Міністерство освіти і науки України
Міжнародний гуманітарний університет
Кафедра комп’ютерних наук

Д.М. Розенвассер

МОДЕЛЮВАННЯ СИСТЕМ

Методичні вказівки до практичних робіт
для здобувачів вищої освіти,
які навчаються за спеціальностями
галузі «Інформаційні технології» та за спеціальністю
«Електронні комунікації та радіотехніка»

Одеса 2025

Затверджено Вченою Радою Міжнародного гуманітарного університету.
Протокол № 5 від 20 січня 2025 р.

Розенвассер Д.М. Моделювання систем. Методичні вказівки до практичних робіт для здобувачів вищої освіти, які навчаються за спеціальностями галузі «Інформаційні технології» та за спеціальністю «Електронні комунікації та радіотехніка» [Електронне видання]. Кафедра комп'ютерних наук Міжнародного гуманітарного університету. Одеса, 2025. 42 с.

Дисципліна «Моделювання систем» формує у здобувачів необхідний обсяг теоретичних і практичних знань про системне моделювання та галузь його застосування в системному інжинірингу, методи концептуального проектування та функціональний аналіз технічної системи, формує знання з архітектурного моделювання системи та її синтезу.

При вивченні дисципліни особливу увагу приділено теоретичному та експериментальному дослідженню, розробці та використанню математичних моделей систем і процесів, математичних методів, а також отриманню знань, основних понять, положень та особливостей математичного моделювання, засвоєння теоретичних знань і формуванню практичних навичок з основ моделювання систем та мереж.

ЗМІСТ

ТЕМА 1. НАУКА МОДЕЛЮВАННЯ

Практична робота 1. Виконання арифметичних операцій над матрицями та векторами в GNU Octave / Scilab

Мета роботи – ознайомитися з основними можливостями GNU Octave та Scilab щодо створення та обробки матриць і векторів, навчитися виконувати арифметичні операції, транспонування та множення матриць і векторів.

Ключові положення

Матриці та вектори є базовими структурами даних у математичному та комп'ютерному моделюванні систем. Вони дозволяють компактно описувати лінійні перетворення, системи рівнянь, моделі мереж та сигнали в телекомунікаціях.

У середовищах GNU Octave та Scilab матриця створюється за допомогою квадратних дужок, елементи рядка розділяються пробілами або комами, а рядки – крапкою з комою. Основні арифметичні операції включають додавання, віднімання, матричне множення, поелементне множення, ділення, транспонування та піднесення до степеня.

Важливо розрізняти матричне множення (*) та поелементне множення (.*). Аналогічно для ділення (/ та ./). Транспонування матриці виконується оператором апострофа ('). Для розв'язування систем лінійних рівнянь ефективно використовується оператор \ (A\b).

Правильне володіння операціями над матрицями є фундаментом для подальшого моделювання динамічних систем, оптимізації та аналізу телекомунікаційних мереж.

Практичне завдання

1. Створити матриці та вектори різних розмірностей (2×2, 3×3, 4×2 тощо).
2. Виконати операції додавання та віднімання матриць.
3. Виконати матричне множення та поелементне множення матриць.
4. Виконати транспонування матриць та векторів.
5. Обчислити визначник квадратної матриці.
6. Знайти обернену матрицю (якщо можливо).
7. Виконати множення матриці на вектор.
8. Порівняти результати обчислень з ручними розрахунками.
9. Виконати операції з використанням вбудованих функцій (size, length, det, inv).
10. Проаналізувати випадки, коли операції неможливі (несумісні розміри матриць).

Методичні вказівки до виконання практичного завдання

Перед початком роботи необхідно ознайомитися з правилами створення матриць і векторів у GNU Octave та Scilab. Рекомендується виконувати роботу в режимі скрипта (файл з розширенням .m для Octave або .sce для Scilab).

Створюйте матриці за допомогою конструкції [ ]. Для введення елементів використовуйте пробіли між елементами рядка та крапку з комою для переходу на новий рядок.

Виконуючи матричні операції, звертайте увагу на розмірність операндів. Матричне множення можливе лише тоді, коли кількість стовпців першої матриці дорівнює кількості рядків другої. Для поелементних операцій розміри матриць повинні співпадати.

Використовуйте команди disp() або printf() для виведення результатів. Для перевірки розмірності матриці використовуйте функцію size().

Рекомендується створити один скрипт, у якому послідовно виконуються всі завдання, з коментарями перед кожним блоком. Після виконання кожного пункту фіксуйте результати у звіті (скріншоти консолі або копіювання виведення).

Особливу увагу приділіть аналізу помилок, які виникають при спробі виконання операцій з матрицями несумісних розмірів.

Контрольні питання

1. Як створюються матриці та вектори в GNU Octave / Scilab?
2. У чому полягає різниця між операторами * та .*?
3. Які умови повинні виконуватися для матричного множення двох матриць?
4. Як виконується транспонування матриці?
5. Для чого використовується оператор \ при роботі з матрицями?
6. Як обчислити визначник матриці?
7. У яких випадках матриця не має оберненої?
8. Які вбудовані функції використовуються для роботи з матрицями?
9. Як перевірити розмірність матриці?
10. Чому операції над матрицями є важливими в моделюванні систем?

Зміст звіту

– назва практичної роботи;
– мета виконання;
– теоретичні основи роботи з матрицями та векторами;
– лістинг програми (скрипту) з коментарями;
– результати виконання всіх операцій;
– скріншоти або копії виведення програми;
– порівняння результатів з ручними розрахунками;
– аналіз помилок та особливих випадків;
– висновки.

Практична робота 2. Розв’язування систем лінійних алгебраїчних рівнянь методом Гаусса та за допомогою вбудованих функцій (GNU Octave / Scilab)

Мета роботи – опанувати метод Гаусса для розв’язування систем лінійних алгебраїчних рівнянь (СЛАР), навчитися реалізовувати його вручну та порівняти результати з розв’язанням за допомогою вбудованих функцій GNU Octave / Scilab.

Ключові положення

Системи лінійних алгебраїчних рівнянь широко застосовуються при моделюванні технічних систем, електричних кіл, телекомунікаційних мереж, задач балансу та оптимізації. Метод Гаусса є одним з найпоширеніших прямих методів розв’язування СЛАР. Він полягає у приведенні розширеної матриці системи до верхнього трикутного вигляду (прямий хід) з подальшим зворотним ходом для знаходження значень невідомих.

У середовищах GNU Octave та Scilab для розв’язування СЛАР рекомендується використовувати оператор \ (A\b), який реалізує ефективніші чисельні методи (LU-розклад або QR-розклад). Це значно швидше та точніше, ніж класичний метод Гаусса, особливо для великих систем. Однак розуміння алгоритму Гаусса необхідне для глибокого розуміння чисельних методів і аналізу стійкості розв’язків.

Важливими поняттями є визначеність системи, умовна стійкість розв’язку та накопичення похибки округлення при виконанні обчислень.

Практичне завдання

1. Скласти розширену матрицю системи лінійних рівнянь.
2. Реалізувати прямий хід методу Гаусса (приведення до верхнього трикутного вигляду).

3. Реалізувати зворотний хід для знаходження розв’язку.
4. Розв’язати систему за допомогою вбудованої функції (оператор \).
5. Порівняти результати, отримані двома способами.
6. Розв’язати систему з нульовим визначником (вироджену) та проаналізувати результат.
7. Розв’язати систему з нескінченно багатьма розв’язками.
8. Виконати розв’язування системи з використанням функції rref() (reduced row echelon form).
9. Оцінити точність розв’язку та вплив похибки округлення.
10. Проаналізувати стійкість методу Гаусса до малих змін у коефіцієнтах.

Методичні вказівки до виконання практичного завдання

Перед початком роботи ознайомтеся з алгоритмом методу Гаусса: виключення невідомих шляхом елементарних перетворень рядків. Рекомендується реалізувати прямий і зворотний ходи у вигляді окремих функцій або блоків скрипта.

У GNU Octave / Scilab для часткового вибору головного елемента можна використовувати вбудовані функції, але для навчальних цілей краще реалізувати простий варіант без нього. Використовуйте циклічні конструкції (for, while) для обробки рядків і стовпців матриці.

Для порівняння результатів розв’язуйте одну й ту саму систему двома способами: вручну методом Гаусса та за допомогою оператора A\b. Обов’язково перевірте правильність розв’язку шляхом підстановки отриманих значень назад у вихідні рівняння.

Для вироджених систем проаналізуйте ранг матриці та кількість розв’язків. Використовуйте функцію rank() для визначення рангу матриці коефіцієнтів.

Усі проміжні матриці після кожного кроку прямого ходу рекомендується виводити на екран за допомогою disp() для наочності.

Контрольні питання

1. У чому полягає сутність методу Гаусса?
2. Які етапи включає метод Гаусса?
3. Що таке прямий і зворотний хід?
4. Для чого використовується оператор \ в Octave / Scilab?
5. Як перевірити правильність отриманого розв’язку?
6. Що відбувається з системою, якщо її визначник дорівнює нулю?
7. Як впливає вибір головного елемента на стійкість методу?
8. Які переваги вбудованих функцій порівняно з класичним методом Гаусса?
9. Що таке ранг матриці і як він визначається?
10. Чому метод Гаусса важливий для моделювання систем?

Зміст звіту

– назва практичної роботи;
– мета виконання;
– теоретичні основи методу Гаусса;
– лістинг програми з реалізацією методу Гаусса;
– результати розв’язування систем двома способами;
– проміжні матриці після кроків прямого ходу;
– порівняльний аналіз отриманих результатів;
– аналіз вироджених систем;
– висновки щодо точності та ефективності методів.

Практична робота 3. Символьне та чисельне обчислення похідних функцій в GNU Octave / Scilab

Мета роботи – навчитися обчислювати похідні функцій символьно та чисельно в GNU Octave / Scilab, порівняти результати обох методів та проаналізувати їхню точність.

Ключові положення

Похідна функції є одним з найважливіших математичних інструментів у моделюванні систем. Вона характеризує швидкість зміни функції, використовується для аналізу стійкості, оптимізації, дослідження динамічних систем та знаходження екстремумів.

У середовищах GNU Octave та Scilab існують два основних підходи до обчислення похідних:

– символьне диференціювання – отримання точного аналітичного виразу похідної;

– чисельне диференціювання – наближене обчислення похідної за допомогою формул скінченних різниць.

Символьні обчислення в Scilab виконуються за допомогою пакету Symbolic Math Toolbox (функція diff), в GNU Octave – через пакет symbolic. Чисельне диференціювання реалізується вручну або за допомогою вбудованих функцій (gradient, diff).

Порівняння символьного та чисельного методів дозволяє зрозуміти переваги й недоліки кожного з них, а також вплив кроку дискретизації на точність чисельних результатів. Ці навички необхідні при моделюванні динамічних процесів у телекомунікаційних системах.

Практичне завдання

1. Задати кілька тестових функцій різної складності (поліноміальні, тригонометричні, експоненціальні).
2. Обчислити першу та другу похідні функцій символьним методом.
3. Обчислити похідні тих самих функцій чисельним методом (методом скінченних різниць).
4. Побудувати графіки вихідної функції та її похідних.
5. Порівняти результати символьного та чисельного диференціювання.
6. Дослідити вплив величини кроку h на точність чисельного диференціювання.
7. Обчислити похідну складеної функції.
8. Знайти екстремуми функції за допомогою похідної.
9. Проаналізувати випадки, коли чисельне диференціювання дає значну похибку.
10. Виконати символьне та чисельне диференціювання для функції, що моделює сигнал у телекомунікаційній системі.

