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

Розенвассер Д.М.

МОДЕЛЮВАННЯ СИСТЕМ

Конспект лекцій

Одеса 2025

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

Розенвассер Д.М. Моделювання систем. Конспект лекцій [Електронне видання]. Кафедра комп’ютерних наук Міжнародного гуманітарного університету. Одеса, 2025. 41 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.

На курсі розглядаються методи системного моделювання та їх застосування в системному інжинірингу; методи концептуального проектування; функціональний аналіз технічної системи; архітектурне моделювання технічної системи; синтез технічної системи. Для виконання завдань курсу здобувачі мають змогу використовувати сучасне відкрите програмне забезпечення GNU Octave та Scilab.

ЗМІСТ

Тема 1. Наука моделювання

1.1. Предмет, зміст і задачі дисципліни

Дисципліна «Моделювання систем» допомагає зрозуміти, як за допомогою математики та комп'ютера описувати, вивчати й покращувати роботу складних технічних об'єктів – від окремих пристроїв до великих телекомунікаційних мереж.

Предмет дисципліни – це математичні моделі систем і процесів, способи їх створення, перевірки й використання в реальній професійній роботі. Особливо важливі моделі для електронних комунікацій та радіотехніки: моделі передачі сигналів, роботи мереж, черг повідомлень, навантаження каналів, затримок, пропускної здатності тощо. Ми вчимося перетворювати реальні проблеми (наприклад, «як зменшити затримку в мережі при заданій кількості користувачів») у математичні задачі, які можна розв'язати на комп'ютері.

Зміст курсу побудовано так, щоб поступово перейти від основ до практичного застосування саме в телекомунікаціях.

Спочатку ви ознайомитеся з тим, що таке моделювання взагалі, які бувають його види, як ставити задачу математичного програмування (лінійне, нелінійне), як працювати з матрицями, векторами, системами рівнянь, похідними та границями – це фундамент, без якого неможливо рухатися далі.

Потім переходите до комп'ютерного моделювання: вчитеся програмувати в GNU Octave або Scilab (це безкоштовні аналоги MATLAB), писати функції, використовувати цикли й умови, генерувати випадкові числа, будувати графіки (2D і 3D), чисельно інтегрувати функції, розв'язувати диференціальні рівняння символьно й чисельно. Саме ці інструменти ви будете використовувати протягом усього курсу.

Далі йде найважливіша для вашої спеціальності частина – моделювання систем і мереж. Ви навчитеся будувати моделі за чіткою технологією, створювати логіко-арифметичні моделі, описувати динамічні процеси, моделювати прикладні системи, а головне – телекомунікаційні системи та мережі: канали зв'язку, черги пакетів, маршрутизацію, навантаження, випадкові процеси в мережах.

Останній блок – оптимізація. Тут розглядаються методи знаходження найкращих рішень: як поставити задачу оптимізації, як записати цільову функцію й обмеження, як розв'язувати задачі лінійного, нелінійного, цілочисельного та стохастичного програмування. Усе це відразу застосовується до реальних задач телекомунікацій: максимізація пропускної здатності, мінімізація затримки, оптимальне розміщення базових станцій, розподіл ресурсів тощо.

Мета дисципліни – дати вам уміння самостійно моделювати об’єкти вашої майбутньої професійної діяльності. Після курсу ви зможете:

– створювати математичні та програмні моделі процесів, які відбуваються в мережах і системах зв’язку;

– обирати відповідний тип моделі залежно від задачі;

– реалізовувати модель у Octave/Scilab;

– аналізувати результати, оцінювати їхню точність;

– застосовувати моделі для проектування та вдосконалення телекомунікаційних систем.

Ці знання безпосередньо потрібні для дисциплін «Комп’ютерні мережі», «Телекомунікаційні системи та мережі», а також для дипломної роботи. Курс побудовано так, щоб ви не просто запам’ятовували формули, а вміли на практиці розв’язувати реальні задачі вашої спеціальності за допомогою сучасних безкоштовних інструментів.

Тому основне, що ви отримаєте: розуміння, як «перекладати» реальну проблему зв’язку на мову математики й комп’ютера, а потім знаходити оптимальне рішення. Саме це зараз дуже цінується в IT та телекомунікаційній галузі.

1.2. Класифікація видів моделювання

Моделювання – це створення спрощеного, але достатньо точного опису реального об’єкта, процесу чи явища з метою його вивчення, аналізу, прогнозування або вдосконалення. Щоб систематизувати підходи, моделі класифікують за кількома основними ознаками. Розуміння цієї класифікації допомагає правильно обрати тип моделі для конкретної задачі в телекомунікаціях чи комп’ютерних системах.

За способом представлення реальності моделі поділяються на фізичні, аналогічні та математичні.

Фізичні моделі – це матеріальні копії об’єкта, які відтворюють його в зменшеному або збільшеному масштабі, з тими самими фізичними процесами. Наприклад, макет антени в безеховій камері для вимірювання діаграми спрямованості або зменшений макет базової станції для тестування покриття. Такі моделі дорогі, трудомісткі й рідко застосовуються в сучасних IT-задачах, але іноді незамінні на етапі експериментальної перевірки.

Аналогічні (фізично подібні) моделі базуються на аналогії між явищами різної природи. Наприклад, гідравлічна модель для опису потоків даних у мережі (де вода – це пакети, труби – канали, насоси – джерела трафіку). У телекомунікаціях цей підхід використовується рідше, але іноді допомагає інтуїтивно зрозуміти складні процеси.

Математичні моделі – найпоширеніший і найважливіший для вашої спеціальності тип. Вони описують об’єкт за допомогою математичних рівнянь, нерівностей, графів, алгоритмів чи статистичних законів. Саме з ними ви працюватимете протягом усього курсу в Octave/Scilab. Математичні моделі поділяються на аналітичні та імітаційні.

Аналітичні моделі дозволяють отримати точний або наближений розв’язок у вигляді формули чи числового значення. Наприклад, формула Ерланга для розрахунку ймовірності блокування виклику в системі з втратами або формула для середньої затримки в черзі M/M/1. Перевага – швидкість обчислень і можливість аналізу чутливості. Недолік – часто потребують суттєвих спрощень (наприклад, припущення про пуассонівський потік, експоненціальний час обслуговування).

Імітаційні моделі відтворюють поведінку системи крок за кроком у часі, імітуючи всі події. Вони не дають готової формули, але дозволяють врахувати дуже складні умови, нелінійності, реальні розподіли, адаптивні алгоритми. Наприклад, імітація роботи LTE-мережі з реальними траєкторіями користувачів, перемиканнями між базовими станціями, завадою від сусідніх осередків. У сучасних телекомунікаціях більшість складних задач (особливо оптимізація 5G/6G, IoT-мереж, SDN) розв’язується саме імітаційними моделями.

За характером відображення часу моделі бувають статичні та динамічні.

Статичні моделі описують систему в певний момент часу або в стаціонарному режимі, без урахування змін у часі. Наприклад, задача розміщення базових станцій для максимального покриття або оптимізація маршрутизації в мережі як графова задача.

Динамічні моделі враховують зміну стану системи в часі. Вони описуються диференціальними рівняннями (неперервний час), різницевими рівняннями або дискретними подіями. У телекомунікаціях це моделі передачі пакетів, зміни навантаження, адаптивного кодування, handover тощо.

За ступенем деталізації виділяють концептуальні, функціональні, структурні та деталізовані моделі.

Концептуальна модель – найвищий рівень абстракції: визначає, що саме моделюється, які основні елементи та зв’язки між ними. Наприклад, блок-схема телекомунікаційної системи «джерело – канал – приймач – завада».

Функціональна модель описує, що саме робить система (вхід – функція – вихід), без деталей реалізації.

Структурна модель показує, з яких компонентів складається система і як вони з’єднані.

Деталізована модель включає повний опис алгоритмів, параметрів, фізичних характеристик.

За характером інформації моделі поділяються на детерміновані та стохастичні.

Детерміновані моделі не враховують випадковість: при однакових вхідних даних завжди однаковий результат. Наприклад, детермінована модель поширення сигналу в ідеальному каналі без шуму.

Стохастичні моделі обов'язково включають випадкові процеси: пуассонівські потоки викликів, експоненціальний час обслуговування, завади з гауссівським розподілом, випадкові рухи абонентів. Більшість реальних телекомунікаційних моделей – стохастичні.

У вашому курсі основний акцент саме на математичних моделях, переважно стохастичних і динамічних, з великою часткою імітаційних підходів. Аналітичні моделі вивчаються для розуміння фундаментальних закономірностей, а імітаційні – для розв'язання практичних, складних задач проектування мереж, де аналітичний розв'язок неможливий або надто грубий.

Знання цієї класифікації дозволяє вже на етапі постановки задачі вирішити: чи можна скористатися простою формулою (аналітична модель), чи потрібна повноцінна імітація в Octave/Scilab (імітаційна стохастична динамічна модель). Це один з ключових навичок системного інженера в галузі електронних комунікацій.

1.3. Етапи математичного моделювання

Математичне моделювання – це систематичний процес, який перетворює реальну проблему на математичну задачу, розв'язує її та повертає результат назад у реальний світ. У вашій спеціальності (електронні комунікації та радіотехніка) цей процес застосовується до задач аналізу мереж, оцінки якості зв'язку, оптимізації ресурсів, прогнозування навантаження тощо. Усі етапи виконуються послідовно, але часто з поверненнями назад, якщо щось не влаштовує.

Перший етап – постановка задачі та якісний аналіз об'єкта.

Ви чітко формулюєте, що саме потрібно зробити: наприклад, «визначити середню затримку пакетів у мережі при заданому трафіку», «знайти оптимальне розміщення 5 базових станцій для покриття району з мінімальними витратами» або «оцінити ймовірність втрати пакета в каналі з перешкодами». На цьому етапі ви вивчаєте реальний об'єкт: збираєте дані про його структуру, параметри, умови роботи, обмеження, мету. Важливо зрозуміти фізичну сутність процесу, основні фактори впливу та що можна спростити без великої втрати точності. Результат – чітка словесна постановка задачі з вказівкою мети, вхідних даних, вихідних показників і обмежень.

Другий етап – побудова концептуальної моделі.

Ви створюєте спрощений опис системи на рівні блоків і зв’язків, без формул. Це може бути блок-схема, діаграма потоків даних, граф мережі, схема черг тощо. Наприклад, для моделі телекомунікаційної мережі ви визначаєте джерела трафіку, вузли комутації, канали передачі, черги, механізми управління доступом. Концептуальна модель показує, які елементи суттєві, а які можна ігнорувати. На цьому етапі часто використовують TOGAF, UML або прості блок-діаграми.

Третій етап – формалізація, тобто перехід до математичної моделі.

