Міністерство освіти і науки України
Прикарпатський національний університет
імені Василя Стефаника

О. В. Махней, Т. П. Гой

МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ АВТОМАТИЗАЦІЇ ПРИКЛАДНИХ ДОСЛІДЖЕНЬ

Навчальний посібник
для студентів галузей знань
«Фізико-математичні науки»,
«Системні науки та кібернетика»
вищих навчальних закладів

Івано-Франківськ
2013

УДК 004.4:51
ББК 32.973.26-018.2
М36

Рекомендовано Вченою радою факультету математики та інформатики Прикарпатського національного університету імені Василя Стефаника як навчальний посібник для студентів галузей знань «Фізико-математичні науки», «Системні науки та кібернетика» (протокол № 6 від 21 лютого 2013 р.).

Рецензенти:

Лютак І. З., доктор технічних наук, професор (Івано-Франківський національний технічний університет нафти і газу);

П'янило Я. Д., доктор фізико-математичних наук, старший науковий співробітник (Інститут прикладних проблем механіки і математики ім. Я. С. Підстригача);

Салій Я. П., доктор фізико-математичних наук, професор (Прикарпатський національний університет імені Василя Стефаника).

М36 Махней О. В. Математичне забезпечення автоматизації прикладних досліджень : навчальний посібник / О. В. Махней, Т. П. Гой. – Івано-Франківськ : Сімик, 2013. – 304 с.

У книзі викладено навчальний курс з вивчення системи комп'ютерної математики Maple. Описано основи роботи з програмою, використання графічних можливостей, застосування комп'ютерної математики до розв'язування різних видів рівнянь та їх систем, задач математичного аналізу, лінійної алгебри, аналітичної геометрії, теорії ймовірностей і математичної статистики тощо. Детальний предметний покажчик дозволяє використовувати посібник як довідник.

Для студентів галузей знань «Фізико-математичні науки», «Системні науки та кібернетика». Буде корисним для студентів природничих та інженерно-технічних напрямів підготовки, аспірантів, науково-технічних працівників.

УДК 004.4:51
ББК 32.973.26-018.2

ISBN 978-966-8067-97-6

Зміст

Передмова

Комп'ютерна математика – новий і перспективний напрям, який виник на стику математики і інформатики. Під комп'ютерною математикою розуміють сукупність теоретичних, алгоритмічних, апаратних і програмних засобів, призначених для ефективного розв'язування на комп'ютерах математичних задач майже необмеженої складності з високим рівнем візуалізації всіх етапів обчислень.

Важливим аспектом комп'ютерної математики є математичне забезпечення автоматизації прикладних досліджень за допомогою таких новітніх математичних систем, як-от MathCAD, Mathematica, Maple, MATLAB, Scilab, Octave, Maxima, Sage, Derive та ін. Використання цих математичних систем позбавляє від рутинної роботи обчислювального характеру й аналітичних перетворень при розв'язуванні різноманітних математичних задач. Без застосування систем комп'ютерної математики вже неможливо уявити як числові, так і аналітичні розрахунки в науково-дослідницьких роботах. Однак завдяки потужній графіці, засобам візуального програмування й використанню техніки мультимедіа роль систем комп'ютерної математики виходить далеко за межі автоматизації тільки математичних розрахунків і прикладних досліджень. Можливість підготовки в системах комп'ютерної математики документів і електронних книг, забезпечених наочними графічними ілюстраціями й ілюстративними прикладами, робить ці системи незамінними в освіті. Вони стали зручними і надійними помічниками для досвідчених користувачів і засобом надання математичних знань для початківців.

Одним з найбільш популярних і потужних є пакет аналітичних обчислень і числових розрахунків Maple. Проект створення системи Maple з'явився в університеті м. Ватерлоо (Канада) у 1980 році. У 1988 році було створено об'єднання Waterloo Maple Inc., яке до теперішнього часу успішно керує координа-

цією зусиль розробників з багатьох країн світу з розвитку та підтримки Maple. Зараз це об'єднання більше відоме під ім'ям свого підрозділу і торгової марки Maplesoft. Майже щороку виходить нова версія Maple (остання версія Maple 16 вийшла 28 березня 2012 р.).

У даний час система Maple стала пріоритетним засобом числових і символьних обчислень у багатьох провідних університетах і науково-дослідних установах світу.

У цій книзі викладено навчальний курс з вивчення системи комп'ютерної математики Maple. Описано основи роботи з програмою, використання графічних можливостей, застосування до розв'язування різних видів рівнянь та їх систем, задач математичного аналізу, лінійної алгебри, аналітичної геометрії, теорії ймовірностей і математичної статистики тощо. Всі розглянуті теми проілюстровано прикладами. Вони дадуть читачу можливість краще засвоїти широкий спектр засобів Maple для розв'язування багатьох прикладних задач. Детальний предметний покажчик дозволяє використовувати посібник як довідник.

Студенти галузей знань, яким адресований цей посібник, зможуть використовувати його з таких нормативних та вибіркових навчальних дисциплін, як «Програмування», «Математичне забезпечення систем автоматизації прикладних досліджень», «Математичне програмне забезпечення», «Числові методи», «Практикум з розв'язування задач на ЕОМ», «Дослідження операцій» та інших.

Для роботи з більшою частиною розглянутих тем достатньо версії Maple 7. Розділ 13 і § 17.3 вимагають версії, не нижчої за Maple 9.5.

Матеріал посібника підібрано так, щоб створити основу для подальшого самостійного опанування системи Maple у потрібному користувачеві напрямку. Автори не мали на меті описати всі команди та можливості Maple. Для глибшого ознайомлення з можливостями Maple радимо ознайомити-

ся з літературою, яку наведено у кінці посібника, де деякі питання розкриті по-іншому або більш повно. Багато цікавої інформації можна отримати із сайту компанії-розробника http://www.maplesoft.com, де є, зокрема, ілюстративні навчальні матеріали, а також матеріали з розв'язування практичних задач.

Всі слова та числа, виділені червоним кольором, працюють як гіперпосилання. При клацанні на них лівою кнопкою миші відбувається перехід на ту частину сторінки, де розміщена відповідна інформація. Нагадуємо, що для повернення до попередньої сторінки після користування гіперпосиланням слід застосовувати стандартну для програми, якою Ви користуєтесь, комбінацію клавіш. У більшості програм такою комбінацією клавіш є Alt-← (стрілка вліво). Весь зміст продубльовано у закладках, що дозволяє швидко переходити до потрібного розділу чи параграфа. Дерево закладок розкривається при клацанні мишею на значку +.

Розділ 1. Інтерфейс користувача Maple

§ 1.1. Основи інтерфейсу

Робота з Maple здійснюється у вигляді інтерактивного сеансу (session): користувач вводить математичні вирази і команди (під ними зараз розуміємо певні інструкції) і натисненням клавіші `Enter` передає їх на виконання програмі. Усі введені команди і результати їх виконання, наприклад, вирази, формули, графіки, формують вміст робочого листа (worksheet) – основного документа Maple. Робочий лист можна зберегти, відкрити, знову виконати команди, що на ньому містяться, чи внести необхідні зміни.

Починаючи з версії Maple 9.5, програма має два графічні інтерфейси: класичний робочий лист (Classic Worksheet), майже незмінний для всіх версій Maple під Windows, і стандартний робочий лист (Standard Worksheet). Останній надає більше можливостей для оформлення, але вимагає додаткових системних ресурсів і тому може працювати повільніше. Математичні можливості обох робочих листів однакові.

Робочий лист складається з областей введення й областей виведення. У перших областях вводяться команди, у других – можемо спостерігати результати їх виконання чи повідомлення про помилки. Вміст областей введення і виведення утворює групу обчислень, яку на робочому листі легко знайти за характерною квадратною дужкою зліва (див. рис. 1 і 2).

За замовчуванням команди в області введення класичного робочого листа відображаються червоним кольором, а стандартного робочого листа – чорним. Формули і повідомлення про нештатні ситуації в області виведення наводяться синім кольором, повідомлення про помилки – рожевим.

На рис. 1 наведено вигляд вікна програми з класичним робочим листом, на якому знайдено первісну функції $y = x \sin x^2$ і побудовано графік функції $y = x^2 - 5\cos 2x$ на проміжку $[-9, 9]$. На рис. 2 – вигляд вікна програми зі стандартним ро-

бочим листом з тими самими діями.

Команди вводяться в області введення після символу-запрошення > у формі синтаксису мови Maple, а відображатись можуть у цій самій формі або у вигляді звичайного математичного запису. За замовчуванням у класичному робочому листі використовується перший варіант запису, у стандартному – другий.

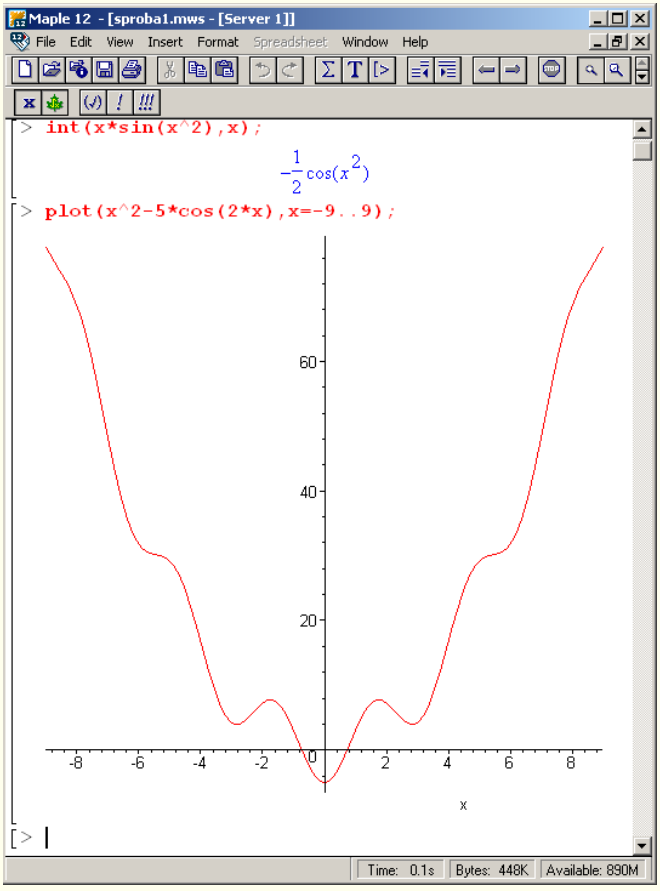


Рис. 1. Класичний робочий лист

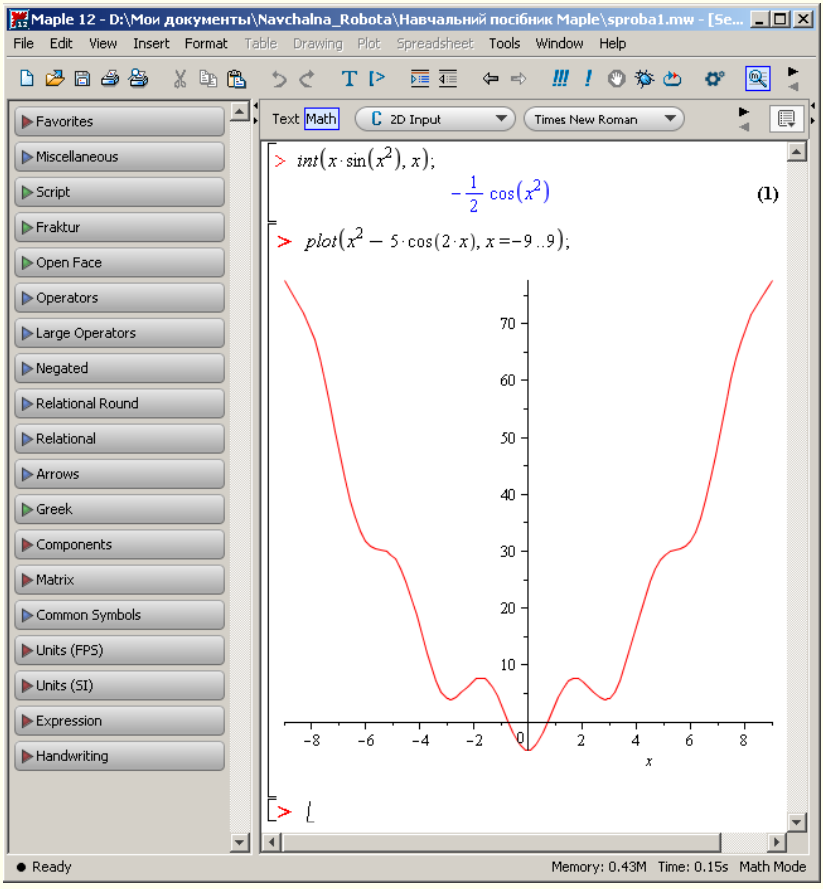


Рис. 2. Стандартний робочий лист

Працюючи з класичним робочим листом Maple, переключатись між режимами введення команд у формі синтаксису Maple і математичного запису можна за допомогою кнопки [x] на контекстній панелі інструментів. Ця кнопка дозволяє в будь-який момент замінити команди у формі синтаксису Maple на їх математичний запис і навпаки, попередньо встановивши на них курсор. При роботі зі стандартним ро-

бочим листом аналогічну функцію виконують команди меню `Format`▶`Convert To`▶`1-D Math Input` для переходу до синтаксису Maple і `Format`▶`Convert To`▶`2-D Math Input` для переходу до звичайного математичного запису.

Кожна команда Maple в області введення повинна закінчуватись крапкою з комою (`;`) або двокрапкою (`:`). Відсутність будь-якого з цих символів у кінці команди в класичному робочому листі викликає повідомлення про помилку, а у стандартному – розглядається як наявність крапки з комою. Якщо команда закінчується крапкою з комою, то результат виконання цієї команди відображатиметься в області виведення, двокрапка в кінці команди використовується для проміжних обчислень, результати яких наводити не потрібно. В одному рядку можна вводити кілька команд, відокремлених крапкою з комою чи двокрапкою. Якщо команда не поміщається у рядку, то вона буде автоматично перенесена у наступний рядок.

Іноді зручно задати кілька команд по одній у рядку, але так, щоб при натисненні клавіші `Enter` вони були одночасно передані на виконання. Це здійснюється натисненням комбінації клавіш `Shift`+`Enter` після завершення введення рядка. У цьому випадку введений рядок не опрацьовується, а система очікує введення інформації користувачем, перемістивши курсор у наступний рядок.

За замовчуванням результати виконання команди відображуються у вигляді звичайного математичного запису. Отриману формулу або її частину можна скопіювати в область введення (вона відобразиться у формі синтаксису мови Maple, якщо у цій формі відображається вміст області введення).

Контекстне меню для області виведення (відкривається правою кнопкою миші на області виведення) містить операції, які можна виконати над її вмістом. Крім звичайних операцій, пов'язаних з копіюванням і вставкою, це вікно містить перелік операцій, які можна застосувати до наявного виразу, наприклад, `Integrate` (зінтегрувати), `Differentiate` (здиференці-

ювати), `Factor` (розкласти на множники), `Plot` (побудувати графік), `Solve` (розв'язати рівняння, нерівність чи їх систему), `Limit` (знайти границю). Для побудованого графіка за допомогою контекстного меню можна змінити деякі параметри відображення графіка. Контекстне меню існує також для області введення.

Результати роботи можна зберегти у файлах різних форматів. Поточний документ записується у файл з розширенням *.mw (Maple Worksheet) або *.mws (Maple Classic Worksheet), однак його можна також експортувати у текстовий файл (*.txt), веб-сторінку (*.html), файл популярного пакету для високоякісного оформлення математичних документів LaTeX (*.tex) та інші формати.

§ 1.2. Коментарі, меню, палітри

Крім груп обчислень, на робочому листі можуть бути коментарі. Для створення текстового коментаря в робочому листі треба виконати команду меню `Insert`▶`Text` або натиснути кнопку T на основній панелі інструментів. Для початку нової групи обчислень треба виконати команду меню `Insert`▶`Maple Input` або натиснути кнопку [> з основної панелі інструментів. Коментарем є також частина рядка в області введення, яка починається з символу #. Весь рядок після цього символу не виконується. У коментарях можна використовувати кириличні букви, різні способи форматування тексту, зокрема різноманітні шрифти, стилі шрифтів, кольори тощо.

На рис. 3 наведено приклад класичного робочого листа з коментарями, додатково оформленими спеціальними шрифтами. На рис. 4 цей самий приклад відображено у стандартному робочому листі.

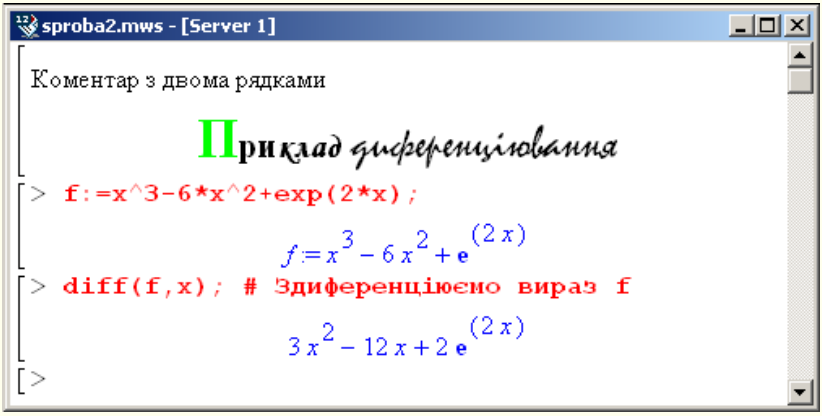


Рис. 3. Класичний робочий лист з коментарями

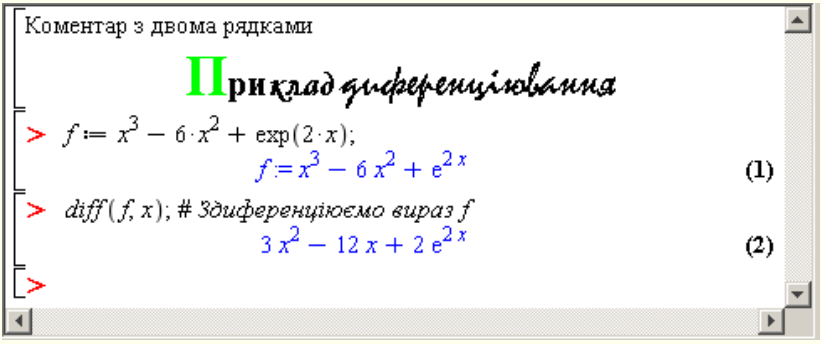


Рис. 4. Стандартний робочий лист з коментарями

Починаючи з версії Maple 10, здійснюється автоматична нумерація формул і виразів, де кожен рядок нумерується числом у круглих дужках (див. рис. 4).

Звертаємо увагу на те, що під час сеансу роботи з робочим листом у пам'яті комп'ютера зберігаються всі результати виконання команд, навіть якщо самі команди або результати їх роботи після цього були видалені (звичайно, якщо не застосовувати спеціальних заходів для знищення цієї інформації).

Водночас, після відкриття робочого листа у пам'яті комп'ютера немає результатів обчислень, хоча вони можуть бути на робочому листі у відповідних областях виведення.

Для видалення результатів обчислень з пам'яті комп'ютера (у тому числі всіх присвоювань значень змінним) у черговій області введення робочого листа можна використати команду `restart` або натиснути кнопку  на основній панелі інструментів. Для повторного виконання команд області введення потрібно покласти курсор у будь-яку точку цієї області введення і натиснути клавішу `Enter` чи кнопку  з основної панелі інструментів. Для перерахунку результатів всього робочого листа використовують кнопку  з основної панелі інструментів. Якщо обчислення тривають надто довго, то їх можна зупинити, натиснувши у класичному робочому листі кнопку  , а у стандартному – кнопку  .

Головне меню програми у класичному робочому листі зазвичай містить пункти: `File` (файл), `Edit` (редагування), `View` (вигляд), `Insert` (вставка), `Format` (формат), `Spreadsheet` (електронна таблиця) – недоступний за відсутності активної електронної таблиці, `Window` (вікно), `Help` (довідка). Якщо виділений графік побудованої функції, то у меню замість `Spreadsheet` з'являються пункти `Style` (стиль), `Legend` (легенда), `Axes` (осі), `Projection` (проекція), `Animation` (анімація), `Export` (експорт).

Головне меню програми у стандартному робочому листі містить пункти: `File`, `Edit`, `View`, `Insert`, `Format`, `Table` (таблиця), `Drawing` (рисування), `Plot` (графік), `Spreadsheet`, `Tools` (інструменти), `Window`, `Help`. Пункти меню `Table`, `Drawing`, `Plot`, `Spreadsheet` будуть недоступними без активних об'єктів, для роботи з якими вони призначені.

Робота з багатьма пунктами меню здійснюється так само, як у більшості програм зі стандартним інтерфейсом. Використання інших пунктів меню має бути зрозумілим для читача, який на достатньому рівні володіє англійською мовою.

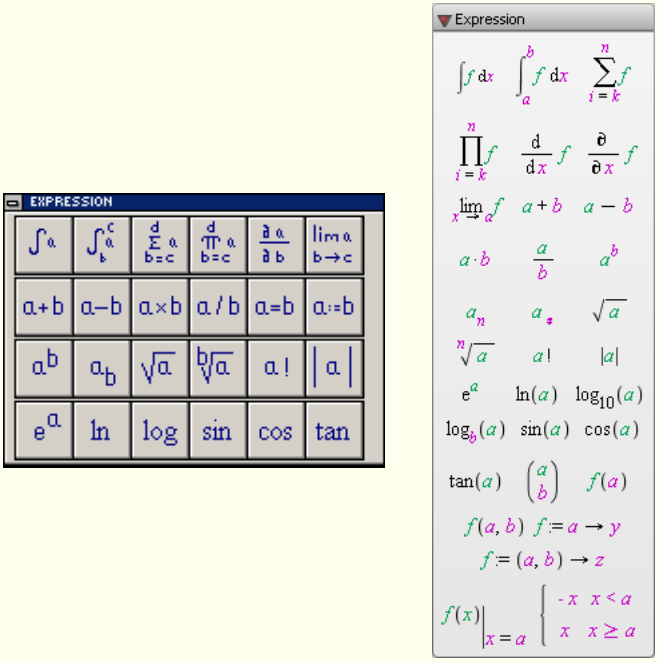


Рис. 5. Палітра `Expression` у класичному (зліва) і стандартному (справа) робочому листі

Команда меню `Insert`▶`Execution Group`▶`Before Cursor` (`Insert`▶`Execution Group`▶`After Cursor`) дозволяє вставити нову групу обчислень вище (нижче) курсора. Ці ж операції можна виконати, використавши комбінації клавіш `Ctrl`+`K` і `Ctrl`+`J` відповідно.

Для набору формул можна використовувати палітри для вставки символів, виразів, матриць тощо. У класичному робочому листі для виведення на екран палітр використовують команди меню: `View`▶`Palettes`▶`Symbol Palette` (для введення грецьких букв і деяких математичних сталих), `View`▶`Palettes`▶`Expression Palette` (для шаблонів математичних операторів і операцій), `View`▶`Palettes`▶`Matrix Palette` (для шаблонів матриць

різних розмірів), `View▶Palettes▶Vector Palette` (для шаблонів векторів різних розмірів і типів).

У стандартному робочому листі назви палітр знаходяться ліворуч від робочого листа (рис. 2). Для відкриття палітри треба клацнути мишею на трикутнику ▶ зліва від її назви. На практиці, як правило, швидше набрати назви потрібних команд чи імен, ніж шукати і вибирати відповідні об'єкти в палітрах. На рис. 5 наведено палітру `Expression` у класичному та стандартному робочих листах.

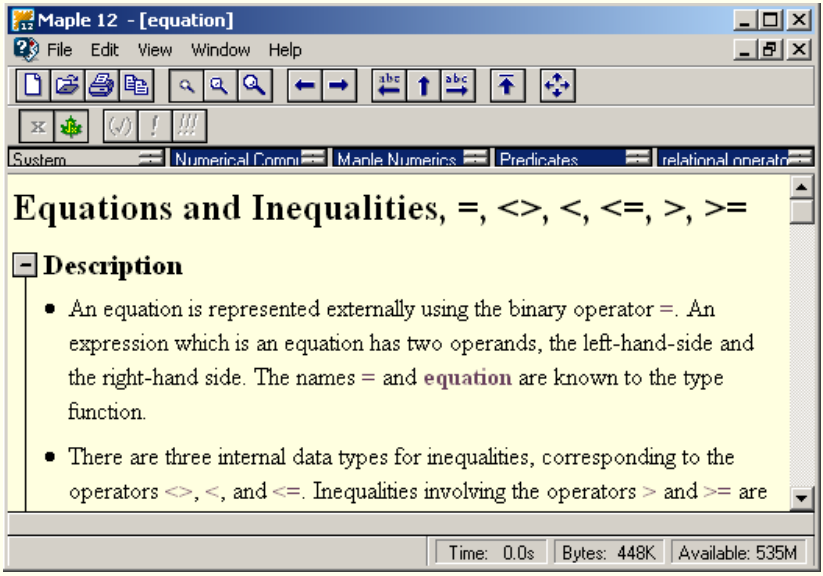


Рис. 6. Вікно довідки у класичному робочому листі

§ 1.3. Довідкова система

Maple має детальну довідкову систему з поясненнями та багатьма прикладами. Документ довідки – це робочий лист з гіперпосиланнями. Команда меню `Help`▶`Topic Search` у класичному робочому листі й комбінація клавіш `Ctrl`+`F1` у стандартному робочому листі відкриває вікно довідки, яке дозволяє знайти потрібний розділ довідки. Якщо на робочому листі навести курсор на деяке слово і натиснути клавішу `F1` (у класичному робочому листі) або `F2` (у стандартному робочому листі), то відкриється вікно довідки з відповідною темою (за умови, що інформація про відповідний термін є у довідці).

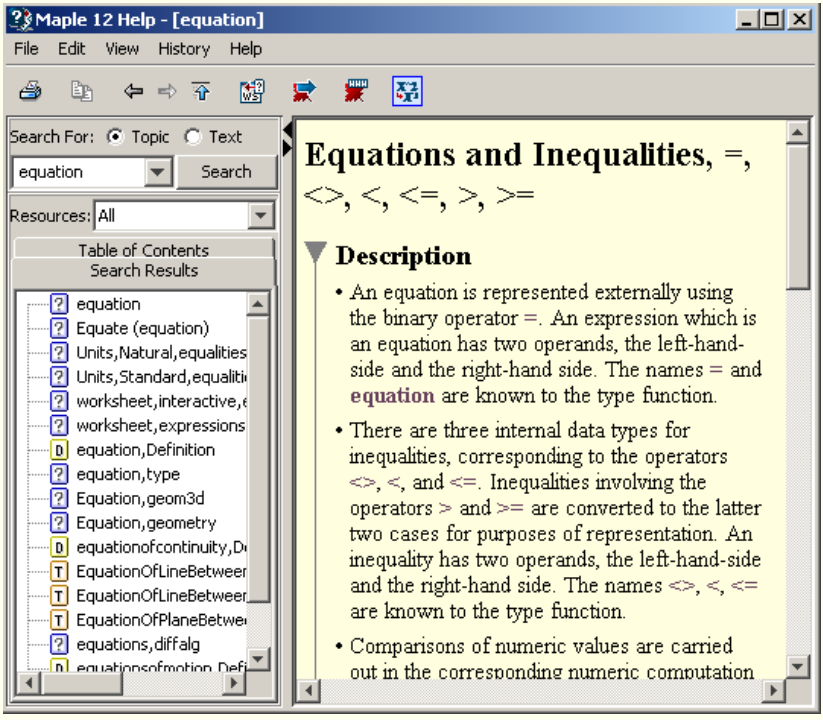


Рис. 7. Вікно довідки у стандартному робочому листі

Ще один спосіб отримати довідкову інформацію – це виконати в області введення на робочому листі команду `?`термін або `help(`термін`)` (такі команди можна не завершувати двокрапкою чи крапкою з комою). Наприклад, довідку за словом `equation` (рівняння) можна отримати, натиснувши клавішу `F1` чи `F2` на цьому слові або виконавши в області введення `?equation` чи `help(equation)`. Вікно довідки у класичному і стандартному робочих листах має різний вигляд (рис. 6, 7 відповідно).

Питання до розділу 1

1. Назвіть загальні особливості інтерфейсу Maple. Які основні відмінності між класичним і стандартним робочими листами Maple?

2. Назвіть основні принципи введення та виведення інформації в Maple.

3. Як змінити спосіб відображення області введення?

4. Як у робочий лист у Maple вставити текстовий коментар?

5. Якими символами має закінчуватись кожна інструкція в області введення? Як виконати команду без виведення результатів на екран?

6. Як між двома групами обчислень вставити ще одну?

7. Як вилучити з пам'яті комп'ютера всі результати обчислень, не закриваючи робочий лист?

8. Як зупинити обчислення, що тривають занадто довго?

9. Як у Maple реалізується спрощений механізм введення символів і математичних структур?

10. Наведіть основні способи використання довідкової системи Maple.

11. У які формати можна експортувати робочий лист Maple?

Розділ 2. Типи даних, змінні та вирази в Maple

§ 2.1. Числа та дії з ними

Числа можуть бути цілими, звичайними дробами, радикалами (алгебричними коренями з чисел), комплексними і числами з плаваючою крапкою. Перші три типи чисел дозволяють виконувати точні обчислення (без заокруглень), що відрізняє систему аналітичних обчислень Maple від систем числового розв'язання задач MathCad, MATLAB та інших. Числа з плаваючою крапкою є наближеними і кількість значущих цифр в них обмежена. Комплексні числа можуть бути як точними (коли дійсна й уявна частини подаються точними числами), так і наближеними (коли дійсна або уявна частина задана у вигляді числа з плаваючою крапкою).

Цілі числа задаються у вигляді послідовності цифр від 0 до 9 практично довільної довжини (кількість цифр обмежена числом 2^{28} чи обсягом наявної пам'яті комп'ютера). Перед від'ємними числами ставиться знак мінус (-). В обчисленнях можна використовувати додавання (+), віднімання (-), множення (*), ділення (/), піднесення до степеня (^ або **) і обчислення факторіала (!). Порядок виконання цих операцій є стандартним: спочатку виконується піднесення до степеня, потім – множення і ділення, далі – додавання і віднімання. Для зміни порядку виконання операцій використовують круглі дужки. Наприклад:

```
> 11^12-12**11;
```

$$2395420006033$$

```
> (19+4*11)/3+4;
```

$$25$$

```
> 60!;
```

832098711274139014427634118322336438075417260636124595\
2\
449277696409600000000000000

Знак \ (зворотний слеш) в останньому прикладі використовується для перенесення довгого числа на наступний рядок.

Надалі у прикладах будемо наводити область введення у формі синтаксису Maple. Саме так потрібно набирати формули, проте відображатись вміст області введення може як у цьому вигляді, так і в стандартному математичному записі.

Звичайні дроби задають за допомогою ділення двох цілих чисел, причому скорочення дробів виконується автоматично. Із звичайними дробами можна виконувати всі арифметичні операції та піднесення до степеня:

```
> 28/12*6;
```

$$14$$

```
> 2/3-1/6+5/9-7/10;
```

$$\frac{16}{45}$$

```
> (2+6/3)^2;
```

$$16$$

Для перетворення звичайного дробу в десятковий можна використовувати команду[1] `evalf`, яка наближує звичайний дріб десятковим, використовуючи десять значущих цифр. Якщо цієї точності недостатньо, то її можна задати другим параметром вказаної команди. Розглянемо приклади:

```
> evalf(13/19);
```

$$0.6842105263$$

```
> evalf(13/19,25);
```

$$0.6842105263157894736842105$$

Радикали задаються як результат піднесення до дробового степеня цілих і дробових чисел або обчислення з них квадратного кореня функцією `sqrt` чи кореня n-го степеня функцією

[1] Детальніше про команди Maple йтиметься у § 2.2.

`surd`(число, n). Якщо основою або показником степеня є дріб, то його потрібно записувати у круглих дужках. Для радикалів також здійснюються можливі спрощення, пов'язані з винесенням з-під знаку радикала максимально можливої величини. Розглянемо приклади:

```
> sqrt(80/9);
```

$$\frac{4\sqrt{5}}{3}$$

```
> surd(64,-6);
```

$$\frac{1}{2}$$

```
> (1/3)^(4/3);
```

$$\frac{3\left(\frac{2}{3}\right)}{9}$$

У останньому прикладі показник степеня $2/3$ записаний так, як він виглядає у класичному робочому листі. У стандартному робочому листі показник степеня запишеться через косу риску:

```
> (1/3)^(4/3);
```

$$\frac{1}{9}\,3^{2/3}$$

Числа з плаваючою крапкою задаються у вигляді відокремлених крапкою цілої та десяткової частин (можливо, зі знаком), наприклад, 2.5671, $-100.$, .234. Числа з плаваючою крапкою можна задавати також в експоненціальній формі, де після мантиси дійсного числа ставиться символ `e` чи `E` з цілим числом (показником степеня) після нього. Наприклад, запис 5.6789e-2 означає число 0,056789, а 32E1000 – число $3{,}2\cdot 10^{1001}$.

Якщо у виразі є хоч одне число з плаваючою крапкою, то результат теж буде таким, окрім випадку, коли у виразі є радикал (тоді радикал обчислюється точно, а коефіцієнт біля

нього може бути точним або числом з плаваючою крапкою):

```
> 4^3*0.1;
```

$$6.4$$

```
> 3e2-1/3+5/7*sqrt(5)*0.3+2^(2/3);
```

$$299.6666667 + 0.2142857143\sqrt{5} + 2^{2/3}$$

Для позначення уявної одиниці $i = \sqrt{-1}$ у записі комплексних чисел використовується константа `I`, наприклад:

```
> 3+4*I;
```

$$3 + 4\,\mathrm{I}$$

Над комплексними числами можна виконувати всі звичайні операції:

```
> (1+I)*(3-2*I);
```

$$5 + \mathrm{I}$$

```
> (1-I)/(2+I);
```

$$\frac{1}{5} - \frac{3}{5}\,\mathrm{I}$$

```
> (4+I)^2;
```

$$15 + 8\,\mathrm{I}$$

```
> sqrt(I);
```

$$\frac{1}{2}\sqrt{2} + \frac{1}{2}\,\mathrm{I}\,\sqrt{2}$$

Maple завжди намагається здійснювати обчислення з максимальною точністю. Лише якщо зробити це не вдається, то використовуються наближені методи.

§ 2.2. Константи, змінні, вирази, команди і типи

Вираз складається з чисел, констант, імен змінних і функцій, поєднаних знаками допустимих операцій, можливо, з круглими дужками.

Якщо вираз містить змінну, якій не надано жодного значення, то вона трактується як деяка невідома величина, а відповідний вираз називають *символьним*. Основна робота у Maple зазвичай пов'язана з різноманітними перетвореннями символьних виразів.

У виразах можна використовувати такі константи: `Pi` – число $\pi = 3{,}1415926\ldots$, `I` – уявна одиниця $\sqrt{-1}$, `infinity` – додатна нескінченність ∞, `gamma` – стала Ейлера ($0{,}5772156\ldots$), `true` – істина, `false` – хибність та деякі інші. Імена цих констант є зарезервованими, а їх значення не можна перевизначати.

Для основи натурального логарифма немає окремої константи, замість неї треба використовувати запис `exp(1)`. У стандартному математичному записі основа натурального логарифма виглядатиме як `e`. Використання у формулі букви `e` замість основи натурального логарифма не приведе до повідомлення про помилку, але результат буде неправильним. Буква `e` як невідома відображається курсивом: e.

Два вирази, поєднані знаком `=`, є рівнянням. Нерівність складається з двох виразів, поєднаних знаками `>`, `<`, `>=`, `<=` або `<>`.

Вирази, рівняння, нерівності та інші об'єкти можна присвоювати змінним операцією присвоювання (`:=`). Кожна *змінна* Maple має ім'я (ідентифікатор), що складається з латинських букв, цифр і символу підкреслення (`_`), але першим символом імені цифра бути не може. Великі і малі букви розрізняються. В якості імен не можна використовувати зарезервовані слова Maple (наприклад, `and`, `by`, `do`, `error`, `fi`, `while`, `next`, `in`, `use`, `proc`, `from`), а також не бажано використовувати назви команд. Обмежень на довжину імені змінної фактично немає (вона може сягати 524275 символів). Приклади імен різних змінних: `newname`, `NewName`, `Newname`, `newname1`, `newname_2`, `new_name`.

Змінна, ім'я якої збігається з ім'ям грецької букви, від-

ображається відповідною грецькою буквою, наприклад, букву α одержимо, якщо набрати `alpha`. Назви малих грецьких букв наведено у таблиці:

α – `alpha`	η – `eta`	ν – `nu`	τ – `tau`
β – `beta`	θ – `theta`	ξ – `xi`	υ – `upsilon`
γ – `gamma`	ι – `iota`	o – `omicron`	ϕ – `phi`
δ – `delta`	κ – `kappa`	π – `pi`	χ – `chi`
ε – `epsilon`	λ – `lambda`	ρ – `rho`	ψ – `psi`
ζ – `zeta`	μ – `mu`	σ – `sigma`	ω – `omega`

Великі грецькі букви можна записати, якщо набрати назву відповідної букви з великої букви, наприклад, Δ – `Delta`, Σ – `Sigma`, Ω – `Omega` і т. д. Грецькі букви можна набирати також за допомогою спеціальної палітри (§ 1.2). Приклад:

```
> h:=alpha*Omega/(Phi(tau)+beta);
```

$$h := \frac{\alpha\,\Omega}{\Phi(\tau)+\beta}$$

Змінна, якій нічого не присвоєно, трактується як невідома. Існують системні змінні, яким певні значення присвоєні відразу. Наприклад, системна змінна `Digits` визначає необхідну кількість значущих цифр при наближених обчисленнях з десятковою крапкою. За замовчуванням їй присвоєно значення 10, але його можна змінити операцією присвоювання, наприклад:

```
> evalf(Pi);
```

$$3.141592654$$

```
> Digits:=30: evalf(Pi);
```

$$3.14159265358979323846264338328$$

Більшість імен системних змінних починається з символу підкреслення (див., наприклад, стор. 123).

Символ `%` (відсоток) можна використовувати для звертання до результату виконання попередньої операції (навіть якщо вона закінчувалась символом двокрапки). У Maple є ще дві подібні операції – `%%` і `%%%`, що означають відповідно звертання до результату виконання «передпопередньої» і «передпередпопередньої» операцій. Під попередньою розуміємо операцію, виконану безпосередньо перед поточною, причому вона не обов'язково має бути записаною у групі обчислень, що безпосередньо передує поточній області введення. Приклади:

```
> 100!/99!;
```

$$100$$

```
> %^2;
```

$$10000$$

```
> %-%%;
```

$$9900$$

Основні можливості Maple реалізовані за допомогою функцій, які також називають *командами*. Кожна команда Maple (їх є декілька тисяч) має назву і кілька аргументів, що записуються у круглих дужках після назви команди через кому:

$$\texttt{command}(par1, par2, \ldots, parn)$$

де *par1*, *par2*, ..., *parn* – аргументи (їх ще називають параметрами) команди. Назви команд треба записувати, дотримуючись регістру, бо великі і малі букви у назвах команд, як і скрізь у Maple, розрізняються. Кількість аргументів, їх зміст, порядок і форма запису закладені розробниками команди. Аргументи бувають обов'язкові і необов'язкові. В якості аргументів можна використовувати вирази, які є деякими математичними функціями, а тому вживатимемо термін «команда» для уникнення плутанини між функціями як командами Maple і виразами як математичними функціями. Саме

цей термін використовується у довідковій системі Maple. Деякі команди (`evalf`, `sqrt`, `surd`) і приклади їх застосування зустрічались на стор. 24 – 26.

Деякі команди можуть мати необов'язкові аргументи, які записуються у квадратних дужках перед круглими. Результат дії команди можна присвоїти змінній, використати у виразі, він може бути аргументом іншої команди. При безпосередньому використанні команди результат її виконання просто відобразиться в області виведення (якщо команда завершується крапкою з комою).

Найважливіші команди знаходяться в стандартній бібліотеці Maple, а решта – у додаткових пакетах (про роботу з пакетами йтиметься у розділі 9).

Цілі числа мають тип `integer`, звичайні дроби – `fraction`, радикали – `radical` і `^`, числа з плаваючою крапкою – `float`, комплексні числа – `complex`, невідома – `symbol`, рівняння – тип `=`, нерівність – тип `<`, `<=` або `<>`, вираз – знак останньої операції, що виконуватиметься після розкриття дужок.

Визначити тип об'єкта *obj* можна за допомогою команди `whattype(`*obj*`)`. Перевірити, чи заданий об'єкт *obj* має певний тип *t*, можна командою `type(`*obj*, *t*`)`. Вона виводить значення `true`, якщо об'єкт *obj* має тип *t*, і значення `false` – у протилежному випадку. Приклади:

```
> whattype(x+5);
```

$$`+`$$

```
> whattype(Pi);
```

$$symbol$$

```
> type(1000000^100000, integer);
```

$$true$$

Оскільки об'єкт може належати до кількох типів одночасно (можуть бути вкладені типи), то команда `type` може давати значення `true` для кількох типів. Крім того, типів, які

можуть використовуватись у команді `type`, є набагато більше, ніж типів, що вертаються командою `whattype`. Назви всіх можливих типів, які можуть використовуватись у команді `type`, є на сторінці довідки цієї команди (`?type`).

§ 2.3. Основні математичні функції

У Maple є великий набір математичних функцій, починаючи від елементарних і закінчуючи спеціальними. У таблиці наведено команди для основних математичних функцій.

Функція	**Синтаксис команди**
$\sin x$	`sin(x)`
$\cos x$	`cos(x)`
$\mathrm{tg}\, x$	`tan(x)`
$\mathrm{ctg}\, x$	`cot(x)`
$\sec x$	`sec(x)`
$\mathrm{cosec}\, x$	`csc(x)`
$\arcsin x$	`arcsin(x)`
$\arccos x$	`arccos(x)`
$\mathrm{arctg}\, x$	`arctan(x)`
$\mathrm{arcctg}\, x$	`arccot(x)`
e^x	`exp(x)`
$\ln x$	`ln(x)` або `log(x)`
$\lg x$	`log10(x)`
$\log_a x$	`log[a](x)`
$\mathrm{sh}\, x$	`sinh(x)`
$\mathrm{ch}\, x$	`cosh(x)`
$\mathrm{th}\, x$	`tanh(x)`
$\mathrm{cth}\, x$	`coth(x)`
$\lvert x \rvert$	`abs(x)`
$\mathrm{sgn}\, x$	`signum(x)`
$\sqrt{x}$	`sqrt(x)`
$\sqrt[n]{x}$	`surd(x,n)`
$[x]$ (ціла частина)	`floor(x)`

Функція	**Синтаксис команди**
$\{x\}$ (дробова частина)	`frac(x)`
заокруглення x до найближчого цілого	`round(x)`
відкидання дробової частини числа x	`trunc(x)`
найближче ціле, більше або рівне за x	`ceil(x)`
$\min(x_1, \ldots, x_n)$	`min`$(x_1, \ldots, x_n)$
$\max(x_1, \ldots, x_n)$	`max`$(x_1, \ldots, x_n)$

Розглянемо кілька прикладів:

```
> floor(Pi); floor(-Pi);
```

$$3$$
$$-4$$

```
> frac(exp(1.)); frac(-exp(1.));
```

$$0.718281828$$
$$-0.718281828$$

```
> round(15.2); round(-15.2);
```

$$15$$
$$-15$$

```
> trunc(tan(1000)); trunc(-tan(1000));
```

$$1$$
$$-1$$

```
> ceil(sqrt(6)); ceil(-sqrt(6));
```

$$3$$
$$-2$$

Зауважимо, що значення аргументів всіх тригонометричних функцій задаються у радіанах. Усі наведені в таблиці функції мають тип `function`. Серед спеціальних математичних функцій є, зокрема, гамма- і бета-функції, функції Бесселя, еліптичні інтеграли, дельта-функція Дірака, функція Хевісайда та багато інших. Довідку про математичні функції Maple можна отримати, виконавши команду `?inifunction`.

§ 2.4. Помилки

Якщо команда введена правильно, то у області виведення матимемо результат виконання цієї команди. Якщо область виведення порожня або у ній відображається те саме чи майже те саме, що було записане у відповідній області введення, то це означає, що програма не може виконати відповідну команду. Так може бути тоді, коли, наприклад, інтеграл не знаходиться у квадратурах, рівняння не розв'язується точними методами, програма не може знайти методу розв'язування задачі, не вказані деякі параметри. Повністю повторений разом з ім'ям команди рядок введення в області виведення може свідчити про те, що ім'я команди задано неправильно, такої команди не існує або не підключений пакет, в якому вона міститься.

Якщо у записі аргументів команди допущено синтаксичну помилку, то інформацію про неї англійською мовою можна буде побачити в області виведення. Наприклад:

```
> sin(1,2);
Error, (in sin) expecting 1 argument, got 2
```

У цьому прикладі замість крапки помилково набрано кому, тому у команду, яка потребує одного аргументу, фактично підставлено два аргументи. Правильний запис:

```
> sin(1.2);
```

$$0.9320390860$$

Часто при помилці набору замість повідомлення про помилку може бути отриманий невірний результат. Наприклад:

```
> sin*(1.2);
```

$$1.2\sin$$

У цьому випадку зайвий знак множення призвів до того, що `sin` було сприйнято як ім'я змінної, якій не присвоєно жодного значення. Аналогічно, пропуск знака множення у виразі `x(y+z)`

```
> x:=3: y:=4: z:=6: x(y+z);
```

$$3$$

дає неправильну відповідь, бо `x(y+z)` трактується як функція x однієї змінної[1], яку потрібно обчислити у точці 10 ($y + z = 4 + 6$). Оскільки змінній x присвоєно значення 3, то вона є сталою, тобто дорівнює 3 у кожній точці. У складніших випадках помилки аналогічного характеру можуть бути не такими очевидними.

При наборі у стандартному робочому листі пропуск сприймається як знак множення. Більше того, цей знак множення відображається пропуском, як це прийнято в математиці, бо область введення за замовчуванням показується у стандартному математичному записі! Таку помилку помітити ще важче. Матимемо знову:

```
> sin (1.2);
```

$$1.2\sin$$

При недостатньому досвіді радимо працювати у класичному робочому листі або при роботі зі стандартним робочим листом область введення відображати у формі синтаксису Maple.

Треба бути уважним: наприклад, використання `pi` замість `Pi`, `i` замість `I` чи `e` замість `exp(1)` означає невідому (якщо їй

[1] Про створення функцій користувача йтиметься в § 3.2.

попередньо не було присвоєно певного значення) π, i чи e, а не число $\pi = 3.14159\ldots$, уявну одиницю чи основу натурального логарифма відповідно. Таку помилку можна одразу й не помітити:

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

$$\pi$$
$$3.141592654$$

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

$$\pi$$
$$\pi$$

Для уникнення грубих помилок при виконанні прикладів можна перед ними викликати команду `restart`, яка знімає присвоювання з усіх раніше використаних змінних (стор. 18).

§ 2.5. Рядки

Maple вміє працювати з *рядками* – будь-якими наборами символів у подвійних лапках. Тип рядка – `string`, а його довжина практично не обмежена. Якщо потрібно використовувати подвійні лапки в рядку, то треба набрати двоє лапок підряд `""` або `\"`. Приклади:

```
> "Приклад рядка";
```

```
"Приклад рядка"
```

```
> "Рядок \"A\""; whattype(%);
```

```
"Рядок "A""
string
```

Об'єднання рядків здійснюють за допомогою операції `||` або команди `cat`:

```
> "Перший рядок"||"Другий рядок";
```

"Перший рядокДругий рядок"

```
> cat("Перший рядок","Другий рядок");
```

"Перший рядокДругий рядок"

Команда `length` дозволяє знайти довжину рядка (кількість символів у ньому), а для виділення підрядка заданого рядка можна використовувати індекси:

```
> "S:="information";
```

$$S := \text{"information"}$$

```
> length(S);
```

$$11$$

```
> S1:=S[3..7];
```

$$S1 := \text{"forma"}$$

§ 2.6. Послідовності

Послідовність у Maple – це група виразів, відокремлених комами. Її тип – `exprseq`. Цей тип буде видавати команда `whattype`, але його не можна використовувати в команді `type`. Приклад:

```
> s1:=1,2,sqrt(5),x,2,2*x+1, "friend";
```

$$s1 := 1, 2, \sqrt{5}, x, 2, 2\,x + 1, \text{"friend"}$$

```
> whattype(s1);
```

$$exprseq$$

Послідовність зберігає порядок виразів у ній. Вона може у довільному порядку містити елементи, що повторюються.

Якщо над послідовністю виконується якась команда, то кожен елемент цієї послідовності розглядається як відповідний параметр команди. Розглянемо, наприклад, використання послідовності в якості аргументу команди `surd`:

```
> s1:=100,3; surd(s1);
```

$$s1 := 100, 3$$
$$10^{2/3}$$

Для багатьох операцій використання в якості одного з операндів послідовності приводить до формування нової послідовності, елементами якої будуть результати застосування заданої операції до відповідних елементів початкової послідовності. Наприклад:

```
> s2:=1,3,2,x; s3:=s2*3;
```

$$s2 := 1, 3, 2, x$$
$$s3 := 3, 9, 6, 3\,x$$

Можна виконувати множинні присвоювання за допомогою послідовності змінних зліва від операції присвоювання `:=` і послідовності виразів справа від неї:

```
> a,b,c:=1,2,5;
```

$$a, b, c := 1, 2, 5$$

```
> a+b+c;
```

$$8$$

Отримати значення будь-якого елемента послідовності можна за допомогою індексу (номера), який записується у квадратних дужках після імені змінної:

```
> s2:=1,3,2,x: s2[2];
```

$$3$$

Якщо використовується від'ємний індекс, то нумерація елементів послідовності здійснюється справа наліво: індекс -1

відповідає останньому елементу, індекс -2 – передостанньому і т. д.

За допомогою індексів із заданої послідовності можна виділити підпослідовність. Для цього у квадратних дужках наводять діапазон $a\,..\,b$, де a і b – натуральні числа:

```
> s:=a,b,c,d,e,f: s[4..6];
```

$$d, e, f$$

Присвоїти нове значення елементу послідовності з використанням індексної форми неможливо. Для об'єднання кількох послідовностей достатньо записати їх через кому.

Для генерування послідовності, елементи якої підпорядковуються деякій закономірності, зручно використовувати команду `seq` або операцію повторення `$`.

Команду `seq` зазвичай використовують у форматі `seq(`f`,`j`=`$m\,..\,n$`)`, де f – вираз загального члена послідовності, залежного від параметра j, m і n – числа, що визначають діапазон зміни змінної j з кроком 1. Приклади:

```
> seq(cos(Pi*j/3), j=0..8);
```

$$1, \frac{1}{2}, -\frac{1}{2}, -1, -\frac{1}{2}, \frac{1}{2}, 1, \frac{1}{2}, -\frac{1}{2}$$

```
> seq(a[k],k=1..4);
```

$$a_1, a_2, a_3, a_4$$

Операція `$` може бути унарною чи бінарною. У першому випадку вона застосовується до числового діапазону $m\,..\,n$, а в другому випадку першим операндом є вираз, що залежить від змінної, діапазон зміни якої вказаний у другому операнді. Приклади:

```
> $2..6;
```

$$2, 3, 4, 5, 6$$

```
> $1/3..7;
```

$$\frac{1}{3}, \frac{4}{3}, \frac{7}{3}, \frac{10}{3}, \frac{13}{3}, \frac{16}{3}, \frac{19}{3}$$

```
> j^2*(-1)^j$j=1..6;
```

$$-1, 4, -9, 16, -25, 36$$

```
> a[i]$i=1..5;
```

$$a_1, a_2, a_3, a_4, a_5$$

```
> x$10;
```

$$x, x, x, x, x, x, x, x, x, x$$

§ 2.7. Списки

Список – це послідовність виразів, задана у квадратних дужках. Тип списку – `list`. Робота зі списками нагадує роботу з послідовностями (§ 2.6). У списку можуть повторюватись елементи, їх порядок є істотним, звертатись до елементів списку можна за допомогою індексу. Нумерація здійснюється зліва направо, якщо номери додатні, а від'ємні індекси дозволяють задавати порядковий номер, рахуючи справа наліво. Змінити значення певного елемента списку можна присвоюванням цьому елементу нового значення за допомогою індексу. Приклади:

```
> L:=[a,b,c,d,e,f,g];
```

$$L := [a, b, c, d, e, f, g]$$

```
> L[2];
```

$$b$$

```
> L[2..5], L[-3..-2];
```

$$[b, c, d, e], [e, f]$$

```
> L[3]:=Omega; L;
```

$$L_3 := \Omega$$
$$[a, b, \Omega, d, e, f, g]$$

```
> [8, 9, -7, 2, 11, 27, 0, -9];
```

$$[8, 9, -7, 2, 11, 27, 0, -9]$$

```
> [cos(x), sin(x), tan(x), cot(x)];
```

$$[\cos(x), \sin(x), \tan(x), \cot(x)]$$

Для отримання зі списку послідовності достатньо поставити після імені списку пару квадратних дужок `[]`:

```
> L[];
```

$$a, b, \Omega, d, e, f, g$$

Об'єднують списки за допомогою переходу до послідовностей:

```
> L1:=[5,6,7]: L2:=[L[],L1[]];
```

$$L2 := [a, b, \Omega, d, e, f, g, 5, 6, 7]$$

§ 2.8. Множини

Множина – це послідовність виразів, задана у фігурних дужках. Вона має тип `set`. Множину у Maple розуміють як невпорядковану сукупність елементів. У множині кільком елементам, що повторюються, відповідає один елемент, а їх порядок є несуттєвим:

```
> {a,b,c},{a,c,b},{c,a,a,c,b,a};
```

$$\{a, b, c\}, \{a, b, c\}, \{a, b, c\}$$

Для вибору елементів множини можна використовувати індекси, однак це недоречно, бо порядок елементів множини формується програмою і залежить від версії Maple.

З'ясувати, чи деякий вираз x є елементом списку або множини s можна командою `member(`x, s`)`. Якщо $x \in s$, то значенням команди буде `true`, інакше – `false`. Операція `union` здійснює об'єднання множин, `minus` – різницю, а `intersect` – перетин. Кількість елементів у множині (списку) A можна визначити за допомогою команди `nops(`A`)`. Приклади:

```
> set1:={x,y,z}: member(X,set1);
```

$$false$$

```
> set1 intersect {u,v,y,z};
```

$$\{y, z\}$$

```
> set1 minus {x} union {2};
```

$$\{2, y, z\}$$

```
> nops(%);
```

$$3$$

Для створення порожньої множини достатньо пари фігурних дужок `{}`.

§ 2.9. Масиви

Узагальненням поняття списку є масив. Масив може мати багато розмірностей, кожну зі своїм індексом. Індексом може бути будь-яке ціле число.

Для оголошення масиву використовують команду

`array(`*indfunc*, *ranges*, *list*`)`

у правій частині операції присвоювання. Змінна в лівій частині і буде створюваним масивом типу `array`. Перший параметр *indfunc* (індексна функція) використовується рідко,

він може набувати значень: `symmetric` (симетрична матриця), `antisymmetric` (кососиметрична матриця), `sparse` (розріджений масив), `diagonal` (діагональна матриця), `identity` (одинична матриця). Параметр *ranges* – це послідовність діапазонів зміни індексів масиву. Значення елементів масиву задаються параметром *list*, який є списком. Усі параметри команди `array` є необов'язковими, але *ranges* або *list* повинні бути обов'язково. Для багатовимірних масивів елементами списку є списки. Глибина вкладеності списків дорівнює кількості розмірностей. Приклад:

```
> ar1:=array(1..2,0..2,[[2,4,5],[3,6,-1]]);
```

$$ar1 := ARRAY([1..2, 0..2], [(1,0) = 2, (1,1) = 4, (1,2) = 5, (2,0) = 3, (2,1) = 6, (2,2) = -1])$$

Якщо значення елементів масиву не задані, то за допомогою індексів елементам масиву можна присвоїти певні значення, причому після імені масиву потрібно задати список індексів, відповідних елементу, якому присвоюється значення:

```
> ar:=array(1..3,-1..2);
```

$$ar := array(1..3, -1..2, [\,])$$

```
> ar[1,-1]:=5; ar;
```

$$ar_{1,-1} := 5$$
$$ar$$

Набір в області введення імені масиву не відображає в області виведення його вміст: буде лише ім'я масиву. Для виведення всіх елементів масиву призначена команда `print`, яка в області виведення відображає вміст об'єкта, вказаного як її аргумент:

```
> print(ar);
```

$$ARRAY([1..3,-1..2],[(1,-1)=5,(1,0)=ar_{1,0},(1,1)=ar_{1,1},$$
$$(1,2)=ar_{1,2},(2,-1)=ar_{2,-1},(2,0)=ar_{2,0},(2,1)=ar_{2,1},$$
$$(2,2)=ar_{2,2},(3,-1)=ar_{3,-1},(3,0)=ar_{3,0},(3,1)=ar_{3,1},$$
$$(3,2)=ar_{3,2}])$$

Крім команди `array`, оголосити масив можна також командою

`Array(`*indfuncs*, *dims*, *init*, *opts*`)`.

Тут *indfuncs* – послідовність індексних функцій, *dims* – послідовність діапазонів зміни індексів масиву або послідовність верхніх меж цих діапазонів (в останньому випадку нижні межі вважаються рівними одиниці), *init* – задання масиву (одним зі способів є той самий, що і для команди `array`), *opts* – опції, які записуються у вигляді *keyword*=*value*, де *keyword* – ключове слово, а *value* – значення. Параметром *init* може бути список, масив, множина рівностей у формі (*індекси*)=*значення*, функція користувача для заповнення масиву (про створення функцій див. § 3.2) або число, яким заповнюються всі елементи масиву. З опціями й індексними функціями можна ознайомитись у довідці Maple. Тип даних – `Array`.

Масив `Array` відображається поелементно і без команди `print`:

```
> ar1:=Array(1..2,0..2,[[2,4,5],[3,6,-1]]): ar1;
```

$$Array([1..2,0..2],\{(1,0)=2,(1,1)=3,(1,2)=5,$$
$$(2,0)=3,(2,1)=6,(2,2)=-1\},datatype=anything,$$
$$storage=rectangular,order=Fortran\_order)$$

Опція `storage=sparse` дозволяє створювати розріджений масив – ті елементи, яким не присвоєно жодних значень, будуть нулями:

```
> ars:=Array(1..4,1..5,{(2,4)=15,(1,5)=-8.1,(4,1)=5},
  storage=sparse);
```

$$ars := \begin{bmatrix} 0 & 0 & 0 & 0 & -8.1 \\ 0 & 0 & 0 & 15 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 5 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Цю саму матрицю можна створити й інакше:

```
> ars1:=Array(1..4,1..5,0): ars1[2,4]:=15:
  ars1[1,5]:=-8.1: ars1[4,1]:=5: ars1;
```

$$\begin{bmatrix} 0 & 0 & 0 & 0 & -8.1 \\ 0 & 0 & 0 & 15 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 5 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Тут скрізь області виведення наведені для стандартного робочого листа, у класичному робочому листі вони виглядають дещо інакше.

