

Міністерство освіти і науки України
Українська інженерно-педагогічна академія

А. Л. Литвинов

**Математичні методи та моделі в розрахунках на ЕОМ.
Система Maple та її використання для моделювання
техніко-економічних систем**

Рекомендовано Міністерством освіти і науки України
як навчальний посібник для студентів спеціальності
“Теплові електричні станції”, а також для магістрів і
аспірантів

Харків 2004

ББК 22.18+32.81+65

УДК 004.78:004.942

Рекомендовано Міністерством освіти і науки України

(лист № 14/18.2-462 від 12.03.2004 р.)

Рецензенти:

О.І.Федоренко, д-р фіз-мат наук, професор.
(Харківський інститут соціального прогресу)

О.Є. Федорович, д-р техн. наук, професор.
(Національний аерокосмічний університет
ім. М.Є. Жуковського – ХАІ)

Литвинов А.Л.

Математичні методи та моделі в розрахунках на ЕОМ. Система Maple та її використання для моделювання техніко-економічних систем: Навчальний посібник. - Харків: УІПА, 2004. - 103 с., іл. 28.

У навчальному посібнику викладаються основи однієї із потужних систем комп'ютерної математики Maple . Ця система дозволяє розв'язувати у діалоговому режимі значну кількість математичних задач від самих простіших розрахунків до складних аналітичних перетворень і обчислень. Має зручний інтерфейс і добре пристосована до потреб користувачів середньої кваліфікації. Приведені основні функції та оператори системи з прикладами їх застосування. Значна увага присвячена використанню Maple для моделювання процесів в енергетиці, техніці та економіці.

Навчальний посібник призначений для студентів вищих учбових закладів спеціальності “Теплові електричні станції”, а також для магістрів і аспірантів.

© А.Л.Литвинов, 2004

© Українська інженерно-педагогічна академія, 2004

Вступ

Для сучасного рівня розвитку економіки та техніки характерне широке використання метода моделювання. При дослідженні складних енергетичних, технічних та економічних систем виникають ситуації, коли неможливо безпосередньо отримати знання або прогнозувати їх поведінку у майбутньому із-за складності системи, відсутності її повної теорії. У цьому випадку в процесі дослідження можна замінити досліджувану систему (оригінал) якимось об'єктом, можливо і іншої природи, східну з оригіналом по поведінку або по опису - моделлю. Оперуючи з моделлю, можливо отримати знання об оригіналі. Усі рівняння та формули теплоенергетики є ні що інше, як моделі функціонування реальних теплоенергетичних систем.

Курс "Математичні методи та моделі в розрахунках на ЕОМ" орієнтовано на здобуття студентами навички реалізації на ЕОМ методів та моделей інженерної та наукової діяльності.

Стрімкий ріст складності енергетичних, технічних та економічних систем, їх значна кількість і швидкий розвиток роблять необхідним використання математичних наук для їх дослідження. При цім використовуються глибинні результати математичних наук, використовуються різні підходи. В сучасних умовах математика і, у частковості, математичні моделі, це не просто інструмент але необхідна складова частина інженерної і наукової діяльності. Ця необхідність обумовлена особливою роллю математики серед інших наук, бо сучасна математика - це і засоби, принципи та методи формального опису будь-яких об'єктів, явищ і процесів. Математичне дослідження об'єкта або процесу дозволяє виявити властиві йому закономірності, основні властивості, закони зміни і взаємодії з іншими об'єктами або явищами. А це, в свою чергу, дозволяє керувати цим об'єктом або процесом, знайти оптимальні засоби керування при заданих цілях і передбачити наслідки цього керування. При цьому усе це відбувається на язику, який не допускає неясності в тлумаченні.

Моделювання реальних процесів потребує значних ресурсів та часу. Тому навчання з систем комп'ютерної математики типа Maple, що дозволяють значно прискорити математичний опис та моделювання різних систем, є насущною потребою для спеціалістів інженерного профілю.

1. Теоретичні аспекти моделювання

1.1. Моделювання як метод наукового пізнання

Оригінал, модель засоби оперування з моделлю можуть бути самими різноманітними. В загальному випадку ніяких обмежень на використання метода моделювання не накладається, крім одного: потрібно, щоб модель дозволяла переносити отримані знання на оригінал із урахуванням його природи.

Особливість тих ситуацій, при яких використовують метод моделювання, сама специфіка використання моделей у пізнавальному процесі, дозволяють зробити висновок, що використання метода моделювання можливе лише в тих випадках, коли у дослідника вже склалися якісь початкові теоретичні представлення о предметі дослідження. Не знаючи хоча б приблизно деякі властивості об'єкта-оригінала, ми не можемо підступатися до вибору моделі. Моделювання є ефективним методом пізнання, тільки коли проведено початкове теоретичне узагальнення знань об досліджуванім об'єкті, коли ці знання вже систематизовані якоюсь теорією. Роль моделі заключається у перевірці істинності цих первісних узагальнень, у їх подальшому розвитку. В якості моделі береться об'єкт, властивості якого досліджені досить повно. Тільки знаючи властивості моделі і оригіналу настільки, що можна установити подібність між моделлю та оригіналом, приступають до подальших досліджень.

При використанні метода моделювання велике значення набуває питання об дійсній оцінці моделей. Якщо розглядати модель як специфічний образ досліджуваного об'єкта, то відношення між ними подібності або неподібності, а також ступень подібності можуть бути оцінені у термінах істинності або неістинності (хибності), а також у відношенні абсолютної та відносної істини. Істинність моделі означає відповідність моделі об'єкту дослідження. Поняття істинності моделі повинно проводитися приймаючи до уваги умови, на основі яких модель того або іншого типа відтворює досліджуване явище. Очевидно, що для різного виду моделей ці умови будуть різними.

Процес моделювання починається з постановки задачі дослідження, розв'язок якої може бути отримано за допомогою моделі. У відповідності до конкретної задачі необхідно виділити деякі властивості і відношення, дослідження яких може привести до цілі.

Після визначення предмета дослідження, перед

дослідником виникає необхідність розширення знань о ній. Результати досліджень об'єкта можуть показати неможливість або великі труднощі розв'язку поставленої задачі шляхом безпосереднього оперування з об'єктом. Щоб подолати ці труднощі дослідник у процесі пізнання створює проміжну ланку – модель. У процесі моделювання дослідник від попереднього дослідження об'єкта, його опису переходить до побудови або вибору такого об'єкта, на основі вивчення якого було б можливо отримати інтересуючі його дані о першій об'єкті. Пошук моделі можливо як на інтуїтивній основі, так і на строго логічній основі.

Після вибору моделі її необхідно дослідити, тобто вона стає об'єктом дослідження. При цьому усі дії виконуються над моделлю і направлені на отримання знань об цій об'єкті, на встановлення законів його розвитку, властивостей і відносин.

Кінцевим етапом моделювання є переніс отриманих з допомогою моделі даних на оригінал. Завдяки відповідності елементів та відносин моделі елементам та відносинам оригіналу існує можливість переносу знань отриманих з допомогою моделі даних на оригінал. Зв'язок відповідних елементів та відносин моделі з елементами та відносинами оригінала встановлюються дослідником в процесі моделювання. Евристичне значення моделювання виявляється у тім, що в результаті констатації наявності ряду загальних властивостей у моделі і оригіналі дослідник приписує оригіналу деяку нову властивість, знайдену при дослідженні моделі.

Уся практика використання метода моделювання свідчить о тім, що запорукою успіху є не здібність дослідника будувати довільні мисленні конструкції, а поперед усього в здібності правильно відображати існуючі зв'язки, розкривати об'єктивні закономірності та на основі цього правильно завбачати невідомі сторони і властивості оригінала. Як ніякому другому методу моделюванню властиве тісна, органічна єдність теорії та практики. Практика у формі побудови моделі та модельного експерименту виступає тут вже не тільки як вихідний пункт пізнання, але поперед усе як форма розвитку теорії. Моделювання характеризується також такої важливою діалектичною особливістю як взаємозв'язок та взаємо проникання старого та нового знань, як посування знання від відомого до невідомого.

Моделювання у порівнянні з іншими методами дослідження має значні переваги. Воно дозволяє досліджувати та імітувати особливості функціонування систем в яких можливих умовах,

віртуально відтворювати ситуації, які неможливі у натурному експерименті. Завдяки цьому зменшується потреба у складному лабораторному обладнанні та експлуатаційних випробуваннях системи.

1.2. Технологія побудови комп'ютерних моделей

Процес математичного моделювання, тобто вивчення явищ за допомогою математичних моделей, можна умовно розділити на чотири етапи.

Перший етап — формулювання законів, що зв'язують основні об'єкти моделі, виявлення істотних і другорядних факторів, що впливають на досліджуване явище. Для цього потрібно глибоке розуміння фактів, що відносяться до досліджуваних явищ. Корисно керуватися так називаним «потрійним правилом» побудови моделі - необхідно відповісти на три питання:

— Чим ми керуємо? (потрібно виділити набір керованих параметрів);

— Як ми керуємо? (потрібно визначити припустима безліч або область зміни параметрів; припустима безліч задається рівностями, нерівностями і додатковими вимогами);

— Навіщо ми керуємо? (потрібно сформулювати мета побудови моделі, наприклад — максимізація прибутку).

У результаті виходить запис у математичних термінах якісних представлень про зв'язки між об'єктами або параметрами моделі.

Другий етап — дослідження математичних задач, до яких зводяться математичні моделі.

Будь-яке явище можна дуже грубо описати в такий спосіб: є деякі вхідні дані, потім відбувається явище, у результаті якого з'являються вихідні дані.

Математична модель — це задача, початковими даними якої є вхідні дані, явище записане в математичній символіці як закон перетворення початкових даних, у результаті цього перетворення виходять вихідні дані. Як правило, задача описує досліджуване явище приблизно.

Другий етап математичного моделювання полягає в рішенні цієї задачі. На цьому етапі застосовується математичний апарат

для аналізу задачі й обчислювальна техніка для одержання кількісному інформації.

Часто буває ситуація, коли математичні задачі, що відбивають різні явища, однакові. Наприклад, багато явищ зводяться приблизно до знаходження мінімуму або максимуму якої-небудь функції. Подібні, часто застосовувані, математичні моделі вивчаються окремо, незалежно від описуваних ними явищ. Наприклад, численні задачі знаходження найкращих дій вивчаються дисциплінами: методи оптимізації, математичне програмування, оптимальне керування, прийняття рішень і ін.

Третій етап — з'ясування, наскільки адекватне побудована модель відбиває досліджуване явище. Це з'ясовують, наприклад, за допомогою контрольного приклада. Контрольний приклад — це окремий випадок явища, що спостерігається, вхідні і вихідні дані якого відомі. Вирішується задача з цими вхідними даними і рішення порівнюється з вихідними даними. У залежності від погрішності рішення робиться висновок про адекватність моделі. Існують і інші способи визначення адекватності моделі.

Четвертий етап — модернізація моделі. Поява нової інформації про досліджуване явище, удосконалювання математичних методів, розвиток обчислювальної техніки роблять математичну модель незадовільною. Виникає необхідність побудови нової, більш досконалою моделі.

Розглянемо приклад побудови математичної моделі. Нехай вивчається процес падіння якого-небудь тіла з висоти. Існує багато математичних моделей цього процесу. Ще Аристотель побудував математичну модель падіння тіла. Він вважав, що швидкість вільно падаючого тіла пропорційна його ваги. Тобто, якщо m — вага тіла, v — його швидкість, то $v = k \cdot m$, де k — деякий коефіцієнт.

Ця модель активно обговорювалася поколіннями філософів, були знайдені спростування і запропоновані інші моделі, що, однак, також не витримували критики. Була потрібна більш адекватна нова модель. Вона була побудована Галилеєм, і ми дотепер користуємося нею: швидкість вільно падаючого тіла пропорційна часу $v = at$, де a — коефіцієнт пропорційності.

Ця модель була отримана Галилеєм методом проб і помилок після чисельних експериментів і міркувань, а потім доповнена й уточнена Ньютоном, що установив, що коефіцієнт пропорційності $a = g \sim 9,8 \text{ м/с}^2$ — прискорення вільного падіння.

Падіння, наприклад, каменю з не дуже великої висоти добре

описується цією моделлю. А падіння листа з дерева в тиху погоду відбувається за іншим законом. Таким чином, модель Галілея не є універсальною для будь-якого падаючого тіла. У неї, як і в будь-якої іншої моделі, є область застосовності. Для пояснення понять області застосовності й адекватності математичної моделі ще раз розглянемо вільно падаюче тіло. Розглянемо кожний з чотирьох етапів побудови математичної моделі.

Перший етап: формулювання законів, що зв'язують основні об'єкти моделі і виділення факторів, що істотно впливають на процес падіння. Об'єктами даного процесу є: саме падаюче тіло; середовище, у якій воно падає (наприклад, повітря); Земля, сила ваги якої діє на тіло; Місяць, Сонце й інші космічні тіла, що впливають на падаюче тіло і так далі. Кожний з перерахованих об'єктів впливає на процес падіння. Процес буде відбуватися по-різному в залежності від: форми тіла (що вплине, наприклад, на опір повітря); кольору тіла (який вплине, зокрема, на світловий тиск на нього); речовини, з якого складається тіло (від цього залежить, наприклад, магнітне поле, створюване тілом, що буде взаємодіяти з магнітними полями інших тіл) тощо. В'язкість середовища, її турбулентність та інше також вплинуть на падіння тіла; географічне місце, де відбувається падіння вплине, наприклад, на величину атмосферного тиску і сили ваги і так далі.

Приведені далеко не усі фактори, що реально впливають на падіння розглянутого тіла. Очевидно, що практично неможливо врахувати їх усі при побудові моделі і тим самим забезпечити абсолютну точність моделі або, іншими словами, абсолютну адекватність її реальному процесові. Отже, необхідно розглядати кожен процес падіння окремо, щоб визначити: які фактори істотно впливають на нього, а впливом яких можна зневажити. Строго говорячи, вирішити, які фактори істотні, а які — ні, можна лише обчисливши внесок кожного фактора. Однак це також практично нездійсненно. Тому доводиться при оцінці більшості факторів покладатися на інтуїцію і досвід, що спрощує задачу, з одного боку, а з іншого боку - вносить деяку частку ризику. Якщо не взяти до уваги істотні фактори, то виникнуть проблеми з адекватністю моделі. Але і якщо врахувати другорядні фактори, то в моделі з'являться зайві рівняння або ускладняться вже наявні. Модель стане більш громіздкою і складною, виникнуть додаткові труднощі при роботі з нею.

Наприклад, досвід і обчислення показують, що при розгляді падіння тіл можна не враховувати вплив небесних тіл, крім Землі,

якщо маса падаючого тіла багато менше, ніж маса Землі, а відстань до небесного тіла багато більше, ніж відстань до Землі. Якщо розглядати падіння тіла невеликих розмірів, але достатньої маси, щоб тіло можна було вважати матеріальною крапкою, то можна зневажити середовищем, у якій падає тіло (наприклад, для каменю, що падає в повітрі). Якщо ж тіло має велику площу поверхні і невелику масу (як лист дерева), то опір повітря стає істотним фактором.

Таким чином, математична модель — це майже завжди наближений опис якого-небудь процесу або явища. Від того, наскільки добре обрані об'єкти моделі й істотні фактори, залежить наскільки адекватна модель реальної ситуації. З'ясування адекватності моделі — складна і творча проблема. Використовуються різні підходи для її рішення (наприклад, проведення експериментів).

Якщо до математичного моделювання підходити з інших позицій, то важливим стає поняття області застосовності моделі. Як згадувалося вище, багато реальних задач різної природи зводяться до одній і тій же математичній моделі. Такі математичні моделі досліджуються і вивчаються незалежно від їхніх задач, що породили. З появою нової задачі іноді простіше з'ясувати, чи не є яка-небудь із уже досліджених математичних моделей адекватній цій задачі. Або, іншими словами, чи задовольняють умови нової задачі обмеження, у яких створювалася математична модель (зрозуміло, за умови, що сам процес або явище описується тими ж законами). Такі обмеження називаються областю застосовності математичної моделі.

Наприклад, задача падіння листа з дерева не входить в область застосовності моделі вільного падіння тіла: $v = g \cdot t$.

Дослідження математичної моделі — другий етап побудови моделі. Отримана математична модель — це математична задача. Бажано до її рішення провести аналіз. Важливо, чи коректно поставлена задача.

Нехай вхідні дані моделі x і вихідні дані в зв'язані функціональною залежністю f :

$$y = f(x).$$

Задача $y = f(x)$ називається коректною задачею, якщо виконані наступні умови:

- 1) розв'язок задачі існує;
- 2) він єдиний;
- 3) розв'язок стійкий.

Перша умова означає, що задача має рішення при будь-яких припустимих вхідних даних, тобто в залежності f немає суперечних один одному умов. Наприклад, у системі

$$\text{EMBED Equation.3} \quad \begin{cases} p + q = 1 \\ p + q = 2 \end{cases}$$

рівняння суперечать один одному, система не має рішення. Ця задача некоректна.

Друга умова означає, що вхідним даним x відповідає тільки одне рішення y , тобто інформації у функціональній залежності / досить для однозначності рішення.

Наприклад, у системі

$$\begin{cases} p + q = 2 \\ p + q = 4 \end{cases}$$

інформації для однозначності рішення недостатньо. Варіанти $p = q = 1$ і $p = 0, q = 2$ - два різні часткові рішення. Ця задача також некоректна.

Розглянемо третю умову коректності. Нехай деякі вхідні дані x_1 дають результат y_1 :

$$y_1 = f(x_1),$$

а вхідні дані x_2 дають результат y_2 .

$$y_2 = f(x_2),$$

Рішення стійке, коли вірна умова: якщо міра розходження між вхідними даними x_1 і x_2 досить мала, то і міра розходження між вихідними даними y_1 і y_2 менше будь якою наперед заданої малою величини. При цьому передбачається, що в просторі припустимих вхідних даних і в просторі можливих рішень встановлене поняття міри розходження (або міри близькості).

Наприклад, розглянемо задачу, формалізовану функціональною залежністю

$$\begin{cases} 1, \text{ при } x > 0, \\ 0, \text{ при } x = 0, \\ -1, \text{ при } x < 0. \end{cases}$$

Така функція називається знаком числа (sgn — «сигнум»). Її значення: 1 для додатних чисел, (-1) — для від'ємних і 0 — для нуля. Вхідні дані цієї задачі — x , а вихідні — y .

Під мірою розходження або близькості з 3-ї умови коректності будемо розуміти звичайну різницю чисел по

абсолютній величині. Наприклад, числа 1 і 10 розрізняються на 9: $|1 - 10| = 9$; а числа 1 і 0,9999 розрізняються на 0,0001: $|1 - 0,9999| = 0,0001$.

Розглянемо два набори вхідних даних $x_1 = 0$ і $x_2 > 0$. Відповідні вихідні дані — $y_1 = 0$ і $y_2 = 1$. Виберемо міру розходження між вихідними даними з 3 - ї умови коректності рівної 0,5. Тоді для ; цієї наперед заданій точності для вхідних даних $x_1 = 0$ і $x_2 > 0$, міра розходження між якими може бути зроблена як завгодно малої, міра розходження між відповідними вихідними даними при цьому дорівнює 1: $|y_1 - y_2| = |0 - 1| = 1$, і більше цієї наперед заданій точності. Для цієї задачі в точці $x = 0$ не виконується 3-я умова коректності.

Крім того, деякі задачі формально є коректними, тобто задовольняють умовам 1)...3), але лише дуже малі розходження між вхідними даними забезпечують малі погрішності вихідних даних. Наприклад, у системі

$$\begin{cases} p + q = 2, \\ p + 1,0001q = 2,0001 + \alpha \end{cases} \quad (1.1)$$

будемо вважати вхідними даними параметр α , а вихідними - p і q . Тобто , задаючи різні α , одержуємо систему двох рівнянь із двома невідомими, її розв'язок — p і q — вихідні даної задачі.

Розглянемо два набори вхідних даних: $\alpha_1 = 0$ і $\alpha_2 = 0,0001$. Тоді система (1.1) відповідно запишеться у вигляді:

$$\begin{cases} p + q = 2, \\ p + 1,0001q = 2,0001 \end{cases} ; \begin{cases} p + q = 2, \\ p + 1,0001q = 2,0002 \end{cases}$$

Розв'язок цих систем: $p_1 = q_1 = 1$ і $p_2 = 0, q_2 = 2$. Невелика погрішність вхідних даних $|\alpha_1 - \alpha_2| = 0,0001$ істотно змінила результат $|q_1 - q_2| = 1$.

Хитливі до погрішності вхідних даних за дачі мають невелику практичну цінність, тому що при можливій невеликій помилці вхідних даних може вийти результат, далекий від справжнього.

У техніці, фізиці й інших областях знань виникає необхідність вирішувати і некоректні задачі , але для їхнього рішення застосовуються спеціальний аналіз і інші методи. Надалі будемо розглядувати тільки коректні задачі.

У задачі падіння каменю рішення існує, тому що будь-якій годині падіння відповідає деяка швидкість. Рішення єдине,

оскільки кожному моментові падіння відповідає тільки одна швидкість. Рішення стійке, тому що невелика зміна вхідних даних спричиняє пропорційну зміну вихідних даних.

Аналіз коректності, звичайно, у компетенції професійних математиків. Дуже часто такий аналіз складний. Багато практичних задач використовуються без подібного дослідження, тому що для його проведення потрібні занадто великі витрати часу і сил. Але задачі, використовувані без аналізу — джерело ризику; можливо, що при деяких початкових даних буде отриманий невірний результат.

Часто корисно розглянути граничні випадки даної задачі або спрощені її постановки як аналіз.

Якщо задача, що представляє дану математичну модель, буде вирішуватися кількісно, з використанням ЕОМ, то необхідно розробити обчислювальний алгоритм.

Алгоритм — це точне розпорядження, що задає обчислювальний процес, що починається з довільного вихідного даного і спрямований на одержання цілком обумовленого цим вихідним даним результату. Є й інші визначення алгоритму, наприклад: алгоритм — це кінцева система правил, що перетворить вихідні дані в однозначний результат. І, нарешті, зовсім приблизно можна розуміти під алгоритмом правила розв'язку задачі.

Фактично обчислювальний алгоритм — це множина алгебраїчних формул і логічних умов на послідовність застосування цих формул. Для рішення однієї і тієї ж математичної задачі можна запропонувати, як правило, багато різних алгоритмів. Вони розрізняються по швидкодії, складності, точності, обумовленості, спільності й іншим ознакам. Розділ обчислювальної математики, що називаються чисельні методи, визначає критерії оцінки алгоритмів і займається побудовою ефективних обчислювальних алгоритмів.

Існують різні способи запису алгоритмів.

1) Словесний опис алгоритму.

Звичайно це пронумероване арабськими цифрами зі знаком «градус» опис етапів рішення задачі на природній людській мові (українській, англійській і ін.) Наприклад, для задачі падіння каменю:

1°. Початок процесу.

2°. Уведення часу падіння каменю.

3°. Знаходження швидкості як добутку часу падіння і

прискорення вільного падіння.

4°. Вивід отриманого результату.

5°. Кінець процесу.

2) Формульний опис алгоритму.

Під цим розуміється сукупність математичних формул, що показують перетворення вихідної і проміжної інформації в заданій послідовності. Наприклад, падіння каменю з висоти описується формулою **$v = gt$** .

