Министерство образования и науки Российской Федерации
Федеральное агентство по образованию
Дальневосточный государственный университет

Л.А. МОЛЧАНОВА

ВВЕДЕНИЕ В MAPLE

Методические указания для студентов
математических специальностей

Владивосток
Издательство Дальневосточного университета
2006

ББК xxx
Mxx

Рецензенты:
А.В. Бажанов, к.ф.-м.н. (ИМКН ДВГУ);
М.А Князева, к.т.н. (ИАПУ ДВО, РАН).

Молчанова Л.А.

М75 **Введение в Maple.** Учебно-методическое пособие. - Владивосток: Изд-во Дальневост. ун-та, 2006. - 36 с.

Методические указания разработаны для студентов Института математики и компьютерных наук ДВГУ. В них дается теоретический материал, позволяющий студентам использовать средства математического пакета Maple в своей практической деятельности при выполнении заданий по спецкурсу Пакеты прикладных программ и программировании задач из различных разделов математики.

Для студентов математических специальностей.

$\mathrm{M}\frac{1702050000}{180(03)-2006}$ ББК xx.xxx

Содержание

1 Основы Maple

Maple - это пакет для аналитических вычислений на компьютере, содержащий более двух тысяч команд, которые позволяют решать задачи алгебры, геометрии, математического анализа, дифференциальных уравнений, статистики, математической физики.

Основными объектами являются формулы и действия с ними.

Работа в Maple проходит в режиме сессии - пользователь вводит предложения (команды, выражения, процедуры), которые воспринимаются условно и обрабатываются Maple. Рабочее поле разделяется на три части:

1) область ввода - состоит из командных строк. Команды вводятся после приглашения - знака >;

2) область вывода - содержит результаты обработки введенных команд в виде аналитических выражений, графических объектов или сообщений об ошибке;

3) область текстовых комментариев - содержит любую текстовую информацию, которая может пояснить выполняемые процедуры. Текстовые строки не воспринимаются Maple и никак не обрабатываются.

Для отмены всех сделанных назначений и начала нового сеанса без выхода из системы используется команда **restart**.

В состав пакета Maple входит большое число библиотек, подключение которых осуществляется командой **with(имя библиотеки)**. Например, пакет **linalg** содержит операции линейной алгебры, а функции для решения дифференциальных уравнений находятся в библиотеке **DETools**.

1.1 Важные символы

Алфавит Maple-языка содержит 26 малых латинских букв от a до z, 26 больших латинских букв от A до Z, 10 арабских цифр (от 0 до 9), 32 специальных символа и ключевые слова. Отметим некоторые специальные символы.

Каждое введенное предложение должно завершаться разделителем: точка с запятой (;) или двоеточием (:). Если ввод предложения завершается точкой с запятой, то в строке под предложением сразу будет отклик: результат исполнения предложения или сообщение об ошибке. Разделитель (:) используется для отмены вывода, когда предложение выполняется системой, но результат не выводится на экран.

В Maple применяются круглые, квадратные и фигурные скобки. С помощью круглых скобок задают порядок при построении математических выражений и обрамляют аргументы функций и параметры в записи ко-

манд. Квадратные скобки нужны для работы с индексными величинами. Фигурные скобки используются для формирования множеств.

Две последовательные точки (..) в параметрах команд применяются для определения интервала изменения переменных.

Знаком процента (%) обозначается предшествующий вывод. Два знака процента относятся к предпоследнему результату, а предшественник предпоследнего результата обозначается тремя знаками процента.

Для арифметических операций используются знаки +, -, *, /, ^(возведение в степень),! (факториал).

Для операций отношения имеются знаки >, <, >=, <=, <>, =, а для конструирования булевых выражений используются команды **not, or, and**.

Обратный слеш (\) используется для переносов, а для комментирования предусмотрен символ #. Вся строка после этого символа не выполняется.

Знак равенства (=) используется при формировании уравнений, а знак присваивания (:=) - при задании значений переменных.

Символ (e) используется для ввода действительных чисел с порядком, а символ D - для задания дифференциального оператора.

Символ (') (апостроф) используется в качестве ограничителя строки. Для включения апострофа в символьную строку нужно поставить два апострофа подряд.

С помощью двойных кавычек(") задается строка. Для включения двойных кавычек в строку их нужно продублировать, при этом их предваряет обратный слеш.

Фраза, заключенная в обратные кавычки (‘), воспринимается Maple, как символ.

При помощи оператора конкатенации (||) и переменной можно организовать комбинированные имена.

При помощи знака-повторителя $ можно создавать последовательности выражений из символов и чисел.

Метки представлены символами %N, где N - номер метки.

Знак тильда (~) используется для отметки предполагаемых переменных.

1.2 Основные объекты

Система Maple оперирует со множеством объектов, используя для работы различные типы данных. Это позволяет применять свои правила обработки для каждого типа.

Простейшими объектами являются числа, константы, строки и переменные.

1.2.1 Числа и константы

бывают действительные и комплексные.

Комплексное число записывается в алгебраической форме z=x+iy, и в командной строке такая запись должна выглядеть так:

> z:=x+I*y;

Функции **Re(x)** и **Im(x)** возвращают действительную и мнимую части комплексных чисел.

Вещественные числа разделяются на целые и рациональные. Целые числа выражаются цифрами в десятичной записи. Рациональные числа могут быть представлены в 3-х видах:

1) рациональной дроби с использованием оператора деления, например: 28/70;

2) с плавающей запятой, например: 2.3;

3) в показательной форме, например: 1,602e-19 означает $1,602\,10^{-19}$.

Десятичная точка в числах имеет особый статус: указание ее в любом месте числа, а также в конце, делает число вещественным и ведет перевод вычислений в режим вычислений с вещественными числами. При этом количеством выводимых после десятичной точки цифр можно управлять, задавая их значение системной переменной **Digits**:

> 75/4;Digits:=5:75/4.0;

$$\frac{75}{4}$$

$$18.750$$

> Digits:=3:12345678/7.;

$$.176\,10^{7}$$

Возможна работа с числами, имеющими различное основание, в частности, с бинарными числами (binary), восьмеричными (octal) и шестнадцатеричными (hex). Функция **convert** позволяет преобразовывать форматы чисел:

```
> x:=61;b:=convert(x,binary);
> o:=convert(x,octal);h:=convert(x,hex);
```

$$61$$

$$b:=111101$$

$$o:=75$$

$$h:=3D$$

```
> convert(o-5,decimal,octal);
```

56

Основные математические константы:
Pi - число π; I - мнимая единица i; infinity - бесконечность.

1.2.2 Переменные

Переменные идентифицируются именем - набором символов, начинающихся с буквы, причем большие и малые буквы различаются. Кроме букв могут употребляться цифры и знак подчеркивания. Если последним символом имени идет тильда (~), то это имя переменной, относительно которой сделаны назначения (определена вещественность или положительность и т.д.). Составные имена могут быть сформированы при помощи оператора конкатенации (||) или команды **cat**.

Для обозначения служебных констант используются имена, начинающиеся с подчеркивания. Неопределенные константы, возникающие при решении дифференциального уравнения, именуются $\_C1$, $\_C2$ и т.д.

Для просмотра содержимого переменной простого типа нужно лишь ввести имя переменной (для массивов и других составных типов используется команда **eval**).

Чтобы освободить конкретную переменную от предшествующих назначений, нужно этой переменной присвоить ее имя, заключенное в прямые одинарные кавычки (') (апострофы). Например:

```
> ex:=x^2+exp(y):ex;
```

$$ex := x^2 + e^y$$

```
> ex:='ex':ex;
```

ex

Если расчеты предполагают, что переменные могут иметь определенные признаки, то для придания им статуса предполагаемых используется оператор **assume**:

assume(x,prop)

где x - переменная, имя или выражение, *prop* - свойство.

```
> assume(x,positive):s:=x->sqrt(x);s(2.);
```

s:=sqrt

1.414213562

Для защиты от изменений существует команда **protect**, а для снятия защиты - **unprotect**.

