МІЖРЕГІОНАЛЬНА
АКАДЕМІЯ УПРАВЛІННЯ ПЕРСОНАЛОМ

А. В. Кузьмін, Н. М. Кузьміна, А. Б. Телейко

СИМВОЛЬНІ ТА НАБЛИЖЕНІ ОБЧИСЛЕННЯ В СИСТЕМІ MAPLE

Частина 2

Навчальний посібник

Київ
ДП «Видавничий дім «Персонал»
2008

ББК 32.973.26-018.2я73
К89

Рецензенти: *М. І. Жалдак*, д-р пед. наук, проф., акад.
В. С. Саженюк, канд. фіз.-мат. наук
В. В. Попов, канд. фіз.-мат. наук, проф.

Схвалено Вченою радою Міжрегіональної Академії управління персоналом (протокол № 1 від 31.01.07)

Кузьмін, А. В.
К89 Символьні та наближені обчислення в системі Maple: Навч. посіб. / А. В. Кузьмін, Н. М. Кузьміна, А. Б. Телейко. — К. : МАУП, 2006–2008.
Ч. 2. — К. : ДП «Видавничий дім «Персонал», 2008. — 128 с.: іл.— Бібліогр.: с. 123.

ISBN 978-966-608-860-7

Навчальний посібник є продовженням частини 1 “Символьні обчислення в системі Maple” і містить навчальний матеріал, приклади розв’язання задач та задачі для самостійного розв’язання за такими розділами: типові засоби програмування в системі Maple; спеціальні функції та робота з ними; наближення функцій; чисельне розв’язування систем нелінійних рівнянь; чисельне інтегрування задач Коші та граничних задач; наближене обчислення інтегралів; знаходження екстремумів функцій; пакет лінійної алгебри. Може використовуватись при вивченні таких дисциплін: “Лінійна алгебра та аналітична геометрія”, “Диференціальні рівняння”, “Чисельні методи”, “Чисельні методи в інформатиці”, “Чисельні методи математичної фізики”, “Символьні обчислення”, “Комп’ютерна алгебра” та ін.

Для студентів освітніх напрямків “Прикладна математика” та “Комп’ютерні науки”.

ББК 32.973.26-018.2я73

ISBN 978-966-608-860-7

ВСТУП

Цей методичний посібник, що є продовженням частини 1 “Символьні обчислення в системі Maple”, присвячений вивченню аналітичних можливостей системи Maple, зокрема можливостей наближених (чисельних) обчислень.

Викладений матеріал значною мірою сприятиме підвищенню продуктивності науково-дослідної роботи, пов’язаної з математичним моделюванням реальних процесів, розробкою алгоритмів та програм, обробкою та поданням результатів дослідження, написанням звітів; засвоєнню таких навчальних дисциплін, як “Чисельні методи”, “Чисельні методи математичної фізики”, “Рівняння математичної фізики”, “Символьні обчислення та комп’ютерна алгебра”, “Лінійна алгебра та аналітична геометрія”, “Диференціальні рівняння”.

Система Maple створена в 1980 р. групою вчених університету Waterloo (Канада) під керівництвом Кейта Геддеса і Гастона Гоне. Спочатку вона була реалізована на великих комп’ютерах і тривалий час апробовувалась, увібравши велику кількість математичних функцій і правил їх перетворення, вироблених математикою за сторіччя розвитку. Реалізації цієї програми є на платформах ПК Macintosh, Unix, Sun, а нещодавно спрощена версія Maple для операційної системи Windows CE почала використовуватися в мініатюрних комп’ютерах фірми “Casio” під назвою “Cassiopeia”.

Цій системі присвячено багато праць [1–15]. Доволі повний список праць (близько 400) із систем Maple розміщено на сайті розроблювача цієї системи — компанії Waterloo Maple Software: www.maplesoft.com. Переважна їх більшість видана за кордоном, а російською мовою, особливо щодо останніх версій цієї програми, дуже мало.

Особливо ефективно використовувати програму Maple при вивченні математики. Найвищий “інтелект” цієї системи поєднується з ефективними засобами чисельного моделювання та можливостями графічної візуалізації. Застосування системи Maple ефективне як при викладанні математики, так і при самостійному її опануванні, причому від основ до вершин.

Практика засвідчує, що найважливіший перший етап засвоєння системи. Перше ознайомлення з програмою зачаровує користувачів, переконуючи в безмежних можливостях системи за відсутності її систематизованого опису. Через це користувачі часто відкладають вивчення системи.

Пропонований посібник дає змогу швидко ознайомитись з великою кількістю функцій системи Maple, що входять у спеціальні пакети прикладних програм (ППП), переглянути десятки прикладів розв'язання задач за допомогою системи Maple та закріпити здобуті знання при самостійному розв'язанні близько 150 задач. Багато уваги в посібнику приділено вивченню додаткових порівняно з першою частиною можливостей інтерфейсу користувача і основних прийомів програмування в системі Maple. Окремо розглядаються ППП для розв'язання спеціалізованих задач. Ця потужна частина можливостей системи MAPLE сприяє значному підвищенню загальної математичної культури студента та ефективності виконання математичних досліджень у теоретичній та прикладній галузях.

Система *комп'ютерної математики* Maple розрахована на широке коло користувачів. Донедавна її називали системою комп'ютерної алгебри, що свідчило про особливу роль символьних обчислень і перетворень, які здатна здійснювати ця система. Але така назва звужує сферу застосування системи. Насправді, по-перше, вона здатна швидко й ефективно виконувати числові розрахунки, причому з необмеженою точністю, а по-друге, її можливості виходять далеко за межі алгебри і вже покривають, наприклад, велику частину математичного аналізу.

Система Maple має широку сферу застосування — від моделювання складних фізичних об'єктів, систем і пристроїв до створення художньої графіки (наприклад, фракталів). Заслужену популярність система Maple зажила у вищих закладах освіти — понад 300 найбільших університетів світу взяли її на озброєння. А зареєстрованих користувачів системи вже понад мільйон.

Maple — одна з найпотужніших інтегрованих систем символьної математики — завоювала лідерство в гострій конкурентній боротьбі з іншою математичною системою — Mathematica. Кожна з цих систем має специфічні особливості, проте загалом вони рівнозначні. Однак треба зазначити, що поява нових версій Maple 9–11 означає черговий виток у змаганні цих систем за місце лідера на світовому ринку, причому ривок цього разу зробила система Maple.

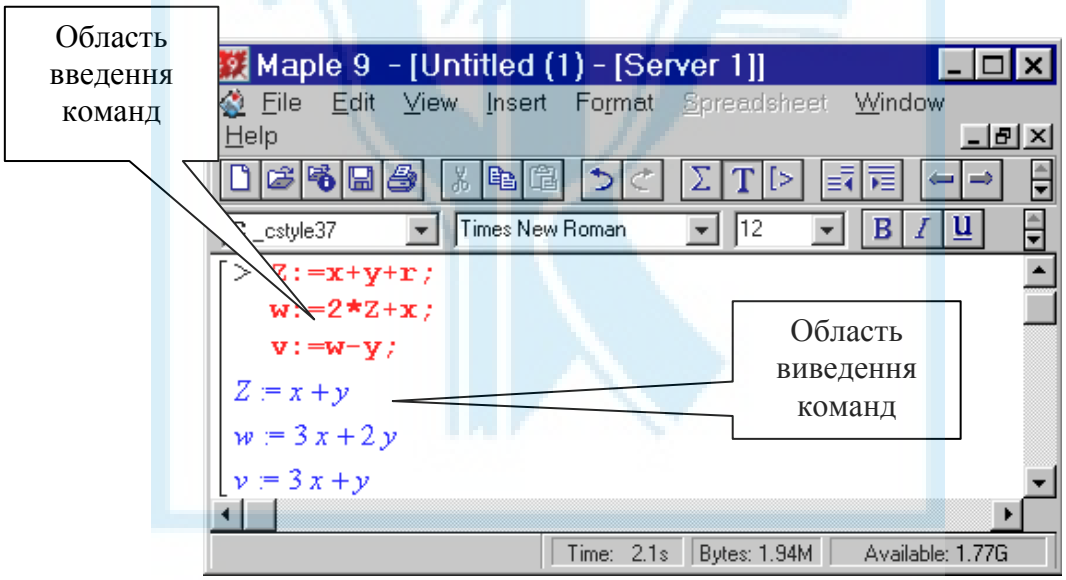
Maple є типовою інтегрованою системою з такими властивостями:

- вбудованим ядром для перетворення математичних виразів;
- потужною довідковою системою з великою кількістю прикладів;
- числовими і символьними процесорами;
- візуалізацією обчислень, науковою та інженерною графікою;
- бібліотекою вбудованих функцій і додаткових пакетів;
- вбудованою мовою програмування.

У цьому зв’язку Maple може застосовуватись як для класичних, так і прикладних математичних досліджень у більшості галузей науки, що використовують математичні моделі, включаючи економіку, фізику, екологію, біологію, соціологію. Іншими словами, Maple — ефективний інструмент для *математичного моделювання*.

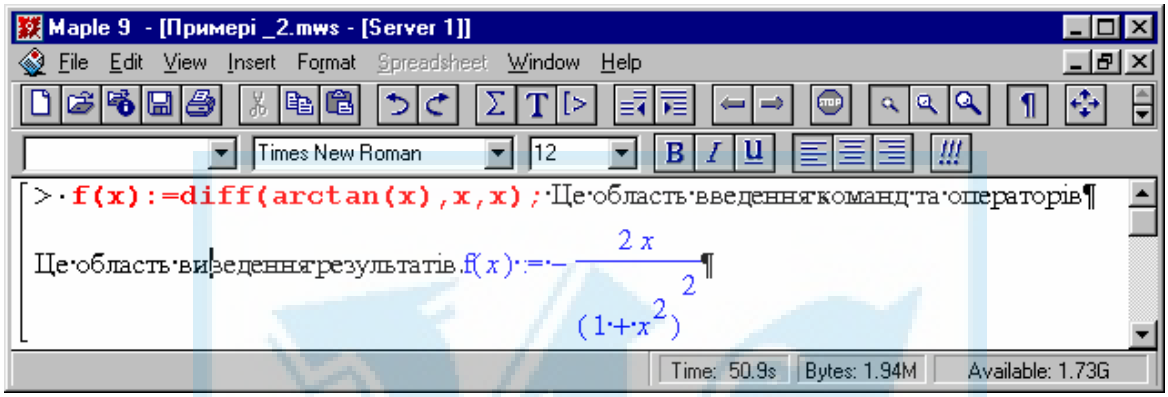
Структура робочого аркуша Maple-програми

Робота в системі Maple означає інтерактивний сеанс, під час якого користувач вводить на робочому аркуші команди і після натискання клавіші <Enter> передає їх для виконання ядру системи. Усі введені команди та відображені результати їх роботи становлять вміст робочого аркуша. Наприкінці сеансу цей аркуш можна зберегти на жорсткому диску у вигляді файла в одному із запропонованих форматів. При черговому сеансі роботи слід відкрити файл і повторно виконати відповідні команди або внести корективи у відповідні команди.



Оскільки команди і результати їх роботи виводяться на одному робочому аркуші, а після кожної команди безпосередньо відображається результат її роботи, робочий аркуш прийнято поділяти на області введення та виведення команд. У першій користувач вводить команди, а у другій отримує результати їх роботи. Вміст областей введення та виведення команд утворює ***групу обчислень***, яка на робочому аркуші зліва помічається квадратною дужкою. Група обчислень може містити кілька команд, при цьому основна властивість групи полягає в тому, що всі її команди і оператори виконуються за одне звернення до ядра системи Maple після натискання клавіші <Enter>.

Для додавання нової групи обчислень необхідно встановити курсор у потрібне місце робочого аркуша, виконати команду ***Insert> Execution Group*** та вибрати одну з двох можливостей вище або нижче положення курсора. Після існуючої групи обчислень наступна група обчислень утворюється автоматично після натискання клавіші <Enter> .



Для запису кількох команд в одній області введення необхідно після введення чергової команди натиснути комбінацію клавіш <Shift> + <Enter>. У цьому разі область введення міститиме кілька команд послідовно одна за одною, а після них міститиметься область виведення послідовності команд.

Знак запрошення передує тільки першій команді.

В одному рядку можуть міститись кілька команд, що розділяються знаком “;” або “:”. Довгі за розміром команди автоматично переносяться системою Maple на наступний рядок; при цьому команда виконуватиметься як звичайна, що міститься в одному рядку.

Команди можна вводити та відображати у формі синтаксису мови Maple або у стандартній математичній формі; при цьому команда може відображатись в активній формі, тобто яка підлягає виконанню, або в пасивній, тобто яка не виконується. Ознакою активної форми команди є знак запрошення у вигляді символу “>”, ознакою пасивної команди — знак питання, що передує команді перед її введенням. Для введення активної команди у стандартній формі синтаксису Maple, яка встановлена в системі Maple за замовчуванням, необхідно ввести текст команди одразу після символу запрошення “>”. Для введення команди у стандартній математичній формі необхідно виконати команду ***Insert> Standard Math Input***; у цьому разі вигляд курсора змінюється на знак запитання і текст команди вводиться в командному рядку панелі інструментів. Команда з’являється в області введення команди після натискання клавіші

<Enter>. Після повторного натискання цієї клавіші виконується синтаксично правильна команда та відображається результат в області виведення команди.

```
> diff(arctan(x),x$2);
```

$$-\frac{2x}{(1+x^2)^2}$$

Активна форма команди у формі синтаксису Maple.

$$\frac{d^2}{dx^2}\arctan(x) \quad -\frac{2x}{(1+x^2)^2}$$

Активна форма команди у стандартній математичній формі.

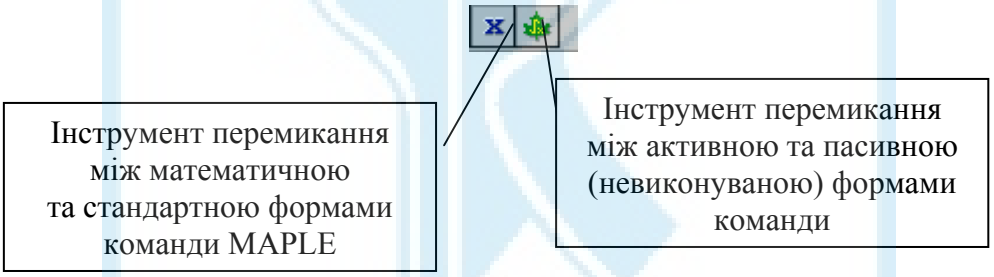
```
diff(arctan(x),x$2);
```

Пасивна форма команди у формі синтаксису Maple.

$$\frac{d^2}{dx^2}\arctan(x)$$

Пасивна форма команди у стандартній математичній формі

Для перемикання між різними формами роботи з командами використовуються кнопки на панелі інструментів, які працюють як звичайні перемикачі.



Робочий аркуш може містити текстові коментарі, за допомогою яких створюють пояснення до математичних обчислень. Для введення в робочий аркуш текстового коментаря необхідно виконати команду ***Insert >Text***. Для додавання до коментаря математичної формули слід виконати команду ***Insert >Standard Math***, після якої можна ввести пасивну команду у стандартній математичній формі. Наведемо приклад відповідного текстового коментаря з математичними формулами.

Значення другої похідної від $\sin(x)$ ***дорівнює*** $\frac{d^2}{dx^2}\sin(x)=\cos(x).$

Структурування робочого аркуша

Структурування робочого аркуша в системі Maple здійснюється за рахунок утворення всередині аркуша ієрархічної деревоподібної струк-

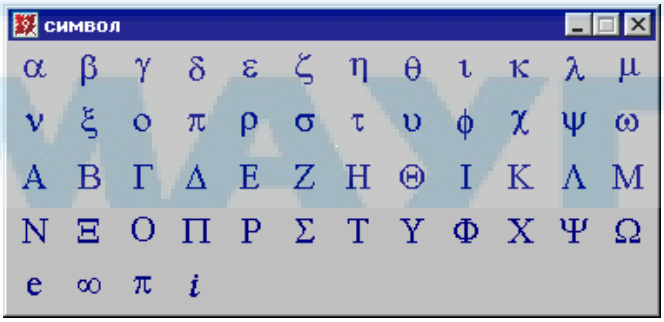
тури, аналогічної структурі директорій файлової системи, що складається із секцій, всередині яких можуть міститись інші секції (підсекції). Глибина вкладання секцій не обмежується. Кожна секція вставляється у вигляді кнопки зі знаком “+” або “–”; це вказує на те, що секція перебуває у згорнутому стані й може бути розгорнута натисканням на кнопку зі знаком “+” або в розгорнутому стані й може бути згорнута натисканням на кнопку зі знаком “–”. Для додавання секції необхідно встановити курсор у потрібне місце робочого аркуша та виконати команду ***Insert > Section***, для додавання підсекції всередині секції необхідно встановити курсор введення праворуч кнопки секції та виконати команду ***Insert > Subsection.***

Кожна секція (підсекція) може мати назву, яка вводиться безпосередньо після кнопки секції.

Для об’єднання існуючих груп обчислень, текстових коментарів у секцію необхідно виділити їх за допомогою миші та виконати команду ***Format > Indent***. Для видалення груп обчислень та текстових коментарів із секції (ліквідація секції, але не її вмісту) необхідно виконати команду ***Format > Outdent***.

Отже, концепції секцій та підсекцій надають користувачеві додаткову свободу управління документом, уможливлюють створення електронних документів, зручних для перегляду на комп’ютері.

Для введення різних команд користувач використовує різні комбінації символів клавіатури. Водночас для зручності система Maple пропонує користувачу чотири палітри для введення символів грецького алфавіту та спеціальних математичних символів, виразів, матриць і векторів. Для використання відповідних палітр необхідно виконати команду ***View > Palettes*** та скористатися відповідним об’єктом палітри.



Наприклад, для введення в команду грецької букви α можна ввести в неї назву “***alpha***” або натиснути кнопку “Символ” на панелі інструментів.

Для введення команди обчислення певного інтеграла можна ввести відповідну команду `>int(f(x),x=a..b);` або натиснути відповід-

ний інструмент > `int(%?, %?=%?..%?);` на панелі інструментів “Expression” та заповнити шаблон необхідними параметрами.

Для управління процесом візуалізації робочого аркуша Maple існує можливість приховати різні елементи робочого аркуша за допомогою пункта головного меню ***Edit***. За допомогою команд ***Hide Content > Hide Input, Hide Content > Hide Output, Hide Content > Hide Graphics*** при їх активізації приховують відповідно області введення, виведення та графічні об’єкти.

Базові засоби програмування в системі Maple

Умовні конструкції

Для створення умовних конструкцій у системі Maple використовується оператор ***if***. Синтаксис цього оператора має загальноприйнятий у мовах програмування вигляд:

```
if <Логічний вираз 1> then <Послідовність операторів 1> |e1if < Логічний вираз 2> then <Послідовність операторів 2>| |e1se <Послідовність операторів n+1>| fi:
```

Семантика оператора має такий вигляд: якщо `<Логічний вираз 1>` має істинне значення, то виконується `<Послідовність операторів 1>` до першого `e1if`, якщо це значення хибне, то перевіряється `< Логічний вираз 2>` і якщо його значення істинне, то виконується `<Послідовність операторів 2>`, якщо значення `< Логічний вираз 2>` хибне, то виконується `<Послідовність операторів n+1>`. Блоків ***elif*** може бути будь-яка кількість, відповідні їм логічні вирази послідовно перевіряються і виконується відповідна група операторів, якщо логічний вираз приймає істинне значення. За допомогою вертикальних рисок | | вказують елементи конструкції ***if***, які можуть бути пропущені.

Наведемо приклад використання умовного оператора:

```
>  x:=2.5;
>if x <0 then f(x):=0
elif (x>=0 and x<1)then f(x):=x
elif (x>=1 and x<2) then f(x):=1
elif x>=2 and x< 3 then f(x):=1+(x-2)
elif x>=3 and x<4 then f(x):=3
else f(x):=3+(x-4)
fi;
```

$$x := 2.5$$

$$f(2.5) := 1.5$$

Оператори циклу

Для організації повторних обчислень у системі Maple існують оператори циклу. Синтаксис оператора циклу має вигляд

`for <ім'я змінної 1> from <вираз 1> |to <вираз 2>| |by <вираз 3>| do <Набір операторів > od;`

Тут `<ім'я змінної 1>` — ім'я змінної управління циклом; `<вираз 1>, <вираз 2>, <вираз 3>` — вирази, що задають відповідно початкове, кінцеве значення та крок зміни змінної `<ім'я змінної 1>`.

Якщо складова циклу `by <вираз 3>` відсутня, то крок зміни змінної управління дорівнює +1.

Наведений оператор циклу виконуватиметься доти, поки змінна управління не перевищить значення `<вираз 2>`, якщо конструкція `to <вираз 2>` відсутня, то цикл виконуватиметься нескінченну кількість разів і для його припинення потребуватимуться додаткові дії.

Цикл можна перервати за допомогою додаткової конструкції `while <вираз 4>`. Синтаксис такого циклу має вигляд

`for <ім'я змінної 1> from <вираз 1> |to <вираз 2>| |by <вираз 3>| while <умова 1> do <Набір операторів> od;`

Цей цикл виконуватиметься доти, поки `<ім'я змінної 1> < <вираз 2>`, і завершиться як тільки `<умова 1>` стане хибною.

Існує ще одна форма оператора циклу з конструкцією `in`, який має синтаксис

`for <ім'я змінної 1> |in <список значень>| |while <умова 1>| do <Набір операторів> od;`

Цей список виконуватиметься доти, поки не вичерпається `<список значень>` та виконуватиметься `<умова 1>`.

Система Maple має також спрощену конструкцію циклу

`while <умова 1> do <Набір операторів > od;`

У цьому разі набір операторів виконуватиметься доти, поки істинною є `умова 1`.

Розглянемо приклад використання оператора циклу для знаходження суми з додатковою умовою:

```
> s:=0;                    s := 0
> for i from 1 to 10000 by 2 while s<1.05 do
s:=s+1/(i^3) od: evalf(s);i;
                1.050075546                  13
```

Оператори переривання та пропускання циклу

Іноді постає потреба пропустити окреме значення змінної циклу. Для цього застосовується оператор ***next.*** Наведемо приклад використання оператора ***next*** у складі операторів ***if – fi***:

```
> for i in [1,2,5,7,9,11] do if i=7 then next else
k[i]:=i^2; print(k[i]) fi od;
```

1 4 25 81 121

Оператор ***break*** перериває виконання фрагменту програми або циклу. Його дію пояснює такий приклад:

```
>  for i in [1,2,5,7,9,11] do if i=7 then break else
k[i]:=i^2; print(k[i]) fi od;
```

1 4 25

Застосування функцій до списків та множин

Коли потрібно виконати деяку операцію або обчислити функцію стосовно кожного елемента списку або множини, окрім можливості застосування оператора циклу система Maple надає можливість використання спеціальних команд:

map(функція або команда, список або множина [,пар1, пар2, … …, парN]);

map2(функція або команда, пар1, список або множина [, пар2, … …, парN]);

Команда ***map*** дає змогу застосувати функцію або команду, що вказана першим параметром, до всіх елементів списку або множини, що вказана другим параметром; при цьому елементи списку або множини становлять перший параметр функції або команди. Якщо функція або команда потребує додаткових аргументів, вони вказуються після списку або множини у вигляді пар1, пар2, …., парN. Результатом роботи команди ***map*** є відповідно список або множина:

```
> map(diff,[sin(x),cos(x),x^2],x$2);
```

$$[-\sin(x), -\cos(x), 2]$$

```
> map(cos,[x,y,z,v,w]);
```

$$[\cos(x), \cos(y), \cos(z), \cos(v), \cos(w)]$$

```
> map(int ,[seq(x^2*sin(k*x),k=1..10)],x=0..2*Pi);
```

$$\left[-4\pi^2, -2\pi^2, -\frac{4}{3}\pi^2, -\pi^2, -\frac{4}{5}\pi^2, -\frac{2}{3}\pi^2, -\frac{4}{7}\pi^2, -\frac{1}{2}\pi^2, -\frac{4}{9}\pi^2, -\frac{2}{5}\pi^2\right]$$

Команда ***map2*** дає змогу застосувати функцію або команду, що вказана першою, до виразу, що вказаний у вигляді пар1; як другий параметр функції або команди використовуються елементи списку або множини, а як можливі додаткові параметри — пар2, …., парN. Команда ***map2*** повертає список або множину:

```
> f:=(x,y,z)->x^y+z;
```

$$f := (x, y, z) \to x^y + z$$

```
> map2(f,x,[1,2,3,4,5,6],sqrt(x));
```

$$[x+\sqrt{x},\ x^2+\sqrt{x},\ x^3+\sqrt{x},\ x^4+\sqrt{x},\ x^5+\sqrt{x},\ x^6+\sqrt{x}]$$

```
> map2(diff,sin(x*y*z)/(x*y),[x,y,z]);
```

$$\left[\frac{\cos(x\,y\,z)\,z}{x}-\frac{\sin(x\,y\,z)}{x^2\,y},\ \frac{\cos(x\,y\,z)\,z}{y}-\frac{\sin(x\,y\,z)}{x\,y^2},\ \cos(x\,y\,z)\right]$$

Процедури та процедури-функції

Мова програмування Maple так само, як і мови програмування високого рівня, містить спеціальні конструкції — процедури. Процедурою називається самостійний модуль програми, призначений для виконання відокремленої сукупності операцій. Процедура є найважливішим елементом структурного програмування і використовується для розширення можливостей системи Maple.

Найпростіша форма процедури має такий вигляд:

name: = proc (Параметри)
тіло процедури
end;

де ***name*** – ім'я процедури.

Параметри процедури задаються у вигляді списку змінних, наприклад proc (x,y,z). При цьому списки формальних і фактичних параметрів повинні відповідати один одному в тому розумінні, що однакові місця в цих списках повинні займати змінні одного типу.

Розглянемо приклад процедури, яка обчислює коефіцієнти Фур'є заданого порядку для заданої функції:

```
> koef:=proc(f,k,ks1,kc1)
ks1:=(int(f(x)*sin(k*x),x=0..2*Pi)/int((sin(k*x))^2,x=0..2*Pi)):
kc1:=(int(f(x)*cos(k*x),x=0..2*Pi)/int((cos(k*x))^2,x=0..2*Pi)):
end proc;
```

```
koef := proc(f, k, ks1, kc1)
    ks1 := 'int'(f(x)*sin(k*x), x = (0 .. 2*Pi))/'int'(sin(k*x)^2, x = (0 .. 2*Pi));
    kc1 := 'int'(f(x)*cos(k*x), x = (0 .. 2*Pi))/'int'(cos(k*x)^2, x = (0 .. 2*Pi))
end proc
```

Звернення до відповідної процедури має вигляд

```
> g:=x->x^7;          #Задання функції
```

$$g := x \to x^7$$

```
> ks:='ks';kc:='kc';
> koef(g,8,ks,kc):
> ks;kc;
```

$$ks := ks \qquad kc := kc$$

```
# Звернення до процедури
# Виведення значень коефіцієнтів Фур'є
```

$$-\frac{\frac{105}{512}\pi^3+\frac{315}{65536}\pi-16\pi^7+\frac{21}{8}\pi^5}{\pi} \qquad \frac{7\pi^6+\frac{315}{8192}\pi^2-\frac{105}{128}\pi^4}{\pi}$$

Локальні та глобальні змінні

Змінні, що використовуються у процедурі, можуть бути або локальними, або глобальними стосовно неї. Усі змінні, що визначені поза тілом процедури, стосовно неї глобальні. Процедура може використовувати та змінювати їх значення, причому змінені значення будуть доступні й поза тілом процедури. Усі змінні, які визначені та використовуються лише в тілі процедури, є локальними і їх імена можна використовувати для визначення поза тілом цієї процедури без жодних обмежень.

У системі Maple існує алгоритм автоматичного визначення, яка зі змінних локальна, а яка глобальна; водночас існує можливість явної об'яви глобальних і локальних змінних за допомогою операторів:

local L1,L2,...,Ln;

global G1,G2,...,Gm;

Якщо ж у процедурі немає об'яв локальних та глобальних змінних, то в Maple діє таке правило для визначення того, чи локальна змінна:

- змінна з'являється в лівій частині оператора присвоювання;
- змінна є змінною параметра циклу ***for*** або індексною змінною операторів ***seq(), add().***

Якщо жодне з правил не можна застосувати до деякої змінної, вона вважається глобальною.

Локальні змінні відрізняються від глобальних також алгоритмом їх обчислення. Для глобальних змінних використовується правило повного обчислення, тобто якщо у вираз, який попередньо був присвоєний змінній, входила невідома величина, якій потім було надано деяке значення, то це значення підставляється в попередній вираз до повного обчислення глобальної змінної, поки обчислення не дійде до невідомих величин або числових значень.

Локальні змінні у процедурах не обчислюються згідно з правилом повного обчислення. Для них у системі Maple передбачено правило обчислення точно на один рівень, тобто якщо при визначенні локальної змінної використовувалась невідома величина, якій пізніше було присвоєно значення, то цей факт не враховується.

Використання окремих спеціальних функцій та ортогональних поліномів

При розв'язуванні звичайних диференціальних рівнянь дуже часто розв'язки задач неможливо знайти у вигляді суперпозиції елементарних функцій і тому в математиці крім елементарних функцій використовуються вищі трансцендентні функції або як їх ще називають — спеціальні функції. Більшість цих функцій є особливими розв'язками звичайного диференціального рівняння другого порядку вигляду $\frac{d}{dx}\left[p(x)\frac{dy}{dx} \right] - q(x)y = 0$, де коефіцієнт $p(x)$ обертається у нуль в одній або кількох точках проміжку зміни x.

Ортогональні поліноми

Класичні ортогональні поліноми $p_n(x),\ n = 0,\infty$, що є ортогональними на відрізку (a,b) з ваговим множником $\rho(x)$, задовольняють диференціальному рівнянню

$$\sigma(x)p_n''(x) + \tau(x)p_n'(x) + \lambda_n p_n(x) = 0,$$

де $\lambda_n = -n[\tau'(x) + 0{,}5(n-1)\sigma''(x)]$; $\tau(x)$ — поліном першого степеня, $\sigma(x)$ залежно від типу інтервалу (a,b) має вигляд

$$\sigma(x) = \begin{cases} (x-a)(b-x), & a \neq -\infty,\ b \neq \infty; \\ (x-a), & a \neq -\infty,\ b = \infty; \\ (b-x), & a = -\infty,\ b \neq \infty; \\ 1, & a = -\infty,\ b = \infty; \end{cases}$$

при цьому вага $\rho(x)$ задовольняє диференціальному рівнянню $[\sigma(x)\rho(x)]' = \tau(x)\rho(x)$. Залежно від значень a, b для $\rho(x)$ існують такі значення:

$$\rho(x) = \begin{cases} (x-a)^{\alpha}(b-x)^{\beta}, & \alpha = -\dfrac{\tau(b)}{b-a} - 1,\ \beta = \dfrac{\tau(a)}{b-a} - 1,\ a \neq -\infty,\ b \neq \infty; \\ (x-a)^{\alpha}e^{x\tau'}(x), & \alpha = \tau(a) - 1, \quad a \neq -\infty,\ b = \infty; \\ (b-x)^{\alpha}e^{-x\tau'}(x), & \alpha = -\tau(b) - 1, \quad a = -\infty,\ b \neq \infty; \\ e^{\int \tau(x)dx}, & a = -\infty,\ b = \infty. \end{cases}$$

Наведемо таблицю для основних характеристик найпоширеніших ортогональних поліномів Якобі, Лагерра та Ерміта.

(a,b)	$\rho(x)$	$\sigma(x)$	$\tau(x)$	$p_n(x)$
$(-1,1)$	$(1-x)^{\alpha}(1+x)^{\beta}$	$1-x^2$	$-(\alpha+\beta+2)x+\beta-\alpha$	$P_n^{(\alpha,\beta)}(x)$ Якобі
$(0,\infty)$	$e^{-x}x^{\alpha}$	x	$-x+\alpha+1$	$L_n^{\alpha}(x)$ Лагерра
$(-\infty,\infty)$	e^{-x^2}	1	$-2x$	$H_n(x)$ Ерміта

Важливими частинними випадками поліномів Якобі є такі поліноми:

- Лежандра $P_n(x)$ $(\alpha=\beta=0)$;
- Чебишева першого та другого роду
  $T_n(x)$ $\left(\alpha=\beta=-\frac{1}{2}\right)$, $U_n(x)$ $\left(\alpha=\beta=\frac{1}{2}\right)$;
- Гегенбауера (ультрасферичні) $C_n^{\lambda}(x)$ $\left(\alpha=\beta=\lambda-\frac{1}{2}\right)$.