Методичні вказівки до виконання практичного завдання

Перед початком роботи встановіть та підключіть пакет символічних обчислень (у Scilab – atomsInstall("symbolic"), у Octave – pkg install symbolic). Для символьних обчислень використовуйте функцію diff() після оголошення символьної змінної (syms x).

Для чисельного диференціювання реалізуйте формулу центральної різниці: $f'(x) \approx [f(x+h) – f(x–h)] / (2h)$

Створіть скрипт, у якому чітко розділені блоки символьного та чисельного обчислення. Використовуйте функцію plot() для побудови графіків вихідної функції та її похідних на одному рисунку (з легендою).

Обов’язково варіюйте крок h у широкому діапазоні (від 1e-1 до 1e-6) і проаналізуйте, як змінюється похибка. Для порівняння результатів виводьте аналітичний вираз похідної та чисельні значення в одній таблиці.

Особливу увагу приділіть вибору тестових функцій: починайте з простих поліномів і поступово переходьте до більш складних (sin, exp, ln).

Контрольні питання

1. У чому полягає різниця між символьним та чисельним диференціюванням?
2. Які переваги символьного обчислення похідних?
3. Які недоліки чисельного диференціювання?
4. Як впливає величина кроку h на точність чисельної похідної?
5. Які формули скінченних різниць ви знаєте?
6. Як у GNU Octave / Scilab виконується символьне диференціювання?
7. Для чого використовується друга похідна функції?
8. Як знайти точки екстремуму за допомогою похідної?
9. У яких задачах моделювання систем похідні відіграють ключову роль?
10. Коли доцільніше використовувати чисельне, а коли символьне диференціювання?

Зміст звіту

– назва практичної роботи;
– мета виконання;
– теоретичні основи символьного та чисельного диференціювання;
– лістинг програми з коментарями;
– аналітичні вирази похідних;
– графіки функцій та їх похідних;
– таблиці порівняння символьних і чисельних результатів;
– аналіз впливу кроку h на точність;
– висновки щодо переваг і недоліків обох методів.

Практична робота 4. Обчислення границь функцій та символьне розв'язування квадратних рівнянь (GNU Octave / Scilab)

Мета роботи – навчитися обчислювати границі функцій символьним методом, опанувати символьне розв'язування квадратних рівнянь та проаналізувати поведінку функцій у точках розриву та невизначеності.

Ключові положення

Границя функції є фундаментальним поняттям математичного аналізу і широко використовується в моделюванні систем для дослідження поведінки функцій у критичних точках, при великих значеннях аргументу, а також для аналізу стійкості та асимптотичної поведінки моделей.

У середовищах GNU Octave та Scilab символьне обчислення границь виконується за допомогою пакету символічних обчислень (функція limit). Границі дозволяють визначати поведінку функції при наближенні аргументу до певної точки або до нескінченності, виявляти розриви, вертикальні та горизонтальні асимптоти.

Символьне розв'язування квадратних рівнянь дає точні аналітичні розв'язки (корені) без чисельних похибок. Це особливо важливо при подальшому моделюванні, коли потрібна висока точність або подальший аналіз виразу (наприклад, для оптимізації чи дослідження динамічних систем).

Поєднання обчислення границь і символьного розв'язування рівнянь дозволяє глибше зрозуміти властивості функцій, які описують реальні процеси в телекомунікаційних та технічних системах.

Практичне завдання

1. Обчислити границі простих функцій у скінченних точках (включаючи односторонні границі).

2. Обчислити границі функцій при $x \to \pm\infty$.
3. Дослідити границі функцій, що мають невизначеність типу 0/0 або ∞/∞.
4. Використовуючи правило Лопіталя (символьно), обчислити границі складних виразів.
5. Символьно розв’язати кілька квадратних рівнянь різної складності.
6. Знайти корені квадратного рівняння та проаналізувати їх характер (дійсні, кратні, комплексні).
7. Побудувати графіки функцій та позначити на них точки, пов’язані з обчисленими границями.
8. Порівняти результати символьного обчислення границь з чисельними значеннями.
9. Дослідити поведінку квадратичної функції біля точок її коренів.
10. Застосувати отримані навички до функції, що моделює характеристику телекомунікаційного сигналу або системи.

Методичні вказівки до виконання практичного завдання

Перед початком роботи підключіть пакет символічних обчислень (symbolic). Для обчислення границь використовуйте функцію limit(f, x, a), де f – функція, x – змінна, a – точка (можна вказувати inf або -inf). Для односторонніх границь використовуйте параметр 'left' або 'right'.

Для розв’язування квадратних рівнянь використовуйте функцію solve() пакету symbolic. Оголошуйте символьні змінні за допомогою syms x.

Рекомендується створити окремий скрипт, у якому чітко розділити розділи «Обчислення границь» та «Розв’язування квадратних рівнянь». Для наочності будуйте графіки функцій за допомогою plot() та позначайте асимптоти і корені.

Порівнюйте символьні результати з чисельним наближенням (підставляйте значення x, близькі до точки границі). Обов’язково фіксуйте випадки, коли границя не існує або дорівнює нескінченності.

Контрольні питання

1. Що таке границя функції?
2. Які типи невизначеностей виникають при обчисленні границь?
3. Як у GNU Octave / Scilab обчислюється границя функції?
4. У чому полягає правило Лопіталя?
5. Як обчислити односторонню границю?
6. Як символьно розв’язати квадратне рівняння?
7. Які види коренів може мати квадратне рівняння?
8. Для чого потрібно обчислювати границі при моделюванні систем?
9. Як графіки допомагають зрозуміти поведінку границь?
10. У яких задачах телекомунікаційного моделювання використовуються границі функцій?

Зміст звіту

– назва практичної роботи;
– мета виконання;
– теоретичні основи обчислення границь та розв’язування квадратних рівнянь;
– лістинг програми з коментарями;
– результати обчислення границь (у тому числі односторонніх та при нескінченності);
– символьні розв’язки квадратних рівнянь та аналіз коренів;
– графіки функцій з позначеними асимптотами та коренями;
– порівняння символьних і чисельних результатів;
– висновки щодо застосування отриманих методів у моделюванні систем.

ТЕМА 2. КОМП'ЮТЕРНЕ МОДЕЛЮВАННЯ

Практична робота 5. Реалізація рекурсивних та ітеративних функцій з використанням циклів for та while в GNU Octave / Scilab

Мета роботи – опанувати реалізацію рекурсивних та ітеративних (циклічних) функцій в GNU Octave / Scilab, навчитися порівнювати їх ефективність, глибинну рекурсії та можливі проблеми переповнення стеку.

Ключові положення

Рекурсія – це метод, при якому функція викликає сама себе для розв'язання задачі меншого розміру. Ітерація – це повторення певного блоку коду за допомогою циклів (for або while). Обидва підходи широко застосовуються в моделюванні систем, обчисленні комбінаторних величин, обробці сигналів та реалізації алгоритмів.

Класичним прикладом для порівняння рекурсії та ітерації є обчислення факторіалу числа n (n!). Рекурсивний варіант є природним з математичної точки зору, але потребує значної пам'яті стеку та може призвести до помилки «stack overflow» при великих значеннях n. Ітеративний варіант економніший за ресурсами та зазвичай швидший.

У GNU Octave та Scilab рекурсія реалізується шляхом виклику функції всередині самої себе, а ітерація – за допомогою циклів for (з відомою кількістю ітерацій) та while (з умовою продовження). Важливо розуміти переваги та недоліки кожного підходу, а також межі їх практичного застосування в задачах моделювання.

Практичне завдання

1. Реалізувати функцію обчислення факторіалу рекурсивним методом.
2. Реалізувати функцію обчислення факторіалу ітеративним методом за допомогою циклу for.
3. Реалізувати функцію обчислення факторіалу ітеративним методом за допомогою циклу while.
4. Порівняти результати роботи всіх трьох реалізацій для невеликих значень n.
5. Дослідити максимальну глибину рекурсії (знайти найбільше n, при якому рекурсія ще працює).
6. Порівняти час виконання рекурсивної та ітеративних функцій для великих n.
7. Реалізувати функцію обчислення чисел Фібоначчі рекурсивним та ітеративним способами.
8. Проаналізувати ефективність рекурсії та ітерації при зростанні n.
9. Обробити помилки переповнення стеку та надто великих значень факторіалу.
10. Застосувати ітеративний підхід для обчислення факторіалу в контексті моделювання комбінаторних задач телекомунікацій.

Методичні вказівки до виконання практичного завдання

Перед початком роботи створіть окремий скрипт-файл. Реалізуйте кожну версію функції (рекурсивну, for, while) як окрему користувацьку функцію.

Для рекурсії обов'язково визначте базовий випадок (n = 0 або n = 1), інакше виникне нескінченна рекурсія. Для ітеративних варіантів використовуйте змінну-акумулятор для накопичення результату.

Використовуйте функцію tic / toc (або clock) для вимірювання часу виконання. Для визначення максимальної глибини рекурсії поступово збільшуйте n до появи помилки.

Рекомендується виводити результати у вигляді таблиці: значення n, результат рекурсії, результат ітерації for, результат ітерації while, час виконання.

Особливу увагу зверніть на обробку великих чисел – у Octave/Scilab факторіал швидко виходить за межі стандартних типів даних, тому використовуйте вбудовані можливості роботи з великими цілими числами або переходьте до подвійної точності.

Контрольні питання

1. У чому полягає принцип рекурсії?
2. Які переваги та недоліки рекурсивних функцій порівняно з ітеративними?
3. Що таке базовий випадок у рекурсії?
4. У чому різниця між циклами for та while?
5. Коли краще використовувати рекурсію, а коли ітерацію?
6. Що таке переповнення стеку (stack overflow)?
7. Як виміряти час виконання функції в GNU Octave / Scilab?
8. Чому факторіал великих чисел викликає проблеми?
9. Для яких задач моделювання систем корисні рекурсивні алгоритми?
10. Як можна оптимізувати рекурсивні функції?

Зміст звіту

– назва практичної роботи;
– мета виконання;
– теоретичні основи рекурсії та ітерації;
– лістинг програм рекурсивної та ітеративних функцій з коментарями;
– таблиця порівняння результатів для різних n;
– графіки часу виконання залежно від n;
– аналіз максимальної глибини рекурсії;
– висновки щодо ефективності рекурсивного та ітеративного підходів.

Практична робота 6. Розробка програми з діалоговим введенням/виведенням: введення коефіцієнтів квадратного рівняння та виведення коренів (GNU Octave / Scilab)

Мета роботи – навчитися створювати програми з діалоговим введенням/виведенням даних у GNU Octave / Scilab, реалізувати введення коефіцієнтів квадратного рівняння, обчислення та коректне виведення його коренів з урахуванням різних випадків.

Ключові положення

Діалогове введення/виведення є важливою складовою інтерактивних програм моделювання. Воно дозволяє користувачеві вводити параметри системи безпосередньо під час виконання програми, робить програмне забезпечення зручнішим та більш універсальним.

Квадратне рівняння $ax^2 + bx + c = 0$ є класичною математичною моделлю, розв’язок якої залежить від дискримінанту $D = b^2 – 4ac$. Залежно від значення дискримінанту можливі три випадки: два дійсні корені, один дійсний корінь (кратний) або два комплексні корені.

У GNU Octave та Scilab для діалогового введення використовуються функції input() (для чисел) та disp() або printf() для виведення результатів. Правильна обробка всіх можливих випадків ($D > 0$, $D = 0$, $D < 0$) є обов’язковою для коректної роботи програми. Також важливо передбачити захист від введення некоректних даних (наприклад, $a = 0$).