Тут ви записуєте всі зв’язки у вигляді рівнянь, нерівностей, графів, імовірнісних законів. Вибираєте тип моделі: детермінована чи стохастична, аналітична чи імітаційна, статична чи динамічна. Наприклад, для черги в базовій станції ви можете записати модель M/M/1 (пуассонівський вхід, експоненціальний час обслуговування, один сервер) або більш складну модель з обмеженою буферною пам’яттю. Визначаєте змінні, параметри, цільову функцію (якщо оптимізація), обмеження. Результат – математична постановка задачі, яку можна програмувати.

Четвертий етап – вибір методу розв’язання та реалізація моделі.

Ви обираєте, як саме розв’язувати задачу: аналітичний метод (формула Ерланга, формули Кендалла), чисельний метод (інтегрування диференціальних рівнянь), імітаційний експеримент (метод Монте-Карло), оптимізаційний алгоритм (симплекс-метод, генетичні алгоритми, градієнтний спуск). Потім реалізуєте модель у GNU Octave або Scilab: пишете код, задаєте параметри, готуєте функції для генерації трафіку, обчислення показників, побудови графіків.

П’ятий етап – проведення обчислень та отримання результатів.

Ви запускаєте модель для різних наборів вхідних даних, змінюєте параметри, проводите серію експериментів. Наприклад, будуєте залежність затримки від інтенсивності трафіку, ймовірності втрати від розміру буфера, пропускної здатності від кількості антен тощо. На цьому етапі дуже важлива правильна генерація псевдовипадкових чисел і забезпечення відтворюваності результатів (random.seed).

Шостий етап – верифікація та валідація моделі.

Верифікація – перевірка, чи правильно реалізована модель (чи код відповідає математичній постановці, чи немає помилок у логіці чи алгоритмах). Валідація – перевірка, чи адекватно модель описує реальність (порівняння з реальними вимірами, даними літератури, відомими аналітичними формулами для простих випадків). Якщо розбіжності великі – повертаєтеся до попередніх етапів: спрощуєте менше, додаєте фактори, змінюєте розподіли, уточнюєте параметри.

Сьомий етап – інтерпретація результатів та використання.

Ви аналізуєте отримані дані, робите висновки, пропонуєте рекомендації: наприклад, «при навантаженні понад 0,8 ерланга затримка зростає експоненційно – потрібно збільшити буфер або додати сервер», «оптимальна кількість базових станцій – 7 при бюджеті 500 тис. грн». Результати візуалізуєте (графіки, теплові карти, діаграми), готуєте звіт або презентацію.

У реальних проектах етапи не завжди йдуть строго послідовно: після валідації часто повертаються до формалізації або навіть до постановки задачі, якщо виявилося, що мета сформульована неправильно. У телекомунікаціях особливо багато ітерацій, бо реальні мережі мають величезну кількість випадкових факторів і нелінійностей.

Цей цикл – основа всієї дисципліни. Кожен ваш практичний чи самостійний роботу буде проходити саме через ці етапи: від реальної задачі зв'язку до коду в Octave/Scilab і висновків. Чим краще ви засвоїте цю послідовність, тим швидше й точніше зможете розв'язувати професійні задачі.

1.4. Задачі математичного програмування

Математичне програмування – це розділ математики, який займається пошуком найкращого (оптимального) рішення задачі за певних обмежень. У вашій спеціальності це один з найважливіших інструментів: майже кожна реальна задача в телекомунікаціях зводиться до того, щоб знайти найкращий розподіл ресурсів, розміщення обладнання, маршрутизацію трафіку, налаштування параметрів мережі тощо при заданих обмеженнях на бюджет, потужність, затримку, енергію, пропускну здатність.

Задача математичного програмування завжди має однакову структуру:

- Цільова функція (що ми хочемо максимізувати або мінімізувати).
- Змінні рішення (що ми можемо змінювати).
- Обмеження (умови, яких обов'язково треба дотримуватися).
- Область допустимих значень змінних (наприклад, невід'ємні, цілі, бінарні).

Найпоширеніші класи задач математичного програмування, які ви зустрінете в курсі та в професійній роботі.

Лінійне програмування (ЛП) – найпростіший і найчастіше використовуваний клас. Усі функції (цільова і обмеження) – лінійні, тобто мають вигляд суми коефіцієнтів помножених на змінні. Змінні зазвичай невід'ємні або вільні. Приклади в телекомунікаціях:

- максимізувати пропускну здатність мережі при обмеженнях на потужність передавачів;
- мінімізувати загальну вартість розміщення базових станцій при вимозі покриття не менше 95% території;

– оптимальний розподіл частот між осередками без взаємних перешкод (задача розфарбовування графу як лінійна);

– розподіл трафіку по шляхах у мережах MPLS або SDN. Для розв'язання використовується симплекс-метод (або його модифікації), який ви будете вивчати на практичних заняттях в Octave/Scilab.

Методи математичного моделювання

Метод	Опис	Область застосування	Переваги методу	Недоліки методу
Лінійне програмування	Складається система лінійних рівнянь з необхідними обмеженням. Типові завдання - складання рідких сумішей, розподілу ресурсів	Не складні лінійні детерміновані задачі (із заздалегідь точно відомими результатами тієї чи іншої стратегії).	Простота, можливість швидко отримати рішення без застосування ЕОМ. При адекватній постановці завдання забезпечується необхідна точність рішення.	Вузьке коло вирішуваних завдань - лише мала частина реальних процесів лінійна, в інших випадках виникає зайва апроксимація.
Нелінійне програмування	Складається система нелінійних рівнянь з необхідними обмеженням. Типові завдання - управління виробничим процесом, виручка від реалізації продукції.	Завдання із заданими нелінійними співвідношеннями змінних.	Можливість задавати залежності змінних - наприклад вплив обсягу продажів на ціну.	Складність рішення. Необхідність великого числа даних.
Динамічне програмування	Як правило, складається мережева модель. Приклад - модель розподілу зусиль.	Багатоетапні задачі. Результатом виконання завдання є рекурентний вираз, що виражає кроки, які слід приймати на будь-якому етапі.	Дозволяють приймати правильне рішення безліч разів без втручання людини (на основі рекурентного виразу) - тобто з допомогою ЕОМ.	Вузьке коло вирішуваних завдань, складність складання рекуррентного виразу.
Мережеві задачі	Мережеві задачі - окремий випадок задач лінійного програмування. Для опису моделі використовується граф. Приклад - транспортна задача.	Мережевий метод зручний для графічного опису оптимізаційних задач і може застосовуватися для складних (тисячі змінних і сотні обмежень) завдань.	Графічний механізм зручний для опису завдань лінійного програмування. Можливість враховувати при вирішенні транспортної задачі сезонності, пропускної здатності, змінної потужності постачальників і т.п.	Більшість завдань, що вирішуються даним методом є варіаціями транспортної задачі.
Імовірнісні оптимізаційні моделі	Метод, що враховує вірогідну компоненту. Включає в себе імовірнісні моделі управління запасами і системи масового обслуговування.	Будь-які моделі, для опису яких потрібні випадкові величини.	Замість зайвого ускладнення моделі вводяться ймовірності подій, що дозволяє вирішувати складні завдання, які не вирішуються іншими методами.	Складність рішення без ЕОМ.
Цілочисельне програмування	Значне число оптимізаційних задач мають обмеження в целочисленном вирішенні. Для їх вирішення потрібні особливі алгоритми. Приклад – задача комівояжера.	Коло задач, що вимагає цілочисельного рішення.	Метод дозволяє вирішувати комбінаторні задачі.	Всі недоліки вирішуваного класу задач.
Імітаційне моделювання	Застосовується на ЕОМ. З огляду на швидкість розрахунків можуть вирішувати завдання практично будь-якої складності.	Клас задач, що не можна вирішити іншими методами.	Широке коло вирішуваних завдань.	Складність опису всіх умов і вимоги обчислювальної потужності.

Нелінійне програмування (НЛП) виникає, коли хоча б одна з функцій (цільова чи обмеження) нелінійна. Це значно складніше, бо немає гарантії глобального оптимуму, часто знаходимо лише локальний. Приклади:

– оптимізація потужності передавачів для максимізації якості сигналу (SINR) з урахуванням нелінійної залежності втрат від відстані;

– мінімізація енергоспоживання мережі при нелінійних моделях підсилювачів;

– налаштування параметрів адаптивного кодування та модуляції (MCS) в 5G/6G. Методи розв’язання: градієнтний спуск, метод Ньютона, генетичні алгоритми, роїнні методи (PSO), які легко реалізувати в Octave.

Цілочисельне програмування (ЦП) – коли деякі або всі змінні рішення мають бути цілими числами. Найчастіше зустрічаються бінарні змінні (0 або 1). Приклади:

– чи встановлювати базову станцію в даній точці (0/1);

– скільки каналів виділити кожному користувачеві;

– вибір маршруту в мережі (бінарні змінні для ребер);

– задача призначення частот (frequency assignment problem). Це один з найскладніших класів, бо навіть невеликі задачі можуть бути NP-важкими. Використовуються методи відсікання та гілок-меж, релаксація (спочатку розв’язуємо як ЛП, потім округляємо), евристики.

Змішане цілочисельне програмування (ЗЦП) – комбінація: частина змінних неперервні, частина цілі. Дуже поширене в телекомунікаціях. Приклад: оптимальне розміщення базових станцій (бінарні – ставити/не ставити) + розподіл потужності (неперервна змінна) при обмеженнях на інтерференцію.

Стохастичне програмування – коли параметри задачі невідомі точно, а відомі лише їх ймовірнісні характеристики (наприклад, випадкове навантаження мережі, випадкові перешкоди). Види:

– з рекурсією (two-stage) – спочатку приймаємо рішення, потім реалізується випадковість і коригуємо;

– chance-constrained – ймовірність порушення обмежень не більше заданого рівня;

– стохастична оптимізація з Монте-Карло. Приклади: планування ресурсів у мережі з випадковим трафіком користувачів, оптимальне резервування каналів при випадкових відмовах.

Багатокритеріальна оптимізація – коли є кілька конфліктуючих цілей одночасно (наприклад, максимізувати пропускну здатність і мінімізувати затримку, або максимізувати покриття і мінімізувати витрати). Розв’язання: метод зваженої суми, метод ε-обмежень, Парето-оптимальність, еволюційні алгоритми.

У вашому курсі основна увага приділяється лінійному та нелінійному програмуванню, а також простим стохастичним підходам. Ви навчитеся формулювати задачі саме в цій формі, реалізовувати їх в Octave/Scilab (наприклад, функція linprog для ЛП, fmincon-подібні для НЛП) і аналізувати результати. Розуміння цих класів задач дозволяє вже на етапі постановки проблеми вирішити, який інструмент застосовувати: чи вистачить симплекс-

методу, чи потрібна імітація з Монте-Карло, чи доведеться використовувати генетичні алгоритми для складних цілочисельних задач.

