Е.М. Артемова, И.А. Бизяев

# СИСТЕМА КОМПЬЮТЕРНОЙ МАТЕМАТИКИ MAPLE: ТЕОРИЯ И РЕШЕНИЕ ЗАДАЧ

## Часть 1

Министерство науки и высшего образования Российской Федерации
ФГБОУ ВО «Удмуртский государственный университет»
Институт математики, информационных технологий и физики
Кафедра теоретической и экспериментальной физики

Е.М. Артемова, И.А. Бизяев

# СИСТЕМА КОМПЬЮТЕРНОЙ МАТЕМАТИКИ MAPLE: ТЕОРИЯ И РЕШЕНИЕ ЗАДАЧ

Часть 1

Учебно-методическое пособие

Ижевск
2023

УДК 519.6:004(075.5)
ББК 22.195.3я73
А861

*Рекомендовано к изданию Учебно-методическим советом УдГУ*

*Пособие подготовлено в Уральском математическом центре (соглашение № 075-02-2023-933).*

**Рецензент:** канд. физ.-мат. наук, ст. науч. сотрудник ИМ УдмФИЦ УрО РАН М.Р. Королева

**Артемова Е.М., Бизяев И.А.**

А861 Система компьютерной математики Maple: теория и решение задач : учеб.-метод. пособие в 2 ч. : Ч. 1. – Ижевск : Удмуртский университет, 2023. – 99 с.

Пособие представляет собой практическое руководство по изучению возможностей пакета аналитических вычислений Maple. Подробные теоретические сведения чередуются с практическими заданиями. Последовательное изучение тем и выполнение заданий позволит постепенно освоить основные команды математической системы Maple.

Пособие предназначено для студентов, аспирантов, преподавателей физико-математических направлений университетов, а также для широкого круга исследователей при первоначальном ознакомлении с пакетом. Кроме того может использоваться как справочник по основным командам Maple.

УДК 519.6:004(075.5)
ББК 22.195.3я73

# Содержание

# Предисловие

Содержание первой части данного учебно-методического пособия составляют разделы, изучаемые студентами Удмуртского государственного университета (УдГУ) при освоении дисциплины «Аналитические методы вычислений». Приведенные в данном учебно-методическом пособии материалы могут быть полезны при изучении дисциплин, требующих дополнительных компьютерных вычислений, например, «Уравнения тепломассопереноса», «Гидродинамика», «Нелинейные уравнения математической физики», «Вычислительная математика в физических задачах». При написании преследовалась цель описать возможности пакета Maple, по возможности, в наиболее простой и ясной форме.

Материал первой части данного пособия разделен на четыре основные главы. Первая посвящена началу работы с пакетом Maple, описаны интерфейс программы и особенности синтаксиса. Во второй и третьей главах изложены основы командного языка Maple и базовые математические команды, часто встречающиеся при решении физических задач. В этих разделах для каждой команды приводится пример ее использования. Четвертая глава посвящена разбору графических возможностей пакета Maple. Рассмотрены команды для построения двумерной и трехмерной графики, а также команды для создания анимации и обработки экспериментальных данных (с графическими иллюстрациями). Кроме краткого изложения теоретического материала, в данном пособии представлены примеры решения типовых задач по каждому разделу, а также перечень задач для самостоятельного решения. Каждый раздел сопровождается наглядными примерами, демонстрирующими использование компьютерной программы для аналитических и численных вычислений Maple.

Пособие предназначено для студентов бакалавриата, магистратуры, аспирантов, преподавателей физико-математических направлений университетов, а также для неспециалистов в работе с пакетом Maple для первоначального ознакомления.

Отдельные разделы пособия будут полезны научным сотрудникам в области математики, химии, биохимии и т. д.

Для изучения некоторых разделов данного пособия читателю будет полезно ознакомиться с содержанием ранее изданных учебно-методических пособий [1, 2], в которых изложены основы синтаксиса других языков программирования. А также с пособиями [3, 4, 5], в которых приведены фрагменты Maple-кода для решения задач математической физики и анализа динамических систем.

Необходимость издания данного пособия обусловлена отсутствием современных методических пособий по данной тематике в библиотечном фонде УдГУ. В основу пособия положены лекции, читаемые авторами в Институте математики, информационных технологий и физики в УдГУ для студентов направлений «Физика» и «Прикладные математика и физика». Содержание данного пособия соответствует учебной программе дисциплины «Аналитические методы вычислений», связанной с изучением пакета Maple и его использованием для решения задач.

В пособии учтены требования Федерального государственного образовательного стандарта к содержанию курса для направлений подготовки бакалавров «Физика» и «Прикладные математика и физика». Методическое пособие рассчитано на средний уровень физико-математической подготовки студентов и необходимо для освоения следующих компетенций:

– способность использовать в профессиональной деятельности базовые знания фундаментальных разделов математики, создавать математические модели типовых профессиональных задач и интерпретировать полученные результаты с учетом границ применимости моделей;

– способность использовать базовые теоретические знания для решения профессиональных задач;

– способность применять на практике базовые общепрофессиональные знания и умения теории и методов физических исследований (в соответствии с профилем подготовки).

# 1. Начало работы с пакетом Maple

**Maple** — программный пакет, система компьютерной математики. Система Maple предназначена для символьных вычислений, хотя имеет ряд средств и для численного решения дифференциальных уравнений и нахождения интегралов. Обладает развитыми графическими средствами. Имеет собственный интерпретируемый язык программирования.

В рамках данного методического пособия будет рассматриваться работа с пакетом Maple версии 2019 года[1]. При запуске программы на экране появляется стартовая страница (см. рис. 1).

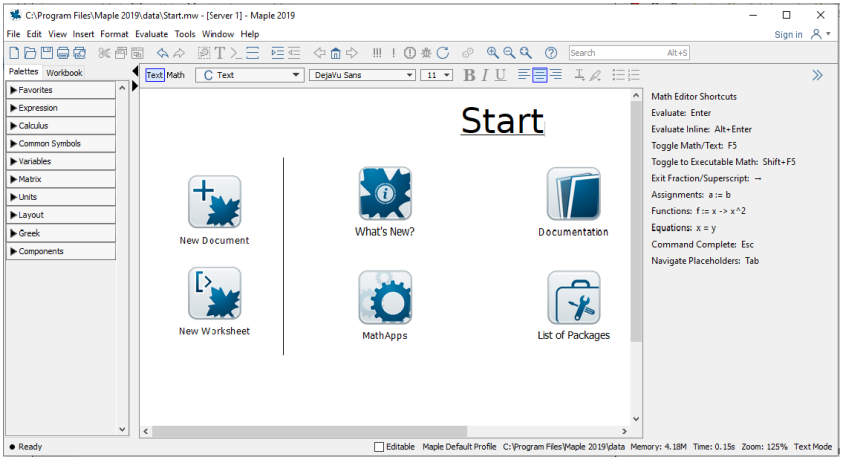



Рис. 1. Стартовая страница Maple 2019

Для начала работы с пакетом Maple необходимо создать **рабочий лист** или новый документ. Для этого предлагается либо выбрать вкладку «New Document» (см. рис. 2a), либо команду «Create a new file» на основной панели инструментов (см. рис. 2b), либо через главное меню «File→New→Document Mode»

[1] Ознакомиться с последней версией пакета Maple можно посылке https://www.maplesoft.com/products/Maple/. Пробная версия дается на 15 дней.

(или быстрая комбинация клавиш «Ctrl+N») (см. рис. 2c). В результате появляется рабочее окно, интерфейс которого приведен на рис. 3.

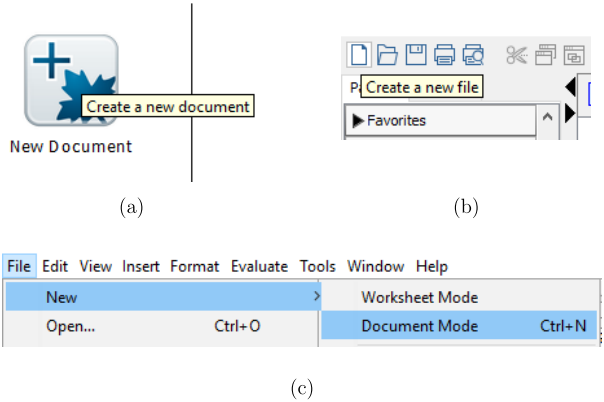



Рис. 2. Варианты создания рабочего листа

Элементы интерфейса рабочего листа подписаны на рис. 3. Пункты **главного меню**:

1) **File** (Файл) – содержит стандартный набор команд для работы с файлами, например: сохранить файл, открыть файл, создать новый файл и т. д.

2) **Edit** (Правка) – содержит стандартный набор команд для редактирования текста, например: копирование, перемещение выделенного текста в буфер обмена, отмена команды/действия и т. д.

3) **View** (Вид) — содержит стандартный набор команд, управляющих структурой окна Maple.

4) **Insert** (Вставка) – служит для вставки полей разных типов: математических или текстовых строк, графиков.

5) **Format** (Формат) – содержит команды оформления документа, например: установка типа, размера и стиля шрифта.

6) **Evaluate** (Вычисление) – служит для выполнения/вычисления команд.

7) **Tools** (Инструменты) – служит для установки различных параметров ввода и вывода информации на экран, принтер, например таких, как качество печати.

8) **Windows** (Окно) – служит для перехода из одного рабочего листа в другой.

9) **Help** (Справка) – содержит подробную справочную информацию о Maple.

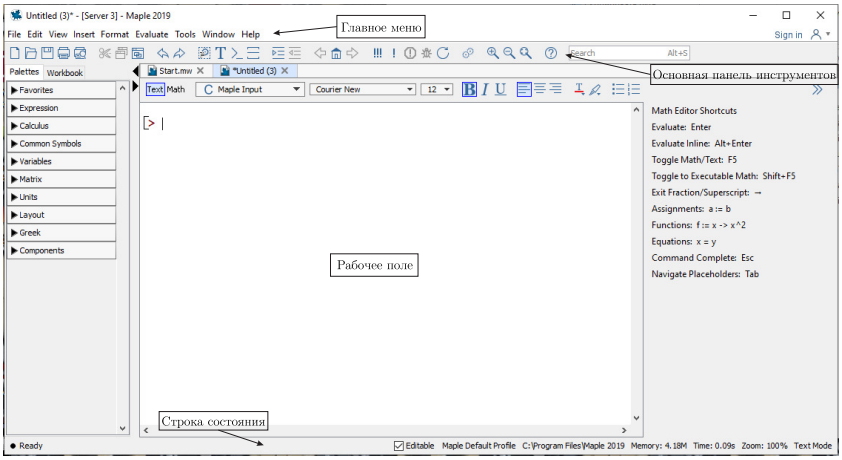



Рис. 3. Новый рабочий лист

Ниже главного меню расположена **основная панель инструментов** с рядом кнопок, дублирующих наиболее часто используемые команды главного меню. Большую часть окна занимает **рабочее поле**, в котором располагаются рабочие листы, где вводятся команды и отображаются результаты их выполнения. В нижней части окна расположена **строка состояния**, в которой отображаются некоторые параметры работы системы Maple.

Рабочий лист состоит из **областей ввода** и **областей вывода**. Области ввода начинаются с символа [>, обозначающего приглашение ко вводу команд[2], а в областях вывода отображаются результаты выполнения команд и операторов, а также двумерная и трехмерная графика. Команды в области ввода стандартно отображаются красными прямыми буквами, результаты этих комманд отображаются в области вывода синими курсивными буквами, если вывод не отключен (см. рис. 4). Если команды в области ввода отображаются другим способом, например, черным курсивом, необходимо выполнить следующие инструкции:

1) В главном меню выбрать **Tools→Options...**.

2) В появившемся окне выбрать вкладку **Display**.

3) Во вкладке **Input display:** изменить «2-D Math Notation» на «Maple Notation».

4) Нажать на кнопку **Apply Globally**.

Содержимое областей ввода и вывода образуют отдельный блок - **группу вычислений**, которая на рабочем листе отмечается слева квадратной скобкой.

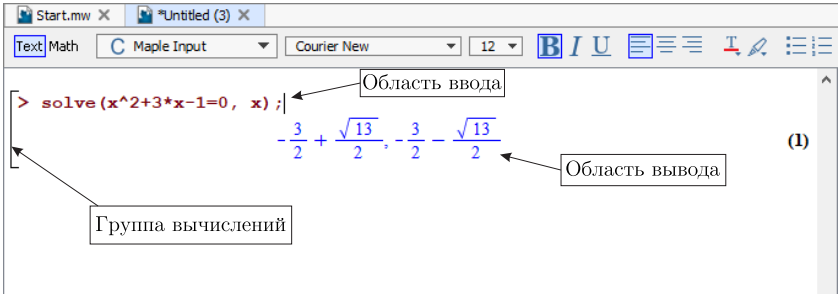



Рис. 4. Области ввода и вывода команд

[2]Если приглашение ко вводу не появилось автоматически, его можно добавить с помощью кнопки >_ на основной панели инструментов.

В начале каждого рабочего файла/листа желательно прописывать команду `restart`, которая очищает внутреннюю память Maple, то есть отменяет все выполненные ранее команды и освобождает использованные переменные.

Каждая команда, прописанная в поле ввода, оканчивается либо символом «`;`» (точка с запятой), либо «`:`» (двоеточие). В зависимости от проставленного знака будет осуществляться вывод в поле ввода по следующему правилу:

- если команда заканчивается точкой с запятой, то результат ее выполнения будет выводиться на экран;
- если команда заканчивается двоеточием, то ее результат не будет выведен на экран, но будет храниться во внутренней памяти Maple.

Если после команды не проставлен никакой знак, то результат ее выполнения по умолчанию будет выведен на экран.

# 2. Основы командного языка

## 2.1. Основные объекты

Как и в любой интерактивной системе, в Maple реализован свой язык, с помощью которого происходит общение пользователя с системой. Базовыми понятиями являются **объекты** и **переменные**.

**Объектами** в Maple могут быть числа, символьные выражения, уравнения, графики и другое. Простейшими объектами, с которыми может работать Maple, являются числа, константы и строки.

**1. Числа.** Числа могут быть целыми, обыкновенными дробями, радикалами и числами с плавающей точкой. Первые три типа чисел представляются точно, без округлений. Числа с плавающей точкой являются приближенными, в которых ограничено число значащих цифр.

- Целые числа:

```
> 2; -57;
```

$$2$$
$$-57$$

- Дроби:

```
> -18/4; 45/7;
```

$$-\frac{9}{2}$$
$$\frac{45}{7}$$

- Радикалы:

```
> 18^(1/2); (4/3)^(1/2);
```

$$\sqrt{18}$$
$$\frac{\sqrt{4}\sqrt{3}}{3}$$

- Числа с плавающей точкой:

```
> 18.4; -0.1;
```

$$18.4$$
$$-0.1$$

Поскольку целые числа, дроби и радикалы представляются точно, то и вычисления, связанными с ними, являются точными, так как при работе с этими типами чисел Maple не производит никаких округлений, в отличие от чисел с плавающей точкой. Таким образом, если в выражении все числа являются целыми, дробями или радикалами, то результат будет представлен также с использованием этих типов данных:

```
> 17^(1/2)*4/7;
```

$$\frac{4\sqrt{17}}{7}$$

Однако если в выражении присутствует число с плавающей точкой, то там, где это возможно, Maple будет производить вычисления точно, но некоторые подвыражения будут вычислены в форме с плавающей точкой:

```
> 17^(1/2)*4.0/7;
```

$$0.5714285714\sqrt{17}$$

**2. Константы.** Константы – это простейшие именованные объекты, несущие заранее предопределенные значения. Их имена (идентификаторы) также заранее определены и не могут меняться. В Maple существуют изменяемые и неизменяемые константы. Имена и значения неизменяемых констант указаны в таблице 1.

Заметим, что в Maple различаются малые и заглавные буквы. Поэтому вызов константы `Pi` возвращает число $\pi \approx 3.1415926540$, а вызов `pi` возвращает символ $\pi$, не содержащий в себе численного значения.

Таблица 1. Неизменяемые константы

| Константа | Значение |
|---|---|
| `true` | логическая константа, обозначающая истинность выражения |
| `false` | логическая константа, обозначающая ложность выражения |
| `gamma` | константа Эйлера $\gamma \approx 0.5772156649$ |
| `Pi` | число $\pi$ |
| `I` | мнимая единица $i = \sqrt{-1}$ |
| `infinity` | бесконечность $\infty$ |
| `D` | дифференциальный оператор |
| `O` | константа, определяющая порядок малости в разложениях |

Константы, значения которых могут быть переопределены, — это константы, задающие необходимые для работы системы настройки. К наиболее важным можно отнести две константы, позволяющие влиять на точность вычислений: `Digits` и `Order`. Первая константа задает число значащих цифр для операций с числами с плавающей точкой, вторая — определяет количество членов в разложении функции в ряд. По умолчанию `Digits = 10`, `Order = 6`.

**3. Строки.** Кроме чисел и констант Maple позволяет работать со строками — наборами символов, заключенными в обратные кавычки «‘»:

```
> ‘строка в Maple‘
```

*строка в Maple*

В некоторых версиях Maple для отображения строк используются двойные кавычки.

Под **переменной** в Maple понимается область в памяти, которой присвоенно некоторое имя и в которой хранится объект. Имя переменной — последовательность символов, обязательно начинающаяся с буквы или символа подчеркивания `_`. При этом

следует помнить, что Maple чувствителен к регистру, то есть следующие имена переменных являются разными

`Var`, `var`, `_Var`, `_var`

Кроме букв в именах переменных могут использоваться также цифры, но следует помнить, что имя переменной не может начинаться с цифры. В качестве имен запрещено использовать зарезервированные слова языка Maple, например, `for`, `end`, `and`, имена существующих команд Maple и т. д. Проверка совпадения с существующим зарезервированным словом:

```
> ?name
```

Здесь `name` — проверяемое имя переменной. Если имя переменной совпадает с каким-либо зарезервированным словом, то Maple выдаст соответствующую страницу в справочной системе.

Важной операцией в Maple является операция присваивания `:=`. Синтаксис этого оператора следующий:

```
> переменная := выражение;
```

В левой части указывается имя переменной, а в правой части — любое выражение, которое может быть числовым, символьным или просто другой переменной. Пример присвоения переменным числовых значений:

```
> a := 5: b := -4;
```

$$-4$$

```
> a;
```

$$5$$

При создании переменных в их названии часто используются греческие буквы. В Maple можно записать буквы греческого алфавита в полиграфическом виде. Для этого в командной строке набирается название греческой буквы. Например, буква $\alpha$ получится, если набрать `alpha`. Пример объявления переменной $\alpha$ символьным выражением:

```
> alpha := t^2 + 4*t - 3;
```

$$\alpha = t^2 + 4t - 3$$

Строчные греческие буквы и их названия в Maple приведены в таблице 2.

