Руководство по языку

«MATHPAR»

Геннадий Малашонок

ВЕРСИЯ 15.00

25.01.2025

ОГЛАВЛЕНИЕ

Глава 1

Введение

Данное руководство по языку Mathpar поможет Вам при решении математических задач. Оно будет всегда Вашим помощником, когда Вам нужно воспользоваться математикой: будь то решение задачи в школе или в университете, выполнение научных расчетов или решение производственной задачи.

Mathpar поможет Вам делать простые числовые или алгебраические операции, строить графики кривых и поверхностей.

Он поможет Вам решать задачи различных разделов математического анализа, алгебры, геометрии, задачи по физике, по химии и другие.

Если же Вы профессионально применяете математику, то он поможет Вам избавиться от рутинных вычислений и оперировать с очень большими математическими объектами, задействуя при этом суперкомпьютеры. Mathpar позволяет оперировать с функциями и функциональными матрицами, получать как точные численно-аналитические решения, так и решения, в которых числовые коэффициенты получаются с требуемой степень точности.

В основе языка Mathpar лежит широко используемый математиками и физиками язык TeX, который обычно используют для набора математических текстов.

Вы можете сохранить как постановку задачи, так и ход ее решения. При этом можете сохранять и текстовый вид (Mathpar, TeX или MathML) и изображение (pdf, jpg).

Весь излагаемый тут материал делится на 14 глав.

Для первого знакомства достаточно ознакомиться с двумя сле-

дующими главами данного руководства.

Во второй главе описывается ввод данных и выполнение простейших вычислений. Даются обозначения для элементарных функций, таких как логарифм, синус, косинус и т.д., и констант — π, e, i, а также констант, которые необходимы для задания числовых множеств. Описываются способы задания векторов и матриц, арифметические операции над ними, команды генерации случайных чисел, полиномов и матриц, команды для решения алгебраических уравнений. Для всех команд приведены примеры.

Третья глава посвящена построению графиков функций. Mathpar позволяет строить графики функций, которые заданны явно или параметрически, кроме того функции могут быть заданы таблично – множеством значений функций на конечном множестве значений аргумента. Можно выполнить построение нескольких графиков в одной системе координат.

В четвертой главе описываются способы задания окружения в системе Mathpar, т.е. того пространства, в котором будут определяться математические объекты. В любой момент Вы можете сменить окружение и задать новое алгебраическое пространство.

В пятой главе описаны команды для задания математических функций одной или нескольких переменных, их композиций, вычисления значений функции в точке, подстановки выражений в функции, вычисление предела функции в точке, символьного интегрирования композиций элементарных функций. Приводятся примеры выполнения команд.

Шестая глава посвящена действиям с рядами. Рассматриваются способы задания ряда. Даются команды для сложения, вычитания, умножения двух рядов и для разложения функции в ряд Тейлора с определенным количеством членов ряда.

В седьмой главе описаны команды для решения обыкновенных дифференциальных уравнений и систем, а также дифференциальных уравнений с частными производными.

Восьмая глава посвящена полиномиальным вычислениям. Рассматриваются команды для вычисления значения полинома в точке, суммирования полинома по переменным, вычисления базиса Гребнера полиномиального идеала над рациональными числами.

В девятой главе описываются матричные функции — вычисление транспонированной матрицы, определителя матрицы, присоеди-

ненной и обратной матриц, эшелонной формы матрицы, ядра оператора, характеристического полинома матрицы и другие.

Десятая глава посвящена функциям теории вероятностей и математической статистики. Описывается задание дискретной случайной величины, команды для вычисления математического ожидания дискретной случайной величины, дисперсии, среднего квадратичного отклонения, суммы, произведения двух дискретных случайных величин, коэффициента ковариации, коэффициента корреляции, построения многоугольника распределения и функции распределения дискретной случайной величины. В этой главе рассматриваются команды для задания выборок и для вычисления функций для них: выборочное среднее, выборочная дисперсия, коэффициент ковариации и коэффициент корреляции для двух выборок.

