МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ
БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ
УНИВЕРСИТЕТ»

А.А. Каменский, А.А. Некипелов, В.В. Чернушкин

КОМПЬЮТЕРНЫЕ ЛАБОРАТОРНЫЕ ЗАНЯТИЯ ПО ТЕОРЕТИЧЕСКОЙ МЕХАНИКЕ
Часть 1

Учебное пособие

Воронеж
Издательский дом ВГУ
2016

ŋqdqɐeɎ bɆ dədɆbɎɜə d ɕɔɑbɔɐ ɈqddɎqɐè

Содержание

введённой команды, но не выводить его на экран.

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.

В Maxima определены арифметические операции + (сложение), – (вычитание или унарный минус как знак отрицательного числа), * (умножение), / (деление). Существует несколько идентичных способов задания возведения в степень (например, **). Операция нахождение факториала обозначается восклицательным знаком, например, 5!. Для увеличения приоритета операции, как и в математике, используются круглые скобки: ().

```
(%i1) 25!;
```

$$15511210043330985984000000$$

```
(%i2) 2**100;
```

$$1267650600228229401496703205376$$

В Maxima для удобства вычислений имеется ряд встроенных констант. Самые распространённые из них — %pi (число пи), %e (число e – основание натуральных логарифмов), %i (мнимая единица). По умолчанию все вычисления проводятся на множестве комплексных чисел. Символ % используется в начале служебных имен Maxima для исключения пересечений с пользовательскими идентификаторами. Одиночный символ % используется для обращения к результату предыдущего вычисления. Кроме того, все ячейки нумеруются символами %i номер и %o номер, что позволяет обращаться из любой ячейки программы к любому ранее вычисленному значению.

```
(%i3) %pi;
```

$$\pi$$

```
(%i4) 180/%pi, numer;
```

$$57.29577951308232$$

Помимо этого, существует и традиционный способ запоминания вычисленных значений – посредством переменных. Имена переменных и функций (идентификаторы) могут включать в себя буквы латинского алфавита, символ подчеркивания и цифры. Идентификаторы не могут начинаться с цифры. Необходимо также подчеркнуть, что прописные и строчные буквы в идентификаторах различаются, так что имена temp, TEMP и TemP задают разные объекты. Присваивание значения переменной осуществляется с использованием символа : (двоеточие), например, x:5.

```
(%i1) a:5;
```

$$5$$

```
(%i2) f:3/a;
```

$$\frac{3}{5}$$

```
(%i3) f, numer;
```

$$0.6$$

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.

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.

Заметим также, что kill(all) полностью очищает память, в том числе и ранее вычисленные значения. Поэтому после использования kill(all) нумерация ячеек начинается сначала.

В Maxima имеется достаточно большой набор встроенных математических функций. Вот некоторые из них:

– тригонометрические функции: sin (синус), cos (косинус), tan(тангенс), cot (котангенс), sec (секанс, sec x = 1/cos x), csc (косеканс, csc x = 1/sin x);

– обратные тригонометрические функции: asin (арксинус), acos (арккосинус), atan (арктангенс), acot (арккотангенс);

– гиперболические функции – sinh (гиперболический синус), cosh (гиперболический косинус), tanh (гиперболический тангенс), coth (гиперболический котангенс), sech (гиперболический секанс), cosh (гиперболический косеканс);

Среди других наиболее часто используемых функций отметим следующие: log (натуральный логарифм), sqrt (квадратный корень), mod

(остаток от деления), abs (модуль), sign (определяет знак аргумента: pos – положительный, neg – отрицательный, pnz – не определен, zero – значение равно нулю), а также min(x1,. . . ,xn) и max(x1,. . . ,xn) – нахождение соответственно минимального и максимального значения в списке аргументов. Для записи функции необходимо указать ее название, а затем, в круглых скобках записать через запятую значения аргументов.

```
--> x:48;
```

$$48$$

```
--> y:sqrt(x+1);
```

$$7$$

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.

```
--> r(x,y,z):=sqrt(x*x+y*y+z*z);
```

$$\mathrm{r}\,(x, y, z) := \sqrt{z \cdot z + y \cdot y + x \cdot x}$$

```
--> ra:r(1,1,8);
```

$$\sqrt{66}$$

```
--> float(ra);
```

$$8.12403840463596$$

К наиболее популярным задачам элементарной математики, решаемых с помощью Maxima, можно отнести вычисление и преобразование арифметических выражений, построение графиков функций, решение уравнений и систем алгебраических уравнений. Рассмотрим ряд полезных функций:

assume — ввод ограничений на значения переменных;

forget — снятие ограничений, заданных командой assume;

При необходимости принудительного задания типа переменной используется функция declare. Например, команда declare(n, integer) – делает переменную n целой;

factor — разложение на множители

expand — раскрытие скобок

```
--> expand((y+4)*5+5*y*(y-9)+1);
```

$$5 \cdot y^2 - 40 \cdot y + 21$$

−−> `factor(x^2+5*x-6);`

$$(x-1)\cdot(x+6)$$

ratsimp —- упрощение выражения. 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.

partfrac —- преобразовать в простые дроби по заданной переменной

trigsimp — тригонометрическое упрощение

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. Всегда, когда возможно, заданное выражение линеаризуется.

trigexpand (тригонометрическое раскрытие скобок) — использует формулы преобразования сумм двух углов для представления введенного выражения в как можно более простом виде – где в качестве аргумента только одна переменная

trigreduce (приведение тригонометрическое) — преобразует тригонометрическое выражение к сумме элементов, каждый из которых содержит только sin или cos

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.

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

−−> `subst(4,r,r+q);`

$$q+4$$

−−> `subst(r=4,r+q);`

$$q+4$$

ŋqdqɐeꓤ bɆ dedɆbꓤɜe d ɕɔɑbɔɐ Ɉqddꓤqɐè

`-->subst([r=4,q=10],r+q);`

$$14$$

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:

solve([список выражений], [список переменных]) — решение системы уравнений относительно заданных переменных.

Несколько упрощённые формы записи:

solve(expr, x) — решение одного уравнения относительно переменной x;

solve(expr) — решение уравнения с одной неизвестной и числовыми коэффициентами;

Уравнение может задаваться выражением со знаком =, в этом случае для выделения левой и правой частей уравнения используется функция part.

`-->eq:100/x+100/(x+1)=45;`

$$\frac{100}{x+1}+\frac{100}{x}=45$$

`-->part(eq,1);`

$$\frac{100}{x+1}+\frac{100}{x}$$

`-->part(eq,2);`

$$45$$

`-->solve(eq);`

$$\left[x=-\frac{5}{9}, x=4\right]$$

Предыдущий вывод представляет собой список. Списки представляют собой набор значений, заключенных в квадратные скобки и разделенных запятыми. Для создания списков используется функция makelist, а для добавления элементов – функция append. Например, создадим список корней уравнения из предыдущего примера:

`-->l:%;`

$$\left[x=-\frac{5}{9}, x=4\right]$$

ŋqdqyeY bE dedEbYse d sɑɑbɔy Jqdqyqè