§ 2.10. Матриці і вектори

Окремим випадком масиву є матриця – двовимірний масив. Матриця

`matrix`($m, n, list$)

є окремим випадком масиву `array`, вона має тип `matrix`. Тут m – кількість рядків, n – кількість стовпців, *list* – список списків для елементів матриці (вони вводяться по рядкам), число, яким заповнюються всі елементи матриці, або функція користувача для заповнення матриці[1]. Список *list* або числа m і n

[1]Про створення функцій користувача див. § 3.2, а приклад створення матриці за допомогою функції є на с. 170.

задавати треба обов'язково. Приклади:

```
> mm:=matrix(2,2);
```

$$mm := array(1..2, 1..2, [\,])$$

```
> mm1:=matrix([[2,3],[-1,0]]);
```

$$mm1 := \begin{bmatrix} 2 & 3 \\ -1 & 0 \end{bmatrix}$$

Звертатись до елементів матриці можна за допомогою індексів, причому першим індексом є номер рядка, а другим – номер стовпця (нумерація здійснюється від одиниці вниз по рядкам і вправо по стовпцям):

```
> mm1[2,1];
```

$$-1$$

Вектор

$$\texttt{vector}(n, list)$$

є окремим випадком матриці `matrix`, він має тип `vector`. Тут *list* – список елементів вектора, n – їх кількість. Приклад:

```
> vv:=vector([4,-7,6]);
```

$$vv := [4 \;\; -7 \;\; 6]$$

Матрицю порядку $m \times n$ можна задати також командою

$$\texttt{Matrix}(m, n, init, opts),$$

яка є окремим випадком масиву `Array` (§ 2.9). Матриця має тип `Matrix`. Параметром *init* може бути список списків для елементів матриці, масив, множина рівностей у формі (i, j)=*значення*, число (ним заповнюються всі елементи матриці) або функція користувача для заповнення матриці (про створення функцій див. § 3.2), *opts* – опції (ті самі, що й для масиву `Array`). Кожен окремий параметр є необов'язковим,

але хоч один з них мусить бути обов'язково. Приклад:

```
> mm1:=Matrix([[2,3,4],[-1,0,1]]);
```

$$mm1 := \begin{bmatrix} 2 & 3 & 4 \\ -1 & 0 & 1 \end{bmatrix}$$

Звертатись до елементів матриці можна за допомогою індексів:

```
> mm1[2,1];
```

$$-1$$

Вектор

$$\texttt{Vector[}d\texttt{](}n, init, opts\texttt{)}$$

є окремим випадком матриці `Matrix`, він має тип `Vector`. Тут d – `row` (рядок) або `column` (стовпець), n – кількість елементів вектора, *init* – список елементів, масив, множина рівностей у формі (i)=*значення*, функція користувача для заповнення вектора або число, яким заповнюються всі елементи вектора, *opts* – опції. Без параметра d буде створений вектор-стовпець. Кожен окремий параметр є необов'язковим, але хоч один з них мусить бути заданий для визначення вектора:

```
> vv:=Vector([4,-7,6]);
```

$$vv := \begin{bmatrix} 4 \\ -7 \\ 6 \end{bmatrix}$$

```
> v1:=Vector[row]([4,-7,6]);
```

$$v1 := [4 \ \ -7 \ \ 6]$$

Вектор-стовпець `Vector` можна утворити також за допомогою конструкції $\texttt{<}x_1, x_2, \ldots, x_n\texttt{>}$, наприклад:

```
> v2:=<3,-2,0>;
```

$$v2 := \begin{bmatrix} 3 \\ -2 \\ 0 \end{bmatrix}$$

Подібну конструкцію можна використовувати для набору матриць `Matrix` по стовпцям, а не по рядкам. Як розділювач стовпців використовують символ `|`. Приклад:

```
> mm2:=<2,-1|3,0>;
```

$$mm2 := \begin{bmatrix} 2 & 3 \\ -1 & 0 \end{bmatrix}$$

§ 2.11. Таблиці

Узагальненням поняття масиву як структури даних є таблиця. У ній як індекс можна використовувати не лише цілі числа, але й, наприклад, імена, дробові числа, списки. Тип таблиці – `table`. Для створення таблиці використовують команду `table(`*indfunc*, *L*`)`, параметрами якої є індексна функція *indfunc* і список чи множина *L* пар *індекс=значення*:

```
> tt:=table([one=1,two=2]); tt[two];
```

$$tt := table([two = 2, one = 1])$$

$$2$$

Елементами таблиці можуть бути не лише числа і вирази, але й дані складних типів – списки, множини, масиви, таблиці. Тому таблиці дозволяють зберігати в одній структурі дані різних типів і при цьому використовувати в якості індексів зручні і зрозумілі назви:

```
> alum:=table([gust=[2.7,kg/m^3],pyt_tepl=[920,
  Dzh/(kg*gC)],temp_pl=[658,gC]]);
```

$$alum := table\left(\left[pyt\_tepl = \left[920, \frac{Dzh}{kg\ gC}\right], temp\_pl = [658, gC], gust = \left[2.7, \frac{kg}{m^3}\right]\right]\right)$$

```
> alum[temp_pl];
```

$$[658, gC]$$

Питання до розділу 2

1. Назвіть основні типи даних, що використовуються у Maple.
2. Яку роль відіграють операції %, %%, %%%?
3. Які типи чисел є точним (наближеними)? Як знайти десяткове наближення точного числа?
4. Які обмеження накладаються на імена змінних?
5. Назвіть основні математичні константи Maple.
6. Чим відрізняються символьні обчислення від числових?
7. Опишіть синтаксис основних математичних функцій у Maple.
8. Що таке послідовність? Як створити послідовність, який її тип? Як звернутись до конкретного елемента послідовності?
9. Що таке список? Як створити список, який його тип? Як звернутись до конкретного елемента списку?
10. Що таке множина? Як створити множину, який її тип? Назвіть операції, які можна виконувати над множинами.
11. Як створити масив, вектор, матрицю і таблицю?

Вправи до розділу 2

1. Обчислити: а) $\frac{2+5^2}{6} - \sqrt{80}$; б) $\sqrt[6]{13!} + \sin\frac{\pi}{4} - \log_3 5$. Знайти десяткові наближення отриманих результатів з десятьма і вісімнадцятьма значущими цифрами.
2. Створити послідовність з сорока перших непарних натуральних чисел, розміщених у порядку зростання.
3. Створити список з двадцяти перших чисел, які є кубами натуральних чисел, кратних трьом, розміщених у порядку зростання. Замінити п'ятий елемент отриманого списку невідомою змінною t.
4. Створити множини $A = \{2, 3, x, y^3, 15.3, 2, abc, 17, x + y, 8.1\}$, $B = \{1, 3, 5, 7, 8.1, x\}$ і знайти $A \cup B$, $A \cap B$, $A\backslash B$, $B\backslash A$.
5. Створити одновимірний масив з перших п'яти простих чисел. Створити вектор з тих самих чисел.
6. Створити матрицю

$$M = \begin{pmatrix} 2 & 3 & x \\ 1.2 & -4 & 5 \\ 6 & 2 & -1 \end{pmatrix}$$

Розділ 3. Обчислення в Maple

§ 3.1. Обчислення виразів

Переважно Maple використовує алгоритм *повного обчислення* імен, який полягає у наступному. Для обчислення значення змінної найперше перевіряється, чи було їй присвоєно яке-небудь значення. У випадку присвоєння значення воно підставляється замість імені змінної та здійснюється перевірка, чи містить підставлене значення невідомі змінні. Якщо містить, то перевіряється, чи було здійснено присвоювання для цих імен і процес продовжується далі рекурсивно, поки замість імен всіх змінних не будуть підставлені присвоєні їм значення (якщо їм нічого не присвоювалось, то такі імена залишаться в остаточному результаті обчислення імені змінної як невідомі величини). Приклад:

```
> x:=y: y:=z: z:=6: x;
```

$$6$$

При обчисленні імені може виявитись необхідним виконати кілька підстановок. Кожну таку підстановку називають *рівнем обчислення* імені, причому всі вони послідовно нумеруються, починаючи з першого присвоювання значення імені.

Команда `eval` має кілька форматів. Команда `eval(`f, n`)` дозволяє обчислити ім'я чи вираз f з використанням n рівнів обчислення імені. Якщо глибину обчислення n не вказано, то використовується алгоритм повного обчислення імені. Продовжуючи попередній приклад, маємо:

```
> eval(x);
```

$$6$$

```
> eval(x,1);
```

$$y$$

```
> eval(x,2);
```

$$z$$

Правилу повного обчислення імені підпорядковуються змінні всіх типів, крім змінних, що містять вектори, масиви, таблиці і матриці. Такі змінні обчислюються до значення останнього присвоєного імені, що містить названі об'єкти. Для повного обчислення цих імен потрібно використовувати команду `eval` або `print`:

```
> x:=y: y:=matrix([[1,2],[0,-3]]): x;
```

$$y$$

```
> print(x);
```

$$\begin{bmatrix} 1 & 2 \\ 0 & -3 \end{bmatrix}$$

Щоб не обчислювати ім'я змінної повністю чи частково, можна скористатися командою `evaln(`f`)` або записати ім'я змінної в одинарних лапках (прямих апострофах):

```
> evaln(x);
```

$$x$$

```
> 'x';
```

$$x$$

Якщо змінній присвоїти її ім'я в одинарних лапках, то будуть анульовані всі попередні присвоювання значень цій змінній, а тому її знову можна буде використовувати як невідому величину:

```
> x:=1: x;
```

$$1$$

```
> x:='x'; x;
```

$$x := x$$

$$x$$

Дія одинарних лапок розповсюджується також на вирази:

```
> z:=5: 'z+3'=z+3;
```

$$z + 3 = 8$$

Команда `assigned(`f`)` перевіряє, чи було змінній f присвоєне деяке значення:

```
> assigned(z);
```

true

Обчислити значення виразу чи математичної функції в точці можна кількома способами. Перший спосіб – присвоїти невідомій деяке значення:

```
> g:=a*x^3+3*x+5;
```

$$g := a\,x^3 + 3\,x + 5$$

```
> a:=5: x:=4: g;
```

337

Недоліком цього способу є «втрата» виразу як виразу з невідомою змінною. Щоб його знову отримати, треба анулювати всі попередні присвоєння цим змінним:

```
> a:='a': x:='x': g;
```

$$a\,x^3 + 3\,x + 5$$

Інший спосіб обчислення значення виразу – це використання команди `eval` у одному з наступних форматів: `eval(`f, v`=`a`)`, `eval(`f, `[`$v1$`=`$a1$, $v2$`=`$a2$, `...])` чи `eval(`f, `{`$v1$`=`$a1$, $v2$`=`$a2$, `...})`, де рівності типу v`=`a визначають значення змінних, для яких треба обчислити вираз f. Команда `eval` не змінює значення виразу f. Приклади:

```
> g:=a*x^3+3*x+5: eval(g,x=3);
```

$$27\,a + 14$$

```
> eval(g,[a=4,x=3]);
```

$$122$$

```
> g;
```

$$a\,x^3 + 3\,x + 5$$

Скрізь, де це можливо, Maple намагається виконувати точні обчислення, використовуючи дроби і радикали. Команда `evalf(`f, n`)` відображає десяткове наближення значення виразу чи змінної f з n значущими цифрами. Якщо другий аргумент команди відсутній, то використовується 10 значущих цифр (цю кількість можна змінити за допомогою системної змінної `Digits`):

```
> evalf(exp(1));
```

$$2.718281828$$

```
> Digits:=20: evalf(exp(1));
```

$$2.7182818284590452354$$

Першим аргументом команди `evalf` може бути також команда `eval`:

```
> evalf(eval(g,[a=1/17,x=4]),30);
```

$$20.764705882352941176470588235 3$$

Команда `evalhf(`f`)` відображає десяткове наближення значення виразу чи змінної f з п'ятнадцятьма значущими цифрами. Працює значно швидше за команду `evalf`, бо використовує апаратні можливості процесора, а не програмну реалізацію обчислень.

§ 3.2. Створення функцій користувача

Ще один спосіб виконання обчислень полягає у створенні функції користувача й обчисленні її значень. Функції користувача створюються такими конструкціями:

$$name\texttt{:=}x\texttt{->}expr$$

$$name\texttt{:=(}x1, x2, \ldots, xn\texttt{)->}expr$$

Тут *x1*, *x2*, ..., *xn* – змінні, *expr* – вираз для обчислення функції. Виклик функції здійснюється у вигляді $name\texttt{(}x1, x2, \ldots, xn\texttt{)}$, тобто найбільш звичним способом, коли у дужках замість аргументів функції вказуються конкретні значення змінних. Приклади:

```
> f:=x->x*ln(x);
```

$$f := x \to x\ln(x)$$

```
> f(x), f(2);
```

$$x\ln(x), 2\ln(2)$$

```
> f1:=(x,y,z)->sqrt(x^2+y^2+z^2);
```

$$f1 := (x, y, z) \to \sqrt{x^2 + y^2 + z^2}$$

```
> f1(x,2*x,3*x); f1(3,4,0);
```

$$\sqrt{14}\sqrt{x^2}$$

$$5$$

Функції можна створювати також за допомогою команди

$$\texttt{unapply(}f, x1, x2, \ldots, xn\texttt{)}.$$

Наприклад:

```
> f:=unapply(x*ln(x),x);
```

$$f := x \to x\ln(x)$$

```
> f1:=unapply(sqrt(x^2+y^2+z^2),x,y,z);
```

$$f1 := (x, y, z) \to \sqrt{x^2 + y^2 + z^2}$$

§ 3.3. Створення кускових функцій

У Maple є можливість задання кускових функцій, тобто функцій вигляду

$$f(x)=\begin{cases} f_1(x), & x<a_1,\\ f_2(x), & a_1<x<a_2,\\ \quad\dots & \\ f_n(x), & x>a_{n-1}. \end{cases}$$

Зробити це можна за допомогою команди

`piecewise`(*cond1*, f_1, *cond2*, f_2, $\dots$, *condn*, f_n, *f_ else*),

де кожна пара *condj*, f_j визначає проміжок зміни незалежної змінної логічною умовою *condj* і значення функції f_j на цьому проміжку, а *f_ else* – це значення функції на інших проміжках. Наприклад, кускову функцію

$$f_1(x)=\begin{cases} x^2+1, & -1\leqslant x<1,\\ 2x-2, & 1\leqslant x<2,\\ 1, & x\geqslant 2 \end{cases}$$

задаємо так:

```
> f1:=piecewise(x>=-1 and x<1, x^2+1, x>=1 and x<2,
  2*x-2, 1);
```

$$f1:=\begin{cases} x^2+1 & -1\le x \textbf{ and } x<1\\ 2\,x-2 & 1\le x \textbf{ and } x<2\\ 1 & otherwise \end{cases}$$

Обчислити значення такої функції у деякій точці можна, наприклад, командою `eval`:

```
> eval(f1,x=2);
```

$$2$$

Графік функції $f_1(x)$ побудовано на с. 70.

§ 3.4. Команди для роботи з цілими і комплексними числами

Наведемо основні команди для роботи з цілими числами:

Команда	Опис
`iquo(`$n, k,$ `'`r`')`	обчислює цілу частину від ділення числа n на k, а остачу присвоює змінній r (якщо вона вказана)
`irem(`$n, k,$ `'`q`')`	знаходить остачу від ділення числа n на k, а цілу частину присвоює змінній q (якщо вона вказана)
`igcd(`$k_1, k_2, \ldots, k_n$`)`	знаходить найбільший спільний дільник цілих чисел k_1, k_2, …, k_n
`ilcm(`$k_1, k_2, \ldots, k_n$`)`	знаходить найменше спільне кратне цілих чисел k_1, k_2, …, k_n
`isprime(`n`)`	перевіряє, чи є ціле число n простим
`ithprime(`n`)`	знаходить n-те просте число, починаючи з числа 2
`ifactor(`n`)`	розкладає ціле або раціональне число n на прості множники
`length(`n`)`	обчислює кількість цифр у числі n
`factorial(`n`)`	обчислює факторіал числа n (аналог оператора `!`)

Приклади:

```
> iquo(267,5);
```

$$53$$

```
> irem(267,5);
```

$$2$$

```
> iquo(267,5,'rrr'); rrr;
```

$$53$$

$$2$$

```
> igcd(156,216,1596,4212);
```

$$12$$

```
> ilcm(-9,12);
```

$$36$$

```
> isprime(1456667);
```

$$true$$

```
> ithprime(33);
```

$$137$$

```
> ifactor(20!);
```

$$(2)^{18}\,(3)^{8}\,(5)^{4}\,(7)^{2}\,(11)\,(13)\,(17)\,(19)$$

```
> ifactor(350/16);
```

$$\frac{(5)^{2}\,(7)}{(2)^{3}}$$

```
> length(55!);
```

$$74$$

```
> [factorial(15), 15!];
```

$$[1307674368000, 1307674368000]$$

Замість команди `irem` можна також використовувати операцію `mod` для відшукання остачі від ділення цілих чисел:

```
> 267 mod 5;
```

$$2$$

Наведемо команди для роботи з комплексними числами:
`Re(`z`)` знаходить дійсну частину числа $z = \operatorname{Re} z + i \operatorname{Im} z$;
`Im(`z`)` знаходить уявну частину числа $z = \operatorname{Re} z + i \operatorname{Im} z$;
`argument(`z`)` знаходить аргумент комплексного числа z;
`conjugate(`z`)` будує комплексно спряжене число до числа z;

`csgn(`z`)` визначає знак комплексного числа z за формулою

$$\operatorname{csgn}(z) = \begin{cases} 1, & \operatorname{Re}(z) > 0 \text{ або } (\operatorname{Re}(z) = 0 \text{ і } \operatorname{Im}(z) > 0), \\ -1, & \operatorname{Re}(z) < 0 \text{ або } (\operatorname{Re}(z) = 0 \text{ і } \operatorname{Im}(z) < 0). \end{cases}$$

Крім того, для роботи з комплексними числами можна використовувати всі математичні функції (§ 2.3), зокрема, модуль комплексного числа z обчислює команда `abs(`z`)`. Приклади:

```
> Im(3+sqrt(5)*I);
```

$$\sqrt{5}$$

```
> Re((1+I)*(1-I));
```

$$2$$

```
> z:=3+sqrt(5)*I: [abs(z), argument(z)];
```

$$\left[\sqrt{14},\ \arctan\left(\frac{1}{3}\sqrt{5}\right)\right]$$

```
> [conjugate(z),csgn(z)];
```

$$[3 - \mathrm{I}\sqrt{5}, 1]$$

```
> log(-4);
```

$$2\ln(2) + \mathrm{I}\,\pi$$

Для комплексних виразів складної будови команди `Re(`z`)` і `Im(`z`)` можуть не дати потрібних результатів. У цьому випадку дійсну й уявну частину комплексного числа допоможе знайти команда перетворення комплексних виразів `evalc(`z`)`. Наприклад:

```
> z:=sin(3+I)^5;
```

$$z := \sin(3 + \mathrm{I})^5$$

```
> Re(z), Im(z);
```

$$\Re\left(\sin(3+\mathrm{I})^5\right), \Im\left(\sin(3+\mathrm{I})^5\right)$$

```
> evalc(Re(z));
```

$$\sin(3)^5\cosh(1)^5 - 10\sin(3)^3\cosh(1)^3\cos(3)^2\sinh(1)^2 + 5\sin(3)\cosh(1)\cos(3)^4\sinh(1)^4$$

```
> evalc(Im(z));
```

$$5\sin(3)^4\cosh(1)^4\cos(3)\sinh(1) - 10\sin(3)^2\cosh(1)^2\cos(3)^3\sinh(1)^3 + \cos(3)^5\sinh(1)^5$$

Питання до розділу 3

1. У чому полягає алгоритм повного обчислення імені змінної? Що таке рівні обчислення? Яка команда дозволяє керувати ними?

2. Як можна обчислити вираз з невідомими при заданих значеннях цих невідомих?

3. Як створити функцію користувача? Як обчислити значення створеної функції в точці?

4. Як створити кускову функцію?

5. Які команди призначені для роботи з цілими числами?

6. Як виділити дійсну і уявну частини комплексного числа, знайти його модуль і аргумент? Як побудувати комплексно спряжене число?

Вправи до розділу 3

1. Обчислити значення функції $f(x) = \sin 3x - 5/3 + \ln(x^2+5)$ у точках $x = 1$, $x = 2$ і $x = 3$ двома способами: за допомогою команди `eval` і з використанням самостійно створеної функції.

2. Знайти цілу частину і остачу від ділення числа 23567 на 451.

3. Розкласти на прості множники число 27824.

4. Знайти модуль, аргумент, дійсну та уявну частину комплексного числа $(3+i)^{10}$ та число, спряжене до нього.

5. Довести, що число $2^{31} - 1$ просте.

Розділ 4. Базова графіка

§ 4.1. Основи роботи з командою plot

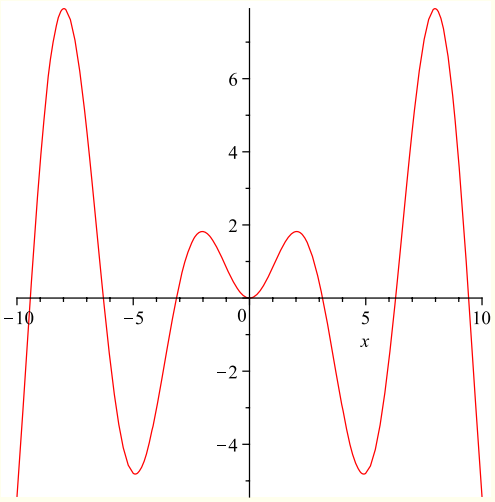
Maple володіє розвиненими графічними можливостями, починаючи від побудови простих двовимірних кривих і закінчуючи складними тривимірними поверхнями та анімацією зображень.

Команда

plot(*f*, *h*, *v*, *opts*)

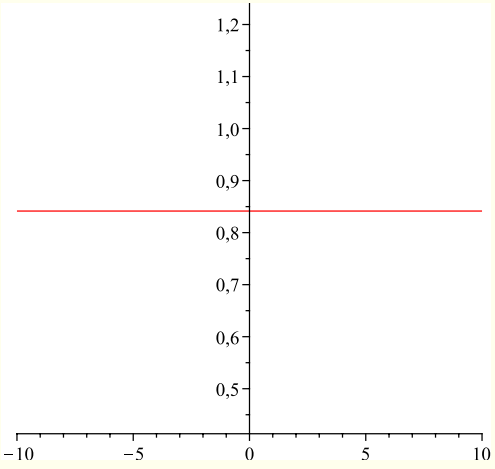
дозволяє будувати графіки функцій однієї незалежної змінної, заданих явно або у параметричній формі, а також відображати скінченну множину точок площини. Через f позначено математичну функцію, графік якої треба побудувати (нею може бути довільний вираз однієї змінної), h – ім'я незалежної змінної або рівність вигляду $x{=}a\,..\,b$, де x – незалежна змінна, a, b – мінімальне і максимальне значення x, для яких будується графік, v – діапазон значень $c\,..\,d$, який визначає частину осі ординат, що відображатиметься на графіку, *opts* – додаткові опції. Числа a, b, c і d можуть бути і невласними ($\pm$ infinity). Обов'язковим у команді plot є лише перший аргумент. За відсутності інших аргументів команда відображає частину графіка на проміжку $[-10, 10]$ осі абсцис, причому вибирається така частина осі ординат, щоб можна було побачити графік для всіх значень функції, відповідних проміжку $x \in [-10, 10]$. При цьому масштаб на осях координат може бути різним, тобто вісь ординат при потребі може розтягатись чи стискатись. Побудуємо, наприклад, графік функції $y = x \sin x$:

```
> plot(x*sin(x));
```



Важливо пам'ятати, що незалежній змінній раніше не повинно бути присвоєно значень, інакше отримаємо неправильний результат або повідомлення про помилку:

```
> x:=1: plot(x*sin(x),x);
Error, (in plot) unexpected option: 1
> x:=1: plot(x*sin(x));
```



Для побудови графіка функції у параметричній формі $x = f_1(t)$, $y = f_2(t)$, $t \in [a, b]$, використовують команду `plot` у форматі:

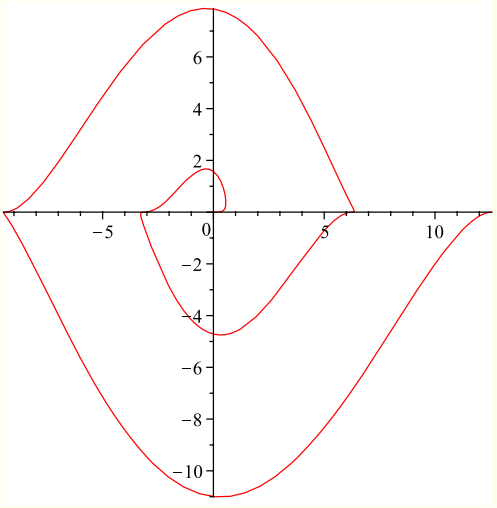
$$\texttt{plot([}f_1(t), f_2(t), t\texttt{=}a\texttt{..}b\texttt{]}, opts\texttt{)},$$

де *opts* – додаткові опції. Приклад:

```
> f1:=t*cos(t); f2:=t*sin(t)^3;
```

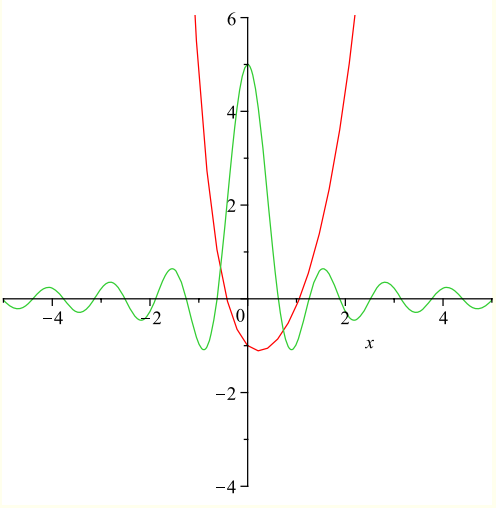
$$f1 := t\cos(t)$$
$$f2 := t\sin(t)^3$$

```
> plot([f1,f2,t=0..4*Pi]);
```



Для побудови графіків кількох функцій в одній системі координат треба задати ці функції у вигляді списку першим аргументом команди `plot`, причому програма автоматично вибере різні кольори для графіків:

```
> plot([exp(x)+exp(-2*x)-3,sin(5*x)/x],x=-5..5,-4..6);
```



§ 4.2. Опції команди `plot`

Опції задаються рівностями *keyword=value*, де *keyword* – назва опції, а *value* – її значення. За відсутності явного задання опцій використовуються їх значення за замовчуванням.

Повний список опцій можна знайти у довідковій системі Maple. Найважливіші опції команди `plot` наведемо у вигляді таблиці.

Опція	Опис
`axes`	Визначає тип осей координат. Можливі значення: `normal` – звичайні осі координат, `boxed` – графік у прямокутнику з нанесеними шкалами зліва і знизу, `framed` – осі з точкою перетину в лівому нижньому куті графіка, `none` – осі координат відсутні. Значення за замовчуванням – `normal`

Опція	Опис
color	Задає колір кривих на графіку. Основні значення: black (чорний), blue (синій), brown (коричневий), cyan (блакитний), gold (золотистий), green (зелений), gray (сірий), khaki (хакі), magenta (пурпурний), navy (темно-синій), orange (оранжевий), pink (рожевий), red (червоний), violet (фіолетовий), white (білий), yellow (жовтий). Значення за замовчуванням – red. Для побудови графіків кількох функцій в одній системі координат значенням опції є список кольорів. Про використання створених користувачем кольорів йтиметься на стор. 67
coords	Змінює тип системи координат. Можливі значення: bipolar, cardioid, cartesian, cassinian, elliptic, hyperbolic, invcassinian, invelliptic, logarithmic, logcosh, maxwell, parabolic, polar, rose, tangent. Опис цих значень можна знайти в довідковій системі Maple. Значення за замовчуванням – cartesian (декартова система координат)
discont	Можливі значення: true, false. Якщо discont=true, то здійснюється аналіз наявності точок розриву графіка функції. Значення за замовчуванням – false

Опція	Опис
`filled`	Можливі значення: `true`, `false`. Якщо `filled=true`, то область, обмежена графіком функції і віссю абсцис, зафарбовується заданим опцією `color` кольором. Значення за замовчуванням – `false`
`font`	Задає шрифт для виведення тексту на рисунку. Значення опції задається у вигляді списку [*family*, *style*, *size*]. Параметр *family* може набувати значень: `TIMES`, `COURIER`, `HELVETICA`, `SYMBOL`. Параметр *style* визначає стиль шрифту: для шрифту `TIMES` можливі значення `BOLD`, `ITALIC`, `BOLDITALIC`, для шрифтів `COURIER` і `HELVETICA` стиль може бути пропущений або мати значення `BOLD`, `OBLIQUE`, `BOLDOBLIQUE`, для шрифту `SYMBOL` стиль не задається. Параметр *size* визначає розмір шрифту у пунктах (1 пт $\approx$ 0,35 мм)
`labeldirections`	Задає напрям відображення назв осей координат у вигляді списку [x, y], елементи якого можуть набувати значень `horizontal` (горизонтально) і `vertical` (вертикально). За замовчуванням назви відображаються горизонтально
`labels`	Задає назви осей координат у вигляді списку [x, y], елементами якого є рядки. За замовчуванням відображаються назви незалежної змінної і функції

Опція	Опис
legend	Задає відображення легенди (позначення) для кількох кривих на одному графіку у вигляді списку рядків, в якому j-ий елемент відповідає j-ій кривій графіка
linestyle	Визначає тип лінії графіка. Значенням може бути ім'я або число: solid (1, суцільна лінія), dot (2, лінія з крапок), dash (3, пунктирна лінія), dashdot (4, штрихпунктирна лінія), longdash (5, пунктирна лінія з довгими штрихами), spacedash (6, пунктирна лінія з довгими пропусками), spacedot (7, лінія з розріджених крапок). Значення за замовчуванням – solid. Для побудови графіків кількох функцій в одній системі координат значенням опції є список значень
numpoints	Визначає мінімальну кількість точок для побудови графіка. За замовчуванням numpoints=50
scaling	Можливі значення: constrained (одна одиниця вимірювання на осі незалежної змінної дорівнює одній одиниці вимірювання на осі значень функції), unconstrained (одиниці вимірювання на осях координат можуть не бути однаковими, щоб область значень функції, відповідна області зміни незалежної змінної, помістилась на графіку). Значення за замовчуванням – unconstrained

Опція	Опис
`style`	Задає відображення графіка функції лініями (`line`) або точками (`point`). Значення за замовчуванням – `line`
`symbol`	Визначає символ для позначення точки графіка функції при опції `style=point`. Можливі значення: `asterisk` для ⋆, `box` для ▫, `circle` для ○, `cross` для +, `diagonalcross` для ×, `diamond` для ◇, `point` для •, `solidbox` для ■, `solidcircle` для ●, `soliddiamond` для ◆. Значення за замовчуванням – `diamond`
`symbolize`	Задає розмір символу для опції `symbol` у пунктах. Значенням є натуральне число. За замовчуванням `symbolize=10`
`thickness`	Задає товщину лінії графіка – невід'ємне ціле число. За замовчуванням `thickness=1`. Для побудови графіків кількох функцій в одній системі координат значенням опції є список чисел
`tickmarks`	Визначає кількість точок, які будуть відмічені на осях координат. Значення задається у вигляді списку `[`n, m`]`. Замість чисел n і m можуть бути також списки точок, які треба відмітити. Ці списки можуть бути як списками значень точок, так і списками рівностей, у яких після знаку «=» у подвійних лапках наводиться текст, що відображатиметься в якості мітки, наприклад, `[5,[0.5="A",1="B",2="C"]]`

Опція	Опис
`title`	Задає заголовок рисунка. Після `\n` у заголовку здійснюється перехід на новий рядок. За замовчуванням заголовок не виводиться
`view`	Визначає мінімальні та максимальні координати кривої, що відображатиметься. Значенням є список діапазонів [$xmin..xmax, ymin..ymax$]
`xtickmarks`	Визначає кількість точок, які будуть відмічені на осі абсцис. Значенням може бути ціле число або список значень точок чи список рівностей
`ytickmarks`	Визначає кількість точок, які будуть відмічені на осі ординат. Значенням може бути ціле число або список значень точок чи список рівностей

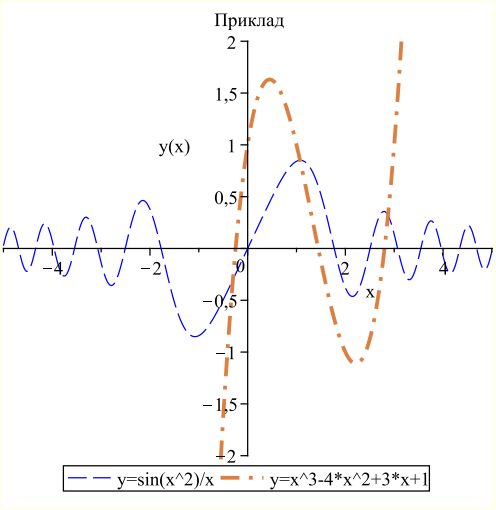
Користувач може визначити власний колір, утворений змішуванням частин червоного, зеленого і синього кольорів. Це робиться за допомогою команди `COLOR(RGB`, a, b, c`)`, де $a, b, c \in [0, 1]$ – інтенсивності червоного, зеленого і синього кольорів відповідно. Дію цієї команди можна присвоїти змінній, а потім використовувати відповідну змінну в якості кольору.

Деякі опції можна змінити після побудови графіка: якщо на ньому клацнути мишею, то зміняться основне меню і кнопки контекстної панелі інструментів, за допомогою яких можна змінити відображення графіка. Також для цього можна користуватись контекстним меню.

Приклад використання опцій:

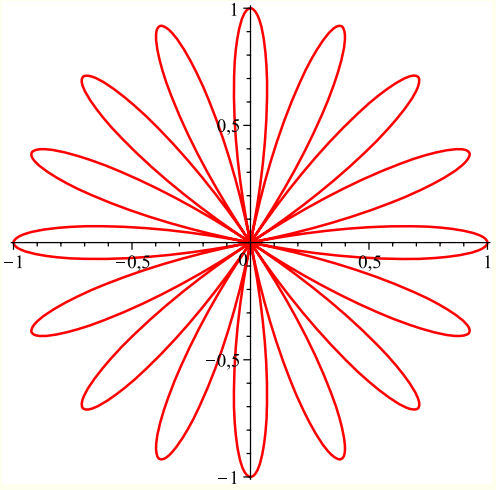
```
> plot([sin(x^2)/x,x^3-4*x^2+3*x+1],x=-5..5,-2..2,
  numpoints=500,color=[blue,COLOR(RGB,0.86,0.5,0.25)],
  linestyle=[dash,dashdot],thickness=[1,3],legend=
  ["y=sin(x^2)/x","y=x^3-4*x^2+3*x+1"],title=
```

```
"Приклад",labels=["x","y(x)"],ytickmarks=9);
```



Побудуємо за допомогою опції `coords` графік функції $r = \cos 8\varphi$, $\varphi \in [0, 2\pi]$, у полярній системі координат:

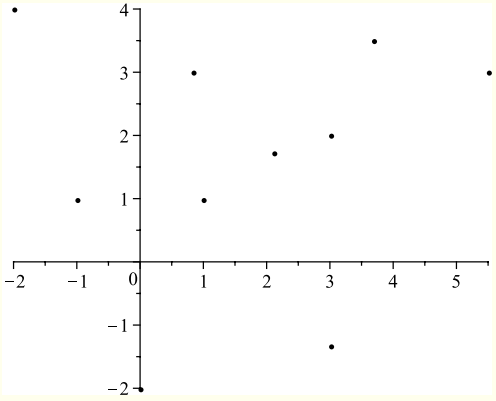
```
> plot(cos(8*phi),phi=0..2*Pi,coords=polar,
  thickness=2,numpoints=1000);
```



§ 4.3. Корисні поради для побудови графіків

Команда `plot` дозволяє будувати окремі точки, які задаються у вигляді списку, елементами якого є двоелементні списки, що визначають координати точок на площині. Замість списку списків можна також використовувати матрицю `Matrix` з двома стовпцями або два списки чи вектори через кому (перший стовпець матриці, список чи вектор містить перші координати точок, а другий – другі координати). За замовчуванням точки з'єднуються лініями у послідовності їх задання, тому графіком є ламана. Якщо для побудови графіка задати опцію `style=point`, то точки відображатимуться спеціальними символами. Тип символу залежить від значення опції `symbol` (стор. 66). Приклад:

```
> plot([[1,1],[3,2],[-2,4],[5.5,3],[0,-2],[3,-4/3],
  [-1,1],[2.1,sqrt(3)],[sin(1),3],[3+ln(2),3.5]],
  style=point,symbol=solidcircle,scaling=constrained,
  color=black);
```

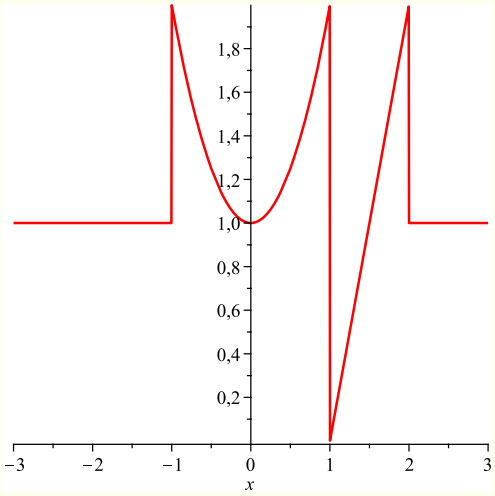


Вкажемо на деякі особливості побудови розривних функцій. Побудуємо, наприклад, графік кускової функції

$$f_1(x) = \begin{cases} x^2+1, & -1 \leqslant x < 1, \\ 2x-2, & 1 \leqslant x < 2, \\ 1, & x \geqslant 2, \end{cases}$$

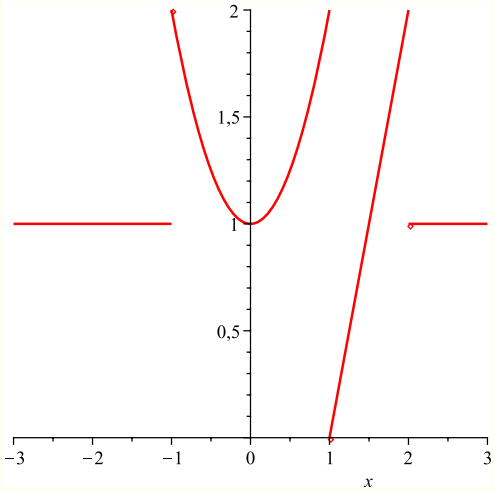
яка розглядалась у § 3.3.

```
> f1:=piecewise(x>=-1 and x<1, x^2+1, x>=1 and x<2,
  2*x-2, 1):
> plot(f1,x=-3..3,thickness=2);
```



Як видно з цього рисунка, Maple будує вертикальні лінії в точках розриву, з'єднуючи значення зліва і справа від точки розриву. Усунути недоречні вертикальні лінії можна за допомогою опції `discont=true`:

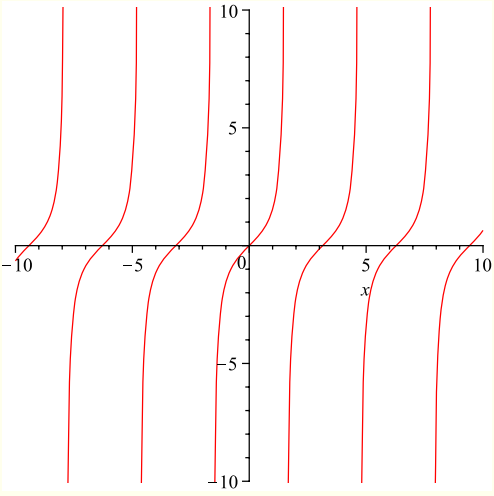
```
> plot(f1,x=-3..3,discont=true,thickness=2);
```



Точками позначаються кінці ліній, які включаються.

Будуючи графіки функцій з розривами другого роду, треба обов'язково вказувати інтервал зміни значень функції і опцію `discont=true`. Приклад:

```
> plot(tan(x),x=-10..10,-10..10,discont=true);
```

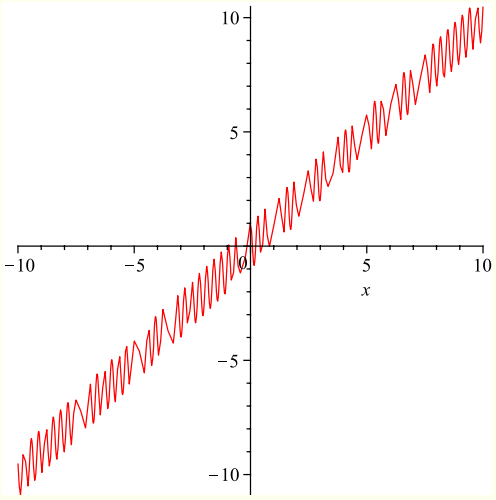


Пропонуємо читачам самостійно пересвідчитись, як виглядатиме графік функції $y = \operatorname{tg} x$, якщо у команді `plot` не вказати область значень і знехтувати опцією `discont=true`:

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

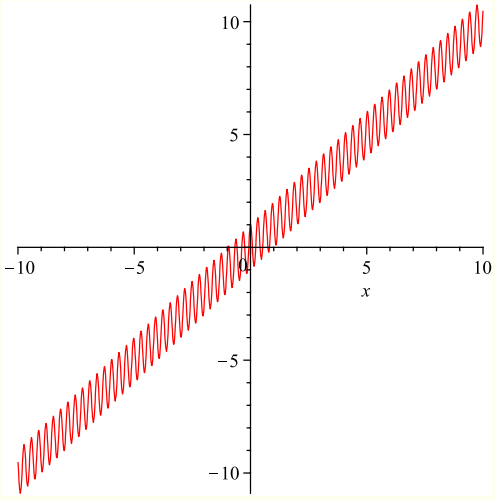
Maple будує графіки функцій, обчислюючи значення функцій у певній кількості точок. На проміжках швидкої зміни функції обчислюються значення функції у додаткових точках. Проте в деяких випадках цей алгоритм не спрацьовує. Побудуємо, наприклад, графік функції $y = x + \cos 20x$:

```
> plot(x+cos(20*x));
```



Бачимо, що на графіку багато зубців пропущені, а інші є скривленими. Для кращого відображення графіка треба збільшити кількість точок, за якими від будується. Це робиться за допомогою опції `numpoints` (стор. 65):

```
> plot(x+cos(20*x),numpoints=1000);
```



§ 4.4. Тривимірна графіка

Функцію двох змінних можна зобразити як поверхню у тривимірному просторі, при цьому дві осі простору відповідатимуть двом незалежним змінним, а третя – значенню функції. Для цього використовують команду

$$\texttt{plot3d(}f(x,y),x\texttt{=}a\,..\,b,y\texttt{=}c\,..\,d,opts\texttt{)},$$

де $f(x,y)$ – це функція, графік якої треба побудувати (нею може бути довільний вираз двох змінних), x, y – незалежні змінні, $a\,..\,b$, $c\,..\,d$ – діапазони їх зміни, *opts* – опції.

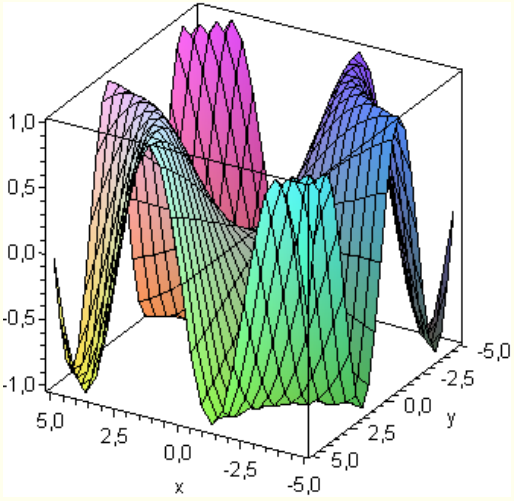
Більшість опцій є такими самими, що й для команди `plot` (§ 4.2). Деякі задаються з очевидними змінами, наприклад, `labels=[`x,y,z`]`, бо є три, а не дві осі координат, опція `axes=boxed` задає поверхню у паралелепіпеді з осями на трьох його ребрах, а не в прямокутнику, `filled=true` будує непрозору область, обмежену поверхнею і площиною для значення функції, рівного нулю, значеннями опції `coords` можуть бути назви тривимірних систем координат.

Опція `style` може набувати таких значень: `patch` (зафарбована поверхня з лініями сітки на ній), `patchnogrid` (зафарбована поверхня без ліній сітки), `line` (каркас поверхні без видалення невидимих ліній), `hidden` (каркас поверхні з видаленням невидимих ліній), `point` (лише точки на поверхні). За замовчуванням `style=patch`.

Є кілька опцій, характерних лише для тривимірної графіки. Зокрема, опція `grid` визначає прямокутну рівномірну сітку значень незалежних змінних функції, по якій обчислюються точки для побудови поверхні. Ця опція задається у вигляді двоелементного списку $[m, n]$, у якому кожен елемент визначає кількість точок на відповідній координаті. За замовчуванням використовується сітка $[25, 25]$. Опція `grid=[round(sqrt(`n`)),round(sqrt(`n`))]` рівносильна опції `numpoints=`n. Є також опції, які задають кольорову схему для відображення поверхні, її орієнтацію у просторі тощо. Змінити ці параметри оформлення для побудованого графіка можна також за допомогою меню.

Приклад:

```
> plot3d(sin(x*y/4),x=-5..5,y=-5..5,axes=boxed);
```



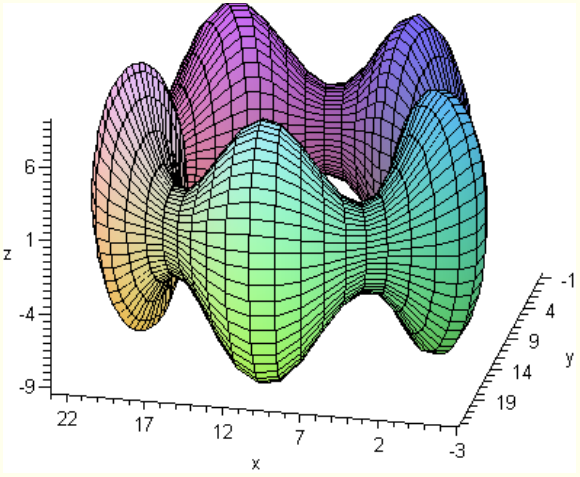
Побудовану поверхню можна повертати за допомогою миші. Тут і далі створена програмою Maple поверхня не обов'язково наводиться так, як вона виглядає зразу після побудови. Часто вона подається у повернутому вигляді.

Для побудови параметрично заданої поверхні $x = x(u, v)$, $y = y(u, v)$, $z = z(u, v)$ потрібно першим аргументом вказати список виразів для трьох осей координат, залежних від двох параметрів:

`plot3d(`$[x(u,v), y(u,v), z(u,v)], u{=}u_1..u_2, v{=}v_1..v_2, opts$`)`

Приклад:

```
> plot3d([10+(10+(2+sin(5*u))*cos(v))*cos(u), 10+(10+
  (2+sin(5*u))*cos(v))*sin(u), 3*(2+sin(5*u))*sin(v)],
  u=0..2*Pi, v=0..2*Pi, axes=framed, grid=[50,50],
  labels=["x","y","z"], scaling=constrained);
```



Додаткові засоби для відображення графіки розглядаються у розділах 11, 12.

Питання до розділу 4

1. Яку команду використовують для побудови двовимірних графіків? Опишіть її формат для побудови графіків явно (параметрично, таблично) заданих функцій.

2. Опишіть основні опції, які використовують для побудови графіків.

3. Назвіть основні особливості побудови графіків функцій з розривами першого (другого) роду, графіків швидкозмінних функцій.

4. Як будувати тривимірні графіки?

Вправи до розділу 4

1. Побудуйте графік функції $y = 3\ln x \sin(2x^2)$ на проміжку $[0, 7]$.

2. Побудуйте графік функції $x = \sin 4t$, $y = \cos^2 t$, $t \in [0, 2\pi]$.

3. Відобразіть на площині точки з координатами $(2; 3)$, $(3; -1)$, $(1/2; 1)$, $(2{,}4; 2)$.

4. Побудуйте графік функції $y = \operatorname{tg} x^2$.

5. Побудуйте в одній системі координат графіки функцій $y = x^3 - 6x^2 + 2x - 1$ і $y = -x^4 + 8x^3 - 18x^2 + 11x - 1$ на проміжку $[-2, 6]$, обмеживши область значень графіків так, щоб їх поведінка була більш зрозумілою. Графік першої функції має бути синім і складатись з крапок подвійної товщини, а другої – коричневим пунктирним потрійної товщини.

6. Побудуйте графік поверхні $z = x\sin(xy + y^2)$, якщо $-3 \leqslant x \leqslant 3$, $-3 \leqslant y \leqslant 3$.

7. Побудуйте у полярних координатах графік функції $\rho = \sin^2(\varphi/3)$, якщо $0 \leqslant \varphi \leqslant 6\pi$.

Розділ 5. Аналітичні перетворення в Maple

§ 5.1. Структура виразів

Усі алгебричні вирази системою Maple зберігаються у вигляді деревовидної структури, що забезпечує доступ до будь-якого її члену чи підвиразу, а також дозволяє виконувати над виразом різноманітні символьні перетворення. Кожен об'єкт Maple складається з підоб'єктів першого рівня, які, в свою чергу, складаються з підоб'єктів другого рівня, і т. д. Підоб'єктами найвищого рівня є базисні (прості) елементи Maple (цілі, дійсні числа, дроби, невідомі тощо).

Нагадаємо, що для визначення типу виразу призначені команди `whattype` і `type` (стор. 30). Наведемо декілька команд, які дозволяють проаналізувати структуру певного виразу. Наприклад, команда `nops(`*expr*`)` виводить кількість операндів (підоб'єктів) першого рівня виразу *expr*.

Команду `op` використовують у різних форматах:

`op(`*expr*`)` виводить операнди першого рівня виразу *expr* у вигляді послідовності підвиразів;

`op(`*i*, *expr*`)` виводить *i*-й операнд першого рівня виразу *expr*;

`op(`*i*`..`*j*, *expr*`)` виводить операнди першого рівня виразу *expr* з *i*-го по *j*-й у вигляді послідовності підвиразів;

`op([`*k1*, *k2*, ..., *kn*, *i*`..`*j*`],` *expr*`)` можна використовувати замість `op(`*i*`..`*j*, `op(`*kn*, ..., `op(`*k2*, `op(`*k1*, *expr*`))))`.

Приклади:

```
> ex:=x^3/y-y+5*sqrt(x)*cos(z^2);
```

$$ex := \frac{x^3}{y} - y + 5\sqrt{x}\cos(z^2)$$

```
> whattype(ex);
```

`+`

```
> nops(ex);
```

3

```
> op(ex);
```

$$\frac{x^3}{y},\ -y,\ 5\sqrt{x}\cos(z^2)$$

```
> op(3,ex);
```

$$5\sqrt{x}\cos(z^2)$$

```
> op(2..3,ex);
```

$$-y,\ 5\sqrt{x}\cos(z^2)$$

Проаналізуємо, наприклад, перший операнд $\frac{x^3}{y}$:

```
> whattype(op(1,ex));
```

$$`*`$$

```
> op([1,1],ex);
```

$$x^3$$

```
> op([1,2],ex);
```

$$\frac{1}{y}$$

```
> whattype(op([1,2],ex));
```

$$`^`$$

```
> op([1,2,1],ex);
```

$$y$$

```
> op([1,2,2],ex);
```

$$-1$$

На рис. 8 схематично зображений вираз

$$\frac{x^3}{y} - y + 5\sqrt{x}\cos(z^2)$$

у вигляді деревовидної структури, у вузлах якої знаходяться операції ($+$, $*$, $\hat{}$, cos), а гілки вказують на операнди:

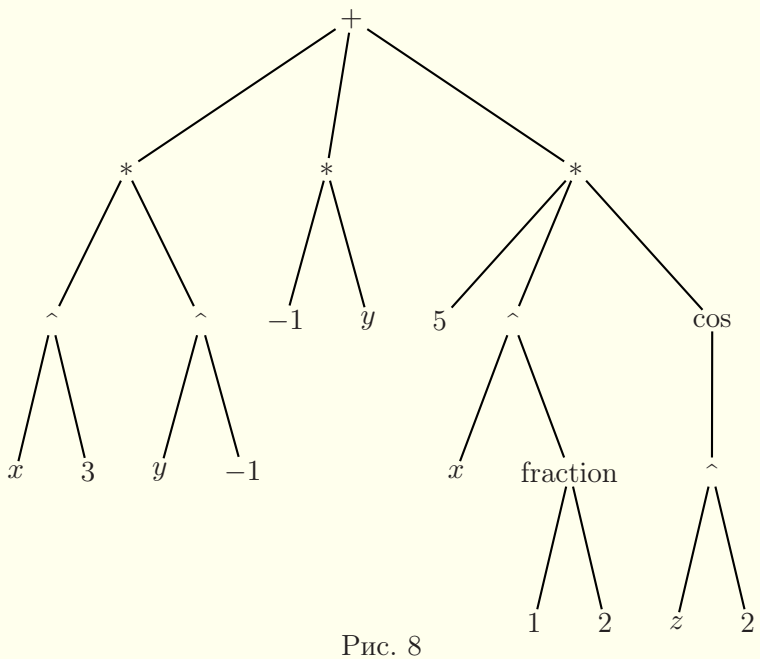


Рис. 8

Операндами списку і множини є їх елементи:

```
> op([2,3,x^2]);
```

$$2,\ 3,\ x^2$$

Команда `has(`*expr*, *x*`)` дозволяє перевірити, чи вираз *expr* містить x – об'єкт, список або множину об'єктів:

```
> ex:=x^3/y-y+5*sqrt(x)*cos(z^2): has(ex,cos);
```

true

Виділити ліву чи праву частину рівняння або нерівності *eq* можна за допомогою команд `lhs(`*eq*`)` і `rhs(`*eq*`)` відповідно:

```
> x+y=2: lhs(%); rhs(%%);
```

$$x + y$$

$$2$$

§ 5.2. Перетворення типів

Команда

`convert(`*expr*, *f*, *arg1*, . . . , *argn*`)`

перетворює (конвертує) вираз *expr* у форму f, якою може бути, наприклад, інший тип даних. Для деяких форм f можна використовувати необов'язкові аргументи *arg1*, . . . , *argn*. В якості параметра f може використовуватись один з багатьох передбачених для цього термінів, наприклад:

`list` – перетворення об'єкта у список;

`set` – перетворення об'єкта у множину;

`rational` – перетворення десяткового дробу в звичайний, може мати додатковий аргумент – кількість цифр;

`float` – перетворення звичайного дробу у десятковий (аналог команди `evalf`);

`degrees` – перетворення числа з радіан у градуси;

`radians` – перетворення числа, заданого у градусах, у радіани.

З усіма термінами можна ознайомитись у довідці. Найбільш характерні випадки використання команди `convert`: перетворення списку в одновимірний масив чи вектор і навпаки, перетворення списку списків у матрицю і навпаки, перетворення списку в множину і навпаки. Приклади:

```
> convert([2,3,4],set);
```

$$\{2, 3, 4\}$$

```
> convert(evalf(Pi), rational);
```

$$\frac{104348}{33215}$$

```
> convert(Pi/9, degrees);
```

$$20\ degrees$$

```
> convert(1250*degrees, radians);
```

$$\frac{125}{18}\,\pi$$

Команда `convert` дозволяє також перетворити число з однієї системи числення в іншу. Для перетворення десяткового числа в іншу систему числення використовують параметри: `binary` (двійкова), `octal` (вісімкова), `hex` (шістнадцяткова), `base`, n (система числення з основою n). В останньому випадку числа, відповідні розрядам, задаються у списку, починаючи з наймолодшого розряду. Для перетворення числа з системи числення з основою n у десяткову систему числення використовують параметри `decimal`, n, де $n \in \{2, 3, \ldots, 36\}$, букви від A до Z відповідають значенням цифр від 10 до 36. Число з буквами обов'язково треба брати у подвійні лапки. Параметри `base`, k, n дозволяють перетворити список чисел, які виконують роль цифр, з системи числення з основою k у список чисел у системі числення з основою n. Приклади:

```
> convert(333,binary);
```

$$101001101$$

```
> convert(%,decimal,2);
```

$$333$$

```
> convert("2E6A",decimal,16);
```

$$11882$$

```
> convert(333,base,45);
```

$$[18, 7]$$

§ 5.3. Розкриття дужок, розклад многочлена на множники, об'єднання виразів

Для розкриття дужок в алгебричному виразі *expr* призначена команда

`expand(`*expr*, *ex1*, *ex2*, . . . , *exn*`)`.

