2. Основи Maxima

2.1 Структура Maxima

Пакет Maxima складається з інтерпретатора макромови, написаного на Lisp, та кількох поколінь пакетів розширень, написаних на макромові пакета або безпосередньо на Lisp. Maxima дозволяє вирішувати досить широке коло завдань, що належать до різних розділів математики.

2.1.1 Області математики, що підтримуються в Maxima

- Операції з поліномами (маніпуляція раціональними та статечними виразами, обчислення коренів тощо)
- Обчислення з елементарними функціями, у тому числі з логарифмами, експоненціальними функціями, тригонометричними функціями
- Обчислення зі спеціальними функціями, зокрема. еліптичними функціями та інтегралами
- Обчислення меж та похідних
- Аналітичне обчислення певних та невизначених інтегралів
- Розв'язання інтегральних рівнянь
- Розв'язання алгебраїчних рівнянь та їх систем
- Операції зі статечними рядами та рядами Фур'є
- Операції з матрицями та списками, велика бібліотека функцій для вирішення задач лінійної алгебри
- Операції з тензорами
- Теорія чисел, теорія груп, абстрактна алгебра

Перелік додаткових пакетів для Maxima, які необхідно завантажувати перед використанням, що істотно розширюють її можливості та коло задач, що вирішуються, наведено в додатку 1.

2.2 Переваги програми

Основними перевагами програми Maxima є:

- можливість вільного використання (Maxima відноситься до класу вільних програм та поширюється на основі ліцензії GNU);
- можливість функціонування під управлінням різних ОС (зокрема Linux та Windows™);
- розмір програми (дистрибутив займає близько 23 мегабайт, у встановленому вигляді з усіма розширеннями потрібно близько 80 мегабайт);
- широкий клас розв'язуваних завдань;

- можливість роботи як у консольній версії програми, так і з використанням одного з графічних інтерфейсів (xMaxima, wxMaxima або як плагін (plug-in) до редактора TexMacs);
- розширення wxMaxima (що входить до комплекту поставки) надає користувачеві зручний і зрозумілий інтерфейс, позбавляє необхідності вивчати особливості введення команд для вирішення типових завдань;
- інтерфейс програми російською мовою;
- наявність довідки та інструкцій по роботі з програмою

2.3 Встановлення та запуск програми

Завантажити останню версію програми можна з її сайту в Інтернеті: http://maxima.sourceforge.net/.

2.4 Інтерфейс wxMaxima

Для зручності роботи відразу звернемося до графічного інтерфейсу wxMaxima, тому що він є найбільш дружнім для користувачів-початківців системи.

Достоїнствами wxMaxima є:

- можливість графічного виведення формул (див. ілюстрації нижче)
- спрощене введення найбільш часто використовуваних функцій (через діалогові вікна), а не набір команд, як у класичній Maxima.
- поділ вікна введення даних та області виведення результатів (у класичній Maxima ці області об'єднані, і введення команд відбувається в єдиній робочій області з отриманими результатами).

Розглянемо робоче вікно програми. Зверху вниз розташовуються: текстове меню програми – доступ до основних функцій та налаштувань програми. У текстовому меню wxMaxima знаходяться функції для вирішення великої кількості типових математичних завдань, розділені за групами: рівняння, алгебра, аналіз, спростити, графіки, чисельні обчислення. Введення команд через діалогові вікна полегшує роботу з програмою для новачків.

При використанні інтерфейсу wxMaxima, Ви можете виділити у вікні виведення результатів необхідну формулу та викликавши контекстне меню правою кнопкою миші скопіювати будь-яку формулу у текстовому вигляді, у форматі TEX або у вигляді графічного зображення для наступної вставки в будь-який документ.

Також у контекстному меню, при виборі результату обчислення, Вам буде запропоновано низку операцій з вибраним виразом (наприклад, спрощення, розкриття дужок, інтегрування, диференціювання та ін.).

2.5 Введення найпростіших команд Maxima

Усі команди вводяться у полі ВВЕДЕННЯ, роздільником команд є символ ; (Крапка з комою). Після введення команди необхідно натиснути клавішу Enter для її обробки та виведення результату. У ранніх версіях Maxima та деяких її оболонках (наприклад, xMaxima) наявність точки з комою після кожної команди суворо обов'язково. Завершення введення символом $ (замість точки з комою) дозволяє обчислити результат введеної команди, але не виводити його на екран. У разі коли вираз треба відобразити, а не обчислити, перед ним необхідно поставити знак '(одинарна лапка). Але цей метод не працює, коли вираз має явне значення, наприклад вираз $sin(\pi)$ замінюється на значення, що дорівнює нулю.

Дві одинарні лапки послідовно, застосовані до виразу у вхідному рядку, призводять до заміщення вхідного рядка результатом обчислення виразу, що вводиться.

Приклад:

```
(%i1)    aa:1024;
```
$$(\%o1) \qquad 1024$$
```
(%i2)    bb:19;
```
$$(\%o2) \qquad 19$$
```
(%i3)    sqrt(aa)+bb;
```
$$(\%o3) \qquad 51$$
```
(% i4)   '(sqrt(aa)+bb);
```
$$(\%o4) \qquad bb + \sqrt{aa}$$
```
(% i5)   ''%;
```
$$(\%o5) \qquad 51$$

2.5.1 Позначення команд та результатів обчислень

Після введення кожній команді надається порядковий номер. У розглянутому прикладі введені команди мають номери 1–5 і позначаються відповідно (%i1), (%i2) тощо.

Результат обчислення також має порядковий номер, наприклад $(\%o1), (\%o2)$ і т.д., де і скорочення від англ. input (введення), а про - англ. output (висновок). Цей механізм дозволяє уникнути в наступних обчисленнях повторення повного запису вже виконаних команд, наприклад (%i1)+(%i2) означатиме додавання до виразу першої команди - виразу другої та наступного обчислення результату. Також можна використовувати номери результатів обчислень, наприклад $(\%o1)*(\%o2)$. Для останньої виконаної команди у Maxima є спеціальне позначення. $\%$.

Приклад:

Обчислити значення похідної функції $y(x) = x^2 \cdot e^{-x}x$:

```
(%i1) diff(x^2*exp(-x),x);
```

```
(%o1)                    2 x e^-x - x^2 e^-x
(%i2) f(x):=''%;
(%o2)                f(x) := 2 x e^-x - x^2 e^-x
```

Подвійна лапка перед символом попередньої операції дозволяє замістити цей символ значенням, тобто. текстовим рядком, отриманим у результаті диференціювання.

Інший приклад (з очевидним змістом):

```
(%i3)    x:4;
(%o3)                          4
(% i4)   sqrt(x);
(%o4)                          2
(% i5)   %^2;
(%o5)                          4
```

2.6 Числа, оператори та константи

2.6.1 Введення цифрової інформації

Правила введення чисел в Maxima такі самі, як і для багатьох інших подібних програм. Ціла і дрібна частина десяткових дробів поділяються символом крапка. Перед від'ємними числами ставиться знак мінус. Чисельник і знаменник звичайних дробів розділяється символом / (прямий слеш). Зверніть увагу, що якщо в результаті виконання операції виходить деякий символьний вираз, а необхідно отримати конкретне числове значення у вигляді десяткового дробу, то вирішити це завдання дозволить застосування прапора numer. Зокрема, він дозволяє перейти від звичайних дробів до десяткових. Перетворення до форми з плаваючою точкою здійснює також функція $float$.

```
(%i1)    3/7+5/3;
                              44
(%o1)                         --
                              21
(%i2)    3/7+5/3, float;
(%o2)              2.095238095238095
(%i3)    3/7+5/3, номер;
(%o3)              2.095238095238095
(% i4)   float(5/7);
(%o4)              0.71428571428571
```

2.6.2 Арифметичні операції

Позначення арифметичних операцій на Maxima нічим відрізняється від класичного уявлення: + , — , * , /. Зведення у ступінь можна позначати кількома способами: ^, ^^, **. Вилучення кореня ступеня n записуємо, як ступінь $\frac{1}{n}$. Операція знаходження факторіалу позначається знаком оклику, наприклад 5!. Для збільшення пріоритету операції, як і математики, використовуються круглі

дужки: (). Список основних арифметичних та логічних операторів наведено у табл. 2.1 та табл. 2.2 нижче.

Таблиця 2.1. Арифметичні оператори

`+`	оператор додавання
`-`	оператор віднімання або зміни знака
`*`	оператор множення
`/`	оператор поділу
`^ або **`	оператор зведення у ступінь

Таблиця 2.2. Логічні оператори

`<`	оператор порівняння менше
`>`	оператор порівняння більше
`<=`	оператор порівняння менше чи одно
`>=`	оператор порівняння більше чи одно
`#`	оператор порівняння не дорівнює
`=`	оператор порівняння одно
`and`	логічний оператор та
`or`	логічний оператор або
`not`	логічний оператор не

2.6.3 Константи

Maxima для зручності обчислень є ряд вбудованих констант. Найпоширеніші з них показані у табл. 2.3:

Таблиця 2.3. Основні константи Maxima

Назва	Позначення
ліворуч (щодо меж)	$minus$
праворуч (щодо меж)	$plus$
плюс нескінченність	inf
мінус нескінченність	$minf$
число π	$\%pi$
e (експонента)	$\%e$
Уявна одиниця $\sqrt{-1}$	$\%i$
Істина	$true$
Брехня	$false$
Золотий перетин $(1+\sqrt{5})/2$	$\%phi$

2.7 Типи даних, змінні та функції

Для зберігання результатів проміжних розрахунків використовуються змінні. Зауважимо, що при введенні назв змінних, функцій та констант важливий регістр букв, так змінні x і X - Дві різні змінні. Присвоювання

значення змінної здійснюється за допомогою символу : (двокрапка), наприклад x:5. Якщо потрібно видалити значення змінної (очистити її), то застосовується метод $kill$:

$kill(x)$ - Видалити значення змінної x ;

$kill(all)$ — видалити значення всіх змінних, що використовуються раніше.

Зарезервовані слова, використання яких як імена змінних викликає синтаксичну помилку:

integrate, next, from, diff, in, at, limit, sum, for, and, elseif, then, else, do, or, if, unless, product, while, thtu, step.

2.7.1 Списки

Списки – базові будівельні блоки для Maxima та Lisp. Усі інші типи даних (масиви, хеш-таблиці, числа) подаються як списки. Щоб задати список, достатньо записати його елементи через кому і обмежити запис квадратними дужками. Список може бути порожнім чи складатися з одного елемента.

```
(%i1) list1:[1,2,3,x,x+y];
```
$$(\%o1) \quad [1, 2, 3, x, y + x]$$
```
(%i2) list2:[];
```
$$(\%o2) \quad []$$
```
(%i3) list3: [3];
```
$$(\%o3) \quad [3]$$

Елементом списку може бути і інший список

```
(% i4)     list4: [1,2, [3,4], [5,6,7]];
```
$$(\%o4) \quad 1, 2, [3, 4], [5, 6, 7]]$$

Посилання на елемент списку здійснюється за номером списку:

```
(% i4)     list4: [1,2, [3,4], [5,6,7]];
```
$$(\%o4) \quad [1, 2, [3, 4], [5, 6, 7]]$$
```
(% i5)     list4[1];
```
$$(\%o5) \quad 1$$
```
(%i6) list4[3];
```
$$(\%o6) \quad [3, 4]$$
```
(%i7) list4[3][2];
```
$$(\%o7) \quad 4$$

2.7.1.1 Функції для елементарних операцій зі списками

Функція $length$ повертає кількість елементів списку (при цьому елементи списку самі можуть бути досить складними конструкціями):

```
(% i8)     length(list4);
```
$$(\%o8) \quad 4$$
```
(% i9)     length(list3);
```
$$(\%o9) \quad 1$$

Функція $copylist(expr)$ повертає копію списку expr:

```
(%i1) list1:[1,2,3,x,x+y];
```
$$(\%o1) \qquad [1, 2, 3, x, y + x]$$
```
(%i2) list2:copylist(list1);
```
$$(\%o2) \qquad [1, 2, 3, x, y + x]$$

Функція $makelist$ створює список, кожен елемент якого генерується з певного виразу. Можливі два варіанти виклику цієї функції:

- $makelist(expr, i, i_0, i_1)$ - Повертає список, j -й елемент якого дорівнює $ev(expr, i = j)$, при цьому індекс j змінюється від i_0 до i_1
- $makelist(expr, x, list)$ - Повертає список, j -й елемент якого дорівнює $ev(expr, x = list[j])$, при цьому індекс j змінюється від 1 до $length(list)$.

Приклади:

```
(%i1) makelist(concat(x,i),i,1,6);
```
$$(\%o1) \qquad [x1, x2, x3, x4, x5, x6]$$
```
(%i2) list: [1,2,3,4,5,6,7];
```
$$(\%o2) \qquad [1, 2, 3, 4, 5, 6, 7]$$
```
(%i3) makelist(exp(i),i,list);
```
$$(\%o3) \qquad [e, e^2, e^3, e^4, e^5, e^6, e^7]$$

Багато в чому аналогічні дії виконує функція $create_list(form, x_1, list_1, ..., x_n, list_n)$.

Ця функція будує список шляхом обчислення виразу $form$, що залежить від x_1 до кожного елемента списку $list_1$ (аналогічно form, що залежить і від x_2, застосовується до $list_2$ і т.д.).

Приклад:

```
(%i1) create_list(x^i,i,[1,3,7]);
```
$$(\%o1) \qquad [x, x^3, x^7]$$
```
(%i2) create_list([i,j],i,[a,b],j,[e,f,h]);
```
$$(\%o2) \qquad [[a, e], [a, f], [a, h], [b, e], [b, f], [b, h]]$$

Функція Append дозволяє склеювати списки. При виклику

```
append (list_1, \dots, list_n)
```

повертається один список, у якому за елементами $list_1$ слідують елементи $list_2$ і т.д. аж до $list_n$.