Для виклику в системі Maple ортогональних поліномів необхідно підключити пакет за допомогою команди ***with(orthopoly)*** та скористатись іменами відповідних їм функцій:

- $G(n,a,x)$ — поліноми Гегенбауера;
- $H(n,x)$ — поліноми Ерміта
- $L(n,a,x)$ — поліноми Лагерра;
- $P(n,x)$ — поліноми Лежандра;
- $P(n,a,b,x)$ — поліноми Якобі;
- $T(n,x)$ — поліноми Чебишева першого роду;
- $U(n,x)$ — поліноми Чебишева другого роду.

Розглянемо приклади використання ортогональних поліномів.

Поліном Чебишева першого роду з першого по п’ятий степінь включно обчислюється так:

```
with(orthopoly);          [G, H, L, P, T, U]
> map(T,[1,2,3,4,5],x);
```

$$[x, -1+2x^2, 4x^3-3x, 1+8x^4-8x^2, 16x^5-20x^3+5x]$$

Корені полінома Ерміта п’ятого степеня

```
solve(H(5,x),x);
```

$$0, -\frac{\sqrt{10+2\sqrt{10}}}{2}, \frac{\sqrt{10+2\sqrt{10}}}{2}, -\frac{\sqrt{10-2\sqrt{10}}}{2}, \frac{\sqrt{10-2\sqrt{10}}}{2}.$$

Поліном Чебишева другого роду з другого по шостий степінь включно обчислюється так:

```
> map(U,[2,3,4,5,6],x);
```

$$[-1+4x^2,\ 8x^3-4x,\ 1+16x^4-12x^2,\ 32x^5-32x^3+6x,\ -1+64x^6-80x^4+24x^2]$$

Перевірка умов ортогональності для поліномів Чебишева другого роду:

```
> int(U(5,x)*U(5,x)*sqrt(1-x^2),x=-1..1);
```
$$\frac{\pi}{2}$$
```
> int(U(5,x)*U(7,x)*sqrt(1-x^2),x=-1..1);
```
$$0$$

Перевірка умов ортогональності для поліномів Лежандра:

```
> (int(P(3,x)*P(10,x),x=-1..1));
```
$$0$$

Перевірка умов ортогональності для поліномів Лагерра:

```
> (int(L(10,1,x)*x^7*exp(-x)*x,x=0..infinity));
```
$$0$$

Формування функції, що становить лінійну комбінацію поліномів Лежандра, та обчислення значення цієї функції в точці 0.5 здійснюються так:

```
> s:=x->sum(i*P(i,x),i=0..5);
```
$$s := x \to \sum_{i=0}^{5} i\,\mathrm{P}(i,x)$$
```
> s(0.5);
```
$$-1.769531250$$

Для роботи з ортогональними поліномами та їх рядами існує пакет, який підключається за допомогою команди

```
>with(OrthogonalSeries);
```
[*Add, ApplyOperator, ChangeBasis, Coefficients, ConvertToSum, Copy, Create, Degree, Derivate, DerivativeRepresentation, Evaluate, GetInfo, Multiply, PolynomialMultiply, ScalarMultiply, SimplifyCoefficients, Truncate*]

Розглянемо окремі функції цього пакета.

Слід зауважити, що при використанні пакета ***OrthogonalSeries*** користувач повинен застосовувати повні імена відповідних ортогональних поліномів, доступних безпосередньо в ядрі системи Maple:

GegenbauerC(*n*,*a*,*x*) — поліноми Гегенбауера;
HermiteH(*n*,*x*) — поліноми Ерміта
LaguerreL(*n*,*a*,*x*) — поліноми Лагерра;
JacobiP(*n*,*a*,*b*,*x*) — поліноми Якобі;
ChebyshevT(*n*,*x*) — поліноми Чебишева першого роду;
ChebyshevU(*n*,*x*) — поліноми Чебишева другого роду.

Функція ***Create(a, F)****;* призначена для створення ряду з ортогональних поліномів, імена яких вказані другим параметром, а коефіцієнти при них — першим. Залежно від першого параметра сума може бути як скінченною, так і нескінченною.

```
>S:=(Create({1/n!,n=1..infinity},
ChebyshevU(n,x)));
```
$$S := \sum_{n=1}^{\infty}\left(\frac{\mathrm{ChebyshevU}(n,x)}{n!}\right)$$

```
>V1:=Create({[1=2,2=4,3=10]},
ChebyshevT(n,x));
```

$$V1 := 2*ChebyshevT(1, x)+10*ChebyshevT(3, x)+4*ChebyshevT(2, x)$$

```
>Create({2^n,n=0..10,
[11=1,12=3]},JacobiP(n,0,0,x));
```

$$3*JacobiP(12, 0, 0, x)+JacobiP(11, 0, 0, x)+Sum(2^n*JacobiP(n, 0, 0, x), n = (0 .. 10))$$

За допомогою функції ***Evaluaate*** можна обчислити значення утвореної оператором ***Create*** зваженої суми ортогональних поліномів у вигляді полінома відповідної змінної, або числове значення, якщо задано значення аргументу.

```
> Evaluate(V1);
```

$$\frac{40 x^3 + 8 x^2 - 28 x - 4}{16}$$

```
> Evaluate(V1,x=1);
> evalf(Evaluate(S,1));
```

(in Evaluate) infinite number of terms

Зауважимо, що при нескінченній сумі команда ***Evaluaate*** не обчислює значення суми і видає про це відповідне повідомлення.

За допомогою команди ***ChangeBasis(***P, R1, R2,…, Rn), де P – поліном *n* змінних; R1, R2, …, Rn — ортогональні поліноми від змінних полінома P, можна подати поліном P у базисі ортогональних поліномів R1, R2,…, Rn:

```
> P1:=10*x^7+2*x^5+6;
```

$$P1 := 10 x^7 + 2 x^5 + 6$$

```
> ChangeBasis(P1,ChebyshevT(n,x));
```

$$6\,ChebyshevT(0, x) + \frac{5}{32}\,ChebyshevT(7, x) + \frac{215}{32}\,ChebyshevT(1, x) + \frac{39}{32}\,ChebyshevT(5, x) + \frac{125}{32}\,ChebyshevT(3, x)$$

```
> ChangeBasis(P1,JacobiP(n,0,0,x));
```

$$6\,JacobiP(0, 0, 0, x) + \frac{160}{429}\,JacobiP(7, 0, 0, x) + \frac{88}{21}\,JacobiP(1, 0, 0, x) + \frac{1888}{819}\,JacobiP(5, 0, 0, x) + \frac{508}{99}\,JacobiP(3, 0, 0, x)$$

```
> P2:=3*x^5*y^5+2*x^2*y+3*x+y+5;
```

$$P2 := 3 x^5 y^5 + 2 x^2 y + 3 x + y + 5$$

```
> ChangeBasis(P2,ChebyshevT(n,x),ChebyshevT(m,y));
```

$$\frac{3}{16}\,ChebyshevT(5, x)\,ChebyshevT(1, y) + 3\,ChebyshevT(1, x)\,ChebyshevT(0, y)$$

$$+\frac{15}{8}\,\mathrm{ChebyshevT}(1,x)\,\mathrm{ChebyshevT}(1,y)+\frac{15}{16}\,\mathrm{ChebyshevT}(3,x)\,\mathrm{ChebyshevT}(1,y)$$
$$+\,\mathrm{ChebyshevT}(2,x)\,\mathrm{ChebyshevT}(1,y)+5\,\mathrm{ChebyshevT}(0,x)\,\mathrm{ChebyshevT}(0,y)$$
$$+2\,\mathrm{ChebyshevT}(0,x)\,\mathrm{ChebyshevT}(1,y)$$

За допомогою функції ***Coefficients*** можна обчислити коефіцієнт (множник) при ортогональному поліномі відповідного степеня:

```
> Coefficients(S,k);
> Coefficients(V1,3);
```

$$\frac{1}{k!}$$
$$10$$

За командою ***ApplyOperatop*** можна застосувати диференціальний оператор з поліноміальними коефіцієнтами до функціонального ряду ортогональних поліномів.

```
> ss:=Create({[5=1]},HermiteH(n,x));
```

$$ss := \mathrm{HermiteH}(5,x)$$

```
> ApplyOperator([x*dx^2, [dx,x]],ss);
```

$$40\,\mathrm{HermiteH}(4,x)+240\,\mathrm{HermiteH}(2,x)$$

```
> Evaluate(%);
```

$$640\,x^4-960\,x^2$$

Цей приклад демонструє застосування диференціального оператора $x\frac{d^2}{dx^2}$ до поліному Ерміта п’ятого степеня та обчислення результату у вигляді степеневої функції.

```
> S:=(Create({1/n!,n=1..infinity},HermiteH(n,x)));
```

$$S := \sum_{n=1}^{\infty}\left(\frac{\mathrm{HermiteH}(n,x)}{n!}\right)$$

```
> SimplifyCoefficients(ApplyOperator([(x^2+2*x+1)*
*dx,[dx,x]],S),simplify);
```

$$11\,\mathrm{HermiteH}(1,x)+6\,\mathrm{HermiteH}(0,x)+\frac{17}{2}\,\mathrm{HermiteH}(2,x)$$
$$+\left(\sum_{n=3}^{\infty}\left(\frac{(n^2+7n+18)\,\mathrm{HermiteH}(n,x)}{2\,\Gamma(n+1)}\right)\right)$$

Цей приклад демонструє застосування оператора $(x^2+2x+1)\frac{d}{dx}$ до ряду ортогональних поліномів Ерміта. З подальшим застосуванням до результату функції ***SimplifyCoefficients***, за допомогою якої можна спростити коефіцієнти результуючого поліному, останній параметр функції ***SimplifyCoefficients simplify*** вказує на те, що до коефіцієнтів полінома застосовуються алгоритми приведення подібних та скорочення дробів.

Окрім застосування параметра ***simplify*** команда ***SimplifyCoefficient*** допускає використання параметрів ***expand, factor, normal.***

За допомогою функції Multiply(S1, S2) можна знайти добуток скінченного ряду ортогональних поліномів S1 та можливо нескінченного ряду ортогональних поліномів S2 та подати результат у базисі ортогональних поліномів функціонального ряду S2.

```
> P1:=10*x^7+2*x^5+6;
```

$$P1 := 10x^7 + 2x^5 + 6$$

```
> S1:=ChangeBasis(P1,ChebyshevT(n,x));
```

$$S1 := \frac{125}{32}\,ChebyshevT(3, x) + \frac{215}{32}\,ChebyshevT(1, x) + 6\,ChebyshevT(0, x) + \frac{39}{32}\,ChebyshevT(5, x) + \frac{5}{32}\,ChebyshevT(7, x)$$

```
> SimplifyCoefficients(Multiply(S1,S2),simplify);
```

$$\frac{136143}{4}\,HermiteH(1, x) + \frac{3427837}{320}\,HermiteH(5, x) + \frac{51421}{4}\,HermiteH(0, x) + \frac{5849093}{4480}\,HermiteH(7, x) + 44289\,HermiteH(2, x) + \frac{1775141}{48}\,HermiteH(3, x) + \frac{5919563}{1440}\,HermiteH(6, x) + \frac{725299}{32}\,HermiteH(4, x) + \frac{18944581}{53760}\,HermiteH(8, x) + \frac{237425099}{2903040}\,HermiteH(9, x) + \left(\sum_{n=10}^{\infty} \left(\frac{1}{128\,\Gamma(n+1)} ((5n^9 - 70n^8 + 1074n^7 - 5052n^6 + 37677n^5 - 22710n^4 + 380156n^3 + 676184n^2 + 1704272n + 1683296)\,HermiteH(n, x)) \right) \right)$$

За командою ***PolynomialMultiply(p, S)*** можна знайти добуток полінома p та функціонального ряду ортогональних поліномів S, а також подати результат у термінах ортогональних поліномів S.

```
> ss:=Create({[5=1]},HermiteH(n,x));
```

$$ss := HermiteH(5, x)$$

```
> PolynomialMultiply(x^2+2*x+1,ss);
```

$$\frac{13}{2}\,HermiteH(5,x)+\frac{1}{4}\,HermiteH(7,x)+20\,HermiteH(3,x)$$
$$+\,HermiteH(6,x)+10\,HermiteH(4,x)$$

За допомогою функції ***Add(S1, S2, a, b)*** можна знайти лінійну комбінацію функціональних рядів ортогональних поліномів S1,S2 з ваговими множниками *a*, *b*. Функціональні ряди S1,S2 мають бути записані за однаковими системами ортогональних поліномів.

```
> S:=(Create({1/n!,n=1..infinity},HermiteH(n,x)));
```

$$S := \sum_{n=1}^{\infty}\left(\frac{HermiteH(n,x)}{n!}\right)$$

```
> R1:=Create({[1=2,2=3,5=6]},HermiteH(n,x));
```

$$R1 := 2\,HermiteH(1,x)+6\,HermiteH(5,x)+3\,HermiteH(2,x)$$

```
> Add(S,R1,2,3);
```

$$6\,HermiteH(1,x)+18\,HermiteH(5,x)+9\,HermiteH(2,x)$$
$$+\left(\sum_{n=1}^{\infty}\left(\frac{2\,HermiteH(n,x)}{n!}\right)\right)$$

З іншими функціями пакета OrthogonalSeries можна ознайомитись, використовуючи довідку.

Циліндричні функції

Важливим класом спеціальних функцій є циліндричні функції дійсного аргументу або функції Бесселя, які визначаються як лінійно незалежні розв'язки звичайного диференціального рівняння другого порядку

$$U''+\frac{1}{z}U'+\left(1-\frac{\nu^2}{z^2}\right)U=0, \qquad (1)$$

де z — комплексна змінна; ν — комплексний параметр.

Перший лінійно незалежний розв'язок цього звичайного диференціального рівняння (функція Бесселя першого роду, ν-го порядку) визначається у вигляді степеневого ряду

$$J_\nu(z)=\sum_{k=0}^{\infty}\frac{(-1)^k\left(\frac{z}{2}\right)^{2k+\nu}}{k!\Gamma(k+\nu+1)},\ |z|<\infty,\ |\arg z<\pi|.$$

Другий лінійно незалежний розв'язок (функція Бесселя другого роду, ν-го порядку) для нецілих ν визначається у вигляді

$$Y_\nu(z)=\frac{J_\nu(z)\cos(\nu\pi)-J_{-\nu}(z)}{\sin(\nu\pi)},\ |z|<\infty,\ |\arg z|<\pi,$$

а для цілих $\nu = n$ у вигляді $Y_n(z) = \frac{1}{\pi}\left[\frac{\partial J_\nu(z)}{\partial \nu}\right]_{\nu=n} - (-1)^n\left[\frac{\partial J_{-\nu}(z)}{\partial \nu}\right]$. Таким чином, загальний розв'язок диференціального рівняння (1) можна записати у вигляді $U_\nu(z) = C_1 J_\nu(z) + C_2 Y_\nu(z)$.

```
>eq1_Cilindric:=diff(U(x),x$2)+1/x*diff(U(x),x)+
(1-nu^2/x^2)*U(x)=0;
```

$$eq1\_Cilindric := \left(\frac{d^2}{dx^2}\mathrm{U}(x)\right) + \frac{\frac{d}{dx}\mathrm{U}(x)}{x} + \left(1 - \frac{\nu^2}{x^2}\right)\mathrm{U}(x) = 0$$

```
> dsolve(eq1_Cilindric);
```

$$\mathrm{U}(x) = \_C1\ \mathrm{BesselJ}(\nu, x) + \_C2\ \mathrm{BesselY}(\nu, x)$$

```
> series(BesselJ(1/2,x),x,10);
```

$$\frac{\sqrt{2}\sqrt{x}}{\sqrt{\pi}} - \frac{\sqrt{2}\,x^{(5/2)}}{6\sqrt{\pi}} + \frac{\sqrt{2}\,x^{(9/2)}}{120\sqrt{\pi}} - \frac{\sqrt{2}\,x^{(13/2)}}{5040\sqrt{\pi}} + \frac{\sqrt{2}\,x^{(17/2)}}{362880\sqrt{\pi}} + \mathrm{O}(x^{(19/2)})$$

```
> series(BesselY(1/2,x),x,10);
```

$$-\frac{\sqrt{2}}{\sqrt{\pi}\sqrt{x}} + \frac{\sqrt{2}\,x^{(3/2)}}{2\sqrt{\pi}} - \frac{\sqrt{2}\,x^{(7/2)}}{24\sqrt{\pi}} + \frac{\sqrt{2}\,x^{(11/2)}}{720\sqrt{\pi}} - \frac{\sqrt{2}\,x^{(15/2)}}{40320\sqrt{\pi}} + \mathrm{O}(x^{(19/2)})$$

Приклад демонструє знаходження розв'язку звичайного диференціального рівняння у вигляді лінійної комбінації функцій Бесселя першого та другого роду і подання функцій Бесселя порядку *0,5* у вигляді степеневого ряду, зберігаючи члени не вище x^{10}.

У систему Maple вбудовані функції, за допомогою яких можна знаходити нулі функцій Бесселя першого та другого роду довільного порядку:

BesselJZeros(**v**, **n**), BesselJZeros(**v**, **n1**..**n2**), BesselYZeros(**v**, **n**), BesselYZeros(**v**, **n1**..**n2**)

Параметр ***n*** задає порядковий номер нуля функції Бесселя, параметри **n1**, **n2** — верхню та нижню межі порядкового номера нулів, параметр **v** — порядок функції Бесселя.

```
> BesselJZeros(1/2, 1..10);
```

$$\pi, 2\pi, 3\pi, 4\pi, 5\pi, 6\pi, 7\pi, 8\pi, 9\pi, 10\pi$$

```
> BesselJZeros(1/4, 1..10);
```

$$\mathrm{BesselJZeros}\left(\frac{1}{4}, 1 .. 10\right)$$

```
> evalf(BesselJZeros(1/4, 1..10));
```

2.780887724, 5.906142699, 9.042383664, 12.18134153, 15.32136983, 18.46192725, 21.60278445, 24.74382780, 27.88499460, 31.02624748

```
> BesselYZeros(1/2, 1..10);
```

$$\frac{\pi}{2}, \frac{3\pi}{2}, \frac{5\pi}{2}, \frac{7\pi}{2}, \frac{9\pi}{2}, \frac{11\pi}{2}, \frac{13\pi}{2}, \frac{15\pi}{2}, \frac{17\pi}{2}, \frac{19\pi}{2}$$

Другий клас циліндричних функцій — циліндричні функції уявного аргументу є лінійно незалежними розв'язками звичайного

диференціального рівняння другого порядку

$$U'' + \frac{1}{x}U' - \left(1 + \frac{\nu^2}{x^2}\right)U = 0,\ 0 < x < \infty. \qquad (2)$$

Останнє перейде у рівняння (1), якщо замінити незалежну змінну $z = ix$. Один з лінійно незалежних розв'язків рівняння (2) можна подати у вигляді степеневого ряду $I_\nu(z) = \sum_{k=0}^{\infty} \frac{\left(\frac{x}{2}\right)^{2k+\nu}}{k!\Gamma(k+\nu+1)}$, $0 < x < \infty$. Ця функція називається функцією Бесселя першого роду уявного аргументу ν-го порядку. Другий лінійно незалежний розв'язок рівняння (2) – функцію Бесселя другого роду уявного аргументу ν-го порядку (функцію Макдональда) для нецілих ν визначають у вигляді $K_\nu(x) = \pi\frac{I_{-\nu}(x) - I_\nu(x)}{2\sin(\nu\pi)}$, $\nu \neq n$, для цілих $\nu = n$ у вигляді $K_n(x) = \frac{(-1)^n}{2}\left[\left(\frac{\partial I_{-\nu}}{\partial \nu}\right)_{\nu=n} - \left(\frac{\partial I_\nu}{\partial \nu}\right)_{\nu=n}\right]$. Таким чином, загальний розв'язок рівняння (2) можна записати у вигляді $U_\nu(x) = C_1 I_\nu(x) + C_2 K_\nu(x)$.

```
>eq2_Cilindric:=diff(U(x),x$2)+1/x*diff(U(x),x)-
(1+nu^2/x^2)*U(x)=0;
```

$$eq2\_Cilindric := \left(\frac{d^2}{dx^2}\mathrm{U}(x)\right) + \frac{\frac{d}{dx}\mathrm{U}(x)}{x} - \left(1 + \frac{\nu^2}{x^2}\right)\mathrm{U}(x) = 0$$

```
> dsolve(eq2_Cilindric);
```

$$\mathrm{U}(x) = \_C1\ \mathrm{BesselI}(\nu, x) + \_C2\ \mathrm{BesselK}(\nu, x)$$

```
> series(BesselI(1/2,x),x,8);
```

$$\frac{\sqrt{2}\sqrt{x}}{\sqrt{\pi}} + \frac{\sqrt{2}\,x^{(5/2)}}{6\sqrt{\pi}} + \frac{\sqrt{2}\,x^{(9/2)}}{120\sqrt{\pi}} + \frac{\sqrt{2}\,x^{(13/2)}}{5040\sqrt{\pi}} + \mathrm{O}(x^{(15/2)})$$

```
>series(BesselK(1/2,x),x,8);
```

$$\frac{\sqrt{2}\sqrt{\pi}}{2\sqrt{x}} - \frac{\sqrt{2}\sqrt{\pi}\sqrt{x}}{2} + \frac{\sqrt{2}\sqrt{\pi}\,x^{(3/2)}}{4} - \frac{\sqrt{2}\sqrt{\pi}\,x^{(5/2)}}{12} + \frac{\sqrt{2}\sqrt{\pi}\,x^{(7/2)}}{48} - \frac{\sqrt{2}\sqrt{\pi}\,x^{(9/2)}}{240}$$
$$+ \frac{\sqrt{2}\sqrt{\pi}\,x^{(11/2)}}{1440} - \frac{\sqrt{2}\sqrt{\pi}\,x^{(13/2)}}{10080} + \mathrm{O}(x^{(15/2)})$$

Інші спеціальні функції

Ейлерові інтеграли першого та другого роду

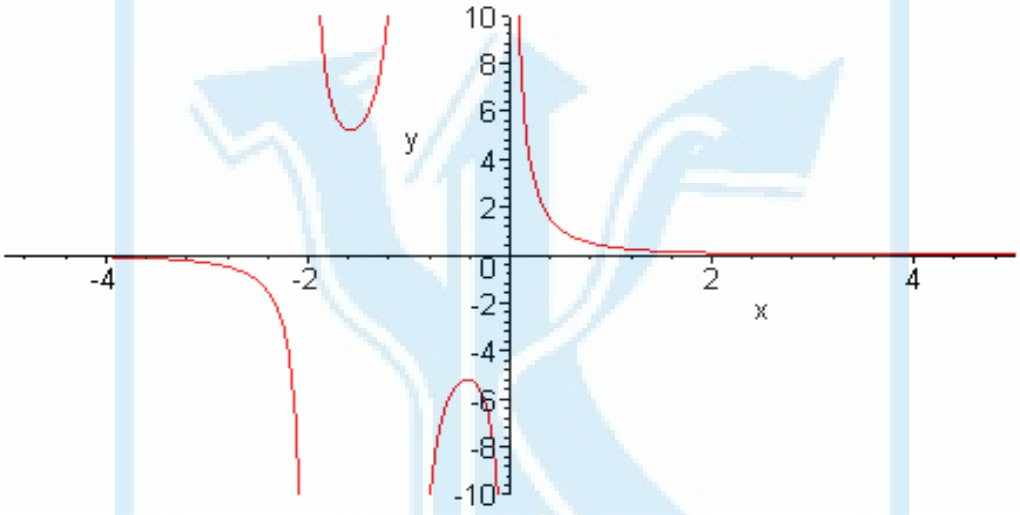
Інтеграли $B(x,y)=\int_0^1 t^{x-1}(1-t)^{y-1}dt$ та $\Gamma(x)=\int_0^\infty e^{-t}t^{x-1}dt$ називають ейлеровими інтегралами відповідно першого та другого роду. Крім того, поширенішою назвою цих функцій є Бета-функція та Гамма-функція.

Система Maple надає можливість звертатися до цих функцій для їх обчислення та застосування в різних виразах.

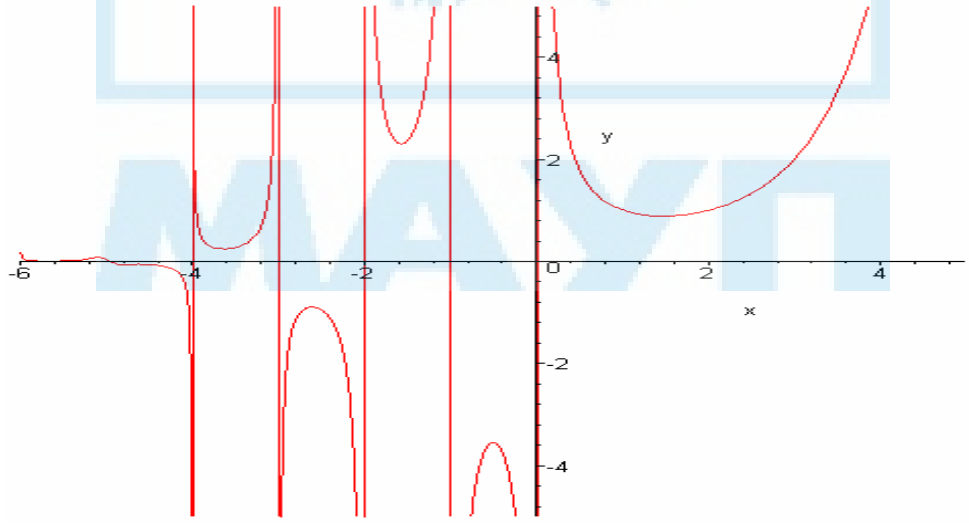
```
> Beta(2,5.2);
```
0.03101736973

```
> GAMMA(3.2);
```
2.423965480

```
> plot(Beta(x,3),x=-5..5,y=-10..10,discont = true);
```




```
> plot(GAMMA(x),x=-6..5,y=-5..5);
```




```
> convert(Beta(x,y),GAMMA);
```
$$\frac{\Gamma(x)\,\Gamma(y)}{\Gamma(y+x)}$$

Останній приклад демонструє формулу, що виражає Бета-функцію через Гамма-функцію.

Інтегральний синус та косинус, інтеграл вірогідності

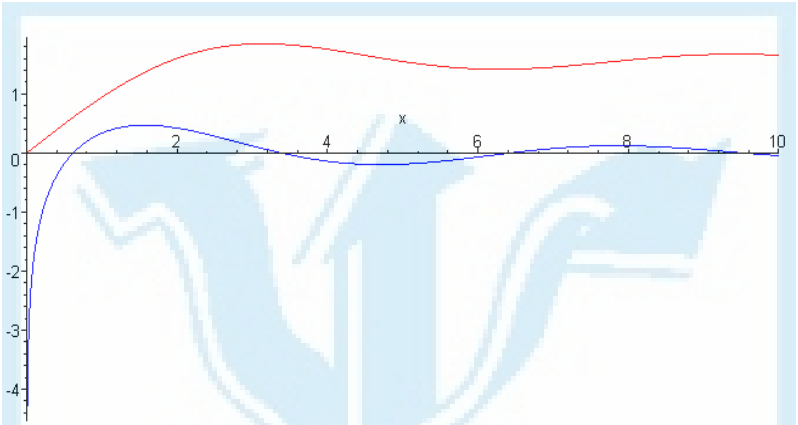
Інтегральними синусом та косинусом називають інтеграли відповідно

$$Si(x)=\int_0^x \frac{\sin(t)}{t}dt; \qquad Ci(x)=-\int_0^x \frac{\cos(t)}{t}dt.$$

```
> evalf(Si(Pi));
```
1.851937052

```
> evalf(Ci(Pi/2));
```
0.4720006514

```
> plot([Si(x),Ci(x)],x=0..10,color=[red,blue]);
```

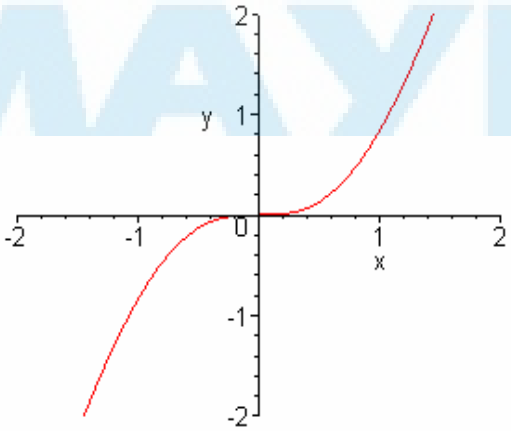


Інтегралом вірогідності в математиці називають функцію, яка при будь-якому комплексному значенні змінної z обчислюється за формулою $\Phi(z)=\frac{2}{\sqrt{\pi}}\int_0^z e^{-\frac{t^2}{2}}dt$.

У системі Maple цю функцію можна викликати за ім'ям erf(x).

```
> erf(3.)+erf(0.25);
```
1.276304300

```
> plot(x^2*erf(x),x=-2..2,y=-2..2);
```



```
> series(erf(x),x=0,10);
```

$$\frac{2}{\sqrt{\pi}}x-\frac{2}{3\sqrt{\pi}}x^3+\frac{1}{5\sqrt{\pi}}x^5-\frac{1}{21\sqrt{\pi}}x^7+\frac{1}{108\sqrt{\pi}}x^9+O(x^{10})$$

```
> diff(erf(x),x);
```

$$\frac{2\,e^{(-x^2)}}{\sqrt{\pi}}$$

Сферичні функції Лежандра першого та другого роду

Узагальненням ортогональних поліномів Лежандра $LegandreP(n,x)$, що були розглянуті раніше, є сферичні функції Лежандра першого та другого роду, які є двома лінійно незалежними розв'язками звичайного диференціального рівняння

$$(1-x^2)y''-2xy'+\nu(\nu+1)y=0,$$

де ν — числовий параметр.

Загальний розв'язок цього рівняння можна записати у вигляді $y(x)=c_1P_\nu(x)+c_2Q_\nu(x)$, де $P_\nu(x)$, $Q_\nu(x)$ — сферичні функції Лежандра відповідно першого та другого роду.

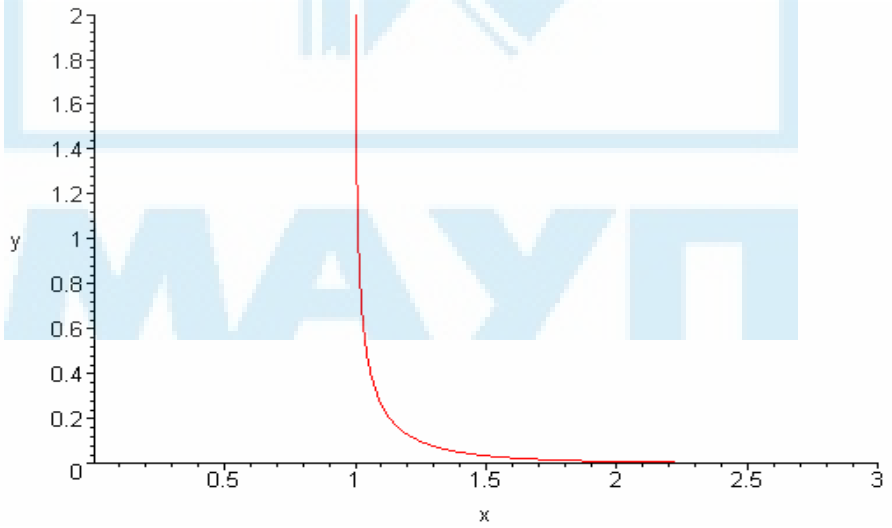
У системі Maple сферичні функції Лежандра можна викликати за їх іменами *LegandreP(ν,x)* та *LegandreQ(ν,x)*.