Таблица 2. Греческие буквы и их наименования

| Буква | Название | Буква | Название | Буква | Название |
|---|---|---|---|---|---|
| $\alpha$ | `alpha` | $\theta$ | `theta` | $\sigma$ | `sigma` |
| $\beta$ | `beta` | $\kappa$ | `kappa` | $\upsilon$ | `upsilon` |
| $\gamma$ | `gamma` | $\lambda$ | `lambda` | $\phi$ | `phi` |
| $\delta$ | `delta` | $\nu$ | `nu` | $\chi$ | `chi` |
| $\epsilon$ | `epsilon` | $\mu$ | `mu` | | `psi` |
| $\zeta$ | `zeta` | $\xi$ | `pi` | $\omega$ | `omega` |
| $\eta$ | `eta` | $\rho$ | `rho` | | |

Заглавные греческие буквы можно записать, если набирать название греческой буквы с заглавной буквы, например, чтобы получить $\Omega$, следует набрать `Omega`. Также некоторые специальные начертания греческих букв можно получить с помощью приставки `var`, например, $\varphi$ — `varphi`, $\varepsilon$ — `varepsilon`.

## 2.2. Математические операции и функции

Стандартными операциями над объектами и переменными в Maple являются **арифметические операции**. С их помощью составляются математические выражения, работу с которыми рассмотрим далее. Символы арифметических операций приведены в таблице 3.

Таблица 3. Арифметические операции

| Символ | Операция |
|---|---|
| `+` | сложение |
| `-` | вычитание |
| `*` | умножение |
| `/` | деление |
| `^` или `**` | возведение в степень |
| `!` | вычисление факториала |

Порядок выполнения арифметических операций соответствует стандартным математическим правилам.

Обычно в математических выражениях, помимо стандартных арифметических операций, используются разнообразные **математические функции**. В Maple изначально определен большой набор стандартных математических функций, начиная от элементарных и заканчивая специальными функциями, возникающими при решении задач математической физики.

В таблице 4 приведены основные математические функции и соответствующий им синтаксис Maple.

Таблица 4. Основные математические функции

| Функция | Синтаксис Maple | Функция | Синтаксис Maple |
|---|---|---|---|
| $e^x$ | `exp(x)` | $\operatorname{sh} x$ | `sinh(x)` |
| $\ln x$ | `ln(x)` или `log(x)` | $\operatorname{ch} x$ | `cosh(x)` |
| $\log_{10} x$ | `log10(x)` | $\operatorname{th} x$ | `tanh(x)` |
| $\log_a x$ | `log[a](x)` | $\operatorname{cth} x$ | `coth(x)` |
| $\lvert x \rvert$ | `abs(x)` | $\arcsin x$ | `arcsin(x)` |
| $\sin x$ | `sin(x)` | $\arccos x$ | `arccos(x)` |
| $\cos x$ | `cos(x)` | $\operatorname{arctg} x$ | `arctan(x)` |
| $\operatorname{tg} x$ | `tan(x)` | $\operatorname{arcctg} x$ | `arccot(x)` |
| $\operatorname{ctg} x$ | `cot(x)` | | |

Отдельно приведем, как в Maple задается кусочная функция, на примере функции

$$\begin{cases} 1 & x > 2 \\ -1 & x \leqslant 2 \end{cases}$$

В Maple данное выражение записывается с помощью команды/функции `piecewise` следующим образом:

```
> piecewise(x > 2, 1, x <= 2, -1);
```

$$\begin{cases} 1 & x > 2 \\ -1 & x \leq 2 \end{cases}$$

Помимо функций из таблицы 4, Maple содержит огромное количество специальных функций, например, функции Бесселя,

интеграл ошибок, эллиптические интегралы и другие. В таблице 5 приведены некоторые специальные функции, часто используемые в математической физике.

Таблица 5. Специальные математические функции

| Функция | Синтаксис Maple |
|---|---|
| Функция Бесселя первого рода $J_\nu(x)$ | `BesselJ(nu, x)` |
| Функция Бесселя второго рода $Y_\nu(x)$ | `BesselY(nu, x)` |
| Модифицированная функция Бесселя первого рода $I_\nu(x)$ | `BesselI(nu, x)` |
| Модифицированная функция Бесселя второго рода $K_\nu(x)$ | `BesselK(nu, x)` |
| Функция ошибок $\mathrm{erf}(x)$ | `erf(x)` |
| Дополнительная функция ошибок $\mathrm{erfc}(x)$ | `erfc(x)` |
| Дельта-функция Дирака $\delta(x)$ | `Dirac(x)` |
| Функция Хевисайда $\theta(x)$ | `Heaviside(x)` |

Задание в Maple других специальных функций можно найти в Справке (клавиша «F1» на клавиатуре или через главное меню **Help→Maple Help**).

## 2.3. Работа с математическими выражениями

Технология работы в Maple заключается в том, что пользователь создает переменные, производит над ними некоторые действия в соответствии с поставленной задачей, используя стандартные или собственные процедуры.

Синтаксис вызова **стандартной команды или функции** следующий:

```
> команда(пар_1, пар_2, ... , пар_n);
```

Здесь `команда` — это имя вызываемой функции, а `пар_1, пар_2, ..., пар_n` означают необходимые для выполнения команды параметры.

Система Maple является достаточно интуитивной системой, поэтому обычно имя функции соответствует действию, которое

она выполняет (следует учесть, что все имена заданы на английском языке).

Далее рассмотрим базовые команды или функции Maple, не требующие дополнительного подключения пакетов.

**1. Функция для приближенных вычислений**. Команда или функция `evalf` численно оценивает выражения, включающие константы и математические функции, и имеет следующий синтаксис:

```
> evalf(expression, n);
```

или

```
> evalf[n](expression);
```

Здесь `expression` — выражение, значение которого должно быть вычислено, `n` — число значащих цифр, используемых при вычислении.

Пример использования команды `evalf`:

```
> evalf(ln(124), 5);
```

$$4.8203$$

```
> evalf[17](ln(124));
```

$$4.8202815656050369$$

Если число `n` не указано, то количество выводимых цифр определяется константой `Digits` (по умолчанию равной 10).

**2. Функции для выделения частей дроби**. Для выделения числителя или знаменателя дробного выражения используются функции `numer` и `denom` соотвественно. Синтаксис этих команд следующий:

```
> numer(expression);
> denom(expression);
```

Здесь `expression` — символьное или числовое выражение (вообще говоря, не обязательно дробное).

Примеры использования команд `numer` и `denom`:

```
> numer(2/3); denom(5);
```

$$2$$
$$1$$

```
> expr:= (h^2 + 3 - h^3) / (h^3 - h^5 + 6);
```

$$\frac{-h^3 + h^2 + 3}{-h^5 + h^3 + 6}$$

```
> numer(expr);
```

$$-h^3 + h^2 + 3$$

```
> denom(expr);
```

$$-h^5 + h^3 + 6$$

Если выражение `expression` не представлено в форме обыкновенной дроби, Maple преобразует его к требуемой форме.

```
> numer(2/x + x);
```

$$x^2 + 2$$

```
> denom(2/x + x);
```

$$x$$

**3. Функции для выделения частей выражения**. Как было сказано выше, объектом в Maple может быть символьное выражение, представляющее собой уравнение или неравенство. Выделение правой и левой частей выражения осуществляется командами `rhs` и `lhs` соответственно. Синтаксис этих команд следующий:

```
> rhs(expression);
> lhs(expression);
```

Здесь `expression` — символьное или числовое выражение, содержащее знак равенства (`=`) или неравенства (`<`, `>`).

Примеры использования команд `rhs` и `lhs`:

```
> rhs(x + y = z^2 + 1);
```

$$x + y$$

```
> lhs(x + y = z^2 + 1);
```

$$z^2 + 1$$

```
> rhs(r^2 + cos(s) > 1);
```

$$r^2 + \cos(s)$$

```
> lhs(r^2 + cos(s) > 1);
```

$$1$$

В выражении `expression` в функциях `rhs` и `lhs` обязательно должен быть разделяющий знак (=, <, >), иначе Maple выведет сообщение об ошибке.

**4. Функция замены переменных**. При выполнении математических преобразований часто необходимо произвести замену переменных в выражении, функции, уравнении и т. д., то есть вместо какой-то переменной подставить ее значение или представление через некоторые другие переменные. Для этих целей в системе Maple существует команда `subs`, синтаксис которой следующий:

```
> subs(old_var = new_var, expr);
```

Здесь `old_var` — переменная в выражении, которая будет заменена на новое значение (или переменную) `new_var`, `expr` — выражение, в котором производится замена. В общем случае `expr` может быть не одним выражением, а списком выражений, то есть задаваться как `{expr1, expr2, ..., exprN}`. Тогда замена старой переменной `old_var` на новую `new_var` будет производиться в каждом выражении из списка.

Если в выражении или наборе выражений требуется сделать замену более чем одной переменной, то синтаксис команды `subs` следующий:

```
> subs({old_v1 = new_v1, old_v2 = new_v2, ...}, expr);
```

Здесь все необходимые замены переменных указаны в фигурных скобках, то есть формируют список замен.

Примеры использования команды `subs`:

```
> subs(x = 5 * Pi, cos(x) + y);
```

$$\cos(5\pi) + y$$

```
> subs(x = 0, {x^2 + 1, y*2^(x+1), exp(x)});
```

$$\{1, 2y, e^0\}$$

```
> subs({x = z^2, y = -z}, x^2 + 2*x - exp(y) = 0);
```

$$z^4 + 2z^2 - e^{-z} = 0$$

Следует обратить внимание, что функция `subs` выполняет только замену переменных, при этом не вычисляет/упрощает полученное выражение. Для упрощения или вычисления могут быть использованы другие функции (например, `evalf`) или опция `eval` функции `subs`:

```
> subs[eval](old_var = new_var, expr);
```

Пример использования приведен ниже:

```
> subs(x = 5 * Pi, cos(x/4));
```

$$\cos\left(\frac{5\pi}{4}\right)$$

```
> evalf(subs(x = 5 * Pi, cos(x/4)), 5);
```

$$-0.70710$$

```
> subs[eval](x = 5 * Pi, cos(x/4));
```

$$-\frac{\sqrt{2}}{2}$$

Более сложные замены (имеется в виду замена целого выражения, а не переменной) могут быть выполнены с помощью команды `algsubs`, синтаксис которой аналогичен `subs`. Пример использования:

```
> algsubs(x^2 = 1-y^2, x^3);
```

$$(-y^2 + 1)x$$

**5. Функция упрощения выражений**. Для упрощения разнообразных выражений, составленных из чисел, переменных и элементарных или специальных функций, в Maple реализована команда `simplify`. Эта команда упрощает выражение, используя

свои внутренние закрытые алгоритмы, результат выполнения которых может не совсем соответствовать взглядам пользователя[3]. Синтаксис функции `simplify` следующий:

```
> simplify(expression, opts);
```

Здесь `expression` — выражение, которое необходимо упростить, `opts` — опции, указывающие команде `simplify` на некоторые свойства выражения. Заметим, что параметр `opts` не обязателен и может не указываться.

Примеры использования команды `simplify` без дополнительных опций:

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

$$A := \frac{x^2 - 2x + 1}{x - 1}$$

```
> simplify(A);
```

$$x - 1$$

```
> simplify(cosh(x)^2 - sinh(x)^2);
```

$$1$$

Рассмотрим некоторые опции команды `simplify`:

- `symbolic` — указывает, что упрощаемое выражение символьное:

  ```
  > simplify(sqrt(x^2));
  ```

  $$\mathrm{csgn}(x)x$$

  ```
  > simplify(sqrt(x^2), symbolic);
  ```

  $$x$$

- `trig` — указывает, что нужно упростить слагаемые с тригонометрическими функциями:

  ```
  > simplify(cos(2*x) + sin(x)^2, trig);
  ```

  $$\cos(x)^2$$

[3]Сложные выражения, как правило, можно представить в различных эквивалентных формах.

```
> simplify(sin(x)^2 + ln(2*x) + cos(x)^2, trig);
```

$$1 + \ln(2x)$$

- `ln` — указывает, что нужно упростить слагаемые с логарифмами:

```
> simplify(sin(x)^2 + ln(2*x) + cos(x)^2, ln);
```

$$\sin(x)^2 + \ln(2) + \ln(x) + \cos(x)^2$$

```
> simplify(ln(342368), ln);
```

$$5\ln(2) + \ln(13) + \ln(823)$$

- `power` — указывает, что упрощаемое выражение содержит степенные функции:

```
> simplify(x^a*x^b, power);
```

$$x^{a+b}$$

```
> simplify(exp(5*ln(x) + 1), power);
```

$$x^5\mathrm{e}$$

Заметим, что не всегда указание одной опции помогает упростить выражение до желаемого вида, например:

```
> simplify((a^b)^c, power);
```

$$(a^b)^c$$

В таких случаях можно дополнительно использовать опцию `symbolic`:

```
> simplify((a^b)^c, power, symbolic);
```

$$a^{bc}$$

**6. Функция для разложения на множители**. Для разложения на множители многочлена с целыми, рациональными, (комплексными) числовыми или алгебраическими коэффициентами используется команда `factor`, синтаксис которой следующий:

```
> factor(expression);
```

Здесь `expression` обозначает полином, который надо разложить на множители.

Примеры использования команды `factor`:

```
> factor(y^5 - y^4 - 7 * y^3 + y^2 + 6 * y);
```

$$y(y-1)(y-3)(y+2)(y+1)$$

```
> factor(1/(x^2 - 1) + 1/(x^2 + 3*x + 2));
```

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

**7. Функция для раскрытия скобок в выражении**. Команда `expand` раскрывает все скобки в алгебраическом выражении. Можно сказать, что команда `expand` противоположна функции `factor`. Синтаксис функции `expand` следующий:

```
> expand(expression);
```

Здесь `expression` — символьное или числовое выражение.

Примеры использования команды `expand`:

```
> q := (y + 1)*(y - 1)*(y^2 - y + 1)*(y^2 + y + 1);
```

$$q := (y+1)(y-1)(y^2-y+1)(y^2+y+1)$$

```
> expand(q);
```

$$y^6 - 1$$

Заметим, что функция `expand` «разворачивает» не только алгебраические выражения, но и тригонометрические:

```
> expand(sin(x + y));
```

$$\sin(x)\cos(y) + \cos(x)\sin(y)$$

**8. Функция для объединения членов выражения**. Команда `combine` пытается привести несколько членов в выражении к одному. Данная функция имеет следующий синтаксис:

```
> combine(expression, opts);
```

Здесь `expression` — преобразуемое выражение, `opts` — опции, указывающие команде `combine` на некоторые свойства выражения. Параметр `opts` не обязателен и может не указываться.

Примеры использования команды `combine` без дополнительных опций:

```
> combine(cos(alpha)*cos(beta)+sin(alpha)*sin(beta));
```

$$\cos(\alpha - \beta)$$

```
> combine(exp(x)^2 * exp(y));
```

$$e^{2x+y}$$

```
> combine(sin(4*x) * cos(2*x));
```

$$\frac{\sin(6x)}{2} + \frac{\sin(2x)}{2}$$

Названия опций функции `combine` совпадают с названиями опций команды `simplify` и имеют тот же смысл.

**9. Функция для приведения подобных членов выражения**. Команда `collect` собирает коэффициенты при заданном выражении или выражениях и имеет следующий синтаксис:

```
> collect(expression, vars, opts);
```

Здесь `expression` — символьное или числовое выражение, `vars` — переменная или список переменных, при которых будут приводиться подобные члены, `opts` — дополнительные опции. Опции `opts` используются, если в `vars` указана не одна переменная, а список переменных.

Примеры использования команды `collect`:

```
> f := a^3*x + a*x^2 + a - x - b*x^2;
```

$$f := a^3x + ax^2 + a - x - bx^2$$

```
> collect(f, x);
```

$$(a-b)x^2+(a^3-1)x+a$$

```
> p := -a*x^2*y + a*x*y + x^2*y + a*x + x*y:
  collect(p, {x, y});
```

$$(-a+1)yx^2+((a+1)y+a)x$$

```
  collect(p, {x, y}, distributed);
```

$$(-a+1)yx^2+(a+1)xy+ax$$

**10. Функция для работы с радикалами**. Не всегда выше перечисленные команды упрощения корректно срабатывают с выражениями, содержащими радикалы, в этих случаях можно использовать команду `radnormal`. Синтаксис этой функции следующий:

```
> radnormal(expression);
```

Здесь `expression` — выражение, содержащее радикалы.

Примеры использования команды `radnormal`:

```
> a := (7 + 5*sqrt(2))^(1/3);
```

$$a := (7+5\sqrt{2})^{1/3}$$

```
> radnormal(a);
```

$$1+\sqrt{2}$$

Для сравнения результат команды `simplify`:

```
> simplify(a);
```

$$a := (7+5\sqrt{2})^{1/3}$$

**11. Функции для определения свойств переменной**. Команда `assume` устанавливает свойства переменных и отношения между переменными. Аналогично работает команда `assuming`. Синтаксис команды `assume` следующий:

```
> assume(x1, prop1, x2, prop2, ...);
```

или

```
> assume(xprop1, xprop2, ...);
```

Здесь `x1, x2` — имена переменных, свойства которых определяются, `prop1, prop2` — названия свойств, `xprop1, xprop2` — неравенства, определяющие свойства переменных.

Пример использования команды `assume`:

```
> assume(a < 0);
> b + a;
```

$$b + a\sim$$

```
> assume(f, 'positive');
> f;
```

$$f\sim$$

После использования команды `assume` во **всем рабочем листе** будет существовать указанное определение. При выводе некоторого выражения с переменной, на которую было наложено ограничение, рядом с этой переменной будет отображаться знак «тильда».

Данная команда удобна при работе с символьными выражениями:

```
> sqrt(a);
```

$$\mathrm{I}\sqrt{-a\sim}$$

Команда `assuming` используется в случае, когда определение переменной необходимо только **при выполнении конкретной команды**, но не во всем рабочем листе. Пример использования команды `assuming`:

```
> abs(b);
```

$$|b|$$

```
> abs(b) assuming b < 0;
```

$$-b$$

**12. Применение команды/функции к нескольким выражениям**. В случае, когда существует список выражений и к нему необходимо применить некоторую команду (например, упрощения), то используется функция `map`, синтаксис которой следующий:

```
> map(command, {x1, x2, ...});
```

Здесь `command` — название команды, `{x1, x2, ...}` — список выражении или переменных, к которым будет применена команда `command`.