Навички створення діалогових програм необхідні для подальшого розроблення інтерактивних моделей систем та мереж телекомунікацій.

Практичне завдання

1. Розробити програму, яка запитує у користувача коефіцієнти a, b, c квадратного рівняння.

2. Реалізувати перевірку коректності введення ($a \neq 0$).
3. Обчислити дискримінант $D = b^2 - 4ac$.
4. Реалізувати розв’язування рівняння для трьох випадків: $D > 0$, $D = 0$, $D < 0$.
5. Вивести корені рівняння у зручному для користувача форматі.
6. Додати повідомлення про тип коренів (дійсні, кратні, комплексні).
7. Передбачити можливість повторного введення коефіцієнтів після отримання результату.
8. Реалізувати обробку помилок при введенні нечислових даних.
9. Вивести графік квадратичної функції з позначенням коренів (за бажанням).
10. Проаналізувати роботу програми при різних значеннях коефіцієнтів.

Методичні вказівки до виконання практичного завдання

Перед початком роботи сплануйте структуру програми: блок введення даних → перевірка коректності → обчислення дискримінанту → розгалуження за значеннями D → виведення результатів.

Використовуйте функцію input('Текст запиту: ') для введення коефіцієнтів. Для повторного запуску обчислень організуйте основний цикл while з можливістю виходу за бажанням користувача.

Для виведення результатів рекомендується використовувати disp() або форматований вивід printf(). При комплексних коренях виводьте дійсну та уявну частини окремо.

Обов’язково обробляйте ситуацію, коли a = 0 (рівняння стає лінійним). Використовуйте конструкції if-elseif-else для розгалуження за значенням дискримінанту.

Рекомендується створити окрему функцію для обчислення коренів квадратного рівняння, яку можна буде використовувати в подальших роботах.

Контрольні питання

1. Які функції використовуються для діалогового введення в GNU Octave / Scilab?
2. Як організувати повторне виконання програми за бажанням користувача?
3. Як обчислюється дискримінант квадратного рівняння?
4. Які три можливі випадки розв’язку квадратного рівняння існують?
5. Як правильно обробити ситуацію, коли a = 0?
6. У чому полягає перевага форматованого виведення (printf)?
7. Як вивести комплексні числа у зручному вигляді?
8. Які помилки можуть виникнути при введенні даних користувачем?
9. Для чого потрібна перевірка введених значень?
10. Де в задачах моделювання систем використовуються діалогові програми?

Зміст звіту

– назва практичної роботи;
– мета виконання;
– теоретичні основи роботи з квадратними рівняннями;
– блок-схема або структура програми;
– лістинг програми з коментарями;
– приклади роботи програми при різних значеннях коефіцієнтів (скріншоти);
– опис обробки особливих випадків;
– висновки щодо зручності та надійності діалогової програми.

Практична робота 7. Генерація псевдовипадкових чисел (rand, grand) з рівномірним та нормальним розподілом, побудова гістограми в GNU Octave / Scilab

Мета роботи – навчитися генерувати псевдовипадкові числа з рівномірним та нормальним розподілом у GNU Octave / Scilab, будувати гістограми розподілів та аналізувати їх статистичні характеристики.

Ключові положення

Псевдовипадкові числа широко застосовуються в імітаційному моделюванні систем, телекомунікаційних мереж, методі Монте-Карло, моделюванні шумів та випадкових процесів.

У GNU Octave та Scilab для генерації псевдовипадкових чисел використовуються такі функції:

- rand() – генерує числа з рівномірним розподілом на інтервалі [0, 1];
- randn() (Octave) або grand() (Scilab) – генерує числа з нормальним (гауссівським) розподілом (математичне сподівання = 0, дисперсія = 1).

Гістограма є графічним представленням розподілу даних і дозволяє візуально оцінити, наскільки згенерована послідовність відповідає теоретичному закону розподілу.

Важливими характеристиками є математичне сподівання, дисперсія, стандартне відхилення. Відтворюваність результатів забезпечується встановленням початкового значення генератора випадкових чисел (rand("seed", ...) або grand("setall", ...)).

Практичне завдання

1. Згенерувати 1000 та 10000 псевдовипадкових чисел з рівномірним розподілом на інтервалі [0, 1] за допомогою rand().
2. Згенерувати 1000 та 10000 чисел з нормальним розподілом за допомогою randn() (Octave) або grand(1, n, "nor", 0, 1) (Scilab).
3. Побудувати гістограми розподілів для обох випадків.
4. Обчислити та вивести основні статистичні характеристики (середнє значення, дисперсію, стандартне відхилення).
5. Налаштувати кількість інтервалів (bins) гістограми та дослідити вплив цього параметра на вигляд розподілу.
6. Встановити початкове значення генератора випадкових чисел та перевірити відтворюваність результатів.
7. Порівняти емпіричний розподіл з теоретичним нормальним розподілом.
8. Згенерувати числа з нормальним розподілом з іншими параметрами (заданим середнім та дисперсією).
9. Побудувати сумісний графік гістограми та теоретичної кривої щільності розподілу.
10. Проаналізувати результати для застосування в моделюванні телекомунікаційних систем (моделювання шумів, навантаження, затримок).

Методичні вказівки до виконання практичного завдання

Перед початком роботи ознайомтеся з синтаксисом функцій rand(), randn() та grand(). У Scilab для нормального розподілу рекомендується використовувати grand(n, "nor", m, sigma).

Для побудови гістограми використовуйте функцію hist() або histogram(). Налаштуйте кількість стовпчиків за допомогою параметра nbins або bins.

Для виведення статистичних характеристик використовуйте функції mean(), var(), std().

Для відтворюваності результатів перед генерацією чисел встановлюйте seed (наприклад, rand("seed", 1234) в Octave або grand("setall", 1234, 1234) в Scilab).

Рекомендується створювати два окремих графіки (для рівномірного та нормального розподілу) або використовувати subplot(). Для порівняння з теоретичним розподілом побудуйте поверхню нормальної кривої за допомогою plot().

Контрольні питання

1. Що таке псевдовипадкові числа і чим вони відрізняються від справжніх випадкових?
2. Які функції використовуються для генерації рівномірного розподілу в Octave / Scilab?
3. Як генеруються числа з нормальним розподілом?
4. Що показує гістограма розподілу?
5. Як впливає кількість інтервалів (bins) на вигляд гістограми?
6. Для чого потрібно встановлювати початкове значення генератора (seed)?
7. Які основні статистичні характеристики розподілу ви знаєте?
8. Як змінити математичне сподівання та дисперсію нормального розподілу?
9. Де в моделюванні телекомунікаційних систем використовуються псевдовипадкові числа?
10. Чому відтворюваність результатів важлива в імітаційному моделюванні?

Зміст звіту

– назва практичної роботи;
– мета виконання;
– теоретичні основи генерації псевдовипадкових чисел;
– лістинг програми з коментарями;
– гістограми рівномірного та нормального розподілів;
– таблиці статистичних характеристик;
– аналіз впливу кількості елементів та кількості інтервалів;
– висновки щодо якості згенерованих розподілів та їх застосування в моделюванні систем.

Практична робота 8. Побудова двовимірних та тривимірних графіків (plot, surf / plot3d) поверхонь

Мета роботи – опанувати інструменти візуалізації даних у GNU Octave / Scilab, навчитися будувати двовимірні графіки функцій та тривимірні поверхні, налаштовувати їх оформлення та використовувати для аналізу математичних моделей.

Ключові положення

Графічна візуалізація є одним з найважливіших етапів комп’ютерного моделювання систем. Вона дозволяє наочно представити поведінку функцій, виявити екстремуми, асимптоти, зони нестійкості та особливості моделей.

У GNU Octave та Scilab для побудови двовимірних графіків використовується функція plot(), для тривимірних поверхонь – surf() (Octave) та plot3d() або surf() (Scilab).

Для створення сітки аргументів застосовуються функції meshgrid() (Octave) або ndgrid() / meshgrid() (Scilab). Налаштування графіків включає додавання заголовків, підписів осей, легенди, сітки та кольорових карт (colormap).

Якісна візуалізація допомагає краще зрозуміти результати моделювання динамічних систем, телекомунікаційних процесів, функцій оптимізації та випадкових процесів.

Практичне завдання

1. Побудувати двовимірні графіки елементарних функцій (поліноміальної, тригонометричної, експоненціальної) на одному рисунку.
2. Налаштувати оформлення графіка: колір ліній, тип ліній, маркери, легенду, підписи осей та заголовок.
3. Побудувати графік функції з розривами та асимптотами.
4. Створити сітку аргументів за допомогою meshgrid().
5. Побудувати тривимірну поверхню простої функції двох змінних (наприклад, $z = x^2 + y^2$).
6. Побудувати поверхні складніших функцій (сідло, хвиля, гауссівський пік).
7. Налаштувати тривимірний графік: кут огляду, кольорову карту, сітку, підпис осей.
8. Побудувати контурний графік (contour()) для тієї самої поверхні.
9. Побудувати кілька поверхонь на одному рисунку за допомогою subplot().
10. Використати отримані навички для візуалізації функції, що описує характеристику телекомунікаційної системи (наприклад, рівень сигналу залежно від координат).

Методичні вказівки до виконання практичного завдання

Перед початком роботи створіть скрипт-файл. Для двовимірних графіків використовуйте команду plot(x, y, 'r--', 'LineWidth', 2) для налаштування стилю лінії.

Для створення рівномірної сітки аргументів x та y використовуйте:

- в Octave: [X, Y] = meshgrid(x, y);
- в Scilab: аналогічно або ndgrid().

Тривимірну поверхню будуйте командою surf(X, Y, Z) або plot3d(x, y, z). Після побудови графіка використовуйте команди:

- xlabel(), ylabel(), zlabel(), title()
- grid on
- colormap() – для зміни кольорової схеми
- view() або rotate3d on – для зміни кута огляду

Рекомендується використовувати функцію subplot(2,2,1) для розміщення кількох графіків на одному вікні. Для кращої наочності додавайте легенду за допомогою legend().

Обов’язково експериментуйте з різними значеннями діапазонів x та y, щоб побачити характер поведінки поверхні.

Контрольні питання

1. Яка функція використовується для побудови двовимірних графіків?
2. Як побудувати кілька функцій на одному графіку?
3. Для чого використовується функція meshgrid()?
4. У чому різниця між surf() та plot3d()?
5. Як налаштувати оформлення тривимірного графіка?
6. Що таке контурний графік і як його побудувати?
7. Як змінити кут огляду тривимірної поверхні?
8. Для чого потрібна функція colormap()?
9. Які можливості візуалізації найкорисніші при моделюванні систем?
10. Як графіки допомагають аналізувати результати моделювання?

Зміст звіту

– назва практичної роботи;
– мета виконання;
– теоретичні основи побудови двовимірних та тривимірних графіків;
– лістинг програми з коментарями;
– двовимірні графіки функцій з оформленням;
– тривимірні поверхні та контурні графіки;

– скріншоти всіх побудованих графіків;
– висновки щодо можливостей візуалізації в GNU Octave / Scilab.

Практична робота 9. Чисельне інтегрування функцій з використанням вбудованих методів (GNU Octave / Scilab)

Мета роботи – опанувати методи чисельного інтегрування функцій у GNU Octave / Scilab, навчитися використовувати вбудовані функції для обчислення визначених інтегралів та оцінювати точність отриманих результатів.