Імена сферичних функцій Лежандра можуть використовуватись у будь-яких виразах та операторах.

```
> evalf(LegendreQ(2.5,1.5));
```

0.03621096718

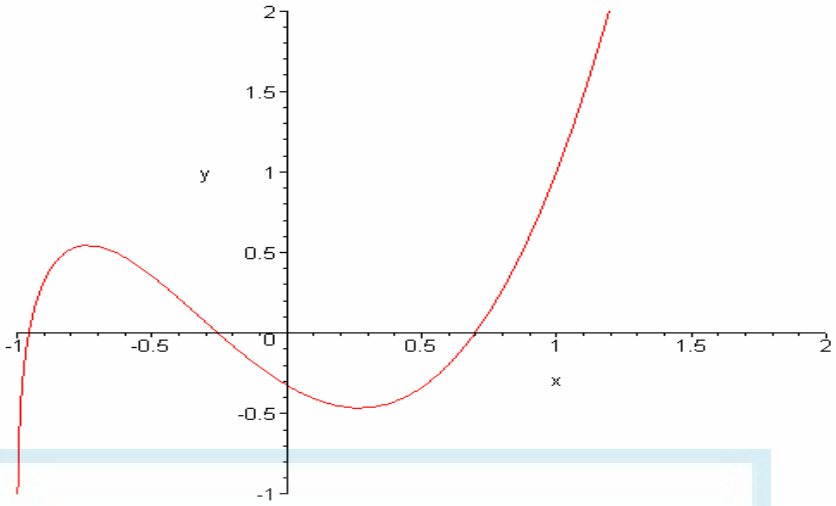
```
> plot(LegendreQ(2.5,x),x=0..3,y=0..2);
```




```
> evalf(LegendreP(2.5,1.5));
```

4.177619139

```
> plot(LegendreP(2.5,x),x=-1..2,y=-1..2);
```



Гіпергеометрична функція

Найзагальнішою трансцендентною функцією, за допомогою якої можна записати розв'язок звичайного диференціального рівняння другого порядку вигляду $\sigma(x)y'' + \tau(x)y' + \lambda y = 0$, де $\sigma(x)$, $\tau(x)$, λ — відповідно довільні поліноми не вище другого та першого ступеня і комплексний параметр є гіпергеометрична функція.

Гіпергеометричну функцію $F(\alpha,\beta,\gamma,x)$ можна визначити як обмежений розв'язок звичайного диференціального рівняння другого порядку $x(1-x)y'' + [\gamma - (\alpha+\beta+1)x]y' - \alpha\beta y = 0$ та подати у вигляді степеневого ряду $F(\alpha,\beta,\gamma,x) = \sum_{n=0}^{\infty} \frac{(\alpha)_n (\beta)_n}{(\gamma)_n} \frac{x^n}{n!}$, де $(\alpha)_n = \alpha(\alpha+1)...$ $...(\alpha+n-1)$.

У системі Maple гіпергеометричну функцію можна викликати за допомогою її імені *hypergeom*([α,β], [γ], x) та використовувати в будь-яких математичних виразах.

```
>eqhipergeom:=x*(x-1)*diff(U(x),x$2)+
+(2*x-1.3)*diff(U(x),x)+0.25*U(x)=0;
```

$$eqhipergeom := x\,(x-1)\left(\frac{d^2}{dx^2}\mathrm{U}(x)\right) + (2\,x-1.3)\left(\frac{d}{dx}\mathrm{U}(x)\right) + 0.25\,\mathrm{U}(x) = 0$$

```
> F:=unapply(op(2,dsolve([eqhipergeom,U(0)=1])),x);
```

$$F := x \to \mathrm{hypergeom}\left(\left[\frac{1}{2}, \frac{1}{2}\right], \left[\frac{13}{10}\right], x\right)$$

```
> evalf(F(1));
```

1.980805992

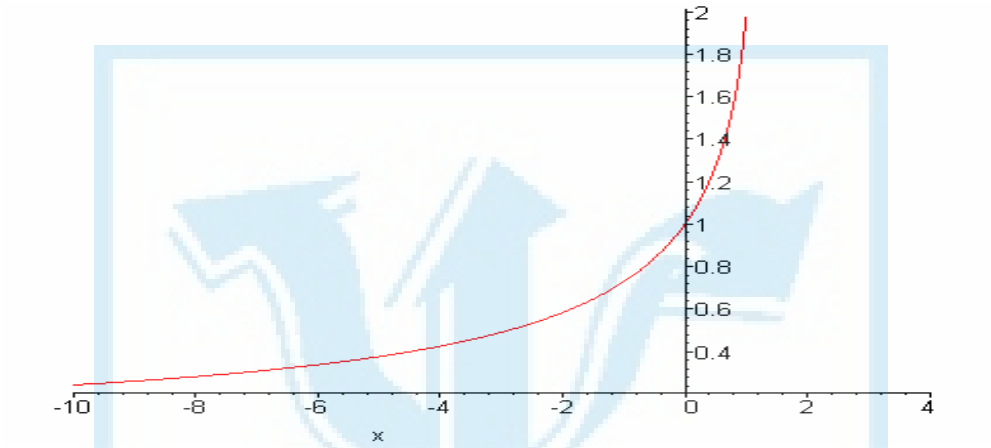
Приклад демонструє знаходження розв'язку звичайного диференціального рівняння другого порядку, його запис за допомогою гіпергеометричної функції та обчислення розв'язку в точці.

```
> series(hypergeom([a,b],[c],x),x=0);
```

$$1+\frac{a\,b}{c}x+\frac{a\,b\,(a+1)\,(b+1)}{2\,c\,(c+1)}x^2+\frac{a\,b\,(a+1)\,(b+1)\,(a+2)\,(b+2)}{6\,c\,(c+1)\,(c+2)}x^3+$$
$$\frac{a\,b\,(a+1)\,(b+1)\,(a+2)\,(b+2)\,(a+3)\,(b+3)}{24\,c\,(c+1)\,(c+2)\,(c+3)}x^4+$$
$$\frac{a\,b\,(a+1)\,(b+1)\,(a+2)\,(b+2)\,(a+3)\,(b+3)\,(a+4)\,(b+4)}{120\,c\,(c+1)\,(c+2)\,(c+3)\,(c+4)}x^5+O(x^6)$$

Приклад демонструє відрізок степеневого ряду, в який розкладається гіпергеометрична функція.

```
> plot(hypergeom([1,2],[5],x),x=-10..4);
```



Приклад демонструє побудову графіка гіпергеометричної функції.

```
>convert(LegendreP(2,x),hypergeom);
```
$$\text{hypergeom}\left([-2,3],[1],\frac{1}{2}-\frac{x}{2}\right)$$

```
> convert(BesselJ(3,x),hypergeom);
```
$$\frac{1}{48}x^3\,\text{hypergeom}\left([\,],[4],-\frac{x^2}{4}\right)$$

Приклад демонструє можливості команди *convert* системи Maple щодо перетворення різних спеціальних функцій через гіпергеометричну функцію.

Завдання для самостійного виконання I

1 Використовуючи оператори map або map2, без застосування оператора циклу обчислити.

1.1 $gradf(x,y,z,w)$ для функції чотирьох змінних.

1.2 $\sin(kx)+\dfrac{\cos(ky)}{2^k},\ k=1,3,5,7,9,11.$

1.3 $\displaystyle\int_{i\frac{\pi}{4}}^{3i\frac{\pi}{4}}\sin(x)dx,\ i=1...20.$

1.4 Коефіцієнти Фур'є функції x^3, $\frac{1}{\pi}\int_{-\pi}^{\pi} x^3 \sin(kx)dx,\ k=1...10.$

1.5 Похідні від $\sin(x), \cos(x), \mathrm{tg}(x), \mathrm{ctg}(x)$ у точці $\pi/4$.

2 Обчислити суму коренів ортогональних поліномів та побудувати графіки цих поліномів.

2.1 Чебишева 6-го степеня першого роду.

2.2 Чебишева 7-го степеня другого роду.

2.3 Ерміта 5-го степеня.

2.4 Лежандра 7-го степеня.

2.5 Лагерра 6-го степеня з параметром $\alpha = 1$.

3 Заданий поліном подати в базисі вказаних ортогональних поліномів та застосувати до результату диференціальний оператор *L*.

3.1 $5x^6 - 3x^2 + 2x - 1$ за поліномами Ерміта, $L = x^2 \frac{d}{dx} + x$.

3.2 $7x^8 + 4x^5 - x^3 + x$ за поліномами Лежандра, $L = x\ \frac{d^2}{dx^2} + \frac{d}{dx}$.

3.3 $5x^6 + 4x^5 - 6x^2 + 1$ за поліномами Чебишева другого роду, $L = x\frac{d^2}{dx^2} + x\frac{d}{dx}$.

3.4 $5x^6 - 3x^2 + 2x - 1$ за поліномами Лагерра з параметром, $\alpha = 2; L = x^2 \frac{d^2}{dx^2}$.

3.5 $7x^8 + 4x^5 - x^3 + x$ за поліномами Чебишева першого роду, $L = x^3 \frac{d^2}{dx^2}$.

4 Знайти два лінійно незалежних розв'язки звичайного диференціального рівняння та побудувати їх графіки на вказаному проміжку.

4.1 $\frac{d}{dx}\left((1-x^2)\frac{dy}{dx}\right) + \frac{4}{9}y = 0, \qquad x \in [0,1]$.

4.2 $U'' + \frac{1}{x}U' - \left(1 + \frac{4}{x^2}\right)U = 0, \qquad x \in [0,5]$.

4.3 $U'' + \frac{1}{z}U' + \left(1 - \frac{4}{z^2}\right)U = 0, \qquad x \in [0,5]$.

4.4 $x(1-x)U'' + (x-4)U' - 0,25U = 0, \qquad x \in [0,1]$.

4.5 $xU'' + (5/2 - x)U' - 2U = 0, \qquad x \in [0,1]$.

5 Розкласти задану функцію $f(x)$ на проміжку $[0, l]$ в узагальнений ряд Фур'є за функціями Бесселя $J_\nu\left(\frac{\mu_n}{l}x\right)$ виду $\sum_{n=1}^{\infty} a_k J_\nu\left(\frac{\mu_n}{l}x\right)$, де $a_k = \frac{\int_0^l f(x) J_\nu(\frac{\mu_n}{l}x) x dx}{\int_0^l (J_\nu(\frac{\mu_n}{l}x))^2 x dx}$; μ_n — n-й за номером додатний корінь функції Бесселя $J_\nu(x)$. Знайти перші N членів ряду. Вивести графіки N-часткової суми ряду Фур'є та вказаної функції.

5.1 $f(x) = \sin\left(\frac{\pi x}{l}\right)x^3,\ \nu = 3,\ N = 5, l = 2$.

5.2 $f(x) = \cos\left(\frac{\pi x}{l}\right)x^2,\ \nu = 1,\ N = 6, l = 4$.

5.3 $f(x) = \ln\left(1 + \frac{x}{l}\right)x^3,\ \nu = 4,\ N = 4, l = 2$.

5.4 $f(x) = \mathrm{tg}\left(\frac{\pi x}{4l}\right)(x-l)^2,\ \nu = 3,\ N = 5, l = 2$.

5.5 $f(x) = \ln\left(1 + \left(\frac{x}{l}\right)^3\right)(x-l),\ \nu = 2,\ N = 5, l = 1$.

6 Знайти обмежений розв'язок звичайного диференціального рівняння $\frac{d}{dx}\left((1-x^2)\frac{dy}{dx}\right) + n(n+1)y = 0,\ -1 < x < 1,\ y(1) < \infty$ і такий, що $y(1) = 1$.

6.1 $n = 6$.
6.2 $n = 8$.
6.3 $n = 7$.
6.4 $n = 5$.
6.5 $n = 10$.

7 Знайти розв'язок диференціального рівняння

$$\frac{d}{dx}\left((1-x^2)\frac{dy}{dx}\right) + \left(\eta(\eta+1) - \frac{\mu^{2.}}{(1-x^2)}\right)y = 0,\ -1 < x < 1.$$

7.1 $\eta = \frac{5}{3},\ \mu = \frac{7}{4},\ y(0) = 1,\ \frac{dy(0)}{dx} = 0.$

7.2 $\eta = \frac{7}{2},\ \mu = \frac{5}{4},\ y(0) = 0,\ \frac{dy(0)}{dx} = 1.$

7.3 $\eta = \frac{7}{2},\ \mu = \frac{6}{7},\ y(0,5) = 0,\ \frac{dy(0,5)}{dx} = 0,5.$

7.4 $\eta = \frac{9}{2},\ \mu = \frac{11}{7},\ y(0,3) = 0,\ \frac{dy(0,3)}{dx} = 0,9.$

7.5 $\eta = \frac{13}{6},\ \mu = \frac{12}{7},\ y(0,4) = 0,\ \frac{dy(0,4)}{dx} = 0,7.$

8 Знайти розв'язок диференціального рівняння $\frac{d}{dx}\left(e^{-x^2}\frac{dy}{dx}\right) + 2ne^{-x^2}y(x) = 0$ такий, що при $x \to \infty$ зростає не швидше x^s, де s — довільне додатне число. Порівняти цей розв'язок з поліномами Ерміта.

8.1 $n = 4.$

8.2 $n = 6.$

8.3 $n = 7.$

8.4 $n = 5.$

8.5 $n = 9.$

9 Знайти розв'язок диференціального рівняння $\frac{d}{dx}\left(xe^{-x}\frac{dy}{dx}\right) + ne^{-x}y(x) = 0$ такий, що при $x \to \infty$ зростає не швидше x^s, де s — довільне додатне число, та задовольняє умову $y(0) < \infty$.

9.1 $n = 6.$

9.2 $n = 4.$

9.3 $n = 5.$

9.4 $n = 9.$

9.5 $n = 7.$

10 Використовуючи твірну функцію $\Phi(x,t)$ для відповідного класу ортогональних поліномів, подати її у вигляді степеневого ряду за змінною t виду $\Phi(x,t) = \sum_{n=0}^{\infty} t^n \mathrm{P}_n(x)$. Перевірити, що коефіцієнт при t^n збігається з відповідним ортогональним поліномом n степеня змінної x.

10.1 Обчислити $\mathrm{P}_{10}(x)$ для твірної функції $\Phi(x,t)=\dfrac{1}{\sqrt{1-2tx+t^2}}$ поліномів Лежандра $\mathrm{P}_n(x)$.

10.2 Обчислити $L_{12}^2(x)$ для твірної функції $\Phi(x,t)=(1-t)^{-3}e^{-\frac{xt}{1-t}}$ поліномів Лагерра $L_n^2(x)$.

10.3 Обчислити $H_9(x)$ для твірної функції $\Phi(x,t)=e^{-2xt-t^2}$ поліномів Ерміта $H_n(x)$.

10.4 Обчислити $T_9(x)$ для твірної функції $\Phi(x,t)=(1-2xt+t^2)^{-\frac{1}{2}}(1-x+(1-2xt+t^2)^{\frac{1}{2}})^{\frac{1}{2}}(1+x+(1-2xt+t^2)^{\frac{1}{2}})^{\frac{1}{2}}$ поліномів Чебишева $T_n(x)$.

10.5 Обчислити $U_9(x)$ для твірної функції $\Phi(x,t)=(1-2xt+t^2)^{-\frac{1}{2}}(1-x+(1-2xt+t^2)^{\frac{1}{2}})^{\frac{1}{2}}(1+x+(1-2xt+t^2)^{\frac{1}{2}})^{\frac{1}{2}}$ поліномів Чебишева $U_n(x)$.

11 Використовуючи ваговий скалярний добуток $(\varphi,\psi)_\rho=\int_a^b\varphi(x)\psi(x)\rho(x)dx$ та таблицю для основних характеристик найпоширеніших ортогональних поліномів Якобі, Лагерра, Ерміта, розвинути задану функцію на відповідному відрізку в узагальнений ряд Фур'є $\sum_{n=1}^{\infty}(f,\Pi_n)_\rho\dfrac{\Pi_n(x)}{(\Pi_n,\Pi_n)_\rho}$ за вказаною системою ортогональних поліномів $\Pi_n(x)$. Обчислити 6 членів ряду Фур'є.

11.1 $f(x)=x^2\sin(\pi x)$ на проміжку $[-1,1]$ за поліномами Чебишева першого роду.

11.2 $f(x)=x^3\cos(\pi x)$ на проміжку $[-1,1]$ за поліномами Чебишева другого роду.

11.3 $f(x)=\operatorname{arctg}(x)$ на проміжку $[-1,1]$ за поліномами Лежандра.

11.4 $f(x)=\operatorname{arctg}(x^2)$ на проміжку $[-1,1]$ за поліномами Гегенбауера, $\lambda=-1$.

11.5 $f(x)=\ln(1+x^2)$ на проміжку $[0,\infty]$ за поліномами Лагерра, $\lambda=-2$.

Наближення функцій

Проблема наближення функцій полягає в заміні складної функції, яка важко обчислюється або задана в окремих точках, на достатньо просту функцію; при цьому норма різниці між цими функціями повинна бути щонайменша.

Поліноміальна інтерполяція функцій

Якщо деяка функція задана таблично, тобто у вигляді масивів X, Y однакової розмірності $\{X(i), Y(i),\ i = 1, N\}$, то існує можливість побудови аналітичної функції y вигляді полінома n – 1 степеня. У системі Maple існує активна форма вбудованої функції interp(X,Y,x) або пасивна форма цієї функції Interp(X,Y,x). Параметри X,Y — списки, які задають масив значень аргументу та масив значень функції, x — ім'я змінної, яка вказує на аргумент інтерполяційного полінома.

$$f1 := x \rightarrow \frac{2x+1}{(x^2+2x+1)^2}$$

```
> f1:=x->(2*x+1)/((x^2+2*x+1)^2);

spx:=evalf([seq(i*Pi/4/10,i=0..10)]);
```

$$spx := [0.,\ 0.07853981635,\ 0.1570796327,\ 0.2356194490,\ 0.3141592654,\ 0.3926990818,\ 0.4712388981,\ 0.5497787144,\ 0.6283185308,\ 0.7068583472,\ 0.7853981635]$$

```
spy:=map(f1,spx);
```

$$spy := [1.000000000,\ 0.8551031742,\ 0.7331536571,\ 0.6311669592,\ 0.5459430729,\ 0.4745761698,\ 0.4145942442,\ 0.3639560905,\ 0.3209994255,\ 0.2843782471,\ 0.2530039700]$$

```
> Int_pol:=x->interp(spx,spy,x);
```

$$Int\_pol := x \rightarrow interp(spx, spy, x)$$

```
> Int_pol(x);
```

$$-1.410129845\,x^{10} + 6.870951225\,x^{9} - 2.000073559\,x - 15.19170886\,x^{8} + 2.002850952\,x^{2} + 20.29164809\,x^{7} - 0.0461013942\,x^{3} - 18.31379063\,x^{6} - 4.580301922\,x^{4} + 11.56293290\,x^{5} + 1.000000000$$

Приклад демонструє побудову інтерполяційного полінома 10-го степеня для функції, заданої аналітично на інтервалі $[0, \frac{\pi}{4}]$. У змінних *spx, spy* обчислюються масиви значень аргументу та функції. Змінна ***Int_pol*** призначена для задавання інтерполяційного полінома, вигляд якого демонструє останній оператор прикладу.

Сплайнова інтерполяція

Сплайнова інтерполяція призначена для наближення таблично заданої функції у вигляді кусково-поліноміальної функції на кожному з елементарних інтервалів. У граничних точках повинні виконуватись певні умови гладкості сплайну залежно від степеня поліномів. У системі Maple для проведення сплайн апроксимації використовується вбудована функція ***spline(X,Y,x,d)***, де ***X,Y,x*** мають такий самий зміст, що й у попередній функції. Параметр *d* задає порядок сплайну і може приймати цифрові значення 1, 2, 3, 4 або символьні значення linear, quadratic, cubic, quartic, що відповідає сплайнам лінійним, квадратичним, кубічним або четвертого порядку.

```
> Sp_1:=x->spline(spx,spy,x,1);
```

$$Sp\_1 := x \to spline(spx, spy, x, 1)$$

```
> map(Sp_1,[0.1,0.2,0.3,0.4]);
```

$$[0.8217817435, 0.6774200590, 0.5613073513, 0.4690003593]$$

```
> Sp_3:=x->spline(spx,spy,x,3);
```

$$Sp\_3 := x \to spline(spx, spy, x, 3)$$

```
> Sp_1(x);
```

$$\begin{cases} 1.000000000 - 1.844883685\,x & x < 0.07853981635 \\ 0.9770526911 - 1.552709476\,x & x < 0.1570796327 \\ 0.9371270528 - 1.298534969\,x & x < 0.2356194490 \\ 0.8868386176 - 1.085104221\,x & x < 0.3141592654 \\ 0.8314106843 - 0.9086716314\,x & x < 0.3926990818 \\ 0.7744857980 - 0.7637135968\,x & x < 0.4712388981 \\ 0.7184231675 - 0.6447449980\,x & x < 0.5497787144 \\ 0.6646527455 - 0.5469412455\,x & x < 0.6283185308 \\ 0.6139688531 - 0.4662753248\,x & x < 0.7068583472 \\ 0.5667467407 - 0.3994697031\,x & otherwise \end{cases}$$

Приклад демонструє наближення функції з попереднього прикладу за допомогою лінійних та кубічних сплайнів і подання лінійного сплайну в явному вигляді, тобто у вигляді кусково-лінійної функції.

Дробово-раціональне рівномірне наближення функцій за алгоритмом Ремеза

У пакеті ***numapprox*** існує функція remez(w, f, a, b, m, n, crit, ‘max-error’), призначена для побудови рівномірного наближення функції у дробово-раціональному вигляді. Параметри цієї функції мають такий зміст:

w-функція аргументу x, $w(x) > 0$, $x \in [a,b]$, яка використовується для обчислення відхилення між функціями $f(x)$ та дробово-раціональ-

ною функцією $r(x)$ у вигляді $\max\limits_{x\in[a,b]} w(x)\left|(f(x)-r(x)\right|$. Найпоширенішими частинними випадками функції $w(x)$ є функція $w(x)=1$ для абсолютної похибки та $w(x)=\dfrac{1}{|f(x)|}$ для відносної похибки;

f – наближена функція;

a, b – відрізок, на якому здійснюється наближення;

m, n – степені поліномів чисельника та знаменника;

crit – масив точок розмірності $m + n + 2$, що належать інтервалу [a,b], в яких реалізується max/min похибки;

'maxerror' – ім'я змінної, якій присвоюється значення найбільшого відхилення на відрізку [a,b].

```
>w:=x->1;f:=x->exp(x);
crit:=array(1..6,[0,0.5,0.7,0.8,1.2,1.5]);
```

$$w := x \to 1 \quad f := x \to e^x \quad crit := [0, 0.5, 0.7, 0.8, 1.2, 1.5]$$

```
> ff:=remez(w,f,0,2,3,1,crit,'maxerror');
```

$$ff := x \to \frac{1.241133455 + (1.009150365 + (0.3493997146 + 0.1183993828\,x)\,x)\,x}{1.241587759 - 0.2415877588\,x}$$

```
> maxerror;
```

$$0.0003659247886$$

Приклад демонструє наближення функції e^x на відрізку $[0,2]$ у вигляді відношення поліномів третього та першого степеня.

Раціональна інтерполяція

У пакеті CurveFitting існує функція ***RationalInterpolation(****xdata****,*** *ydata****,*** *x****,*** *opts****)***, за допомогою якої можна побудувати дробово-раціональну інтерполюючу функцію. Призначення параметрів функції CurveFitting:

xdata – список раціональних значень аргументів розмірності $n + 1$;

ydata – список раціональних значень функцій розмірності $n + 1$;

x – незалежна змінна інтерполюючої функції або числове значення, в якому обчислюється інтерполююча функція;

opts – параметр, який може мати вигляд *degrees=[d1,d2]*, де $n \ge d1 + d2$.

```
> f2:=x->(2*x+1)/sin((x^2+2*x+1)^2);
```

$$f2 := x \to \frac{2\,x+1}{\sin((x^2+2\,x+1)^2)}$$

```
> spx:=convert(evalf([seq(i*Pi/4/10,i=0..10)]),rational);
```

$$spx := \left[0, \frac{13008}{165623}, \frac{26087}{166075}, \frac{12866}{54605}, \frac{52174}{166075}, \frac{13221}{33667}, \frac{25945}{55057}, \frac{26087}{47450}, \frac{104348}{166075}, \frac{13079}{18503}, \frac{26087}{33215}\right]$$

```
> spy:=convert(evalf(map(f2,spx)),rational);
```

$$spy := \left[\frac{46717}{39311}, \frac{12783}{10787}, \frac{73037}{54217}, \frac{51709}{25471}, \frac{106999}{10405}, \frac{-53285}{17353}, \frac{-111139}{57194}, \frac{-181613}{42565}, \frac{101403}{30526}, \frac{59376}{19823}, \frac{-47120}{12309}\right]$$

```
> with(CurveFitting):
evalf(RationalInterpolation(spx,spy,0.4,degrees=[5,4]));
                    -2.793889501
> evalf(f2(0.4));      -2.794062219
```

Приклад демонструє обчислення наближеного значення функції $f_2(x)$ за допомогою дробово-раціональної інтерполюючої функції в точці $x = 0,4$ та порівняння цього наближеного значення з точним значенням функції в цій точці.

Середньоквадратичне наближення за методом найменших квадратів

У пакеті ***CurveFitting*** існує функція ***LeastSquares***, за допомогою якої можна встановити функцію заданого вигляду, яка лінійно залежить від деякого набору параметрів і має найменше середньоквадратичне відхилення на заданій множині точок. Функція ***LeastSquares(xdate, ydate, x, curve = f, weight = s)*** має такий зміст параметрів:

xdate, ydate — відповідно списки або масиви однакової розмірності точок незалежної змінної та значення функції;

x — незалежна змінна, відносно якої записується функція найкращого середньоквадратичного наближення;

f — функція аргументу *x*, що містить деякий набір параметрів;

s — список або масив вагових множників однакової розмірності з ***xdate, ydate***, який враховує точність вимірювання значення функції (необов'язковий параметр).

Приклад демонструє побудову найкращого полінома середньоквадратичного наближення 5-го степеня у вигляді лінійної комбінації поліномів Ерміта. Значення параметрів ***spx, spy*** узяті з прикладу підрозділу "Поліноміальна інтерполяція функцій".

```
> Hermit_5:=sum(a[n]*HermiteH(n,x),n=0..5);
```

$$Hermit\_5 := a_0\, HermiteH(0, x) + a_1\, HermiteH(1, x) + a_2\, HermiteH(2, x) + a_3\, HermiteH(3, x) + a_4\, HermiteH(4, x) + a_5\, HermiteH(5, x)$$

```
> LeastSquares(spx,spy,x,curve=Hermit_5);
```

$$2.2766309503509769\, HermiteH(0, x) - 1.8457018203944495\, HermiteH(1, x) + 0.7334758824044491\, HermiteH(2, x) - 0.11880686121719048\, HermiteH(3, x) + 0.015861962695508713\, HermiteH(4, x) + 0.0021398328239113058\, HermiteH(5, x)$$

Чисельне інтегрування

Система Maple прагне виконати аналітичне обчислення заданого інтеграла навіть тоді, коли значення цього інтеграла виражається через спеціальні функції. Якщо визначений інтеграл неможливо обчислити точно, використовуючи елементарні і спеціальні функції, система Maple надає користувачу можливість виконати наближене (чисельне) обчислення відповідного інтеграла. При застосуванні чисельних процедур інтегрування головна процедура ***int( )*** або ***Int( )*** стає параметром процедури ***evalf( )***. При цьому використовують таке правило. Якщо параметром процедури ***evalf( )*** є процедура ***int( )***, спочатку робиться спроба виконати точне аналітичне обчислення, якщо воно неможливе, виконується чисельне обчислення. Якщо ж використовується неактивна форма процедури інтегрування ***Int( ),*** одразу відбувається наближене обчислення відповідного інтеграла.

Процедура обчислення інтеграла ***int( )*** або ***Int( )*** окрім стандартних параметрів, які використовуються при аналітичному обчисленні визначених інтегралів ***int(f(x), x = a..b)***, використовує додаткові необов'язкові параметри ***int(f(x), x = a..b, необов'язкові параметри).***

Необов'язкові параметри процедури ***int*** мають такий вигляд:

digits = n	n — ціле додатне число, яке задає кількість правильних цифр у результаті обчислення інтеграла;
epsilon = ε	ε — дійсне число, яке задає відносну похибку обчислення результату. Ці два параметри взаємопов'язані співвідношенням $\varepsilon = 0.5 \cdot 10^{1-n}$;
method = назва методу	назва методу приймає значення назви методу наближеного обчислення інтеграла; назва методу може вказуватися і без посилання на значення параметра method;

method = _CCquad — Clenshaw-Curtis квадратурний метод;

method = _Dexp — адаптивний метод подвійного показника;

method = _Gquad — адаптивний метод Гаусса;

method = _Sinc — адаптивний ***sinc*** метод квадратур;

method = _NCrule — адаптивний метод Ньютона – Котеса **"*quanc8*"**;

method = _d01ajc — адаптивний 10-точковий метод Гаусса, що застосовується для скінченних інтервалів інтегрування;

method = _d01akc — адаптивний 30-точковий метод Гаусса, що застосовується для скінченних інтервалів інтегрування з осцилюючими підінтегральними функціями;

method = _d01amc — метод обчислення інтегралів з нескінченними межами інтегрування.

При чисельному обчисленні багатовимірних інтегралів у багатовимірних паралелепіпедах можна використовувати спеціальну форму процедури ***Int( )*** у поєднанні з процедурою *evalf(): evalf(Int(f, [x1= = a1..b1, x2=a2..b2, ..., xn = an..bn])).*

У цьому випадку система Maple надає можливість використовувати чисельні методи обчислення багатовимірних інтегралів:

method = _cuhre — метод обчислення багатовимірних інтегралів у паралелепіпедах скінченного розміру розмірністю від 2 до 15, що базується на алгоритмі ACM TOMS Algorithm 698;

method = _метод Монте–Карло — метод обчислення багатовимірних інтегралів. Цей метод дає невисоку відносну похибку обчислення.

Для чисельного обчислення багатовимірних інтегралів можна також використовувати стандартний підхід зведення багатовимірних інтегралів до повторних.

Наведемо приклади чисельного обчислення одновимірних інтегралів:

```
> evalf(Int( exp(-x^5)/(x^3+1),
x = 0..1 ));
```

0.7543143320

```
> evalf(Int( sin(x)/(x^2+1),
x = 0..infinity,digits=20));
```

0.64676112277913007155

```
> evalf(Int( sin(x)*ln(x+1)*
exp(-x^3), x = 0..infinity));
```

0.2090363674

```
> e1 := 1/GAMMA(x)^2:
Int(e1, x = 0..2) =
evalf(Int(e1, x=0..2, digits=14,
method=_d01ajc));
```

$$\int_0^2 \frac{1}{\Gamma(x)^2}\,dx = 1.5630430689991$$

Приклад демонструє обчислення багатовимірного інтеграла методом Монте – Карло.

```
> f:=(x,y,z,u)-> piecewise(x^2+y^2+z^2+u^2<
1,1/(x^3+y^3+z^3+u^3+1),0);
```

$$f := (x, y, z, u) \to \text{piecewise}\left(x^2+y^2+z^2+u^2<1, \frac{1}{x^3+y^3+z^3+u^3+1}, 0\right)$$

```
> evalf(Int(f(x,y,z,u),[x=-1..1, y=-1..1, z=-1..1,
u=-1..1],method=_MonteCarlo, digits=3));
```

5.96

Завдання для самостійного виконання II

1 Побудувати інтерполяційний багаточлен Лагранжа $L_n(x)$ для функції:

1.1 $Erf(x)=\frac{2}{\sqrt{\pi}}\int_0^x e^{-z^2}dz$ на проміжку [0.45;0.50] за рівновіддаленими вузлами n = 5, n = 10. Обчислити $L_5(0.4555)$, $L_5(0.4756)$.

1.2 $f(x)=\frac{1}{1+40x^2}, -1\le x\le 1$ у точках: а) рівновіддалених вузлів; б) вузлів, що є нулями полінома Чебишева першого роду, $n=5, n=10$. Побудувати графік функції $f(x)$ та багаточлена $L_n(x)$.

1.3 $f(x)=\frac{\sin^2(\pi x)e^x}{x}, 0<x\le 2, f(0)=0$ за системою рівновіддалених вузлів, $n=11, n=21$. Обчислити значення інтерполяційного багаточлена в точках, що лежать посередині між вузлами інтерполяції, порівняти з точним значенням функції.

1.4 $f(x)=Sinc(5x), -2\le x\le 2$, де $Sinc(x)=\begin{cases}\frac{\sin x}{x}, & x\neq 0\\ 1, & x=0\end{cases}$ за системою рівновіддалених вузлів, $n=11, n=21$. Обчислити значення інтерполяційного багаточлена в точках, що лежать посередині між вузлами інтерполяції, порівняти з точним значенням функції.