Пример использования команды `map`:

```
> map(lhs, {a = b, x^2 + 5 = 8});
```

$$\{a, x^2 + 5\}$$

Команда `map` может применять к списку не только базовые команды, существующие в Maple, но и команды, объявленные пользователем[4], например:

```
> map(f, {x, y, z});
```

$$\{f(x), f(y), f(z)\}$$

В случае, если применяемая к списку команда имеет два входных значения (например, функция `subs`), то используется команда `map2`:

```
> map2(subs, x=0, {cos(x), x, exp(x + y)});
```

$$\{0, \cos(0), e^y\}$$

**13. Функции для работы с комплексными числами**. Объявление переменной, которая содержит комплексное число или выражение, выполняется так же, как и во всех выше рассмотренных примерах

```
> a := 8 + I*1;
```

$$a := 8 + \mathrm{I}$$

[4]Создание пользовательских команд/функций рассмотрено в разделе 6.3 второй части методического пособия.

```
> b := ln(x + I*y);
```

$$b := \ln(x + \mathrm{I}y)$$

Основные команды Maple, используемые для работы с комплексными выражениями:

- `evalc(complex_expr)` — вычисление выражения, например, нахождения суммы или возведения в степень

```
> k := 1 + I*7:
  l := 3 - I*4:
  evalc(k + l);
```

$$4 + 3\mathrm{I}$$

```
> evalc(k^3);
```

$$-146 - 322\mathrm{I}$$

- `Re(complex_expr)` — выделение действительной части комплексного числа

```
> a := ln(x + I*y);
```

$$a := \ln(x + \mathrm{I}y)$$

```
> Re(a);
```

$$\ln(|x + y\mathrm{I}|)$$

```
> evalc(Re(a));
```

$$\frac{\ln(x^2 + y^2)}{2}$$

- `Im(complex_expr)` — выделение мнимой части комплексного числа

```
> Im(a);
```

$$arg(x + y\mathrm{I})$$

```
> evalc(Im(a));
```

$$\arctan(y, x)$$

- `conjugate(complex_expr)` — вычисление комплексного сопряжения

```
> conjugate(15 + I*8);
```

$$15 - 8\mathrm{I}$$

## Задачи

2.1. Вычислить с точностью до $10^{-10}$:

$$\cos\frac{\pi}{8}, \quad \frac{1}{241}, \quad \frac{\pi^3}{e^2}, \quad \sqrt{\ln 5}.$$

2.2. Упростить выражения:

$$\frac{8a^{\frac{3}{2}} + 27b^{\frac{3}{2}}}{2a^{\frac{1}{2}} - 3b^{\frac{1}{2}}} \cdot \frac{4a - 9b}{4a - 6a^{\frac{1}{2}}b^{\frac{1}{2}} + 9b} - 12a^{\frac{1}{2}}b^{\frac{1}{2}},$$

$$\left(\frac{m}{m-6} - \frac{2m}{m^2 - 12m + 36}\right)\frac{36 - m^2}{m - 8} + \frac{12m}{m-6},$$

$$\frac{1 - \mathrm{tg}^2\,\alpha + \mathrm{tg}^4\,\alpha}{\cos^2\alpha}.$$

2.3. Подставить в выражение

$$\cos^2 t - \cos^4 t + \sin^4 t$$

вместо переменной $t$ выражение $y^2 + \pi$ и упростить его.

# 3. Базовые математические команды

## 3.1. Пределы, производные, интегралы

**1.** Для нахождения пределов в Maple реализована команда/функция `limit` со следующим синтаксисом:

```
> limit(expression, var = a, opts);
```

Здесь `function` — выражение, предел которого вычисляется, `var` — имя переменной, которая стремится к `a`, `opts` — дополнительная опция, задающая стремление слева или справа. Параметр `opts` может не указываться, если в нем нет необходимости.

Пример вычисления пределов $\lim\limits_{x\to 0}\dfrac{\sin x}{x}$, $\lim\limits_{n\to 0}\dfrac{1}{n}(e^n-1)$:

```
> limit(sin(x)/x, x = 0);
```

$$1$$

```
> limit((exp(n) - 1)/n, n = 0);
```

$$1$$

Пример вычисления предела $\lim\limits_{z\to 0} z^{-1}$ при стремлении $z$ к нулю слева и справа:

```
> limit(1/z, z = 0, left);
```

$$-\infty$$

```
> limit(1/z, z = 0, right);
```

$$\infty$$

Часто необходимо вычислить предел от выражения, содержащего параметр, при этом этот параметр может влиять на результат. В этом случае удобно использовать команду `assuming`, описанную в разделе 2.3. Пример вычисления предела $\lim\limits_{x\to 0}\left|\dfrac{\sin ax}{x}\right|$, считая $a<0$:

```
> limit(abs(sin(a*x)/x), x = 0) assuming a < 0;
```

$$-a$$

**2.** Для вычисления производных первого, второго или $n$-ого порядка, а также вычисления частных производных используется функция `diff`.

Синтаксис команды `diff` для вычисления производной по **одной** переменной первого, второго или $n$-ого порядка соответственно:

```
> diff(expression, x);
> diff(expression, x, x);
> diff(expression, x$n);
```

Здесь `expression` — выражение, производная которого вычисляется, `x` — имя переменной, по которой вычисляется производная, `n` — порядок производной.

Пример вычисления производных $\dfrac{d}{dx}\cos x$, $\dfrac{d^2}{dx^2}\ln(\cos x)$:

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

$$-\sin(x)$$

```
> diff(ln(cos(x)), x$2);
```

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

Функция `diff`, как и многие функции в Maple, предназначенные для символьных вычислений, не упрощает полученные выражения. Поэтому совместно с `diff` полезно использовать функции упрощения выражений, описанные в разделе 2.3:

```
> simplify(diff(ln(cos(x)), x$2));
```

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

Команда `diff` также может вычислить частную производную от функции нескольких переменных. Пример вычисления частных производных $\dfrac{\partial}{\partial x}\cos(xy^2+1)$, $\dfrac{\partial}{\partial y}\cos(xy^2+1)$:

```
> diff(cos(x*y^2+1), x);
```

$$-y^2\sin(xy^2+1)$$

```
> diff(cos(x*y^2+1), y);
```

$$-2xy\sin(xy^2+1)$$

Синтаксис команды `diff` для вычисления производной по **нескольким** переменным первого или $n$-ого порядка соответственно:

```
> diff(expression, [x1, x2, ... xN]);
> diff(expression, [x1$n1, x2$n2, ... xN$nN]);
```

Здесь `expression` — выражение, производная которого вычисляется, `[x1, x2, ... xN]` — список переменных, по которым вычисляется производная, `n1, n2, ..., nN` — порядок дифференцирования для каждой переменной.

Пример вычисления производных $\dfrac{\partial^2}{\partial x\partial y}xy$, $\dfrac{\partial^3}{\partial x^2\partial y}\ln\cos(xy)$:

```
> diff(x*y, [x, y]);
```

$$1$$

```
> A := diff(ln(cos(x*y)), [x$2, y]);
```

$$A := -2y - \frac{2y\sin(xy)^2}{\cos(xy)^2} - \frac{2y^2\sin(xy)x}{\cos(xy)} - \frac{2y^2\sin(xy)^3x}{\cos(xy)^3}$$

```
> simplify(A);
```

$$-\frac{2y(xy\sin(xy)+\cos(xy))}{\cos(xy)^3}$$

**3.** Вычисление интегралов реализовано в Maple через функцию/команду `int`. Данная функция может использоваться для:

1) Аналитического и численного вычисления неопределенных и определенных интегралов. Соответствующий синтаксис команды `int`:

   ```
   > int(expression, x);
   ```

   ```
   > int(expression, x = a..b, numeric);
   ```

   Здесь `expression` — подынтегральное выражение, `x` — переменная интегрирования, `a..b` — пределы интегрирования,

`numeric` — опция для вычисления интеграла численными методами (только для определенных интегралов). Если опция `numeric` не указана, то функция `int` будет пытаться построить решение аналитически.

2) Аналитического и численного вычисления кратных интегралов. Синтаксис команды `int`:

```
> int(expression, [x1, x2, ..., xN]);
> int(expression, [x1 = a1..b1, ...,
                                    xN = aN..bN], numeric);
```

Здесь `expression` — подынтегральное выражение, `[x1, x2, ..., xN]` — список переменных, по которым ведется интегрирование, `a1..b2, ..., aN..bN` — пределы интегрирования для каждой переменной, `numeric` — опция численного вычисления интеграла (только для определенных интегралов). Так же, как и для однократного интеграла, если опция `numeric` не указана, то функция `int` будет вычислять кратный интеграл аналитически.

Примеры вычисления интегралов $\int \sin x dx$, $\int\limits_a^b x dx$, $\int\limits_1^2 \mathrm{ctg}(\sin x) dx$:

```
> int(sin(x), x);
```

$$-\cos(x)$$

```
> int(x, x = a..b);
```

$$-\frac{a^2}{2}+\frac{b^2}{2}$$

```
> int(cot(sin(x)), x = 1..2);
```

$$\int\limits_1^2 \cot(\sin x)\mathrm{d}x$$

Из примера видно, что Maple не удалось построить точное решение для последнего интеграла, поэтому в качестве результата пакет выводит математическую форму записи интеграла. В таких случаях предлагается вычислить интеграл численно:

```
> int(cot(sin(x)), x = 1..2, numeric);
```

$$0.7073342093$$

Примеры вычисления кратных интегралов $\iiint xyzdxdydz$, $\int_{\pi}^{2\pi}\int_{0}^{y} y\cos x dxdy$, $\int_{-1}^{1}\int_{0}^{1}\int_{0}^{1}\cos(\sin(xyz))dxdydz$:

```
> int(x*y*z, [x, y, z]);
```

$$\frac{x^2y^2z^2}{8}$$

```
> int(y*cos(x), [x = 0..y, y = Pi..2*Pi]);
```

$$-3\pi$$

```
> int(cos(sin(x*y*z)), [x = 0..1, y = 0..1,
      z = -1..1], numeric);
```

$$1.966024690$$

## 3.2. Суммы, последовательности, произведения

**1.** Для вычисления конечных и бесконечных сумм (рядов) в Maple реализована команда `sum`, синтаксис которой следующий:

```
> sum(expression, n = a..b);
```

Здесь `expression` — общее слагаемое вычисляемой суммы, зависящее от индекса суммирования $n$, вторым параметром указывается диапазон значений, в котором изменяется индекс `n = a..b`.

Пример вычисления конечных сумм $\sum_{k=1}^{5} k$ и $\sum_{n=-50}^{50}\cos(n\pi)$:

```
> sum(k, k = 1..5);
```

$$15$$

```
> sum(cos(n * Pi), n = -50..50);
```

$$1$$

Особенностью команды `sum` является то, что пределы суммирования необязательно должны быть числовыми значениями. В качестве пределов суммирования может быть указана переменная, в которую ничего не записано, или знак бесконечности. В этом случае функция `sum` попытается построить аналитическую формулу для результата суммирования.

Пример вычисления конечных рядов с неопределенным верхним пределом суммирования $\sum_{k=1}^{n} k^3$ и $\sum_{k=0}^{n-1} \sin\frac{2\pi k}{n} \cos\frac{\pi k}{n}$:

```
> sum(k^3, k = 1..n);
```

$$\frac{(n+1)^4}{4} - \frac{(n+1)^3}{2} + \frac{(n+1)^2}{4}$$

```
> A := sum(sin(k*2*Pi/n) * cos(k*Pi/n), k = 0..n-1);
```

$$A := -\frac{2\left(2\cos\left(\frac{\pi}{n}\right)-1\right)\sin\left(\frac{\pi}{n}\right)}{\left(1+2\cos\left(\frac{\pi}{n}\right)\right)\left(\cos\left(\frac{\pi}{n}\right)-1\right)} + \frac{2\sin\left(\frac{\pi}{n}\right)}{1+2\cos\left(\frac{\pi}{n}\right)}$$

Следует заметить, что функция `sum` не упрощает полученное выражение, поэтому необходимо использовать функции, описанные в разделе 2.3:

```
> simplify(A);
```

$$-\frac{2\sin\left(\frac{\pi}{n}\right)\cos\left(\frac{\pi}{n}\right)}{2\cos\left(\frac{\pi}{n}\right)^2 - \cos\left(\frac{\pi}{n}\right) - 1}$$

Пример вычисления бесконечной суммы (ряда) $\sum_{k=1}^{\infty} \frac{1}{k(3k+1)}$:

```
> sum(1/(k*(3*k+1)), k = 1..infinity);
```

$$3 - \frac{\pi\sqrt{3}}{6} - \frac{3\ln(3)}{2}$$

Функция `sum` может представить результат вычисления ряда (или конечной суммы) не только через элементарные функции,

но и через специальные функции, например, $\sum_{k=0}^{\infty} \frac{(-1)^k}{k!(k+a)}$:

```
> sum((-1)^k/(k!*(k + a)), k = 0..infinity);
```

$$-\Gamma(\ln(x + \mathrm{I}y), 1) + \Gamma(\ln(x + \mathrm{I}y))$$

где Γ обозначает гамма-функцию. Однако следует понимать, что возможности Maple ограничены и не каждый ряд может быть просуммирован, в этом случае Maple выведет математическую запись суммы:

```
> sum(sinh(k * x + a), k = 1..infinity);
```

$$\sum_{k=1}^{\infty} \sinh(kx + a)$$

**2.** Помимо функции `sum`, для того чтобы суммировать последовательность значений, может использоваться функция `add`. Синтаксис этой функции (частично совпадает с `sum`):

```
> add(expression, n = a..b);
```

или

```
> add(expression, n = {n1, ..., nN});
```

Здесь `expression` — общее слагаемое суммы, зависящее от $n$, вторым параметром может быть диапазон значений индекса `n = a..b` или список значений, которые он должен принимать `n = {n1, ..., nN}`.

Пример вычисления суммы $\sum_{k=1}^{5} k$ :

```
> add(k, k = 1..5);
```

15

```
> add(k, k = {1, 2, 3, 4, 5});
```

15

Между функциями `sum` и `add`, несмотря на схожесть, существует принципиальная разница. При вызове команды `sum` суммирование производится в символьной форме. Другими словами, Maple в этом случае пытается получить аналитическую

формулу. Функция же `add` используется для выполнения суммирования в явном виде, результат ее выполнения — число. Поэтому в качестве параметров команды `add` символьные значения использоваться не могут.

Примеры **неправильного** использования функции `add`:

```
> add(1/r^2, r = 1..infinity);
Error, unable to execute add
> sum(1/r^2, r = 1..infinity);
```

$$\frac{\pi^2}{6}$$

```
> add(k, k = 1..N);
Error, unable to execute add
> sum(k, k = 1..N);
```

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

**3.** Для создания последовательностей, элементы которых подчиняются определенной закономерности, используется команда `seq`, синтаксис которой следующий:

```
> seq(expression, n = a..b, step);
```

Здесь `expression` — общий вид элемента последовательности, зависящий от $n$, `n = a..b` — диапазон значений параметра $n$ ($b > a$), `step` — шаг, с которым изменяется $n$. Если параметр `step` не указан, то шаг по умолчанию считается равным единице.

Примеры использования функции `seq`:

```
> seq(2*Pi*i/5, i = 0..5);
```

$$0, \frac{2\pi}{5}, \frac{4\pi}{5}, \frac{6\pi}{5}, \frac{8\pi}{5}, 2\pi$$

```
> seq(k, k = 0.3..2);
```

$$0.3, 1.3$$

```
> seq(k, k = 0.3..2, 0.2);
```

$$0.3, 0.5, 0.7, 0.9, 1.1, 1.3, 1.5, 1.7, 1.9$$

**4.** Для вычисления конечных и бесконечных произведений в Maple реализована команда `product`:

```
> product(expression, n = a..b);
```

Здесь `expression` — общий вид элемента последовательности, зависящий от $n$, `n = a..b` — диапазон значений параметра $n$.

Пример вычисления конечного произведения $\prod_{n=0}^{5}(1+n)^2$:

```
> product((1 + n)^2, n = 0..5);
```

$$518400$$

Функция `product` может вычислять некоторые произведения с неопределенными или бесконечными пределами по `n`. Примеры вычисления произведений $\prod_{k=1}^{n}(k+a)$ и $\prod_{k=1}^{\infty}\left(1+\frac{1}{k^3}\right)$:

```
> product((k + a), k = 1..n);
```

$$\frac{\Gamma(1+n+a)}{\Gamma(1+a)}$$

```
> A := product((1 + 1/(k^3)), k = 1..infinity);
```

$$A := \frac{\sin\left(\pi\left(\frac{1}{2}+\frac{\mathrm{I}\sqrt{3}}{2}\right)\right)}{\pi}$$

Команда `product`, как и команда `sum`, не упрощает полученное выражение, поэтому совместно с `product` полезно использовать функции упрощения (см. раздел 2.3):

```
> simplify(A);
```

$$\frac{\cosh\left(\frac{\pi\sqrt{3}}{2}\right)}{\pi}$$

Следует помнить, что Maple имеет ограниченный функционал и не может вычислить все произведения, в этом случае будет выведена математическая форма записи вычисляемого произведения:

```
> product(cos(Pi*j/n), j = 1..n);
```

$$\prod_{j=1}^{n}\cos\left(\frac{\pi j}{n}\right)$$

## 3.3. Разложения в ряды

**1.** Для разложения выражения (функции $f(x)$) в степенной ряд или ряд Лорана в окрестности некоторой точки $x = a$ используется функция `series`. Синтаксис этой команды следующий:

```
> series(expression, var = a, order);
```

Здесь `expression` — выражение, разлагаемое в ряд, `var = a` — указывает, по какой переменной `var` выражение раскладывается в ряд и в окрестности какого значения `a`[5], `order` — порядок разложения или «порядок усечения» вычислений ряда. Параметры `a` и `order` могут не указываться, по умолчанию они имеют значения:

– `a = 0`, то есть разложение будет строится в окрестности 0.

– `order = 6`, то есть разложение строится до пятой степени.