Приклад:

```
(%i1) append([1],[2,3],[4,5,6,7]);
```

```
(%o1)                    [1, 2, 3, 4, 5, 6, 7]
```

Створити новий список, компонуючи елементи двох списків по черзі в порядку, дозволяє функція $join(l, m)$. Новий список містить l_1, потім m_1, потім l_2, m_2 і т.д.

Приклад:

```
(%i1) join([1,2,3],[10,20,30]);
(%o1)                [1, 10, 2, 20, 3, 30]
   (% i2) join ([1,2,3], [10,20,30,40]);
(%o2)                [1, 10, 2, 20, 3, 30]
```

Довжина одержаного списку обмежується мінімальною довжиною списків l і m.

Функція $cons(expr, list)$ створює новий список, першим елементом якого буде $expr$, а решта — елементи списку $list$. Функція $endcons(expr, list)$ також створює новий список, перші елементи якого – елементи списку $list$, а останній - новий елемент $expr$.

Приклад:

```
(%i1) cons(x,[1,2,3]);
(%o1)                   [x, 1, 2, 3]
(%i2) endcons(x,[1,2,3]);
(%o2)                   [1, 2, 3, x]
```

Функція $reverse$ змінює порядок елементів у списку на зворотний

```
(% i5)     list1: [1,2,3, x];
(%o5)                   [1, 2, 3, x]
(%i6) list2: reverse (list1);
(%o6)                   [x, 3, 2, 1]
```

Функція $member(expr_1, expr_2)$ повертає $true$, якщо $expr_1$ є елементом списку $expr_2$, і $false$ інакше.

Приклад:

```
(%i1)    member (8, [8, 8.0, 8b0]);
(%o1)                     true
(%i2)    member (8, [8.0, 8b0]);
(%o2)                     false
(%i3)    member (b, [[a, b], [b, c]]);
(%o3)                     false
(% i4)   member ([b, c], [[a, b], [b, c]]);
(%o4)                     true
```

Функція $rest(expr)$ виділяє залишок після видалення першого елемента списку $expr$. Можна видалити перші n елементів, використовуючи виклик

$rest(expr, n)$. Функція $last(expr)$ виділяє останній елемент списку $expr$ (аналогічно $first$ - Перший елемент списку).

Приклади:

```
(%i1) list1: [1,2,3,4, a, b];
```
$$(\%o1) \qquad [1, 2, 3, 4, a, b]$$
```
(%i2) rest(list1);
```
$$(\%o2) \qquad [2, 3, 4, a, b]$$
```
(%i3) rest(%);
```
$$(\%o3) \qquad [3, 4, a, b]$$
```
(% i4)     last(list1);
```
$$(\%o4) \qquad b$$
```
(% i5)     rest(list1,3);
```
$$(\%o5) \qquad [4, a, b]$$

Підсумовування та перемноження списків (як та інших виразів) здійснюється функціями sum і $product$. Функція $sum(expr, i, in, ik)$ підсумовує значення виразу expr при зміні індексу i від in до ik. Функція $product(expr, i, in, ik)$ перемножує значення виразу $expr$ при зміні індексу i від in до ik.

Приклад:

```
(%i1) product (x + i*(i+1)/2, i, 1, 4);
```
$$(\%o1) \qquad (x+1)\,(x+3)\,(x+6)\,(x+10)$$
```
(%i2) sum (x + i*(i+1)/2, i, 1, 4);
```
$$(\%o2) \qquad 4\,x + 20$$
```
(%i3) product (i^2, i, 1, 4);
```
$$(\%o3) \qquad 576$$
```
(% i4)     sum (i^2, i, 1, 4);
```
$$(\%o4) \qquad 30$$

2.7.1.2 Функції, що оперують із елементами списків

Функція $map(f, expr_1, ..., expr_n)$ дозволяє застосувати функцію (оператор, символ операції) f до частин виразів $expr_1, expr_2, ..., expr_n$. При використанні зі списками застосовує f до кожного списку. Слід звернути увагу, що f саме ім'я функції (без вказівки змінних, від яких вона залежить).

Приклади:

```
(%i1) map(ratsimp, x/(x^2+x)+(y^2+y)/y);
```
$$(\%o1) \qquad y + \frac{1}{x+1} + 1$$
```
(%i2) map("=",[a,b],[-0.5,3]);
```
$$(\%o2) \qquad [a = -0.5, b = 3]$$

```
(%i3) map (exp, [0,1,2,3,4,5]);
```
$$(\%o3) \qquad [1, e, e^2, e^3, e^4, e^5]$$

Функція f може бути і заданою користувачем, наприклад:

```
(% i5) f(x):=x^2;
```
$$(\%o5) \qquad f(x) := x^2$$
```
(%i6) map(f,[1,2,3,4,5]);
```
$$(\%o6) \qquad [1, 4, 9, 16, 25]$$

Функція $apply$ застосовує задану функцію до всього списку (список стає списком аргументів функції; під час виклику ($F, [x_1, ..., x_n]$ обчислюється вираз $F(arg_1, ..., arg_n)$). Слід враховувати, що $apply$ не розпізнає ординарні функції та функції від масиву.

Приклад:

```
(%i1) L: [1, 5, -10.2, 4, 3];
```
$$(\%o1) \qquad [1, 5, -10.2, 4, 3]$$
```
(%i2) apply(max,L);
```
$$(\%o2) \qquad 5$$
```
(%i3) apply(min,L);
```
$$(\%o3) \qquad -10.2$$

Щоб знайти максимальний чи мінімальний елемент набору чисел, потрібно викликати функції max чи min. Проте, обидві функції як аргумент чекають кілька чисел, а чи не список, складений із чисел. Застосовувати подібні функції до списків та дозволяє функцію $apply$.

2.7.2 Масиви

Масиви в Maxima - сукупності однотипних об'єктів з індексами. Кількість індексів має перевищувати п'яти. У Maxima існують функції з індексами (функції масиву).

Можливе створення та використання змінних з індексами до оголошення відповідного масиву. Такі змінні розглядаються як елементи масивів із невизначеними розмірностями (так звані хеш-масиви). Розміри невизначених масивів зростають динамічно в міру надання значень елементам. Цікаво, що індекси масивів із невизначеними межами не обов'язково мають бути числами. Для підвищення ефективності обчислень рекомендується перетворювати масиви з невизначеними межами на звичайні масиви (для цього використовується функція $array$).

Створення масиву виконується функцією array. Синтаксис звернення до функції: $array(name, dim_1, ..., dim_n)$ - Створення масиву з ім'ям name і розмірностями $dim_1, ..., dim_n$; array(name,type,dim_1,... ,dim_n) — створення масиву з ім'ям name та елементами типу $type$;

$array([name_1, ..., name_m], dim_1, ..., dim_n)$- Створення декількох масивів однакової розмірності.

Індекси звичайного масиву - цілі числа, що змінюються від 0 доdim_i

Приклад:

```
(%i1) array(a,1,1);
```
$(\%o1)$ a
```
(%i2) a[0,0]:0; a[0,1]:1; a[1,0]:2; a[1,1]:3;
```
$(\%o5)$ 0123
```
(%i6) listarray(a);
```
$(\%o6)$ $[0, 1, 2, 3]$

Функція$listarray$, використана у прикладі, перетворює масив на список. Синтаксис виклику:$listarray(A)$.

АргументA може бути певним чи невизначеним масивом, функцією масиву чи функцією з індексами. Порядок включення елементів масиву до списку по рядках.

Функція$arrayinfo$виводить інформацію про масивA. Синтаксис виклику:$arrayinfo(A)$АргументA, як і у випадку$listarray$може бути певним або невизначеним масивом, функцією масиву або функцією з індексами.

прикладвикористання:

```
(%i1) array (aa, 2, 3);
```
$(\%o1)$ aa
```
(%i2) aa [2, 3]: % pi;
```
$(\%o2)$ π
```
(%i3) aa [1, 2]: % e;
```
$(\%o3)$ e
```
(% i4)     arrayinfo (aa);
```
$(\%o4)$ $[declared, 2, [2, 3]]$
```
(% i5)     bb [FOO]: (a + b) ^ 2;
```
$(\%o5)$ $(b + a)^2$
```
(%i6) bb [BAR]: (c - d) ^ 3;
```
$(\%o6)$ $(c - d)^3$
```
(%i7) arrayinfo (bb);
```
$(\%o7)$ $[hashed, 1, [BAR], [FOO]]$
```
(% i8)     listarray (bb);
```
$(\%o8)$ $[(c - d)^3, (b + a)^2]$

Функції$listarray$і$arrayinfo$застосовні і до функцій масиву:

```
(% i9)     cc [x, y]: = y/x;
```

$$(\%o9) \qquad cc_{x,y} := \frac{y}{x}$$

```
(%i10)     cc[1,2];
```
$$(\%o10) \qquad 2$$
```
(%i11)     cc[2,1];
```
$$(\%o11) \qquad \frac{1}{2}$$
```
(%i12)     arrayinfo(cc);
```
$$(\%o12) \qquad [hashed, 2, [1, 2], [2, 1]]$$
```
(%i13)     listarray(cc);
```
$$(\%o13) \qquad [2, \frac{1}{2}]$$

Ще один приклад — створення та виведення інформації про функції з індексами:

```
(%i1) dd [x] (y): = y^x;
```
$$(\%o1) \qquad dd_x\,(y) := y^x$$
```
(%i2) dd[1](4);
```
$$(\%o2) \qquad 4$$
```
(%i3) dd[a+b];
```
$$(\%o3) \qquad lambda\left([y], y^{b+a}\right)$$
```
(% i4)     arrayinfo(dd);
```
$$(\%o4) \qquad [hashed, 1, [1], [b + a]]$$
```
(% i5)     listarray(dd);
```
$$(\%o5) \qquad [lambda\left([y], y\right), lambda\left([y], y^{b+a}\right)]$$

Функція $make_array(type, dim_1, ..., dim_n)$ створює та повертає масив Lisp. Тип масиву може бути $any,\ flonum,\ fixnum,\ hashed,\ functional$. Індекс i може змінюватися в межах від 0 до $dim_i - 1$.

Гідність $make_array$ в порівнянні з $array$ - Можливість динамічно керувати розподілом пам'яті для масивів. Привласнення $y : make_array(...)$ створює посилання масив. Коли масив більше не потрібний, посилання знищується привласненням $y : false$, пам'ять звільняється потім збирачем сміття.

Приклади:

```
(%i1) A1: make_array (fixnum, 8);
```
$$(\%o1) \qquad Lisp\ Array:\ \#(0\ 0\ 0\ 0\ 0\ 0\ 0\ 0)$$
```
(%i2) A1[1]:8;
```
$$(\%o2) \qquad 8$$
```
(%i3) A3: make_array (any, 8);
```

$(\%o3)$
$Lisp\ Array:\ \#(NIL\ NIL\ NIL\ NIL\ NIL\ NIL\ NIL\ NIL)$
```
(% i4)     arrayinfo(A3);
```
$(\%o4)$ $[declared, 1, [7]]$

Змінна $arrays$ містить список імен масивів першого та другого видів, визначених на даний момент.

Приклад:
```
(%i1) array(a,1,1);
```
$(\%o1)$ a
```
(%i2) array(b,2,3);
```
$(\%o2)$ b
```
(%i3) arrays;
```
$(\%o3)$ $[a, b]$

Функція $fillarray$ дозволяє заповнювати масиви значеннями іншого масиву або списку. Заповнення провадиться по рядках.

Приклади:
```
(%i1) array(a,1,1);
```
$(\%o1)$ a
```
(%i2) fillarray(a,[1,2,3,4]);
```
$(\%o2)$ a
```
(%i3) a[1,1];
```
$(\%o3)$ 4
```
(% i4)     a2: make_array (fixnum, 8);
```
$(\%o4)$ $Lisp\ Array\ \#(0\ 0\ 0\ 0\ 0\ 0\ 0\ 0)$
```
(% i5)     fillarray (a2, [1, 2, 3, 4, 5]);
```
$(\%o5)$ $Lisp\ Array\ \#(1\ 2\ 3\ 4\ 5\ 5\ 5\ 5)$

Як очевидно з розглянутих прикладів, довжина списку може і збігатися з розмірністю масиву. Якщо вказано тип масиву, він повинен заповнюватись елементами того самого типу. Видалення масивів із пам'яті здійснюється функцією $remarray$.

Крім того, для зміни розмірності масиву є функція $rarray(A, dim_1, ..., dim_n)$. Новий масив заповнюється елементами старого рядками. Якщо розмір старого масиву менший за новий, залишок нового заповнюється нулями або $false$ (Залежно від типу масиву).

2.7.3 Матриці та найпростіші операції з ними

У Maxima визначено прямокутні матриці.

Основний спосіб створення матриць – використання функції matrix. Синтаксис виклику: $matrix(row_1, ..., row_n)$. Кожен рядок – список виразів, усі рядки однакової довжини. На безлічі матриць визначені операції складання,

віднімання, множення та поділу. Ці операції виконуються поелементно, якщо операнди - дві матриці, скаляр та матриця або матриця та скаляр.

Зведення в ступінь можливе, якщо один із операндів - скаляр. Перемноження матриць (загалом некомутативна операція) позначається символом ".". Операція множення матриці на себе може розглядатися як зведення в ступінь. Зведення у ступінь -1 - як звернення (якщо це можливо).

прикладстворення двох матриць:

```
(%i1) x: matrix ([17, 3], [-8, 11]);
```

$$(\%o1)\begin{bmatrix} 17 & 3 \\ -8 & 11 \end{bmatrix}$$

```
(%i2) y: matrix ([%pi, %e], [a, b]);
```

$$(\%o2)\begin{bmatrix} \pi & e \\ a & b \end{bmatrix}$$

Виконання арифметичних операцій із матрицями:

```
(%i3) x+y;
```

$$(\%o3)\begin{bmatrix} \pi+17 & e+3 \\ a-8 & b+11 \end{bmatrix}$$

```
(% i4)      xy;
```

$$(\%o4)\begin{bmatrix} 17-\pi & 3-e \\ -a-8 & 11-b \end{bmatrix}$$

```
(% i5)      x*y;
```

$$(\%o5)\begin{bmatrix} 17\,\pi & 3\,e \\ -a\,8 & 11\,b \end{bmatrix}$$

```
(%i6) x/y;
```

$$(\%o6)\begin{bmatrix} \frac{17}{\pi} & 3\,e^{-1} \\ -\frac{8}{a} & \frac{11}{b} \end{bmatrix}$$

Зверніть увагу – операції виконуються поелементно. При спробі виконувати арифметичні операції, як вище, над матрицями різних розмірів, видається помилка.