1.5 $f(x)=e^{-\omega^2x^2}, -1\le x\le 1, \omega=\overline{1,4}$ за системою вузлів, що є нулями полінома Чебишева першого роду, $n=11, n=20$. Побудувати графік функції $f(x)$ та багаточлена $L_n(x)$.

2 Написати процедуру побудови таблиці поділених різниць $f(x_0,x_1,...,x_k),\ k=\overline{1..n}$ вигляду

$$\begin{array}{c|cccc} x_0 & f(x_0) & & & \\ x_1 & f(x_1) & f(x_0,x_1) & & \\ x_2 & f(x_2) & f(x_1,x_2) & f(x_0,x_1,x_2) & \\ \vdots & \vdots & \vdots & \vdots & \\ x_n & f(x_n) & f(x_{n-1},x_n) & f(x_{n-2},x_{n-1},x_n) & \ldots f(x_0,x_1,...,x_n) \end{array}$$

де задана таблиця:

x	x_0	x_1	...	x_n
$f(x)$	$f(x_0)$	$f(x_1)$	...	$f(x_n)$

Врахувати можливу кратність вузлів, приймаючи

$$f(x_i,x_i)=f'(x_i),\ f(x_i,x_i,x_i)=\frac{f''(x_i)}{2!},\ f\underbrace{(x_i,\ldots,x_i)}_{k+1}=\frac{f^k(x_i)}{k!}$$ і т. д.

3 Використовуючи попередню процедуру обчислення поділених різниць, записати процедуру побудови інтерполяційного полінома Ерміта з кратними вузлами інтерполяції.

$H_n(x)=f_0+(x-x_0)f(x_0,x_1)+(x-x_0)(x-x_1)f(x_0,x_1\ x_2)+\ldots$

$\ldots+(x-x_0)\cdots(x-x_{n-1})f(x_0,x_1,\ldots,x_n).$

Побудувати інтерполяційний багаточлен за наступною таблицею

3.1 $x_0=-1: f_0=0,\ f_0'=2;$
$x_1=\ 0: f_1=2,\ f_1'=4, f_1''=-4;$
$x_2=\ 1: f_2=-1,\ f_2'=-14.$

3.2 $x_0=0: f_0=1,\ f_0'=0, f_0''=-2;$
$x_1=\ 0.3: f_1=0.9140,\ f_1'=-0.5484.$

3.3 $x_0=-1: f_0=0,\ f_0'=2;$
$x_1=\ 0: f_1=2,\ f_1'=4, f_1''=-4;$
$x_2=\ 1: f_2=-1,\ f_2'=-14.$

3.4 $x_0=-1: y_0=1, y_0'=2;$
$x_1=0:\ \ y_1=1, y_1'=3; y_1''=4;$
$x_2=1: y_2=2.$

3.5 $x_0=0: y_0=0, y_0'=1;$
$x_1=1:\ \ y_1=2, y_1'=9, y_1''=44;$
$x_2=2: y_2=82.$

4 Побудувати графіки лінійного та кубічного сплайнів, що інтерполює значення заданої функції $f(x), a\le x\le b$ на рівномірній сітці $x_i=a+ih, i=\overline{0,n}, h=\frac{b-a}{n}$ для значень $n=25,32$. Варіанти функцій:

4.1 $f(x)=\frac{1}{1+40x^2}, -1\le x\le 1.$

4.2 $f(x)=\begin{cases}0, -1\le x\le 0,\\ x^k,\ 0\le x\le 1.\end{cases}\quad k=\overline{1,4}.$

4.3 $f(x)=1-2^k(x-0,5)^k, 0\le x\le 1, k=2,4,6.$

4.4 $f(x)=|x|, x\in[-1,1].$

4.5 $f(x) = e^{-\omega^2 x^2}$, $-1 \le x \le 1$, $\omega = \overline{1,4}$.

5 Побудувати рівномірне наближення до заданої функції у вигляді полінома першого степеня.

5.1 $f(x) = \text{arctg}x$, $x \in [0,1]$.

5.2 $f(x) = \sqrt[3]{x}$, $x \in [0,1]$.

5.3 $f(x) = \sin(x), x \in [0, \frac{\pi}{2}]$.

5.4 $f(x) = x\sin(x)$, $x \in [-\pi, \pi]$.

5.5 $f(x) = \sqrt{x}$, $x \in [0,1]$.

6 Побудувати багаточлен найкращого рівномірного наближення третього степеня та знайти відхилення для функції, побудувати графіки.

6.1 $f(x) = 3x^4 - 2x^2 + x + 1$, $x \in [0,1]$.

6.2 б) $f(x) = x^4 - 2$, $x \in [0,1]$.

6.3 $f(x) = 2x^4$, $x \in [-1,1]$.

6.4 $f(x) = 5x^4 + x - 4$, $x \in [2,5]$.

6.5 $f(x) = x^4$, $x \in [0,1]$.

7 На заданій множині точок, використовуючи рівномірну сітку, на відрізку $[-1, 1]$ побудувати багаточлен найкращого середньоквадратичного наближення за вказаною системою ортогональних поліномів.

7.1 $f(x) = x^2 \sin(\pi x)$, $x_i = -1 + ih$, $i = \overline{0..N}$, $h = \frac{2}{N}$, за поліномами Чебишева першого роду.

7.2 $f(x) = x\cos(\pi x)$, $x_i = -1 + ih$, $i = \overline{0..N}$, $h = \frac{2}{N}$, за поліномами Чебишева другого роду.

7.3 $f(x) = x\text{arctg}(x)$, $x_i = -1 + ih$, $i = \overline{0..N}$, $h = \frac{2}{N}$, за поліномами Лежандра.

7.4 $f(x) = \text{arctg}(x^3)$, $x_i = -1 + ih$, $i = \overline{0..N}$, $h = \frac{2}{N}$, за поліномами Гегенбауера з $\lambda = \frac{3}{2}$.

7.5 $f(x)=x\mathrm{tg}(x),\quad x_i=-1+ih,\ i=\overline{0..N},\ h=\dfrac{2}{N}$, за поліномами Лежандра.

8 Використовуючи процедури чисельного інтегрування, експериментально вибрати метод для обчислення матриці та обчислити її з п'ятьма правильними знаками після десяткової крапки. $a_{i,j}=\int\limits_{-1}^{1}\int\limits_{-1}^{1}K(x,y)\varphi_i(y)\varphi_j(x)\rho(x)dx,\ i,j=\overline{1..n}$ для заданого ядра $K(x,y)$ заданої системи функцій $\varphi_i(x),\ i=\overline{1..n}$ та вагової функції $\rho(x)\geq 0$.

Варіант	$K(x,y)$	$\varphi_i(x)$	$\rho(x)$
8.1	$\sin\left(\sqrt{x^2+y^2}\right)$	$T_i(x)$	$\dfrac{1}{\sqrt{1-x^2}}$
8.2	$\ln\left(1+x^2y^2\right)$	$U_i(x)$	$\sqrt{1-x^2}$
8.3	$\mathrm{tg}\left(\dfrac{x^2+y^2}{2}\right)$	$P_i(x)$	1
8.4	$\mathrm{arctg}\left(1-x^2-y^2\right)$	$G_i^{(2)}(x)$	$(1-x^2)^{\frac{3}{2}}$
8.5	$\dfrac{xy}{(1+x^2y^2)}$	$T_i(x)$	$\dfrac{1}{\sqrt{1-x^2}}$

Чисельне розв'язування нелінійних рівнянь та їх систем

Для знаходження чисельного розв'язку нелінійних рівнянь та систем рівнянь система Maple використовує базову процедуру ***fsolve (****eqns****,*** *vars****,*** *options****)***:

eqns – одне рівняння або система рівнянь, що задається у вигляді множини;

vars — змінна або множина змінних, відносно яких здійснюється знаходження розв'язку;

options — набір додаткових необов'язкових опцій, які можуть включати значення, наведені в таблиці:

avoid	avoid ={список рівностей}. Список рівностей визначає ті значення змінної, які слід ігнорувати при пошуку розв'язків.
complex	Якщо ця опція вказана, то пошук розв'язків відбуватиметься на множині комплексних чисел.

fulldigits	Опція дає змогу підтримувати високу точність заокруглення при проміжних розрахунках.
maxsols	maxsols=n. Ця опція використовується при роботі з поліномами, *n* — кількість коренів, які необхідно обчислити. Як корені вибирають найменші;
interval	{x = a..b, y = c..d,…z = l..s}. Ця опція задає паралелепіпед, якому має належати розв'язок системи рівнянь.

Продемонструємо на прикладах можливості використання процедури *fsolve* для обчислення розв'язків систем рівнянь.

```
> eq1:={sin(2*x-y)-1.2*x=0.4,0.8*x^2+1.5*y^2=1};
```

$$eq1 := \{0.8\,x^2 + 1.5\,y^2 = 1,\ \sin(2\,x - y) - 1.2\,x = 0.4\}$$

```
> fsolve(eq1,{x,y},fulldigits);
```

$$\{x = 0.4912379505,\ y = -0.7334613013\}$$

```
> eq2:={3*x^2+3/2*y^2+z^2-5=0,  6*x*y*z-x+5*y+3*z=0,5*x*z-y*z-1=0};
```

$$eq2 := \{3\,x^2 + \frac{3\,y^2}{2} + z^2 - 5 = 0,\ 6\,x\,y\,z - x + 5\,y + 3\,z = 0,\ 5\,x\,z - y\,z - 1 = 0\}$$

```
> fsolve(eq2,{x,y,z});
```

$$\{x = -1.284457050,\ y = -0.1297565120,\ z = -0.1589186226\}$$

```
> fsolve(eq2,{x,y,z}, {x=1..1.5,y=-0.5..0.5,z=0..0.5},fulldigits);
```

$$\{z = 0.1589186226,\ x = 1.284457050,\ y = 0.1297565120\}$$

Продемонструємо приклад знаходження наближеного розв'язку для одного трансцендентного рівняння.

```
> eq3:=tan(x)=(x^2-1)/x;
```

$$eq3 := \tan(x) = \frac{x^2 - 1}{x}$$

```
> fsolve(eq3,x,{x=Pi/2..3*Pi/2});
```

4.481749781

Приклади демонструють знаходження коренів поліномів.

```
> with(orthopoly);
```

$$[G, H, L, P, T, U]$$

```
> eq5:=T(7,x);
```

$$eq5 := 64\,x^7 - 112\,x^5 + 56\,x^3 - 7\,x$$

```
> fsolve(eq5,x,maxsols=5);
```

-0.9749279122, -0.7818314825, -0.4338837391, 0., 0.4338837391

```
> eq4:=5*x^7+3*x^4-25;
```

$$eq4 := 5\,x^7 + 3\,x^4 - 25$$

```
> fsolve(eq4,x,maxsols=3);
```

1.206786284

```
> fsolve(eq4,x,maxsols=3,complex);
```

-1.166245256 – 0.5018463345 *I*, -1.166245256 + 0.5018463345 *I*,
-0.2334153741 – 1.202374214 *I*

Чисельне розв'язування задач Коші та граничних задач для звичайних диференціальних рівнянь

Чисельне розв'язування задач Коші для систем звичайних диференціальних рівнянь у системі Maple здійснюється за допомогою процедури ***dsolve (odesys, numeric, [method = namemethod], vars, options)***.

У цій процедурі параметр ***odesys*** задає у вигляді множини диференціальне рівняння або систему звичайних диференціальних рівнянь та додаткові умови (початкові або граничні).

Параметр ***numeric*** вказує, що задача Коші, або гранична задача, розв'язуватиметься чисельно.

Параметр ***namemethod*** приймає значення, що задають метод чисельного розв'язування задачі:

bvp — використовується при розв'язанні граничних задач і використовує метод трапецій bvp[trapdefer] або метод середньої точки bvp[middefer], або покращений метод трапецій bvp[traprich], або покращений метод середньої точки bvp[midrich];

classical — використовуються класичні методи Рунге – Кутта 1–4-го порядку точності;

rkf45 — використовується метод Рунге - Кутта - Фелберга 4–5-го порядку точності;

lsode — використовуються процедури для інтегрування жорстких систем диференціальних рівнянь на базі методу Адамса;

gear — використовується метод простої інтерполяції.

Параметр ***options*** задає додаткові параметри чисельного розв'язування:

range = numeric..numeric — діапазон інтегрування для граничних задач;

abserr = numeric — абсолютна точність інтегрування на кроці при інтегруванні задачі Коші* або оцінка абсолютної похибки розв'язку для граничної задачі;

relerr = numeric — відносна точність інтегрування на кроці при інтегруванні задачі Коші** та оцінка відносної похибки розв'язку для граничної задачі.

Параметр ***vars*** (необов'язковий) задає у вигляді множини залежні змінні системи диференціальних рівнянь.

---

* Для всіх методів, крім тих, що задаються параметром *classical.*

** Для всіх методів, крім тих, що задаються параметрами *classical* та *taylorseries*

Розглянемо приклад застосування процедури ***dsolve*** до знаходження розв'язку граничної задачі для звичайного диференціального рівняння другого порядку.

```
> a:=10;k:=1;h1:=1;h2:=1;H1:=2;H2:=1;f:=x->10*cos(Pi*x);
```

$a := 10 \quad k := 1 \quad h1 := 1 \quad h2 := 1 \quad H1 := 2 \quad H2 := 1 \quad f := x \to 10\cos(\pi x)$

```
> eq:=diff(y(x),x$2)+a*sin(k*Pi*x)*diff(y(x),x)+y(x)=f(x);
      ly:=h1*D(y)(0)-h2*y(0)=0,H1*D(y)(1)+H2*y(1)=1;
```

$$eq := \left(\frac{d^2}{dx^2}\mathrm{y}(x)\right) + 10\sin(\pi x)\left(\frac{d}{dx}\mathrm{y}(x)\right) + \mathrm{y}(x) = 10\cos(\pi x)$$

$$ly := \mathrm{D}(y)(0) - \mathrm{y}(0) = 0,\ 2\,\mathrm{D}(y)(1) + \mathrm{y}(1) = 1$$

```
> nsol:=dsolve([eq,ly],y(x),range=0..1,numeric,abserr=1.0E-8);
```

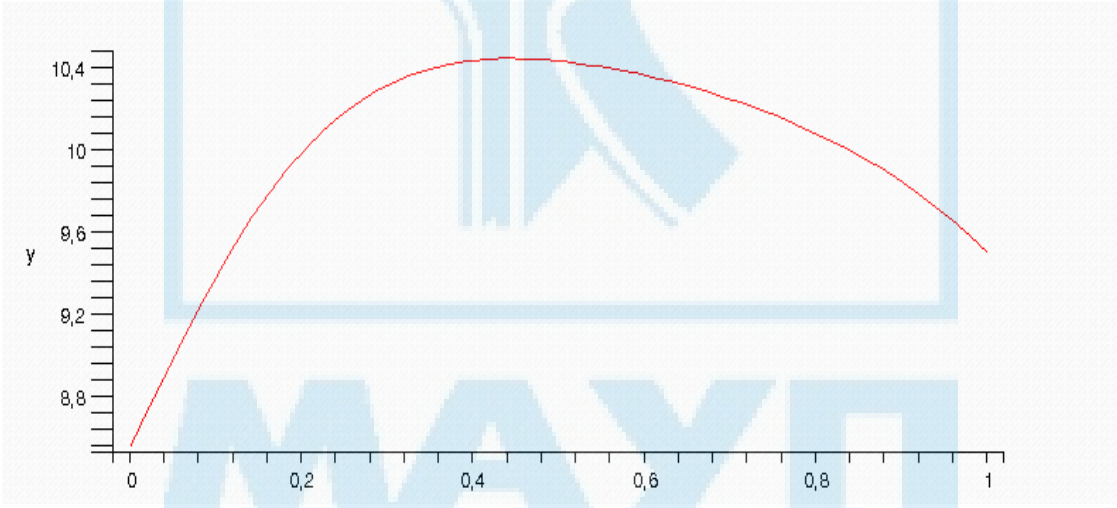
nsol := **proc** (*x_bvp*) ... **end proc**

```
> seq([i*0.05,op(2,nsol(i*0.1))],i=0..10);
```

[0., y(x) = 8.56340856206129076], [0.05, y(x) = 9.38222958731479650],
[0.10, y(x) = 9.98961288200069930], [0.15, y(x) = 10.3177704832928790],
[0.20, y(x) = 10.4354440213655853], [0.25, y(x) = 10.4330459877079971],
[0.30, y(x) = 10.3644030670559584], [0.35, y(x) = 10.2483527136566917],
[0.40, y(x) = 10.0824556006399249], [0.45, y(x) = 9.84800766007484718],
[0.50, y(x) = 9.50269067330200556]

Підключимо графічний пакет plots та побудуємо графік розв'язку.

```
> with(plots):        > odeplot(nsol);
```



Розглянемо приклад знаходження наближених розв'язків для системи шести звичайних диференціальних рівнянь з додатковими умовами Коші або граничними умовами

```
> seq(y[i],i=1..6);
```

$[y_1, y_2, y_3, y_4, y_5, y_6]$

Задамо список параметрів системи рівнянь:

```
> a:=[-1,-0.001,1,1,2,3];
```

$a := [-1, -0.001, 1, 1, 2, 3]$

Задамо систему диференціальних рівнянь першого порядку (6 рівнянь):

```
>eq:=diff(y[1](x),x)=a[1]*(y[1](x))^2+y[2](x),diff(y[2]
(x),x)= a[3]*y[1](x)*y[2](x)+a[1]*y[2](x),
```

```
diff(y[3](x),x)=
a[2]*y[3](x),
diff(y[4](x),x)=a[4]*y[3](x)+a[2]*y[4](x),
diff(y[5](x),x)=a[5]*y[4](x)+a[2]*y[5](x),
diff(y[6](x),x)=a[6]*y[5](x)+a[2]*y[6](x);
```

$$eq := \frac{d}{dx}y_1(x) = -y_1(x)^2 + y_2(x),\ \frac{d}{dx}y_2(x) = y_1(x)\,y_2(x) - y_2(x),\ \frac{d}{dx}y_3(x) = -0.001\,y_3(x),$$

$$\frac{d}{dx}y_4(x) = y_3(x) - 0.001\,y_4(x),\ \frac{d}{dx}y_5(x) = 2\,y_4(x) - 0.001\,y_5(x),$$

$$\frac{d}{dx}y_6(x) = 3\,y_5(x) - 0.001\,y_6(x)$$

Задамо початкові умови

```
> NU:=y[1](0)=1, y[2](0)=2,y[3](0)=0.5,y[4](0)=0.5,
y[5](0)=0.5, y[6](0)=0.5;
```

$$NU := y_1(0) = 1, y_2(0) = 2, y_3(0) = 0.5, y_4(0) = 0.5, y_5(0) = 0.5, y_6(0) = 0.5$$

та граничні умови

```
> KU:=y[1](0)=1,y[2](0)=1,y[3](0)=0.5,y[4](0)=0.5,
y[5](2)=10,y[6](0)=0.5;
```

$$KU := y_1(0) = 1, y_2(0) = 1, y_3(0) = 0.5, y_4(0) = 0.5, y_5(2) = 10, y_6(0) = 0.5$$

Зробимо спробу обчислити аналітичний розв'язок задачі Коші та граничної задачі:

```
> asol:=dsolve([eq,NU],func);
```

asol :=

```
> asol_G:=dsolve([eq,KU],func);
```

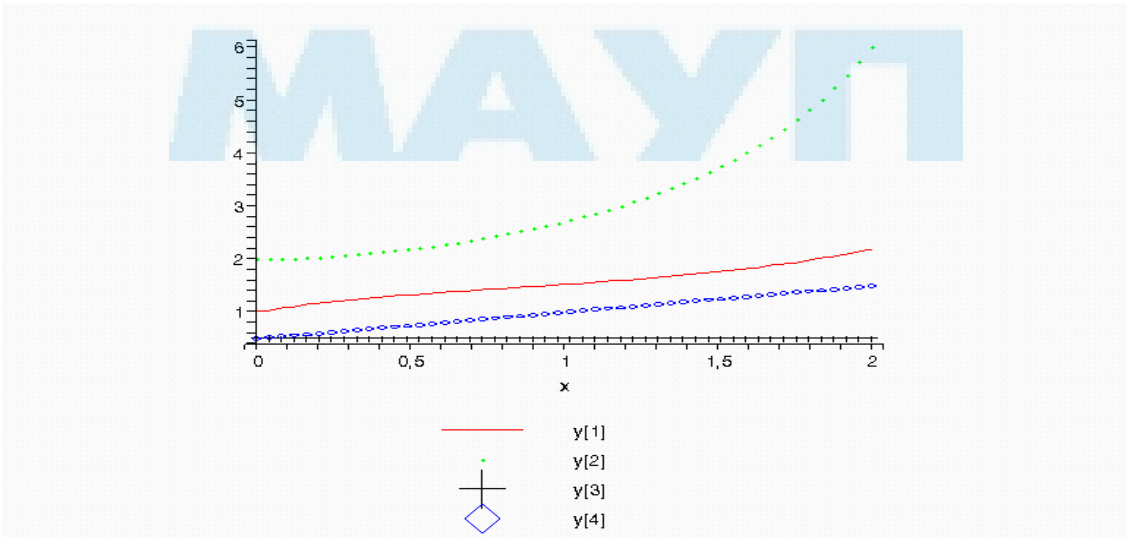
asol_G :=

Оскільки знаходження точного розв'язку неможливе, наведемо команди обчислення наближеного розв'язку задачі Коші методами Рунге – Кута, Адамса – Бешфорда, розвинення в ряд Тейлора та побудову графіків окремих компонент розв'язку.

```
> nsol1:=dsolve({eq,NU},func,range=0..2,
numeric,method=rkf45);
```

nsol1 := **proc** (*x_rkf45*) ... **end proc**

```
> odeplot(nsol1,[[x,y[1](x)],[x,y[2](x)],[x,y[3](x)],
[x,y[4](x)]]);
```



```
> nsol2:=dsolve({eq,NU},func,
numeric,method=classical[adambash]);
```

$$nsol2 := \mathbf{proc}(x\_classical) \ \ldots \ \mathbf{end\ proc}$$

```
> nsol3:=dsolve({eq,NU},func, numeric,method=taylorseries);
```

$$nsol3 := \mathbf{proc}(x\_taylorseries) \ \ldots \ \mathbf{end\ proc}$$

Обчислимо розв'язок граничної задачі:

```
> nsol1_G:=dsolve({eq,KU},func,range=0..2, numeric);
```

$$nsol1\_G := \mathbf{proc}(x\_bvp) \ \ldots \ \mathbf{end\ proc}$$

Знайдемо значення наближених розв'язків у точці $x = 1$ для задачі Коші

```
> nsol1(1);
```

$$[x = 1., y_1(x) = 0.99999999999999988, y_2(x) = 0.99999999999999988,$$
$$y_3(x) = 0.49950024991668745 4, y_4(x) = 0.99900049983337480 0,$$
$$y_5(x) = 1.99800099966673538, y_6(x) = 3.99600199938870260]$$

```
> nsol2(1);
```

$$[x = 1., y_1(x) = 1., y_2(x) = 1., y_3(x) = 0.49950024991668740 0,$$
$$y_4(x) = 0.99900049983337468 8, y_5(x) = 1.99800099966674848,$$
$$y_6(x) = 3.99600199933348942]$$

```
> nsol3(1);
```

$$[x = 1., y_1(x) = 1., y_2(x) = 1., y_3(x) = 0.49950024991668700 0, y_4(x) = 0.99900049983337500 0,$$
$$y_5(x) = 1.99800099966675, y_6(x) = 3.99600199933350]$$

та розв'язки граничної задачі в цій самій точці:

```
> nsol1_G(1);
```

$$[x = 1., y_1(x) = 0.99999999999999978, y_2(x) = 0.99999999999999978,$$
$$y_3(x) = 0.49950024991668756 6, y_4(x) = 0.99900049983337457 8,$$
$$y_5(x) = 7.51250375208348409, y_6(x) = 20.5395102556668796]$$

Знаходження екстремумів функцій

Знаходження розв'язків задач лінійного програмування

Для знаходження розв'язків задач лінійного програмування в системі Maple існує пакет *simplex,* який викликається за допомогою стандартного оператора:

```
> with(simplex);
Warning, the protected names maximize and minimize have been redefined
and unprotected
```

[*basis*, *convexhull*, *cterm*, *define_zero*, *display*, *dual*, *feasible*, *maximize*, *minimize*, *pivot*, *pivoteqn*, *pivotvar*, *ratio*, *setup*, *standardize*]

Після підключення пакета виводиться попередження про перевизначення двох стандартних функцій ядра системи Maple, а саме функцій

maximize та *minimize*. Функції з такими самими назвами існують у системі Maple.

Головними функціями пакета *simplex* є функції maximize та minimize, призначені для знаходження оптимального розв'язку задачі лінійного програмування. Виклик цих функцій здійснюється в таких формах:

maximize(ЦФ, МЛО, ТЗ), minimize(ЦФ, МЛО, ТЗ),

де ЦФ — цільова функція, яка задається у вигляді деякого лінійного виразу; МЛО — множина лінійних нерівностей стандартного вигляду; ТЗ — тип ведучих змінних (необов'язковий параметр), який може приймати значення NONNEGATIVE — невід'ємні (за замовчуванням) UNRESTRICTED — обмеження на знак ведучих змінних не накладається.

Результати своєї роботи процедури maximize та minimize повертають у вигляді множини рівнянь за її ведучими змінними, на яких цільова функція досягає максимального або мінімального значення. За відсутності оптимального розв'язку повертається порожня множина, а за необмеженого розв'язку — ***NULL.***

Розглянемо приклад знаходження розв'язку задачі лінійного програмування.

$$4x+5y+9z+11u \to \max \qquad \begin{cases} x+y+z+u \le 15, \\ 7x+5y+3z+2u \le 80, \\ 3x+5y+10z+15u \le 60; \end{cases}$$

```
> a:=4*x+5*y+9*z+11*u;
```
$$a := 2x - 3y - 3z$$

```
> L:={x+y+z+u<=15,7*x+5*y+3*z+2*u<=80,3*x+5*y+10*z+15*u<=60};
```
$$L := \{x+y+z+u \le 15,\ 7x+5y+3z+2u \le 80,\ 3x+5y+10z+15u \le 60\}$$

```
> b:=evalf(maximize(a,L,NONNEGATIVE));
```
$$b := \{y = 0.,\ u = 0.,\ z = 2.950819672,\ x = 10.16393443\}$$

```
> evalf(subs(%,a));
```
$$67.21311477$$

Функція ***dual***(ЦФ, МЛО, Y) повертає спряжену (двоїсту) лінійну задачу до задачі, заданої параметрами функції ***dual*** ЦФ, МЛО (параметр Y задає ім'я змінної, яка буде присвоєна змінним спряженої задачі).

Розглянемо приклад застосування функції ***dual*** для побудови двоїстої задачі для задачі лінійного програмування попереднього прикладу.

```
f:=dual(a,L,w);
```
$$15\,w1 + 80\,w2 + 60\,w3,\ \{4 \le w1 + 7\,w2 + 3\,w3,\ 5 \le w1 + 5\,w2 + 5\,w3,$$
$$11 \le w1 + 2\,w2 + 15\,w3,\ 9 \le w1 + 3\,w2 + 10\,w3\}$$

Знайдемо розв'язок двоїстої задачі лінійного програмування та значення цільової функції:

```
> d:=minimize(f[1],(f[2]),NONNEGATIVE);
```
$$d := \{w2 = \frac{13}{61},\ w3 = \frac{51}{61},\ w1 = 0\}$$

```
> evalf(subs(%,f[1]));
```
$$67.21311475$$

Використана функція standardize дає змогу привести лінійні обмеження, що записані у вигляді лінійних нерівностей або рівностей, до стандартного вигляду:

```
> standardize(f[2]);
```

$$\{-w1 - 7\,w2 - 3\,w3 \le -4,\ -w1 - 3\,w2 - 10\,w3 \le -9,\ -w1 - 5\,w2 - 5\,w3 \le -5,\ -w1 - 2\,w2 - 15\,w3 \le -11\}$$

Обчислення умовних екстремумів, команда extrema

Для обчислення умовних екстремумів аналітичних функцій у системі Maple існує функція ***extrema(constr, vars, 's')***, де ***extr*** — вираз, екстремум якого необхідно знайти; ***constr*** — множина обмежень, записаних у вигляді рівностей (за відсутності обмежень — порожня множина); ***vars*** — множина змінних, за якими відбувається пошук екстремуму; ***'s'*** — змінна, які присвоюється значення змінних ***vars***, в яких вираз ***extr*** набуває екстремального значення. Розглянемо приклади застосування функції ***extrema.***

```
> evalf(extrema(5*x^2-30*x*y+41*y^2-2*x-10*y+z^2-
3*z*x*y,{x^2+y=1},{x,y,z},'s'));
```

$$\{-5.53258943\}$$

```
> evalf(s);
```

$$\{\{z = 0.4468093938,\ x = 0.7889515842,\ y = 0.3775553978\}\}$$

```
> extrema(sin(z^2+x^2+y),{x+y-z=2},{x,y,z},'s');
```

$$\left\{1,\ \sin\left(\frac{3}{2}\right)\right\}$$

```
> s;
```

$$\left\{\{x = \mathrm{RootOf}(2\,\_Z^2 + 2\,\_Z^2 - 2\,\_Z + 2\,z + 4 - \pi),\ z = z,\ y = -\mathrm{RootOf}(2\,\_Z^2 + 2\,\_Z^2 - 2\,\_Z + 2\,z + 4 - \pi) + z + 2\},\right.$$

$$\left\{y = 1,\ x = \frac{1}{2},\ z = \frac{-1}{2}\right\},\ \left\{y = \frac{3}{2} + \%1,\ z = \%1,\ x = \frac{1}{2}\right\},$$

$$\left.\left\{y = -\%2 + \frac{3}{2},\ x = \%2,\ z = \frac{-1}{2}\right\}\right\}$$

$$\%1 := \frac{1}{2}\,\mathrm{RootOf}(\_Z^2 + 7 + 2\,\_Z - 2\,\pi,\ label = \_L15)$$

$$\%2 := \frac{1}{2}\,\mathrm{RootOf}(7 + \_Z^2 - 2\,\_Z - 2\,\pi,\ label = \_L16)$$

```
> allvalues(s)[3];
```

$$\left\{ \left\{ x = \frac{1}{2} + \frac{1}{2}\sqrt{-6+2\pi}, y = 1 - \frac{1}{2}\sqrt{-6+2\pi}, z = \frac{-1}{2} \right\},$$

$$\left\{ x = \frac{1}{2} + \frac{1}{2}\sqrt{-7-4z^2+2\pi-4z}, y = \frac{3}{2} - \frac{1}{2}\sqrt{-7-4z^2+2\pi-4z}+z, z = z \right\}, \left\{ y = 1, x = \frac{1}{2}, z = \frac{-1}{2} \right\},$$

$$\left\{ y = 1 - \frac{1}{2}\sqrt{-6+2\pi}, z = -\frac{1}{2} - \frac{1}{2}\sqrt{-6+2\pi}, x = \frac{1}{2} \right\} \right\}$$

Знаходження максимуму та мінімуму функцій

У системі Maple для знаходження максимуму та мінімуму аналітичних функцій однієї або багатьох змінних використовуються функції ***minimize(expr, opt1,opt2,optn)*** та ***maximize(expr, opt1,opt2,optn).*** Параметри цих функцій мають такий зміст:

expr — функція або вираз, мінімальне або максимальне значення якої необхідно обчислити;

opt1,opt2,optn — дають змогу задавати додаткові параметри цих функцій.

Так, для обмеження множини пошуку за деякими змінними, від яких залежить ***expr***, можна задавати опції вигляду t = a..b, де t — ім'я змінної; a — ліва межа; b — права межа зміни t. Якщо потрібно отримати розширений результат роботи функцій, можна задати параметр ***location***, який дає змогу вивести не тільки максимальне або мінімальне значення функції, а й значення змінних у цих точках.

Розглянемо приклади застосування цих функцій.

```
> (minimize(3*x^2+2*y^2+z^2+2*x*z-14*x-8*y-
100,x=0..2,y=1..3,z=0..5,location));
```

$$-124, \{[\{z = 0, y = 2, x = 2\}, -124]\}$$

```
>minimize(3*x^2+2*y^2+z^2+2*x*z-14*x-8*y-100,location);
```

$$\frac{-265}{2}, \left\{\left[\{x = \frac{7}{2}, z = \frac{-7}{2}, y = 2\}, \frac{-265}{2}\right]\right\}$$

```
>maximize(y^3-x^2-27*y+3*x+16,x=-5..5,y=-5..5,location);
```

$$\frac{289}{4}, \left\{\left[\{y = -3, x = \frac{3}{2}\}, \frac{289}{4}\right]\right\}$$

```
> (maximize(2*ln(x)+4*ln(y)+ln(7-x-y),x=0.1..2,y=0.1..4));
```

$$\text{maximize}(2\ln(x) + 4\ln(y) + \ln(7 - x - y), x = 0.1 \ldots 2, y = 0.1 \ldots 4)$$

Наведені приклади демонструють доволі обмежений характер застосування функцій пошуку мінімуму та максимуму функцій кількох

змінних. Останній приклад повинен мати максимальне значення, водночас знайти його значення і точку локалізації системі Maple не вдається*.