Пример разложения функции $\dfrac{1}{\sin^2 x}$ в степенной ряд в окрестности точки $x = 0$:

```
> series(1/sin(x)^2, x = 0);
```

$$x^{-2} + \frac{1}{3} + \frac{1}{15}x^2 + \frac{2}{189}x^4 + \mathrm{O}(x^6)$$

```
> series(1/sin(x)^2, x = 0, 8);
```

$$x^{-2} + \frac{1}{3} + \frac{1}{15}x^2 + \frac{2}{189}x^4 + \frac{1}{675}x^6 + \mathrm{O}(x^8)$$

Функция `series` предусматривает разложение в окрестности некоторой переменной (не числа), а также может работать со специальными функциями. Пример разложения гамма-функции $\Gamma(x)$ в окрестности точки $x = a$ до первого порядка (включительно):

```
> series(GAMMA(x), x = a, 2);
```

$$\Gamma(a) + \psi(a)\Gamma(a)(x - a) + \mathrm{O}((x - a)^2)$$

[5]Значение `a` может быть как действительным, так и комплексным.

Здесь $\psi(a)$ — дигамма-функция.

**2.** Разложение функции в ряд Тейлора в окрестности некоторой точки в Maple выполняется с помощью функции `taylor`, синтаксис которой полностью совпадает с синтаксисом команды `series`:

```
> taylor(expression, var = a, order);
```

Здесь `expression` — разлагаемое в ряд выражение, `var = a` — указывается, по какой переменной `var` идет разложение и в окрестности какой точки `a` (по умолчанию `a = 0`), `order` — порядок разложения.

Примеры разложения в ряд Маклорена функций $\sin x$ и $e^{\cos x}$ приведены ниже:

```
> taylor(sin(x), x = 0, 4);
```

$$x - \frac{1}{6}x^3 + \mathrm{O}(x^5)$$

```
> taylor(exp(cos(x)), x = 0);
```

$$\mathrm{e} - \frac{1}{2}\mathrm{e}x^2 + \frac{1}{6}\mathrm{e}x^4 + \mathrm{O}(x^6)$$

Команда `taylor` также может разлагать функцию в ряд Тейлора в окрестности нечисловой переменной. Пример разложения функции $\sin(\cos x)$ в окрестности точки $x = a$:

```
> taylor(sin(cos(x)), x = a, 2);
```

$$\sin(\cos(a)) - \cos(\cos(a))\sin(a)(x-a) + \mathrm{O}((x-a)^2)$$

**3.** Команда `taylor` не может быть использована для разложения в ряд Тейлора функции нескольких переменных. Для этого в Maple реализована функция `mtaylor`, синтаксис которой следующий:

```
> mtaylor(expr, {var1 = a, ..., varN = aN}, order);
```

Здесь `expr` — выражение от $N$ переменных, разлагаемое в ряд, `{var1 = a1, ..., varN = aN}` — список переменных, по которым выражение `expr` раскладывается в ряд, для каждой переменной указывается значение `a1, ..., aN`, в окрестности которого ведется разложение (по умолчанию `a1 = 0, ..., aN = 0`), `order`

— порядок разложения (указывает, что члены с порядком малости $\geqslant$`order` отбрасываются).

Пример разложения функции двух переменных $\sin(xy)$ в ряд Тейлора в окрестности точки $(x, y) = (0, \pi)$:

```
> mtaylor(sin(y*x), {x = 0, y = Pi}, 5);
```

$$\pi x + (y - \pi)x - \frac{\pi^3 x^3}{6} - \frac{((y - \pi)x^3\pi^2}{2}$$

```
> mtaylor(sin(y*x), {x, y = Pi}, 5);
```

$$\pi x + (y - \pi)x - \frac{\pi^3 x^3}{6} - \frac{((y - \pi)x^3\pi^2}{2}$$

Пример разложения функции трех переменных $e^{xyz}$ в ряд Тейлора в окрестности нуля:

```
> mtaylor(exp(y*x*z), {x, y, z}, 10);
```

$$1 + yxz + \frac{1}{2}y^2x^2z^2 + \frac{1}{6}y^3x^3z^3$$

**4.** Следует отметить, что в пакете Maple не предусмотрена функция/команда для разложения функций в ряд Фурье. Однако эта функция может быть реализована пользователем самостоятельно, создание пользовательских функций рассматривается в разделе 6.3 второй части учебно-методического пособия.

Далее рассмотрим некоторые команды, которые могут быть полезными при работе с разложениями.

**4.1.** Для дальнейшей работы с разложениями, полученными с помощью функций `series`, `taylor`, необходимо сначала использовать команду `convert`, со следующим синтаксисом:

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

Здесь `expression` — результат разложения некоторой функции в ряд. Данное преобразование необходимо, поскольку по умолчанию результат функций `series` и `taylor` представляется в Maple как объект специального типа series. После применения команды `convert` разложение представляется в Maple как обычный полином, при этом слагаемое $O(x^n)$ отбрасывается.

Следует отметить, что функция `mtaylor`, в отличие от `series`, `taylor`, сразу возвращает разложение в виде полинома и не требует использования функции `convert`.

Пример использования команды `convert` для разных разложений:

```
> A := series(1/sin(x), x = 0);
```

$$A := x^{-1} + \frac{1}{6}x + \frac{7}{360}x^3 + \frac{31}{15120}x^5 + \mathrm{O}(x^7)$$

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

$$\frac{1}{x} + \frac{x}{6} + \frac{7x^3}{360}$$

```
> convert(taylor(exp(x), x = 1, 3), polynom);
```

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

**4.2.** Функции `op` и `nops` для выделения слагаемых и подсчета их количества в разложении (полиноме) соотвественно, имеют следующий синтаксис:

```
> op(polynom);
> nops(polynom);
```

Здесь `polynom` — анализируемый полином, например, полученный из разложения в ряд. Функция `op` возвращает список всех слагаемых в полиноме, ниже приведен пример обращения к конкретному слагаемому в списке.

Примеры использования функций `op` и `nops`:

```
> A := series(tan(x), x = 0):
  B := convert(A, polynom);
```

$$B := x + \frac{1}{3}x^3 + \frac{2}{15}x^5$$

```
> op(B);
```

$$x, \frac{x^3}{3}, \frac{2x^5}{15}$$

```
> nops(B);
```

3

```
> C := mtaylor(exp(x*y), {x, y}, 7);
```

$$C := 1 + xy + \frac{1}{2}x^2y^2 + \frac{1}{6}x^3y^3$$

```
> L := op(C);
```

$$L := 1, xy, \frac{x^2y^2}{2}, \frac{x^3y^3}{6}$$

Для того чтобы получить конкретное слагаемое, необходимо обратиться к переменной, в которой хранятся все слагаемые, через квадратные скобки `[]` с указанием в них номера слагаемого, считая, что нумерация начинается с единицы:

```
> L[2]; L[4];
```

$$xy$$
$$\frac{x^3y^3}{6}$$

**4.3.** Часто при построении разложения функции в ряд необходимо выделить коэффициенты при членах разложения. В Maple для этого реализованы функции `coeff` (выделение коэффициента при конкретной степени) и `coeffs` (выделение всех коэффициентов в разложении/полиноме). Синтаксис этих функций следующий:

```
> coeffs(polynom);
> coeff(polynom, var^n);
```

или

```
> coeff(polynom, var, n);
```

Здесь `polynom` — полином, `var` и `n` — имя переменной и ее степень, при которой выделяется коэффициент.

Примеры использования функций `coeff` и `coeffs`:

```
> A := convert(taylor(ln(1 - 2*x), x), polynom);
```

$$A := -2x - 2x^2 - \frac{8}{3}x^3 - 4x^4 - \frac{32}{5}x^5$$

```
> coeffs(A);
```

$$-2, -2, -\frac{8}{3}, -4, -\frac{32}{5}$$

```
> coeff(A, x^4);
```

$$-4$$

```
> coeff(A, x, 2);
```

$$-2$$

## 3.4. Решение алгебраических уравнений

**1.** Решение алгебраических уравнений и системы алгебраических уравнений выполняется с помощью функции `solve`. Синтаксис команды `solve` для решения **одного** уравнения:

```
> solve(equation, x);
```

Здесь `equation` — решаемое уравнение (должно содержать знак равенства `=`, иначе по умолчанию выражение `equation` будет приравниваться нулю), `x` — переменная, относительно которой разрешается уравнение.

Пример решения уравнений $x^2 + 2x - 1 = 0$, $\cos x = 0$:

```
> solve(x^2 + 2*x - 1 = 0, x);
```

$$\sqrt{2} - 1, -1 - \sqrt{2}$$

```
> solve(cos(x) = 0, x);
```

$$\frac{\pi}{2}$$

Из последнего примера видно, что Maple находит не общее решение уравнения, равное $\pi/2 + \pi n$, $n \in \mathbb{Z}$, а только частное при $n = 0$.

Синтаксис команды `solve` для решения **системы уравнений** следующий:

```
> solve({eq1, eq2, ..., eqN}, {x1, x2, ..., xN});
```

Здесь `{eq1, eq2, ..., eqN}` — список уравнений, `{x1, x2, ..., xN}` — список переменных, относительно которых эти уравнения надо разрешить.

Пример решения систем уравнений

$$\begin{cases} 3x + 4y = -2 \\ 2x - y = 1 \end{cases} \quad \text{и} \quad \begin{cases} xy = 0 \\ x + y = -2 \end{cases} :$$

```
> solve({3*x + 4*y = -2, 2*x - y = 1}, {x, y});
```

$$\left\{x = \frac{2}{11}, y = -\frac{7}{11}\right\}$$

```
> solve({x*y = 0,x + y = -2}, {x, y});
```

$$\{x = 0, y = -2\}, \{x = -2, y = 0\}$$

Следует понимать, что команда `solve` не всегда может найти решение уравнения или системы уравнений. Один из случаев, когда Maple не может построить решение, это нахождение корней полиномов высокой степени. В этом случае решения могут быть очень громоздкими, поэтому в качестве результата выводится функция *RootOf*. Пример такого уравнения $x^4 - x + 1 = 0$:

```
> solve(x^4 - x + 1 = 0, x);
```

$$\begin{gathered} RootOf(\_Z^4 - \_Z + 1, index = 1), \\ RootOf(\_Z^4 - \_Z + 1, index = 2), \\ RootOf(\_Z^4 - \_Z + 1, index = 3), \\ RootOf(\_Z^4 - \_Z + 1, index = 4) \end{gathered}$$

**2.** Специальная функция *RootOf*, которая применяется для обозначения любого корня уравнения относительно заданной переменной. В примере выше функция $RootOf(\_Z^4 - \_Z + 1)$ представляет любое решение уравнения $x^4 - x + 1 = 0$ относительно переменной $x$.

Если результат решения представлен через функцию *RootOf*, то получить все корни можно через функцию `allvalues` или с помощью функции `evalf` (описана в разделе 2.3), которая может вывести приближенные числовые значения корней уравнения. Пример использования функции `evalf`[6]:

```
> A := solve(x^4 - x + 1 = 0, x):
```

[6]Пример использования команды `allvalues` не приводится, поскольку выводимый результат слишком громоздкий. Читателю предлагается самостоятельно посмотреть работу данной функции.

```
> evalf(A);
```

$$0.7271360845 + 0.4300142883\mathrm{I}, -0.7271360845 + 0.9340992895\mathrm{I},$$
$$-0.7271360845 - 0.9340992895\mathrm{I}, 0.7271360845 - 0.4300142883\mathrm{I}$$

**3.** Не в каждом случае, когда при решении уравнения или системы уравнений с помощью функции `solve` появляется функция *RootOf*, явный вид решения можно получить через функцию `allvalues`. Примеры решения таких уравнений:

```
> A := solve(cos(x^2) = 2*cos(x) + x, x);
```

$$A := RootOf(2\cos(\_Z) - \cos(\_Z^2) + \_Z)$$

```
> allvalues(A);
```

$$RootOf(2\cos(\_Z) - cos(\_Z^2) + \_Z, -0.6680246832), ...$$
$$..., RootOf(2\cos(\_Z) - \cos(\_Z^2) + \_Z, 2.600945398)$$

```
> B := solve(tan(x) = x, x);
```

$$B := RootOf(-\tan(\_Z) + \_Z)$$

```
> allvalues(B);
```

$$RootOf(-\tan(\_Z) + \_Z, -4.493409458), ..., 0$$

Когда при вызове функции `allvalues` вместо явного вида решения опять появляется специальная функция *RootOf*, то это говорит о невозможности найти аналитическое решение уравнения.

В случае, когда уравнение или система уравнений не имеет точного аналитического решения, используется функция `fsolve` для приближенного нахождения численного решения. Синтаксис этой команды следующий:

```
> fsolve(equations, {x1, x2, ..., xN}, opts);
```

или

```
> fsolve(equations, {x1 = a1, ..., xN = aN}, opts);
```

Здесь `equations` — уравнение или список уравнений, которые надо решить, `{x1, x2, ..., xN}` — список переменных, относительно которых решаются уравнения (если решается одно уравнение относительно одной переменной, то фигурные

скобки можно опустить), `opts` — необязательные дополнительные опции (например, для поиска комплексных корней)[7]. Иногда уравнение или система уравнений имеет не единственное решение, в этих случая бывает удобно указать начальное приближение `{a1, a2, ..., aN}` для численного поиска корней.

Пример решения трансцендентного уравнения $\operatorname{ctg}(x) = -x$:

```
> fsolve(cot(x) = -x, x);
```

$$-2.798386046$$

```
> fsolve(cot(x) = -x, x = 2);
```

$$2.798386046$$

```
> fsolve(cot(x) = -x, x = 10);
```

$$9.317866462$$

Пример численного решения системы уравнений

$$\begin{cases} 3x^4y^2 = 17, \\ x^3y - 5xy^2 - 2y = 1 \end{cases}$$

```
> fsolve({3*x^4*y^2 = 17, x^3*y - 5*x*y^2 - 2*y = 1},
         {x, y});
```

$$\{x = 2.118203038, y = 0.5305528603\}$$

```
> fsolve({3*x^4*y^2 = 17, x^3*y - 5*x*y^2 - 2*y = 1},
         {x = -10, y = -10});
```

$$\{x = -1.619502631, y = 0.9076126019\}$$

[7]Названия этих опций читателю предлагаем посмотреть самостоятельно в Справке: `fsolve, details`.

## 3.5. Примеры решения задач

**1.** Найти точки минимума/максимума следующей функции $f(x) = (x^2 + 2x - 1)e^x$.

Начнем с того, что создадим переменную `y`, в которую запишем функцию $f(x)$:

```
> restart;
> y := (x^2 + 2*x - 1) * exp(x);
```

$$y := (x^2 + 2x - 1)\mathrm{e}^x$$

Для определения точек экстремума необходимо вычислить первую производную функции, для этого используем функцию `diff`. Результат вычисления упростим и запишем в переменную `dy`:

```
> dy := simplify(diff(y, x));
```

$$dy := \mathrm{e}^x(x^2 + 4x + 1)$$

Полученное выражение для производной необходимо приравнять к нулю и разрешить полученное уравнение относительно переменной $x$. Для решения уравнения используем функцию `solve`:

```
> xs := solve(dy = 0, x);
```

$$xs := -2 + \sqrt{3}, -2 - \sqrt{3}$$

Запишем найденные точки в переменные `x1` и `x2` соотвественно: `> x1 := evalf(xs[1]); x2 := evalf(xs[2]);`

$$x1 := -0.267949192$$
$$x2 := -3.732050808$$

Для определения, какая из точек является точкой минимума, а какая точкой максимума, вычислим вторую производную:

```
> ddy := simplify(diff(y, x, x));
```

$$dy := \mathrm{e}^x(x^2 + 6x + 5)$$

Найдем значение второй производной в точках `x1` и `x2` с помощью функции `subs`:

```
> evalf(subs(x = x1, ddy));
```

$$2.649852912$$

```
> evalf(subs(x = x2, ddy));
```

$$-0.08294334596$$

Поскольку вторая производная в точке `x1` положительна, то `x1` — точка минимума. В точке `x2` вторая производная отрицательна, следовательно, `x2` — точка максимума.

**2.** Найти площадь фигуры, ограниченной кривыми, приведенными на рис. 5.

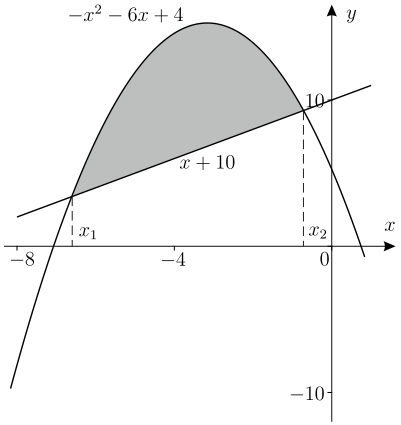



Рис. 5. Фигура, площадь которой требуется вычислить

Для начала создадим две переменные `y1` и `y2`, в которые запишем функции, ограничивающие фигуру:

```
> restart;
> y1 := x + 10;
```

$$y1 := x + 10$$

```
> y2 := -x^2 - 6*x + 4;
```

$$y2 := -x^2 - 6x + 4$$

Определим точки пересечения этих функций с помощью функции `solve`:

```
> xs := solve(y1 = y2, x);
```

$$xs := -6, -1$$

```
> x1 := xs[1]; x2 := xs[2];
```

$$x1 := -6$$
$$x2 := -1$$

Зная координаты $x$ точек пересечения, составим кратный интеграл для вычисления площади фигуры:

```
> S := int(1, [y = y1..y2, x = x1..x2]);
```

$$S := \frac{125}{6}$$

**3.** Проанализировать на сходимость ряд $\sum\limits_{n=1}^{\infty} \left(\frac{6n+1}{7n+5}\right)^n$.

Прежде чем переходить к провереке признаков сходимости, попробуем вычислить сумму с помощью функции `sum` (если сумма вычисляется, то соотвественно ряд сходящийся):

```
> restart;
> sum(((6*n + 1)/(7*n + 5))^n, n = 1..infinity);
```

$$\sum_{n=1}^{\infty} \left(\frac{6n+1}{7n+5}\right)^n$$

Видно, что Maple не может вычислить сумму этого ряда. Проверим необходимое условие сходимости ряда $\lim\limits_{n\to\infty} a_n = 0$, где $a_n$ — общий член ряда:

```
> an := ((6*n + 1)/(7*n + 5))^n;
```

$$an := \left(\frac{6n+1}{7n+5}\right)^n$$

```
> limit(an, n = infinity);
```

$$0$$

Поскольку необходимое условие сходимости выполняется, воспользуемся признаком Коши ($\lim\limits_{n\to\infty} \sqrt[n]{a_n} = D$):

```
> limit(an^(1/n), n = infinity);
```

$$\frac{6}{7}$$

Получается, что $D = 6/7 < 1$, следовательно, по признаку Коши ряд сходится.