І Словесний і формульний опис застосовуються рідко, частіше використовується їхня комбінація:

3) Словісно-формульний опис алгоритму.

. Застосовується в тому випадку, якщо задача складна. У , формульному описі алгоритму даються пояснення, що уточнюють або конкретизують шлях рішення. Словісно-формульний опис падіння каменю з висоти:

1°. Початок процесу.

2°. Уведення часу t .

3°. Обчислення швидкості по формулі **$v = gt$** .

4°. Вивід швидкості v .

5°. Кінець процесу.

Перераховані способи запису алгоритму застосовуються, як правило, на початкових етапах рішення задачі. При цьому ступінь деталізації може бути різної і залежить від багатьох причин.

4) Опис алгоритму у вигляді блок-схеми.

Зручним інструментом є блок-схема, якою можна представити графічно спроектований алгоритм. У блок-схемах для позначення різних фрагментів алгоритму використовуються стандартні символи. Ці символи прості у використанні і наочні. З їхньою допомогою оформляється алгоритм у графічному вигляді, що дозволяє швидко і легко охопити весь процес. Символи, з яких проектується схеми даних, алгоритмів, програм і систем, визначаються стандартами. Графічні алгоритми, оформлені відповідно стандарту у різних країнах розуміються однозначно. У табл. 1.1 приведені основні елементи, з яких проектується блок-схеми.

Якщо в алгоритмі кілька розривів лінії потоку, то в з'єднувачі розміщують ідентифікатор (цифру, букву і т.д.) для однозначної ідентифікації розірваних ліній.

У разі потреби (якщо алгоритм складний і складні процеси обробки інформації і необхідні пояснення і доповнення не тільки в символі коментарю, але й в інших місцях, наприклад, у

пояснювальній записці) символи позначаються ідентифікаторами — буквами, цифрами й ін. Ідентифікатори розташовуються над символами алгоритму ліворуч.

Таблиця 1.1		
Символ	Найменування	Застосування
	термінатор	відображає вхід у зовнішнє середовище і вихід із зовнішнього середовища (початок або кінець схеми алгоритму)
	дані	відображає дані (ввід і вивід інформації)
	процес	відображає процес обробки інформації комп'ютером
	рішення	відображає операцію вибору наступної дії
	визначений процес	відображає процес, що складається з операцій, що визначені в іншому місці (наприклад, у підпрограмі)
	лінія	відображає потік даних або керування (з'єднуючи символи між собою, лінії вказують, яке дія виконується наступної)
	з'єднувач	відображає вихід у частину схеми і вхід з іншої частини, якщо при цьому обривається лінія і продовжується в іншому місці (наприклад, якщо алгоритм не міститься на одному листі, то в розриві лінії потоку розміщують

		з'єднувач)
--	--	------------

На рис. 1.1 зображений алгоритм розв'язання задачі падіння каменю з висоти в графічному виді.

Наступний етап — складання програми, зводиться до перекладу алгоритму на обрану мову програмування.

Тому що сучасне програмування — це наука, то розумно використовувати її досягнення. Наприклад, такі технології програмування, як структурне, об'єктно-орієнтоване або функціональне програмування практично дають можливість використовувати методи, що помітно впливають на правильність прог

початок
І під правильністю програм різні школи програмування розуміють різне. Деякі правильністю програми вважають відсутність помилок у ній, інші — можливість формального висновку програми з формального набору передумов. Видимо, для практичних цілей більш важливе розуміння правильності програми в першій мірі. Різні технології програмування дозволяють використовувати закінчені фрагменти як елементи програми. Тому що правильність таких фрагментів доведена, то для написання правильної програми досить їх правильно застосувати.

швидкість
в
зילו, складні, точніше великого обсягу, програми розробляють і оформляють належним образом професійні програмісти. Однак програмісти - компетентні фахівці у своїй області, можуть бути слабкі в предметній області. Рідко зустрічаються фахівці, здатні розробити сучасну, досить об'ємну комп'ютерну програму, так це від них і не потрібно. Кожен фахівець повинний займатися своєю справою і виконувати її добре. Але сучасний фахівець повинний у достатній мірі бути підготовлений в області математики, інформатики і програмування, добре орієнтуватися в навколишньому його світі і володіти комп'ютерними технологіями як ефективним інструментом для

рішення своїх професійних задач. При цьому він зобов'язаний уміти не тільки усвідомлено і грамотно використовувати готові програми рішення технічних задач, довідкові й оформлювальні пакети, але і вміти поставити задачу, розробити алгоритм її рішення і консультувати програміста, що пише по заданому алгоритмі програму, по змістовних питаннях.

Сучасні програмні продукти крім предметної частини містять ще й елементи взаємодії з операційною системою, периферійними пристроями, користувачем і інші технічні і сервісні елементи.

Аналіз результату поєднує третій і четвертий етапи побудови моделі.

Інтерпретація отриманих рішень полягає в обробці вихідних даних, їхньому аналізі. Для цього потрібне розуміння сутності рішення задачі, відповідних математичних методів і особливостей застосовуваного обчислювального алгоритму.

За допомогою контрольних прикладів, окремих випадків, теоретичних розумінь, попередніх оцінок, зразкових розрахунків і іншого - з'ясовується, наскільки адекватна модель реальності, у якому місці виникли невідповідності. Шукаються можливості уточнення і поліпшення моделі. Потім знову виконується уся описана вище робота, починаючи зі зміненого місця. Зрештою виникає ситуація, коли модель задовольняє пред'явленим вимогам і результати її роботи можуть використовуватися практично.

Модель експлуатується доти, поки в ній є потреба, або поки вона задовольняє вимогам. Зміни в постановці задачі, удосконалювання математичних методів, розвиток обчислювальної техніки й інше підвищують вимоги до математичної моделі і роблять необхідної її модернізацію.

Математична модель модифікується, враховуються нові фактори і зміни в старих, і починається новий цикл побудови математичної моделі.

Наприклад, Аристотелева модель падіння тіла не узгоджувалася з реальністю: виходило, що швидкість вільно падаючого тіла не залежить від відстані, що воно пролетіло. Цей факт і деякі інші зажадали модернізації моделі, що і зробив Галілей.

1.3. Моделювання у техніці

Інженери звернулися до математики ще в XVI столітті. У XIX столітті наростаюче зростання виробництва супроводжувалося ускладненням технологій, структури виробничих зв'язків, що

волокло збільшення обсягу обчислень. Спочатку інженери звернулися до математики для полегшення і прискорення таких обчислень. Крім того, невеликі і відносно нескладні виробничі комплекси, керувати якими можливо, спираючи тільки на інтуїцію і досвід людини, змінилися складними комплексами, цілісне представлення про які неможливо мати, не прибігаючи до точних методів і технічних засобів уже хоча б тому, що фізичні можливості людини обмежені. Керування такими комплексами зажадало залучення математичних методів.

Оптимізація технологічних процесів, виконана за допомогою математичних методів, враховувала різноманітні факторів, недоступні для аналізу тільки на основі досвіду, давала значну вигоду.

На перших етапах уведення математичних методів у виробництво переслідувалася мета - перевірка залежностей і зіставлення результатів розрахунків по моделі з реальними показниками. Потім формулювалися гіпотези, створювалися моделі і ставився обчислювальний експеримент.

Обчислювальним експериментом називають проведення комплексних розрахунків при розв'язанні задач. Обчислювальний експеримент на відміну від фізичного проводиться не над реальними об'єктами, а над їхніми математичними моделями. Обчислювальний експеримент має настільки значні практичні результати у всіх областях людської діяльності, що правомірно говорити про обчислювальний експеримент як про нову технологію і методологію досліджень. Обчислювальний експеримент має істотні переваги в порівнянні з фізичним: він, як правило, дешевше; у нього можна легко і безпечно втручатися; його можна повторювати і переривати в будь-який момент; легко досягається зміна масштабу часу, що дозволяє швидко одержувати результати довго триваючих у реальних умовах процесів (наприклад, що вимагають часу життя декількох поколінь дослідників при одержанні нових видів рослин, живих організмів, аналізі процесів зміни атмосферних, кліматичних і інших наддовгих процесів); у ході його можна змодельовувати умови, який не можна створити в лабораторії або на полігоні.

У ряді випадків фізичний експеримент узагалі неможливий. Вивчення швидкодіючих процесів, важкодоступних і недоступних об'єктів, явищ з непередбаченими наслідками можна легко і безпечно виконувати, розглядаючи їхні моделі. Деякі енергетичні процеси відносяться до тих, натурні експерименти в які взагалі

неможливі. Обчислювальний експеримент дозволяє вивчати їх.

Один з істотних недоліків обчислювального експерименту — обмеження області застосовності прийнятою моделлю. Однак, незважаючи на недоліки, обчислювальний експеримент є ефективним засобом дослідження з безупинно розширююся областю застосування.

1.4. Інструментарії комп'ютерного моделювання

Для моделювання використовуються багато засобів. Але в енергетиці сформувався чітку виражений круг засобів для моделювання.

По перше це потужний апарат сучасної вищої математики, за допомогою якого можливо обробляти експериментальні залежності, знаходити екстремуми функцій, складати і розв'язувати звичайні та диференціальні рівняння, досліджувати отримані залежності.

З вищої математики поступово виділився круг задач прикладного аспекту як математичне програмування, теорія масового обслуговування, управління запасами, теорія ігор, які складають сучасну прикладну математику. Суттєвим моментом задач прикладної математики є те, що вони мають велику розмірність і розв'язок реальних задач можливий лише за допомогою електронних обчислювальних машин.

Використання електронних обчислювальних машин, особливо персональних, зробило революційний крок у моделюванні технічних та економічних процесів. З'явилася можливість використання алгоритмів моделювання, які неможливо було використовувати традиційними способами. З'явився новий метод моделювання, який базується на прямому відтворенні моделюваної системи програмними засобами з штучною імітацією параметрів, від яких залежить система. Цей метод отримав назву імітаційне моделювання і він використовується там, де аналітичні методи не дають належної точності. Слід зауважити, що імітаційне моделювання потребує багато часу на програмування, налагодження, прогін моделей та обробку отриманого чисельного матеріалу.

Розвиток обчислювальної техніки та програмування дав можливість створити так звані системи комп'ютерної математики, які поєднують у собі властивості редакторів тексту, мов програмування і мають велику кількість убудованих математичних функцій та методів розв'язку основних задач математики. До цих

систем належать: **Mathcad, Mathematica, Matlab, Maple**. Вони мають сучасний інтерфейс, чудові властивості по побудові графіків. Значним досягненням є те, що деякі з них здібні виконувати аналітичні перетворення. Найбільш потужними властивостями у цьому напрямку володіє система **Maple**, особливо її останні модифікації **Maple 6 - Maple 9**, які будуть розглянуті нижче.

Контрольні запитання і вправи

1. Що таке модель? Приведіть приклади фізичних і абстрактних моделей.
2. Що таке математична модель? Приведіть приклади математичних моделей природних явищ і енергетичних процесів.
3. Чому з'явилася необхідність у вивченні систем?
4. Наведіть приклади моделей у фізиці.
5. Що є об'єктом вивчення й інструментом дослідження системного аналізу як наукової дисципліни?
6. Як функціонує математична теорія систем?
7. Опишіть системний аналіз як технологію дослідження.
8. Використовуючи п'ять кроків аналізу системи, розгляньте механізм роботи: а) турбіни; б) електричного ланцюга; в) фірми.
9. Назвіть основні етапи створення математичної моделі.
10. Що таке алгоритм? Приведіть приклад алгоритму. Складіть алгоритм розв'язку квадратного рівняння.
11. Що таке комп'ютерне моделювання ?
12. Які сучасні математичні системи використовуються у комп'ютерному моделюванні ?

2. Математична системи Maple

2.1. Загальна характеристика математичної системи Maple

Maple – це інтегрована математична система, орієнтована на символічний та чисельний розв'язок математичних задач, що має могутню убудовану мову програмування (подібну Бейсик). Кількість використовуваних функцій, що охоплюють усі розділи математики і суміжних дисциплін, приблизно дорівнює 3000, що практично виключає програмування при розв'язанні конкретних задач.

Основні можливості:

- Розв'язок задач лінійної алгебри.

- Розв'язок задач математичного аналізу.
- Інтегральні перетворення Фур'є, Лапласа.
- Розв'язок диференціальних рівнянь.
- Розв'язок задач теорії ймовірностей і математичної статистики.
- Розв'язок задач статистичної обробки інформації.
- Числова апроксимація.
- Графічна візуалізація обчислень, побудова двовірних і тривірних графіків у різних системах координат.
- Завдання анімаційних графіків і їхнє програвання.
- Інтеграція з матричною системою MATLAB, яка дозволяє виконувати деякі операції значно ефективніше.

Maple дозволяє підготовляти файли в HTML форматі для передачі по мережі Internet, а також перетворювати файли у формат LaTeX.

Система **Maple** реалізована в середовищі операційної системи Windows і має сумісний з нею інтерфейс. Система має прекрасно виконану довідкову систему з великою кількістю прикладів.

Основним об'єктом **Maple** для введення інформації, що згодом заноситься у файл, є документ - робочий лист (**Worksheet**). Новий документ утворюється по команді **File/New**. Спочатку він має ім'я **Untitled(N)**, $N=1,2,\dots$. Після редагування документ може бути записаний на диск під своїм ім'ям командою **File/Save As...**

Мінімальними структурними одиницями документа **Maple** є параграф (Paragraph) і група (Execution Group). Параграфи складаються з текстових рядків і використовуються для підготовки текстової інформації. Група є основним діалоговим об'єктом **Maple** і складається з осередку введення, куди заносяться команди, текстові коментарі, і осередку виведення, куди система **Maple** заносить результати обчислень, графіки, різні повідомлення. У групу можуть включатися і параграфи. Група обрамляється ліворуч прямокутною скобою ([). Запрошенням **Maple** для введення команди служить знак >, при введенні тексту він відсутній. Позиція для введення чергового символу відзначається курсором (|). Для перемикання осередку введення між режимами введення команди або текстового коментарю використовується команда

Edit/Input mode або клавіша **F5**. Для введення параграфа використовується команда **Insert/Paragraph►** (до курсору чи після курсору). Для введення групи в довільне місце робочого листа використовується команда **Execution Group►** (до курсору чи після курсору). Для введення параграфів і груп після поточного положення курсору можна використовувати відповідні кнопки панелі інструментів.

Робота з **Maple** в основному ведеться в інтерактивному режимі: запит у вигляді математичного виразу, команди – результат, запит - результат і т.д., хоча можливий і чисто програмний підхід до розв'язку конкретних задач. Запис функцій і команд ведеться на Maple-мові, яка багато в чому схожа на звичайні мови програмування. У **Maple** можливий запис математичних виразів у природній математичній формі з використанням спеціальної палітри математичних символів. Якщо запис ведеться на Maple – мові, то отриманий вираз можна перетворити в природну математичну форму. Для цього необхідно поставити курсор на початок команди і клацнути мишею по кнопці **X** що, знаходиться на початку панелі форматування вхідних виразів.

Для підготовки наукових публікацій використовується так звана інертна (невиконувальна) форма команд. Запис виразів у інертній формі практично нічим не відрізняється від звичайного тексту, крім того, що функції в інертній формі починаються з великої букви; мається можливість за допомогою другої кнопки на панелі форматування вхідних виразів перейти від інертної форми математичного виразу в активну.

Розглянемо копію екрана **Maple** з типовими прикладами роботи із системою, зображену на рис.2.1.

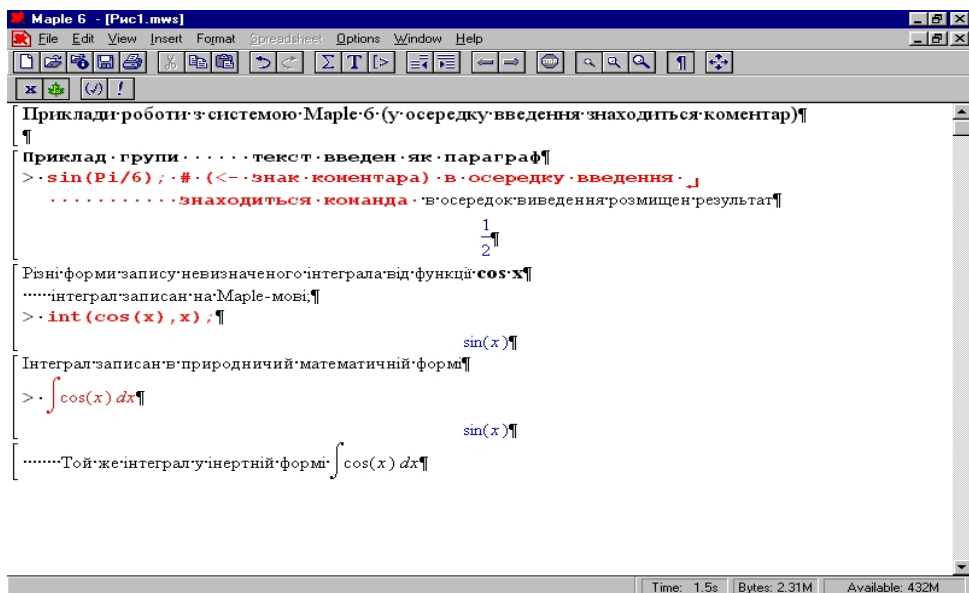


Рис.2.1

Увесь екран поділяється на кілька областей. Нагорі знаходиться титульний рядок з назвою системи та ім'ям поточного документа. Під ним знаходиться рядок головного меню. Ще нижче знаходиться панель інструментів, а нижче її знаходиться панель форматування вхідних виразів. При введенні тексту, панель форматування вхідних виразів замінюється звичайною панеллю форматування. Основну частину вікна займає робоча область, заповнена групами і параграфами. Математичні вирази, записані на Maple-мові, закінчуються знаком ; або : (так звані фіксатори).

Процес роботи з робочим листом називається сеансом; як видно з рис.2.1 практично весь сеанс складається з окремих груп, що реалізують відпрацювання математичних виразів, що вводяться.

2.2. Основні засоби керування Maple

До засобів керування **Maple** відносяться: головне меню, панель інструментів, панель форматування (для текстових виразів), панель форматування вхідних виразів (для команд на **Maple-мові**), контекстні меню, викликувані клацанням правої кнопки

миші по тому чи іншому об'єкті. Для набору математичних виразів можуть використовуватися символна палітра (для набору грецьких букв), палітра математичних виразів (рис. 2.5) і палітра для заповнення матриць. Виклик цих палітр здійснюється командою **View/Palettes►**. У **Maple** запроваджена можливість роботи з електронною таблицею (**Spreadsheet**). Виклик режиму роботи з електронною таблицею здійснюється командою **Insert/Spreadsheet**, після чого стають доступними команди пункту головного меню **Spreadsheet**.

Для швидкого виконання команд можна використовувати гарячі клавіші.

При виведенні графіка і клацанням по ньому мишею, з'являється панель керування графіками, яка дозволяє вибрати вісі координат, тип лінії графіка. За допомогою миші можливо змінити розмір графіка, а тривимірні графіки повернути у просторі, для кращого спостереження.

Пункти головного меню **Maple** максимально уніфіковані з операційними системами Windows. Природно, додані специфічні команди.

Призначення кнопок панелі інструментів (**Tool Bar**) представлено на рис. 2.2.



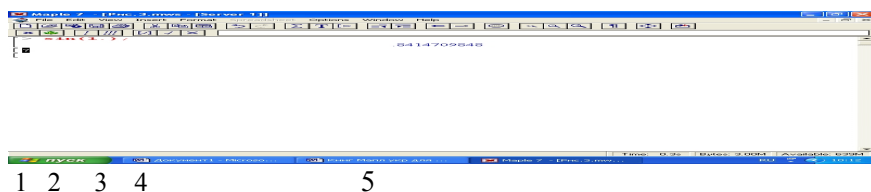
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
22 23

- | | |
|--|--|
| 1 - Утворення нового документа | 2 – Відкриття документа |
| 3 – Відкрити WEB сторінку | 4 - збереження активного документа |
| 5 - Печатка активного документа | 6 - Вирізання 7 - Копіювання 8 - Вставка |
| 9 - Скасування останньої операції | 10 – Відновлення скасованої операції |
| 11 – Перемикання осередку введення на введення математичного виразу в інертній формі | |
| 12 - Перемикання осередку введення на введення тексту | |

- 13 - Вставка нової групи
- 14 – Вилучення поточної секції
- 15 - Створення секції або підсекції
- 16 – Вилучення підсекції, до якої входить поточний вираз.
- 17 – Включення поточного виразу у підсекцію.
- 18 - Переривання обчислень
- 19 - Виведення малими символами (100%)
- 20 - Виведення середніми символами (150%)
- 21 - Виведення великими символами (200%)
- 22 - Виведення символів, що не друкуються
- 23 - Повне розкриття активного вікна

Рис. 2.2

Призначення основних кнопок панелі форматування вхідних математичних виразів приведене на рис. 2.3.

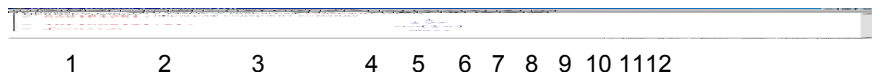


- 1 – Переключення між стандартною і текстовою формою математичного виразу.
- 2 – Переключення між активною і інертною формою математичного виразу.
- 3 – Виконання поточного виразу.
- 4 – Виконання робочого листа.
- 5 – Поле виразу, що редагується.

Рис.2.3

Набір команд у цій панелі різний для різних версій **Maple**.

На рис.2.4 представлена панель форматування текстових виразів із призначенням окремих кнопок

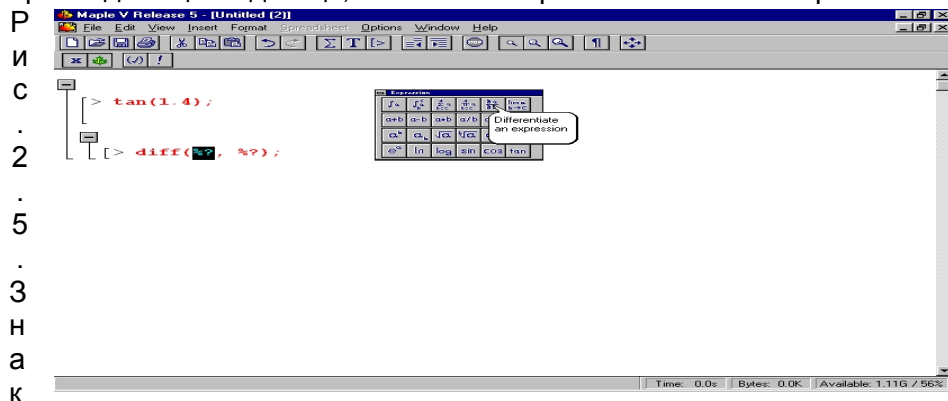


- 1 – Поле стилю
- 2 – Перемикач стилю
- 3 - Поле вибору шрифту
- 4 - Перемикач шрифтів
- 5 - Поле розміру символів шрифту
- 6 - Перемикач розмірів шрифтів
- 7 - Символи напівжирні
- 8 - Символи похилі
- 9 - Символи підкреслені знизу
- 10 - Вирівнювання тексту ліворуч
- 11 - Вирівнювання тексту по середині
- 12 - Вирівнювання тексту праворуч

Рис. 2.4

Математичні вирази, що вводяться в осередок введення повинні закінчуватися фіксаторами (; чи :). Символ «;» задає вивід результатів обчислень в осередок виведення, символ «:» блокує вивід результатів обчислень. Закінчення введення виразу фіксується натисканням клавіші Enter. Якщо при введенні вираз не міститься на одному рядку, то він автоматично переноситься на наступний рядок. Натисканням комбінації клавіш Shift+Enter можна задати примусовий переніс виразу на новий рядок. Це робиться для додання документу наочності.