Це основа для теми 4 курсу («Оптимізація систем та мереж»), де ви будете безпосередньо розв’язувати реальні телекомунікаційні задачі оптимізації.

Тема 2. Комп’ютерне моделювання

2.1. Середовища моделювання

У дисципліні «Моделювання систем» для практичної реалізації математичних моделей використовуються два основні відкриті (безкоштовні) середовища: GNU Octave та Scilab. Вони спеціально рекомендовані робочою програмою, бо дозволяють виконувати всі необхідні обчислення, графіку, оптимізацію та імітацію без придбання комерційного програмного забезпечення на кшталт MATLAB. Обидва інструменти працюють на Windows, Linux та macOS, мають схожий синтаксис, орієнтований на матричні обчислення, і активно розвиваються.

GNU Octave – це відкритий аналог MATLAB, створений з метою максимальної сумісності з ним. Його синтаксис майже ідентичний MATLAB: більшість скриптів, написаних для MATLAB (без спеціалізованих toolbox-ів), запускаються в Octave без змін або з мінімальними правками. Це робить Octave ідеальним вибором, якщо ви плануєте працювати з кодами, які можуть бути перенесені в MATLAB у майбутньому (наприклад, на роботі в компаніях, де використовують комерційну версію). Octave має потужний інтерпретатор для матриць і векторів, вбудовану 2D/3D-графіку, інструменти для розв’язання рівнянь, оптимізації, статистики, символьних обчислень (через пакет symbolic), генерації випадкових чисел, чисельного інтегрування та розв’язання диференціальних рівнянь. У 2025–2026 роках вийшли версії 10.x та 11, де покращено сумісність з MATLAB, графічний інтерфейс (GUI), backend графіки, додані нові функції пошуку пакетів, оптимізована робота з Java та пам’яттю. Octave підтримує пакети (як у MATLAB – через pkg install), наприклад, control, signal, communications, optimization, statistics, що безпосередньо потрібні для моделювання телекомунікаційних систем і мереж.

Scilab – ще один потужний відкритий інструмент для наукових обчислень, розроблений спеціально для інженерних задач. Його синтаксис близький до MATLAB, але з деякими відмінностями (наприклад, індексація з 1, інші назви функцій у деяких випадках). Scilab не прагне 100% сумісності з MATLAB, зате має власні сильні сторони: вбудований графічний редактор моделей Xcos (аналог Simulink для блокових діаграм), який ідеально підходить для моделювання динамічних систем, черг, телекомунікаційних процесів, керування. У версіях 2025.0.0 та новіших додано lambda-функції, новий синтаксис для комплексних чисел, функцію kmeans для кластеризації, vander для матриць Вандермонда, покращено графіку (анти-аліасинг за замовчуванням, повернення handle-ів від plotting-функцій, підтримка формул у текстах графіків). Scilab швидший у деяких паралельних обчисленнях, має добру продуктивність для великих матриць і добре інтегрується з іншими

мовами. Він особливо зручний для імітаційного моделювання складних систем, де потрібні блокові схеми.

Обидва середовища дозволяють:

– працювати з матрицями та векторами як з основними об’єктами;

– будувати 2D/3D-графіки для візуалізації результатів (plot, surf, contour тощо);

– використовувати оператори керування (if, for, while), функції, псевдовипадкові числа (rand, randn);

– розв’язувати системи рівнянь, диференціальні рівняння (odepkg в Octave, ode в Scilab);

– виконувати оптимізацію (linprog, fminsearch, ga-подібні);

– проводити імітацію методом Монте-Карло.

У курсі ви можете обрати будь-яке з них залежно від уподобань: Octave – якщо важлива максимальна сумісність з MATLAB і простота переходу; Scilab – якщо плануєте використовувати Xcos для блокового моделювання динамічних систем і мереж. Обидва встановлюються безкоштовно з офіційних сайтів (octave.org та scilab.org), мають документацію українською/англійською, велику кількість прикладів і спільноту. На практичних заняттях ви будете писати код саме в одному з цих середовищ, щоб моделювати процеси в телекомунікаціях: трафік, черги, затримки, оптимізацію ресурсів тощо.

Головне – освоїти базовий синтаксис (матричні операції, функції, графіку), бо саме на ньому будується все подальше комп’ютерне моделювання в дисципліні. Після курсу ви зможете самостійно використовувати ці інструменти для наукових розрахунків, курсових, дипломної роботи чи професійних задач.

2.2. Основи програмування в Octave/Scilab

Octave і Scilab мають дуже схожий синтаксис, орієнтований на матричні обчислення та наукові розрахунки, тому основи програмування в обох середовищах майже однакові. Ви можете використовувати будь-яке з них для курсу – код, який ви напишете, буде працювати в обох з мінімальними правками. Головна відмінність для початківця: Octave ближчий до MATLAB (майже повна сумісність), Scilab має вбудований Xcos для блокового моделювання, але базовий скриптовий стиль однаковий.

Програма запускається в консолі (командному рядку) або через редактор скриптів (.m-файли в Octave, .sce або .sci в Scilab). Усі команди можна вводити безпосередньо в консоль для швидких перевірок або писати в файл і запускати.

Змінні створюються простим присвоєнням без попереднього оголошення типу. Імена чутливі до регістру, наприклад: a = 5; A = [1 2 3; 4 5 6]; (матриця 2×3) pi – вбудована константа, ans – автоматична змінна для останнього результату без ;.

Матриці та вектори – основа мови. Створюються квадратними дужками, елементи розділяються пробілами або комами, рядки – крапкою з комою: v = [1 2 3 4]; (рядковий вектор) w = [1; 2; 3]; (стовпцевий вектор) M = [1 2; 3 4];

Доступ до елементів: M(1,2) – елемент першого рядка, другого стовпця. M(:,2) – весь другий стовпець. M(2,:) – другий рядок. M(end) – останній елемент.

Арифметичні операції: + - * / ^ для матриць працюють поелементно, якщо додати крапку: .+ .- .* ./ .^ Наприклад: A .* B – поелементне множення. Матричне множення: A * B. Транспонування: A' або A.' (для комплексних – Hermitian).

Коментарі починаються з % (Octave) або // (Scilab). % Це коментар // Це теж коментар

Виведення: disp(x) – виводить значення змінної. printf("Значення = %f\n", x) – форматований вивід. Крапка з комою ; в кінці рядка приховує вивід у консоль.

Введення даних: x = input("Введіть число: "); – запитує значення з консолі.

Оператори керування: Умовний оператор: if умова команди elseif інша_умова команди else команди end

Цикли: for i = 1:10 disp(i); end

while умова команди if щось, break; end (вихід з циклу) if щось, continue; end (наступна ітерація) end

Функції генерації випадкових чисел: rand() – рівномірний розподіл [0,1] rand(n,m) – матриця n×m випадкових чисел randn() – нормальний (гауссів) розподіл rand("seed", значення) або rand("state", значення) – установка початкового стану для відтворюваності (дуже важливо для імітації!)

Графіка – проста й потужна: x = 0:0.1:10; y = sin(x); plot(x, y); – базовий графік plot(x, y, "r--", "linewidth", 2); – червона пунктирна лінія, товщина 2 title("Синусоїда"); xlabel("Час"); ylabel("Амплітуда"); grid on; Для кількох графіків: hold on; або plot(x,y1,x,y2); 3D: mesh(X,Y,Z); або surf(X,Y,Z);

Щоб створити функцію, пишіть окремий файл, наприклад sin_func.m: function y = sin_func(t) y = sin(t) + 0.5*cos(2*t); endfunction Потім викликайте: sin_func(pi/2)

Скрипти (.m або .sce) просто виконуються командою run("ім'я_файлу") або натисканням кнопки в редакторі.

Для очищення: clear – видаляє всі змінні, clc – очищає консоль, clf – очищає графічне вікно.

Ці базові елементи – основа всього комп’ютерного моделювання в курсі. На практичних ви будете використовувати саме їх для створення моделей: генерація трафіку (rand), цикли для імітації подій, графіки для візуалізації затримок чи пропускної здатності, функції для повторного використання коду. Почніть з простих скриптів у консолі, потім переходьте до файлів – і швидко освоїте середовище для задач телекомунікацій.

2.3. Генерація псевдовипадкових чисел

Генерація псевдовипадкових чисел – ключовий елемент імітаційного моделювання в телекомунікаціях. Більшість реальних процесів у мережах (прихід пакетів, час обслуговування, завади, рухи абонентів, помилки в каналі) мають випадковий характер. Щоб імітувати їх на комп’ютері, ми використовуємо псевдовипадкові послідовності – числа, які виглядають випадковими, але генеруються детерміновано за певним алгоритмом. Це дозволяє відтворювати експерименти: запустивши модель з тим самим початковим станом, ви отримаєте точно ті самі «випадкові» події.

У GNU Octave та Scilab основні функції для генерації – rand і randn.

Функція rand генерує числа з рівномірного розподілу на інтервалі [0, 1).

- rand() – одне число
- rand(5) – матриця 5×5
- rand(3,4) – матриця 3×4
- rand(n,m,"single") – тип даних single (рідко потрібен)

Щоб отримати числа в іншому діапазоні, просто масштабуємо та зсуваємо: uniform = a + (b - a) * rand(); – рівномірний розподіл на [a, b]

Для дискретних величин (наприклад, кількість пакетів): poisson = poissrnd(lambda); (в Octave через пакет statistics, в Scilab – grand("poi", lambda))

Функція **randn** генерує числа зі стандартного нормального (гауссівського) розподілу – середнє 0, дисперсія 1.

- randn() – одне число
- randn(1000,1) – 1000 значень для гістограми шуму

Це ідеально для моделювання адитивного гауссівського шуму в каналі зв’язку: noise = sigma * randn(size(signal)); received = signal + noise;

Щоб забезпечити відтворюваність результатів (дуже важливо для налагодження, порівняння моделей, наукових публікацій), потрібно фіксувати початковий стан генератора – seed або state.

У GNU Octave рекомендований спосіб – rand("seed", значення); – встановлює простий цілочисельний seed (старий, але простий спосіб) Або повний стан: rand("state", v); – де v –

вектор 625 елементів (повний стан Mersenne Twister) v = rand("state"); – отримати поточний стан rand("state", "reset"); – скинути до заводського стану

Найпростіший і найчастіше використовуваний у курсі: rand("seed", 12345); – тоді rand завжди видаватиме одну й ту саму послідовність при кожному запуску скрипту.

У Scilab аналогічно: rand("seed", значення); або rand("setsd", sd); (grand для розширених генераторів) Для нормального: rand("normal"); перемикає генератор на нормальний, але seed встановлюється так само. grand("setsd", sd); – для деяких розподілів (poisson, exponential тощо).