За допомогою параметрів *ex1*, *ex2*, . . . , *exn* можна задати вирази, розкривати дужки у яких не потрібно. Для раціональної функції (відношення двох многочленів) команда розкриває дужки в чисельнику і ділить кожен член отриманого у чисельнику виразу на знаменник, який залишається без змін. Крім того, команда `expand` вміє розкривати дужки у тригонометричних, логарифмічних, показникових та інших функціях. Приклади:

```
> expand((x+1)*(x-3)^3);
```

$$x^4 - 8\,x^3 + 18\,x^2 - 27$$

```
> expand(((x+4)*(2*x+1))/(x+3)^2);
```

$$\frac{2\,x^2}{(x+3)^2} + \frac{9\,x}{(x+3)^2} + \frac{4}{(x+3)^2}$$

```
> expand(tan(x+y));
```

$$\frac{\tan(x) + \tan(y)}{1 - \tan(x)\tan(y)}$$

```
> expand((x+4)^2*(y+3*z),x+4);
```

$$(x+4)^2\,y + 3\,(x+4)^2\,z$$

Якщо параметр *expr* задати у вигляді списку, множини або рівняння, то команда `expand` застосовуватиметься до всіх компонентів *expr*. Це зауваження стосується і решти команд

§§ 5.3 – 5.5, 5.7, однак нагадувати про це для кожної команди не будемо. Приклад:

```
> expand([(x-3)*(x+4),(y+12)^2]);
```

$$[x^2 + x - 12, y^2 + 24\,y + 144]$$

Команда

`factor(`mn, K`)`

розкладає на множники многочлен mn над числовим полем K. Невідома у многочлені може бути функцією від невідомої. Якщо параметр K не заданий, то многочлен розкладатиметься на множники над числовим полем, до якого належать коефіцієнти многочлена. В якості поля K можна використовувати терміни `real`, `complex`, радикал чи список (множину) радикалів. Приклади:

```
> factor(x^2*y^3+x*y^3-3*x^2*y^2-9*x*y^2+3*x^2*y+
  y^4*x+11*x*y-x^2-4*x+4*y^4-12*y^3+12*y^2-4*y);
```

$$(y-1)^3\,(4+x)\,(x+y)$$

```
> factor(sin(z)^2-3*sin(z)+2);
```

$$(\sin(z)-1)\,(\sin(z)-2)$$

```
> factor(x^3+3);
```

$$x^3 + 3$$

```
> factor(x^3+3,real);
```

$$(x + 1.442249570)\,(x^2 - 1.442249570\,x + 2.080083822)$$

```
> factor(x^3+3,complex);
```

$$(x + 1.442249570)\,(x - 0.7211247852 + 1.249024766\,I)\,(x - 0.7211247852 - 1.249024766\,I)$$

```
> factor(x^3+3,3^(1/3));
```

$$(x^2 - x\,3^{1/3} + 3^{2/3})\,(x + 3^{1/3})$$

Якщо команду `factor` застосувати до раціональної функції, то спочатку відбудеться скорочення спільних множників чисельника та знаменника, а потім чисельник та знаменник будуть розкладені на множники над полем K. Наприклад:

```
> f:=(x^7-x^6*y^1-y^6*x^1+y^7)/(x^5-x^4*y-2*y^2*x^3+
  2*y^3*x^2+y^4*x-y^5);
```

$$f := \frac{x^7 - x^6\,y - y^6\,x + y^7}{x^5 - x^4\,y - 2\,y^2\,x^3 + 2\,y^3\,x^2 + y^4\,x - y^5}$$

```
> factor(f);
```

$$\frac{(y^2 + x\,y + x^2)\,(y^2 - x\,y + x^2)}{(x - y)\,(x + y)}$$

Команда

`combine`(*expr*, *names*, *opt*)

виконує дію, протилежну дії команди `expand` при роботі з різними математичними функціями: вона об'єднує вираз, перетворюючи його у більш компактну форму. Необов'язковий аргумент *names* може визначати правило чи список правил, які будуть застосовуватись при роботі команди `combine` (ними можуть бути: `abs`, `arctan`, `conjugate`, `exp`, `icombine`, `ln`, `piecewise`, `polylog`, `power`, `Psi`, `radical`, `range`, `signum`, `trig`). Для функцій, правила перетворення яких залежать від значень їх аргументів чи мають на них обмеження, останнім аргументом команди `combine` можна задати слово `symbolic`, яке змусить команду виконувати формальні перетворення без врахування області визначення. Приклади:

```
> combine(cos(x)*sin(y)+sin(x)*cos(y));
```

$$\sin(x + y)$$

```
> combine(ln(x)+ln(y));
```

$$\ln(x) + \ln(y)$$

```
> combine(ln(x)+ln(y),symbolic);
```

$$\ln(x\,y)$$

```
> combine(exp(cos(x)*cos(y))*exp(sin(y)*sin(x)), exp);
```

$$\mathrm{e}^{\cos(x)\cos(y)+\sin(y)\sin(x)}$$

```
> combine(exp(cos(x)*cos(y))*exp(sin(y)*sin(x)),trig);
```

$$\mathrm{e}^{\frac{1}{2}\cos(x-y)+\frac{1}{2}\cos(x+y)}\,\mathrm{e}^{\frac{1}{2}\cos(x-y)-\frac{1}{2}\cos(x+y)}$$

```
> combine(exp(cos(x)*cos(y))*exp(sin(y)*sin(x)), exp,
  trig);
```

$$\mathrm{e}^{\cos(x-y)}$$

§ 5.4. Зведення подібних доданків

Для зведення подібних доданків використовують команду

`collect(`*expr*, *obj*`)`,

де *obj* – об'єкт чи об'єкти, відносно яких треба звести (згрупувати за степенями) вираз *expr*. В якості *obj* можна використовувати ім'я невідомої величини, ім'я функції з аргументом-невідомою, ім'я функції без аргументу, список чи множину таких об'єктів. Maple розглядає вираз *expr* як многочлен відносно об'єктів *obj* і збирає коефіцієнти біля однакових раціональних (додатних, від'ємних і дробових) степенів. Сортування многочлена за зростанням чи спаданням степенів не здійснюється. Приклади:

```
> g:=x^2*exp(x)+2*x*exp(x)-x*sin(x)+2*cos(x)-sin(x)+
  exp(2*x)-3*x*exp(2*x);
```

$$g := x^2\,\mathrm{e}^x + 2\,x\,\mathrm{e}^x - x\sin(x) + 2\cos(x) - \sin(x) + \mathrm{e}^{2\,x} - 3\,x\,\mathrm{e}^{2\,x}$$

```
> collect(g,x);
```

$$x^2\,\mathrm{e}^x + (-\sin(x) + 2\,\mathrm{e}^x - 3\,\mathrm{e}^{2\,x})\,x - \sin(x) + \mathrm{e}^{2\,x} + 2\cos(x)$$

```
> collect(g,exp(x));
```

$$(2\,x + x^2)\,\mathrm{e}^x - \sin(x) + \mathrm{e}^{2\,x} - x\sin(x) + 2\cos(x) - 3\,x\,\mathrm{e}^{2\,x}$$

```
> collect(g,exp);
```

$$(2\,x + x^2)\,\mathrm{e}^x + (1 - 3\,x)\,\mathrm{e}^{2\,x} - \sin(x) - x\sin(x) + 2\cos(x)$$

```
> gg:=x^2*y-3*x^2+4*y-12+6*x*y+a*x^2*y;
```

$$gg := x^2\,y - 3\,x^2 + 4\,y - 12 + 6\,x\,y + a\,x^2\,y$$

```
> collect(gg,x); collect(gg,y);
```

$$(-3 + y + a\,y)\,x^2 + 6\,x\,y + 4\,y - 12$$
$$(x^2 + 4 + 6\,x + a\,x^2)\,y - 3\,x^2 - 12$$

```
> collect(gg,[x,y]); collect(gg,[y,x]);
```

$$(-3 + (1 + a)\,y)\,x^2 + 6\,x\,y + 4\,y - 12$$
$$(4 + (1 + a)\,x^2 + 6\,x)\,y - 3\,x^2 - 12$$

§ 5.5. Скорочення і раціоналізація дробів

Команда `normal(`f`)` призначена для зведення алгебричних дробів до спільного знаменника та скорочення чисельника і знаменника на найбільший спільний дільник. Нагадаємо, що

самé скорочення дробу здійснюється автоматично, без застосування команди `normal`. Команда може мати своїм необов'язковим другим параметром термін `expanded`, який вказує на те, що у чисельнику і знаменнику додатково потрібно розкрити дужки. Якщо у команді `normal` f – список, множина, послідовність, рівняння, функція, то ця команда послідовно застосовується до всіх елементів (компонентів) f. Приклади:

```
> ff:=1/(x+y)-(x+y^2)/(x-3*y);
```

$$\frac{1}{x+y}-\frac{x+y^2}{x-3\,y}$$

```
> normal(ff);
```

$$-\frac{-x+3\,y+x^2+xy+y^2x+y^3}{(x+y)\,(x-3\,y)}$$

```
> normal(ff,expanded);
```

$$\frac{x-3\,y-x^2-xy-y^2x-y^3}{x^2-2\,xy-3\,y^2}$$

```
> normal(1/x+1/y=1/(x+1)-1/(y-1));
```

$$\frac{y+x}{x\,y}=-\frac{-y+2+x}{(x+1)\,(y-1)}$$

Команди `numer(`f`)` і `denom(`f`)` вертають відповідно чисельник і знаменник алгебричного дробу, отриманого після зведення алгебричних дробів до спільного знаменника та скорочення чисельника і знаменника на найбільший спільний дільник:

```
> ff:=1/(x+y)-(x+y^2)/(x-3*y): numer(ff);
```

$$x-3\,y-x^2-xy-y^2x-y^3$$

```
> denom(ff);
```

$$(x+y)\,(x-3\,y)$$

Під раціоналізацією дробів розуміють позбавлення від ірраціональності у знаменнику. Команда `rationalize(`f`)` призначена саме для цього:

```
> 1/(2-sqrt(3));
```

$$\frac{1}{2-\sqrt{3}}$$

```
> rationalize(%);
```

$$2+\sqrt{3}$$

§ 5.6. Обмеження на невідомі

За замовчуванням вважається, що невідомі величини можуть набувати довільних комплексних значень. Команда `assume` дозволяє накласти на невідомі певні обмеження, тобто встановити їх належність до деякої числової множини. Ці обмеження можна задавати в одному з трьох форматів:

$$\texttt{assume}(x1, prop1, x2, prop2, \ldots, xn, propn),$$
$$\texttt{assume}(x1::prop1, x2::prop2, \ldots, xn::propn),$$
$$\texttt{assume}(xrel1, xrel2, \ldots, xreln),$$

або скористатись їх комбінацією. Тут $x1$, $x2$, …, xn – невідомі, $prop1$, $prop2$, …, $propn$ – властивості, $xrel1$, $xrel2$, …, $xreln$ – невідомі з накладеними на них за допомогою логічних відношень обмеженнями. Властивості $prop1$, $prop2$, …, $propn$ можуть визначати тип даних або один з термінів, наприклад:

Властивість	**Значення**
`real`	дійсні числа
`negative`	від'ємні дійсні числа
`nonnegative`	невід'ємні дійсні числа
`positive`	додатні дійсні числа
`complex`	комплексні числа

Властивість	Значення
`imaginary`	комплексні числа з нульовою дійсною частиною
`NumeralNonZero`	комплексні числа, відмінні від нуля
`integer`	цілі числа
`rational`	раціональні числа (дроби і цілі)
`irrational`	ірраціональні числа
`fraction`	тільки дробові числа
`natural`	невід'ємні цілі числа
`posint`	натуральні числа
`odd`	непарні числа
`even`	парні числа
`prime`	прості числа
`scalar`	скалярні величини

Крім того, властивості можуть визначати проміжок $[a, b]$ дійсної осі, він позначається через `RealRange(`a`,` b`)`, інтервал (a, b) – `RealRange(Open(`a`),Open(`b`))`, а також півінтервали `RealRange(Open(`a`),` b`)` і `RealRange(`a`,Open(`b`))`. Є також обмеження для функцій і матриць. З повним переліком обмежень можна ознайомитись у довідковій системі на сторінці, що відображається командою `?property`.

Якщо на змінну накладаються обмеження, то до неї можна застосовувати значно більше перетворень, а біля її імені відображається знак тильда ($\sim$). Приклади:

```
> ln(a^2);
```

$$\ln(a^2)$$

```
> assume(a>0); ln(a^2);
```

$$2\ln(a\sim)$$

```
> assume(p::negative); sqrt(p^6);
```

$$-p\sim^3$$

```
> assume(p,nonnegative,q>=0); sqrt((p+q)^2);
```

$$p\sim + q\sim$$

```
> assume(n, integer): cos(Pi*n);
```

$$(-1)^{n\sim}$$

```
> assume(k, even): sin(Pi*k/2);
```

$$0$$

Кожне нове обмеження, накладене командою `assume`, скасовує всі попередні обмеження щодо тієї ж змінної. Щоб не скасовувати вже накладені обмеження, використовують команду `additionally`, параметри якої повністю збігаються з параметрами команди `assume`. Обмеження, накладені командою `additionaly`, додаються до обмежень, які введені командами `assume` та попередніми командами `additionaly`. Для зняття всіх обмежень потрібно змінній присвоїти її ім'я в одинарних лапках, наприклад, `x:='x'`.

Команда `is` з одним або двома аналогічними аргументами дозволяє перевірити, чи має змінна певну властивість. Якщо хоч одне з можливих значень змінної не задовольняє обмеження, то відповіддю буде `false`. Команда `coulditbe` з аналогічними аргументами перевіряє, чи може вказана змінна задовольняти задану властивість. Якщо хоч одне з можливих значень змінної може мати задану властивість, то отримаємо `true`. Команда `hasassumptions(`x`)` дозволяє визначити, чи на змінну x накладались які-небудь обмеження. Після виконання команди `about(`x`)` одержимо інформацію про всі обмеження, які накладались на змінну x. Приклади:

```
> assume(a>0): is(a>0);
```

$$true$$

```
> additionally(a<=1); coulditbe(a=2);
```

false

```
> hasassumptions(a);
```

true

```
> about(a);
```

Originally a, renamed a~:
  is assumed to be: RealRange(Open(0),1)

```
> assume(x>=2,x<7); about(x);
```

Originally x, renamed x~:
  is assumed to be: RealRange(2,Open(7))

Замість команди `assume` для накладання обмежень лише на один вираз *expr* можна використовувати одну з конструкцій:

$$expr\ \texttt{assuming}\ xrel,$$
$$expr\ \texttt{assuming}\ x\texttt{::}prop,$$

де *xrel*, *x*, *prop* – ті самі, що й у команді `assume`. Приклад:

```
> sqrt(p^2) assuming p::real;
```

$$p$$

На стор. 126, 163 йтиметься про те, як зробити так, щоб усі невідомі вважались дійсними.

§ 5.7. Спрощення виразів

Елементарні спрощення, наприклад, скорочення дробів, зведення подібних доданків в очевидних випадках і деякі інші виконуються у Maple автоматично. Для спрощення виразів у складніших випадках призначена команда `simplify`. Вона

вміє спрощувати вирази з дробами, тригонометричними, оберненими тригонометричними функціями, логарифмами, експонентами тощо.

Формат цієї команди

`simplify(`*expr*, *opts*`)`

де *expr* – вираз, який потрібно спростити, а *opts* – необов'язкові додаткові параметри, використання яких значно збільшує ефективність команди `simplify`.

По-перше, параметри *opts* можуть бути конкретними процедурами, які дозволяється використовувати для спрощення. Наприклад, `power` дозволяє здійснювати спрощення, пов'язані зі степенями, логарифмами і експонентами, `sqrt` – спрощення, пов'язані лише з квадратними коренями, `trig` – лише тригонометричні спрощення. З іншими процедурами, як-от `radical`, `polar`, `hypergeom`, `ln`, можна ознайомитись у довідці. Замість цих процедур або додатково до них можна вказувати опцію `assume=`*prop*, яка дозволяє здійснювати спрощення при деяких додаткових обмеженнях (це, як правило, розширює множину допустимих перетворень). В якості *prop* можна використовувати певний тип даних чи будь-яку з перелічених у попередньому параграфі властивостей, наприклад, `integer`, `positive`, `RealRange(`*a*, *b*`)` тощо. При використанні опції `assume` вважатиметься, що всі невідомі виразу *expr* задовольняють вказану властивість. Додатково до згаданих процедур перетворення виразів або замість них можна використовувати опцію `symbolic`, яка означає, що для багатозначних функцій буде здійснюватись формальне символічне спрощення, при цьому, скоріш за все, результат не буде правильним на всій комплексній площині. Ця опція несумісна з опцією `assume`. Замість розглянутих необов'язкових параметрів можна використовувати список або множину рівностей, які задають спрощення. Приклади:

```
> simplify(sin(x)^4+cos(x)^4+sin(2*x)^2/2);
```

$$1$$

```
> simplify(sqrt(x^2));
```

$$\mathrm{csgn}(x)\,x$$

```
> simplify(sqrt(x^2),assume=real);
```

$$|x|$$

```
> simplify(sqrt(x^2),assume=negative);
```

$$-x$$

```
> simplify(sqrt(x^2),symbolic);
```

$$x$$

```
> simplify(x^3+x*y^2,[x=2*y]);
```

$$10\,y^2$$

Зауважимо, що результат спрощення може зовсім не відповідати сподіванням користувача щодо очікуваного результату. Наприклад, виконуючи спрощення тригонометричних виразів, нерідко використовують прийом, коли 1 замінюють на $\sin^2 x + \cos^2 x$, а Maple може замінити навпаки.

Якщо команда `simplify` не може спростити відповідний вираз, то вона його просто повторить у області виведення. У цьому випадку можна спробувати використати параметри *opts* для уточнення перетворень.

Більш детально з правилами спрощення виразів можна ознайомитись у довідці Maple, виконавши команду `?simplify`.

Для спрощення виразів, які містять не тільки квадратні, але й корені інших степенів, краще використовувати команду `radnormal(`f`)`. Наприклад:

```
> b:=sqrt(3+sqrt(3)+(10+6*sqrt(3))^(1/3));
```

$$b := \sqrt{3+\sqrt{3}+\left(10+6\sqrt{3}\right)^{1/3}}$$

```
> simplify(b,power);
```

$$\sqrt{3+\sqrt{3}+\left(10+6\sqrt{3}\right)^{1/3}}$$

```
> radnormal(b);
```

$$1+\sqrt{3}$$

§ 5.8. Підстановки

Команда `subs`$(s_1, \ldots, s_n, expr)$ призначена для виконання синтаксичних підстановок у вираз *expr*. Підстановки задаються одним чи кількома параметрами s_1, ..., s_n, якими можуть бути рівності вигляду *oldexpr*=*newexpr*, множини або списки таких рівностей. У цих рівностях до знаку «=» ставиться старий вираз *oldexpr*, а після нього – новий вираз *newexpr*, на який буде замінено старий. Якщо підстановки задані у вигляді списку чи множини, то вони виконуються одночасно, інакше – по черзі. Приклади:

```
> f:=x^3+a*x^2+sin(3*x);
```

$$f := x^3 + a\,x^2 + \sin(3\,x)$$

```
> subs(x=1-y^2,f);
```

$$(1-y^2)^3 + a\,(1-y^2)^2 + \sin(3-3\,y^2)$$

```
> subs([x=a,a=x],f);
```

$$a^3 + x\,a^2 + \sin(3\,a)$$

```
> subs(x=a,a=x,f);
```

$$2\,x^3 + \sin(3\,x)$$

```
> subs(x^2=t,f);
```

$$x^3 + a\,t + \sin(3\,x)$$

Остання команда не підставила у вираз x^3 замість x^2 величину t. Аналогічно, залишився x під знаком функції $\sin 3x$.

Є кілька інших способів виконання замін. Наприклад, команда `eval(f,x^2=t)` (стор. 51) дозволяє отримати той самий результат, що й команда `subs(x^2=t,f)`.

Команда `algsubs(`*oldexpr*`=`*newexpr*, *expr*`)` виконує алгебричну підстановку *newexpr* замість *oldexpr* у вираз *expr*:

```
> algsubs(x^2=t,f);
```

$$t\,x + a\,t + \sin(3\,x)$$

Подібний результат дає команда `simplify`:

```
> simplify(f,[x^2=t]);
```

$$4\sin(x)\cos(x)^2 - \sin(x) + x\,t + a\,t$$

Кращі результати одержимо, якщо замість x підставити $\sqrt{t}$:

```
> subs(x=sqrt(t),f);
```

$$t^{3/2} + a\,t + \sin(3\sqrt{t})$$

```
> algsubs(x=sqrt(t),f);
```

$$t^{3/2} + a\,t + \sin(3\sqrt{t})$$

```
> eval(f,x=sqrt(t));
```

$$t^{3/2} + a\,t + \sin(3\sqrt{t})$$

Якщо відомо, який операнд виразу потрібно замінити, то використовують команду

`subsop(`x_1`=`*newexpr1*, $\ldots$, x_n`=`*newexprn*, *expr*`)`,

де $x_1, \ldots, x_n$ – номери операндів або списки номерів операндів у виразі *expr*, складені за тими самими правилами, що й для команди `op` (стор. 77). Ці операнди буде замінено новими виразами *newexpr1*, ..., *newexprn*:

```
> subsop(3=cos(4*y),f);
```

$$x^3 + a\,x^2 + \cos(4\,y)$$

```
> subsop([2,2,2]=4,f);
```

$$x^3 + a\,x^4 + \sin(3\,x)$$

У багатьох випадках корисною є команда `trigsubs(`*expr*`)`, яка знаходить всі тригонометричні еквіваленти виразу *expr*, наприклад:

```
> trigsubs(cos(2*x));
```

$$\left[\cos(2\,x), \cos(-2\,x), 2\cos(x)^2 - 1, 1 - 2\sin(x)^2, \cos(x)^2 - \sin(x)^2, \right.$$
$$\left. \frac{1}{\sec(2\,x)}, \frac{1}{\sec(-2\,x)}, \frac{1-\tan(x)^2}{1+\tan(x)^2}, \frac{1}{2}\mathrm{e}^{2\,I\,x} + \frac{1}{2}\mathrm{e}^{-2\,I\,x} \right]$$

§ 5.9. Команди `map`, `zip`, `select`, `remove`, `sort`

Команда

`map(`*fnc*, *expr*, *arg1*, ..., *argn*`)`

застосовує команду або функцію *fnc* до кожного операнда виразу, елемента списку, множини або масиву *expr*. Необов'язкові аргументи *arg1*, ..., *argn* команди `map` використовують як додаткові аргументи команди *fnc*. Приклади:

```
> map(x->x^3+1,[1,2,3,4,5]);
```

$$[2, 9, 28, 65, 126]$$

```
> map(surd,x+y+z,3);
```

$$\sqrt[3]{x} + \sqrt[3]{y} + \sqrt[3]{z}$$

Аналогічною до команди `map` є команда

`map2(`*fnc*, *arg1*, *expr*, *arg2*, ..., *argn*`)`.

У цьому випадку першим аргументом команди *fnc* буде *arg1*, а другим – один з операндів виразу, елементів списку, множини або масиву *expr*. При необхідності в якості решти аргументів можна використовувати *arg2*, ..., *argn*. Приклад:

```
> map2(op,1,{q+p,r*s,w-z});
```

$$\{q, r, w\}$$

Команда `select(`*fnb*, *expr*, *arg1*, ..., *argn*`)` створює новий вираз, список, множину або масив, вибираючи їх операнди чи елементи з виразу, списку, множини або масиву *expr* відповідно до логічної функції *fnb*. При необхідності в якості решти аргументів команди *fnb* можна використовувати *arg1*, ..., *argn*. Команда `remove(`*fnb*, *expr*, *arg1*, ..., *argn*`)` діє аналогічно, але вибирає замість елементів, що задовольняють логічну функцію *fnb*, елементи, які її не задовольняють.

Команда `selectremove(`*fnb*, *expr*, *arg1*, ..., *argn*`)` формує послідовність з двох об'єктів, відповідних командам `select` і `remove`. Приклади:

```
> select(isprime,[$20..35]);
```

$$[23, 29, 31]$$

```
> remove(isprime,[$20..35]);
```

$$[20, 21, 22, 24, 25, 26, 27, 28, 30, 32, 33, 34, 35]$$

```
> selectremove(isprime,[$20..35]);
```

$$[23, 29, 31], [20, 21, 22, 24, 25, 26, 27, 28, 30, 32, 33, 34, 35]$$

Команда `zip(`*fnc*, *u*, *v*, *d*`)` застосовує бінарну функцію *fnc* до двох списків, векторів, масивів чи матриць *u* і *v*. Об'єкти *u* і *v* повинні мати однаковий тип. У результаті дії команди `zip` формується новий список, вектор, масив чи матриця, який містить таку саму кількість елементів, що й найменший з об'єктів *u*, *v*. Якщо використати необов'язковий скалярний аргумент *d*, то створений об'єкт матиме стільки елементів, скільки їх є в найбільшому з об'єктів *u*, *v*, а замість відсутніх значень буде використовуватись *d*. Приклади:

```
> zip((x,y)->x^2+y^4,[2,3,4,5,s],[k,7,2,-3,8,1,6]);
```

$$[4 + k^4, 2410, 32, 106, s^2 + 4096]$$

```
> zip((x,y)->x^2+y^4,[2,3,4,5,s],[k,7,2,-3,8,1,6],2);
```

$$[4 + k^4, 2410, 32, 106, s^2 + 4096, 5, 1300]$$

Команда `sort(`L, f`)` сортує числовий список L за спаданням елементів, якщо немає аргументу f, і за зростанням – якщо в якості необов'язкового аргументу f використовується `'>'` (разом зі зворотними апострофами). Крім того, ця команда дозволяє сортувати многочлен L за спаданням степенів, причому параметром f треба обов'язково вказати змінну (або їх список для многочлена кількох змінних), за степенями якої здійснюється сортування. Якщо необов'язковим третім аргументом команди `sort` вказати термін `ascending`, то многочлен сортуватиметься за зростанням степенів. Приклади:

```
> sort([4.6,7,-2,3,10,3/2]);
```

$$\left[-2, \frac{3}{2}, 3, 4.6, 7, 10\right]$$

```
> sort([4.6,7,-2,3,10,3/2],`>`);
```

$$\left[10, 7, 4.6, 3, \frac{3}{2}, -2\right]$$

```
> sort(x^3-3*x^2+a*x+4*x^4-5,x);
```

$$4\,x^4 + x^3 - 3\,x^2 + a\,x - 5$$

```
> sort(x^3-3*x^2+a*x+4*x^4-5,x,ascending);
```

$$-5 + a\,x - 3\,x^2 + x^3 + 4\,x^4$$

§ 5.10. Команди для роботи з многочленами

Крім описаних у §§ 5.3 – 5.9 команд (наприклад `expand`, `factor`, `sort`), що дозволяють обробляти різні вирази, зокрема і многочлени, є низка команд, які застосовують лише до многочленів. Наведемо основні з них:

`coeff(`*mn*, *var*, *n*`)` або `coeff(`*mn*, *var*`^`*n*`)` визначає коефіцієнт біля *n*-го степеня змінної *var* у многочлені *mn*;

`coeffs(`*mn*, *var*`)` визначає коефіцієнти многочлена *mn* відносно змінної *var* або множини (списку) змінних *var*;

`lcoeff(`*mn*, *var*`)` визначає коефіцієнт біля найстаршого степеня змінної *var* у многочлені *mn*;

`tcoeff(`*mn*, *var*`)` визначає вільний член у многочлені *mn* змінної *var*;

`degree(`*mn*, *var*`)` і `ldegree(`*mn*, *var*`)` визначають степінь і найменший степінь многочлена *mn* відносно змінної *var* відповідно;

`discrim(`*mn*, *var*`)` обчислює дискримінант многочлена *mn* відносно змінної *var*.

Якщо у наведених командах зрозуміло, що виступає змінною, то параметр *var* можна не вказувати.

Приклади:

```
> g:=x^5-5*x^4+3*x^3-2*x+6;
  g1:=3*x^3*y^4-x^2*y+y^6-7*x*y^3;
```

$$g := x^5 - 5\,x^4 + 3\,x^3 - 2\,x + 6$$
$$g1 := 3\,x^3\,y^4 - x^2\,y + y^6 - 7\,x\,y^3$$

```
> coeff(g,x,4);
```

$$-5$$

```
> coeff(g1,y^3);
```

$$-7\,x$$

```
> coeffs(g);
```

$$6,\ -2,\ 1,\ -5,\ 3$$

```
> coeffs(g1);
```

$$1,\ -7,\ 3,\ -1$$

```
> coeffs(g1,[x,y]);
```

$$1,\ -7,\ 3,\ -1$$

```
> coeffs(g1,x);
```

$$y^6,\ -7\,y^3,\ -y,\ 3\,y^4$$

```
> lcoeff(g1);
```

$$3$$

```
> degree(g1);
```

$$7$$

```
> ldegree(g1);
```

$$3$$

```
> discrim(x^3+2*x^2-4*x+3,x);
```

$$-451$$

Команда `divide(`*mn1*, *mn2*, '*name*'`)` вертає значення `true`, якщо многочлен *mn1* ділиться націло на многочлен *mn2*. Крім того, після виконання команди змінна *name* міститиме частку від ділення двох многочленів, якщо вони діляться націло. Одинарні лапки є обов'язковими. Приклад:

```
> divide(x^4-27*x^2+14*x+120,x^2+7*x+10,'r');
```

true

```
> r;
```

$$x^2 - 7\,x + 12$$

Команда `gcd(`*mn1*, *mn2*`)` знаходить найбільший спільний дільник многочленів *mn1*, *mn2*:

```
> gcd(x^3+3*x-3*x^2-9,x^4+7*x^2+5*x^3+15*x+12);
```

$$x^2 + 3$$

Команда `lcm(`*mn1*, *mn2*, . . . , *mnk*`)` визначає найменше спільне кратне многочленів *mn1*, *mn2*, . . . , *mnk*. Приклад:

```
> lcm(x^2-x-12,x^3-4*x^2+4*x-16);
```

$$(x + 3)\,(x^3 - 4\,x^2 + 4\,x - 16)$$

Команда `proot(`*mn*, *n*`)` знаходить корінь степеня n з многочлена *mn*, якщо цей многочлен є повним n-м степенем деякого іншого многочлена. У протилежному разі маємо значення *_NOROOT*. Приклади:

```
> proot(x^4+12*x^3*y+54*x^2*y^2+108*x*y^3+81*y^4,4);
```

$$x + 3\,y$$

```
> proot(x^4+12*x^3*y+54*x^2*y^2+108*x*y^3+81*y^4,3);
```

$$\_NOROOT$$

Команда `psqrt(`*mn*`)` знаходить квадратний корінь з многочлена *mn*, якщо цей многочлен є повним квадратом деякого іншого многочлена. У протилежному разі виводиться *_NOSQRT*. Приклад:

```
> psqrt(x^6*y^2+4*x^3*y+4);
```

$$x^3\,y + 2$$

Команда `quo(`*mn1*, *mn2*, *var*`)` знаходить цілу частину від ділення многочлена *mn1* на многочлен *mn2*, де *var* – змінна

многочленів. У форматі `quo(`*mn1*, *mn2*, *var*, '*r*'`)` додатково знаходиться остача r від ділення многочленів. Приклади:

```
> quo(x^5-4*x^3+2*x^2+5*x-2,x^3+2*x^2-6*x+3,x);
```

$$x^2 - 2\,x + 6$$

```
> quo(x^5-4*x^3+2*x^2+5*x-2,x^3+2*x^2-6*x+3,x,'r'): r;
```

$$-20 - 25\,x^2 + 47\,x$$

Команда `rem(`*mn1*, *mn2*, *var*`)` знаходить остачу від ділення многочлена *mn1* на многочлен *mn2*, де *var* – змінна. У форматі `rem(`*mn1*, *mn2*, *var*, '*q*'`)` додатково визначається ціла частина q від ділення многочленів. Приклад:

```
> rem(x^5-4*x^3+2*x^2+5*x-2,x^3+2*x^2-6*x+3,x,'r');
```

$$-20 - 25\,x^2 + 47\,x$$

```
> r;
```

$$x^2 - 2\,x + 6$$

Команда `realroot(`*mn*, *width*`)` генерує список проміжків, на яких містяться відокремлені дійсні нулі многочлена *mn*. Проміжки задаються як списки $[a, b]$ двох чисел, $a \leqslant b$, причому список $[a, a]$ визначає точний корінь многочлена *mn* – число a. Необов'язковим параметром *width* можна задати обмеження зверху на величину проміжків. Приклад:

```
> realroot(x^5-4*x^4+3*x^2+2*x-2);
```

$$[[1, 1], [0, 1], [2, 4]]$$

```
> realroot(x^5-4*x^4+3*x^2+2*x-2,1/4);
```

$$\left[[1, 1], \left[\frac{1}{2}, \frac{3}{4}\right], \left[\frac{15}{4}, 4\right]\right]$$

Команда `roots(`*mn*, *var*, *K*`)` знаходить всі нулі многочлена *mn* та їх кратність над числовим полем K, *var* – це змінна

многочлена. Обов'язковим аргументом є лише многочлен mn. Якщо параметр K не вказати, то шукатимуться лише раціональні нулі многочлена. В якості поля K можна використовувати радикал чи список (множину) радикалів. Результатом роботи команди є список двоелементних списків, першим елементом кожного з яких є черговий нуль многочлена, а другим – його кратність. Приклад:

```
> roots(4*x^8-8*x^7-59*x^6+263*x^5-520*x^4+770*x^3-
  812*x^2+424*x-80);
```

$$\left[[2,3],\left[\frac{1}{2},2\right],[-5,1]\right]$$

Питання до розділу 5

1. Назвіть основні команди, призначені для аналізу структури виразів.
2. Як перетворити вираз з одного типу в інший, число з однієї системи числення в іншу?
3. Як розкрити дужки у виразі?
4. Яка команда призначена для зведення подібних доданків?
5. Яка команда дозволяє спростити вираз?
6. Яким чином можна накласти припущення на невідомі?
7. Як розкласти многочлен на множники?
8. Як виконувати підстановки?
9. Назвіть основні команди для роботи з многочленами.

Вправи до розділу 5

1. Розкрийте дужки у виразі $(x^2+2)(x+5)(x-4)$.
2. Розкладіть многочлен $x^5-4x^3+2x^4+3x-18$ на множники окремо над полем цілих, дійсних та комплексних чисел.
3. Зведіть подібні доданки у виразі $3t^2-6t+at-bt^2+5a$ відносно змінної t.
4. Позбудьтесь ірраціональності у знаменнику виразу $\frac{1}{\sqrt{2}+\sqrt[5]{3}}$.
5. Спростіть вираз $2(\sin^6 x+\cos^6 x)-3(\sin^4 x+\cos^4 x)$.
6. Відокремте всі дійсні нулі многочлена $x^4+4x^3-3x^2-12x+8$.

Розділ 6. Застосування Maple до розв'язування задач математичного аналізу

§ 6.1. Границі послідовностей і функцій

У Maple для деяких математичних операцій існують дві команди: перша – прямої, а інша – відкладеної дії. Імена відповідних команд складаються з однакових букв, крім першої: команди прямої дії починаються з малої букви, команди відкладеної дії – з великої. Після звертання до команди відкладеної дії математичні операції, наприклад, знаходження інтегралів, границь, похідних, виводяться на екран у вигляді їх стандартного аналітичного запису. Обчислення у цьому випадку не здійснюються. Команди прямої дії видають результат одразу.

Для знаходження границі $\lim\limits_{x\to a} f(x)$ є дві команди:

1) команда прямої дії `limit(`f`,`x`=`a`,`par`)`, де f – вираз, границю якого потрібно знайти, x – змінна, a – точка, у якій обчислюється границя (число a може бути й невласним), par – необов'язковий параметр для відшукання однобічних границь (`left` – для лівобічної, `right` – для правобічної) або вказування типу змінної (`real` – дійсна, `complex` – комплексна).

2) команда відкладеної дії `Limit(`f`,`x`=`a`,`par`)` з тими самими аргументами.

Розглянемо приклади:

```
> Limit(ln(sin(4*x))/ln((1-exp(2*x))), x=0);
```

$$\lim_{x\to 0} \frac{\ln\left(\sin\left(4\,x\right)\right)}{\ln\left(1-\mathrm{e}^{2\,x}\right)}$$

```
> limit(ln(sin(4*x))/ln((1-exp(2*x))), x=0);
```

$$1$$

Якщо границя функції не існує, то результат може бути такий:

```
> limit(tan(x), x=infinity);
```

$$\textit{undefined}$$

```
> limit(cos(x/(x-Pi/2)), x=Pi/2);
```

$$-1..1$$

В останньому випадку повідомлено, що границю не знайдено (вона не існує), однак вказано діапазон $-1..1$ значень, які набуває функція $\cos\frac{x}{x-\pi/2}$ в околі граничної точки $x=\frac{\pi}{2}$.

Об'єднуючи команди `limit` і `Limit`, можна записувати математичні вирази з границями у звичному аналітичному вигляді, наприклад:

```
> Limit(sin(x)^tan(x), x=Pi/2)=limit(sin(x)^tan(x),
  x=Pi/2);
```

$$\lim_{x\to\frac{1}{2}\,\pi}\sin(x)^{\tan(x)}=1$$

```
> Limit((a^x-a^b)/(x-b),x=b)=limit((a^x-a^b)/(x-b),
  x=b);
```

$$\lim_{x\to b}\frac{a^x-a^b}{x-b}=\ln(a)\,a^b$$

Команда `value(`*expr*`)` обчислює вираз *expr*, навіть якщо він містить відкладені команди, наприклад:

```
> Limit(sqrt(n+1)-sqrt(n-1), n=+infinity): %=value(%);
```

$$\lim_{n\to\infty}\left(\sqrt{n+1}-\sqrt{n-1}\right)=0$$

Нагадаємо, що двокрапка наприкінці команди означає, що результат виконання команди не відображатиметься, а знак `%` – це звертання до результату виконання попередньої команди (працює навіть тоді, коли результат не виводиться).

Наведемо приклади відшукання однобічних границь:

```
> Limit(arctan(1/(1-x)), x=1,left): %=value(%);
```

$$\lim_{x\to1^-}\arctan\left(\frac{1}{1-x}\right)=\frac{1}{2}\,\pi$$

```
> Limit(arctan(1/(1-x)), x=1,right): %=value(%);
```

$$\lim_{x\to 1^+} \arctan\left(\frac{1}{1-x}\right) = -\frac{1}{2}\,\pi$$

Команду `limit` можна використовувати також для обчислення границь нескладних функцій багатьох аргументів. Наприклад,

```
> limit((x+y)/(x^2+y^2), {x=5, y=infinity});
```

$$0$$

§ 6.2. Суми, ряди і добутки

Для знаходження скінченних і нескінченних сум $\sum_{n=a}^{b} S(n)$ (у другому випадку $b=\infty$) у Maple існують команди `sum` і `Sum` прямої та відкладеної дії відповідно. Формат обох команд однаковий, наприклад, `sum(`f`,`n`=`a`..`b`)`, де f – вираз, залежний від індексу підсумовування, $a..b$ – межі індексу підсумовування. Для обчислення суми ряду замість верхньої межі потрібно ввести `infinity`.

Знайдемо, наприклад, суму кубів всіх натуральних чисел від 1 до 1000:

```
> Sum(k^3, k=1..1000)=sum(k^3, k=1..1000);
```

$$\sum_{k=1}^{1000} k^3 = 250500250000$$

Наведені команди знаходження сум можна використовувати також до функціональних рядів. Знайдемо, наприклад, суму ряду $\sum_{n=1}^{\infty} (-1)^n\, n^2 x^n$:

```
> Sum((-1)^n*n^2*x^n, n=1..infinity): %=value(%);
```

$$\sum_{n=1}^{\infty} (-1)^n\, n^2 x^n = \frac{x\,(x-1)}{(x+1)^3}$$

Аналогічно обчислюють скінченні та нескінченні добутки $\prod_{n=a}^{b} P(n)$ за допомогою команд `product(`f`,`n`=`a`..`b`)` і `Product(`f`,`n`=`a`..`b`)` (прямої і відкладеної дії відповідно).

Знайдемо, наприклад, добуток всіх непарних натуральних чисел від 1 до 30:

```
> Product(2*n-1,n=1..15)=product(2*n-1,n=1..15);
```

$$\prod_{n=1}^{15}(2\,n-1)=6190283353629375$$

Знайдемо $\prod_{n=2}^{\infty}\frac{n^3+1}{n^3-1}$:

```
> product((n^3+1)/(n^3-1), n=2..infinity);
```

$$\frac{3}{2}$$

§ 6.3. Похідні функцій однієї змінної

Для знаходження похідної функції однієї змінної в Maple є дві команди:

1) команда прямої дії `diff(`f`,`x`)`, де f – функція, яка диференціюється за змінною x;

2) команда відкладеної дії `Diff(`f`,`x`)` з тими самими параметрами.

Для обчислення значення похідної функції $f(x)$ у заданій точці $x=a$ можна скористатись командою `eval(diff(f(x),x), x=a)`.

Наведемо приклади:

```
> Diff(arcsin(x)^2,x)=diff(arcsin(x)^2,x);
```

$$\frac{\mathrm{d}}{\mathrm{d}x}\left(\left(\arcsin\left(x\right)\right)^2\right)=2\,\frac{\arcsin\left(x\right)}{\sqrt{1-x^2}}$$

```
> Diff(x^x,x)=diff(x^x,x);
```

$$\frac{\mathrm{d}}{\mathrm{d}x}\left(x^x\right) = x^x\left(\ln\left(x\right)+1\right)$$

```
> eval(%, x=1);
```

$$\left.\frac{\mathrm{d}}{\mathrm{d}x}\left(x^x\right)\right|_{x=1} = 1$$

Для знаходження похідних вищих порядків у командах `diff` або `Diff` другим параметром потрібно вказати $x\$n$, де n – порядок похідної, або змінну x повторити n разів: `diff(`$f, x, x, \ldots, x$`)` чи `Diff(`$f, x, x, \ldots, x$`)`.

Отриманий результат диференціювання залежно від потрібного вигляду можна спростити за допомогою команд `simplify`, `factor` або `expand`, які вивчались у §§ 5.3, 5.7. Наприклад:

```
> Diff(sin(x)^4, x$3)=diff(sin(x)^4, x$3);
```

$$\frac{d^3}{dx^3}\left(\left(\sin\left(x\right)\right)^4\right) = 24\,\sin\left(x\right)\left(\cos\left(x\right)\right)^3 - 40\,\left(\sin\left(x\right)\right)^3\cos\left(x\right)$$

```
> simplify(%);
```

$$\frac{d^3}{dx^3}\left(\left(\sin\left(x\right)\right)^4\right) = 8\,\sin\left(x\right)\cos\left(x\right)\left(8\,\left(\cos\left(x\right)\right)^2 - 5\right)$$

```
> combine(%);
```

$$\frac{d^3}{dx^3}\left(\frac{3}{8}+\frac{1}{8}\,\cos\left(4\,x\right)-\frac{1}{2}\,\cos\left(2\,x\right)\right) = 8\,\sin\left(4\,x\right)-4\,\sin\left(2\,x\right)$$

Для знаходження похідних можна використовувати також диференціальний оператор `D(`f`)`, де f – функція:

```
> D(cos);
```

$$-\sin$$

Знайдемо, наприклад, похідну функції $y = \cos x$ у точці $x = \pi/2$:

```
> D(cos)(Pi/2): eval(%);
```

$$-1$$

Оператор диференціювання `D` можна застосовувати до створених користувачем функцій (§ 3.2):

```
> f:=x->log[2](x)^2+exp(sqrt(x));
```

$$f := x \to \log_2(x)^2 + \mathrm{e}^{\sqrt{x}}$$

```
> D(f);
```

$$x \to \frac{2\log_2(x)}{x\ln(2)} + \frac{1}{2}\frac{\mathrm{e}^{\sqrt{x}}}{\sqrt{x}}$$

Оператор `D` у форматі `(D@@`n`)(`f`)` можна використовувати для знаходження похідних n-го порядку. Знайдемо, наприклад, третю похідну функції $y(x) = \cos(e^{2x})$:

```
> f:=x->cos(exp(2*x)): (D@@3)(f);
```

$$x \to 8\sin(\mathrm{e}^{2\,x})\,(\mathrm{e}^{2\,x})^3 - 24\cos(\mathrm{e}^{2\,x})\,(\mathrm{e}^{2\,x})^2 - 8\sin(\mathrm{e}^{2\,x})\,\mathrm{e}^{2\,x}$$

§ 6.4. Частинні похідні

Для знаходження частинних похідних функції f багатьох змінних використовується команда `diff` у форматі: `diff(`$f, x_1, x_2, \ldots, x_m$`)`, де x_1, x_2, …, x_m – змінні, за якими здійснюється диференціювання. Замість n однакових змінних x можна писати x`$`n. Наприклад, частинну похідну $\frac{\partial^3 f}{\partial x^2 \partial y}$ записуємо у вигляді `diff(f,y,x,x)` або `diff(f,y,x$2)`.

В якості прикладу знайдемо частинні похідні f''_{x^2} і f''_{xy} функції $f = \operatorname{arctg}(x^2 y)$:

```
> f:=arctan(x^2*y): Diff(f,x$2)=diff(f,x$2);
```

$$\frac{\partial^2}{\partial x^2}\arctan\left(x^2y\right)=\frac{2\,y}{1+x^4y^2}-\frac{8\,x^4y^3}{\left(1+x^4y^2\right)^2}$$

```
> simplify(%);
```

$$\frac{\partial^2}{\partial x^2}\arctan\left(x^2y\right)=-2\,\frac{y\left(-1+3\,x^4y^2\right)}{\left(1+x^4y^2\right)^2}$$

```
> Diff(f,x,y)=simplify(diff(f,x,y));
```

$$\frac{\partial^2}{\partial x\partial y}\arctan\left(x^2y\right)=-2\,\frac{x\left(-1+x^4y^2\right)}{\left(1+x^4y^2\right)^2}$$

Для обчислення значення частинної похідної у заданій точці можна скористатися командою `eval`, наприклад:

```
> Diff(x^y,x,y)=diff(x^y,x,y);
```

$$\frac{\partial^2}{\partial y\,\partial x}\,x^y=\frac{x^y\ln(x)\,y}{x}+\frac{x^y}{x}$$

```
> eval(%,[x=exp(1),y=1]);
```

$$\left.\frac{\partial^2}{\partial y\,\partial x}\,x^y\right|_{x=\mathrm{e},\,y=1}=2$$

§ 6.5. Неперервність функцій та точки розриву

Перевірити на неперервність (відсутність розривів) функцію $f(x)$ на проміжку $(a,\,b)$ можна за допомогою команди `iscont`. Її зазвичай записують у форматі `iscont(`f`,`x`=`a`..`b`)`. Якщо неперервність функції $f(x)$ перевіряється на всій числовій осі, задають інтервал x`=-infinity..+infinity`. Якщо функція $f(x)$ є неперервною на заданому проміжку, то у робочому полі з'явиться відповідь `true`, інакше – `false`.

Розглянемо приклади:

```
> iscont(x/sin(x), x=1..2);
```

true

```
> iscont(x/sin(x), x=1..4);
```

false

За замовчуванням вважається, що неперервність у крайніх точках проміжку (a, b) не перевіряється. В іншому випадку використовують опцію '`closed`' (разом з апострофами). Приклади:

```
> iscont(1/x, x=0..1);
```

true

```
> iscont(1/x, x=0..1, 'closed');
```

false

Точки розриву функції $f(x)$ на заданому проміжку можна знайти за допомогою команд `discont`(f, x) або `singular`(f, x), причому першу з них можна використовувати для знаходження точок розриву першого і другого родів, а другу – точок розриву другого роду (дійсних і комплексних значень). Обидві команди видають результат у вигляді множини (тип `set`). Для подальшого використання отриманих значень точок розриву потрібно з типу `set` за допомогою команди `convert` перевести їх у числовий тип. Команда `discont` може знайти зайві точки.

Знайдемо точки розриву функцій $f(x) = \frac{1}{x^3-2x^2-x+2}$ і $g(x) = \operatorname{tg} \frac{x}{x-3}$:

```
> f:=1/(x^3-2*x^2-x+2): discont(f,x);
```

$$\{-1, 1, 2\}$$

```
> g:=tan(x/(x-1)): singular(g,x);
```

$$\{x = 1\},\ \left\{x = \frac{\pi\,(2\,\_Z2\sim + 1)}{2\,\pi\,\_Z2\sim + \pi - 2}\right\}$$

Тут $\_Z2\sim$ – нова згенерована змінна, яка може набувати лише цілих значень (детальніше див. на стор. 123). Таким чином, функція $f(x)$ має розриви у точках $x = \pm 1$, $x = 2$, а функція $g(x)$ – у точках $x = 1$, $x = \frac{\pi(2k+1)}{\pi(2k+1)-2}$, де $k \in \mathbb{Z}$.

§ 6.6. Екстремуми, найбільше і найменше значення функцій однієї змінної

Для дослідження функції $f(x)$ на екстремуми використовують команду `extrema(`f`,{`$cond$`},`x`,'`s`')`, де `{cond}` (у фігурних дужках) – обмеження для незалежної змінної x, а s – це змінна, якій будуть присвоєні координати критичних точок (точок, у яких $f'(x)$ дорівнює нулю). Якщо фігурні дужки залишити порожніми, то пошук екстремуму здійснюватиметься на всій числовій осі. Команда `extrema` виводить у вигляді множини значення функції в точках, підозрілих на екстремум. Достатні умови екстремуму не перевіряються. Для того, щоб побачити самі точки, в яких може бути екстремум, потрібно вивести значення додаткової змінної s.

Знайдемо критичні точки функцій $f(x) = ax^2 + bx + c$ і $g(x) = 2\operatorname{arctg} x - \ln(x^2 + 1)$:

```
> f:=a*x^2+b*x+c: g:=2*arctan(x)-ln(x^2+1):
> extrema(f,{},x, 's'); s;
```

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

```
> y0:=extrema(g,{},x,'x0'): print(x0,y0);
```

$$\{\{x=1\}\},\ \left\{\frac{1}{2}\,\pi-\ln(2)\right\}$$

Для функції $g(x)$ спершу наведено точку, в якій може бути екстремум, а потім – значення функції у цій точці.

Команда `extrema` має ще один суттєвий недолік: вона не визначає тип екстремуму (мінімум чи максимум).

Для відшукання найбільшого значення функції $f(x)$ на проміжку $x\in[a,b]$ використовують команду `maximize(`f,x`=`a`..`b`)`, а для визначення найменшого значення – команду `minimize(`f,x`=`a`..`b`)`. Якщо не вказати інтервал зміни x, то пошук максимумів і мінімумів здійснюватиметься на всій числовій осі. Якщо через кому вказати опцію `location` (або `location=true`), то у рядку виведення після найбільшого (найменшого) значення функції у фігурних дужках будуть вказані координати точки, в якій це значення досягається.

Наведемо приклади:

```
> f:=cos(x)+2*sin(x):
> M:=maximize(f,x); m:=minimize(f,x);
```

$$M:=\sqrt{5}$$
$$m:=-\sqrt{5}$$

```
> M1:=maximize(x*exp(-x));
```

$$M1:=\mathrm{e}^{-1}$$

```
> M2:=maximize(x*exp(-x), x=4..10);
```

$$M2:=4\,\mathrm{e}^{-4}$$

```
> maximize(x*exp(-x), location);
```

$$\mathrm{e}^{-1},\left\{[\{x=1\},\mathrm{e}^{-1}]\right\}$$

```
> minimize(cos(x)+2*sin(x),location);
```

$$-\sqrt{5}, \left\{\left[\{x = \arctan(2) + \pi(2\_Z2 + 1)\}, -\sqrt{5}\right]\right\}$$

Тут *_Z2~* – нова згенерована змінна, яка може набувати лише цілих значень (детальніше див. на стор. 123).

§ 6.7. Локальний та умовний екстремуми функцій багатьох змінних

Для відшукання локальних та умовних екстремумів функції $f(x_1, \ldots, x_m)$ використовують команду

$$\texttt{extrema}(f, \{cond\}, \{x_1, \ldots, x_m\}, 's'),$$

де *cond* – записані у вигляді рівностей обмеження для пошуку умовного екстремуму, s – змінна, якій буде присвоєно координати точок екстремуму. Якщо обмежень не вказати, то відбуватиметься пошук локального екстремуму.

Команда `extrema` має суттєвий недолік: вона видає усі критичні точки, тобто й ті, у яких екстремуму немає. Вилучити критичні точки, у яких немає екстремуму, можна з допомогою безпосередньої підстановки цих точок у функцію, наприклад, за допомогою команди `eval`.

Дослідимо на екстремум функцію

$$f(x, y) = x^4 + 2y^4 - 2x^2 - y^2 :$$

```
> f:=x^4+2*y^4-2*x^2-y^2:
> extrema(f,{},{x,y},'s'); s;
```

$$\left\{0, -\frac{9}{8}\right\}$$

$$\left\{\{x = -1, y = 0\}, \left\{x = -1, y = -\frac{1}{2}\right\}, \left\{x = -1, y = \frac{1}{2}\right\},\right.$$

$$\{x=0,y=0\},\left\{x=0,y=-\frac{1}{2}\right\},\left\{x=0,y=\frac{1}{2}\right\},$$
$$\{x=1,y=0\},\left\{x=1,y=-\frac{1}{2}\right\},\left\{x=1,y=\frac{1}{2}\right\}\Big\}$$

Одержали лише два екстремуми 0 і $-9/8$, тому, очевидно, $f_{max}=0$, $f_{min}=-9/8$, причому максимум досягається в точці $(0,0)$. Інші критичні точки потрібно перевірити. Враховуючи парність функції f відносно своїх змінних, обмежимося перевіркою критичних точок з невід'ємними координатами:

```
> eval(f,[x=0,y=1/2]);
```

$$-\frac{1}{8}$$

```
> eval(f,[x=1,y=0]);
```

$$-1$$

```
> eval(f,[x=1,y=1/2]);
```

$$-\frac{9}{8}$$

Отже, задана функція має такі локальні екстремуми:

$$f_{max}=f(0,0)=0,\quad f_{min}=f\left(\pm 1,\pm 1/2\right)=-9/8.$$

Найбільше і найменше значення функції $f(x_1,\ldots,x_n)$ знаходимо за допомогою команд `maximize` і `minimize` відповідно, записаних у таких форматах:

$$\texttt{maximize}(f,x_1{=}a_1..b_1,\ldots,x_n{=}a_n..b_n),$$
$$\texttt{minimize}(f,x_1{=}a_1..b_1,\ldots,x_n{=}a_n..b_n).$$

Для визначення точки, в якій досягається це значення, додатково використовують опцію `location`.

Знайдемо, наприклад, найбільше і найменше значення функції $f(x,y)=x^2+2xy-2x+4y$ у прямокутнику D, утвореному прямими $x=1$, $x=5$, $y=0$, $y=3$:

```
> f:=x^2+2*x*y-2*x+4*y:
> maximize(f,x,y, x=1..5, y=0..3);
```

$$57$$

```
> minimize(f,x,y, x=1..5, y=0..3,location);
```

$$-1,\ \{[\{x=1, y=0\}, -1]\}$$

Таким чином, задана функція у прямокутнику D має найбільше значення $f_{max}=57$ і найменше значення $f_{min}=-1$, причому останнє досягається в точці $(1;0)$.

Розглянемо ще один приклад: знайдемо умовні екстремуми функції $f(x,y,z)=xy+2yz$, якщо $x^2+y^2=2$, $y+z=2$, $x>0$, $y>0$, $z>0$:

```
> f:=x*y+2*y*z: assume(x>0, y>0, z>0):
> extrema(f, {x^2+y^2=2, y+z=2}, {x,y,z}, 's');
```

$$\left\{3, -\frac{49}{5}\right\}$$

```
> s;
```

$$\left\{\{x\sim=-1, y\sim=1, z\sim=1\}, \{x\sim=1, y\sim=1, z\sim=1\}, \left\{x\sim=\frac{1}{5}, y\sim=-\frac{7}{5}, z\sim=\frac{17}{5}\right\}\right\}$$

```
> eval(f,s[1]), eval(f,s[2]), eval(f,s[3]);
```

$$1, 3, -\frac{49}{5}$$

Таким чином, задана функція має два умовні екстремуми: $f_{max}=f(1,1,1)=3$ і $f_{min}=f\left(\frac{1}{5},-\frac{7}{5},\frac{17}{5}\right)=-\frac{49}{5}$.

Про дослідження на екстремум лінійної функції багатьох змінних за певних обмежень, заданих лінійними рівностями чи нерівностями, йтиметься в § 17.2.

§ 6.8. Первісна і визначений інтеграл

Невизначений інтеграл $\int f(x)\,dx$ у Maple можна записати і знайти за допомогою команди відкладеної дії `Int(`f,x`)` і команди прямої дії `int(`f,x`)` відповідно, де f – функція, x – змінна інтегрування. Розглянемо приклади:

```
> y:=tan(x)^2: Int(y,x)=int(y,x);
```

$$\int \tan(x)^2\,\mathrm{d}x = \tan(x) - x$$

```
> Int(cos(a*x)*exp(b*x), x)=int(cos(a*x)*exp(b*x), x);
```

$$\int \cos(a\,x)\mathrm{e}^{b\,x}\,\mathrm{d}x = \frac{b\,\mathrm{e}^{b\,x}\cos(a\,x)}{b^2+a^2} + \frac{a\,\mathrm{e}^{b\,x}\sin(a\,x)}{b^2+a^2}$$

Звертаємо увагу на те, що в аналітичному представленні невизначених інтегралів відсутня довільна стала C (тобто програма знаходить насправді первісну), що може призвести до помилок під час знаходження повторних інтегралів.

Для обчислення визначеного інтеграла $\int\limits_a^b f(x)\,dx$ у командах `int` і `Int` потрібно додати межі інтегрування у вигляді діапазону x=a`..`b, причому числа a і b можуть бути й невласними. Наведемо приклади:

```
> int((1+x)*cos(x)*sin(2*x),x=0..Pi);
```

$$\frac{4}{3} + \frac{2}{3}\,\pi$$

```
> Int(x^2*log(2*x), x=0..1)=int(x^2*log(2*x), x=0..1);
```

$$\int_0^1 x^2 \ln(2\,x)\,\mathrm{d}x = \frac{1}{3}\,\ln(2) - \frac{1}{9}$$

```
> p:=infinity: Int(exp(-x^2), x=-p..p): %=value(%);
```

$$\int_{-\infty}^{\infty} \mathrm{e}^{-x^2}\,\mathrm{d}x = \sqrt{\pi}$$

```
> Int((x-1)/ln(x), x=0..1);
```

$$\int_0^1 \frac{x-1}{\ln(x)}\,\mathrm{d}x$$

```
> value(%);
```

$$\ln(2)$$

Звертаємо увагу на те, що два останні інтеграли є невласними: першого і другого роду відповідно.

Додаткова опція `CauchyPrincipalValue` дозволяє обчислювати невласні інтеграли першого і другого роду в сенсі головного значення Коші:

```
> Int(x^(-4), x=-1..3)=int(x^(-4), x=-1..3);
```

$$\int_{-1}^3 \frac{1}{x^4}\,\mathrm{d}x = \infty$$

```
> Int(x^(-4), x=-1..3)=int(x^(-4), x=1..3,
  CauchyPrincipalValue);
```

$$\int_{-1}^3 \frac{1}{x^4}\,\mathrm{d}x = \frac{26}{81}$$

Якщо визначений інтеграл «не береться», то його можна спробувати знайти наближено за допомогою команди `evalf(int(`f`,`x`=`a`..`b`),`s`)`, де s – точність обчислення (кількість значущих цифр), наприклад

```
> y:=sin(x)/x: Int(y,x=1..2)=evalf(int(y,x=1..2),5);
```

$$\int_1^2 \frac{\sin(x)}{x}\,\mathrm{d}x = 0.65932$$

Про обчислення кратних, криволінійних і поверхневих інтегралів йтиметься у розділі 13.