Mathpar не только активный математический язык, но он еще и процедурный язык программирования. Одиннадцатая глава посвящена программированию в языке Mathpar. В этой главе описаны правила записи процедур и основной части программы, правила записи операторов ветвления и цикла. Вы можете написать программу, содержащую Ваши новые процедуры и функции, и потом много раз использовать эти процедуры и функции для выполнения необходимых Вам вычислений. Mathpar можно использовать для обучения программированию в школе.

В главе двенадцатой описываются команды, которые управляют вычислениями на суперкомпьютере. Для решения вычислительных задач, которые требуют большого времени вычислений или больших объемов памяти, разработаны специальные функции, которые предоставляют Вам ресурсы суперкомпьютера. При использовании этих функций вычисления производятся не на одном процессоре, а на выделенном множестве ядер суперкомпьютера, количество которых заказывает пользователь. Это такие операции, как вычисление базиса Гребнера, присоединенной матрицы, ступенчатого вида матрицы, обратной матрицы, определителя, ядра линейного оператора, характеристического полинома и др. На момент подготовки этой редакции руководства пользователя вычисления на суперкомпьютере не поддерживаются.

В тринадцатой главе приведен список основных операторов в языке Mathpar.

В четырнадцатой главе приведены примеры решения задач по

физике.

Глава 2

Знакомство и первые шаги

Эта глава посвящена первому знакомству с возможностями, которые Вам открывает Mathpar. Язык Mathpar, который описывается ниже, может рассматриваться, как некоторое развитие языка TeX. Язык TeX предназначен для записи математических текстов и подготовке их к публикации. Его можно считать пассивным по сравнению с языком Mathpar, который позволяет делать вычисления, то есть является активным языком математики. Как формулировка задачи, так и результаты вычислений, записываются на языке Mathpar.

Сразу после вычислений Вы видите весь математический текст в виде pdf-изображения, в том виде, как принято представлять математическую запись в научных и технических публикациях.

Этот результат может быть использован дальше несколькими способами.

(1) Можно кликнуть по тексту мышкой, и он вернется к исходному виду языка Mathpar. Есть и другой способ переключения изображения текста: при помощи кнопки«  », расположенной между кнопками «▶» и «+».

(2) Можно кликнуть по **изображению математической записи** правой кнопкой мышки. В этом случае появится выпадающее меню. Верхнее поле «Show Math As» позволяет перейти к выбору языка вывода. Предлагается выбрать TeX или MathML. И затем

открыть поле с желаемым текстом.

Например, матрица A, размера 2×2, в Mathpar будет записана так:

A=[[a, b], [c, d]];

в TeX она выглядит так:

A= \left(\begin{*array*}{*cc*}*a*&*b* *c*&*d*\\ \end{*array*}\right).

В MathML это еще более громоздкое выражение.

Полученный текст на языке TeX или MathML можно скопировать и поместить в TeX- или html-файл и использовать для публикации. Кроме того, можно получить обычное изображение и разместить его в любом документе. Это необходимо, например, когда требуется сохранить график функции или решение задачи.

2.1. Ввод данных, решение задачи

В центральной части экрана находится поле ввода. Здесь Вы размещаете математические выражения. Для решения задачи надо нажать на кнопку «►», которая расположена над полем ввода. Кроме того, можно использовать сочетание клавиш *Ctrl+Enter*. Например, можно набрать 2+2 и нажать «►».

В верхней части экрана находятся активные поля Помощь и Руководство. Указывая мышкой на эти поля, Вы можете перейти к страницам Помощи или открыть Руководство по языку Mathpar.

На рисунке приведен пример простого задания.