Найкраща практика для імітаційного моделювання в курсі:

1. На початку скрипту завжди ставте фіксований seed: rand("seed", 42); (або будь-яке число, наприклад, рік або номер студента)
2. Якщо потрібно кілька незалежних послідовностей – змінюйте seed для кожної серії експериментів.
3. Для порівняння різних моделей – використовуйте той самий seed, щоб «випадкові» події були однаковими.

Приклади застосування в телекомунікаціях:

- Моделювання пуассонівського потоку викликів: кількість подій за інтервал = poissrnd(lambda * T)
- Час між прибуттями пакетів: експоненціальний розподіл – -log(rand()) / lambda
- Помилки в каналі з BER: помилка = (rand() < BER)
- Рух користувача (модель random waypoint): координати = rand() * область
- Метод Монте-Карло для оцінки ймовірності блокування: багаторазово генеруємо трафік rand і рахуємо частку втрат.

Без фіксації seed результати будуть різними при кожному запуску – це добре для фінальної оцінки, але погано для налагодження та порівняння. Завжди фіксуйте seed на початку, а для статистики запускайте багато реалізацій (наприклад, 1000–10000) і усереднюйте результати.

Освоївши rand, randn і seed, ви зможете реалізовувати повноцінні стохастичні моделі черг, мереж, каналів – саме те, що потрібно для тем 3 і 4 курсу. На практичних заняттях ви будете використовувати ці функції для імітації трафіку, черг M/M/1, оцінки затримок і втрат у мережах.

2.4. Графіка

Візуалізація результатів – один з найважливіших етапів моделювання. У GNU Octave та Scilab графіка дозволяє швидко побачити залежності, виявити закономірності, порівняти

сценарії та представити результати зрозуміло. Обидва середовища мають потужні вбудовані інструменти для 2D- і 3D-графіків, які використовуються майже в кожній практичній роботі курсу: побудова залежності затримки від навантаження, гістограми часу обслуговування, поверхонь пропускної здатності, діаграм розподілу сигналу тощо.

Основна функція для 2D-графіків – **plot**. Вона найпростіша й найчастіше використовувана.

Приклади базового використання:

x = 0:0.1:10; y = sin(x); plot(x, y);

Це побудує синусоїду. Щоб налаштувати вигляд: plot(x, y, "r--", "linewidth", 2, "markersize", 8);

- "r--" – червона пунктирна лінія
- "linewidth", 2 – товщина лінії
- "markersize", 8 – розмір маркерів (якщо є)

Кілька кривих на одному графіку: y1 = sin(x); y2 = cos(x); plot(x, y1, "b-", x, y2, "g--"); Або через hold on: plot(x, y1, "b-"); hold on; plot(x, y2, "g--"); hold off;

Підписи та оформлення: title("Залежність затримки від навантаження"); xlabel("Інтенсивність трафіку, ерланг"); ylabel("Середня затримка, мс"); grid on; legend("M/M/1", "M/D/1", "location", "northwest");

Для логарифмічних осей: semilogx(x, y); – логарифм по X semilogy(x, y); – логарифм по Y loglog(x, y); – логарифм по обох осях (дуже зручно для BER vs SNR)

Гістограми – для аналізу розподілів (наприклад, час між пакетами): data = -log(rand(10000,1))/lambda; % експоненціальний розподіл hist(data, 50); % 50 стовпчиків Або сучасніший варіант: histogram(data, "Normalization", "pdf"); % для порівняння з теоретичною щільністю

Точкові графіки (scatter): scatter(x, y, 20, "filled"); – точки розміром 20, заповнені scatter(x, y, 30, c, "filled"); – колір залежить від вектора c

3D-графіки потрібні для поверхонь (наприклад, пропускна здатність від двох параметрів): [X, Y] = meshgrid(0:0.1:5, 0:0.1:5); Z = X .* Y ./ (X + Y + 1); % приклад функції mesh(X, Y, Z); % сітка surf(X, Y, Z); % кольорова поверхня shading interp; % плавне забарвлення colorbar;

Контурні графіки: contour(X, Y, Z, 20); – 20 рівнів contourf(X, Y, Z); – заповнені контури

Інші корисні функції: bar(x, y); – стовпчикові діаграми (наприклад, ймовірність блокування для різних навантажень) stairs(x, y); – сходинкова функція (для кумулятивних

розподілів) errorbar(x, y, e); – графік з похибками subplot(2,2,1); – кілька графіків в одному вікні figure; – нове вікно clf; – очистити поточне графічне вікно

Збереження графіка: print -dpng "graph.png" -r300; % PNG з роздільністю 300 dpi print -depsc "graph.eps"; % для публікацій

У Scilab додатково є Xcos для блокових діаграм, але для простих графіків plot та surf працюють майже ідентично Octave.

У курсі графіка використовується постійно:

- побудова кривих затримки vs навантаження для різних типів черг;
- гістограми часу обслуговування та міжприбуття;
- поверхні оптимальних параметрів (наприклад, потужність vs відстань vs SINR);
- порівняння теоретичних формул з імітаційними результатами.

Головне правило: завжди додавайте підписи осей, заголовок, легенду та сітку – це робить графік професійним і зрозумілим. Починайте з простих plot, потім додавайте оформлення. На практичних заняттях ви будете будувати десятки графіків саме для аналізу моделей телекомунікаційних систем – це один з найкращих способів зрозуміти, як змінюються характеристики мережі при зміні параметрів.

2.5. Чисельне інтегрування та диференціальні рівняння

Чисельне інтегрування та розв’язання диференціальних рівнянь – це два ключові інструменти для моделювання динамічних процесів у телекомунікаціях. Багато характеристик мереж (середня затримка, пропускна здатність, енергоспоживання, ймовірність помилки) обчислюються через інтеграли, а поведінка систем у часі (зміна навантаження, поширення сигналу, адаптивні алгоритми) описується диференціальними рівняннями. У GNU Octave та Scilab ці задачі розв’язуються вбудованими функціями, які дозволяють швидко отримати точні наближення без аналітичного розв’язку.

Чисельне інтегрування

Чисельне інтегрування потрібне, коли функцію не вдається проінтегрувати аналітично або коли дані задані таблично (наприклад, експериментальні виміри сигналу, трафіку).

Найпростіший метод – трапеціальний (trapz, cumtrapz). Він працює з набором точок і не потребує аналітичного виразу функції: x = 0:0.01:10; y = exp(-x.^2); I = trapz(x, y); % наближений інтеграл від 0 до 10 cumtrapz(x, y) – дає кумулятивний інтеграл (корисний для накопичення енергії сигналу чи трафіку). Це швидкий метод, але менш точний на грубих сітках.

Для точнішого інтегрування функцій (коли є аналітичний вираз) використовуйте адаптивні квадратури: У Octave – quadgk (рекомендований, Gauss-Kronrod, добре працює з

осциляціями та нескінченними інтервалами), quad, quadl, quadcc. Приклад: f = @(x) sin(x)./x; % sinc-функція [I, err] = quadgk(f, 0, pi); disp(I); disp(err); % err – оцінка похибки

У Scilab – intg (адаптивна квадратура, подібна до quad): deff("y=f(x)","y=sin(x)/x"); I = intg(0, %pi, f);

Для табличних даних – inttrap (аналог trapz): v = inttrap(x, s);

Для подвійних/потрійних інтегралів – dblquad, triplequad в Octave або вкладені intg в Scilab.

У телекомунікаціях чисельне інтегрування застосовується для:

– обчислення середньої потужності сигналу (інтеграл квадрата амплітуди);

– розрахунку ймовірності помилки (BER) через інтеграл по гауссівському розподілу;

– оцінки пропускної здатності каналу з шумом (інтеграл по формулі Шеннона);

– накопичення трафіку в чергах.

Розв’язання диференціальних рівнянь

Диференціальні рівняння описують динаміку: зміну стану системи в часі. У курсі найчастіше зустрічаються звичайні диференціальні рівняння (ЗДР) першого порядку або системи ЗДР.

У Octave основні солвери (сумісні з MATLAB):

– lsode – класичний солвер (Adams/BDF, добре для жорстких і нежорстких систем).

– ode45 – Runge-Kutta 4(5) Dormand-Prince, високоточний для нежорстких задач, змінний крок.

– ode23 – Runge-Kutta 2(3) Bogacki-Shampine, швидший, але менш точний.

Синтаксис: function dydt = myode(t, y) dydt = -2*y + sin(t); % приклад: y' = -2y + sin(t) endfunction

[t, y] = ode45(@myode, [0 10], 1); % від t=0 до 10, початкове y(0)=1 plot(t, y);

Для жорстких систем (наприклад, швидкі перехідні процеси в мережах) краще ode15s, ode23s.

У Scilab основний солвер – ode: function dydt = myode(t, y) dydt = -2*y + sin(t); endfunction

y = ode(1, 0, tspan, myode); % y0=1, t0=0, tspan – вектор часу plot(tspan, y);

Scilab також підтримує типи: "adams" (нежорсткі), "stiff" (жорсткі), "rk" (Runge-Kutta), "bdf" тощо. Для жорстких – dasrt, daskr (DAE з нульовими переходами).

У телекомунікаціях диференціальні рівняння моделюють:

– поширення сигналу в каналі з fading (диференціальні рівняння для амплітуди);

– динаміку черг (fluid-flow моделі: dq/dt = arrival_rate - service_rate);

– адаптивне регулювання потужності або швидкості передачі;

– моделі руху абонентів у мобільних мережах (наприклад, рівняння руху з випадковими прискореннями).

Поради для курсу:

– Починайте з ode45 (Octave) або ode з типом "adams"/"rk" (Scilab) – вони найуніверсальніші.

– Завжди фіксуйте seed для випадкових збурень.

– Перевіряйте точність: зменшуйте крок, порівнюйте з аналітичним розв’язком для простих випадків.

– Використовуйте графіку для візуалізації: plot(t, y) показує еволюцію стану.

Ці інструменти – основа для моделювання динамічних систем і мереж у темах 3 і 4. На практичних заняттях ви будете розв’язувати саме такі задачі: інтегрувати функції ймовірностей, моделювати динаміку черг чи поширення сигналу в часі. Освоївши їх, ви зможете переходити від простих формул до повноцінних імітаційних моделей реальних телекомунікаційних процесів.

Тема 3. Моделювання систем та мереж

3.1. Принципи побудови моделей

Побудова моделі – це не просто запис рівнянь чи написання коду, а систематичний процес, який дозволяє адекватно описати реальну систему (наприклад, телекомунікаційну мережу, чергу в базовій станції, канал зв'язку з перешкодами) для аналізу, прогнозування чи оптимізації. У вашій спеціальності принципи побудови моделей особливо важливі, бо реальні мережі 4G/5G/6G, IoT-системи чи SDN мають величезну складність, випадковість і динаміку, тому неправильна модель може призвести до хибних висновків.