Групи можуть поєднуватися в так називані секції і підсекції. Секція починається з кнопки зі знаками + і – . На рис. 2.5 приведен приклад секції і підсекції, а також палітри математичних виразів.



+ показує, що секція закрита і усе, що в ній знаходиться, невидимо, але бере участь у всіх обчисленнях. Знак – показує, що секція відкрита. У секції доцільно поміщати допоміжні обчислення, що не варто обдивлятися постійно. Створення секцій і підсекцій здійснюється командами **Insert/Section**, **Insert/Subsection** і відповідної кнопкою на панелі форматування вхідного вираз. Відкриття і закриття секцій здійснюється клацанням по кнопці початку секції, а також командами **View/Expand All Sections** і **View/Collapse All Sections**.

2.3. Основи роботи з Maple

У загальному випадку **Maple** є програмною системою для роботи з математичними виразами. Математичний вираз може містити оператори, операнди і функції з параметрами. Вирази в

Maple можна змінювати (еволюціонувати) відповідно до математичних законів або заданими правилами перетворення. Наприклад, функція спрощення виразів **simplify** здатна спрощувати багато математичних виразів, записаних в якості її параметрів:

```
>simplify((a^2-2*a*b+b^2)/(a-b));
```

a-b

Як вже відзначалося раніше, математичний вираз повинен закінчуватися так званим фіксатором: це чи крапка з комою ; чи двокрапка : . Символ ; закінчує (фіксує) вираз і задає вивід результатів, символ : блокує вивід результатів (самі обчислення при цьому можуть виконуватися). Фіксатори виконують роль роздільників виразів, якщо в рядку їх декілька. Уведення виразу закінчується натисканням клавіші **Enter**. При цьому курсор уведення може бути в будь-якій позиції рядка.

За допомогою від одного до трьох знаків % (" у реалізації R4) можна викликати на виконання перший, другий або третій вираз з кінця сеансу (навіть якщо знак % знаходиться в середині сеансу):

```
>sin(2.);
.9092974268
>tan(1.4);
5.797883715
>%%;
.9092974268
```

Строкові дані, якщо вони знаходяться не в параграфі або у текстовому коментарі, обрамляються знаками зворотного апострофа ' :

```
>int(cos(x),x);
sin(x)
>'int(cos(x),x)';
int(cos(x),x)
```

Для вставки коментарю в рядок, що містить математичний вираз, використовується знак # :

```
>diff(x^n,x);# це похідна
n x^n/x
```

Коментар можна вставити безпосередньо в оператор.

```
diff(x^n,x);'this is the differentiation ';
```

n x^n/x

this is the differentiation

Варто помітити, що з українськими буквами в командах система працює нестійко.

У **Maple** використовується ряд **визначених констант**:

Pi – $\pi = 3,141\dots$;

I – мнима одиниця;

infinity – нескінченність;

як основу натурального алгоритму - числа **e** рекомендується використовувати **exp(1)** .

Maple виводить результати обчислень з певною кількістю значущих цифр (за замовчуванням 10 цифр). Командою

>Digits:=n :

можна задати вивід **n** значущих цифр у чисельних виразах. Цією командою можна регулювати точність виводимих чисел.

Maple складається з ядра, написаного мовою програмування **Ci**, в якому знаходяться ретельно оптимізовані процедури, і пакетів розширення, написаних на Maple-мові, які поставляються із системою. Наведемо назву деяких із цих пакетів: лінійної алгебри; лінійної оптимізації; розв'язку диференціальних рівнянь; функцій комбінаторики; фінансово-економічних функцій; ортогональних багаточленів; чисельної апроксимації; інтегральних перетворень; тензорів і теорію відносності; симетрії Лі. Пакети можуть доповнюватися, модифікуватися, чим забезпечується розширення й адаптація системи.

При звертанні до ядра відразу указується відповідна функція, а при звертанні до пакета спочатку вказується ім'я пакета, а потім потрібна функція. Наприклад,

>with('linalg'); # виклик пакета Розв'язок задач лінійної алгебри

>M:=matrix(2,2,[a,b,c,d]); # формування квадратної матриці і її заповнення

$$M = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

Результати, що виводяться при виконанні аналітичних і чисельних перетворень не завжди виводяться в потрібній формі. Якщо вираз - скалярна змінна, то її значення буде виведено в осередок виведення. Для змінних більш складних типів виводяться

не їхні значення, а просто повторюється ім'я змінної. Просто повторюються також ім'я невизначених змінних. Для одержання результату в потрібному вигляді, його у ряді випадків необхідно піддати спеціальній процедурі оцінювання. Ця процедура виконується за допомогою декількох спеціальних функцій: **eval(expr)**, **evalf(expr,n)**, **evalm(mexpr)** та ін. Наприклад

```
>sin(1.);
.8414709848
>sin(1);
sin(1)
>evalf(sin(1),4);
.8415
```

Багато математичних виразів мають різні тотожні форми. Часто перетворення виразу з однієї форми у іншу дозволяє одержати результат більш наочний і зручний для аналізу. Крім того, різні функції **Maple** працюють з різними формами виразів і різних типів даних. Тому велике значення має цілеспрямоване перетворення виразів і даних. Основною функцією для такого перетворення є функція **convert(expr, form, opt1,opt2,...)**, де **expr** – вираз, яких необхідно перетворити, **form** – найменування форми, у яку буде перетворен вираз, **opt** – опції перетворення. Список форм нараховує 67 найменувань:

decimal – задає перетворення числа в десяткове представлення з двоїчного, восьмеричного і шістнадцятиричного;
float - задає перетворення числа у дійсне представлення;
fraction - задає перетворення числа в раціональне у вигляді відносини цілих чисел;

exp - задає перетворення функції в експонентний вид та інше. Приклад:

```
>convert(2.46,fraction); # перетворення числа з дійсного в дробове
```

123/50

Ще однією могутньою функцією перетворення виразів є функція **combine**. Вона забезпечує об'єднання показників показових функцій, перетворення тригонометричних і деяких інших функцій.

Ця функція може записуватися у трьох формах:

```
combine(f)
combine(f, name)
combine(f, name, opt1, opt2, ...)
```

де: **f** — будь-який вираз, список виразів, **name** - ім'я, чи список імен, **opt1**, **opt2**, ... — імена опцій.

В другому аргументі можна використовувати наступні функції:

exp **ln** **power** **trig** **Psi** **polylog** **radical** **abs**
signum

plus **atatsign** **conjugate** **plot** **product** **range**

Як опції можуть вказуватися наступні:

arctan **atatsign** **conjugate** **exp** **ln** **polylog** **power**

Psi **radical** **trig**

Приклади застосування функції **combine** представлені

нижче:

```
> combine(exp(2*x)^2,exp);
                                exp(4 x)
> combine(2*sin(x)^2+2*cos(x)^2);
                                2
> combine(sin(x)*cos(x));
                                0.5 sin(2x)
```

Maple працює з числами, символьними даними, списками, векторами, матрицями, наборами і таблицями.

Числа, з якими працює **Maple** можуть бути: цілими десятковими (1.325), раціональними у вигляді відносини цілих чисел (123/50), дійсними (1.56, 123.5623E-12), комплексними (4+6*I), двоїчними (10011101), восьмиричними і шістнадцятиричними (B23A).

Символьні дані - це просто ланцюжка символів, що звичайно використовуються для створення текстових блоків. В одних випадках **Maple** сам ідентифікує рядок як текст (при введенні параграфа, при введенні інформації в осередок уведення як текст - немає символу >). При цьому у рядок додається внутрішня ознака. В інших випадках необхідно явно вказати, що послідовність символів є рядок, обрамляючи її зворотними апострофами ' ('sin(x)') .

Об'єкти **Maple** можна поєднувати у множини і списки.

Множини створюються за допомогою фігурних дужок { , } :

```
>B:={c,f,w+n,3,7,7,i};
```

```
B := {3, 7, i, f,3, w+n}
```

У множині автоматично усуваються повторювані елементи й

елементи множини можуть переставлятися відповідно до внутрішніх правил.

Для двох множин можна виконувати оператори об'єднання (**union**), перетинання (**intersect**) і виключення (**minus**). Приклади:

```
> {d,u,o,r,r,e} union {d,t,q,r};  
      {t, u, r, d, o, e, q}  
  
> {d,u,o,r,r,e} intersect {d,t,q,r};  
      {r,d}  
  
> {d,u,o,r,r,e} minus {d,r};  
      {u, o, e}
```

Доступ до елементів множини **B** с *i*-го по *j*-й можна здійснити виразом **B[i..j]**, до *i*-го елемента – **B[i]**. Порожня множина задається виразом **[]**.

Списки створюються квадратними дужками **[,]**.

```
> D1:=[5,7,3,5,f,w,b,h];  
      D1 := [5, 7, 3, 5, f, w, b, h]
```

Списки перетворюються і виводяться строго у тім порядку, в якому вони були задані. Списки широко застосовуються для завдання векторів і матриць. Доступ до елементів списку здійснюється аналогічно множинам.

2.4. Робота з довідковою системою Maple

Maple має дві взаємозв'язані системи довідок. Одна з них оперативна (online) дозволяє викликати потрібну довідку в командному режимі. Для цього досить набрати в командному рядку слово **help** чи **?** (знак питання) з наступною вказівкою слова, по якому шукається довідка, наприклад:

```
> help(sin);  
  
чи  
  
> ?sin
```

У даному випадку після натискання клавіші **Enter** буде отримана довідка по функції синус. Таке звертання застосовується для бібліотечних функцій і зарезервованих (ключових) слів системи. Його недоліком є те, що потрібно знати ім'я оператора

функції, по якій очікується одержання довідки. На екран дисплея буде виведене докладне повідомлення (довідка) англійською мовою про призначення і правила запису (синтаксис) відповідної функції, чи оператора пакета застосувань.

Дуже корисним засобом є виділення і вибір прикладів з довідок. Для цього використовується клавіша **Ins**, за допомогою якої помічають початок і кінець виділень. Можна також виділити приклад, переміщаючи мишу з натиснутою лівою клавішею. Нарешті, використовуючи команду **Copy Examples** у позиції **Edit** головного меню вікна довідки, можна скопіювати в буфер усі приклади. Після цього, закривши вікно довідки, можна перенести в документ наявні в буфері приклади.

Можна також виділити будь-як слово в тексті, по якому необхідно одержати довідку, і вибрати пункт **Help** головного меню. Крім того можна установити курсор на будь-яке слово і натиснувши клавішу **F1**, одержати довідку про це слово (якщо вона мається в системі).

Основні засоби керування довідковою системою **Maple** зосереджені в позиції **Help** головного меню. Довідкова база даних має зручну деревоподібну структуру контекстно-залежного пошуку.

У ході пошуку інформації спочатку вказуються її загальні ознаки, потім частки – і так далі (див. рис. 2.6).

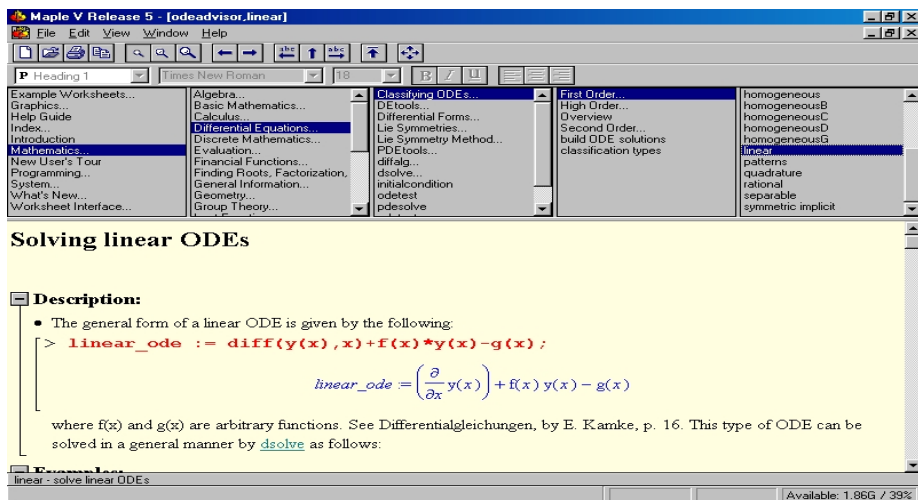


Рис. 2.6

2.5. Основні оператори, команди і функції ядра Maple

Оператори **Maple** служать для конструювання виразів і застосовуються разом з операндами. До основних операторів **Maple** відносяться:

+ додавання; **-** віднімання; ***** множення; **/** ділення;
****** або **^** зведення у степінь; **:=** присвоєння; **mod** обчислення модуля (залишку від ділення цілого числа **a** на ціле число **b**);
! факторіал, **.** або **cat** конкатенація (об'єднання); **and** логічне І;
or логічне АБО і т.д.

Крім того користувач має можливість створювати свої оператори за допомогою оператора **define**.

Для скасування присвоєння значень окремим змінним (наприклад **x**), які надалі можна не використовувати, можна використовувати вираз **>x:=`x`**. Для одночасного скасування присвоєння значень усім змінним можна використовувати команду **restart**.

У відмінності від інших мов програмування, в системі **Maple** істотно розширені функції оператора присвоювання (**:=**). З його допомогою змінної можна привласнити значення якої-небудь константи, виразу, рівняння або навіть системи рівнянь. Наприклад

T1:=diff(y(x),x)-y(x)=sin(x)*x

T1 привласнене значення диференціального рівняння. Це дозволить при розв'язанні даного диференціального рівняння звертатися до нього за допомогою змінної **T1**.

Maple має повний набір елементарних математичних функцій, функцій математичного аналізу, лінійної алгебри і т.п.

Більшість елементарних функцій має позначення, прийняте в математиці : **sin(x)**, **cos(x)**, **tan(x)**, **exp(x)**, **ln(x)**, **arcsin(x)**, **cot(x)** – котангенс, **sqrt(x)**, **!** - факторіал. Їх повний список можна знайти у довідці в розділі

Mathimatics/Basic Mathimatics/initially known functions.

Їхні інертні аналоги починаються з великої букви.

Суми ряду обчислення (**f(m)+f(m+1)+f(m+2)+...+f(n-1)+f(n)**) можна виконати за допомогою функцій **sum(f,k)** і **sum(f,k=m..n)**, де **f** – функція, що задає члени сумуємого ряду, **k** – індекс підсумовування, **m** і **n** – цілочисельні межі зміни **k**. Якщо **n=∞**, тоді для **n** використовується константа **infinity**. Приклад:

> sum(40*(0.5)^k,k=0..infinity);

Добутків членів послідовності обчислення можна виконати за допомогою функції **product(f,k=m..n)**.

Границя функції обчислюється функціями границь, що мають вигляд:

limit(f,x=a) чи **limit(f,x=a,dir)**

де **f** – алгебраїчний вираз, **x** – ім'я змінної, **a** – точка, у якій обчислюється межа, **dir** – опція, що вказує на напрямок пошуку границі (**left** – ліворуч, **right** – праворуч). Приклад

>limit(sin(x)/x,x=0);

1

Похідної функції обчислення ($f'(x) = df(x)/dx$) реалізується функцією **diff(f(x),x)**

Приклад:

>diff(sin(cos(x)),x);

-cos(cos(x))*sin(x)

Для обчислення похідних **n-го** порядку, а також для обчислення частинних похідних від функції з декількома аргументами використовуються функції:

diff(f(x),x\$n) , **diff(f(x,y,z...,u),x,z)**.

Поряд з функцією **diff** може використовуватися диференціальний оператор **D**. У формі **D(f(x))**, де **f** – тільки ім'я функції, він еквівалентний **diff(f(x),x)**. З його допомогою можна вказувати значення похідних при конкретних значеннях змінних і тим самим запроваджувати початкові умови в диференціальні рівняння. Приклади:

> D(sin)(x);

cos(x)

D(sin)(Pi/3);

1/2

> D(D(sin))(3.141/6);

-.50

Інтегралів невизначених і визначених обчислення можна виконати за допомогою функцій **int(f(x),x)** і **int(f(x),x=a..b)**. Для обчислення кратних інтегралів функція **int** використовується

багаторазово.

Приклад: обчислити подвійний інтеграл

$$\int_0^1 \int_{\sqrt{y}}^{y^2} (2-x-y) dx dy$$

```
> int(int((2-x-y),x=sqrt(y)..y^2),y=0..1);
```

-11/30

Рівнянь лінійних і нелінійних розв'язок можна одержати в аналітичному вигляді за допомогою функції **solve(eqn, var)** або **solve({eqn1,eqn2,...},{var1,var2,...})**, де **eqn** – рівняння, що містить функцію однієї чи декількох змінних, **var** – змінна, по якій шукається розв'язок. Якщо розв'язок містить повторювані вирази, то **Maple** позначає їх як **%1**, **%2**, ..та виводить розв'язок з такими змінними.

При розв'язку система рівнянь і список змінних задаються як множини у фігурних дужках. При цьому і результат розв'язку виводиться у вигляді множини. Щоб перетворити його до звичайного розв'язка, потрібно використовувати функцію **assign**, що забезпечує привласнення змінним значення, узятих з множини. Приклади:

```
> solve(x^2+p*x+q=0,x);
```

$$-\frac{1}{2}p + \frac{1}{2}\sqrt{p^2 - 4q} ; \frac{1}{2}p - \frac{1}{2}\sqrt{p^2 - 4q} ;$$

```
> s:=solve({3*x+y=a,x-y=b},{x,y});
```

$$s:=\{y= -3/4 b + 1/4 a , x=1/4 b + 1/4 a\}$$

```
> x;
```

x

Доступу до елемента множини поки немає.

```
> assign(s):
```

```
> x;y;
```

$$1/4 b + 1/4 a$$

$$-3/4 b + 1/4 a$$

```
> x;
```

$$1/4 b + 1/4 a$$

Чудовою можливістю функції **solve** є можливість розв'язку систем лінійних рівнянь щодо обмеженого числа змінних.

Наприклад, систему трьох рівнянь із змінними **x, y, z, v, t** можна вирішити відносно тільки перших трьох змінних **x, y, z**. При цьому розв'язок буде функціями щодо змінних **t** і **v**:

$$> e1 := 2 \cdot x + 10 \cdot y + 16 \cdot z - 6 \cdot t + 8 \cdot v = 20:$$

$$> e2 := 3 \cdot x + 2 \cdot y + 4 \cdot z + 8 \cdot v = 20:$$

$$> e3 := 6 \cdot x - 4 \cdot y + z + 3 \cdot t + 5 \cdot v = 10:$$

$$> e4 := 8 \cdot x - 2 \cdot y + 4 \cdot z - 3 \cdot t + v = 0:$$

$$> e5 := 3 \cdot x + 2 \cdot y + 4 \cdot z + t + 5 \cdot v = 30:$$

$$> S1 := \text{solve}(\{e1, e2, e3, e4, e5\});$$

$$S1 := \left\{ t = \frac{431}{65}, z = \frac{-122}{39}, v = \frac{-73}{65}, x = \frac{266}{39}, y = \frac{2051}{195} \right\}$$

$$> S2 := \text{solve}(\{e1, e2, e3\}, \{x, y, z\});$$

$$S2 := \left\{ y = \frac{-64}{23}v + \frac{530}{69} - \frac{1}{23}t, x = \frac{-68}{23}v + \frac{520}{69} - \frac{14}{23}t, \right.$$

$$\text{EMBED Equation.3} \quad \left. z = \frac{37}{23}v - \frac{310}{69} + \frac{11}{23}t \right\}$$

$$> \text{Digits} := 3:$$

$$> \text{convert}(\%, \text{float});$$

$$\{y = -2.78v + 7.68 - .0435t, x = -2.96v + 7.54 - .609t,$$

$$\text{EMBED Equation.3} \quad z = 1.61v - 4.49 + .478t\}$$

$$\text{assign}(S2):$$

$$x;$$

$$x = \frac{-68}{23}v + \frac{520}{69} - \frac{14}{23}t$$

Призначення для імен виконується за допомогою функції **assign**, що має форми **assign(a,b)**, **assign(a=b)**, де **a** – ім'я, **b** – вираз, та **assign(t)**, де **t** – множина або набір рівнянь. Остання форма часто використовується при розв'язку

систем рівнянь, коли треба явно привласнити змінним розв'язок, узятий з множини отриманого розв'язку, як у попередньому прикладі. Ще приклади:

```
>assign(a,2):
```

```
>a;
```

2

Після використання змінної призначення для неї доцільно скасувати. Виконати це можна за допомогою функції **unassign** або шляхом узяття в прямі апострофи:

```
>unassign('x');
```

```
>y:='y';
```

```
>x;
```

x

```
>y;
```

y

У розв'язках рівнянь нерідко з'являється функція **RootOf**, що означає, що корені є розв'язком визначеного рівняння. Іноді вдається одержати ці корені в явному виді за допомогою функції **allvalues**:

```
> solve({x*y=a,x+y=b},{x,y});
```

```
{y = RootOf(_Z^2 - _Zb + a), x = -RootOf(_Z^2 - _Zb + a) + b}
```

```
> allvalues(%);
```

$$\left\{ y = \frac{1}{2}b + \frac{1}{2}\sqrt{b^2 - 4a}, x = \frac{1}{2}b - \frac{1}{2}\sqrt{b^2 - 4a} \right\}$$

$$\left\{ y = \frac{1}{2}b - \frac{1}{2}\sqrt{b^2 - 4a}, x = \frac{1}{2}b + \frac{1}{2}\sqrt{b^2 - 4a} \right\}$$

Нерівностей і систем нерівностей розв'язок можна також одержати за допомогою функції **solve** :

```
> solve({x*y*z>0,x>-1,y+z>10},{x,y,z});
```

```
{y = 0, -1 < x, 10 < z}, {-1 < x, z = 0, 10 < y}
```

Виділення правої або лівої частини виразу

здійснюється за допомогою функцій **rhs** та **lhs**. Ці функції застосовуються при конструюванні функцій разом з функцією **unapply**, коли результат розв'язка диференціального рівняння або якого-небудь перетворення виходить у форматі, що не обчислюється.

```
> a2:=y=x^3-2*x-6:
```

```
> z2:=rhs(a2);
```

$$z2 := x^3 - 2x - 6$$

Конструювання функцій.

Система **Maple** орієнтована на символічні перетворення. Тому запис **f(t)** означає просто символічний вираз. Щоб придати **f(t)** зміст функції від аргументу **x** використовується функція **unapply**. Звичайно вона застосовується за схемою:

>f:=unapply(expr,t,v,w,..), де **f** – позначення майбутньої функції, **expr** математичний вираз, що описує майбутню функцію, **t, v, w,..** – її змінні. Приклад:

```
> m(x,y):=sqrt(x^2+y^2);
```

$$m(x, y) := \sqrt{x^2 + y^2}$$

```
> m(3,4);
```

$$m(3, 4)$$

```
m:=unapply(sqrt(x^2+y^2),x,y);
```

$$m := (x, y) \rightarrow \sqrt{x^2 + y^2}$$

```
> m(3,4);
```

$$\sqrt{25}$$

```
> evalf(%);
```

$$5.000000000$$

Диференціальних рівнянь

розв'язок виконується за допомогою функції **dsolve({eqns} {vars}, option)**, де **deqns** – одне

диференціальне рівняння або система із диференціальних рівнянь першого порядку з зазначенням початкових умов; **vars** – змінна або змінні, щодо яких шукається розв'язок; **option** – необов'язковий параметр, що вказує на метод розв'язку.