На страницах Помощи все поля с примерами являются активными полями, содержащиеся там задачи можно тут же решить и увидеть ответ. Для этого нужно кликнуть по кнопке «►» или же можно поставить курсор на поле примера и нажать *Ctrl+Enter*. Можно копировать текст из любого примера и перенести его в Ваше основное поле ввода. Для этого можно использовать выделение текста мышкой, копирование и вставку этого текста с помощью сочетания клавиш *Ctrl+C* и *Ctrl+V*, соответственно, для копирования и вставки.

Текст, который Вы можете вводить в поле ввода, состоит из комментариев и математических операторов.

При вводе комментариев, то есть любого поясняющего текста, необходимо брать его в кавычки. Например: (" это комментарий "). Кавычки разрешается использовать только для комментариев. В

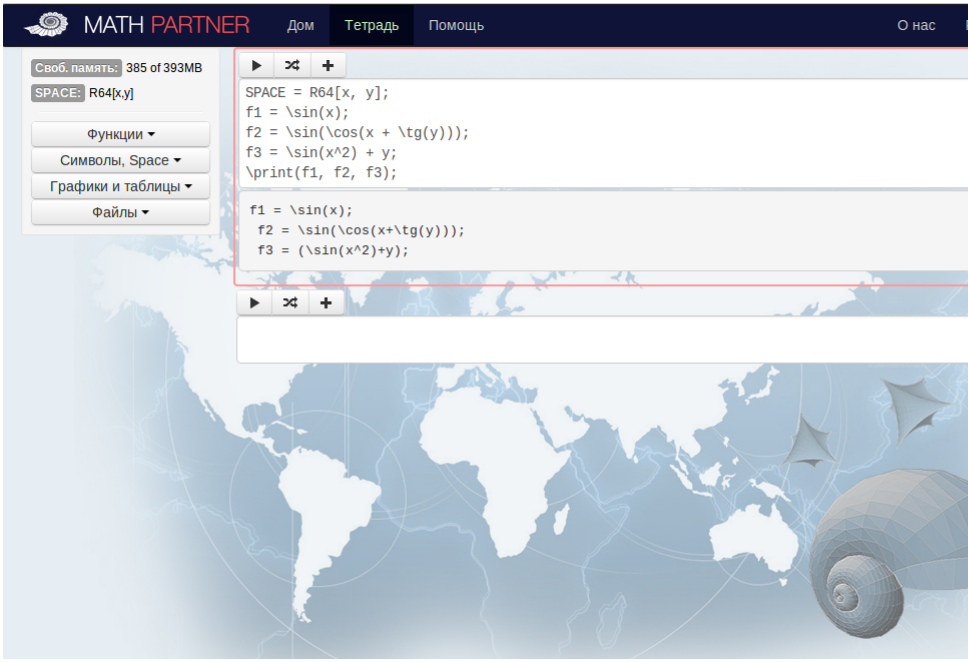


Рисунок 2.1: Запись задания в поле ввода

тексте комментариев можно использовать, например, такие кавычки « ». Когда в комментариях нужно иметь математическое выражение, как часть комментария, то его необходимо окаймить знаками доллара ($). Например, можно написать такой комментарий:
"Два обозначения $\backslash exp(x)$ и $\backslash e\hat{}x$ применяются для экспоненциальной функции."

При вводе математических выражений их необходимо разделять точкой с запятой (;) или текстом комментариев, которые заключены в кавычки. Можно не ставить точку с запятой после последнего оператора. В математических операторах, когда необходимо вставить текст, нужно использовать апострофы: ('текст в математическом операторе').

Для вывода результатов можно использовать команду **print**(), где в качестве аргументов, разделенных запятыми, необходимо указать имена тех выражений, которые требуется вывести. Если среди команд не встретился оператор печати **print**() или какой-нибудь другой оператор вывода (**plot**(), **prints**() и т. д.), то будет выводится результат, полученный в последнем операторе или последней новой переменной.

Команды и операторы начинаются с символа «back slash» (\).

Кнопка [+] предназначена для добавления полей ввода. Для удаления поля ввода можно воспользоваться сочетанием клавиш *Ctrl+Del* или крестиком [x], расположенным над полем ввода справа.