прикладоперацій з матрицями та скалярами:

```
(% i9)      x^3;
```

$$(\%o9)\begin{bmatrix} 4913 & 27 \\ -512 & 1331 \end{bmatrix}$$

```
(%i10)      3^x;
```

$$(\%o10)\begin{bmatrix} 129140163 & 27 \\ \frac{1}{6561} & 177147 \end{bmatrix}$$

Розмноження матриці на матрицю:

```
(%i11)      xy;
```

$$(\%o11)\ \begin{bmatrix} 3\,a+17\,\pi & 3\,b+17\,e \\ 11\,a-8\,\pi & 11\,b-8\,e \end{bmatrix}$$

```
(%i12)      yx;
```

$$(\%o12)\ \begin{bmatrix} 17\,\pi-8\,e & 3\,\pi+11\,e \\ 17\,a-8\,b & 11\,b+3\,a \end{bmatrix}$$

Очевидно, що для успішного перемноження матриці мають бути узгоджені за розмірами. Зведення до ступеня -1 дає зворотну матрицю:

```
(%i13)      x^^-1;
```

$$(\%o13)\ \begin{bmatrix} \frac{11}{211} & -\frac{3}{211} \\ \frac{8}{211} & \frac{17}{211} \end{bmatrix}$$

```
(%i14)      x.(x^^-1);
```

$$(\%o14)\ \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Варто звернути увагу, що операції x^^-1 та x^-1 дають різний результат!

Приклад:

```
(%i2) x^-1;
```

$$(\%o2)\ \begin{bmatrix} \frac{1}{17} & \frac{1}{3} \\ -\frac{1}{8} & \frac{1}{11} \end{bmatrix}$$

```
(%i3) x^^-1;
```

$$(\%o3)\ \begin{bmatrix} \frac{11}{211} & -\frac{3}{211} \\ \frac{8}{211} & \frac{17}{211} \end{bmatrix}$$

Функція $genmatrix$ повертає матрицю заданої розмірності, складену з елементів двоіндексного масиву. Синтаксис виклику:

- $genmatrix(a, i_2, j_2, i_1, j_1)$
- $genmatrix(a, i_2, j_2, i_1)$
- $genmatrix(a, i_2, j_2)$

Індекси i_1, j_1 і i_2, j_2 вказують лівий та правий нижній елементи матриці у вихідному масиві.

Приклад:

```
(%i1) h [i, j]: = 1 / (i + j - 1);
```

$$(\%o1)\qquad h_{i,j} := \frac{1}{i+j-1}$$

```
(%i2) genmatrix(h,3,3);
```

$$(\%o2)\ \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \end{bmatrix}$$

```
(%i3) array (a, fixnum, 2, 2);
```
$$(\%o3) \qquad a$$
```
(% i4)     a [1, 1]: % e;
```
$$(\%o4) \qquad e$$
```
(% i5)     a [2, 2]: % pi;
```
$$(\%o5) \qquad \pi$$
```
(%i6) genmatrix (a, 2, 2);
```
$$(\%o6) \begin{bmatrix} e & 0 \\ 0 & \pi \end{bmatrix}$$

Функція $zeromatrix$ повертає матрицю заданої розмірності, складену з нулів (синтаксис виклику $zeromatrix(m, n)$).

```
(%i7) zeromatrix(2,2);
```
$$(\%o7) \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Функція $ident$ повертає одиничну матрицю заданої розмірності (синтаксис $ident(n)$)

```
(% i9)     ident(2);
```
$$(\%o9) \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Функція $copymatrix(M)$ створює копію матриці M. Зверніть увагу, що присвоєння не створює копії матриці (як і присвоєння копії списку).

Приклад:

```
(%i1) a: matrix ([1,2], [3,4]);
```
$$(\%o1) \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$
```
(%i2) b:a;
```
$$(\%o2) \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$
```
(%i3) b [2,2]: 10;
```
$$(\%o3) \qquad 10$$
```
(% i4)     a;
```
$$(\%o4) \begin{bmatrix} 1 & 2 \\ 3 & 10 \end{bmatrix}$$

Привласнення нового значення елементу матриці b змінює значення відповідного елемента матриці a. Використання $copymatrix$ дозволяє уникнути цього ефекту.

Функції row і col дозволять витягти відповідно рядок та стовпець заданої матриці, отримуючи список. Синтаксис виклику:

- $row(M, i)$ - Повертає i -ю рядок;
- $col(M, i)$ - Повертає i -й стовпець.

Функції $addrow$ і $addcol$ додають до матриці рядок або стовпець відповідно. Синтаксис виклику:

- $addcol(M, list_1, ..., list_n)$
- $addrow(M, list_1, ..., list_n)$

Тут $list_1, ..., list_n$ — рядки, що додаються, або стовпці.

Приклад:

```
(%i1) a: matrix ([1,2], [3,4]);
```

$$(\%o1) \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

```
(%i2) b:addrow(a,[10,20]);
```

$$(\%o2) \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 10 & 20 \end{bmatrix}$$

```
(%i3) addcol(b,[x,y,z]);
```

$$(\%o3) \begin{bmatrix} 1 & 2 & x \\ 3 & 4 & y \\ 10 & 20 & z \end{bmatrix}$$

Функція $submatrix$ повертає нову матрицю, що складається з заданої підматриці. Синтаксис виклику:

- $submatrix(i_1, ..., i_m, M, j_1, ..., j_n)$
- $submatrix(i_1, ..., i_m, M)$
- $submatrix(M, j_1, ..., j_n)$

Підматриця будується так: з матриці M видаляються рядки $i_1, ..., i_m$ і $j_1, ..., j_n$.

приклад(використовуємо останній результат з попереднього прикладу, видаляємо третій рядок та третій стовпець):

```
(%i6) submatrix(3,%,3);
```

$$(\%o6) \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

Для заповнення матриці значеннями певної функції використовується функція $matrixmap$ (аналог $map, apply, fullmap$). Синтаксис виклику:

matrixmap(f,M). Функція matrixmap повертає матрицю з елементами i, j, рівними $f(M[i,j])$.

Приклад:

```
(%i1) a: matrix ([1,2], [3,4]);
```

$$(\%o1)\ \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

```
(%i2) f(x):=x^2;
```

$$(\%o2)\ \ f(x) := x^2$$

```
(%i3) matrixmap(f,a);
```

$$(\%o3)\ \begin{bmatrix} 1 & 4 \\ 9 & 16 \end{bmatrix}$$

Для роботи з матрицями існує ще багато функцій, але вони відносяться до вирішення різних завдань лінійної алгебри, тому обговорюються нижче, 3.2.

2.7.4 Математичні функції

Maxima має досить великий набір вбудованих математичних функцій. Перелік основних класів вбудованих функцій наведено нижче:

- тригонометричні функції: sin (синус), cos (Косінус), tan (тангенс), cot (Котангенс);
- зворотні тригонометричні функції: $asin$ (арксинус), $acos$ (Арккосинус), $atan$ (Арктангенс), $acot$ (Арккотангенс);
- sec (секанс, $sec(x) = \frac{1}{cos(x)}$), csc (Косеканс, $csc(x) = \frac{1}{sin(x)}$);
- $sinh$ (Гіперболічний синус), $cosh$ (гіперболічний косинус), $tanh$ (Гіперболічний тангенс), $coth$ (гіперболічний котангенс), $sech$ (Гіперболічний секанс), $cosh$ (Гіперболічний косеканс);
- log (натуральний логарифм);
- $sqrt$ (квадратний корінь);
- mod (Залишок від розподілу);
- abs (модуль);
- $min(x_1, ..., x_n)$і $max(x_1, ..., x_n)$— знаходження мінімального та максимального значення у списку аргументів;
- $sign$ (Визначає знак аргументу: pos - Позитивний, neg - Негативний, pnz - не визначений, $zero$ - значення дорівнює нулю);
- Спеціальні функції – функції Бесселя, гамма-функція, гіпергеометрична функція та ін;
- Еліптичні функції різних типів.

2.7.5 Обчислення та перетворення аналітичних виразів

Функція ev є основною функцією, яка обробляє вирази. Синтаксис виклику: $ev(expr, arg_1, ..., arg_n)$

Функція ev обчислює вираз expr в оточенні, що визначається аргументами $arg_1, ..., arg_n$. Аргументи можуть бути ключами (булівськими прапорами, присвоюваннями, рівняннями та функціями. Функція ev повертає результат (інший вираз).

У багатьох випадках можна опускати ім'я функції ev (Тобто застосовувати значення змінних до деякого виразу)

$expr, flag1, flag2, \ldots$

$expr, x = val1, y = val2, \ldots$

$expr, flag1, x = val1, y = val2, flag2, \ldots$

На вираз expr за замовчуванням діє функція спрощення. Необхідність виконання спрощення регулюється прапором $simp$ (якщо встановити $simp = false$, спрощення буде вимкнено). Крім того, використовують прапори $float$ і $numer$, Що визначають формат подання раціональних чисел (у вигляді дробів або з плаваючою точкою) та результатів обчислення математичних функцій. Прапор $pred$ визначає необхідність обчислення стосовно логічних виразів.

Аргументами ev можуть бути і вбудовані функції, що виконують спрощення або перетворення виразів ($expand, factor, trigexpand, trigreduce$) або функція $diff$.

Якщо вказані підстановки (у вигляді $x = val1$ або $x : val2$), то вони виконуються.

При цьому повторний виклик функції ev цілком здатний вкотре змінити вираз, тобто. обробка виразу не триває остаточно при одноразовому виклику функції ev.

Приклад:

```
(%i1) ev((a+b)^2,expand);
```

$$(\%o1) \qquad b^2 + 2\,a\,b + a^2$$

```
(%i2) ev((a+b)^2,a=x);
```

$$(\%o2) \qquad (x + b)^2$$

```
(%i3) ev((a+b)^2,a=x,expand,b=7)
```

$$(\%o3) \qquad x^2 + 14\,x + 49$$

Інший приклад показує застосування $diff$ до відкладеного обчислення похідної:

```
(%i1) sin(x) + cos(y) + (w+1)^2 + 'diff (sin(w), w);
```

$$(\%o1) \qquad cos\,(y) + sin\,(x) + \frac{d}{d\,w}\,sin\,(w) + (w+1)^2$$

```
(%i2) ev (%, sin, expand, diff, x = 2, y = 1);
```

$$(\%o2) \quad cos\,(w) + w^2 + 2\,w + cos\,(1) + 1.909297426825682$$

Прапор $simp$ дозволяє або забороняє спрощення виразів. Спочатку він дорівнює $true$, якщо встановити його рівним $false$, то спрощення не проводитимуться:

```
(%i1) f:a+2*a+3*a+4*a;
```
$$(\%o1) \qquad 10\,a$$
```
(%i2) simp: false;
```
$$(\%o2) \qquad false$$
```
(%i3) f:a+2*a+3*a+4*a;
```
$$(\%o3) \qquad a+2\,a+3\,a+4\,a$$

Функцію ev не обов'язково вказувати явно, наприклад:

```
(%i3) x+y, x: a+y, y: 2;
```
$$(\%o3) \qquad y+a+2$$

Оператор примусового обчислення, позначений двома апострофами, є синонімом до функції ev **(вираз)**. Сама функція ev надає набагато ширші можливості, ніж просте примусове обчислення заданого виразу: вона може приймати довільне число аргументів, перший з яких — вираз, що обчислюється, а інші — спеціальні опції, які якраз і впливають на те, як саме буде здійснюватися обчислення.

У термінології Maxima необчислена форма виразу називається "noun form", обчислена - "verb form". Зберігаючи лінгвістичні паралелі, російською це можна перекласти як "недосконала форма" і "скоєна форма". Значення виразу, що вводиться в Maxima закономірно зберігається до його обчислення (тобто в недосконалій формі). А значення виразу, що виводиться, — після (тобто в скоєній). Іншими словами, тут є природний порядок "введення - обчислення - висновок".

Функція $factor$ факторизує (тобто представляє у вигляді добутку деяких співмножників) заданий вираз (функція $gfactor$ - Аналогічно, але на безлічі комплексних чисел і виразів).