Параметр **option** задає один з методів розв'язку:

exact – аналітичний розв'язок (прийняте за умовчуванням);

explicit – розв'язок в явному вигляді;

laplace – розв'язок через перетворення Лапласа;

series – розв'язок у вигляді ряду з порядком, що вказується значення змінної **Order**;

numeric – розв'язок в чисельному вигляді.

Для розв'язку задачі Коші в **{deqns}** треба включати початкові умови, а при розв'язку крайових задач – крайові умови. Якщо **Maple** має можливість знайти розв'язок при числі початкових чи крайових умов менше порядку системи, то у розв'язку будуть з'являтися невизначені константи виду **_C1**, **_C2** і т.д. Вони ж можуть бути і при аналітичному рішенні системи, коли початкові умови не задані. Якщо розв'язок знайдене в неявному вигляді, то в ньому з'явиться параметр **_T**. Вираз **{deqns}** повинне мати структуру множини і містити крім самої системи рівнянь їхні початкові умови.

Розв'язок диференціальних рівнянь виводиться, як правило, у вигляді, що не обчислюється. Якщо потрібна подальша робота з розв'язком, то його треба перетворити в повновагому функцію за допомогою функції **unapply**. У **Maple** знаходиться пакет розв'язку диференціальних рівнянь **DEtools**, що значно розширює можливості ядра **Maple** для розв'язку диференціальних рівнянь.

Завдання початкових значень похідної функції необхідно виконувати за допомогою оператора **D**. Так оператор **D(D(y))(1.) = -3** задає початкове значення другої похідної від функції **f** у точці 1, рівне -3... Приклад:

Розв'язати диференціальне рівняння

y'' - y = x sin(x) при **y(1) = 3**, **y'(1) = -1**, тобто розв'язати задачу Коші.

> **restart**:

> **dsolve({diff(y(x),x\$2)-y(x)=x*sin(x),y(1.)=3.,D(y)(1.)=-1.},y(x));**

$$y(x) = \frac{1 - e^x \cos(x) - e^x x \sin(x) + 1.0893(e^x)^2 + 12.0168}{e^x}$$

> **y(2.);**

y(2.)

Отримана в розв'язку функція **y(x)** не є повною, її не можна використовувати для одержання числових значень розв'язка, будувати по ній графіки. Одержимо розв'язок, використовуючи функцію **unapply**:

```
R:=dsolve({diff(y(x),x$2)-y(x)=x*sin(x),y(1.)=3.,D(y)(1.)=-1.},y(x));
```

$$R := y(x) = -\frac{1}{2} \frac{e^x \cos(x) + e^x x \sin(x) - 1.0893(e^x)^2 - 12.0168}{e^x}$$

Скасовуємо присвоєння для **y**:

```
>y='y':
```

Формуємо повною функцію:

```
> y:=unapply(rhs(R),x);
```

$$y := x \rightarrow -\frac{1}{2} \frac{e^x \cos(x) + e^x x \sin(x) - 1.0893(e^x)^2 - 12.0168}{e^x}$$

Обчислюємо значення **y(x)** у точці 2.

```
> y(2.);
```

4.136392464

Будуємо графік функції (рис.2.7):

```
>plot(y(x),x=-10..10);
```

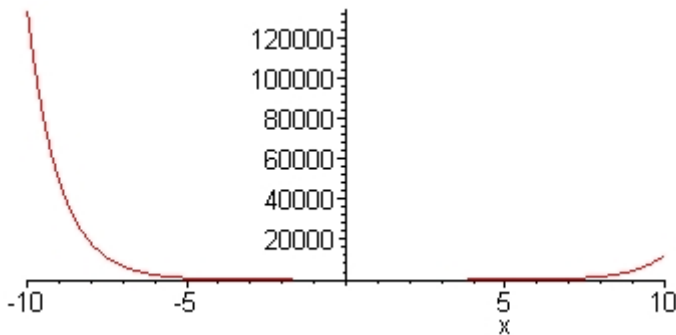


Рис. 2.7.

Екстремуміє функцій пошук реалізується за допомогою стандартної бібліотеки. Спочатку викликається бібліотека, а потім функція пошуку. У **Maple** можна окремо шукати екстремуми без вказівки виду, максимуми і мінімуми.

Бібліотечна функція **extrema(expr, constr, vars, 'nv')** дозволяє знайти екстремум виразу **expr** при обмеженнях **constr** для змінних **vars**, по яких шукається екстремум. Знайдені координати точок екстремуму привласнюються змінної **'nv'**.

Приклади:

```
> readlib(extrema):
> extrema(x*exp(-x), {}, x, 's');
                                     {exp(-1)}
> s;
                                     {{x = 1}}
> extrema(x^3.-2.*x^2.+5., {}, x, 's');
                                     {3.8, 5.}
> s;
                                     {{x = 0}, {x = 1.3}}
> extrema(x^2+y^2+3., {}, {x,y}, 's');
                                     {3}
> s;
                                     {{y = 0, x = 0}}
```

Якщо обмежень не має, то це вказується порожньою множиною **{ }**.

Для пошуку мінімумів і максимумом виразів (функцій) **expr** служать функції:

minimize(expr)	maximize(expr)
minimize(expr, vars)	maximize(expr, vars)
minimize(expr, vars, 'infinite')	maximize(expr, vars, 'infinite')
minimize(expr, vars, ranges)	maximize(expr, vars, ranges)

Ці функції можуть розшукувати максимуми і мінімуми як функцій однієї змінної, так і ряду змінних; вони вказуються списком змінних **vars**. Опція **'infinite'** означає, що пошук мінімуму або максимуму виконується по всій числовій вісі, а опція **ranges** дозволяє задавати межі зміни змінних при пошуку. Функції не

видають точок максимуму і мінімуму! Приклади:

```
> minimize(x^2 + y^2, {x, y}, {x=-10..10, y=10..20});
```

100

```
> maximize(x^3-2.*x^2+5.)
```

5.

Статистичні розрахунки в **Maple** можна виконати, використовуючи пакет **stats**.

Пакет **stats** для таких розрахунків представлений всього двома багатоцільовими статистичними функціями:

stats[subpackage, function](args) subpackage[function](args)

Однак, завдяки спеціальній формі завдання параметрів (зокрема, у виді субпакетів **subpackages**), можливе обчислення найрізноманітніших статистичних функцій. Можливе звертання до наступних субпакетам:

anova — дисперсійний аналіз;

describe — функції розподілу ймовірності;

fit лінійна регресія;

random — генерація випадкових чисел з різними законами розподілу;

statevalf — обчислення статистичних функцій і одержання оцінок для масивів даних;

statplots — побудова графіків статистичних функцій;

transform — функції перетворення даних .

Одним з найважливіших додатків пакета **Stats** є **регресійний аналіз**.

Під регресійним аналізом (чи просто регресією) звичайно мають на увазі знаходження деякої формальної аналітичної залежності, що приблизно (за критерієм мінімуму середньоквадратичної помилки) апроксимує деяку вихідну залежність. Найчастіше вона буває представлена набором (хмарою) точок, наприклад, даних експерименту.

Для проведення регресійного аналізу служить функція **fit**, що викликається в такий спосіб:

```
stats[fit,leastsquare[vars,eqn,parms]](data)
```

або

fit[leastsquare[vars,eqn,parms]](data)

де: **data** — список даних, **vars** — список змінних для представлення даних, **eqn** — рівняння, що задає апроксимуючу залежність (за замовчуванням лінійну), **parms** — безліч параметрів, що будуть замінені обчисленими значеннями.

На прикладах нижче показано проведення регресій за допомогою функції **fit** для залежностей виду **y(x)**. Нехай отримані експериментальні дані залежності **y** від **x**:

	1.00	1.20	1.30	1.40	1.50	1.60	1.70
	1.04	1.06	1.16	1.10	1.03	0.99	0.95

Тоді реалізацію регресії можна виконати в наступній послідовності

```
>with(stats): # виклик пакета
>Digits:=5:   # завдання виводу п'яти значущих цифр
>f:=fit[leastsquare][x,y]([
    [1.00,1.10,1.20,1.30,1.40,1.50,1.60,1.70],
    [1.04,1.06,1.16,1.15,1.10,1.03,0.99,0.95]]);
    f := y = -.16905 x + 1.2882
>plot(y,x=2..9);
```

Plotting error, empty plot

У прикладі функція регресії не задана, тому реалізується найпростіша лінійна регресія і функція **fit** повертає отримане рівняння регресії для початкових даних, представлених списками координат вузлових точок. Це рівняння апроксимує дані з найменшою середньоквадратичною похибкою. Побудована функція виводиться у форматі, що не обчислюється. Для подальшої роботи її треба представити у вигляді повновагомій функції, як це зроблено нижче.

```
R:=fit[leastsquare][x,y,y=a*x^2+b*x+c]([
    [1.00,1.10,1.20,1.30,1.40,1.50,1.60,1.70],
    [1.04,1.06,1.16,1.15,1.10,1.03,0.99,0.95]]);
    R := y = -1.0640 x2 + 2.7036 x - .59498
> plot(y,x=2..9);
```

Plotting error, empty plot

```
y:=`y`:
> y:=unapply(rhs(R),x);
```

```

y := x -> -1.0640 x2 + 2.7036 x - .59498
> plot([y(x),[[1.,1.04],[1.1,1.06],[1.2,1.16],[1.3,1.15],
[1.4,1.1],[1.5,1.03],[1.6,0.99],[1.7,0.99]]],
x=1.0..1.7,
style=[line,point],thickness=1);

```

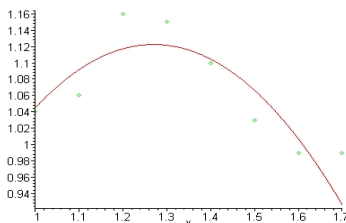


Рис. 2.8

У другому прикладі задане наближення вихідних даних степеневим багаточленом другого порядку. Узагалі говорячи, функція **fit** забезпечує наближення будь-якою функцією і, таким чином, реалізує практично усі види регресії для функцій однієї змінної.

Однак функція **fit** може забезпечувати регресію і для функцій декількох змінних. При цьому просто треба збільшити розмірність масивів початкових даних. Як приклад, нижче приведе приклад регресії для функції двох змінних

```

>f=fit[leastsquare[[x,y,z],z=a+b*x+c*y,{a,b,c}]]\
([1,2,3,5,5],[2,4,6,8,8],[3,5,7,10,15]);
f = (z = 1+7/2*x-3/4*y)
> fa:=unapply(rhs(f),x,y);
fa := (x, y) -> 1+7/2*x-3/4*y
>fa(1.,2.);
2.9999999999
>fa(2,3);

```

37/6

У даному випадку рівняння регресії задане у вигляді **z=a+b*x+c*y**. Звернемо увагу на важливий момент наприкінці цього приклада — застосування отриманої функції регресії для обчислення значень функції та побудови її графіка. Пряме застосування функції **f** у даному випадку неможливо, тому що вона представлена у форматі, що не обчислюється. Для одержання формату, що обчислюється, вона перетворена у функцію двох

змінних **fa(x,y)** шляхом виділення правої частини у виразу для функції **f**. Після цього можливо обчислення значень функції **fa(xy)** для будь-яких заданих значень **x** та **y**

З **матрицями і векторами** робота здійснюється командами ядра **Maple** і пакета розв'язка задач лінійної алгебри **linalg**. При роботі з матрицями варто мати на увазі, що вивід значень матриці можна здійснити функцією **evalm(expr)**.

Створення векторів і матриць можна виконати функцією ядра **array**. Звичайно вона використовується у формі:

array(a..b,s1) повертає вектор з індексами від **a** до **b** і значеннями в одномірному списку **s1**;

array(a..b,c..d,s21) повертає матрицю з номерами рядків від **a** до **b**, стовпців від **c** до **d** і значеннями в двовимірному списку **s2**:

> **A:=array(1..3,[x,y,`cos`]);**

$A := [x, y, \cos]$

> **B:=array(1..2,1..2,[[a,b],[c,d]]);**

$$B := \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

> **B;**

B

> **evalm(B);**

$$B := \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

Елементи векторів і матриць є індексованими змінними, тобто місце кожного елемента вектора визначається його індексом, а у матриці – двома індексами. Допускаються операції виклику потрібного елемента і присвоєння йому нового значення:

V[i] - виклик **i** – го елемента вектора **V**;

M[i, j] - виклик елемента матриці **M**,

розташованого на **i** – му рядку **j** – го стовпця;

M[i, j]:=x - присвоювання нового значення **x** елементу матриці **M**.

З матрицями можна виконувати типові операції додавання,

віднімання, множення, зведення в ступінь, а також вирішувати з їх допомогою системи лінійних рівнянь.

Приклад – розв’язати квадратну систему лінійних рівнянь:

$$x_1 + 2x_2 + 3x_3 = 2,$$

$$4x_1 + 5x_2 + 6x_3 = 4,$$

$$-4x_1 + 2x_2 + x_3 = 2.$$

Сформуємо матрицю системи **A** та вектор-стовпець вільних членів **B**.

> **A:=array(1..3,1..3,[[1,2,3],[4,5,6],[-4,2,1]]);**

$$A := \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ -4 & 2 & 1 \end{bmatrix} \quad \text{EMBED Equation.3}$$

> **B:=array(1..3,1..1,[[2],[4],[3]]);**

$$B := \begin{bmatrix} 2 \\ 4 \\ 3 \end{bmatrix}$$

Значення невідомих **x** можна визначити як добуток матриці, зворотної до матриці **A** на вектор-стовпець вільних членів **B**:

> **X:=A^(-1)*B;**

$$X := \frac{B}{A}$$

> **evalm(X);**

$$\begin{bmatrix} -\frac{1}{3} \\ \frac{2}{3} \\ \frac{1}{3} \end{bmatrix}$$

> **convert(evalm(X),float);**

$$\begin{bmatrix} -.3333333333 \\ .6666666667 \\ .3333333333 \end{bmatrix}$$

Відповідно $x_1 = -0,333333..$, $x_2 = 0,66666..$, $x_3 = 0,33333..$.

Більш ефективно операції лінійної алгебри реалізуються за допомогою пакета розв'язка задач лінійної алгебри **linalg**. Він містить понад сто функцій по формуванню матриць і векторів, виконанню всіх можливих операцій над ними, розв'язку лінійних рівнянь як у матричному вигляді, так і через спеціальні функції. Роботі з цим пакетом повинен передувати виклик пакета командою **with(linalg)**. Розглянемо основні функції цього пакета.

Формування матриці з числом рядків **n**,

A := matrix([[1, 2, 3], [4, 5, 6], [-4, 2, 1]]),

де **n=3**, **[1, 2, 3]** – перший ряд матриці, **[4, 5, 6]** – другий ряд, **[-4, 2, 1]** – третій ряд.

Ранг матриці обчислюється функцією

>rank(A);

3

Вектора створення реалізується функцією **vector**:

vector([5,4,6,3]) → [5, 4, 6, 3]

Кут між векторами **A** и **B** можна обчислити за допомогою функції **angle(A, B)**.

Визначник (детермінант) матриці A обчислюється функцією **det(A)**:

>det(A);

21

2.6. Побудова графіків у Maple

Maple дозволяє будувати двовимірні і тривимірні графіки, знаходити перетинання поверхонь у просторі, задавати анімаційні ефекти.

Найпростішими є двовимірні графіки. Вони будуються за допомогою функції **plot**, у якій задається функція графіка, його вид, колір і інші параметри.

Загальний вид завдання графіка - **plot(f,h,v,opt1,opt2,...)**, де:

f – функція, графік якої ми будуємо;

h – діапазон зміни аргументу функції по горизонталі, звичайно задається у вигляді **x1..x2**;

v – діапазон зміни функції по вертикалі (може не вказуватися, тоді **Maple** підбирає діапазон автоматично);
opt – різні опції, за допомогою яких можна задати різні параметри.

Типова команда для побудови графіка однієї функції має вид:

```
plot(tan(x),x=-6*Pi..6*Pi,y=-10..10,title="Графік тангенса",
color=green,style=point,symbol=cross,thickness=2);
```

Ця команда побудує графік функції тангенса з наступними параметрами: діапазон зміни **x** – від **-6π** до **+6π**, діапазон зміни самої функції – від **-10** до **+10**, заголовок – **Графік тангенса**, колір графіка – зелений, графік буде виводитися крапками, символ виведення однієї крапки графіка – хрестик, товщина графіка – 2 умовні одиниці (може бути 0, 1, 2, 3, за замовчуванням 0). У команді обов'язковим є завдання функції і діапазону зміни аргументу. Всі інші опції необов'язкові або встановлюються **Maple** за замовчуванням. У **Maple** можливий одночасний вивід графіків декількох функцій. У цьому випадку самі функції та їхні опції задаються множинами вигляду **[a1,a2,...,an]**; можлива побудова графіків по окремих точках, як у прикладі на рис. 2.10.

На рис. 2.9 представлена команда виведення двох графіків **sin x** і **(sin x)/x** і самі графіки з різними опціями:

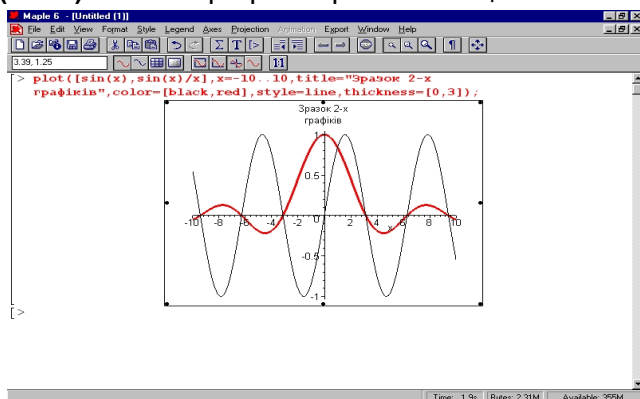


Рис. 2.9

діапазон зміни аргументу **x** по горизонталі: **-10 - +10**;
діапазон зміни функцій по вертикалі – підбране **Maple**;
заголовок графіка «Зразок 2-х графіків»;

графік функції **$\sin x$** будується чорним (black) кольором,
 графік функції **$(\sin x)/x$** будується червоним (**red**) кольором;
 графіки функцій будуються суцільною лінією (**line**)
 товщиною 0 і 3, відповідно;

умовна товщина лінії 1 (можливо 0, 1, 2, 3).

Якщо клацнути лівою кнопкою миші по графіку, то вгорі з'являється додаткова панель керування видом графіка.

Графік можна побудувати по точках, заданих списком координат аргументу і функції.

На рис. 2.10 зображено графік такої функції, окремі точки графіка з'єднані ламаною лінією зеленого кольору.

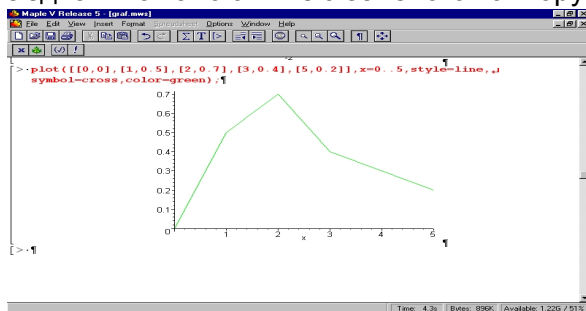


Рис.2.10.

Для побудови графіків тривимірних поверхонь **Maple** має убудовану в ядро функцію **plot3d**. Приклад її використання для побудови графіка пересічних поверхонь приведе на рис.2.11:

> plot3d({2*sin(x*y)+1, x^2+2*y^2-5},x=-Pi..Pi,y=-Pi..Pi);

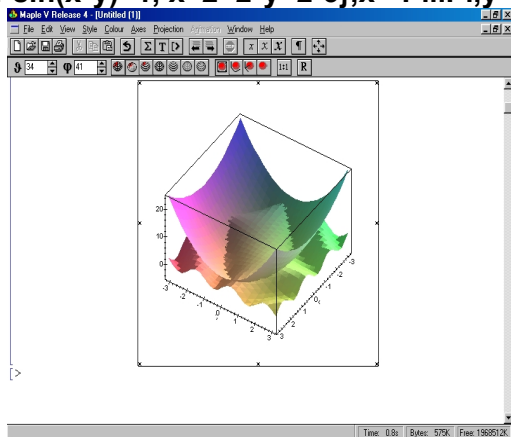


Рис. 2.11

Статистичний пакет **stats** має власні можливості для побудови графіків. Нижче реалізована побудова гістограми набору випадкових чисел:

```
> with(stats):
```

```
> with(stats[statplots]):
```

Завдання випадкових величин

```
> data2:= [-1.96,-.814,1.86,1.96,.519,.739,-.0540,.702,.663,  
.591,.580,.475,.589,-1.33,.0420,-.460,-.482,1.58,.778,.530,-  
.507,-.233,-.195, 193, -.136];
```

```
data2 := [-1.96, -.814, 1.86, 1.96, .519, .739, -.0540,  
.702, .663, .591, .580, .475, .589, -1.33, .0420, -.460, -  
.482, 1.58, .778, .530, -.507, -.233, -.195, .193, -.136]
```

Побудова гістограми

```
> histogram[count](data2);
```

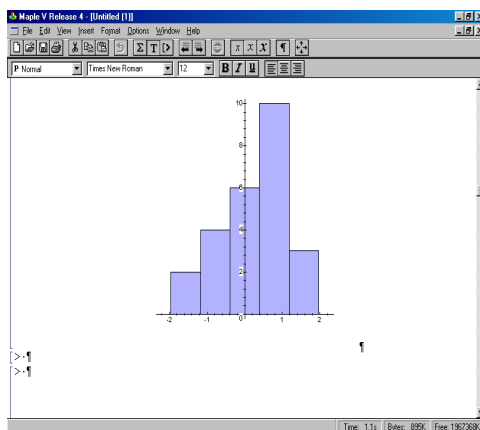
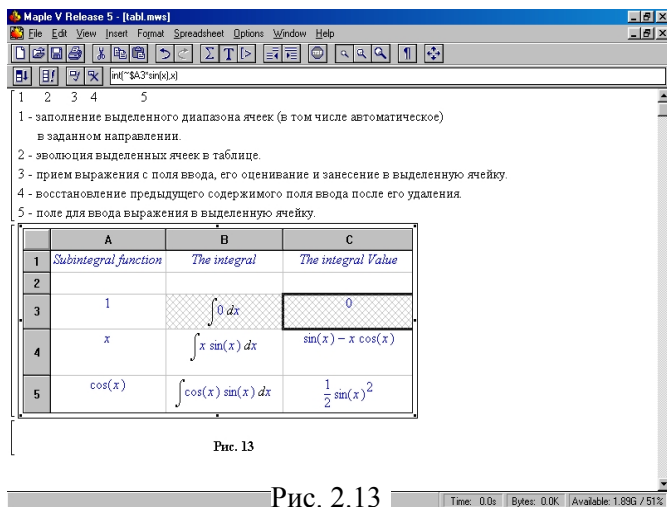


Рис. 2.12

2.7. Робота з електронними таблицями

У **Maple** впроваджені обмежені засоби роботи з електронними таблицями. Для завдання електронних таблиць у головному меню в позиції **Insert** з'явилася операція **Spreadsheet** (вставка шаблону електронної таблиці). При виконанні цієї операції з'являється типовий шаблон електронної таблиці (рис. 2.13), що має розмір 100 (рядів) x 52 (колонки).



