Тема2. Основы Maxima

2.1 Структура Maxima

Пакет **Maxima** состоит из интерпретатора макроязыка, написанного на **Lisp**, и нескольких поколений пакетов расширений, написанных на макроязыке пакета или непосредственно на **Lisp.** Maxima позволяет решать достаточно широкий круг задач, относящихся к различным разделам математики.

2.1.1 Области математики, поддерживаемые в Maxima

- Операции с полиномами (манипуляция рациональными и степенными выражениями, вычисление корней и т.п.)
- Вычисления с элементарными функциями, в том числе с логарифмами, экспоненциальными функциями, тригонометрическими функциями
- Вычисления со специальными функциями, в т.ч. эллиптическими функциями и интегралами
- Вычисление пределов и производных
- Аналитическое вычисление определённых и неопределённых интегралов
- Решение интегральных уравнений
- Решение алгебраических уравнений и их систем
- Операции со степенными рядами и рядами Фурье
- Операции с матрицами и списками, большая библиотека функций для решения задач линейной алгебры
- Операции с тензорами
- Теория чисел, теория групп, абстрактная алгебра

Перечень дополнительных пакетов для **Maxima**, которые необходимо загружать перед использованием, существенно расширяющих её возможности и круг решаемых задач, приведён в приложении 1.

2.2 Достоинства программы

Основными преимуществами программы **Maxima** являются:

- возможность свободного использования (**Maxima** относится к классу свободных программ и распространяется на основе лицензии GNU);
- возможность функционирования под управлением различных ОС (в частности **Linux** и **Windows™**);
- размер программы (дистрибутив занимает порядка 23 мегабайт, в установленном виде со всеми расширениями потребуется около 80 мегабайт);
- широкий класс решаемых задач;
- возможность работы как в консольной версии программы, так и с использованием одного из графических интерфейсов (**xMaxima, wxMaxima** или как плагин (plug-in) к редактору **TexMacs**);
- расширение **wxMaxima** (входящее в комплект поставки) предоставляет пользователю удобный и понятный интерфейс, избавляет от необходимости изучать особенности ввода команд для решения типовых задач;
- интерфейс программы на русском языке;
- наличие справки и инструкций по работе с программой (русскоязычной версии справки нет, но в сети Интернет присутствует большое количество статей с примерами использования **Maxima**);

2.3 Установка и запуск программы

Скачать последнюю версию программы можно с её сайта в сети Интернет: http://maxima.sourceforge.net/. Русская локализация сайта: http://maxima.sourceforge.net/ru/.

Система компьютерной алгебры **Maxima** присутствует в большинстве дистрибутивов, однако зачастую в списке дополнительных программ, которые можно скачать в Интернете в версии для данного дистрибутива. Примеры и расчёты в данной книге выполнены с использованием дистрибутива **Alt Linux 4.1 Desktop**[1]Некоторые примеры проверялись в более поздней версии Maxima 5.26.0.

2.4 Интерфейс wxMaxima

Для удобства работы сразу обратимся к графическому интерфейсу **wxMaxima**, т. к. он является наиболее дружественным для начинающих пользователей системы.

Достоинствами **wxMaxima** являются:

- возможность графического вывода формул (см. иллюстрации ниже)
- упрощённый ввод наиболее часто используемых функций (через диалоговые окна), а не набор команд, как в классической **Maxima.**
- разделение окна ввода данных и области вывода результатов (в классической **Maxima** эти области объединены, и ввод команд происходит в единой рабочей области с полученными результатами).

Рассмотрим рабочее окно программы. Сверху вниз располагаются: текстовое меню программы — доступ к основным функциям и настройкам программы. В текстовом меню **wxMaxima** находятся функции для решения большого количества типовых математических задач, разделённые по группам: уравнения, алгебра, анализ, упростить, графики, численные вычисления. Ввод команд через диалоговые окна упрощает работу с программой для новичков.

При использовании интерфейса **wxMaxima**, Вы можете выделить в окне вывода результатов необходимую формулу и вызвав контекстное меню правой кнопкой мыши скопировать любую формулу в текстовом виде, в формате TEX или в виде графического изображения, для последующей вставки в какой-либо документ.

Также в контекстном меню, при выборе результата вычисления, Вам будет предложен ряд операций с выбранным выражением (например, упрощение, раскрытие скобок, интегрирование, дифференцирование и др.).

2.5 Ввод простейших команд Maxima

Все команды вводятся в поле ВВОД, разделителем команд является символ ; (точка с запятой). После ввода команды необходимо нажать клавишу Enter[2]В **wxMaxima** нужно нажать Shift+Enter. для её обработки и вывода результата. В ранних версиях **Maxima** и некоторых её оболочках (например, **xMaxima**) наличие точки с запятой после каждой команды строго обязательно. Завершение ввода символом $ (вместо точки с запятой) позволяет вычислить результат введённой команды, но не выводить его на экран. В случае, когда выражение надо отобразить, а не вычислить, перед ним необходимо поставить знак ' (одинарная кавычка). Но этот метод не работает, когда выражение имеет явное значение, например, выражение $sin(\pi)$ заменяется на значение равное нулю.

Две одинарных кавычки последовательно, применённые к выражению во входной строке, приводят к замещению входной строки результатом вычисления вводимого выражения.

Пример:

```
(%i1)   aa:1024;
```

$$(\%o1) \qquad 1024$$

```
(%i2)   bb:19;
```

$$(\%o2) \qquad 19$$

```
(%i3)   sqrt(aa)+bb;
```

$$(\%o3) \qquad 51$$

```
(%i4)   '(sqrt(aa)+bb);
```

(%o4) $bb + \sqrt{aa}$

```
(%i5)   ''%;
```

(%o5) 51

2.5.1 Обозначение команд и результатов вычислений

После ввода, каждой команде присваивается порядковый номер. В рассмотренном примере, введённые команды имеют номера 1–5 и обозначаются соответственно (%i1), (%i2) и т.д.

Результат вычисления также имеет порядковый номер, например $(\%o1), (\%o2)$ и т.д., где **i** — сокращение от англ. input (ввод), а **o** — англ. output (вывод). Этот механизм позволяет избежать в последующих вычислениях повторения полной записи уже выполненных команд, например (%i1)+(%i2) будет означать добавление к выражению первой команды — выражения второй и последующего вычисления результата. Также можно использовать и номера результатов вычислений, например $(\%o1)*(\%o2)$. Для последней выполненной команды в Maxima есть специальное обозначение — $\%$.

Пример:

Вычислить значение производной функции $y(x) = x^2 \cdot e^{-x}x$:

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

(%o1) $2\,x\,e^{-x} - x^2\,e^{-x}$

```
(%i2)   f(x):=''%;
```

(%o2) $f\left(x\right) := 2\,x\,e^{-x} - x^2\,e^{-x}$

Двойная кавычка перед символом предыдущей операции позволяет заместить этот символ значением, т.е. текстовой строкой, полученной в результате дифференцирования.

Другой пример (с очевидным содержанием):

```
(%i3)   x:4;
```

(%o3) 4

```
(%i4)   sqrt(x);
```

(%o4) 2

```
(%i5)   %^2;
```

(%o5) 4

2.6 Числа, операторы и константы

2.6.1 Ввод числовой информации

Правила ввода чисел в **Maxima** точно такие, как и для многих других подобных программ. Целая и дробная часть десятичных дробей разделяются символом точка. Перед отрицательными числами ставится знак минус. Числитель и знаменатель обыкновенных дробей разделяется при помощи символа / (прямой слэш). Обратите внимание, что если в результате выполнения операции получается некоторое символьное выражение, а необходимо получить конкретное числовое значение в виде десятичной дроби, то решить эту задачу позволит применение флага $numer$. В частности он позволяет перейти от обыкновенных дробей к десятичным. Преобразование к форме с плавающей точкой осуществляет также функция $float$.

```
(%i1)   3/7+5/3;
```

(%o1) $\frac{44}{21}$

```
(%i2)   3/7+5/3, float;
```

(%o2) 2.095238095238095

```
(%i3)   3/7+5/3, numer;
```

(%o3) 2.095238095238095

```
(%i4)   float(5/7);
```

(%o4) 0.71428571428571

2.6.2 Арифметические операции

Обозначение арифметических операций в **Maxima** ничем не отличается от классического представления: + , — , * , /. Возведение в степень можно обозначать несколькими способами: ^, ^^, **. Извлечение корня степени n записываем, как степень $\frac{1}{n}$. Операция нахождение факториала обозначается восклицательным знаком, например 5!. Для увеличения приоритета операции, как и в математике, используются круглые скобки: (). Список основных арифметических и логических операторов приведён в табл. 2.1 и табл. 2.2 ниже.