Приклад:

```
(%i1) x^3-1, factor;
```
$$(\%o1) \qquad (x-1)\left(x^2+x+1\right)$$
```
(%i2) factor(x^3-1);
```
$$(\%o2) \qquad (x-1)\left(x^2+x+1\right)$$

Ще приклади факторизації різних виразів:

```
(%i3) factor (-8 * y - 4 * x + z ^ 2 * (2 * y + x));
```
$$(\%o3) \qquad (2\,y+x)\,(z-2)\,(z+2)$$
```
(% i4)     factor (2^63 - 1);
```
$$(\%o4) \qquad 7^2\,73\,127\,337\,92737\,649657$$
```
(% i5)     factor (1 + % e ^ (3 * x));
```
$$(\%o5) \qquad \left(e^x+1\right)\left(e^{2\,x}-e^x+1\right)$$

прикладвикористання функції $gfactor$:

```
(%i6) gfactor(x^2+a^2);
```
$$(\%o6) \qquad (x - i\,a)\,(x + i\,a)$$
```
(%i7) gfactor(x^2+2*%i*x*aa^2);
```
$$(\%o7) \qquad (x + i\,a)^2$$

Функція $factorsum$ факторизує окремі доданки у виразі.

```
(% i8)     expand ((x + 1) * ((u + v) ^ 2 + a * (w + z) ^
2));
```
$$(\%o8)$$
$$axz^2+az^2+2awxz+2awz+aw^2x+v^2x+2uvx+u^2x+aw^2+v^2+2uv+u^2$$
```
(% i9)     factorsum(%);
```
$$(\%o9) \qquad (x+1)\left(a\,(z+w)^2+(v+u)^2\right)$$

Функція $gfactorsum$ відрізняється від $factorsum$ тим самим, чим $gfactor$ відрізняється від $factor$:

```
(%i10)     gfactorsum(
a^3+3*a^2*b+3*a*b^2+b^3+x^2+2*%i*x*yy^2);
```
$$(\%o10) \qquad (b+a)^3-(y-i\,x)^2$$

Функція $expand$ розкриває дужки, виконує множення, зведення у ступінь, наприклад:

```
(%i1) expand((xa)^3);
```
$$(\%o1) \qquad x^3-3\,a\,x^2+3\,a^2\,x-a^3$$
```
(%i2) expand((xa)*(yb)*(zc));
```
$$(\%o2) \quad x\,y\,z-a\,y\,z-b\,x\,z+a\,b\,z-c\,x\,y+a\,c\,y+b\,c\,x-a\,b\,c$$
```
(%i3) expand((xa)*(yb)^2);
```
$$(\%o3) \qquad x\,y^2-a\,y^2-2\,b\,x\,y+2\,a\,b\,y+b^2\,x-a\,b^2$$

Функція $combine$ поєднує доданки з ідентичним знаменником

```
(% i5)     combine(x/(1+x^2)+y/(1+x^2));
```
$$(\%o5) \qquad \frac{y+x}{x^2+1}$$

Функція $xthru$ наводить вираз до спільного знаменника, не розкриваючи дужок і не намагаючись факторизувати доданки

```
(%i6) xthru( 1/(x+y)^10+1/(x+y)^12 );
```
$$(\%o6) \qquad \frac{(y+x)^2+1}{(y+x)^{12}}$$
```
(%i1) ((x+2)^20 - 2*y)/(x+y)^20 + (x+y)^(-19) -
x/(x+y)^20;
```
$$(\%o1) \qquad \frac{1}{(y+x)^{19}}+\frac{(x+2)^{20}-2\,y}{(y+x)^{20}}-\frac{x}{(y+x)^{20}}$$
```
(%i2) xthru (%);
```

(%o2) $$\frac{(x+2)^{20}-y}{(y+x)^{20}}$$

Функція $multthru$ множить кожне доданок у сумі на множник, причому при множенні дужки у виразі не розкриваються. Вона допускає два варіанти синтаксису:

- $multthru(mult, sum);$
- $multthru(expr);$

В останньому випадку вираз expr включає множник і суму (див. (%i4) у прикладі нижче).

Приклад:

```
(%i1) x/(xy)^2 - 1/(xy) - f(x)/(xy)^3;
```

(%o1) $$-\frac{1}{x-y}+\frac{x}{(x-y)^2}-\frac{f(x)}{(x-y)^3}$$

```
(%i2) multthru ((xy)^3, %);
```

(%o2) $$-(x-y)^2+x\,(x-y)-f(x)$$

```
(%i3) ((a+b)^10*s^2 + 2*a*b*s + (a*b)^2)/(a*b*s^2);
```

(%o3) $$\frac{(b+a)^{10}\,s^2+2\,a\,b\,s+a^2\,b^2}{a\,b\,s^2}$$

```
(% i4)      multthru (%);
```

(%o4) $$\frac{2}{s}+\frac{a\,b}{s^2}+\frac{(b+a)^{10}}{a\,b}$$

Функції assume (введення обмежень) та $forget$ (зняття обмежень) дозволяють керувати умовами виконання (контекстом) інших функцій та операторів.

Приклад:

```
(% i20)     sqrt(x^2);
(%o20)            |x|
(%i21)      assume (x<0);
(%o21)            [ x < 0 ]
(%i22)      sqrt(x^2);
(%o22)            -x
(%i23)      forget(x<0);
(%o23)            [ x < 0 ]
(%i24)      sqrt (x ^ 2);
(%o24)            |x|
```

Функція $divide$ дозволяє обчислити приватне та залишок від розподілу одного багаточлена на інший:

```
(%i1) divide(x^3-2,x-1);
```

(%o1) $[x^2+x+1,-1]$

Перший елемент отриманого списку - приватний, другий - залишок від поділу.

Функція gcd дозволяє знайти найбільший спільний дільник багаточленів.

Підстановки здійснюються функцією $subst$. Виклик цієї функції: $subst(a,b,c)$ (підставляємо a замість b у виразі c).

Приклад:

```
(%i1) subst (a, x + y, x + (x + y) ^ 2 + y);
```

(%o1) $y+x+a^2$

2.7.6 Перетворення раціональних виразів

Для виділення чисельника та знаменника дробових виразів використовуються функції num і $denom$:

```
(%i1) expr:(x^2+1)/(x^3-1);
```

(%o1) $$\frac{x^2+1}{x^3-1}$$

```
(%i2) num (expr);
```

(%o2) x^2+1

```
(%i3) denom(expr);
```

(%o3) x^3-1

Функція rat наводить вираз до канонічного уявлення. Вона полегшує будь-який вираз, розглядаючи його як дробораціональну функцію, тобто. працює з операціями "+", "-", "*", "/" та зі зведенням у цілий ступінь.

Синтаксис виклику:

- $rat(expr)$
- $rat(expr, x_1, ..., x_n)$

Змінні впорядковуються відповідно до списку $x_1, ..., x_n$. При цьому вид відповіді залежить від способу впорядкування змінних. Спочатку змінні впорядковані в алфавітному порядку.

прикладвикористання rat:

```
(%i1) ((x - 2*y)^4/(x^2 - 4*y^2)^2 + 1)*(y + a)*(2*y + x)
/ (4*y^2 + x ^2);
```

(%o1) $$\frac{(y+a)\,(2\,y+x)\left(\frac{(x-2\,y)^4}{\left(x^2-4\,y^2\right)^2}+1\right)}{4\,y^2+x^2}$$

```
(%i2) rat(%);
```

(%o2) $$\frac{2\,y+2\,a}{2\,y+x}$$

Після вказівки порядку використання змінних одержуємо такий вираз:

```
(%i3) rat(%o1,y,a,x);
```

$$(\%o3) \qquad \frac{2\,a+2\,y}{x+2\,y}$$

Функція $ratvars$ дозволяє змінити алфавітний порядок переваги змінних, прийнятий за умовчанням. Виклик $ratvars(z,y,x,w,v,u,t,s,r,q,p,o,n,m,l,k,j,i,h,g,f,e,d,c,b,a)$ змінює порядок переваги точно на зворотний, а виклик $ratvars(m,n,a,b)$ впорядковує змінні m,n,a,b у порядку зростання пріоритету.

Прапор $ratfac$ включає або вимикає часткову факторизацію виразів під час зведення їх до стандартної форми (CRE). Спочатку встановлено значення $false$. Якщо встановити значення $true$, то проводитиметься часткова факторизація.

Функція $ratsimp$ приводить усі частини (у тому числі аргументи функцій) виразу, який не є дробово-раціональною функцією, до канонічного подання, виробляючи спрощення, які не виконує функція rat. Повторний виклик функції у випадку може змінити результат, тобто. не обов'язково спрощення проводиться остаточно. Застосуванням спрощення до експоненційних виразів керує прапор $ratsimexpons$, за умовчанням рівний $false$, якщо його встановити у $true$, Спрощення застосовується і до показників ступеня або експоненти.

```
(%i1) sin (x/(x^2 + x)) = exp ((log(x) + 1)^2 -
log(x)^2);
```

$$(\%o1) \qquad sin\left(\frac{x}{x^2+x}\right) = e^{(log(x)+1)^2-log(x)^2}$$

```
(%i2) ratsimp(%);
```

$$(\%o2) \qquad sin\left(\frac{1}{x+1}\right) = e\,x^2$$

```
(%i3) ((x - 1) ^ (3/2) - (x + 1) * sqrt (x - 1)) / sqrt
((x - 1) * (x + 1));
```

$$(\%o3) \qquad \frac{(x-1)^{\frac{3}{2}} - \sqrt{x-1}\,(x+1)}{\sqrt{(x-1)\,(x+1)}}$$

```
(% i4)     ratsimp(%);
```

$$(\%o4) \qquad -\frac{2\sqrt{x-1}}{\sqrt{x^2-1}}$$

```
(% i5)     x^(a + 1/a), ratsimpexpons: true;
```

$$(\%o5) \qquad x^{\frac{a^2+1}{a}}$$

Функція $fullratsimp$ викликає функцію $ratsimp$ доти, доки вираз не перестане змінюватися.

Приклад:

```
(%i1) expr: (x^(a/2) + 1)^2*(x^(a/2) - 1)^2/(x^a - 1);
```

$$(\%o1) \quad \frac{\left(x^{\frac{a}{2}}-1\right)^2\left(x^{\frac{a}{2}}+1\right)^2}{x^a-1}$$

```
(%i2) ratsimp(expr);
```

$$(\%o2) \quad \frac{x^{2\,a}-2\,x^a+1}{x^a-1}$$

```
(%i3) fullratsimp(expr);
```

$$(\%o3) \quad x^a-1$$

```
(% i4)     rat(expr);
```

$$(\%o4) \quad \frac{\left(x^{\frac{a}{2}}\right)^4-2\left(x^{\frac{a}{2}}\right)^2+1}{x^a-1}$$

Приклад впливу прапора $ratsimpexponds$ на результат обчислень:

```
(%i1) fullratsimp( exp((x^(a/2)-1)^2 *(x^(a/2)+1)^2 /
(x^a-1) ) );
```

$$(\%o1) \quad e^{\frac{x^{2\,a}}{x^a-1}-\frac{2\,x^a}{x^a-1}+\frac{1}{x^a-1}}$$

```
(%i2) ratsimpexpons:true;
```

$$(\%o2) \quad true$$

```
(%i3) fullratsimp( exp((x^(a/2)-1)^2 *(x^(a/2)+1)^2 /
(x^a-1) ) );
```

$$(\%o3) \quad e^{x^a-1}$$

Функція $ratexpand$ розкриває дужки у виразі. Відрізняється від функції $expand$ тим, що наводить вираз до канонічної форми, тому відповідь може відрізнятись від результату застосування функції $expand$:

```
(%i1) ratexpand ((2 * x - 3 * y) ^ 3);
```

$$(\%o1) \quad -27\,y^3+54\,x\,y^2-36\,x^2\,y+8\,x^3$$

```
(%i2) expr: (x - 1) / (x + 1) ^ 2 + 1 / (x - 1);
```

$$(\%o2) \quad \frac{x-1}{(x+1)^2}+\frac{1}{x-1}$$

```
(%i3) expand(expr);
```

$$(\%o3) \quad \frac{x}{x^2+2\,x+1}-\frac{1}{x^2+2\,x+1}+\frac{1}{x-1}$$

```
(% i4)     ratexpand(expr);
```

$$(\%o4) \quad \frac{2\,x^2}{x^3+x^2-x-1}+\frac{2}{x^3+x^2-x-1}$$

Підстановка у раціональних виразах здійснюється функцією$ratsubst$. Синтаксис виклику:$ratsubst(a,b,c)$Виразaпідставляється замість виразуbу виразіc(bможе бути сумою, твором, ступенем тощо).

прикладвикористання$ratsubst$:

```
(%i1)   ratsubst (a, x*y^2, x^4*y^3 + x^4*y^8);
```

$$(\%o1) \qquad a\,x^3\,y + a^4$$

```
(%i2)   cos(x)^4 + cos(x)^3 + cos(x)^2 + cos(x) + 1;
```

$$(\%o2) \qquad cos\,(x)^4 + cos\,(x)^3 + cos\,(x)^2 + cos\,(x) + 1$$

```
(%i3)   ratsubst (1 - sin(x)^2, cos(x)^2, %);
```

$$(\%o3) \quad sin\,(x)^4 - 3\,sin\,(x)^2 + cos\,(x)\,\left(2 - sin\,(x)^2\right) + 3$$

2.7.7 Перетворення тригонометричних виразів

Функція$trigexpand$розкладає всі тригонометричні та гіперболічні функції від сум та творів у комбінації відповідних функцій одиничних кутів та аргументів. Для посилення контролю користувача один виклик$trigexpand$виконує спрощення однією рівні. Для керування обчисленням є прапор $trigexpand$. Спочатку прапор$trigexpand$встановлений у$false$. Якщо прапор$trigexpand$встановити в$true$, то функція$trigexpand$буде працювати доти, доки вираз не перестане змінюватися.

```
(%i1)   x + sin (3 * x) / sin (x), trigexpand = true,expand;
```

$$(\%o1) \qquad -sin\,(x)^2 + 3\,cos\,(x)^2 + x$$

```
(%i2)   trigexpand(sin(10*x+y));
```

$$(\%o2) \qquad cos\,(10\,x)\;sin\,(y) + sin\,(10\,x)\;cos\,(y)$$

```
(%i3)   trigexpand(sin(3*x)+cos(4*x));
```

$$(\%o3)$$

$$sin\,(x)^4 - sin\,(x)^3 - 6\,cos\,(x)^2\,sin\,(x)^2 + 3\,cos\,(x)^2\,sin\,(x) + cos\,(x)^4$$

Функція trigreduce згортає всі твори тригонометричних та гіперболічних функцій у комбінації відповідних функцій від сум. Функція працює не до кінця, тому повторний виклик може змінити вираз. При виклику функції trigreduce(expr,x) перетворення здійснюються щодо функцій x.