Для роботи з таблицями в головному меню **Maple** стає доступною позиція **SpreadSheet**, що містить набір операцій для роботи з табличними даними. Крім того з'являється окрема панель для роботи з таблицею.

В осередки таблиць можуть уводитися будь-які вирази **Maple**, а також текстові дані англійською мовою. Осередки характеризуються своєю адресою — буквою стовпця (**A – AZ**) і номером рядка (**1 – 100**), наприклад **AB4**. У середині таблиці курсор миші перетворюється в жирний хрест і його можна використовувати для вказівки потрібного осередку і фіксації в ній уведення. При цьому введення даних у виділений осередок виконується шляхом їхнього набору в спеціальному полі уведення виразів панелі для роботи з таблицями. Натиснувши праву клавішу миші, можна вивести контекстно-залежне меню з переліком операцій, доступних для заданого осередку.

Уведення виразу у виділений осередок здійснюється через поле введення панелі для роботи з таблицями. Уведення виразів у ряд суміжних осередків по вертикалі або горизонталі істотно прискорюється шляхом використання операції автозаповнення (**Fill**). Для цього необхідно відповідний вираз занести в перший осередок ряду та виділити її. Натиснувши й утримуючи клавішу **shift** відзначається останній осередок ряду. Після виділення осередків ряду

необхідно виконати одну з команд **Down, Up, Left**, чи **Right** пункту меню **SpreadSheet**. Виділення осередків ряду можна виконати мишею, а замість вибору команди заповнення натиснути першу кнопку панелі для роботи з таблицями.

Ряд суміжних осередків можна заповнити числами, що зростають чи убують з визначеним кроком. Для цього вводиться визначене число в перший осередок ряду, ряд осередків виділяється і натискається перша кнопка панелі для роботи з таблицями (можна також виконати команду **SpreadSheet/Fil/Detailed...**З'являється діалогове вікно для вибору кроку ряду (**Step Size**), напрямку заповнення. Можна також ввести останнє значення ряду (**Stop Value**). Натискання кнопки приводить до заповнення ряду.

Вираз з одного осередку в інший можна переносити через буфер обміну, а також перетаскувати або копіювати мишею.

Для перетаскування виразу з одного осередку в інший необхідно виділити початковий осередок, підвести до нього покажчик миші так, щоб він прийняв форму стрілки, натиснути ліву кнопку миші (при цьому внизу стрілки з'являється квадратик), перетягнути зміст осередку в потрібний осередок і відпустити ліву кнопку миші.

Для копіювання виразу за допомогою миші виконуються ті ж дії, але перед натисканням лівої кнопки миші необхідно натиснути й утримувати клавішу **Ctrl**. При цьому біля стрілки з'явиться невеликий хрестик.

У математичних виразах, що заносяться в осередки таблиці, можна використовувати посилання у вигляді адресів осередків на зміст інших осередків. Ознакою посилання є знак ~ (тильда) перед адресою. Використовуються абсолютні посилання, що відзначаються знаком \$ і відносні посилання, що спеціально не відзначаються. Так вираз **~\$B5** позначає абсолютне посилання на стовпчик **B** і відносне посилання на п'ятий осередок у ряді цього стовпчика. Застосування різних видів посилань на осередки таблиці аналогічно табличним процесорам, таких як, наприклад, Excel.

Для зміни розміру видимої на екрані частини таблиці, необхідно активізувати саму таблицю, підвести покажчик миші у вигляді стрілки зовні до будь-якого краю таблиці і клацнути

лівою кнопкою миші. Навколо таблиці з'являться маркери зміни границь. Перетаскуючи їх мишею, можна змінювати розміри таблиці.

2.8. Засоби програмування у Maple

Хоча ядро *Maple*, бібліотека і пакети розширення системи містять понад три тисячі функцій, завжди може виявитися, що саме потрібної користувачеві функції все-таки немає. Тоді виникає необхідність у створенні власної функції, іменованою функцією користувача.

Основним засобом розширення Maple-мови є модулі - процедура. Однак на перший погляд вони досить складні. Є і більш прості способи завдання функцій користувача. Один з таких способів полягає в присвоюванні уведеної функції (у вигляді виразу) деякої перемінної

Name:=Вираз

Цей прийом фактично означає просто операцію присвоювання. Наступні приклади ілюструють техніку роботи з такими функціями:

m:=sqrt(x^2+y^2);

$$m := \sqrt{x^2 + y^2}$$

x:=3:y:=4:m;

$$\sqrt{25}$$

evalf(m);

5.000000000

Заданий у такий спосіб програмний об'єкт усе-таки не є повноцінною функцією користувача, і, насамперед, тому, що в ньому використовуються тільки глобальні змінні (x і y) і немає оголошеного списку параметрів, від яких залежить значення функції. При цьому значення змінних функції приходить свідомо задавати окремо, використовуючи операції присвоювання. Подібні конструкції не можна ввести в бібліотеки Maple .

Більш гнучкий спосіб завдання повноцінних функцій користувача базується на застосуванні функціонального оператора. При цьому використовується наступна конструкція:

name : = (x, y,...) ->expr

Після цього виклик функції здійснюється у виді **name (x, y,...)**, де (**x, y,...**) список формальних параметрів функції користувача з ім'ям **name**. Змінні, зазначені в списку формальних параметрів, є локальними. При підстановці на їхнє місце фактичних параметрів вони зберігають їхнього значення тільки в тілі функції (**expr**). За межами цієї функції перемінні цими іменами виявляються або невизначеними, або зберігають раніше привласнені їм значення. Наступні приклади ілюструє сказане:

```
> restart;
> x:=0;y:=0;
      x:= 0
      y:= 0
> m:=(x,y)->sqrt(x^2+y^2);
      m := (x, y) → √x2 + y2
> m(3,4) ;
      5
> m(3.,4) ;
      5.0000000000
> x,y;
      [0, 0]
```

Неважко помітити, що при обчисленні функції **m(x,y)** перемінні **x** і **y** мали значення 3 і 4, однак за межами функції вони зберігають нульові значення, задані їм перед введенням визначення функції користувача. Ще один спосіб завдання функції користувача базується на застосуванні функції **unapply**:

name:=unapply (expr, var1, var2,...)

Нижче дані приклади такого завдання функції користувача.

```
> restart;
> fm:=unapply(sqrt(x^2+y^2),x,y);
      fm := (x, y) → √x2 + y2
fm(4.,3);
      5.0000000000
```

Для підготовки програм, що розгалужуються, у Maple-мову програмування включений оператор **if**, що дозволяє створювати наступну конструкцію:

```

if <Умова порівняння> then <Елементи>
| elif <Умова порівняння> then <Елементи>|
| else <Елементи>|
fi;

```

У вертикальних рисах | | зазначені необов'язкові елементи цієї конструкції. Наступні дві конструкції умовних виражень найчастіше використовуються на практиці:

- 1) **if <Умова> then <Елементи 1> fi** — якщо **Умова** виконується, те виконуються **Елементи 1**, інакше нічого не виконується;
- 2) **if <Умова> then <Елементи 1> else <Елементи 2> fi** — якщо **Умова** виконується, те виконуються **Елементи 1**, інакше виконуються **Елементи 2**.

В умовах використовуються будь-як логічні конструкції зі знаками порівняння (<, <=, >, >=, =, o) і логічних операторів **and**, **or** і **not**; у цих конструкціях повертаються логічні значення **true** і **false**. Розглянемо наступний простий приклад:

```

> x:=-5:
> if x<0 then print('Negative') fi;

```

Negative

```

> x:=1:
> if x<0 then print('Negative4') fi;

```

У цьому прикладі аналізується значення **x**. Якщо воно від'ємне, то за допомогою функції висновку **print** на екран дисплея виводиться повідомлення «Negative». А от якщо **x** невід'ємно, то не виводиться ніякого повідомлення. В іншому прикладі, якщо **x** невід'ємно, то виводиться повідомлення «Positive»:

```

> x:=-5:
> if x<0 then print('Negative') else print('Positive') fi;
Positive
> if (x<0, print('Negative'),print('Positive')) ;..
Positive

```

Досить часто необхідно циклічне повторення деяких виражень задане число раз або доти, поки виконується визначена умова. **Maple** має узагальнену конструкцію циклу, що задається в такий спосіб:

```
l for <name>| l from <expr1>| l to <expr3>| l by <expr2>|  
l while <expr4>|  
do <statement sequence> od;
```

Тут **name** — ім'я керуючої перемінної циклу, **expr1**, **expr2** і **expr3** — вираження, що задають початкове значення, кінцеве значення і крок зміни перемінної **name**, **expr4** — вираження, що задає умова, поки цикл (набір об'єктів між словами **do** і **od**) буде виконуватися.

У ході виконання циклу керуюча перемінна міняється від значення **expr1** до значення **expr2** із кроком, заданим **expr3**. Якщо блок **by <expr2>** відсутній, то керуюча перемінна буде мінятися з кроком +1 при **expr1 < expr2**. Це наочно пояснює наступний приклад:

```
> for i from 1 to 5 do print(i) od;  
1  
2  
3  
4  
5
```

Він виводить значення керуючої перемінної **i** у ході виконання циклу. Неважко помітити, що вона і справжні міняється від значення 1 до значення 5 із кроком +1. Наступний приклад показує, що границі зміни керуючої перемінної можна задати арифметичними вираженнями:

```
> for i from 7/(2+5) to 2+3 do print(i) od;  
1  
2  
3  
4  
5
```

А в іншому прикладі показано завдання циклу, у якого керуюча перемінна міняється від значення 1 до 10 із кроком 2:

```
.> for i from 1 to 10 by 2 do print(i) od;  
1  
3  
5  
7  
9
```

У цьому випадку виводяться непарні числа від 1 до 9. Слід

зазначити, що виконання циклу в зворотному порядку, коли $\text{expr1} > \text{expr2}$, не передбачено. Але зате цикл можна перервати за допомогою додаткового блоку **while <expr4>**. Цикл із таким блоком виконується доти, поки умова expr4 істинно. Наприклад

```
> for i from 1 to 10 by 2 while i<6 do print(i) od;  
1  
3  
4
```

Таким чином, конструкція циклу в Maple-мові програмування увібрала в себе основні конструкції циклів **for** і **while**. Є ще одна, більш специфічна конструкція циклу:

```
I for <name>I I in <expr1>| |while <expr2>| do,  
<statement sequence> od;
```

Тут expr1 задає список значень, що буде приймати керуюча перемінна **name**. Цикл буде виконуватися, поки не буде вичерпані список і поки виконується умова, задане вираженням expr2 . Наступний приклад ілюструє сказане:

```
> for i in [1,2,5,-1,7,12] do print(i) od;  
1  
2  
5  
-1  
7  
12  
  
> for i in [1,2,5,-1,7,12] while i>0 do print(i) od;  
1  
2  
5
```

У циклі цього виду керуюча перемінна може мінятися довільно, приймаючи убутні і від'ємні значення.

Цикли можуть бути вкладеними. Це ілюструє наступний приклад, що створює одиничну матрицю на базі заданого масиву **M**:

```
> M:=array(1..3,1..3) ;  
M:=array(1 ..3,1 .. 3, [ ])  
> for i to 3 do for j to 3 do M[i,j]:=0;if i=j then
```

```
M[i,j]:=1 fi; od od;
> evalm(M);
```

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Можлива спрощена приватна конструкція циклу типу **while**:

```
while expr do statseq od;
```

Тут вираз **statseq** виконуються, поки виконується логічна умова **expr**. Приклад такого циклу:

```
> n:=1;
> while n<16 do n:=2*n od;
```

n:=2
n:=4
n:=8
n:=16

У цьому прикладі виконується подвоєння числа n з початковим значенням $n = 1$ доти поки значення n менше 16.

Іноді буває потрібним пропустити визначений цикл. Для цього використовується оператор **next** (наступний). Приведений нижче приклад ілюструє застосування оператора **next** у складі вираження **if-fi** для виключення виводу значення $i = -2$:

```
> for i in [1, 2, 3, -2, 4.] do if i=-2 then
  next else print(i) fi od;
```

1
2
3
4

Інший оператор - **break** - перериває виконання фрагмента програми (або циклу). Його дія пояснює злегка модифікований попередній приклад:

```
> for i in [1, 2, 3, -2, 4.] do if i=-2 then break
  else print(i) fi od;
```

У даному випадку при значенні $i = -2$ відбулося повне припинення виконання циклу. Тому наступне значення 4 змінної i привласнене не було, і це значення на вихід циклу не потрапило.

Який завгодно з операторів `quit`, `done` або `stop` забезпечує також переривання виконання поточної програми (зокрема, циклу), але при цьому вікно поточного документа закривається і всі наявні в ньому визначення зникають.

Для складних обчислень використовуються процедури. Процедурою називають модуль програми, що має самостійне значення і виконує одну або кілька операцій, звичайно досить складних і відмінних від операцій, виконуваних убудованими операторами і функціями.

Процедури є найважливішим елементом структурного програмування і служать засобом розширення можливостей системи Maple користувачем. Кожна процедура має своє унікальне ім'я і список параметрів (навіть якщо він порожній). Процедури викликаються так само, як убудовані функції — указівкою їхнього імені зі списком фактичних параметрів. При цьому просто процедури звичайно не повертають яких-небудь значень після свого виконання, хоча можуть привласнювати значення вхідної в них змінним. Процедури-функції у відповідь на звертання до них повертають деяке значення. Вони як би є новими функціями, що задаються користувачем. Визначені раніше функції користувача фактично є процедурами-функціями з трохи спрощеною структурою.

Найпростіша форма завдання процедури наступна:

name := proc(Параметри)

Тіло процедури

end;

Параметри процедури задаються перерахуванням імен змінних, наприклад `proc (x)` або `proc (x, y, z)`. За допомогою знака `:` після імені перемінної можна визначити її тип, наприклад, в оголошенні `proc (n : integer)` з'являється, що змінна **n** є цілочисельна.

При виклику процедури виразом виду

name(Фактичні_параметри)

фактичні параметри підставляються на місце формальних. Невідповідність фактичних параметрів типові заданих змінних веде до повідомлення про помилку і відмовлення від виконання процедури.

Як приклад нижче приведена процедура обчислення модуля комплексного числа z — у даному випадку це єдиний параметр процедури:

```
> mode:=proc(z)
>   evalf(sqrt(Re(z)^2+Im(z)^2))
> end;
mode := proc(z) evalf(sqrt(R(z)^2 + I(z)^2)) end proc
```

Після введення заголовка процедури під рядком введення з'являється повідомлення: «Warning, premature end of input». Воно вказує на те, що введення листинга процедури не закінчений і повинний бути продовжений доти, поки не буде введене завершальне слово **end** листинга процедури. Якщо після цього слова поставити символ крапки з коми, то листинг процедури буде виведений на екран дисплея.

Тепер для обчислення модуля досить задати звертання до процедурі **mode** (z), указавши замість z конкретне комплексне число:

```
> mode(3.+1*4.);
5.0000000000
```

Неважко помітити, що при знаку **;** після завершального слова **end** текст процедури повторюється в рядку висновку (у загальному випадку в трохи іншому виді). Якщо це повторення не потрібно, після слова **end** треба поставити знак двокрапки. Звернете також увагу на те, що для позначення дійсної і мнимі частин процедури в її тексті використовуються готичні символи.

Як відзначалося, процедури, що повертають значення результату у відповідь на звертання до них, багато в чому тотожні функціям. Будемо називати їх процедурами-функціями. Звичайно процедура повертає значення останнього вираження в її тілі або вираження, наміченого до повернення спеціальним оператором **RETURN**:

```
> mode:=proc(z)
>   evalf(sqrt(Re(z)^2+Im(z)^2)) :
>   RETURN(%)
```

```

> end;
mode := proc(z) evalf(sqrt( R(z)^2 + I(z)^2)); RETURN (%) end proc
> mode(3.+1*4.);
5.000000000

```

Параметром оператора RETURN може бути будь-який вираз. У Maple не прийнято виділяти процедури-функції в якийсь окремий клас. Діє правило — якщо не використаний оператор RETURN, процедура повертає значення останнього вираження в її тілі. Для усунення видачі значень виражень після них просто треба установити знак двокрапки.

Змінні, котрі вказуються в списку параметрів (наприклад, **z** в останньому прикладі), є локальними. Це означає, що зміна їхніх значень відбувається лише в тілі процедури, тобто локально. За межами тіла процедури ці змінні мають те значення, яке у них було до використання процедури. Це добре пояснює наступний приклад:

```

> restart: z:=1;
z=1
> modc:=proc(z)
> evalf(sqrt(Re(z)^2+Im(z)^2));
> end;
modc := proc(z) evalf(sqrt(R(z)^2 + I(z)^2)) end proc
> mode(3.+1*4.);
5.000000000
> z;
1

```

Неважко помітити, що усередині процедури $z = 3 + i \cdot 4$, тоді як поза нею значення $z=1$. Таким чином, імена змінних у списку параметрів процедури можуть збігатися з іменами глобальних змінних, використовуваних за межами процедури.

Змінні, котрим уперше привласнюється значення в процедурі, також відносяться до локального. Крім того, змінні, застосовувані для організації циклів, є локальними. Всі інші змінні — глобальні.

Якщо у тілі процедури є операції привласнення для раніше визначених (глобальних) змінних, то зміна їхніх значень у ході виконання процедури створює так називаний побічний ефект. Він здатний істотно змінити алгоритм розв'язання складних задач і, це як правило, неприпустимо.

Тому Maple-мова програмування має убудовані засоби для

виключення побічних ефектів. Зустрічаючи такі операції присвоювання, Maple-мова коректує текст процедури і вставляє в неї оголошення змінних локальними за допомогою ключового слова `local` і видає попереджувачий напис про подібне застосування:

```
> restart: m:=0;
           m:=0
> modc:=proc(z)
> m:=evalf(sqrt(Re(z)^2+Im(z)^2)):RETURN(m)
> end;
```

Warning, `m` is implicitly declared local to procedure `modc`
`modc := proc(z) local m; m := evalf(sqrt(R(z)^2 + I(z)^2)); RETURN(m) end proc`
> modc(3.+1*4.);

5.0000000000

> m

0

Зауважимо, що у телі процедури було автоматично уставлене визначення `local m`, що задає локальний статус змінної `m`. Оператором `Print` можна вивести текст процедури:

> print(modc) ;

`proc(z) local m; m := evalf(sqrt(R(z)^2 + I(z)^2)); RETURN(m) end proc`

У ряді випадків доводиться повідомляти деякі змінні усередині процедури глобальними. Щоб зробити змінні усередині процедури глобальними, досить оголосити них за допомогою ключового слова `global`, після якого перелічуються ідентифікатори змінних. Наступний приклад пояснює застосування оператора `global` у процедурі:

>a:=1; b:=1;

a=1

b=1

>fg:=proc(x,y)

>global a,b;

a:=x^2;b:=y^2:

>RETURN(sqrt(a+b));

>end;

`fg := proc(x,y) global a, b; a := x^2; b := y^2; RETURN(sqrt(a + b)) end proc`

>fg(3,4);

5

>[a,b] ;

[9,16]

У прикладі змінним a і b спочатку привласнені значення 1, Оскільки вони у процедурі оголошені глобальними, те усередині процедури вони приймають нові значення x^2 і y^2 . У результаті при виході з процедури вони мають уже нові значення. Це і є побічний ефект при виконанні даної процедури. Якщо користувач не знає (або не пам'ятає), що та або інша процедура має побічний ефект, то він ризикує одержати самі несподівані (і невірні) результати своїх обчислень.

При професійній підготовці процедур користувач повинний передбачити їхнє поведіння при можливих помилках. Наприклад, якщо він готує процедуру або функцію, що обчислює квадратний корінь з дійсних чисел, то треба врахувати, що такий корінь не можна витягати з негативних чисел (будемо, винятково в навчальних цілях, вважати, що комплексні числа в даному прикладі неприпустимі!).

Для контролю за типом даних звичайно використовуються різні функції оцінки і тестування. При виявленні ними помилки, як правило, передбачається виведення відповідного повідомлення. Для цього використовується функція **ERROR**:

ERROR (expr_1, expr_2, ...)

де $\text{expr}_1, \dots$ — ряд виражень (можливо, порожній). Найбільше часто **ERROR** виводить просто строкове повідомлення про помилку, наприклад **ERROR ('strings'~)**. Повне повідомлення про помилку має вигляд:

Error, (in name) string,...

Приведемо приклад процедури, у якій передбачений виведення повідомлення про помилку при завданні змінної $x < 0$:

```
> f := proc (x) if x<0 then error "invalid variable x: %1",  
  x else x^(1/2) end if end proc;  
f:=proc(x) if x<0 then error "invalid variable x: %1"x else sqrt(x) end if end proc  
> f(3.); :  
1.732050808  
> f(-3.);  
Error, (in f) invalid variable x: -3.  
> lasterror;  
"invalid variable x: %1" , -3.  
> lastexception;
```

f, "invalid variable x: % 1", -3.

Ця процедура обчислює квадратний корінь з числа x . При $x < 0$ виводиться задане повідомлення про помилку.

В оголошенні процедури можна включити опції, що вводяться словом `options opseq`. Іноді ці опції називають розширювальними ключами. Передбачено наступні опції:

- `remember` — визначає таблицю пам'яті для процедури;
- `builtin` — визначає функцію як убудовану;
- `system` — визначає процедуру як системну;
- `operator` — повідомляє процедуру — функціональний оператор;
- `arrow` — визначає процедуру в нотації $\rightarrow$;
- `trace` — задає трасування процедури;
- `, package` — визначає процедуру для пакета розширення;
- `copyright` — захищає процедуру від копіювання.

Більш детальну інформацію по опціям можливо дізнатися з довідкової системи. Використання цих опцій значно розширює можливості системи Maple в галузі програмування.

Контрольні запитання і вправи

1. Які основні можливості системи **Maple** ?
2. Перечисліть основні засоби керування системою **Maple**.
3. Як користуватися довідковою системою **Maple** ?

4. Знайти похідну від функції $\sin^2 x$.

5. Знайти інтеграл $\int x^3 e^x dx$.

6. Побудувати двовірний графік функції
 $y = \sin x / x, x = -10..10$.

7. Побудувати тривірний графік функції
 $y = \sin(xy), x = -2..2, y = -2..2$.

8. Розв'язати систему рівнянь
$$\begin{cases} 5nx + 10y^2 = 20, \\ 10x^2 - xy = 5, \end{cases}$$
 де n - номер комп'ютера.

9. Розв'язати диференціальне рівняння
 $y'' - 20y' - 6ny = \sin nx$, при $y(2) = -2, y'(2) = 3$;

побудувати графік функції $y(x)$, n - номер комп'ютера.

10. Розв'язати систему рівнянь

$$\begin{cases} -3nx + 10y + 25z = -1, \\ 2x - 4ny + 2nz = 10, \\ 15x + 5y + 2nz = 7 \end{cases}$$

матричним способом.