1.2.3 Строки и символы

Строкой является любой набор символов, заключенный в двойные кавычки. Для включения двойных кавычек их нужно продублировать, при этом их предваряет обратный слеш (\). Строка отличается от символа, который получается, если фразу заключить в обратные кавычки. Символ воспринимается **Maple** как единое целое, а строка состоит из символов, и с каждым из них можно работать отдельно. Например:

```
> v1:="String";v2:=`Symbol`;
```

v1:="String"

v2 := Symbol

```
> v1[2];v2[1];
```

"t"

$Symbol_1$

1.2.4 Типы переменных

В системе существует множество типов переменных: от известных вещественного, целого и строкового до тех, которые необходимы для выполнения и программирования аналитических преобразований: дробь, функции, индексная переменная, процедура, множество, разложение, последовательность выражений, массив, списки и некоторые другие. Комбинируя данные разных типов, можно получить, например, списки множеств или последовательность списков и т.д.

Рассмотрим базовые типы: последовательность выражений, списки, множества и массивы и некоторые команды для работы с ними.

1.2.5 Последовательность выражений

Последовательности выражений удобны для накопления уравнений, переменных и т.д. Переменная этого типа получается как последовательность выражений, разделенных запятыми и завершенная фиксатором. Могут встречаться одинаковые элементы. При помощи операции конкатенации (||) и переменной можно организовывать комбинированные имена. Для объединения нескольких последовательностей выражений достаточно записать их через запятые. Для обозначения пустой последовательности имеется специальная константа $NULL$.

Для автоматического формирования последовательности выражений применим специальный оператор $ (доллар), после которого можно указывать число выражений или задавать диапазон формирования выражений:

```
> f$5;
```

f,f,f,f,f

```
> V[i^2+1]$i=1..5;
```

$$V_2, V_5, V_{10}, V_{17}, V_{26}$$

Для организации последовательностей имеется команда

seq(EX,k=N..M)

Первый параметр ex задает k-ый член последовательности, а второй параметр определяет диапазон изменения целой переменной. Например:

```
> seq(k,k=-1..11);
```

-1,0,1,2,3,4,5,6,7,8,9,10,11

Команда **seq** позволяет создавать последовательность выражений, состоящих из символьных величин. Например, можно сформировать систему уравнений с неизвестными d_1, d_2, d_3, d1, d2, d3 одной командой:

```
> eq:=seq(d[k]=k^2+d||k,k=1..3);
```

$$eq := d_1 = 1 + d1, d_2 = 4 + d2, d_3 = 9 + d3$$

Для выбора элемента последовательности нужно указать его номер (целое положительное число), считая слева направо. При указании отрицательного числа нумерация идет от конца последовательности. Для выбора нескольких последовательных элементов нужно указать диапазон. В примере вычисляется производная от ряда тригонометрических функций и выбираются второй и предпоследний элементы и группа элементов от третьего до предпоследнего:

```
> Df:=seq(D(f),f=[sin,cos,tan,cot,sec,csc]);
```

$$Df := \cos, -\sin, 1 + tan^2, -1 - cot^2, sec\, tan, -csc\, cot$$

```
> DF[2];Df[3..-2];
```

-sin

$$1 + tan^2, -1 - cot^2, sec\, tan$$

1.2.6 Списки и множество

Список - это последовательность выражений, заключенная в квадратные скобки. При выборе элемента из списка в квадратных скобках указывается номер или диапазон номеров. Нумерация ведется слева направо, если номера положительные. Отрицательные числа применяются для указания порядкового номера справа налево. Для работы со списками применяются команды выбора по заданному правилу (**select**) и удаления (**remove**). Для превращения списка в последовательность выражений следует использовать команду **op**.Этой же командой можно сделать объединение данных двух списков. Например:

```
> LA:=[2,3,5,7]:LB:=[4,6,8,9]:LAB:=[op(LA),op(LB)];
```

LAB:=[2,3,5,7,4,6,8,9]

Множество - это последовательность выражений, заключенная в фигурные скобки. В этом виде задают систему уравнений и получают найденные Maple решения уравнений.

При работе с множествами можно пользоваться операциями объединения **union**, пересечения **intersect**, вычитания **minus**. Для образования пустого множества достаточно пары фигурных скобок.

```
> empt:={};a:={2,3};new:=empt union {abc,3};new minus {};
```

$$empt := \{\}$$
$$a:=\{2,3\}$$
$$new:=\{3,abc\}$$
$$\{3,abc\}$$

```
> new intersect a;
```

$$\{3\}$$

1.2.7 Массив

Массивы позволяют организовать данные, используя для индексации отрицательные числа и нуль. Массив создается по команде

array(FUN,DIA,LIS)

Параметры ее имеют следующее назначение: функция FUN задает свойство массива, переменная DIA определяет диапазоны изменения индексов, а LIS есть список элементов массива. Каждый из параметров может быть опущен, но по крайней мере один диапазон или список элементов должны быть заданы. В качестве имени FUN могут быть использованы следующие стандартные методы: symmetric, antisymmetric, sparse, diagonal, identity, package. Это позволяет определить соответственно массивы симметричные и кососимметричные, разряженные (нули для упомянутых элементов), массивы с ненулевой диагональю и единичные. Имя package указывает на процедуру, служащую для ввода элементов.

Если массив многомерный, то соответствующие диапазоны должны задаваться подряд через запятую. Для двумерного массива элементами списка являются списки.

Если значения элементов массива не заданы, то с использованием индексной формы можно присвоить элементам соответствующие значения. Для отображения в области вывода значений элементов массива следует воспользоваться командой **print**.

Например, для создания массива из четырех элементов с именем x используется следующая конструкция:

```
> x:=array(-1..2);x[1]:=1:x[2]:=2:x[3]:=3:print(x);
```

x:=array(-1..2,[])

[1,2,3]

Создадим массив C и заполним его поэлементно:

```
> C:=array(1..3,1..3);
```

$$C := \text{array}(1..\ 3,\ 1\ ..\ 3,\ [])$$

```
> for i to 3 do
> for j to 3 do
> C[i,j]:=i+j
> od
> od;
```

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

1.2.8 Стандартные функции

Обращение к стандартной функции - элементарная операция, если известно имя нужной функции.

Стандартные математические функции

Из огромного списка функций перечислим достаточно представительное подмножество из известных математических функций. Все функции имеют один аргумент x. Как правило, если аргументом функции является фундаментальная константа, целое или рациональное число, то функция выводится с таким аргументом без получения результата в форме действительного числа с плавающей точкой.

Алгебраические функции

- `exp(x)` Экспоненциальная функция
- `ln(x)` Натуральный логарифм
- `log10(x)` Логарифм по основанию 10
- `sqrt(x)` Квадратный корень
- `abs(x)` Абсолютное значение x

Например:

```
> exp(1);exp(1.0);
```

$$e$$

$$2.718281828$$

Функции с элементами сравнения

- `ceil` Наименьшее целое, большее или равное x
- `floor` Наибольшее целое, меньшее или равное x
- `frac` Дробная часть числа
- `trunc` Меньшее целое, округленное в направлении x=0
- `round` Округленное значение числа

- `signum  Функция знака(-1 при x<0,0 при x=0 и +1 при x>0`

Приведем примеры работы этих функций, для компактности вывода организовав последовательность выражений из результатов вычислений:

```
> x:=-1.4:round(x);trunc(x);floor(x);ceil(x);
```

-1,-1,-2,-1

```
> x:=-1.5:round(x);trunc(x);floor(x);ceil(x);
```

-2,-1,-2,-1

```
> x:=1.4:round(x);trunc(x);floor(x);ceil(x);
```

1,1,1,2

```
> x:=1.5:round(x);trunc(x);floor(x);ceil(x);
```

2,1,1,2

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

π-3

0.141592654

Аргументы тригонометрических (sin, cos, tan, cot, sec, csc), обратных тригонометрических (arcsin, arccos, arctan, arccot, arcsec, arccsc), гиперболических(sinh, cosh, tanh, coth, sech, csch), обратных гиперболических (arcsinh, arccosh, arctanh, arccoth, arcsech, arccsch) функций должны быть заданы в радианах.