§ 6.9. Формула Тейлора

Для розвинення функції $f(x)$ за формулою Тейлора у точці $x = a$ до членів порядку n використовують команду `taylor(`f, x`=`a, n`)`. Якщо останній параметр не вказано, то кількість членів розвинення визначається значенням системної змінної `Order` (за замовчуванням `Order=6`). Замість x`=0` можна писати просто x. Приклади:

```
> taylor(sin(Pi*cos(x)),x);
```

$$\frac{1}{2}\,\pi\,x^2 - \frac{1}{24}\,\pi\,x^4 + \mathrm{O}\left(x^6\right)$$

```
> f:=taylor(1/x,x=3,4);
```

$$f := \frac{1}{3} - \frac{1}{9}\,(x-3) + \frac{1}{27}\,(x-3)^2 - \frac{1}{81}\,(x-3)^3 + \mathrm{O}\left((x-3)^4\right)$$

Результат розвинення за формулою Тейлора має тип `series`. Для перетворення виразу *ser* цього типу в многочлен використовують команду `convert(`*ser*`,polynom)`, наприклад:

```
> convert(f,polynom);
```

$$\frac{1}{3} - \frac{1}{9}\,(x-3) + \frac{1}{27}\,(x-3)^2 - \frac{1}{81}\,(x-3)^3$$

Команда `series(`f, x`=`a, n`)` виконує розвинення функції $f(x)$ в узагальнений степеневий ряд у точці $x = a$, де n – порядок розвинення. В узагальненому степеневому ряді можуть бути степені як з додатними, так і з від'ємними показниками, тобто він може виявитись рядом Лорана. Приклади:

```
> series((x+3)/(x^2+1),x=infinity);
```

$$\frac{1}{x} + \frac{3}{x^2} - \frac{1}{x^3} - \frac{3}{x^4} + \frac{1}{x^5} + \mathrm{O}\left(\frac{1}{x^6}\right)$$

```
> series(64/(z^4-1),z=I,4);
```

$$\frac{16\,\mathrm{I}}{z-\mathrm{I}} - 24 - 20\,\mathrm{I}\,(z-\mathrm{I}) + 10\,(z-\mathrm{I})^2 + \mathrm{I}\,(z-\mathrm{I})^3 + \mathrm{O}\left((z-\mathrm{I})^4\right)$$

Розвинення функції $f(x_1,\ldots,x_m)$ багатьох змінних у ряд Тейлора у точці $(a_1,\ldots,a_m)$ знаходять за допомогою команди

$$\texttt{mtaylor(}f\texttt{, [}x_1\texttt{=}a_1,\ldots,x_m\texttt{=}a_m\texttt{], }n\texttt{, }w\texttt{)},$$

де n – порядок розвинення (максимальна сума степенів усіх змінних), а w – список ваг змінних. Зокрема, якщо вага змінної x дорівнює 2, то в розвиненні будуть використовуватись члени степеня не вище $x^{\left[\frac{n}{2}\right]}$. Два останніх аргументи команди `mtaylor` є необов'язковими. Приклади:

```
> mtaylor(exp(x)*sin(y),[x=ln(2),y=Pi],4);
```

$$-2\,y+2\,\pi-2\,(x-\ln(2))\,(y-\pi)+\frac{1}{3}\,(y-\pi)^3-(x-\ln(2))^2\,(y-\pi)$$

```
> mtaylor(exp(x)*sin(y),[x=ln(2),y=Pi],4,[1,2]);
```

$$-2\,y+2\,\pi-2\,(x-\ln(2))\,(y-\pi)$$

Питання до розділу 6

1. Яка різниця між командами прямої і відкладеної дії?

2. Як знайти границю послідовності, границю функції та відобразити їх у вигляді рівності «границя=значення»? Як знайти однобічні границі функції?

3. Як знайти скінченну суму, суму ряду, скінченний і нескінченний добуток? Як відобразити їх у вигляді рівностей?

4. Які команди призначені для відшукання похідних функцій однієї і багатьох змінних? Чим відрізняється команда `diff` від оператора `D`? Як відобразити знайдені похідні у вигляді рівностей?

5. Як віднайти екстремуми функцій однієї і багатьох змінних? Як шукати умовний екстремум? Як визначити найменше і найбільше значення функції?

6. Як знайти невизначений і визначений інтеграл?

7. Як подати функцію однієї змінної (кількох змінних) за формулою Тейлора? Як розвинути функцію в узагальнений степеневий ряд?

Вправи до розділу 6

1. Знайдіть границю $\lim\limits_{x\to 0}\left(\frac{2^x+3^x+7^x}{3}\right)^{\frac{1}{x}}$, відобразивши результат за допомогою рівності.

2. Знайдіть частинну суму та суму ряду

$$\sum_{n=1}^{\infty}\frac{1}{(3n-1)(3n+2)}.$$

3. Знайдіть добуток

$$\prod_{n=3}^{50}\frac{n^2-7n+5}{n^2+2n+1}.$$

4. Знайдіть частинні похідні першого і другого порядку функції $z=x^3y\ln(xy+y^4+1)$ і відобразіть результати за допомогою рівностей.

5. Дослідіть функцію $z=(5x+7y-25)e^{-(x^2+xy+y^2)}$ на екстремум.

6. В крузі $x^2+y^2\leqslant 9$ знайдіть найбільше і найменше значення функції $z=x^2+y^2+5x-4y+2$, врахувавши, що такі значення можуть досягатись лише в точці екстремуму всередині області або на межі області (останній випадок зводиться до умовного екстремуму).

7. Знайдіть інтеграли

$$\int\frac{\sqrt{x^2+4x-7}+x}{3x^2+2x-5}\,dx,\qquad \int\limits_5^{10}\frac{\sqrt{x^2+4x-7}+x}{3x^2+2x-5}\,dx,$$

відобразивши результати за допомогою рівностей.

8. Запишіть за формулою Тейлора за степенями $x-3$ функцію $y=\sqrt{x+1}$ до члена сьомого порядку включно.

9. Знайдіть перші десять членів розвинення функції $y=\frac{1}{\ln^2 x}$ в узагальнений степеневий ряд за степенями $x-1$.

Розділ 7. Розв'язування рівнянь, нерівностей та їх систем у Maple

§ 7.1. Команда `solve` для розв'язування рівнянь та систем рівнянь

Для точного розв'язування алгебричних і трансцендентних рівнянь та їх систем призначена універсальна команда

`solve(`*eqs*, *vars*`)`,

де *eqs* – рівняння, вираз або множина рівнянь чи виразів, а *vars* – змінна, відносно якої потрібно розв'язати рівняння, або множина таких змінних. Множину рівнянь задають для розв'язування системи рівнянь.

Команда `solve` знаходить усі корені рівняння у вигляді послідовності чисел чи виразів. У випадку системи кожен член послідовності у відповіді містить один розв'язок системи, який задається множиною рівностей *змінна=значення*.

Якщо замість рівняння задано вираз, то вважається, що задано рівняння, ліва частина якого є цим виразом, а права дорівнює нулю. Якщо змінну (змінні) *vars* не вказувати, то рівняння чи систему буде розв'язано відносно всіх невідомих. Якщо невідомих більше, ніж рівнянь, то частину невідомих буде виражено через інші. Для перевірки розв'язків рівняння чи системи можна використати команду `eval`. Приклади:

```
> f:=6^x-4*2^(x-2)=8*2^x; solve(f,x);
```

$$f := 6^x - 4\,2^{x-2} = 8\,2^x$$

$$2$$

```
> solve(a*x^2+3*x-5,x);
```

$$\frac{1}{2}\,\frac{-3+\sqrt{9+20\,a}}{a},\ -\frac{1}{2}\,\frac{3+\sqrt{9+20\,a}}{a}$$

```
> solve({x+y=3,x-2*y=5},{x,y});
```

$$\left\{x = \frac{11}{3}, y = -\frac{2}{3}\right\}$$

```
> S:={x^2+y^2=17,x+y=5}: p:=solve(S,{x,y});
```

$$p := \{x = 4, y = 1\}, \{x = 1, y = 4\}$$

```
> eval(S, p[1]), eval(S, p[2]);
```

$$\{5 = 5, 17 = 17\}, \{5 = 5, 17 = 17\}$$

За замовчуванням тригонометричні рівняння Maple розв'язує на проміжку $[-\pi, \pi]$. Для одержання всіх розв'язків тригонометричних рівнянь потрібно системній змінній `_EnvAllSolutions` надати значення `true`. Приклади:

```
> f:=sin(x)^2+sin(2*x)^2=sin(3*x)^2;
```

$$f := \sin(x)^2 + \sin(2\,x)^2 = \sin(3\,x)^2$$

```
> solve(f);
```

$$0,\ \pi,\ \frac{1}{2}\,\pi,\ -\frac{1}{2}\,\pi,\ \frac{1}{6}\,\pi,\ \frac{5}{6}\,\pi,\ -\frac{1}{6}\,\pi,\ -\frac{5}{6}\,\pi$$

```
> _EnvAllSolutions:=true: solve(f);
```

$$2\,\pi\_Z1\sim,\ \pi + 2\,\pi\_Z2\sim,\ \frac{1}{2}\,\pi + 2\,\pi\_Z3\sim,\ -\frac{1}{2}\,\pi + 2\,\pi\_Z4\sim,$$
$$\frac{1}{6}\,\pi + 2\,\pi\_Z5\sim,\ \frac{5}{6}\,\pi + 2\,\pi\_Z5\sim,$$
$$-\frac{1}{6}\,\pi + 2\,\pi\_Z6\sim,\ -\frac{5}{6}\,\pi + 2\,\pi\_Z6\sim$$

В останньому прикладі розв'язки побудовані з використанням згенерованих системних змінних *_Z1~*, *_Z2~*, ..., *_Z6~*, де знак тильда (~) означає, що на змінні накладено

обмеження (у даному випадку вони можуть набувати значень цілих чисел, про це свідчить префікс $\_Z$ в іменах змінних). Можуть також генеруватись змінні з префіксом $\_NN$ (невід'ємні цілі) і $\_B$ (бінарні числа, які можуть набувати лише значень 0 і 1).

Алгебричні рівняння п'ятого і вищих степенів можуть не мати розв'язків, які подаються через радикали. Так само, далеко не завжди можна знайти точні розв'язки трансцендентного рівняння. У цих випадках Maple для подання результатів використовує спеціальну функцію `RootOf`:

```
> s:=solve(x^5-2*x^4+6*x^2+2=0);
```

$$\begin{aligned} s := \mathit{RootOf}\big(\_Z^5 - 2\,\_Z^4 + 6\,\_Z^2 + 2, \mathit{index} = 1\big), \\ \mathit{RootOf}\big(\_Z^5 - 2\,\_Z^4 + 6\,\_Z^2 + 2, \mathit{index} = 2\big), \\ \mathit{RootOf}\big(\_Z^5 - 2\,\_Z^4 + 6\,\_Z^2 + 2, \mathit{index} = 3\big), \\ \mathit{RootOf}\big(\_Z^5 - 2\,\_Z^4 + 6\,\_Z^2 + 2, \mathit{index} = 4\big), \\ \mathit{RootOf}\big(\_Z^5 - 2\,\_Z^4 + 6\,\_Z^2 + 2, \mathit{index} = 5\big) \end{aligned}$$

```
> evalf(s[2]);
```

$$-0.006334728273 + 0.5499341421\,\mathrm{I}$$

```
> solve(x+3=cos(x));
```

$$\mathit{RootOf}(\_Z + 3 - \cos(\_Z))$$

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

$$-3.794388613$$

Опція `index` з цілочисловим значенням використовується для впорядкування коренів рівняння. Функція `evalf` обчислює наближені числові значення функції `RootOf`, як це було зроблено в останніх прикладах. Нагадаємо, що індексна форма дає доступ до конкретного члена послідовності (в нашому випадку – до конкретного розв'язку рівняння).

Розв'язок системи може бути записаний через функцію `RootOf` навіть тоді, коли можна знайти точні розв'язки. У цьому випадку точні розв'язки легко отримати за допомогою команди `allvalues(`*expr*`)`, де *expr* – вираз, матриця, список чи множина виразів. Приклад:

```
> solve({x^2+y^2=4, x-3*y=3});
```

$$\{x = 3\,RootOf\left(10\_Z^2 + 18\_Z + 5, label = \_L2\right) + 3,$$
$$y = RootOf\left(10\_Z^2 + 18\_Z + 5, label = \_L2\right)\}$$

```
> aa:=allvalues(%);
```

$$aa := \left\{x = \frac{3}{10} + \frac{3}{10}\sqrt{31},\ y = -\frac{9}{10} + \frac{1}{10}\sqrt{31}\right\},$$
$$\left\{x = \frac{3}{10} - \frac{3}{10}\sqrt{31},\ y = -\frac{9}{10} - \frac{1}{10}\sqrt{31}\right\}$$

Для присвоєння невідомим знайдених значень використовують команду `assign(`t`)`, де t – послідовність, список чи множина рівностей вигляду *невідома=значення*. Фактично команда `assign` замінює у множині, списку чи послідовності знак рівності (`=`) на знак присвоювання (`:=`). Продовжуючи приклад, маємо:

```
> assign(aa[1]); 3*x+y;
```

$$\sqrt{31}$$

Для розв'язування рівнянь потрібно, щоб невідомим, які в них входять, попередньо не було присвоєно жодних значень. Нагадаємо, що для того щоб позбутись вже виконаних присвоювань, потрібно присвоїти невідомим їх імена в одинарних лапках. Наприклад:

```
> x:=5: solve(2*x+4=0,x);
Warning, solving for expressions other than names or functions is
not recommended.
```

```
Error, (in solve) a constant is invalid as a variable, 5
> x:='x': solve(2*x+4=0);
```

$$-2$$

Треба мати на увазі, що Maple розв'язує рівняння над полем комплексних чисел. Тому іноді можна отримати «дивні» (у припущенні того, що розв'язок є дійсним) відповіді:

```
> solve(sin(x)^2+4*sin(x)=5);
```

$$\frac{1}{2}\,\pi,\ -\arcsin(5)$$

```
> evalf(%[2]);
```

$$-1.570796327 + 2.292431670\,\mathrm{I}$$

У §5.6 йшла мова про можливість накладання окремих обмежень на невідомі, але для рівнянь ці обмеження не працюватимуть. Для того, щоб програма знаходила лише дійсні корені рівнянь, треба підключити пакет `RealDomain`. Про роботу з пакетами йтиметься у розділі 9.

Відсутність відповіді означає, що розв'язку не існує або програма не може його знайти. Розв'язки трансцендентних рівнянь можуть бути знайдені через спеціальні функції:

```
> solve(exp(-x)=x);
```

$$\mathrm{LambertW}(1)$$

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

$$0.5671432904$$

§ 7.2. Розв'язування нерівностей

Нерівністю є два вирази, поєднані знаками `>=`, `<=`, `>` або `<`. Для розв'язування нерівностей також використовують команду `solve` (§ 7.1). Якщо нерівність чи невідома у команді

`solve` вказується у фігурних дужках (тобто у вигляді множини), то і відповідь буде даватись як множина, інакше відповідь буде представлена за допомогою функцій `RealRange` і `Open` (стор. 89). Перша з цих функцій задає замкнений відрізок дійної осі, а друга вказує на те, що крайня точка не належить розв'язку. Наприклад:

```
> solve(x^3>4*x, x);
```

$$RealRange(Open(-2), Open(0)), RealRange(Open(2), \infty)$$

```
> solve(x^3>4*x, {x});
```

$$\{-2 < x,\ x < 0\},\ \{2 < x\}$$

За допомогою команди `solve` можна також розв'язувати системи нерівностей:

```
> solve({log[3](x)+log[x](3)<10/3, 5^x+5^(-x)>26/5});
```

$$\{x < 27,\ 3^{1/3} < x\}$$

§ 7.3. Команда `fsolve`

Якщо точні розв'язки рівняння або системи знайти неможливо, то для відшукання десяткових наближень коренів використовують команду `fsolve` в одному з форматів

`fsolve(`*eqs*, *vars*`)`, `fsolve(`*eqs*, *vars*, *a*`..`*b*`)`,
`fsolve(`*eqs*, *vars*`=`*a*`..`*b*`)`.

На відміну від команди `solve` у команді `fsolve` є можливість задати проміжок $[a, b]$, що містить шуканий корінь рівняння. Для алгебричних рівнянь Maple знаходить всі дійсні корені, а для інших – один корінь на вказаному проміжку або один довільний (зазвичай найменший) дійсний корінь, якщо проміжок не вказано. Якщо на заданому проміжку розв'язку рівняння чи системи не існує, то результатом виконання

команди `fsolve` буде повторена область введення. Для алгебричних рівнянь замість проміжку можна використовувати опцію `complex`, – тоді будуть знайдені всі комплексні розв'язки рівняння. Приклади:

```
> f:=sin(x)=x/4+1;
```

$$f := \sin(x) = \frac{1}{4}x + 1$$

```
> fsolve(f);
```

$$-7.972119390$$

```
> fsolve(f,x=-7..5);
```

$$-3.313955876$$

```
> fsolve(f,x,-3..100);
```

$$fsolve\left(\sin(x) = \frac{1}{4}\,x + 1,\ x,\ -3..100\right)$$

```
> fsolve(f,x,-100..-8);
```

$$fsolve\left(\sin(x) = \frac{1}{4}\,x + 1,\ x,\ -100..-8\right)$$

```
> fsolve(x^5-4*x^4+2*x^3-6*x^2+7*x-2=0);
```

$$0.4700053142,\ 0.6284163620,\ 3.770918441$$

```
> fsolve(x^5-4*x^4+2*x^3-6*x^2+7*x-2=0,x,complex);
```

$$-0.4346700587 - 1.267579215\,\mathrm{I},\ -0.4346700587 + 1.267579215\,\mathrm{I},$$
$$0.4700053142,\ 0.6284163620,\ 3.770918441$$

```
> fsolve({sin(x+y)=exp(x-y^2),cos(x)=y^5},{x,y});
```

$$\{x = -2.270500701, y = -0.9157481165\}$$

```
> fsolve({sin(x+y)=exp(x-y^2),cos(x)=y^5},
  {x=-100..100,y=0..100});
```

$$\{x = -0.7430079047, y = 0.9406479715\}$$

Про більш потужні засоби наближеного розв'язування рівнянь і систем йтиметься у § 17.3.

§ 7.4. Команди isolve, rsolve, msolve

Команда

`isolve`(*eqs*, *nvars*)

знаходить цілі розв'язки рівняння чи системи рівнянь *eqs*. Необов'язковим параметром *nvars* можна задати нові змінні, які будуть використовуватись замість згенерованих. Розв'язки можуть будуватись з використанням згенерованих цілих змінних з префіксом _*Z*. Приклади:

```
> isolve(x^2-4*y=5);
```

$$\{x = 1 - 4\,\_Z1, y = -1 - 2\,\_Z1 + 4\,\_Z1^2\},$$
$$\{x = 3 - 4\,\_Z1, y = 1 - 6\,\_Z1 + 4\,\_Z1^2\}$$

```
> isolve({x-4*y=5,x+y^3+3*z=6},k);
```

$$\left\{x = 13 + 12\,k, y = 2 + 3\,k, z = -\frac{7}{3} - 4\,k - \frac{1}{3}\,(2 + 3\,k)^3\right\}$$

Якщо цілого розв'язку не існує, то область виведення буде порожньою:

```
> isolve({x+7*y=4,3*x+6*y=4});
```

Команда

`msolve`(*eqs*, *m*)

знаходить цілі розв'язки рівняння чи системи *eqs* за модулем *m* ($m \in \mathbb{N}$). Приклади:

```
> msolve(x^2+4*x-7=0, 5);
```

$$\{x = 2\}, \{x = 4\}$$

```
> msolve(3^i=5, 11);
```

$$\{i = 3 + 5\,\_Z2\}$$

```
> msolve(x^3=5, 7);  # Розв'язку немає
> msolve({x^3=8, 3*x-4*y^2=12}, 12);
```

$$\{x = 8, y = 0\},\ \{x = 8, y = 3\},\ \{x = 8, y = 6\},\ \{x = 8, y = 9\}$$

Подібна за аргументами до команди `solve` команда

`rsolve(`*eqs*, *fns*`)`

знаходить розв'язок рекурентного рівняння *eqs* чи системи таких рівнянь, можливо з початковими умовами, які задаються разом з рівняннями в множині *eqs* (*fns* – шукана функція чи множина функцій). Приклади:

```
> rsolve(f(n+1)=3*f(n)-n^2,f(n));
```

$$f(0)\,3^n - \frac{1}{2} + (n+1)\left(\frac{1}{2}\,n + 1\right) - \frac{1}{2}\,3^n - n$$

```
> rsolve({f(n+1)=g(n),g(n+1)=f(n)+n^4,f(0)=a,g(0)=b},
  {f(n),g(n)});
```

$$\left\{ f(n) = \frac{1}{10}\,n^5 - \frac{1}{2}\,n^4 + \frac{2}{3}\,n^3 - \frac{4}{15}\,n + \right.$$
$$+ \frac{1}{2}\,(-1)^n a - \frac{1}{2}\,(-1)^n b + \frac{1}{2}\,a + \frac{1}{2}\,b,$$
$$\left. g(n) = \frac{1}{10}\,n^5 - \frac{1}{3}\,n^3 + \frac{7}{30}\,n - \frac{1}{2}\,(-1)^n a + \frac{1}{2}\,(-1)^n b + \frac{1}{2}\,a + \frac{1}{2}\,b \right\}$$

Якщо розв'язок рівняння чи системи знайти не вдається, то результатом виконання команди `rsolve` буде повністю повторена область введення.

§ 7.5. Відшукання точних розв'язків звичайних диференціальних рівнянь

Для відшукання розв'язків звичайних диференціальних рівнянь та їх систем у пакеті Maple використовують команду

`dsolve(`*deqs*, *fns*, *opts*`)`,

де *deqs* – звичайне диференціальне рівняння, множина таких рівнянь або множина рівнянь і початкових чи крайових умов, *fns* – невідома функція чи множина невідомих функцій диференціального рівняння або системи (можна не вказувати), *opts* – опція, яка задає форму подання розв'язку і методи його відшукання.

Для задання похідної у диференціальному рівнянні можна використовувати команду `diff` або оператор диференціювання `D` (§ 6.3), причому саму функцію треба записувати з явним вказуванням незалежної змінної, наприклад, $y(x)$. Оператор `D` має такий синтаксис:

`(D@@`n`)(`функція`)(`змінна`)`

У цьому записі n – порядок похідної. Наприклад, $f''(x)$ можна задати як `(D@@2)(f)(x)`. Якщо $n = 1$, то замість першого виразу у дужках можна писати просто `D`.

Команда `dsolve`, як правило, знаходить лише загальний розв'язок і не завжди наводить особливі. Це пов'язано, зокрема, з існуванням різних, не еквівалентних між собою означень особливих розв'язків. Згенеровані величини *_C1*, *_C2* і т. д. позначають довільні сталі. Maple завжди намагається знайти розв'язок диференціального рівняння у явному вигляді. Якщо це зробити не вдається, то буде знайдено розв'язок у неявній формі. Приклади:

```
> eq1:=(2*y(x)-x^2*y(x))*diff(y(x),x)=x-x*y(x)^2;
```

$$eq1 := \left(2\,y(x) - x^2\,y(x)\right)\left(\frac{\mathrm{d}}{\mathrm{dx}}y(x)\right) = x - x\,y(x)^2$$

```
> dsolve(eq1,y(x));
```

$$y(x) = \sqrt{1 - 2\,\_C1 + \_C1\,x^2},\ y(x) = -\sqrt{1 - 2\,\_C1 + \_C1\,x^2}.$$

```
> eq2:=(D@@2)(y)(x)-k^2*y(x)=0;
```

$$eq2 := D^{(2)}(y)(x) - k^2\,y(x) = 0$$

```
> dsolve(eq2);
```

$$y(x) = \_C1\,\mathrm{e}^{-k\,x} + \_C2\,\mathrm{e}^{k\,x}$$

```
> dsolve({D(y)(x)=6*y(x)+z(x),D(z)(x)=-15*y(x)-
  2*z(x)}, {y(x),z(x)});
```

$$\left\{y(x) = \_C1\,\mathrm{e}^{3\,x} + \_\mathrm{C2}\,\mathrm{e}^{x},\ z(x) = -3\,\_C1\,\mathrm{e}^{3\,x} - 5\,\_C2\mathrm{e}^{x}\right\}$$

Якщо потрібно розв'язати задачу Коші чи крайову задачу, то першим параметром команди `dsolve` має бути множина, яка складається з рівняння і початкових чи крайових умов (через кому у фігурних дужках). У цих умовах для задання похідних використовують оператор `D`. Приклади:

```
> dsolve({eq2,y(0)=6,D(y)(0)=10});  # Задача Коші
```

$$y(x) = \frac{(3\,k-5)\,\mathrm{e}^{-k\,x}}{k} + \frac{(5+3\,k)\,\mathrm{e}^{k\,x}}{k}$$

```
> eq3:=diff(y(x),x$2)-diff(y(x),x)=x;
```

$$eq3 := \frac{\mathrm{d}^2}{\mathrm{d}x^2}\,y(x) - \left(\frac{\mathrm{d}}{\mathrm{d}x}\,y(x)\right) = x$$

```
> bconds:=y(0)=0, D(y)(1)=1;  # Крайова задача
```

$$bconds := y(0) = 0, \mathrm{D}(y)(1) = 1$$

```
> dsolve(({eq3, bconds}, y(x)));
```

$$y(x) = -\frac{1}{2}\,x^2 + \frac{3\,\mathrm{e}^{x}}{\mathrm{e}} - x - \frac{3}{\mathrm{e}}$$

Часто Maple знаходить розв'язок диференціального рівняння у явній формі через спеціальні функції. Для зображення розв'язку у неявній формі використовують опцію `implicit`. Опція `parametric` (часто використовується разом з опцією `implicit`) вказує на те, що розв'язок потрібно подати у параметричній формі. Опція `explicit` вимагає подання розв'язку у явній формі. При цьому він може бути записаний через функцію `RootOf`. Опція `useInt` призначена для запису розв'язку у квадратурах (через невизначені інтеграли без їх знаходження). Приклади:

```
> ode:=diff(y(x),x)=(x-y(x)+5)/(2*x-2*y(x));
```

$$ode := \frac{\mathrm{d}}{\mathrm{d}x}\,y(x) = \frac{x - y(x) + 5}{2\,x - 2\,y(x)}$$

```
> dsolve(ode,y(x));
```

$$y(x) = x - 5\,\mathrm{LambertW}\left(-\frac{1}{5}\,\mathrm{e}^{\frac{1}{10}\,x}\_C1\,\mathrm{e}^{-1}\right) - 5$$

```
> dsolve(ode,y(x), implicit);
```

$$-x + 2\,y(x) - 10\ln(-x + y(x) + 5) - \_C1 = 0$$

```
> eq4:=y(x)^2+(x*y(x)+2*y(x)^2-3)*diff(y(x),x)=0;
```

$$eq4 := y(x)^2 + \left(x\,y(x) + 2\,y(x)^2 - 3\right)\left(\frac{\mathrm{d}}{\mathrm{d}x}\,y(x)\right) = 0$$

```
> dsolve(eq4,y(x));
```

$$y(x) = \mathrm{e}^{RootOf\left(3\_Z - x\,\mathrm{e}^{-\_Z} - \left(\mathrm{e}^{-\_Z}\right)^2 + \_C1\right)}$$

```
> dsolve(eq4,y(x), implicit);
```

$$x - \frac{-y(x)^2 + 3\ln(y(x)) + \_C1}{y(x)} = 0$$

```
> eq5:=x+5*D(y)(x)^4+2*D(y)(x)=0; dsolve(eq5, y(x));
```

$$eq5 := x + 5\,\mathrm{D}(y)(x)^4 + 2\,\mathrm{D}(y)(x) = 0$$

$$y(x) = \frac{4}{5}\,x\,RootOf\big(x + 5\,\_Z^4 + 2\,\_Z\big) + \frac{3}{5}\,RootOf\big(x + 5\,\_Z^4 + 2\,\_Z\big)^2 + \_C1$$

```
> dsolve(eq5, y(x), implicit, parametric);
```

$$\big[x(\_T) = 5\,\_T^4 + 2\,\_T,\ y(\_T) = 4\,\_T^5 + \_T^2 + \_C1\big]$$

Команда

`odetest(`*sol*, *deqs*`)`

перевіряє, чи рівність (множина рівностей) *sol* є розв'язком звичайного диференціального рівняння (системи рівнянь) *deqs*. Якщо це справді так, то виводиться значення 0, інакше виводиться вираз, який визначає величину нев'язки (різницю між точним розв'язком і *sol*). У множину *deqs* можна включати також додаткові умови. Приклади:

```
> ode:=D(y)(x)+2*x*y(x)=2*x*exp(-x^2);
```

$$ode := \mathrm{D}(y)(x) + 2\,x\,y(x) = 2\,x\,\mathrm{e}^{-x^2}$$

```
> odetest(y(x)=x^2*exp(-x^2), ode);
```

$$0$$

```
> odetest(y(x)=x*exp(-x^2), ode);
```

$$\mathrm{e}^{-x^2} - 2\,x\,\mathrm{e}^{-x^2}$$

§ 7.6. Наближене розв'язування звичайних диференціальних рівнянь

Якщо зінтегрувати диференціальне рівняння у скінченному вигляді не вдалося, то можна знайти його наближений розв'язок за допомогою ряду Тейлора. Для цього в команді `dsolve` потрібно задати опцію `type=series` або просто `series`, а також вказати, до якого порядку малості треба будувати наближений розв'язок, присвоївши відповідне значення системній змінній `Order`. Якщо відсутні початкові чи крайові умови, то розв'язок будується у вигляді ряду Тейлора в околі точки 0, а його коефіцієнти виражаються через довільні сталі $\_C1$, $\_C2$ і т. д. Приклад:

```
> Order:=7: dsolve({(D@@2)(y)(x)+3*x^3*D(y)(x)-
  y(x)/x=1,y(0)=0,D(y)(0)=1},y(x),series);
```

$$y(x) = x + x^2 + \frac{1}{6}\,x^3 + \frac{1}{72}\,x^4 - \frac{43}{288}\,x^5 - \frac{1771}{8640}\,x^6 + O(x^7)$$

Для відшукання наближеного числового розв'язку задачі Коші чи крайової задачі треба задати опцію `type=numeric` або просто `numeric`. Числовий розв'язок будується у формі процедури, тому результат виконання команди `dsolve` потрібно присвоїти деякій змінній. Надалі ім'я цієї змінної можна використовувати як ім'я процедури для обчислення значення розв'язку задачі Коші чи крайової задачі в деякій точці. Якщо використовується опція `type=numeric`, то диференціальне рівняння не повинно містити жодних невідомих параметрів, крім шуканої функції і незалежної змінної. Приклад:

```
> eq:=(D@@2)(y)(x)-2*sin(x^2)*D(y)(x)+y(x)=x^2;
```

$$eq := \mathrm{D}^{(2)}(y)(x) - 2\sin\left(x^2\right)\mathrm{D}(y)(x) + y(x) = x^2$$

```
> F:=dsolve({eq,y(0)=1,y(3)=5},y(x),type=numeric);
```

$$F := \mathbf{proc}(x\_bvp)\ \ldots\ \mathbf{end\ proc}$$

```
> F(0); F(1);
```

$$\left[x = 0., y(x) = 1., \frac{\mathrm{d}}{\mathrm{dx}} y(x) = 1.3227535683262241 5\right]$$

$$\left[x = 1., y(x) = 1.85001087793757968, \frac{\mathrm{d}}{\mathrm{dx}} y(x) = 0.576053425720693624\right]$$

Для відшукання наближених числових розв'язків можна використовувати також опцію `method=`*met*, де *met* – метод побудови розв'язку. За замовчуванням для розв'язування початкових задач використовується метод Рунге – Кутти четвертого або п'ятого порядку. Найточнішим, але й найповільнішим, є метод `taylorseries` (розвинення у ряд Тейлора). Досить точним є метод `dverk78` (Рунге – Кутти сьомого або восьмого порядку). Можна використовувати і класичні методи, наприклад, класичний метод Ейлера – `classical[foreuler]`. Для розв'язування крайових задач використовується лише метод `bvp`. Кількість вузлів для обчислення розв'язку в останньому випадку обмежується значенням опції `maxmesh`. У деяких випадках його доводиться збільшувати, але не більше, ніж до 8192.

Засоби графічного подання наближених розв'язків звичайних диференціальних рівнянь розглядаються у §§ 11.2, 12.1, 12.2.

§ 7.7. Відшукання точних розв'язків диференціальних рівнянь з частинними похідними

Для відшукання точних розв'язків рівнянь з частинними похідними та їх систем використовується команда

`pdsolve(`*pdeqs*, *fns*, *opts*`)`,

де *pdeqs* – рівняння з частинними похідними або множина таких рівнянь, *fns* – шукана функція чи їх множина (необов'язковий параметр), *opts* – опції. Приклади:

```
> eq1:=(x+2*y)*diff(z(x,y),x)=y*diff(z(x,y),y);
```

$$(x+2\,y)\left(\frac{\partial}{\partial x}\,z(x,y)\right)=y\left(\frac{\partial}{\partial y}\,z(x,y)\right)$$

```
> pdsolve(eq1);
```

$$z(x,y)=\_F1\big(x\,y+y^2\big)$$

```
> pdsolve(diff(u(t,x),t,t)=4*diff(u(t,x),x,x)+x*t);
```

$$u(t,x)=\_F1(x+2\,t)+\_F2(x-2\,t)+\frac{1}{6}\,t^3\,x$$

Команда `pdsolve` використовує префікс $\_F$ з наступним натуральним числом для подання довільних функцій у загальному розв'язку рівняння. Довільні сталі, як і раніше, позначаються через $\_C1$, $\_C2$ і т. д.

Якщо програма не може визначити метод розв'язування диференціального рівняння, то можна використати опцію `HINT=`*hint*, де в якості підказки *hint* може бути `strip` для методу характеристик, `X(x)*T(t)`, `X(x)+T(t)` або подібний вираз для відокремлення змінних. Якщо замість змінної t використовується, наприклад, змінна y, то у підказці *hint* треба писати `Y(y)`. Якщо Maple зможе відокремити змінні, то замість розв'язку вказуються звичайні диференціальні рівняння. Щоб їх одразу зінтегрувати, використовують опцію `INTEGRATE`. Для побудови загального розв'язку рівняння з частинними похідними з використанням методу Фур'є (методу відокремлення змінних) застосовують опцію `build`. Приклад:

```
> eq:=diff(u(t,x),t)=4*x*diff(u(t,x),x,x);
```

$$eq:=\frac{\partial}{\partial t}u(t,x)=4\,x\left(\frac{\partial^2}{\partial x^2}u(t,x)\right)$$

```
> pdsolve(eq);
```

$$(u(t,x) = \_F1(t)\,\_F2(x))\ \&\text{where}\left[\left\{\frac{\mathrm{d}}{\mathrm{dt}}\,\_F1(t) = \_c_1\,\_F1(t),\ \frac{\mathrm{d}^2}{\mathrm{dx}^2}\,\_F2(x) = \frac{1}{4}\,\frac{\_c_1\,\_F2(x)}{x}\right\}\right]$$

```
> pdsolve(eq,INTEGRATE);
```

$$(u(t,x) = \_F1(t)\,\_F2(x))\ \&\text{where}\left[\left\{\left\{\_F1(t) = \_C1\,\mathrm{e}^{-\_c_1\,t}\right\},\ \left\{\_F2(x) = \_C2\sqrt{x}\,\mathrm{BesselJ}\left(1,\sqrt{-\_c_1}\sqrt{x}\right) + \_C3\sqrt{x}\,\mathrm{BesselY}\left(1,\sqrt{-\_c_1}\sqrt{x}\right)\right\}\right\}\right]$$

```
> pdsolve(eq,build);
```

$$u(t,x) = \_C1\,\mathrm{e}^{-\_c_1\,t}\,\_C2\sqrt{x}\,\mathrm{BesselJ}\left(1,\sqrt{-\_c_1}\sqrt{x}\right) + \_C1\,\mathrm{e}^{-\_c_1\,t}\,\_C3\sqrt{x}\,\mathrm{BesselY}\left(1,\sqrt{-\_c_1}\sqrt{x}\right)$$

Команда

`pdetest(`*sol*, *pdeqs*`)`

перевіряє, чи рівність (їх множина) *sol* є розв'язком рівняння з частинними похідними (системи) *pdeqs*. Якщо це справді так, то виводиться 0, інакше – вираз, який визначає величину нев'язки. Приклади:

```
> pdetest(u(t,x)=42*x^2*t-12*x*t^2+56*t^3,
  diff(u(t,x),t,t)=4*diff(u(t,x),x,x)-24*x);
```

$$0$$

```
> pdetest(u(t,x)=x^2*t, diff(u(t,x),t,t)=
  4*diff(u(t,x),x,x)-24*x);
```

$$-8\,t + 24\,x$$

§ 7.8. Наближене розв'язування диференціальних рівнянь з частинними похідними

Для наближеного розв'язування мішаних задач для залежних від часу рівнянь з частинними похідними та їх систем використовується команда `pdsolve` з трохи іншим, ніж у § 7.7, синтаксисом:

`pdsolve(`*pdeqs*, *conds*, `numeric`, *opts*`)`,

де *pdeqs* – рівняння або множина рівнянь, *conds* – множина початкових і крайових умов, *opts* – опції. Похідні в початкових і крайових умовах потрібно задавати за допомогою оператора

`D[`*n*`](`*f*`)(`*vars*`)`,

де f – функція, *vars* – змінні, n – натуральне число чи їх послідовність. Кожне число у послідовності n задає диференціювання за змінною з відповідним йому номером, наприклад:

```
> D[1,2,2](u)(t,x);
```

$$\mathrm{D}_{1,2,2}(u)(t,x)$$

```
> convert(D[1,2,2](u)(t,x),diff);
```

$$\frac{\partial^3}{\partial x^2 \partial t} u(t,x)$$

Команда `pdsolve` у режимі наближеного розв'язування диференціальних рівнянь створює модуль, тому результат її виконання потрібно присвоїти деякій змінній. Створений модуль містить кілька методів. Для доступу до методу потрібно після імені модуля (тобто змінної, якій його присвоєно) поставити символи `:-`, а після них – ім'я методу. Для обчислення значення розв'язку у певній точці використовують метод

`value(`*v*`=`*val*`)`,

який створює процедуру для обчислення значення розв'язку при *v=val*. Для побудови двовимірного графіка, просторового графіка й анімації (про анімацію йтиметься також у § 11.7) використовуються відповідно методи

$$\texttt{plot(}v1\texttt{=}val, v2\texttt{=}a\texttt{..}b, popts\texttt{)},$$
$$\texttt{plot3d(}v1\texttt{=}a\texttt{..}b, v2\texttt{=}c\texttt{..}d, popts\texttt{)},$$
$$\texttt{animate(}v1\texttt{=}a\texttt{..}b, v2\texttt{=}c\texttt{..}d, popts\texttt{)},$$

де *popts* – опції команд `plot` і `plot3d` (§ 4.2, 4.4). Детальніше з описом цих методів можна ознайомитись у довідковій системі Maple. Там можна також знайти опис використання опцій *opts* команди `pdsolve` для наближеного розв'язування рівнянь з частинними похідними.

Розв'яжемо мішану задачу

$$\frac{\partial}{\partial t}u(t,x) = 4x\frac{\partial^2}{\partial x^2}u(t,x),$$
$$u(0,x) = 0, \quad u(t,0) = \sin t, \quad u(t,2) = \cos 2t:$$

```
> PF:=pdsolve(diff(u(t,x),t)=4*x*diff(u(t,x),x,x),
  {u(0,x)=0,u(t,0)=sin(t),u(t,2)=cos(2*t)}, numeric);
```

$$PF := \textbf{module}(\,)\ \textbf{export}\ plot, plot3d, animate, value, settings;$$
$$\ldots\ \textbf{end module}$$

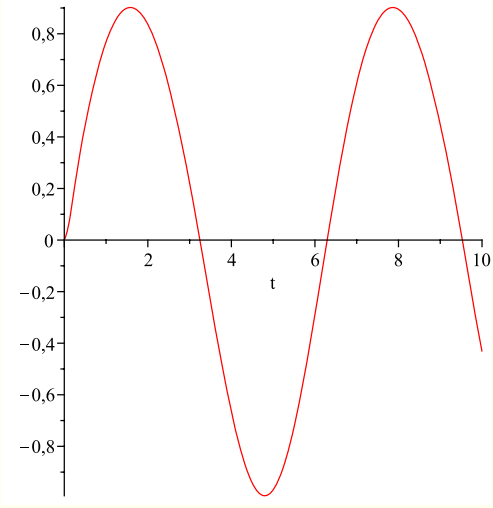
```
> Fv:=PF:-value(t=0.1);
```

$$Fv := \textbf{proc}(x\_pde)\ \ldots\ \textbf{end proc}$$

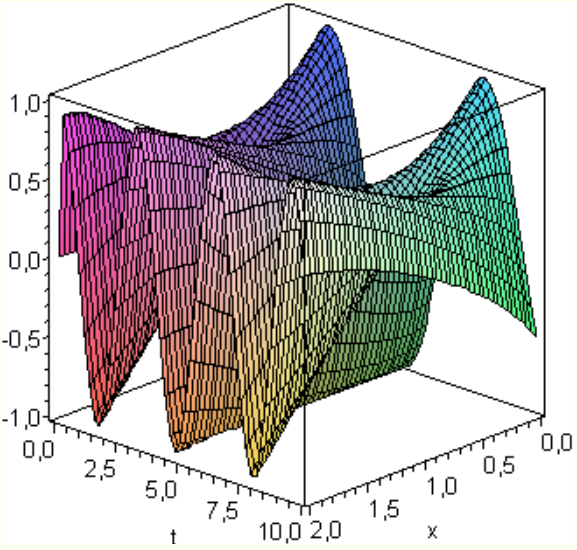
```
> Fv(0.2);
```

$$[t = 0.1,\ x = 0.2,\ u(t,x) = 0.04053821697449643 00]$$

```
> PF:-plot(x=0.1,t=0..10,numpoints=100);
```




```
> PF:-plot3d(t=0..10,x=0..2,axes=boxed,grid=[50,50]);
```



Інший спосіб графічного подання наближених розв'язків рівнянь з частинними похідними першого порядку наведено в

§ 12.3.

§ 7.9. Розв'язування інтегральних рівнянь

Для розв'язування лінійних інтегральних рівнянь використовують команду

`intsolve`(*ieq*, *fn*, *method*),

де *ieq* – інтегральне рівняння, *fn* – шукана функція, *method* – опція `method`, значенням якої вказується метод розв'язування рівняння (`Neumann` – метод ітерованих ядер, `Laplace` – операційний метод Лапласа, `differentialequation` – зведення до звичайного диференціального рівняння, `eigenfunction` – зведення до крайової задачі на власні значення). Зазвичай Maple самостійно вибирає найкращий метод розв'язування інтегрального рівняння.

Команда `intsolve` знаходить точний розв'язок таких типів інтегральних рівнянь: однорідні й неоднорідні рівняння Фредгольма I і II роду, однорідні й неоднорідні рівняння Вольтерри I і II роду. У деяких випадках розв'язок може залежати від довільних сталих, які позначаються *_C1*, *_C2* і т. д.

Операційний метод (параметр `Laplace`) застосовується тільки до однорідних і неоднорідних лінійних рівнянь Вольтерри з ядром типу згортки.

Метод ітерованих ядер (параметр `Neumann`) застосовується для інтегральних рівнянь Вольтерри II роду, Фредгольма II роду і деяких рівнянь Вольтерри I роду.

Приклади:

```
> eq1:=y(x)+int(cos(x+t^2)*y(t),t=0..Pi)=x;
```

$$eq1 := y(x) + \int_0^{\pi} \cos(x + t^2)\, y(t)\, \mathrm{d}t = x$$

```
> intsolve(eq1,y(x));
```

$$y(x) = x + \frac{4\cos(x)}{\pi + 2} - \frac{2\,\pi \sin(x)}{\pi - 2}$$

```
> eq2:=int((5*x-t)^2*y(t),t=1..x)=16*x;
```

$$eq2 := \int_1^x (5\,x - t)^2\, y(t)\,\mathrm{d}t = 16\,x$$

```
> intsolve(eq2,y(x));
```

$$y(x) = -\frac{7\sin\left(\frac{1}{4}\ln(x)\right)}{x^{11/4}} + \frac{\cos\left(\frac{1}{4}\ln(x)\right)}{x^{11/4}}$$

```
> eq3:=x^2+x=int((3*x^2*t+2*x*t^2)*y(t),t=-1..1);
```

$$eq3 := x^2 + x = \int_{-1}^1 \left(3\,x^2\,t + 2\,x\,t^2\right) y(t)\,\mathrm{d}t$$

```
> intsolve(eq3,y(x));
```

$$y(x) = \frac{3}{4} - \frac{3}{5}\,\_C1 + \_C1\,x^2 + \frac{1}{2}\,x$$

```
> eq4:=y(x)-int(x*t*y(t),t=0..1)=2*x;
```

$$eq4 := y(x) - \left(\int_0^1 x\,t\,y(t)\,\mathrm{d}t\right) = 2\,x$$

```
> intsolve(eq4,y(x));  # Точний розв'язок
```

$$y(x) = 3\,x$$

```
> intsolve(eq4,y(x),method=Neumann);
```

$$y(x) = \frac{2186}{729}\,x$$

```
> Order:=25: intsolve(eq4,y(x),method=Neumann);
```

$$y(x) = \frac{2541865828328}{847288609443}\,x$$

Питання до розділу 7

1. Яка команда використовується для відшукання точних розв'язків алгебричних і трансцендентних рівнянь та їх систем? Опишіть її формат.

2. Як знайти всі розв'язки тригонометричного рівняння?

3. Як розв'язати нерівність?

4. Яку команду використовують для наближеного розв'язування алгебричних і трансцендентних рівнянь? Опишіть особливості її застосування.

5. Як знайти цілі розв'язки рівняння чи системи?

6. Як розв'язати рекурентне рівняння?

7. Як знайти розв'язок звичайного диференціального рівняння чи їх системи? Як розв'язати початкову чи крайову задачу для цього рівняння чи системи? У чому полягають особливості наближеного розв'язування таких рівнянь?

8. Як знайти розв'язок диференціального рівняння з частинними похідними чи їх системи? У чому полягають особливості наближеного розв'язування початкових і крайових задач для таких рівнянь?

9. Яка команда призначена для розв'язування лінійних інтегральних рівнянь?

Вправи до розділу 7

1. Розв'яжіть рівняння $4^x + 4^{2-x} = 17$.

2. Знайдіть всі розв'язки системи

$$\begin{cases} x^3 + y^3 = 28, \\ xy = 3. \end{cases}$$

3. Знайдіть всі розв'язки рівняння

$$\sin^4 x + \cos^4 x + 3 \sin x \cos x = 2.$$

4. Розв'яжіть нерівність $\log_{2x}\left(\frac{2-3|x|}{x^2-9}\right) > 0$.

5. Знайдіть дійсний розв'язок рівняння $\operatorname{tg} x + x = 5$ на проміжку $[-2, -1]$.

6. Розв'яжіть у цілих числах рівняння $3x + y^4 = 7$.

7. Знайдіть загальний розв'язок рівняння Ейлера

$$x^2y'' + 5xy' + 4y = 2x,$$

а також частинний розв'язок цього рівняння, який задовольняє початкові умови $y(1) = 5$, $y'(1) = 4$.

8. Знайдіть наближений числовий розв'язок задачі Коші з попередньої вправи у точках $x = 1{,}5$ і $x = 2$ та наближений розв'язок цієї задачі у вигляді степеневого ряду з точністю до члена десятого порядку малості.

9. Знайдіть загальний розв'язок диференціального рівняння

$$x\frac{\partial z}{\partial x} + (xz + y)\frac{\partial z}{\partial y} = z.$$

10. Знайдіть розв'язок інтегрального рівняння

$$y(x) + \int\limits_0^{\frac{\pi}{2}} (3x + \cos t)y(t)dt + 5 = 0.$$

Розділ 8. Основи програмування в Maple

§ 8.1. Maple-мова програмування

Maple розв'язує велику кількість задач без жодного програмування у загальноприйнятому розумінні цього поняття. Більш того, існують тисячі задач, алгоритми яких вже реалізовані у вигляді функцій чи команд системи. Але це зовсім не означає, що в Maple не можна програмувати.

Maple підтримує три власні мови: вхідну мову, мову реалізації та мову програмування. Вхідна мова орієнтована на розв'язування математичних задач практично довільної складності і призначена для задання системі вхідних даних для наступного їх опрацювання. Вхідна мова має велику кількість визначених математичних і графічних функцій, а також велику бібліотеку.

Має Maple і свою мову процедурного програмування – Maple-мову з традиційними засобами структурування програм: операторами циклів і розгалуження, командами керування, функціями користувача, процедурами тощо. Вона також включає в себе всі команди і функції вхідної мови, їй доступні всі спеціальні оператори та функції.

Багато з цих функцій самі є доволі складними програмами, наприклад, символьне диференціювання, інтегрування, розвинення у ряд Тейлора тощо.

Не треба плутати вхідну мову і Maple-мову програмування з мовою її реалізації. Нею є одна з найпотужніших мов програмування – C. На цій мові написане ядро системи, яке містить ретельно оптимізовані процедури.

Синтаксис структурних операторів мови Maple нагадує синтаксис операторів мов програмування Бейсік і Паскаль. Це значно полегшує знайомство з нею тим, хто має певний досвід програмування на цих мовах. У § 8.2 – 8.4 розглянемо базові можливості програмування у Maple.

§ 8.2. Оператор розгалуження

Програми, всі інструкції яких виконуються послідовно, називають лінійними. Однак переважна більшість програм є розгалуженими, тобто в них залежно від результатів обчислення й умов роботи можливі переходи від однієї вітки з командами до іншої.

Найпростішу конструкцію розгалуженої програми задає оператор розгалуження (умовний оператор) `if`, загальна форма якого має такий синтаксис:

```
if <Умова 1> then <Оператори 1>
|elif <Умова 2> then <Оператори 2>|
|else <Оператори 3>|
end if;
```

Тут і надалі у вертикальних рисках вказуватимемо необов'язкові елементи конструкції.

Умови (логічні вирази) задаються за допомогою логічних відношень (знаків порівняння `<`, `>`, `=`, `<=`, `>=`, `<>`) і логічних операторів `and`, `or`, `not`, `xor`, конструкції з яких набувають логічних значень `true` або `false`. При побудові логічних виразів можна використовувати круглі дужки.

Дія оператора `if`: якщо логічний вираз `<Умова 1>` набуває значення `true` (`<Умова 1>` справджується), то виконується послідовність операторів `<Оператори 1>`; якщо логічний вираз `<Умова 1>` набуває значення `false` (`<Умова 1>` не виконується), то перевіряється на істинність вираз `<Умова 2>` (якщо він заданий) і у випадку його істинності виконується послідовність операторів `<Оператори 2>`. Якщо жоден з логічних виразів не виконується, то виконується блок `<Оператори 3>` (якщо цей блок заданий). Для відокремлення один від одного операторів, як завжди, використовують двокрапку або крапку з комою.

Наведемо приклад програми, яка знаходить значення ку-

скової функції

$$f(x) = \begin{cases} -x, & x < 0, \\ x^2, & 0 \leqslant x < 2, \\ 6 - x, & x \geqslant 2 \end{cases}$$

в заданій точці області визначення:

```
> x:=4:
> if x<0 then f:=-x
  elif x<2 then f:=x^2
  else f:=6-x
  end if;
```

$$f := 2$$

Писати оператор розгалуження в кілька рядків необов'язково – це зроблено задля наочності. Нагадаємо, що для переходу на наступний рядок в одній області введення використовують комбінацію клавіш `Shift`+`Enter`.

Блок `elif` можна повторювати в операторі розгалуження довільну кількість разів, але блок `else` може бути тільки один.

Замість `end if` можна писати «симетричне» `fi`.

На практиці найчастіше використовуються такі типи умовних виразів:

1) `if <Умова> then <Оператори> end if`; – якщо виконується `<Умова>`, то виконуються `<Оператори>`, інакше не виконується нічого;

2) `if <Умова> then <Оператори 1> else <Оператори 2> end if`; – якщо виконується `<Умова>`, то виконуються команди з `<Оператори 1>`, інакше виконуються `<Оператори 2>`.

Використовуючи конструкцію 2), складемо програму для знаходження десяткового наближення значення функції $f(m) = m^3 + \cos 2m$, де m – найбільше з двох заданих чисел a і b.

```
> a:=100/101: b:=101/102:
> if a>b then m:=a else m:=b end if:
```

```
> f(m)=evalf(m^3+cos(2*m));
```

$$f\left(\frac{101}{102}\right) = 0.5726370207$$

§ 8.3. Цикли

Часто у програмі необхідно виконати циклічне повторення певного виразу задану кількість разів доти, поки виконується певна умова.

У Maple є кілька операторів циклу. Для їх запису використовують службові слова `for`, `from`, `by`, `to`, `while`, `do`, `end do`. Тілом всіх операторів циклу є команди між словами `do` та `end do`. Замість слів `end do` можна використовувати `od`. У Maple використовуються дві форми операторів циклу: `for-from` та `for-in`. Перший оператор циклу є універсальним і включає як цикли, що повторюються задану кількість разів, так і цикли, які виконуються, поки деякий логічний вираз є істинним. За другою формою оператора циклу реалізується цикл за елементами послідовності виразів, списку чи множини.

Найбільш загальний опис оператора циклу `for-from` має вигляд:

|`for <Ім'я>`| |`from <Вираз 1>`|
|`to <Вираз 2>`| |`by <Крок>`|
|`while <Вираз 3>`|
`do <Оператори> end do;`

Тут `<Ім'я>` – ім'я параметра (змінної) циклу, `<Вираз 1>`, `<Вираз 2>` – початкове і кінцеве значення параметра циклу відповідно, `<Крок>` – приріст (додатний або від'ємний) параметра циклу після завершення етапу циклу, `<Вираз 3>` – логічний вираз, який задає умову, доки тіло циклу буде виконуватись, `<Оператори>` – послідовність операторів, які виконуються в циклі (тіло циклу).

Отже, у ході виконання циклу його змінна змінюється від значення `<Вираз 1>` до значення `<Вираз 2>` з кроком, заданим

виразом <Крок>. Якщо блок by <Крок> відсутній, то змінна циклу буде змінюватися з кроком +1 (якщо значення <Вираз 1> менше значення <Вираз 2>). Виконання циклу починається з надання його змінній початкового значення, після чого, якщо крок циклу додатний (від'ємний), перевіряється, чи не більше (не менше) воно від кінцевого значення, й у випадку позитивної відповіді виконуються оператори тіла циклу, змінна циклу змінюється на значення кроку й алгоритм перевірки починається знову. Якщо значення змінної циклу стає більшим або меншим від кінцевого значення, то виконання циклу закінчується. Якщо <Вираз 1> більше <Вираз 2> і заданий додатний <Крок>, то цикл не виконається жодного разу. Якщо заданий блок while, то одночасно з перевіркою значення параметра циклу перевіряється на істинність значення логічного виразу <Вираз 3> цього блоку, і виконання команд циклу також припиняється, якщо <Вираз 3> набуває значення false.

Наприклад, знайдемо добуток всіх непарних чисел від 5 до 15:

```
> d:=1: for i from 5 by 2 to 15 do d:=d*i end do:
> d;
```

675675

Якщо деякий блок циклу не заданий, то його параметр набуває значення за замовчуванням: for фіктивна змінна, from 1, by 1, to infinity, while true.

При використанні циклу for-from обов'язковим є тільки блок do, що визначає тіло циклу, причому він може бути єдиним блоком циклу:

do <Оператори> end do;

Ця конструкція визначає нескінчений цикл, зупинити який можна лише за допомогою операторів break, return або виникнення помилки при виконанні операторів циклу. Для дострокового припинення виконання циклу потрібно використовувати оператор break. Про оператор return йтиметься у § 8.4.

Можлива спрощена конструкція циклу типу `while`:
`while <Умова> do <Оператори> end do;`
Тут вираз `<Оператори>` виконується доти, поки виконується логічна умова `<Умова>`. Розглянемо приклад, в якому число n збільшується втричі доти, поки $n < 28$:

```
> n:=1: while n<28 do n:=3*n end do;
```

$$n := 3$$
$$n := 9$$
$$n := 27$$
$$n := 81$$

Крапка з комою після `end do` означає, що треба відображати результати виконання операторів у тілі циклу (крім оператора розгалуження), незалежно від того, крапкою з комою чи двокрапкою вони завершуються в тілі циклу.

За другою формою оператора циклу `for-in` організовується цикл за елементами об'єкта, який може бути послідовністю, списком, множиною, сумою, добутком або рядком. Синтаксис такого циклу:
`for <Ім'я> in <Дані> |while <Вираз>|`
`do <Оператори> end do;`
Параметр циклу, визначений у блоці `for-in`, послідовно набуває значень складових об'єкта `<Дані>`. Цикл виконується стільки разів, скільки елементів задано в об'єкті `<Дані>`, якщо тільки логічний `<Вираз>` у необов'язковому блоці `for-in` не набуде значення `false` раніше, ніж будуть послідовно використані всі елементи об'єкта `<Дані>`. Наприклад:

```
> summa:=0: for j in [1,x,3,2*x,x^2,x^3,sqrt(y)]
  do summa:=summa+j end do:
> summa;
```

$$4 + 3\,x + x^2 + x^3 + \sqrt{y}$$

Як правило, завершення будь-якого циклу відбувається, коли значення параметра циклу перевищує задане кінцеве

значення (якщо крок додатний), стає меншим кінцевого значення (якщо крок від'ємний) або логічний вираз умови у блоці `<Дані>` набуває значення `false`.

Інколи виникає потреба за деяких умов пропустити виконання частини або всіх операторів тіла циклу. Таку можливість надає оператор `next`, за яким здійснюється перехід до наступного кроку. Наприклад, у результаті виконання наступної програми друкуватимуться факторіали простих чисел від 1 до 10:

```
> for i from 1 to 10 do
  if not isprime(i) then next end if; i! end do;
```

$$2$$
$$6$$
$$120$$
$$5040$$

Цикли можуть бути вкладеними. Проілюструємо це на прикладі, коли створюється діагональна матриця M четвертого порядку, елементами якої є квадрати натуральних чисел:

```
> M:=matrix(4,4):
> for i to 4 do for j to 4 do
  if i=j then M[i,j]:=i^2 else M[i,j]:=0 end if;
  end do end do;
> evalm(M);
```

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 \\ 0 & 0 & 9 & 0 \\ 0 & 0 & 0 & 16 \end{bmatrix}$$

Звичайно, діагональну матрицю можна створити раціональніше (с. 170), але цей спосіб дозволяє виконувати досить загальні перетворення матриць.

У § 5.9 розглядалися команди `map`, `map2`, `select`, `remove`, `selectremove`, `zip`, які дозволяють виконувати циклічні дії без використання циклів `for-from` і `for-in`.

§ 8.4. Процедури

Крім функцій користувача (§ 3.2), у Maple можна створювати і власні процедури. Процедурою називають об'єкт програми, який має самостійне значення і виконує одну або декілька операцій. Процедури є важливим елементом структурного програмування і засобом розширення користувачем можливостей Maple.

Процедура починається з заголовку. Заголовок має ім'я процедури (ім'я визначає користувач), далі йде обов'язковий оператор присвоювання `:=` і службове слово `proc`, після якого в круглих дужках через кому вказуються формальні параметри процедури.

Процедура обов'язково повинна закінчуватися службовими словами `end proc`.

Опис процедури має такий синтаксис:

name`:=proc(|<Формальні параметри>|)`
`|local <Локальні змінні>|`
`|global <Глобальні змінні>|`
`|option <Опції>|`
`|description <Рядок опису>|`
`<Тіло процедури>`
`end proc;`

При оголошенні процедури єдиним обов'язковим параметром є послідовність операторів, які формують `<Тіло процедури>`. Решта параметрів, що визначають локальні і глобальні змінні, список формальних параметрів, задають спеціальні опції режиму виконання процедури і рядок опису (по суті коментар), можуть бути відсутні. За відсутності формальних параметрів круглі дужки все одно треба ставити.

Найпростіше процедуру задати так:

name`:=proc(<Формальні параметри>)`
`<Тіло процедури>`
`end proc;`

де *name* – ім'я процедури.

Виклик процедури здійснюється вказуванням її імені *name* зі списком фактичних параметрів:

name`(<Фактичні параметри>)`,

тобто практично не відрізняється від виклику команд Maple.

Фактичні параметри процедури задаються послідовністю імен змінних, наприклад, `proc(x)`, `proc(x,y)`, `proc(x1,x2,x3)` тощо. При виклику процедури фактичні параметри підставляються на місце формальних.

Результат останнього виконання оператора в процедурі є значенням, яке вертає процедура.