3. Моделювання техніко-енергетичних систем

3.1. Моделювання процесу розповсюдження тепла у стержні

Функція $u(x, t)$ – температура у точці x стержня в момент t , розповсюдження тепла у стержні з тепло ізолюваною бічною поверхнею при відсутності джерел тепла визначається з диференціального рівняння в частинних похідних теплопровідності

$$\frac{\partial u}{\partial t} = a^2 \frac{\partial^2 u}{\partial x^2}, \quad (3.1)$$

де a – коефіцієнт теплопровідності. До рівняння (3.1) приєднуються граничні та початкова умови.

Початкова умова має вигляд

$$u(x, 0) = f(x). \quad (3.2)$$

Для стержня скінченної довжини, $0 \leq x \leq l$, кінці якого перебувають при нульовій температурі, граничні умови мають вигляд

$$\begin{aligned} u(0, t) &= 0, \\ u(l, t) &= 0. \end{aligned} \quad (3.3)$$

Рівняння теплопровідності (1) з початковою умовою (3.2) та граничними умовами (3.3) складає математичну модель процесу розповсюдження тепла у стержні. Розв'язок цього рівняння є дуже складною задачею, але її можливо розв'язати у системі Maple за допомогою пакета розв'язки диференціальних рівнянь у частинних похідних PDEtools. Цей пакет дозволяє розв'язувати диференціальні рівняння у частинних похідних як аналітично, але без обмежень, так і чисельно, будувати різноманітні графіки. Послідовність рішення наступна:

> **restart:**

Виклик пакета

> **with(PDEtools):**

> Опис диференціального рівняння розповсюдження тепла з

довільною константою a

> **PDE1:=diff(u(x,t),t)-a*diff(u(x,t),x,x)=0;**

$$PDE1 := \left(\frac{\partial}{\partial t} u(x, t) \right) - a \left(\frac{\partial^2}{\partial x^2} u(x, t) \right) = 0$$

Аналітичний розв'язок цього рівняння

> **pdsolve(PDE1,build);**

$$u(x, t) = e^{\left(\sqrt{-C1}x\right)} _C3 e^{(a - C1t)} _C1 + \frac{e^{(a - C1t)} _C2}{e^{\left(\sqrt{-C1}x\right)}},$$

де $_C1$, $_C2$, $_C3$ – невизначені константи.

Чисельний розв'язок рівняння (1) з граничними умовами (3) при $a = 419$ В/мхград, $l = 2$ і $f(x) = -40(x-1)^2 + 40$ можна отримати наступним чином

> **PDE2:=diff(u(x,t),t)-417*diff(u(x,t),x,x)=0;**

$$PDE2 := \left(\frac{\partial}{\partial t} u(x, t) \right) - 417 \left(\frac{\partial^2}{\partial x^2} u(x, t) \right) = 0$$

Опис граничних та початкової умови

> **IBC:={u(x,0)=-40*(x-1)^2+40,u(0,t)=0,u(2,t)=0};**

$$IBC := \{u(0, t) = 0, u(x, 0) = -40(x - 1)^2 + 40, u(2, t) = 0\}$$

Розв'язок рівняння і привласнення його змінній **ans**. Слід зауважити, що пакет **PDEtools** формує розв'язок у вигляді програмної процедури, яку можна використовувати для різних цілей.

> **ans:=pdsolve(PDE2,IBC,numeric,time=t,range=0..2);**

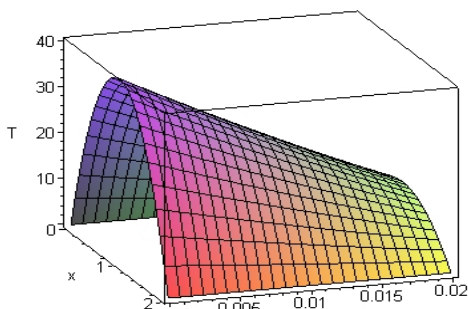
ans := module()export plot,plot3d,animate,value,settings...end module

Побудова тривимірного графіка (рис. 3.1) зміни температури у різних точках стержня в проміжку часу $[0, 0.02]$

> **ans:-value(t=0.02,output=listprocedure);**

> **ans:-plot3d(t=0..0.02,x=0..2,labels=['x','t','T']);**

[x=(proc(x) ... end proc), t=0.02, u(x,t) = (proc(x) ... end proc)



Можна побудувати двовимірні графіки розподілу температури вздовж довжини стержня у різні моменти часу. Це можна зробити наступним чином

```
> p1:=ans:-plot(t=0,thickness=0,title='Розподіл температури по  
стержню у різні моменту часу`):  
> p2:=ans:-plot(t=0.01,thickness=2):  
> p3:=ans:-plot(t=0.02,thickness=3):  
> plots[display]({p1,p2,p3},labels=['x`,`T']);
```

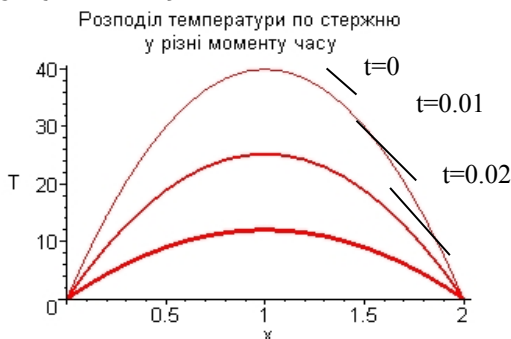


Рис. 3.2

Слід зауважити, що використання пакету PDEtools потребує його ретельного вивчення по довідці системи Maple.

3.2. Розрахунок середньої температури вздовж дротового зонда

Система Maple безперервно поповнюється новими спеціалізованими пакетами, які значно розширюють можливості системи та спрощують роботу з нею. Одним з таких пакетів є пакет Vec_calc для роботи з векторами. З його допомогою дослідимо розподіл температури вздовж дротового зонда, який помістили у ванну з нерівномірно нагрітою рідиною.

Нехай температурний зонд помістили у ванну з рідиною; тоді він буде відображати середню температуру вздовж зонда. Ванна має розмір 27x36x21 (умовних одиниць). Температура рідини у ванні задається функцією: $T(x,y,z) = xy - xz$. Зонд має форму дуги, яка описується параметричним рівнянням :

$$r(t) = [3t^2, t^3 + 3t, t^3 - 3t + 3] \text{ при } 0 < t < 3.$$

Потрібно знайти середню температуру вздовж зонда та порівняти її з температурою рідини у ванні.

По перше визначимо функції, які описують форму зонда та температуру у ванні.

```
> r:=MF(t, [3*t^2, t^3+3*t, t^3-3*t+3]);
```

$$r := [t \rightarrow 3t^2, t \rightarrow t^3 + 3t, t \rightarrow t^3 - 3t + 3]$$

```
> T:=MF([x,y,z], x*y-x*z);
```

$$T := (x, y, z) \rightarrow xy - xz \quad .3 \quad T := (x, y, z) \rightarrow xy - xz$$

Побудуємо графік дуги зонда за допомогою команди

```
> spacecurve(r(t), t=0..3, axes=boxed, color=blue,
thickness=3, orientation=[-30,80],
scaling=constrained);
```

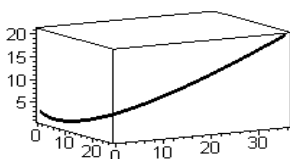


Рис.3.3

Знайдемо вектор дотичної до дуги або швидкість у кожній точці дуги зонда

```
> v:=D(r);
```

$$v := [t \rightarrow 6t, t \rightarrow 3t^2 + 3, t \rightarrow 3t^2 - 3] \quad .3$$

```
v := [t → 6t, t → 3t^2 + 3, t → 3t^2 - 3. > Tr:=T(op(r(t)));
```

```
Tr:=simplify(%);
```

Знаходимо його довжину

```
> interface(showassumed=0);
```

```
> assume(t, real); lv:=len(v(t));
```

$$lv := 3\sqrt{2}(1+t^2) \quad .3 \quad lv := 3\sqrt{2}(1+t^2) \quad .$$

Знаходимо температуру вздовж дуги зонда

```
> Tr:=T(op(r(t))); Tr:=simplify(%);
```

$$Tr := 3t^2 (t^3 + 3t) - 3t^2 (t^3 - 3t + 3)$$

$$Tr := 18t^3 - 9t^2$$

Середнє значення температури визначається виразом

$$T_{ave} = \frac{\int T ds}{\int 1 ds} = \frac{\int_0^3 T(r)|v| dt}{\int_0^3 |v| dt}.$$

Відмітимо, що знаменник є довжиною дуги.

Визначимо середню температуру зонда

> **Tave:=Int (Tr*lv, t=0..3) /Int (lv, t=0..3) ;**

$$T_{ave} := \frac{\int_0^3 3(18t^3 - 9t^2)\sqrt{2}(1+t^2)dt}{\int_0^3 3\sqrt{2}(1+t^2)dt}.$$

> **Tave:=value(%) ; evalf(%) ;**

$$Tave := \frac{6777}{40}$$

$$169.4250000$$

Зараз ми можемо порівняти з температурою у ванні:

> **1/(27*36*21)*Muint(T(x,y,z), x=0..27, y=0..36, z=0..21) ; value(%) ; evalf(%) ;**

$$\frac{1}{20412} \int_0^{21} \int_0^{36} \int_0^{27} (xy - xz) dx dy dz$$

$$\frac{405}{4}$$

$$101.2500000$$

Таким чином, температура, яку видає зонд є вищою, ніж на самому ділі.

Іншим способом обчислити ці інтеграли по лінії є використання команди `Line_int_scalar` (або її скорочення `Lis`) з пакета `vec_calc`, яка прямо працює з параметрично заданою

кривою та скалярними функціями:

```
> Lis (T,r,t=0..3)/Lis (1,r,t=0..3) ;  
Tave:=value (%) ;
```

$$\frac{\int_0^3 3(18t^3 - 9t^2)\sqrt{2}(1+t^2)dt}{\int_0^3 3\sqrt{2}(1+t^2)dt} \sim$$

Це значно скорочує запис виразів.

3.3. Розповсюдження тепла через багатошарову стінку

Зовнішня стінка холодильного складу складається з 0,5 м цегли, за яким слідує 0,1 м мінеральної вати, за нею з внутрішньої сторони стінки розташовується 0,05 м гіпсовій штукатурці, як показано на рис. 3.4.

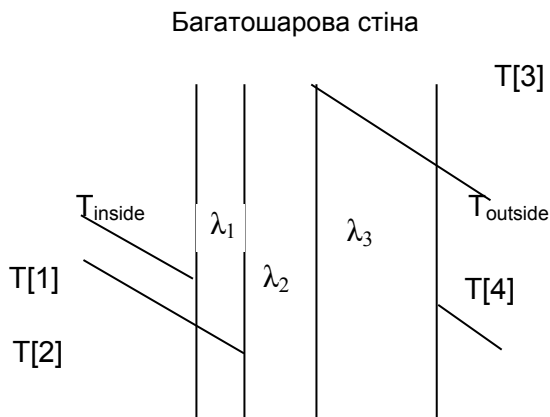


Рис. 3.4

$T[1]$, $T[2]$, $T[3]$, $T[4]$ це значення температурі на границях шарів стінки.

Зовнішня температура $T_{outside}$ дорівнює 25°C , внутрішня температура T_{inside} дорівнює -5°C . Коефіцієнт зовнішньої конвекційної передачі тепла – $\alpha_{outside}$, коефіцієнт внутрішньої конвекційної передачі тепла – α_{inside} . Потрібно знайти розподіл температури у стінці. Виконаємо це за допомогою системи Maple у відповідному робочому листу.

>restart:

Визначимо потік тепла усередині приміщення

> $Q_inside := \alpha[inside] * A * (T[inside] - T[1])$:

Потік тепла у стінці описується виразом

> $Q_i := \lambda[i] * A * (T[i] - T[i+1]) / d[i]$;

$$Q_i := \frac{\lambda_i A (T_i - T_{i+1})}{d_i}$$

Потік тепла ззовні стінки описується виразом

> $Q_outside := \alpha[outside] * A * (T[4] - T[outside])$:

Потік тепла через багатошарову стінку описується рівнянням:

> $eq := (T[inside] - T[outside]) / (1 / (\alpha[inside] * A) + \sum(d[i] / (\lambda[i] * A), i=1..n) + 1 / (\alpha[outside] * A))$:

Температурна провідність трьох шарів $\lambda - \left[\frac{W}{mK} \right]$, товщина трьох шарів – m , внутрішня температура, зовнішня температура, коефіцієнт зовнішньої конвекційної передачі тепла – $\alpha_{outside}$, коефіцієнт внутрішньої конвекційної передачі тепла – α_{inside} .
Задамо

> $\lambda := [1, 0.04, 0.75]$:

$d := [0.05, 0.1, 0.5]$:

$T[inside] := -5$; $T[outside] := 25$:

$\alpha[inside] := 7$; $\alpha[outside] := 20$:

> $A := 1$; $n := 3$:

Розрахувавши удільний потік тепла через ділянку A , привласнимо змінній q

> $q := eq$;

$q := -8.798882680$

Підставивши це значення у перше рівняння, ми отримаємо поверхню температуру внутрішнього шару у градусах Цельсія

> $T[1] := \text{solve}(Q_inside = q, T[1])$;

$T_1 := -3.743016760$.3 $T_1 := -3.743016760$

Підставивши обчислене значення рекурсивно у друге рівняння, отримаємо температуру у з'єднанні шару **i+1** з шаром **i** у градусах Цельсія

```
> for i from 1 to n do  
T[i+1]:=solve(Q_i=q,T[i+1]); od;
```

$$T_2 := -3.303072626$$

$$T_3 := 18.69413408$$

$$T_4 := 24.56005587$$

Зобразимо графічно отриманий розподіл температури у багат шаровій стіні. Для цього скористаємось графічним пакетом `plottools`.

```
> with(plottools): with(plots):  
Warning, the names arrow and changecoords have been  
redefined  
> boundary:=[0]:  
for i from 1 to n do  
boundary:=[op(boundary),boundary[i]+d[i]]: od:  
pl1:=plot([[0,T[inside]], [boundary[n+1],T[outside]]],  
          style=point, symbol=circle,  
          color=green):  
pl2:=plot([seq([boundary[i],T[i]],i=1..n+1)],style=line, thickness=2):  
pl3 := seq(line([boundary[i],-6],  
               [boundary[i],26], color=blue,  
               linestyle=3),i=1..n+1):  
pl4 := textplot([ seq([boundary[i]/2,10,  
                      convert(i-1,string)],  
                      i=2..n+1)], color=blue):  
plots[display](pl1,pl2,pl3,pl4,  
               title="Розподіл температури у  
багат шаровій стіні",  
               labels=["      Відстань від внутрішньої  
стіни [m]","Temp. [°C] "]);
```

Розподіл температури у багат шаровій стіні

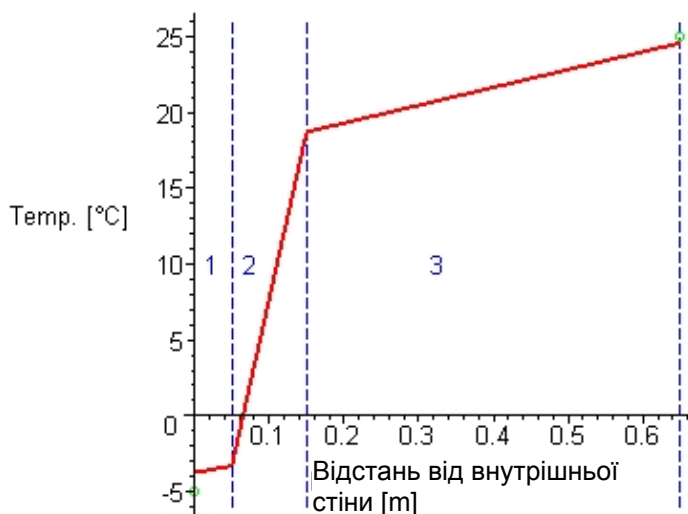


Рис. 3.5.

Бачимо, що найбільш температура змінюється у внутрішньому шарі стіни і вона потребує використання матеріалу з більшими теплоізоляційними властивостями.

3.4. Дослідження технічної системи з обслуговуванням

Нехай технічна система складається з n апаратів обслуговування і a – завантаження одного апарата. Система працює у імовірнісному режимі. Потрібно дослідити імовірність відмовлення в обслузі при різних значеннях кількості апаратів обслуговування і завантаження a .

Моделлю досліджуваної технічної системи є система масового обслуговування з відказами.

Позначимо через $[i]$ стан системи, коли на обслуговуванні знаходиться i запитів, а через P_i – стаціонарну ймовірність цього стану. Якщо λ – інтенсивність потоку запитів на обслуговування, а μ – інтенсивність обслуговування, то для ймовірностей P_i

справедлива система рівнянь [2]

$$-\lambda p_0 + \mu p_1 = 0,$$

$$-(\lambda + \mu)p_1 + \lambda p_0 + 2\mu p_2 = 0,$$

.....

$$-(\lambda + i\mu)p_i + \lambda p_{i-1} + (i+1)\mu p_{i+1} = 0,$$

.....

$$-n\mu p_n + \lambda p_{n-1} = 0.$$

Ця система доповнюється умовою нормування

$$\sum_{i=0}^n p_i = 1.$$

Послідовно розв'язуючи рівняння, отримаємо

$$p_1 = \frac{\lambda}{\mu} p_0 = a p_0, p_2 = \frac{a^2}{2!} p_0, \dots, p_i = \frac{a^i}{i!} p_0, i = 1, 2, \dots, n.$$

Підставивши отримані вирази p_i в умову нормування, отримаємо

$$p_0 = \left[\sum_{i=0}^n \frac{a^i}{i!} \right]^{-1}.$$

Тоді імовірність відмовлення в обслузі обчислюється за формулою

$$p_n = \frac{a^n / n!}{\sum_{k=0}^n a^k / k!}.$$

Розрахунку і побудова графіків по цій формулі потребує багато зусиль на програмування. У системі Maple, як показано нижче, це виконується однією командою

```
>plot([(0.5^n/n!)/(sum((0.5^k/k!),k=0..n)),(0.7^n/n!)/(sum((0.7^k/k!),k=0..n)),(0.95^n/n!)/(sum((0.95^k/k!),k=0..n)),0.05],n=1..10,p=0..0.3,labels=['n','p'],thickness=[2,2,2,0],color=black);
```

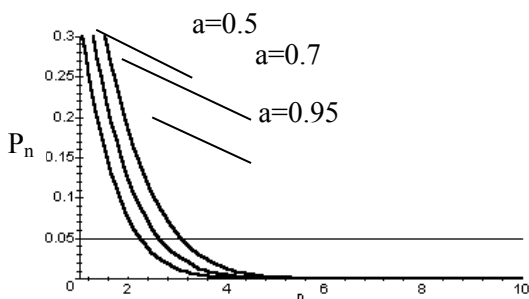


Рис. 3.6

Тонкою горизонтальною лінією відмічено рівень значимості у 0.05 по якому можна вибирати кількість апаратів у системі для надійної роботи.

3.5. Моделювання гнучкої виробничої системи

Гнучка виробнича система (ГВС) складається з станків з числовим програмним управлінням, транспортного роботу та автоматизованого складу, які керуються електронною обчислювальною машиною. Схема ГВС зображена на рис. 3.7.

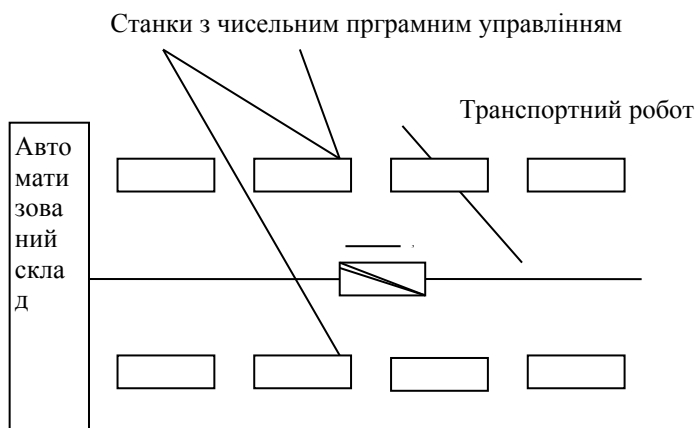


Рис. 3.7. Схема ГВС

Станки обробляють заготовки, які привозить їм транспортний робот. Час обробки станком однієї заготовки випадковий, його середнє значення w . Інтенсивність надходження запитів від одного станку $L=1/w$.

Транспортний робот обслуговує запити від станків на заготовки, підвозячи їх із складу. Час підвозу однієї заготовки

випадковий, його середнє значення v . Швидкість обслуговування транспортним роботом запитів від станків дорівнює $M = 1/v$. Кількість станків n . Якщо два або більш станків послали запити на заготовки, то транспортний робот обслуговує їх послідовно. Таким чином, поки транспортний робот підвозить заготовку одному станку, усі інші, які послали запити на заготовки, чекають.

Потрібно побудувати модель ГВС і з знайти коефіцієнт навантаження станків K_n та середню кількість працюючих станків S та побудувати відповідні графіки.

Для побудови моделі ГВС будемо використовувати апарат марківських випадкових процесів [2]. Для системи введемо

множину станів $\{i\}$ $i = 0, 1, 2, \dots, n$. Стан $\{i\}$ означає, що i станків обробляють заготовки, відповідно $n-i$ станків простоюють. Кожен стан $\{i\}$ характеризується ймовірністю p_i . Для ймовірностей p_i для стаціонарного стану можна скласти систему лінійних рівнянь:

$$\begin{aligned} -nLp_n + Mp_{n-1} &= 0, \\ -[(n-1)L + M]p_{n-1} + nLp_n + Mp_{n-2} &= 0, \\ -[(n-2)L + M]p_{n-2} + (n-1)Lp_{n-1} + Mp_{n-3} &= 0, \\ &\dots\dots\dots (3.4) \\ -(2L + M)p_2 + 3Lp_3 + Mp_1 &= 0, \\ -(L + M)p_1 + 2Lp_2 + Mp_0 &= 0, \\ -Mp_0 + Lp_1 &= 0. \end{aligned}$$

Всього рівнянь $n+1$. Крім того

$$\sum_{i=0}^n p_i = 1. \quad (3.5)$$

Рівняння (3.4) – (3.5) можна відразу розв'язувати на ЕОМ. Але у даному випадку можливо отримати аналітичний розв'язок, а потім аналізувати його на ЕОМ у системі **Maple**.
Із першого рівняння системи (3.4) випливає:

$$p_{n-1} = \frac{nL}{M} p_n = nbp_n = \frac{n!}{(n-1)!} b^1 p_n, \text{ де } b = \frac{L}{M}.$$

Якщо підставити p_{n-1} у друге рівняння системи (3.4), то після перетворень отримаємо

$$p_{n-2} = \frac{n!}{(n-2)!} b^2 p_n .$$

І нарешті

$$p_i = \frac{n!}{i!} b^{n-i} p_n .$$

Підставивши p_i у рівняння (3.5), отримаємо:

$$p_n = \left\{ \sum_{i=0}^n \frac{n!}{i!} b^{n-i} \right\}^{-1}, \text{ звідки } p_i = \frac{n!}{i!} b^{n-i} \left/ \sum_{k=0}^n \frac{n!}{k!} b^{n-k} \right. .$$

Коефіцієнт навантаження усіх станків у процентах

$$K_n = p_n \cdot 100\% = 100\% \left\{ \sum_{i=0}^n \frac{n!}{i!} b^{n-i} \right\}^{-1}$$

Середня кількість працюючих станків

$$S = \sum_{i=1}^n i p_i = \sum_{i=1}^n \frac{i n! b^{n-i}}{i!} \left/ \sum_{k=0}^n \frac{n!}{k!} b^{n-k} \right. .$$

Знайдемо середню кількість станків, що очікують обслуговування - Q .