Ключові положення

Чисельне інтегрування (квадратура) – це процес наближеного обчислення визначеного інтеграла, коли аналітичне розв'язання неможливе або складне. Воно широко застосовується в моделюванні систем для розрахунку площ, об'ємів, середніх значень, енергії сигналів, ймовірностей та характеристик телекомунікаційних процесів.

У GNU Octave та Scilab основними вбудованими функціями чисельного інтегрування є:

- quad() – адаптивний метод Гаусса-Кронрода (рекомендований);
- quadl() – метод Лобатто (більш точний для гладких функцій);
- trapz() – метод трапецій (простий, але менш точний);
- simps() (Scilab) або реалізація правила Сімпсона.

Точність чисельного інтегрування залежить від методу, кількості розбиття інтервалу, гладкості функції та наявності особливостей (розривів, осциляцій). Важливо вміти порівнювати результати різних методів та оцінювати похибку.

Практичне завдання

1. Обчислити визначений інтеграл простої функції (полінома) за допомогою різних методів (quad, trapz).
2. Обчислити інтеграл тригонометричної та експоненціальної функцій.
3. Обчислити інтеграл функції з осциляціями (наприклад, sin(x)/x).
4. Дослідити залежність точності результату від параметра допустимої похибки (tol).
5. Порівняти результати вбудованих функцій з аналітичним розв'язком (де можливо).
6. Реалізувати власну просту функцію чисельного інтегрування методом трапецій.
7. Порівняти точність і швидкість роботи власної реалізації та вбудованих методів.
8. Обчислити подвійний інтеграл за допомогою вкладених викликів quad.
9. Обчислити інтеграл функції, що моделює потужність сигналу або ймовірність у телекомунікаційній системі.
10. Проаналізувати вплив кількості точок розбиття та вибору методу на точність обчислення.

Методичні вказівки до виконання практичного завдання

Перед початком роботи ознайомтеся з синтаксисом функцій quad(f, a, b, tol) та trapz(x, y). Функція quad приймає як перший аргумент дескриптор функції (@f або inline-функцію).

Для порівняння точності обчислюйте аналітичне значення інтеграла (якщо можливо) і знаходьте абсолютну похибку.

Реалізуйте метод трапецій самостійно за формулою: I ≈ h/2 * (y0 + 2*y1* + ... + 2yn-1 + yn)

Використовуйте цикл for або векторизацію для обчислень.

Для подвійного інтегрування використовуйте вкладені виклики quad. Обов'язково експериментуйте зі значенням tol (наприклад, 1e-6, 1e-8, 1e-10) і спостерігайте за зміною результату та часу виконання.

Результати оформлюйте у вигляді таблиць: функція, аналітичне значення, результат quad, результат trapz, похибка.

Контрольні питання

1. Що таке чисельне інтегрування і коли воно застосовується?
2. Які вбудовані функції чисельного інтегрування є в GNU Octave / Scilab?
3. У чому полягає метод трапецій?
4. Який метод реалізує функція quad()?
5. Як впливає параметр tol на точність і швидкість обчислення?
6. Чому для осцилюючих функцій потрібна більша кількість точок?
7. Як обчислити подвійний інтеграл?
8. У чому переваги вбудованих методів порівняно з власними реалізаціями?
9. Де в моделюванні телекомунікаційних систем використовується чисельне інтегрування?
10. Як оцінити точність отриманого чисельного інтеграла?

Зміст звіту

– назва практичної роботи;
– мета виконання;
– теоретичні основи чисельного інтегрування;
– лістинг програми з коментарями;
– таблиця порівняння результатів різних методів;
– графіки підінтегральних функцій;
– аналіз залежності точності від параметрів;
– висновки щодо вибору методів чисельного інтегрування.

Практична робота 10. Символьне та чисельне розв'язування звичайних диференціальних рівнянь (lsode в GNU Octave / ode в Scilab)

Мета роботи – опанувати методи символьного та чисельного розв'язування звичайних диференціальних рівнянь (ЗДР) у GNU Octave / Scilab, навчитися використовувати вбудовані функції lsode та ode, порівняти результати обох підходів та проаналізувати їх застосування в моделюванні динамічних систем.

Ключові положення

Звичайні диференціальні рівняння (ЗДР) є основним математичним апаратом для опису динамічних процесів у технічних, фізичних та телекомунікаційних системах (зміна струму, напруги, сигналу, навантаження мережі, поширення хвиль тощо).

Існує два основних підходи до розв'язування ЗДР:

- Символьне розв’язування – отримання аналітичного розв’язку (можливе лише для простих рівнянь).
- Чисельне розв’язування – наближене інтегрування системи диференціальних рівнянь з використанням методів Рунге-Кутта, Адамса та ін.

У GNU Octave основною функцією чисельного розв’язування є lsode() (Livermore Solver for Ordinary Differential Equations). У Scilab використовується функція ode() (з підтримкою методів rk, adams, bdf тощо).

Для символьного розв’язування застосовується пакет symbolic. Важливо правильно задати початкові умови, інтервал інтегрування та параметри точності. Чисельні методи вимагають розуміння стабільності, кроку інтегрування та можливих осциляцій розв’язку.

Практичне завдання

1. Символьно розв’язати просте диференціальне рівняння першого порядку (наприклад, y' = ky).
2. Символьно розв’язати лінійне диференціальне рівняння другого порядку.
3. Чисельно розв’язати рівняння першого порядку з використанням lsode() (Octave) або ode() (Scilab).
4. Чисельно розв’язати систему двох диференціальних рівнянь (модель Лотки-Вольтерра або коливальний контур).
5. Побудувати графіки аналітичного та чисельного розв’язків на одному рисунку.
6. Дослідити вплив кроку інтегрування та методу на точність і стабільність розв’язку.
7. Задати різні початкові умови та проаналізувати поведінку системи.
8. Розв’язати жорстке диференціальне рівняння та порівняти результати різних методів.
9. Промоделювати динамічний процес у телекомунікаційній системі (наприклад, затухання сигналу або зміна навантаження).
10. Порівняти результати символьного та чисельного розв’язування за точністю та можливостями застосування.

Методичні вказівки до виконання практичного завдання

Перед початком роботи підключіть пакет символічних обчислень. Для символьного розв’язування використовуйте функцію dsolve() з пакету symbolic.

Для чисельного розв’язування в **GNU Octave**:

- Використовуйте lsode(f, y0, t)
- Функція f повинна повертати похідну як вектор.

Для **Scilab**:

- Використовуйте ode(y0, t0, t, f) або ode("rk", ...).

Створіть окрему функцію, яка описує праву частину диференціального рівняння (наприклад, function dy = f(t, y)).

Для побудови графіків порівняння використовуйте plot(t, y_analytical, 'r', t, y_numeric, 'b--') з легендою. Обов’язково експериментуйте з різними методами (rk4, adams, bdf) та кроками часу.

Особливу увагу приділіть жорстким рівнянням (з різко змінюваними коефіцієнтами) – вони вимагають спеціальних методів.

Контрольні питання

1. Що таке звичайне диференціальне рівняння?
2. У чому різниця між символьним та чисельним розв’язуванням ЗДР?

3. Які функції використовуються для чисельного розв’язування в Octave та Scilab?
4. Як правильно задати праву частину диференціального рівняння у функції?
5. Що таке початкові умови і як їх задавати?
6. Які методи інтегрування підтримуються функцією ode()?
7. Що таке жорсткі диференціальні рівняння?
8. Як оцінити точність чисельного розв’язку?
9. Де в моделюванні телекомунікаційних систем застосовуються диференціальні рівняння?
10. Які переваги та обмеження символьного розв’язування ЗДР?

Зміст звіту

– назва практичної роботи;
– мета виконання;
– теоретичні основи розв’язування звичайних диференціальних рівнянь;
– лістинг програми (символьне та чисельне розв’язування) з коментарями;
– графіки аналітичного та чисельного розв’язків;
– таблиця порівняння результатів для різних методів і початкових умов;
– аналіз стабільності та точності розв’язків;
– висновки щодо застосування методів у моделюванні динамічних систем.

ТЕМА 3. МОДЕЛЮВАННЯ СИСТЕМ ТА МЕРЕЖ

Практична робота 11. Побудова логіко-арифметичної моделі простої системи черги (M/M/1) в Scilab

Мета роботи – навчитися будувати логіко-арифметичну модель простої системи масового обслуговування типу M/M/1 в Scilab, розраховувати основні характеристики черги та аналізувати її роботу в умовах різних навантажень.

Ключові положення

Системи масового обслуговування (СМО) широко застосовуються в телекомунікаціях для моделювання черг пакетів, запитів до серверів, буферів маршрутизаторів та каналів зв'язку.

Модель M/M/1 є найпростішою марківською системою черги:

- вхідний потік – пуассонівський з інтенсивністю λ (заявок/одиницю часу);
- час обслуговування – експоненціально розподілений з інтенсивністю μ (обслуговувань/одиницю часу);
- один канал обслуговування (single server);
- нескінченна черга.

Логіко-арифметична модель описує систему за допомогою формул стаціонарного режиму (при $\rho = \lambda/\mu < 1$):

- Середня кількість заявок у системі: $L = \rho / (1 - \rho)$
- Середня кількість заявок у черзі: $Lq = \rho^2 / (1 - \rho)$
- Середній час перебування в системі: $W = 1 / (\mu - \lambda)$
- Середній час очікування в черзі: $Wq = \rho / (\mu - \lambda)$
- Ймовірність простою сервера: $P0 = 1 - \rho$

Модель дозволяє аналізувати, як змінюються характеристики черги при збільшенні навантаження ρ.

Практичне завдання

1. Розробити програму для введення інтенсивностей λ та μ.
2. Перевірити умову існування стаціонарного режиму ($\rho < 1$).
3. Обчислити основні характеристики системи M/M/1 за аналітичними формулами.
4. Побудувати залежність середньої довжини черги Lq від навантаження ρ.
5. Побудувати залежність середнього часу очікування Wq від ρ.
6. Побудувати графік ймовірності простою сервера P0 залежно від ρ.
7. Дослідити поведінку системи при $\rho \to 1$ (критичне навантаження).
8. Реалізувати простий імітаційний експеримент для порівняння з аналітичними результатами (за бажанням).
9. Проаналізувати вплив зміни інтенсивності вхідного потоку на характеристики черги.
10. Зробити висновки щодо застосування моделі M/M/1 для телекомунікаційних систем.

Методичні вказівки до виконання практичного завдання

Виконуйте роботу в Scilab. Використовуйте функцію input() для діалогового введення значень λ та μ.

Обов’язково передбачте перевірку умови $\rho = \lambda/\mu < 1$. При порушенні умови виводьте відповідне попередження.

Розрахунок характеристик реалізуйте у вигляді окремих змінних або функцій:

- rho = lambda / mu
- L = rho / (1 - rho)
- Lq = rho^2 / (1 - rho)
- W = 1 / (mu - lambda)
- Wq = rho / (mu - lambda)
- P0 = 1 - rho

Для візуалізації створіть вектор значень ρ від 0.1 до 0.95 з певним кроком і обчисліть відповідні Lq, Wq, P0. Використовуйте функцію plot() для побудови графіків.

Рекомендується використовувати subplot() для розміщення кількох графіків на одному вікні. Додавайте сітку (grid on), підписи осей та заголовки.