## Задачи

3.1. Вычислить следующие интегралы:

$$\int \frac{dx}{\sin^2\left(2x+\frac{\pi}{4}\right)}, \quad \int \sqrt{\frac{x-a}{x+a}}dx \text{ (считая что } a<x)$$

$$\int\limits_{-1}^{8} \sqrt[3]{x}dx, \quad \int\limits_{1}^{2} dx \int\limits_{x}^{2} xydy$$

3.2. Найти производные:

$$\frac{d}{dx}(\cos^2 x + \operatorname{tg}\ln x^2), \quad \frac{\partial^2}{\partial x \partial y}(x \operatorname{ch} y - y^2 \operatorname{arctg}\sqrt{x})$$

3.3. Разложить в ряд Тейлора функцию:

$$f(x) = \cos x, \text{ в точке } x = 0$$

3.4. Найти следующие пределы:

$$\lim_{x\to\infty} \frac{5x+\sin x}{4x-3\cos x}, \quad \lim_{a\to 0} \frac{5a+8}{4a-1}$$

3.5. Найти корни следующих уравнений и проверить их:

$$x^2 + 14x - 15 = 0,$$
$$\operatorname{ctg} y = -2y, \quad y > 2\pi$$

3.6. Найти решение системы

$$\begin{cases} 4x + y - 2z = -3 \\ 6x + 3y - z = 4 \\ 8x + 7y + 5z = 34 \end{cases}$$

3.7. Вычислить следующие интегралы с использованием теоремы о вычетах:

$$\oint\limits_{L} \frac{dz}{(z+2)^2(z^2+1)}, \quad L : |z+2| = 2,$$

$$\oint\limits_{L} \frac{\sin z dz}{z^2 - z}, \quad L : |z| = 3, \quad \oint\limits_{L} \frac{dz}{4+z^2}, \quad L : |z| = 3.$$

# 4. Графика в Maple

## 4.1. Двумерные графики

### 4.1.1. Явно и параметрически заданные кривые

**1.** Для построения одномерных **явных функций**, то есть функций вида $y = f(x)$, используется функция/команда `plot`. Синтаксис этой команды для отрисовки графика одной функции следующий:

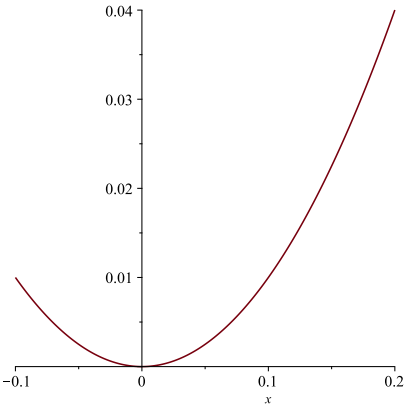
```
> plot(f(x), x = a..b, options);
```

Здесь `f(x)` — явная функция, зависящая от `x`, `x = a..b` — имя независимой переменной `x` и диапазон значений этой переменной `a..b`, на котором строится график, `options` — дополнительные опции для описания параметров отрисовки кривой (цвет, ширина и т. д.).

Если диапазон переменной `x` не будет указан явно, Maple по умолчанию построит график в диапазоне $[-10, 10]$, если функция `f(x)` не содержит тригонометрических функции, и в диапазоне $[-2\pi, 2\pi]$, если в `f(x)` есть тригонометрические функции.

Пример построения графика функции $y = \operatorname{sh} x^2$ в диапазоне $x \in [-0.1, 0.2]$ без дополнительных опций:

```
> plot(sinh(x^2), x = -0.1..0.2);
```



По умолчанию Maple рисует график функции темно-красной непрерывной линией, а также подписывает горизонтальную ось именем переменной, от которой строится график (в примере $x$).

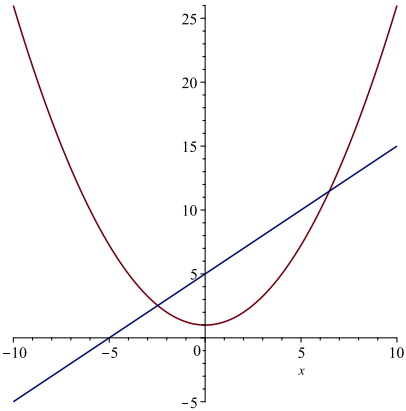
Для построения нескольких явно заданных функций, например, двух $y = f(x)$, $y = g(x)$ на одном графике, также используется функция `plot`:

```
> plot([f(x), g(x)], x = a..b, options);
```

Здесь `[f(x), g(x)]` — список функций, зависящих от одной переменной `x`. Заметим, что данный список может состоять из любого количества явно заданных функций.

Пример построения функций $y = \frac{1}{4}x^2 + 1$, $y = x + 5$ на одном графике:

```
> plot([0.25 * x^2 + 1, x + 5], x);
```




По умолчанию Maple рисует график первой функции темно-красной непрерывной линией, второй функции — темно-синей непрерывной линией. Все последующие графики также будут различных цветов. Данный алгоритм Maple неудобен тем, что пользователь заранее не знает, какой цвет ставится в соответствие каждой указанной функции, поэтому удобно использовать дополнительные опции `plot`.

Рассмотрим наиболее часто используемые опции `plot`:

1) `color = n` — опция для задания цвета графиков, в качестве значения `n` может указываться название цвета (или список названий) на английском языке (black, red, green и др.). Названия базовых цветов в Maple можно посмотреть, вызвав следующую команду:

```
> ColorTools:-DisplayPalette("MapleV", aliases);
```

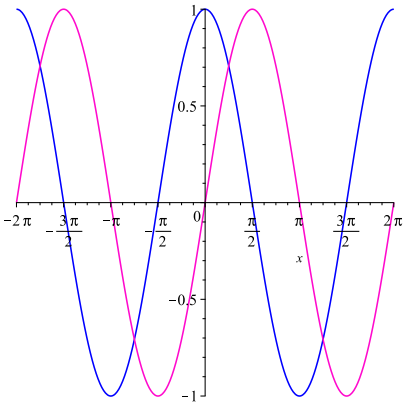
Также в качестве значения `n` может указываться RGB-код (или список RGB-кодов) с помощью следующей команды:

```
> ColorTools:-Color([R, G, B]);
```

где значения `R`, `G`, `B` есть целые числа в диапазоне от 0 до 255.

Пример отрисовки графиков функции $y = \cos x$ синим цветом (с использованием стандартной палитры Maple) и функции $y = \sin x$ (с RGB-кодом [255, 10, 205]):

```
> myColor := ColorTools:-Color([255, 10, 205]):
> plot([cos(x), sin(x)], x, color = [blue, myColor]);
```

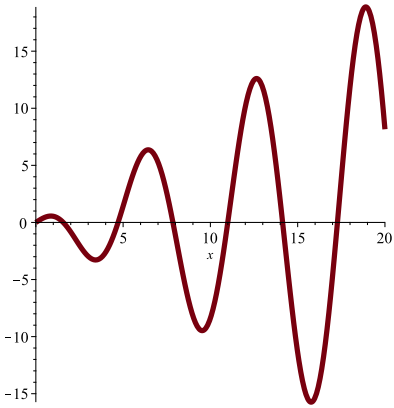



Цвета, указанные в опции `color`, применяются к указанным функциям по порядку, то есть к первой функции — первый цвет в списке цветов, ко второй функции — второй цвет и т. д.

2) `thickness = n` — опция для указания толщины линии графика функции. Значение `n` должно быть неотрицательным целым числом, при `n = 0` строится самая тонкая линия.

   Пример использования этой опции при отрисовке функции $y = x\cos x$:

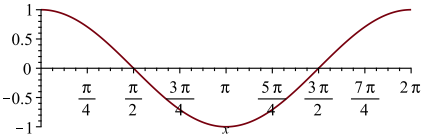
```
> plot(x*cos(x), x = 0..20, thickness = 5);
```




3) `scaling = s` — опция, управляющая масштабированием графика. Параметр `s` может иметь два значения: `constrained` (график выводится в масштабе 1:1) и `unconstrained` (по умолчанию). По умолчанию график масштабируется, чтобы соответствовать окну графика.

   Пример использования этой опции при построении функции $y = \cos x$:

```
> plot(cos(x), x = 0..2*Pi, scaling = constrained);
```



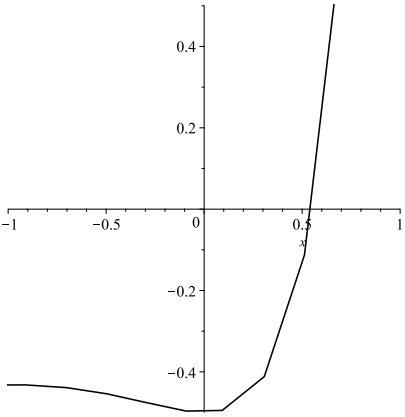
4) `coords = n` — опция для определения системы координат, в которой строится график или графики. Наиболее часто задаваемые значения `n`: `elliptic` (эллиптическая), `hyperbolic` (гиперболическая), `logarithmic` (логарифмическая), `polar` (полярная), также можно «создать» свою систему координат с помощью функции `addcoords`. Примеры использования этой опции читателю предлагается посмотреть самостоятельно в **Справке**.

5) `numpoints = n` — опция для задания количества точек в графике, по умолчанию `n = 200`. Эта опция полезна для «сглаживания» графика функции. Пример её использования будет представлен далее.

6) `view = [xmin..xmax, ymin..ymax]` — опция для ограничения графика по оси $Ox$ в пределах [`xmin`, `xmax`] и по оси $Oy$ в пределах [`ymin`, `ymax`].

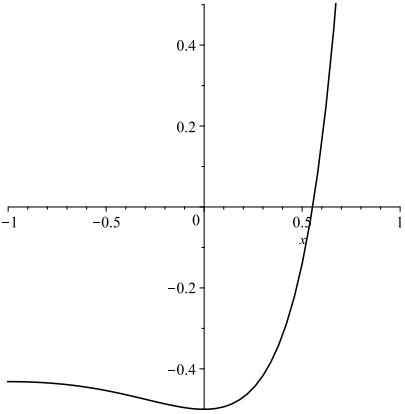
   Пример построения функции $y = \mathrm{ch}(xe^x) - \frac{3}{2}$ черным цветом в пределах $x \in [-1, 1]$, $y \in [-1, 1]$:

```
> plot(cosh(x*exp(x)) - 1.5, x, color = black,
       view = [-1..1,-0.5..0.5]);
```



Из рисунка видно, что построенный график не такой гладкий, как в предыдущих примерах, поэтому используем дополнительно опцию `numpoints`:

```
> plot(cosh(x*exp(x)) - 1.5, x, color = black,
       numpoints = 500, view = [-1..1,-0.5..0.5]);
```




**2.** Для построения графика **параметрически заданной** функции, то есть функции $x = x(t)$, $y = y(t)$, также используется команда `plot` со следующим синтаксисом (для построения одной функции):

```
> plot([x(t), y(t), t = a..b], options);
```

Здесь `x(t), y(t)` — параметрически заданная функция, зависящая от `t`, `t = a..b` — имя параметра `t` и его диапазон значений `a..b`, `options` — дополнительные опции для описания параметров кривой (цвет, ширина и т. д.).

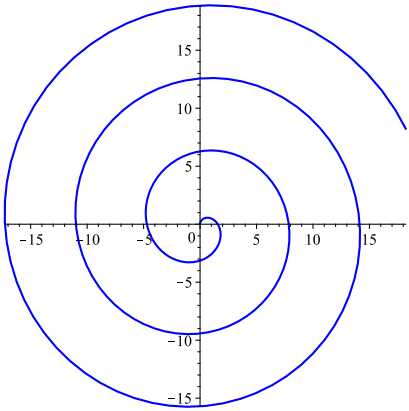
Следует обратить внимание на то, что выражения `x(t)`, `y(t)` заключаются в квадратные скобки **вместе** с параметром `t`. Также отметим, что диапазон параметра должен быть определен, иначе вместо параметрически заданной функции на одном графике будут нарисованы три функции `x(t)`, `y(t)` и `t`, явно зависящие от `t`.

Пример построения параметрически заданной функции

$$\begin{cases} x(t) = t\sin t \\ y(t) = t\cos t \end{cases}$$

синим цветом:

```
> plot([t*sin(t), t*cos(t), t = 0..20],
                    color = blue, thickness = 2);
```




Для построения нескольких параметрически заданных функций (например, $x_1 = x_1(t)$, $y_1 = y_1(t)$ и $x_2 = x_2(t)$, $y_2 = y_2(t)$) на одном графике используется следующий синтаксис команды `plot`:

```
> plot([[x1(t), y1(t), t = a..b],
        [x2(t), y2(t), t = c..d]], options);
```

Здесь `[x1(t), y1(t), t = a..b]` — задание одной параметрической функции, `[x2(t), y2(t), t = c..d]` — задание второй параметрической функции. Количество параметрически заданных функций на одном графике не ограничено.

Дополнительные опции `options` команды `plot` для построения параметрически заданных функций не отличаются от опций, описанных выше для построения явных функций.

### 4.1.2. Кривые, заданные неявно

Функции построения графиков **неявно заданных** кривых, то есть функций вида $f(x, y) = C$, в Maple хранятся в пакете `plots`. Для подключения этого (и любого другого) пакета используется ключевое слово `with`, после которого в скобках указывается имя подключаемого пакета. Пример подключения пакета для графики:

```
> with(plots):
```

Если вместо «:» поставить «;», то будут выведены имена всех функций/команд, содержащихся в этом пакете. Для построения неявных функции рассмотрим команды `implicitplot` и `contourplot` из пакета `plots`.

**1.** Функция `implicitplot` используется для построения неявно заданных функций, которые заданы **уравнением**, и имеет следующий синтаксис (для построения одной функции):

```
> implicitplot(f(x, y) = C, x = a..b, y = c..d, opts);
```

Здесь `f(x, y) = C` — выражение, задающее неявную кривую, `x = a..b, y = c..d` — имена переменных `x`, `y`, от которых зависит функция и их диапазоны значений `a..b`, `c..d`, `opts` — дополнительные опции. Отметим, что указание диапазонов значений аргументов для функции `implicitplot` обязательно, в отличие от функции `plot`, в противном случае Maple выведет сообщение об ошибке.

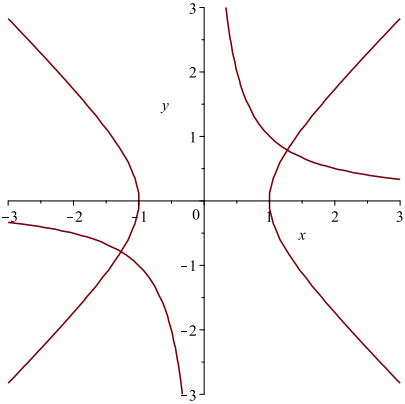
Для построения нескольких неявно заданных функций (например, $f(x, y) = C_1$, $g(x, y) = C_2$) на одном графике используется команда `implicitplot` со следующим синтаксисом:

```
> implicitplot([f(x, y) = C1, g(x, y) = C2],
                                  x = a..b, y = c..d, opts);
```

Здесь `[f(x, y) = C1, g(x, y) = C2]` — список неявно заданных функций, который в общем случае может содержать любое количество функций.

Пример построения неявных функций $x^2 - y^2 = 1, xy = 1$ на одном графике:

```
> implicitplot([x^2 - y^2 = 1, x*y = 1],
                                x = -3..3, y = -3..3);
```




Так же, как и для функции `plot`, по умолчанию Maple строит линию темно-красным цветом и подписывает оси именами переменных, которые были указаны (в примере $x$ и $y$). Однако в отличие от `plot` при отрисовке нескольких графиков функция `implicitplot` не рисует их разными цветами.

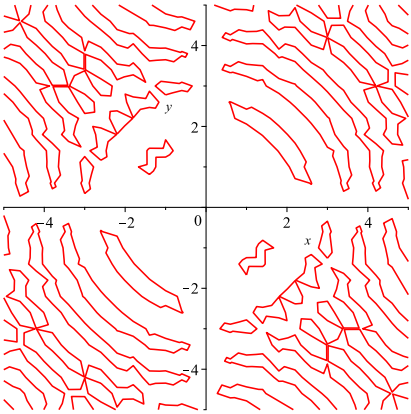
Дополнительные опции `opts` функции/команды `implicitplot` совпадают с опциями для функции `plot`, за исключением опции `numpoints`. Помимо этих опций функция `implicitplot` имеет собственные опции, например:

1) `grid = [m, n]` — опция для указания размеров сетки, на которой строится график. По умолчанию для построения графиков используется сетка 26 на 26.

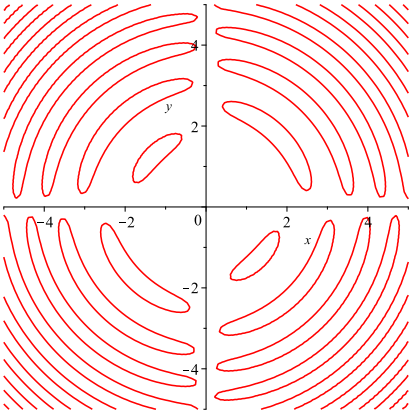
   Также следует отметить связанную с `grid` опцию `gridrefine = k`, где `k` связан с размерами сетки следующим образом `[2^k*26, 2^k*26]` (по умолчанию `k = 0`).