Таблица 2.1. Арифметические операторы

+	оператор сложения
-	оператор вычитания или изменения знака
*	оператор умножения
/	оператор деления
^ или **	оператор возведения в степень

Таблица 2.2. Логические операторы

<	оператор сравнения меньше
>	оператор сравнения больше
<=	оператор сравнения меньше или равно
>=	оператор сравнения больше или равно
#	оператор сравнения не равно
=	оператор сравнения равно
and	логический оператор и
or	логический оператор или

not логический оператор не

2.6.3 Константы

В **Maxima** для удобства вычислений имеется ряд встроенных констант. Самые распространённые из них показаны в табл. 2.3:

Таблица 2.3. Основные константы Maxima

Название	Обозначение
слева (в отношении пределов)	$minus$
справа (в отношении пределов)	$plus$
плюс бесконечность	inf
минус бесконечность	$minf$
число π	$\%pi$
e (экспонента)	$\%e$
Мнимая единица $\sqrt{-1}$	$\%i$
Истина	$true$
Ложь	$false$
Золотое сечение $(1+\sqrt{5})/2$	$\%phi$

2.7 Типы данных, переменные и функции

Для хранения результатов промежуточных расчётов применяются переменные. Заметим, что при вводе названий переменных, функций и констант важен регистр букв, так переменные x и X — две разные переменные. Присваивание значения переменной осуществляется с использованием символа : (двоеточие), например x:5. Если необходимо удалить значение переменной (очистить её), то применяется метод $kill$:

$kill(x)$ — удалить значение переменной x ;

$kill(all)$ — удалить значения всех используемых ранее переменных.

Зарезервированные слова, использование которых в качестве имён переменных вызывает синтаксическую ошибку:
$integrate,\ next,\ from, diff,\ in,\ at,\ limit,\ sum,\ for,\ and,\ elseif,\ then,\ else,\ do,\ or,\ if,\ unless,\ product,\ while,\ thru,\ step$
.

2.7.1 Списки

Списки — базовые строительные блоки для **Maxima** и **Lisp.** Все прочие типы данных (массивы, хэш-таблицы, числа) представляются как списки. Чтобы задать список, достаточно записать его элементы через запятую и ограничить запись квадратными скобками. Список может быть пустым или состоять из одного элемента.

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

Элементом списка может и другой список

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

Ссылка на элемент списка производится по номеру элемента списка:

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

2.7.1.1 Функции для элементарных операций со списками

Функция $length$ возвращает число элементов списка (при этом элементы списка сами могут быть достаточно сложными конструкциями):

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

Функция $copylist(expr)$ возвращает копию списка expr:

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

$$(\%o2) \qquad [1,2,3,x,y+x]$$

Функция $makelist$ создаёт список, каждый элемент которого генерируется из некоторого выражения. Возможны два варианта вызова этой функции:

- $makelist(expr,i,i_0,i_1)$ — возвращает список, j -й элемент которого равен $ev(expr,i=j)$, при этом индекс j меняется от i_0 до i_1
- $makelist(expr,x,list)$ — возвращает список, j -й элемент которого равен $ev(expr,x=list[j])$, при этом индекс j меняется от 1 до $length(list)$.

Примеры:

```
(%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);
```

$$(\%o3) \qquad [e,e^2,e^3,e^4,e^5,e^6,e^7]$$

Во многом аналогичные действия выполняет функция

$$create_list(form,x_1,list_1,...,x_n,list_n).$$

Эта функция строит список путём вычисления выражения $form$, зависящего от x_1, к каждому элементу списка $list_1$ (аналогично form, зависящая и от x_2, применяется к $list_2$ и т.д.).

Пример:

```
(%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]);
```

$$(\%o2) \qquad [[a,e],[a,f],[a,h],[b,e],[b,f],[b,h]]$$

Функция append позволяет склеивать списки. При вызове

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

возвращается один список, в котором за элементами $list_1$ следуют элементы $list_2$ и т.д. вплоть до $list_n$.

Пример:

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

$$(\%o1) \qquad [1,2,3,4,5,6,7]$$

Создать новый список, компонуя элементы двух списков поочерёдно в порядке следования, позволяет функция $join(l,m)$. Новый список содержит l_1, затем m_1, затем l_2, m_2 и т.д.

Пример:

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

$$(\%o1) \qquad [1,10,2,20,3,30]$$

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

$$(\%o2) \qquad [1,10,2,20,3,30]$$

Длина полученного списка ограничивается минимальной длиной списков l и m.

Функция $cons(expr,list)$ создаёт новый список, первым элементом которого будет $expr$, а остальные — элементы списка $list$. Функция $endcons(expr,list)$ также создаёт новый список, первые элементы которого — элементы списка $list$, а последний — новый элемент $expr$.

Пример:

```
(%i1)   cons(x,[1,2,3]);
```

$$(\%o1) \qquad [x,1,2,3]$$

```
(%i2)   endcons(x,[1,2,3]);
```

$$(\%o2) \qquad [1,2,3,x]$$

Функция $reverse$ меняет порядок элементов в списке на обратный

```
(%i5)   list1:[1,2,3,x];
```

$$(\%o5) \qquad [1,2,3,x]$$

```
(%i6)   list2:reverse(list1);
```

$$(\%o6) \qquad [x,3,2,1]$$

Функция $member(expr_1,expr_2)$ возвращает $true$, если $expr_1$ является элементом списка $expr_2$, и $false$ в противном случае.

Пример:

```
(%i1)   member (8, [8, 8.0, 8b0]);
```

$$(\%o1) \qquad true$$

```
(%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) $true$

Функция $rest(expr)$ выделяет остаток после удаления первого элемента списка $expr$. Можно удалить первые n элементов, используя вызов $rest(expr, n)$. Функция $last(expr)$ выделяет последний элемент списка $expr$ (аналогично $first$ — первый элемент списка).

Примеры:

```
(%i1)   list1:[1,2,3,4,a,b];
```

(%o1) $[1, 2, 3, 4, a, b]$

```
(%i2)   rest(list1);
```

(%o2) $[2, 3, 4, a, b]$

```
(%i3)   rest(%);
```

(%o3) $[3, 4, a, b]$

```
(%i4)   last(list1);
```

(%o4) b

```
(%i5)   rest(list1,3);
```

(%o5) $[4, a, b]$

Суммирование и перемножение списков (как и прочих выражений) осуществляется функциями sum и $product$. Функция $sum(expr, i, in, ik)$ суммирует значения выражения expr при изменении индекса i от in до ik. Функция $product(expr, i, in, ik)$перемножает значения выражения $expr$ при изменении индекса i от in до ik.

Пример:

```
(%i1)   product (x + i*(i+1)/2, i, 1, 4);
```

(%o1) $(x + 1)\ (x + 3)\ (x + 6)\ (x + 10)$

```
(%i2)   sum (x + i*(i+1)/2, i, 1, 4);
```

(%o2) $4\,x + 20$

```
(%i3)   product (i^2, i, 1, 4);
```

(%o3) 576

```
(%i4)   sum (i^2, i, 1, 4);
```

(%o4) 30

2.7.1.2 Функции, оперирующие с элементами списков

Функция $map(f, expr_1, ..., expr_n)$ позволяет применить функцию (оператор, символ операции) f к частям выражений $expr_1, expr_2, ..., expr_n$. При использовании со списками применяет f к каждому элементу списка. Следует обратить внимание, что f — именно имя функции (без указания переменных, от которых она зависит).

Примеры:

```
(%i1)   map(ratsimp, x/(x^2+x)+(y^2+y)/y);
```

(%o1) $$y + \frac{1}{x + 1} + 1$$

```
(%i2)   map("=",[a,b],[-0.5,3]);
```

(%o2) $[a = -0.5, b = 3]$

```
(%i3)   map(exp,[0,1,2,3,4,5]);
```

(%o3) $[1, e, e^2, e^3, e^4, e^5]$

Функция f может быть и заданной пользователем, например:

```
(%i5) f(x):=x^2;
```

(%o5) $f\,(x) := x^2$

```
(%i6)   map(f,[1,2,3,4,5]);
```

(%o6) $[1, 4, 9, 16, 25]$

Функция $apply$ применяет заданную функцию ко всему списку (список становится списком аргументов функции; при вызове ($F, [x_1, ..., x_n]$ вычисляется выражение $F(arg_1, ..., arg_n)$). Следует учитывать, что $apply$ не распознаёт ординарные функции и функции от массива.

Пример:

```
(%i1)   L : [1, 5, -10.2, 4, 3];
```

(%o1) $[1, 5, -10.2, 4, 3]$

```
(%i2)   apply(max,L);
```

(%o2) 5

```
(%i3)  apply(min,L);
```

(%o3) -10.2

Чтобы найти максимальный или минимальный элемент набора чисел, надо вызвать функции max или min. Однако, обе функции в качестве аргумента ожидают несколько чисел, а не список, составленный из чисел. Применять подобные функции к спискам и позволяет функция $apply$.

2.7.2 Массивы

Массивы в **Maxima** — совокупности однотипных объектов с индексами. Число индексов не должно превышать пяти. В **Maxima** существуют и функции с индексами (функции массива).

Возможно создание и использование переменных с индексами до объявления соответствующего массива. Такие переменные рассматриваются как элементы массивов с неопределёнными размерностями (так называемые хэш-массивы). Размеры неопределённых массивов растут динамически по мере присваивания значений элементам. Интересно, что индексы массивов с неопределёнными границами не обязательно должны быть числами. Для повышения эффективности вычислений рекомендуется преобразовывать массивы с неопределёнными границами в обычные массивы (для этого используется функция $array$).

Создание массива производится функцией array. Синтаксис обращения к функции: $array(name, dim_1, ..., dim_n)$ — создание массива с именем name и размерностями $dim_1, ..., dim_n$; **array(name,type,dim_1,... ,dim_n)** — создание массива с именем name и элементами типа $type$; $array([name_1, ..., name_m], dim_1, ..., dim_n)$ — создание нескольких массивов одинаковой размерности.

Индексы обычного массива — целые числа, изменяющиеся от 0 до dim_i.

Пример:

```
(%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) $[0, 1, 2, 3]$

Функция $listarray$, использованная в примере, преобразует массив в список. Синтаксис вызова: $listarray(A)$.

Аргумент A может быть определённым или неопределённым массивом, функцией массива или функцией с индексами. Порядок включения элементов массива в список — по строкам.

Функция $arrayinfo$ выводит информацию о массиве A. Синтаксис вызова: $arrayinfo(A)$ Аргумент A, как и в случае $listarray$, может быть определённым или неопределённым массивом, функцией массива или функцией с индексами.

Пример использования:

```
(%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) $[(c - d)^3, (b + a)^2]$

Функции $listarray$ и $arrayinfo$ применимы и к функциям массива:

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

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

```
(%i10)  cc[1,2];
```

(%o10) 2

```
(%i11)  cc[2,1];
```

(%o11) $\frac{1}{2}$

```
(%i12)  arrayinfo(cc);
```

(%o12) $[hashed, 2, [1, 2], [2, 1]]$

```
(%i13)  listarray(cc);
```

$$(\%o13) \qquad [2, \frac{1}{2}]$$

Ещё один пример — создание и вывод информации о функциях с индексами:

```
(%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);
```

$$(\%o5) \qquad [lambda\,([y], y)\,, lambda\left([y], y^{b+a}\right)]$$

Функция $make_array(type, dim_1, ..., dim_n)$ создаёт и возвращает массив **Lisp.** Тип массива может быть $any,\ flonum,\ fixnum,\ hashed,\ functional$. Индекс i может изменяться в пределах от 0 до $dim_i - 1$.

Достоинство $make_array$ по сравнению с $array$ — возможность динамически управлять распределением памяти для массивов. Присваивание $y : make_array(...)$ создаёт ссылку на массив. Когда массив больше не нужен, ссылка уничтожается присваиванием $y : false$, память освобождается затем сборщиком мусора.

Примеры:

```
(%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);
```

$$(\%o4) \qquad [declared, 1, [7]]$$

Переменная $arrays$ содержит список имён массивов первого и второго видов, определённых на данный момент.

Пример:

```
(%i1)   array(a,1,1);
```

$$(\%o1) \qquad a$$

```
(%i2)   array(b,2,3);
```

$$(\%o2) \qquad b$$

```
(%i3)   arrays;
```

$$(\%o3) \qquad [a, b]$$

Функция $fillarray$ позволяет заполнять массивы значениями из другого массива или списка. Заполнения производится по строкам.

Примеры:

```
(%i1)   array(a,1,1);
```

$$(\%o1) \qquad a$$

```
(%i2)   fillarray(a,[1,2,3,4]);
```

$$(\%o2) \qquad a$$

```
(%i3)   a[1,1];
```

$$(\%o3) \qquad 4$$

```
(%i4)   a2 : make_array (fixnum, 8);
```

$$(\%o4) \qquad Lisp\ Array\ \#(0\ 0\ 0\ 0\ 0\ 0\ 0\ 0)$$

```
(%i5)   fillarray (a2, [1, 2, 3, 4, 5]);
```

$$(\%o5) \qquad Lisp\ Array\ \#(1\ 2\ 3\ 4\ 5\ 5\ 5\ 5)$$

Как видно из рассмотренных примеров, длина списка может и не совпадать с размерностью массива. Если указан тип массива, он должен заполняться элементами того же типа. Удаление массивов из памяти осуществляется функцией $remarray$.

Кроме того, для изменения размерности массива имеется функция $rarray(A, dim_1, ..., dim_n)$. Новый массив заполняется элементами старого по строкам. Если размер старого массива меньше, чем нового, остаток нового заполняется нулями или $false$ (в зависимости от типа массива).

2.7.3 Матрицы и простейшие операции с ними

В **Maxima** определены прямоугольные матрицы.

Основной способ создания матриц — использования функции matrix. Синтаксис вызова: $matrix(row_1, ..., row_n)$. Каждая строка — список выражений, все строки одинаковой длины. На множестве матриц определены операции сложения, вычитания, умножения и деления. Эти операции выполняются поэлементно, если операнды — две матрицы, скаляр и матрица или матрица и скаляр. Возведение в степень возможно, если один из операндов — скаляр. Перемножение матриц (в общем случае некоммутативная операция) обозначается символом ". ". Операция умножения матрицы самой на себя может рассматриваться как возведение в степень. Возведение в степень -1 — как обращение (если это возможно).

Пример создания двух матриц:

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

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

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

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

Выполнение арифметических операций с матрицами:

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

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

```
(%i4)   x-y;
```

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

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

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

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

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

Обратите внимание — операции выполняются поэлементно. При попытке выполнять арифметические операции, как представлено выше, над матрицами различных размеров, выдаётся ошибка.

Пример операций с матрицами и скалярами:

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

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

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

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

Умножение матрицы на матрицу:

```
(%i11)  x.y;
```

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

```
(%i12)  y.x;
```

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

Очевидно, что для успешного перемножения матрицы должны быть согласованы по размерам. Возведение в степень -1 даёт обратную матрицу:

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

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

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

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

Стоит обратить внимание, что операции x^^-1 и x^-1 дают разный результат!

Пример:

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

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

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

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

Функция $genmatrix$ возвращает матрицу заданной размерности, составленную из элементов двухиндексного массива. Синтаксис вызова:

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

Индексы i_1, j_1 и i_2, j_2 указывают левый и правый нижний элементы матрицы в исходном массиве.

Пример:

```
(%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);
```

$$(\%o6) \begin{bmatrix} e & 0 \\ 0 & \pi \end{bmatrix}$$

Функция $zeromatrix$ возвращает матрицу заданной размерности, составленную из нулей (синтаксис вызова $zeromatrix(m, n)$).

