1, padx=10, pady=5, sticky="ew")
                self.entries.append(entry)
            ttk.Button(self, text="Запустити модель",
                    command=self.run_model).grid(row=len(labels), column=0,
                                                 columnspan=2, pady=10)
            self.result_box = tk.Text(self, height=10)
            self.result_box.grid(row=len(labels) + 1, column=0, columnspan=2, padx=10,
pady=10)
            self.grid_columnconfigure(0, weight=1)
            self.grid_columnconfigure(1, weight=1)
        def run_model(self):
            try:
                d = float(self.entries[0].get())
                K = float(self.entries[1].get())
                h = float(self.entries[2].get())
                b = float(self.entries[3].get())
                mu = float(self.entries[4].get())
                historical = np.random.poisson(mu, 50)
                self.model = DelayedDemandModel(d, K, h, b, mu, historical)
                self.model.forecast_demand()
                self.model.calculate_optimal_order()
                self.result_box.delete(1.0, tk.END)
                self.result_box.insert(tk.END, self.model.display_results())
                self.model.plot_inventory_levels()
                self.model.plot_forecasted_demand()
                param_range = np.linspace(mu * 0.5, mu * 1.5, 10)
                res = self.model.sensitivity_analysis(param_range, "mu")
                self.model.plot_sensitivity_analysis(res, "μ")
            except Exception as e:
                messagebox.showerror("Помилка", str(e))
    if __name__ == "__main__":
        app = App()
        app.mainloop()
```

ДОДАТОК Б

Лістинг коду реалізації моделі з втраченим попитом

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm
from statsmodels.tsa.arima.model import ARIMA
import tkinter as tk
from tkinter import ttk, messagebox
class DelayedDemandModel:
    def __init__(self, d, K, h, b, mu, historical_data=None):
        self.d = d
        self.K = K
        self.h = h
        self.b = b
        self.mu = mu
        self.historical_data = historical_data if historical_data is not None else []
    def forecast_demand(self):
        if self.historical_data is not None and len(self.historical_data) > 5:
            try:
                model = ARIMA(self.historical_data, order=(1, 1, 1))
                model_fit = model.fit()
                forecast = model_fit.forecast(steps=12)
                self.forecasted_demand = np.array(forecast)
            except Exception:
                self.forecasted_demand = np.full(12, self.mu)
        else:
            self.forecasted_demand = np.full(12, self.mu)
    def calculate_optimal_order(self):
        if self.h <= 0 or (self.h + self.b) <= 0:
            raise ValueError("Некоректні параметри витрат")
        Q = np.sqrt((2 * self.d * self.K) / (self.h + self.b))
        r = self.mu - Q / 2
        B = 0
        if self.mu != 0:
            B = (self.mu - r) * (1 - (r / self.mu))
        C = (self.K * self.d / Q) + (self.h * Q / 2) + (self.b * B)
        self.Q_opt = Q
        self.B = B
        self.r_opt = r
        self.C = C
    def display_results(self):
```

```
    if not hasattr(self, "Q_opt"):
        return "Спочатку виконайте розрахунок"
    return (
        f"Оптимальний розмір замовлення (Q): {self.Q_opt:.2f}\n"
        f"Очікуваний рівень дефіциту (B): {self.B:.2f}\n"
        f"Очікувані витрати (C): {self.C:.2f}"
    )
def plot_inventory_levels(self):
    if not hasattr(self, "Q_opt"):
        return
    t = np.linspace(0, 10, 100)
    inventory = self.r_opt + self.Q_opt * (1 - (t % 1))
    plt.figure()
    plt.plot(t, inventory)
    plt.title("Рівень запасів")
    plt.xlabel("Час")
    plt.ylabel("Запаси")
    plt.grid()
    plt.show()
def plot_forecasted_demand(self):
    if not hasattr(self, "forecasted_demand"):
        return
    plt.figure()
    if self.historical_data is not None and len(self.historical_data) > 0:
        plt.plot(self.historical_data, label="Історичні дані")
    hist_len = len(self.historical_data) if self.historical_data is not None else 0
    future_x = np.arange(hist_len, hist_len + len(self.forecasted_demand))
    plt.plot(future_x, self.forecasted_demand, label="Прогноз")
    plt.title("Прогноз попиту")
    plt.xlabel("Час")
    plt.ylabel("Попит")
    plt.legend()
    plt.grid()
    plt.show()
def sensitivity_analysis(self, param_range, param_name):
    if not hasattr(self, param_name):
        raise ValueError("Невірний параметр")
    results = []
    original = getattr(self, param_name)
    try:
        for val in param_range:
```

```
                setattr(self, param_name, val)
                self.calculate_optimal_order()
                results.append((val, self.Q_opt, self.r_opt, self.C))
            finally:
                setattr(self, param_name, original)
            return results
        def plot_sensitivity_analysis(self, results, param_name):
            if not results:
                return
            vals, Qs, rs, Cs = map(np.array, zip(*results))
            fig, axes = plt.subplots(3, 1, figsize=(8, 6))
            axes[0].plot(vals, Qs)
            axes[0].set_ylabel("Q")
            axes[1].plot(vals, rs)
            axes[1].set_ylabel("r")
            axes[2].plot(vals, Cs)
            axes[2].set_ylabel("C")
            axes[2].set_xlabel(param_name)
            for ax in axes:
                ax.grid()
            plt.tight_layout()
            plt.show()
    class App(tk.Tk):
        def __init__(self):
            super().__init__()
            self.title("Delayed Demand Model")
            self.geometry("420x400")
            self.create_widgets()
            self.model = None
        def create_widgets(self):
            labels = [
                "Інтенсивність попиту (d)",
                "Витрати на замовлення (K)",
                "Витрати зберігання (h)",
                "Втрати від дефіциту (b)",
                "Середній попит (μ)"
            ]
            self.entries = []
            for i, label in enumerate(labels):
                ttk.Label(self, text=label).grid(row=i, column=0, padx=10, pady=5,
sticky="w")
```

```
                entry = ttk.Entry(self, width=25)
                entry.grid(row=i, column=1, padx=10, pady=5, sticky="ew")
                self.entries.append(entry)
            ttk.Button(self, text="Запустити модель",
                    command=self.run_model).grid(row=len(labels), column=0,
                                                 columnspan=2, pady=10)
            self.result_box = tk.Text(self, height=10)
            self.result_box.grid(row=len(labels) + 1, column=0, columnspan=2, padx=10,
pady=10)
            self.grid_columnconfigure(0, weight=1)
            self.grid_columnconfigure(1, weight=1)
        def run_model(self):
            try:
                d = float(self.entries[0].get())
                K = float(self.entries[1].get())
                h = float(self.entries[2].get())
                b = float(self.entries[3].get())
                mu = float(self.entries[4].get())
                historical = np.random.poisson(mu, 50)
                self.model = DelayedDemandModel(d, K, h, b, mu, historical)
                self.model.forecast_demand()
                self.model.calculate_optimal_order()
                self.result_box.delete(1.0, tk.END)
                self.result_box.insert(tk.END, self.model.display_results())
                self.model.plot_inventory_levels()
                self.model.plot_forecasted_demand()
                param_range = np.linspace(mu * 0.5, mu * 1.5, 10)
                res = self.model.sensitivity_analysis(param_range, "mu")
                self.model.plot_sensitivity_analysis(res, "μ")
            except Exception as e:
                messagebox.showerror("Помилка", str(e))
    if __name__ == "__main__":
        app = App()
        app.mainloop()
```