Приведем примеры работы некоторых из этих функций:

```
> Digits:=7:a:=15*0.01745329;
```

a := 0.2617994

```
> sec(a)=sqrt(1+(tan(a))^2);
```

1.035276 = 1.035276

```
> (csc(a))^2-(cot(a))^2=1;
```

1.00000 = 1

```
> Digits:=3:
> sin(I*a)=I*sinh(a);(sinh(a))^2=(cosh(2*a)-1)/2.;
```

0.265 I = 0.265 I

0.0702 = 0.070

```
> Digits:=5:
> arcsin(a)=arctan(a/sqrt(1-a^2));
```

0.26489 = 0.26489

```
> arctanh(a)=ln((a+1)/(1-a))*0.5;
```

0.26804 = 0.26804

Функции для работы со строковыми данными

Строковые данные - это совокупность любых символов в обратных апострофах (‘).

Для интерактивного ввода строк можно использовать функцию **readline (filename)**

задав в качестве имени файла **filename** или опустив его. В этом случае строка вводится с клавиатуры, например:

```
> s:=readline();
> Привет мой друг!
```

s:=Привет мой друг

Имеется ряд функций для работы со строками, из которых наиболее важные указывают в ниже.

Обработка строк

- `length(str)` Возвращает число символов в строке str
- `substring(str,a..b)` Возвращает подстроку в виде символов
- от a-го до b-го строки str
- `cat(str1,str2,..)` Возвращает строку, полученную
- объединением строк str1, str2..

Пример применения этих функций:

```
> str:=‘Hello‘;length(str);substring(str,4..length(str));
```

str := Hello

5

lo

```
> s:=cat(str,‘ my‘,‘ friend‘);
```

s := Hello my friend

2 Аналитические преобразования в Maple

Структура выражений

Maple - это система для манипулирования с выражениями. Выражение в системе - это объект, вполне соответствующий сути обычного математического выражения. Оно может содержать операторы, операнды и функции с параметрами. Выражения могут эволюционировать, т.е. изменяться в соответствии с заданными математическими законами и правилами преобразования. Например, это можно осуществить с помощью функции упрощения **simplify**.

Для выполнения любых математических операций необходимо обеспечить ввод в систему исходных данных - в общем случае математических выражений. Для ввода таких выражений и текстовых комментариев служат два типа строк ввода - для ввода математических выражений и тестовых комментариев. Переключение типа текущей строки ввода осуществляется клавишей F5. Строка ввода математических выражений имеет отличительный символ >, а строка ввода текста никаких признаков не имеет. В строке ввода может быть несколько выражений.

Во внутреннем представлении Maple каждый объект (формула, уравнение, таблица и др.) состоит из подобъектов, каждый их которых, в свою очередь, может состоять из подобъектов, и т.д., вплоть до базисных элементов, так что получается древовидная структура, При работе с большими выражениями часто требуется извлечь отдельные элементы структуры и преобразовать их. Команда **nops** выводит число элементов первого уровня, а команда **op** выводит их в виде последовательных выражений. Пример:

```
> ob:=x^3-y+sin(z^2);nops(ob);op(ob);
```

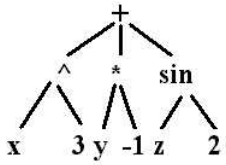
$$ob := x^3 - y + \sin(z^2)$$

$$3$$

$$x^3, -y, sin(z^2)$$

Объект хранится в виде древовидной структуры, в узлах которой находятся операции (+, *, , sin), а ветви указывают на операнды.

Для введенного выражения ob дерево схематически выглядит так:



Для подстановки выражения *new* в *n*-й операнд первого уровня выражения *ex* используется команда

>subsop(n=new,ex)

При анализе выражений полезна команда **has(ex,var)**, позволяющая определить содержит ли объект *ex* выражение или переменную *var*. Результатом выполнения команды является булево выражение. Проверить, входит ли в объект *ex* объект типа *typ*, позволяет команда **hastype(ex,typ)**. Эти команды особенно полезны при программировании, когда для работы существенна информация о типе или составных частях выражения.

Определить тип выражения помогает команда **whattype(ex)**

Таблица 1. Перечень типов

Термин	Значение
">"*"	Произведение выражений
"+"	Сумма выражения
"^"	Степень
".."	Диапазон
"::"	Определение типа
"<:","<=","<>","="	Операция отношения

"and","not","or"	Булевы операции
array	Массив
exprseq	Последовательность выражений
float	Вещественное число
fraction	Дробь
function	Функция
indexed	Индексная величина
integer	Целое число
list	Список
procedure	Процедура
series	Разложение
set	Множество
string	Строка
symbol	Символьная величина
uneval	Невычисленное выражение
Matrix	Матрица

2.1 Преобразование математических выражений

Maple обладает широкими возможностями для проведения аналитических преобразований математических формул. К ним относятся такие операции, как приведение подобных, разложение на множители, раскрытие скобок, приведение рациональной дроби к нормальному виду и многие другие.

2.2 Выделение частей выражений

В процессе преобразований может потребоваться выделить левую или правую часть уравнения, определить числитель или знаменатель дроби. Для этих действий применяются команды, перечисленные ниже, где приняты следующие обозначения: *eq* - уравнение или диапазон, *rat* - рациональная дробь.

Команды выделения

- `rhs(eq)` выделение правой части выражения
- `lhs(eq)` выделение левой части выражения
- `denom(rat)` выделение знаменателя рациональной дроби
- `numer(rat)` выделение числителя рациональной дроби

Математическая формула, над которой будут производиться преобразования, записывается в следующей форме:

$>$ **eq:=exp1=exp2**

Здесь eq - произвольное имя выражения, $exp1$ - условное обозначение левой части формулы, $exp2$ - условное обозначение правой части формулы. Примеры:

```
> eq:=a^2-b^2=c; lhs(eq); rhs(eq);
```

$$eq := a^2 - b^2 = c$$

$$a^2 - b^2$$

$$c$$

```
> f:=(a^2+b)/(2*a-b); numer(f); denom(f);
```

$$f := \frac{a^2 + b}{2a - b}$$

$$a^2 + b$$

2a-b

2.3 Тождественные преобразования выражений

Ниже в алфавитном порядке перечислены основные команды, применяемые для упрощения выражений, и использованы следующие обозначения: $expr$ - выражение, var - переменная или список переменных, rat - рациональная дробь, opt - дополнительные условия.