Кнопка [C], расположенная над полем ввода справа, предназначена для отмены всех введенных раннее обозначений. Отмена обозначений позволяет получать формулы, в которых стоят символы, а не числа.

В левой верхней части экрана находятся поля, в которых указано текущее окружение и объем оперативной памяти в мегабайтах. Окружение фиксируется числовым множеством и именами основных переменных. Под этим полем расположены различные меню, которые облегчают ввод функций и операторов.

2.1.1. Работа с файлами

Функции для работы с файлами доступны из раскрывающейся панели «Файлы», расположенной в меню слева.

Существуют следующие возможности для обработки файлов:

1) Сохранение последней выполненной секции в виде файла PDF с помощью кнопки «Сохранить PDF». Можно указать собственный размер страницы (в сантиметрах), по умолчанию указан размер A4 (21x29.7 см).

2) Загрузка текстовых файлов на сервер Mathpar с помощью кнопки «Загрузить файл». Под этой кнопкой расположен список загруженных файлов. Файлы должны содержать выражения на языке Mathpar или таблицы в специальном формате.

Таблица состоит из строки с заголовком (произвольные строки) и строк с числами. Столбцы отделяются знаком табуляции. Функции для работы с таблицами доступны в разделе «Графики и таблицы» (см. также раздел 3.1 Построение графиков функций системы помощи).

3) Ввод выражений на языке Mathpar из загруженного файла с помощью функции **fromFile**(). Например, создать выражение из загруженного файла myfile.txt и присвоить это выражение переменной a можно командой a = **fromFile**($'myfile.txt'$).

2.2. Математические функции

Приняты следующие обозначения для элементарных функций и констант.

2.2.1. Константы

\i — мнимая единица,
\e — основание натурального логарифма,
\pi — число π, то есть отношение длины окружности к диаметру,
\infty – знак бесконечности.

2.2.2. Функции одного аргумента

\ln — натуральный логарифм,
\lg — десятичный логарифм,
\sin — синус,
\cos — косинус,
\tg — тангенс,

\ctg — котангенс,
\arcsin — арксинус,
\arccos — арккосинус,
\arctg — арктангенс,
\arcctg — арккотангенс,
\sh — синус гиперболический,
\ch — косинус гиперболический,
\th — тангенс гиперболический,
\cth — котангенс гиперболический,
\arcsh — арксинус гиперболический,
\arcch — арккосинус гиперболический,
\arcth — арктангенс гиперболический,
\arccth — арккотангенс гиперболический,
\exp — экспонента,
\sqrt — корень квадратный,
\abs — абсолютное значение для действительных чисел, модуль для комплексного числа,
\sign — знак числа. Возвращает 1, 0, -1, когда число положительное, ноль или отрицательное, соответственно,
\unitStep(x) — это функция, которая при $x \geqslant 0$ принимает значение 1, а при $x < 0$ принимает значение 0;
\fact — факториал. Определен для целых положительных чисел. Равносильная запись — «n!».

2.2.3. Функции двух аргументов

^ — степень,
\log — логарифм от функции по указанному основанию,
\rootOf(x, n) — корень степени n из x,
\Gamma — функция Гамма,
\Gamma2 — функция Гамма 2,
\binomial — число сочетаний.

Примеры.

```
SPACE = R64[x, y];
f1 = \sin(x);
f2 = \sin(\cos(x + \tg(y)));
f3 = \sin(x^2) + y;
\print(f1, f2, f3);
```

Результат выполнения:
$SPACE = R64[x, y]$;
$f1 = sin(x)$;
$f2 = sin(cos(x + tg(y)))$;
$f3 = sin(x^2) + y$.

2.3. Действия с функциями

Для перечисленных выше функций и их композиций можно вычислить значение функции в точке, подставить выражения в функцию вместо аргументов, вычислить предел функции, ее производную. Для этого определены следующие команды.