Контрольні питання

1. Що таке система масового обслуговування типу M/M/1?
2. Які потоки описуються літерами M у позначенні M/M/1?
3. Яка умова існування стаціонарного режиму для M/M/1?
4. Як обчислюється навантаження системи ρ?
5. Які основні характеристики системи M/M/1 ви знаєте?
6. Як змінюється середня довжина черги при наближенні ρ до 1?
7. Чому модель M/M/1 важлива для телекомунікацій?
8. У чому різниця між часом очікування в черзі Wq та часом перебування в системі W?
9. Як впливає збільшення інтенсивності обслуговування μ на характеристики черги?
10. Які обмеження має аналітична модель M/M/1?

Зміст звіту

– назва практичної роботи;
– мета виконання;
– теоретичні основи моделі M/M/1;
– лістинг програми з коментарями;
– результати розрахунку характеристик для різних значень λ та μ;
– графіки залежностей Lq(ρ), Wq(ρ), P0(ρ);
– аналіз поведінки системи при різних навантаженнях;
– висновки щодо практичного застосування моделі в телекомунікаційних системах.

Практична робота 12. Моделювання простої телекомунікаційної мережі з двома вузлами та розрахунок характеристик (GNU Octave / Scilab)

Мета роботи – навчитися будувати просту математичну модель телекомунікаційної мережі з двома вузлами, розраховувати її основні характеристики (пропускну здатність, затримку, завантаження каналу) та аналізувати залежність цих характеристик від параметрів мережі.

Ключові положення

Проста телекомунікаційна мережа з двома вузлами (джерело – приймач) є базовою моделлю для розуміння процесів передачі даних. Така мережа складається з джерела даних, каналу зв'язку та приймача.

Основними характеристиками мережі є:

- Інтенсивність вхідного потоку (λ, пакетів/с);
- Пропускна здатність каналу (C, біт/с);
- Середній розмір пакета (L, біт);
- Інтенсивність обслуговування (μ = C / L, пакетів/с);
- Навантаження каналу (ρ = λ / μ);
- Середня затримка передачі пакета (T);
- Середня кількість пакетів у черзі та в системі.

Модель часто базується на теорії масового обслуговування (M/M/1 або M/D/1). При передачі даних через канал затримка складається з часу обслуговування, часу очікування в черзі та часу розповсюдження сигналу.

Аналіз таких моделей дозволяє оцінити, як зміна швидкості каналу, розміру пакетів або інтенсивності потоку впливає на якість обслуговування в мережі.

Практичне завдання

1. Розробити програму для введення параметрів мережі: інтенсивність потоку λ, пропускна здатність каналу C, середній розмір пакета L.
2. Обчислити інтенсивність обслуговування μ та навантаження каналу ρ.
3. Розрахувати основні характеристики: середню затримку передачі, середню довжину черги, середню кількість пакетів у системі.
4. Побудувати графіки залежності середньої затримки від навантаження ρ.
5. Побудувати графіки залежності середньої довжини черги від ρ.
6. Дослідити поведінку мережі при наближенні навантаження до 1 (ρ → 1).
7. Реалізувати розрахунок для двох різних типів каналів (різні швидкості).
8. Порівняти результати для моделі M/M/1 та M/D/1 (детермінований час обслуговування).
9. Проаналізувати вплив розміру пакета на затримку в мережі.
10. Зробити висновки щодо оптимального навантаження каналу в телекомунікаційних мережах.

Методичні вказівки до виконання практичного завдання

Виконуйте роботу в GNU Octave або Scilab (рекомендується Scilab для цієї теми). Використовуйте діалогове введення параметрів за допомогою input().

Основні розрахункові формули (для моделі M/M/1):

- rho = lambda / mu
- mu = C / L
- Середня затримка: T = 1 / (mu - lambda)
- Середня кількість пакетів у системі: N = rho / (1 - rho)
- Середня кількість у черзі: Nq = rho^2 / (1 - rho)

Передбачте перевірку умови rho < 1. При ρ ≥ 1 виводьте попередження про перевантаження каналу.

Для візуалізації створіть вектор значень ρ від 0.05 до 0.95 і обчисліть відповідні характеристики. Використовуйте функцію plot() з чітким оформленням (підписи осей, заголовок, сітка).

Рекомендується використовувати subplot(2,1,1) та subplot(2,1,2) для одночасного відображення залежностей затримки та довжини черги.

Контрольні питання

1. Які основні елементи входять до простої телекомунікаційної мережі з двома вузлами?
2. Що таке навантаження каналу ρ і як воно обчислюється?

3. Від чого залежить пропускна здатність каналу?
4. Як обчислюється середня затримка передачі пакета в моделі M/M/1?
5. Чому затримка різко зростає при наближенні ρ до 1?
6. У чому різниця між моделями M/M/1 та M/D/1?
7. Як розмір пакета впливає на характеристики мережі?
8. Які основні характеристики телекомунікаційної мережі вираховуються в цій роботі?
9. Для чого використовується моделювання мереж з двома вузлами?
10. Яке оптимальне навантаження каналу рекомендується для стабільної роботи мережі?

Зміст звіту

– назва практичної роботи;
– мета виконання;
– теоретичні основи моделювання простої телекомунікаційної мережі;
– лістинг програми з коментарями;
– результати розрахунку характеристик для різних параметрів;
– графіки залежності затримки та довжини черги від навантаження ρ;
– аналіз поведінки мережі при різних рівнях навантаження;
– висновки щодо практичного застосування моделі.

Практична робота 13. Імітаційне моделювання комп'ютерної мережі з використанням принципів дискретно-подійного моделювання (Scilab)

Мета роботи – опанувати принципи дискретно-подійного імітаційного моделювання (Discrete Event Simulation – DES) для аналізу роботи комп'ютерної мережі, навчитися реалізовувати просту імітаційну модель у Scilab та оцінювати її характеристики.

Ключові положення

Дискретно-подійне моделювання – це метод імітації, при якому стан системи змінюється лише в дискретні моменти часу, що відповідають настанню певних подій (прихід пакета, завершення обслуговування, вихід пакета з системи тощо).

На відміну від неперервного моделювання, час у DES просувається стрибкоподібно – від однієї події до наступної. Це робить метод особливо ефективним для моделювання телекомунікаційних мереж, черг пакетів, маршрутизаторів та серверів.

Основні компоненти дискретно-подійної моделі:

- Генератор подій (прихід пакетів – зазвичай пуассонівський потік);
- Черга (буфер);
- Сервер (канал обслуговування);
- Календар подій (список майбутніх подій, відсортований за часом);
- Статистичні накопичувачі (для підрахунку середньої довжини черги, затримки, завантаження тощо).

У Scilab така модель реалізується за допомогою циклів, структур даних (списків або матриць) та генераторів псевдовипадкових чисел (grand()). Основна логіка – обробка подій у хронологічному порядку з оновленням поточного часу моделювання.

Практичне завдання

1. Реалізувати генератор подій приходу пакетів за пуассонівським законом (експоненціальний інтервал між прибуттями).
2. Реалізувати чергу (буфер) з обмеженою або необмеженою ємністю.
3. Реалізувати сервер з експоненціальним часом обслуговування.
4. Створити календар подій та механізм просування модельного часу.
5. Реалізувати обробку двох типів подій: «прихід пакета» та «завершення обслуговування».

6. Зібрати статистику: середню довжину черги, середню затримку пакетів, коефіцієнт завантаження сервера, ймовірність втрат (якщо черга обмежена).
7. Провести імітаційний експеримент протягом заданого модельного часу (наприклад, 1000 або 5000 одиниць часу).
8. Порівняти результати імітаційного моделювання з аналітичними формулами моделі M/M/1.
9. Дослідити вплив інтенсивності вхідного потоку λ на характеристики мережі.
10. Проаналізувати отримані результати та зробити висновки щодо поведінки комп'ютерної мережі.

Методичні вказівки до виконання практичного завдання

Виконуйте роботу виключно в Scilab. Для генерації експоненціально розподілених інтервалів часу використовуйте grand(1,1,"exp", mean).

Основна структура програми:

- Ініціалізація змінних (поточний час, черга, статистика);
- Цикл моделювання до досягнення кінцевого модельного часу;
- Обробка наступної події (вибір події з найменшим часом);
- Оновлення статистики після кожної події.

Рекомендується використовувати список або вектор для зберігання черги (часи прибуття пакетів). Для календаря подій можна використовувати просту змінну «наступна подія» або масив подій.

Для накопичення статистики використовуйте змінні для інтегрального підрахунку (метод «площі під кривою» для середньої довжини черги).

Обов'язково додайте вивід поточних результатів кожні 500–1000 одиниць модельного часу для наочності.

Контрольні питання

1. Що таке дискретно-подійне імітаційне моделювання?
2. Які основні компоненти DES-моделі комп'ютерної мережі?
3. Як просувається модельний час у дискретно-подійному моделюванні?
4. Які типи подій реалізуються в простій моделі мережі?
5. Як генерується пуассонівський потік пакетів?
6. У чому перевага імітаційного моделювання порівняно з аналітичним?
7. Як збирається статистика середньої довжини черги?
8. Чому важливо порівнювати результати імітації з аналітичними формулами M/M/1?
9. Які труднощі виникають при реалізації календаря подій?
10. Де в реальних телекомунікаційних системах застосовується дискретно-подійне моделювання?

Зміст звіту

– назва практичної роботи;
– мета виконання;
– теоретичні основи дискретно-подійного моделювання;
– блок-схема алгоритму моделі;
– лістинг програми з коментарями;
– результати імітаційного експерименту (таблиці та графіки);
– порівняння результатів імітації з аналітичною моделлю M/M/1;
– аналіз впливу параметрів на характеристики мережі;
– висновки.

Практична робота 14. Моделювання випадкових процесів у телекомунікаційних мережах методом Монте-Карло (Scilab)

Мета роботи – опанувати метод Монте-Карло для моделювання випадкових процесів у телекомунікаційних мережах, навчитися оцінювати ймовірності, середні значення та інші характеристики систем за допомогою великої кількості випадкових випробувань у Scilab.

Ключові положення

Метод Монте-Карло – це чисельний метод, заснований на багаторазовому повторенні випадкових випробувань для отримання наближених розв'язків задач, особливо тих, що містять випадкові величини або складні ймовірнісні процеси.

У телекомунікаційних мережах метод Монте-Карло широко застосовується для:

- оцінки ймовірності втрат пакетів;
- розрахунку середньої затримки при наявності завад;
- моделювання впливу шумів на якість сигналу;
- оцінки пропускної здатності каналу в умовах випадкового навантаження;
- аналізу надійності мережі.

Основна ідея методу: замість аналітичного розв'язання складних рівнянь проводять тисячі або мільйони імітаційних експериментів з використанням генераторів псевдовипадкових чисел (grand() в Scilab), а потім обчислюють статистичні оцінки шуканих величин.

Перевага методу – універсальність і можливість враховувати складні розподіли та залежності, які важко описати аналітично. Недолік – висока обчислювальна складність і статистична похибка, яка зменшується зі збільшенням кількості випробувань N.

Практичне завдання

1. Реалізувати генерацію пуассонівського вхідного потоку пакетів методом Монте-Карло.
2. Промоделювати процес передачі пакетів через канал з експоненціальним часом обслуговування.
3. Оцінити ймовірність втрати пакета при обмеженому розмірі буфера.
4. Обчислити середню затримку пакетів, що успішно пройшли через мережу.
5. Оцінити коефіцієнт завантаження каналу.
6. Провести серію експериментів при різних значеннях навантаження ρ (від 0.3 до 0.95).
7. Дослідити залежність ймовірності втрат і середньої затримки від розміру буфера.
8. Побудувати графіки залежності ключових характеристик від навантаження та розміру буфера.
9. Порівняти результати методу Монте-Карло з аналітичними формулами моделі M/M/1/K (з обмеженою чергою).
10. Проаналізувати точність оцінки залежно від кількості імітаційних випробувань.