Завдання для самостійного виконання III

1 Знайти чисельний розв'язок задачі Коші для звичайного диференціального рівняння другого порядку з використанням методів Рунге – Кутта – Фелберга 4–5 та Адамса. Результат обчислень вивести у вигляді таблиці на проміжку [0,2] з кроком 0.2. Побудувати графіки чисельного розв'язку.

1.1 $y'' + 3y'y + y^2 = \dfrac{9e^{3x}}{1+e^{3x}},\ y(0) = \ln 4,\ y'(0) = 3(1-\ln 2).$

1.2 $y'' + 4y(y')^2 + y^3 = 8\operatorname{ctg} 2x,\ y\left(\dfrac{\pi}{4}\right) = 5,\ y'\left(\dfrac{\pi}{4}\right) = 4.$

1.3 $y'' - 9(y')y^2 + 18y^4 = \dfrac{9e^{3x}}{1+e^{-3x}},\ y(0) = 0\ \ y'(0) = 0.$

1.4 $y'' + 6(y')^3 + 8y = \dfrac{4e^{-2x}}{2+e^{2x}},\ y(0) = 0,\ y'(0) = 0.$

1.5 $y'' - 3\sin(y') + 2y = \dfrac{1}{3+e^{-x}},\ y(0) = 1 + 8\ln 2,\ y'(0) = 14\ln 2.$

2 Знайти розв'язок задачі Коші, задачі для системи звичайних диференціальних рівнянь методами Адамса та Гіра. Вивести таблицю результатів на проміжку [0,3] з кроком 0.1.

2.1
$$\begin{cases} \dfrac{dx}{dt} = 2xy - z - yz + 2t^2, \\ \dfrac{dy}{dt} = 3x - 2y^3 - 3zx - \sin t, \\ \dfrac{dz}{dt} = 2z - x + y, \\ x(0) = 1,\ y(0) = 1,\ z(0) = 0. \end{cases}$$

2.2
$$\begin{cases} \dfrac{dx}{dt} = 2x^3 + 2zy^2 - \sin y, \\ \dfrac{dy}{dt} = \operatorname{arctg} x + 2z + 3t^2, \\ \dfrac{dz}{dt} = y - 2x - z^2 - \cos t, \\ x(0) = 1,\ y(0) = 2,\ z(0) = 1. \end{cases}$$

* В систему MAPLE 9.5 включений новий пакет оптимізації Optimization, який має розширені можливості знаходження розв'язку лінійних і нелінійних задач оптимізації.

2.3
$$\begin{cases} \dfrac{dx}{dt} = \cos x + z - y - 2t^2 + t, \\ \dfrac{dy}{dt} = x + \operatorname{arctg}(y^2) - z, \\ \dfrac{dz}{dt} = 2xy - y + \sin t, \\ x(0) = 0,\ y(0) = 1,\ z(0) = 2. \end{cases}$$

2.4
$$\begin{cases} \dfrac{dx}{dt} = \cos(x - z) - y + (t-1)^2, \\ \dfrac{dy}{dt} = \ln(x^2 + y^2), \\ \dfrac{dz}{dt} = 3xy + z + \sin t^2, \\ x(0) = 1,\ y(0) = 0,\ z(0) = 2. \end{cases}$$

2.5
$$\begin{cases} \dfrac{dx}{dt} = \operatorname{arctg}(2x - y^2) + z + \sin t, \\ \dfrac{dy}{dt} = (x + 2y - z)^3, \\ \dfrac{dz}{dt} = (x - y)2z + \dfrac{t^2}{2} + 1, \\ x(0) = 2,\ y(0) = 0,\ z(0) = 1. \end{cases}$$

3 Знайти чисельно розв'язок граничної задачі, побудувати графік розв'язку на вказаному проміжку.

$$\begin{cases} -\dfrac{d}{dx}(k(x)\dfrac{d}{dx}y\ ) + q(x,y) = f(x), \quad 0 < x < 1, \\ h_1 y'(0) - h_2 y(0) = 0, \\ H_1 y'(1) + H_2 y(1) = 0. \end{cases}$$

Варіант	$k(x)$	$q(x,y)$	h1,h2	H1, H2	$f(x)$
3.1	$1 + x^2 + 2x$	$\ln(1 + y^2)x$	0,1	1,1	$x^2 + \sin x$
3.2	$1 + \sin(\pi x)$	$\operatorname{arctg}(yx)$	1,2	0,1	$2x^2 + \sin 2x$
3.3	$1 + x^2 + 2x$	$x \cdot \sin(y)$	1,1	1,0	$x^3 + e^{2x}$
3.4	$2 - \sin(\pi x)$	$\cos(x^2 + y^2)$	1,2	0,1	$2x^3 - e^{2x}$
3.5	$1 + \sin(\pi x)$	$\sqrt{1 + y^2}\sin x$	2,1	1,0	$\operatorname{arctg}(x/2)$

4 Знайти чисельно всі дійсні розв'язки нелінійного рівняння.

4.1 $T_5(x) = 2e^{-x}$, $T_5(x)$ — поліном Чебишева першого роду 5 степеня.

4.2 $U_6(x) = \text{arctgx}$, $U_6(x)$ – поліном Чебишева другого роду 6 степеня.

4.3 $P_4(x) = \dfrac{1}{1+x^2}$, $P_4(x)-$ поліном Лежандра 4 степеня.

4.4 $H_5(x) = \dfrac{x^2}{1+x^4}$, $H_5(x)-$поліном Ерміта 5 степеня.

4.5 $T_6(x) = 1 + e^{-x^2}$, $T_6(x)-$поліном Чебишева першого роду 6 степеня.

5 Знайти чисельно розв'язок системи нелінійних рівнянь з точністю $\varepsilon = 10^{-6}$, який розміщується в околі точки $(x^0, y^0) = (1.25, 0)$:

5.1 $\begin{cases} \sin(2x-y) - 1.2x = 0.4; \\ 0.8x^2 + 1.5y^2 = 1. \end{cases}$ 5.2 $\begin{cases} \sin(x-0.6) - y = 1.6; \\ 3x - \cos y = 0.9. \end{cases}$

5.3 $\begin{cases} \text{tg}(xy+0.1) = x^2; \\ x^2 + 2y^2 = 1. \end{cases}$ 5.4 $\begin{cases} \sin(x-y) - xy = -1; \\ x^2 - y^2 = 0.75. \end{cases}$

5.5 $\begin{cases} \sin x + 2y = 1.6; \\ \cos(y-1) = 1. \end{cases}$.

6 Знайти розв'язок задачі лінійного програмування та записати двоїсту задачу лінійного програмування.

6.1 $\begin{cases} x_1 + x_2 + x_3 \to \min, \\ x_1 - x_4 - 2x_6 = 5, \\ x_2 + 2x_4 - 3x_5 + x_6 = 3, \\ x_3 + 2x_4 - 5x_5 + 6x_6 = 5, \\ x_i \ge 0, i = 1..6. \end{cases}$ 6.2 $\begin{cases} 2x_1 + x_2 - x_3 - x_4 \to \min, \\ x_1 + x_2 + 2x_3 - x_4 = 2, \\ 2x_1 + x_2 - 3x_3 + x_4 = 6, \\ x_1 + x_2 - x_3 + x_4 = 7, \\ x_i \ge 0, i = 1..4. \end{cases}$

6.3 $\begin{cases} x_1 - 2x_2 + 3x_3 \to \min, \\ -2x_1 + x_2 + x_3 = 2, \\ 2x_1 + 3x_2 + 4x_3 = 1, \\ x_i \ge 0, i = 1..3. \end{cases}$ 6.4 $\begin{cases} 4x_1 + 5x_2 + 9x_3 + 11x_4 \to \max, \\ x_1 + x_2 + x_3 + x_4 \le 15, \\ 7x_1 + 5x_2 + 3x_3 + 2x_4 \le 80, \\ 3x_1 + 5x_2 + 10x_3 + 15x_4 \le 60, \\ x_i \ge 0, i = 1..4. \end{cases}$

6.5
$$\begin{cases} -3x_1 + x_2 + 3x_3 - 34x_4 \to \min, \\ x_1 + 2x_2 - x_3 + x_4 = 0, \\ 2x_1 - 2x_2 3x_3 + 3x_4 = 9, \\ x_1 - x_2 + 2x_3 - x_4 = 6, \\ x_i \ge 0, i = 1..4. \end{cases}$$

7 Знайти умовний екстремум функції.

7.1 $u(x,y,z) = xyz,\ x^2 + y^2 + z^2 = 3.$

7.2 $u(x,y,z) = x^2 y^3 z^2,\ x + y + z = 6.$

7.3 $u(x,y,z) = x^2 + y^2 + z^2,\ \frac{x^2}{4} + \frac{y^2}{9} + \frac{z^2}{1} = 1.$

7.4 $u(x,y,z) = xyz,\ x^2 + y^2 + z^2 = 1,\ x + y + z = 0.$

7.5 $u(x,y,z) = xy + yz,\ x^2 + y^2 = 2,\ y + z = 2.$

Структури даних пакета LinearAlgebra

У системі Maple є дві базові структури для організації табличних даних: rtable та table. На основі першої з них базуються функції Array, Matrix і Vector. Саме дані типу Matrix, Vector, а також числові дані (типу algebraic) використовуються в пакеті LinearAlgebra.

Матриці та вектори в LinearAlgebra створюють за допомогою команд відповідно Matrix() та Vector() або коротких позначень.

Короткі позначення для матриць та векторів

Для того щоб задати матрицю або вектор, можна використовувати трикутні дужки “<” і “>”. При цьому запис вигляду <a,b,c> означає, що структура даних організовується за стовпцями, а <a|b|c> — за рядками. Наприклад:

```
> <1,2>;
```

$$\begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

```
> <1|2|3>;
```

$$[1, 2, 3]$$

```
> <<1,2,3>|<4,5,6>>;
```

$$\begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$$

```
> <<1|2|3>,<4|5|6>>;
```

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

Функція Matrix()

Команда Matrix() має такий синтаксис:

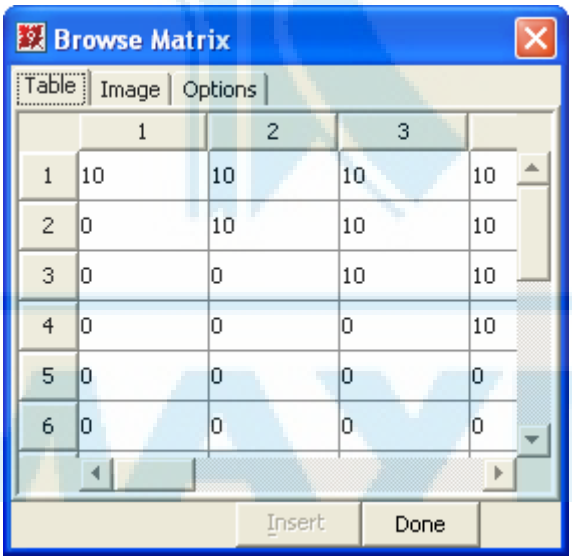
```
> Matrix(r,c,init,ro,sym,sc,sh,st,o,dt,f,a);
```

Тут r, c — розміри матриці відповідно за рядками та стовпцями. У випадку квадратної матриці параметр c можна опустити. За замовчуванням Maple повністю виводить на екран лише матриці, розміри яких не перевищують 10x10. Більші матриці виводяться так:

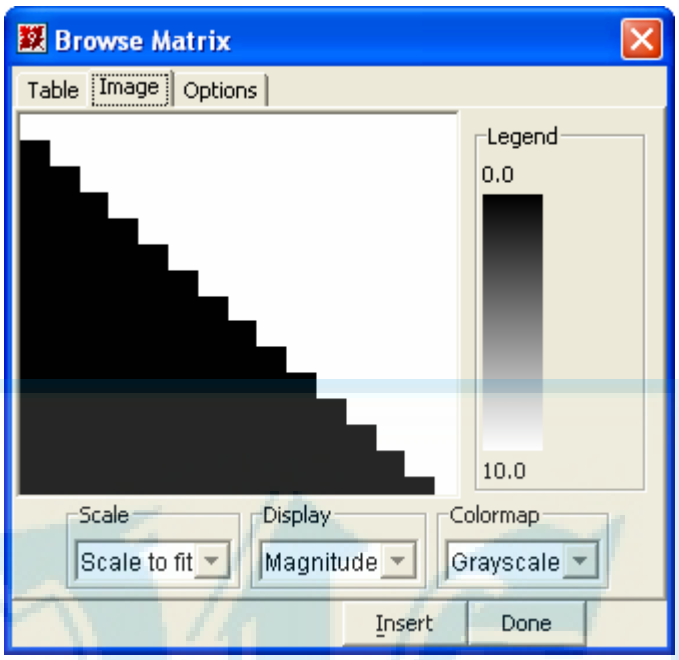
```
> Matrix(15,15,10,shape=triangular[upper]);
```

$$\begin{bmatrix} \textit{15 x 15 Matrix} \\ \textit{Data Type: anything} \\ \textit{Storage: triangular[upper]} \\ \textit{Order: Fortran\_order} \end{bmatrix}$$

Для їх перегляду потрібно двічі клацнути мишею на отриманому об’єкті. У результаті відкриється вікно Browse Matrix, за допомогою якого можна переглянути як елементи матриці (на вкладці Table)



так і її структуру (на вкладці Image)



Максимальні розміри матриць, які безпосередньо виводяться на екран, можна змінити за допомогою команди interface(rtablesize= MaxValue).

Елементи матриці можна задати за допомогою параметра init. Якщо його опущено, матриця заповнюється нулями. Наприклад:

```
> Matrix(2);
```

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

```
> Matrix(2,3);
```

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

Параметр init можна задати кількома способами. Це може бути процедура або функція, яка кожній парі індексів ставить у відповідність деякий об’єкт типу algebraic; об’єкт типу table, array, Array або Matrix; множину рівностей виду (i,j)=value, за допомогою яких можна явно вказати елементи матриці з відповідними індексами; об’єкт Vector для діагональної матриці; список list елементів або списків list, за допомогою яких перелічено елементи матриці. В останньому випадку порядок перерахунку задається параметром sc.

Розглянемо приклади.

```
> mA:=Matrix(2,2,(i,j)->i+2*j);
```

$$mA := \begin{bmatrix} 3 & 5 \\ 4 & 6 \end{bmatrix}$$

```
> Matrix(3,3,mA);
```

$$\begin{bmatrix} 3 & 5 & 0 \\ 4 & 6 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

```
> Matrix(3,3,{(2,3)=10,(3,2)=100});
```

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 10 \\ 0 & 100 & 0 \end{bmatrix}$$

Параметр ro — це рівність виду readonly=true (або false), яка забороняє (або дозволяє) зміну елементів матриці.

Параметр sym виду symbol=name задає символ для позначення елементів матриці.

```
> mB:=Matrix(2,2,readonly=true,symbol=b);
```

$$mB := \begin{bmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{bmatrix}$$

```
> mB[1,1]:=0;
```

Error, cannot assign to a read-only Matrix

Параметр sc використовується тоді, коли init є списком. Він має вигляд scan=name або scan=list. Значеннями параметра scan можуть бути службові слова rectangular, triangular[upper], triangular[lower], Hessenberg[upper], Hessenberg[lower], band[m,n], band[m], diagonal. Додатково у списку scan може задаватись порядок заповнення матриці: rows — за рядками, columns — за стовпцями, diagonals — за діагоналями.

Параметр scan=rectangular означає, що значення списку списків init заповнюють матрицю як прямокутну починаючи з верхнього лівого елемента. Заповнення може відбуватися за рядками (за замовчуванням) або стовпцями. Якщо розміри матриці більші від розмірів прямокутника, що заданий параметром init, залишкові елементи заповнюються нулями.

```
> Matrix(2,2,[[1,2],[3,4]],scan
=[rectangular,rows]);
```

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

```
> Matrix(3,4,[[1,2],[3,4]], scan
=[rectangular,columns]);
```

$$\begin{bmatrix} 1 & 3 & 0 & 0 \\ 2 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Параметр scan=triangular[upper] означає, що значення списку списків init заповнюють матрицю як верхню трикутну починаючи з верхнього лівого елемента. При цьому перший елемент кожного списку всередині init лежить на діагоналі. Заповнення може відбуватися за рядками (за замовчуванням) або стовпцями. Якщо матриця не вичерпується параметром init, залишкові елементи заповнюються нулями. Параметр scan=triangular[lower] працює подібно.

```
> Matrix(3,3,[[1,2,3],[4,5],[6]],
scan=[triangular[upper],rows]);
```

$$\begin{bmatrix} 1 & 2 & 3 \\ 0 & 4 & 5 \\ 0 & 0 & 6 \end{bmatrix}$$

```
> Matrix(3,3,[[1],[2,3],[4,5,6]],
scan=[triangular[upper],columns]);
```

$$\begin{bmatrix} 1 & 2 & 4 \\ 0 & 3 & 5 \\ 0 & 0 & 6 \end{bmatrix}$$

```
> Matrix(3,3,[[1],[2,3],[4,5,6]],
scan=[triangular[lower],rows]);
```

$$\begin{bmatrix} 1 & 0 & 0 \\ 2 & 3 & 0 \\ 4 & 5 & 6 \end{bmatrix}$$

```
> Matrix(3,3,[[1,2,3],[4,5],[6]],
scan=[triangular[lower],columns]);
```

$$\begin{bmatrix} 1 & 0 & 0 \\ 2 & 4 & 0 \\ 3 & 5 & 6 \end{bmatrix}$$

Параметр scan=Hessenberg[upper] означає, що значення списку списків init заповнюють матрицю як гесенбергову, тобто як верхню трикутну з піддіагоналлю. При цьому останній елемент кожного списку всередині init лежить на піддіагоналі. Заповнення може відбуватися за рядками (за

замовчуванням) або стовпцями. Якщо матриця не вичерпується параметром init, залишкові елементи заповнюються нулями. Параметр scan=Hessenberg[lower] працює подібно.

```
> Matrix(4,4,[[1,2,3,4],[5,6,7,8],
[9,10,11],[12,13]],
scan=[Hessenberg[upper],rows]);
```

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 0 & 9 & 10 & 11 \\ 0 & 0 & 12 & 13 \end{bmatrix}$$

```
> Matrix(4,4,[[1,2],[3,4,5],
[6,7,8,9], [10,11,12,13]],
scan=[Hessenberg[upper],columns]);
```

$$\begin{bmatrix} 1 & 3 & 6 & 10 \\ 2 & 4 & 7 & 11 \\ 0 & 5 & 8 & 12 \\ 0 & 0 & 9 & 13 \end{bmatrix}$$

```
> Matrix(4,4,[[1,2],[3,4,5],
[6,7,8,9], [10,11,12,13]],
scan=[Hessenberg[lower],rows]);
```

$$\begin{bmatrix} 1 & 2 & 0 & 0 \\ 3 & 4 & 5 & 0 \\ 6 & 7 & 8 & 9 \\ 10 & 11 & 12 & 13 \end{bmatrix}$$

```
> Matrix(4,4,[[1,2,3,4],[5,6,7,8],
[9,10,11],[12,13]],
scan=[Hessenberg[lower],columns]);
```

$$\begin{bmatrix} 1 & 5 & 0 & 0 \\ 2 & 6 & 9 & 0 \\ 3 & 7 & 10 & 12 \\ 4 & 8 & 11 & 13 \end{bmatrix}$$

Параметр scan=band[m,n] означає, що значення списку списків init заповнюють матрицю як (m+n+1)-діагональну. При цьому під головною діагоналлю розташовано m піддіагоналей, а над нею — n наддіагоналей. За замовчуванням заповнення здійснюється за діагоналями, починаючи з найнижчої. Заповнення може відбуватися також за рядками або стовп-

цями. Якщо матриця не вичерпується параметром init, залишкові елементи заповнюються нулями. Параметр scan=band[m] еквівалентний scan=band[m,m], а scan=diagonal — scan=band[0].

```
> Matrix(4,4,[[1,2,3],[4,5,6,7],
[8,9,10],[11,12]],
scan=[band[1,2],diagonals]);
```

$$\begin{bmatrix} 4 & 8 & 11 & 0 \\ 1 & 5 & 9 & 12 \\ 0 & 2 & 6 & 10 \\ 0 & 0 & 3 & 7 \end{bmatrix}$$

```
> Matrix(4,4,[[1,2,3],[4,5,6,7],
[8,9,10],[11,12]],
scan=[band[1,2],rows]);
```

$$\begin{bmatrix} 1 & 2 & 3 & 0 \\ 4 & 5 & 6 & 7 \\ 0 & 8 & 9 & 10 \\ 0 & 0 & 11 & 12 \end{bmatrix}$$

```
> Matrix(4,4,[[1,2],[3,4,5],
[6,7,8,9],[10,11,12]],
scan=[band[1,2],columns]);
```

$$\begin{bmatrix} 1 & 3 & 6 & 0 \\ 2 & 4 & 7 & 10 \\ 0 & 5 & 8 & 11 \\ 0 & 0 & 9 & 12 \end{bmatrix}$$

```
> Matrix(4,4,[[1,2],[3,4,5],
[6,7,8,9],[10,11,12]],
scan=[band[2,1],diagonals]);
```

$$\begin{bmatrix} 6 & 10 & 0 & 0 \\ 3 & 7 & 11 & 0 \\ 1 & 4 & 8 & 12 \\ 0 & 2 & 5 & 9 \end{bmatrix}$$

```
> Matrix(4,4,[[1,2,3],[4,5,6,7],
[8,9,10]], scan=band[1]);
```

$$\begin{bmatrix} 4 & 8 & 0 & 0 \\ 1 & 5 & 9 & 0 \\ 0 & 2 & 6 & 10 \\ 0 & 0 & 3 & 7 \end{bmatrix}$$

На відміну від параметра scan, який лише забезпечує зчитування початкових елементів матриці, параметр sh описує математичну структуру матриці і має вид shape=name або shape=list. Тут name — вбудована або створена користувачем індексна функція, тобто функція, яка при введенні або зміні елемента матриці виконує з матрицею деякі додаткові операції. Якщо функцій кілька, їх записують як список. Параметр shape використовує такі вбудовані індексні функції: antihermitian (для антиермітової матриці з $a_{ij} = -\overline{a_{ji}}$), antisymmetric (для антисиметричної матриці з $a_{ij} = -a_{ji}$), band (для n-діагональної матриці), constant (для матриці, усі елементи якої дорівнюють заданому скаляру), diagonal (для діагональної матриці), identity (для одиничної матриці), hermitian (для ермітової матриці з $a_{ij} = \overline{a_{ji}}$), scalar (для матриці, усі діагональні елементи якої дорівнюють заданому скаляру), symmetric (для симетричної матриці з $a_{ij} = a_{ji}$), triangular (для верхньої або нижньої трикутної матриці), zero (для нульової матриці), Hessenberg (для верхньої або нижньої матриці Гесенберга).

Розглянемо приклади.

```
> Matrix(3,3,shape=identity);
```

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

```
> Matrix(2,3,shape=zero);
```

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

```
> Matrix(3,3,shape=scalar[5]);
```

$$\begin{bmatrix} 5 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 5 \end{bmatrix}$$

```
> Matrix(2,3,shape=constant[5]);
```

$$\begin{bmatrix} 5 & 5 & 5 \\ 5 & 5 & 5 \end{bmatrix}$$

```
> mAsh:=Matrix(3,3,[1,2,3],
scan=diagonal, shape=diagonal);
```

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

```
> mAsh[1,2]:=5;
```

Error, attempt to assign non-zero to off-diagonal entry of a diagonal Matrix

```
> mBsh:=Matrix(2,2,shape=symmetric);
```

$$mBsh := \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

```
> mBsh[1,2]:=10;
```

$$mBsh_{1,2} := 10$$

```
> mBsh;
```

$$\begin{bmatrix} 0 & 10 \\ 10 & 0 \end{bmatrix}$$

```
> mCsh:=Matrix(2,2,shape=
antisymmetric);
```

$$mCsh := \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

```
> mCsh[2,1]:=10;
```

$$mCsh_{2,1} := 10$$

```
> mCsh;
```

$$\begin{bmatrix} 0 & -10 \\ 10 & 0 \end{bmatrix}$$

```
> mDsh:=Matrix(2,2,shape=hermitian);
```

$$mDsh := \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

```
> mDsh[1,2]:=5+I;
```

$$mDsh_{1,2} := 5 + I$$

```
> mDsh;
```

$$\begin{bmatrix} 0 & 5+I \\ 5-I & 0 \end{bmatrix}$$

```
> mEsh:=Matrix(2,2,shape=
antihermitian);
```

$$mEsh := \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

```
> mEsh[1,2]:=5+I;
```

$$mEsh_{1,2} := 5 + I$$

```
> mEsh;
```

$$\begin{bmatrix} 0 & 5+I \\ -5+I & 0 \end{bmatrix}$$

```
> mM:=<<1|2|3>,<4|5|6>,<7|8|9>>;
```

$$mM := \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

```
> Matrix(3,3,mM,shape=
triangular[upper]);
```

$$\begin{bmatrix} 1 & 2 & 3 \\ 0 & 5 & 6 \\ 0 & 0 & 9 \end{bmatrix}$$

```
> Matrix(3,3,mM,shape=
triangular[lower]);
```

$$\begin{bmatrix} 1 & 0 & 0 \\ 4 & 5 & 0 \\ 7 & 8 & 9 \end{bmatrix}$$

```
> Matrix(3,3,mM,shape=
Hessenberg[upper]);
```

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 0 & 8 & 9 \end{bmatrix}$$

```
> Matrix(3,3,mM,shape=
Hessenberg[lower]);
```

$$\begin{bmatrix} 1 & 2 & 0 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

```
> Matrix(3,3,mM,shape=band[0,1]);
```

$$\begin{bmatrix} 1 & 2 & 0 \\ 0 & 5 & 6 \\ 0 & 0 & 9 \end{bmatrix}$$

Параметри st (storage=name) і o (order=name) задають розташування матриці у фізичній пам'яті. Ці параметри важливі лише при роботі з дуже великими матрицями, тому не зупинятимемося на них.

Параметр dt (datatype=name) визначає тип даних елементів матриці. Параметр f (fill=value) задає значення, яке використовується для невизначених елементів матриці. Наприклад:

```
> Matrix(3,3,{(1,3)=100,(3,2)=200},fill=-1);
```

$$\begin{bmatrix} -1 & -1 & 100 \\ -1 & -1 & -1 \\ -1 & 200 & -1 \end{bmatrix}$$

Параметр a має вигляд attributes=list і задає список додаткових атрибутів матриці. Зокрема, деякі алгоритми пакета LinearAlgebra використовують певні атрибути матриць для підвищення ефективності обчислень.

```
> mAat:=Matrix(2,2,[[1,2],[3,4]],
attributes=[matrix_of_system]);
```

$$mAat := \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

```
> attributes(mAat);
```

$$matrix\_of\_system$$

Функція Vector()

Команда Vector() має такий синтаксис:

```
Vector[o](d,init,ro,sym,sh,st,dt,f,a,o)
```

Тут [o] — параметр вигляду [row] або [column], який задає відповідно вектор-рядок або вектор-стовпець.

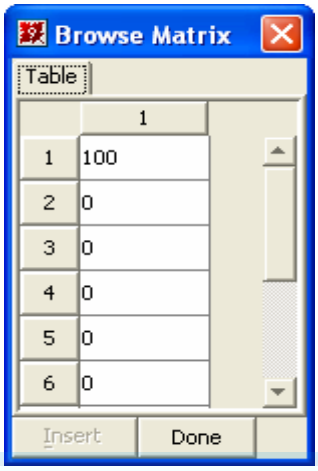
Орієнтацію вектора можна задати також за допомогою параметра o, який записується так: orientation=row або orientation=column. Зауважимо, що за наявності обох цих параметрів перевага надається [o]. За замовчуванням використовується орієнтація у стовпець.

Параметр d — це невід'ємне ціле число, яке задає розмірність вектора. За замовчуванням система Maple виводить лише вектори, розмірність яких не перевищує 10.

```
> Vector(11,shape=scalar[1,100]);
```

$$\begin{bmatrix} 11 \ Vector[column] \\ Data\ Type: anything \\ Storage: empty \\ Order: Fortran\_order \end{bmatrix}$$

Для перегляду елементів великих векторів потрібно викликати вікно Browse Matrix:



Інші параметри подібні до відповідних у конструкторі Matrix(). Тому наведемо лише відмінності. Зокрема, init не може бути матрицею або двовимірним списком, список списків зчитується як один рядок. Параметр sh виду shape=name або shape=list допускає такі вбудовані індексні функції: constant, scalar, unit, zero.

Розглянемо приклади.

```
> Vector([1,2]);
```

$$\begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

```
> Vector[row](5,{2=10,4=100});
```

$$[0, 10, 0, 100, 0]$$

```
> Vector[row](5,{2=10,4=100},fill=-1);
```

$$[-1, 10, -1, 100, -1]$$

```
> Vector[row](6,shape=constant[2]);
```

$$[2, 2, 2, 2, 2, 2]$$

```
> Vector[row](5,shape=unit[3]);
```

$$[0, 0, 1, 0, 0]$$

```
> Vector[row](5,shape=scalar[2,10]);
```

$$[0, 10, 0, 0, 0]$$

```
> vZ:=Vector(2,shape=zero,
readonly=true);
```

$$vZ := \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

```
> vZ[1]:=1;
```

```
Error, cannot assign
to a read-only Vec-
tor
```

```
> Vector[row](6,i->x^i);
```

$$[x, x^2, x^3, x^4, x^5, x^6]$$

```
> Vector(2,symbol=u);
```

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

Перегляд та зміна параметрів матриць і векторів

Пакет LinearAlgebra надає дві функції для перегляду та встановлення параметрів матриць і векторів — MatrixOptions() та VectorOptions(). Їх синтаксис такий:

```
MatrixOptions(A,opt1,opt2,...)
```

```
VectorOptions(V,opt1,opt2,...)
```

Тут A — матриця; V — вектор; opt1, opt2, ... — назви параметрів або рівності виду option = value. Якщо ці функції викликаються з назвою параметра, система Maple виводить значення цього параметра для певної матриці чи вектора. Якщо ж параметр задається рівністю option = value, для матриці чи вектора встановлюється нове значення цього параметра. Опції dimensions, datatype, shape та storage змінювати не можна. Параметр readonly можна лише встановлювати.