Якщо система знаходиться у стані **{n-1}**, тоді один станок обслуговується і **0** чекають у черзі. Якщо система знаходиться у стані **{n-2}**, тоді один станок обслуговується і **1** чекає у черзі. Якщо система знаходиться у стані **{n-3}**, тоді один станок обслуговується і **2** чекають у черзі. Якщо система знаходиться у стані **{1}**, тоді один станок обслуговується і **n-2** чекають у черзі. Якщо система знаходиться у стані **{0}**, тоді один станок обслуговується і **n-1** чекають у черзі. Отже

$$Q = \sum_{i=0}^{n-1} p_i (n-i-1) = \sum_{i=0}^{n-1} n p_i - \sum_{i=0}^{n-1} i p_i - \sum_{i=0}^{n-1} p_i = n - S - (1 - p_n) .$$

Середня кількість запитів, що чекають Q , пропорційна середньому часу чекання обслуговування у черзі T , а середня кількість працюючих станків S пропорційна середньому часу обробки заготовки $w = 1/I$. Таким чином, середній час чекання обслуговування у черзі визначається виразом

$$T = \frac{Q_w}{S}.$$

Час функціонування станка складається із трьох частин: часу безпосередньої роботи w , часу чекання обслуговування T та часу обслуговування v . Тому коефіцієнт навантаження станків в процентах можна обчислити по наступному виразу:

$$K_n = \frac{w}{w + T + v}.$$

Програма на Maple мові та відповідні графіки шуканих залежностей приведені нижче.

```
> restart:
> w:=10:v:=2:b:=v/w:
> pn:=1/(sum(n!*b^(n-i)/i!,i=0..n)):
> S:=sum((n!*k*b^(n-k))/k!)*pn,k=1..n):
> Q:=n-S-(1-pn):
> T:=Q*w/S:
> Kz:=w/(w+v+T)*100:
> plot(S,n=0..10,y=0..6,color=black,thickness=2,
title="Середня кількість працюючих станків,
в залежності від їх числа ",labels=['n','S']));
```

Середня кількість працюючих станків,
в залежності від їх числа

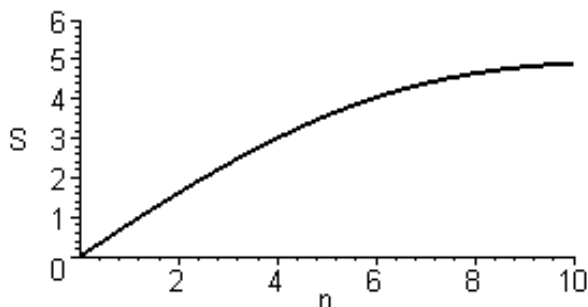


Рис. 3.8

```
plot(Kz,n=0..10,y=0..100,color=black,thickness=2
,title="Коефіцієнт навантаження станків в %
в залежності від їх числа ",labels=['n','Kn']));
```

Рис. 3.9

Коефіцієнт навантаження станків в %
в залежності від їх числа



Видно, що починаючи з деякого числа (у даному випадку вісім) подальше збільшення числа станків у ГВС недоцільно. Потрібно використовувати або більш продуктивний транспортний робот або добавляти ще один транспортний робот та переробляти транспортну схему ГВС.

Контрольні запитання і вправи

1. З яких законів виводиться рівняння теплопровідності?
2. Що таке диференціальне рівняння у частинних похідних і як вони розв'язуються аналітично?

3. Побудуйте модель розповсюдження тепла у стержні та дослідити її, якщо кінці стержня тепло ізолювані, тобто

$$u'_x(0, t) = 0,$$

$$u'_x(l, t) = 0.$$

4. Побудуйте модель розповсюдження тепла у стержні та дослідити її, якщо один кінець стержня тепло ізолюваний, а другий знаходиться при нульовій температурі, тобто

$$u'_x(0, t) = 0,$$

$$u_x(l, t) = 0.$$

5. Складіть вираз для середнього числа апаратів технічної системи, чекаючи обслуговування та дослідити його.

6. У чому переваги гнучкої виробничої системи перед звичайними виробничими системами? Назвіть склад ГВС.

7. Дослідити залежність основних характеристик ГВС від параметра b – завантаження транспортного робота запитами від одного станка.

4. Моделювання економічних процесів

4.1. Моделювання простішої виробничої діяльності фірми

Фірма планує випуск сонячних батарей. На підставі досліджень була встановлена залежність попиту Q від ціни P за батарею :

$$Q = 100000 - (200 - 10 \cdot N) \cdot P,$$

де N – довільне число.

Фірма випускає Q сонячних батарей і її витрати складають

$$C=(150000-100\ N)+100\ Q+0.003\ Q^2.$$

Потрібно розрахувати прибуток **S** і визначити оптимальний обсяг виробництва **Q** за допомогою функцій **extrema** і **maximize** системи **Maple**. Побудувати графік залежності **S=f(Q)** зеленим кольором із заголовком.

Розв'язання

Нехай **N=6**, тоді

> restart;

Розв'язуємо рівняння відносно ціни **p**

> p:=solve(100000-(200-10*6)*p-q=0,p);

$$p := \frac{5000}{7} - \frac{1}{140}q$$

Розраховуємо витрати

> c:=(150000-100*6)+100*q+0.003*q^2;

$$c := 149400 + 100q + 0.003q^2$$

Розраховуємо доход

> r:=q*p;

$$r := \left(q \frac{5000}{7} - \frac{1}{140} q^2 \right)$$

Обчислюємо прибуток

> s:=r-c;

$$s := \left(\frac{5000}{7} - \frac{1}{140}q \right) q - 149400 - 100q - .003q^2$$

Знаходимо екстремум

> readlib(extrema):extrema(s,{q},q,`y`);y;

Будуємо графік функції
 $\{.9151404837 \cdot 10^7, \{ \{q = 30281.69014\} \}$

> plot(s,q=0..100000,color=green,title=`Прибуток`);

Знаходимо тип екстремуму та його значення



```
> readlib(maximize):maximize(s,q=1..1000000);
.9151404837 107
```

Система **Maple** дозволила не тільки обчислити оптимальні параметри, а й з допомогою графіка дослідити залежність прибутку від об'єму випуску.

4.2. Моделювання виробничої діяльності багато профільної фірми

Фірма виготовляє два види товарів **G1** і **G2** і продає їх за цінами 1000 грн. і 800 грн., відповідно. Обсяги випуску товарів **G1** і **G2** - **Q1** і **Q2**, відповідно. Функція витрат має вигляд

$$c = 2q_1^2 + 2q_1q_2 + q_2^2$$

Знайти обсяги випуску **q1** і **q2** при яких прибуток фірми максимальний. Знайти значення цього прибутку. Побудувати графік залежності прибутку від обсягів виробництва.

Розв'язання

```
> restart:
```

Разраховуємо доход

```
> r:=q1*1000+q2*800;
```

$$r := 1000 q_1 + 800 q_2$$

Розраховуємо витрати

```
> c:=2*q1^2+2*q1*q2+q2^2;
```

$$c := 2 q_1^2 + 2 q_1 q_2 + q_2^2$$

Знаходимо прибуток

```
> s:=r-c;
```

$$s := 1000 q_1 + 800 q_2 - 2 q_1^2 - 2 q_1 q_2 - q_2^2$$

Обчислюємо екстремум

```
> readlib(extrema):extrema(s,{},{q1,q2},`x`);x;
```

$$\{170000\}, \{q_1 = 100, q_2 = 300\}$$

Будуємо графік

```
> plot3d(s(q1,q2),q1=0..1000,q2=0..1000);
```

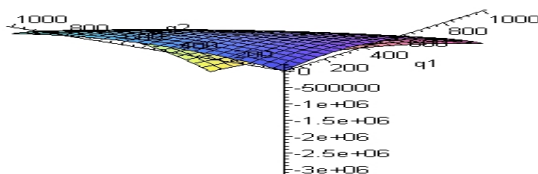


Рис. 4.2

Знову обчислюємо екстремум та знаходимо його вигляд
> readlib(maximize):maximize(s,q1=1..100000,q2=1..100000);

170000

Із графіка видно, що екстремум носить не явно виражений характер

4.3. Моделювання виробничої діяльності фірми в умовах монополістичної змови

Фірма виготовляє два види товарів **G1** і **G2** у кількості **Q1** і **Q2**. Функція витрат має вигляд

$$C=10 Q_1+Q_1 Q_2+10 Q_2,$$

а залежності попиту кожного товару описуються виразами

$$Q_1=P_1 + P_2 - 80$$

$$Q_2=2*P_1 + P_2 - 130,$$

де **P1** і **P2** - ціна одиниці товару видів **G1** і **G2**, відповідно. Фірма випускає товари у відповідності з попитом. Крім того фірма зв'язана обмеженнями з конкурентами на загальний обсяг виробництва товарів **G1** і **G2**; її квота згідно змови складає 15 одиниць. Треба визначити максимальний прибуток при цих умовах і оптимальний план виробництва товарів .

Розв'язання

> restart;

Нехай **Q1** і **Q2** обсяги виробництва товарів **G1** і **G2** відповідно

Формуємо функцію витрат

> C:=10*Q1+Q1*Q2+10*Q2;

Визначаю ціни **P1** і **P2** через обсяги виробництва **Q1** і **Q2**

> s:=solve({Q1=P1+P2-80,Q2=2*P1+P2-130},{P1,P2});

$$s := \{P_2 = -Q_2 + 2 Q_1 + 30, P_1 = -Q_1 + Q_2 + 50\}$$

Привласнюю змінним **P1** і **P2** розв'язок рівняння

> **assign(s);P1;P2;**

$$-Q_1 + Q_2 + 50$$

$$-Q_2 + 2 Q_1 + 30$$

Формуємо доходи

> **R1:=P1*Q1;**

$$R_1 := (- Q_1 + Q_2 + 50) Q_1$$

> **R2:=P2*Q2;**

$$R_2 = (- Q_2 + 2 Q_1 + 30) Q_2$$

> **R:=R1+R2;**

$$R := (- Q_1 + Q_2 + 50) Q_1 + (- Q_2 + 2 Q_1 + 30) Q_2$$

Визначаємо прибуток і формуємо функцію користувача **W(Q1,Q2)**

> **W:=unapply(R-C,Q1,Q2);**

$$W := (Q_1, Q_2) \rightarrow (- Q_1 + Q_2 + 50) + (- Q_2 + 2 Q_1 + 30) - 10 Q_1 - Q_1 Q_2 - 10 Q_2$$

Перевірка

> **W(9.99,5.01);**

$$474.9996$$

Обчислюємо екстремум

> **readlib(extrema);extrema(W(Q1,Q2),15-Q1-Q2=0,{Q1,Q2},`V`);V;**

$$\{475\}$$

$$\{Q_2 = 5, Q_1 = 10\}$$

Будуємо графік

> **plot(W(Q1,15-Q1),Q1=1..14);**

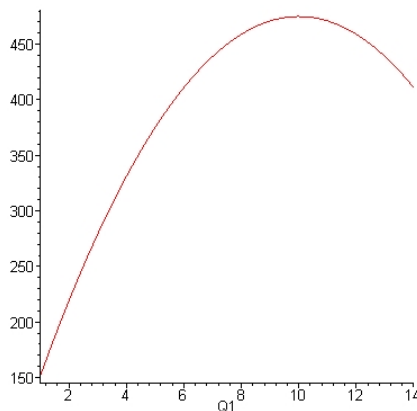


Рис. 4.3

У даному випадку моделювання діяльності фірми виконано при умові додаткового обмеження на сумарний об'єм продукції.

4.4. Дослідження еластичності попиту на товар, залежною від часу

За спостереженнями попит на товар залежить від часу і визначається співвідношенням

$$D(t) = 2 \left(\frac{\partial^2}{\partial t^2} P(t) \right) - \left(\frac{\partial}{\partial t} P(t) \right) - P(t) + 15$$

Пропозиція товару описується співвідношенням

$$S(t) = 3 \left(\frac{\partial^2}{\partial t^2} P(t) \right) + \left(\frac{\partial}{\partial t} P(t) \right) + P(t) + 5$$

де $P(t)$ - функція ціни від часу і $P(0)=6$, $D(0)=S(0)=10$. У припущенні, що $D(t)=S(t)$, знайти залежність ціни від часу. Знайти еластичність попиту за ціною і побудувати її графік.

Розв'язання

> restart;

Описуємо залежність попиту від часу. Так як буква **D** у системі **Maple** є зарезервованою для системних цілей, замість неї використовуємо **Q**

> Q:=2*diff(P(t),t,t)-diff(P(t),t)-P(t)+15;

$$Q(t) = 2 \left(\frac{\partial^2}{\partial t^2} P(t) \right) - \left(\frac{\partial}{\partial t} P(t) \right) - P(t) + 15$$

Описуємо залежність пропозиції від часу

> S:=3*diff(P(t),t,t)+diff(P(t),t)+P(t)+5;

$$S(t) = 3 \left(\frac{\partial^2}{\partial t^2} P(t) \right) + \left(\frac{\partial}{\partial t} P(t) \right) + P(t) + 5$$

Формуємо рівняння рівності попиту і пропозиції

> W:=Q-S=0;

$$W(t) = -\left(\frac{\partial^2}{\partial t^2} P(t)\right) - 2\left(\frac{\partial}{\partial t} P(t)\right) - 2P(t) + 10 = 0$$

Розв'язуємо це рівняння відносно ціни товару з урахуванням початкових умов і привласнюємо цей розв'язок (його праву частину) проміжній змінній **P2**

>P2:=rhs(dsolve({Q-S=0,P(0)=6},P(t)));

$$P2 := \frac{5e^t + \cos(t) + _C2\sin(t)}{e^t}$$

Зазначимо, що поки ні **P2**, ні **P(t)** не є повноцінними функціями. Дійсно

>P(0);

P(0)

Формуємо вираз для залежності ціни від часу у вигляді повноцінної функції

>P:=unapply(P2,t);

$$P := t \rightarrow \frac{5e^t + \cos(t) + _C2\sin(t)}{e^t}$$

Зараз **P(t)** вже є повноцінною функцією. Дійсно

P(0)=6

Знаходимо остаточний вигляд для попиту.

>Q:=unapply(Q,t);

$$Q := t \rightarrow 2 \frac{5e^t - \cos(t) + _C2 \sin(t)}{e^t} - 5 \frac{5e^t - \sin(t) + _C2 \cos(t)}{e^t} + 2 \frac{5e^t + \cos(t) + _C2 \sin(t)}{e^t} + 15$$

Із початкових умов знаходимо константу **_C2**

> _C2:=solve(Q(0)=10,_C2);

_C2:=0 .

Формуємо остаточний вираз для ціни

> P(t);

$$\frac{5e^t + \cos(t)}{e^t}$$

Формуємо залежність пропозиції від часу

> S:=unapply(S,t);

$$S := t \rightarrow 3 \frac{5e^t - \cos(t)}{e^t} - 5 \frac{5e^t - \sin(t)}{e^t} + \frac{5e^t + \cos(t)}{e^t} + 15$$

Спростуємо цей вираз

> simplify(S(t));

$$10 + 5 \sin(t) e^{(-t)}$$

Формуємо і спростуємо залежність попиту від часу

> simplify(Q(t));

$$10 + 5 \sin(t) e^{(-t)}$$

Знаходимо еластичність попиту по ціні, враховуючи, що і

попит і ціна є функціями параметру t , і одночас спрощуємо його

```
> E2:=simplify((diff(Q(t),t)/diff(P(t),t))*P(t)/Q(t),
- 2et sin(t) - 1 - sin(t) cos(t) - 2 cos(t) et + cos(t)2)
```

$$E2 = \frac{-5 \cos(t) - e^{(-t)} \cos(t)^2 + 5 \sin(t) + e^{(-t)} \sin(t) \cos(t)}{2 \sin(t) + e^{(-t)} - e^{(-t)} \cos(t)^2 + 2 \cos(t) + e^{(-t)} \sin(t) \cos(t)}$$

Формуємо остаточний вираз для еластичності у вигляді повноцінної функції

```
> E:=unapply(E2(t),t);
```

Будуємо сумісний графік ціни та еластичності в залежності від часу (для ціни вибираємо товсту лінію)

```
> plot([P(t),E(t)],t=0..2*Pi,E=-20..20, thickness=[3,0];
```

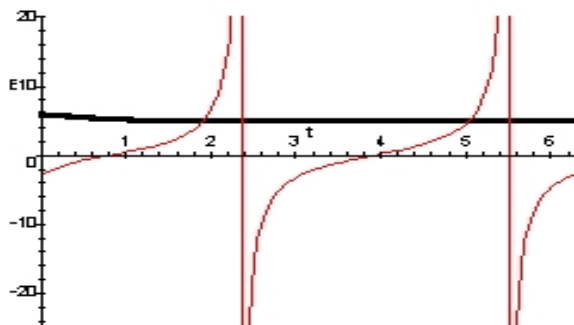


Рис. 4.4

Бачимо, що з часом еластичність змінюється у діапазоні від мінус нескінченності до плюс нескінченності, так звані сезонні коливання. Водночас бачимо суттєвий недолік побудованої моделі: як правило цінова еластичність лежить в діапазоні від 0 до $-\infty$.

4.5. Дослідження рекламної діяльності фірми

Торгова фірма реалізує товар, про який в момент часу t з числа потенційних покупців N знає тільки x покупців. Фірма проводить гучну рекламну компанію, у результаті якій кількість потенційних покупців, знайомих з товаром безперервно збільшується. За період часу Δt число потенційних покупців, знайомих з товаром, збільшиться на величину, пропорційну Δt , числу покупців, що нічого не знають про товар і числу покупців, знайомих з товаром, з коефіцієнтом пропорційності k . Відомо, що в початковий момент часу $t=0$ про товар знали N/g покупців (час відраховується від моменту початку рекламних заходів), g - задане число.

Треба найти закон зміни числа потенційних покупців, знайомих з товаром, у залежності від часу, побудувати графік процесу.

Розв'язання

> restart:

Кількість потенційних покупців, знайомих з товаром у момент часу $t + \Delta t$ обчислюється по формулі

$$n(t + \Delta t) = n(t) + kn(t)(N - n(t))\Delta t$$

Переносячи $n(t)$ у ліву частину, ділячи на Δt і переходячи до границі, коли Δt прямує до нуля, отримаємо наступне диференціальне рівняння

$$\text{diff}(n(t),t)-k*n(t)*(N-n(t))=0;$$

Привласнюємо його змінній **Purch**

$$> \text{Purch}:=\text{diff}(n(t),t)-k*n(t)*(N-n(t))=0;$$

$$\text{Purch} := \left(\frac{\partial}{\partial t} n(t) \right) - kn(t)(N - n(t)) = 0$$

Розв'язуємо це диференціальне рівняння у загальному вигляді

$$> \text{p1}:=\text{dsolve}(\{\text{Purch},n(0)=N/g\},n(t));$$

$$p1 := n(t) = \frac{N}{1 + e^{(-kNt)}(g - 1)}$$

Формуємо повноцінну функцію

$$> \text{p}:=\text{unapply}(\text{rhs}(\text{p1}),t);$$

$$p := t \rightarrow \frac{N}{1 + e^{(-kNt)}(g - 1)}$$

Привласнюємо параметрам виразу конкретні значення

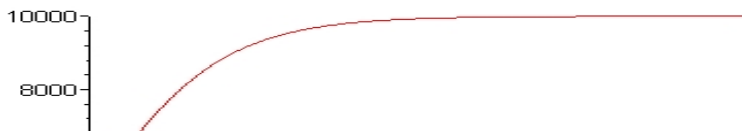
$$> \text{N}:=10000:k:=0.0001:g:=2:$$

$$> \text{p}(t);$$

$$1000 \frac{1}{1 + e^{(-1.0000t)}}$$

Отримана залежність називається рівнянням логістичної кривої, яка широко використовується у техніці та на транспорті. Будуємо її графік

$$> \text{plot}(\text{p}(t),t=0.0..10,p=0..10000);$$



Бачимо, що з деякого часу усі потенційні покупці дізнаються про рекламований товар і подальша реклама недоцільна.

4.6. Комп'ютерна модель міжгалузевого балансу (модель Леонтьева В.В.)

На прикладі цієї моделі буде показано використання операцій над матрицями та елементів програмування у моделюванні техніко-економічних систем.

Будь яке сучасне національне господарство розвивається у складній системі міжгалузевих взаємозв'язків, зрозуміти які у всій їх сукупності шляхом простого складання неможливо. Наприклад, попит на автомобілі чинить вплив не тільки на автомобільну промисловість, але й побічним шляхом на металургію та галузі, які виробляють шини, бортову електроніку, а через них на виробництво мікросхем. Зміни в одній галузі неминуче позначається на усім народнім господарстві.

Національне господарство, якщо воно замкнуте, знаходиться у балансі виробництва та споживання. Для аналізу міжгалузевих зв'язків такого господарства видатний економіст Леонтьєв В.В. розробив спеціальний метод – “аналіз витрати-випуск” (Input-output analysis або I/O analysis), який дозволяє дослідити процеси, зв'язані з міжгалузевими взаємодіями.

Нехай увесь виробничий сектор національній економіці поділено на n “чистих галузей”. Словосполучення “чиста галузь” означає, що продукція кожній галузі є однорідною.

Чиста галузь є економічною абстракцією, не обов'язково існуюча у вигляді якихось організаційних форм, тобто це є модель. Наприклад, під галуззю “електроенергетика” можна розуміти сукупність усіх електростанцій, незалежно від їх належності. Подібна ідеалізація дозволяє провести ретельний аналіз сформованої технологічної структури суспільного виробництва та розподілу.

Припустимо, що кожна галузь випускає продукт тільки одного виду і різні галузі виробляють різні продукти. В процесі виробництва свого виду продукту кожна галузь потребує в продукції інших галузей.

Нехай у якийсь момент часу T_0 складено балансовий звіт по національній економіці по підсумковим даним за фіксований період часу, наприклад за минулий рік, по наступній формі.