   Пример построения функции $xy\cos(x^2 + y^2) = 1$ с опцией `grid` и без нее красным цветом:

```
> implicitplot(x*y*cos(x^2 + y^2) = 1, x = -5..5,
                          y = -5..5, color = red);
```



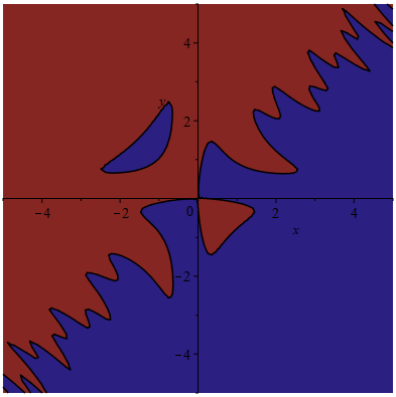

```
> implicitplot(x*y*cos(x^2 + y^2) = 1, x = -5..5,
                y = -5..5, grid = [100, 100], color = red);
```




2) `filledregions = s` — опция, определяющая, будет график закрашен или нет. Параметр `s` имеет только два возможных значения: `true` и `false` (по умолчанию). Пример использования этой опции при построении кривой

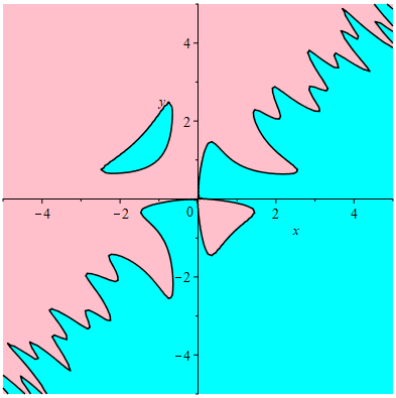
$$x - y + \sin(2.5xy) = \sin x - \sin y + \sin(xy) :$$

```
> implicitplot(x-y+sin(2.5*x*y)=sin(x)-sin(y)+sin(x*y),
              x = -5..5, y = -5..5, grid = [100, 100],
              filledregions = true);
```




Для задания цветов областей используется опция `coloring = [color1, color2, ...]`, где `[color1,...]` — список из названий цветов.

```
> implicitplot(x-y+sin(2.5*x*y)=sin(x)-sin(y)+sin(x*y),
       x = -5..5, y = -5..5, grid = [100, 100],
       filledregions = true, coloring = [pink, cyan]);
```



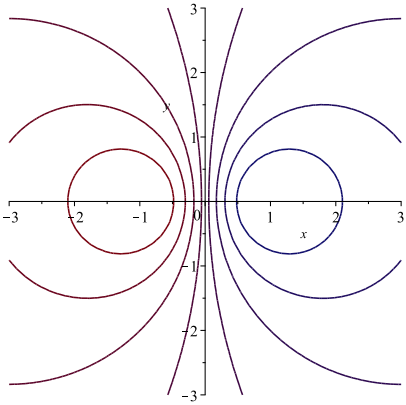
**2.** Функция `contourplot` используется для построения последовательности неявно заданных функций вида $f(x, y) = C$, где $C$ принимает некоторый набор значений, другими словами, команда `contourplot` строит линии уровня явной функции двух переменных $z = f(x, y)$. Синтаксис этой функции/команды совпадает с синтаксисом функции `implicitplot`:

```
> contourplot(f(x, y), x = a..b, y = c..d, opts);
```

Здесь `f(x, y)` — неявно заданная функция, `x = a..b, y = c..d` — имена переменных `x`, `y`, от которых зависит функция, и их диапазоны значений `a..b`, `c..d`, `opts` — дополнительные опции.

Пример построения линий уровня функции $\dfrac{2x}{x^2 + y^2 + 1}$:

```
> contourplot(2*x/(x^2+y^2+1), x = -3..3, y = -3..3);
```




Для построения линий уровня нескольких функций (например, функции $f(x, y)$, $g(x, y)$), на одном графике используется команда `contourplot` со следующим синтаксисом:

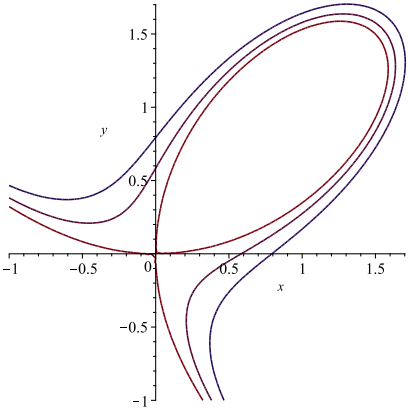
```
> contourplot({f(x, y), g(x, y)},
                          x = a..b, y = c..d, opts);
```

Здесь `{f(x, y), g(x, y)}` — список неявно заданных функций, линии уровня которых будут построены. Следует обратить внимание, что список заключается в фигурные скобки `{}`, а не в квадратные `[]`, как во всех вышеописанных случаях.

Дополнительные опции `opts` для команды `contourplot` совпадают с опциями функции `impliciplot`. Однако, функция `contourplot` имеет собственную дополнительную опцию `contours = s`, отвечающую за количество линий уровня на графике. Параметр `s` может быть целым числом, указывающим количество линий, либо списком значении, явно задающих уровни функции.

Пример построения конкретных уровней функции $3xy - x^3 - y^3 = [0, -0.2, -0.5]$:

```
> contourplot(3*x*y - x^3 - y^3, x = -1..3, y = -1..3,
        contours = [0, -0.2, -0.5], grid = [100, 100]);
```




### 4.1.3. Графики по точкам

Ниже рассмотренные функции располагаются в пакете `plots` (если не оговорено обратное), поэтому прежде следует подключить этот пакет:

```
> with(plots):
```

**1.** Для построения точек на плоскости в Maple используется команда `pointplot` со следующим синтаксисом:

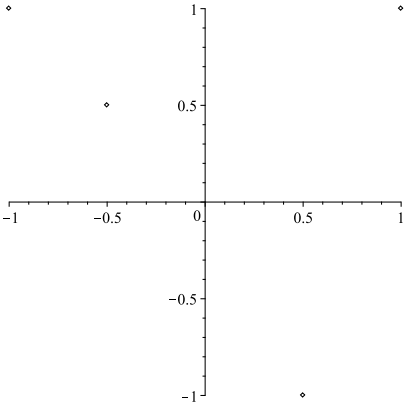
```
> pointplot(L, options);
```

Здесь `L` — список точек, то есть список пар $(x, y)$, задающих координаты каждой точки, `opts` — дополнительные опции. Список

точек может быть прописан в ручную `L = [[x1, y1], ...]` или с помощью функции `seq`, если координаты можно представить в виде некоторой последовательности `L = [seq([xn, yn], n)]`.

Пример построения на плоскости точек с координатами $(-1, 1)$, $(-0.5, 0.5)$, $(0.5, -1)$, $(1, 1)$:

```
> pointplot([[-1,1], [-0.5,0.5], [0.5,-1], [1,1]]);
```



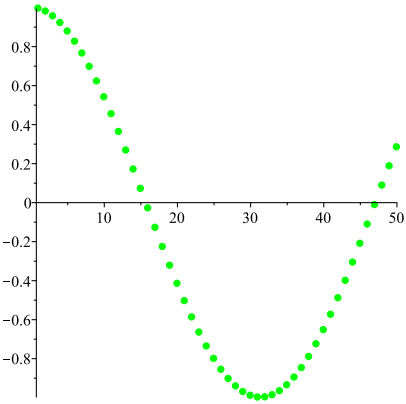

Для функции `pointplot` в качестве дополнительных опций могут указываться цвет точек (`color`), система координат (`coords`), ограничение по осям (`view`), масштабирование осей (`scaling`) — все эти опции описаны выше в разделе 4.1.1, также в опциях `options` можно указать тип маркера (`symbol`), которым изображается точка, и его размер (`symbolsize`). Возможные значения опции `symbol`:

- `asterisk` — символ звездочки;
- `box` — незакрашенный квадрат;
- `circle` — незакрашенная окружность;
- `cross` — прямой крест;
- `diagonalcross` — повернутый крест;
- `diamond` — незакрашенный ромб (значение по умолчанию);

- `point` — точка;
- `solidbox` — закрашенный квадрат;
- `solidcircle` — закрашенная окружность;
- `soliddiamond` — закрашенный ромб.

Пример построения на плоскости 50 точек, координаты которых заданы формулой $(x_n, y_n) = (n, \cos\frac{n}{10})$, зеленными кружками:

```
> pointplot([seq([n, cos(0.1*n)],n = 1..50)],
            symbol = solidcircle, symbolsize = 15,
            color = green);
```




**2.** Для построения непрерывной линии по точкам в Maple реализованы функции `listplot` и `dataplot` (базовая функция, не требующая подключения дополнительных пакетов). Синтаксис этих комманд следующий:

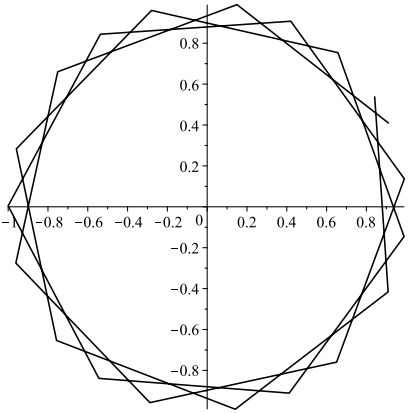
```
> listplot(L, options);
> dataplot(Lx, Ly, options);
```

Здесь `L` — список точек, то есть список пар $(x,\ y)$, задающих координаты каждой точки, `Lx`, `Ly` — списки координат $x$ и $y$ всех точек соответственно, `opts` — дополнительные опции. Списки точек, как и для функции `pointplot`, могут быть прописаны

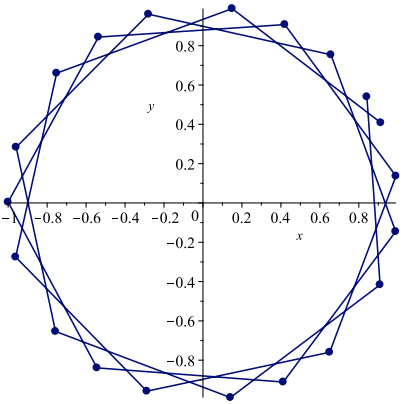
в ручную или с помощью функции `seq`. Дополнительные опции `opts` такие же, как и для функций `plot` и `pointplot`.

Пример построения линии по 20 точкам, координаты которых задаются последовательностями $(x_n, y_n) = (\sin n, \cos n)$:

```
> listplot([seq([sin(n), cos(n)], n = 1..20)]);
```




```
> dataplot([seq(sin(n), n = 1..20)],
                        [seq(cos(n), n = 1..20)]);
```



### 4.1.4. Другие функции

**1.** Функция для совмещения нескольких графиков на одном рисунке называется `display` и имеет следующий синтаксис:

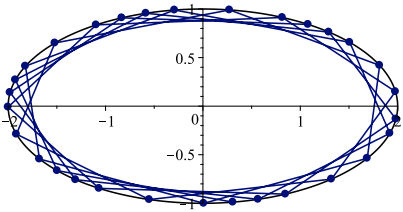
```
> display(A, B, C, ..., opts);
```

Здесь `A, B, C, ...` — графики разных функций (например, явных, неявных и т. д.), `opts` — дополнительные опции, которые читателю предлагается изучить самостоятельно.

Данная функция оказывается полезна на практике, поскольку при решении математических и физических задач часто возникает необходимость наложить друг на друга графики, полученные, например, из формулы и из данных эксперимента.

Пример построения эллипса из параметрического задания кривой и по 30 точкам:

```
> A := plot([2*cos(x), sin(x), x = 0..2*Pi],
                    color = black,
                    scaling = constrained):
> B := dataplot([seq(2*cos(n), n = 1..30)],
                    [seq(sin(n), n = 1..30)]):
> display(A, B);
```



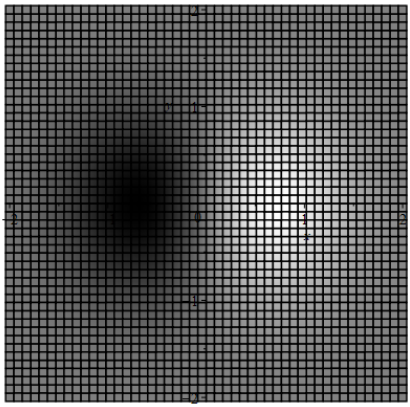
**2.** Функция `densityplot` в Maple для построения распределения по заданной функции двух переменных (заливка по цветам между линиями уровня неявной функции) и имеет следующий синтаксис:

```
> densityplot(f(x, y), x = a..b, y = c..d, opts);
```

Здесь `f(x, y)` — неявная функция, `x = a..b, y = c..d` — имена переменных `x`, `y`, от которых зависит функция, и их диапазоны значений `a..b`, `c..d`, `opts` — дополнительные опции.

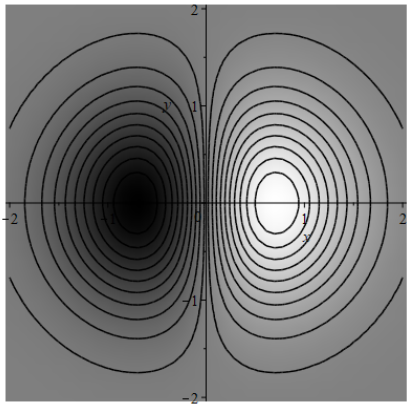
По умолчанию `densityplot` строит цветовое распределение в черно-белой гамме. Пример построения для функции $xe^{-x^2-y^2}$:

```
> densityplot(x*exp(-x^2 - y^2),
                    x = -2..2, y = -2..2);
```



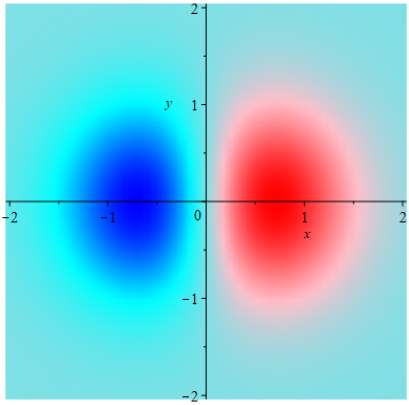
Для отключения сетки воспользуемся опцией `style=patchnogrid` и наложим на это распределение линии уровня:

```
> A := densityplot(x*exp(-x^2 - y^2),
                   x = -2..2, y = -2..2,
                   style = patchnogrid);
> B := contourplot(x*exp(-x^2 - y^2),
                   x = -2..2, y = -2..2,
                   contours = 20, color = black);
> display(A, B);
```




Для задания другой цветовой палитры используется опция `colorscheme = [color1, ..., colorN]`, в которой указывается список цветов, определяющих цветовую гамму (от наименьшего значения функции к наибольшему):

```
> densityplot(x*exp(-x^2 - y^2), x = -2..2, y = -2..2,
      style = patchnogrid,
      colorscheme = [blue, cyan, pink, red]);
```




Данная функция оказывается полезной при решении задач математической физики на двумерных поверхностях, а именно

при отрисовке распределения температуры или распределения модуля скорости на плоской поверхности.

**3.** Функция `fieldplot` используется для отрисовки двумерного векторного поля, синтаксис этой функции следующий:

```
> fieldplot([fx(x, y), fy(x, y)],
                          x = a..b, y = c..d, opts);
```

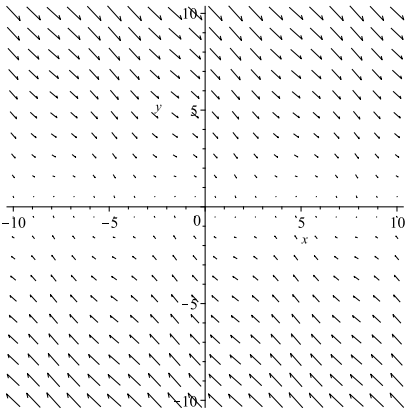
Здесь `[fx(x, y), fy(x, y)]` — компоненты векторного поля, `x = a..b, y = c..d` — имена переменных `x`, `y`, от которых зависят компоненты, и их диапазоны значений `a..b`, `c..d`, `opts` — дополнительные опции.

Отметим опции, которые существуют только для функции/команды `fieldplot`:

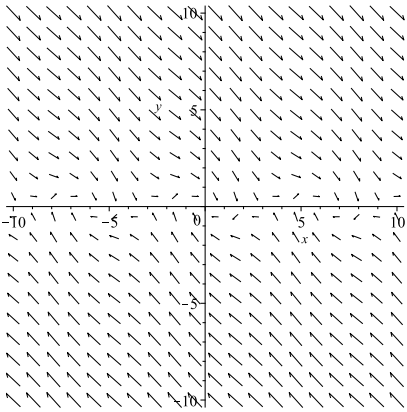
1) `arrows = s` — опция, определяющая тип стрелочек, которыми изображается векторное поле. Возможные значения параметра `s`: `LINE`, `THICK`, `SLIM` и `THIN` (по умолчанию).
2) `fieldstrength = s` — опция, указывающая длину стрелочки при отрисовке векторного поля. Возможные значения параметра `s`: `maximal` (по умолчанию), `average`, `fixed`, `log`, `scaleto(n)` (где `n` — целое число, определяющее масштаб).

Пример построения векторного поля, компоненты которого задаются выражениями $(y, \sin x - y)$:

```
> fieldplot([y, -sin(x)-y], x = -10..10, y = -10..10);
```



```
> fieldplot([y, -sin(x)-y], x = -10..10,
                    y = -10..10, fieldstrength = log);
```




## 4.2. Трехмерные графики

### 4.2.1. Явно и неявно заданные поверхности

**1.** Для построения графиков **явно заданных функций** двух переменных, то есть функций вида $z = f(x, y)$, используется функция/команда `plot3d`. Синтаксис этой команды для отрисовки одной функции следующий:

```
> plot3d(f(x, y), x = a..b, y = c..d, opts);
```

Здесь `f(x, y)` — явная функция, зависящая от переменных `x, y`, `x = a..b, y = c..d` — имена независимых переменных `x, y` и их диапазоны значений `a..b,c..d` соответственно, `opts` — дополнительные опции.

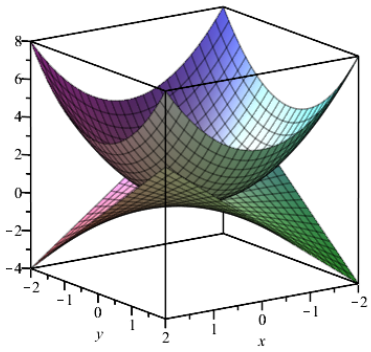
Для отрисовки нескольких явно заданных поверхностей, например, $z = f(x, y)$ и $z = g(x, y)$ также используется команда `plot3d` со следующим синтаксисом:

```
> plot3d([f(x, y), g(x, y)], x = a..b, y = c..d, opts);
```

Здесь `[f(x, y), g(x, y)]` — список явно заданных функций, в общем случае состоящий из любого количества функций.

Пример построения функций $xy$, $x^2 + y^2$ на одном графике:

```
> plot3d([(x*y), (x^2+y^2)], x = -2..2, y = -2..2);
```




По умолчанию Maple строит поверхность в своей базовой палитре цветов и подписывает оси именами переменных, указанных при вызове `plot3d`, в примере $x$ и $y$.