```
(%i7)   zeromatrix(2,2);
```

$$(\%o7) \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Функция $ident$ возвращает единичную матрицу заданной размерности (синтаксис $ident(n)$)

```
(%i9)   ident(2);
```

$$(\%o9) \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Функция $copymatrix(M)$ создаёт копию матрицы M. Обратите внимание, что присваивание не создаёт копии матрицы (как и присваивание не создаёт копии списка).

Пример:

```
(%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;
```

$$(\%o4) \begin{bmatrix} 1 & 2 \\ 3 & 10 \end{bmatrix}$$

Присваивание нового значения элементу матрицы b изменяет и значение соответствующего элемента матрицы a. Использование $copymatrix$ позволяет избежать этого эффекта.

Функции row и col позволят извлечь соответственно строку и столбец заданной матрицы, получая список. Синтаксис вызова:

- $row(M, i)$ — возвращает i -ю строку;
- $col(M, i)$ — возвращает i -й столбец.

Функции $addrow$ и $addcol$ добавляют к матрице строку или столбец соответственно. Синтаксис вызова:

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

Здесь $list_1, ..., list_n$ — добавляемые строки или столбцы.

Пример:

```
(%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]);
```

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

Функция $submatrix$ возвращает новую матрицу, состоящую из подматрицы заданной. Синтаксис вызова:

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

Подматрица строится следующим образом: из матрицы M удаляются строки $i_1, ..., i_m$ и $j_1, ..., j_n$.

Пример (используем последний результат из предыдущего примера, удаляем третью строку и третий столбец):

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

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

Для заполнения матрицы значениями некоторой функции используется функция $matrixmap$ (аналог $map, apply, fullmap$). Синтаксис вызова: matrixmap(f,M). Функция matrixmap возвращает матрицу с элементами i, j, равными $f(M[i, j])$.

Пример:

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

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

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

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

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

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

Для работы с матрицами существует ещё много функций, но они относятся к решению различных задач линейной алгебры, поэтому обсуждаются ниже, в главе 3.2.

2.7.4 Математические функции

В **Maxima** имеется достаточно большой набор встроенных математических функций. Перечень основных классов встроенных функций приведён ниже:

- тригонометрические функции: sin (синус), cos (косинус), tan (тангенс), cot (котангенс);
- обратные тригонометрические функции: $asin$ (арксинус), $acos$ (арккосинус), $atan$ (арктангенс), $acot$ (арккотангенс);
- sec (секанс, $sec(x) = \frac{1}{cos(x)}$, csc (косеканс, $csc(x) = \frac{1}{sin(x)}$);
- $sinh$ (гиперболический синус), $cosh$ (гиперболический косинус), $tanh$ (гиперболический тангенс), $coth$ (гиперболический котангенс), $sech$ (гиперболический секанс), $cosh$ (гиперболический косеканс);
- log (натуральный логарифм);
- $sqrt$ (квадратный корень);
- mod (остаток от деления);
- abs (модуль);
- $min(x_1, ..., x_n)$ и $max(x_1, ..., x_n)$ — нахождение минимального и максимального значения в списке аргументов;
- $sign$ (определяет знак аргумента: pos — положительный, neg — отрицательный, pnz — не определён, $zero$ — значение равно нулю);
- Специальные функции — функции Бесселя, гамма-функция, гипергеометрическая функция и др.;
- Эллиптические функции различных типов.

2.7.5 Вычисление и преобразование аналитических выражений

Функция ev является основной функцией, обрабатывающей выражения. Синтаксис вызова: $ev(expr, arg_1, ..., arg_n)$

Функция ev вычисляет выражение expr в окружении, определяемом аргументами $arg_1, ..., arg_n$. Аргументы могут быть ключами (булевскими флагами, присваиваниями, уравнениями и функциями. Функция ev возвращает результат (другое выражение).

Во многих случаях можно опускать имя функции ev (т.е. применять значения переменных к некоторому выражению)

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

На выражение expr по умолчанию действует функция упрощения. Необходимость выполнения упрощения регулируется флагом $simp$ (если установить $simp = false$, упрощение будет отключено). Кроме того, используют флаги $float$ и $numer$, определяющие формат представления рациональных чисел (в виде дробей или с плавающей точкой) и результатов вычисления математических функций. Флаг $pred$ определяет необходимость вычисления применительно к логическим выражениям.

Аргументами ev могут быть и встроенные функции, выполняющие упрощение или преобразование выражений ($expand, factor, trigexpand, trigreduce$) или функция $diff$.

Если указаны подстановки (в виде $x = val1$ или $x : val2$), то они выполняются.

При этом повторный вызов функции ev вполне способен ещё раз изменить выражение, т.е. обработка выражения не идёт до конца при однократном вызове функции ev.

Пример:

```
(%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)
```

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

Другой пример показывает применение $diff$ к отложенному вычислению производной:

```
(%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);
```

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

Флаг $simp$ разрешает либо запрещает упрощение выражений. Изначально он равен $true$, если установить его равным $false$, то упрощения производиться не будут:

```
(%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;
```

$$(\%o3) \qquad a + 2\,a + 3\,a + 4\,a$$

Функцию ev не обязательно указывать явно, например:

```
(%i3)   x+y, x: a+y, y: 2;
```

$$(\%o3) \qquad y + a + 2$$

Оператор, принудительного вычисления, обозначенный двумя апострофами, является синонимом к функции ev (выражение). Сама функция ev предоставляет гораздо более широкие возможности, нежели простое принудительное вычисление заданного выражения: она может принимать произвольное число аргументов, первый из которых — вычисляемое выражение, а остальные — специальные опции, которые как раз и влияют на то, как именно будет производиться вычисление.

В терминологии **Maxima** невычисленная форма выражения называется "noun form", вычисленная — "verb form". Сохраняя лингвистические параллели, на русский это можно перевести как "несовершенная форма" и "совершённая форма". Значение вводимого выражения в **Maxima** закономерно сохраняется до его вычисления (т. е. в несовершенной форме), а значение выводимого выражения — после (т. е. в совершённой); другими словами, тут имеется естественный порядок "ввод — вычисление — вывод".

Функция $factor$ факторизует (т.е. представляет в виде произведения некоторых сомножителей) заданное выражение (функция $gfactor$ — аналогично, но на множестве комплексных чисел и выражений).

Пример:

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

$$(\%o1) \qquad (x - 1)\,\left(x^2 + x + 1\right)$$

```
(%i2)   factor(x^3-1);
```

$$(\%o2) \qquad (x - 1)\,\left(x^2 + x + 1\right)$$

Ещё примеры факторизации различных выражений:

```
(%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));
```

$$(\%o5) \qquad (e^x + 1)\,\left(e^{2\,x} - e^x + 1\right)$$

Пример использования функции $gfactor$:

```
(%i6)   gfactor(x^2+a^2);
```

$$(\%o6) \qquad (x - i\,a)\,(x + i\,a)$$

```
(%i7)   gfactor(x^2+2*%i*x*a-a^2);
```

(%o7) $$(x+i\,a)^2$$

Функция $factorsum$ факторизует отдельные слагаемые в выражении.