Приклади:

```
(% i8)      trigreduce(cos(x)^4 + cos(x)^3 + cos(x)^2 +
cos(x) + 1);
```

$$(\%o8)$$

$$\frac{cos\,(4\,x) + 4\,cos\,(2\,x) + 3}{8} + \frac{cos\,(3\,x) + 3\,cos\,(x)}{4} + \frac{cos\,(2\,x) + 1}{2} + cos\,(x) + 1$$

```
(% i9)      trigreduce(-sin(x)^2+3*cos(x)^2+x);
```

$$(\%o9) \qquad \frac{cos\,(2\,x)}{2} + 3\,\left(\frac{cos\,(2\,x)}{2} + \frac{1}{2}\right) + x - \frac{1}{2}$$

Функція $trigsimp$ спрощує тригонометричні та гіперболічні вирази, застосовуючи до них правила $sin(x)^2 + cos(x)^2 = 1$ і $cosh(x)^2 - sinh(x)^2 = 1$.

Пример :

```
(%i1) trigsimp(sin(x)^2+3*cos(x)^2);
```

$$(\%o1) \qquad 2\cos(x)^2 + 1$$

```
(%i2) trigsimp(sinh(x)^2+3*cosh(x)^2);
```

$$(\%o2) \qquad 4\cosh(x)^2 - 1$$

Функція $trigrat$ (синтаксис виклику $trigrat(expr)$) наводить заданий тригонометричний вираз $expr$ до канонічної спрощеної квазілінійної форми. Цей вираз розглядається як раціональний, що містить sin, cos, tan, аргументи яких лінійні форми деяких змінних та $\frac{\pi}{n}$ (n - ціле). Завжди, коли можливо, заданий вираз лінеаризується.

Приклад:

```
(%i1) trigrat((1+sin(2*b)-cos(2*b))/sin(b));
```

$$(\%o1) \qquad 2\sin(b) + 2\cos(b)$$

2.7.8 Перетворення статечних та логарифмічних виразів

Функція radcan спрощує вирази, що містять експоненти, логарифми та радикали, шляхом перетворення до форми, яка є канонічною для широкого класу виразів. Змінні у виразі впорядковуються. Еквівалентні вирази в цьому класі не обов'язково однакові, але їхня різниця спрощується застосуванням radcan до нуля.

Приклади:

```
(%i1) (log(x+x^2)-log(x))^a/log(1+x)^(a/2);
```

$$(\%o1) \qquad \frac{\left(log\left(x^2+x\right) - log\left(x\right)\right)^a}{log\left(x+1\right)^{\frac{a}{2}}}$$

```
(%i2) radcan(%);
```

$$(\%o2) \qquad log\left(x+1\right)^{\frac{a}{2}}$$

```
(%i10)      (%e^x-1)/(1+%e^(x/2));
```

$$(\%o10) \qquad \frac{e^x - 1}{e^{\frac{x}{2}} + 1}$$

```
(%i11)      radcan(%);
```

$$(\%o11) \qquad e^{\frac{x}{2}} - 1$$

Функція $logcontract(expr)$ рекурсивно сканує вираз $expr$, перетворюючи вирази виду $a1 * log(b1) + a2 * log(b2) + c$ до форми $log(ratsimp(b1^{a1} * b2^{a2})) + c$.

Приклад:

```
(%i1) 2*(a*log(x)+3*b*log(y));
```
$$(\%o1) \qquad 2\,(3\,b\,log\,(y)+a\,log\,(x))$$
```
(%i2) logcontract(%);
```
$$(\%o2) \qquad b\,log\left(y^6\right)+a\,log\left(x^2\right)$$

Якщо оголосити змінну n цілою (використовуючи $declare(n, integer)$), функція $logcontract$ дозволяє включити цю змінну до показника ступеня:
```
(%i1) declare(n,integer);
```
$$(\%o1) \qquad done$$
```
(%i2) logcontract(3*a*n*log(x));
```
$$(\%o2) \qquad a\,log\left(x^{3\,n}\right)$$

2.7.9 Користувальницькі функції

Для запису функції необхідно вказати її назву, а потім у круглих дужках записати через кому значення аргументів. Якщо значенням аргументу є список, він полягає у квадратні дужки, а елементи списку також поділяються комами.

Приклад:
```
sin(x);
integrate(sin(x),x,-5,5);
plot2d([sin(x)+3,cos(x)],[x,-%pi,%pi],[y,-5,5]);
```

Користувач може встановити власні функції. Для цього спочатку вказується назва функції, у дужках перераховуються назви аргументів, після знаків := (двокрапка і одно) слідує опис функції. Після завдання функція користувача викликається точно так, як і вбудовані функції Maxima.

Приклад:
```
(%i44)     f(x):=x^2;
```
$$(\%o44) \qquad f(x) := x^2$$
```
(%i45)     f(3 + 7);
```
$$(\%o45) \qquad 100$$

Не використовуйте функції назви, зарезервовані для вбудованих функцій Maxima. Для створення функцій також використовується вбудована функція $define$ яка дозволяє перетворити вираз у функцію. Синтаксис виклику $define$ досить різноманітний:

- $define(f(x_1, ..., x_n), expr)$
- $define(f[x_1, ..., x_n], expr)$
- $define(funmake(f, [x_1, ..., x_n]), expr)$
- $define(arraymake(f, [x_1, ..., x_n]), expr)$
- $define(ev(expr_1), expr_2)$

Варіанти виклику функції $define$ розрізняються, який саме об'єкт створюється: ординарна функція (аргументи у круглих дужках) чи масив

(аргументи у квадратних дужках). Якщо перший аргумент – оператори $funmake, arraymake$, то функція створюється та обчислюється (аналогічно і ev).

Приклади:

Ординарна функція:

```
(%i1) expr: cos(y) - sin(x);
```
$$(\%o1) \qquad cos\,(y) - sin\,(x)$$
```
(%i2) define (F1 (x, y), expr);
```
$$(\%o2) \qquad F1\,(x, y) := cos\,(y) - sin\,(x)$$
```
(%i3) factor(F1(a,b));
```
$$(\%o3) \qquad cos\,(b) - sin\,(a)$$

Створення функції-масиву:

```
(%i1) define (G2 [x, y], xy - yx);
```
$$(\%o1) \qquad G2_{x,y} := x.y - y.x$$

Створення масиву:

```
(%i2) define (arraymake (F, [u]), cos(u) + 1);
```
$$(\%o2) \qquad F_u := cos\,(u) + 1$$

Використання функції ev для завдання функції користувача:

```
(%i3) define (ev (foo(x, y)), sin(x) - cos(y));
```
$$(\%o3) \qquad foo\,(x, y) := sin\,(x) - cos\,(y)$$

2.8 Розв'язання задач елементарної математики

2.8.1 Знаходження коренів рівнянь та систем рівнянь алгебри

Розв'язання рівнянь алгебри та їх систем здійснюється за допомогою функції $solve$, як параметри. У перших квадратних дужках вказується список рівнянь через кому, у других - список змінних, через кому (або дещо спрощені форми запису):

$solve(expr, x)$ - Вирішення одного рівняння щодо змінної x;

$solve(expr)$ - Рішення рівняння з однією невідомою та числовими коефіцієнтами;

$solve([eqn_1, ..., eqn_n], [x_1, ..., x_n])$ - Вирішення системи рівнянь.

Приклади:

Вирішення одного рівняння з одним невідомим

```
(%i7) solve(x^2-5*x+4);
```
$$(\%o7) \qquad [x = 1, x = 4]$$

Розв'язання одного рівняння у символьному вигляді:

```
(%i2) solve([xa/x+b], [x]);
```
$$(\%o2) \qquad [x = -\frac{\sqrt{b^2 + 4\,a} + b}{2}, x = \frac{\sqrt{b^2 + 4\,a} - b}{2}]$$

Розв'язання системи рівнянь у символьному вигляді:

```
(%i10)      solve([x*y/(x+y)=a,x*z/(x+z)=b,y*z/(y+z)=c],
[x,y,z]);
```

$$(\%o10)[[x=0,y=0,z=0],[x=\frac{2abc}{(b+a)c-ab},y=\frac{2abc}{(b-a)c+ab},z=-\frac{2abc}{(b-a)c-ab}]]$$

В останньому прикладі рішень кілька, і Maxima видає результат у вигляді списку.

Функція $solve$ застосовна і на вирішення тригонометрических рівнянь. При цьому у разі безлічі рішень у тригонометричних рівнянь видається відповідне повідомлення тільки одне з рішень.

Приклад:

```
(% i13) solve ([sin (x) = 0], [x]);

solve: використовуючи arc-trig функцій, щоб отримати
рішення.
Кілька рішень буде втрачено.
```

$$(\%o13)\qquad [x=0]$$

Також Maxima дозволяє знаходити комплексне коріння

```
(%i18)      solve([x^2+x+1], [x]);
```

$$(\%o18)\qquad [x=-\frac{\sqrt{3}\,i+1}{2},x=\frac{\sqrt{3}\,i-1}{2}]$$

2.9 Побудова графіків та поверхонь

Для виведення графіків на екран або на друк за допомогою Maxima є кілька варіантів форматів і, відповідно, програм виведення графіки, а саме:

- $openmath$ (Tcl/Tk програма з графічним інтерфейсом користувача; елемент xMaxima)
- $gnuplot$ (Потужна утиліта для побудови графіків, обмін з Maxima - через канал)
- $mgnuplot$ (Tk-інтерфейс до $gnuplot$ із рудиментарним графічним інтерфейсом користувача; включений до дистрибутиву Maxima)
- $wxMaxima$ (вбудовані можливості $frontend$ -А до Maxima)

Усі варіанти інтерфейсу (крім wxMaxima) для побудови графіків використовують дві базові функції: $plot2d$ (побудова двовимірних графіків) та $plot3d$ (Побудова тривимірних графіків).

При використанні wxMaxima крім них використовуються ще дві аналогічні команди: $wxplot2d$ і $wxplot3d$. Усі команди дозволяють вивести графік на екран, або (залежно від параметрів функції) у файл.

2.9.1 Побудова графіка явної функції y = f(x)

Графік функції $y = f(x)$ на відрізку $[a, b]$ можна побудувати за допомогою функції $plot2d(f(x), [x, a, b], \text{опции})$ або $plot2d(f(x), [x, a, b], [y, c, d], \text{опции})$. Опції не обов'язкові, проте, зміни властивостей графіка їх потрібно задавати. Параметр $[y, c, d]$ можна не ставити, тоді висота графіка вибирається за умовчанням. Побудуємо графік функції $y = sin(x)$ на відрізку $[-4\pi, 4\pi]$.

```
(%i2) plot2d(sin(x), [x, -4*%pi, 4*%pi]);
(%i3) plot2d(sin(x), [x, -4*%pi, 4*%pi],[y,-2,2]);
```

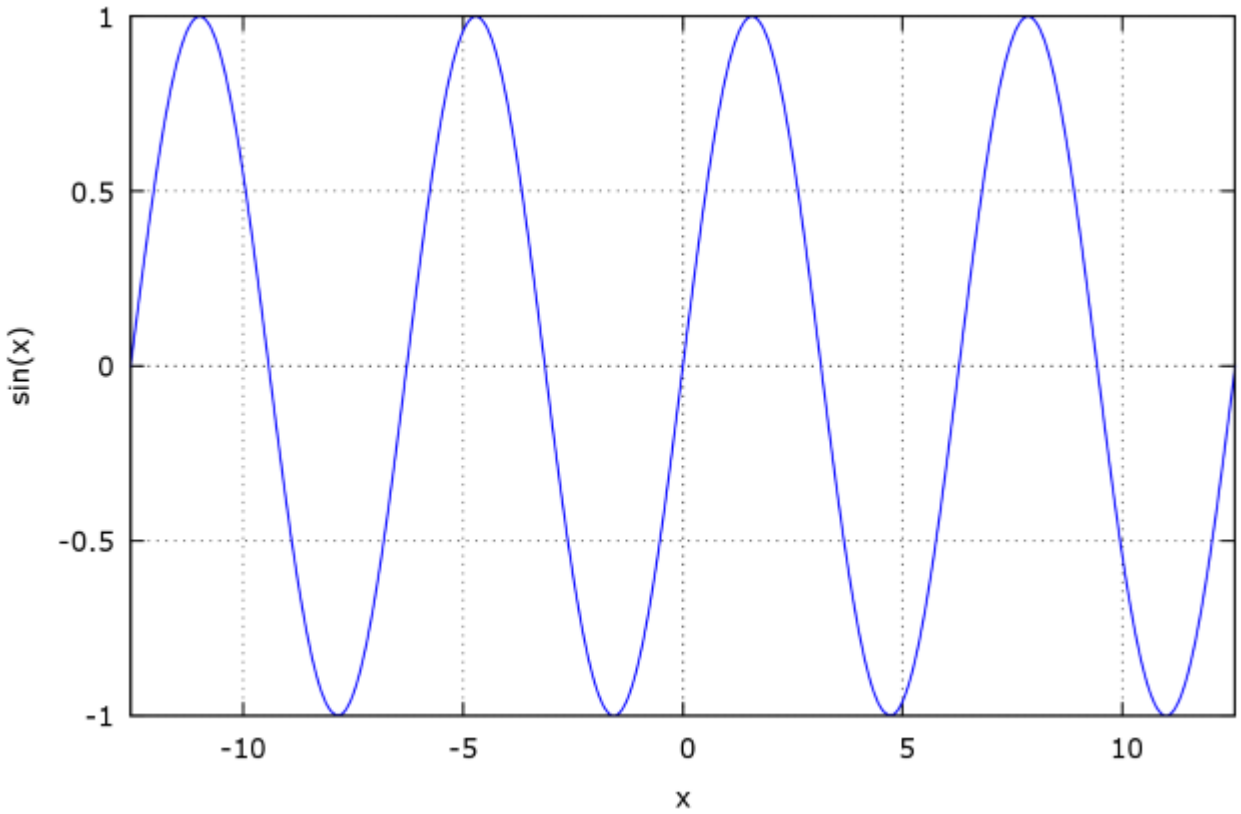
Результати наведено на рис. 2.1, рис. 2.2