Часто используемые дополнительные опции, такие как `color` (цвет поверхности), `coords` (задание системы координат), `view` (ограничение графика по осям), `scaling` (масштабирование осей), `grid` (задание узлов сетки), совпадают с опциями, описанными в разделе двумерной графики 4.1. Однако помимо этих опций существуют опции, определенные только для `plot3d`, рассмотрим некоторые из них:

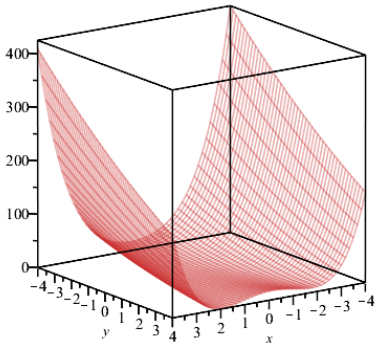
1) `style = s` — опция, определяющая способ отображения поверхности. Параметр `s` может иметь следующие значения:

   - `surface` — поверхность без сетки;
   - `surfacewireframe` — поверхность с сеткой на ней (по умолчанию);
   - `contour` — линии уровня в пространстве;
   - `surfacecontour` — поверхность с линиями уровня на ней;

- `wireframe` — отрисовка поверхности мелкой сеткой;
- `wireframeopaque` — отрисовка поверхности крупной сеткой;
- `point` — отрисовка поверхности точками;
- `pointline` — отрисовка поверхности мелкой сеткой с точками на ней.

Пример построения графика функции $z = (1-x)^2+(y-x^2)^2$ мелкой сеткой оранжевого цвета:

```
> plot3d((1-x)^2+(y-x^2)^2, x = -4..4, y = -4..4,
                        style = wireframe, color = orange);
```



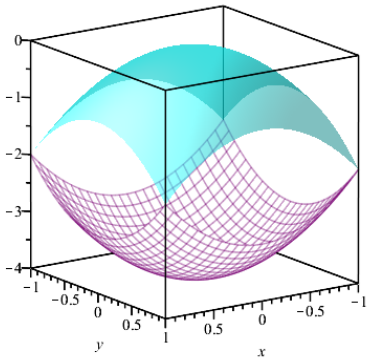

2) `lightmodel = s` — опция, определяющая тип освещения графика. Возможные значения параметра `s`: `none`, `light1`, `light2`, `light3` (по умолчанию), `light4`. Читателю предлагается самостоятельно изучить работу данной опции.

3) `transparency = t` — опция для указания прозрачности поверхности. Возможные значения параметра `t` лежат в пределах от 0 до 1, где 0 соответствует абсолютно непрозрачной поверхности, а 1 — абсолютно прозрачной.

   Пример построения графика функции $z = -x^2 - y^2$ полупрозрачной голубой поверхностью без сетки и функции

$z = x^2 + y^2 - 4$ фиолетовой крупной сеткой на одном графике:

```
> A := plot3d(-x^2-y^2, x = -1..1, y = -1..1,
        style = surface, color = cyan,
        transparency = 0.5):
> B := plot3d(x^2 + y^2 - 4, x = -1..1, y = -1..1,
        style = wireframeopaque, color = purple):
> with(plots):
> display(A, B);
```




4) `labels = [Name_x, Name_y, Name_z]` — опция для подписи осей. Ось $Ox$ будет подписана как `Name_x`, $Oy$ — `Name_y`, $Oz$ — `Name_z`.

Другие не упомянутые дополнительные опции можно найти в Справке во вкладке `plot3d/option`.

**2.** Помимо функции `plot3d` для построения явно заданной поверхности может использоваться функция/команда `contourplot3d`, которая требует подключения пакета `plots` и имеет следующий синтаксис (совпадает с синтаксисом команды `plot3d`):

```
> contourplot3d(f(x, y), x = a..b, y = c..d, opts);
```

Здесь `f(x, y)` — явная функция, зависящая от переменных `x, y`, `x = a..b, y = c..d` — имена независимых переменных `x, y` и их диапазоны значений `a..b, c..d` соответственно, `opts` — дополнительные опции.

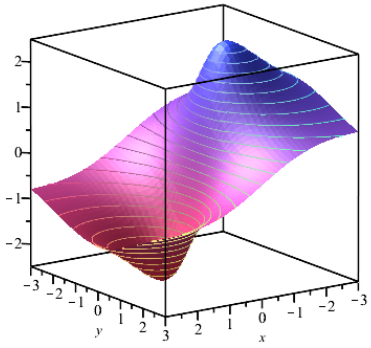
По умолчанию команда `contourplot3d` строит линии уровня заданной функции в пространстве. Для построения поверхности необходимо использовать дополнительную опцию `filledregions = true`, которая подробно описана в разделе 4.1.2 (также из этого раздела могут быть полезны опция `coloring` и опции команды `contourplot`).

Пример построения явно заданной функции

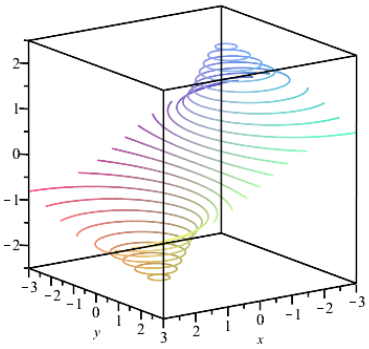
$$z = -\frac{5x}{x^2 + y^2 + 1}$$

с использованием опции `filledregions` и без нее:

```
> with(plots):
> contourplot3d(-5*x/(x^2 + y^2 + 1), x = -3..3,
                y = -3..3, contours = 25,
                filledregions = true);
```




```
> contourplot3d(-5*x/(x^2 + y^2 + 1), x = -3..3,
                y = -3..3, contours = 25);
```




**3.** Для построения неявно заданных поверхностей (то есть функции вида $f(x, y, z) = C$) в Maple используется команда/функция `implicitplot3d`, расположенная в пакете `plots` и имеющая следующий синтаксис:

```
> implicitplot3d(f(x, y, z) = C,
                 x = a..b, y = c..d, z = p..q, opts);
```

Здесь `f(x, y, z) = C` — уравнение неявно заданной функции, зависящей от переменных `x, y, z`, `x = a..b, y = c..d, z = p..q` — имена независимых переменных `x, y, z` и их диапазоны значений `a..b, c..d, p..q` соответственно, `opts` — дополнительные опции.

Для построения нескольких неявно заданных функций, например, $f(x, y, z) = C_1$ и $g(x, y, z) = C_2$, также используется команда `implicitplot3d` со следующим синтаксисом:

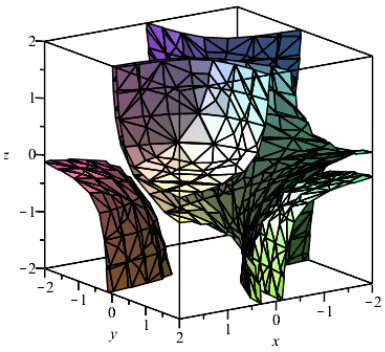
```
> implicitplot3d([f(x, y, z) = C1, g(x, y, z) = C2],
                 x = a..b, y = c..d, z = p..q, opts);
```

Здесь `[f(x, y, z) = C1, g(x, y, z) = C2]` — список неявно заданных функций (может состоять из большего количества функций).

Пример построения функций $xyz = \frac{1}{2}$, $y = e^{-xz}$ без использования дополнительных опций:

```
> with(plots):
```

```
> implicitplot3d([x*y*z = 1/2, y = exp(-x*z)],
                 x = -2..2, y = -2..2, z = -2..2);
```



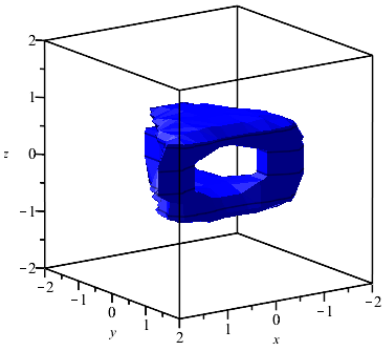

Дополнительные опции `opts` функции `implicitplot3d` совпадают с опциями, описанными для функции/команды `plot3d`. Однако все равно отметим опцию `grid = [m, n, k]`, задающую сетку, по которой строится поверхность. Эта опция наиболее часто используются при построении поверхностей, заданных неявно, поскольку по умолчанию сетка имеет размеры [10, 10, 10] и поверхность строится достаточно грубо.

Пример построения поверхности 2-го рода, задаваемой уравнением

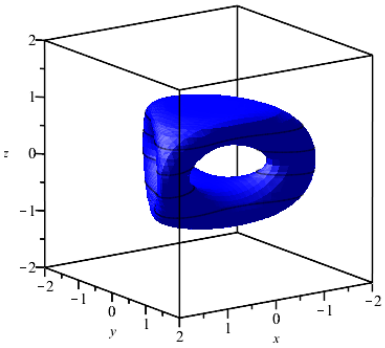
$$2y(y^2 - 3x^2)(1 - z^2) + (x^2 + y^2)^2 - (9z^2 - 1)(1 - z^2) = 0$$

без использования опции `grid` и вместе с ней:

```
> with(plots):
> implicitplot3d(2*y*(y^2 - 3*x^2)*(1 - z^2)
     + (x^2 + y^2)^2 - (9*z^2 - 1)*(1 - z^2),
     x = -2..2, y = -2..2, z = -2..2,
     color = blue,
     style = surfacecontour);
```




```
> implicitplot3d(2*y*(y^2 - 3*x^2)*(1 - z^2)
     + (x^2 + y^2)^2 - (9*z^2 - 1)*(1 - z^2),
     x = -2..2, y = -2..2, z = -2..2,
     color = blue, grid = [40, 40, 40],
     style = surfacecontour);
```



### 4.2.2. Кривые в пространстве

Ниже рассмотренные функции располагаются в пакете `plots`, поэтому сначала следует подключить данный пакет:

```
> with(plots):
```

**1.** Функция `spacecurve` используется для построения параметрически заданной кривой или нескольких кривых в пространстве и имеет следующий синтаксис:

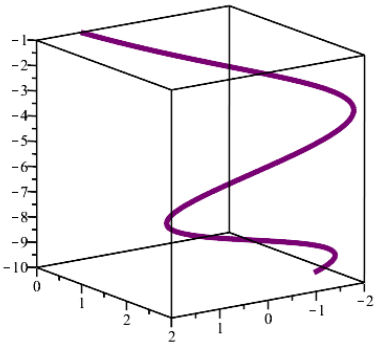
```
> spacecurve([x(t), y(t), z(t)], t = t1..t2, opts);
```

Здесь `[x(t), y(t), z(t)]` — явные зависимости координат `x, y, z` от параметра `t`, `t = t1..t2` — диапазон значений параметра `t`, `opts` — дополнительные опции, полностью совпадающие с опциями для функции `plot`.

Пример построения кривой в пространстве, задаваемой следующим образом:

$$\begin{cases} x(t) = 2\cos t, \\ y(t) = 3\sin(\ln t), \\ z(t) = -t. \end{cases}$$

```
> spacecurve([2*cos(t), 3*sin(ln(t)), -t], t = 1..10,
                color = purple, thickness = 5);
```




Для построения нескольких кривых в пространстве используется функция/команда `spacecurve` со следующим синтаксисом:

```
> spacecurve([[x1(t), y1(t), z1(t)],
          [x2(t), y2(t), z2(t)], ...], t = t1..t2, opts);
```

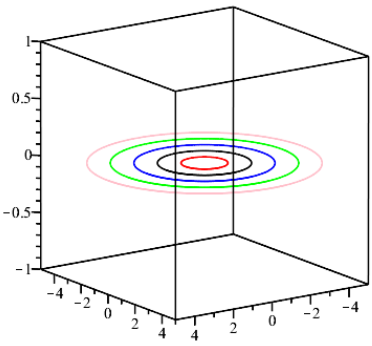
Здесь `[[x1(t), y1(t), z1(t)], [x2(t), y2(t), z2(t)], ...]` — список параметрически заданных кривых в пространстве.

Пример построения семейства кривых

$$\begin{cases} x(t) = n\cos t, \\ y(t) = n\sin t, \\ z(t) = 0, \end{cases} \qquad n = 1, 2, 3, 4, 5.$$

на одном графике разными цветами:

```
> spacecurve([seq([n*cos(t), n*sin(t), 0], n = 1..5)],
    t = 0..8, color = [red, black, blue, green, pink]);
```




Отметим, что похожей на функцию `spacecurve` является функция `tubeplot`, с которой читателю предлагается познакомиться самостоятельно.

**2.** Кривая в пространстве может задаваться как линия пересечения поверхностей, для построения таких кривых используется функция `intersectplot` со следующим синтаксисом:

```
> intersectplot(surf1, surf2, opts);
```

Здесь `surf1, surf2` — поверхности, которые могут быть заданы явно, параметрически или неявно, `opts` — дополнительные опции для полученной кривой. Поверхности `surf1, surf2` задаются с помощью функции/команды `surface`:

- `surface(f(x,y), x=a..b, y=c..d)`, если поверхность задается явно;

- `surface([x(s,t), y(s,t), z(s,t)],s = a..b,t = c..d)`, если поверхность задана параметрически;

- `surface(F(x,y,z) = C, x = a..b, y = c..d, z = e..f)`, если поверхность задается неявно.

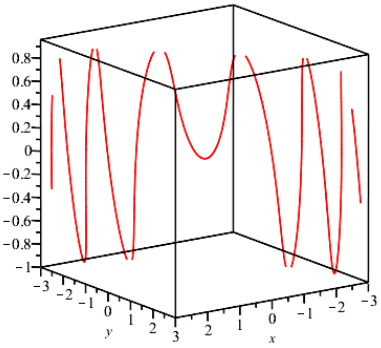
Отдельно выделим случай, когда обе поверхности заданы неявными функциями, тогда синтаксис функции `intersectplot` следующий:

```
> intersectplot(f(x, y, z) = C1, g(x, y, z) = C2,
                    x = a..b, y = c..d, z = p..q, opts);
```

Здесь `f(x, y, z) = C1, g(x, y, z) = C2` — неявные функции, зависящие от переменных `x, y, z`, `x = a..b`,`y = c..d`, `z = p..q` — имена независимых переменных `x, y, z` и их диапазоны значений `a..b, c..d, p..q` соответственно. Дополнительные опции `opts` совпадают с опциями функции `plot`.

Пример построения пересечения явных поверхностей $\sin(x^2+y^2)$ и $x+y$:

```
> intersectplot(surface(sin(x^2+y^2), x = -3..3,
      y = -3..3), surface(x+y, x = -3..3, y = -3..3));
```



Пример построения кривой как пересечения неявно заданных поверхностей: сферы $x^2 + y^2 + z^2 = 3$ и цилиндра $(y^2 + x^2) = 1$ (с отображением поверхностей):

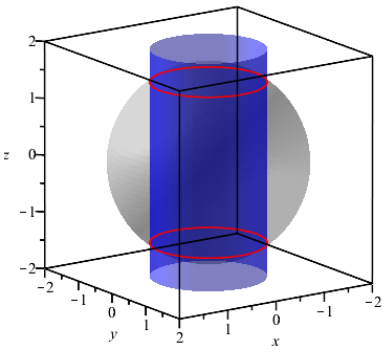
```
> A := x^2 + y^2 + z^2 = 3:
> B := y^2 + x^2 = 1:
> P1 := intersectplot(A, B, x = -2..2, y = -2..2,
                            z = -2..2, color = red):
> P2 := implicitplot3d(x^2 + y^2 + z^2 = 3, x = -2..2,
                       y = -2..2, z = -2..2, color = gray,
                       grid = [40,40,40], style = surface,
                       transparency = 0.5);
```

[*Length of output exceeds limit of* 1000000]

```
> P3 := implicitplot3d(y^2 + x^2 = 1, x = -2..2,
                       y = -2..2, z = -2..2, color = blue,
                       grid = [40,40,40], style = surface,
                       transparency = 0.5);
```

[*Length of output exceeds limit of* 1000000]

```
> display(P1, P2, P3);
```



### 4.2.3. Точки в пространстве

Ниже рассмотренная функция располагается в пакете `plots`, поэтому сначала следует подключить данный пакет:

```
> with(plots):
```

Для построения точек в трехмерном пространстве в Maple используется команда `pointplot3d`, синтаксис которой следующий:

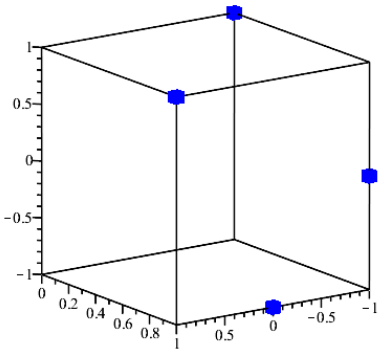
```
> pointplot3d(L, options);
```

Здесь `L` — список точек, то есть список координат $x$, $y$, $z$ для каждой точки, `opts` — дополнительные опции. Так же, как и для функции `pointplot`, список точек может быть прописан вручную `L = [[x1, y1, z1], [x2, y2, z2], ...]` или с помощью функции `seq`, если координаты могут представленны в виде некоторой последовательности `L = [seq([xn, yn, zn], n)]`.

Дополнительные опции `opts` полностью совпадают с опциями для функции построения точек на плоскости `pointplot`.

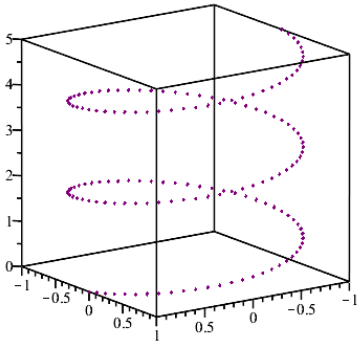
Пример построения точек с координатами $(1,1,1)$, $(-1,1,0)$, $(0,1,-1)$, $(-1,0,1)$ синими кубическими маркерами:

```
> pointplot3d([[1,1,1], [-1,1,0], [0,1,-1], [-1,0,1]],
              color = blue, symbol = solidbox,
              symbolsize = 25):
```




Пример построения точек в пространстве, координаты которых задаются последовательностями $x_n = \cos\frac{n\pi}{30}$, $y_n = \sin\frac{n\pi}{30}$, $z_n = \frac{n\pi}{30}$, $n = 1, \ldots, 150$.

```
> pointplot3d([seq([cos(Pi*n/30), sin(Pi*n/30), n/30],
                   n = 0 .. 150)], color = purple);
```




## 4.3. Анимация

**1.** Для построения анимации (на плоскости и в пространстве) используется команда `animate`, расположенная в пакете `plots`. Синтаксис этой команды следующий:

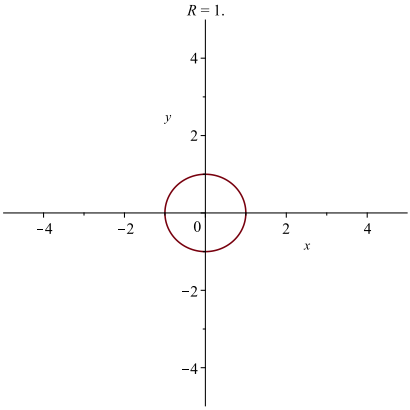
```
> animate(plotcommand, [plotargs], t = a..b, options);
```

Здесь `plotcommand` — функция/команда для построения графика анимируемой функции (то есть, если строится анимация явной функции, то используется функция `plot`, если строится анимация кривой в пространстве — функция `spacecurve`), `plotargs` — аргументы, необходимые функции/команде `plotcommand` для построения графика, также здесь могут указываться дополнительные опции для `plotcommand`, `t = a..b` — имя параметра анимации `t` и его диапазон значений `a..b`, `options` — дополнительные опции для команды `animate`.