Розглянемо приклади.

```
> with(LinearAlgebra):
> mA:=Matrix(3,[[1,2,3],[4,5],[6]],
scan=triangular[upper],shape=triangular[upper]);
```

$$mA := \begin{bmatrix} 1 & 2 & 3 \\ 0 & 4 & 5 \\ 0 & 0 & 6 \end{bmatrix}$$

```
> MatrixOptions(mA);
```

$shape = [triangular_{upper}]$, $datatype = anything$, $storage = triangular_{upper}$, $order = Fortran\_order$

```
> MatrixOptions(mA,storage=rectangular);
    Error, (in MatrixOptions) storage cannot be changed
> MatrixOptions(mA,readonly=true);
> MatrixOptions(mA);
```

$shape = [triangular_{upper}]$, $datatype = anything$, $storage = triangular_{upper}$, $order = Fortran\_order$, $readonly$

```
> MatrixOptions(mA,readonly=false);
   Error, (in MatrixOptions) readonly can only be set, not
                          cleared
```

```
> vV:=Vector([2,4]);
```

$$vV := \begin{bmatrix} 2 \\ 4 \end{bmatrix}$$

```
> VectorOptions(vV,orientation);
```

$$column$$

```
> VectorOptions(vV,orientation=row);
> VectorOptions(vV);
```

$shape = [], datatype = anything, orientation = row, storage = rectangular, order = Fortran\_order$

```
> vV;
```

$$[2, 4]$$

Визначення розмірності матриць та векторів

Пакет LinearAlgebra надає такі функції для визначення розмірності матриць та векторів: RowDimension(M) повертає кількість рядків матриці M, ColumnDimension(M) — стовпців, Dimension(A) — розмірність вектора A або кількість рядків і стовпців матриці A.

```
> with(LinearAlgebra):
> mA:=Matrix(2,7);
```

$$mA := \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

```
> RowDimension(mA);
```

$$2$$

```
> ColumnDimension(mA);
```

$$7$$

```
> Dimension(mA);
```

$$2, 7$$

```
> vV:=Vector[row]([1,2,3,4,5,6,7,8]);
```

$$vV := [1, 2, 3, 4, 5, 6, 7, 8]$$

```
> Dimension(vV);
```

$$8$$

Перевірка рівності матриць та векторів

Перевірити, чи збігаються дві матриці (два вектора) як поелементно, так і за структурою, можна за допомогою функції Equal(). Її синтаксис такий:

```
Equal(A,B,compare=method)
```

Тут A, B — дві матриці (або два вектори); method — один з методів перевірки рівності: entries, structure чи all. Метод entries (береться за замовчуванням) означає перевірку рівності типів (Matrix або Vector) об'єктів, їх розмірностей і елементів. Метод structure перевіряє, чи однакові па-

раметри dimension, datatype, shape, storage, attributes і orientation. Метод all об'єднує методи entries і structure.

```
mA:=Matrix(2,[[0,1],[1,0]],shape=symmetric);
```

$$mA := \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

```
> mB:=<<0|1>,<1|0>>;
```

$$mB := \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

```
> Equal(mA,mB);
```

true

```
> Equal(mA,mB,compare=entries);
```

true

```
> Equal(mA,mB,compare=structure);
```

false

```
> Equal(mA,mB,compare=all);
```

false

```
> vV:=<1|2>;
```

$vV := [1, 2]$

```
> vU:=<1,2>;
```

$$vU := \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

```
> Equal(vV,vU,compare=entries);
```

false

Функції IsMatrixShape() та IsVectorShape()

Ці функції перевіряють, чи має матриця (вектор) вказану форму. Їх синтаксис такий:

```
IsMatrixShape(A,shape)
IsVectorShape(V,shape)
```

Тут A — матриця; V — вектор; shape — назва форми, на яку перевіряється об'єкт.

За замовчуванням система Maple розпізнає такі форми: zero, identity, scalar, scalar[x], diagonal, constant, constant[x], band[m], band[m,n], symmetric, antisymmetric (skewsymmetric), hermitian, antihermitian (skewhermitian), triangular, triangular, triangular[upper], triangular[lower], Hessenberg, Hessenberg[upper], Hessenberg[lower] для матриць та zero, unit, unit[j], scalar, scalar[j,x], constant, constant[x] для векторів. Зміст цих форм описаний у підрозділах "Функція Matrix()" та "Функція Vector()".

Зауважимо, що зазначені функції розпізнають форму, а не просто перевіряють значення параметра shape.

Розглянемо приклади.

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

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

```
> IsMatrix=
Shape(mA,triangular);
```

true

```
> IsMatrixShape(mA,band);
```

```
Error, (in IsMatrixShape/band)
band expects one or two inte-
ger indices
```

```
> IsMatrix-
Shape(mA,band[0,2]);
```

true

```
> IsMatrix-
Shape(mA,symmetric);
```

false

```
> vV:=Vector[row]([0,0,5]);
```

$$vV := [0, 0, 5]$$

```
> IsVector-
Shape(vV,constant);
```

false

```
> IsVectorShape(vV,scalar);
```

true

```
> IsVector-
Shape(vV,scalar[3,5]);
```

true

Крім стандартно передбачених форм користувач може задавати форми у вигляді процедур з назвами типу ‘IsMatrixShape/usershape’ або ‘IsVectorShape/usershape’. Розглянемо приклад. Квадратну матрицю називають нульпатентною другого порядку, якщо її квадрат дорівнює нульовій матриці.

```
> 'IsMatrixShape/nilpotent2':=proc(M)
evalb(type(M,'Matrix'(square)) and LinearAlgebra[Equal]
(M . M,LinearAlgebra[ZeroMatrix](LinearAlgebra[RowDimension]
(M))));
end:
```

```
> mB:=<<1|2>,<-2|1>>;
```

$$mB := \begin{bmatrix} 1 & 2 \\ -2 & 1 \end{bmatrix}$$

```
> mB.mB;
```

$$\begin{bmatrix} -3 & 4 \\ -4 & -3 \end{bmatrix}$$

```
> IsMatrixShape(mB,nilpotent2);
```

$$false$$

```
> mC:=<<0|1|1>,<0|0|0>,
<0|0|0>>;
```

$$mC := \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

```
> mC.mC;
```

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

```
> IsMatrixShape(mC,nilpotent2);
```

$$true$$

Перетворення типів

Як зауважувалось, система Maple має два базових типи для побудови матриць та векторів: table і rtable. Розглянемо процедуру конвертації типів, побудованих на основі table, у Matrix або Vector. Для цього використовуємо функцію convert, записану в одній з таких форм:

```
convert(A,Matrix,…)
convert(V,Vector,…)
convert(V,Vector[o],…)
```

Тут A, V — об'єкти, які потрібно конвертувати в типи відповідно Matrix та Vector; [o] — орієнтація вектора; … — додаткові параметри функції Matrix() або Vector().

```
> A := [[1,2],[3,4]];
```

$$A := [[1, 2], [3, 4]]$$

```
> whattype(A);
```

$$list$$

```
> mA:=convert(A, Matrix);
```

$$mA := \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

```
> whattype(mA);
```

$$Matrix$$

```
> V:=Array([0,0,1]);
```

$$V := [0, 0, 1]$$

```
> whattype(V);
```

$$Array$$

```
> vV:=convert(V,Vector[column],
shape=unit[3]);
```

$$vV := \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

```
> whattype(vV);
```

$$Vector_{column}$$

```
> VectorOptions(vV,shape);
```

$$[unit_3]$$

Робота з елементами матриць та векторів

Для виділення елементів матриць та векторів використовують команди відповідно M[L1,L2] та V[L]. Тут M — матриця; V — вектор; L,L1,L2 — цілі індекси або списки цілих індексів. При цьому допускаються діапазони індексів виду i..j, від'ємні індекси (–1 означає останній індекс, –2 — передостанній і т. д.).

Якщо L — один індекс, то V[L] повертає відповідну координату вектора як число. Якщо L — список, то результатом V[L] буде вектор, який складається з відповідних координат вектора V.

Для матриці якщо L1, L2 є індексами, то результатом буде (L1,L2)-елемент матриці. Якщо один з них індекс, а інший — список, отримаємо вектор. Якщо обидва параметри є списками, то результатом буде матриця.

Розглянемо приклади.

```
> vV:=Vector[row]
(10,i->i^2);
```

$$vV := [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]$$

```
> vV[5];
```

$$25$$

```
> vV[[5]];
```

$$[25]$$

```
> whattype(%);
```

$$Vector_{row}$$

```
> vV[[1,-1]];
```

$$[1, 100]$$

```
> vV[[2..-2]];
```

$$[4, 9, 16, 25, 36, 49, 64, 81]$$

```
> vV[[4..6]]:=
Vector[row]
(3,[0,0,0]);
```

$$vV_{[4\ ..\ 6]} := [0, 0, 0]$$

```
> vV;
```

$$[1, 4, 9, 0, 0, 0, 49, 64, 81, 100]$$

```
> mA:=Matrix(4,6,
(i,j)->i+2*j);
```

$$mA := \begin{bmatrix} 3 & 5 & 7 & 9 & 11 & 13 \\ 4 & 6 & 8 & 10 & 12 & 14 \\ 5 & 7 & 9 & 11 & 13 & 15 \\ 6 & 8 & 10 & 12 & 14 & 16 \end{bmatrix}$$

```
> mA[3,4];
```

$$11$$

```
> mA[[3],[4]];
```

$$[11]$$

```
> whattype(%);
```

$$Matrix$$

```
> mA[[3],4];
```

$$[11]$$

```
> whattype(%);
```

$$Vector_{column}$$

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

$$mA_{[2\,..\,-1],\,[3\,..\,-1]} := \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

```
> mA;
```

$$\begin{bmatrix} 3 & 5 & 7 & 9 & 11 & 13 \\ 4 & 6 & 0 & 0 & 0 & 0 \\ 5 & 7 & 0 & 0 & 0 & 0 \\ 6 & 8 & 0 & 0 & 0 & 0 \end{bmatrix}$$

```
> mA[1,[4,3,2,1]];
```

$$[9, 7, 5, 3]$$

Побудова стандартних матриць

Пакет LinearAlgebra надає низку функцій для побудови матриць базових форм, а саме: BandMatrix(), ConstantMatrix(), ConstantVector(), DiagonalMatrix(), IdentityMatrix(), JordanBlockMatrix(), ScalarMatrix(), ScalarVector(), UnitVector(), ZeroMatrix(), ZeroVector().

Всюди в цьому підрозділі параметр outputoptions=list задає опції orientation, readonly, shape, storage, order, datatype або attributes, які описані в пунктах “Функція Matrix()” та “Функція Vector()”. Параметр cpt виду compact=true або false відповідає за збереження вектора або матриці в пам’яті відповідно якомога компактніше згідно з їх структурою або як повністю заповнених.

Функція BandMatrix()

Ця функція призначена для побудови n-діагональних матриць. Її синтаксис такий:

```
BandMatrix(L,n,r,c,outputoptions=list)
```

Тут L — список list скалярів або списків list скалярів. У першому випадку кожний елемент списку визначає відповідну діагональ, заповнену цим елементом, у другому кожний список є відповідною діагоналлю матриці. Невід'ємне ціле число n вказує кількість діагоналей під головною, тобто головна діагональ визначається n+1-м скаляром або списком залежно від вигляду L. Кількість рядків та стовпців матриці задаються невід'ємними цілими числами відповідно r, c.

Розглянемо приклади.

```
> BandMatrix([[x],[y,y],[z]]);
```

$$\begin{bmatrix} y & z \\ x & y \end{bmatrix}$$

```
> BandMatrix([[x,x],[y,y,y],
[z,z,z]]);
```

$$\begin{bmatrix} y & z & 0 & 0 \\ x & y & z & 0 \\ 0 & x & y & z \end{bmatrix}$$

```
> BandMatrix([[x,x],[y,y,y],
[z,z,z]],2);
```

$$\begin{bmatrix} z & 0 & 0 \\ y & z & 0 \\ x & y & z \\ 0 & x & y \end{bmatrix}$$

```
> BandMatrix([[x,x],[y,y,y],[z,z,z]],
2,5,4);
```

$$\begin{bmatrix} z & 0 & 0 & 0 \\ y & z & 0 & 0 \\ x & y & z & 0 \\ 0 & x & y & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

```
> BandMatrix([x],0,3);
```

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

```
> BandMatrix([x,y],0,3);
```

$$\begin{bmatrix} x & y & 0 \\ 0 & x & y \\ 0 & 0 & x \end{bmatrix}$$

```
> BandMatrix([x,y],1,3);
```

$$\begin{bmatrix} y & 0 & 0 \\ x & y & 0 \\ 0 & x & y \end{bmatrix}$$

```
> A:=BandMatrix([x,y],1,3,4,
outputoptions=[readonly=true]);
```

$$A := \begin{bmatrix} y & 0 & 0 & 0 \\ x & y & 0 & 0 \\ 0 & x & y & 0 \end{bmatrix}$$

```
> A[1,1]:=1;
```

```
Error, cannot assign to
a read-only Matrix
```

Функція ConstantMatrix()

Ця функція заповнює матрицю вказаним значенням згідно з її структурою. Вона має такий синтаксис:

```
ConstantMatrix(s,r,c,cpt,outputoptions=list)
```

Тут s — значення, яким заповнюється матриця; r, c — кількість її відповідно рядків і стовпців.

Розглянемо приклади.

```
> ConstantMatrix(Pi,2,4);
```

$$\begin{bmatrix} \pi & \pi & \pi & \pi \\ \pi & \pi & \pi & \pi \end{bmatrix}$$

```
> ConstantMatrix(a,4,4, outputop-
tions=[shape=Hessenberg[upper]]);
```

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

```
> ConstantMatrix(5,2,2, outputop-
tions=[shape=antisymmetric]);
```

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

Функція ConstantVector()

Синтаксис цієї функції, яка заповнює вектор вказаним значенням, такий:

```
ConstantVector[o](s,d,cpt,outputoptions=list)
```

Тут s — значення, яким заповнюється вектор; d — розмірність вектора; [o] — параметр виду [row] або [column], який задає відповідно вектор-рядок або вектор-стовпець (за замовчуванням використовується орієнтація у стовпець).

Розглянемо приклади.

```
> ConstantVector(1,2);
```

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

```
> ConstantVector[row](1/2,5);
```

$$\begin{bmatrix} \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix}$$

```
> vA:=ConstantVector[row](5,3,
outputoptions=[readonly=true]);
```

$$vA := [5, 5, 5]$$

```
> vA[1]:=0;
```

```
Error, cannot assign to a
    read-only Vector
```

Функція DiagonalMatrix()

Ця функція створює матрицю, заповнену в напрямку головної діагоналі блоками значень. Вона має такий синтаксис:

```
DiagonalMatrix(V,r,c,outputoptions=list)
```

Тут V — вектор або список блоків, якими заповнюється діагональ; r, c — розміри матриці відповідно за рядками та стовпцями.

Скалярним елементам у V відповідають блоки 1x1. Блоки значень розміщуються вздовж головної діагоналі починаючи з лівого верхнього кута матриці.

```
> vV:=Vector[row] (3,[1,5,25]);
```

$$vV := [1, 5, 25]$$

```
> DiagonalMatrix(vV);
```

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 25 \end{bmatrix}$$

```
> DiagonalMatrix([10,100]);
```

$$\begin{bmatrix} 10 & 0 \\ 0 & 100 \end{bmatrix}$$

```
> listV:=[<1,2>,
<3|4>,10,<<-1|-2>,
<-3|-4>>];
```

$$listV := \left[\begin{bmatrix} 1 \\ 2 \end{bmatrix}, [3, 4], 10, \begin{bmatrix} -1 & -2 \\ -3 & -4 \end{bmatrix} \right]$$

```
> DiagonalMatrix(listV,6,8);
```

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 10 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & -2 & 0 & 0 \\ 0 & 0 & 0 & 0 & -3 & -4 & 0 & 0 \end{bmatrix}$$

Функція IdentityMatrix()

Ця функція створює прямокутну матрицю, на головній діагоналі якої розміщені 1, а решта елементів дорівнюють нулю. Її синтаксис такий:

```
IdentityMatrix(r,c,cpt,outputoptions=list)
```

Тут r, c — розміри матриці відповідно за рядками та стовпцями. Якщо параметр c опущено, створюється одинична матриця порядку r.

```
> IdentityMatrix(3);
```

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

```
> MatrixOptions(%,storage);
```

empty

```
> IdentityMatrix(2,4,compact=false);
```

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

```
> MatrixOptions(%,storage);
```

rectangular

JordanBlockMatrix()

Ця функція призначена для побудови жорданових матриць. Її синтаксис такий:

```
JordanBlockMatrix(K,d,outputoptions=list)
```

Тут K — це список впорядкованих пар виду [a,b]; а задає власне значення жорданової клітки, b — її розмір, d — параметр, що визначає порядок матриці.

```
> JordanBlockMatrix([[0,2],[10,1],[-5,2]],6);
```

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & -5 & 1 & 0 \\ 0 & 0 & 0 & 0 & -5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

```
> MatrixOptions(%,shape);
```

$$[band_{0,\,1}]$$

Функція ScalarMatrix()

Ця функція створює скалярну матрицю. Її синтаксис такий:

```
ScalarMatrix(s,r,c,cpt,outputoptions=list)
```

Тут s — скаляр, яким заповнюється головна діагональ; r, c — кількість відповідно рядків та стовпців матриці.

```
> ScalarMatrix(lambda, 2);
```

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

```
> ScalarMatrix(5,2,3);
```

$$\begin{bmatrix} 5 & 0 & 0 \\ 0 & 5 & 0 \end{bmatrix}$$

```
> MatrixOptions(%,shape);
```

$$[scalar_5]$$

Функція ScalarVector()

Ця функція створює скалярний вектор, тобто вектор, усі координати якого, за винятком заданого, дорівнюють нулю. Її синтаксис такий:

```
ScalarVector[o](s,i,d,cpt,outputoptions=list)
```

Тут d — розмірність вектора; i — ненульова координата; s — її значення.

Параметр [o] має вид [row] або [column] і задає відповідно вектор-рядок або вектор-стовпець.

```
> ScalarVector[row](Pi,5,10);
```

$$[0, 0, 0, 0, \pi, 0, 0, 0, 0, 0]$$

```
> VectorOptions(%,shape);
```

$$[scalar_{5, \pi}]$$

```
> ScalarVec-
tor(1000,1,2,compact=false);
```

$$\begin{bmatrix} 1000 \\ 0 \end{bmatrix}$$

```
> VectorOptions(%,shape);
```

$$[]$$

Функція UnitVector()

Ця функція задає одиничний вектор розмірності. Її синтаксис такий:

```
UnitVector[o](i,d,cpt,outputoptions=list)
```

Тут d — розмірність вектора; i — координата, i = 1.

Параметр [o] має вигляд [row] або [column] і задає відповідно вектор-рядок або вектор-стовпець.

```
> UnitVector[row](8,10);
```

$$[0, 0, 0, 0, 0, 0, 0, 1, 0, 0]$$

```
> VectorOptions(%, shape);
```

$$[unit_8]$$

```
> UnitVector(2,2,compact=false);
```

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

```
> VectorOptions(%, shape);
```

$$[]$$

Функція ZeroMatrix()

Ця функція створює нульову матрицю заданого розміру. Її синтаксис такий:

```
ZeroMatrix(r,c,cpt,outputoptions=list)
```

Наприклад:

```
> ZeroMatrix(2);
```

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

```
> ZeroMatrix(2,3);
```

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

```
> MatrixOptions(%,shape);
```

$$[zero]$$

Функція ZeroVector()

Ця функція створює нульовий вектор заданого розміру. Її синтаксис такий:

```
ZeroVector[o](d,cpt,outputoptions=list)
```

Наприклад:

```
> ZeroVector[row](3);
```

$$[0, 0, 0]$$

Арифметика матриць та векторів

У цьому підрозділі параметр outputoptions=list задає опції orientation, readonly, shape, storage, order, datatype або attributes, які описано в пунктах “Функція Matrix()” та “Функція Vector()”. Параметр cpt виду compact=true або false відповідає за збереження вектора або матриці в пам’яті відповідно якомога компактніше згідно з їх структурою або як повністю заповнених. Параметр ip має вид inplace=true або inplace=false (за замовчуванням). У першому випадку результат записується на місце першого аргумента, а у другому створюється новий об’єкт.

Додавання і віднімання матриць та векторів

Найпростіше додавати або віднімати матриці (однакової розмірності) та вектори (однакової розмірності і орієнтації), використовуючи знаки “+” та “–”. Наприклад:

```
> mA:=Matrix(2,{(1,2)=5},
shape=antihermitian);
```

$$mA := \begin{bmatrix} 0 & 5 \\ -5 & 0 \end{bmatrix}$$

```
> va:=Vector(2,[10,100]):mB:= Ma-
trix(2,va,shape=diagonal);
```

$$mB := \begin{bmatrix} 10 & 0 \\ 0 & 100 \end{bmatrix}$$

```
> mA+mB;
```

$$\begin{bmatrix} 10 & 5 \\ -5 & 100 \end{bmatrix}$$

```
> mA-mB;
```

$$\begin{bmatrix} -10 & 5 \\ -5 & -100 \end{bmatrix}$$

```
> Matrix(2,shape=constant[1000]);mA+%;
```

$$\begin{bmatrix} 1000 & 1005 \\ 995 & 1000 \end{bmatrix}$$

```
> vb:=Vector[row](3,[1,2,3]);
```

$$vb := [1, 2, 3]$$

```
> vc:=<100|100|100>;
```

$$vc := [100, 100, 100]$$

```
> vb+vc;
```

$$[101, 102, 103]$$

```
> vc-vb;
```

$$[99, 98, 97]$$

```
> vd:=Vector[row](4,shape=constant[1]);
```

$$vd := [1, 1, 1, 1]$$

```
> vd+vb;
```

```
Error, (in
rtable/Sum) inva-
lid arguments
```

```
> ve:=Vector[column](3,shape= con-
stant[1]);
```

$$ve := \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

```
> ve+vb;          Error, (in rtable/Sum) invalid arguments
```

Множення матриць та векторів на скаляр

Для того щоб помножити скаляр на матрицю або вектор, можна використати знак “*”. Наприклад:

```
> mA1:=<<1|2>,<3|4>>;
```

$$mA1 := \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

```
> 10*mA1;
```

$$\begin{bmatrix} 10 & 20 \\ 30 & 40 \end{bmatrix}$$

```
> mA1*10;
```

$$\begin{bmatrix} 10 & 20 \\ 30 & 40 \end{bmatrix}$$

```
> va1:=<1|2|3>;
```

$$va1 := [1, 2, 3]$$

```
> lambda*va1;
```

$$[\lambda, 2\lambda, 3\lambda]$$

```
> va1*lambda;
```

$$[\lambda, 2\lambda, 3\lambda]$$

Лінійні комбінації матриць або векторів

Якщо A, B — дві матриці (або вектори) однакової розмірності (і орієнтації), c1, c2 — скаляри, то для обчислення лінійної комбінації c1A+c2B можна використовувати не лише команду c1*A+c2*B, а й дещо потужнішу функцію Add. Синтаксис цієї функції такий:

```
Add(A,B,c1,c2,ip,outputoptions=list)
```

Тут A, B – дві матриці (вектори або скаляри) однакового розміру; c1, c2 — скаляри (за замовчуванням c1 = 1, c2 = 1).

Наприклад:

```
> ve1:=UnitVector(1,2);
```

$$ve1 := \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

```
> ve2:=UnitVector(2,2);
```

$$ve2 := \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

```
> Add(ve1,ve2);
```

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

```
> Add(ve1,ve2,10,20);ve1;
```

$$\begin{bmatrix} 10 \\ 20 \end{bmatrix} \qquad \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

```
> Add(ve1,ve2,10,20,inplace=true);ve1;
```

$$\begin{bmatrix} 10 \\ 20 \end{bmatrix} \qquad \begin{bmatrix} 10 \\ 20 \end{bmatrix}$$

У команді Add() допускається, щоб один з параметрів A, B був скаляром, а інший матрицею. У такому разі вважається, що параметр-скаляр задає відповідну скалярну матрицю.

Наприклад:

```
> mC:=Matrix(2,2,10,shape= triangu-
lar[upper]);
```

$$mC := \begin{bmatrix} 10 & 10 \\ 0 & 10 \end{bmatrix}$$

```
> Add(2,mC);
```

$$\begin{bmatrix} 12 & 10 \\ 0 & 12 \end{bmatrix}$$

Крім функції Add() пакет LinearAlgebra надає ще дві функції: MatrixAdd() для обчислення лінійної комбінації матриць та VectorAdd() — для обчислення лінійної комбінації векторів. Їх синтаксис такий:

```
MatrixAdd(A,B,c1,c2,ip,outputoptions=list)
VectorAdd(U,V,c1,c2,ip,outputoptions=list)
```

Тут перші два параметри — відповідно матриці чи вектори, інші параметри та дія цих функцій такі самі, як і функції Add().

Наприклад:

```
> mA2:=<<1|2|3>,<4|5|6>>; mB2:=ConstantMatrix(1,2,3);
```

$$mA2 := \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \qquad mB2 := \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

```
> MatrixAdd(mA2,mB2,100,5);
```

$$\begin{bmatrix} 105 & 205 & 305 \\ 405 & 505 & 605 \end{bmatrix}$$

```
> vV1:=<1,10>;vU1:=<5,5>;
```

$$vV1 := \begin{bmatrix} 1 \\ 10 \end{bmatrix} \qquad vU1 := \begin{bmatrix} 5 \\ 5 \end{bmatrix}$$

```
> VectorAdd(vV1,vU1,2,100);
```

$$\begin{bmatrix} 502 \\ 520 \end{bmatrix}$$

Множення матриць та векторів

Над матрицями та векторами можна виконувати операцію некомутативного множення. У системі Maple вона задається знаком «.» Нагадаємо, що необхідною умовою при такому множенні двох об'єктів (матриць або векторів) є рівність кількості стовпців першого об'єкта і рядків другого. Для того щоб відрізнити оператор «.» від десяткової крапки, прийнято обрамляти його символами пробілу.

Для некомутативного множення в пакеті LinearAlgebra наявні ще кілька функцій:

MatrixMatrixMultiply(A,B,ip,outputoptions=list)

для множення матриці A на матрицю B,

MatrixVectorMultiply(A,U,outputoptions=list)

— матриці A на вектор U,

VectorMatrixMultiply(V,A,outputoptions=list)

— вектора V на матрицю A,

MatrixScalarMultiply(A,s,ip,outputoptions=list)

— скаляра s на матрицю A,

VectorScalarMultiply(V,s,ip,outputoptions=list)

— скаляра s на вектор V,

ScalarMultiply(A,s,ip,outputoptions=list)

— скаляра s на матрицю або вектор A,

OuterProductMatrix(U,V,cpt,outputoptions=list)

— вектор-стовпчика U на вектор-рядок V.

Крім того, є функція Multiply(), яка узагальнює всі перелічені випадки. Її синтаксис такий:

Multiply(A,B,ip,outputoptions=list)

Приклади:

```
> mX:=BandMatrix([[1,2],[3,4]],0);
```

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

```
> mY:=<<6,5,4>|<3,2,1>>;
```

$$mY := \begin{bmatrix} 6 & 3 \\ 5 & 2 \\ 4 & 1 \end{bmatrix}$$

```
> mX . mY;
```

$$\begin{bmatrix} 21 & 9 \\ 26 & 8 \end{bmatrix}$$

```
> with(LinearAlgebra): Multi-
ply(mX,mY);
```

$$\begin{bmatrix} 21 & 9 \\ 26 & 8 \end{bmatrix}$$

```
> MatrixMatrixMultiply(mX,mY);
```

$$\begin{bmatrix} 21 & 9 \\ 26 & 8 \end{bmatrix}$$

```
> mYX:=mY . mX;
```

$$mYX := \begin{bmatrix} 6 & 24 & 12 \\ 5 & 19 & 8 \\ 4 & 14 & 4 \end{bmatrix}$$

```
> mY . mYX;
```

Error, (in MatrixMatrixMultiply) first matrix column dimension (2) <> second matrix row dimension (3)

```
> vU:=<1,2,3>;
```

$$vU := \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

```
> mYX . vU;
```

$$\begin{bmatrix} 90 \\ 67 \\ 44 \end{bmatrix}$$

```
> MatrixVectorMultiply(mYX,vU);
```

$$\begin{bmatrix} 90 \\ 67 \\ 44 \end{bmatrix}$$

```
> vV:=Vector[row](3,[1,2,3]);
```

$$vV := [1, 2, 3]$$

```
> vV . mYX;
```

$$[28, 104, 40]$$

```
> VectorMatrixMultiply(vV,mYX);
```

$$[28, 104, 40]$$

```
> vU . mYX;
```

Error, (in VectorMatrixMultiply) invalid input: VectorMatrixMultiply expects its 1st argument, v, to be of type Vector[row] but received Vector[Column](1..3, [123] , datatype = Anything, storage = rectangular, order = C_order)

```
> 5 . vV;
```

$$[5, 10, 15]$$

```
> VectorScalarMultiply(vV,5);
```

$$[5, 10, 15]$$

```
> 10 . mYX;
```

$$\begin{bmatrix} 60 & 240 & 120 \\ 50 & 190 & 80 \\ 40 & 140 & 40 \end{bmatrix}$$

```
> MatrixScalarMultiply(mYX,10);
```

$$\begin{bmatrix} 60 & 240 & 120 \\ 50 & 190 & 80 \\ 40 & 140 & 40 \end{bmatrix}$$

```
> vV1:=<1,2>;vU1:=<10|20>;
```

$$vV1 := \begin{bmatrix} 1 \\ 2 \end{bmatrix} \qquad vU1 := [10, 20]$$

```
> vV1 . vU1;
```

$$\begin{bmatrix} 10 & 20 \\ 20 & 40 \end{bmatrix}$$

```
> OuterProductMatrix(vV1,vU1);
```

$$\begin{bmatrix} 10 & 20 \\ 20 & 40 \end{bmatrix}$$

Зауважимо, що до об'єктів типу Array можна застосовувати також операцію поелементного (комутативного) множення "*". Для об'єктів типу Matrix чи Vector ця операція вважатиметься помилковою.

Приклади:

```
> aA:=Array([[1,2],[3,4]]);
```

$$aA := \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

```
> aB:=Array([[1,10],
[100,1000]]);
```

$$aB := \begin{bmatrix} 1 & 10 \\ 100 & 1000 \end{bmatrix}$$

```
> whattype(aB);
```

$$Array$$

```
> aA*aB;
```

$$\begin{bmatrix} 1 & 20 \\ 300 & 4000 \end{bmatrix}$$

```
> mA:=<<1|2>,<3|4>>;
```

$$mA := \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

```
> mB:=<<1|10>,<100|1000>>;
```

$$mB := \begin{bmatrix} 1 & 10 \\ 100 & 1000 \end{bmatrix}$$

```
> whattype(mB);
```

Matrix

```
> mA*mB;
```

Error, (in rtable/Product) invalid arguments

Векторна алгебра

У цьому підрозділі параметр outputoptions=list задає опції orientation, readonly, shape, storage, order, datatype або attributes, які описано в пункті "Функція Vector()". Параметр ip має вид inplace=true або inplace=false (за замовчуванням). У першому випадку результат записується на місце першого аргумента, а у другому створюється новий об'єкт. Параметр conj виду conjugate=value використовується у функціях, що посилаються на обчислення скалярного добутку. Якщо conjugate=false, обчислюється дійсний скалярний добуток, в іншому випадку — комплексний. За замовчуванням використовується conjugate=true.

Скалярний добуток

У пакеті LinearAlgebra міститься функція для обчислення скалярного добутку двох векторів однакової розмірності. Її синтаксис такий:

```
DotProduct(U,V,conj)
```

Тут U,V — вектори; value — true (за замовчуванням) або false. При conjugate=true, якщо U — вектор-стовпець, то комплексне спряження у скалярному добутку застосовується до U, в іншому випадку — до V.

Приклади:

```
> vU:=Vector([x,y]);
```

$$vU := \begin{bmatrix} x \\ y \end{bmatrix}$$

```
> vV:=Vector([z,t]);
```

$$vV := \begin{bmatrix} z \\ t \end{bmatrix}$$

```
> DotProduct(vU,vV);
```
$$\bar{x}\,z+\bar{y}\,t$$

```
> vV:=Vector[row]([z,t]);
```
$$vV := [z, t]$$

```
> DotProduct(vU,vV);
```
$$\bar{x}\,z+\bar{y}\,t$$

```
> vU:=Vector[row]([x,y]);
```
$$vU := [x, y]$$

```
> DotProduct(vU,vV);
```
$$\bar{z}\,x+\bar{t}\,y$$

```
> DotProduct(vU,vV,conjugate=false);
```
$$z\,x+t\,y$$

Векторний добуток

Для обчислення векторного добутку тривимірних векторів пакет LinearAlgebra пропонує функцію CrossProduct(). Її синтаксис такий:

```
CrossProduct(U,V,outputoptions=list)
```

Тут U, V — тривимірні вектори. Якщо вони обидва є векторами-рядками, таким самим буде й результат, в іншому випадку результатом є вектор-стовпець.

Аналогом функції CrossProduct() є операція &x, так само доступна з пакета LinearAlgebra.