2.9.2 Побудова графіків функцій, заданих параметрично

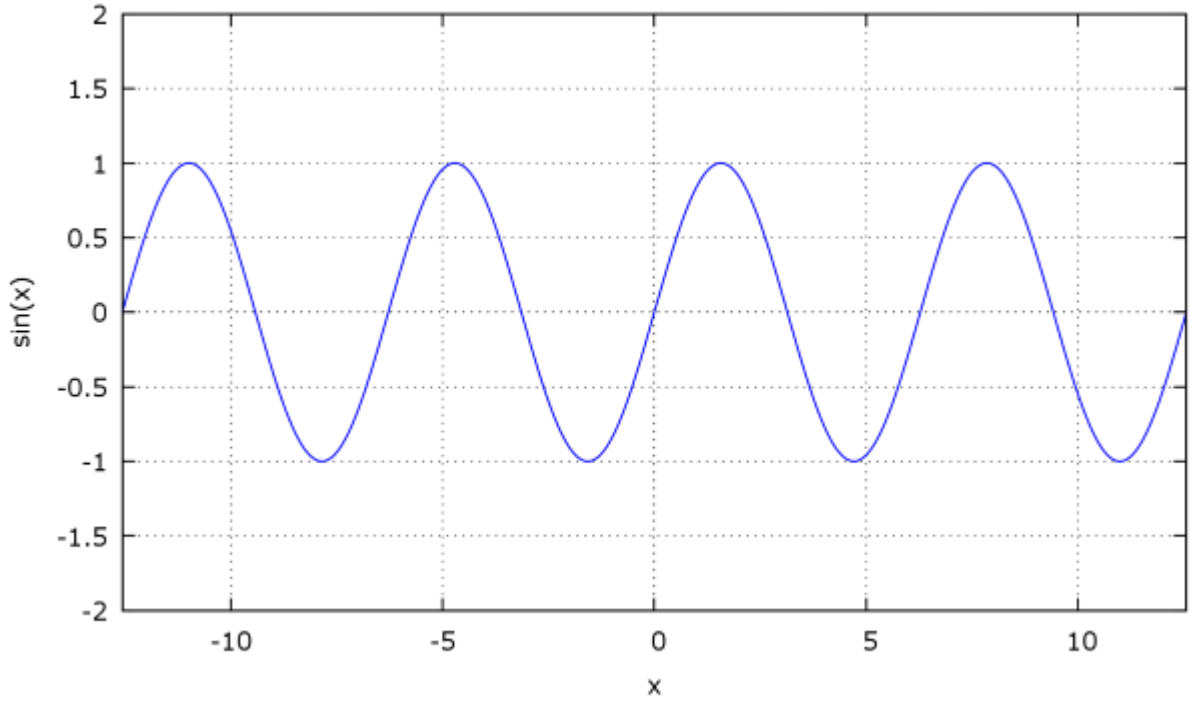
Для побудови графіків функцій, заданих параметрично, використовується опція $parametric$. Для побудови графіка вказується область зміни параметра. Приклад графіка найпростішої параметричної функції на рис. 2.3.

Команда побудови графіка:

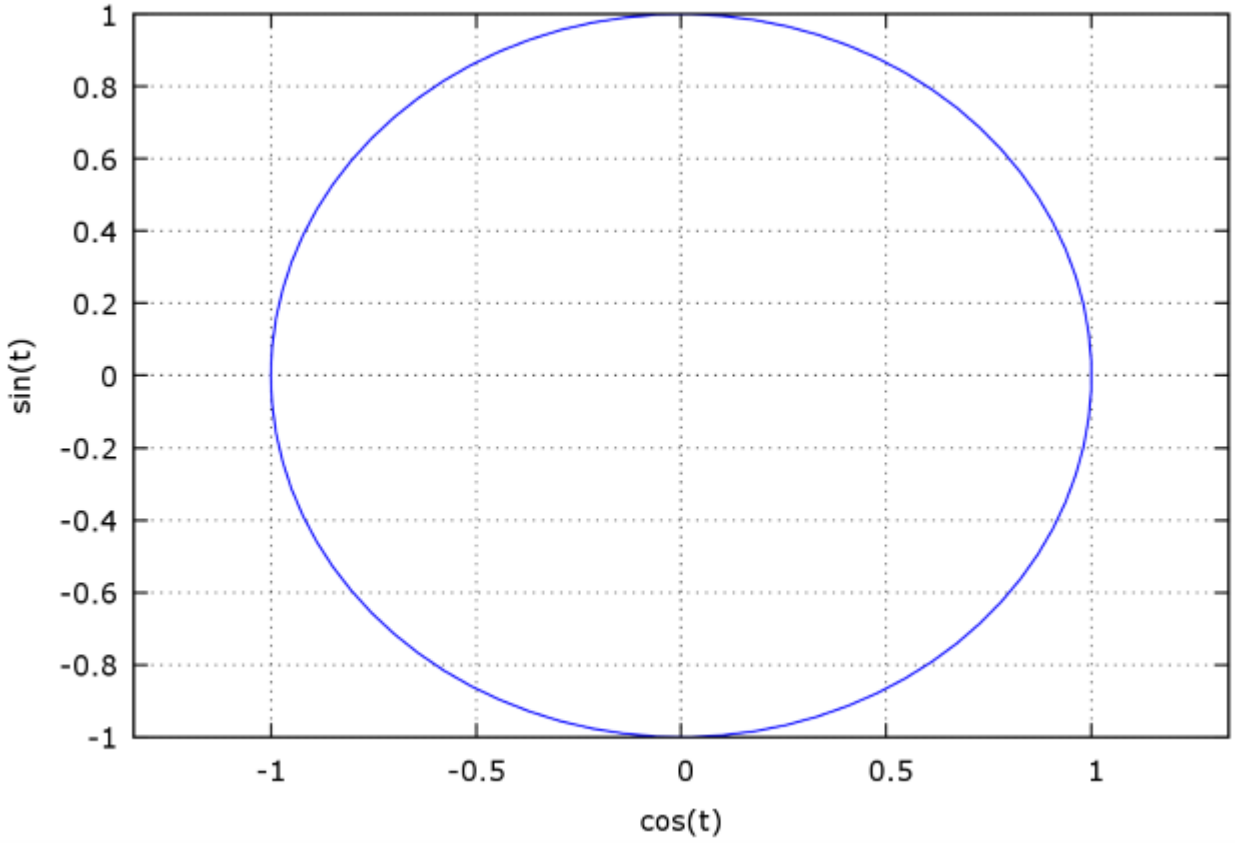
```
plot2d ([parametric, cos(t), sin(t), [t,-%pi,%pi],
[nticks,80]],[x, -4/3, 4/3])
```



Мал. 2.1.Найпростіша команда побудови графіка



Мал. 2.2.Найпростіша команда побудови графіка із зазначенням інтервалу по осі Oy



Мал. 2.3.Найпростіша команда побудови графіка функції, заданої параметрично

Опція $ntics$ вказує кількість точок, якими проводиться крива.

Розглянемо деякі опції.

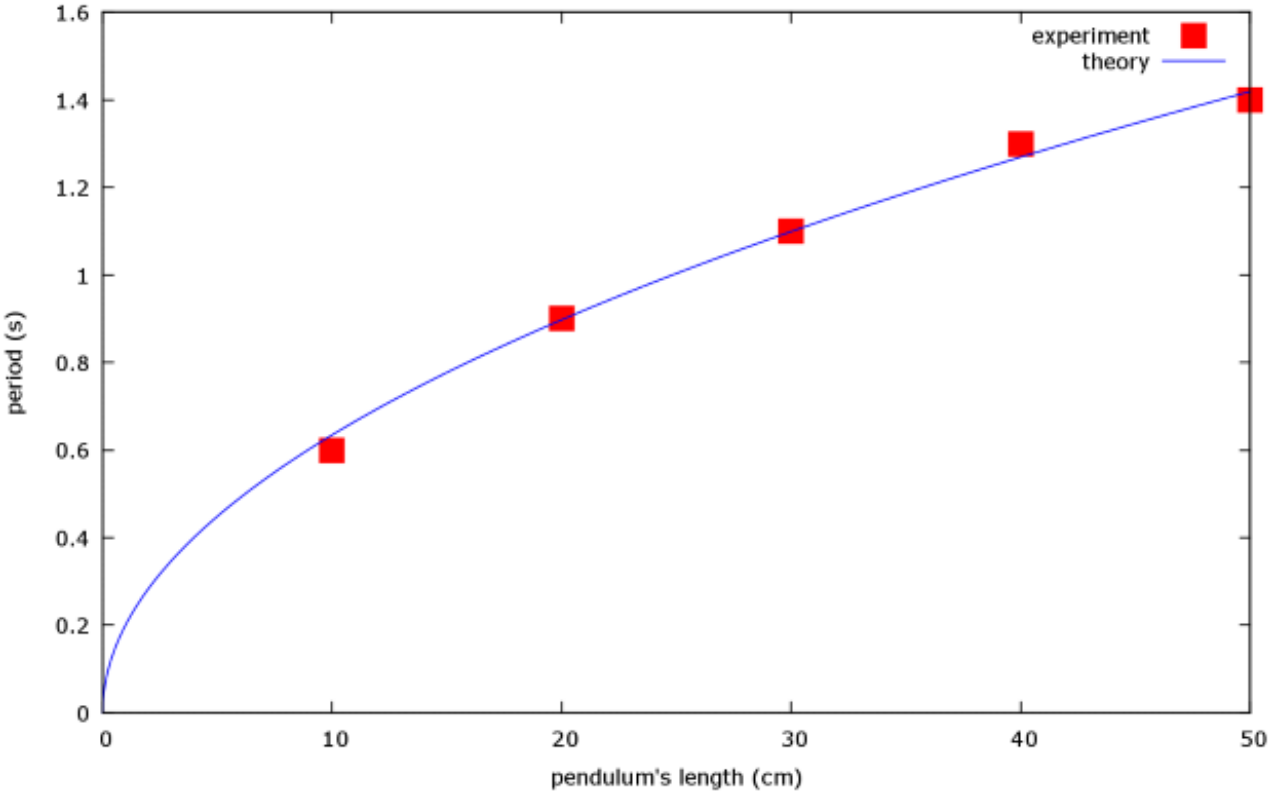
Опції вказуються як аргументів функції $plot2d$ у квадратних дужках. Можливе встановлення легенди, міток на осях, кольору та стилю графіка. Застосування кількох опцій характеризує наступний приклад:

```
(%i17)plot2d([[discrete,xy], 2*%pi*sqrt(l/980)],
[l,0,50],
             [style, [points,5,2,6], [lines,1,1]],
               [legend, experiment, theory],
                [xlabel,"pendulum's length (cm)"],
                                [ylabel, "period (s)"]);
```

У цьому прикладі в одних осях будуються два графіки. Перший $([discrete, xy])$ будується у вигляді крапок по масиву xy із зазначенням стилю $points$. Другий будується за рівнянням функції $2 * \%pi * sqrt(l/980)$ із зазначенням стилю $lines$. Опція $legend$ вказує підписи кривих, опції $xlabel$ і $ylabel$ - Підписи осей. Результат наведено на рис. 2.4.

Формування масивів для побудови графіка здійснюється так:

```
(%i12)      xx:[10, 20, 30, 40, 50];
(%i13)      yy: [.6, .9, 1.1, 1.3, 1.4];
(%i14)      xy:[[10,.6], [20,.9], [30,1.1], [40,1.3],
                                                [50,1.4]];
```



Мал. 2.4.Поєднання на одному графіку дії серії опцій

Можна комбінувати в одних осях графіки кривих різного типу: $y = f(x)$ або параметричні

$$\begin{cases} x = \varphi(t), \\ y = \psi(t). \end{cases}$$

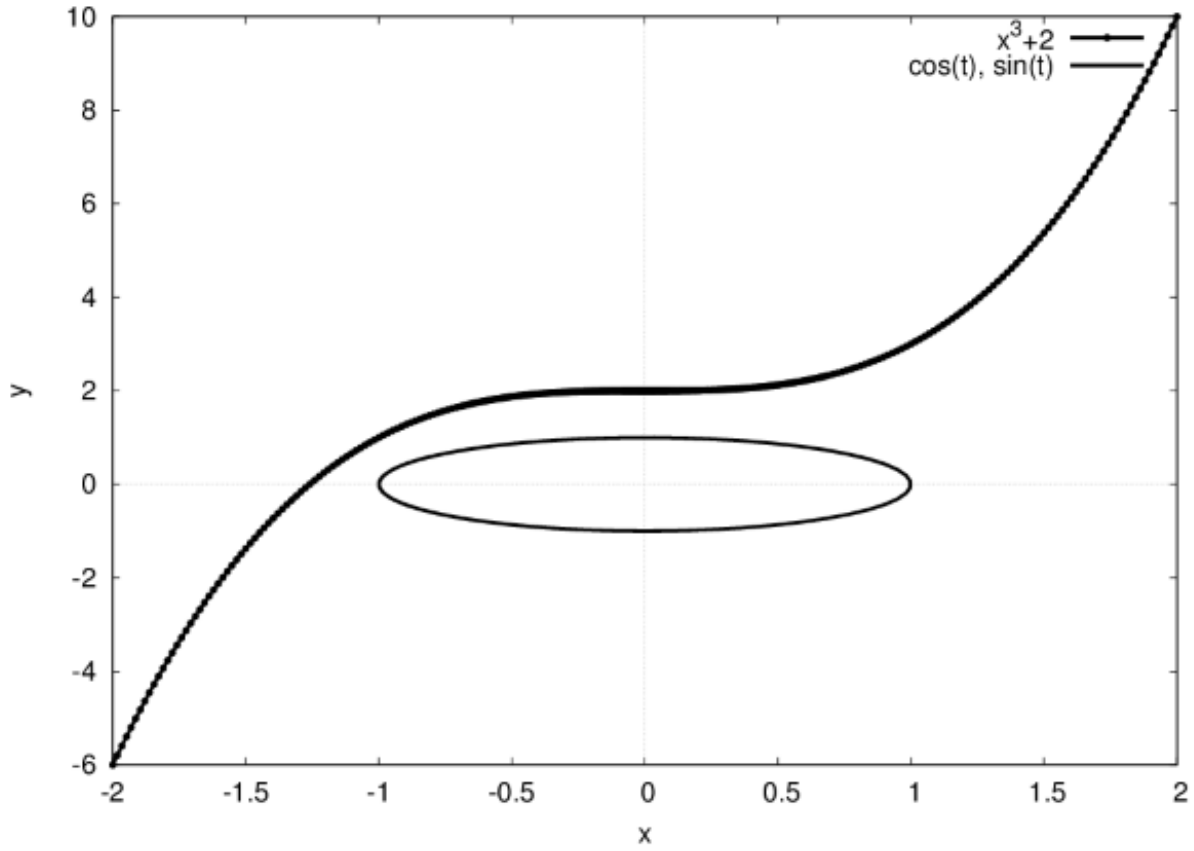
наприклад(див. рис. 2.5):

```
plot2d ([x^3+2, [parametric, cos(t), sin(t), [t, -5, 5],
[nticks,80]]], [x, -2, 2], [xlabel, "x"], [ylabel, "y"],
[style, [linespoints,3,2], [lines,3,1]],
                                [gnuplot_term, s],
                          [gnuplot_out_file, "test.eps"]);
```

Опції [gnuplot_term, ps], [gnuplot_out_file, "test.eps"] вказують, що графічна ілюстрація виводиться у файл $test.eps$ у форматі $postscript$ (Бекенд для виведення графіків - $gnuplot$).