Основні принципи побудови моделей, які ви будете застосовувати протягом усієї теми 3 і на практичних заняттях:

1. Принцип адекватності (відповідності реальності). Модель повинна достатньо точно відтворювати ті властивості об'єкта, які важливі для поставленої задачі. Не потрібно моделювати все до найдрібніших деталей – це неможливо й непотрібно. Наприклад, якщо вас цікавить середня затримка пакетів у мережі, не обов'язково моделювати кожен біт сигналу, достатньо описати чергу та розподіл часу обслуговування. Але якщо задача – аналіз інтерференції в 5G, то треба враховувати розташування антен, частотні смуги, тип модуляції та fading. Адекватність перевіряється на етапі валідації: порівнянням з реальними вимірами або відомими результатами.

2. Принцип необхідної повноти. Модель повинна включати всі суттєві фактори, що впливають на вихідні показники, і ігнорувати несуттєві. Визначити «суттєвість» допомагає аналіз чутливості: якщо зміна параметра на 10% змінює результат менше ніж на 1%, його можна спростити. У телекомунікаціях типові суттєві фактори: інтенсивність трафіку, розподіл часу між пакетами, тип черги, пропускна здатність каналу, ймовірність помилки, механізм повторної передачі. Несуттєві на першому етапі: мікроскопічні деталі фізичного шару (якщо не моделюємо BER на рівні бітів).

3. Принцип ієрархічності. Складні системи моделюють на різних рівнях абстракції:

- концептуальний рівень (блок-схема: джерело → канал → приймач → завада);
- функціональний рівень (що саме робить кожен блок);
- структурний рівень (компоненти та зв'язки);
- деталізований рівень (математичні рівняння, алгоритми, параметри). У курсі ви починаєте з концептуальної моделі (наприклад, система з чергою M/M/1), потім переходите до математичної (формули Кендалла) або імітаційної (код в Octave/Scilab).

4. Принцип декомпозиції. Складну систему розбивають на підсистеми, моделюють окремо, а потім об’єднують. Наприклад, мережа 5G: спочатку моделюємо один осередок (черга + канал), потім кілька осередків з handover, потім всю мережу з інтерференцією. Це полегшує перевірку та налагодження.

5. Принцип балансу між точністю та складністю. Чим точніша модель, тим складніша, довше обчислюється й більше параметрів потрібно оцінювати. У телекомунікаціях часто використовують аналітичні моделі для швидкої оцінки (формула Ерланга для ймовірності блокування), а для складних сценаріїв – імітаційні (Монте-Карло). Баланс знаходять експериментально: починають з простої моделі, додають деталі, поки похибка не стане прийнятною.

6. Принцип стохастичності для реальних систем. Більшість процесів у мережах – випадкові: прихід пакетів (пуассонівський), час обслуговування (експоненціальний або важкохвостовий), завади, рухи користувачів. Тому більшість моделей у курсі – стохастичні: використовують rand, randn, poissrnd, grand для генерації подій.

7. Принцип ітеративності та верифікації. Побудова моделі – це цикл: концептуальна модель → математична → реалізація → обчислення → порівняння з реальністю → уточнення. На кожному кроці перевіряють: чи правильно реалізована модель (верифікація), чи адекватна реальності (валідація). Якщо результати сильно відрізняються – повертаються назад: змінюють припущення, додають фактори, уточнюють розподіли.

У контексті телекомунікаційних систем і мереж типова послідовність побудови моделі виглядає так:

– Визначити мету (наприклад, оцінити затримку при навантаженні 0,8 ерланга).

– Зібрати дані про систему (тип трафіку, параметри каналу, дисципліна обслуговування).

– Побудувати концептуальну модель (джерело пакетів → черга → сервер → вихід).

– Вибрати тип моделі (аналітична M/M/1 чи імітаційна з реальними розподілами).

– Записати математичну постановку (рівняння, параметри).

– Реалізувати в Octave/Scilab (генерація трафіку rand, цикли імітації, обчислення статистики).

– Провести експерименти, побудувати графіки.

– Перевірити адекватність (порівняти з формулою Поллачека–Хінчина для M/G/1 або з літературою).

Ці принципи – фундамент теми 3. Вони застосовуються до всіх видів моделювання: логіко-арифметичного, динамічного, прикладних систем і, головне, телекомунікаційних мереж. На практичних заняттях ви будете будувати моделі саме за цими правилами: від простої черги до складної імітації мережі з випадковими процесами. Якщо дотримуватися цих принципів, ваша модель буде не просто «працюючим кодом», а надійним інструментом для аналізу та оптимізації реальних систем зв'язку.

3.2. Логіко-арифметичне моделювання

Логіко-арифметичне моделювання – це метод опису систем, який поєднує логічні умови (якщо-то, перемикання станів, прийняття рішень) з арифметичними обчисленнями (формули, рівняння, накопичення значень). У телекомунікаціях цей підхід особливо корисний для моделювання систем з дискретними подіями, чергами, протоколами передачі даних, механізмами управління доступом, повторними передачами та адаптивними алгоритмами – тобто там, де поведінка системи залежить від комбінації логічних умов і числових параметрів.

На відміну від чисто аналітичних моделей (наприклад, формули Ерланга), логіко-арифметичні моделі не намагаються отримати готову формулу для всіх випадків. Замість цього вони описують правила роботи системи в явному вигляді, а потім реалізують ці правила в коді (Octave/Scilab), виконуючи імітацію крок за кроком. Це робить їх гнучкими для складних, нелінійних і умовних процесів, які неможливо описати одним рівнянням.

Основні елементи логіко-арифметичного моделювання:

– Стани системи. Система перебуває в одному з чітко визначених станів (наприклад, «канал вільний», «канал зайнятий передачею», «очікування ACK», «повторна передача»). Перехід між станами відбувається за логічними умовами.

– Події. Дискретні моменти часу, коли щось змінюється: прибуття пакета, завершення передачі, тайм-аут, помилка в каналі, надходження підтвердження.

– Логічні умови. Якщо-то конструкції, які визначають, що робити в кожній ситуації. Наприклад: якщо буфер повний – пакет відкидається; якщо SINR > порогу – використовується вища модуляція; якщо отримано NACK – запускається повторна передача.

– Арифметичні операції. Обчислення часу очікування, накопичення статистики (кількість переданих пакетів, сумарна затримка, кількість помилок), оновлення лічильників, розрахунок поточної пропускної здатності.

– Час. Моделювання часу може бути дискретним (кроки фіксованої тривалості) або подієво-орієнтованим (час стрибає до наступної події). У телекомунікаціях частіше

використовують подієво-орієнтовану імітацію – це ефективніше для мереж з низькою інтенсивністю подій.

Типовий приклад логіко-арифметичної моделі в телекомунікаціях – імітація простої системи з чергою та ARQ (Automatic Repeat reQuest):

– Стани: idle (вільний), transmitting (передача), waiting_ack (очікування підтвердження), retransmitting (повтор).

– Події: arrival (прибуття нового пакета), end_transmission (завершення передачі), timeout (тайм-аут), ack_received, nack_received.

– Логіка: якщо канал вільний і є пакет у черзі → перейти в transmitting, запустити таймер передачі якщо отримано ACK → видалити пакет, перейти в idle якщо отримано NACK або тайм-аут → перейти в retransmitting, збільшити лічильник повторів якщо кількість повторів > max → відкинути пакет

– Арифметика: накопичувати сумарний час очікування, кількість успішних передач, кількість повторів, обчислювати середню затримку як сумарний час очікування / кількість пакетів.

У коді це реалізується через цикл while або for з поточним часом t, чергою подій (event queue), генерацією часу наступної події за допомогою rand (експоненціальний розподіл для прибуття, фіксований або випадковий для передачі).

Переваги логіко-арифметичного підходу в курсі:

– Дозволяє моделювати реальні протоколи (ALOHA, CSMA/CA, TCP, HARQ у 5G) без спрощень, які потрібні для аналітичних формул.

– Легко додавати нові умови (наприклад, обмеження на буфер, пріоритети, handover, інтерференцію).

– Добре поєднується з методом Монте-Карло: багато реалізацій з різними seed дають статистичні оцінки затримки, пропускної здатності, ймовірності втрати.

– Результат – не одна формула, а розподіл показників, гістограми, залежності від параметрів.

Недоліки:

– Потребує більше часу на обчислення, ніж аналітичні моделі.

– Потрібна ретельна верифікація (перевірка на простих випадках, де є точна формула).

– Код може стати складним, тому важливо структурувати його: окремі функції для генерації подій, обробки станів, збору статистики.

У практичних заняттях ви будете будувати саме такі моделі: імітацію черги з втратами (M/M/1/K), слотованої ALOHA, простого ARQ, моделі з випадковими помилками в

каналі. Кожна модель починається з опису станів, подій і правил переходу (логіка), потім переходить до коду з циклами, умовами if і накопиченням статистики (арифметика). Це основа для розуміння, як працюють реальні телекомунікаційні системи на рівні подій і рішень, а не тільки середніх значень. Після освоєння логіко-арифметичного підходу ви зможете моделювати значно складніші процеси, ніж дозволяють класичні формули теорії масового обслуговування.

3.3. Моделювання динамічних систем

Динамічні системи – це системи, стан яких змінюється в часі під впливом зовнішніх і внутрішніх факторів. У телекомунікаціях майже всі реальні об’єкти є динамічними: навантаження в мережі змінюється протягом доби, кількість активних користувачів коливається, потужність сигналу падає через fading, handover відбувається при русі абонента, адаптивні алгоритми змінюють модуляцію та кодування в реальному часі. Моделювання динамічних систем дозволяє прогнозувати поведінку таких процесів, оцінювати перехідні режими, стабільність і оптимізувати параметри.

У курсі динамічні системи моделюються двома основними підходами: неперервними (диференціальними рівняннями) та дискретними (подієво-орієнтованими або час-дискретними моделями). Обидва підходи реалізуються в GNU Octave та Scilab, і ви будете використовувати їх на практичних заняттях.

Неперервні динамічні моделі (диференціальні рівняння)

Найчастіше використовуються звичайні диференціальні рівняння (ЗДР) першого порядку або системи ЗДР. Вони описують швидкість зміни стану системи.

Приклади застосування в телекомунікаціях:

– Fluid-flow модель черги: dq/dt = λ(t) – μ(t), де q – довжина черги, λ(t) – інтенсивність прибуття, μ(t) – інтенсивність обслуговування.

– Динаміка потужності передавача в адаптивному контролі потужності (power control): dp/dt = k · (SINR_цільовий – SINR_поточний).

– Зміна кількості користувачів у осередку: dn/dt = прибуття – відхід – handover.