Приклади:

```
> vV:=Vector[row]([1,2,3]);
```
$$vV := [1, 2, 3]$$

```
> vU:=Vector[row]([2,4,6]);
```
$$vU := [2, 4, 6]$$

```
> CrossProduct(vV,vU);
```
$$[0, 0, 0]$$

```
> vW:=Vector([3,-1,5]);
```
$$vW := \begin{bmatrix} 3 \\ -1 \\ 5 \end{bmatrix}$$

```
> CrossProduct(vV,vW);
```
$$\begin{bmatrix} 13 \\ 4 \\ -7 \end{bmatrix}$$

```
> vV &x vW;
```
$$\begin{bmatrix} 13 \\ 4 \\ -7 \end{bmatrix}$$

Кут між векторами

Обчислити кут між двома векторами U, V однакового розміру можна за допомогою функції VectorAngle(). Її синтаксис такий:

```
VectorAngle(U,V,conj)
```

Наприклад:

```
> vV:=Vector[row]([1,-2]);
```

$$vV := [1, -2]$$

```
> vU:=Vector[row]([6,3]);
```

$$vU := [6, 3]$$

```
> VectorAngle(vV,vU);
```

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

```
> vV:=Vector[row]([1,I,1+I]);
```

$$vV := [1, I, 1 + I]$$

```
> vU:=Vector[row]([I,-1,0]);
```

$$vU := [I, -1, 0]$$

```
> VectorAngle(vV,vU);
```

$$\frac{1}{2}\pi + I\,\mathrm{arcsinh}\left(\frac{1}{2}\sqrt{2}\right)$$

```
> vX:=Vector([2,x]);
```

$$vX := \begin{bmatrix} 2 \\ x \end{bmatrix}$$

```
> vY:=Vector([0,1]);
```

$$vY := \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

```
> VectorAngle(vX,vY);
```

$$\arccos\left(\frac{\overline{x}}{\sqrt{4+|x|^2}}\right)$$

```
> VectorAngle(vX,vY,conjugate=false);
```

$$\arccos\left(\frac{x}{\sqrt{4+x^2}}\right)$$

Норма вектора

У пакеті LinearAlgebra міститься функція VectorNorm(), за допомогою якої можна обчислити p-норму вектора. Нагадаємо, що p-норма визначена для значень p від одиниці до нескінченності. Розглядувана функція має такий синтаксис:

```
VectorNorm(V,p,conj)
```

Тут V — вектор; p — норма.

Система Maple не забороняє використовувати значення p між 0 і 1, хоча в цьому разі обчислюється не p-норма, а p-відстань від вектора до початку координат.

Випадок p = 2 має синоніми Frobenius і Euclidean.

Випадок $p = \infty$ задається константою infinity. Нескінченність-норма обчислюється також, якщо значення p опущено.

Аналогічно VectorNorm() працює функція Norm().

Приклади:

```
> v_a:=Vector[row]([x,y]);
```

$$v\_a := [x, y]$$

```
> VectorNorm(v_a,2);
```

$$\sqrt{|x|^2 + |y|^2}$$

```
> VectorNorm(v_a,Frobenius);
```

$$\sqrt{|x|^2 + |y|^2}$$

```
> VectorNorm(v_a,Euclidean);
```

$$\sqrt{|x|^2 + |y|^2}$$

```
> VectorNorm(v_a,2,conjugate=false);
```

$$\sqrt{x^2 + y^2}$$

```
> VectorNorm(v_a,10);
```

$$\left(|x|^{10} + |y|^{10}\right)^{(1/10)}$$

```
> VectorNorm(v_a,infinity);
```

$$\max(|x|, |y|)$$

```
> VectorNorm(v_a);
```

$$\max(|x|, |y|)$$

```
> Norm(v_a,infinity);
```

$$\max(|x|, |y|)$$

```
> VectorNorm(Vector([1,-2+I,I,5, 8-I]),0.25);
```

$$6.403248671$$

Нормалізація вектора

Пакет LinearAlgebra містить функцію Normalize(), що призначена для нормалізації вектора за його p-нормою. Синтаксис цієї функції такий:

```
Normalize(V,p,ip,conj,outputoptions=list)
```

Тут V — вектор; p — норма. Значення параметра p можуть бути такі, як у функції VectorNorm() (див. пункт “Норма вектора”).

Параметри ip та outputoptions=list взаємовиключні.

Приклади:

```
> vX:=Vector[row]([4,-1]);
```

$$vX := [4, -1]$$

```
> Normalize(vX,2,inplace=true);
```

$$\left[\frac{4}{17}\sqrt{17}, -\frac{1}{17}\sqrt{17}\right]$$

```
> vX;
```

$$\left[\frac{4}{17}\sqrt{17}, -\frac{1}{17}\sqrt{17}\right]$$

```
> VectorNorm(vX,2);
```

$$1$$

```
> VectorNorm(vX,3);
```

$$\frac{1}{289}65^{(1/3)}\,289^{(2/3)}\,17^{(1/6)}$$

```
> vY:=Normalize(vX,3,
outputoptions=
[readonly=true]);
```

$$vY := \left[\frac{4}{1105}65^{(2/3)}\,289^{(1/3)}\,17^{(1/3)}, -\frac{1}{1105}65\right.$$

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

$$[0.9948452691, -.2487113173]$$

```
> VectorNorm(vY,3);
```

$$1$$

Властивості матриць

У цьому підрозділі параметр outputoptions=list задає опції orientation, readonly, shape, storage, order, datatype або attributes, які описані в пункті “Функція Matrix()”. Параметр cpt виду compact=true або false відповідає за збереження матриці в пам’яті відповідно якомога компактніше згідно з її структурою або як повністю заповненої. Параметр ip має вид inplace=true або inplace=false (за замовчуванням). У першому випадку результат записується на місце першого аргумента, а у другому — створюється новий об’єкт.

Операція транспонування

Пакет LinearAlgebra надає можливість швидко знайти транспоновану матрицю за допомогою функції Transpose(). Її синтаксис такий:

```
Transpose(A,ip,outputoptions=list)
```

Тут A — матриця, яку потрібно транспонувати.

Наприклад:

```
> mA:=Matrix([[1,3,5],[2,4,6]]);
```

$$mA := \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}$$

```
> Transpose(mA);
```

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$$

Комплексно спряжене транспонування

Ця операція використовується за допомогою функції Hermitian-Transpose(). Її синтаксис такий:

```
HermitianTranspose(A,ip,outputoptions=list)
```

Тут A — матриця, яку потрібно транспонувати.

Наприклад:

```
> mC:=<<1+I|2+10*I>,<3|4-100*I>>;
```

$$mC := \begin{bmatrix} 1+I & 2+10\,I \\ 3 & 4-100\,I \end{bmatrix}$$

```
> HermitianTranspose(mC,inplace=true):
> mC;
```

$$\begin{bmatrix} 1-I & 3 \\ 2-10\,I & 4+100\,I \end{bmatrix}$$

Визначник матриці

Для знаходження визначника квадратної матриці призначена функція Determinant(). Її синтаксис такий:

```
Determinant(A,method=value)
```

Тут A — матриця, а другий параметр задає метод обчислення визначника.

Не вдаючись у деталі, зауважимо, наприклад, що метод integer призначений для обчислення визначника цілочисельної матриці, rational — матриці з раціональними елементами, float видасть результат у дійсній формі, modular[m] обчислює визначник за модулем числа m.

Приклади:

```
> mA:=Matrix([[1,1/2,1/3],[1/4,0,1],[3,1/7,1/13]]);
```

$$mA := \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} \\ \frac{1}{4} & 0 & 1 \\ 3 & \frac{1}{7} & \frac{1}{13} \end{bmatrix}$$

```
> Determinant(mA);
```

$$\frac{2969}{2184}$$

```
> Determinant(mA,method=float);
```

$$1.359432235$$

```
> mB:=<<10|3>,<-2|5>>;
```

$$mB := \begin{bmatrix} 10 & 3 \\ -2 & 5 \end{bmatrix}$$

```
> Determinant(mB);
```

$$56$$

```
> Determinant(mB,method=modular[5]);
```

$$1$$

Мінори

У пакеті LinearAlgebra міститься також функція Minor() для обчислення мінорів матриці. Її синтаксис такий:

```
Minor(A,r,c,output=value,method=value, outputoptions=list)
```

Ця функція для заданої матриці A вилучає r-й рядок та c-й стовпець і залежно від значення matrix чи determinant параметра output повертає відповідно матрицю чи значення мінору. При цьому якщо output=matrix, можна задати додаткові параметри результату за допомогою outputoptions=list, а якщо output=determinant — метод обчислення мінору (див. пункт “Визначник матриці”).

Приклади:

```
> mA:=Matrix([[5,1,2],[-3,9,2],
[1,-1,7]]);
```

$$mA := \begin{bmatrix} 5 & 1 & 2 \\ -3 & 9 & 2 \\ 1 & -1 & 7 \end{bmatrix}$$

```
> Minor(mA,2,2,output=matrix);
```

$$\begin{bmatrix} 5 & 2 \\ 1 & 7 \end{bmatrix}$$

```
> Minor(mA,2,2,output=determinant);
```

33

```
>Minor(mA,2,2,output=determinant,method=
modular[3]);
```

0

Ранг матриці

Для обчислення рангу матриці M призначена функція `Rank(M);`

Наприклад:

```
> mA:=Matrix(3,[1,2,3], scan=diagonal);
```

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

```
> Rank(mA);
```

3

```
> mB:=Matrix(3,fill=10);
```

$$mB := \begin{bmatrix} 10 & 10 & 10 \\ 10 & 10 & 10 \\ 10 & 10 & 10 \end{bmatrix}$$

```
> Rank(mB);
```

1

```
> mB[1,1]:=100;
```

$mB_{1,1} := 100$

```
> Rank(mB);
```

2

Обернена матриця

Для знаходження оберненої матриці в пакеті LinearAlgebra передбачена потужна функція MatrixInverse(). На жаль, через обмеженість обсягу немає можливості детально проаналізувати цю функцію. Зауважимо лише, що її параметрами є матриця, для якої потрібно знайти обернену, метод пошуку та його опції, список властивостей матриці результату та ін. Детальніше це описано у спеціальній літературі.

За замовчуванням, якщо матриця A (параметр функції) є квадратною і невиродженою, шукається її обернена матриця A^{-1}. Якщо матриця прямокутна або задано метод pseudo, шукається так звана інверсна мат-

риця, тобто матриця X, яка задовольняє рівності AXA=A і XAX=X. При цьому матриці AX і XA повинні бути ермітові.

Щодо припустимих методів, то для цілочисельної матриці доцільно використовувати метод integer, для раціонально чисельної — univar, для трикутної — subs.

Приклади:

```
> mS:=<<a,c>, <b,d>>:
```

```
> MatrixInverse(%);
```

$$\begin{bmatrix} \frac{d}{a\,d-b\,c} & -\frac{b}{a\,d-b\,c} \\ -\frac{c}{a\,d-b\,c} & \frac{a}{a\,d-b\,c} \end{bmatrix}$$

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

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

```
> mAinv:=MatrixInverse
(mA, method=integer);
```

$$mAinv := \begin{bmatrix} \frac{-1}{21} & \frac{1}{21} & \frac{4}{21} & \frac{2}{21} \\ \frac{4}{21} & \frac{-4}{21} & \frac{-16}{21} & \frac{13}{21} \\ \frac{2}{63} & \frac{19}{63} & \frac{-8}{63} & \frac{-4}{63} \\ \frac{1}{7} & \frac{-1}{7} & \frac{3}{7} & \frac{-2}{7} \end{bmatrix}$$

```
> mA.mAinv;
```

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

```
> mB:=Matrix([[1,2,3],[4,5,6],
[5,7,9]]);
```

$$mB := \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 5 & 7 & 9 \end{bmatrix}$$

```
> MatrixInverse(mB);
```

Error, (in MatrixInverse) singular matrix

```
> mX:=MatrixInverse(mB,method=pseudo);
```

$$mX := \begin{bmatrix} \frac{-7}{9} & \frac{11}{18} & \frac{-1}{6} \\ \frac{-1}{9} & \frac{1}{9} & 0 \\ \frac{5}{9} & \frac{-7}{18} & \frac{1}{6} \end{bmatrix}$$

```
> mB.mX.mB=mB;
```

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 5 & 7 & 9 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 5 & 7 & 9 \end{bmatrix}$$

```
> mX.mB.mX=mX;
```

$$\begin{bmatrix} \frac{-7}{9} & \frac{11}{18} & \frac{-1}{6} \\ \frac{-1}{9} & \frac{1}{9} & 0 \\ \frac{5}{9} & \frac{-7}{18} & \frac{1}{6} \end{bmatrix} = \begin{bmatrix} \frac{-7}{9} & \frac{11}{18} & \frac{-1}{6} \\ \frac{-1}{9} & \frac{1}{9} & 0 \\ \frac{5}{9} & \frac{-7}{18} & \frac{1}{6} \end{bmatrix}$$

```
> mB.mX;
```

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

```
> mX.mB;
```

$$\begin{bmatrix} \frac{5}{6} & \frac{1}{3} & \frac{-1}{6} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{-1}{6} & \frac{1}{3} & \frac{5}{6} \end{bmatrix}$$

```
>mC:=Matrix([[1,10,100],
[0,2,2000],[0,0,3]]);
```

$$mC := \begin{bmatrix} 1 & 10 & 100 \\ 0 & 2 & 2000 \\ 0 & 0 & 3 \end{bmatrix}$$

```
> mCinv:=MatrixInverse
(mC,method=subs);
```

$$mCinv := \begin{bmatrix} 1 & -5 & 3300 \\ 0 & \frac{1}{2} & \frac{-1000}{3} \\ 0 & 0 & \frac{1}{3} \end{bmatrix}$$

```
> mC.mCinv;
```

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

Приєднана матриця

Для квадратної матриці A її приєднаною називають таку квадратну матрицю B, що $AB = \det AE$. Приєднану матрицю знаходять за допомогою функції Adjoint(). Її синтаксис такий:

```
Adjoint(A,outoptions=list)
```

Приклад:

```
> mA:=Matrix([[2,3,5],
[-1,5,7],[1,8,3]]);
```

$$mA := \begin{bmatrix} 2 & 3 & 5 \\ -1 & 5 & 7 \\ 1 & 8 & 3 \end{bmatrix}$$

```
> mB:=Adjoint(mA);
```

$$mB := \begin{bmatrix} -41 & 31 & -4 \\ 10 & 1 & -19 \\ -13 & -13 & 13 \end{bmatrix}$$

```
> mA.mB;
```

$$\begin{bmatrix} -117 & 0 & 0 \\ 0 & -117 & 0 \\ 0 & 0 & -117 \end{bmatrix}$$

```
> Determinant(mA);
```

$$-117$$

Слід матриці

Суму діагональних елементів квадратної матриці називають її слідом. Слід можна обчислити за допомогою функції Trace() пакета LinearAlgebra. Наприклад:

```
> mA:=Matrix([[5,10,20],[30,-2,40],[50,60,1]]);
```

$$mA := \begin{bmatrix} 5 & 10 & 20 \\ 30 & -2 & 40 \\ 50 & 60 & 1 \end{bmatrix}$$

```
> Trace(mA);
```

$$4$$

Перевірка ортогональності та унітарності

За допомогою пакета LineearAlgebra легко перевірити, чи є квадратна матриця ортогональною (унітарною). Нагадаємо, що квадратна матриця A називається ортогональною, якщо $AA^T=E$. Якщо добуток квадратної матриці на її комплексно спряжену дорівнює одиничній матриці, її називають унітарною. Функції IsOrthogonal() і IsUnitary() перевіряють, якою є матриця: ортогональною або унітарною. (Насправді ці функції дають змогу перевірити ортогональність і унітарність у загальнішому розумінні. Детальніше див. Help системи Maple.)

```
> mA:=Matrix(2,[[sqrt(3)/2,1/2],
[-1/2,sqrt(3)/2]]);
```

$$mA := \begin{bmatrix} \frac{1}{2}\sqrt{3} & \frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2}\sqrt{3} \end{bmatrix}$$

```
> IsOrthogonal(mA);
```

true

```
> mA . Transpose(mA);
```

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

```
> mB:=Matrix(2,[[sqrt(3)/2,1/2+I],
[-1/2+I,sqrt(3)/2]]);
```

$$mB := \begin{bmatrix} \frac{1}{2}\sqrt{3} & \frac{1}{2}+I \\ -\frac{1}{2}+I & \frac{1}{2}\sqrt{3} \end{bmatrix}$$

```
> mB . HermitianTranspose(mB);
```

$$\begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

```
> IsUnitary(mB);
```

false

Перевірка подібності матриць

Нагадаємо, що дві квадратні матриці називаються подібними, якщо відображають один і той самий оператор у різних базах. Алгебраїчно матриці A, B подібні, якщо існує матриця C така, що $CAC^{-1} = B$.

Для перевірки подібності матриць використовують функцію IsSimilar(). Її синтаксис такий:

```
IsSimilar(A,B,out,outputoptions[C]=list)
```

Тут out — параметр виду output=query, output=C або output=[query,C]. При цьому query означає перевірку подібності матриць A, B, а C — знаходження матриці C, яка фігурує в означенні подібності. Параметр outputoptions[C]=list задає опції orientation, readonly, shape, storage, order, datatype або attributes, які описані в пункті “Функція Matrix()”.

Приклади:

```
> mA:=Matrix(2,[[1,2],[3,4]]);
```

$$mA := \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

```
> mB:=Matrix(2,[[1,6],[1,5]]);
```

$$mB := \begin{bmatrix} 1 & 6 \\ 1 & 5 \end{bmatrix}$$

```
> IsSimilar(mA,mB,output=query);
```

$$false$$

```
> mB[2,2]:=4;
```

$$mB_{2,2} := 4$$

```
> IsSimilar(mA,mB,output=
[query,C]);
```

$$true, \begin{bmatrix} 1 & 0 \\ 0 & \frac{1}{3} \end{bmatrix}$$

```
> mC:=IsSimilar(mA,mB,output=C,
outputoptions[C]=[readonly=true]);
```

$$mC := \begin{bmatrix} 1 & 0 \\ 0 & \frac{1}{3} \end{bmatrix}$$

```
> mC.mA=mB.mC;
```

$$\begin{bmatrix} 1 & 2 \\ 1 & \frac{4}{3} \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 1 & \frac{4}{3} \end{bmatrix}$$

```
> mC[1,1]:=2;
```

Error, cannot assign to a read-only Matrix

Додатна та від’ємна визначеності матриць

Пакет LinearAlgebra дає змогу перевірити додатну чи від’ємну визначеність квадратної матриці. Нагадаємо, що симетрична (ермітова в комплексному випадку) квадратна матриця є додатно визначеною, якщо всі її власні значення перевищують нуль, додатно напіввизначеною, як-

що всі її власні значення невід'ємні, від'ємно визначеною при від'ємних та від'ємно напіввизначеною — при недодатних власних значеннях. Для цього використовують функцію

```
IsDefinite(A,query=attribute)
```

де A — квадратна матриця; attribute — одна з назв positive_definite, positive_semidefinite, negative_definite або negative_semidefinite. Значення positive_definite приймається за замовчуванням.

Приклади:

```
> mA:=Matrix(2,[[1,2],[2,5]]);
```

$$mA := \begin{bmatrix} 1 & 2 \\ 2 & 5 \end{bmatrix}$$

```
> IsDefinite(mA);
```

true

```
> IsDefinite(mA,query=positive_definite);
```

true

```
> mA[2,2]:=4;
```

$$mA_{2,2} := 4$$

```
> mA;
```

$$\begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix}$$

```
> IsDefinite(mA);
```

false

```
> IsDefinite(mA,query=positive_semidefinite);
```

true

```
> mB:=<<-1|0>,<0|0>>;
```

$$mB := \begin{bmatrix} -1 & 0 \\ 0 & 0 \end{bmatrix}$$

```
> IsDefinite(mB,query=negative_semidefinite);
```

true

```
> IsDefinite(mB,query=negative_definite);
```

false

Білінійна форма

Нагадаємо, що білінійною формою називають координатний запис функції двох аргументів-векторів U, V, що лінійний за кожним з аргументів. Цей запис у конкретній системі координат задається матрицею A білінійної форми. У припущенні, що U,V — вектори-стовпці, значення білінійної форми на них обчислюється як U^TAV. Пакет LinearAlgebra містить функцію BilinearForm(), яка обчислює значення білінійного відображення в заданих векторах за матрицею білінійної форми. Її синтаксис такий:

BilinearForm(U,V,A,conjugate=value)

Якщо вектор U — рядок, а V — стовпець, обчислення здійснюється за формулою UAV. Якщо V — рядок, обчислення здійснюється за формулою UA^TV^T або $U^TA^TV^T$ залежно від виду U.

Якщо conjugate=true (це значення за замовчуванням), транспонування є комплексно-спряженим (детальніше див. Help системи Maple).

Наприклад:

```
> mA:=Matrix([[3,-1],[2,5]]);
```

$$mA := \begin{bmatrix} 3 & -1 \\ 2 & 5 \end{bmatrix}$$

```
> vX:=Vector([x1,x2]);
```

$$vX := \begin{bmatrix} x1 \\ x2 \end{bmatrix}$$

```
> vY:=Vector([y1,y2]);
```

$$vY := \begin{bmatrix} y1 \\ y2 \end{bmatrix}$$

```
>a:=BilinearForm(vX,vY,mA,conjugate=false);
```

$$a := x1\,(3\,y1 - y2) + x2\,(2\,y1 + 5\,y2)$$

```
> b:=Transpose(vX) . mA . vY;
```

$$b := y1\,(3\,x1 + 2\,x2) + y2\,(-x1 + 5\,x2)$$

```
> simplify(a-b);
```

$$0$$

```
> mA:=Matrix([[1,-2],[-2,5]]);
```

$$mA := \begin{bmatrix} 1 & -2 \\ -2 & 5 \end{bmatrix}$$

```
> IsDefinite(mA,query=positive_definite);
```

$$true$$

```
>BilinearForm(vX,vX,mA,conjugate= false);
```

$$x1\,(x1 - 2\,x2) + x2\,(-2\,x1 + 5\,x2)$$

```
> expand(%);
```

$$x1^2 - 4\,x1\,x2 + 5\,x2^2$$

Функції від квадратних матриць

Якщо f(x) — аналітична* функція, має сенс вираз f(A), де A — квадратна матриця. Під f(A) розуміють підстановку A в ряд Тейлора функції f. Пакет LinearAlgebra містить функцію MatrixFunction(), яка виконує цю процедуру. Крім того, є дві часткові форми цієї функції MatrixPower() для обчислення степеневої функції від матриці та MatrixExponential() для обчислення e^A. Синтаксис цих функцій такий:

---

* Розкладається в абсолютно збіжний степеневий ряд.

```
MatrixFunction(A,F,x,outputoptions=list)
MatrixPower(A,n,outputoptions=list)
MatrixExponential(A,outputoptions=list)
```

Тут A — квадратна матриця; F — вираз, який задає функцію f; x — аргумент функції F; n — степінь, до якого потрібно піднести матрицю A.

Приклади:

```
> mA:=Matrix([[-5,-2],[21,8]]);
```

$$mA := \begin{bmatrix} -5 & -2 \\ 21 & 8 \end{bmatrix}$$

```
> Matrix=
Function(mA,ln(x),x);
```

$$\begin{bmatrix} 7\ln(1) - 6\ln(2) & -2\ln(2) + 2\ln(1) \\ 21\ln(2) - 21\ln(1) & -6\ln(1) + 7\ln(2) \end{bmatrix}$$

```
> MatrixFunction(mA,f(x),x);
```

$$\begin{bmatrix} 7f(1) - 6f(2) & -2f(2) + 2f(1) \\ 21f(2) - 21f(1) & -6f(1) + 7f(2) \end{bmatrix}$$

```
> MatrixFunction(mA,x^2,x);
```

$$\begin{bmatrix} -5 & -2 \\ 21 & 8 \end{bmatrix}^2$$

```
> %;
```

$$\begin{bmatrix} -17 & -6 \\ 63 & 22 \end{bmatrix}$$

```
> mA . mA;
```

$$\begin{bmatrix} -17 & -6 \\ 63 & 22 \end{bmatrix}$$

```
> MatrixPower(mA,10);
```

$$\begin{bmatrix} -6137 & -2046 \\ 21483 & 7162 \end{bmatrix}$$

```
> MatrixFunction(mA,x^10,x);
```

$$\begin{bmatrix} -5 & -2 \\ 21 & 8 \end{bmatrix}^{10}$$

```
> %;
```

$$\begin{bmatrix} -6137 & -2046 \\ 21483 & 7162 \end{bmatrix}$$

```
> Matrix=
Function(mA,exp(x),x);
```

$$\begin{bmatrix} 7e^{2}-6e^{2} & -2e^{2}+2e^{2} \\ 21e^{2}-21e^{2} & -6e^{2}+7e^{2} \end{bmatrix}$$

```
> MatrixExponential(mA);
```

$$\begin{bmatrix} 7e^{2}-6e^{2} & -2e^{2}+2e^{2} \\ 21e^{2}-21e^{2} & -6e^{2}+7e^{2} \end{bmatrix}$$

Лінійні простори та оператори

У цьому підрозділі параметр outputoptions=list задає опції orientation, readonly, shape, storage, order, datatype або attributes, які описані в пункті “Функція Vector()”. Значення цих опцій призначаються всім векторам результату.

База лінійного простору

Однією з найпоширеніших задач лінійної алгебри є знаходження бази лінійної оболонки, породженої цією системою векторів. Пакет LinearAlgebra містить функцію Basis(), яка розв’язує задану задачу. Синтаксис цієї функції такий:

```
Basis(V,outputoptions=list)
```

Тут V — вектор, список векторів або множина векторів, які породжують лінійну оболонку. Усі вектори у V повинні бути одного розміру та орієнтації.

Якщо V — список, результат так само повертається у вигляді списку, в іншому випадку — у вигляді множини базисних векторів.

Наприклад:

```
> v1:=Vector[row]([1,-1,3,5]);
```

$$v1 := [1, -1, 3, 5]$$

```
> v2:=Vector[row]([6,2,-3,2]);
```

$$v2 := [6, 2, -3, 2]$$

```
> v3:=Vector[row]([7,1,0,7]);
```

$$v3 := [7, 1, 0, 7]$$

```
> Basis([v1,v2,v3]);
```

$$[[1, -1, 3, 5], [6, 2, -3, 2]]$$

```
> Basis({v1,v2,v3});
```

$$\{[1, -1, 3, 5], [6, 2, -3, 2]\}$$

База суми лінійних підпросторів

Знайти базу суми лінійних просторів можна за допомогою функції SumBasis() пакета LinearAlgebra. Її синтаксис такий:

```
SumBasis(VS,outputoptions=list)
```

Тут VS — список, елементами якого є вектори, списки векторів або множини векторів, які породжують простори, базис суми яких шукається. Усі вектори повинні бути одного розміру та орієнтації.

Якщо всі елементи у VS — списки, результат так само повертається у вигляді списку, в іншому випадку — у вигляді множини базисних векторів.

Наприклад:

```
> a1:=Vector[row]([1,-1,2,1]);
```
$$a1 := [1, -1, 2, 1]$$
```
> a2:=Vector[row]([3,2,-1,5]);
```
$$a2 := [3, 2, -1, 5]$$
```
> b1:=Vector[row]([4,1,1,6]);
```
$$b1 := [4, 1, 1, 6]$$
```
> b2:=Vector[row]([1,3,7,-2]);
```
$$b2 := [1, 3, 7, -2]$$
```
> SumBasis([[a1,a2],
[b1,b2]]);
```
$$[[1, -1, 2, 1], [3, 2, -1, 5], [1, 3, 7, -2]]$$
```
> SumBasis([{a1,a2},
[b1,b2]]);
```
$$\{[3, 2, -1, 5], [4, 1, 1, 6], [1, 3, 7, -2]\}$$

База перетину лінійних просторів

Для знаходження бази перетину лінійних просторів використовують функцію IntersectionBasis() пакету LinearAlgebra. Її синтаксис такий:

```
IntersectionBasis(VS,outputoptions=list)
```

Тут VS — список, елементами якого є вектори, списки або множини векторів, які породжують простори, базис перетину яких шукається. Усі вектори повинні бути одного розміру та орієнтації.

Якщо всі елементи у VS — списки, результат так само повертається у вигляді списку, в іншому випадку — у вигляді множини базисних векторів.

Наприклад:

```
> a1:=Vector[row]([1,-1,1]);
```
$$a1 := [1, -1, 1]$$
```
> a2:=Vector[row]([5,1,2]);
```
$$a2 := [5, 1, 2]$$
```
> b1:=Vector[row]([3,7,-1]);
```
$$b1 := [3, 7, -1]$$
```
> b2:=Vector[row]([2,1,-2]);
```
$$b2 := [2, 1, -2]$$
```
> IntersectionBasis([[a1,a2],[b1,b2]]);
```
$$\left[\left[\frac{19}{2}, \frac{37}{2}, \frac{-9}{2}\right]\right]$$
```
> IntersectionBasis([{a1,a2},[b1,b2]]);
```
$$\left\{\left[\frac{19}{2}, \frac{37}{2}, \frac{-9}{2}\right]\right\}$$

Бази просторів рядків та стовпців матриці

Базу простору, породженого векторами рядків (стовпців) матриці, можна знайти за допомогою функції RowSpace() (відповідно ColumnSpace()) пакета LinearAlgebra. Результатом виконання цих функцій є список канонічних векторів відповідної бази. Їх синтаксис такий:

```
RowSpace(A,outputoptions=list)
ColumnSpace(A,outputoptions=list)
```

де A — задана матриця.

Наприклад:

```
> mA:=Matrix(4,6,[[2,-1,2,1,1,4],[0,2,0,
-1,2,2],[1,2,1,3,1,0],[3,1,3,4,2,4]]);
```

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

```
> RowSpace(mA);
```

$$\left[\left[1, 0, 1, 0, \frac{17}{15}, \frac{14}{5} \right], \left[0, 1, 0, 0, \frac{11}{15}, \frac{2}{5} \right], \left[0, 0, 0, 1, \frac{-8}{15}, \frac{-6}{5} \right] \right]$$

```
> ColumnSpace(mA);
```

$$\left[\begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} \right]$$

Ортогоналізація системи векторів

Якщо розглядаємо евклідовий (або унітарний) лінійний простір, породжений деякою системою векторів, природно поставити задачу опису цього простору в термінах ортонормованої або хоча б ортогональної системи векторів. Процедуру переходу від системи довільних твірних векторів до ортогональної бази простору в алгебрі називають ортогоналізацією цієї системи. Відомий алгоритм Грама — Шмідта, за допомогою якого можна ортогоналізувати задану систему векторів. У пакеті LinearAlgebra цей алгоритм реалізований за допомогою функції GramSchmidt(). Її синтаксис такий:

```
GramSchmidt(V,conjugate=value,normalized=value,
outputoptions=list)
```

Тут V — список або множина векторів, які породжують лінійний простір. Усі вектори у V повинні бути одного розміру та орієнтації.

Якщо V — список, результат так само повертається у вигляді списку, в іншому випадку — у вигляді множини векторів ортогональної бази.

Кількість векторів результату може бути меншою від вхідної. Це характерно для лінійної залежності вхідних векторів.

Якщо conjugate=false, припускається, що заданий простір евклідовий, тобто в обчисленнях використовується дійсний скалярний добуток. За замовчуванням використовується conjugate=true і в цьому разі припускається унітарність простору, тобто використовується комплексний скалярний добуток.

Якщо вказано normalized=true (за замовчуванням значення цього параметра false), результатом буде ортонормована база.

Наприклад:

```
> vA1:=Vector[row]([1,0,-1,2]);
```

$$vA1 := [1, 0, -1, 2]$$

```
> vA2:=Vector[row]([-1,-2,1,0]);
```

$$vA2 := [-1, -2, 1, 0]$$

```
> vA3:=Vector[row]([2,2,-2,2]);
```

$$vA3 := [2, 2, -2, 2]$$

```
> GramSchmidt([vA1,vA2,vA3]);
```

$$\left[[1, 0, -1, 2], \left[\frac{-2}{3}, -2, \frac{2}{3}, \frac{2}{3} \right] \right]$$

```
> ortA:=GramSchmidt({vA1,vA2,vA3},normalized=true);
```

$$ortA := \left\{ \left[\frac{1}{6}\sqrt{3}, -\frac{1}{6}\sqrt{3}, -\frac{1}{6}\sqrt{3}, \frac{1}{2}\sqrt{3} \right], \left[-\frac{1}{6}\sqrt{6}, -\frac{1}{3}\sqrt{6}, \frac{1}{6}\sqrt{6}, 0 \right] \right\}$$

```
> DotProduct(ortA[1],ortA[2]);
```

$$0$$

```
> VectorNorm(ortA[1],Euclidean);
```

$$1$$

```
> VectorNorm(ortA[2],Euclidean);
```

$$1$$

```
> vB1:=Vector[row]([x,1]);
```

$$vB1 := [x, 1]$$

```
> vB2:=Vector[row]([2,x]);
```

$$vB2 := [2, x]$$

```
> GramSchmidt([vB1,vB2]);
```

$$\left[[x, 1], \left[-\frac{(2\bar{x} + x)\,x}{1 + x\bar{x}} + 2, -\frac{2\bar{x} + x}{1 + x\bar{x}} + x \right] \right]$$

```
> GramSchmidt([vB1,vB2],
conjugate=false);
```

$$\left[[x, 1], \left[-\frac{3x^2}{1+x^2}+2, -\frac{3x}{1+x^2}+x\right]\right]$$

Норма оператора

Нагадаємо, що нормою оператора називають максимальну (у скінченно вимірному випадку) довжину образів векторів одиничної довжини. Якщо лінійний оператор заданий матрицею в деякій системі координат, то його норму можна обчислити за допомогою функції

```
MatrixNorm(A,p,conj)
```

Тут A — матриця оператора; p визначає, як обчислюється норма оператора. Значенням p може бути 1, 2, infinity, Frobenius та Euclidean.