Для вычисления значения функции в точке необходимо выполнить команду
$\backslash\mathbf{value}(f, [var1, var2, \ldots, varn])$, где f — функция, а $var1, var2, \ldots, varn$ — значения соответствующих переменных кольца. Для подстановки выражений в функцию необходимо выполнить команду
$\backslash\mathbf{value}(f, [func1, func2, \ldots, funcn])$, где f — это функция, $func1, func2, \ldots, funcn$ — функции, которые подставляются вместо соответствующих переменных.

Для вычисления предела функции в точке необходимо выполнить команду
$\backslash\mathbf{lim}(f, var)$, где f — это функция, а var — точка, в которой требуется найти предел.

Для вычисления производной функции f по переменной y из кольца $\mathrm{Z}[x,\ y,\ z]$ необходимо выполнить команду $\backslash\mathbf{D}(f, y)$. Для нахождения смешанной производной первого порядка от функции f существует команда $\backslash\mathbf{D}(f, [x, y])$, для нахождения производной высших порядков нужно использовать команду $\backslash\mathbf{D}(f, [x\hat{}k, z\hat{}m, y\hat{}n])$, где k, m, n указывают, какого порядка по соответствующей переменной вычисляется производная.

Примеры.

```
SPACE = R[x, y];
f = \sin(x^2 + \tg(y^3 + x));
g = \value(f, [1, 2]);
\print(g);
```

Результат выполнения:
in: $SPACE = R[x, y]$;
$f = sin(x^2 + tg(y^3 + x))$;
$g = value(f,\ [1,\ 2])$;
$print(g)$;
out: $g = 0.52$;

```
SPACE = Z[x, y];
f = x + y;
g = f^2;
r = \value(f, [x^2, y^2]);
\print(g, f, r);
```

Результат выполнения:
in: $SPACE = Z[x, y]$;
$f = x + y$;
$g = f^2$;
$r = value(f, [x^2, y^2])$;
$print(g, f, r)$;
out: $g = y^2 + 2yx + x^2$;
$f = y + x$;
$r = x^2 + y^2$

```
SPACE = R64[x];
f = \sin(x) / x;
g = \lim(f, 0);
\print(g);
```

Результат выполнения:
in: $SPACE = R64[x]$;
$f = sin(x)/x$;
$g = lim(f, 0)$;
$print(g)$;
out: $g = 1.00$;

```
SPACE = Z[x, y];
f = \sin(x^2 + \tg(y^3 + x));
h = \D(f, y);
\print(h);
```

Результат выполнения:
in: $SPACE = Z[x, y]$;

$f = sin(x^2 + tg(y^3 + x))$;
$h = D(f, y)$;
$print(h)$;
out: $h = 3y^2 cos(x^2 + tg(y^3 + x))/(cos(y^3 + x))^2$;

2.4. Решение алгебраических уравнений

Для решения алгебраических уравнений нужно выполнить команду **solve**. Ниже используется команда настройки окружения «FLOATPOS=N». Она устанавливает число десятичных знаков после запятой (N), которые должны появиться при выводе числового результата приближенных вычислений. Она не связана с процессом вычислений, а только с выводом. По умолчанию $FLOATPOS = 2$.

Примеры.

```
SPACE = R64[x];
b = \solve(x^2 - 5x + 6 = 0);
```

Результат выполнения:
in: $SPACE = R64[x]$;
$b = solve(x^2 - 5x + 6 = 0)$;
out: $[2.00, 3.00]$;

```
SPACE = R64[x];
FLOATPOS = 6;
b = \solve(x^4 + 2x + 1 = 0);
```

Результат выполнения:
in: $SPACE = R64[x]$;
$FLOATPOS = 6$;
$b = solve(x^4 + 2x + 1 = 0)$;
out: $[-0.543689, -1.000000]$;

```
SPACE = R64[x];
FLOATPOS = 0;
b = \solve(x^3 + 3x^2 + 3x + 1 = 0);
```

Результат выполнения:
in: $SPACE = R64[x]$;
$FLOATPOS = 0$;
$b = solve(x^3 + 3x^2 + 3x + 1 = 0)$;
out: -1.