Після імені змінної в рядку заголовку процедури можна визначити її тип (§ 5.6), наприклад, `n::integer` (n – ціле), `n::posint` (n – натуральне), `n::scalar` (n – скалярна величина). Якщо цього не зробити, то перевірку коректності введення аргументів доведеться виконувати в ручному режимі за допомогою команди `type` чи `is`, інакше при некоректних фактичних параметрах буде отримано малозрозуміле повідомлення про помилку або неправильний (несподіваний) результат.

Створимо, наприклад, процедуру для знаходження довжини вектора $\overrightarrow{a} = (x, y)$:

```
> leng:=proc(x::numeric, y::numeric)
  sqrt(x^2+y^2)
  end proc:
> leng(3,5);
```

$$\sqrt{34}$$

```
> leng(a,b);
```

Error, invalid input: leng expects its 1st argument, x, to be of type numeric, but received a

У цьому прикладі невідповідність фактичних параметрів типу заданих змінних призвела до повідомлення про помилку і відмови від виконання процедури.

Змінні, вказані у списку формальних параметрів, є локальними, тобто їх зміна відбувається тільки у тілі процедури. Для того, щоб фактичним параметрам можна було присвоїти певні значення в результаті роботи процедури, використовують тип `evaln`:

```
> AAA:=proc(x::evaln)
  x:=5; x+1 end proc:
> AAA(b);
```

$$b + 1$$

```
> b;
```

$$5$$

Кількість фактичних параметрів не обов'язково повинна дорівнювати кількості формальних параметрів процедури. Якщо їх менше, то помилка при виконанні процедури виникне лише тоді, коли при обчисленні тіла процедури справді вимагатиметься значення цього відсутнього параметра. Якщо фактичних параметрів більше, ніж формальних, то додаткові параметри будуть зігноровані.

Для явного означення змінних у тілі процедури локальними призначене ключове слово `local`, а глобальними – ключове слово `global`.

```
> m:=0:
> modcompl:=proc(z)
  m:=evalf(sqrt(Re(z)^2+Im(z)^2))
  end proc;
Warning, `m` is implicitly declared local to procedure `modcompl`
```

$$modcompl := \mathbf{proc}(z)\ \mathbf{local}\ m;$$
$$m := evalf(\mathrm{sqrt}(\mathrm{Re}(z)\hat{}2 + \mathrm{Im}(z)\hat{}2))\ \mathbf{end\ proc}$$

У цьому прикладі система Maple повідомляє про те, що змінна m повинна бути оголошена локальною, і сама робить це у рядку виведення.

Як складніший приклад розглянемо побудову процедури для розвинення функції у тригонометричний ряд Фур'є (такої команди у Maple немає). Нехай потрібно $2l$-періодичну функцію $f(x)$ розвинути у ряд Фур'є на інтервалі $[x_1, x_2]$. Тоді, як відомо,

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty} a_k \cos\frac{k\pi x}{l} + \sum_{k=1}^{\infty} b_k \sin\frac{k\pi x}{l},$$

де $l = \frac{1}{2}(x_2 - x_1)$, $a_0 = \frac{1}{l}\int\limits_{x_1}^{x_2} f(x)dx$,

$$a_k = \frac{1}{l}\int_{x_1}^{x_2} f(x)\cos\frac{k\pi x}{l}dx, \quad b_k = \frac{1}{l}\int_{x_1}^{x_2} f(x)\sin\frac{k\pi x}{l}dx.$$

Отримати перші n членів ряду Фур'є можна за допомогою такої процедури:

```
> fourier:=proc(f::scalar,x::symbol,x1::scalar,
  x2::scalar,n::posint) local k,a,b,l,S:
  l:=(x2-x1)/2: a[0]:=1/l*int(f,x=x1..x2):
  a[k]:=1/l*int(f*cos(k*Pi*x/l),x=x1..x2):
  b[k]:=1/l*int(f*sin(k*Pi*x/l),x=x1..x2):
  S:=a[0]/2+sum(a[k]*cos(k*Pi*x/l)+
  b[k]*sin(k*Pi*x/l), k=1..n);
  end proc:
```

Звернутись до цієї процедури можна за допомогою команди `fourier`(f, x, x_1, x_2, n), де f – функція, яку потрібно розвинути у ряд, x – незалежна змінна, $[x_1, x_2]$ – інтервал розвинення, n – кількість членів ряду. У процедурі передбачено перевірку коректності аргументів.

Розвинемо у ряд Фур'є функцію $f(x) = 2x$ з періодом π на інтервалі $[0, \pi]$, обмежившись першими чотирма членами ряду:

```
> f[4]:=fourier(2*x,x,0,Pi,4);
```

$$f_4 = \pi - 2\sin(2x) - \sin(4x) - \frac{2}{3}\sin(6x) - \frac{1}{2}\sin(8x)$$

Інколи необхідно, щоб деякі змінні після виконання процедури набули значень, отриманих у ході її виконання. Тоді у процедурі ці змінні потрібно оголосити глобальними. Розглянемо приклад:

```
> a:=1: b:=1:
  fg:=proc(x,y)  global a,b;
  a:=x^2; b:=y^2; sqrt(a+b);
  end proc:
> fg(3,4); a; b;
```

$$5$$
$$9$$
$$16$$

Зазвичай процедура вертає значення останнього виразу її тіла. Для виведення деякого іншого значення використовують оператор `return <Вираз>`. Цей оператор взагалі припиняє виконання процедури і виводить `<Вираз>`:

```
> modcompl:=proc(z)
  evalf(sqrt(Re(z)^2+Im(z)^2)); return Re(z);
  z^2; end proc:
> modcompl(3+I*4);
```

$$3$$

Розглянемо ще один приклад. Кожен правильний дріб $\frac{n}{m}$ можна подати у вигляді

$$\frac{n}{m} = \frac{1}{x_1} + \frac{1}{x_2} + \ldots + \frac{1}{x_k},$$

де $x_1, x_2, \ldots, x_k \in \mathbb{N}$, $x_1 < x_2 < \ldots < x_k$.

Для відшукання чисел x_1, x_2, …, x_k створимо процедуру `ndivm`, виконаємо обчислення і перевірку їх правильності:

```
> ndivm:=proc(n,m) local k,nn,mm,dr,s;
  if not is(n,posint) then return "Перший аргумент
  повинен бути натуральним числом!" end if:
  if not is(m,posint) then return "Другий аргумент
  повинен бути натуральним числом!" end if:
  if n>=m then return "Перший аргумент повинен бути
  меншим від другого!" end if:
  nn:=n: mm:=m: k:=1:
  while not is(mm/nn,posint) do
  s[k]:=floor(mm/nn)+1; dr:=nn/mm-1/s[k]:
  nn:=numer(dr): mm:=denom(dr): k:=k+1:
  end do:
  s[k]:=mm/nn; convert(s,list);
  end proc:
> ss:=ndivm(353,45954);
```

$$ss := [131, 20831, 677849073, 765798942267083142]$$

```
> w:=0: for i to nops(ss) do w:=w+1/ss[i] end do: w;
```

$$\frac{353}{45954}$$

```
> ndivm(5,3);
```

"Перший аргумент повинен бути меншим від другого!"

Насправді алгоритм працюватиме і для $n \geqslant m$, а перевірку типів можна було зробити так само, як у прикладі для членів ряду Фур'є (стор. 156).

При створенні рекурсивних процедур рекомендується використовувати опцію `remember` (для цього пишуть на початку процедури `option remember`), яка змушує результати об-

числень процедури зберігати у спеціальній таблиці. Це істотно пришвидшує обчислення при виклику процедури з тими самими фактичними параметрами, що й раніше.

Детальніше з програмуванням у Maple можна ознайомитись у книгах [5, 9, 11].

Питання до розділу 8

1. Як створювати розгалуження? Опишіть формат оператора розгалуження.

2. Які види циклів можна використовувати? Опишіть їх формати.

3. Як достроково перервати виконання циклу? Як достроково припинити виконання чергової ітерації циклу і перейти до наступної?

4. Як створювати процедури? Опишіть формат опису процедури. Як достроково перервати виконання процедури? Чим процедура відрізняється від функції користувача?

Вправи до розділу 8

1. За допомогою циклу обчисліть кількість простих чисел на проміжку $[1000, 2000]$ і виведіть їх список у порядку зростання.

2. Створіть двома способами команду `listprod`, єдиним аргументом якої є список, а результатом – добуток елементів списку, передбачивши при цьому перевірку коректності введення аргументу команди.

3. Напишіть команду `matrixzero`, єдиним аргументом якої є матриця, а результатом – множина двоелементних списків, що містять координати нульових елементів матриці.

4. Створіть команду `matrixnzplot`, яка б своїм єдиним аргументом приймала матрицю, а виводила графік, на якому точками будуть позначені ненульові елементи матриці.

5. Створіть команду `atan2`, яка б для точки (x, y) виводила кут (у радіанах) проти годинникової стрілки між додатним напрямом осі абсцис і радіус-вектором точки (відрізком, що з'єднує початок координат і задану точку). Врахувати, що для точки $(0, 0)$ кут визначити неможливо. Передбачити перевірку коректності аргументів.

Розділ 9. Робота з пакетами

При запуску програми у пам'ять комп'ютера завантажується лише ядро Maple, написане на мові C. Решта компонентів, які реалізують функціональність системи, написані на мові Maple і знаходяться в основній бібліотеці та додаткових бібліотеках, які називають *пакетами*. Команди з основної бібліотеки завантажуються автоматично, тобто одразу після звертання до них. Всі команди Maple, які розглядались у попередніх розділах, містяться в основній бібліотеці програми.

Якщо потрібна команда знаходиться в пакеті, то його потрібно підключити командою

`with(`*package*`)`,

де *package* – ім'я пакета.

Якщо команда `with` закінчується крапкою з комою, то в області виведення відображатиметься список команд, які містить пакет. Усі ці команди тепер можна використовувати безпосередньо. Інколи пакети при підключенні перевизначають команди з основної бібліотеки. У такому випадку в області виведення відображається відповідне попередження.

Переважну більшість пакетів можна відключити командою

`unwith(`*package*`)`,

але вона працює не для всіх пакетів. Команда `restart` чи відповідна кнопка на панелі інструментів, крім звільнення пам'яті від результатів виконання команд, відключає всі пакети.

Можна виконати окрему команду з пакета, не підключаючи його. Для цього використовується повний виклик команди:

package`[`*command*`](`*args*`)`,

де *package* – ім'я пакета, *command* – ім'я команди, *args* – аргументи команди.

Існує понад 100 пакетів Maple. Розглянемо деякі з них.

Пакет	Опис
Algebraic	Команди для роботи з многочленами і раціональними функціями, коефіцієнти яких є алгебричними числами
ArrayTools	Команди для низькорівневих операцій з векторами, матрицями і масивами
AudioTools	Команди для роботи з аудіофайлами
Bits	Команди для ефективних побітових операцій
combinat	Команди комбінаторики
combstruct	Команди для роботи з комбінаторними структурами
CurveFitting	Команди для наближення даних кривими, зокрема, для інтерполяції
DEtools	Засоби для перетворень звичайних диференціальних рівнянь, їх інтегрування, побудови інтегральних кривих та полів напрямів систем диференціальних рівнянь
difforms	Команди для роботи з диференціальними формами у диференціальній геометрії
DiscreteTransforms	Команди для перетворень дискретних даних, зокрема, дискретного перетворення Фур'є
DynamicSystems	Команди для створення, моделювання і графічного відображення лінійних систем
ExcelTools	Команди для доступу до даних, збережених у форматі Microsoft Excel

Пакет	Опис
FileTools	Команди для роботи з файлами
finance	Команди для фінансових обчислень (фінансова математика)
geom3d	Команди для побудов і розв'язування задач аналітичної геометрії у тривимірному евклідовому просторі
geometry	Команди для побудов і розв'язування задач аналітичної геометрії на евклідовій площині
GraphTheory	Команди для роботи з графами
group	Команди для роботи з групами перестановок і скінченними групами
ImageTools	Команди для роботи з рисунками
inttrans	Команди для роботи з інтегральними перетвореннями (Фур'є, Лапласа та ін.)
linalg	Команди для роботи з матрицями і векторами
LinearAlgebra	Команди для роботи з матрицями Matrix і векторами Vector
ListTools	Команди для роботи зі списками
Logic	Команди для перетворення виразів з використанням двозначної логіки
Matlab	Команди для взаємодії з системою Matlab
MmaTranslator	Засоби для перетворення областей введення і робочих листів, створених у системі Mathematica, в області введення і робочі листи Maple
numapprox	Команди для апроксимації функцій многочленами

Пакет	Опис
numtheory	Команди для розв'язування задач теорії чисел
Optimization	Команди для числового розв'язування задач оптимізації
orthopoly	Команди для побудови ортогональних многочленів
PDEtools	Засоби для перетворень рівнянь з частинними похідними, їх розв'язування і графічного відображення розв'язків
plots	Команди побудови спеціальних видів графіків функцій
plottools	Команди для створення графічних об'єктів та роботи з ними
PolynomialTools	Команди для роботи з многочленами
powseries	Команди для роботи з формальними степеневими рядами
RandomTools	Інструменти для створення випадкових об'єктів і роботи з ними
RealDomain	Після підключення пакета обчислення і перетворення будуть здійснюватись у припущенні, що всі невідомі є дійсними
RootFinding	Команди для наближеного розв'язування алгебричних і трансцендентних рівнянь та систем спеціальними методами
ScientificConstants	Команди для отримання інформації про фізичні константи і властивості хімічних елементів

Пакет	Опис
simplex	Команди для розв'язування задач лінійної оптимізації на основі симплекс-методу
Slode	Команди для побудови формальних розв'язків лінійних диференціальних рівнянь у вигляді степеневих рядів
Statistics	Удосконалені інструменти для математичної статистики, розподілів випадкових величин і аналізу даних
stats	Колекція підпакетів для математичної статистики, розподілів випадкових величин і аналізу даних
Student	Колекція підпакетів для допомоги студентам у вивченні курсу вищої математики
student	Застарілий пакет для допомоги студентам у вивченні курсу вищої математики
tensor	Команди для роботи з тензорами
Tolerances	Пакет для виконання обчислень з похибками
Units	Команди для перетворення величин з одних у інші одиниці вимірювання і виконання обчислень з одиницями вимірювання
VariationalCalculus	Команди для розв'язування задач варіаційного числення
VectorCalculus	Команди для векторного числення, в тому числі для відшукання кратних, поверхневих і криволінійних інтегралів

Обчислимо, наприклад, величину банківського вкладу на суму 1000 грн, покладеного під 5% річних на 4,5 роки з умовою, що відсотки нараховуються один раз у кінці року на всю суму разом з попередніми нарахованими відсотками (так звані складні відсотки). За неповний рік відсотки нараховуються аналогічно. Для обчислень скористаємось командою `futurevalue(`A`,` p`,` n`)` з пакета `finance` (A – величина вкладу, p – річні відсотки у частках одиниці, n – кількість років):

```
> finance[futurevalue](1000,0.05,4.5);
```

1245.523270

або

```
> with(finance): futurevalue(1000,0.05,4.5);
```

1245.523270

У наступних розділах посібника частково описані пакети `linalg`, `plots`, `DEtools`, `PDEtools`, `VectorCalculus`, `student`, `Student`, `geometry`, `geom3d`, `stats`, `combinat`, `RootFinding`, `simplex`, `CurveFitting`. З використанням інших пакетів можна ознайомитись у книгах [1 – 5] і довідковій системі Maple.

Питання до розділу 9

1. Як підключити пакет? Як можна звернутись до команди, не підключаючи пакет, в якому вона міститься?

2. Як відключити пакет?

3. Назвіть пакети, призначені для побудов і розв'язування задач аналітичної геометрії (лінійної алгебри, теорії графів, комбінаторики).

4. Яку назву має пакет для допомоги студентам у вивчення вищої математики?

Розділ 10. Застосування Maple до розв'язування задач лінійної алгебри

§ 10.1. Вектори та операції з ними

Основна частина команд для розв'язування задач лінійної алгебри міститься у бібліотеці `linalg`, тому перед розв'язуванням задач з векторами чи матрицями потрібно завантажити цю бібліотеку командою

```
> with(linalg):
```

Нагадаємо, що вектор у Maple можна задати за допомогою команд `vector` або `array` зі стандартної бібліотеки (стор. 41). Наприклад,

```
> v:=vector([2,0,-1]); w:=array([a,b,c]);
```

$$v := \begin{bmatrix} 2 & 0 & -1 \end{bmatrix}$$
$$w := \begin{bmatrix} a & b & c \end{bmatrix}$$

Координату вже визначеного вектора v отримуємо, використавши запис `v[i]`, де i – номер координати, а після виконання інструкції `v[i]:=a` i-му елементу вектора v буде присвоєно нове значення a. Наприклад,

```
> v[1];
```

$$2$$

```
> v[1]:=5: v[1];
```

$$5$$

Нагадаємо (§ 5.2), що вектор можна перетворити у список, а список у вектор за допомогою команд `convert(`*vector*`,list)` і `convert(`*list*`,vector)` відповідно. Наприклад, команда

```
> s:=convert(w,list);
```

перетворює вектор w у список:

$$s := [a, b, c]$$

Додати два вектори v і w однакової розмірності можна за допомогою команд `evalm(v+w)` або `matadd(v, w)`. Наприклад,

```
> v:=vector([1,2,3]): w:=vector([3,2,1]):
> s:=evalm(v+w);
```

$$s := [4 \;\; 4 \;\; 4]$$

Команду `evalm` можна використовувати для множення вектора на число, віднімання векторів, для знаходження лінійної комбінації двох або більшої кількості векторів. Наприклад,

```
> v:=vector([0,1,2]): w:=vector([-3,-2,1]):
> evalm(0.5*v);
```

$$[0 \;\; 0.5 \;\; 1.0]$$

```
> z:=evalm(2*v-3*w);
```

$$z := [9 \;\; 8 \;\; 1]$$

Для знаходження скалярного добутку (v, w) векторів v і w використовується команда `dotprod(v, w)`, а для знаходження векторного добутку $[v, w]$ тих самих векторів – команда `crossprod(v, w)`. Кут α між векторами v і w легко знайти за допомогою команди `angle(v, w)`. Приклади:

```
> v:=vector([2,1,3]): w:=vector([1,4,-2]):
> (v,w)=dotprod(v,w); [v,w]=crossprod(v,w);
```

$$(v, w) = 0$$

$$[v, w] = [-14 \;\; 7 \;\; 7]$$

```
> alpha:=angle(v,w);
```

$$\alpha := \frac{1}{2}\pi$$

Довжину $|v|$ вектора v можна знайти командою `norm(v, 2)`, а пронормувати цей вектор (знайти орт $|v|^{-1}v$) – командою

`normalize(`v`)`. Наприклад,

```
> v:=vector([0,3,4]): n:=norm(v,2);
```

$$n := 5$$

```
> n1:=normalize(v);
```

$$n1 := \left[0 \quad \frac{3}{5} \quad \frac{4}{5}\right]$$

Команда `norm(`v, p`)` дозволяє знайти норму вектора v за формулою

$$\|v\|_p = \left(\sum_{i=1}^{n} |v_i|^p\right)^{1/p},$$

де n – розмірність вектора, v_i – його i-й елемент, а p – натуральне число. Якщо пропустити параметр p, то норма вектора v буде обчислюватись за формулою $\|v\| = \max\limits_{1 \leqslant i \leqslant n} |v_i|$. Наприклад,

```
> norm(v,1); norm(v);
```

$$7$$
$$4$$

Для знаходження базису сукупності векторів v_1, ..., v_n використовується команда `basis([`$v_1, \ldots, v_n$`])`. Команда `GramSchmidt([`$v_1, \ldots, v_n$`])` ортогоналізує сукупність лінійно незалежних векторів v_1, ..., v_n за процедурою Грамма – Шмідта. Виділимо, наприклад, з сукупності векторів $v_1 = (1, 2, 2, -1)$, $v_2 = (1, 1, 5, 2)$, $v_3 = (2, 2, 3, 1)$, $v_4 = (1, 0, 0, 1)$, $v_5 = (5, 1, 0, 0)$ базис і ортогоналізуємо його:

```
> v1:=vector([1,2,2,-1]): v2:=vector([1,1,5,2]):
> v3:=vector([2,2,3,1]): v4:=vector([1,0,0,1]):
> v5:=vector([5,1,0,0]): basis([v1,v2,v3,v4,v5]);
```

$$[v1, v2, v3, v4]$$

```
> GramSchmidt(%);
```

$$\left[[1,2,2,-1],\ \left[-\frac{1}{10},-\frac{6}{5},\frac{14}{5},\frac{31}{10}\right],\ \left[\frac{179}{189},\frac{23}{63},-\frac{14}{27},\frac{121}{189}\right],\ \left[\frac{44}{323},-\frac{48}{323},\frac{12}{323},-\frac{28}{323}\right]\right]$$

§ 10.2. Матриці та операції з ними

Задати матрицю у Maple можна декількома способами (стор. 44). Зокрема, матрицю порядку $n \times m$ можна задати, використовуючи команду

$$\texttt{matrix(}n, m, \texttt{[[}a_{11}, \ldots, a_{1m}\texttt{]}, \ldots, \texttt{[}a_{n1}, \ldots, a_{nm}\texttt{]]),}$$

де n, m – кількість рядків і стовпців (обидва ці параметри не обов'язкові). Наприклад,

```
> A:=matrix([[1,-2,3], [2,3,0]]);
```

$$A := \begin{bmatrix} 1 & -2 & 3 \\ 2 & 3 & 0 \end{bmatrix}$$

Цю саму матрицю можна задати й інакше:

```
> A:=<<1,2>|<-2,3>|<3,0>>;
```

$$A := \begin{bmatrix} 1 & -2 & 3 \\ 2 & 3 & 0 \end{bmatrix}$$

або

```
> A:=<<1|-2|3>,<2|3|0>>;
```

$$A := \begin{bmatrix} 1 & -2 & 3 \\ 2 & 3 & 0 \end{bmatrix}$$

У Maple існує можливість інтерактивного задання матриці. Для цього призначена команда `entermatrix`. Після її використання діалог з користувачем при роботі з класичним робочим листом матиме, наприклад, такий вигляд:

```
> A:=array(1..2, 1..3): entermatrix(A);
enter element 1,1 > 0;
enter element 1,2 > 1;
enter element 1,3 > 2;
enter element 2,1 > 3;
enter element 2,2 > 4;
enter element 2,3 > 5;
```

$$\begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \end{bmatrix}$$

```
> M:=%;
```

$$M := \begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \end{bmatrix}$$

У стандартному робочому листі команда `entermatrix` відкриває окреме вікно для введення елементів матриці.

Створити матрицю можна також з допомогою деякої функції $f(i,j)$, де i,j – індекси матриці `matrix(`n, m, f`)`. Наприклад,

```
> f:=(i,j)->x^i+y^(j+1): A:=matrix(2,3,f);
```

$$A := \begin{bmatrix} x+y^2 & x+y^3 & x+y^4 \\ x^2+y^2 & x^2+y^3 & x^2+y^4 \end{bmatrix}$$

У Maple є можливість генерувати деякі матриці спеціального виду. Зокрема, діагональну матрицю третього порядку можна задати так:

```
> J:=Matrix(3,3,shape=diagonal):
> J[1,1]:=2: J[2,2]:=-1: J[3,3]:=Pi: print(J);
```

$$J := \begin{bmatrix} 2 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & \pi \end{bmatrix}$$

Для створення симетричної матриці можна використати таку послідовність команд:

```
> S:=Matrix(3,3,shape=symmetric):
> S[1,1]:=-2: S[1,3]:=4: S[2,3]:=5: print(S);
```

$$S := \begin{bmatrix} -2 & 0 & 4 \\ 0 & 0 & 5 \\ 4 & 5 & 0 \end{bmatrix}$$

Треба мати на увазі, що деякі команди пакета `linalg` працюють тільки з матрицями типу `matrix` і не працюють з матрицями типу `Matrix`.

Команда `augment(`A, B`)` створює нову матрицю, об'єднуючи дві задані матриці A і B по горизонталі. Команда `stackmatrix(`A, B`)` формує нову матрицю, об'єднуючи дві задані матриці A і B по вертикалі. Звичайно, матриці повинні мати у першому випадку однакову кількість рядків, а в другому – стовпців. Наприклад, для матриць S, J з попередніх прикладів маємо:

```
> with(linalg): augment(S,J), stackmatrix(S,J);
```

$$\begin{bmatrix} -2 & 0 & 4 & 2 & 0 & 0 \\ 0 & 0 & 5 & 0 & -1 & 0 \\ 4 & 5 & 0 & 0 & 0 & \pi \end{bmatrix}, \begin{bmatrix} -2 & 0 & 4 \\ 0 & 0 & 5 \\ 4 & 5 & 0 \\ 2 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & \pi \end{bmatrix}$$

Кількість рядків і стовпців у матриці A можна визначити за допомогою команд `rowdim(`A`)`, `coldim(`A`)` відповідно. Ранг матриці A дозволяє знайти команда `rank(`A`)`. Слід квадратної матриці (суму її діагональних елементів) знаходимо за допомогою команди `trace(`A`)`. Наприклад,

```
> M:=Matrix([[3,1,-2], [2,-2,0], [1,3,-2]]);
```

$$M := \begin{bmatrix} 3 & 1 & -2 \\ 2 & -2 & 0 \\ 1 & 3 & -2 \end{bmatrix}$$

```
> r:=rank(M); t:=trace(M);
```

$$r := 2$$
$$t := -1$$

Додавання двох матриць A і B однакової розмірності можна виконати за допомогою тих самих команд, які використовувались для додавання векторів: `evalm(`A`+`B`)` або `matadd(`A, B`)`. Для матриць типу `Matrix` (а не `matrix`) можна просто писати A`+`B. За допомогою команди `evalm` можна також помножити матрицю на число. Знайдемо, наприклад, лінійну комбінацію двох матриць:

```
> M1:=Matrix([[3,0,-2],[2,-1,0],[4,3,-2]]);
```

$$M1 := \begin{bmatrix} 3 & 0 & -2 \\ 2 & -1 & 0 \\ 4 & 3 & -2 \end{bmatrix}$$

```
> M2:=Matrix([[1,3,1],[1,-1,2],[0,1,0]]);
```

$$M2 := \begin{bmatrix} 1 & 3 & 1 \\ 1 & -1 & 2 \\ 0 & 1 & 0 \end{bmatrix}$$

```
> M:=evalm(2*M1-3*M2);
```

$$M := \begin{bmatrix} 3 & -9 & -7 \\ 1 & 1 & -6 \\ 8 & 3 & -4 \end{bmatrix}$$

Пропонуємо читачам самостійно переконатися, що до таких самих результатів приводять також команди `matadd(2*M1,-3*M2)`, `matadd(M1,M2,2,-3)` та `2*M1-3*M2`.

Якщо існує добуток матриць A і B, то його можна знайти за допомогою команд `evalm(`A`&*`B`)` або `multiply(`A, B`)`.

Наприклад,

```
> A:=matrix([[0,1,2],[2,-3,4]]);
```

$$A := \begin{bmatrix} 0 & 1 & 2 \\ 2 & -3 & 4 \end{bmatrix}$$

```
> B:=matrix([[2,1],[2,-3],[2,0]]);
```

$$B := \begin{bmatrix} 2 & 1 \\ 2 & -3 \\ 2 & 0 \end{bmatrix}$$

```
> P:=multiply(A,B);
```

$$P := \begin{bmatrix} 6 & -3 \\ 6 & 11 \end{bmatrix}$$

Матрицю A^T, транспоновану до заданої матриці A, отримуємо за допомогою команди `transpose(`A`)`. Знайдемо, наприклад, матрицю, транспоновану до наведеної матриці P:

```
> transpose(P);
```

$$\begin{bmatrix} 6 & 6 \\ -3 & 11 \end{bmatrix}$$

Команда `norm(`A`)` знаходить норму матриці $A = (a_{ij})_{i,j=1}^{n,m}$, що обчислюється за формулою $\|A\| = \max\limits_{1\leqslant i\leqslant n} \sum\limits_{j=1}^{m} |a_{ij}|$. Команда `norm(`$A, 1$`)` обчислює норму матриці A за формулою $\|A\|_1 = \max\limits_{1\leqslant j\leqslant m} \sum\limits_{i=1}^{n} |a_{ij}|$. Команда `norm(`$A$`,frobenius)` дозволяє знайти норму матриці A, що обчислюється як

$$\|A\| = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{m} a_{ij}^2}.$$

Наприклад, для матриці A зі стор. 173:

```
> norm(A), norm(A,1), norm(A,frobenius);
```

$$9, 6, \sqrt{34}$$

Команда `cond(`A`)`, `cond(`A`,1)` чи `cond(`A`,frobenius)` знаходить число обумовленості квадратної невиродженої матриці A, обчислене за формулою $\|A\|\cdot\|A^{-1}\|$. Другий аргумент може визначати норму матриці. Мале число обумовленості матриці означає, що при обертанні цієї матриці з наближеними числами або розв'язуванні відповідної системи алгебричних рівнянь похибка заокруглення буде невеликою. Наприклад, для матриці M зі стор. 172:

```
> cond(M), cond(M,1);
```

$$\frac{228}{43}, \frac{3094}{473}$$

§ 10.3. Визначники матриці, мінори та алгебричні доповнення

Визначник квадратної матриці A обчислюється за допомогою команди `det(`A`)`. Наприклад,

```
> A:=matrix([[2,1,2],[-2,-1,0],[1,2,-4]]);
```

$$A := \begin{bmatrix} 2 & 1 & 2 \\ -2 & -1 & 0 \\ 1 & 2 & -4 \end{bmatrix}$$

```
> Delta:=det(%);
```

$$\Delta := 6$$

Знайдемо визначник Вандермонда третього порядку:

```
> M:=Matrix(3,3,(i,j)->b[j]^(i-1));
```

$$M := \begin{bmatrix} 1 & 1 & 1 \\ b_1 & b_2 & b_3 \\ b_1^2 & b_2^2 & b_3^2 \end{bmatrix}$$

```
> d:=det(M);
```

$$d := b_2 b_3^2 - b_3 b_2^2 - b_1 b_3^2 + b_1 b_2^2 + b_1^2 b_3 - b_1^2 b_2$$

```
> d:=factor(d);
```

$$d := (b_2 - b_1)\,(-b_1 + b_3)\,(b_3 - b_2)$$

Команда `minor(`A, i, j`)` дозволяє знайти матрицю, утворену з заданої матриці A викреслюванням i-го рядка та j-го стовпця, а отже, за допомогою цієї команди можна знайти мінор M_{ij} матриці A. Наприклад, для матриці A з цього параграфа маємо:

```
> minor(A,2,3);
```

$$\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$$

```
> M[23]=det(%);
```

$$M_{23} = 3$$

Алгебричне доповнення A_{ij} елемента a_{ij} матриці A легко знайти, пригадавши, що $A_{ij} = (-1)^{i+j} M_{ij}$.

§ 10.4. Функції від матриць

Матрицю A^{-1}, обернену до матриці A, можна знайти за допомогою однієї з команд: `evalm(1/`A`)`, `evalm(`A`^(-1))`, `inverse(`A`)` (або просто A`^(-1)` чи `1/`A для матриці типу `Matrix`). Наприклад,

```
> A:= matrix([[1, -2, 3], [-3, 1, 2], [1, 0, -1]]);
```

$$A := \begin{bmatrix} 1 & -2 & 3 \\ -3 & 1 & 2 \\ 1 & 0 & -1 \end{bmatrix}$$

```
> inverse(A);
```

$$\begin{bmatrix} \frac{1}{2} & 1 & \frac{7}{2} \\ \frac{1}{2} & 2 & \frac{11}{2} \\ \frac{1}{2} & 1 & \frac{5}{2} \end{bmatrix}$$

```
> multiply(A,1/A); #Перевірка
```

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

Піднесення матриці A до натурального степеня n здійснюється командою `evalm(A^n)` (або просто `A^n` для матриці типу `Matrix`). Наприклад:

```
> A:=Matrix([[0,a,a^2],[0,0,a],[0,0,0]]);
```

$$A := \begin{bmatrix} 0 & a & a^2 \\ 0 & 0 & a \\ 0 & 0 & 0 \end{bmatrix}$$

```
> A^2, A^3;
```

$$\begin{bmatrix} 0 & 0 & a^2 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & a^2 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Для знаходження матричної експоненти e^A використовується команда `exponential(A)`. Наприклад, для останньої матриці A з цього параграфа маємо:

```
> exponential(A);
```

$$\begin{bmatrix} 1 & a & \frac{3}{2}a^2 \\ 0 & 1 & a \\ 0 & 0 & 1 \end{bmatrix}$$

Відзначимо також вже знайому команду `map` (стор. 96), що дозволяє застосовувати задану функцію до кожного елемента матриці. Наведемо приклади застосування цієї команди:

```
> M:=Matrix([[1,x],[x^2,x^3]]);
```

$$M := \begin{bmatrix} 1 & x \\ x^2 & x^3 \end{bmatrix}$$

```
> map(diff,M,x);
```

$$\begin{bmatrix} 0 & 1 \\ 2\,x & 3\,x^2 \end{bmatrix}$$

```
> map(int,M,x);
```

$$\begin{bmatrix} x & \frac{1}{2}\,x^2 \\ \frac{1}{3}\,x^3 & \frac{1}{4}\,x^4 \end{bmatrix}$$

```
> map(sqrt,M);
```

$$\begin{bmatrix} 1 & \sqrt{x} \\ \sqrt{x^2} & \sqrt{x^3} \end{bmatrix}$$

Аналогічно можна виконувати над матрицями й інші, значно складніші перетворення.

§ 10.5. Спектральний аналіз матриць

Для відшукання власних чисел і власних векторів матриці A використовують команди `eigenvalues(`A`)` і `eigenvectors(`A`)` відповідно. У результаті виконання останньої команди отримаємо власні числа, їх кратність і відповідні власні вектори. Розглянемо приклад:

```
> K:=Matrix([[3,-1,1],[-1,5,-1],[1,-1,3]]);
```

$$K := \begin{bmatrix} 3 & -1 & 1 \\ -1 & 5 & -1 \\ 1 & -1 & 3 \end{bmatrix}$$

```
> eigenvectors(K);
```

$$\left[6, 1, \left\{\left[\begin{array}{ccc} 1 & -2 & 1 \end{array}\right]\right\}\right], \left[3, 1, \left\{\left[\begin{array}{ccc} 1 & 1 & 1 \end{array}\right]\right\}\right], \left[2, 1, \left\{\left[\begin{array}{ccc} -1 & 0 & 1 \end{array}\right]\right\}\right]$$

У кожній з трьох великих квадратних дужок наведені власне число, його кратність і відповідний власний вектор у фігурних дужках. Отже, матриця K має три власні вектори: $a_1 = (-1, 0, -1)$, який відповідає власному числу $\lambda_1 = 2$ кратності 1, $a_2 = (1, 1, 1)$, який відповідає власному числу $\lambda_2 = 3$

кратності 1, і $a_3 = (1, -2, 1)$, який відповідає власному числу $\lambda_3 = 6$ кратності 1.

Характеристичну матрицю $F_A = \lambda E - A$ можна знайти за допомогою команди `charmat(`A, λ`)`. Для знаходження характеристичного многочлена $P_A = \det(\lambda E - A)$ матриці A використовують команду `charpoly(`A, λ`)`. Мінімальний многочлен (дільник) матриці A можна знайти з допомогою команди `minpoly(`A, λ`)`. Наприклад:

```
> A:=matrix([[3,-I,0],[I,3,0],[0,0,4]]);
```

$$A := \begin{bmatrix} 3 & -\mathrm{I} & 0 \\ \mathrm{I} & 3 & 0 \\ 0 & 0 & 4 \end{bmatrix}$$

```
> F['A']:=charmat(A,lambda);
```

$$F_A := \begin{bmatrix} \lambda - 3 & \mathrm{I} & 0 \\ -\mathrm{I} & \lambda - 3 & 0 \\ 0 & 0 & \lambda - 4 \end{bmatrix}$$

```
> P(lambda):=charpoly(A,lambda);
```

$$P(\lambda) := \lambda^3 - 10\,\lambda^2 + 32\,\lambda - 32$$

```
> d(lambda):=minpoly(A,lambda);
```

$$d(\lambda) := 8 - 6\,\lambda + \lambda^2$$

До канонічної форми матрицю A можна звести командою `jordan(`A`)`. Наприклад, для останньої матриці A:

```
> jordan(A);
```

$$\begin{bmatrix} 2 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 4 \end{bmatrix}$$

До трикутного вигляду матрицю A можна звести за допомогою трьох команд:

1) `gausselim(`A`)` – використовує метод Гаусса;

2) `ffgausselim(`A`)` – використовує метод Гаусса без ділення (цю команду рекомендуємо використовувати до символьних матриць, бо вона не здійснює нормування елементів і виключає помилки, пов'язані з діленням на нуль);

3) `gaussjord(`A`)` – використовує метод Гаусса-Жордана.

Розглянемо зведення матриці третього порядку до трикутного вигляду на прикладі:

```
> B:= Matrix([[3, 2, -2],[1, 3, 4],[-2, -3, 4]]);
```

$$B := \begin{bmatrix} 3 & 2 & -2 \\ 1 & 3 & 4 \\ -2 & -3 & 4 \end{bmatrix}$$

```
> ffgausselim(B);
```

$$\begin{bmatrix} 3 & 2 & -2 \\ 0 & 7 & 14 \\ 0 & 0 & 42 \end{bmatrix}$$

Пропонуємо самостійно перевірити, чим відрізнятимуться результати виконання команд `gausselim(B)` і `gaussjord(B)` на прикладі тієї самої матриці B.

§ 10.6. Матричні рівняння (системи лінійних алгебричних рівнянь)

Систему лінійних алгебричних рівнянь можна розв'язати двома способами:

1) за допомогою стандартної команди `solve`, яка знаходить розв'язок системи, записаної у розгорнутому вигляді

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1, \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2, \\ \qquad\qquad \ldots \quad \ldots \quad \ldots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n = b_m; \end{cases}$$

2) за допомогою команди `linsolve(`A, B`)` з пакета `linalg`, яка знаходить розв'язок матричного рівняння $AX = B$, де A, B – задані матриці.

Знайдемо першим способом загальний розв'язок системи

$$\begin{cases} 3x_1 - 4x_2 - x_3 - 10x_4 = 2, \\ 3x_1 - x_2 + 5x_3 + 6x_4 = -1, \\ 2x_1 - x_2 + 2x_3 + 3x_4 = 2, \end{cases}$$

```
> s:={3*x[1]-4*x[2]-x[3]-10*x[4]=2, 3*x[1]-x[2]+
  5*x[3]+6*x[4]=-1, 2*x[1]-x[2]+2*x[3]+3*x[4]=2}:
> q:=solve(s,{x[1],x[2],x[3]});
```

$$q := \left\{ x_1 = \frac{15}{2} - \frac{13}{2}\,x_4,\ x_2 = 6 - \frac{23}{3}\,x_4,\ x_3 = -\frac{7}{2} + \frac{7}{6}\,x_4 \right\}$$

Розв'яжемо за допомогою команди `linsolve` сумісну неоднорідну систему двох лінійних алгебричних рівнянь

$$\begin{cases} ax + by = f_1, \\ cx + dy = f_2 \end{cases}$$

у символьному вигляді:

```
> A, B:=matrix([[a,b],[c,d]]), vector([f1,f2]);
```

$$A, B := \begin{bmatrix} a & b \\ c & d \end{bmatrix}, \begin{bmatrix} f1 & f2 \end{bmatrix}$$

```
> X:=linsolve(A,B);
```

$$X := \left[-\frac{b\,f2 - f1\,d}{a\,d - c\,b} \quad \frac{a\,f2 - c\,f1}{a\,d - c\,b} \right]$$

Розв'язок невизначеної системи рівнянь подається у параметричній формі з параметрами `_t`[1], `_t`[2], …, `_t`[k], наприклад:

```
> A:=matrix([[1,3,2],[2,6,4],[1,2,0]]):
> B:=vector([6,12,-2]): linsolve(A,B);
```

$$\begin{bmatrix} -18 + 4\,\_t_1 & 8 - 2\,\_t_1 & \_t_1 \end{bmatrix}$$

§ 10.7. Додаткові можливості пакета `linalg`

Наведемо ще декілька команд пакета `linalg` для роботи з векторами і матрицями, які не використовувалися у цьому розділі:

`addcol(`$A, j1, j2, expr$`)` створює нову матрицю з матриці A шляхом додавання до стовпця з номером $j2$ стовпця з номером $j1$, помноженого на $expr$;

`addrow(`$A, i1, i2, expr$`)` створює нову матрицю з матриці A шляхом додавання до рядка з номером $i2$ рядка з номером $i1$, помноженого на $expr$;

`col(`A, j`)` виділяє з матриці A j-й стовпець як вектор;

`curl(`f, v`)` обчислює ротор трьохелементного вектора або списку f за змінними, заданими трьохелементним вектором або списком v;

`definite(`$A, kind$`)` визначає додатну (від'ємну) визначеність матриці A, параметр $kind$ може бути одним з термінів: `positive_def`, `positive_semidef`, `negative_def`, `negative_semidef`;

`diverge(`f, v`)` обчислює дивергенцію векторної функції (параметри є аналогічними параметрам команди `curl`);

`equal(`A, B`)` перевіряє рівність матриць A і B;

`geneqns(`A, v, b`)` генерує систему лінійних алгебричних рівнянь з матриці коефіцієнтів A (ім'я чи їх список v визначають невідомі у системі, необов'язковий вектор b може містити праві частини рівнянь);

`genmatrix(`$eqns$, v, 'b'`)` генерує елементи матриці з коефіцієнтів множини чи списку лінійних алгебричних рівнянь $eqns$ з невідомими з множини чи списку v (необов'язковий параметр b може містити ім'я змінної, якій буде присвоєно вектор з правих частин у системі рівнянь, якщо в якості b використовується термін `flag`, то неоднорідності будуть розміщені у додатковому правому стовпці створеної матриці);

`grad(`f, v`)` обчислює градієнт скалярної функції f, пара-

метр v – це вектор або список змінних для диференціювання;

`iszero(`A`)` визначає, чи є задана матриця A нульовою;

`mulcol(`$A, j, expr$`)` створює нову матрицю з матриці A множенням її j-го стовпця на вираз $expr$;

`mulrow(`$A, i, expr$`)` створює нову матрицю з матриці A множенням її i-го рядка на вираз $expr$;

`orthog(`A`)` визначає, чи є задана квадратна матриця A ортогональною;

`potential(`$f, v,$ `'`P`')` визначає, чи є вектор f градієнтом деякої скалярної функції за змінними зі списку v (якщо це так і необов'язковим параметром P задано ім'я, то воно після виконання команди міститиме функцію, градієнт якої дорівнює f);

`row(`A, i`)` виділяє з матриці A i-й рядок як вектор;

`scalarmul(`$A, expr$`)` множить матрицю або вектор A на заданий вираз $expr$;

`submatrix(`A, rr, cc`)` добуває вказану діапазонами або списками рядків rr і стовпців cc підматрицю із заданої матриці A;

`subvector(`A, rr, cc`)` добуває вказаний підвектор із заданої матриці A;

`swapcol(`$A, j1, j2$`)` створює нову матрицю з матриці A, переставляючи в ній стовпці $j1$ і $j2$;

`swaprow(`$A, i1, i2$`)` створює нову матрицю з матриці A, переставляючи в ній рядки $i1$ і $i2$;

`vandermonde(`L`)` утворює матрицю Вандермонда зі списку L;

`vectdim(`a`)` визначає розмірність вектора a;

`wronskian(`f, v`)` формує матрицю Вронського (визначник якої є визначником Вронського) для заданого вектора чи списку f функцій, диференціювання здійснюється за змінною v.

З іншими командами пакета `linalg` читач може ознайомитись у довідковій системі Maple. Призначення багатьох команд зрозуміле з назви.

Питання до розділу 10

1. Для чого використовують команди `angle`, `norm`, `normalize`, `dotprod`, `crossprod`, `basis`, `GramSchmidt`? Опишіть їх формати.

2. Як виконувати арифметичні операції над векторами і матрицями? Як знайти обернену матрицю? Як піднести матрицю до степеня?

3. З якою метою використовують команди `rank`, `rowdim`, `coldim`, `trace`, `augment`, `stackmatrix`, `cond`, `transpose`, `det`, `minor`? Опишіть їх формати та дію.

4. Які команди для спектрального аналізу матриць Ви знаєте? Опишіть їх.

5. Як можна розв'язувати системи лінійних алгебричних рівнянь за допомогою пакета `linalg`?

Вправи до розділу 10

1. Задано вектори $u = (2, 4, 6)$ і $v = (-1, 2, 5)$. Знайдіть:

а) лінійну комбінацію $2u - 3v$;

б) кут між векторами u і v;

в) їх скалярний і векторний добутки;

г) довжину вектора u;

д) пронормуйте вектор u.

2. Перевірте, чи вектори u і v з попереднього завдання є лінійно незалежними (для цього можна скористатись командою `basis`).

3. Створіть матриці

$$A = \begin{pmatrix} 2 & -4 & 7 \\ 3 & 1 & 5 \\ -1 & 5 & 8 \end{pmatrix}, \quad B = \begin{pmatrix} 3 & 7 & -2 \\ 2 & 5 & 4 \\ 0 & 3 & 5 \end{pmatrix}.$$

Знайдіть:

а) $A + B$; б) $4B - A$; в) A^2; г) A^{-1}; д) A^T; е) e^A;

є) ранг, слід, визначник матриці A;

ж) мінор верхнього правого елемента матриці A;

з) норми і числа обумовленості матриці A;

и) власні значення і відповідні їм власні вектори, характеристичну матрицю, характеристичний многочлен, канонічну форму матриці A.

4. Створіть матриці $C1$ і $C2$, об'єднуючи елементи матриць A і B з вправи 3 по горизонталі і по вертикалі відповідно.

5. Розв'яжіть систему лінійних алгебричних рівнянь з матрицею A з вправи 3 і вектором правих частин u з вправи 1.

6. Створіть матрицю N з матриці B з вправи 3 шляхом додавання до її першого рядка подвоєного другого рядка.

7. Знайдіть градієнт функції $u(x, y, z) = x^3yz - \cos(x + y - z)$.

8. Знайдіть ротор і дивергенцію вектора $(x^2y - z, xyz, x^2 + z^5)$.

9. Знайдіть матрицю і визначник Вронського функцій x^4, $\sin(x^2)$, $\operatorname{tg} x$.

Розділ 11. Спеціальні засоби для відображення графіки

§ 11.1. Побудова графіків функцій однієї змінної

Крім команд `plot` і `plot3d`, які розглядались у розділі 4, для побудови графіків функцій у Maple є спеціальний пакет `plots`. Розглянемо основні команди цього пакету (всього їх понад 50). Пакет `plots`, як і всі інші, треба підключати командою `with`:

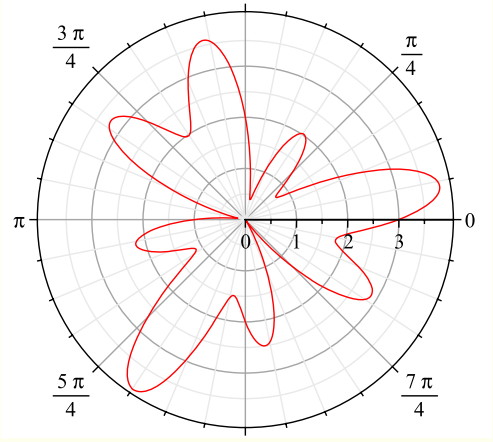
```
> with(plots):
```

Команда

$$\texttt{polarplot}(r(\varphi), \varphi{=}a\,..\,b, opts)$$

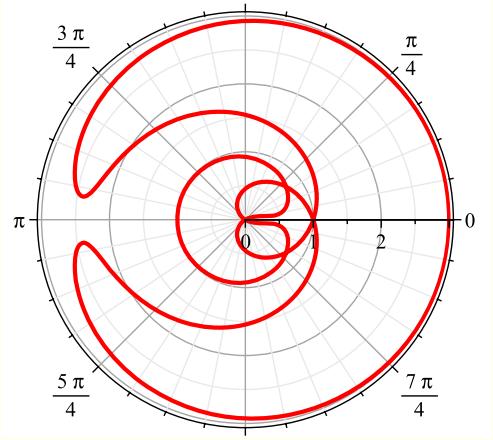
будує графік функції $r(\varphi)$ у полярній системі координат. Якщо діапазон зміни полярного кута φ не вказати, то вважатиметься, що $\varphi \in [0, 2\pi]$. В якості опцій *opts* тут і нижче в цьому розділі можна використовувати опції команди `plot` (§ 4.2). Приклад:

```
> polarplot(sin(8*phi)+cos(3*phi)+2);
```



У полярній системі координат можна будувати графіки параметрично заданих функцій. Для цього, як і у випадку команди `plot`, використовують дещо інший формат: `polarplot([`$r(t), \varphi(t), t$`=`a`..`b`],` *opts*`)`, де $r(t)$, $\varphi(t)$ – вирази полярного радіуса r і полярного кута φ через параметр t. Приклад:

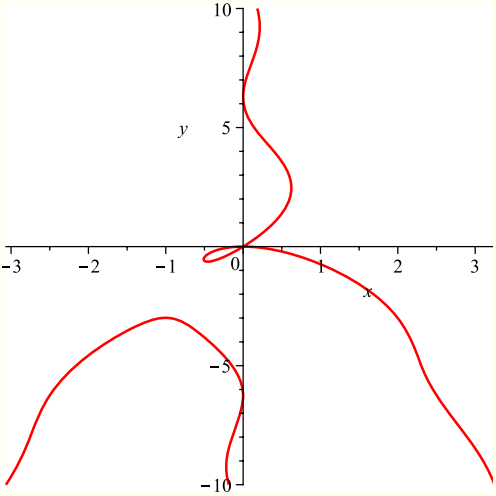
```
> polarplot([2*cos(t)+1,3*sin(2*t),t=0..2*Pi],
  thickness=3);
```



Побудова графіків кількох функцій і відображення дискретної множини точок у полярній системі координат здійснюється аналогічно до команди `plot` (§ 4.1). Команда `plot` з опцією `coords=polar` дозволяє виконувати ті самі дії, що й команда `polarplot`.

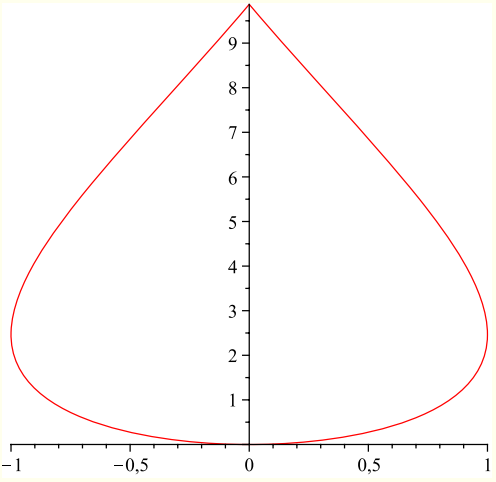
Команда `implicitplot(`*eq*, x`=`a`..`b, y`=`c`..`d, *opts*`)` будує графік функції, неявно заданої рівнянням *eq* двох змінних x, y, у прямокутнику $a \leqslant x \leqslant b$, $c \leqslant y \leqslant d$, де *opts* – опції команди `plot` (§ 4.2) або опція `grid` команди `plot3d` (с. 74). Приклад:

```
> implicitplot(x^3+x*y+cos(y)=1,x=-10..10,y=-10..10,
  grid=[300,300],thickness=2);
```



Команда `complexplot(`$f(x), x$`=`$a..b, opts$`)` будує графік комплекснозначної функції $f(x)$ дійсної змінної x на відрізку $[a, b]$. Приклад:

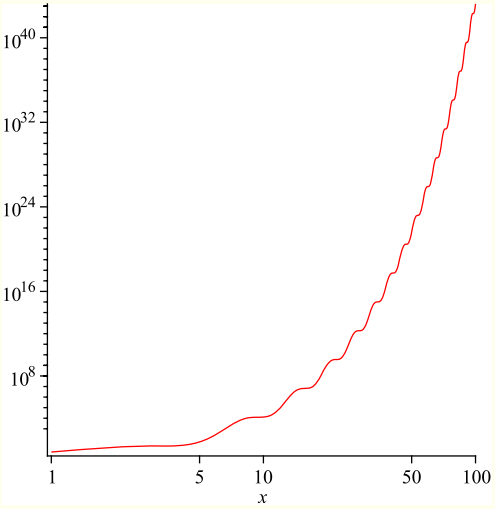
```
> complexplot(sin(x)+I*x^2,x=-Pi..Pi);
```



Команда `loglogplot` має аналогічні команді `plot` (§ 4.1)

аргументи і працює подібно до неї, але обидві осі є логарифмічними, тобто на них відкладаються десяткові логарифми відповідних значень. Приклад:

```
> loglogplot(exp(x+sin(x)),x=1..100,numpoints=1000);
```



При використанні команд `logplot` і `semilogplot` лише одна вісь буде логарифмічною: у випадку команди `logplot` нею буде вісь ординат, а у випадку команди `semilogplot` – вісь абсцис.

§ 11.2. Графіки розв'язків звичайних диференціальних рівнянь

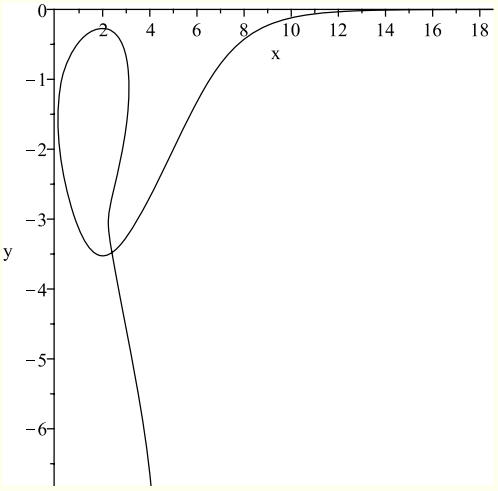
Команда

`odeplot`(*proc*, *vars*, *range*, *opts*)

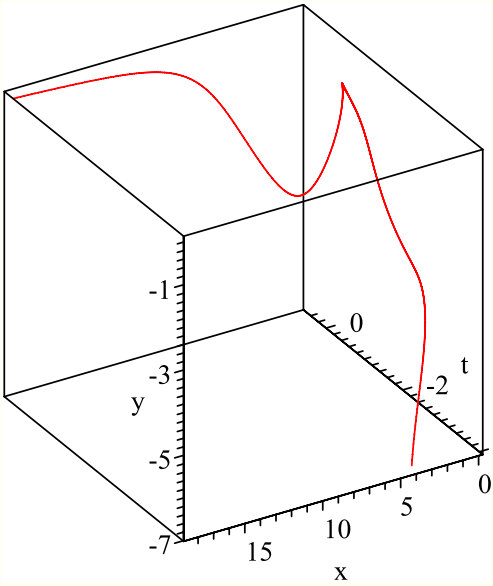
будує інтегральну криву – графік розв'язку задачі Коші або крайової задачі для звичайного диференціального рівняння чи системи таких рівнянь на площині або у просторі. Процедура *proc* для числового розв'язування задачі Коші попередньо

створюється командою `dsolve` з опцією `type=numeric`. Необов'язковим аргументом *vars* можна задати дво- або триелементний список, який визначатиме зміст (назви) осей координат. Аргументом *range* задається діапазон зміни незалежної змінної у формі $xmin..xmax$. Приклади:

```
> F:=dsolve({D(x)(t)=3*x(t)-5*cos(y(t))-t^2, D(y)(t)=
  4*y(t)-2*x(t)*y(t), x(0)=5, y(0)=-2}, {x(t),y(t)},
  type=numeric):
> odeplot(F, [x(t),y(t)], t=-4..0.5, numpoints=1000,
  color=black);
```




```
> odeplot(F, [t,x(t),y(t)], t=-4..0.5, numpoints=1000,
  axes=boxed);
```



Інший спосіб графічного подання розв'язків звичайних диференціальних рівнянь наведено у §§ 12.1, 12.2.