Опції [style,[linespoints,3,2],lines,3,1]] дозволяють вказати стиль ліній на графіку (лінія з точками або суцільна лінія).

Для виведення результатів у формат png можна використовувати опції (вказівка розмірів 400,400 у випадку необов'язково): [gnuplot_term, png size 400,400],[gnuplot_out_file, max.png]



Мал. 2.5.Поєднання на одному графіку параметричної та заданої явно кривих

2.9.3 Побудова кривих у полярній системі координат

Для побудови графіка в полярних координатах потрібно задати зміну значень полярного радіусу та полярного кута. Нехай $r = r(f)(a \le f \le b)$ - Залежність полярного радіусу r від полярного кута f . Тоді графік цієї функції в полярних координатах можна побудувати, поставивши у функції $plot2d$ опцію [gnuplot_preamble, set polar; set zeroaxis]. Ця опція діятиме лише за умови, що вибрано формат графіка $gnuplot$.

Приклад:побудувати у полярних координатах графік функції $r = 3(1 - \varphi + \varphi^2), 0 \le \varphi \le 2\pi$.

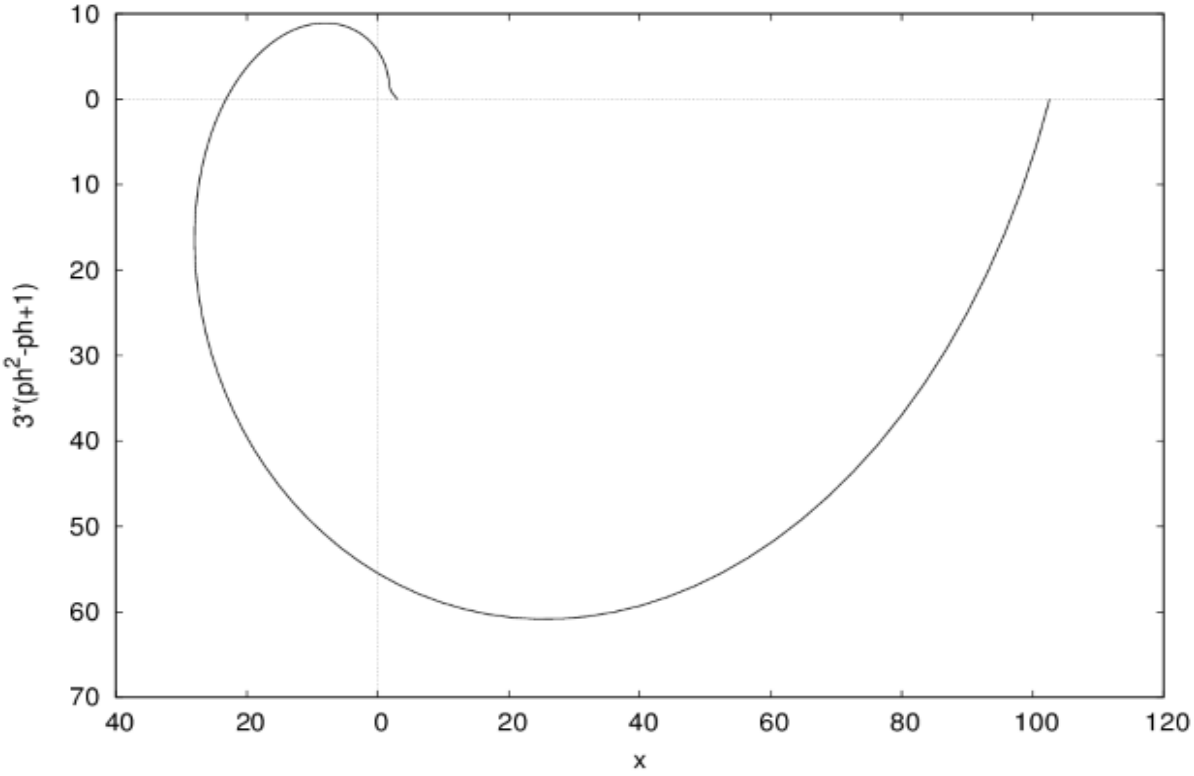
Для створення графіка використовуємо команду:

```
plot2d([3*(1-
ph+ph^2)],[ph,0,2*%pi],[gnuplot_preamble,"set polar",
"set zeroaxis", "set encoding koi8r"], [xlabel, x],
[gnuplot_term, ps],
[gnuplot_out_file, "max.eps"], [plot_format, gnuplot]);
```

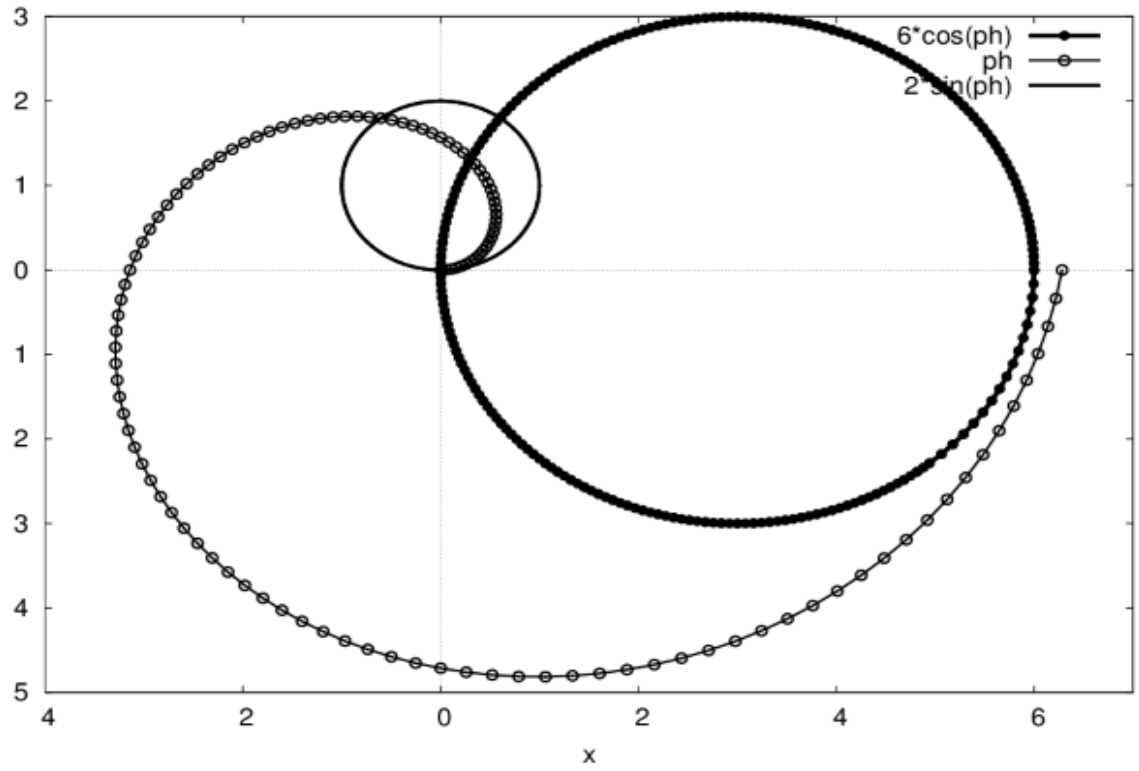
Результат наведено на рис. 2.6. Товщину та стиль лінії можна регулювати, використовуючи опцію $style$ (наприклад, опція [style, [lines,3,1]] встановлює ширину лінії 3 та синій колір).

Приклад:побудувати у полярних координатах графіки трьох функцій $r = 6cos(\varphi), r = \varphi, r = 2sin(\varphi), 0 \le \varphi \le 2\pi$.

Для створення графіка використовуємо команду:



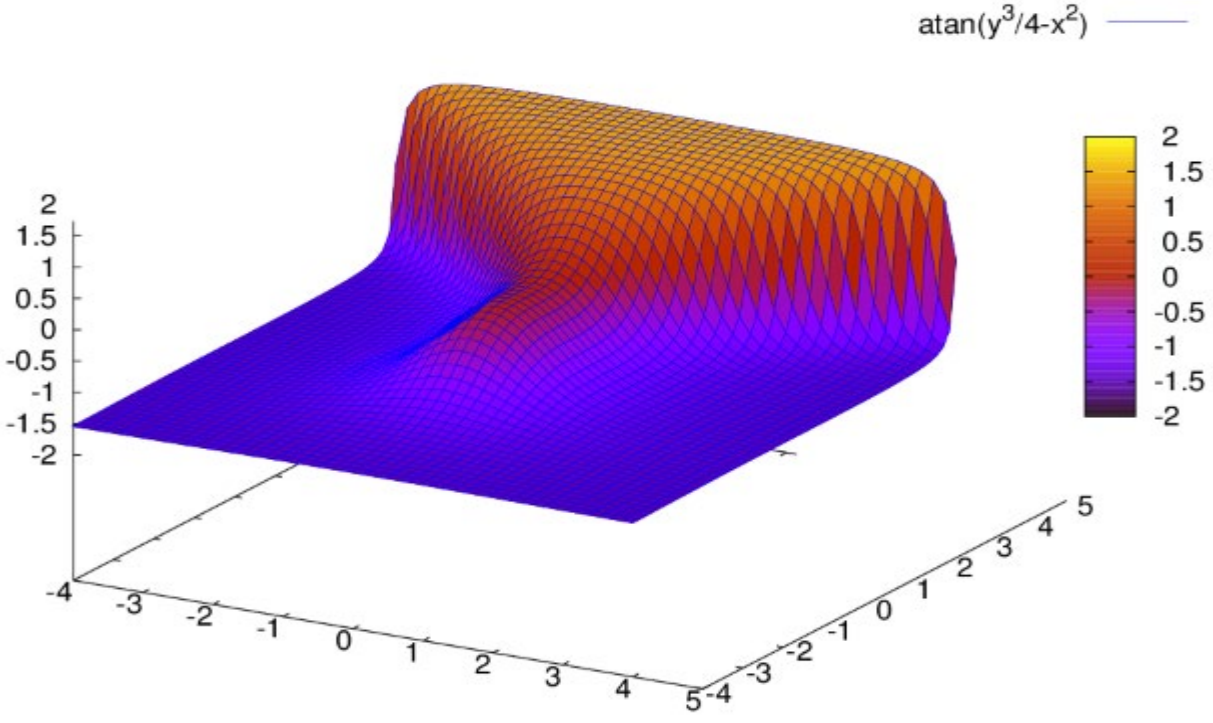
Мал. 2.6.Крива у полярних координатах



Мал. 2.7.Поєднання на одному графіку кількох параметричних кривих

```
plot2d([6*cos(ph),ph,2*sin(ph)],[ph,0,2*%pi],[gnuplot_pre
amble,
"set polar", "set zeroaxis", "set encoding koi8r"],
[xlabel, x],
[gnuplot_term,ps],[gnuplot_out_file, "max3.eps"],
[plot_format, gnuplot]);
```

Результат наведено на рис. 2.7.



Мал. 2.8.Графік функції двох змінних із забарвленням поверхні

2.9.4 Побудова тривимірних графіків

Основна команда для побудови тривимірних графіків $plot3d$. Розглянемо технологію побудови графіків із використанням інтерфейсу $gnuplot$. Поверхня функції у кольоровому зображенні будується за допомогою опції $pm3d$ (Рис. 2.8).