Методичні вказівки до виконання практичного завдання

Виконуйте роботу в **Scilab**. Використовуйте функцію grand() для генерації випадкових величин:

- grand(n, "poi", lambda) – для пуассонівського розподілу;
- grand(n, "exp", mu) – для експоненціального розподілу.

Структура програми:

1. Введення параметрів (λ, μ, розмір буфера K, кількість випробувань N, загальний час моделювання).
2. Цикл Монте-Карло (велика кількість реалізацій).
3. У кожній реалізації – імітація потоку подій протягом модельного часу.

4. Накопичення статистики (кількість втрачених пакетів, сумарна затримка, час зайнятості сервера).
5. Обчислення середніх значень та ймовірностей.

Для ефективності рекомендується проводити 5000–20000 імітаційних прогонів. Використовуйте векторизацію Scilab там, де можливо, щоб пришвидшити обчислення.

Обов'язково фіксуйте кількість випробувань і обчислюйте довірчі інтервали або стандартну похибку оцінки.

Контрольні питання

1. У чому полягає сутність методу Монте-Карло?
2. Чому метод Монте-Карло особливо корисний для моделювання телекомунікаційних мереж?
3. Як генеруються випадкові моменти приходу пакетів у методі Монте-Карло?
4. Як оцінюється ймовірність події за допомогою Монте-Карло?
5. Від чого залежить точність оцінки в методі Монте-Карло?
6. Які характеристики мережі можна оцінити методом Монте-Карло?
7. У чому відмінність між аналітичною моделлю M/M/1 та моделлю Монте-Карло?
8. Як впливає розмір буфера на ймовірність втрат пакетів?
9. Як збільшити точність результатів методу Монте-Карло?
10. Де в реальних телекомунікаційних системах застосовується метод Монте-Карло?

Зміст звіту

– назва практичної роботи;
– мета виконання;
– теоретичні основи методу Монте-Карло;
– опис алгоритму імітаційної моделі;
– лістинг програми з коментарями;
– таблиці та графіки отриманих характеристик (залежність від ρ та розміру буфера);
– порівняння результатів Монте-Карло з аналітичними формулами;
– аналіз точності залежно від кількості випробувань;
– висновки щодо застосування методу в моделюванні телекомунікаційних мереж.

Практична робота 15. Аналіз та розрахунок характеристик телекомунікаційної мережі (пропускна здатність, затримка, ймовірність втрат)

Мета роботи – навчитися комплексно аналізувати та розраховувати ключові характеристики телекомунікаційної мережі (пропускну здатність, середню затримку передачі даних та ймовірність втрат пакетів) з використанням аналітичних формул і результатів імітаційного моделювання в Scilab / GNU Octave.

Ключові положення

Основними характеристиками якості функціонування телекомунікаційної мережі є:

- Пропускна здатність – максимальна кількість даних, яку мережа може передати за одиницю часу (біт/с, пакетів/с).
- Середня затримка (latency) – середній час від моменту надходження пакета в мережу до його доставки отримувачу (включає час обслуговування, час очікування в чергах та час розповсюдження).
- Ймовірність втрат пакетів (packet loss probability) – частка пакетів, які втрачаються через переповнення буферів або помилки передачі.

Ці характеристики тісно пов’язані між собою. При збільшенні навантаження на мережу (ρ) пропускна здатність зростає до певної межі, але різко збільшуються затримка та ймовірність втрат.

Для розрахунку використовуються:

- Аналітичні формули теорії масового обслуговування (M/M/1, M/M/1/K, M/D/1);
- Результати імітаційного моделювання (дискретно-подійного або Монте-Карло).

Аналіз цих характеристик дозволяє оцінити ефективність мережі, знайти оптимальне робоче навантаження та визначити необхідні параметри обладнання (розмір буферів, швидкість каналів).

Практичне завдання

1. Розрахувати пропускну здатність каналу для заданих параметрів (швидкість каналу, середній розмір пакета).
2. Обчислити навантаження мережі ρ для різних значень інтенсивності вхідного потоку λ.
3. Розрахувати середню затримку передачі даних за аналітичними формулами M/M/1.
4. Розрахувати ймовірність втрат пакетів для моделі з обмеженою чергою (M/M/1/K).
5. Використовуючи результати попередніх імітаційних робіт (роботи 11–14), порівняти аналітичні розрахунки з даними імітаційного моделювання.
6. Побудувати графіки залежності:
   - середньої затримки від навантаження ρ;
   - ймовірності втрат від розміру буфера;
   - ефективної пропускної здатності від ρ.
7. Дослідити вплив розміру пакета на затримку та пропускну здатність.
8. Знайти оптимальне значення навантаження ρ, при якому затримка залишається прийнятною, а ймовірність втрат – низькою.
9. Провести порівняльний аналіз для мереж з одним та двома каналами (паралельне обслуговування).
10. Зробити висновки щодо рекомендацій щодо проектування телекомунікаційної мережі.

Методичні вказівки до виконання практичного завдання

Виконуйте роботу в Scilab або GNU Octave. Використовуйте результати та функції, розроблені в попередніх роботах (модель M/M/1, дискретно-подійну модель, метод Монте-Карло).

Основні розрахункові формули:

- Пропускна здатність: C = швидкість_каналу (біт/с), μ = C / L (пакетів/с)
- Навантаження: $\rho = \lambda / \mu$
- Середня затримка (M/M/1): $T = 1 / (\mu - \lambda)$
- Ймовірність втрат (M/M/1/K): формула Ерланга або результат імітації

Створіть єдиний скрипт, який:

- дозволяє вводити або задавати параметри мережі;
- виконує аналітичні розрахунки;
- завантажує або повторює результати імітаційного моделювання;
- будує порівняльні графіки.

Використовуйте subplot() для розміщення кількох графіків. Обов’язково додавайте легенди, сітку та чіткі підписи.

Контрольні питання

1. Які основні характеристики телекомунікаційної мережі ви знаєте?
2. Що таке пропускна здатність каналу і від чого вона залежить?
3. Як розраховується середня затримка в моделі M/M/1?
4. Чому затримка різко зростає при $\rho \to 1$?
5. Що таке ймовірність втрат пакетів і як вона залежить від розміру буфера?

6. У чому полягає взаємозв’язок між затримкою, пропускною здатністю та ймовірністю втрат?
7. Як розмір пакета впливає на характеристики мережі?
8. Яке оптимальне навантаження рекомендується для телекомунікаційних мереж?
9. Чому важливо порівнювати аналітичні розрахунки з результатами імітаційного моделювання?
10. Як результати аналізу характеристик використовуються при проектуванні мереж?

Зміст звіту

– назва практичної роботи;
– мета виконання;
– теоретичні основи розрахунку характеристик телекомунікаційної мережі;
– лістинг програми з коментарями;
– таблиці розрахункових характеристик для різних параметрів;
– графіки залежностей затримки, ймовірності втрат та пропускної здатності;
– порівняння аналітичних та імітаційних результатів;
– аналіз оптимального режиму роботи мережі;
– висновки та рекомендації.

ТЕМА 4. ОПТИМІЗАЦІЯ СИСТЕМ ТА МЕРЕЖ

Практична робота 16. Розв'язування задачі лінійного програмування за допомогою glpk (GNU Octave) або SciGLPK (Scilab)

Мета роботи – навчитися формулювати та розв'язувати задачі лінійного програмування (ЛП) у GNU Octave за допомогою функції glpk() або в Scilab за допомогою пакету SciGLPK, а також інтерпретувати отримані оптимальні розв'язки в контексті оптимізації телекомунікаційних систем.

Ключові положення

Лінійне програмування – це метод оптимізації, який дозволяє знайти екстремум лінійної цільової функції за наявності лінійних обмежень-нерівностей. Задача ЛП записується у стандартній формі:

Максимізувати (або мінімізувати) $\mathbf{Z = c_1x_1 + c_2x_2 + \ldots + c_nx_n}$

при обмеженнях: $\mathbf{A \cdot x \leq b \; x \geq 0}$

У телекомунікаціях задачі лінійного програмування часто виникають при:

- максимізації пропускної здатності мережі;
- мінімізації вартості обладнання;
- оптимальному розподілі ресурсів (каналів, частот, потужності);
- максимізації прибутку від надання послуг за обмежених ресурсів.

У GNU Octave для розв'язування задач ЛП використовується функція glpk() (GNU Linear Programming Kit). У Scilab – пакет **SciGLPK**, який надає аналогічну функцію glpk().

Функція glpk() дозволяє розв'язувати задачі як у стандартній, так і в канонічній формі, з підтримкою цілочисельних змінних.

Практичне завдання

1. Сформулювати математичну модель задачі лінійного програмування: максимізація прибутку провайдера при обмежених ресурсах (канали, обладнання, бюджет).
2. Записати задачу у стандартній формі (цільова функція + обмеження).
3. Реалізувати розв'язання задачі за допомогою glpk() в GNU Octave або SciGLPK в Scilab.
4. Проаналізувати отриманий оптимальний розв'язок (значення змінних, значення цільової функції).
5. Виконати чуттєвий аналіз: як змінюється оптимальний розв'язок при зміні коефіцієнтів цільової функції або обмежень.
6. Розв'язати задачу при різних значеннях обмежень (бюджет, кількість каналів).
7. Порівняти результати при максимізації прибутку та мінімізації витрат.
8. Реалізувати задачу з цілочисельними змінними (Integer Linear Programming).
9. Побудувати графічну інтерпретацію розв'язку для задачі з двома змінними.
10. Зробити висновки щодо застосування лінійного програмування в оптимізації телекомунікаційних мереж.

Методичні вказівки до виконання практичного завдання

У GNU Octave:

octave

```
[xopt, fmin, status] = glpk(c, A, b, lb, ub, ctype, vartype, sense)
```

У Scilab (після встановлення SciGLPK): Аналогічний синтаксис.

Рекомендована постановка задачі (приклад для роботи):

- x_1, x_2 – кількість послуг двох типів (наприклад, домашній інтернет і корпоративний зв'язок);
- Максимізувати прибуток $Z = 50x_1 + 80x_2$;

– Обмеження: кількість каналів, бюджет, час фахівців тощо.

Перед розв’язанням обов’язково:

1. Записати задачу в математичній формі (цільова функція + система обмежень).
2. Привести до стандартної форми glpk (вектор c, матриця A, вектор b).
3. Вказати типи обмежень (ctype) та типи змінних (vartype).

Для графічної інтерпретації (при n=2) використовуйте plot() для побудови області допустимих рішень та позначення оптимальної точки.

Контрольні питання

1. Що таке лінійне програмування і в яких задачах воно застосовується?
2. Яка стандартна форма задачі лінійного програмування?
3. Які функції використовуються для розв’язування ЛП в GNU Octave та Scilab?
4. Що таке цільова функція та обмеження?
5. Як інтерпретувати результат функції glpk()?
6. У чому різниця між неперервним та цілочисельним лінійним програмуванням?
7. Як виконується аналіз чутливості в задачах ЛП?
8. Які задачі оптимізації телекомунікаційних мереж можна розв’язувати за допомогою ЛП?
9. Що означає статус розв’язку (status) у функції glpk?
10. Які переваги та обмеження методу лінійного програмування?