– Моделі поширення сигналу в каналі з Rayleigh fading (рівняння для огинаючої амплітуди).

У Octave/Scilab це розв’язується функціями ode45, ode23, lsode (Octave) або ode (Scilab). Приклад простої моделі черги з змінним навантаженням: function dq = queue_dynamics(t, q) lambda = 5 + 3*sin(0.5*t); % змінне навантаження mu = 8; % постійна швидкість обслуговування dq = lambda - mu * (q > 0); % якщо черга порожня – не обслуговуємо endfunction

[t, q] = ode45(@queue_dynamics, [0 100], 0); plot(t, q); title("Динаміка довжини черги"); xlabel("Час"); ylabel("Кількість пакетів");

Така модель показує, як черга накопичується в періоди піку та спорожнюється в періоди спаду.

Дискретні динамічні моделі

Більшість телекомунікаційних процесів краще моделювати дискретно, бо події (прибуття пакета, завершення передачі, помилка) відбуваються в окремі моменти часу.

Види дискретних моделей:

1. Час-дискретні (фіксований крок часу Δt): на кожному кроці перевіряються умови, генеруються події. Підходить для синхронних систем (наприклад, слотовані протоколи як TDMA або slotted ALOHA).

2. Подієво-орієнтовані (event-driven): час стрибає до наступної події. Найефективніший для мереж з низькою інтенсивністю трафіку. Використовується в більшості імітаційних моделей курсу.

Подієво-орієнтована модель будується за принципами логіко-арифметичного моделювання:

– Список подій (event list) – черга з часом і типом події.

– Поточний час t.

– Стани системи (довжина черги, стан каналу, таймери).

– Генерація часу наступної події (наприклад, для пуассонівського потоку: Δt = -log(rand()) / λ).

Приклад структури коду для подієво-орієнтованої моделі простої черги M/M/1:

– Ініціалізація: t = 0, черга порожня, канал вільний.

– Цикл while (кількість подій < N або t < T_max):

– Знайти найближчу подію (прибуття або завершення обслуговування).

– Оновити t до часу події.

– Обробити подію: якщо прибуття – додати пакет у чергу, якщо канал вільний – почати обслуговування, запланувати завершення через експоненціальний час. якщо завершення – видалити пакет, оновити статистику затримки, якщо черга не порожня – почати наступний.

– Збирати статистику: сумарний час очікування, кількість відправлених пакетів, час зайнятості сервера.

Такий підхід дозволяє точно моделювати випадкові процеси без фіксованого кроку часу, економлячи обчислювальні ресурси.

Застосування в телекомунікаціях

У курсі ви моделюєте динамічні процеси:

– Зміну навантаження в мережі протягом доби (синусоїдальне або реальне трафік-профіль).

– Перехідні режими після зміни параметрів (наприклад, після збільшення кількості користувачів).

– Адаптивні механізми: зміна MCS у 5G залежно від SINR, контроль перевантаження (congestion control).

– Динаміку черг у вузлах мережі (буфер роутера, черга в базовій станції).

– Мобільність: траєкторії користувачів, handover, зміна осередку.

Переваги динамічного моделювання:

– Враховує перехідні процеси, де аналітичні формули (стаціонарний режим) не працюють.

– Дозволяє бачити коливання, піки, нестабільність.

– Легко додаються нелінійності, умовні переходи, випадкові фактори.

Недоліки:

– Потребує більше часу обчислень.

– Результат – не формула, а набір траєкторій, тому потрібні багато реалізацій (Монте-Карло) для статистики.

На практичних заняттях ви будете поєднувати обидва підходи: неперервні моделі для fluid-аналізу черг і подієво-орієнтовані для точної імітації протоколів. Після освоєння цих методів ви зможете переходити до моделювання повноцінних телекомунікаційних мереж з динамічним трафіком, мобільністю та адаптацією – це основа для реальних задач проектування та оптимізації систем зв’язку.

3.4. Моделювання телекомунікаційних систем та мереж

Моделювання телекомунікаційних систем та мереж – це центральна частина дисципліни для вашої спеціальності 172 «Електронні комунікації та радіотехніка». Саме тут поєднуються всі попередні теми: математичні моделі, комп’ютерне програмування в Octave/Scilab, логіко-арифметичні підходи, стохастичні процеси, динамічне моделювання та оптимізація. Мета – отримати кількісні оцінки ключових показників якості (QoS, QoE): затримка, пропускна здатність, ймовірність втрати пакета, ймовірність блокування, енергоспоживання, покриття, інтерференція, надійність тощо.

Телекомунікаційні системи та мережі моделюють на різних рівнях абстракції, залежно від задачі:

1. Фізичний рівень (PHY) Моделювання поширення сигналу, втрат на трасі, fading (Rayleigh, Rician, Nakagami), інтерференції, шумів (AWGN), модуляції, кодування, MIMO. Типові моделі:

– детерміновані (free space path loss, Okumura-Hata, COST-231);

– стохастичні (shadowing з логнормальним розподілом, багатопроменеве поширення);

– імітаційні (генерація Rayleigh fading за допомогою Jakes-моделі або фільтра з кореляцією). У Octave/Scilab це реалізується через генерацію randn для шуму, обчислення SINR = P·G / (N + I), BER через Q-функцію або Монте-Карло.

2. Канальний рівень (MAC) Моделювання доступу до середовища: ALOHA, slotted ALOHA, CSMA/CA (Wi-Fi), TDMA, FDMA, OFDMA (LTE/5G), HARQ, ARQ. Ключові показники: throughput, collision probability, backoff-алгоритми, енергоефективність. Найчастіше – подієво-орієнтована імітація: генеруємо прибуття пакетів (poisson), слоти часу, перевірка колізій, повторні передачі. Приклад: slotted ALOHA – ймовірність успіху G·e^(-G), максимум 1/e ≈ 0.368 при G=1.

3. Рівень мережі (NET) Моделювання маршрутизації, черг у вузлах, управління перевантаженням, handover у мобільних мережах, SDN/NFV. Типові моделі:

– черги M/M/1, M/M/k, M/G/1, G/G/1, черги з втратами M/M/k/k (Ерланг B), з чергою M/M/1/K (Ерланг C);

– мережеві графі (shortest path, Dijkstra, OSPF, BGP);

– fluid-flow моделі для великих мереж (диференціальні рівняння для потоків). У імітації: черги пакетів, генерація трафіку (rand для експоненціального міжприбуття), обчислення затримки за Поллачеком–Хінчином або Монте-Карло.

4. Транспортний та прикладний рівні Моделювання TCP (reno, cubic, BBR), UDP, QUIC, потокового відео, VoIP, IoT-трафіку. Ключові аспекти: congestion window, RTT, packet loss, retransmission timeout, jitter. Часто – гібридні моделі: TCP-модуль + нижчі рівні.

5. Системний рівень (end-to-end) Повна мережа: базові станції, core-мережа, користувачі з мобільністю (random waypoint, Gauss-Markov), трафік-мікс (web, video, VoIP). Показники: E2E delay, packet delivery ratio, handover success rate, energy per bit, spectral efficiency. Зазвичай імітаційні моделі з Монте-Карло (тисячі реалізацій для статистики).

Основні інструменти моделювання в курсі:

– Аналітичні (для простих випадків): формули Ерланга, Кендалла, Поллачека–Хінчина, Шеннона, фреймворк Little’s Law (L = λW).

– Імітаційні (основний метод для складних систем): метод Монте-Карло, подієво-орієнтована симуляція, fluid-апроксимація.

– Гібридні: аналітична оцінка для перевірки, імітація для точності.

Типова послідовність моделювання телекомунікаційної системи:

1. Постановка задачі (наприклад, «оцінити затримку в 5G-мережі при 100 користувачах на осередок»).
2. Визначення рівня абстракції (PHY+MAC+NET чи тільки MAC).
3. Побудова концептуальної моделі (блоки: UE → BS → Core → сервер).
4. Вибір розподілів: прибуття – Пуассон, обслуговування – експоненціальне або Weibull, помилки – Bernoulli.
5. Реалізація в Octave/Scilab: функції для генерації подій, цикл імітації, збір статистики (середнє, дисперсія, CDF).
6. Проведення експериментів: варіювання навантаження, параметрів каналу, розміру буфера.
7. Візуалізація: графіки затримки vs навантаження, гістограми, boxplot, поверхні.
8. Валідація: порівняння з літературою, формулами, реальними даними.

У практичних заняттях ви будете моделювати:

– Просту чергу M/M/1 та M/M/1/K у базовій станції.
– Слотовану ALOHA або CSMA.
– Мережу з кількома вузлами та маршрутизацією.
– Динаміку handover у мобільній мережі.
– Оцінку пропускної здатності каналу з fading.

Після цієї теми ви зможете самостійно створювати моделі реальних телекомунікаційних сценаріїв, оцінювати їхню продуктивність і пропонувати покращення. Це безпосередньо готує до дипломної роботи та професійної діяльності в проектуванні, аналізі та оптимізації мереж 4G/5G/6G, IoT, Wi-Fi, приватних мереж тощо.

Тема 4. Оптимізація систем та мереж

4.1. Основні поняття

Оптимізація – це процес знаходження найкращого (оптимального) рішення задачі за наявних обмежень. У телекомунікаційних системах і мережах майже всі практичні задачі є оптимізаційними: ми хочемо максимізувати пропускну здатність, мінімізувати затримку, зменшити витрати енергії чи грошей, забезпечити максимальне покриття при обмеженому бюджеті тощо. Оптимізація дозволяє перетворити інтуїтивні бажання («щоб мережа працювала краще») на чіткі математичні рекомендації для проектування, налаштування та експлуатації.

Основні поняття оптимізації, які ви будете використовувати в темі 4:

Цільова функція (objective function, criterion) – математичний вираз, який ми хочемо максимізувати або мінімізувати. Приклади в телекомунікаціях:

- максимізувати сумарну пропускну здатність усіх користувачів (sum-rate);
- мінімізувати середню затримку пакетів;
- мінімізувати загальну потужність передавачів;
- максимізувати покриття території при фіксованому числі базових станцій;
- мінімізувати інтерференцію в мережі.

Змінні рішення (decision variables) – параметри, які ми можемо змінювати для досягнення оптимального результату. Вони можуть бути:

- неперервними (потужність передавача в мВт, кут нахилу антени в градусах, поріг handover);
- цілочисельними (кількість каналів, кількість базових станцій);
- бінарними (0/1 – встановлювати чи не встановлювати станцію в даній точці, активувати чи не активувати beamforming для користувача).