Команды упрощения

- `collect(ex,var)` Приведение подобных членов
- `combine(expr,var)` Объединение членов
- `expand(expr)` Раскрытие скобок
- `factor(expr)` Разложение выражения на множители
- `normal(rat)` Нормализация дроби
- `radnormal(expr)` Упрощение выражений с корнями разных степеней
- `simplify(expr,opt)` Упрощение выражений

Пример на раскрытие выражения:

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

$$eq := (x+1)(x-1)(x^2 - x + 1)(x^2 + x + 1)$$

```
> expand(ex);
```

$$x^6 - 1$$

Пример разложения многочлена на множители:

```
> p:=x^5-x^4-7*x^3+x^2+6*x; factor(p);
```

$$p := x^5 - x^4 - 7x^3 + x^2 + 6x$$

(x-1)(x-3)(x+2)(x+1)

Команда **expand** может иметь дополнительный параметр, позволяющий при раскрытии скобок оставлять определенное выражение без изменений. Например, пусть требуется каждое слагаемое выражения умножить на выражение (x+a). Тогда в командной строке следует написать:

```
> expand((x+a)*(ln(x)+exp(x)-y^2), (x+a));
```

$$(x+a)\ln x + (x+a)e^x - (x+a)^2$$

Присваивания результата для дальнейшей работы

```
> f:=(a^4-b^4)/((a^2+b^2)*a*b); nf:=normal(f);
```

$$f := \frac{a^4 - b^4}{(a^2+b^2)ab}$$

$$nd := \frac{a^2 - b^2}{ab}$$

Пример на упрощение выражений:

```
> ex:=(cos(x)-sin(x))*(cos(x)+sin(x)):simplify(ex);
```

$$2\cos(x)^2 - 1$$

В команде **simplify** в качестве параметров можно указать, какие выражения преобразовывать. Например, при указании **simplify(ex,trig)** будет производиться упрощение при использовании большого числа тригонометрических соотношений.

Стандартные параметры имеют названия:

power - для степенных преобразований;

radical или sqrt - для преобразования корней;

exp - преобразование экспонент;

ln - преобразование логарифмов.

Объединить показатели степенных функций или понизить степень тригонометрических функций можно при помощи команды **combine.**

Параметр *var* указывает, какой тип функций преобразовать, например, trig - для тригонометрических, power - для степенных. Пример:

```
> combine(4*sin(x)^3, trig);
```

-sin(3x)+3sin(x)

Для упрощения выражений, содержащих не только квадратные корни, но и корни других степеней, лучше использовать команду **radnormal(ex).** Пример:

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

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

С помощью команды **convert(exp, param),** где *exp* - выражение, которое будет преобразовано в указанный тип *param*. В частности, можно преобразовать выражение, содержащее sinx и cosx, в выражение, содержащее только tgx, если указать в качестве параметра tan, или, наоборот, tgx, ctgx можно перевести в sinx и cosx, если в параметрах указать sincos.

Вообще, команда **convert** имеет более широкое назначение. Она осуществляет преобразование выражения одного типа в другой. Например: **convert(list, vector)** - преобразование некоторого списка *list* в вектор с

теми же элементами; **convert(expr, string)** - преобразование математического выражения в его текстовую запись.

Подстановка

Для замены переменной OLD выражением NEW в объекте EX (выражение, список, множество, ...) применяется команда

subs(OLD=NEW, EX)

В одной команде разрешается использовать несколько выражений вида $OLD = NEW$, и этот набор подстановок может быть оформлен в виде множества.

Подстановка не изменяет исходный объект EX, а позволяет получить новое выражение. При нескольких подстановках замещение производится слева направо. Например:

```
> subs(x=y,z=x+y,z+x);subs(z=x+y,x=y,z+x);
```

x+2y

3y

Вместо последовательной подстановки можно использовать операцию одновременного замещения. Для этого подставляемые тождества нужно взять в фигурные скобки - превратить в множество. Например, после выполнения подстановки **subs(x=y,y=x,[x,y])** исходный список $[x, y]$ будет преобразован в $[x, x]$, тогда как при использовании команды **subs({x=y,y=x},[x,y])** переменные x и y в списке поменяются местами: список будет иметь вид $[y, x]$.

Команда **subs** не позволяет проводить замену в выражении производной, не действует под знаком многократного интеграла, в выражении предела и некоторых других случаях; для таких подстановок нужно применять команду **dchange** из пакета **PDEtools**.

3 Математический анализ

Предварительные сведения

В **Maple** для некоторых математических операций существует по две команды: одна прямого, а другая - отложенного исполнения, причем имена этих команд состоят из одинаковых букв. Команды прямого исполнения, как правило, начинаются с маленькой буквы и выполняются немедленно. Отложенные команды часто начинаются с большой буквы, обычно в том случае, когда существует команда - синоним прямого действия. После обращения к команде отложенного действия заданная математическая

операция (интеграл, производная, предел и т.д.) выводится в стандартном математическом виде и сразу не вычисляется. Для выполнения отложенной операции нужно использовать команду **value**. Перед использованием некоторых команд необходимо предварительно подключить соответствующую библиотеку командой **with**.

Напомним, что результат последней выполненной операции сохраняется в переменной %, предпоследней - в переменной %%, а ей предшествующей - %%%.

Часто необходимо наложить условия на переменные, например для задания области определения функции; в **Maple** это можно сделать, используя команду

assume(expr,prop)

Здесь *expr* - выражение, а *prop* - свойство. В качестве свойств могут выступать *real*, *continuos* и другие (см. справку **Maple**).

3.1 Суммы, ряды

Конечные и бесконечные суммы $\sum_{n=a}^{b} S(n)$ вычисляются командой прямого исполнения **sum** и отложенного исполнения **Sum**. Аргументы этих команд одинаковые: **sum(expr, n=a..b)**, где *expr* - выражение, зависящее от индекса суммирования, *a..b* - пределы индекса суммирования, указывающие, что суммировать следует от $n = a$ до $n = b$.

Напомним, что различие между этими двумя командами заключается в том, что в варианте, начинающемся с маленькой буквы, суммирование производится сразу, а для выполнения **Sum** нужно дополнительно дать команду **value**.

Если требуется вычислить сумму бесконечного ряда, то в качестве верхнего предела вводится *infinity*. Например:

```
> Sum(1/n!,n=1..infinity)=sum(1/n!,n=1..infinity);
```

$$\sum_{n=1}^{\infty} \frac{1}{n!} = e - 1$$

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

$$1.718281828 = 1.718281828$$

3.2 Дифференцирование и интегрирование

Для вычисления производных в Maple имеются две команды:

1) прямого исполнения - **diff(f,x)**, где f - функция, которую следует продифференцировать, x - имя переменной, по которой производится дифференцирование.

2) отложенного исполнения - **Diff(f,x)**, где параметры команды такие же, как и в предыдущей. Действие этой команды сводится к аналитической записи производной в виде $\partial f(x)/\partial x$.

Пример:

```
> Diff(sin(x^2),x)=diff(sin(x^2),x);
```

$$\frac{\partial}{\partial x}\sin(x^2) = 2\cos(x^2)x$$

Для вычисления производных старших порядков следует указать в параметрах $x\$n$, где n - порядок производной; например:

```
> Diff(cos(2*x)^2,x$4)=diff(cos(2*x)^2,x$4);
```

$$\frac{\partial^4}{\partial x^4}\cos(2x)^2 = -128\sin(2x)^2 + 128\cos(2x)^2$$

Для вычисления частных производных функций многих переменных используется следующий формат:

diff(expr,x1$n1,x2$n2,...)

Здесь *expr* - выражение, зависящее от переменных $x1$, $x2$..., а $n1$, $n2$,... - порядки дифференцирования по соответствующим переменным. Для команды **Diff** параметры аналогичны.

Пример с комбинированной последовательностью выдачи результата:

```
>u:=cos(x)*y^3;diff(u,x);diff(u,x$2,y$2);
```

$$u := \cos(x)y^3 \ -\sin(x)y^3 \ -6cos(x)y$$

Если правило дифференцирования функции пакету неизвестно или одна из переменных неявно зависит от другой, то в ответе будут сохраняться символы дифференцирования. Пример:

```
> alias(y=y(x)):diff(x^2+y^2=g(x),x);
```

$$2x + 2y(\frac{\partial}{\partial x}y) = \frac{d}{dx}g(x)$$

Операторное дифференцирование **D** применяется в тех случаях, когда ищется производная функции, а не выражение, и результатом действия оператора будет также функция. Например:

```
> f:=x-> ln(x^2)+exp(3*x): fd:=D(f);fd(1);evalf(%);
```

$$f := x- > ln(x^2) + exp(3x)$$

$$fd := x \to 2\frac{1}{x} + 3e^{(3x)}$$

$$2 + 3e^3$$

$$62.25661076$$

Неопределенный интеграл $\int f(x)dx$ вычисляется с помощью 2-х команд:

1) прямого исполнения - **int(f, x)**, где f - подынтегральная функция, x - переменная интегрирования;

2) отложенного исполнения - **Int(f, x)** - где параметры команды такие же, как и в команде прямого исполнения **int**. Команда **Int** выдает на экран интеграл в аналитическом виде математической формулы.