2.5. Решение алгебраических неравенств

Для решения алгебраических неравенств нужно выполнить команду **solve**, в которой записано неравенство. Можно решать строгие и не строгие алгебраические неравенства. Открытый интервал обозначается круглыми скобками ( ), а закрытый интервал – квадратными скобками [ ], множество обозначается фигурными скобками { }.

```
SPACE = R[x];
b = \solve(x^2-5x+6 < 0);
```

Результат выполнения:
in: $SPACE = R[x]$;
$b = solve(x^2 - 5x + 6 < 0)$;
out: $(2, 3)$.

```
SPACE = R[x];
b = \solve((x+1)^2(x-3)(x+5) \ge 0);
```

Результат выполнения:
in: $SPACE = R[x]$;
$b = solve((x + 1)^2(x - 3)(x + 5) \geq 0)$;
out: $(-\infty, -5] \cup -1 \cup [3, \infty)$.

```
SPACE = R[x];
b = \solve((x^2+11x+28)/(x+5) \le 0);
```

Результат выполнения:
in: $SPACE = R[x]$;
$b = solve((x^2 + 11x + 28)/(x + 5) \leq 0)$;
out: $(-\infty, -7] \cup (-5, -4]$.

```
SPACE = Q[x];
b = \solve(x^2 + 4x - 7 = 0);
```

Результат выполнения:
in: $SPACE = Q[x]$;
$b = solve((x^2 + 4x - 7 = 0)$;
out: $[(\sqrt{11} + (2)), (2 - \sqrt{11})]$;.

2.6. Решение систем алгебраических неравенств

Для решения систем алгебраических неравенств нужно выполнить команду **solve**$[In1, In2, ..., Ink]$, где $[In1, In2, ..., Ink]$ — вектор неравенств. Система может содержать строгие и не строгие алгебраические неравенства. Открытый интервал обозначается круглыми скобками ( ), а закрытый интервал — квадратными скобками [ ], множество обозначается фигурными скобками { }.

```
SPACE = R[x];
b = \solve([x^2+4x-5 > 0, x^2-2x-8 < 0]);
```

Результат выполнения:
in: $SPACE = R[x]$;
$b = solve([x^2 + 4x - 5 > 0, x^2 - 2x - 8 < 0])$;
out: $(1, 4)$.

```
SPACE = R[x];
b = \solve([x^2-x-6 \ge 0, x^2-4x-12 < 0]);
```

Результат выполнения:
in: $SPACE = R[x]$;
$b = solve([x^2 - x - 6 \geq 0, x^2 - 4x - 12 < 0])$;
out: $(-4, -2] \cup [3, 4)$.

```
SPACE = R[x];
b = \solve([x^2-4 < 0, x+1 > 0, 0.5-x > 0]);
```

Результат выполнения:
in: $SPACE = R[x]$;
$b = solve([x^2 - 4 < 0, x + 1 > 0, 0.5 - x > 0])$;
out: $(-1, 0.5)$.

2.7. Операции на подмножествах действительных чисел

Подмножество, содержащее несколько интервалов можно задать так **set**$((a, b), (c, d])$, где a, b, c, d — числа. Здесь интервал обозначается круглыми скобками ( ), полуоткрытый интервал – одной круглой и одной квадратной скобкой [ ) или ( ], а отрезок – квадратными

скобками [ ]. Точка обозначается фигурными скобками { } или как закрытый интервал.