При p = 1 норма оператора обчислюється як найбільша у стовпцях сума модулів елементів стовпця. При $p = \infty$ (так само, якщо значення p опущено) норма оператора обчислюється як найбільша в рядках сума модулів елементів рядка.

При p = 2 (або Euclidean) норма обчислюється як квадратний корінь з найбільшого власного значення матриці AA^T. При цьому можна використовувати параметр conj виду conjugate=value. Якщо conjugate=false, транспонування вважається ермітовим, в іншому випадку звичайним. За замовчуванням використовується conjugate=true.

При значенні Frobenius для p норма оператора обчислюється як корінь квадратний із суми квадратів модулів елементів його матриці.

Аналогічно MatrixNorm() працює функція Norm().

Приклади:

```
> mA:=Matrix([[10,0,1],
[0,5,5], [0,4,4]]);
```

$$mA := \begin{bmatrix} 10 & 0 & 1 \\ 0 & 5 & 5 \\ 0 & 4 & 4 \end{bmatrix}$$

```
> MatrixNorm(mA,1);
```

$$10$$

```
> MatrixNorm(mA,infinity);
```

$$11$$

```
> MatrixNorm(mA,2);
```

$$\sqrt{\text{RootOf}(\_Z^2 - 183\,\_Z + 8241, index = 2)}$$

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

$$10.14674525$$

```
> MatrixNorm(mA,Frobenius);
```

$$\sqrt{183}$$

```
> evalf(%);
> mB:=Matrix([[x,1],[0,2]]);
```

$$13.52774926$$

$$mB := \begin{bmatrix} x & 1 \\ 0 & 2 \end{bmatrix}$$

```
> MatrixNorm(mB,2);
```

$$\mathrm{sqrt}\left(\max\left(\left|\mathrm{RootOf}(\_Z^2 + (-5 - x\,\bar{x})\,\_Z + 4\,x\,\bar{x}, index = 1)\right|, \left|\mathrm{RootOf}(\_Z^2 + (-5 - x\,\bar{x})\,\_Z + 4\,x\,\bar{x}, index = 2)\right|\right)\right)$$

```
> MatrixNorm(mB,2,conjugate=false);
```

$$\mathrm{sqrt}\left(\max\left(\left|\mathrm{RootOf}(\_Z^2 + (-5 - x^2)\,\_Z + 4\,x^2, index = 2)\right|, \left|\mathrm{RootOf}(\_Z^2 + (-5 - x^2)\,\_Z + 4\,x^2, index = 1)\right|\right)\right)$$

```
> MatrixNorm(mB,Frobenius);
```

$$\sqrt{5 + |x|^2}$$

Внутрішні операції над матрицями

У цьому підрозділі параметр outputoptions=list задає опції orientation, readonly, shape, storage, order, datatype або attributes, які описані в пункті “Функція Matrix()”.

Видалення стовпців або рядків з матриці

Для видалення кількох рядків або стовпців матриці можна скористатися функціями відповідно

```
DeleteRow(A,L,outputoptions=list)
DeleteColumn(A,L,outputoptions=list)
```

Тут A — матриця; L — номер, інтервал номерів або список номерів чи їх інтервалів відповідно рядків або стовпців, що вилучаються.

Наприклад:

```
> mA:=Matrix(3,8,(i,j)->i*j);
```

$$mA := \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 2 & 4 & 6 & 8 & 10 & 12 & 14 & 16 \\ 3 & 6 & 9 & 12 & 15 & 18 & 21 & 24 \end{bmatrix}$$

```
> DeleteColumn(mA,[1,3..5,-2]);
```

$$\begin{bmatrix} 2 & 6 & 8 \\ 4 & 12 & 16 \\ 6 & 18 & 24 \end{bmatrix}$$

```
> DeleteRow(mA,2..3);
```

$$[1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7 \quad 8]$$

Операції отримання рядків, стовпців та діагоналей матриці

Для отримання з матриці рядків, стовпців чи діагоналей використовують функції відповідно

```
Row(A,L,outputoptions=list)
Column(A,L,outputoptions=list)
Diagonal(A,L,outputoptions=list)
```

Тут A — матриця; L — номер, інтервал номерів або список номерів чи їх інтервалів відповідно рядків, стовпців або діагоналей, які потрібно отримати. При цьому головна діагональ має номер 0.

Наприклад:

```
> mA:=Matrix(3,8,(i,j)->i*j);
```

$$mA := \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 2 & 4 & 6 & 8 & 10 & 12 & 14 & 16 \\ 3 & 6 & 9 & 12 & 15 & 18 & 21 & 24 \end{bmatrix}$$

```
> Row(mA,[-1,-2]);
```

$$[3, 6, 9, 12, 15, 18, 21, 24], [2, 4, 6, 8, 10, 12, 14, 16]$$

```
> Column(mA,-1);
```

$$\begin{bmatrix} 8 \\ 16 \\ 24 \end{bmatrix}$$

```
> Diagonal(mA);
```

$$\begin{bmatrix} 1 \\ 4 \\ 9 \end{bmatrix}$$

```
> Diagonal(mA,[-2,0..2,6]);
```

$$[3], \begin{bmatrix} 1 \\ 4 \\ 9 \end{bmatrix}, \begin{bmatrix} 2 \\ 6 \\ 12 \end{bmatrix}, \begin{bmatrix} 3 \\ 8 \\ 15 \end{bmatrix}, \begin{bmatrix} 7 \\ 16 \end{bmatrix}$$

Елементарні перетворення матриць

У пакеті LinearAlgebra містяться три функції RowOperation(), ColumnOperation() і Pivot(), за допомогою яких можна реалізувати елементарні перетворення матриць. Перші дві з них мають однаковий синтаксис та дію і застосовуються для перетворень відповідно рядків і стовпців. Розглянемо детально функцію RowOperation(). Її синтаксис такий:

```
RowOperation(A,K,s,ip,outputoptions=list)
```

Тут A — матриця; K — один номер або список з двох номерів рядків; s — скаляр.

При цьому у випадку RowOperation(A,[m,n]) рядки з номерами m та n міняються місцями; у випадку RowOperation(A,m,s) рядок з номером m помножується на скаляр s; у випадку RowOperation(A,[m,n],s) до m-го рядка додається n-й, домножений на число s.

Параметр ip має вид inplace=true або inplace=false (за замовчуванням). У першому випадку результат записується на місце матриці, а у другому створюється новий об'єкт.

Наприклад:

```
> mA:=Matrix([[7,-3,1,7],
[2,5,2,8],[9,4,-1,8]]);
```

$$mA := \begin{bmatrix} 7 & -3 & 1 & 7 \\ 2 & 5 & 2 & 8 \\ 9 & 4 & -1 & 8 \end{bmatrix}$$

```
> ColumnOperation(mA,[1,3],
inplace);
```

$$\begin{bmatrix} 1 & -3 & 7 & 7 \\ 2 & 5 & 2 & 8 \\ -1 & 4 & 9 & 8 \end{bmatrix}$$

```
> RowOperation(mA,[2,1],-2,
inplace);
```

$$\begin{bmatrix} 1 & -3 & 7 & 7 \\ 0 & 11 & -12 & -6 \\ -1 & 4 & 9 & 8 \end{bmatrix}$$

```
> RowOperation(mA,[3,1],1,
inplace);
```

$$\begin{bmatrix} 1 & -3 & 7 & 7 \\ 0 & 11 & -12 & -6 \\ 0 & 1 & 16 & 15 \end{bmatrix}$$

```
> RowOperation(mA,[2,3],
inplace);
```

$$\begin{bmatrix} 1 & -3 & 7 & 7 \\ 0 & 1 & 16 & 15 \\ 0 & 11 & -12 & -6 \end{bmatrix}$$

```
> RowOperation(mA,[3,2],
-11,inplace);
```

$$\begin{bmatrix} 1 & -3 & 7 & 7 \\ 0 & 1 & 16 & 15 \\ 0 & 0 & -188 & -171 \end{bmatrix}$$

```
> RowOperation(mA,3,-1/188,
inplace);
```

$$\begin{bmatrix} 1 & -3 & 7 & 7 \\ 0 & 1 & 16 & 15 \\ 0 & 0 & 1 & \frac{171}{188} \end{bmatrix}$$

```
> RowOperation(mA,[2,3],-16,
inplace);
```

$$\begin{bmatrix} 1 & -3 & 7 & 7 \\ 0 & 1 & 0 & \frac{21}{47} \\ 0 & 0 & 1 & \frac{171}{188} \end{bmatrix}$$

```
> RowOperation(mA,[1,3],-7,
inplace);
```

$$\begin{bmatrix} 1 & -3 & 0 & \frac{119}{188} \\ 0 & 1 & 0 & \frac{21}{47} \\ 0 & 0 & 1 & \frac{171}{188} \end{bmatrix}$$

```
> RowOperation(mA,[1,2],3,
inplace);
```

$$\begin{bmatrix} 1 & 0 & 0 & \frac{371}{188} \\ 0 & 1 & 0 & \frac{21}{47} \\ 0 & 0 & 1 & \frac{171}{188} \end{bmatrix}$$

Функція Pivot() використовується для отримання нулів у стовпці із вказаним елементом за допомогою елементарних перетворень рядків. Вона має такий синтаксис;

```
Pivot(A,i,j,L,ip,outputoptions=list)
```

Тут A — матриця; i,j — індекси ведучого елемента матриці; L — список рядків, в яких у ведучому стовпці буде отримано нулі.

Приклади:

```
> mA:=Matrix([[7,-3,1,7],
[2,5,2,8],[9,4,-1,8]]);
```

$$mA := \begin{bmatrix} 7 & -3 & 1 & 7 \\ 2 & 5 & 2 & 8 \\ 9 & 4 & -1 & 8 \end{bmatrix}$$

```
> Pivot(mA,1,3,inplace);
```

$$\begin{bmatrix} 7 & -3 & 1 & 7 \\ -12 & 11 & 0 & -6 \\ 16 & 1 & 0 & 15 \end{bmatrix}$$

```
> Pivot(mA,3,2,inplace);
```

$$\begin{bmatrix} 55 & 0 & 1 & 52 \\ -188 & 0 & 0 & -171 \\ 16 & 1 & 0 & 15 \end{bmatrix}$$

```
> Pivot(mA,2,1,inplace);
```

$$\begin{bmatrix} 0 & 0 & 1 & \frac{371}{188} \\ -188 & 0 & 0 & -171 \\ 0 & 1 & 0 & \frac{21}{47} \end{bmatrix}$$

Типові задачі лінійної алгебри

Власні значення та власні вектори

Нагадаємо, що власне значення лінійного оператора A — це таке число λ, що Ax = λx для деякого ненульового вектора x, який називають власним вектором, що відповідає λ. Для знаходження власних значень використовують характеристичну матрицю A-λE, визначник якої дорівнює нулю лише для власних значень λ. Цю матрицю можна обчислити, використовуючи таку функцію:

```
CharacteristicMatrix(A,lambda,outputoptions=list)
```

Тут A — задана матриця; lambda — змінна, параметр характеристичної матриці, outputoptions=list задає додаткові опції readonly, shape, storage, order, datatype або attributes, які описано в пункті “Функція Matrix()”.

Приклади:

```
> A := Matrix([[-2,1,2],
[3,-1,0],[4,3,5]]);
```

$$A := \begin{bmatrix} -2 & 1 & 2 \\ 3 & -1 & 0 \\ 4 & 3 & 5 \end{bmatrix}$$

```
> A_ch:=CharacteristicMatrix
(A,lambda);
```

$$A\_ch := \begin{bmatrix} -\lambda - 2 & 1 & 2 \\ 3 & -\lambda - 1 & 0 \\ 4 & 3 & -\lambda + 5 \end{bmatrix}$$

```
> A_ch_pol:=Determinant(A_ch);
```

$$A\_ch\_pol := -\lambda^3 + 2\lambda^2 + 24\lambda + 21$$

```
> solve(A_ch_pol=0,lambda);
```

$$-1, \frac{3}{2} + \frac{1}{2}\sqrt{93}, \frac{3}{2} - \frac{1}{2}\sqrt{93}$$

Останні дві команди обчислюють відповідно характеристичний многочлен, тобто визначник характеристичної матриці, та власні значення матриці A як його корені.

Характеристичний многочлен можна обчислити також, використовуючи безпосередньо функцію CharacteristicPolynomial(). Її синтаксис такий:

```
CharacteristicPolynomial(A,x)
```

Тут A — матриця оператора; x — змінна характеристичного многочлена.

Продовжуючи попередній приклад, отримуємо

```
> A_ch_pol1:=CharacteristicPolynomial(A,lambda);
```

$$A\_ch\_pol1 := \lambda^3 - 2\lambda^2 - 24\lambda - 21$$

Як бачимо, функція шукає характеристичний поліном зі старшим коефіцієнтом, що дорівнює одиниці.

У пакеті LinearAlgebra містяться також функції для безпосереднього обчислення власних значень та векторів: Eigenvalues() та Eigenvectors(). Синтаксис першої з них такий:

```
Eigenvalues(A,implicit,output=obj,outputoptions=list)
```

Тут A — матриця оператора, необов'язковий параметр implicit обчислює власні значення в неявній формі RootOf, параметр output вказує, як вивести результат. Можливі значення параметра output: 'Vector', 'Vector[row]', 'Vector[column]' і 'list'.

Наприклад, продовжуючи попередні обчислення, отримаємо

```
> Eigenvalues(A,output='list');
```

$$\left[-1, \frac{3}{2} + \frac{1}{2}\sqrt{93}, \frac{3}{2} - \frac{1}{2}\sqrt{93}\right]$$

```
> Eigenvalues(A,implicit,output='Vector[column]');
```

$$\begin{bmatrix} -1 \\ \text{RootOf}(\_Z^2 - 3\_Z - 21, index = 1) \\ \text{RootOf}(\_Z^2 - 3\_Z - 21, index = 2) \end{bmatrix}$$

Синтаксис функції Eigenvectors() подібний:

```
Eigenvectors(A,implicit,output=obj,outputoptions=list)
```

Уточнення тут потребує лише параметр output=obj. За замовчуванням функція повертає результат у вигляді вектора власних значень та матриці, стовпці якої формують відповідні власні вектори. Якщо значення параметра output='list', то результати виводяться у вигляді списку векторів, перша координата яких формується власним значенням, друга — кратністю власного значення, а третя — відповідними власному значенню векторами. Значення 'values' чи 'vectors' означають, що потрібно виводити власні значення чи вектори.

Наприклад:

```
> B:=Matrix([[-235, -357, -204, -357], [189, 359, 252, 441],
[-45, -105, -142, -105], [-45, -105, -60, -187]]);
```

$$B := \begin{bmatrix} -235 & -357 & -204 & -357 \\ 189 & 359 & 252 & 441 \\ -45 & -105 & -142 & -105 \\ -45 & -105 & -60 & -187 \end{bmatrix}$$

```
> Eigenvectors(B);
```

$$\begin{bmatrix} -82 \\ -82 \\ -82 \\ 41 \end{bmatrix}, \begin{bmatrix} -\frac{7}{3} & -\frac{4}{3} & -\frac{7}{3} & \frac{17}{5} \\ 1 & 0 & 0 & -\frac{21}{5} \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

```
> Eigenvectors(B,output='list');
```

$$\left[\left[-82, 3, \left\{\begin{bmatrix}-\frac{7}{3}\\1\\0\\0\end{bmatrix}, \begin{bmatrix}-\frac{4}{3}\\0\\1\\0\end{bmatrix}, \begin{bmatrix}-\frac{7}{3}\\0\\0\\1\end{bmatrix}\right\}\right], \left[41, 1, \left\{\begin{bmatrix}\frac{17}{5}\\-\frac{21}{5}\\1\\1\end{bmatrix}\right\}\right]\right]$$

Розв'язування систем лінійних рівнянь

У пакеті LinearAlgebra містяться кілька функцій, які можна використовувати при розв'язуванні систем лінійних рівнянь. Зокрема, функція GaussianElimination() зводить задану матрицю до східчастого виду методом Гаусса. Якщо вихідна матриця системи розширена, то після її зведення до верхньої східчастої форми для знаходження розв'язків методом зворотної підстановки можна використати функцію BackwardSubstitute().

Якщо розширену матрицю системи зведено до нижньої східчастої форми, то розв'язки системи можна знайти методом прямої підстановки, який реалізований у функції ForwardSubstitute.

І нарешті, одразу отримати за розширеною матрицею розв'язок системи можна за допомогою функції ReducedRowEchelonForm, яка використовує метод Йордана — Гаусса.

Перелічені функції мають такий синтаксис:

```
GaussianElimination(A,m,outputoptions=list)
BackwardSubstitute(A,B,t,ip,outputoptions=list)
ForwardSubstitute(A,B,t,ip,outputoptions=list)
ReducedRowEchelonForm(A)
```

Тут A — матриця (або розширена матриця системи); B — вектор (або матриця у випадку матричного рівняння) правої частини системи. Якщо навести параметр m у вигляді method='FractionFree', зведена методом Гаусса матриця не міститиме раціональних дробів. Параметр t має вигляд free=symbol і використовується для позначення параметрів загального розв'язку системи.

Параметр outputoptions=list задає додаткові опції readonly, shape, storage, order, datatype або attributes, які описані в пункті "Функція Ma-

trix()". Параметр ip має вигляд inplace=true або inplace=false (за замовчуванням). У першому випадку результат записується на місце другого аргумента (необхідно, щоб матриця A була квадратною), а у другому створюється новий об'єкт.

Приклади:

```
> mA:=Matrix([[1,5,-3],[2,6,-2],
[-1,0,9]]);
```

$$mA := \begin{bmatrix} 1 & 5 & -3 \\ 2 & 6 & -2 \\ -1 & 0 & 9 \end{bmatrix}$$

```
> vB:=Vector([7,1,-14]);
```

$$vB := \begin{bmatrix} 7 \\ 1 \\ -14 \end{bmatrix}$$

```
> mAb:=<mA|vB>;
```

$$mAb := \begin{bmatrix} 1 & 5 & -3 & 7 \\ 2 & 6 & -2 & 1 \\ -1 & 0 & 9 & -14 \end{bmatrix}$$

```
> GE_mAb:=
GaussianElimination(mAb);
```

$$GE\_mAb := \begin{bmatrix} 1 & 5 & -3 & 7 \\ 0 & -4 & 4 & -13 \\ 0 & 0 & 11 & \frac{-93}{4} \end{bmatrix}$$

```
> GE_mAb1:=
GaussianElimination
(mAb,method=
'FractionFree');
```

$$GE\_mAb1 := \begin{bmatrix} 1 & 5 & -3 & 7 \\ 0 & -4 & 4 & -13 \\ 0 & 0 & -44 & 93 \end{bmatrix}$$

```
> vX:=BackwardSubstitute(GE_mAb);
```

$$vX := \begin{bmatrix} \frac{-221}{44} \\ \frac{25}{22} \\ \frac{-93}{44} \end{bmatrix}$$

```
> mA.vX-vB;
```

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

```
> JGE_mAb:=
ReducedRowEchelonForm
(mAb);
```

$$JGE\_mAb := \begin{bmatrix} 1 & 0 & 0 & \frac{-221}{44} \\ 0 & 1 & 0 & \frac{25}{22} \\ 0 & 0 & 1 & \frac{-93}{44} \end{bmatrix}$$

```
> mA1:= <<2,-1,0>|<0,1,
-5>|<0,0,-7>|<0,0,3>>;
```

$$mA1 := \begin{bmatrix} 2 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & -5 & -7 & 3 \end{bmatrix}$$

```
> mB1:= <<1,2,-1>|<5,0,3>>;
```

$$mB1 := \begin{bmatrix} 1 & 5 \\ 2 & 0 \\ -1 & 3 \end{bmatrix}$$

```
> mX1:=ForwardSubstitute
(mA1,mB1,free=lambda);
```

$$mX1 := \begin{bmatrix} \frac{1}{2} & \frac{5}{2} \\ \frac{5}{2} & \frac{5}{2} \\ \lambda_{1,1} & \lambda_{1,2} \\ \frac{23}{6}+\frac{7}{3}\lambda_{1,1} & \frac{31}{6}+\frac{7}{3}\lambda_{1,2} \end{bmatrix}$$

```
> mA1.mX1-mB1;
```

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

Крім того, у пакеті містяться дві функції, які дають змогу за системою рівнянь побудувати розширену матрицю систему і, навпаки, за матрицею системи отримати рівняння, які її породжують, а саме:

```
GenerateMatrix(eqns,vars,aug,outputoptions=list)
GenerateEquations(A,v,B)
```

Тут eqns — список або множина рівнянь системи; vars — список або множина невідомих, які входять у систему; якщо aug задано у вигляді augmented=false (прийнятому за замовчуванням), то результат виведеться як пара: матриця системи, вектор правої частини, якщо значення цього параметра true, результатом буде розширена матриця.

У випадку другої функції A, B — це відповідно ліва і права частини системи, v — список символів-невідомих.

Наприклад:

```
> eq1:=3*x-4*y+5*z=-1;eq2:=2*x+5*y-3*z=7;eq3:=-x+y+5*z=4;
```

$$eq1 := 3x - 4y + 5z = -1$$
$$eq2 := 2x + 5y - 3z = 7$$
$$eq3 := -x + y + 5z = 4$$

```
> mAb2:=GenerateMatrix([eq1,eq2,eq3],[x,y,z],
augmented=true);
```

$$mAb2 := \begin{bmatrix} 3 & -4 & 5 & -1 \\ 2 & 5 & -3 & 7 \\ -1 & 1 & 5 & 4 \end{bmatrix}$$

```
> GE_mAb2:=GaussianElimination(mAb2,method='FractionFree');
```

$$GE\_mAb2 := \begin{bmatrix} 3 & -4 & 5 & -1 \\ 0 & 23 & -19 & 23 \\ 0 & 0 & 147 & 92 \end{bmatrix}$$

```
> GenerateEquations(GE_mAb2,[x,y,z]);
```

$$[3x - 4y + 5z = -1, 23y - 19z = 23, 147z = 92]$$

```
> vX2:=ReducedRowEchelonForm(GE_mAb2);
```

$$vX2 := \begin{bmatrix} 1 & 0 & 0 & \frac{95}{147} \\ 0 & 1 & 0 & \frac{223}{147} \\ 0 & 0 & 1 & \frac{92}{147} \end{bmatrix}$$

```
> GenerateEquations(vX2,[x,y,z]);
```

$$\left[x = \frac{95}{147}, y = \frac{223}{147}, z = \frac{92}{147}\right]$$

Неописані функції

Поза межами огляду залишилась низка функцій пакета LinearAlgebra. Коротко наведемо їх призначення (їх синтаксис та приклади використання можна знайти в Help системи Maple).

BidiagonalForm	Зводить матрицю до дводіагональної форми
BezoutMatrix	Задає матрицю Безу двох многочленів
CompanionMatrix	Повертає супроводжуючу матрицю многочлена
Copy	Створює копію матриці або вектора
EigenConditionNumbers	Використовується при чисельному пошуку узагальнених власних значень та векторів
FrobeniusForm	Зводить матрицю до фробеніусової форми
GivensRotationMatrix	Використовується при лінійних перетвореннях простору
HankelMatrix	Задає матрицю Хенкеля
HermiteForm	Обчислює ермітову нормальну форму матриці

HessenbergForm	Зводить матрицю до верхньої гесенбергової форми
HilbertMatrix	Повертає узагальнену матрицю Гільберта
HouseholderMatrix	Задає матрицю Хаузхолдера
JordanForm	Зводить матрицю до жорданової форми
LeastSquares	Використовується в задачі мінімізації нев'язки
LinearSolve	Надає низку можливостей при розв'язуванні лінійних систем
CreatePermutation LUDecomposition QRDecomposition	Використовуються в задачах факторизації матриць
Map, Map2	Застосування до кожного елемента матриці чи вектора заданого відображення
MinimalPolynomial	Обчислює мінімальний многочлен матриці
Permanent	Подібна до функції обчислення визначника, за винятком того, що всі доданки визначника беруться зі знаком "+"
PopovForm	Обчислює нормальну форму Попова матриці
RandomMatrix RandomVector	Генерують випадкову відповідно матрицю чи вектор
SchurForm	Зводить матрицю до форми Шура
SingularValues	Обчислює сингулярні значення матриці
SmithForm	Зводить матрицю до смітової нормальної форми
SubMatrix SubVector	Використовуються для отримання частини відповідно матриці чи вектора
SylvesterMatrix	Задає матрицю Сильвестра двох многочленів
ToeplitzMatrix	Задає матрицю Тьоплиця
TridiagonalForm	Зводить матрицю до тридіагональної форми
VandermondeMatrix	Задає матрицю Вандермонда
Zip	Застосовується двоаргументна процедура до пари матриць чи векторів

Завдання для самостійного виконання IV

1. Задати матрицю, використовуючи коротку нотацію.

$$A1 := \begin{bmatrix} 1 & 2 & -1 & 3 \\ 0 & 1 & 2 & 4 \\ 0 & 0 & 1 & -1 \end{bmatrix} \quad A2 := \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} \quad A3 := \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$A4 := \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 2 & -1 & 0 & 0 & 0 \end{bmatrix} \quad A5 := \begin{bmatrix} 0 & 0 & 0 \\ 1 & 2 & 0 \\ 0 & -1 & 3 \\ 0 & 0 & 1 \end{bmatrix}$$

2. Виконати завдання 1, використовуючи функцію Matrix. При цьому врахувати структуру матриці.
3. Для матриці завдання 1 обчислити її добуток на транспоновану до неї матрицю.
4. Скориставшись відповідною функцією, обчислити лінійну комбінацію векторів із вказаними скалярами.
   4.1 (2,4,5), (7,-1,9), -2, 8.
   4.2 (1,-1), (5,9), 7, 10.
   4.3 (7, 4, -4, 2), (1,3, 5, 7), -1, 2.
   4.4 (6, 0, 0, -1), (5, -1, 2, 2), 10,9.
   4.5 (2, 5), (3,7), 7, -1.
5. Задати вектор-стовпці

$$V := \begin{bmatrix} 7 \\ 3 \\ 4 \end{bmatrix} \quad U := \begin{bmatrix} 6 \\ -1 \\ 1 \end{bmatrix}$$

   5.1 Знайти кут між цими векторами.
   5.2 Обчислити їх скалярний добуток.
   5.3 Обчислити їх векторний добуток.
   5.4 Обчислити p-норму вектора V при p = 5.
   5.5 Знайти евклідову норму вектора U.

6. Використовуючи функцію BandMatrix, побудувати матрицю

$$A := \begin{bmatrix} -2 & 1 & 0 & 0 \\ 5 & -2 & 1 & 0 \\ 0 & 5 & -2 & 1 \\ 0 & 0 & 5 & -2 \end{bmatrix}$$

6.1 Обчислити її визначник.
6.2 Знайти обернену до неї матрицю.
6.3 Знайти слід матриці.
6.4 Перевірити, чи додатно визначена матриця.
6.5 Знайти ранг цієї матриці.

7. Для матриці із завдання 6 обчислити:
7.1 e^A.
7.2 A^{20}.
7.3 ln(A).
7.4 sin(A).
7.5 cos(A).

8. Задати матрицю

$$M := \begin{bmatrix} -1 & 3 & 1 \\ 1 & -2 & 5 \\ 1 & 3 & 0 \end{bmatrix}$$

8.1 Обчислити характеристичну матрицю для неї.
8.2 Обчислити характеристичний многочлен.
8.3 Обчислити власні значення матриці.
8.4 Знайти власні вектори матриці.
8.5 Знайти власні значення матриці M^2.

9. Написати таку процедуру:
9.1 Яка для заданої матриці обчислює суму її елементів.
9.2 Яка для заданої матриці перевіряє, чи квадратна вона.
9.3 Яка для заданого вектора перевіряє, чи впорядковані його координати за спаданням.
9.4 Яка для заданої матриці її від'ємні елементи змінює на нулі.
9.5 Яка для заданих матриць перевіряє, чи можна їх перемножити в якомусь порядку.

Список використаної та рекомендованої літератури

Основна

1. *Аладьев В., Шишаков М.* Автоматизированное рабочее место математика. — М.: Лаборат. баз. знаний, 2000.
2. *Васильев А.* Maple 8. Самоучитель. — М.: Диалектика, Вильямс, 2003.
3. *Дьяконов В.* Maple 9 в математике, физике и образовании. — М.: СОЛОН-Пресс, 2004.
4. *Прохоров Г., Леденев М., Колбеев В.* Пакет символьных вычислений Maple. — М.: Петит, 1997.

Додаткова

5. *Аладьев В.* Эффективная работа в Maple 6 и 7. — М.: Лаборат. баз. знаний, 2002.
6. *Аладьев В., Богдявичюс М.* Maple 6: Решение математических, статистических и инженерно-физических задач. — М.: Лаборат. баз. знаний, 2001.
7. *Говорухин В., Цибулин В.* Введение в Maple. Математический пакет для всех. — М.: Мир, 1997.
8. *Говорухин В., Цибулин В.* Компьютер в математическом исследовании: Maple, MATLAB, LaTeX. — СПб.: Питер, 2001.
9. *Голоскоков Д.* Уравнения математической физики. Решение задач в системе Maple. — СПб.: Питер, 2004.
10. *Дьяконов В.* Компьютерная математика. Теория и практика. — М.: Нолидж, 2000.
11. *Дьяконов В.* Maple 8 в математике, физике и образовании. — М.: СОЛОН-Пресс, 2003.
12. *Кирсанов М.* Решебник. Теоретическая механика. — М.: Физматлит, 2002.
13. *Матросов А.* Maple 6. Решение задач высшей математики и механики. — СПб.: БХВ-Петербург, 2001.
14. *Очков В.* Физические и экономические величины в MathCAD и Maple. — М.: Финансы и статистика, 2002.
15. *Сдвижков О. А.* Математика на компьютере: Maple 8. — М.: СОЛОН-Пресс, 2003.

ЗМІСТ

Book is the sequel of part 1 ”Symbolic calculus in Maple’s system” and contains educational material, samples of task solutions and tasks for individual solutions from such parts: typical ways of programming in Maple’s system, specific functions and work with them, approximation function , numerous solutions of non-linear equation systems, numerous integration of Koshi tasks, approximate calculation of integrals, finding extremum functions, linear algebra. Book can be used in studying of such disciplines: “Linear algebra and analytical geometry”, “ Differential equation”, ” Quantity methods ”, ”Quantity methods in information technologies”, “Quantity methods of mathematical physics”, “Symbolic calculus”, “Computer algebra” etc.

For students of “Applied mathematics” and ” Computer science” educational spheres.

Навчальне видання

Кузьмін Анатолій Володимирович
Кузьміна Наталія Миколаївна
Телейко Андрій Богданович

СИМВОЛЬНІ ТА НАБЛИЖЕНІ ОБЧИСЛЕННЯ В СИСТЕМІ MAPLE

Частина 2

Навчальний посібник

Educational Edition

Kuzmin, Anatoliy V.
Kuzmina, Nataliya M.
Teleyko, Andriy B.

SYMBOLIC AND APPROXIMATE CALCULUS IN MAPLE’S SYSTEM

Part 2

Educational supplier

Відповідальний редактор *С. Г. Рогузько*
Редактор *Л. В. Логвиненко*
Комп’ютерне верстання *Т. Г. Замура*
Оформлення обкладинки *А. В. Ясиновський*

Підп. до друку 28.01.08. Формат 60×84/$_{16}$. Папір офсетний.
Друк ротаційний трафаретний. Ум. друк. арк. 8,6. Обл.-вид. арк. 7,07. Тираж 1000 пр.

Міжрегіональна Академія управління персоналом (МАУП)
03039 Київ-39, вул. Фрометівська, 2, МАУП

ДП «Видавничий дім «Персонал»
03039 Київ-39, просп. Червонозоряний, 119, літ. ХХ

Свідоцтво про внесення до Державного реєстру суб’єктів видавничої справи ДК № 8 від 23.02.2000