Зміст звіту

– назва практичної роботи;
– мета виконання;
– математична постановка задачі лінійного програмування;
– лістинг програми з коментарями;
– результати розв’язання (оптимальні значення змінних, значення цільової функції);
– графічна інтерпретація області допустимих рішень (для $n \leq 2$);
– аналіз чутливості та порівняння різних варіантів;
– висновки щодо застосування лінійного програмування в телекомунікаціях.

Практична робота 17. Розв’язування задачі цілочисельного програмування

Мета роботи – навчитися формулювати та розв’язувати задачі цілочисельного лінійного програмування (Integer Linear Programming – ILP), використовувати функцію glpk() з цілочисельними змінними та аналізувати отримані оптимальні розв’язки в контексті оптимізації телекомунікаційних мереж.

Ключові положення

Цілочисельне програмування – це розширення лінійного програмування, в якому всі або частина змінних повинні набувати тільки цілих значень ($x \in \mathbb{Z}$). Такі задачі виникають у випадках, коли змінні мають фізичний зміст: кількість обладнання, кількість каналів, кількість серверів, кількість частотних каналів тощо.

На відміну від звичайного лінійного програмування, задача ILP є NP-складною. Для її розв’язання використовуються методи гілок і меж (branch and bound), відсікання Гоморі та інші комбінаторні методи.

У GNU Octave та Scilab функція glpk() підтримує цілочисельне програмування через параметр vartype, де для цілочисельних змінних вказується символ 'I' (Integer).

Типові задачі цілочисельного програмування в телекомунікаціях:

– Оптимальне розміщення серверів або базових станцій;
– Розподіл каналів зв’язку між користувачами;

– Вибір кількості маршрутизаторів або комутаторів;
– Оптимізація кількості ліцензій на програмне забезпечення.

Практичне завдання

1. Сформулювати задачу цілочисельного програмування: максимізація прибутку від розгортання телекомунікаційної мережі при обмежених ресурсах (кількість маршрутизаторів, серверів, бюджет).
2. Записати математичну модель задачі (цільова функція + система обмежень).
3. Реалізувати розв'язання задачі за допомогою glpk() з цілочисельними змінними.
4. Порівняти розв'язок цілочисельної задачі з розв'язком її неперервної релаксації (звичайне ЛП).
5. Проаналізувати, наскільки погіршується значення цільової функції через вимогу цілочисельності.
6. Дослідити вплив зміни обмежень (бюджету, кількості доступних пристроїв) на оптимальне рішення.
7. Розв'язати задачу при різних значеннях параметрів і порівняти результати.
8. Побудувати графічну інтерпретацію області допустимих рішень для випадку з двома змінними.
9. Проаналізувати оптимальне рішення з точки зору практичної реалізації в телекомунікаційній мережі.
10. Зробити висновки щодо особливостей розв'язування цілочисельних задач порівняно зі звичайним лінійним програмуванням.

Методичні вказівки до виконання практичного завдання

У функції glpk() для цілочисельного програмування необхідно:

- У векторі vartype вказати 'I' для кожної цілочисельної змінної (наприклад, vartype = "II" для двох змінних).
- Нижня межа змінних lb зазвичай дорівнює 0.

Приклад постановки задачі (рекомендований для роботи): Максимізувати прибуток $Z = 120x_1 + 150x_2$ при обмеженнях:

– $2x_1 + 3x_2 \leq 18$ (кількість портів/каналів)
– $4x_1 + 2x_2 \leq 16$ (бюджетні обмеження)
– $x_1, x_2 \geq 0$, x_1, x_2 – цілі числа (кількість маршрутизаторів різних типів)

У програмі:

1. Задати вектор коефіцієнтів цільової функції c.
2. Задати матрицю обмежень A та вектор правих частин b.
3. Вказати типи обмежень (ctype = "UU") та типи змінних (vartype = "II").
4. Викликати [xopt, fopt] = glpk(c, A, b, lb, ub, ctype, vartype, 1) (1 – максимізація).

Порівняйте отриманий цілочисельний розв'язок з розв'язком, отриманим без обмеження цілочисельності (vartype = "CC").

Контрольні питання

1. У чому полягає відмінність цілочисельного програмування від звичайного лінійного програмування?
2. Які задачі в телекомунікаціях вимагають цілочисельних змінних?
3. Як задаються цілочисельні змінні у функції glpk()?
4. Що таке релаксація задачі цілочисельного програмування?
5. Чому значення цільової функції в ILP зазвичай гірше, ніж у неперервній релаксації?
6. Які методи використовуються для розв'язування задач ILP?
7. Як виконується аналіз чутливості в цілочисельних задачах?

8. Які труднощі виникають при розв’язуванні великих задач цілочисельного програмування?

9. Для чого потрібне порівняння ILP з неперервним ЛП?

10. Де в оптимізації телекомунікаційних мереж застосовується цілочисельне програмування?

Зміст звіту

– назва практичної роботи;
– мета виконання;
– математична постановка задачі цілочисельного програмування;
– лістинг програми з коментарями;
– результати розв’язання цілочисельної задачі;
– порівняння з неперервною релаксацією;
– графічна інтерпретація (для задач з двома змінними);
– аналіз впливу зміни обмежень на оптимальне рішення;
– висновки щодо особливостей цілочисельного програмування в телекомунікаціях.

Практична робота 18. Багатокритеріальна оптимізація параметрів телекомунікаційної мережі (GNU Octave / Scilab)

Мета роботи – навчитися формулювати та розв’язувати задачі багатокритеріальної оптимізації параметрів телекомунікаційної мережі, опанувати методи зведення багатокритеріальної задачі до однокритеріальної (метод головної критерію, метод зважених сум, метод ідеальної точки) та аналізувати компромісні рішення.

Ключові положення

Багатокритеріальна оптимізація (Multi-Objective Optimization) виникає тоді, коли потрібно одночасно оптимізувати кілька, часто суперечливих, цілей. У телекомунікаційних мережах типовими критеріями є:

- Максимізація пропускної здатності;
- Мінімізація середньої затримки передачі даних;
- Мінімізація ймовірності втрат пакетів;
- Мінімізація вартості розгортання мережі;
- Максимізація надійності (мінімізація простою).

Оскільки критерії конфліктують між собою (наприклад, збільшення пропускної здатності часто призводить до зростання затримки при високому навантаженні), не існує єдиного оптимального рішення. Замість цього шукається множина Парето (Pareto front) – набір компромісних рішень, у яких покращення одного критерію неможливе без погіршення іншого.

Для практичного розв’язування багатокритеріальних задач найчастіше використовують методи зведення до однокритеріальної задачі:

1. Метод головної критерію (один критерій оптимізується, інші переводяться в обмеження);
2. Метод зважених сум (введення вагових коефіцієнтів);
3. Метод ідеальної (еталонної) точки.

У GNU Octave / Scilab такі задачі розв’язуються шляхом багаторазового виклику функції glpk() при різних значеннях ваг або обмежень.

Практичне завдання

1. Сформулювати багатокритеріальну задачу оптимізації параметрів телекомунікаційної мережі (максимізація пропускної здатності, мінімізація затримки, мінімізація вартості).
2. Записати математичну модель з трьома критеріями.
3. Реалізувати метод зважених сум: розв'язати задачу для різних наборів вагових коефіцієнтів (наприклад, 10–15 варіантів).
4. Реалізувати метод головної критерію (оптимізувати пропускну здатність при обмеженнях на затримку та вартість).
5. Побудувати множину Парето в координатах «пропускна здатність – затримка».
6. Побудувати множину Парето в координатах «вартість – затримка».
7. Проаналізувати компромісні рішення: як змінюється оптимальне рішення при зміні пріоритетів (ваг).
8. Знайти компромісне рішення за методом ідеальної точки.
9. Порівняти результати різних методів багатокритеріальної оптимізації.
10. Зробити висновки щодо практичного вибору параметрів телекомунікаційної мережі при наявності кількох конфліктуючих критеріїв.

Методичні вказівки до виконання практичного завдання

Використовуйте GNU Octave або Scilab. Основний підхід – метод зважених сум:

Мінімізувати (або максимізувати) $F = w_1 \cdot f_1 + w_2 \cdot f_2 + w_3 \cdot f_3$ при $\sum w_i = 1$, $w_i \geq 0$

Для зручності нормалізуйте критерії (приведіть до безрозмірного вигляду).

Структура програми:

- Задати діапазони вагових коефіцієнтів (наприклад, w1 від 0 до 1 з кроком 0.1).
- У циклі змінювати ваги і розв'язувати однокритеріальну задачу за допомогою glpk().
- Зберігати отримані значення критеріїв у вектори.
- Побудувати графіки множини Парето за допомогою plot() або scatter().

Рекомендована постановка задачі:

- x_1, x_2 – кількість обладнання двох типів (маршрутизатори, сервери);
- Критерії: максимізація пропускної здатності, мінімізація середньої затримки, мінімізація загальної вартості.

Обов'язково виводьте таблицю з вагами та значеннями всіх трьох критеріїв для кожного розв'язку.

Контрольні питання

1. Що таке багатокритеріальна оптимізація і чому вона складніша за однокритеріальну?
2. Що таке множина Парето?
3. Які методи зведення багатокритеріальної задачі до однокритеріальної ви знаєте?
4. У чому полягає метод зважених сум?
5. Як виконується нормалізація критеріїв?
6. Які критерії найчастіше конфліктують у телекомунікаційних мережах?
7. Як змінюється оптимальне рішення при збільшенні ваги затримки?
8. У чому перевага методу головної критерію?

9. Як інтерпретувати результати багатокритеріальної оптимізації?

10. Де в проектуванні телекомунікаційних мереж застосовується багатокритеріальна оптимізація?

Зміст звіту

– назва практичної роботи;
– мета виконання;
– математична постановка багатокритеріальної задачі;
– опис використаних методів (зважених сум, головної критерію);
– лістинг програми з коментарями;
– таблиця результатів для різних вагових коефіцієнтів;
– графіки множини Парето;
– аналіз компромісних рішень;
– висновки щодо вибору параметрів мережі.

Практична робота 19. Оптимізація розміщення серверів у мережі для мінімізації середньої затримки

Мета роботи – навчитися формулювати та розв'язувати задачу оптимізації розміщення серверів у телекомунікаційній мережі з метою мінімізації середньої затримки передачі даних, використовуючи методи лінійного та цілочисельного програмування.

Ключові положення

Оптимізація розміщення серверів (facility location problem) – одна з найважливіших задач у проектуванні сучасних телекомунікаційних мереж, дата-центрів та хмарних систем.

Середня затримка передачі даних складається з:

– часу обробки запиту на сервері;
– часу передачі даних мережею (залежить від відстані між клієнтом і сервером);
– часу очікування в чергах.

Мета оптимізації – вибрати оптимальні місця розміщення обмеженої кількості серверів серед можливих локацій, щоб мінімізувати сумарну (або середню) затримку для всіх клієнтів.

Задача відноситься до класу цілочисельного лінійного програмування, оскільки змінні розміщення серверів є бінарними (0 або 1). Математична модель включає:

– Бінарні змінні xj x_j xj – чи розміщений сервер у локації j j j;
– Бінарні змінні yij y_{ij} yij – чи обслуговується клієнт i i i сервером j j j;
– Обмеження на кількість серверів;
– Обмеження на те, що кожен клієнт обслуговується рівно одним сервером.

Функція мети – мінімізація сумарної затримки: min⁡∑i∑jdij·yij \min \sum_{i} \sum_{j} d_{ij} \cdot y_{ij} min∑i∑jdij·yij де dij d_{ij} dij – затримка між клієнтом i i i i сервером j j j.