Таблиця 4.1

Галузі виробництва								Кінцевий попит (споживання, інвестиції, експорт(+), імпорт(-))	Об'єм випуску
		<i>1</i>	<i>2</i>		<i>j</i>		<i>n</i>		
Галузі попуки (сектора продажі ту)	Галузі попи (сектора пропозицій)								
Галузі виробництва	<i>1</i>	α_{11}	α_{12}		α_{1j}		α_{1n}	d_1	q_1
	<i>2</i>	α_{21}	α_{22}		α_{2j}		α_{2n}	d_2	q_2
		..	..	.	.	..	..		
		..	..	.	.	..	..		
	<i>i</i>	α_{i1}	α_{i2}		α_{ij}		α_{in}	d_i	q_i
		..	..	.	.	..	..		
		..	..	.	.	..	..		
	<i>n</i>	α_{n1}	α_{n2}		α_{nj}		α_{nn}	d_n	q_n
Додавлена вартість (прибуток зайнятих по найму, → підприємницький прибуток, амортизаційні відрахунки, побічні податки та інше)		g_1	g_2		g_j		g_n		
Об'єм витрат →		q_1	q_2		q_j		q_n	D	V

Строки приведеної таблиці показують розподіл випуску

(output) кожного виду продукції. Величина α_{ij} показує об'єм продукції галузі i , яку закупила у неї галузь j ($i, j = 1, 2, \dots$) в якості проміжних продуктів. Кінцевий попит d_i показує об'єм продукції i -ї галузі, який був споживам на інвестиції, експорт, для створення запасів. Число q_i дорівнює загальному об'єму продукції (валовому випуску) i -ї галузі за звітний період. Кожен рядок характеризується наступним балансом:

Випуск даного виду продукції = проміжний попит + кінцевий попит,

що математично може бути записано як

$$q_i = (\alpha_{i1} + \alpha_{i2} + \dots + \alpha_{in}) + d_i, \quad i = 1, 2, \dots, n. \quad (4.1)$$

Стовпці таблиці показують структуру затрат (input), або структуру використовуваних ресурсів, необхідних для кожної галузі. Для стовпців установлюється наступний баланс:

витрати галузі = проміжні витрати + добавлена вартість,
що в математичному запису виглядає як

$$q_j = (\alpha_{1j} + \alpha_{2j} + \dots + \alpha_{nj}) + g_j, \quad j = 1, 2, \dots, n. \quad (4.2)$$

Проміжні витрати є початкові матеріали, закуплені галуззю у галузей (секторів) $1, 2, \dots, n$. Добавлена вартість є факторними витратами галузі, тобто знов побудована вартість, яка розпадається на заробітну плату, підприємницький прибуток, амортизаційні відрахування.

Для національної економіки, виходячи із закону збереження, повинно виконуватися відношення:

Випуск галузі = витрати галузі,

Загальна сума кінченого попиту = загальна сума добавленої вартості.

Математично це записується наступним чином:

$$q_i = \sum_{j=1}^n \alpha_{ij} + d_i = \sum_{j=1}^n \alpha_{ji} + g_i, \quad i = 1, 2, \dots, n. \quad (4.3)$$

$$\sum_{i=1}^n d_i = \sum_{i=1}^n g_i.$$

EMBED Equation.3

(4.4)

Рівняння (4.3) та (4.4) називаються балансовими рівняннями. Одиниці виміру усіх вказаних величин можуть бути або натуральними (тони, штуки) або вартісними, в залежності від чого розрізняють натуральний і вартісний баланс. В наступному будимо мати на увазі вартісний баланс.

Таблиця 4.1 також називається таблицею міжгалузевого балансу і вона дозволяє вивчати структуру потоків ресурсів у національній економіці. Але більш інформативними є відносні величини. Якщо усі елементи a_{ij} таблиці 4.1 поділити на величини q_j (об'єм продукції j – ї галузі), а числа d_i на q_i , то число $a_{ij} = a_{ij} / q_j, i, j = 1, 2, \dots, n$, можна розуміти як об'єм продукції i – ї галузі, потрібної для виробництва однієї одиниці продукту галузі з номером j ; число $c_i = d_i / q_i, i = 1, 2, \dots, n$, як долю продукції i – ї галузі, яка пішла на невиробниче споживання.

Числа $a_{ij}, i = 1, 2, \dots, n$, носять назву коефіцієнтів прямих, безпосередніх витрат j – ї галузі і у деякому сенсі повністю характеризують технологію j – ї галузі у звітному періоді: випуск одиниці продукції можливо при структурі витрат, характеризованими величинами a_{ij} . Виходячи з економічного глузду, вважаємо коефіцієнти a_{ij} невід'ємними.

Матриця $A = [a_{ij}]$, яка складається з коефіцієнтів прямих витрат a_{ij} , називається структурною матрицею економіки, і вона несе багато інформації о структурі міжгалузових зв'язків та існуючої технології суспільного виробництва, які склалися у народному господарстві. Порівнюючи такі матриці, складені у різні моменти часу, можна простежити напрямок зміни та розвитку технології.

Наприклад, деякі коефіцієнти витрат праці при семи десяти шести галузевій схемі класифікації галузей народного господарства США мають вигляд

Таблиця 4.2

Назва галузі	1947 р.	1958 р.
Автомобілі та обладнання	0,2149	0,1569

Ремонт автомобілів	0,4769	0,4969
Нафтопереробка	0,1328	0,0881

З таблиці 4.2 видно, що внаслідок впровадження прогресивної технології різко зменшилися витрати на виробництво автомобілів та нафтопереробку, але у сфері ремонту автомобілів, де переважає ручна праця, відносні витрати зросли.

Матрицю **A** можна використовувати для поточного та довгострокового планування. Зробимо наступні припущення:

- Існуюча технологія виробництва незмінна вздовж деякого проміжку часу $[T_0, T]$, де $T > T_0$. В залежності від поставленої задачі проміжок $[T_0, T]$ може дорівнювати одному календарному періоду (рік) або декільком.

- Технологія виробництва лінійна, тобто будемо вважати, що для випуску продукції **i-ї** галузі об'ємом x_i необхідно і достатньо зробити витрати в об'ємах $x_i a_{ij}$, $j=1,2,...,n$, продукції кожної галузі.

Усе це разом приводить до того, що ми будемо розглядати ідеалізацію реальних процесів - модель. Ми не можемо вироблювати який завгодно об'єм продукції, так як не вистачить перш за все виробничих потужностей.

Нехай **i-та** галузь повинна виробляти об'єм x_i , $i=1,2,...,n$, валового випуску своєї продукції. Позначимо через **[X]** матрицю-

$$\begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix}$$

стовпець валового випуску усіх секторів. Скориставшись припущенням о лінійності, розрахуємо витрати галузі **i** на випуск продукції у інших галузях. Це буде

$$a_{i1}x_1 + a_{i2}x_2 + \dots + a_{in}x_n \quad (4.5)$$

З рівняння балансу випливає, що вираз (4.5) дорівнює x_i мінус кінцевий попит, тобто

$$\sum_{j=1}^n a_{ij} x_j = x_i - d_i, \quad i = 1, 2, \dots, n,$$

де d_i кінцевий попит на продукцію i -ї галузі.

У матричному вигляді $[A] \cdot [X] = [X] - [D]$, або

$$([I] - [A]) \cdot [X] = [D], \quad (4.6)$$

де $[D]$ вектор кінцевого попиту національної економіки, $[I]$ одинична матриця.

Рівняння (4.6) разом із інтерпретацією матриці $[A]$ та векторів $[X]$, $[D]$ називається моделлю Леонтьєва. При плануванні ставиться завдання: знайти вектор $[X]$ валового випуску, щоб задовольнити вектор кінцевого попиту $[D]$ при існуючій структурі економіки, яка задається матрицею $[A]$.

Якщо задано вектор кінцевого попиту $[D]$ і існує невід'ємний вектор виробництва $[X]$, який задовольняє рівнянню (4.6), то модель Леонтьєва називається продуктивною.

Для встановлення продуктивності моделі Леонтьєва можна використовувати наступні умови:

- Матриця $[A]$ повинна бути нерозкладаною, тобто вона не повинна мати ізольованих підмножин, окрім самої матриці та пустої матриці.
- Сума елементів кожної строки матриці $[A]$ повинна бути не більш одиниці і хоча б для однієї строки строго менш одиниці.

Теоретичний розв'язок рівняння (9) вельми компактний:

$$[X] = ([I] - [A])^{-1} \cdot [D] = [Zv] \cdot [D],$$

де $[Zv] = ([I] - [A])^{-1}$ - зворотна матриця Леонтьєва. $[Zv]$ є ні що інше, як матриця коефіцієнтів повних витрат. Економічний зміст її елементів z_{vj} міститься в наступному: коефіцієнт z_{vj} показує загальну потребу у валовому випуску продукції галузі i для виробництва одиниці кінцевої продукції галузі j .

Наприклад, для випуску автомобілів потрібна продукція металургійної галузі, це буде первинний попит; також потрібні підшипники, для випуску яких також потрібна продукція металургійної галузі – це утворює вторинний попит з боку галузі автомобілебудування на продукцію металургійної галузі і так далі.

Розглянемо процедуру планування з використанням моделі Леонтьєва на прикладі даних економіки США за 1958 рік у системі *Maple*.

>restart:Digits:=7:

>with('linalg'): # виклик пакета лінійної алгебри

Модель міжгалузевого балансу економіки США за даними 1958 р. (млн. дол.) задана наступною таблицею

> ls:=array(1..12,1..12,[

```
[Sek, 1, 2, 3, 4, 5, 6, 7, 8, Sum, Prod, ValVup],
[1, 8565, 8069, 8843, 3045, 1124, 276, 230, 3464, 33616, 3992, 37608],
[2, 1505, 6996, 6895, 3530, 3383, 365, 219, 2946, 25839, 19261, 45100],
[3, 98, 39, 5, 429, 5694, 7, 376, 327, 6975, 39347, 46322],
[4, 999, 1048, 120, 9143, 4460, 228, 210, 2226, 18434, 22625, 41059],
[5, 4373, 4488, 8325, 2729, 29671, 1733, 5757, 14756, 71832, 137572, 209404],
[6, 2150, 36, 640, 1234, 165, 821, 90, 6717, 11853, -654, 11199],
[7, 506, 7, 180, 0, 2352, 0, 18091, 26529, 47665, 8327, 55992],
[8, 5315, 1895, 2993, 1071, 13941, 434, 6096, 46338, 78083, 82997, 161080],
[Sum, 23511, 22578, 28001, 21181, 60790, 3864, 31069, 103303, 294297, 313467, 607764],
[Doh, 14097, 22522, 18321, 19878, 148614, 7335, 24923, 57777, 313467, xxxxxx, xxxxxx],
[Zat, 37608, 45100, 46322, 41059, 209404, 11199, 55992, 161080, 607764, xxxxxx, 607764]]];
```

	Sek	1	2	3	4	5	6	7	8	Sum	Prod	ValVup
	1	8565	8069	8843	3045	1124	276	230	3464	33616	3992	37608
	2	1505	6996	6895	3530	3383	365	219	2946	25839	19261	45100
	3	98	39	5	429	5694	7	376	327	6975	39347	46322
	4	999	1048	120	9143	4460	228	210	2226	18434	22625	41059
ls=	5	4373	4488	8325	2729	29671	1733	5757	14756	71832	137572	209404
	6	2150	36	640	1234	165	821	90	6717	11853	-654	11199
	7	506	7	180	0	2352	0	18091	26529	47665	8327	55992
	8	5315	1895	2993	1071	13941	434	6096	46338	78083	82997	161080
	Sum	23511	22578	28001	21181	60790	3864	31069	103303	294297	313474	607764
	Doh	14097	22522	18321	19878	148614	7335	24923	57777	313467	xxxxxx	xxxxxx
	Zat	37608	45100	46322	41059	209404	11199	55992	161080	607764	xxxxxx	607764

Де:

Sek – галузі (сектори), **Sum** – проміжні витрати або проміжний попит; **Doh** – добавлена вартість, **Zat** витрати галузі, **Prod** – кінцевий попит, **ValVup** – випуск продукції.

1 - галузь матеріалів,

2 - металообробні галузі,

3 - будівництво,

4 - галузь, що поєднує транспортне устаткування, електрику і газ,

5 - послуги і транспорт,

6 - гірничодобувна промисловість,

7 - сільське господарство і суміжні галузі,

8 - усі інші галузі .

На відміну від таблиці 1, матриця **Is** (початкова) має додатковий стовпець **Sum** та додаткову строку **Sum** – сума коефіцієнтів α по стовпцям та по строкам.

Сплануємо випуск продукції у галузях промисловості у млн. доларів США, щоб задовольнити споживанню по кожній галузі, заданого таблицею

Номер галузі	Об'єм продукції галузі, що іде на споживання (млн. дол.)
1	6000
2	40000
3	39000
4	22000
5	300000
6	500
7	8000
8	90000

Формуємо матрицю коефіцієнтів прямих витрат , що відповідає міжгалузовому балансу:

- Утворюємо матрицю **RashM** розміром **8x8** з елементів α шляхом їх вилучення з матриці **Is** за допомогою функції **submatrix**

>RashM:=submatrix(IshodnM,2..9,2..9);

$$\text{RashM} = \begin{bmatrix} 8565 & 8069 & 8843 & 3045 & 1124 & 276 & 230 & 3464 \\ 1505 & 6996 & 6895 & 3530 & 3383 & 365 & 219 & 2946 \\ 98 & 39 & 5 & 429 & 5694 & 7 & 376 & 327 \\ 999 & 1048 & 120 & 9143 & 4460 & 228 & 210 & 2226 \\ 4373 & 4488 & 8325 & 2729 & 29671 & 1733 & 5757 & 14756 \\ 2150 & 36 & 640 & 1234 & 165 & 821 & 90 & 6717 \\ 506 & 7 & 180 & 0 & 2352 & 0 & 18091 & 26529 \\ 5315 & 1895 & 2993 & 1071 & 13941 & 434 & 6096 & 46338 \end{bmatrix}$$

Утворюємо матрицю прямих витрат шляхом ділення елементів

стовпців матриці **RashM** на загальний об'єм витрат по кожній галузі, які вибираються з **12-тої** строки матриці **Is**. Так як операції ділення у пакеті **linalg** немає заміняємо її множенням на зворотну величину. Так як операцію множення можна виконувати лише по окремим стовпцям, оформляємо її у циклі програмним шляхом. При цьому буде виводитися багато проміжних даних, тому усе оформляємо у секцію (вона зараз відкрита).

```
- #Формування матриці коефіцієнтів витрати-випуск
```

```
> EE:=matrix(8,8):
> copyinto(RashM,EE,1,1):
> for i from 1 to 8 do
    EE:=mulcol(EE,i,(1/(IshodnM[12,i+1]))) od:
> A:=matrix(8,8):
> A:=convert(eval(EE),float):
```

```
>evalm(A);
```

```
[.2277441 .1789135 .1909069 .07416338 .005367615 .02462527 .004107730 .02150484
.04001808 .1551220 .1488526 .08597594 .01615537 .03256602 .003911273 .01828905
.002605829 .00086474 .0001079424 .01044863 .02719146 .00062455 .006715245 .00203004
.02656350 .02323725 .002590618 .2226850 .02129854 .02034261 .003750536 .01381922
.1162785 .09951220 .1797241 .06646695 .1416926 .1546217 .1028183 .09160666
.05716869 .00079822 .01381663 .03005504 .000787950 .07325125 .001607372 .04169978
.01345458 .00015521 .003885926 0 .01123188 0 .3230997 .1646946
.1413263 .04201774 .06461432 .02608505 .06657466 .03872234 .1088727 .2876707]
```

Знаходимо зворотну матрицю Леонтьєва. Для цього сформуємо одиничну діагональну матрицю:

```
>I:=band([1],8);
```

$$I = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

За допомогою матричних операцій обчислюємо зворотну матрицю Леонтьєва

```
>Zv:=evalm((I-A)^(-1));
```

1.339392	.2960476	.3115558	.1721120	.03372644	.05840473	.02991055			
.06693522									
.08865971	.213901	.2090991	.1527271	.03817543	.05702074	.02466648			
.05135275									
.01289261	.0089487	1.011064	.01945329	.03398887	.00796947	.01750241			
.01276199									
.06457548	.0562216	.0342750	.305546	.03842964	.04049442	.02080906			
.04094215									
Zv:=	.2648177	.2155171	.3195975	.1735424	1	.207020	.2300262	.2293987	.2395344
	.09986208		.02884338	.0454198	.05887385		.01106307	1.088982	
	.01776728	.07430657							
	.1093064	.04931747	.0684206	.03503676			.05376374	.02995881	
	1.547186	.3718269							
	.3213846	.1624581	.2099102	.1175035	.1350609	.1024312	.2686170	1.506068	

Прорахування контрольного приклада.

Формуємо матрицю кінцевого попиту по 11-му стовпцю матриці **Is**, яка описує економіку США.

>**Dem:=matrix(8,1,[3992,19261,39347,22625,137572,-654,8327,82997]):**

3992
19261
39347
Dem:= 22625
137572
-654
8327
82997

Так як символ **D** зарезервовано за позначенням диференціального оператора, то для позначення вектора-стовпця кінцевого попиту (demand) вибране сполучення **Dem**.

Обчислюємо об'єми випуску по кожній галузі

>**X1:= multiply(Zv,dd);**

X1:=
37608.00
45100.01
46321.99
41059.00
209404.9
11199.00
55992.01
161080.0

Порівнюючи цей результат з реальним валовим випуском (остання колонка матриці **Is**), бачимо, що модель працює з досить великою точністю.

Розв'яжемо поставлену задачу. Формуємо матрицю кінцевого попиту

> Dem:=matrix(8,1,[6000,40000,39000,22000,300000,500,8000,90000]);

$$Dem := \begin{bmatrix} 6000 \\ 40000 \\ 39000 \\ 22000 \\ 300000 \\ 500 \\ 8000 \\ 90000 \end{bmatrix}$$

Обчислюємо потрібний випуск продукції по кожній галузі, щоб задовольнити цей попит

X2:=multiply(Zv,Dem);

$$X2 := \begin{bmatrix} 52226.17 \\ 76903.39 \\ 51784.10 \\ 48095.57 \\ 412108.5 \\ 15513.64 \\ 68054.02 \\ 197463.7 \end{bmatrix}$$

Аналогічно можна планувати виробництво і при інших показниках.

Контрольні запитання і вправи

1. Що таке прибуток фірми і як він обчислюється?
2. Побудуйте повноважну функцію для прибутку фірми.
3. Що таке еластичність і який її економічний зміст?
4. Як обчислюється екстремум аналітично і як можна знайти екстремум функції за допомогою системи Maple?
5. Що таке логістична крива і де ще вона застосовується?
6. Що таке міжгалузевий баланс в економіці?
7. Що описує таблиця міжгалузевого балансу?
8. Яка модель міжгалузевого балансу називається продуктивною?
9. Складіть програму дослідження моделі міжгалузевого балансу на продуктивність.

Бібліографічний опис

1. Дьяконов В.П. Математична система Maple: учебный курс. – СПб.: Питер. – 2001. -608 с.
2. Литвинов А.Л. Вероятностное моделирование производственных систем. Харьков: ХАИ. -1989. 95 с.
3. Малыхин В.И. Математическое моделирование в экономике. Учебно-практическое пособие для ВУЗов. – М.: Изд-во УРАО, 1998. -160 с.
4. Манзон Б. М. Maple V. Power Edition. М.: Филинь, 1998.- 405 с.
5. Математичні методи та моделі в розрахунках на ПЕОМ. Програма, методичні вказівки та контрольні завдання для студентів – заочників інженерно-технічних спеціальностей ./ І. В. К. Крикова, О.М.. Литвин, Л.С. Лобанова, І.С. Храповицький. – Харків: УІПА, 2000. -128 с.
6. Попов В.А. Розв'язування математичних задач у системі комп'ютерної алгебри Maple V. Київ: VIP, 2001 с. – 310 с.

Предметний покажчик

Алгоритм 12	Оригінал 2
Баланс 88	Палітра математичних виразів 22
Блок-схема 13	Пакет 26
Вектор 43	Параграф 20
Гнучка виробнича система 73	Перетворення 27
Графік 45	Помилки 61
Границя 32	Похідна 32
Група 20	Призначення 35
Диференціальні рівняння 37	Похідна 32
Добуток членів послідовності 32	Програмування 50
Довідка 29	Процедура 50, 57
Екстремум 39	Рівняння 34
Еластичність 82	Регресійний аналіз 40
Електронні таблиці 48	Робочий лист 19
Зонд 63	Розповсюдження тепла 63
Залежність 88	Розв'язання нерівностей 36
Ідентифікатор 13	Розв'язання рівнянь 34, 37,
Інертна форма 21	Сеанс 22
Інтеграл 33	Секція 24
Інтерактивний режим 20	Середня температура 65
Інтерфейс 20	Символьні дані 28
Коментар 25	Система 71
Константи 26	Списки 28
Конструювання функцій 36	Статистичні розрахунки 40
Коректна задача 9	Строкові дані 25
Лінійна технологія 93	Стійке рішення 10
Лінійні рівняння 34	Сума ряду 32
Матриця 43	Температура 70
Модель 2, 9	Транспортний робот
Моделювання 4	Таблиці 28, 48
Можливості Maple 19	Файл 19
Мнима одиниця 26	Фіксатор 21
Множина 28	Формат файлів 19
Обчислювальний експеримент 17	
Фірма 78	
Обслуговування 72	Функція 51
Операнд 25	Цикл 53
Оператор 25	Числа 28
Операції 23	Ядро системи 26

Зміст

Вступ	3
1. Теоретичні аспекти моделювання	4
1.1. Моделювання як метод наукового пізнання	4
1.2. Технологія побудови комп'ютерних моделей	6
1.3. Моделювання у техніці	16
1.4. Інструментарії комп'ютерного моделювання	17
Контрольні запитання і вправи1	18
2. Математична системи Maple	19
2.1. Загальна характеристика математичної системи Maple	19
2.2. Основні засоби керування Maple	22
2.3. Основи роботи з Maple	25
2.4. Робота з довідковою системою Maple	29
2.5. Основні оператори, команди і функції ядра Maple	31
2.6. Побудова графіків у Maple	45
2.7. Робота з електронними таблицями	48
2.8. Засоби програмування у Maple	51
Контрольні запитання і вправи	62
3. Моделювання техніко-енергетичних систем	63
3.1. Моделювання процесу розповсюдження тепла у стержні	63
3.2. Розрахунок середньої температури вздовж дротового зонда	65
3.3. Розповсюдження тепла через багат шарову стінку	68
3.4. Дослідження технічної системи з обслуговуванням	71
3.5. Моделювання гнучкої виробничої системи	73
Контрольні запитання і вправи	77
4. Моделювання економічних процесів	78
4.1. Моделювання простішої виробничої діяльності фірми	78
4.2. Моделювання виробничої діяльності багато профільної фірми	79
4.3. Моделювання виробничої діяльності фірми в умовах монополістичної змови	81
4.4. Дослідження еластичності попиту на товар, залежною від часу	83
4.5. Дослідження рекламної діяльності фірми	86
4.6. Комп'ютерна модель міжгалузевого балансу (модель Леонтьєва В.В.)	88
Контрольні запитання і вправи	99
Бібліографічний опис	100
Предметний покажчик	101

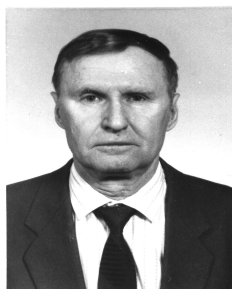
Учбове видання

Математичні методи та моделі в розрахунках на ЕОМ.
Система Maple та її використання для моделювання
техніко-економічних систем

Навчальний посібник

Для студентів спеціальності “Теплові електричні станції”

В авторській редакції



Автор: Литвинов Анатолій Леонідович,
доктор технічних наук, професор.
Українська інженерно-педагогічна
академія.
Спеціаліст в галузях моделювання,
комп'ютерної техніки та інформаційних
технологій. Автор понад 100 наукових
робіт, винаходів, учбових посібників.

Формат 60x84/16. Умовн. друк. арк. 6. Тираж 100 прим.

© Українська інженерно-педагогічна академія.

61003, м. Харків, вул.. Університетська, 16.