Обмеження (constraints) – жорсткі умови, яких обов’язково потрібно дотримуватися. Вони бувають:

- лінійні (сумарна потужність ≤ максимальна потужність передавача);
- нелінійні (SINR ≥ порогу для кожного користувача);
- цілочисельні (кількість ресурсів – ціле число);
- стохастичні (ймовірність блокування ≤ 0,01). Приклади: бюджет не більше 1 млн грн, затримка не більше 10 мс, покриття не менше 95%, інтерференція від сусідніх осередків не перевищує допустимого рівня.

Область допустимих рішень (feasible region) – множина всіх комбінацій змінних, які задовольняють усім обмеженням. Оптимальне рішення лежить саме в цій області (або на її межі).

Оптимальне рішення (optimal solution) – значення змінних, при яких цільова функція досягає найкращого значення (максимуму чи мінімуму) в межах допустимої області.

– Глобальний оптимум – найкраще можливе рішення.

– Локальний оптимум – найкраще в деякій околиці (часто в нелінійних задачах ми знаходимо саме локальний).

Однокритеріальна оптимізація – одна цільова функція (наприклад, тільки максимізувати пропускну здатність). Багатокритеріальна оптимізація (multi-objective) – кілька конфліктуючих цілей одночасно (наприклад, максимізувати пропускну здатність і мінімізувати енергоспоживання). Тут немає єдиного оптимального рішення – є множина Парето-оптимальних рішень, з яких обирають компроміс за допомогою методів зваженої суми, ε-обмежень чи візуалізації фронту Парето.

Детермінована оптимізація – всі параметри відомі точно (наприклад, фіксоване навантаження, відомі позиції користувачів). Стохастична оптимізація – параметри випадкові (випадкове навантаження, випадкові позиції користувачів, випадкові завади). Тут використовують:

– очікування (очікувана пропускна здатність);

– ймовірнісні обмеження (chance constraints);

– сценарії (Monte-Carlo з багатьма реалізаціями);

– robust optimization (найгірший випадок).

Класифікація задач за типом функцій та змінних (це ключові поняття для вибору методу розв’язання):

– Лінійне програмування (LP) – лінійна цільова функція та лінійні обмеження, змінні неперервні.

– Нелінійне програмування (NLP) – хоча б одна функція нелінійна.

– Цілочисельне програмування (IP/MIP) – змінні цілі або бінарні.

– Змішане цілочисельне програмування (MILP/MINLP) – комбінація неперервних і цілих змінних.

– Стохастичне програмування – з випадковими параметрами.

– Динамічне програмування – для задач з послідовними рішеннями в часі (наприклад, розподіл ресурсів у часі).

У телекомунікаціях типові оптимізаційні задачі:

– розміщення базових станцій (бінарні змінні + нелінійні обмеження на покриття та інтерференцію);

– розподіл ресурсів (RB у 5G, частоти, потужність);

– контроль потужності та beamforming (нелінійне);

– маршрутизація трафіку в SDN (лінійне або цілочисельне);

– оптимізація параметрів handover (пороги, таймери);

– енергозбереження в IoT-мережах.

У курсі ви будете вчитися формулювати задачі саме в цій термінології: чітко визначати цільову функцію, змінні, обмеження, тип задачі. Потім реалізовувати їх в Octave/Scilab (linprog для LP, fmincon-подібні для NLP, ga для генетичних алгоритмів тощо) і аналізувати результати. Розуміння цих базових понять дозволяє вже на етапі постановки задачі вирішити, який метод застосовувати, чи можна спростити модель до лінійної, чи потрібні стохастичні підходи, і як інтерпретувати отриманий оптимум у реальній мережі. Це фундамент для всіх подальших розділів теми 4 і для професійної роботи в проектуванні та оптимізації телекомунікаційних систем.

4.2. Лінійне програмування – симплекс-метод

Лінійне програмування (ЛП) – це клас оптимізаційних задач, де цільова функція та всі обмеження є лінійними, а змінні рішення – неперервними (або невід'ємними). У телекомунікаціях ЛП застосовується дуже часто, бо багато реальних задач можна достатньо точно апроксимувати лінійними моделями: розподіл ресурсів, розміщення обладнання, маршрутизація трафіку, контроль потужності в лінійній зоні тощо. Симплекс-метод – класичний і найефективніший алгоритм для точного розв'язання задач лінійного програмування.

Стандартна форма задачі лінійного програмування

Будь-яку задачу ЛП можна привести до канонічної (стандартної) форми:

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

за умов $a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \le b_1$ $a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n \le b_2$ … $a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n \le b_m$

$x_1 \ge 0, x_2 \ge 0, \ldots, x_n \ge 0$

Щоб застосувати симплекс-метод, вводимо додаткові змінні-«слабкості» (slack variables) $s_1, s_2, \ldots, s_m \ge 0$, перетворюючи нерівності на рівності: $a_{11}x_1 + \ldots + a_{1n}x_n + s_1 = b_1$ $a_{21}x_1 + \ldots + a_{2n}x_n + s_2 = b_2$ … $a_{m1}x_1 + \ldots + a_{mn}x_n + s_m = b_m$

Тоді система стає системою лінійних рівнянь, а симплекс-метод працює з базисними та небазисними змінними.

Ідея симплекс-методу

Симплекс-метод – це ітеративний алгоритм, який рухається від одного вершинного рішення (кута багатогранника допустимих рішень) до сусіднього, покращуючи значення цільової функції на кожному кроці, доки не буде досягнуто оптимум.

Ключові кроки:

1. Початкове базисне рішення (початкова таблиця симплекса). Зазвичай беруть нульове рішення $x = 0$, $s = b$ (якщо всі $b \geq 0$). Якщо є вільні змінні або $\geq$-обмеження – вводять штучні змінні (метод штучного базису або двофазний метод).

2. Перевірка оптимальності. У рядку цільової функції (Z-рядок) дивимося на коефіцієнти небазисних змінних.

– Якщо всі коефіцієнти ≤ 0 (для максимізації) – поточне рішення оптимальне.

– Якщо є хоча б один додатний коефіцієнт – можна покращити.

3. Вибір ведучої змінної (вхідної в базис). Оберіть небазисну змінну з найбільшим додатним коефіцієнтом у Z-рядку (правило Бленда або просто максимум).

4. Вибір ведучої рядка (вихідної змінної з базису). Для кожного рядка з додатним коефіцієнтом у стовпці ведучої змінної обчисліть відношення b_i / a_{ij} (де $a_{ij} > 0$). Оберіть рядок з найменшим додатним відношенням (правило мінімального відношення) – це запобігає негативним значенням змінних.

5. Перетворення таблиці (pivot-операція). Ведучий елемент (на перетині ведучої змінної та ведучого рядка) робимо = 1, інші елементи стовпця – 0, перераховуємо весь рядок і стовпець.

6. Повторюємо кроки 2–5, доки не досягнемо оптимальності.

Приклад простої задачі в телекомунікаціях

Задача: розподіл двох частотних каналів між трьома базовими станціями для максимізації сумарної пропускної здатності. Нехай x_1, x_2 – кількість каналів, виділених станціям 1 і 2 ($x_3 = 2 – x_1 – x_2$ автоматично). Максимізувати $Z = 15x_1 + 20x_2 + 10x_3$ за умов $x_1 + x_2 \leq 2$ $x_1 \leq 1.5$ $x_2 \leq 1.2$ $x_1, x_2 \geq 0$

Після введення slack-перемінних і приведення до стандартної форми симплекс-таблиця починається з базису $\{s_1, s_2, s_3\}$. Після кількох ітерацій оптимальне рішення: $x_1 = 0.8$, $x_2 = 1.2$, $Z = 36$ (максимальна сумарна пропускна здатність).

У Octave/Scilab для розв’язання ЛП використовується функція linprog: c = [-15; -20]; % для максимізації – мінус коефіцієнти A = [1 1; 1 0; 0 1]; b = [2; 1.5; 1.2]; lb = [0; 0]; [x, fval] = linprog(c, A, b, [ ], [ ], lb);

fval – значення цільової функції (від’ємне для максимізації).

Переваги та застосування симплекс-методу в телекомунікаціях

- Гарантує глобальний оптимум (якщо задача обмежена та допустима).
- Дуже ефективний для задач з сотнями змінних і обмежень.
- Застосовується для:
  - розподілу ресурсів (RB у 5G, частот, потужності);
  - розміщення базових станцій (якщо спростити до лінійної моделі);
  - оптимізації маршрутів у мережах з фіксованими потоками;
  - лінійної апроксимації нелінійних задач (наприклад, логарифмічна апроксимація для sum-rate);
  - релаксації цілочисельних задач (потім округлення або branch-and-bound).

У курсі симплекс-метод вивчається як базовий інструмент для розуміння, як працюють точні методи оптимізації. Ви будете формулювати задачі ЛП для телекомунікаційних сценаріїв, будувати таблиці симплекса вручну для маленьких задач і розв'язувати більші за допомогою linprog в Octave/Scilab. Це основа для переходу до нелінійного, цілочисельного та стохастичного програмування в наступних підтемах. Після освоєння симплекс-методу ви зможете швидко оцінювати, чи можна вашу задачу розв'язати точно й ефективно, чи потрібні наближені методи.

4.3. Нелінійна, цілочисельна, стохастична оптимізація

У реальних телекомунікаційних задачах рідко вдається обмежитися чистим лінійним програмуванням. Більшість практичних проблем містять нелінійні залежності (наприклад, пропускна здатність від SINR за логарифмічною формулою Шеннона), цілочисельні параметри (кількість базових станцій, каналів, RB) або випадкові фактори (навантаження, позиції користувачів, завади). Тому в курсі розглядаються три ключові розширення лінійного програмування: нелінійна, цілочисельна та стохастична оптимізація. Кожна з них має власні методи розв'язання та інструменти в Octave/Scilab.

Нелінійна оптимізація (NLP – Nonlinear Programming)

Нелінійна оптимізація виникає, коли хоча б одна з функцій (цільова або обмеження) нелінійна. У телекомунікаціях це найпоширеніший випадок.

Типові приклади:

- максимізація сумарної пропускної здатності (sum-rate) = $\Sigma \log_2(1 + SINR_i)$, де SINR залежить від потужностей нелінійно;
- мінімізація інтерференції при контролі потужності (SINR-залежність);
- оптимізація beamforming (нелінійні обмеження на норму вектора ваги);
- розміщення базових станцій з нелінійними моделями покриття (path loss ~ d^(-α));

– адаптивна модуляція та кодування (MCS) з дискретними, але нелінійними залежностями.

Методи розв’язання:

– Градієнтні методи (gradient descent, conjugate gradient) – прості, але повільно сходяться.

– Метод Ньютона або квазі-Ньютона (BFGS) – швидше, але потребує гессіана або його апроксимації.

– Interior-point методи – ефективні для великих задач.

– Безградієнтні методи (Nelder-Mead, pattern search) – коли похідні важко обчислити.