```
(%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) $$(x+1)\left(a\,(z+w)^2+(v+u)^2\right)$$

Функция $gfactorsum$ отличается от $factorsum$ тем же, чем $gfactor$ отличается от $factor$:

```
(%i10)  gfactorsum( a^3+3*a^2*b+3*a*b^2+b^3+x^2+2*%i*x*y-y^2 );
```

(%o10) $$(b+a)^3-(y-i\,x)^2$$

Функция $expand$ раскрывает скобки, выполняет умножение, возведение в степень, например:

```
(%i1)   expand((x-a)^3);
```

(%o1) $$x^3-3\,a\,x^2+3\,a^2\,x-a^3$$

```
(%i2)   expand((x-a)*(y-b)*(z-c));
```

(%o2) $$x\,y\,z-a\,y\,z-b\,x\,z+a\,b\,z-c\,x\,y+a\,c\,y+b\,c\,x-a\,b\,c$$

```
(%i3)   expand((x-a)*(y-b)^2);
```

(%o3) $$x\,y^2-a\,y^2-2\,b\,x\,y+2\,a\,b\,y+b^2\,x-a\,b^2$$

Функция $combine$ объединяет слагаемые с идентичным знаменателем

```
(%i5)   combine(x/(1+x^2)+y/(1+x^2));
```

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

Функция $xthru$ приводит выражение к общему знаменателю, не раскрывая скобок и не пытаясь факторизовать слагаемые

```
(%i6)   xthru( 1/(x+y)^10+1/(x+y)^12 );
```

(%o6) $$\frac{(y+x)^2+1}{(y+x)^{12}}$$

```
(%i1)   ((x+2)^20 - 2*y)/(x+y)^20 + (x+y)^(-19) - x/(x+y)^20;
```

(%o1) $$\frac{1}{(y+x)^{19}}+\frac{(x+2)^{20}-2\,y}{(y+x)^{20}}-\frac{x}{(y+x)^{20}}$$

```
(%i2)   xthru (%);
```

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

Функция $multthru$ умножает каждое слагаемое в сумме на множитель, причём при умножении скобки в выражении не раскрываются. Она допускает два варианта синтаксиса:

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

В последнем случае выражение expr включает и множитель и сумму (см. (%i4) в примере ниже).

Пример:

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

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

```
(%i2)   multthru ((x-y)^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) $$\frac{2}{s}+\frac{a\,b}{s^2}+\frac{(b+a)^{10}}{a\,b}$$

Функции assume (ввод ограничений) и $forget$ (снятие ограничений) позволяют управлять условиями выполнения (контекстом) прочих функций и операторов.

Пример:

```
(%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|
```

Функция $divide$ позволяет вычислить частное и остаток от деления одного многочлена на другой:

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

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

Первый элемент полученного списка — частное, второй — остаток от деления.

Функция gcd позволяет найти наибольший общий делитель многочленов.

Подстановки осуществляются функцией $subst$. Вызов этой функции: $subst(a,b,c)$ (подставляем a вместо b в выражении c).

Пример:

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

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

2.7.6 Преобразование рациональных выражений

Для выделения числителя и знаменателя дробных выражений используются функции num и $denom$:

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

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

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

$$(\%o2) \qquad x^2+1$$

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

$$(\%o3) \qquad x^3-1$$

Функция rat приводит выражение к каноническому представлению. Она упрощает любое выражение, рассматривая его как дробнорациональную функцию, т.е. работает с операциями "+", "-", "*", "/" и с возведением в целую степень.

Синтаксис вызова:

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

Переменные упорядочиваются в соответствии со списком $x_1, ..., x_n$. При этом вид ответа зависит от способа упорядочивания переменных Изначально переменные упорядочены в алфавитном порядке.

Пример использования rat:

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

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

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

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

После указания порядка использования переменных получаем следующее выражение:

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

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

Функция $ratvars$ позволяет изменить алфавитный порядок предпочтения переменных, принятый по умолчанию. Вызов $ratvars(z,y,x,w,v,u,t,s,r,q,p,o,n,m,l,k,j,i,h,g,f,e,d,c,b,a)$ меняет порядок предпочтения в точности на обратный, а вызов $ratvars(m,n,a,b)$ упорядочивает переменные m,n,a,b в порядке возрастания приоритета.

Флаг $ratfac$ включает или выключает частичную факторизацию выражений при сведении их к стандартной форме (CRE). Изначально установлено значение $false$. Если установить значение $true$, то будет производиться частичная факторизация.

Функция $ratsimp$ приводит все части (в том числе аргументы функций) выражения, которое не является дробно-рациональной функцией, к каноническому представлению, производя упрощения, которые не выполняет функция rat. Повторный вызов функции в общем случае может изменить результат, т.е. не обязательно упрощение проводится до конца. Применением упрощения к экспоненциальным выражениям управляет флаг $ratsimexpons$, по умолчанию равный $false$, если его установить в $true$, упрощение применяется и к показателям степени или экспоненты.

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

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

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

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

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

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

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

Функция $fullratsimp$ вызывает функцию $ratsimp$ до тех пор, пока выражение не перестанет меняться.

Пример:

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

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

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

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

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

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

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

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

Пример влияния флага $ratsimpexponds$ на результат вычислений:

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

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

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

(%o2) $$true$$

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

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

Функция $ratexpand$ раскрывает скобки в выражении. Отличается от функции $expand$ тем, что приводит выражение к канонической форме, поэтому ответ может отличаться от результата применения функции $expand$:

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

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

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

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

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

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

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

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

Подстановка в рациональных выражениях осуществляется функцией $ratsubst$. Синтаксис вызова: $ratsubst(a,b,c)$ Выражение a подставляется вместо выражения b в выражении c (b может быть суммой, произведением, степенью и т.п.).

Пример использования $ratsubst$:

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

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

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

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

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

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

2.7.7 Преобразование тригонометрических выражений

Функция $trigexpand$ раскладывает все тригонометрические и гиперболические функции от сумм и произведений в комбинации соответствующих функций единичных углов и аргументов. Для усиления пользовательского контроля один вызов $trigexpand$ выполняет упрощение на одном уровне. Для управления вычислением имеется флаг $trigexpand$. Изначально флаг $trigexpand$ установлен в $false$. Если флаг $trigexpand$ установить в $true$, то функция $trigexpand$ будет работать до тех пор, пока выражение не перестанет меняться.

```
(%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));
```

$$(\%o3)$$
$$sin(x)^4-sin(x)^3-6\,cos(x)^2\,sin(x)^2+3\,cos(x)^2\,sin(x)+cos(x)^4$$

Функция trigreduce свёртывает все произведения тригонометрических и гиперболических функций в комбинации соответствующих функции от сумм. Функция работает не до конца, так что повторный вызов может изменить выражение. При вызове функции в формате trigreduce(expr,x) преобразования осуществляются относительно функций x.

Примеры:

```
(%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);
```

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

Функция $trigsimp$ упрощает тригонометрические и и гиперболические выражения, применяя к ним правила $sin(x)^2+cos(x)^2=1$ и $cosh(x)^2-sinh(x)^2=1$.

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);
```

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

Функция $trigrat$ (синтаксис вызова $trigrat(expr)$) приводит заданное тригонометрическое выражение $expr$ к канонической упрощённой квазилинейной форме. Это выражение рассматривается как рациональное, содержащее sin, cos, tan, аргументы которых линейные формы некоторых переменных и $\frac{\pi}{n}$ (n — целое). Всегда, когда возможно, заданное выражение линеаризуется.

Пример:

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

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

2.7.8 Преобразование степенных и логарифмических выражений

Функция **radcan** упрощает выражения, содержащие экспоненты, логарифмы и радикалы, путём преобразования к форме, которая является канонической для широкого класса выражений. Переменные в выражении упорядочиваются. Эквивалентные выражения в этом классе не обязательно одинаковы, но их разность упрощается применением **radcan** до нуля.

Примеры:

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

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

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

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

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

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

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

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

Функция $logcontract(expr)$ рекурсивно сканирует выражение $expr$, преобразуя выражения вида $a1*log(b1)+a2*log(b2)+c$ к форме $log(ratsimp(b1^{a1}*b2^{a2}))+c$.