Пример построения анимации неявной функции $x^2 + y^2 = R^2$, которая задает окружность на плоскости, по параметру $R \in [1, 5]$. Поскольку функция неявная, то команда для отрисовки графика — `implicitplot`, синтаксис этой функции/команды описан в разделе 4.1.2. Команда для построения такой анимации[8]:

[8] Анимации, описанные в данном методическом пособии, можно посмотреть, скачав файл ExampleAnimation.mw по ссылке: https://drive.google.com/file/d/1AuuFRrDZp49kpbieh_wqMVFpeG6Jx0Ux/view?usp=sharing

```
> animate(implicitplot, [x^2 + y^2 = R^2, x = -5..5,
                  y = -5..5, grid = [70, 70]], R = 1..5);
```




По умолчанию параметр анимации и его начальное значение указывается вверху графика (в примере, $R = 1.$). Для запуска анимации необходимо нажать левой кнопкой мыши на выведенный график, после чего под основной панелью инструментов появится дополнительное меню (см. рис. 6), на котором надо нажать на кнопку «play» (обозначается треугольником).

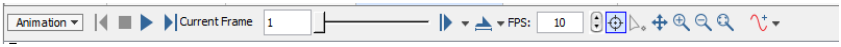



Рис. 6. Дополнительное меню анимации

Также построенную анимацию можно запустить, нажав на выведенный график правой кнопкой мыши и выбрав в выпавшем меню: **Animation→Play** (см. рис. 7).

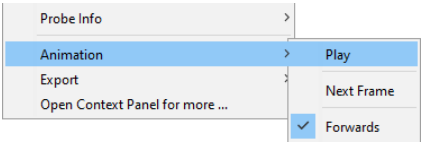
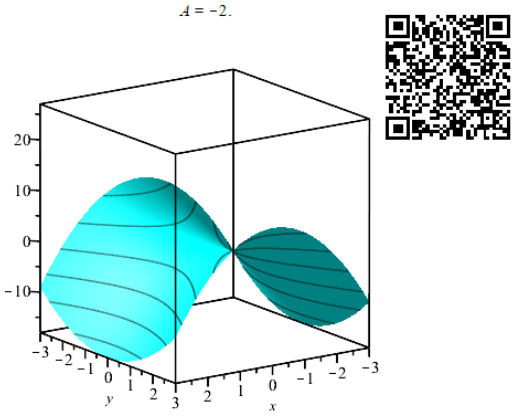



Рис. 7. Запуск анимации

Пример построения анимации явно заданной поверхности $z = Ax^2 + y^2$ по параметру $A \in [-2, 2]$. Поскольку функция явно задает поверхность, используем для отрисовки команду `plot3d` и построим поверхность голубым цветом:

```
> animate(plot3d, [A*x^2 + y^2, x = -3..3, y = -3..3,
        style = patchcontour, color = cyan], A = -2..2);
```




Рассмотрим наиболее часто используемые опции функции/команды `animate`:

1) `frames = n` — опция, указывающая количество кадров в анимации. Параметр `n` является целым числом и по умолчанию равен 25. Пример использования этой опции будет приведен ниже.

2) `trace = n` — опция, указывающая количество предыдущих кадров, которые должны быть сохранены в анимации. Параметр `n` является целым числом и по умолчанию равен 0, то есть предыдущие кадры не сохраняются. Пример использования этой опции будет приведен ниже.

3) `background = s` — опция для указания фона анимации. По умолчанию значение параметра `s = NULL`, что соответствует отсутствию фона. В качестве параметра `s` может быть

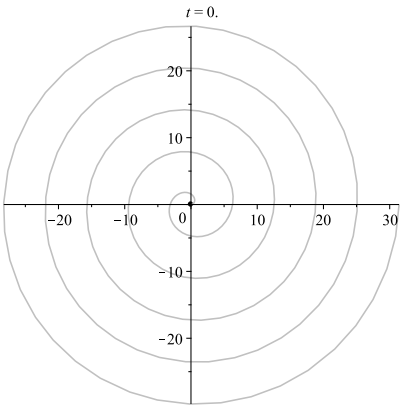
указан цвет (базовые цвета в Maple, RGB-код), картинка или график.

Пример построения анимации точки, движущейся по траектории, задаваемой следующими параметрическими уравнениями:

$$\begin{cases} x(t) = t\cos t \\ y(t) = t\sin t \end{cases} \qquad t \in [0, 10\pi]$$

Поскольку анимироваться будет точка на плоскости, то будем использовать функцию `pointplot`. В качестве фона анимации укажем график кривой, по которой движется точка, с сохранением кадров и общим количеством 100 кадров:

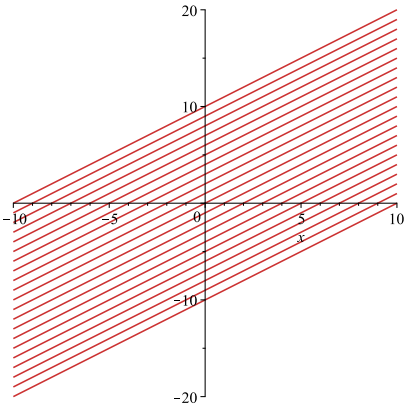
```
> back := plot([t*cos(t), t*sin(t), t = 0..10*Pi],
          color = gray):
> animate(pointplot, [[t*cos(t), t*sin(t)]],
          symbol = solidcircle], t = 0..10*Pi,
          trace = 20, background = back,
          frames = 100);
```



**2.** Помимо функции/команды `animate` для построения анимации может использоваться команда `display` с опцией `insequence`. Данная опция указывает, будут выводиться на экран графики, указанные внутри функции `display`, последовательно или нет. По умолчанию значение `insequence = false`, и все указанные в `display` графики выводятся одновременно.

Пример создания последовательности графиков явной функции $y = x + n$ при $n = -10, \ldots, 10$ и отображение этой последовательности через функцию `display`:

```
> display(seq(plot(x + n, x = -10..10,
                        color = orange), n = -10..10));
```




Из примера видно, что без указания опции `insequence = true` все кривые выведены на одном графике. Добавив указанную опцию, будет выведена только одна кривая, а при нажатии на картинку появится дополнительное меню анимации, как и при использовании команды `animate`.

## 4.4. Примеры решения задач

**1.** Построить график функции $y(x) = \dfrac{(x-1)^3}{4} + 2$, отметив на нем точки перегиба черным цветом. В тех областях, где функция вогнута, график рисовать синим цветом, в областях, где функция выпукла — красным цветом.

Создадим переменную `y`, в которую запишем рассматриваемую функцию:

```
> restart;
> y := (x - 1)^3 / 4 + 2;
```

$$y := \frac{(x-1)^3}{4} + 2$$

Определим точки перегиба, для этого вычислим вторую производную. Приравнем производную к нулю и получим уравнение для определения точек перегиба:

```
> ddy := diff(y, x, x);
```

$$ddy := \frac{3x}{2} - \frac{3}{2}$$

```
> xP := solve(ddy = 0, x);
```

$$xP := 1$$

В результате получили, что функция имеет только одну точку перегиба при $x = 1$. Для определения, где функция вогнута/выпукла, вычислим вторую производную в точках левее и правее точки перегиба:

```
> yL := subs(x = xP - 2, ddy);
```

$$yL := -3$$

```
> yR := subs(x = xP + 2, ddy);
```

$$yR := 3$$

Получаем, что левее точки перегиба функция выпукла, а правее — вогнута. Следовательно, в области $x \in (-\infty, 1)$ график

следует рисовать красным цветом, а в области $x \in [1, +\infty)$ — синим цветом.

Рассматриваемая функция $y(x)$ задана явно, поэтому для отрисовки ее графика воспользуемся функцией `plot`. Для отрисовки точки перегиба воспользуемся функцией `pointplot`, также для объединения всех графиков в одном изображении потребуется функция `display`. Последние две функции располагаются в пакете `plots`, поэтому начнем с подключения этого пакета:

```
> with(plots):
```

Построим выпуклую и вогнутую части графика и запишем их в переменные `P1` и `P2`:

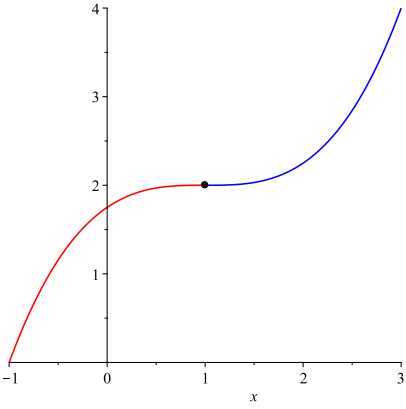
```
> P1 := plot(y, x = -1..xP, color = red):
> P2 := plot(y, x = xP..3, color = blue):
```

Изображение точки перегиба на плоскости запишем в переменную `P3`:

```
> P3 := pointplot([xP, subs(x = xP, y)], color = black,
              symbol = solidcircle, symbolsize = 15):
```

Объединяем все графики на одном:

```
> display(P1, P2, P3);
```




**2.** Найти точки пересечения следующих поверхностей:

$$x^2 + y^2 + z^2 = 49, \quad y - 3 = 0, \quad z + 6 = 0.$$

Для начала создадим переменные `surf1, surf2, surf3`, в которые запишем рассматриваемые функции:

```
> surf1 := x^2 + y^2 + z^2 = 49;
```

$$surf1 := x^2 + y^2 + z^2 = 49$$

```
> surf2 := y - 3;
```

$$surf2 := y - 3$$

```
> surf3 := z + 6;
```

$$surf3 := z + 6$$

Для нахождения точек пересечения необходимо решать систему уравнений, составленную из этих уравнений:

```
> sys := {surf1, surf2 = 0, surf3 = 0};
```

$$sys := \{y - 3 = 0, z + 6 = 0, x^2 + y^2 + z^2 = 49\}$$

```
> solve(sys, {x, y, z});
```

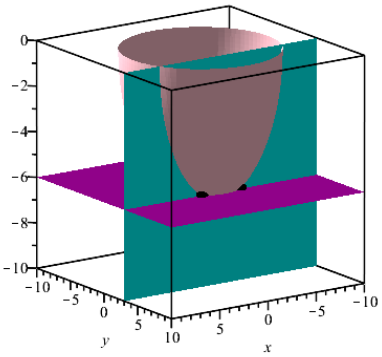
$$\{x = 2, y = 3, z = -6\}, \{x = -2, y = 3, z = -6\}$$

Полученные точки и есть точки пересечения, нарисуем все поверхности и отметим найденные точки:

```
> with(plots):
> Ps := pointplot3d([[2, 3,-6], [-2, 3, -6]],
        color = black, symbol = solidcircle,
        symbolsize = 30):
> surfs := implicitplot3d([surf1, surf2, surf3],
         x = -10..10, y = -10..10, z = -10..0,
         grid = [50, 50, 50], style = surface,
         color = [pink, cyan, purple]);
```

$$[Length\ of\ output\ exceeds\ limit\ of\ 1000000]$$

```
> display(surfs, Ps);
```




## Задачи

4.1. Построить графики следующих функций:

$$y = x^2 + 5x - 5$$

$$\begin{cases} x(t) = t \sin t \\ y(t) = \cos t \end{cases} \quad t \in [0, 10\pi]$$

$$\operatorname{arctg} \tfrac{x}{y} = xy$$

4.2. Построить на одном графике:

$$\begin{cases} x(t) = a \sin t \\ y(t) = a \cos t \end{cases} \quad t \in [0, 10\pi]$$

$$a = 1, 1.5, 2$$

4.3. Построить поверхность

$$z = -\sin(x^2 + y^2) + 1$$

4.4. Нарисовать и отметить точки пересечения следующих поверхностей:

$$x^2 + 5y^2 = 1$$

$$z = x^2 - y^2$$

$$(x-1)^2 + y^2 + z^2 = 1$$

## Список команд

1) `piecewise` — задание кусочной функции.
2) `evalf` — приближенное вычисление выражения.
3) `numer` — выделение числителя в дроби.
4) `denom` — выделение знаменателя в дроби.
5) `rhs` — выделение правой части равенства.
6) `lhs` — выделение левой части равенства.
7) `subs` — замена переменной или переменных в выражении.
8) `simplify` — упрощение выражения.
9) `factor` — разложение выражения на множители.
10) `expand` — раскрытие скобок.
11) `combine` — объединение членов выражения.
12) `collect` — приведение подобных членов выражения.
13) `radnormal` — преобразования выражения с радикалами.
14) `assume` — определение свойств переменных.
15) `map` — применение команды к списку выражений.
16) `evalc` — вычисление комплексного числа.
17) `Re` — выделение действительной части комплексного числа.
18) `Im` — выделение мнимой части комплексного числа.
19) `conjugate` — сопряжение комплексного числа.
20) `limit` — вычисление предела.
21) `diff` — вычисление производной.

22) `int` — вычисление интеграла.

23) `sum` — вычисление конечной суммы или ряда.

24) `add` — вычисление суммы последовательности.

25) `seq` — создание последовательности.

26) `product` — вычисление конечного или бесконечного произведения.

27) `series` — разложение функции в степенной ряд.

28) `taylor` — разложение функции в ряд Тейлора по одной переменной.

29) `mtaylor` — разложение функции в ряд Тейлора по нескольким переменным.

30) `op` — выделение слагаемых в полиноме.

31) `nops` — подсчет количества слагаемых в полиноме.

32) `coeff` — выделение коэффициента в полиноме при заданной степени.

33) `coeffs` — выделение всех коэффициентов в полиноме.

34) `solve` — аналитическое решение алгебраического уравнения или системы алгебраических уравнений.

35) `fsolve` — численное решение уравнения или системы уравнений.

36) `plot` — построение двумерного графика явно или параметрически заданной кривой.

37) `implicitplot` — построение двумерного графика неявно заданной кривой.

38) `contourplot` — построение линий уровня неявно заданной функции.

39) `pointplot` — построение точек на плоскости.

40) `listplot`, `dataplot` — построение двумерного графика по точкам.

41) `display` — отображение нескольких графиков на одном рисунке.

42) `densityplot` — построение распределения неявно заданной функции.

43) `fieldplot` — построение двумерного векторного поля.

44) `plot3d` — построение поверхности по явно заданной функции.

45) `implicitplot3d` — построение поверхности по неявно заданной функции.

46) `spacecurve` — построение кривой в пространстве.

47) `intersectplot` — построение кривой в пространстве, заданной как пересечение поверхностей.

48) `pointplot3d` — построение точек в пространстве.

49) `animate` — построение анимации.

# Список литературы


[1] Ветчанин Е. В., Артемова Е. М. Программирование в системах GNU Octave и MatLAB: учеб.-метод. пособие. — Ижевск: Изд-во «Удмуртский университет», 2019. — 99 с.

[2] Ветчанин Е. В., Артемова Е. М. Процедурное программирование на языках C/C++: учеб.-метод. пособие. — Ижевск: Изд-во «Удмуртский университет», 2020. — 124 с.

[3] Лебедев, В. Г., Иванова, Т. Б., Васькин, В. В., Пивоварова, Е. Н. Методы математической физики. Параболические уравнения: учеб.-метод. пособие. — Ижевск: Изд-во «Удмуртский университет», 2022. — 123 с.

[4] Лебедев В. Г., Иванова Т. Б., Васькин В. В. Уравнения теплопереноса: учеб.-метод. пособие. — Ижевск: Изд-во «Удмуртский университет», 2018. — 104 с.

[5] Килин А. А. Введение в теорию точечных отображений. Динамический хаос: учеб. пособие. — Ижевск: Изд-во «Удмуртский университет», 2021. — 100 с.

[6] Матросов А. В. Основы работы в Maple V — Спб.: Санкт-Петербургский государственный университет водных коммуникаций, 1999. — 61 с.

[7] Корольков О. Г., Чеботарев А. С., Щеглова Ю. Д. Maple в примерах и задачах. — Воронеж: Издательско-полиграфический центр Воронежского государственного университета, 2011. — 132 с.

[8] Голоскоков Д. П. Практический курс математической физики в системе Maple: учебное пособие для студентов высших учебных заведений. — СПб.: ООО «ПаркКом», 2010. — 643 с.

[9] Стребуляев С. Н., Миронова Д. А. Использование системы аналитических вычислений Maple для решения задач прикладной математики. — Нижний Новгород: Изд-во «Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского», 2007.

[10] Савотченко С. Е., Кузьмичева Т. Г. Методы решения математических задач в Maple: учебное пособие. — Белгород, 2001. — 116 с.

[11] Матросов А. В. Maple 6. Решение задач высшей математики и механики. — СПб.: БХВ-Петербург, 2001. — 528 с.

*Учебное издание*

Артемова Елизавета Марковна
Бизяев Иван Алексеевич

**СИСТЕМА КОМПЬЮТЕРНОЙ МАТЕМАТИКИ MAPLE: ТЕОРИЯ И РЕШЕНИЕ ЗАДАЧ**
Часть 1

Учебно-методическое пособие

*Авторская редакция*

Подписано в печать 30.06.2023. Формат 60x84 1/16.
Усл. печ. л. 5,75. Уч. изд. л. 5,26.
Тираж 33 экз. Заказ № 1131.

Издательский центр «Удмуртский университет»
426034, г. Ижевск, Ломоносова, 4Б, каб. 021
Тел.: + 7 (3412) 916-364, E-mail: editorial@udsu.ru

Типография Издательского центра «Удмуртский университет»
426034, Ижевск, ул. Университетская, 1, корп. 2.
Тел. 68-57-18