Для вычисления определенных интегралов используется команда **int(f,var=a..b)**

Пределы интегрирования в командах **int** и **Int** могут принимать значения бесконечности. Для вычисления двойных, тройных и т.д. интегралов нужно применить эту команду несколько раз. Поясним это примерами:

```
> Int(sinh(x)^4,x=0..ln(2))=int(sinh(x)^4,x=0..ln(2));
```

$$\int_0^{\ln(2)} \sinh(x)^4 dx = -\frac{225}{1024} + \frac{3}{8} ln(2)$$

```
> Int(Int(x+2*y,x=y^2-4..5),y=-3..3):%=value(%);
```

$$\int_{-3}^{3} \int_{y^2-4}^{5} x + 2\,y \, dx \, dy = \frac{252}{5}$$

Численное интегрирование выполняется командой **evalf(int(f, x=a..b),digits,flag)**

Здесь обязательными является подынтегральная функция f, зависящая только от переменной x и пределов интегрирования a и b, а необязательным - число значащих цифр **Digits**. Пример:

```
> evalf(Int(cos(x),x=0..3.14/4,5));
```

$$0.70683$$

3.3 Разложение функций в ряд

Команда **series(expr,eqn,n)** выполняет разложение выражения *expr* в степенной ряд; здесь *eqn* - выражение вида $x = a$, где x - переменная разложения, a - точка, в окрестности которой выписывается разложение, n - порядок, до которого строится разложение. Пример:

```
> p:=series((1+x)^(1/x),x=0,4);
```

$$e - \frac{1}{2}ex + \frac{11}{24}ex^2 + O(x^3)$$

Остаточная погрешность разложения задается членом вида $O(x^n)$. При точном разложении этот член отсутствует. В общем случае для его удаления можно использовать функцию **convert**. Например:

```
>convert(p,polynom);
```

$$e - \frac{1}{2}ex + \frac{11}{24}ex2\frac{7}{16}ex^3$$

Для разложения выражения expr, зависящего от переменной x, в ряд Тейлора в точке x=a до членов порядка n, можно воспользоваться командой

taylor(expr,x=a,n)

Отметим, что если последний параметр не указан, то порядок разложения определяется значением константы Order:

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

$$2 + (1 - 2ln(2))(x-1) + (-1 + ln(2) + ln(2)^2)(x-1)^2 + O((x-1)^3)$$

Выражения, зависящие от нескольких переменных, разлагаются в ряд Тейлора при помощи команды

mtaylor(expr,vars,k,w)

Здесь *expr* - разлагаемое в ряд выражение, *vars* - список пар <имя переменной> = <точка> для переменных разложения, k - порядок разложения, а w - вес. Например:

```
> q:=3*a*(x+1)^2+sin(a)*x^2*y-y^2*x+x- a;
```

$$q := 3a(x+1)^2 sin(a)x^2y - y^2x + x - a$$

```
> taylor(q,x=-1);
```

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

```
> mtaylor(q, [x,y]);
```

$$2a + (6a+1)x + 3ax^2 + sin(a)x^2y - y^2x$$

```
>mtaylor(q, [x=0,y=1]);
```

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

Если нужно получить коэффициент при члене порядка k, то надо воспользоваться командой

coeftayl(expr,vars,k)

Здесь *vars* - список имен переменных разложения, k - порядок члена разложения, при котором выписывается коэффициент. Для функции нескольких переменных параметр k является списком. Примеры:

```
> coeftayl(q, [x,y]=[0,0], [0,0]);
```

2a

```
> coeftayl(q, [x,y]=[0,0], [2,1]);
```

sin(a)

```
>coeftayl(q, x=-1, 2);
```

$$3a + \sin(a)y$$

Команда **sympt(f,x,n)** позволяет выписать асимптотическое разложение f по степеням переменной x порядка n, когда x стремится к бесконечности.

3.4 Интерполяция и аппроксимация

В пакете Maple имеется несколько команд, реализующих обычную и сплайн-интерполяцию, а также метод наименьших квадратов для приближения данных. В результате выполнения любой из этих команд формируется выражение, которое затем можно преобразовать в процедуру. Для оформления выражения в виде процедуры используется команда **unhappy** или команда из пакета **codegen**

fproc:=codegen[makeproc](f,x)

Здесь *fproc* - имя формируемой процедуры, x - независимая переменная.

Для построения интерполяционного многочлена относительно переменной *var* по таблице, заданной векторами X, Y используется команда **interp(X,Y,var)**. Массивы, задающие узлы интерполяции, могут быть не упорядочены, но массив X не должен содержать одинаковых элементов. Векторы X и Y должны содержать $n+1$ координат точек исходной зависимости, где n - степень интерполяционного полинома.

Покажем на примере технику применения полиномиальной аппроксимации на основе функции **interp** с построением аппроксимирующего полинома.

```
> X:=[0,1,2,3,4,5]:Y:=[0,1,4,3,2,1]:f:=interp(X,Y,x);
```

$$f := -\frac{7}{60}\,x^5 + \frac{19}{12}\,x^4 - \frac{91}{12}\,x^3 + \frac{173}{12}\,x^2 - \frac{73}{10}\,x$$

Построение сплайна с переменной *var* по таблице, заданной векторами X, Y, производится при помощи команды **spline(X,Y,var,d)**. Здесь параметр d определяет порядок сплайна, который может быть линейным (linear), квадратичным (quadratic), кубическим (cubic) и четвертой степени (quartic). По умолчанию строится кубический сплайн. Результатом действия команды будет построение сплайна в виде кусочно-гладкой функции (**piecewise**). Пример построения сплайнов четырех видов.

```
> readlib(spline):X:='X':Y:='Y':
> X:=[0,1,2,3,4,5]:Y:=[0,1,4,3,2,1]:
> f1:=spline(X,Y,x,linear);fc:=spline(X,Y,x,cubic):
```

```
> fq:=spline(X,Y,x,quadratic):fq1:=spline(X,Y,x,quartic):
```

$$f1 := \begin{cases} x & x < -1 \\ -2 + 3x & x < 2 \\ 6 - x & 0x < 3 \\ 6 - x & x < 4 \\ 6 - x & otherwise \end{cases}$$

```
> plot([f1,fc,fq,fq1],x=-1..6,y=-1..5,color=black);
```

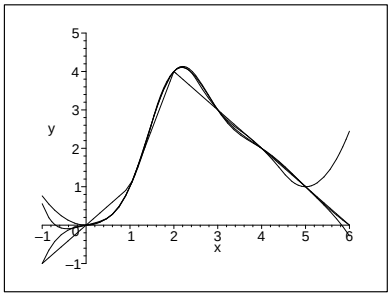


Рис 1. Графики полученных приближений.