§ 11.3. Побудова плоских областей

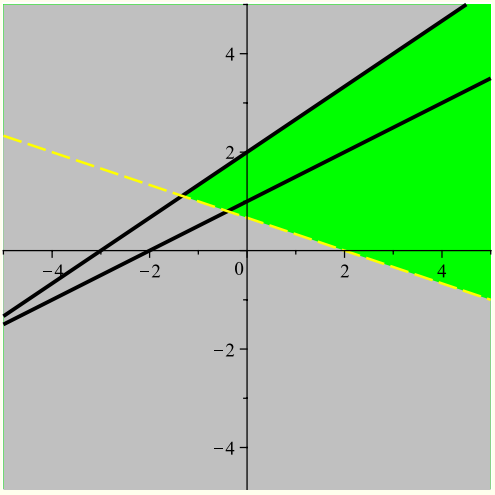
Команда

`inequal`(*ineqs*, *x*=*a*..*b*, *y*=*c*..*d*, *inopts*)

зображує у прямокутнику $a \leqslant x \leqslant b$, $c \leqslant y \leqslant d$ площини розв'язок системи лінійних рівнянь і нерівностей двох незалежних змінних x, y. Система задається множиною *ineqs*. Опції *inopts* записують у вигляді *name*=(*opts*), де в якості *name* можуть використовуватись терміни: `optionsfeasible` (область, що задовольняє систему нерівностей), `optionsexcluded` (область, яка не задовольняє систему нерівностей), `optionsopen` (область межі строгих нерівностей), `optionsclosed` (область межі нестрогих нерівностей і рівностей), а *opts* – опції команди `plot` (§ 4.2), які дозволяють задати колір та інші параметри

відображення відповідної області. Приклад:

```
> inequal({x+3*y>2, 3*y-2*x<=6, 2*y-x=2}, x=-5..5,
  y=-5..5, optionsfeasible=(color=green),
  optionsexcluded=(color=gray), optionsopen=
  (color=yellow,linestyle=3,thickness=2),
  optionsclosed=(color=black,thickness=3));
```

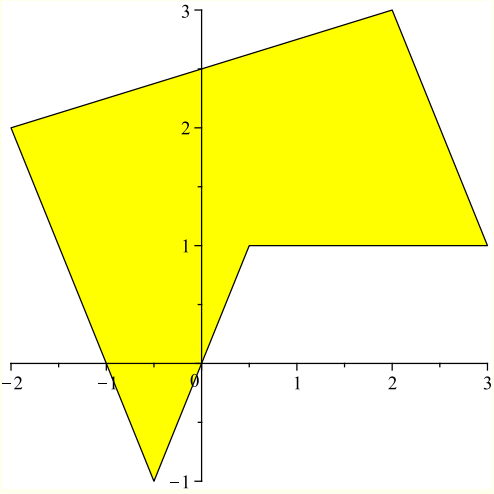


Команда

`polygonplot([[`x_1, y_1`], [`x_2, y_2`], . . . , [`x_n, y_n`]],` *opts*`)`

відображає на площині многокутник з вершинами у точках (x_1, y_1), (x_2, y_2), . . . , (x_n, y_n). Приклад:

```
> polygonplot([[2,3],[-2,2],[-0.5,-1],[0.5,1],[3,1]],
  color=yellow);
```

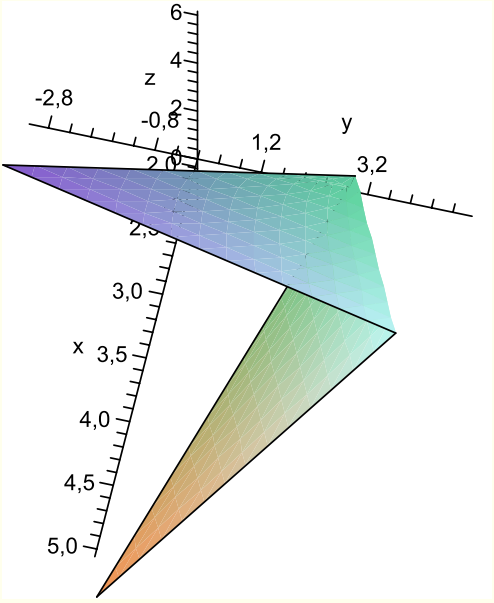


Можна будувати також многокутники в просторі командою

`polygonplot3d([[`x_1`,`y_1`,`z_1`],[`x_2`,`y_2`,`z_2`],...,[`x_n`,`y_n`,`z_n`]],`$opts$`).`

Приклад:

```
> polygonplot3d([[2,3,1],[3,-3,4],[4,5,6],[5,0,-2]],
  axes=normal,labels=[x,y,z]);
```



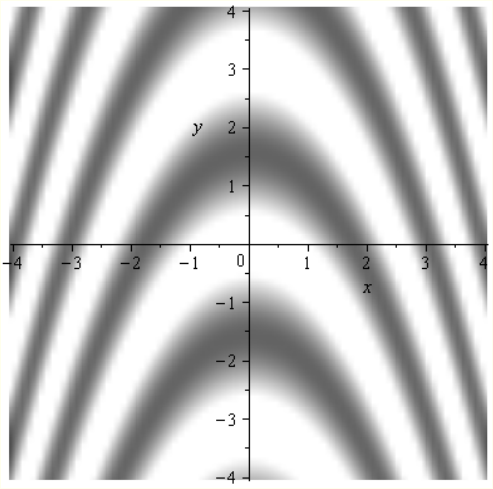
§ 11.4. Функція щільності, лінії рівня, поле градієнтів і векторне поле на площині

Команда

`densityplot(`$f(x,y)$`,`x`=`a`..`b`,`y`=`c`..`d`,`$opts$`)`

відображає у прямокутнику $a \leqslant x \leqslant b$, $c \leqslant y \leqslant d$ площини функцію щільності для функції $f(x,y)$. Світліший колір використовується для більших значень функції, а темніший – для менших. Крім різноманітних опцій команди `plot` (§ 4.2) та опцій `style` і `grid` команди `plot3d` (§ 4.4), починаючи з версії Maple 11, для керування яскравістю і контрастом додатково можна використовувати опції `brightness=`k і `contrast=`k ($0 \leqslant k \leqslant 1$, за замовчуванням $k = 0{,}5$). Приклад:

```
> densityplot(cos(x^2+2*y), x=-4..4, y=-4..4,
  grid=[80,80], brightness=0.8, style=patchnogrid);
```



Команда

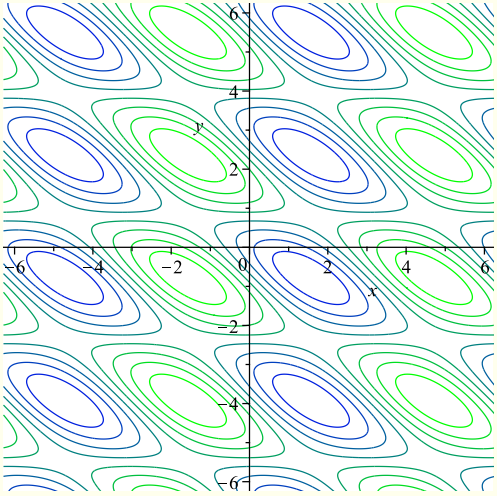
$$\texttt{contourplot(}f(x,y),x\texttt{=}a\,..\,b,y\texttt{=}c\,..\,d,opts\texttt{)}$$

будує у прямокутнику $a \leqslant x \leqslant b$, $c \leqslant y \leqslant d$ площини лінії рівня функції $f(x,y)$. Крім різноманітних опцій команди `plot` (§ 4.2) і опції `grid` команди `plot3d` (с. 74), додатково можна використовувати опцію `contours=`k для встановлення кількості ліній рівня (за замовчуванням $k = 8$), опцію `coloring=[`*c1*, *c2*`]` – для кольорів *c1* і *c2* ліній рівня з найменшим і найбільшим значеннями, опцію `filled=true` – для одночасної побудови функції щільності та ліній рівня. Приклад:

```
> f:=(x,y)->cos(x+2*y)+sin(x);
```

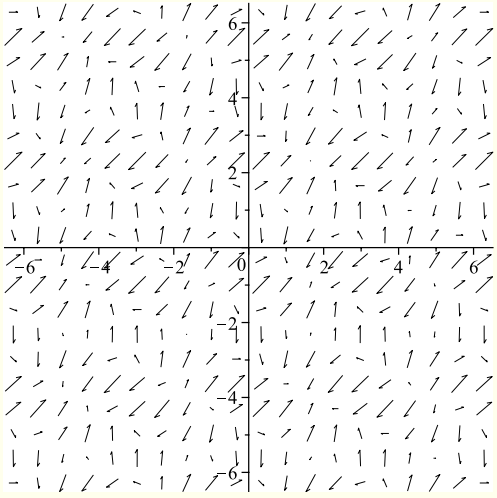
$$f := (x,y) \to \cos(x+2\,y)+\sin(x)$$

```
> contourplot(f(x,y), x=-2*Pi..2*Pi, y=-2*Pi..2*Pi,
  grid=[100,100],coloring=[green,blue]);
```



Команда `gradplot(`$f(x,y)$`, x=`$a..b$`, y=`$c..d$`, opts)` будує у прямокутнику $a \leqslant x \leqslant b$, $c \leqslant y \leqslant d$ площини поле градієнтів функції $f(x,y)$. Приклад:

```
> gradplot(cos(x+2*y)+sin(x),x=-2*Pi..2*Pi,y=-2*Pi..
  2*Pi);
```



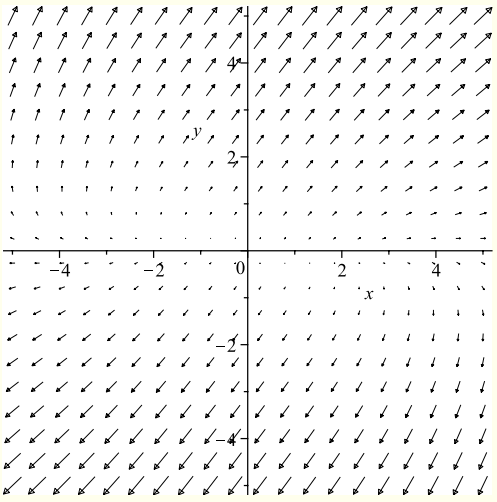
У тих командах пакета `plots`, де для позначення векторів використовуються стрілки, можна керувати відображенням цих стрілок за допомогою значень опції `arrows`: `LINE` (стрілка у вигляді риски), `THIN` (тонка стрілка – значення за замовчуванням), `SLIM` (тонка стрілка), `THICK` (товста стрілка).

Команда

$$\texttt{fieldplot([}f_1(x,y), f_2(x,y)\texttt{]}, x\texttt{=}a\texttt{..}b, y\texttt{=}c\texttt{..}d, opts\texttt{)}$$

будує у прямокутнику $a \leqslant x \leqslant b$, $c \leqslant y \leqslant d$ площини векторне поле для функцій $f_1(x,y)$, $f_2(x,y)$ (перша координата вектора є функцією $f_1(x,y)$, а друга – $f_2(x,y)$). В якості кольору можна використовувати функцію $f_3(x,y)$. Фактично команда `fieldplot` створює імітацію графіка функції $f : \mathbb{R}^2 \to \mathbb{R}^2$, а якщо враховувати колір, то навіть $f : \mathbb{R}^2 \to \mathbb{R}^3$. Приклад:

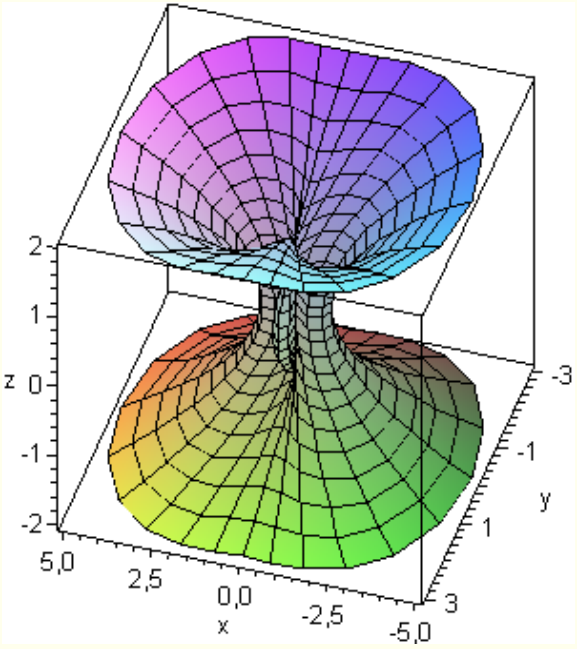
```
> fieldplot([x+3*y,sin(x)+4*y], x=-5..5, y=-5..5,
  arrows=SLIM);
```



§ 11.5. Побудова графіків функцій у просторі

Команда `cylinderplot(`$r(\varphi,z)$`,`φ`=`$a..b$`,`z`=`$c..d$`,`*opts*`)` будує поверхню $r(\varphi,z)$ у циліндричній системі координат в області зміни кута φ і аплікати z: $a \leqslant \varphi \leqslant b$, $c \leqslant z \leqslant d$. В якості опцій *opts* можна використовувати опції команд `plot` і `plot3d` (§§ 4.2, 4.4). Приклад:

```
> cylinderplot(cos(2*phi)+z^2, phi=0..2*Pi, z=-2..2,
  axes=boxed);
```



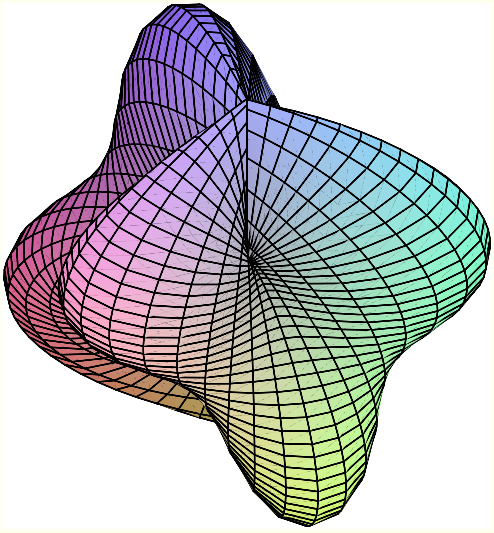
Для побудови поверхонь параметрично заданих у циліндричній системі координат функцій використовують запис

`cylinderplot(`$[r(u,v),\varphi(u,v),z(u,v)]$`,`$u$`=`$a..b$`,`$v$`=`$c..d$`,`*opts*`)`

з двома параметрами u і v.

Команда `sphereplot(`$r(\varphi,\theta)$`,`φ`=`$a..b$`,`θ`=`$c..d$`,`$opts$`)` будує поверхню $r(\varphi,\theta)$ у сферичній системі координат для $a \leqslant \varphi \leqslant b$, $c \leqslant \theta \leqslant d$. Тут θ – кут, утворений радіус-вектором точки $r(\varphi,\theta)$ з полярною віссю, а φ – кут, утворений з віссю абсцис проекцією $r(\varphi,\theta)\sin\theta$ радіус-вектора на площину, перпендикулярну до полярної осі. Приклад:

```
> sphereplot((cos(3*phi))*cos(theta)+1, phi=0..2*Pi,
  theta=0..Pi, grid=[50,50]);
```



Для побудови поверхонь параметрично заданих у сферичній системі координат функцій використовують запис

`sphereplot([`$r(u,v),\varphi(u,v),\theta(u,v)$`],`$u$`=`$a..b$`,`$v$`=`$c..d$`,`$opts$`)`

з двома параметрами u і v.

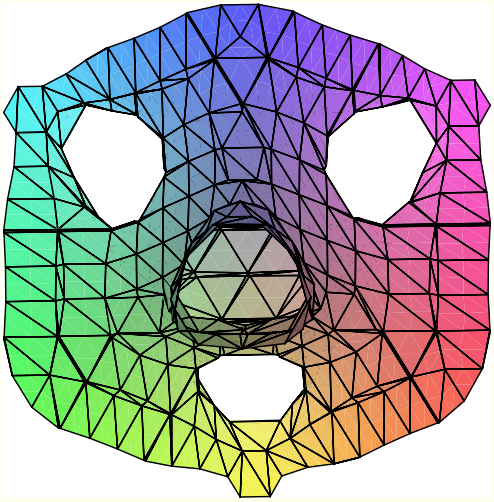
Команда

`implicitplot3d(`eq`,`x`=`$a..b$`,`y`=`$c..d$`,`z`=`$g..h$`,`$opts$`)`

будує поверхню функції, неявно заданої рівнянням eq трьох змінних x, y, z, у паралелепіпеді $a \leqslant x \leqslant b$, $c \leqslant y \leqslant d$,

$g \leqslant z \leqslant h$. Приклад:

```
> implicitplot3d(x^5+y^5+z^5+2=(x+y+z+1)^3, x=-2..2,
  y=-2..2, z=-2..2);
```



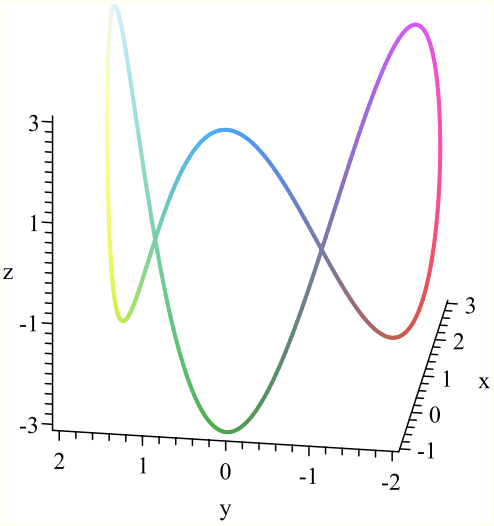
Опцією `coords` можна визначати побудову неявно заданих поверхонь у різних системах координат, наприклад, у циліндричній (`cylindrical`) або сферичній (`spherical`).

Команда

$$\texttt{spacecurve([}x(t), y(t), z(t)\texttt{],}\, t\texttt{=}a\texttt{..}b, \mathit{opts}\texttt{)}$$

дозволяє побудувати просторову криву, задану в параметричній формі рівняннями $x = x(t)$, $y = y(t)$, $z = z(t)$, $a \leqslant t \leqslant b$. Приклад:

```
> spacecurve([1+2*cos(2*t),2*cos(t),3*sin(3*t)],
  t=0..2*Pi,axes=framed,labels=[x,y,z],numpoints=200,
  thickness=2);
```



Команда

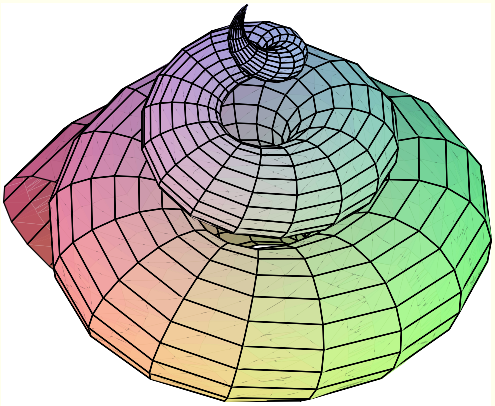
$$\texttt{tubeplot([}x(t), y(t), z(t)\texttt{],}\, t\texttt{=}a\texttt{..}b, opts\texttt{)}$$

будує циліндричну поверхню вздовж просторової кривої, заданої в параметричній формі рівняннями $x = x(t)$, $y = y(t)$, $z = z(t)$, $a \leqslant t \leqslant b$. В якості опцій *opts* можна використовувати також `radius=`r або `radius=`$f(t)$ і `tubepoints=`n для визначення радіуса і кількості точок для побудови кругового перерізу циліндричної поверхні. Приклад:

```
> x:=t->(t-6*Pi)*sin(t)/3; y:=t->(t-6*Pi)*cos(t)/3;
  z:=t->(t-6*Pi)*0.9;
```

$$x := t \to \frac{1}{3}(t - 6\,\pi)\sin(t)$$
$$y := t \to \frac{1}{3}(t - 6\,\pi)\cos(t)$$
$$z := t \to (t - 6\,\pi)\cdot 0.9$$

```
> tubeplot([x(t),y(t),z(t)], t=-0.1*Pi..6*Pi,
  radius=(t-6*Pi)*0.2,tubepoints=20);
```



Команда `complexplot3d` дозволяє будувати графіки функцій комплексної змінної, тобто графіки функцій, які діють з $\mathbb{C}$ в $\mathbb{C}$, та графіки функцій дійсної змінної, які діють з $\mathbb{R}^2$ в $\mathbb{R}^2$.

Для функцій комплексної змінної $f(z)$ використовують формат

`complexplot3d(`$f(z)$, z=a+b`*I..`c+d`*I`, *opts*`)`.

У цьому випадку зміст координатних осей наступний: $\operatorname{Re} z$, $\operatorname{Im} z$, $|f(z)|$, тобто відображається поверхня, яка показує абсолютну величину функції комплексної змінної у паралелепіпеді $a \leqslant \operatorname{Re} z \leqslant c$, $b \leqslant \operatorname{Im} z \leqslant d$.

Колір поверхні у команді `complexplot3d` залежить від значення аргументу функції $f(z)$. Відповідність кольорів аргументам комплексних чисел можна побачити на рис. 9. Для проміжних променів використовуються проміжні відтінки кольорів.

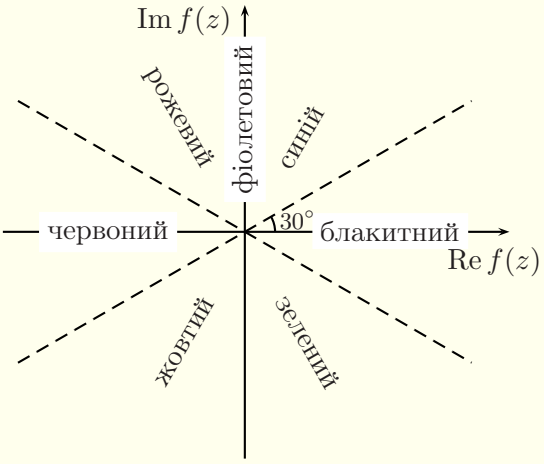
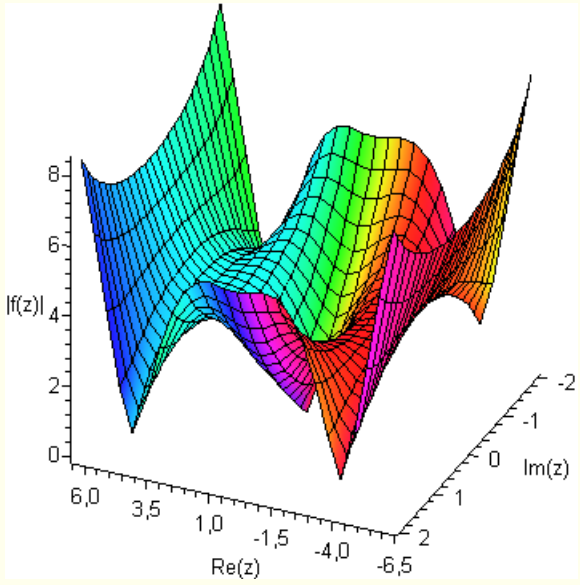


Рис. 9

Приклад:

```
> complexplot3d(sin(z)+z, z=-2*Pi-2*I..2*Pi+2*I,
  axes=framed, labels=[Re(z),Im(z),"|f(z)|"]);
```

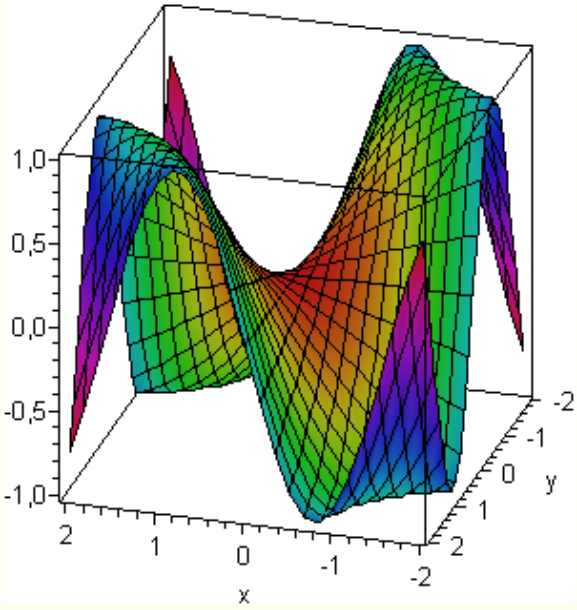


Для функцій дійсної змінної, які діють з $\mathbb{R}^2$ в $\mathbb{R}^2$, команда `complexplot3d` має дещо інший формат:

`complexplot3d([`$f_1(x,y), f_2(x,y)$`],`x`=`a`..`b`,`y`=`c`..`d`,`$opts$`).`

У цьому випадку осі координат мають такий зміст: x, y, $f_1(x,y)$. Функція $f_2(x,y)$ визначає колір поверхні, який задається наступним чином. Близькі до нуля значення відображаються блакитним кольором, найбільші додатні значення – червоним, а проміжні – синім, фіолетовим і рожевим кольорами. Найменші від'ємні значення відображаються світло-коричневим (майже червоним) кольором, а проміжні між найменшими і близькими до нуля значеннями – оранжевим, жовтим і зеленим кольорами. Приклад:

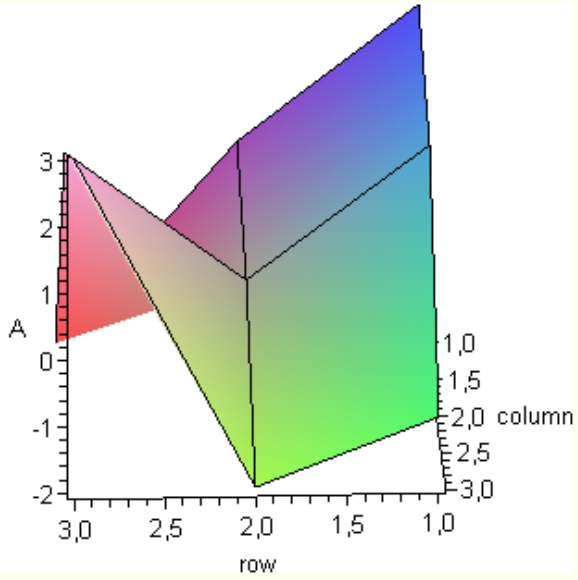
```
> complexplot3d([sin(x*y),x^2+y^2], x=-2..2, y=-2..2,
  axes=boxed);
```



Команда `matrixplot(`A`,`$opts$`)` будує поверхню, що проходить через точки, задані матрицею A. Крім звичайних

опцій *opts*, можна використовувати опцію `heights=histogram` (у цьому випадку матриця зображуватиметься гістограмою). Приклад:

```
> A:=matrix([[3,2,-1],[1,0,-2],[-2,2,3]]):
  matrixplot(A,axes=framed);
```



§ 11.6. Лінії рівня, поле градієнтів і векторне поле у просторі

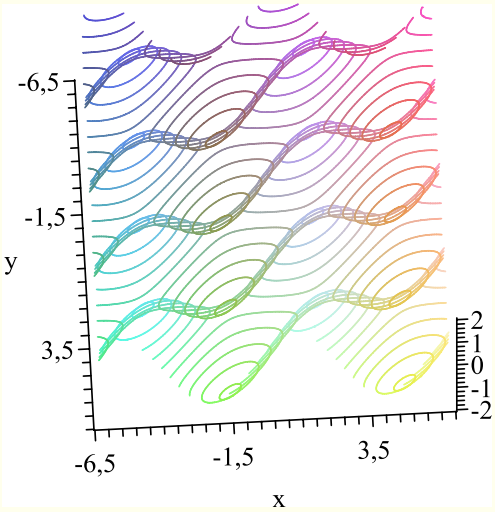
Команда

$$\texttt{contourplot3d}(f(x,y), x\texttt{=}a\texttt{..}b, y\texttt{=}c\texttt{..}d, opts)$$

будує лінії рівня функції $z = f(x,y)$ $(a \leqslant x \leqslant b,\ c \leqslant y \leqslant d)$ у просторі в площинах $z = \text{const}$, де стала const дорівнює значенню, якого набуває функція $f(x,y)$ на лінії рівня.

Для прикладу побудуємо у просторі лінії рівня функції $z = \cos(x+2y) + \sin x$ (на стор. 194 вони побудовані на площині):

```
> contourplot3d(cos(x+2*y)+sin(x), x=-2*Pi..2*Pi,
  y=-2*Pi..2*Pi, grid=[100,100], axes=framed);
```



Команда

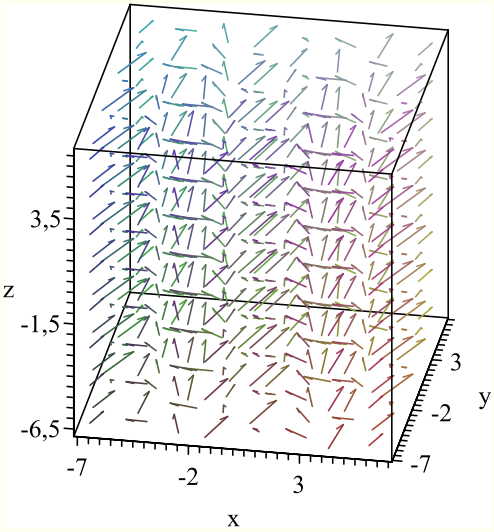
$$\texttt{gradplot3d(}f(x,y,z),x\texttt{=}a\texttt{..}b,y\texttt{=}c\texttt{..}d,z\texttt{=}g\texttt{..}h,opts\texttt{)}$$

будує у паралелепіпеді $a \leqslant x \leqslant b$, $c \leqslant y \leqslant d$, $g \leqslant z \leqslant h$ простору поле градієнтів функції $f(x,y,z)$. Приклад:

```
> w:=cos(x+2*y)+sin(x)+z;
```

$$w := \cos(x+2\,y)+\sin(x)+z$$

```
> gradplot3d(w,x=-2*Pi..2*Pi,y=-2*Pi..2*Pi,
  z=-2*Pi..2*Pi, axes=boxed, labels=[x,y,z]);
```

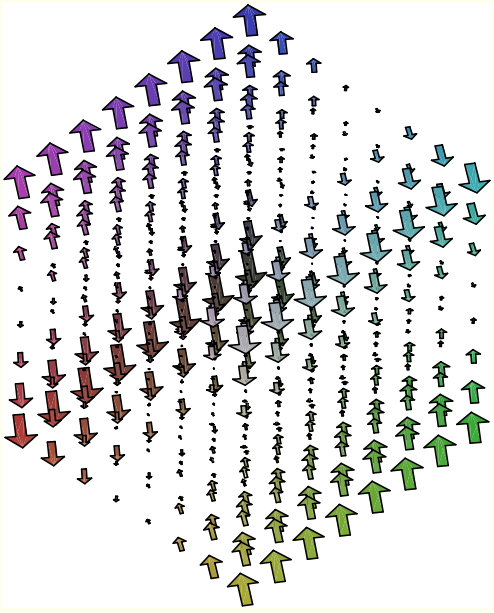


Команда

`fieldplot3d([`$f_1(x,y,z), f_2(x,y,z), f_3(x,y,z)$`],`$x$`=`$a..b$`,`$y$`=`$c..d$`,`$z$`=`$g..h$`,`$opts$`)`

будує у паралелепіпеді $a \leqslant x \leqslant b$, $c \leqslant y \leqslant d$, $g \leqslant z \leqslant h$ простору векторне поле для функцій $f_1(x,y,z)$, $f_2(x,y,z)$, $f_3(x,y,z)$ (перша координата вектора є функцією $f_1(x,y,z)$, друга – $f_2(x,y,z)$, а третя – $f_3(x,y,z)$). Фактично команда `fieldplot3d` створює імітацію графіка функції $f : \mathbb{R}^3 \to \mathbb{R}^3$. Приклад:

```
> fieldplot3d([x+3*y*z,sin(x)+4*y*z,x+y+z],x=-5..5,
  y=-5..5,z=-5..5,arrows=THICK);
```



§ 11.7. Анімація

Часто функція залежить від деякого параметра. Якщо це функція однієї змінної, то, трактуючи параметр як другу змінну, можна побудувати графік функції у тривимірному просторі. Якщо ж параметром є час, то логічно побачити картину поведінки графіка функції в часі, що можна зробити за допомогою розглянутих команд, лише побудувавши кілька статичних графіків для різних значень параметра. У випадку функції трьох змінних її графік взагалі не можна відобразити, але якщо трактувати одну зі змінних як параметр, то можна побудувати кілька статичних графіків для конкретних значень параметра. Є інший спосіб – скористатись анімацією.

Команда

$$\texttt{animate(}f(x,t)\texttt{,}x\texttt{=}a\,..\,b\texttt{,}t\texttt{=}t_1\,..\,t_2\texttt{,}\mathit{opts}\texttt{)}$$

створює двовимірну анімацію для функції $f(x,t)$ $(a \leqslant x \leqslant b)$ з параметром $t \in [t_1, t_2]$. Кількість кадрів для анімації задається опцією `frames` (за замовчуванням – 16 кадрів). Для «запуску» анімації геометричного об'єкта потрібно на контекстній панелі інструментів (яка з'явиться після виконання команди `animate` і виділення мишкою графіка) натиснути кнопку  . Призначення інших інструментів з панелі є інтуїтивно зрозумілим. Приклад:

```
> animate(sin(x+t),x=-Pi..Pi,t=-Pi..Pi);
```

Оскільки зобразити рухомі об'єкти у книзі неможливо, пропонуємо читачам самостійно виконати усі приклади цього параграфа.

Аналогічно до багатьох інших команд відображення графіки у команді `animate` можна використовувати параметрично задані функції. Приклад:

```
> animate([cos(a*t),sin(3*t)^3,t=0..2*Pi],a=0..5,
  numpoints=1000,frames=100);
```

Команда

$$\texttt{animate3d}(f(x,y,t), x\texttt{=}a\texttt{..}b, y\texttt{=}c\texttt{..}d, t\texttt{=}t_1\texttt{..}t_2, opts)$$

створює тривимірну анімацію для функції $f(x,y,t)$ $(a \leqslant x \leqslant b,$ $c \leqslant y \leqslant d)$ з параметром $t \in [t_1, t_2]$. За замовчуванням створюється 8 кадрів. Приклад:

```
> animate3d(sin((x^2+y^2)*t),x=-Pi..Pi,y=-Pi..Pi,
  t=0.1..2,frames=20,grid=[50,50]);
```

Команда `animatecurve` з тими самими аргументами, що і в команди `plot`, відображає хід побудови графіка функції однієї змінної. Приклад:

```
> animatecurve(x*sin(2/x),x=-2..2,numpoints=1000);
```

§ 11.8. Створення написів і об'єднання кількох рисунків в один

Команда

$$\texttt{textplot([}x, y, \texttt{"}\textit{text}\texttt{"]}, \textit{opts}\texttt{)}$$

створює текстовий напис *text* у точці площини з декартовими координатами (x, y). За замовчуванням напис розміщується так, щоб точка (x, y) знаходилась в його центрі. В якості *opts*, крім опцій команди `plot` (§ 4.2), можна використовувати опцію `align=`*value*, де *value* – одне зі значень, які визначають розташування напису відносно точки (x, y), або їх множина. Цими значеннями можуть бути терміни: `left` (лівіше), `right` (правіше), `above` (вище), `below` (нижче).

Команда

$$\texttt{textplot3d([}x, y, z, \texttt{"}\textit{text}\texttt{"]}, \textit{opts}\texttt{)}$$

створює текстовий напис *text* у точці простору з декартовими координатами (x, y, z).

Пакет `plots` містить команду

$$\texttt{display([}p_1, p_2, \ldots, p_n\texttt{]}, \textit{opts}\texttt{)},$$

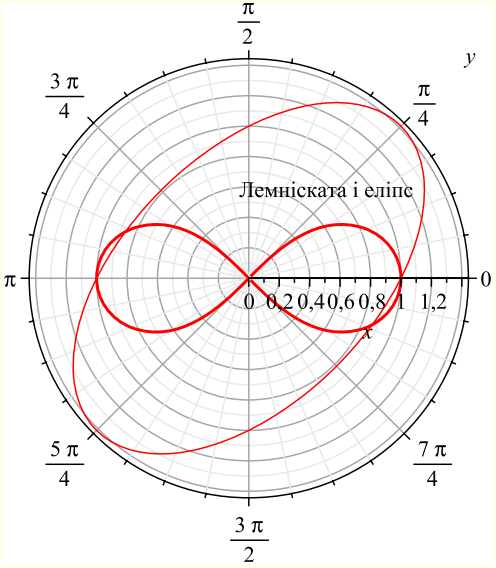
що дозволяє відобразити в одній системі координат кілька графіків p_1, p_2, …, p_n, побудованих командами `plot`, `plot3d` та іншими. Серед p_i можуть бути і команди для створення написів на площині чи у просторі. Опції *opts* діють для всіх команд відображення графіків p_1, p_2, …, p_n. Опція `insequence=true` призначена для створення спеціальної анімації, яка відображатиме графіки p_1, p_2, …, p_n по черзі.

Побудуємо, наприклад, в одній системі координат лемніскату $r = \sqrt{\cos 2\varphi}$ і еліпс $x^2 - xy + y^2 = 1$ з відповідним написом:

```
> p1:=polarplot(sqrt(cos(2*phi)),phi=0..2*Pi,
  thickness=2);
```

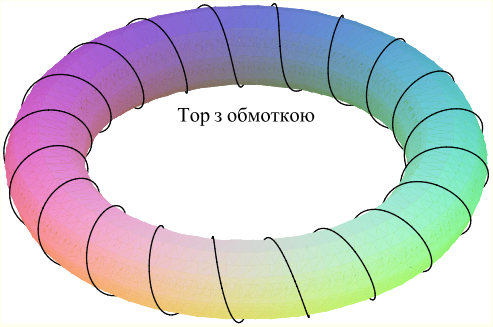
$$p1 := PLOT(\dots)$$

```
> p2:=implicitplot(x^2-x*y+y^2=1,x=-2..2,y=-2..2):
> p3:=textplot([-0.1,0.5,"Лемніската і еліпс"],
  align={above,right}):
> display([p1,p2,p3]);
```



Наведемо тепер приклад створення кількох просторових об'єктів на одному рисунку:

```
> p3d1:=spacecurve([cos(t)*(10+2.1*sin(20*t)),
  sin(t)*(10+2.1*sin(20*t)),2.1*cos(20*t)],t=0..2*Pi,
  numpoints=1000,color=black):
> p3d2:=tubeplot([10*cos(t),10*sin(t),0],t=0..2*Pi,
  radius=2,numpoints=100,style=patchnogrid):
> p3d3:=textplot3d([0,0,3,"Тор з обмоткою"],
  color=black):
> display([p3d1,p3d2,p3d3],scaling=constrained);
```



Питання до розділу 11

1. Як побудувати графік функції у полярній системі координат?
2. Як побудувати графік неявно заданої функції однієї змінної?
3. Як відобразити інтегральну криву задачі Коші для диференціального рівняння чи системи таких рівнянь?
4. Як відобразити графічно розв'язок системи лінійних нерівностей та рівнянь на площині?
5. Які команди використовуються для побудови функції щільності, ліній рівня, поля градієнтів і векторного поля на площині?
6. Як побудувати графіки поверхонь функцій у циліндричній (сферичній) системі координат?
7. Як побудувати поверхню неявно заданої функції двох змінних?
8. Які команди використовуються для побудови ліній рівня, поля градієнтів і векторного поля у просторі?
9. Як створити анімацію на площині та у просторі?
10. Як в одній системі координат відобразити кілька графіків, створених різними командами? Як додавати до графіків написи?

Вправи до розділу 11

1. Побудуйте графік функції $r = 5(1 + \cos^8(18\varphi) + 10\sin^4(12\varphi))$ у полярній системі координат.
2. Побудуйте інтегральну криву задачі Коші $y'' - x\cos y = \cos 3x$, $y(0) = 0$, $y'(0) = 1,5$ на проміжку $[-10, 10]$.
3. Зобразіть на площині розв'язок системи рівнянь і нерівностей $4x + 5y < 3$, $2x + y \geqslant 3$, $x - y = 3$ в області $-1 \leqslant x \leqslant 6$,

$-5 \leqslant y \leqslant 2$. Область, яка задовольняє систему нерівностей, повинна бути зображена зеленим кольором, область, яка не задовольняє систему нерівностей, – чорним кольором, лінія межі строгих нерівностей повинна мати товщину 3 та червоний колір, а лінія межі нестрогих нерівностей та рівностей – товщину 2 та синій колір.

4. Побудуйте функцію щільності, лінії рівня і поле градієнтів функції $z = \sin(x^3 + y^3 - xy)$ у квадраті $-4 \leqslant x \leqslant 4$, $-4 \leqslant y \leqslant 4$.

5. Створіть векторне поле функцій $u = (x - 5y)^3 \sin x$, $v = e^{x+y}$, $-5 \leqslant x, y \leqslant 5$.

6. Побудуйте поверхню $r = \cos^3(\varphi + \theta)$ $(0 \leqslant \varphi \leqslant 2\pi,\ 0 \leqslant \theta \leqslant \pi)$ у сферичній системі координат.

7. Зобразіть поверхню функції $x^3 + y^3 + z^3 = 5 - xyz$ в області $-5 \leqslant x, y, z \leqslant 5$.

8. Створіть анімацію графіка функції $y = x^5 - x^4 \sin(a^3) + 3ax + 1$, $-2 \leqslant x \leqslant 2$, з параметром $-5 \leqslant a \leqslant 5$.

9. Створіть анімацію графіка функції $z = 5t \cos x^3 + ty^2$, $-5 \leqslant x \leqslant 5$, $-5 \leqslant y \leqslant 5$, з параметром $0 \leqslant t \leqslant 4$.

10. Побудуйте в одній системі координат еліпс $\frac{x^2}{16} + \frac{y^2}{9} = 1$ і астроїду $x = 4\cos^3 t$, $y = 3\sin^3 t$, $0 \leqslant t \leqslant 2\pi$. Додайте напис «Астроїда в еліпсі» всередину астроїди.

Розділ 12. Спеціальні засоби для графічного відображення розв'язків диференціальних рівнянь

§ 12.1. Команда DEplot

Команди `dsolve` (§ 7.5) часто не достатньо для відшукання розв'язків звичайних диференціальних рівнянь, адже переважна більшість таких рівнянь не інтегруються у квадратурах. За допомогою команди `dsolve` з опцією `type=numeric` (§ 7.6) можна отримати числовий розв'язок задачі Коші, а потім за допомогою команди `odeplot` (§ 11.2) з пакета `plots` (розділ 11) побудувати графік цього розв'язку (інтегральну криву), але і цього не завжди достатньо.

Пакет `DEtools` містить різні засоби для дослідження розв'язків звичайних диференціальних рівнянь. Наприклад, команда `intfactor(`*deqn*`)` знаходить інтегрувальні множники диференціального рівняння *deqn*:

```
> ode:=(2+x^3/y(x))*D(y)(x)+y(x)/x+3*x^2=0;
```

$$ode := \left(2+\frac{x^3}{y(x)}\right)\mathrm{D}(y)(x)+\frac{y(x)}{x}+3\,x^2=0$$

```
> with(DEtools): intfactor(ode);
```

$$\frac{1}{7\,y(x)+6\,x^3}$$

Ми розглянемо лише одну групу команд пакета `DEtools`, призначену для побудови різних видів графіків розв'язків диференціальних рівнянь.

Команда `DEplot` дає можливість знайти наближений розв'язок одного звичайного диференціального рівняння чи системи таких рівнянь і побудувати його графік на площині. Формат команди:

`DEplot(`*deqns*, *vars*, *trange*, *inits*, *xranges*, *opts*`)`

Параметр *deqns* задає звичайне диференціальне рівняння або множину звичайних диференціальних рівнянь, розв'язаних відносно старших похідних (канонічну систему). Рівняння записуються так само, як і для команди `dsolve` (§ 7.5). Шукані функції подаються параметром *vars*, а у випадку системи вони записуються у вигляді множини. Параметром *trange* визначається діапазон зміни незалежної змінної у вигляді t=a..b.

Початкові умови задають параметром-списком *inits*, елементами якого є списки. Кожен елемент-список містить початкові умови, які визначають інтегральну криву (чи її проекцію) диференціального рівняння або системи, що відображається на графіку. Кількість елементів-списків параметра *inits* відповідає кількості інтегральних кривих. Навіть якщо початкові умови задаються лише для однієї інтегральної кривої, то зовнішні квадратні дужки зберігаються. Необов'язкові параметри *xranges* задають діапазони зміни невідомих функцій і записуються у вигляді $x(t)$=x_1..x_2.

Для автономної системи двох диференціальних рівнянь першого порядку (праві частини не залежать від незалежної змінної) або одного диференціального рівняння, розв'язаного відносно похідної, додатково будується відповідне поле напрямів. У цьому випадку задавати початкові умови *inits* необов'язково, але замість них треба вказувати параметри *xranges*.

В якості *opts* можна використовувати більшість опцій команди `plot` (§ 4.2) і опцію `method` команди `dsolve` (§ 7.6, с. 136). Крім того, є кілька специфічних опцій, опис яких наведемо у вигляді таблиці.

Опція	Опис
`arrows`	Задає вигляд стрілок для відображення поля напрямів. Доступні значення: `small` (за замовчуванням), `medium`, `large`, `line`, `none`
`color`	Вказує колір стрілок для поля напрямів

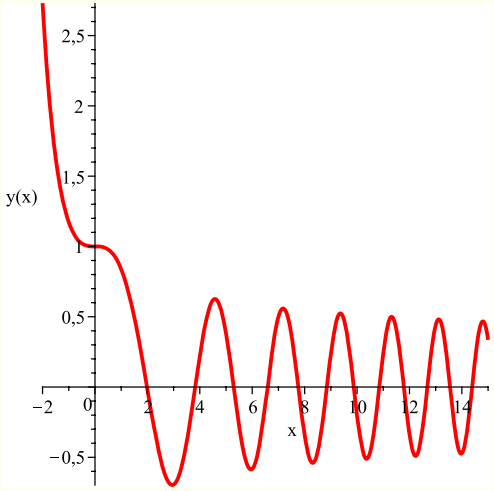
Опція	Опис
dirgrid	Визначає кількість точок сітки в горизонтальному і вертикальному напрямах для відображення векторів поля напрямів. Значення задається у формі списку з двох цілих чисел, перше з яких відповідає горизонтальному напрямку, а друге – вертикальному. Мінімальне допустиме значення [2,2], максимальне – [20,20]
linecolor	Задає колір ліній розв'язків. Якщо на одному графіку будується кілька інтегральних кривих, то можна задати список значень кольорів для відображення відповідних розв'язків
scene	Задається у вигляді двоелементного списку шуканих функцій чи незалежної змінної і визначає, що виводиться на графіку. Зокрема, значення за замовчуванням scene=[x(t),y(t)] означає, що по горизонтальній осі графіка відображається функція $x(t)$, а по вертикальній – $y(t)$ (буде фазовий портрет). Щоб побачити, наприклад, залежність $y(t)$ від t, треба вказати scene=[t,y(t)]
stepsize	Визначає крок зміни незалежної змінної для відображення точок графіка. За замовчуванням дорівнює |b-a|/20, де a і b – межі діапазону зміни незалежної змінної

Приклад 1. Побудувати інтегральну криву задачі Коші $y'' + xy = 0$, $y(0) = 1$, $y'(0) = 0$ на відрізку $x \in [-2, 15]$:

```
> deq:=(D@@2)(y)(x)+x*y(x)=0;
```

$$deq := D^{(2)}(y)(x) + x\,y(x) = 0$$

```
> DEplot(deq,y(x),x=-2..15, [[y(0)=1,D(y)(0)=0]],
  stepsize=0.01,linecolor=red);
```

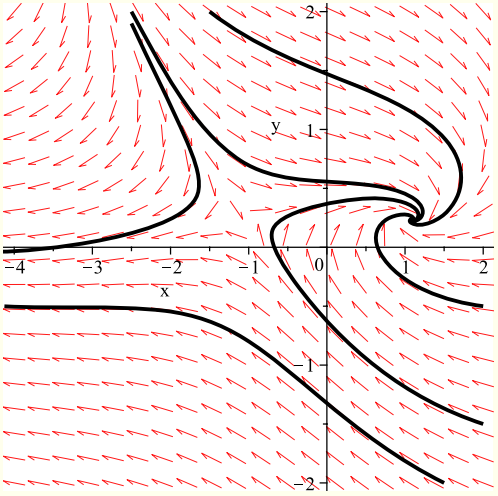


Приклад 2. Побудувати графіки функцій $y = y(x)$, $x = x(t)$, $y = y(t)$ розв'язків системи диференціальних рівнянь

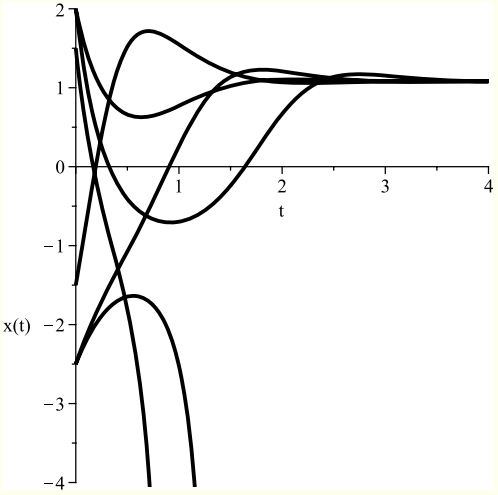
$$\begin{cases} \frac{dx}{dt} = 5y - x^2, \\ \frac{dy}{dt} = -2y + \cos x \end{cases} \qquad (t \in [0,4], \;\; x \in [-4,2], \;\; y \in [-2,2]),$$

які відповідають шести наборам початкових умов: $x(0) = 2$, $y(0) = -0{,}5$; $x(0) = 2$, $y(0) = -1{,}5$; $x(0) = 1{,}5$, $y(0) = -2$; $x(0) = -2{,}5$, $y(0) = 2$; $x(0) = -2{,}5$, $y(0) = 1{,}9$; $x(0) = -1{,}5$, $y(0) = 2$.

```
> deqs:={D(x)(t)=5*y(t)-x(t)^2, D(y)(t)=-2*y(t)+
  cos(x(t))}:
> initconds:=[[x(0)=2,y(0)=-0.5], [x(0)=2,y(0)=-1.5],
  [x(0)=1.5,y(0)=-2], [x(0)=-2.5,y(0)=2],
  [x(0)=-2.5,y(0)=1.9], [x(0)=-1.5,y(0)=2]]:
> DEplot(deqs,{x(t),y(t)},t=0..4,initconds,x(t)=-4..2,
  y(t)=-2..2,stepsize=0.01,linecolor=black);
```



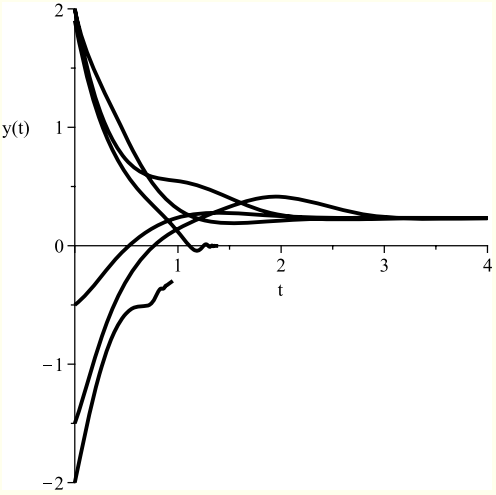

```
> DEplot(deqs,{x(t),y(t)},t=0..4,initconds,x(t)=-4..2,
  stepsize=0.01,linecolor=black,scene=[t,x(t)]);
```




```
> DEplot(deqs,{x(t),y(t)},t=0..4,initconds,y(t)=-2..2,
  stepsize=0.01,linecolor=black,scene=[t,y(t)]);
```

Warning, plot may be incomplete, the following errors(s) were

issued:
cannot evaluate the solution further right of .95269920, probably a singularity
Warning, plot may be incomplete, the following errors(s) were issued:
cannot evaluate the solution further right of 1.3995443, probably a singularity



У останньому випадку графіки побудовані, але не повністю, крім того, наведені повідомлення про нештатні ситуації: два розв'язки з шести не можуть бути продовжені на весь відрізок $t \in [0, 4]$. Для нелінійних звичайних диференціальних рівнянь (систем) така ситуація є доволі типовою.

§ 12.2. Додаткові можливості пакета `DEtools` для графічного подання розв'язків

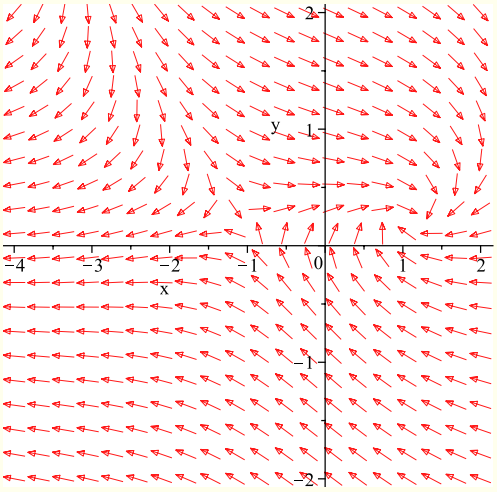
Для побудови поля напрямів автономної системи двох диференціальних рівнянь першого порядку або одного диференціального рівняння, розв'язаного відносно похідної, синоні-

мом до команди DEplot є команда

dfieldplot(*deqns*, *vars*, *trange*, *xranges*, *opts*)

з аналогічними аргументами. Побудуємо поле напрямів системи диференціальних рівнянь з попереднього прикладу:

```
> dfieldplot(deqs,{x(t),y(t)},t=0..4,x(t)=-4..2,
  y(t)=-2..2,arrows=medium);
```



Для побудови фазового портрета системи диференціальних рівнянь або інтегральної кривої одного диференціального рівняння синонімом до команди DEplot є команда

phaseportrait(*deqns*, *vars*, *trange*, *inits*, *xranges*, *opts*)

з аналогічними аргументами. Пропонуємо читачам самостійно побудувати фазовий портрет системи диференціальних рівнянь з початковими умовами з прикладу 2 § 12.1 і порівняти його з першим рисунком на стор. 217.

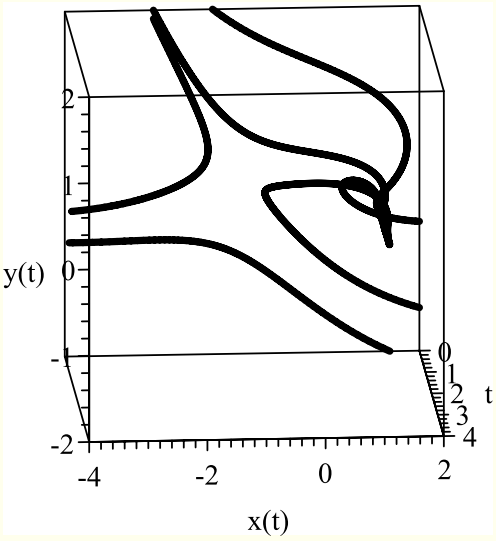
Команда

DEplot3d(*deqns*, *vars*, *trange*, *inits*, *xranges*, *opts*)

відображає у просторі розв'язки системи диференціальних рівнянь. Зміст всіх її аргументів такий самий, як і у команди `DEplot`. Поле напрямів не будується, тому не використовуються опції `arrows`, `dirgrid` і `color`. Значення опції `scene` задають у вигляді трьохелементного списку.

Побудуємо інтегральні криві з прикладу 2 § 12.1 у просторі:

```
> DEplot3d(deqs,{x(t),y(t)},t=0..4,initconds,
  x(t)=-4..2,y(t)=-2..2,stepsize=0.1,linecolor=black);
```



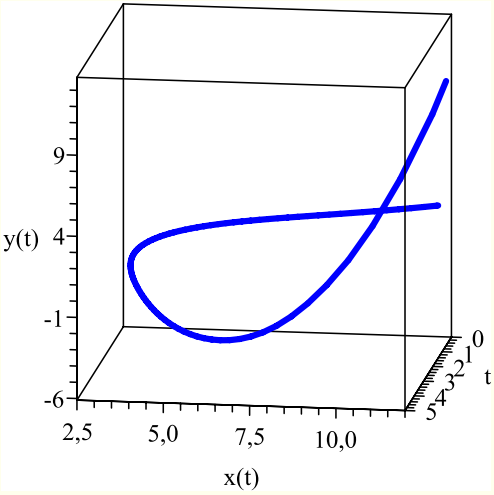
Побудуємо у просторі залежність між t, $x(t)$, $y(t)$ ($t \in [0,5]$, $x \in [2.5,12]$) для системи диференціальних рівнянь

$$\begin{cases} \frac{dx}{dt} = x + \sin y + 5z, \\ \frac{dy}{dt} = 4x - 3y - z^2, \\ \frac{dz}{dt} = 5x + 2y - xz \end{cases}$$

з початковими умовами $x(1) = 3$, $y(1) = -1$, $z(1) = 0$:

```
> DEplot3d({D(x)(t)=x(t)+sin(y(t))+5*z(t),
```

```
D(y)(t)=4*x(t)-3*y(t)-z(t)^2,
D(z)(t)=5*x(t)+2*y(t)-x(t)*z(t)},
{x(t),y(t),z(t)}, t=0..5, [[x(1)=3,y(1)=-1,z(1)=0]],
x(t)=2.5..12, linecolor=blue, scene=[t,x(t),y(t)],
stepsize=0.01);
```



§ 12.3. Команда PDEplot

Для побудови інтегральної поверхні диференціального рівняння з частинними похідними першого порядку, яка проходить через задану криву, у пакеті `PDEtools` призначена команда

`PDEplot(`*PDE*, *inits*, *srange*, *opts*`)`.

Тут *PDE* – рівняння з частинними похідними першого порядку відносно невідомої функції n ($n > 1$) змінних. Рівняння *PDE* записують так само, як для команди `pdsolve` (§ 7.7). Початкові умови задають параметром *inits* у вигляді списку з $n+1$ елементів, які визначають у параметричній формі криву

у $(n+1)$-вимірному просторі, через яку проходить інтегральна поверхня диференціального рівняння. Ці елементи повинні бути виразами, залежними від $n-1$ параметра. Параметром *srange* задають список діапазонів (або один діапазон) зміни кожного параметра, який використовується у початкових умовах, у вигляді $s{=}s_1..s_2$, $t{=}t_1..t_2$ і т. д. Зокрема, у випадку двох незалежних змінних список *inits* повинен містити три вирази, залежні від одного параметра.

В якості *opts* може використовуватись більшість опцій команди `plot3d` (§ 4.4) і опція `method` команди `dsolve` (§ 7.6, с. 136). Одна з найважливіших додаткових опцій $xi{=}ximin..ximax$ дозволяє задати діапазон, що обмежуватиме за змінною xi частину поверхні, яку потрібно відобразити. Аналогічно, опція $u{=}umin..umax$ накладає обмеження на функцію u. Кількість таких опцій може бути від 0 до $n+1$. Без цих опцій необмежену поверхню відобразити неможливо. Інші опції команди `PDEplot` наведено у таблиці.

Опція	Опис
`animate`	Опція `animate=true` означає, що крім поверхні, відображатиметься також анімація. Опція `animate=only` вказує на те, що замість поверхні відображається лише анімація. Для $n>2$ за замовчуванням використовується `animate=only`
`basechar`	Опція `basechar=true` означає, що потрібно додатково побудувати криві характеристик. Якщо `basechar=only`, то відображатимуться лише крива початкових даних і криві характеристик (сама поверхня будуватись не буде)
`color`	Задає колір інтегральної поверхні
`initcolor`	Задає колір лінії (поверхні) початкових умов

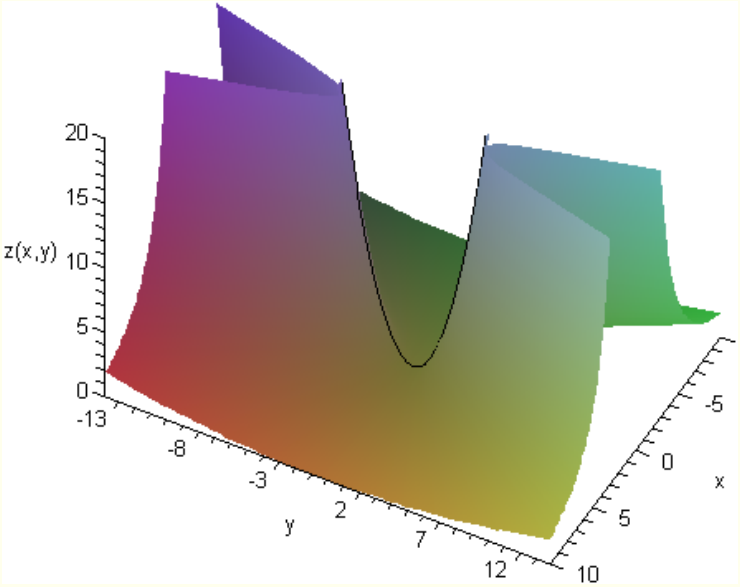
Опція	Опис
numchar	Визначає кількість точок з початкових умов, через які проходять криві, що формують інтегральну поверхню. За замовчуванням numchar=20 для $n = 2$, що може бути недостатнім для акуратного відображення складної поверхні. Для $n > 2$ значенням опції має бути список з $n - 1$ чисел
scene	Задається у вигляді трьохелементного списку шуканої функції та незалежних змінних і визначає те, що виводиться на графіку, наприклад, scene=[x,y,u] для функції $u(x, y, z)$
stepsize	Визначає відстань між обчислюваними точками вздовж кожної з характеристик для відображення поверхні. За замовчуванням дорівнює 0,25

Приклад 1. Знайти інтегральну поверхню рівняння з частинними похідними $(x^2 + 1)\frac{\partial z}{\partial x} + xy\frac{\partial z}{\partial y} = 0$, яка при $x = 0$ проходить через криву $z = y^2$:

```
> eq:=(x^2+1)*diff(z(x,y),x)+x*y*diff(z(x,y),y)=0;
```

$$eq := \left(x^2+1\right)\left(\frac{\partial}{\partial x}z(x,y)\right) + x\,y\left(\frac{\partial}{\partial y}z(x,y)\right) = 0$$

```
> PDEtools[PDEplot](eq, [0,t,t^2], t=-10..10, z=0..20,
  x=-10..10, numchar=200, scaling=constrained);
```



Приклад 2. Графічно зобразити розв'язок рівняння з частинними похідними $(y+z)\frac{\partial u}{\partial x}+(z+x)\frac{\partial u}{\partial y}+(x+y)\frac{\partial u}{\partial z}=u$, якщо $u(x,y,x+y)=\sin y$:

```
> PDEtools[PDEplot]((y+z)*diff(u(x,y,z),x)+(z+x)*
  diff(u(x,y,z),y)+(x+y)*diff(u(x,y,z),z)=u(x,y,z),
  [t2,t1,t1+t2,sin(t1)], t1=-10..10, t2=-10..10,
  x=-10..10, y=-10..10, u=-10..10, numchar=[20,20]);
```

В результаті цієї команди (її пропонуємо читачам виконати самостійно) буде створена анімація за змінною z залежності $u(x,y,z)$ від x і y. Тут не використовувалась опція `scene=[x,y,u]`, бо її роль виконують діапазони `x=-10..10, y=-10..10, u=-10..10`.