Пример:

```
(%i1)   2*(a*log(x)+3*b*log(y));
```

$$(\%o1) \qquad 2\ (3\,b\,log(y)+a\,log(x))$$

```
(%i2)   logcontract(%);
```

$$(\%o2) \qquad b\,log\left(y^6\right)+a\,log\left(x^2\right)$$

Если объявить переменную n целой (используя $declare(n,integer)$), функция $logcontract$ позволяет включить эту переменную в показатель степени:

```
(%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 Пользовательские функции

Для записи функции необходимо указать её название, а затем, в круглых скобках записать через запятую значения аргументов. Если значением аргумента является список, то он заключается в квадратные скобки, а элементы списка также разделяются запятыми.

Пример:

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

Пользователь может задать собственные функции. Для этого сначала указывается название функции, в скобках перечисляются названия аргументов, после знаков := (двоеточие и равно) следует описание функции. После задания пользовательская функция вызывается точно так, как и встроенные функции **Maxima.**

Пример:

```
(%i44)  f(x):=x^2;
```

$$(\%o44) \qquad f(x):=x^2$$

```
(%i45)  f(3 + 7);
```

$$(\%o45) \qquad 100$$

Не следует использовать для функций названия, зарезервированные для встроенных функций **Maxima.** Для создания функций используется также встроенная функция $define$, которая позволяет преобразовать выражение в функцию. Синтаксис вызова $define$ довольно многообразен:

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

Варианты вызова функции $define$ различаются, какой именно объект создаётся: ординарная функция (аргументы в круглых скобках) или массив (аргументы в квадратных скобках). Если первый аргумент — операторы $funmake, arraymake$, то функция создаётся и вычисляется (аналогично и ev).

Примеры:

Ординарная функция:

```
(%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));
```

$$(\%o3) \qquad cos\,(b)-sin\,(a)$$

Создание функции-массива:

```
(%i1)   define (G2 [x, y], x.y - y.x);
```

$$(\%o1) \qquad G2_{x,y}:=x.y-y.x$$

Создание массива:

```
(%i2)   define (arraymake (F, [u]), cos(u) + 1);
```

$$(\%o2) \qquad F_u:=cos\,(u)+1$$

Использование функции ev для задания пользовательской функции:

```
(%i3)   define (ev (foo (x, y)), sin(x) - cos(y));
```

$$(\%o3) \qquad foo\,(x,y):=sin\,(x)-cos\,(y)$$

2.8 Решение задач элементарной математики

2.8.1 Нахождение корней уравнений и систем алгебраических уравнений

Решение алгебраических уравнений и их систем осуществляется при помощи функции $solve$, в качестве параметров. В первых квадратных скобках указывается список уравнений через запятую, во вторых — список переменных, через запятую (либо несколько упрощённые формы записи):

- $solve(expr,x)$ — решение одного уравнения относительно переменной x;
- $solve(expr)$ — решение уравнения с одной неизвестной и числовыми коэффициентами;
- $solve([eqn_1,...,eqn_n],[x_1,...,x_n])$ — решение системы уравнений.

Примеры:

Решение одного уравнения с одним неизвестным

```
(%i7)   solve(x^2-5*x+4);
```

$$(\%o7) \qquad [x = 1, x = 4]$$

Решение одного уравнения в символьном виде:

```
(%i2)   solve([x-a/x+b], [x]);
```

$$(\%o2) \qquad [x = -\frac{\sqrt{b^2+4\,a}+b}{2}, x = \frac{\sqrt{b^2+4\,a}-b}{2}]$$

Решение системы уравнений в символьном виде:

```
(%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 последнем примере решений несколько, и **Maxima** выдаёт результат в виде списка.

Функция $solve$ применима и для решения тригонометрических уравнений. При этом в случае множества решений у тригонометрических уравнений выдаётся соответствующее сообщение только и одно из решений.

Пример:

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

solve: using arc-trig functions to get a solution.
Some solutions will be lost.
```

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

Также Maxima позволяет находить комплексные корни

```
(%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 Построение графиков и поверхностей

Для вывода графиков на экран или на печать при помощи **Maxima** существуют несколько вариантов форматов и, соответственно, программ вывода графики, а именно:

- $openmath$ (Tcl/Tk программа с графическим интерфейсом пользователя; элемент **xMaxima**)
- $gnuplot$ (мощная утилита для построения графиков, обмен с **Maxima** — через канал)
- $mgnuplot$ (Tk-интерфейс к $gnuplot$ с рудиментарным графическим интерфейсом пользователя; включён в дистрибутив **Maxima**)
- $wxMaxima$ (встроенные возможности $frontend$ -а к Maxima)

Все варианты интерфейса (кроме **wxMaxima**) для построения графиков используют две базовых функции: $plot2d$ (построение двумерных графиков) и $plot3d$ (построение трехмерных графиков).

При использовании **wxMaxima** кроме них используются ещё две аналогичные команды: $wxplot2d$ и $wxplot3d$. Все команды позволяют либо вывести график на экран, либо (в зависимости от параметров функции) в файл.

2.9.1 Построение графика явной функции y = f(x)

График функции $y = f(x)$ на отрезке $[a, b]$ можно построить с помощью функции $plot2d(f(x), [x, a, b], \text{опции})$ или $plot2d(f(x), [x, a, b], [y, c, d], \text{опции})$. Опции не обязательны, однако, для изменения свойств графика их нужно задавать. Параметр $[y, c, d]$ можно не задавать, тогда высота графика выбирается по умолчанию. Построим график функции $y = sin(x)$ на отрезке $[-4\pi, 4\pi]$.

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

Результаты приведены на рис. 2.1, рис. 2.2

2.9.2 Построение графиков функций, заданных параметрически

Для построения графиков функций, заданных параметрически, используется опция $parametric$. Для построения графика указывается область изменения параметра. Пример графика простейшей параметрической функции представлен на рис. 2.3.

Команда построения графика: plot2d ([parametric, cos(t), sin(t), [t,-%pi,%pi], [nticks,80]],[x, -4/3, 4/3])

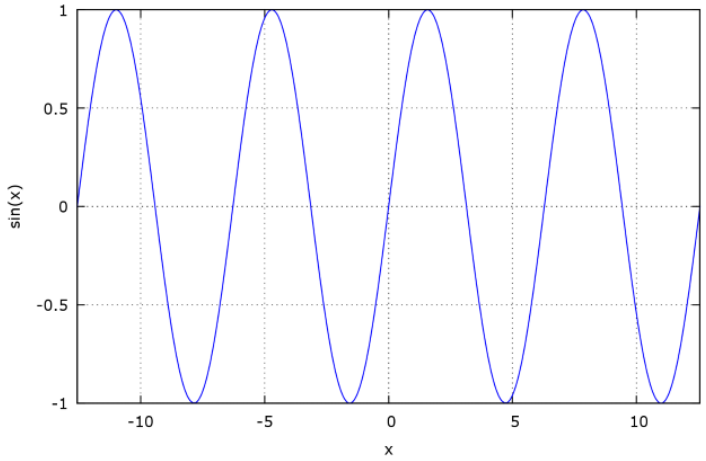

Рис. 2.1. Простейшая команда построения графика

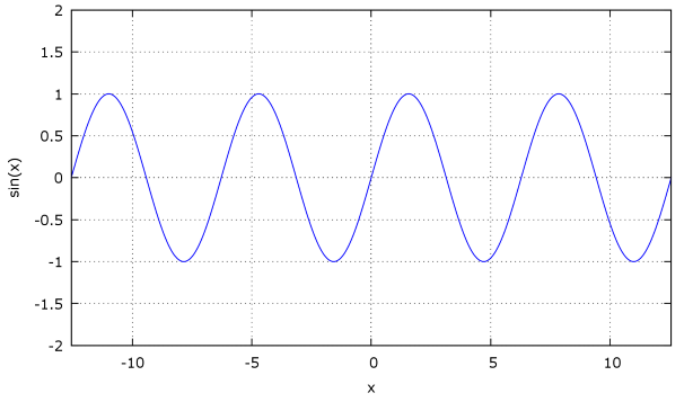

Рис. 2.2. Простейшая команда построения графика с указанием интервала по оси Oy

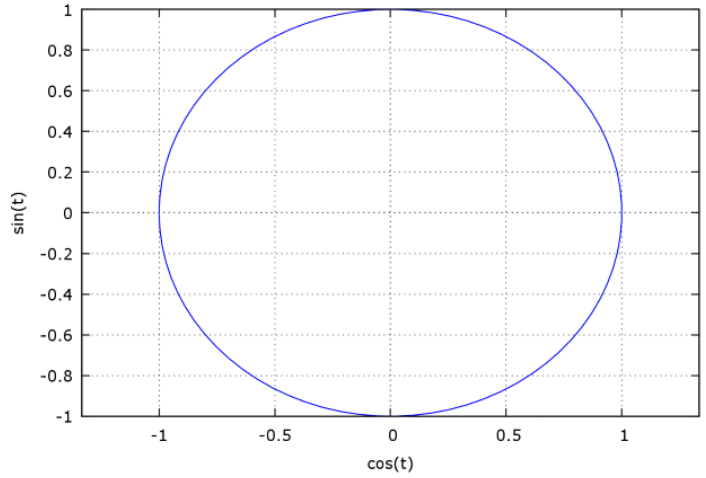

Рис. 2.3. Простейшая команда построения графика функции, заданной параметрически

Опция $ntics$ указывает число точек, по которым проводится кривая.