Простые подмножества обозначаются такими же скобками, но перед каждой скобкой необходимо добавлять backslash (\). Например \(3, 4.5)\] или \[7, 7\]. Оператор **set** не требуется.

```
SPACE = R64[x];
a = \set((-2,1),[2,5),(5.75,6],{8});
```

Результат выполнения:
in: $SPACE = R[x]; a = set((-2, 1), [2, 5), (5.75, 6], 8);$
out: $((-2), 1) \cup [2, 5) \cup (5.75, 6] \cup \{8\}$.

```
SPACE = R64[x];
a = \set((-2,1),(0,5));
```

Результат выполнения:
in: $SPACE = R[x];$
$a = set((-2, 1), (0, 5));$
out: $((-2), 5);$.

С подмножествами можно совершать операции объединения, пересечения, вычитания, вычисления симметрической разности и дополнения при помощи команд $\backslash cup$ и $\backslash cap$, $\backslash setminus$, $\backslash triangle$ и знака апостроф (') соответственно.

```
SPACE = R64[x];
A=\(1,3\)\cup\[5,16\);
B=\(2,4\)\cup\[10,20\);
C=A\cup B;
D=A\cap B;
E=A\triangle B;
F=A \setminus B;
G=A';
\print(C,D,E,F,G);
```

Результат выполнения:
in: $SPACE = R64[x];$
$A = (1, 3) \cup [5, 16);$
$B = (2, 4) \cup [10, 20);$
$C = A \cup B;$

$D = A \cap B$;
$E = A \triangle B$;
$F = A \setminus B$;
$G = A'$;
$print(C, D, E, F, G)$;
out: $C = (1, 4) \cup [5, 20)$
$D = (2, 3) \cup [10, 16)$
$E = (1, 2] \cup [3, 4) \cup [5, 10) \cup [16, 20)$
$F = (1, 2] \cup [5, 10)$
$G = (-\infty, 1] \cup [3, 5) \cup [16, \infty)$

2.8. Векторы и матрицы

Для задания вектора нужно перечислить его элементы в квадратных скобках. Так задаются вектор-строки. Для задания матрицы нужно заключить в квадратные скобки ее вектор-строки, разделенные запятыми, например, $A = [[1, 2], [3, 4]]$.

Подматрицу размера $Nr \times Nc$ матрицы A определяет команда $\backslash\mathbf{submatrix}(A, r1, Nr, c1, Nc)$. Здесь $r1, c1$ – это позиция верхнего левого элемента.

Элемент матрицы можно получить, указав номер строки и столбца в нижних индексах у элемента матрицы, а элемент вектора можно получить указав один индекс. Например, можно определить элементы матрицы так: $a = \backslash elementOf(A)$, и потом обращаться к отдельным элементам: $a\_\{i, j\}$. Или же определить элементы вектора B так: $b = \backslash elementOf(B)$, затем обращаться к ним $b\_\{i\}$.

Можно получить строку i матрицы в виде вектор-строки: $a\_\{i, ?\}$ или столбец матрицы j в виде вектор-столбца: $a\_\{?, j\}$.

Имена некоммутативных объектов, например матриц и векторов, положено писать с первым символом backslash (\) и заглавной латинской буквы, если предполагается их использовать в таких выражениях, в которых нельзя допускать перестановок. Например $\backslash A * \backslash B - \backslash B * \backslash A$ не приведет автоматически к нулю, в отличие от $A * B - B * A$, что сразу упростится в 0.

Для обозначения нулевой и единичной матрицы используются заглавные буквы $\backslash O$ и $\backslash I$, у которых указаны два индекса, обозначающих число строк и столбцов. С помощью символа $\backslash I$ можно создавать прямоугольные матрицы любого размера, у которых элементы на главной диагонали равны 1, а остальные элементы нулевые. Например, $\backslash I\_\{2, 3\}$ и $\backslash O\_\{2, 2\}$ обозначают матрицы $\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$ и $\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$. Можно задавать нулевые векторы, указывая в индексе число элементов: $\backslash O\_\{3\}$ обозначает вектор [0, 0, 0], а $I\_\{3\}$ обозначает вектор [1, 0, 0].