Інший спосіб графічного подання наближених розв'язків рівнянь з частинними похідними наведено в § 7.8.

Питання до розділу 12

1. Як графічно подати розв'язок задачі Коші для звичайного диференціального рівняння чи системи таких рівнянь на площині? Які необов'язкові параметри і з якою метою при цьому використовують?

2. Як графічно подати розв'язок задачі Коші для звичайного диференціального рівняння чи системи таких рівнянь у просторі?

3. Як графічно подати розв'язок задачі Коші для рівняння з частинними похідними?

Вправи до розділу 12

1. Побудувати акуратні графіки функцій $y = y(x)$, $x = x(t)$, $y = y(t)$ розв'язків системи диференціальних рівнянь

$$\begin{cases} \frac{dx}{dt} = 4xy^3, \\ \frac{dy}{dt} = 2x + 7y^3 \end{cases} \quad (t \in [-1,1], \quad x \in [-10,10], \quad y \in [-4,4]),$$

які відповідають двом наборам початкових умов: $x(0) = 2$, $y(0) = 1$; $x(0) = -2{,}5$, $y(0) = 0{,}9$.

2. Побудувати у просторі акуратну залежність між t, $x(t)$, $y(t)$ ($t \in [-1,1]$) для системи диференціальних рівнянь з попередньої вправи з тими самими початковими умовами.

3. Побудувати інтегральну поверхню рівняння з частинними похідними

$$(xy - 3)\frac{\partial z}{\partial x} - 2xy\frac{\partial z}{\partial y} = 0$$

з початковою умовою $z(x,1) = x + 1$ ($x \in [-10,10]$, $y \in [-10,10]$).

Розділ 13. Використання спеціальних пакетів для інтегрування функцій багатьох змінних

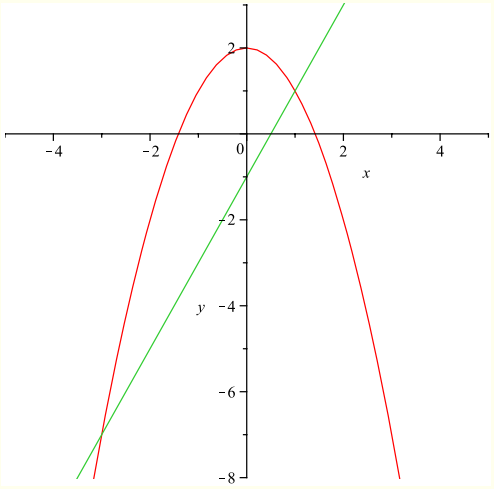
§ 13.1. Подвійні інтеграли

Якщо подвійний інтеграл зведений до повторного, то для його знаходження можна використати команду `int` (§ 6.8).

Обчислимо, наприклад, подвійний інтеграл $\iint\limits_D (x-y)\,dxdy$, де область D обмежена кривими $y = 2 - x^2$, $y = 2x - 1$.

Побудуємо спочатку область інтегрування і знайдемо точки перетину ліній $y = 2 - x^2$ і $y = 2x - 1$:

```
> plot([2-x^2, 2*x-1], x=-5..5, y=-8..3);
```




```
> solve({y=2-x^2,y=2*x-1}, {x,y});
```

$$\{x = -3, y = -7\},\ \{x = 1, y = 1\}$$

Отже, криві $y = 2 - x^2$, $y = 2x - 1$ перетинаються у точках $(-3, -7)$ і $(1, 1)$, а тому двократний інтеграл по області

D записуємо у вигляді

$$\iint\limits_D (x-y)\,dxdy = \int\limits_{-3}^{1} dx \int\limits_{2x-1}^{2-x^2} (x-y)\,dy.$$

Тоді

```
> Int(Int(x-y, y=2*x-1..2-x^2), x=-3..1)=
  int(int(x-y, y=2*x-1..2-x^2), x=-3..1);
```

$$\int_{-3}^{1}\int_{2\,x-1}^{2-x^2} (x-y)\,\mathrm{d}y\,\mathrm{d}x = \frac{64}{15}$$

Для обчислення подвійних інтегралів $\iint\limits_D f(x,y)\,dx\,dy$ можна використовувати спеціальну команду бібліотеки `student`:

$$\texttt{Doubleint}(f(x,y), D),$$

де D – область інтегрування, записана в одному з трьох форматів:

1) x=x_1..x_2, y=y_1..y_2, де числа x_1, x_2, y_1, y_2 задають прямокутну область інтегрування;

2) x=$f_1(y)$..$f_2(y)$, y=y_1..y_2, де лінії $f_1(y)$, $f_2(y)$ обмежують область інтегрування зліва і справа на проміжку $y \in [y_1, y_2]$;

3) x=x_1..x_2, y=$g_1(x)$..$g_2(x)$, де лінії $g_1(x)$, $g_2(x)$ обмежують область інтегрування знизу і зверху на проміжку $x \in [x_1, x_2]$.

Команда `Doubleint` – це команда відкладеної дії, а тому для одержання значення інтеграла потрібно використати команду `value(%)`. Знайдемо, наприклад, попередній подвійний інтеграл:

```
> with(student):
> Doubleint((x-y),y=2*x-1..2-x^2,x=-3..1): %=value(%);
```

$$\int_{-3}^{1}\int_{2\,x-1}^{2-x^2} (x-y)\,\mathrm{d}y\,\mathrm{d}x = \frac{64}{15}$$

Ще один спосіб обчислення подвійних інтегралів надає команда `int(`f`,`dom`)` з пакета `VectorCalculus`, яка розширює можливості команди `int` зі стандартної бібліотеки. Цей спосіб особливо зручний, коли область інтегрування dom має стандартний вигляд, тобто є:

1) трикутником `[`x,y`]=Triangle(<`x_1,y_1`>,<`x_2,y_2`>,<`x_3,y_3`>)`, заданим координатами вершин;

2) прямокутником `[`x,y`]=Rectangle(`x_1`..`x_2`,` y_1`..`y_2`)`;

3) кругом `[`x,y`]=Circle(<`x_0,y_0`>,`R`)`, заданим центром (x_0,y_0) і радіусом R;

4) еліпсом `[`x,y`]=Ellipse(`x`^2/`a`^2+`y`^2/`b`^2-`R`^2)`, заданим у канонічному вигляді;

5) сектором круга або еліпса від кута φ_1 до φ_2, наприклад, `[`x,y`]=Sector(Circle(<`x_0,y_0`>,` R`),`φ_1,φ_2`)`.

Для областей загальнішого вигляду використовують функцію `Region`, наприклад, область D з попереднього прикладу можна задати виразом `Region(-3..1, 2*x-1..2-x^2)`, а відповідний інтеграл обчислюємо так:

```
> with(VectorCalculus): Q:=Region(-3..1,2*x-1..2-x^2):
> int(x-y, [x,y]=Q);
```

$$\frac{64}{15}$$

Додаткова опція `inert` перетворює команду `int` з пакета `VectorCalculus` (як і кожну команду цього пакета для обчислення інтегралів) на відкладену, наприклад,

```
> int(x-y, [x,y]=Q,inert)=int(x-y, [x,y]=Q);
```

$$\int_{-3}^{1}\int_{2\,x-1}^{2-x^2}(x-y)\,\mathrm{d}y\,\mathrm{d}x=\frac{64}{15}$$

§ 13.2. Потрійні інтеграли

Якщо потрійний інтеграл зведений до повторного, то для його обчислення можна тричі використати команду `int`. Знайдемо, наприклад, потрійний інтеграл від функції $(x+1)y^2z^2$,

якщо областю інтегрування є паралелепіпед $0 \leqslant x \leqslant 1$, $1 \leqslant y \leqslant 2$, $2 \leqslant x \leqslant 3$:

```
> Int(Int(Int((x+1)*y^2*z^2,z=2..3), y=1..2), x=0..1)=
  int(int(int((x+1)*y^2*z^2,z=2..3), y=1..2), x=0..1);
```

$$\int_0^1 \int_1^2 \int_2^3 (x+1)y^2 z^2 \mathrm{d}z\,\mathrm{d}y\,\mathrm{d}x = \frac{133}{6}$$

У пакеті `student` для знаходження потрійних інтегралів призначена команда відкладеної дії `Tripleint`$(f(x,y,z),Q)$, де Q – область інтегрування, формат запису якої є аналогічним до формату області інтегрування для команди `Doubleint`. Для одержання числового значення потрійного інтеграла після команди `Tripleint` потрібно виконати команду `value(%)`.

Знайдемо потрійний інтеграл $\iiint\limits_Q x^2 y\sqrt{z}\,dx\,dy\,dz$, де Q – тіло, обмежене поверхнями $z = 0$, $z = y$, $y = x^2$, $y = 1$ (рисунок пропонуємо зробити самостійно).

```
> with(student):
> Tripleint(x^2*y*sqrt(z), z=0..y, y=x^2..1, x=-1..1):
> %=value(%);
```

$$\int_{-1}^1 \int_{x^2}^1 \int_0^y x^2 y\sqrt{z}\,\mathrm{d}z\,\mathrm{d}y\,\mathrm{d}x = \frac{4}{45}$$

Кратні інтеграли можна також шукати за допомогою команди

`MultiInt`$(f(x,y,z),Q,opts)$

з пакета `Student[MultivariateCalculus]`. Область Q тут задається так само, як і в команді `Tripleint`. Опцією `coordinates` у команді `MultiInt` можна вказати систему координат: `cartesian`$[x,y,z]$ (декартову), `cylindrical`$[r,\varphi,z]$ (циліндричну), `spherical`$[r,\varphi,\theta]$ (сферичну). Крім того, є три опції для виведення результату:

`output=value` – видає значення інтеграла (діє за замовчуванням);

`output=integral` – видає формулу інтеграла, не обчислюючи його;

`output=steps` – видає окремі етапи обчислення.

Знайдемо, наприклад, інтеграл $\iiint\limits_Q (y+z)\,dxdydz$, де Q – область, обмежена еліптичним параболоїдом $z = x^2+y^2$ і площиною $z = 4$. Оскільки проекцією області Q на площину Oxy є круг радіуса 2, то інтеграл зручно шукати у циліндричній системі координат:

```
> simplify(subs([x=r*cos(t),y=r*sin(t)], [x^2+y^2,
  y+z]));
```

$$\left[r^2, r\sin(t)+z\right]$$

```
> with(Student[MultivariateCalculus]):
> MultiInt(r*sin(t)+z, z=r^2..4, r=0..2, t=0..2*Pi,
  coordinates=cylindrical[r,t,z], output=value);
```

$$\frac{64}{3}\pi$$

За допомогою команди `int` з пакета `VectorCalculus` можна обчислювати також потрійні інтеграли. Області інтегрування при цьому повинні бути стандартними. До таких областей належать:

1) `[`x, y, z`]=Sphere(<`x_0, y_0, z_0`>,` R`)` – сфера радіуса R з центром у точці (x_0, y_0, z_0);

2) `[`x, y, z`]=Parallelepiped(`$x_1..x_2,\ y_1..y_2,\ z_1..z_2$`)` – прямокутний паралелепіпед $x_1 \leqslant x \leqslant x_2$, $y_1 \leqslant y \leqslant y_2$, $z_1 \leqslant z \leqslant z_2$;

3) `[`x, y, z`]=Tetrahedron(<`x_1, y_1, z_1`>,<`x_2, y_2, z_2`>,<`x_3, y_3, z_3`>,<`x_4, y_4, z_4`>)` – піраміда, задана координатами вершин;

4) `[`x, y, z`]=Region(`$a..b,\ f_1(x)..f_2(x),\ g_1(x,y)..g_2(x,y)$`)` – область з межами $x = a$, $x = b$, $y = f_1(x)$, $y = f_2(x)$, $z = g_1(x,y)$, $z = g_2(x,y)$.

Знайдемо, наприклад, потрійний інтеграл $\iiint\limits_Q xyz\,dxdydz$, де Q – область, обмежена координатними площинами і площиною $x+y+z=3$:

```
> with(VectorCalculus):
> int(x*y*z, [x,y,z]=Tetrahedron(<0,0,0>,<3,0,0>,
  <0,3,0>,<0,0,3>));
```

$$\frac{81}{80}$$

Команда `int` пакета `VectorCalculus` дозволяє також обчислювати кратні інтеграли кратності більшої від трьох. Але при цьому область можна задавати лише за допомогою команд `Parallelepiped` і `Region`, кількість аргументів у яких має відповідати кратності інтеграла.

§ 13.3. Криволінійні інтеграли першого роду

Криволінійні інтеграли першого роду можна обчислювати за допомогою команди `PathInt` з пакета `VectorCalculus`. Аргументи цієї команди є подібними до аргументів команди `int` з цього самого пакета. Команда `PathInt` дозволяє обчислювати криволінійні інтеграли як на площині, так і у просторі.

Обчислимо, наприклад, криволінійний інтеграл

$$\int\limits_L \sqrt{1+4y+9xz}\,dl,$$

де L – дуга кривої, заданої у параметричній формі $x=t$, $y=t^2$, $z=t^3$, $0\leqslant t\leqslant 2$:

```
> with(VectorCalculus): f:=(x,y,z)->sqrt(1+4*y+9*x*z):
> PathInt(f(x,y,z),[x,y,z]=Path(<t,t^2,t^3>, t=0..2));
```

$$\frac{1054}{15}$$

Обчислення криволінійного інтеграла на площині здійснюється аналогічно. Команда `PathInt` особливо зручна тоді, коли крива інтегрування є відрізком прямої, ламаною, колом, еліпсом, дугою кола чи еліпса. Розглянемо форми представлення команди `PathInt` у цих випадках.

Якщо крива інтегрування є відрізком прямої, то команда `PathInt` задається з аргументом `[`x, y, z`]=Line(<`x_1, y_1, z_1`>, <`x_2, y_2, z_2`>)`, де (x_1, y_1, z_1) і (x_2, y_2, z_2) – початок і кінець відрізка відповідно.

Знайдемо, наприклад, інтеграл $\int\limits_{AB} (x + y^2 + z^3)\, dl$, де AB – відрізок, який з'єднує точки $A(1, 1, 1)$, $B(3, 3, 3)$:

```
> PathInt(x+y^2+z^3, [x,y,z]=Line(<1,1,1>,<3,3,3>));
```

$$\frac{98}{3}\sqrt{3}$$

Якщо кривою інтегрування є ламана $A_1A_2 \dots A_n$ з вершинами $A_j = (x_j, y_j, z_j)$, $j = 1, \dots n$, то команда `PathInt` повинна бути задана з аргументом `[`x, y, z`]=LineSegments()`, де у круглих дужках через кому потрібно вказати координати вершин $A_1A_2 \dots A_n$: `<`x_1, y_1, z_1`>`, `<`x_2, y_2, z_2`>` і т. д.

Знайдемо інтеграл $\int\limits_{L} (x+y+z)\, dl$, де L – ламана $A_1A_2A_3A_4$ з вершинами $A_1(0, 0, 0)$, $A_2(3, 0, 1)$, $A_3(1, 1, 2)$, $A_4(3, -1, -2)$:

```
> PathInt(x+y+z,[x,y,z]=LineSegments(<0,0,0>, <3,0,1>,
<1,1,2>, <3,-1,-2>));
```

$$2\sqrt{10} + 8\sqrt{6}$$

Для обчислення криволінійного інтеграла по колу команда `PathInt` задається з аргументом `[`x, y`]=Circle(<`x_0, y_0`>,`R`)`, де `<`x_0, y_0`>` – центр кола, R – радіус.

Знайдемо інтеграл $\int\limits_{L} xy\, dl$, де L – коло $(x-1)^2+(y-3)^2 = 16$, скориставшись відкладеною формою команди `PathInt`:

```
> PathInt(x*y,[x,y]=Circle(<1,3>,4),inert):%=value(%);
```

$$\int_0^{2\,\pi} 4\,(1+4\cos(t))\,(3+4\sin(t))\,\sqrt{\sin(t)^2+\cos(t)^2}\,\mathrm{d}t = 24\,\pi$$

Як бачимо, відкладена форма команди `PathInt` відображає не криволінійний інтеграл першого роду, а відповідний йому інтеграл Рімана. Зрозуміло, що підінтегральний вираз тут можна спростити.

Для обчислення криволінійного інтеграла по еліпсу у команді `PathInt` використовується аргумент `[`x, y`]=Ellipse()`, де у круглих дужках розміщується ліва частина рівняння еліпса (якщо права частина дорівнює нулю).

Знайдемо у такий спосіб інтеграл $\int\limits_L |xy|\,dl$, де L – еліпс $x^2 + y^2/4 = 1$:

```
> PathInt(abs(x*y),[x,y]=Ellipse(x^2+y^2/4-1));
```

$$\frac{56}{9}$$

Якщо потрібно знайти криволінійний інтеграл першого роду по дузі кола чи еліпса, то у команді `PathInt` необхідно писати `[`x, y`]=Arc(Circle(<`x_0, y_0`>,`R`),`φ_1, φ_2`)` або `[`x, y`]=Arc(Ellipse(`x`^2/`a`^2+`y`^2/`b`^2-1),`φ_1, φ_2`)` відповідно.

Знайдемо, наприклад, інтеграл $\int\limits_L xy\,dl$, де L – дуга кола $x^2 + y^2 = 1$, яка належить другій чверті ($\pi/2 \leqslant x \leqslant \pi$):

```
> PathInt(x*y,[x,y]=Arc(Circle(<0,0>, 1), Pi/2, Pi));
```

$$-\frac{1}{2}$$

§ 13.4. Криволінійні інтеграли другого роду

Криволінійні інтеграли другого роду можна обчислювати за допомогою команди `int` зі стандартної бібліотеки. Обчислимо, наприклад, криволінійний інтеграл $\int\limits_L y^2dx + x^2dy$, де L – дуга першої арки циклоїди $x = a(t-\sin t)$, $y = a(1-\cos t)$, $0 \leqslant t \leqslant 2\pi$:

```
> f:=y^2*diff(x,t)+x^2*diff(y,t):
> x:=a*(t-sin(t)): y:=a*(1-cos(t)):
> Int(f,t=0..2*Pi)=int(f,t=0..2*Pi);
```

$$\int_0^{2\,\pi} \left(a^3 \left(1-\cos(t)\right)^3 + a^3 \left(t-\sin(t)\right)^2 \sin(t) \right) \mathrm{d}t = -6\,a^3\pi^2 + 5\,a^3\pi$$

Для обчислення криволінійних інтегралів другого роду $\int\limits_L P\,dx + Q\,dy + R\,dz$ у пакеті `VectorCalculus` призначена команда

$$\texttt{LineInt(VectorField(<}P, Q, R\texttt{>},c\texttt{),}$$
$$\texttt{Path(<}x(t), y(t), z(t)\texttt{>},t\texttt{=}t_1\texttt{..}t_2\texttt{)).}$$

Команда `Path` використовується для задання кривої у параметричній формі. Для правильного задання команди `LineInt` потрібно вказати координати: для плоского поля – за допомогою глобальної команди `SetCoordinates(cartesian[`x,y`])` або локально, за допомогою `cartesian[`x,y`]` в якості параметра c команди `VectorField`, для векторного поля у просторі – команди `SetCoordinates(cartesian[`x,y,z`])` або `cartesian[`x,y,z`]` в якості параметра c команди `VectorField`. Крім того, можна використовувати будь-яку з допустимих систем координат, наприклад, `polar[`r,φ`]` (полярну), `cylindrical[`r,φ,z`]` (циліндричну).

Обчислимо інтеграл з попереднього прикладу:

```
> with(VectorCalculus):SetCoordinates(cartesian[x,y]):
```

```
> LineInt(VectorField(<y^2,x^2>), Path(<a*(t-sin(t)),
  a*(1-cos(t))>, t=0..2*Pi));
```

$$-6\,a^3\pi^2 + 5\,a^3\pi$$

Команда `LineInt` зручна тоді, коли кривою інтегрування є відрізок, ламана, коло, еліпс, дуга кола чи еліпса. У цих випадках для задання кривої у вираз інтеграла замість функції `Path()` потрібно підставити такі конструкції:

1) `Line(<`x_1, y_1, z_1`>,<`x_2, y_2, z_2`>)` – відрізок прямої, заданий координатами своїх кінців;

2) `LineSegments(<`x_1, y_1, z_1`>,<`x_2, y_2, z_2`>,...,<`x_n, y_n, z_n`>)` – ламана, задана координатами своїх вершин;

3) `Circle(<`x_0, y_0`>,`R`)` – коло з заданим центром і радіусом;

4) `Ellipse(`x`^2/`a`^2+`y`^2/`b`^2-1)` – еліпс, заданий у канонічному вигляді;

5) `Arc(Ellipse(`x`^2/`a`^2+`y`^2/`b`^2-1),`φ_1`,`φ_2`)` – дуга еліпса або `Arc(<`x_0, y_0`>,`R`),`φ_1`,`φ_2`)` – дуга кола.

Знайдемо інтеграл $\int\limits_L y^2\,dx - xy\,dy$, де L – розташована у першій чверті дуга еліпса $x^2/a^2 + y^2/b^2 = 1$.

Спосіб 1. Використаємо команду `LineInt` зі стандартною лінією інтегрування у вигляді дуги еліпса:

```
> SetCoordinates(cartesian[x,y]):
> LineInt(VectorField(<y^2,-x*y>),Arc(Ellipse(x^2/a^2+
  y^2/b^2-1),0,Pi/2)) assuming a>0, b>0;
```

$$-b^2a$$

Спосіб 2. Подамо дугу L у параметричній формі ($x = a\cos t$, $y = b\sin t$, $0 \leqslant t \leqslant \pi/2$) за допомогою функції `Path`:

```
> LineInt(VectorField(<y^2,-x*y>,cartesian[x,y]),
  Path(<a*cos(t),b*sin(t)>, t=0..Pi/2));
```

$$-b^2a$$

§ 13.5. Поверхневі інтеграли першого роду

Оскільки обчислення поверхневого інтеграла першого роду здійснюється за допомогою подвійного інтеграла по проекції поверхні на деяку координатну площину, то після утворення такого інтеграла можна застосовувати будь-яку команду, призначену для обчислення подвійних інтегралів (§ 13.1).

Знайдемо поверхневий інтеграл $\iint\limits_Q \sqrt{1+\frac{y^2}{b^2}+\frac{z^2}{c^2}}\,d\sigma$, де Q – частина гіперболічного параболоїда $x=\frac{y^2}{2b}-\frac{z^2}{2c}$, вирізана циліндром $\left(\frac{y^2}{b^2}+\frac{z^2}{c^2}\right)^2 = a^2\left(\frac{y^2}{b^2}-\frac{z^2}{c^2}\right)$. Легко бачити, що проекцією поверхні інтегрування на площину Oyz є лемніската Бернуллі $\left(\frac{y^2}{b^2}+\frac{z^2}{c^2}\right)^2 = a^2\left(\frac{y^2}{b^2}-\frac{z^2}{c^2}\right)$. Для обчислення інтеграла виконуємо заміну $y=br\cos t$, $z=cr\sin t$.

```
> f:=(y,z)->sqrt(1+y^2/b^2+z^2/c^2):
> g:=(y,z)->y^2/(2*b)-z^2/(2*c):
> A:=(y^2/b^2+z^2/c^2)^2-a^2*(y^2/b^2-z^2/c^2):
> B:=simplify(subs({y=b*r*cos(t), z=c*r*sin(t)}, A));
```

$$B := -r^2\left(2\,a^2\cos(t)^2 - a^2 - r^2\right)$$

```
> solve(B=0,r);
```

$$0,\ 0,\ \sqrt{2\cos(t)^2-1}\,a,\ -\sqrt{2\cos(t)^2-1}\,a$$

Тому $0 \leqslant r \leqslant a\sqrt{\cos 2t}$, $-\frac{\pi}{4} \leqslant t \leqslant \frac{\pi}{4}$, $\frac{3\pi}{4} \leqslant t \leqslant \frac{5\pi}{4}$, причому обидва проміжки по t дають той самий інтеграл. Знайдемо поверхневий інтеграл, звівши його до повторного інтеграла. При цьому для обчислення якобіана переходу від змінних y, z до змінних r, t скористаємось командою `Jacobian` з пакета `VectorCalculus`:

```
> with(VectorCalculus): with(linalg):
> h:=subs({y=b*r*cos(t), z=c*r*sin(t)}, f(y,z)):
> J:=abs(simplify(det(Jacobian(<b*r*cos(t),
```

```
  c*r*sin(t)>,[r,t]))))) assuming b>0, c>0, r>0:
> ds:=sqrt(1+diff(g(y,z),y)^2+diff(g(y,z),z)^2):
> ds1:=simplify(subs({y=b*r*cos(t),z=c*r*sin(t)},ds)):
> Result:=2*int(int(h*ds1*J,r=0..a*sqrt(cos(2*t))),
  t=-Pi/4..Pi/4);
```

$$Result := \frac{1}{8}\, b\, c\, a^4\, \pi + b\, c\, a^2$$

Для обчислення поверхневих інтегралів першого роду призначена команда `SurfaceInt` з пакета `VectorCalculus`. Найчастіше цю команду використовують у форматі:

$$\texttt{SurfaceInt(}f(x,y,z),$$
$$\texttt{[}x,y,z\texttt{]=Surface(<}s,t,z(s,t)\texttt{>, [}s,t\texttt{]=}Obl\texttt{()))}.$$

При цьому поверхня задається графіком функції $z(x,y)$, а в якості параметра *Obl* може бути одна зі стандартних областей для команди `int`, про які йшлося у § 13.1. Крім того, у команді `Surface` можна вказати опції переходу до циліндричної чи сферичної системи координат, наприклад, `coords=cylindrical`.

Обчислимо, наприклад, інтеграл $\iint\limits_G (x^4+2y^2-4x^2z^2-z^4)d\sigma$, де G – частина площини $x+y+z=4$, вирізана циліндром $x^2+z^2=4$.

```
> with(VectorCalculus):
> SurfaceInt(x^4+2*y^2-4*x^2*z^2-z^4,[x,y,z]=Surface
  (<s,4-s-t,t>, [s,t]=Circle(<0,0>,2)));
```

$$\frac{400}{3}\sqrt{3}\,\pi$$

Якщо область інтегрування є сферою або прямокутним паралелепіпедом, то замість команди `Surface` можна використовувати `Sphere(<`x_0, y_0, z_0`>,`R`)` або `Box(`$x_1..x_2$, $y_1..y_2$, $z_1..z_2$`)` відповідно.

§ 13.6. Поверхневі інтеграли другого роду

Обчислення поверхневого інтеграла другого роду також зводиться до обчислення подвійного інтеграла по проекції поверхні на координатну площину. Тому після зведення поверхневого інтеграла до подвійного для його обчислення можна використати будь-яку з команд, призначених для обчислення подвійних інтегралів (§ 13.1). Нагадаємо, що поверхневі інтеграли другого роду по внутрішній стороні поверхні відрізняються від поверхневих інтегралів по зовнішній стороні поверхні лише знаком.

Знайдемо інтеграл $\iint\limits_Q (y^2+2z^2)\,dxdy$, де Q – зовнішня частина поверхні $z=\sqrt{9-x^2}$, яку відтинають площини $z=0$, $y=0$, $y=4$. Нескладно показати, що проекцією заданої поверхні на площину Oxy є прямокутник $-3\leqslant x\leqslant 3$, $0\leqslant y\leqslant 4$, а тому

$$\iint\limits_Q (y^2+2z^2)\,dxdy=\iint\limits_{Q_{xy}}\left(y^2+2\left(\sqrt{9-x^2}\right)^2\right)dxdy=$$
$$=\int\limits_{-3}^{3}dx\int\limits_0^4\left(y^2-2x^2+18\right)\big)\,dy.$$

Застосовуючи команду `int`, маємо

```
> Int(Int(y^2-2*x^2+18, y=0..4), x=-3..3)=
  int(int(y^2-2*x^2+18, y=0..4), x=-3..3);
```

$$\int_{-3}^{3}\int_0^4\left(y^2-2\,x^2+18\right)\mathrm{d}y\,\mathrm{d}x=416$$

У пакеті `VectorCalculus` для обчислення поверхневих ін-

тегралів другого роду

$$\iint\limits_S P\,dydz + Q\,dxdz + R\,dxdy$$

можна використовувати команду `Flux`, записану, наприклад, у такому форматі:

`Flux(VectorField(<`P, Q, R`>, cartesian[`x, y, z`]),`
`Surface(<`$x, y, z(x, y)$`>, `x`=`$x_1 \,..\, x_2$`, `y`=`$y_1(x) \,..\, y_2(x)$`)).`

Вона дозволяє обчислити потік векторного поля, заданого функцією `VectorField(<`P, Q, R`>)`, через зовнішню сторону поверхні $z(x, y)$.

Знайдемо за допомогою команди `Flux` попередній інтеграл:

```
> with(VectorCalculus):
> Flux(VectorField(<0,0,y^2+2*z^2>, cartesian[x,y,z]),
  Surface(<x,y,sqrt(9-x^2)>, x=-3..3, y=0..4));
```

416

Питання до розділу 13

1. Назвіть способи знаходження подвійних інтегралів у Maple. У чому полягають особливості використання пакета `VectorCalculus` для обчислення подвійних інтегралів?

2. Назвіть способи знаходження потрійних інтегралів у Maple. У чому полягають особливості використання пакета `VectorCalculus` для обчислення потрійних інтегралів?

3. Як обчислити криволінійний інтеграл першого роду за допомогою пакета `VectorCalculus`? Назвіть способи задання кривої для криволінійного інтеграла.

4. Як обчислити криволінійний інтеграл другого роду за допомогою пакета `VectorCalculus`?

5. Як обчислювати поверхневі інтеграли першого роду?

6. Як обчислювати поверхневі інтеграли другого роду?

Вправи до розділу 13

1. Обчисліть подвійний інтеграл

$$\iint\limits_{x^2+y^2\leqslant 9} \sin(x^2+y^2+1)\,dxdy.$$

2. Обчисліть потрійний інтеграл

$$\iiint\limits_{E} y\,dxdydz,$$

де E – множина, обмежена поверхнями $x^2+y^2=z$, $x=0$, $y=0$, $z=3$.

3. Обчисліть криволінійний інтеграл першого роду

$$\int\limits_{\Gamma} xy\,dl,$$

де Γ – контур прямокутника, обмеженого прямими $x\pm y=1$, $x\pm y=-1$.

4. Обчисліть криволінійний інтеграл

$$\int\limits_{\Gamma} x\,dy-y\,dx,$$

де крива $\Gamma=\big\{(x,y): x=a\cos^3 t,\ y=a\sin^3 t,\ 0\leqslant t\leqslant 2\pi\big\}$.

5. Обчисліть поверхневий інтеграл першого роду

$$\iint\limits_{x^2+y^2+z^2=a^2} (x^2+y^2)\,ds.$$

6. Обчисліть поверхневий інтеграл другого роду

$$\iint\limits_{S} dxdy,$$

де S – внутрішня поверхня конуса $z=\sqrt{x^2+y^2}$, $0\leqslant z\leqslant 1$.

Розділ 14. Застосування Maple до розв'язування задач аналітичної геометрії на площині

§ 14.1. Створення геометричних об'єктів

У Maple включено спеціальний пакет `geometry`, призначений для розв'язування задач аналітичної геометрії та виконання геометричних побудов на площині.

В усіх командах створення геометричних об'єктів першим параметром задають ім'я створюваного об'єкта, по якому в подальшому можна буде звертатись до нього. Інші параметри залежать від типу об'єкта і способу його визначення. Після створення геометричного об'єкта в області виведення відображатиметься його ім'я. Для відображення створеного об'єкта на площині потрібно використовувати спеціальну команду `draw`, мова про яку йтиме у § 14.2.

Для створення точки P, заданої координатами (Px, Py) на площині, призначена команда `point`, яка має два формати: `point(`P`, [`Px`, `Py`])` і `point(`P`, `Px`, `Py`)`. Наприклад:

```
> with(geometry):
> point(p1,-1,0);
```

p1

Команди `line(`l`, [`A`, `B`], `dv`)` і `line(`l`, `eqn`, `dv`)` визначають пряму з ім'ям l, яка проходить через точки A і B (створені командами `point` чи іншими) або задана рівнянням *eqn*. В останньому випадку в якості параметра *dv* бажано використовувати двоелементний список імен незалежних змінних, які застосовуються у рівнянні, інакше у класичному робочому листі відображатимуться підказки для введення імен осей координат (після кожної з них треба ввести ідентифікатор змінної для відповідної осі координат, що завершується крапкою з комою, і натиснути клавішу `Enter`). При роботі зі стандартним робочим листом імена змінних для осей координат вказуються у спеціальних вікнах. Неправильні відповіді на поставлені програмою запитання можуть призвести до некоректної

роботи програми. Приклади:

```
> point(p1,[2,4]):point(p2,[-3.1,4]):line(L1,[p1,p2]):
> line(L2,x+3*y=8,[x,y]):
> line(L3,2*x-4*y=5):
enter name of the horizontal axis > x;
enter name of the vertical axis > y;
```

Якщо параметр *dv* доводиться використовувати часто, то можна замість нього присвоїти певні значення іменам системних змінних `_EnvHorizontalName` і `_EnvVerticalName`, наприклад:

```
> _EnvHorizontalName:=x: _EnvVerticalName:=y:
```

Список *dv* доводиться застосовувати також у деяких інших командах, коли використовується рівняння геометричного об'єкта. У цих випадках при наявності виконаних присвоювань іменам змінних `_EnvHorizontalName` і `_EnvVerticalName` список *dv* можна не вказувати.

Команди `segment(`*seg*, `[`*P1*, *P2*`])` і `segment(`*seg*, *P1*, *P2*`)` визначають відрізок *seg* з кінцями у точках *P1* і *P2*.

Команди `dsegment(`*seg*, `[`*P1*, *P2*`])` та `dsegment(`*seg*, *P1*, *P2*`)` задають вектор *seg* з початком у точці *P1* і кінцем у точці *P2*.

Коло створюють за допомогою команди `circle`, яка може використовуватись в одному з таких форматів:

1) `circle(`*c*, `[`*A*, *B*, *C*`]`, *dv*, *opt*`)` – коло *c* за трьома точками *A*, *B* і *C*, що лежать на ньому;

2) `circle(`*c*, `[`*A*, *B*`]`, *dv*, *opt*`)` – коло *c* за двома точками *A* і *B* на колі, які сприймаються як кінці діаметра;

3) `circle(`*c*, `[`*A*, *rad*`]`, *dv*, *opt*`)` – коло *c* радіуса *rad* з центром у точці *A*;

4) `circle(`*c*, *eqn*, *dv*, *opt*`)` – коло *c* за рівнянням кола *eqn*.

У команді `circle` в якості *dv* можна використовувати список незалежних змінних для рівняння кола, а *opt* – необов'язкова опція `centername=`*O*, яка задає точку *O* для центра кола. Приклади:

```
> circle(c1,[point(p1,[-2,0]),point(p2,[3,1]),
  point(p3,[3,4])],centername=O1):
> circle(c2,x^2+y^2-4*x+3*y+5,[x,y]):
```

Еліпс, гіперболу і параболу можна створити за допомогою команд `ellipse`, `hyperbola` і `parabola` відповідно, причому ці криві другого порядку можна задати, зокрема, за рівнянням чи п'ятьма точками, що належать кривій. Наприклад, команда `ellipse(`p, `[`A, B, C, D, F`]`, dv`)` визначає еліпс p за п'ятьма точками A, B, C, D, F, що йому належать, а команда `hyperbola(`p, eqn, dv`)` визначає гіперболу p за її рівнянням eqn.

Еліпс, гіперболу і параболу можна задавати також іншими способами, про які можна прочитати в довідковій системі Maple. Наприклад, еліпс можна визначити двома фокусами і довжиною більшої чи меншої осі, двома фокусами і сумою відстаней від будь-якої точки еліпса до фокусів, кінцями більшої і меншої осей, директрисою, фокусом і ексцентриситетом.

Довільну криву другого порядку p можна створити за допомогою команди `conic(`p, `[`A, B, C, D, F`]`, dv`)` за п'ятьма точками або `conic(`p, eqn, dv`)` – за рівнянням eqn. Нагадаємо, що крива p (якщо вона існує) може виявитись еліпсом, гіперболою, параболою, колом, однією прямою, двома паралельними чи перетинними прямими або навіть точкою.

Створити квадрат P можна такими способами:

1) `square(`P, `[`A, B, C, D`])` – за чотирма вершинами A, B, C, D;

2) `MakeSquare(`P, `[`$p1, p2$, `diagonal])` – за двома протилежними вершинами $p1$ і $p2$;

3) `MakeSquare(`P, `[`$p1$, `center=`c`])` – за однією з вершин $p1$ і центром c.

4) `MakeSquare(`P, `[`$p1, p2$, `adjacent])` – список P двох квадратів, для яких точки $p1$ і $p2$ є суміжними вершинами;

Трикутник T також можна визначити по-різному:

1) `triangle(`T, `[`A, B, C`])` – за вершинами A, B і C;

2) `triangle(`T, `[`$l1, l2, l3$`])` – за трьома прямими $l1$, $l2$ і $l3$.

Команда `RegularPolygon(`P, n, A, rad`)` визначає правильний n-кутник з центром у точці A і радіусом описаного кола *rad*. Одна з вершин буде розташована справа від центру многокутника на одній з ним горизонталі.

Команда `RegularStarPolygon(`P, n, A, rad`)` робить те саме, що й команда `RegularPolygon`, якщо число n є цілим. Якщо n – дріб, то його чисельник визначає кількість вершин, а знаменник – через скільки вершин з'єднувати. Наприклад:

```
> RegularStarPolygon(P,5/2,point(K,[0,0]),3):
```

Якщо вже існує змінна з ім'ям, що збігається з ім'ям створюваного геометричного об'єкта, то програма видасть повідомлення про помилку. В такому разі це ім'я треба задавати в команді створення геометричного об'єкта в одинарних лапках, наприклад:

```
> K:=125: point('K',[2,4]);
```

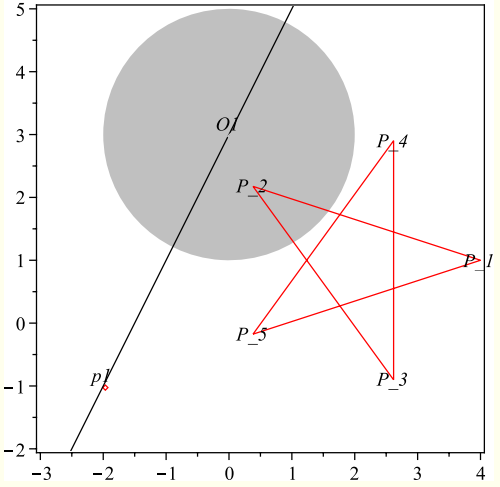
$$K$$

§ 14.2. Візуалізація графічних об'єктів

Для візуалізації (побудови на площині) геометричних об'єктів пакета `geometry` призначена команда `draw(`*obj*, *opts*`)`, першим параметром *obj* якої задають об'єкт, множину чи список об'єктів для відображення, а *opts* – опції, дія яких розповсюджується на всі відображувані об'єкти. В якості *opts* можна застосовувати майже всі опції команди `plot` (§ 4.2), а також опції `printtext` і `filled`. Опція `printtext=true` забезпечує відображення імен точок, а опція `filled=true` зафарбовує внутрішність замкненої фігури *obj* кольором, визначеним опцією `color`. Для кожного об'єкта списку (множини) можна після його імені у круглих дужках визначити опції, які застосовуватимуться саме до нього. Приклад:

```
> circle(c1,x^2+(y-3)^2=4,[x,y],centername=O1):
```

```
> point(p1,[-2,-1]): line(l1,[O1,p1],[x,y]):
> RegularStarPolygon(P,5/2,point(K,[2,1]),2):
> draw([c1(filled=true,color=gray),l1(color=black),
  p1,P],printtext=true,view=[-3..4,-2..5]);
```



§ 14.3. Визначення характеристик і взаємного розташування геометричних об'єктів

Для визначення характеристик геометричних об'єктів існують спеціальні команди. Дві з них працюють з об'єктами всіх типів:

`form(`P`)` визначає тип геометричного об'єкта P, наприклад, `point2d`, `parabola2d`, `square2d` тощо;

`detail(`P`)` виводить детальну інформацію про геометричний об'єкт P. Наприклад, для гіперболи буде виведено тип об'єкта (`hyperbola2d`), координати центра, фокусів, вершин, рівняння самої гіперболи та її асимптот.

Наведемо деякі інші команди, які використовуються для геометричних об'єктів окремих типів:

`area(`P`)` – площа круга, еліпса, трикутника, квадрата чи іншого правильного многокутника P;

`coordinates(`P`)` – список координат точки P;

`HorizontalCoord(`P`)` – абсциса точки P;

`VerticalCoord(`P`)` – ордината точки P;

`diagonal(`P`)` – довжина діагоналі квадрата P;

`radius(`P`)` – радіус кола P або описаного кола навколо правильного многокутника P;

`MajorAxis(`P`)` – довжина великої осі еліпса P;

`MinorAxis(`P`)` – довжина малої осі еліпса P;

`perimeter(`P`)` – периметр правильного многокутника P;

`Equation(`P, dv`)` – рівняння кола, еліпса, параболи, гіперболи чи прямої P.

Для визначення відстані між двома точками u і v або точкою u і прямою v використовують команду `distance(`u, v`)`. Наприклад:

```
> point(A,[2,4]): line(l,3*x-2*y+5=0,[x,y]):
> distance(A,l);
```

$$\frac{3}{13}\sqrt{13}$$

Команда `FindAngle(`u, v`)` знаходить кут у радіанах між двома прямими u і v. Наприклад:

```
> line(l1,2*x-5*y+7=0,[x,y]): line(l2,4*x+y=3,[x,y]):
> FindAngle(l1,l2);
```

$$\arctan\left(\frac{22}{3}\right)$$

Команди `slope(`l`)` і `slope(`A, B`)` обчислюють кутовий коефіцієнт прямої l або відрізка AB, визначеного точками A і B.

Наступні команди перевіряють умови взаємного розташування геометричних об'єктів або певні їх характеристики і виводять значення `true` (істина) або `false` (хибність):

`AreCollinear(`*p1*, *p2*, *p3*`)` перевіряє приналежність точок *p1*, *p2*, *p3* одній прямій;

`AreConcurrent`$(l1, l2, l3)$ перевіряє, чи перетинаються прямі $l1$, $l2$, $l3$ в одній точці;

`AreConcyclic`$(p1, p2, p3, p4)$ перевіряє, чи існує коло, якому належать задані точки $p1$, $p2$, $p3$, $p4$;

`AreParallel`$(l1, l2)$ перевіряє паралельність прямих $l1$, $l2$;

`ArePerpendicular`$(l1, l2)$ перевіряє перпендикулярність прямих $l1$, $l2$;

`AreSimilar`$(t1, t2)$ перевіряє подібність трикутників $t1$, $t2$;

`AreTangent`(l, c) перевіряє, чи дотикаються пряма l і коло c;

`IsEquilateral`(T) перевіряє рівносторонність трикутника T;

`IsRightTriangle`(T) перевіряє, чи трикутник T є прямокутним;

`IsOnLine`(p, l) перевіряє, чи лежить точка p на прямій l;

`IsOnCircle`(p, c) перевіряє, чи лежить точка p на колі c.

§ 14.4. Створення нових геометричних об'єктів за допомогою існуючих

Окремі базові випадки створення нових геометричних об'єктів за допомогою вже існуючих точок і прямих розглядалися у § 14.1. Тепер розглянемо складніші команди, пов'язані зі створенням різноманітних геометричних об'єктів для розв'язування задач аналітичної геометрії:

`intersection`(p, u, v) – точка (список точок) p перетину прямих чи кіл u і v;

`projection`$(p1, p, l)$ – точка $p1$ – проекція точки p на пряму l;

`midpoint`(C, A, B) чи `midpoint`(C, seg) – точка C – середина відрізка AB чи seg;

`OnSegment`(C, A, B, k) – точка C, яка ділить відрізок AB у співвідношенні $k : 1$ від точки A до точки B;

`randpoint`(p, u, r) – випадкова точка p на прямій чи колі u з абсцисою з діапазону r (у форматі `randpoint`$(p, r1, r2)$

команда створює випадкову точку p з абсцисою з діапазону *r1* і ординатою з діапазону *r2*);

`ParallelLine`(pl, p, l) – пряма pl, що проходить через точку p паралельно до прямої l;

`PerpendicularLine`(pl, p, l) – пряма pl, що проходить через точку p перпендикулярно до прямої l;

`PerpenBisector`(pl, A, B) – пряма pl, що проходить через середину відрізка AB перпендикулярно до нього;

`altitude`(hA, A, ABC) – пряма hA, що проходить через точку A перпендикулярно до сторони BC трикутника ABC (з додатковим четвертим параметром H команда створює точку H і висоту hA трикутника ABC);

`bisector`(bA, A, ABC) – пряма bA, що ділить кут A трикутника ABC навпіл (з додатковим четвертим параметром H команда створює точку H і бісектрису bA трикутника ABC);

`median`(mA, A, ABC) – пряма mA, що проходить через точку A і ділить сторону BC трикутника ABC навпіл (з додатковим четвертим параметром H команда створює точку H і медіану mA трикутника ABC);

`centroid`(G, T) – точка G перетину медіан (центр мас) у трикутнику T;

`orthocenter`(G, T) – точка G перетину висот (ортоцентр) у трикутнику T;

`incircle`(c, T, opt) – вписане в трикутник T коло c (опцією opt так само, як і в команді `circle`, можна задавати центр кола);

`circumcircle`(c, T, opt) – описане навколо трикутника T коло c;

`TangentLine`(ll, p, c) – список ll двох дотичних до кола c, які проходять через точку p;

`tangentpc`(l, p, c) – дотична l до кола c, яка проходять через точку p на колі;

`center`(cn, c) – точка cn – центр кола, еліпса чи гіперболи c (центр гіперболи чи еліпса – це середина відстані між фокуса-

ми, за відсутності параметра *cn* створюється точка `center_`c);

`asymptotes(`f`)` – список асимптот гіперболи f;

`foci(`f`)` – список фокусів еліпса чи гіперболи f;

`focus(`f`)` – фокус параболи;

`vertex(`f`)` – вершина параболи;

`vertices(`f`)` – список вершин еліпса чи гіперболи f;

`DefinedAs(`P`)` – список точок, які є кінцями відрізка, вершинами трикутника, квадрата чи іншого правильного многокутника P;

`reflection(`Q, P, c`)` – об'єкт Q, симетричний до об'єкта P відносно точки чи прямої c;

`translation(`Q, P, AB`)` – об'єкт Q, утворений паралельним перенесенням об'єкта P на вектор AB;

`rotation(`$Q, P,$ *angle*, *dir*, c`)` – об'єкт Q, отриманий з об'єкта P обертанням його на кут *angle* (в радіанах) у напрямі *dir* відносно центру обертання – точки c. Напрямом *dir* може бути `clockwise` (за годинниковою стрілкою) або `counterclockwise` (проти годинникової стрілки). Якщо точка c не задана, то нею вважається початок координат.

Задача 1. Задано точки $A(2,4)$, $B(-5,1/2)$, $C(3,2)$. Знайти площу трикутника ABC, рівняння медіани AM трикутника ABC, радіус вписаного кола і координати точки N перетину прямої KN, що проходить через центр K вписаного кола паралельно до прямої BC, з прямою AB. Побудувати на площині трикутник ABC, медіану AM, вписане коло, пряму KN, точку N (з позначенням усіх точок).

```
> point(A,[2,4]): point(B,[-5,1/2]): point(C,[3,2]):
> triangle(ABC,[A,B,C]): area(ABC);
```

$$\frac{35}{4}$$

```
> median(AM,A,ABC): Equation(AM,[x,y]);
```

$$\frac{13}{2} + \frac{11}{4}\,x - 3\,y = 0$$

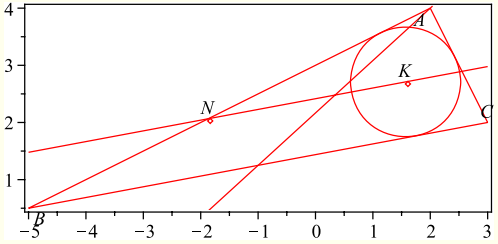
```
> incircle(c,ABC,centername=K): radius(c);
```

$$\frac{2}{35}\left(21+\frac{7}{2}\,\frac{2\sqrt{53}+11}{\sqrt{53}+9}-\frac{7\left(15+4\sqrt{53}\right)}{\sqrt{53}+9}\right)\sqrt{5}$$

```
> line(BC,[B,C]): line(AB,[A,B]):
> ParallelLine(KN,K,BC): intersection(N,KN,AB):
> coordinates(N);
```

$$\left[\frac{2\sqrt{53}-45}{\sqrt{53}+9},\ \frac{1}{2}\,\frac{8\sqrt{53}+9}{\sqrt{53}+9}\right]$$

```
> draw([ABC,AM,c,KN],printtext=true);
```



Задача 2. Скласти рівняння прямої, рівновіддаленої від двох паралельних прямих $x-y+2=0$, $2x-2y+3=0$.

```
> _EnvHorizontalName:=x: _EnvVerticalName:=y:
> line(L1,x-y+2=0): line(L2,2*x-2*y+3=0):
> point(A,[0,2]): # Довільна точка прямої L1
> projection(B,A,L2):
> PerpenBisector(L3,A,B): Equation(L3);
```

$$\frac{7}{16}+\frac{1}{4}\,x-\frac{1}{4}\,y=0$$

Задача 3. Криву $x^2+3xy+5y^2+6x-3y+12=0$ повернути на кут 30° за годинниковою стрілкою навколо точки $P(2,3)$. Визначити тип і рівняння отриманої кривої.

```
> conic(Q,x^2+3*x*y+5*y^2+6*x-3*y+12=0,[x,y]):
> point(P,[2,3]): rotation(QQ,Q,Pi/6,clockwise,P):
> form(QQ);
```

ellipse2d

```
> Equation(QQ);
```

$$\left(\frac{3}{4}\sqrt{3}+2\right)x^2+\left(2\sqrt{3}+\frac{3}{2}\right)xy+\left(-\frac{3}{4}\sqrt{3}+4\right)y^2+$$
$$+\left(4+\frac{1}{2}\sqrt{3}\right)x+\left(17\sqrt{3}-\frac{73}{2}\right)y+\frac{261}{2}-\frac{241}{4}\sqrt{3}=0$$

Питання до розділу 14

1. Який пакет використовується у Maple для розв'язування задач аналітичної геометрії на площині?

2. Як створити точку, пряму, відрізок, вектор, коло, еліпс, гіперболу, параболу, довільну криву другого порядку, трикутник, квадрат чи інший правильний многокутник?

3. Які команди призначені для визначення геометричних характеристик об'єктів? Як знайти координати точки, рівняння прямої чи кривої другого порядку?

4. Як відобразити створені геометричні об'єкти?

5. Як знайти відстань між точками, кут між прямими, визначити, чи лежать три точки на одній прямій?

6. Як знайти точки перетину прямих чи кіл, проекцію точки на пряму, створити паралельну, перпендикулярну пряму, поділити відрізок точкою на частини у заданому співвідношенні, створити медіану, висоту, бісектрису у трикутнику, дотичну до кола, вписати у трикутник коло, описати навколо трикутника коло?

7. Як повернути геометричний об'єкт на заданий кут, виконати паралельне перенесення, відображення симетрії геометричного об'єкта?

Вправи до розділу 14

1. Дано три точки: $A(2,1)$, $B(6,-3)$, $C(2,-1)$. Переконавшись, що ці точки не лежать на одній прямій, знайдіть площу і периметр

трикутника ABC, величину кута A, складіть рівняння сторони AB, висоти, медіани і бісектриси внутрішнього кута трикутника при вершині C.

2. Складіть рівняння бісектриси гострого кута, утвореного двома прямими $3x + 4y - 5 = 0$, $5x - 12y + 3 = 0$.

Примітка. Виберіть дві довільні точки на прямих, які разом з точкою їх перетину утворюватимуть трикутник з гострим кутом у цій точці.

3. Задано рівняння $3x - 2y + 1 = 0$, $x - y + 1 = 0$ двох сторін трикутника і рівняння $2x - y - 1 = 0$ медіани, що виходить із вершини, яка не належить першій стороні. Складіть рівняння третьої сторони трикутника.

4. Виконайте відображення симетрії кривої другого порядку $x^2 - 3x + 12y = 20$ відносно прямої $5x - 4y = 7$. Визначте рівняння і тип отриманої кривої.

Розділ 15. Застосування Maple до розв'язування задач аналітичної геометрії у просторі

§ 15.1. Створення геометричних об'єктів

Для розв'язування задач аналітичної геометрії та виконання побудов геометричних об'єктів у просторі використовують пакет `geom3d`.

Робота з пакетом `geom3d` є аналогічною до роботи з пакетом `geometry` (розділ 14). В усіх командах створення геометричного об'єкта першим параметром є ім'я цього об'єкта. Одночасна робота з пакетами `geometry` і `geom3d` неможлива.

Для створення точки і прямої у просторі використовують команди `point` і `line` (§ 14.1) з деякими змінами у порівнянні з пакетом `geometry`. Зокрема, для точки, очевидно, треба вказувати три координати. Про відмінності, які стосуються команди `line`, йтиметься нижче у цьому параграфі. Можна створити також відрізок `segment`, вектор `dsegment`, трикутник `triangle` (§ 14.1).

Для створення площини *pl*, яка проходить через задані геометричні об'єкти, використовують команду `plane` в одному з таких форматів:

1) `plane(`*pl*, [*A*, *dseg1*]`,` *dv*`)` – через точку *A* і вектор *dseg1*;

2) `plane(`*pl*, [*l1*, *l2*]`,` *dv*`)` – через прямі *l1* і *l2* (якщо вони мимобіжні, то площина проходитиме через пряму *l1* паралельно до прямої *l2*);

3) `plane(`*pl*, [*dseg1*, *dseg2*]`,` *dv*`)` – через вектори *dseg1* і *dseg2*;

4) `plane(`*pl*, [*A*, *B*, *C*]`,` *dv*`)` – через три точки *A*, *B* і *C*, що не лежать на одній прямій;

5) `plane(`*pl*, *eqn*, *dv*`)` – задається рівнянням *eqn*.

Необов'язковий трьохелементний список *dv* у команді `plane` задає імена незалежних змінних для використання їх у рівнянні площини, наприклад `[x,y,z]`. Якщо список *dv* не вказати, то у класичному робочому листі відображатимуться

підказки для введення імен осей координат, а у стандартному робочому листі імена змінних для осей координат треба буде вказати у спеціальних вікнах. Якщо параметр *dv* потрібно використовувати часто, то можна замість нього присвоїти певні значення іменам глобальних змінних `_EnvXName`, `_EnvYName` і `_EnvZName`, наприклад:

```
> _EnvXName:=x: _EnvYName:=y: _EnvZName:=z:
```

Пряму у просторі можна задати не лише двома точками, але й точкою і вектором, двома непаралельними площинами, точкою і площиною (в цьому випадку пряма проходитиме через задану точку перпендикулярно до площини), а також рівнянням прямої в параметричній формі: `line(`*l*, `[`*a1*`+`*b1*`*`*t*, *a2*`+`*b2*`*`*t*, *a3*`+`*b3*`*`*t*`]`, *t*`)`. В останньому випадку бажано останнім аргументом *t* вказувати ім'я параметра, інакше програма це ім'я потім все одно запитає. У будь-якому випадку другим аргументом команди `line` повинен бути список, який містить дві точки, точку і вектор, дві площини, точку і площину або три вирази, залежні від параметра, а третім аргументом може бути ім'я параметра. Приклади:

```
> with(geom3d):
> plane(pl,2*x+4*y-z=15,[x,y,z]):
> line(l1,[point(A,[2,5,-3]),pl]):
> line(l2,[3*t-1,2*4+8,5-t],t):
```

Якщо часто доводиться використовувати параметр *t*, то можна замість нього присвоїти певне значення імені глобальної змінної `_EnvTName`, наприклад:

```
> _EnvTName:=t:
```

Сферу *S* створюють за допомогою команди `sphere`, яка може використовуватись у таких форматах:

1) `sphere(`*S*, `[`*A*, *B*, *C*, *D*`]`, *dv*, *opt*`)` – за чотирма точками *A*, *B*, *C* і *D* сфери;

2) `sphere(`*S*, `[`*A*, *B*`]`, *dv*, *opt*`)` – за двома точками *A* і *B* – кінцями діаметра сфери;

3) `sphere`(S, [A, *rad*], *dv*, *opt*) – за радіусом *rad* і центром A сфери;

4) `sphere`(S, *eqn*, *dv*, *opt*) – за рівнянням *eqn* сфери.

У команді `sphere` в якості *dv* можна використовувати список незалежних змінних для рівняння сфери, а *opt* – опція `centername=`*O1*, яка задає точку *O1* для центра сфери.

Команда `gtetrahedron`(T, [A, B, C, D]) визначає піраміду T за її чотирма вершинами або площинами A, B, C і D.

Команда `parallelepiped`(*pp*, [*d1*, *d2*, *d3*]) створює паралелепіпед *pp* за трьома векторами *d1*, *d2* і *d3* із спільним початком.

Крім того, є багато команд для створення правильних многогранників, наприклад `cube`(P, A, r) задає куб P з центром у точці A і радіусом описаної навколо нього сфери r. З іншими командами для створення правильних многогранників можна ознайомитись у довідці Maple.

Для відображення геометричних об'єктів у просторі використовується команда `draw` (§ 14.2).

§ 15.2. Визначення характеристик і взаємного розташування геометричних об'єктів

Для визначення характеристик геометричних об'єктів у просторі використовуються ті самі команди, що й для об'єктів на площині: `form`, `detail`, `area`, `coordinates`, `radius`, `Equation` та інші (§ 14.3). Команда `area` знаходить площу трикутника або площу поверхні сфери чи правильного многогранника P. Команда `volume` обчислює об'єм кулі або многогранника.

Для визначення відстані між двома точками, двома прямими, двома площинами, точкою і прямою, точкою і площиною, прямою і площиною u і v використовують команду `distance`(u, v).

Команда `FindAngle`(u, v) знаходить кут у радіанах між прямими, площинами, сферами (між дотичними площинами

до сфер у точках їх перетину) або між прямою і площиною u і v.