Приклад:

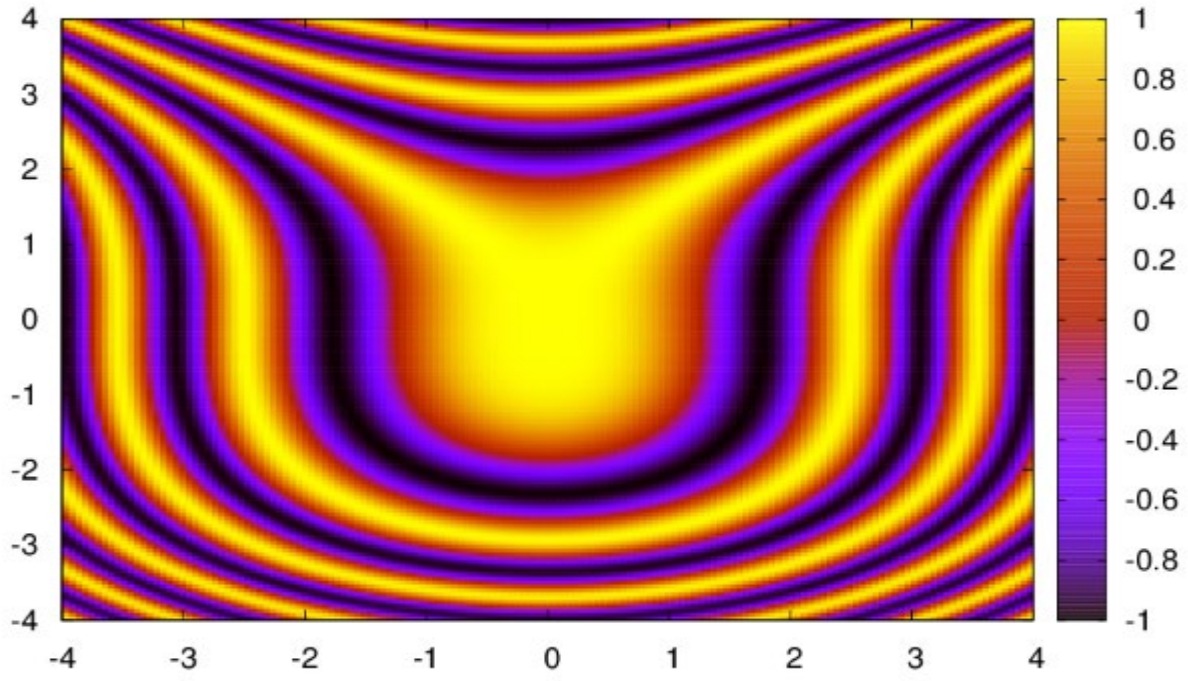
```
(%i2) plot3d (atan (-x^2 + y^3/4), [x, -4, 4], [y, -4,
4],
      [grid, 50, 50],
[gnuplot_pm3d,true],[gnuplot_term,ps],
      [gnuplot_out_file,"plot31.eps"]);
```

З використанням цієї опції та особливостей програми $gnuplot$ можна побудувати зображення ліній рівня функції. Приклад (рис. 2.9):

```
(%i3) plot3d (cos (-x^2 + y^3/4), [x, -4, 4], [y, -4, 4],
      [gnuplot_preamble,"set view map"],
      [gnuplot_pm3d, true], [grid, 150, 150],
[gnuplot_term,ps],
      [gnuplot_out_file,"plot32.eps"]);
```

Більш строгий результат можна отримати, використовуючи стандартний формат функції $plot3d$. Приклад (рис. 2.10):

```
(% i4)     plot3d (2^(-u^2 + v^2), [u, -3, 3], [v, -2,
2]);
```



Мал. 2.9.Графік ліній рівня функції двох змінних із забарвленням поверхні

Для виведення графіка у файл все одно необхідно використовувати опції $gnuplot$ (Встановити термінал $gnuplot$ та ім'я файлу результату). Необхідна команда:

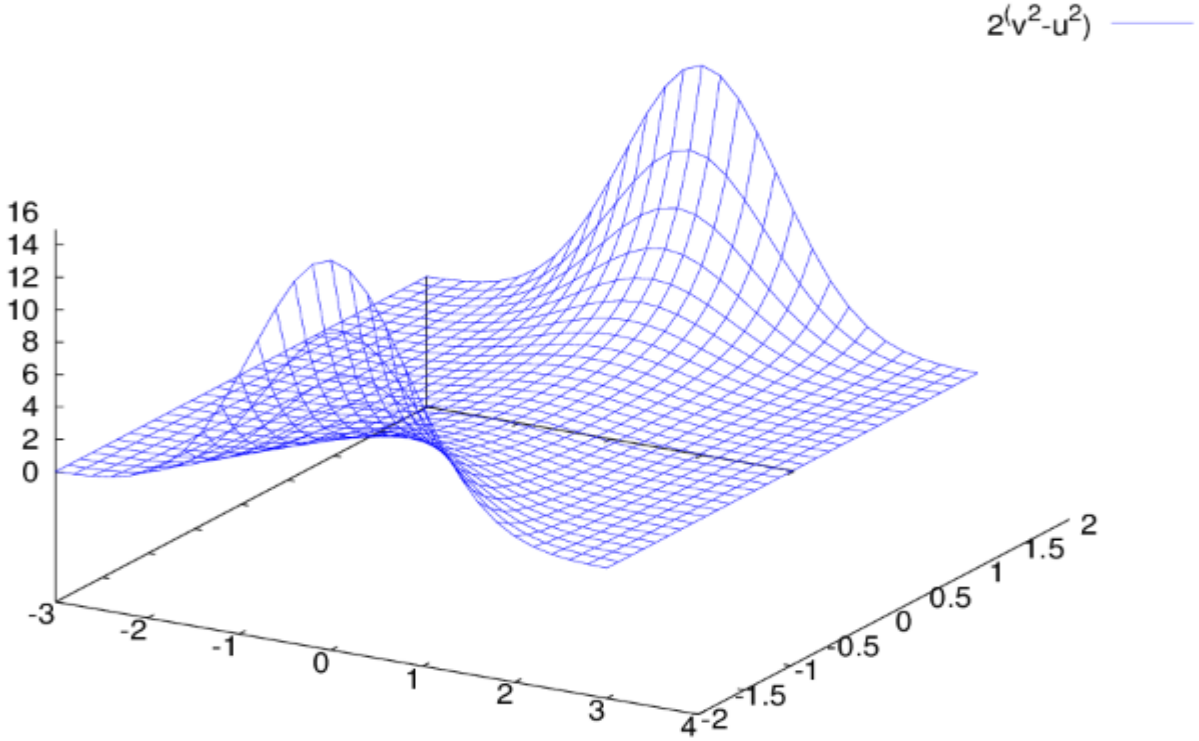
```
(% i5)     plot3d (2^(-u^2 + v^2), [u, -3, 3], [v, -2,
2],
     [gnuplot_term,ps],[gnuplot_out_file,"plot33.eps"]);
```

Зміна формату графіки також можлива за рахунок використання опцій $plot3d$. Приклад (виведення графіки у форматі $openmath$ - Мал. 2.11):

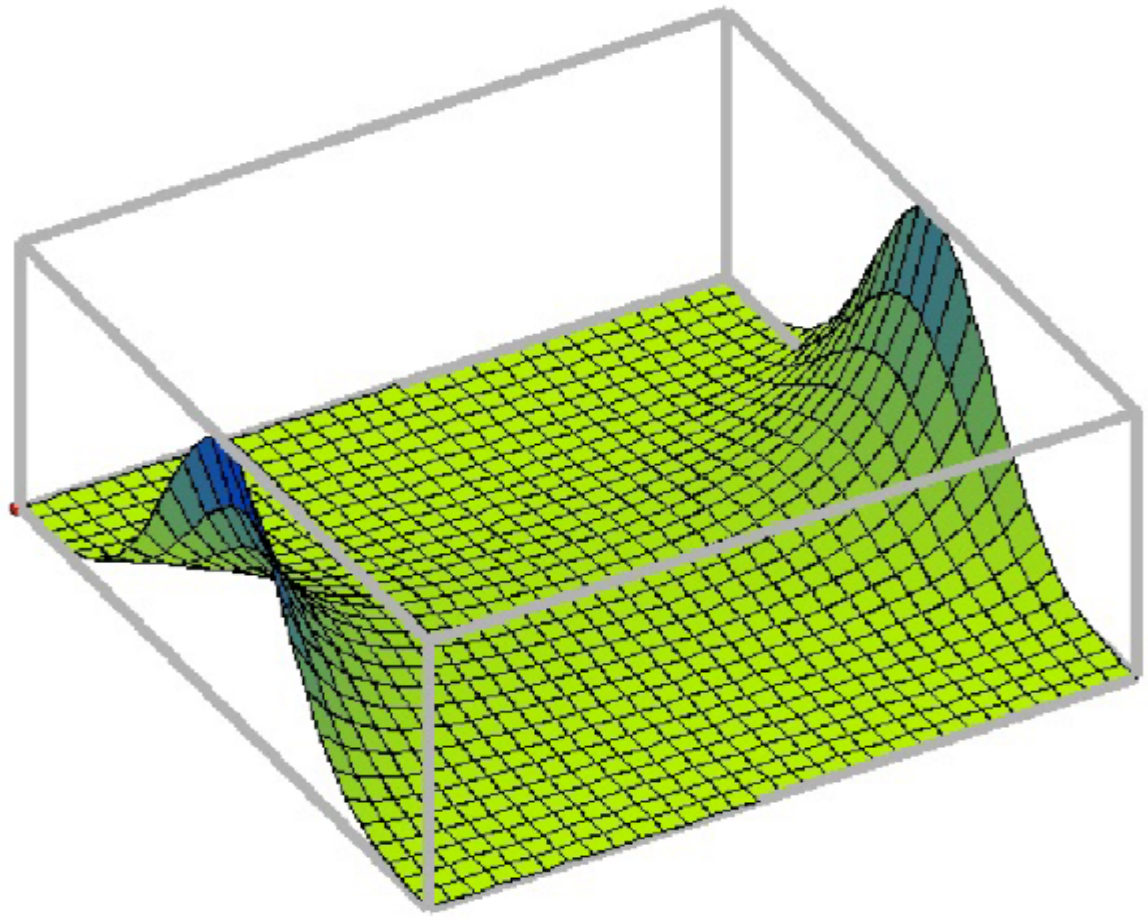
```
(%i6) plot3d (2^(-u^2 + v^2), [u, -3, 3], [v, -2, 2],
     [plot_format, openmath]);
```

Перевагою цього формату є вбудована можливість збереження копії графічного зображення у файл, редагування та повороту побудованого графіка.

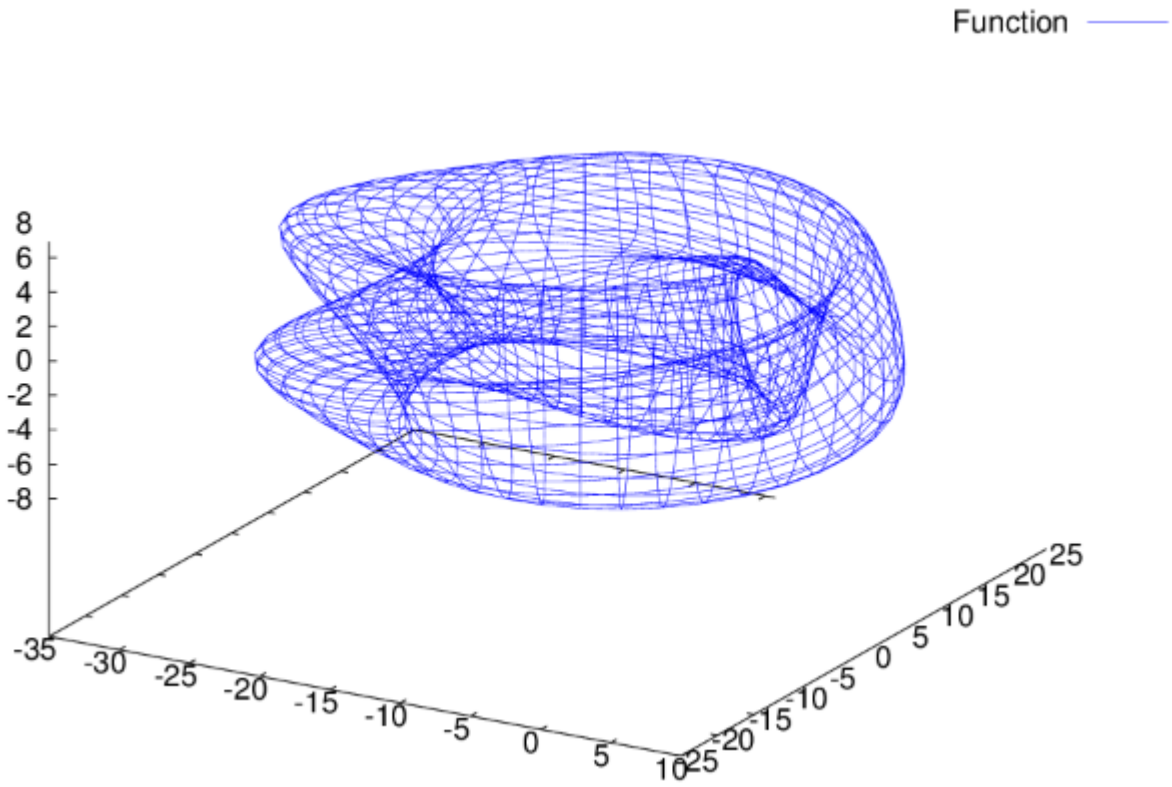
Функція, для якої будується тривимірний графік, може задаватися як Maxima або Lisp-функція, лямбда-функція або вираз Maxima загального вигляду. При використанні формату $plot3d(f, ...)$ вираз f сприймається як функція двох змінних. При використанні формату $plot3d([f_1, f_2, f_3], ...)$, кожна функція (f_1, f_2, f_3) сприймається як функція трьох змінних.



Мал. 2.10.Простий графік функції двох змінних



Мал. 2.11.Простий графік функції двох змінних (формат OpenMath)



Мал. 2.12.Графік функції, визначеної у форматі [f1, f2, f3]

Приклад використання формату $plot3d([f_1, f_2, f_3], ...)$ (рис. 2.12):

Функція $plot3d$ дозволяє будувати графіки функцій, заданих у циліндричних чи сферичних координатах рахунок використання перетворення координат (опція [transform_xy, polar_to_xy] чи функція make_transform (vars, fx, fy, fz)).

Певні переваги забезпечує формат $wxplot$, наявний у графічному інтерфейсі wxMaxima ($wxplot2d$ і $wxplot3d$). Команда побудови графіка у форматі wxMaxima за синтаксисом мало відрізняється від синтаксису команд $plot2d$ і $plot3d$. Якість відтворення графіків на екрані wxMaxima відносно невисока, але легко, виділивши графік клацанням миші, зберегти його у файл (за замовчуванням) $maxout.png$). Якість копії у файлі набагато краща, ніж малюнку у вікні wxMaxima.