– Глобальні методи (генетичні алгоритми, PSO, simulated annealing) – для задач з багатьма локальними оптимумами.

У Octave/Scilab:

– fminunc або fminsearch (Nelder-Mead) – для безобмежених задач.

– fmincon (Octave – через пакет optim, Scilab – nlopt або fmincon-подібні) – з обмеженнями.

– ga (genetic algorithm) – для глобальної оптимізації.

Проблеми: немає гарантії глобального оптимуму, можливі локальні мінімуми, чутливість до початкового наближення. Тому часто запускають кілька разів з різних стартових точок або використовують гібридні методи (локальний + глобальний).

Цілочисельна оптимізація (IP/MIP/MINLP)

Коли деякі змінні рішення мають бути цілими числами (кількість антен, каналів, базових станцій) або бінарними (встановити/не встановити станцію, активувати/не активувати користувача).

Типові задачі:

– розміщення базових станцій (бінарні змінні $x_i = 1$, якщо станція в точці і встановлена);

– призначення частот або RB (integer assignment);

– вибір маршрутів у мережі (бінарні змінні для ребер);

– кластеризація користувачів у CoMP (цілочисельні).

Методи:

– Релаксація LP (розв’язати як неперервну задачу, потім округлити) – швидкий, але не завжди точний.

– Branch-and-Bound (гілки та межі) – точний метод, але експоненційний час у гіршому випадку.

– Branch-and-Cut, Cutting Plane – покращені версії.

– Евристики: генетичні алгоритми, tabu search, simulated annealing, greedy.

– Для змішаних задач (MIP/MINLP) – комбінація симплекс + branch-and-bound.

У Octave/Scilab:

– intlinprog (Octave – пакет optim) або intlinprog-подібні в Scilab.

– ga для евристичного розв'язання складних цілочисельних задач.

– Для малих задач – ручне округлення після LP-релаксації.

Проблеми: NP-важкі задачі, час розв'язання зростає експоненційно з розміром. Тому часто використовують наближені рішення або релаксацію.

Стохастична оптимізація

Коли параметри задачі випадкові (навантаження мережі, позиції користувачів, канальні коефіцієнти, завади). Замість фіксованих значень працюємо з розподілами або сценаріями.

Типи:

– Стохастичне програмування з двома етапами (two-stage): перше рішення приймається до реалізації випадковості, друге – після (наприклад, розміщення станцій до появи користувачів).

– Chance-constrained оптимізація: ймовірність порушення обмежень $\leq \varepsilon$ (наприклад, P(затримка > 10 мс) ≤ 0.01).

– Robust optimization: оптимізація для найгіршого випадку в межах невизначеності.

– Монте-Карло оптимізація: багаторазове розв'язання детермінованої задачі для різних реалізацій випадковості, потім усереднення або вибір найкращого.

– Stochastic gradient descent (SGD) – для великих даних і машинного навчання в мережах.

Приклади в телекомунікаціях:

– планування ресурсів при випадковому трафіку;

– beamforming з невідомими канальними станами;

– розміщення станцій з випадковими користувачами;

– енергозбереження в IoT з випадковими подіями.

Методи реалізації в Octave/Scilab:

– Монте-Карло: цикл з rand/randn, усереднення результатів.

– ga або particleswarm з випадковими оцінками функції.

– Для chance-constraints – апроксимація через Монте-Карло або сценарії.

У курсі ви будете:

– формулювати нелінійні задачі (наприклад, максимізація sum-rate з обмеженнями на потужність);

– розв’язувати їх за допомогою fmincon або ga;

– перетворювати цілочисельні задачі розміщення станцій на MIP і використовувати релаксацію + округлення;

– застосовувати Монте-Карло для стохастичних сценаріїв (наприклад, оцінка середньої пропускної здатності при випадковому навантаженні).

Ці три типи оптимізації охоплюють більшість реальних задач у телекомунікаціях. Після вивчення ви зможете переходити від простих лінійних моделей до повноцінних нелінійних, цілочисельних і стохастичних постановок, реалізовувати їх у Octave/Scilab і отримувати практичні рекомендації для проектування мереж 5G/6G, IoT, приватних мереж тощо.

Задача комівояжера є однією з класичних задач комбінаторної оптимізації та широко використовується під час моделювання транспортних, логістичних і телекомунікаційних систем. Вона формулюється як задача пошуку найкоротшого маршруту, що проходить через задану множину пунктів один раз і повертається у вихідний пункт.

Суть задачі полягає в тому, що комівояжер повинен відвідати кілька міст, причому кожне місто лише один раз, після чого повернутися до початкового міста. Відомі відстані між усіма парами міст, і необхідно визначити такий порядок їх відвідування, за якого сумарна довжина маршруту буде мінімальною. Таким чином, задача зводиться до знаходження оптимального циклу мінімальної довжини в повному зваженому графі.

З математичної точки зору задача комівояжера формулюється як задача пошуку гамільтонового циклу мінімальної ваги в графі, вершини якого відповідають містам, а ребра — відстаням між ними. Ця задача належить до класу NP-складних задач, тому для великих розмірностей її точний розв’язок потребує значних обчислювальних ресурсів.

Практичне значення задачі комівояжера полягає в тому, що вона використовується як модель для розв’язання широкого класу задач оптимального маршрутування. Зокрема, вона застосовується під час оптимізації маршрутів доставки вантажів, планування пересування транспортних засобів, побудови оптимальних маршрутів обслуговування клієнтів, організації руху мобільних сервісних груп та прокладання кабельних трас у телекомунікаційних мережах.

У телекомунікаційних системах задача комівояжера використовується під час оптимального планування маршрутів прокладання волоконно-оптичних ліній зв’язку, оптимізації обслуговування вузлів мережі технічним персоналом, а також під час побудови

маршрутів збору інформації в сенсорних мережах. Крім того, її математичний апарат застосовується у задачах маршрутизації пакетів даних у складних мережевих структурах.

Для розв'язання задачі комівояжера застосовуються як точні, так і наближені методи. До точних методів належать метод повного перебору, метод гілок і меж та метод динамічного програмування. Однак їх використання є ефективним лише для задач невеликої розмірності. Для задач великої розмірності застосовуються евристичні та метаевристичні алгоритми, серед яких широко використовуються жадібні алгоритми, генетичні алгоритми, метод імітації відпалу та мурашині алгоритми.

4.4. Застосування в телекомунікаціях

Сучасні телекомунікаційні системи є складними технічними об'єктами, ефективність функціонування яких значною мірою залежить від раціонального використання наявних ресурсів. До таких ресурсів належать пропускна здатність каналів зв'язку, частотний спектр, обчислювальні можливості вузлів мережі, енергетичні ресурси обладнання та просторове розміщення мережевої інфраструктури. Саме тому задачі оптимізації займають важливе місце на всіх етапах життєвого циклу телекомунікаційних систем – від проектування до експлуатації.

На етапі проектування мережі важливо визначити оптимальну топологію, яка забезпечує необхідний рівень зв'язності між вузлами при мінімальних витратах на побудову мережі. У цьому випадку застосовуються методи математичного програмування та теорії графів, що дозволяють знаходити структури мереж із мінімальною довжиною каналів зв'язку або мінімальною вартістю їх реалізації за умови забезпечення заданих технічних характеристик. Такі задачі виникають під час побудови транспортних мереж операторів зв'язку, корпоративних мереж передачі даних та мереж доступу.

Важливим напрямом застосування оптимізаційних методів є маршрутизація інформаційних потоків. У процесі функціонування мережі необхідно визначати такі маршрути передачі даних, які забезпечують мінімальні затримки, раціональне використання пропускної здатності каналів та необхідний рівень надійності передачі інформації. Формалізація задач маршрутизації здійснюється за допомогою моделей лінійного або нелінійного програмування, а також моделей теорії графів. Оптимізація маршрутів особливо актуальна для пакетних мереж передачі даних, зокрема мереж Інтернет та корпоративних мереж зв'язку.

Не менш важливою задачею є ефективний розподіл частотного ресурсу, який є обмеженим і дорогим ресурсом у радіосистемах зв'язку. Оптимізація використання частот дозволяє зменшити рівень взаємних завад між каналами зв'язку та підвищити пропускну

здатність мережі. У цьому випадку застосовуються методи дискретної та комбінаторної оптимізації, що дозволяють забезпечити електромагнітну сумісність радіоелектронних засобів і підвищити ефективність використання спектра.

У мобільних системах зв'язку важливу роль відіграє задача оптимального розміщення базових станцій. Вона полягає у визначенні таких координат розташування передавальних пристроїв, за яких забезпечується максимальне покриття території обслуговування при мінімальних витратах на створення інфраструктури. Одночасно необхідно враховувати вплив інтерференції між сусідніми базовими станціями та забезпечувати необхідний рівень якості обслуговування абонентів. Розв'язання таких задач здійснюється методами цілочисельного та стохастичного програмування.

У процесі експлуатації телекомунікаційних мереж важливе значення має оптимальний розподіл навантаження між вузлами та каналами зв'язку. Раціональне балансування трафіку дозволяє підвищити ефективність використання мережевих ресурсів, зменшити затримки передачі інформації та підвищити надійність функціонування мережі. Для розв'язання задач балансування навантаження застосовуються методи лінійної та нелінійної оптимізації, які дозволяють враховувати зміну параметрів мережі в реальному масштабі часу.

Особливе місце серед задач оптимізації займає забезпечення показників якості обслуговування користувачів, які характеризуються такими параметрами, як затримка передачі пакетів, варіація затримки, імовірність втрати пакетів та пропускна здатність каналів зв'язку. У цьому випадку задача оптимізації має багатокритеріальний характер, оскільки необхідно одночасно враховувати декілька показників якості, що можуть суперечити один одному. Застосування методів багатокритеріальної оптимізації дозволяє знаходити компромісні рішення, які забезпечують необхідний рівень функціонування телекомунікаційної системи.

Методи оптимізації широко застосовуються в сучасних телекомунікаційних технологіях, зокрема в мобільних мережах четвертого та п'ятого поколінь, супутникових системах зв'язку, оптичних транспортних мережах, мережах Інтернет та програмно-конфігурованих мережах. У таких системах оптимізаційні алгоритми часто працюють у режимі реального часу та забезпечують адаптацію параметрів мережі до зміни навантаження і умов функціонування.

Таким чином, застосування методів математичної оптимізації є необхідною складовою процесу проектування та експлуатації телекомунікаційних систем. Використання оптимізаційних моделей дозволяє підвищити ефективність використання ресурсів мережі, забезпечити необхідну якість обслуговування користувачів і зменшити витрати на створення та розвиток телекомунікаційної інфраструктури.

Навчальне видання

Розенвассер Денис Михайлович

МОДЕЛЮВАННЯ СИСТЕМ

Конспект лекцій

Українською мовою