Є низка команд (частина з яких використовується й у випадку площини – § 14.3), що перевіряють умови взаємного розташування геометричних об'єктів або певні характеристики цих об'єктів і виводять значення `true` (істина) або `false` (хибність): `AreCollinear`, `AreConcurrent`, `AreParallel`, `ArePerpendicular`, `AreCoplanar` (перевіряє, чи належать чотири точки або дві прямі одній площині), `IsEquilateral`, `IsRightTriangle`, `IsOnObject`(p, obj) (перевіряє, чи належить точка p геометричному об'єкту obj), `IsTangent`(pl, sph) (перевіряє, чи площина pl є дотичною до сфери sph).

§ 15.3. Створення нових геометричних об'єктів за допомогою існуючих

Окремі базові випадки створення нових геометричних об'єктів у просторі за допомогою вже існуючих точок, прямих і площин розглядались у § 15.1. Частину важливих для розв'язування задач аналітичної геометрії команд § 14.4 використовують і в просторовому випадку. Однак деякі з них мають певні відмінності.

Зокрема, команда `intersection`(p, u, v) знаходить перетин p (точку або пряму) двох прямих, двох площин, прямої та площини або прямої та сфери u і v. Ця команда у форматі `intersection`(p, u, v, w) знаходить точку p перетину трьох площин u, v і w (якщо така точка існує).

Команда `projection`(Q, P, l) створює проекцію Q точки P на пряму чи площину l або проекцію Q відрізка чи прямої P на площину l.

Команда `reflection`(Q, P, c) створює об'єкт Q, симетричний до об'єкта P відносно точки, прямої чи площини c.

Команда `rotation`$(Q, P, angle, l)$ створює об'єкт Q, отриманий обертанням об'єкта P на кут $angle$ (в радіанах) відносно осі обертання – прямої l.

Команда `parallel(`Q, u, v`)` створює пряму або площину Q, що проходить через задану точку або пряму u паралельно до прямої чи площини v.

Команда `TangentPlane(`Q, P, S`)` створює дотичну до сфери S площину Q, яка проходить через точку P сфери.

Задача. Піраміда $ABCK$ задана координатами вершин $A(2,4,-3)$, $B(-5,3,1)$, $C(3,2,3)$, $K(2,-4,2)$. Довести, що всі чотири точки не лежать в одній площині, а точки A, B і C не лежать на одній прямій. Знайти рівняння площини, що проходить через грань ABC, величину кута між ребром KB і гранню ABC та її десяткове наближення у градусах, рівняння та довжину висоти, проведеної з вершини K на грань ABC, площу грані ABC та об'єм піраміди. Побудувати піраміду $ABCK$.

```
> point(A,[2,4,-3]): point(B,[-5,3,1]):
> point(C,[3,2,3]): point(K,[2,-4,2]):
> AreCoplanar(A,B,C,K);
```

$$\mathit{false}$$

```
> AreCollinear(A,B,C);
```

$$\mathit{false}$$

```
> gtetrahedron(ABCK,[A,B,C,K]):
> plane(pl_ABC,[A,B,C]): Equation(pl_ABC,[x,y,z]);
```

$$-143 + 2\,x + 46\,y + 15\,z = 0$$

```
> line(l_KB,[K,B]): FindAngle(l_KB,pl_ABC);
```

$$\arcsin\left(\frac{293}{77385}\sqrt{25795}\right)$$

```
> evalf(%*180/Pi);
```

$$37.45261046$$

```
> projection(M,K,pl_ABC): line(l_KM,[K,M]):
> Equation(l_KM,t);
```

$$\left[2+\frac{586}{2345}\,t,\ -4+\frac{13478}{2345}\,t,\ 2+\frac{879}{469}\,t\right]$$

```
> distance(K,M);
```

$$\frac{293}{2345}\sqrt{2345}$$

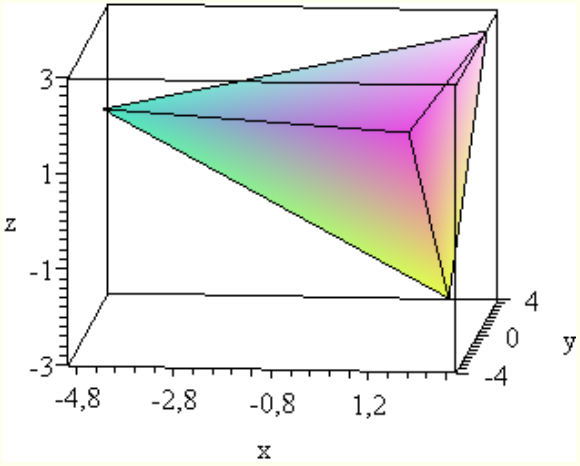
```
> triangle(ABC,[A,B,C]): area(ABC);
```

$$\frac{1}{2}\sqrt{2345}$$

```
> volume(ABCK);
```

$$\frac{293}{6}$$

```
> draw(ABCK,axes=boxed,labels=[x,y,z]);
```



Питання до розділу 15

1. Який пакет використовується у Maple для розв'язування задач аналітичної геометрії у просторі?

2. Як створити точку, пряму, площину, відрізок, вектор, сферу, паралелепіпед, піраміду?

3. Які команди призначені для визначення геометричних характеристик об'єктів? Як знайти координати точки, рівняння прямої, площини, сфери?

4. Як відобразити створені геометричні об'єкти?

5. Як знайти відстань між точками, кут між прямими чи площинами, визначити, чи лежать три точки на одній прямій?

6. Як знайти перетин прямих чи площин, проекцію точки чи прямої на площину, створити паралельну пряму чи площину, дотичну площину до сфери?

7. Як повернути геометричний об'єкт на заданий кут, виконати паралельне перенесення, відображення симетрії геометричного об'єкта?

Вправи до розділу 15

1. Піраміда $ABCM$ задана координатами своїх вершин: $A(4, 2, -3)$, $B(0, 3, 1)$, $C(5, -1, 2)$, $M(2, 6, 0)$. Довести, що всі вершини не лежать в одній площині. Знайти площу і периметр перерізу піраміди $ABCM$ площиною $3x + 4y - 6z = 5$. Зробити рисунок.

Примітка. Ребра піраміди, які перетинає площина, найлегше визначити з рисунка.

Розділ 16. Теорія ймовірностей і математична статистика (пакет stats)

§ 16.1. Генерація випадкових об'єктів

Команда `rand(`n`)` або `rand(`a`..`b`)` зі стандартної бібліотеки створює процедуру для генерації рівномірно розподілених випадкових цілих чисел з проміжку $[0, n-1]$ або $[a, b]$. Команда `rand()` без аргументів генерує випадкове дванадцятицифрове ціле невід'ємне число. Приклади:

```
> rr:=rand(45..95);
```

$$rr := \text{proc}(\,)\ \text{proc}(\,)\ \text{option}\ \mathit{builtin} = \mathit{RandNumberInterface};\ \text{end proc}(6, 51, 6) + 45\ \text{end proc}$$

```
> rr();
```

$$57$$

```
> rr();
```

$$72$$

```
> ra:=evalf(rand(10000000000000)/10000000000000):
  ra();
```

$$0.2338436154$$

Останній приклад показує, як можна отримати дійсне число, рівномірно розподілене на проміжку $[0, 1)$.

Про генерацію випадкових чисел з різними законами розподілу йтиметься в § 16.6.

Команда

`randpoly(`*var*, *opts*`)`

генерує многочлен змінної *var* або списку чи множини змінних *var* з випадковими коефіцієнтами. Опції *opts* дозволяють задати деякі властивості випадкового многочлена. Наведемо їх у таблиці.

Опція	Опис	Значення за замовчуванням
`coeffs`	Метод вибору коефіцієнтів	`rand(-99..99)`
`expons`	Метод вибору показників степеня	`rand(6)`
`terms`	Кількість членів многочлена, які будуть генеруватись	6
`degree`	Загальний степінь многочлена	5

В якості опції *opts* у команді `randpoly` можна також використовувати термін `homogeneous`, який означає, що многочлен кількох змінних буде однорідним (сума показників степенів множників у всіх членах многочлена буде однаковою). Приклади:

```
> randpoly(x);
```

$$-56 - 5\,x^5 + 36\,x^4 - 8\,x^3 + 30\,x^2 - 3\,x$$

```
> randpoly(x);
```

$$17 + 72\,x^5 - 40\,x^4 - 68\,x^3 - 15\,x^2 - 32\,x$$

```
> randpoly(x,coeffs=rand(-1000..1000),expons=rand(8));
```

$$-1321 + 446\,x^7 - 550\,x^4 - 1000\,x^3 + 677\,x$$

```
> randpoly([x,y],degree=10,terms=7);
```

$$-53\,x\,y^2 + 37\,y^3 + 76\,x^4\,y^3 - 44\,x^8\,y + 67\,x^7\,y^3 - 72\,x^6\,y^4 + 2\,y^{10}$$

```
> randpoly([x,y,z],homogeneous,degree=7);
```

$$24\,x^4\,y\,z^2 - 18\,x\,y^3\,z^3 - 8\,y^4\,z^3 + 37\,y^5\,z^2 - 96\,x^5\,y\,z - 82\,x^3\,y^2\,z^2$$

Команда `randmatrix(`$n, m, opts$`)` з пакета `linalg` генерує випадкову матрицю розмірності $n \times m$. Як опції *opts* можуть використовуватись рівність `entries=rand(`$a..b$`)` і один

з термінів: `sparse` (розріджена матриця), `symmetric` (симетрична матриця), `antisymmetric` (кососиметрична матриця), `unimodular` (цілочислова матриця з визначником, що дорівнює $+1$ або -1).

Команда `randvector(`n, opt`)` з пакета `linalg` генерує випадковий n-елементний вектор. В якості опції opt може використовуватись рівність вигляду `entries=rand(`a`..`b`)`.

§ 16.2. Статистичні списки і підпакети

Пакет `stats` підтримує різноманітні статистичні обчислення, графічно подає статистичні дані, генерує випадкові послідовності з заданими законами розподілу і надає доступ до функцій розподілу та інших функцій багатьох розподілів випадкових величин.

Пакет `stats` працює з даними, які організовані у статистичні списки, що включають у себе і звичайні списки типу `list`. Спеціальні статистичні списки можуть містити окремі елементи, діапазони, а також так звані зважені величини. Для створення зважених величин застосовується команда `Weight(`x, n`)` з основної бібліотеки, яка вказує, що величина чи діапазон x повинна з'явитись у списку n разів (має вагу n). Приклад:

```
> statlist:=[0.51,Weight(1.05,3),Weight(0.45..0.47,4),
  0.63..0.66];
```

$$statlist := [0.51, Weight(1.05, 3), Weight(0.45..0.47, 4), 0.63..0.66]$$

Пакет `stats` включає такі підпакети: `describe` – для обчислення статистичних характеристик даних; `fit` – для задач регресійного аналізу (апроксимації даних заданими залежностями); `transform` – для перетворення даних; `random` – для генерування випадкових чисел з заданими законами розподілу; `statevalf` – для доступу до функцій розподілу та інших функцій багатьох розподілів випадкових величин; `statplots` – для графічного подання статистичних даних.

Після підключення пакета `stats` командою `with(stats)` для доступу до його команд потрібно набрати:

підпакет`[`*команда*`](`*параметри*`)`.

Потрібний підпакет можна підключити також командою

`with(stats[`*підпакет*`])`.

§ 16.3. Підпакет `describe`

Цей підпакет призначений для обчислення різноманітних статистичних характеристик. Формат виклику команд залежить від того, чи підключені пакет `stats` і підпакет `describe`:

`stats[describe,` *command*`](`*data*`)`

або

`describe[`*command*`](`*data*`)`

або

command`(`*data*`)`.

У наведених форматах *data* – статистичний список, а замість *command* може бути одна з команд підпакета, наприклад:

`mean` – середнє арифметичне;

`geometricmean` – середнє геометричне;

`harmonicmean` – середнє гармонічне;

`quadraticmean` – середнє квадратичне;

`median` – медіана;

`mode` – мода;

`count` – кількість елементів у статистичному списку;

`linearcorrelation` – лінійна кореляція двох статистичних списків *data*;

`percentile[`k`]` – k-та перцентиль;

`range` – діапазон зміни даних;

`variance[`Nc`]` – дисперсія;

`standarddeviation[`Nc`]` – середньоквадратичне відхилення;

`coefficientofvariation[`Nc`]` – коефіцієнт варіації;

`moment[`k, c, Nc`]` – момент k-го порядку в точці c (зокрема, для початкового моменту береться $c = 0$ – значення за замовчуванням, а для центрального моменту – $c =$ `mean`, в якості c можна використовувати також інші статистичні функції);

`skewness[`Nc`]` – коефіцієнт асиметрії.

У п'яти останніх командах замість Nc вказується 0 (значення за замовчуванням) для генеральної сукупності або 1 для вибірки з генеральної сукупності.

Деякі особливості використання цих функцій пояснимо на прикладах списків `data1` і `data2`:

```
> with(stats):
> data1:=[1,2,3,4,2,3,4,5,6,7,2,3,2]:
> data2:=[1.,3.,4.,4.,7.,8.,4.,5.,5.,6.,2.,3.,6.]:
> describe[count](data1);
```

$$13$$

```
> describe[coefficientofvariation](data1);
```

$$\frac{1}{44}\sqrt{482}$$

```
> describe[harmonicmean]([1,4,5,7]);
```

$$\frac{560}{223}$$

```
> describe[mean](data2);
```

$$4.461538462$$

```
> describe[median](data2);
```

$$4.$$

```
> describe[mode](data1);
```

$$2$$

```
> describe[linearcorrelation](data1,data2);
```

$$0.3510958229$$

```
> describe[percentile[55]](data1);
```

$$3$$

```
> describe[range](data1);
```

$$1..7$$

```
> describe[standarddeviation[1]](data1); evalf(%);
```

$$\frac{1}{78}\sqrt{18798}$$

$$1.757766649$$

```
> describe[variance](data2);
```

$$3.633136095$$

```
> describe[skewness[1]](data1);
```

$$\frac{3843}{755053}\sqrt{18798}$$

Обчислимо початковий момент шостого порядку і центральний момент другого порядку:

```
> describe[moment[6,0]]([1,2,3,4,5]);
```

$$4103$$

```
> describe[moment[2,mean]]([1,2,3,4,5]);
```

$$2$$

Можна створити функцію для обчислення декількох статистичних показників одночасно. Наприклад, створимо функцію G, яка визначить кількість елементів у списку даних,

середнє значення і дисперсію:

```
> G:=[describe[count], describe[mean],
  describe[variance[1]]]:
> G(data1);
```

$$\left[13, \frac{44}{13}, \frac{241}{78}\right]$$

```
> G(data2);
```

$$[13, 4.461538462, 3.935897436]$$

§ 16.4. Підпакет fit

Підпакет `fit` використовують для регресійного аналізу й апроксимації статистичних даних вибраними залежностями. Формат виклику основної команди підпакета:

`stats[fit,leastsquare[`*vars*, *eqn*, *pars*`]](`*ldata*`)`

або

`fit[leastsquare[`*vars*, *eqn*, *pars*`]](`*ldata*`)`,

де *ldata* – список статистичних списків, *vars* – список змінних, відповідних (по порядку) спискам з *ldata*, *eqn* – апроксимуюче лінійне відносно параметрів рівняння (за замовчуванням лінійне рівняння з останньою змінною з *vars* як залежною змінною), *pars* – множина параметрів, які будуть замінені конкретними значеннями. Команда `leastsquare` будує рівняння кривої для апроксимації даних методом найменших квадратів.

Розглянемо приклади апроксимації трьох списків, де змінними є x, y, z:

```
> A:=[1,2,3,5]: B:=[2,4,6,8]: C:=[3,5,7,10]:
> fit[leastsquare[[x,y,z]]]([A,B,C]);
```

$$z = 1 + x + \frac{1}{2}\,y$$

```
> fit[leastsquare[[x,y], y=a*x^2+b*x+c,{a,b,c}]]
  ([A,B]);
```

$$y = -\frac{5}{22}\,x^2 + \frac{317}{110}\,x - \frac{39}{55}$$

```
> fit[leastsquare[[x,y], y=a*x^2+b*x+c]]([A,B]);
```

$$y = -\frac{5}{22}\,x^2 + \frac{317}{110}\,x - \frac{39}{55}$$

```
> fit[leastsquare[[x,y], y=a*x^2+b*x+c,{a,b}]]([A,C]);
```

$$y = \left(-\frac{105}{569} + \frac{125}{1138}\,c\right)x^2 + \left(\frac{1659}{569} - \frac{837}{1138}\,c\right)x + c$$

§ 16.5. Підпакет `transform`

Цей підпакет використовують для виконання різноманітних перетворень статистичних даних. Наведемо найчастіше вживані команди підпакета `transform`:

`apply[`*fc*`](`*data*`)` застосовує функцію *fc* до списку *data*;

`classmark(`*data*`)` замінює інтервали значень їх середніми точками;

`cumulativefrequency(`*data*`)` знаходить накопичену частотність даних;

`frequency(`*data*`)` знаходить частотність кожного елемента даних;

`moving[`*k*`](`*data*`)` обчислює середні значення послідовних порцій з *k* елементів;

`multiapply[`*fc*`](`*ldata*`)` застосовує функцію кількох змінних *fc* до списку кількох статистичних списків *ldata*;

`scaleweight[`*k*`](`*data*`)` множить ваги даних на число *k*;

`split[`*n*`](`*data*`)` розбиває список *data* на *n* списків однакової ваги;

`statsort(`*data*`)` сортує дані за зростанням;

`statvalue(`*data*`)` замінює всі ваги одиницями;

`tally(`*data*`)` групує дані, які мають однакові значення;

`tallyinto(`*data*, *partition*`)` групує дані *data* відповідно до складеного з діапазонів списку *partition*, який визначає спосіб групування (включаються лише ліві кінці проміжків).

Розглянемо приклади:

```
> data1:=[1,1,1,2,3,3,3,4,5,6,6,6,6]:
> transform[tally](data1);
```

$$[Weight(1,3), 2, Weight(3,3), 4, 5, Weight(6,4)]$$

```
> h1:=transform[tallyinto](data1,[0..5,5..10,10..15]);
```

$$h1 := [Weight(0..5, 8), Weight(5..10, 5), Weight(10..15, 0)]$$

```
> transform[scaleweight[1/5]](h1);
```

$$\left[Weight\left(0..5, \frac{8}{5}\right), 5..10, Weight(10..15, 0)\right]$$

```
> transform[frequency](h1);
```

$$[8, 5, 0]$$

```
> transform[cumulativefrequency]([5,Weight(1..6,10),2,
  2]);
```

$$[1, 11, 12, 13]$$

```
> transform[moving[3]](data1);
```

$$\left[1, \frac{4}{3}, 2, \frac{8}{3}, 3, \frac{10}{3}, 4, 5, \frac{17}{3}, 6, 6\right]$$

```
> M1:=transform[apply[x->x^2]](data1);
```

$$M1 := [1, 1, 1, 4, 9, 9, 9, 16, 25, 36, 36, 36, 36]$$

```
> M2:=multiapply[(x,y,z)->x+y-2*z]([[1,2,3,4],
  [2,3,4,7],[4,6,9,1]]);
```

$$M2 := [-5, -7, -11, 9]$$

```
> transform[statsort](M2);
```

$$[-11, -7, -5, 9]$$

```
> transform[statvalue]([Weight(3,10),2,2,1,1,1]);
```

$$[3, 2, 2, 1, 1, 1]$$

```
> transform[split[4]]([2,3,4,5,-1,2,6,10]);
```

$$[[2, 3], [4, 5], [-1, 2], [6, 10]]$$

§ 16.6. Підпакет `random`

Підпакет `random` призначений для генерування випадкових послідовностей з наперед заданими законами розподілів. Формат виклику:

`stats[random,`*distribution*`](`*quantity*`)`

або

`random[`*distribution*`](`*quantity*`)`

або

distribution`(`*quantity*`)`,

де *distribution* – закон розподілу чисел, *quantity* – кількість випадкових чисел, які потрібно отримати.

Підпакет `random` містить, зокрема, такі дискретні розподіли:

`binomiald[`n, p`]` – біноміальний (кількість успіхів у серії n незалежних випробувань, якщо p – ймовірність успіху в одному випробуванні);

`discreteuniform[`a, b`]` – дискретний рівномірний на проміжку $[a, b]$;

`negativebinomial[`k, p`]` – від'ємний біноміальний (кількість невдалих випробувань до k-го успішного при ймовірності успіху p в одному незалежному випробуванні);

`poisson[`λ`]` – розподіл Пуассона;

`empirical[list]` – емпіричний.

Наведемо деякі неперервні розподіли з підпакета `random`:

`uniform[`a,b`]` – рівномірний на відрізку $[a,b]$;

`normald[`μ,σ`]` – нормальний розподіл з середнім значенням μ і середньоквадратичним відхиленням σ;

`beta[`α_1,α_2`]` – бета-розподіл з параметрами α_1 і α_2;

`chisquare[`k`]` – χ^2 (хі-квадрат) з k ступенями свободи;

`exponential[`λ,a`]` – зміщений вправо на a експоненціальний розподіл з інтенсивністю λ;

`gamma[`α,β`]` – гамма-розподіл з параметрами α і β;

`lognormal[`μ,σ`]` – логарифмічний нормальний (логнормальний) розподіл з середнім значенням μ і середньоквадратичним відхиленням σ відповідного нормального розподілу.

Деякі можливості підпакета `random` детальніше розглянемо на прикладах.

1. Згенерувати чотири випадкові числа, які мають нормальний розподіл з середнім значенням 0 і середньоквадратичним відхиленням 1:

```
> r:=random[normald](4): map(evalf,[r])[];
```

$$0.2127564698, -0.7279468705, 0.6855773713, -0.3760280547$$

Для останніх версій Maple випадкові числа з нормальним законом розподілу виводяться у неспрощеному вигляді. Для їх спрощення використано команду `map` (§ 5.9).

2. Згенерувати вісім випадкових чисел, які мають біноміальний розподіл, якщо ймовірність успіху в кожному з дванадцяти незалежних випробувань становить 0,2:

```
> random[binomiald[12,0.2]](8);
```

$$4, 2, 1, 1, 3, 3, 3, 2$$

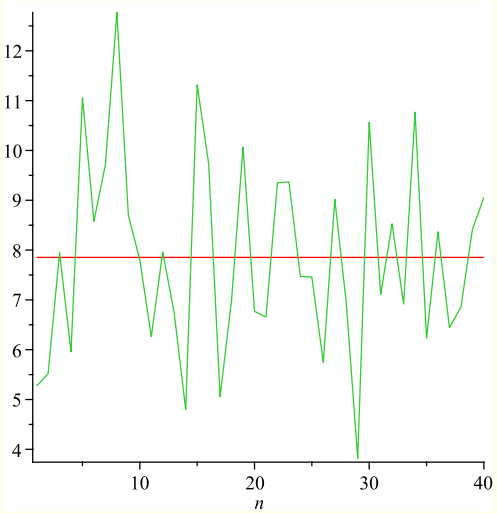
3. Створити масив з 40 елементів з такими параметрами нормального розподілу: середнє значення – 8, середнє квадратичне відхилення – 2,1. Оцінити середнє значення в отрима-

ному масиві, побудувати графік з послідовно з'єднаних елементів, а також лінію, яка відповідає середньому значенню:

```
> k:=40: data:=stats[random, normald[8,2.1]](k):
> describe[mean]([data]);
```

7.852942158

```
> plot({describe[mean]([data]),[[n,op(n,[data])]
  $n=1..k]}, n=1..k);
```



§ 16.7. Підпакет statevalf

Цей підпакет використовується для доступу до функцій розподілу ймовірностей, функцій щільності, функцій імовірностей і обернених функцій для різноманітних розподілів випадкових величин. Формат виклику:

`stats[statevalf,`*function*`,`*distribution*`](`*arg*`)`

або

`statevalf[`*function*`,`*distribution*`](`*arg*`)`,

де *function* – функція, *distribution* – закон розподілу (§ 16.6), *arg* – точка, в якій обчислюється значення функції.

В якості *function* використовують функції:

– для неперервних розподілів:

`cdf` – функція розподілу ймовірностей (інтегральна функція);

`icdf` – обернена функція до функції розподілу ймовірностей;

`pdf` – функція щільності розподілу ймовірностей (диференціальна функція);

– для дискретних розподілів:

`dcdf` – функція розподілу ймовірностей;

`idcdf` – обернена функція до функції розподілу ймовірностей;

`pf` – функція ймовірностей.

Наведемо приклади використання підпакета `statevalf`.

Значення ймовірності розподілу Пуассона для $\lambda = 3$ в точці 4:

```
> stats[statevalf,pf,poisson[3]](4);
```

$$0.1680313557$$

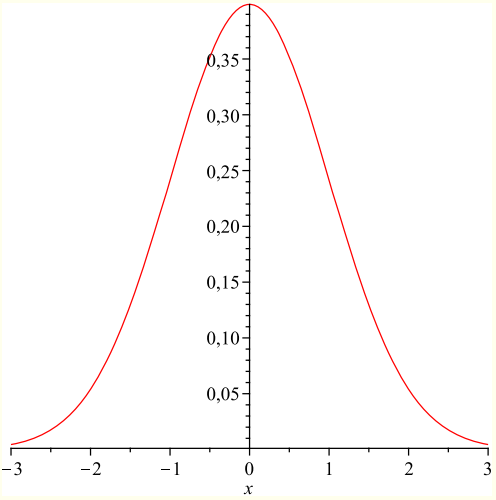
Значення функції щільності ймовірностей нормального розподілу з середнім значенням 0 і середньоквадратичним відхиленням 1 в точці 0,3:

```
> stats[statevalf,pdf,normald](0.3);
```

$$0.3813878155$$

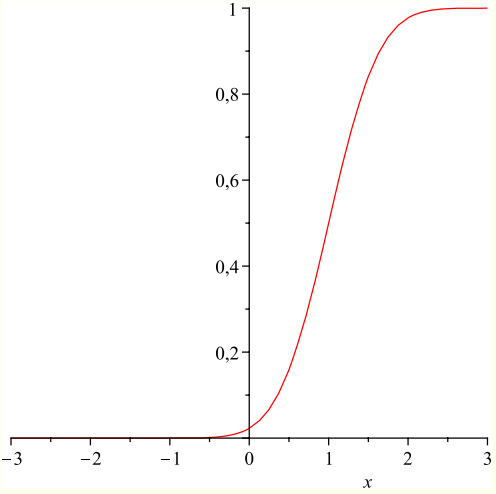
Графік функції щільності ймовірностей нормального розподілу з середнім значенням 0 і середньоквадратичним відхиленням 1 на відрізку $[-3, 3]$:

```
> plot(stats[statevalf,pdf,normald](x), x=-3..3);
```



Графік функції розподілу ймовірностей для нормального закону розподілу з середнім значенням 1 і середньоквадратичним відхиленням 0,5 на відрізку $[-3, 3]$:

```
> plot(stats[statevalf,cdf,normald[1,0.5]](x),
  x=-3..3);
```



§ 16.8. Підпакет statplots

Цей підпакет призначений для побудови графіків статистичних функцій.

Команда

`histogram`(*datas*, *opts*)

будує гістограму послідовності списків *datas*. Якщо є лише один список, то гістограма буде плоскою, якщо більше, – просторовою. Опції *opts* можуть бути опціями команди `plot` (§ 4.2). Крім них можна використовувати опції `area=`x і `numbars=`y. За замовчуванням всі прямокутники в гістограмі мають однакову площу, але неоднакову ширину. При використанні опції `area=`x всі прямокутники матимуть однакову ширину, а їх сумарна висота дорівнюватиме x. Опція `numbars=`y задає кількість прямокутників.

Команда

`scatterplot`(*datas*, *opts*)

будує точковий графік послідовності списків *datas*. Якщо є лише один список, то точки розміщуються вздовж осі x. Якщо є два чи три списки, то точки відображаються зі значеннями абсциси з першого списку, ординати – з другого, аплікати – з третього. Ваги для відповідних елементів у різних списках мають бути однаковими. Опції *opts* тут ті самі, що й для команди `plot` (§ 4.2).

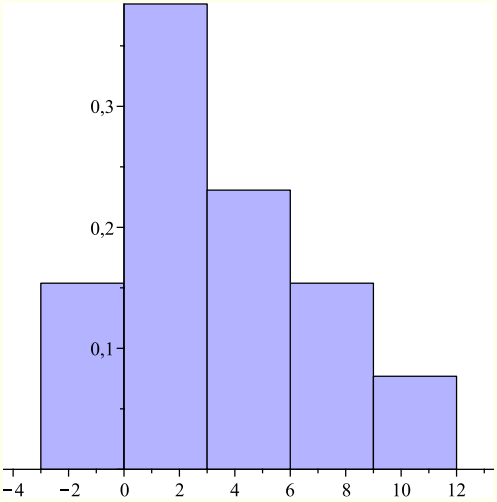
Команди `xshift`(x, *plot*), `yshift`(y, *plot*), `zshift`(z, *plot*) зсувають графік *plot* на x одиниць вздовж осі абсцис, на y – вздовж осі ординат, на z – вздовж осі аплікат відповідно.

Команди `xscale`(x, *plot*), `yscale`(y, *plot*), `zscale`(z, *plot*) масштабують графік *plot* в x разів вздовж осі абсцис, y – вздовж осі ординат, z – вздовж осі аплікат відповідно.

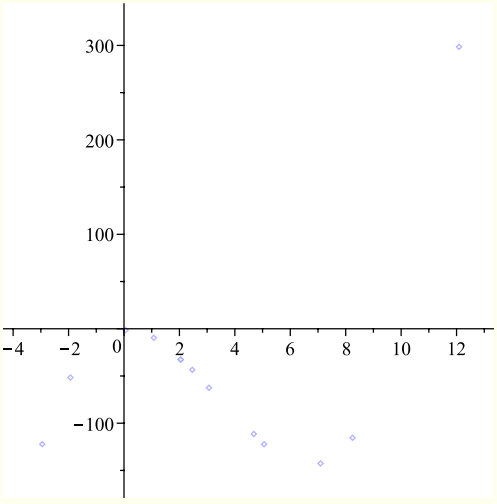
Для заміни координатних осей використовуються команди `xyexchange`(*plot*), `yzexchange`(*plot*), `xzexchange`(*plot*).

Приклади:

```
> data1:=[2,3,7,0,1,2,-2,5,2.4,12,-3,4.6,8.2]:
> statplots[histogram](data1,numbars=5,area=1);
```




```
> data2:=transform[apply[x->x^3-10*x^2+x]](data1):
> statplots[scatterplot](data1,data2);
```



Питання до розділу 16

1. Як згенерувати рівномірно розподілене випадкове число?

2. Що можуть містити статистичні списки?

3. Для чого призначений пакет `stats`? Назвіть підпакети цього пакета та їх основні можливості?

4. Який підпакет пакета `stats` призначений для обчислення статистичних характеристик? Наведіть приклади обчислення середнього арифметичного (геометричного, квадратичного, гармонічного), моди, медіани розподілу, дисперсії вибірки.

5. Для чого призначений підпакет `fit`? Як ним користуватись?

6. Який підпакет пакета `stats` використовують для перетворень статистичних даних? Назвіть декілька прикладів використання цього підпакета.

7. Для чого призначений підпакет `random`? Назвіть декілька прикладів генерування випадкових чисел з заданими законами розподілів (нормального, біноміального, гамма-розподілу тощо).

8. Як отримати доступ до функцій щільності розподілу ймовірностей, функцій розподілу ймовірностей і обернених функцій до функцій розподілу для різних законів розподілу випадкових величин?

9. Як побудувати гістограму для статистичних даних? Які засоби графічного подання статистичних даних надає пакет `stats`?

Вправи до розділу 16

1. Згенерувати тридцять випадкових чисел з логнормальним законом розподілу з середнім значенням 2 і середньоквадратичним відхиленням 1,5 відповідного нормального закону розподілу. Перетворити отримані числа в список `splogn` за допомогою функції $f(x) = 3\sqrt{x} - 1$. Для отриманого списку обчислити середнє арифметичне, середнє квадратичне, середнє гармонічне, середнє геометричне, медіану, моду, діапазон зміни даних, середньоквадратичне відхилення вибірки, коефіцієнт асиметрії вибірки і побудувати гістограму з шістьма стовпцями.

2. Згенерувати список `spgamma` тридцяти випадкових чисел з гамма-розподілом з параметрами $\alpha = 2$, $\beta = 3$. Знайти пряму регресії для списків `spgamma` і `splogn` (з попередньої вправи). Створити точковий графік залежності для цих списків, відобразити також пряму регресії.

3. Побудувати графіки функції щільності розподілу ймовірностей і функції розподілу ймовірностей для гамма-розподілу з параметрами $\alpha = 2$, $\beta = 3$.

4. Список $[2, 3, 4, 2, 0.7, 6, 8, 5, 7/2]$ згрупувати відповідно до списку діапазонів $[0..3, 3..6, 6..10]$ (команда `tallyinto`).

Розділ 17. Деякі пакети Maple

§ 17.1. Комбінаторика

Широкий спектр задач комбінаторного аналізу (комбінаторики) можна розв'язати за допомогою пакетів `combinat` і `construct`.

В основній бібліотеці є команда `binomial(`n,k`)`, яка знаходить біноміальні коефіцієнти $\binom{n}{k}$, причому якщо $n,k\in\mathbb{Z}$ і $0\leqslant k\leqslant n$, то $\binom{n}{k}=C_n^k=\frac{n!}{k!\,(n-k)!}$; у загальному випадку $\binom{n}{k}=\frac{\Gamma(n+1)}{\Gamma(k+1)\Gamma(n-k+1)}$, де $\Gamma(z)=\int\limits_0^\infty s^{z-1}e^{-s}\,ds$ – гамма-функція. Приклади:

```
> binomial(10,3);
```

120

```
> binomial(5,.5);
```

2.58689939247

Ми розглянемо лише пакет `combinat`, який містить 33 команди. Їх назви можна побачити після виконання команди

```
> with(combinat);
```

Наведемо основні команди та відповідні приклади.

Команда `numbcomb(`n,k`)` обчислює кількість комбінацій (сполук) з елементів n, взятих по k, де n – множина чи список виразів або невід'ємне ціле число, k – необов'язковий параметр (невід'ємне ціле число). Якщо k не задане, то обчислюється кількість всіх можливих комбінацій (по всім k від 0 до n чи до кількості елементів в n). Елементи множини чи списка n можуть повторюватись, однак у множині вони, очевидно, рахуватимуться по одному разу. Однакові елементи у списку n дають можливість обчислювати кількість комбінацій з повтореннями. Приклади:

```
> numbcomb([a,b,c,d,f,g],3);
```

$$20$$

```
> numbcomb([a,b,c,d,f,g]);
```

$$64$$

```
> numbcomb([a,b,c,d,f,a,g],3);
```

$$25$$

```
> numbcomb({a,b,c,d,f,a,g},3);
```

$$20$$

Команда `choose(`n, k`)` з аналогічними аргументами виводить всі можливі комбінації з елементів n, взятих по k. Якщо k не задане, то виводяться всі можливі комбінації, включаючи порожню множину (по всім k від 0 до n чи до кількості елементів в n). Треба мати на увазі, що кількість всеможливих комбінацій може перевищити обсяг оперативної пам'яті комп'ютера. Приклади:

```
> choose(4,2);
```

$$[[1,2],[1,3],[1,4],[2,3],[2,4],[3,4]]$$

```
> choose([x,x,y]);
```

$$[[\,],[x],[y],[x,y],[x,x],[x,x,y]]$$

Команда `randcomb(`n, k`)` будує одну випадкову комбінацію k елементів з n:

```
> randcomb(10,5);
```

$$\{1,2,8,9,10\}$$

```
> randcomb([a,b,c,d,e,f], 4);
```

$$[a,b,d,f]$$

Команда `numbperm(`n, k`)` з аналогічними аргументами обчислює кількість розміщень з елементів n, взятих по k. Якщо k відсутнє, то воно вважається рівним кількості елементів у множині (списку) n чи числу n, тобто обчислюється кількість перестановок. Елементи множини (списку) n можуть повторюватись. Кілька однакових елементів множини n вважаються одним елементом. Приклади:

```
> numbperm([x,x,y,z],2);
```

$$7$$

```
> numbperm([x,x,y,z]);
```

$$12$$

Команда `permute(`n, k`)` з аналогічними аргументами будує всі можливі розміщення з елементів n, взятих по k (з повтореннями чи без них) або всі перестановки з елементів n. Приклади:

```
> permute([x,x,y,z],2);
```

$$[[x,x],[x,y],[x,z],[y,x],[y,z],[z,x],[z,y]]$$

```
> permute([x,x,y]);
```

$$[[x,x,y],[x,y,x],[y,x,x]]$$

Команда `randperm(`n, k`)` виводить одне випадкове розміщення чи перестановку:

```
> randperm(5,3);
```

$$[4,3,5,1,2]$$

Команда `multinomial(`$n, n_1, \ldots, n_k$`)` знаходить мультиноміальні (поліноміальні) коефіцієнти за формулою $\binom{n}{n_1,\ldots,n_k} = \frac{n!}{n_1!\ldots n_k!}$, причому $n = n_1 + \ldots + n_k$:

```
> multinomial(5,2,3);
```

$$10$$

Команда `composition`(n, m) видає множину, яка складається з усіх різних впорядкованих m-ок натуральних чисел, сума яких дорівнює n:

```
> composition(6,2);
```

$$\{[1,5],[2,4],[3,3],[4,2],[5,1]\}$$

Команда `numbcomp`(n, m) знаходить кількість різних впорядкованих наборів m натуральних чисел, сума яких дорівнює n (кількість елементів у множині після виконання команди `composition`(n, m)):

```
> numbcomp(6,2);
```

$$5$$

Команда `partition`(n, m) виводить список списків натуральних чисел, сума яких дорівнює n ($m \leqslant n$ – найбільше число у списках):

```
> partition(5,3);
```

$$[[1,1,1,1,1],[1,1,1,2],[1,2,2],[1,1,3],[2,3]]$$

```
> partition(5);
```

$$[[1,1,1,1,1],[1,1,1,2],[1,2,2],[1,1,3],[2,3],[1,4],[5]]$$

Команда `numbpart`(n, m) знаходить кількість всеможливих наборів натуральних чисел, сума яких дорівнює n (m – найбільший доданок у сумах). Іншими словами, ця команда знаходить кількість списків у списку після виконання команди `partition`(n, m):

```
> numbpart(5,3);
```

$$5$$

```
> numbpart(5);
```

$$7$$

Команда `powerset(`s`)` виводить список (множину), що складається зі списків (множин) – усіх підмножин списку чи множини s (кількість таких підмножин дорівнює 2^n, де n – кількість елементів множини чи списку s). В якості s можна використовувати натуральне число. Приклади:

```
> powerset([a,a,a,b]);
```

$$[[\,],[a],[a,a],[b],[a,b],[a,a,b],[a,a,a],[a,a,a,b]]$$

```
> powerset(3);
```

$$\{\{\,\},\{1\},\{2\},\{3\},\{1,2\},\{1,3\},\{2,3\},\{1,2,3\}\}$$

Команда `fibonacci(`n`)` знаходить число Фібоначчі за рекурентною формулою $F(n)=F(n-1)+F(n-2)$, де $F(0)=0$, $F(1)=1$:

```
> fibonacci(50);
```

$$12586269025$$

```
> seq(fibonacci(i), i=1..15);
```

$$1,1,2,3,5,8,13,21,34,55,89,144,233,377,610$$

Команда `fibonacci(`n,x`)` знаходить формулу многочлена Фібоначчі за рекурентною формулою $F(n,x)=xF(n-1,x)+$ $+F(n-2,x)$, де $F(0,x)=0$, $F(1,x)=1$; при цьому $F(n)=$ $=F(n,1)$:

```
> fibonacci(10,x);
```

$$x^9+8\,x^7+21\,x^5+20\,x^3+5\,x$$

Команда `bell(`n`)` знаходить число Белла, тобто число B_n всіх невпорядкованих розбиттів n-елементної множини (за означенням вважають, що $B_0=1$):

```
> bell(4);
```

```
> seq(bell(i), i=1..10);
```

$$1, 2, 5, 15, 52, 203, 877, 4140, 21147, 115975$$

Команда `stirling1(`n, m`)` обчислює число Стірлінга першого роду, тобто коефіцієнт $s(n, m)$ біля степеня x^m у розвиненні многочлена $x(x-1)\dots(x-n+1) = \sum\limits_{m=0}^{n} s(n, m)x^m$:

```
> stirling1(10,5);
```

$$-269325$$

Команда `stirling2(`n, m`)` обчислює число Стірлінга другого роду, тобто число невпорядкованих розбиттів n-елементної множини на m непорожніх підмножин:

```
> stirling2(8,4);
```

$$1701$$

§ 17.2. Пакет simplex

Можливості дослідження функцій на локальний і умовний екстремуми розглядались у §§ 6.6, 6.7. Якщо потрібно знайти змінні, при яких лінійна функція багатьох змінних досягає свого максимуму (мінімуму) за певних обмежень, заданих у вигляді лінійних рівностей або нерівностей, то використовують призначений для розв'язування задач лінійної оптимізації пакет `simplex`. Після його завантаження команди `maximize(`f`,{`*cond*`})` і `minimize(`f`,{`*cond*`})` (§§ 6.6, 6.7) змінюють свою дію: вони видають координати точок, для яких лінійна функція f має максимум або мінімум для заданої системи обмежень лінійними рівностями чи нерівностями *cond*. При цьому можна використовувати додаткову опцію `NONNEGATIVE` для пошуку тільки невід'ємних розв'язків. Якщо розв'язок необмежений, то відповіді не буде. Якщо обмеження *cond* є суперечливими, то відповіддю буде порожня множина {}. Обчислити максимальне (мінімальне) значення, якого набуває функція f можна, як звичайно, за допомогою команди `eval`.

З'ясуємо, наприклад, для яких невід'ємних значень змінних функція $f(x,y) = x + 2y + 3z$ має максимум і знайдемо його, якщо $x + y - 3z \leqslant 4$, $5x - y + 3z \leqslant 8$, $2x + 5z \leqslant 10$.

```
> with(simplex): f:=x+2*y+3*z:
> cond:={x+y-3*z<=4, 5*x-y+3*z<=8, 2*x+5*z<=10}:
> maximize(f, cond, NONNEGATIVE);
```

$$\{x = 0, y = 10, z = 2\}$$

```
> eval(f,%);
```

$$26$$

Якщо потрібно одночасно використовувати команди `maximize` чи `minimize` з основної бібліотеки і з пакета `simplex`, то сам пакет `simplex` не підключають, а для відповідної команди з цього пакета використовують «довгий» запис, наприклад, `simplex[maximize]`(f,{$cond$}).

Команда `feasible`({$cond$}) пакета `simplex` перевіряє на несуперечливість систему обмежень *cond*. Додатково може вказуватись опція `NONNEGATIVE`. Для попереднього прикладу маємо:

```
> feasible(cond,NONNEGATIVE);
```

true

§ 17.3. Спеціальні засоби для наближеного розв'язування рівнянь і систем рівнянь

Пакет `RootFinding` містить кілька спеціальних команд, які дозволяють наближено знаходити розв'язки алгебричних і трансцендентних рівнянь, а також систем алгебричних рівнянь, і доповнюють команду `fsolve` (§ 7.3). Нагадаємо, що команда `fsolve` для трансцендентних рівнянь знаходить лише один дійсний розв'язок на заданому проміжку. Пакет `RootFinding` є лише у версіях Maple, починаючи з Maple 9.5.

Команду `Analytic` можна записувати в одному з двох форматів:

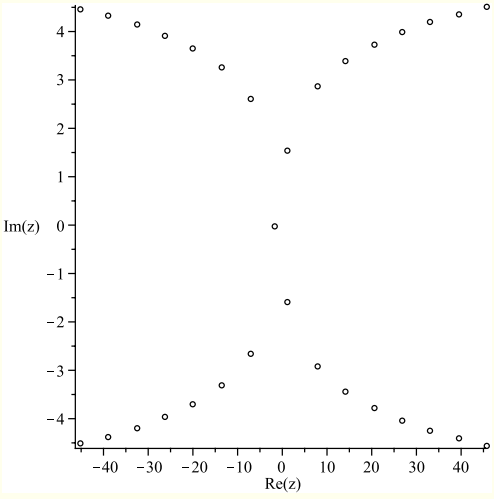
$$\texttt{Analytic(}f(z), z, a\texttt{+}b\texttt{*I..}b\texttt{+}d\texttt{*I}, opts\texttt{)},$$
$$\texttt{Analytic(}f(z), z, \texttt{re=}a\texttt{..}b, \texttt{im=}c\texttt{..}d, opts\texttt{)}.$$

Вона знаходить десяткові наближення всіх (дійсних і комплексних) нулів аналітичної (нескінченну кількість разів неперервно диференційовної) функції $f(z)$ у прямокутнику комплексної площини $a \leqslant \operatorname{Re} z \leqslant b$, $c \leqslant \operatorname{Im} z \leqslant d$. В якості *opts* може бути опція `digits=`n, яка задає кількість значущих цифр n, або опція `plot` для побудови знайдених нулів на комплексній площині. Приклади:

```
> Analytic(sin(z)-z-1,z=-5-5*I..5+5*I, digits=8);
```

$0.81646350 - 1.5635847\,\mathrm{I},\ -1.9345632,\ 0.81646355 + 1.5635847\,\mathrm{I}$

```
> Analytic(sin(z)-z-1,z=-50-50*I..50+50*I,plot);
```



Команда `BivariatePolynomial(`$fg, vars, opts$`)` знаходить десяткові наближення всіх нулів системи двох чи більшої кількості многочленів двох змінних. Тут fg – список чи множина многочленів, *vars* – список чи множина невідомих. Опція

`exact=true` призначена для знаходження точного розв'язку, але відшукати його вдається не завжди. Приклад:

```
> BivariatePolynomial({x^2+y^2-3*x*y+5,x+y^2+x^2+
  6*x*y+5*y-8},{x,y});
```

$$\begin{aligned}
&\{x = 0.5276122757 - 1.283754500\,\mathrm{I},\\
&y = 0.4671508936 + .6853475787\,\mathrm{I}\},\\
&\{x = 0.5276122757 + 1.283754500\,\mathrm{I},\\
&y = 0.4671508936 - 0.6853475787\,\mathrm{I}\},\\
&\{x = -1.249834498 - 0.5675077093\,\mathrm{I},\\
&y = -1.411595338 + 1.063023490\,\mathrm{I}\},\\
&\{x = -1.249834498 + 0.5675077093\,\mathrm{I},\\
&y = -1.411595338 - 1.063023490\,\mathrm{I}\}
\end{aligned}$$

Команда `Homotopy(`fg`)` знаходить десяткові наближення всіх нулів системи n многочленів fg від n змінних:

```
> Homotopy({x^3+y^2-6*x*y*z-z^3,x^2+y^2+z^2+4*x*y*x-7,
  x^3-y^3-z^3+x-5});
```

У зв'язку з громіздкістю область виведення для останнього прикладу не наводимо.

§ 17.4. Інтерполяція та апроксимація

В основній бібліотеці Maple є дві команди для інтерполяції, тобто знаходження проміжних значень величини за наявним дискретним набором відомих значень у вузлах інтерполяції.

Команда `interp(`x, y, v`)` будує інтерполяційний многочлен для списку (вектора) x вузлів інтерполяції і списку (вектора) y значень функції у вузлах інтерполяції, v – змінна многочлена або точка, у якій обчислюватиметься многочлен. Наприклад:

```
> interp([0,1,2,4],[0,4,2,4],x);
```

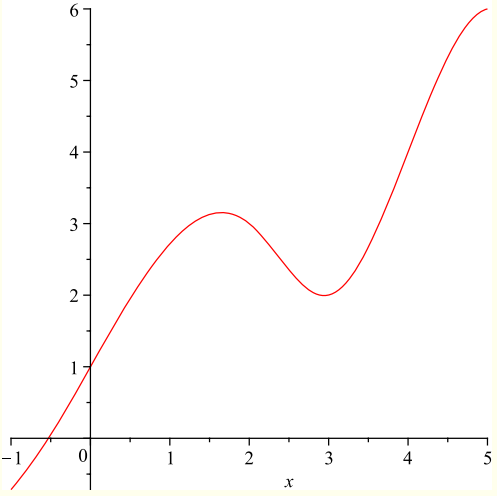
$$x^3 - 6\,x^2 + 9\,x$$

Команда `spline`(x, y, v, d) виконує інтерполяцію сплайнами для списку (вектора) x вузлів інтерполяції й списку (вектора) y значень функції у вузлах інтерполяції, v – змінна, яка використовуватиметься у сплайнах, або число для обчислення значення сплайн-інтерполяції у точці, d – степінь сплайнів (за замовчуванням $d = 3$). Наприклад:

```
> spline([0,2,3,4],[1,3,2,4],x);
```

$$\begin{cases} 1 + \dfrac{45}{23}\,x - \dfrac{11}{46}\,x^3 & x < 2 \\ -\dfrac{269}{23} + 21\,x - \dfrac{219}{23}\,x^2 + \dfrac{31}{23}\,x^3 & x < 3 \\ \dfrac{1108}{23} - \dfrac{894}{23}\,x + \dfrac{240}{23}\,x^2 - \dfrac{20}{23}\,x^3 & otherwise \end{cases}$$

```
> plot(%,x=-1..5);
```



Крім того, є спеціальний пакет `CurveFitting`, призначений для інтерполяції та апроксимації функцій. Розглянемо найважливіші команди цього пакета.

Побудувати інтерполяційний многочлен можна також за допомогою команди `PolynomialInterpolation`(x, y, v, opt) з

тими самими аргументами, що у команді `interp`. Замість двох списків x і y тут та в інших командах пакета можна використовувати список двоелементних списків відповідних вузлів інтерполяції й значень функції в них. Опцією `form=`f, де f – `Lagrange`, `Newton` або `power` (за замовчуванням), можна задати форму відображення інтерполяційного многочлена. Наприклад:

```
> with(CurveFitting):
> PolynomialInterpolation([0,1,2,4],[0,4,2,4],x,
  form=Lagrange);
```

$$\frac{4}{3}\,x\,(x-2)\,(x-4)-\frac{1}{2}\,x\,(x-1)\,(x-4)+\frac{1}{6}\,x\,(x-1)\,(x-2)$$

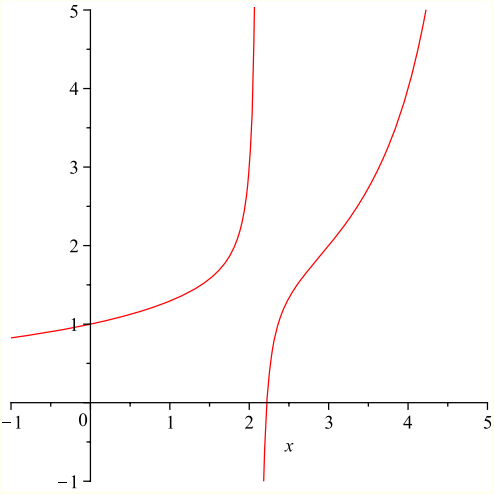
Команда `Spline(`x, y, v, *opts*`)`, як і команда `spline`, виконує інтерполяцію сплайнами. Опція `degree=`d вказує степінь сплайна. Опція `endpoints=`k, де k – `natural` (за замовчуванням), `periodic`, `notaknot` або список, масив, вектор чи матриця, задає поведінку сплайна на кінцях проміжку інтерполяції. Детальніше з використанням опції `endpoints` можна ознайомитись у довідковій системі.

Команда `RationalInterpolation(`x, y, v, *opts*`)` виконує раціональну інтерполяцію, тобто будує частку двох многочленів. Опція `degrees=[`*d1*, *d2*`]` дозволяє вказати степені многочленів у чисельнику і знаменнику. Приклад:

```
> RationalInterpolation([0,2,3,4],[1,3,2,4],x);
```

$$\frac{-54\,x+120}{120-80\,x+11\,x^2}$$

```
> plot(%,x=-1..5,-1..5,discont=true);
```



Команда `LeastSquares`$(x, y, v, opts)$ будує наближення функції методом найменших квадратів. Можливі опції: `curve=`f, де f – лінійна відносно параметрів функція (за замовчуванням $a \cdot v + b$); `params=`$pset$, де $pset$ – множина невідомих параметрів; `weight=`$wlist$, де $wlist$ – список n ваг для всіх n вузлів. Приклади:

```
> LeastSquares([0,2,3,4],[1,3,2,4],x);
```

$$\frac{38}{35} + \frac{22}{35}\,x$$

```
> LeastSquares([0,2,3,4],[1,3,2,4],x,
  curve=a*x^2+b*x+c);
```

$$\frac{123}{110} + \frac{119}{220}\,x + \frac{1}{44}\,x^2$$

Питання до розділу 17

1. Як обчислити кількість комбінацій, розміщень і перестановок? Як вивести всеможливі комбінації, розміщення і перестановки? Як отримати випадкову комбінацію, розміщення, перестановку?

2. Як обчислювати числа Фібоначчі, Белла, Стірлінга першого і другого роду?

3. Як знайти розв'язок, при якому задана лінійна цільова функція досягає екстремуму для заданої системи обмежень у вигляді лінійних рівностей чи нерівностей?

4. Як знайти у певній області всі наближені числові розв'язки трансцендентного рівняння з аналітичними функціями?

5. Як знайти всі наближені числові розв'язки системи алгебричних рівнянь?

6. Як створити інтерполяційний многочлен, виконати сплайн-інтерполяцію, раціональну інтерполяцію, наближення функції методом найменших квадратів?

Вправи до розділу 17

1. Знайдіть кількість комбінацій і розміщень з елементів a, b, b, c, d по 3 елементи, а також кількість перестановок цих елементів. Виведіть всі ці комбінації, розміщення і перестановки.

2. Знайдіть двадцять третє число Фібоначчі.

3. Обчисліть кількість усіх невпорядкованих розбиттів множини з п'ятнадцяти елементів.

4. Знайдіть такі значення невідомих x, y, z, при яких цільова функція $f = 4x - 6y + 2z$ набуває мінімального значення, якщо $3x - 4y + z \leqslant 6$, $2x + 7y - z \geqslant 1$, $x + y + z \leqslant 9$, $x \geqslant 0$, $y \geqslant 0$, $z \geqslant 0$. Обчисліть значення цільової функції у знайденій точці.

5. Знайдіть і побудуйте на комплексній площині всі розв'язки рівняння $e^{3z+1} = z^3 - 3z$ серед тих z, які належать квадрату $-10 \leqslant \operatorname{Re} z \leqslant 10$, $-10 \leqslant \operatorname{Im} z \leqslant 10$.

6. Знайдіть десяткові наближення всіх розв'язків системи

$$\begin{cases} xy + x + y = 25, \\ x^3 + y^3 + x^2y - 15y^2 + 6 = 0. \end{cases}$$

7. Створіть інтерполяційний многочлен, виконайте інтерполяцію сплайнами, раціональну інтерполяцію, лінійне і квадратичне наближення методом найменших квадратів для таблично заданої функції:

x	-1	0	$2/3$	1	2	3	4
y	5	4	3	3	$9/4$	$3/2$	$5/2$

Побудуйте графіки отриманих функцій.

Список рекомендованої літератури

Основна література

1. Білоусова Л. І. Курс вищої математики у середовищі Maple / Л. І. Білоусова, М. М. Горонескуль. – Х. : УЦЗУ, КП «Міська друкарня», 2009. – 412 с.
2. Говорухин В. Компьютер в математическом исследовании: Maple, MATLAB, LaTeX / В. Говорухин, Б. Цибулин. – СПб. : Питер, 2001. – 624 с.
3. Дьяконов В. П. Maple 9.5/10 в математике, физике и образовании / В. П. Дьяконов. – М. : СОЛОН-Пресс, 2006. – 720 с.
4. Кобильник Т. П. Системи комп'ютерної математики: Maple, Mathematica, Maxima / Т. П. Кобильник. – Дрогобич : Редакційно-видавничий відділ ДДПУ імені Івана Франка, 2008. – 315 с.
5. Матросов А. В. Maple 6. Решение задач высшей математики и механики / А. В. Матросов. – СПб. : BHV-Санкт-Петербург, 2001. – 528 с.
6. Попов Б. О. Розв'язування математичних задач у системі комп'ютерної алгебри Maple V / Б. О. Попов. – К. : ViP, 2001. – 312 с.
7. Сдвижков О. А. Математика на компьютере: Maple 8 / О. А. Сдвижков. – М. : СОЛОН-Пресс, 2003. – 176 с.
8. Maple 7. Основи практичного застосування / [Гірник М. О., Костенко А. В., Лучко М. В., Плеша М. І.]. – Львів : ВНТЛ-Класика, 2002. – 174 с.

Додаткова література

9. Аладьев В. З. Системы компьютерной алгебры: Maple: Искусство программирования / В. З. Аладьев. – М. : Лаборатория Базовых Знаний, 2006. – 792 с.

10. Алексеев Е. Р. Решение задач вычислительной математики в пакетах Mathcad 12, MATLAB 7, Maple 9 / Е. Р. Алексеев, О. В. Чеснокова. – М. : НТ Пресс, 2006. – 496 с.

11. Дьяконов В. П. Maple 7: Учебный курс / В. П. Дьяконов. – СПб. : Питер, 2002. – 672 с.

12. Махней О. В. Лабораторний практикум у Maple : методичні рекомендації до проведення лабораторних занять / О. В. Махней. – Івано-Франківськ : Видавничо-дизайнерський відділ Центру інформаційних технологій Прикарпатського національного університету імені Василя Стефаника, 2010. – 32 с.

13. Основи роботи в пакеті Maple. Частина 2. Інструментальні засоби математичного моделювання / [уклад. Гірник М. О., Лучко М. В.]. – Львів : Видавництво Львівської комерційної академії, 2006. – 39 с.

14. Савотченко С. Е. Методы решения математических задач в Maple / С. Е. Савотченко, Т. Г. Кузьмичева. – Белгород : Белаудит, 2001. – 116 с.

15. Сараев П. В. Основы использования математического пакета MAPLE в моделировании / П. В. Сараев. – Липецк : Международный институт компьютерных технологий, 2006. – 119 с.

16. Abell M. L. Maple by Example / M. L. Abell, J. P. Braselton. – Elsevier Academic Press, 2005. – 550 p.

17. Adams P. Introduction to Mathematics with Maple / P. Adams, K. Smith, R. Vyborny. – World Scientific Publishing Company, 2004. – 544 p.

18. Borwein J. M. An Introduction to Modern Mathematical Computing: With Maple/ J. M. Borwein, M. P. Skerritt. – Springer Science+Business Media, 2011. – 216 p.

Предметний покажчик

Навчальне видання

Махней Олександр Володимирович
Гой Тарас Петрович

МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ АВТОМАТИЗАЦІЇ ПРИКЛАДНИХ ДОСЛІДЖЕНЬ

Технічний редактор *О. В. Махней*

Підписано до друку 07.06.2013. Формат 60×84/16.
Папір офсетний. Друк цифровий.
Гарнітура CM Roman. Умовн. друк. арк. 17,67.
Тираж 100. Зам. № 78 від 07.06.2013.

Видавництво «Сімик»
76018, м. Івано-Франківськ,
вул. Теодора Цьоклера, 9а,
тел.: (0342) 78-91-26, e-mail: symyk@com.if.ua.

Свідоцтво про внесення до Державного реєстру суб'єкта видавничої справи серія ІФ № 11 від 27.03.2001 року.

Віддруковано: Приватний підприємець Голіней О. М.
76000, м. Івано-Франківськ,
вул. Галицька, 128,
тел.: (0342) 58-04-32.