Преобразуем полученное выражение для линейного сплайна в процедуру:

```
> f1proc:=codegen[makeproc](f1,x);
```

f1proc := proc(x)
piecewise(x<1,x,x<2,-2+3*x,x<3,6-x,x<4,6-x,6-x)
end proc

Другим способом приближения данных является метод наименьших квадратов. Результатом применения этого метода является функция заданного вида, наименее уклоняющаяся от исходных точек. Для применения этого метода в Maple можно воспользоваться командами линейной алгебры или использовать команду **leastsquare** из пакета stats. Входными параметрами команды **leastsquare** являются имена переменных, вид функции и набор точек. Построим методом наименьших квадратов приближение кубическим полиномом тестового набора данных X, Y, подключив предварительно пакет статистики:

```
> with(stats): fqproc:=codegen[makeproc](f1,x):
```

$> h := fit[leastsquare[[x, y], y = a1 * x^3 + a2 * x^2 + a3 * x + a4]]([X, Y]);$

$$h := y = \frac{1}{54}\,x^3 - \frac{38}{63}\,x^2 + \frac{1049}{378}\,x - \frac{17}{63}$$

4 Решение уравнений в Maple

4.1 Решение алгебраических уравнений

Команда **solve** является многоцелевой и применяется для аналитического решения алгебраических уравнений и их систем, неравенств и функциональных уравнений, вычисления тождеств. Команда **fsolve** предназначена для получения численных решений, а для работы с разностными уравнениями имеется специальная команда **rsolve**.

Вызов команды **solve** имеет следующий вид:

solve(eqn,var)

Здесь *eqn* - уравнение или система уравнений, *var* - переменная или группа переменных. Если параметр *var* отсутствует, то будут найдены решения относительно всех неизвестных, входящих в уравнение *eqn*. Под неизвестными понимаются все символьные переменные. Система уравнений и группа переменных задаются в виде множеств.

Уравнения могут быть заданы непосредственно в теле команды, а могут быть присвоены некоторой переменной. Если в качестве уравнения указано выражение без знака равенства, то считается, что это одна часть равенства, а другая равна нулю.

```
> sol:=solve(x^2-5*x+6,x);
```

$$\text{sol} := 3, 2$$

Для хранения решений удобно ввести переменную и обращаться к конкретному решению по индексу, например:

```
> sol[1]+sol[2];
```

$$5$$

Отметим, что результатом решения одного уравнения будет переменная типа *exprseq* (последовательность выражений). Если в качестве параметра указана переменная типа *set* (множество), то решение будет представлено в виде равенств, где в левой части стоит имя переменной, а справа - значение:

```
> s:=solve({x^2-5*x+6},x);
```

$$s := \{x = 3\}, \{x = 2\}$$

Ответ в виде множества удобен для подстановки нужного решения в выражения, зависящие от той переменной, относительно которой разыскивается решение:

```
> subs(s[-2],x^2);
```

$$9$$

Чтобы присвоить найденные значения переменным на весь сеанс, применяется команда **assign**. Для отмены назначается команда **unassing** или операция присваивания переменной ее имени, взятого в кавычки:

```
> sl:=solve({x^2-1},{x});
```

sl:= {x = 1}, {x =-1}

```
> assign(sl):x;
```

-1

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

x

Для нелинейных уравнений может быть несколько решений (но не обязательно все), а может оказаться, что ни одного решения не найдено. В последнем случае **Maple** просто выведет приглашение ввода, ожидая новой команды.

Если в выражении ответа появилась функция **RootOf**, это означает, что система либо не может выразить корни в радикалах, либо это требует дополнительных усилий:

```
> restart:sx:=solve(x^4+3*x^3-8*x+3,{x});
 sx:={x = RootOf(_Z^4+3*_Z^3-8*_Z+3,index = 1)},
     {x = RootOf(_Z^4+3*_Z^3-8*_Z+3,index = 2)},
     {x = RootOf(_Z^4+3*_Z^3-8*_Z+3,index = 3)},
     {x = RootOf(_Z^4+3*_Z^3-8*_Z+3,index = 4)};
```

Само решение при этом выражается через корни аргумента функции, стоящей внутри **RootOf**.

Команда **allvalues** позволяет представить решение, используя радикалы. Например, для первого решения **sx** получим:

```
> allvalues(sx[1]);
```

$$\{x = -\frac{3}{4} + \frac{\sqrt{37}}{4} - \frac{\sqrt{-10 + 2\sqrt{37}}}{4}\}$$

К аналогичному результату приводит использование команды **convert** с параметром *radical*:

```
> convert(sx[3],radical);
```

$$\{x = -\frac{3}{4} - \frac{\sqrt{37}}{4} + \frac{\sqrt{-10 - 2\sqrt{37}}}{4}\}$$

Команда **solve**, примененная для решения тригонометрического уравнения, выдает только главные решения, то есть решения в интервале $[0,2\pi]$. Для того, чтобы получить все решения, следует предварительно ввести дополнительную команду **_EnvAllSolutions:=true**. Например:

```
> _EnvAllSolutions:=true:
> solve(sin(x)=cos(x),x);
```

$$\frac{1}{4}\pi + \pi\_Z \sim$$

В Maple символ $\_Z \sim$ обозначает константу целого типа, поэтому решение данного уравнения в привычной форме имеет вид $x := \pi/4 + \pi n$, где n - целые числа.

При решении трансцендентных уравнений для получения решения в явном виде перед командой **solve** следует ввести дополнительную команду **_EnvExplicit:=true**. Пример решения сложной системы трансцендентных уравнений и упрощения вида решений:

```
> eq:={ 7*3^x-3*2^(z+y-x+2)=15, 2*3^(x+1)+
 3*2^(z+y-x)=66, ln(x+y+z)-3*ln(x)-ln(y*z)=-ln(4) }:
> _EnvExplicit:=true:
> s:=solve(eq,{x,y,z}):
> simplify(s[1]);simplify(s[2]);
```

$$x = 2, y = 3, z = 1, x = 2, y = 1, z = 3$$

4.2 Обыкновенные дифференциальные уравнения

Для получения аналитических, приближенных и численных решений дифференциальных уравнений применяется команда **dsolve**, причем во всех случаях используется единый формат команды:

dsolve(eq,var,opt)

Здесь *eq* - дифференциальное уравнение или система дифференциальных уравнений относительно неизвестных функций *var*. Для решения задачи Коши в уравнения *eq* нужно включить начальные условия, а для краевой задачи - соответственно краевые условия. Дополнительные условия *opt* позволяют указать способ решения ($type = ...$) и используемый метод $method = ...$ Например, для получения численного решения в качестве *opt* задается $type = numeric$, степенные разложения будут строится при $type = series$, а метод Рунге - Кутты - Фельберга применяется для численного интегрирования, если введена строка $method = rkf45$. При определении способа решения левая часть "$type =$"может быть опущена.

В уравнениях для указания производной применяется команда **diff** и оператор **D**, а для обозначения производной в начальных и краевых условиях используется оператор **D**. Система оформляется в виде множества: уравнения, начальные и краевые условия записываются через запятые и берутся в фигурные скобки.

По умолчанию (дополнительные условия *opt* не указаны), так как принято, что $type = exact$, **Maple** старается найти явное выражение для неизвестной функции (функций). При невозможности выделить искомую функцию (например, решение выражается через дробные степени) решение выводится в неявном виде. Если требуется решение в явном виде, то

необходимо указать дополнительное условие $type = explicit$. Константа - параметр $\_EnvExplicit$ определяет способ представления решения.