Практичне завдання

1. Сформулювати задачу оптимізації розміщення серверів як задачу цілочисельного лінійного програмування.

2. Задати координати клієнтів і можливих локацій серверів (наприклад, 8–12 клієнтів і 5–7 можливих місць розміщення).

3. Обчислити матрицю затримок dij d_{ij} dij між кожним клієнтом і кожною локацією.

4. Реалізувати математичну модель у GNU Octave / Scilab з використанням glpk().

5. Розв’язати задачу при обмеженні на кількість серверів (наприклад, 2 або 3 сервери).

6. Визначити оптимальні місця розміщення серверів та призначення клієнтів.

7. Обчислити значення середньої затримки в оптимальному варіанті.

8. Провести аналіз чутливості: як змінюється рішення при зміні кількості доступних серверів.

9. Порівняти отриманий результат з «жадібним» алгоритмом (розміщення серверів у найближчих до найбільших груп клієнтів точках).

10. Зробити висновки щодо ефективності оптимального розміщення серверів для зменшення затримки в мережі.

Методичні вказівки до виконання практичного завдання

Використовуйте GNU Octave або Scilab з функцією glpk().

Математична модель:

– Змінні: xj∈{0,1} x_j \in \{0,1\} xj∈{0,1} – розміщення сервера в локації j j j;

– Змінні: yij∈{0,1} y_{ij} \in \{0,1\} yij∈{0,1} – призначення клієнта i i i до сервера j j j.

Обмеження:

– Кожен клієнт обслуговується рівно одним сервером: ∑jyij=1 \sum_j y_{ij} = 1 ∑jyij=1;

– Клієнт може бути призначений тільки до відкритого сервера: yij≤xj y_{ij} \leq x_j yij≤xj;

– Обмеження на кількість серверів: ∑jxj≤p \sum_j x_j \leq p ∑jxj≤p (p – максимальна кількість серверів).

Цільова функція: мінімізація ∑i∑jdij·yij \sum_i \sum_j d_{ij} \cdot y_{ij} ∑i∑jdij·yij.

У glpk():

– vartype повинен містити 'I' для всіх бінарних змінних;

– Нижня межа lb = 0, верхня ub = 1.

Рекомендується спочатку розв’язати задачу як мішане цілочисельне програмування (MILP). Для невеликої кількості клієнтів і локацій задача розв’язується швидко.

Для візуалізації побудуйте графік розміщення клієнтів і оптимальних серверів (використовуйте plot() з маркерами різних кольорів).

Контрольні питання

1. У чому полягає задача оптимального розміщення серверів?

2. Які змінні використовуються в математичній моделі цієї задачі?

3. Чому задачу розміщення серверів відносять до цілочисельного програмування?

4. Як формулюється обмеження «кожен клієнт обслуговується рівно одним сервером»?

5. Як впливає кількість дозволених серверів на середню затримку?

6. У чому різниця між оптимальним рішенням і «жадібним» алгоритмом?

7. Які фактори (крім відстані) можуть впливати на затримку в реальній мережі?
8. Як можна розширити модель для врахування місткості серверів?
9. Для чого потрібен аналіз чутливості в задачі розміщення?
10. Де в сучасних телекомунікаційних системах застосовується оптимізація розміщення серверів?

Зміст звіту

– назва практичної роботи;
– мета виконання;
– математична постановка задачі оптимізації розміщення серверів;
– опис вхідних даних (координати клієнтів і локацій);
– лістинг програми з коментарями;
– результати оптимального розміщення серверів;
– значення мінімальної середньої затримки;
– візуалізація розміщення клієнтів і серверів;
– порівняння з жадібним методом;
– висновки щодо ефективності оптимального розміщення.

Практична робота 20. Стохастична оптимізація параметрів мережі з використанням методу Монте-Карло (Scilab)

Мета роботи – опанувати методи стохастичної оптимізації параметрів телекомунікаційної мережі за допомогою методу Монте-Карло, навчитися знаходити оптимальні значення параметрів у умовах невизначеності та випадкових впливів.

Ключові положення

Стохастична оптимізація – це оптимізація в умовах випадкових (стохастичних) факторів, коли параметри системи або вхідні дані є випадковими величинами. У телекомунікаційних мережах такими факторами є:

- випадковий вхідний потік пакетів;
- випадкові затримки в каналах;
- випадкові збої обладнання;
- флуктуації навантаження тощо.

Метод Монте-Карло в стохастичній оптимізації полягає у проведенні великої кількості імітаційних експериментів для кожного набору параметрів і виборі того набору, який дає найкращий середній результат (або найкращий результат за заданим критерієм).

Типова постановка: знайти оптимальні значення параметрів мережі (наприклад, розмір буфера, кількість серверів, швидкість каналу, поріг навантаження), які забезпечують мінімум середньої затримки або мінімум ймовірності втрат при випадковому навантаженні.

Практичне завдання

1. Сформулювати задачу стохастичної оптимізації параметрів мережі (наприклад, оптимальний розмір буфера роутера або оптимальна кількість паралельних каналів).
2. Реалізувати імітаційну модель роботи мережі методом Монте-Карло (використовуючи розробки з попередніх робіт).

3. Визначити діапазон варіювання параметра, що оптимізується.

4. Провести серію імітаційних експериментів (500–2000 прогонів) для кожного значення параметра.

5. Обчислити середні значення ключових характеристик: середню затримку, ймовірність втрат пакетів, коефіцієнт використання каналів.

6. Побудувати графіки залежності середньої затримки та ймовірності втрат від значення оптимізованого параметра.

7. Знайти оптимальне значення параметра за критерієм мінімуму середньої затримки при обмеженні на ймовірність втрат.

8. Провести багатофакторну стохастичну оптимізацію (одночасно два параметри).

9. Порівняти результати стохастичної оптимізації з детермінованими розрахунками.

10. Зробити висновки щодо стійкості оптимальних параметрів до випадкових впливів.

Методичні вказівки до виконання практичного завдання

Виконуйте роботу в Scilab. Основна структура програми:

1. Зовнішній цикл – по значеннях оптимізованого параметра (наприклад, розмір буфера від 5 до 50).

2. Внутрішній цикл Монте-Карло – велика кількість імітаційних прогонів (N = 1000–3000).

3. У кожному прогоні – дискретно-подійна імітація мережі протягом заданого модельного часу.

4. Накопичення статистики (середня затримка, втрати тощо).

5. Обчислення середніх значень та довірчих інтервалів.

Рекомендована задача для роботи:

– Оптимізувати розмір буфера маршрутизатора при випадковому пуассонівському навантаженні.

– Критерій: мінімізація середньої затримки при обмеженні ймовірності втрат пакетів $\leq 1–2\%$.

Використовуйте функцію grand() для генерації випадкових величин. Для прискорення обчислень використовуйте векторизацію там, де можливо.

Обов’язково побудуйте графіки з довірчими інтервалами (середнє ± стандартне відхилення).

Контрольні питання

1. У чому полягає сутність стохастичної оптимізації?

2. Чому метод Монте-Карло ефективний для стохастичних задач?

3. Які параметри мережі доцільно оптимізувати стохастичними методами?

4. Як оцінюється якість рішення в стохастичній оптимізації?

5. Як впливає кількість імітаційних прогонів на точність результату?

6. У чому відмінність стохастичної оптимізації від детермінованої?

7. Як враховувати обмеження (наприклад, на ймовірність втрат) при стохастичній оптимізації?

8. Які труднощі виникають при стохастичній оптимізації реальних мереж?

9. Як інтерпретувати графіки залежності характеристик від оптимізованого параметра?

10. Де в сучасних телекомунікаціях застосовується стохастична оптимізація?

Зміст звіту

– назва практичної роботи;
– мета виконання;
– математична постановка задачі стохастичної оптимізації;
– опис імітаційної моделі;
– лістинг програми з коментарями;
– таблиці результатів для різних значень параметра;
– графіки залежностей з довірчими інтервалами;
– оптимальне значення параметра та відповідні характеристики мережі;
– порівняння зі звичайною (детермінованою) оптимізацією;
– висновки щодо практичного застосування.

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

Основна

1. Томашевський В.М. Моделювання систем. Підручнік / В.М. Томашевський.- К.: Видавнича група BHV, 2015. - 352с.
2. Awad M., Khanna R. Efficient learning machines: theories, concepts, and applications for engineers and system designers. eBook: Springer, 2015. 263 p.
3. Desfray P., Raymond G. Modeling enterprise architecture with TOGAF: A practical guide using UML and BPMN. Waltham: Elsevier, 2014. 285 p.
4. Ptolemaeus C. System design, modeling, and simulation using Ptolemy: 1st ed. Mountain View: Ptolemy, 2014. 690 p.
5. Fouss F., Saerens M., Shimbo M. Algorithms and models for network data and link analysis. Cambridge: Cambridge University Press, 2016. 547 p.
6. Комп'ютерне моделювання систем та процесів. Методи обчислень. Частина 1 / Р. Н. Квєтний та ін. Вінниця: ВНТУ, 2012. 193 с.
7. Комп'ютерне моделювання систем та процесів. Методи обчислень. Частина 2 / Р. Н. Квєтний та ін. Вінниця: ВНТУ, 2012. 193 с.
8. Стеценко І .В. Моделювання систем: навч. посіб. [Електронний ресурс, текст]. Черкаси: ЧДТУ, 2010. 399 с.
9. Хусаінов Д. Я., Харченко І. І., Шатирко А. В. Введення в моделювання динамічних систем: навч. посіб. Київ: КНУ, 2010. 132 с.
10. Чуйко Г. П., Дворник О. В., Яремчук О. М. Математичне моделювання систем і процесів: навч. посіб. Миколаїв: Вид-во ЧДУ імені Петра Могили, 2015. 244 с.

Допоміжна

1. Baudin M. Programming in Scilab. eBook: DIGITEO, 2011. 155 p.
2. Bellomo N., De Angelis E., Delitala M. Lecture notes on mathematical modelling from applied sciences to complex systems. Roma: SIMAI, 2010. 171 p.
3. GNU Octave documentation: 5th ed. / J.W. Eaton et al. Boston: Free Software Foundation, Inc., 2020. 1094 p.
4. Задачин В. М., Конюшенко І. Г. Моделювання систем: конспект лекцій. Харків: ХНЕУ, 2010. 268 с.
5. Павлов В. А., Носовець О. К. Методичні вказівки до виконання комп'ютерних практикумів з навчальної дисципліни «Моделювання систем». Київ: НТУУ «КПІ ім. І. Сікорського», 2017. 61 с.

Інформаційні ресурси

1. GNU Octave. Scientific programming language [Електронний ресурс] // GNU General Public License. URL: https://www.gnu.org/software/octave/. Дата звернення: 08.08.2025

2. Scilab tutorials [Електронний ресурс] // GNU General Public License (GPL) v2.0. URL: https://www.scilab.org/tutorials

3. Scicos: Block diagram modeler/simulator [Електронний ресурс] // INRIA. URL: https://www.rocq.inria.fr/scicos/downloads.html

4. Simulink for system modeling and simulation [Електронний ресурс] // The MathWorks, Inc. URL: https://www.mathworks.com/solutions/system-design-simulation.html

Навчальне видання

Розенвассер Денис Михайлович

МОДЕЛЮВАННЯ СИСТЕМ

Методичні вказівки до практичних робіт для здобувачів вищої освіти,
які навчаються за спеціальностями галузі «Інформаційні технології»
та за спеціальністю «Електронні комунікації та радіотехніка»

Українською мовою