Рассмотрим некоторые опции.

Опции указываются в виде аргументов функции $plot2d$ в квадратных скобках. Возможна установка легенды, меток на осях, цвета и стиля графика. Применение нескольких опций характеризует следующий пример:

```
(%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)"]);
```

В данном примере в одних осях строятся два графика. Первый $([discrete, xy])$ строится в виде точек по массиву xy с указанием стиля $points$. Второй строится по уравнению функции $2 * \%pi * sqrt(l/980)$ с указанием стиля $lines$. Опция $legend$ указывает подписи кривых, опции $xlabel$ и $ylabel$ — подписи осей. Результат приведён на рис. 2.4.

Формирование массивов для построения графика осуществляется следующим образом:

```
(%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]];
```

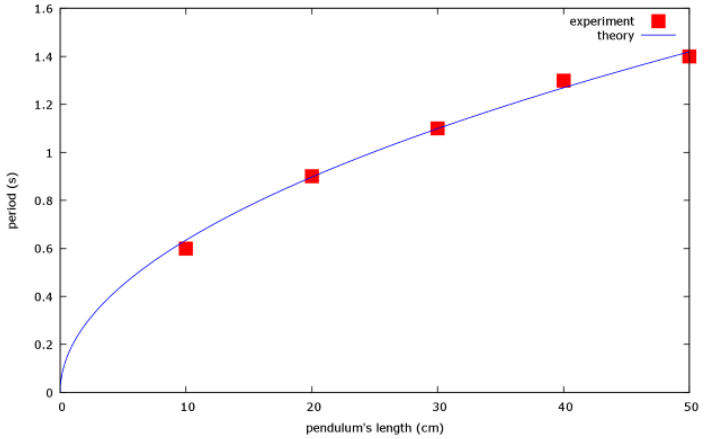

Рис. 2.4. Совмещение на одном графике действия серии опций

Можно комбинировать в одних осях графики кривых различного типа: функции $y = f(x)$ или параметрические

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

например (см.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, ps],
[gnuplot_out_file, "test.eps"]);
```

Опции [gnuplot_term, ps], [gnuplot_out_file, "test.eps"] указывают, что графическая иллюстрация выводится в файл $test.eps$ в формате $postscript$ (бэкенд для вывода графиков — $gnuplot$).

Опции [style,[linespoints,3,2],lines,3,1]] позволяют указать стиль линий на графике (линия с точками или сплошная линия).

Для вывода результатов в формат png можно использовать опции (указание размеров 400,400 в общем случае необязательно): [gnuplot_term, png size 400,400],[gnuplot_out_file, max.png]

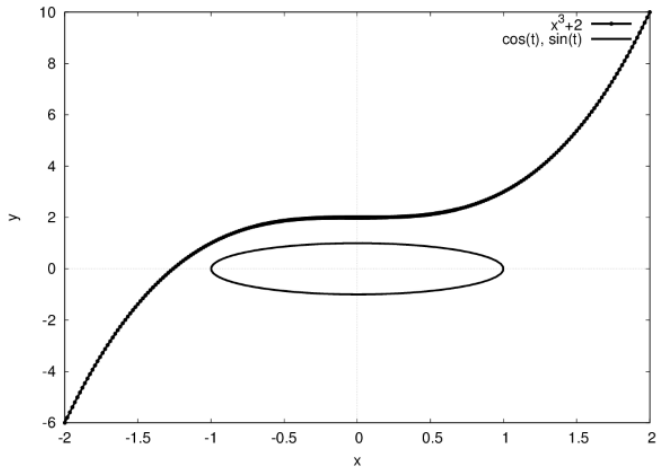

Рис. 2.5. Совмещение на одном графике параметрической и заданной явно кривых

2.9.3 Построение кривых в полярной системе координат

Для построения графика в полярных координатах нужно задать изменение значений полярного радиуса и полярного угла. Пусть $r = r(f)(a \leq f \leq b)$ — зависимость полярного радиуса r от полярного угла f . Тогда график этой функции в полярных координатах можно построить, задав у функции $plot2d$ опцию [gnuplot_preamble, set polar; set zeroaxis]. Данная опция будет действовать лишь при условии, что выбран формат графика $gnuplot$.

Пример: построить в полярных координатах график функции $r = 3(1 - \varphi + \varphi^2),\ 0 \leq \varphi \leq 2\pi$.

Для создания графика используем 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 приведён на рис. 2.6. Толщину и стиль линии можно регулировать, используя опцию $style$ (например, опция [style, [lines,3,1]] устанавливает ширину линии 3 и синий цвет).

Пример: построить в полярных координатах графики трёх функций $r = 6cos(\varphi), r = \varphi, r = 2sin(\varphi), 0 \leq \varphi \leq 2\pi$.

Для создания графика используем команду:

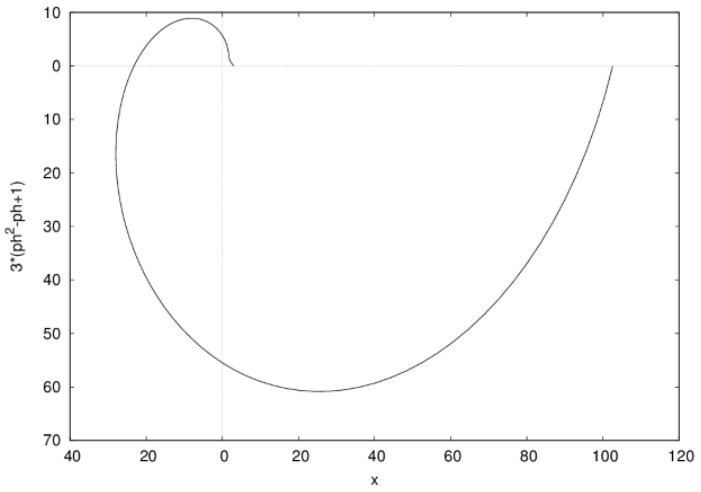

Рис. 2.6. Кривая в полярных координатах

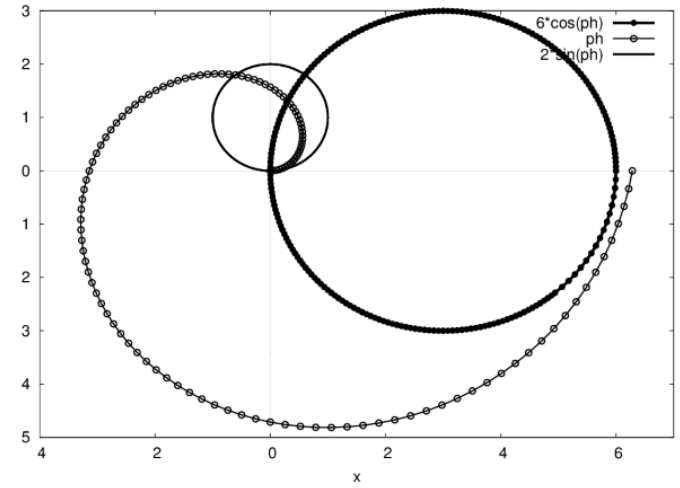

Рис. 2.7. Совмещение на одном графике нескольких параметрических кривых

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

Результат приведён на рис. 2.7.

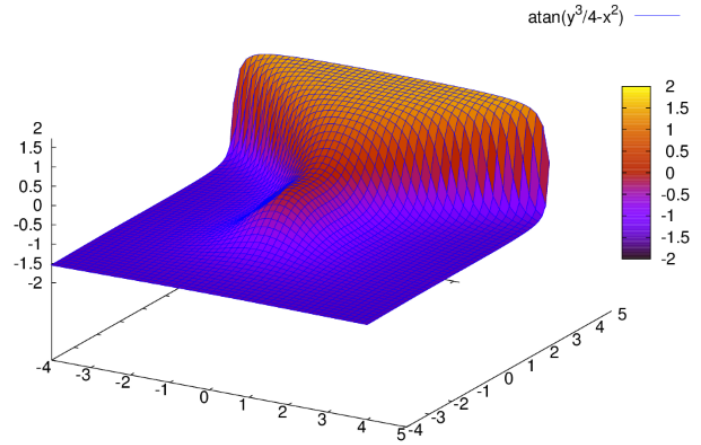

Рис. 2.8. График функции двух переменных с окраской поверхности

2.9.4 Построение трёхмерных графиков

Основная команда для построения трёхмерных графиков — $plot3d$. Рассмотрим технологию построения графиков с использованием интерфейса $gnuplot$. Поверхность функции в цветном изображении строится с использованием опции $pm3d$ (рис. 2.8).

Пример:

```
(%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"]);
```

С использованием этой опции и особенностей программы $gnuplot$ можно построить и изображение линий уровня функции. Пример (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"]);
```

Более строгий результат можно получить, используя стандартный формат функции $plot3d$. Пример (рис. 2.10):

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

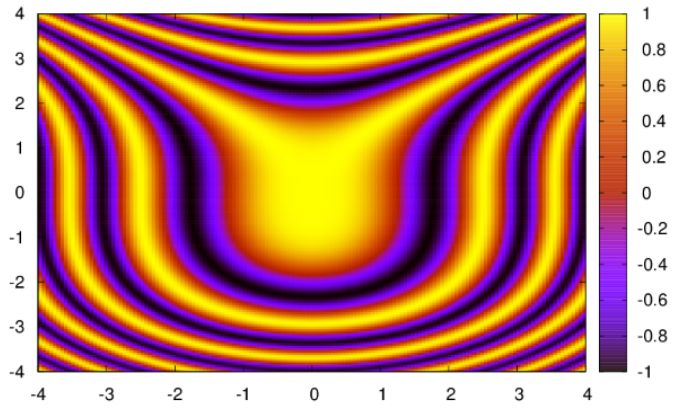

Рис. 2.9. График линий уровня функции двух переменных с окраской поверхности

Для вывода графика в файл всё равно необходимо использовать опции $gnuplot$ (установить терминал $gnuplot$ и имя файла результата). Необходимая команда:

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

Смена формата графики также возможна за счёт использования опций $plot3d$. Пример (вывод графики в формате $openmath$ — рис. 2.11):

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

Достоинством данного формата является встроенная возможность сохранения копии графического изображения в файл, редактирования и поворота построенного графика.

Функция, для которой строится трёхмерный график, может задаваться как **Maxima** или **Lisp**-функция, лямбда-функция либо выражение **Maxima** общего вида. При использовании формата $plot3d(f, ...)$ выражение f рассматривается как функция двух переменных. При использовании формата $plot3d([f_1, f_2, f_3], ...)$, каждая функция (f_1, f_2, f_3) рассматривается как функция трёх переменных.

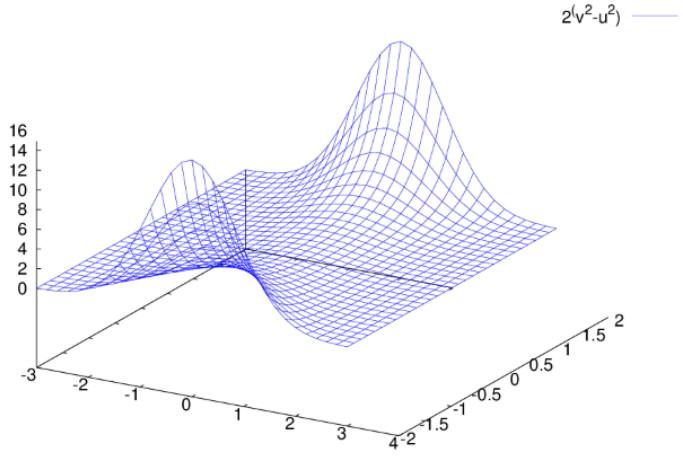

Рис. 2.10. Простой график функции двух переменных

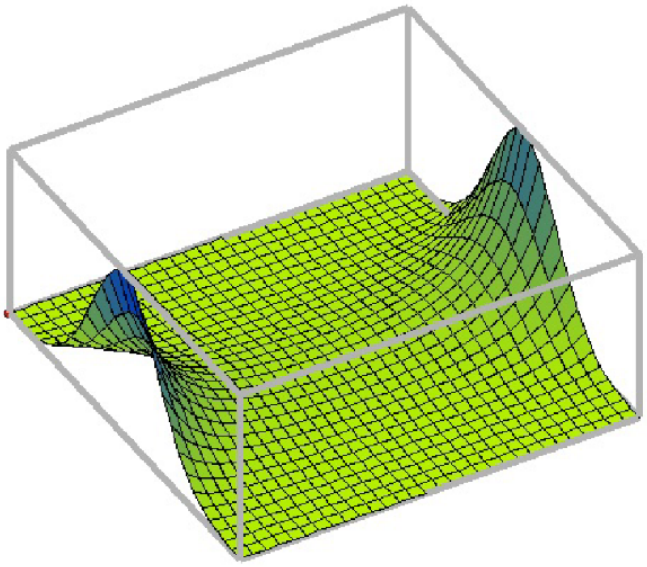

Рис. 2.11. Простой график функции двух переменных (формат OpenMath)

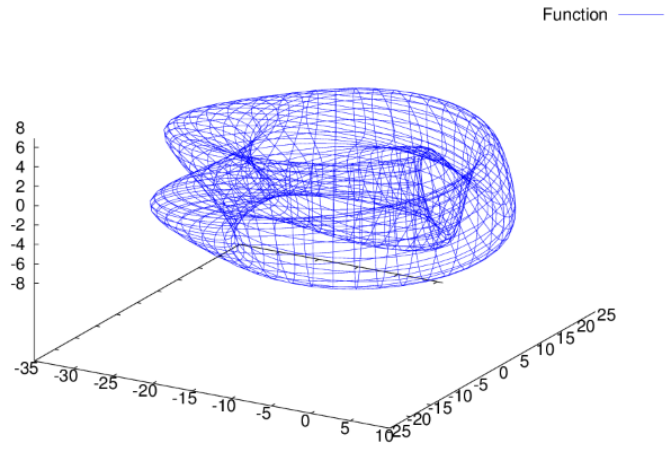

Рис. 2.12. График функции, определённой в формате [f1,f2,f3]

Пример использования формата $plot3d([f_1, f_2, f_3], ...)$ (рис. 2.12):

Функция $plot3d$ позволяет строить графики функций, заданных в цилиндрических или сферических координатах за счёт использования преобразования координат (опция [transform_xy, polar_to_xy] или функция make_transform (vars, fx, fy, fz)).

Определённые преимущества обеспечивает формат $wxplot$, имеющийся в графическом интерфейсе **wxMaxima** ($wxplot2d$ и $wxplot3d$). Команда построения графика в формате **wxMaxima** по синтаксису мало отличается от синтаксиса команд $plot2d$ и $plot3d$. Качество воспроизведения графиков на экране **wxMaxima** относительно невысокое, но легко, выделив график щелчком мыши, сохранить его в файл (по умолчанию $maxout.png$). Качество копии в файле намного лучше, чем рисунка в окне **wxMaxima.**