Если число краевых или начальных условий меньше порядка системы, то в ответе будут фигурировать неопределенные константы $\_C1$, $\_C2$ и т.д.

```
> de:=diff(y(x),x$2)+4*diff(y(x),x)+3*y(x)+2;
```

$$de := (\frac{d^2}{dx^2}y(x)) + 4(\frac{d}{dx}y(x)) + 3y(x) + 2$$

```
>dsolve(de,y(x));
```

$$\mathrm{y(x)} = \exp(\text{-x})\ \_\mathrm{C2} + \exp(\text{-3 x})\_\mathrm{C1} - 2/3$$

То же уравнение при помощи оператора **D** запишется в следующем виде:

```
> d:=(D@@2)(y)(x)+4*D(y)(x)+3*y(x)+2;
```

$$\mathrm{d}:=(D^{(2)})(\mathrm{y})(\mathrm{x}) + 4\ \mathrm{D(y)(x)} + 3\ \mathrm{y(x)} + 2$$

```
> dsolve(d,y(x));
```

$$y(x) = e^{(-x)}\_C2 + e^{(-3x)}\_C1 - \frac{2}{3}$$

Для решения задачи Коши определим начальные условия и обратимся к команде **dsolve**:

```
> ic:=D(y)(0)=0,y(0)=-1;dsolve({d,ic},y(x));
```

$$\mathrm{ic} := \mathrm{D(y)(0)} = 0,\ \mathrm{y(0)} = \text{-1}$$

$$y(x) = \frac{1}{6}e^{(-3x)} - \frac{1}{2}e^{(-x)} - \frac{2}{3}$$

Для решения краевой задачи зададим краевые условия в виде множества и обратимся к команде **dsolve**:

```
> bvp:={D(y)(0)=0,y(1)=-1};dsolve({de}union bvp,y(x));
```

$$\mathrm{bvp} := \{\mathrm{D(y)(0)} = 0,\ \mathrm{y(1)} = \text{-1}\}$$

$$y(x) = -\frac{e^{(-x)}}{3e^{(-1)} - e^{(-3)}} + \frac{1}{3}\frac{e^{(-3x)}}{e^{(-1)} - e^{(-3)}} - \frac{2}{3}$$

Для проверки того, что найденное решение SOL удовлетворяет уравнению или системе ODE, имеется функция

odetest(SOL,ODE)

Проверим решение краевой задачи

```
> odetest(%,de);
```

$$0$$

Команда **dsolve** выдает решение дифференциального уравнения в невычисляемом формате. Для того, чтобы с решением можно было бы работать далее (например, построить график решения) следует отделить правую часть полученного решения командой **rhs(%)**.

5 Двумерная графика в Maple

Для построения графиков функции $f(x)$ одной переменной (в интервале по оси Ox и в интервале по оси Oy) используется команда **plot(f(x), x=a..b, y=c..d, parameters)**, где *parameters* - параметры управления изображением. Если их не указывать, то будут использованы установки по умолчанию. Настройка изображения также может осуществляться с панели инструментов.

Некоторые основные параметры команды **plot**:

1) $title = "text"$, где *text* - заголовок рисунка.

2) *axes* - установка типа координатных осей: axes=NORMAL - обычные оси; axes=BOXED - график в рамке со шкалой; axes=FRAME - оси с центром в левом нижнем углу рисунка; axes=NONE - без осей.

3)$style = LINE(POINT)$ - вывод линиями (или точками).

4) $numpoints = n$ - число вычисляемых точек графика (по умолчанию n=49).

5) *olor* - установка цвета линии; например, yellow - желтый.

6) $thickness = n$, где n=1,2,3... - толщина линии (по умолчанию n=1).

7) $linestyle = n$ - тип линии: непрерывная, пунктирная и т.д. (n=1 - непрерывная, установлено по умолчанию).

8) $symbol = s$ - тип символа, которым помечают точки: BOX, CROSS, CIRCLE, POINT, DIAMOND.

9) $font = [f, style, size]$ - установка типа шрифта для вывода текста: f задает название шрифтов: TIMES, COURIER, HELVETICA, SYMBOL; *style* задает стиль шрифта: BOLD, ITALIC, UNDERLINE; *size* - размер шрифта в pt.

10) $labels = [tx, ty]$ - надписи по осям координат: tx - по оси Ox и ty - по оси Oy.

В пакете **plots** имеется команда **textplot** для вывода текстовых комментариев на рисунок: **textplot([xo,yo,'text'], options)**, где xo, yo - координаты точки, с которой начинается вывод текста 'text'.

Часто бывает необходимо совместить на одном рисунке несколько графических объектов, полученных при помощи различных команд, например, добавить графику, нарисованному командой **plot**, текстовые надписи, полученные командой **textplot**. Для этого результат действия команды присваивается некоторой переменной:

```
> p:=plot(...): t:=textplot(...):
```

При этом на экран вывод не производится. Для вывода графических изображений необходимо выполнить команду из пакета **plots**:

```
> with(plots): display([p,t], options).
```

6 Типичные приемы программирования

Пользователь может воспользоваться набором команд и инструкций, аналогичный существующим в известных языках программирования (например, в языках Паскаль и С), чтобы писать собственные программы.

6.1 Условные операторы

Условный оператор начинается с зарезервированного слова **if** и обязательно должен заканчиваться словом **fi** или **end if,** которые являются синонимами. Этот оператор имеет несколько форм написания, которые отличаются детальностью проверки условий:

if bool then expr1 fi;

if bool then expr1 else expr2 end if;

Эта конструкция дает возможность в зависимости от значения логического условия **bool** выполнять выражение $expr1$ (когда $bool = true$) или $expr2$ (если $bool = false$ в случае ветки **else**). В качестве выражений $expr1$ и $expr2$ может выступать любая последовательность **Maple**-команд. Пример:

```
>x:=2:if x<0 then x:=5 else x:=-5:fi;
```

x := -5

Для реализации сложных условий можно использовать полный вариант условного оператора, который имеет следующий вид:

if bool1 then expr1

elif bool2 then expr2

...

elif boolN then exprN

else expr0 end if;

Таким образом, вложенность условий может быть практически неограниченной и реализуется при помощи вставки **else**. В качестве выражений $expr0$, $expr1$,..., $exprN$ могут выступать любые последовательности **Maple**-команд. Как и для простого условного оператора, у этой конструкции может отсутствовать ветка **else**. Приведем пример полного условного оператора:

```
> x:=7: if x<0 then x:=a;elif x=0 then x:=b;
                        elif x<10 then x:=c;
                        else x:=d:fi;
```

x := c

Существует вариант условного оператора, который предназначен для использования в выражениях. Его синтаксис имеет вид:

"if"(bool,expr1,expr2)

При обращении к этому оператору сначала проверяется логическое выражение **bool**, затем при его истинности (**true**) выполняется оператор **expr1**, а при ложности (**false**) - **expr2**. Например:

```
> a:=-5:b:=4:`if`(a>b,a,b);
```

4

6.2 Операторы цикла

В Maple имеется несколько операторов цикла, и для их записи используются служебные слова **for, from, by, to, while, do, od** и **end do**. Телом всех операторов цикла является последовательность команд, заключенных между **do** и **end do**. Рассмотрим цикл перечисления:

for VAR from VAL1 by VAL2 to VAL3 do EXPR end do;

Тело цикла **EXPR** выполняется при каждом значении параметра цикла **VAR**, который изменяется от **VAL1** с шагом **VAL2** до тех пор, пока не станет больше **VAL3**. Если шаг изменения **VAL2** равен единице, то оператор цикла допускает сокращенную форму:

for VAR from VAL1 to VAL3 do EXPR od;

А если и начальное значение **VAL1** равно единице, то возможен еще более короткий вариант:

for VAR to VAL3 do EXPR od;

Пример:

```
> for i from 0 by 4 to 8 do i:od;
```

0
4
8

Оператор цикла типа "пока"имеет вид:

while BOOL do EXPR od;

Тело цикла **EXPR**выполняется, пока значение логического выражения **Bool** истинно (**true**), и выполнение прекращается, если **BOOL** ложно (**false**). Пример:

```
> j:=0:while j<5 do j:=(j+1)^j:od;
```

j := 1
j:=2
j:=9

Третий оператор цикла является некоторым симбиозом двух предыдущих:

for VAR from VAL1 by VAL2
while BOOL do EXPR end do;

Тело цикла **EXPR** выполняется, пока логическое выражение **BOOL** является истинным,а переменная (**VAR**) изменяется от значения **VAL1** с с шагом **VAL2**).Отметим, что **Maple** вначале проверяет условие цикла, а затем выполняет выражение **EXPR**. Пример:

```
> for i from 1 by 2 while i<6 do i^2: od;
```

$$1$$
$$9$$
$$25$$

Четвертый оператор цикла ориентирован на работу с символьными выражениями и имеет следующую форму:

for VAR in EXPR1 do EXPR2 od;

Тело цикла **EXPR2** выполняется, когда символьная переменная **VAR** последовательно принимает значение каждого из операндов алгебраического выражения или списка **EXPR1**. Работа этой конструкции зависит от внутреннего представления выражения **EXPR1**. Если **EXPR1** - сумма, то переменная **VAR** принимает поочередно значения каждого слагаемого, если произведение - то каждого сомножителя и т.д. Пример:

```
> f:=x^2+3*x+1/x;g:=simplify(f);
```

$$f := x^2 + 3x + 1/x$$
$$\frac{x^3 + 3x^2 + 1}{x}$$

```
> for s in f do s: od;
```

$$x^2$$
$$3\ \mathrm{x}$$
$$1/\mathrm{x}$$

```
> for s in g do s:od;
```

$$x^3 + 3x^2 + 1$$
$$\frac{1}{x}$$

Команды **break** и **next** предназначены для управления операторами цикла. По команде **break** осуществляется немедленный выход из цикла, а по команде **next** - переход к следующему шагу. Примеры:

```
> i:=0:do i:=i+1:if i=3 then break fi end do;
```

$$\mathrm{i} := 1$$
$$\mathrm{i} := 2$$
$$\mathrm{i} := 3$$

```
> L:=[1,2,5,100]:for i in L do
> if i=5 then next end if:i^2:od;
```

$$1$$
$$4$$
$$10000$$

Иногда бывает полезным бесконечный цикл, который можно реализовать, опустив все управляющие циклами параметры:

do EXPR2 end do;

Этот цикл будет работать до тех пор, пока внутри него не встретится ошибка или не будет осуществлен выход с помощью команд **break** или **return**.

6.3 Процедуры-функции

Процедуры-функции можно задавать несколькими способами. Первый напоминает задание отображения - некоторое число входных параметров при помощи знака → формирует результирующее выражение **expr**:

NAME:=(VAR1, VAR,...)->expr

Здесь **NAME** - имя функции, **VAR1, VAR2,...** - имена формальных параметров, а **EXPR** - выражение, реализующее тело функции. Типы формальных параметров и результата работы функции могут быть любыми.

```
> f:=(x,y)->simplify(x^2+y^2); f(sin(x),cos(x));
```

$$\text{f:=(x,y)->simplify}(x^2 + y^2)$$

$$1$$

Второй способ задания при тех же входных параметрах использует команду **unapply**:

NAME:=unapply(EXPR, VAR1, VAR2,...)

Здесь **VAR1, VAR2,...** - переменные, а **EXPR** - выражение или операция. Эта команда полезна при определении новой функции через известную или когда вычисленное выражение предполагается использовать как функцию. Например:

```
>f:=unapply(diff(z(x)^2,x),z);f(sin^2);
```

$$f := z- > 2z(x)(\frac{d}{dx}z(x))$$

$$4sin(x)^3cos(x)$$

6.4 Процедуры

Всякая процедура в **Maple** начинается с заголовка, который состоит из имени процедуры, за которым следует знак присваивания и служебное слово **proc**, затем в круглых скобках через запятую указываются формальные параметры. Процедура должна обязательно заканчиваться оператором **end proc**. Все команды и выражения, стоящие после заголовка

и до оператора **end**, составляют тело процедуры. Простейшая процедура имеет вид:

(NAME:=proc(VAR1, VAL2,...);
EXPR1; EXPR2;...
end proc;

Здесь **(NAME** - имя процедуры, **VAR1, VAL2,...**- имена формальных параметров, а **EXPR1**, **EXPR2**, ... - выражения, реализующие тело процедуры.

Текст процедуры должен быть набран в одной группе. По нажатию клавиши **Enter** происходит синтаксический анализ текста и в случае ошибки выводится сообщение о ней. После того как процедура загружена в память, к ней можно обращаться по имени. Возвращаемым значением по умолчанию является результат последнего оператора из тела процедуры, однако существуют и другие возможности, о которым будет сказано ниже. Тип результата работы процедуры зависит от типа возвращаемого значения.

Формальный параметр процедуры можно явно описать, указав его тип после двух двоеточий, следующих за именем параметра. В этом случае при обращении к процедуре **Maple** проверит тип фактического параметра и выведет сообщение об ошибке в случае несовпадения с типом формального параметра. Пример:

>f:=proc(i::integer) i^2 end proc:f(4);f(-3);

16

9

Еще одним способом передачи данных из процедуры является изменение значения входных параметров. При помощи описания **evaln**можно присваивать переменным, выступающим в качестве входных параметров, значения внутри процедуры. Отметим, что присваивать фактическим параметрам возвращаемые значения следует только перед выходом из процедуры, что связано со спецификой обработки выражений внутри процедур. Пример:

>f:=proc(x::evaln) x:=10;x end proc:f(a);a;

a

10

После заголовка процедуры может следовать описательная часть процедуры, отделяющаяся от него пробелом. В этой части описываются локальные и глобальные переменные, используемые процедурой.

Для определения локальных переменных применяется описатель **local**. Перечислить глобальные переменные можно при помощи описателя **global**. Неописанные переменные **Maple** автоматически описывает как локальные. Пример:

```
   >restart:i;j;k;l:=4;a:=proc() local i:global j;
                    i:=1;j:=2-l;k:=3; end proc:
```

$$i$$
$$j$$
$$k$$
$$l := 4$$

```
Warning, `k` is implicitly declared local to procedure `a`
>a();i;j;k;
```

$$i$$
$$-2$$
$$k$$

Для выхода из процедуры в любом месте ее тела используется команда **return(VAL)**. Здесь **VAL** - возвращаемое значение. Если в процедуре несколько операторов **return**, то возвращаемые значения могут быть даже различных типов. Например:

```
   >restart:prim:=proc(x::numeric) local y,w:global z;
 options remember;
 if x<0 then RETURN(-x):
 elif x=0 then ERROR(x=0"):fi;"
[x,x^2,x^3]
end proc:
>prim(-1);prim(0);prim(5);
```

$$1$$

```
Error, (in prim) x=0
```

$$[5,\ 25,\ 125]$$

Литература

1. Говорухин В.Н., Цибулин В.Г. Компьютер в математическом исследовании. Учебный курс. - СПб.: Питер, 2001.

2. Дьяконов В.П. Математическая система Maple V R3/R4/R5. М.: Солон, 1998.

3. Дьяконов В.П. Maple 6. -СПб.: Питер, 2001.

4. Савотченко С.Е., Кузьмичева Т.Г. Методы решения математических задач в Maple. -Белгород: БГУ, 2001.

5. Тарасевич Ю.Ю. Информационные технологии в математике. М.: Солон-Пресс, 2003.

Учебное издание

Лилия Александровна Молчанова

ВВЕДЕНИЕ В MAPLE

Методические указания для студентов
математических специальностей

В авторской редакции
Технический редактор Л.М. Гурова
Компьютерный набор и верстка автора

Подписано в печать xx.xx.06
Формат 60 × 84 1/16. Усл. печ. л. x,x. Уч.-изд. л. x,x.
Тираж 25 экз.

Издательство Дальневосточного университета
690950, Владивосток, ул. Октябрьская, 27.
Отпечатано в лаборатории
кафедры компьютерных наук ИМКН ДВГУ
690950, Владивосток, ул. Октябрьская, 27, к. 132.