Отметим, что в качестве одномерных и двумерных массивов в языке Mathpar используются векторы и матрицы, например, O_{n}, O_{n, m}.

Вектор-столбец может быть образован транспонированием

вектор-строки, например, $D = [7,\ 2,\ 3]^T$ — это вектор-столбец из трех элементов. Кроме обычных арифметических операций (+,-,*) можно вычислять функции от векторов поэлементно.

Пример 1.

```
SPACE = Z[x];
A = [[x, 4], [y, 5]];
V = [x, y, 1, 2, x^6];
\print(A, V);
```

Результат выполнения:

in: $SPACE = Z[x]$;

$A = \begin{pmatrix} x & 4 \\ y & 5 \end{pmatrix}$;

$V = [x, y, 1, 2, x^6]$;

$print(A, V)$;

out: $A = \begin{pmatrix} x & 4 \\ y & 5 \end{pmatrix}$;

$V = [x, y, 1, 2, x^6]$;

Пример 2.

```
SPACE = Z[x, y];
A = [[3, 4], [3, 1]];
B = [[2, 5], [4, 7]];
C = A + B;
G = A - B;
T = A * B;
\print(C, G, T);
```

Результат выполнения:

in: $SPACE = Z[x]$;

$A = \begin{pmatrix} 3 & 4 \\ 3 & 1 \end{pmatrix}$;

$B = \begin{pmatrix} 2 & 5 \\ 4 & 7 \end{pmatrix}$;

$C = A + B$;

$G = A - B$;

$T = A * B$;

$print(C, G, T)$;

out: $C = \begin{pmatrix} 5 & 9 \\ 7 & 8 \end{pmatrix}$;

$G = \begin{pmatrix} 1 & -1 \\ -1 & -6 \end{pmatrix}$;

$T = \begin{pmatrix} 22 & 43 \\ 10 & 22 \end{pmatrix}$;

Пример 3.

```
SPACE = Z[x];
A = [[1, 4], [-4, 5]];
a = \elementOf(A);
det = a_{1, 1} * a_{2, 2} - a_{1, 2} * a_{2, 1};
\print(det);
```

Результат выполнения:

in: $SPACE = Z[x]$;

$A = \begin{pmatrix} 1 & 4 \\ -4 & 5 \end{pmatrix}$;

$det = a_{1,1} * a_{2,2} - a_{1,2} * a_{2,1}$;

$print(det)$;

out: $det = 21$;

Пример 4.

```
SPACE = Z[x, y];
A = [[x^2, y], [4, x+y]];
a = \elementOf(A);
B = a_{1, ?};
C = a_{?, 2};
b = \elementOf(B);
c = \elementOf(C);
h = b_{2} * c_{1, 1};
\print(B, C, h);
```

Результат выполнения:

in: $SPACE = Z[x, y]$;

$A = \begin{pmatrix} x^2 & y \\ 4 & x+y \end{pmatrix}$;

$a = \backslash elementOf(A)$;

$B = a_{1,?}$;

$C = a_{?,2}$;

$b = \backslash elementOf(B); c = \backslash elementOf(C);$
$h = b_2 \cdot c_1;$
$print(B, C, h);$
out: $B = \begin{pmatrix} x^2 \\ 4 \end{pmatrix};$
$C = \begin{pmatrix} 4 & y + x \end{pmatrix};$
$h = (y^2);$

Пример 5.

```
SPACE = Z[x, y];
A = 3x * \I_{2, 2};
B = \O_{3, 3};
\print(A, B);
```

Результат выполнения:
in: $SPACE = Z[x, y];$
$A = 3x * \mathbf{I}_{2,2};$
$B = \mathbf{O}_{3,3};$
$print(A, B);$
out: $A = \begin{pmatrix} 3x & 0 \\ 0 & 3x \end{pmatrix};$
$B = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}.$

Пример 6.

```
SPACE = R64[x];
A = [\pi / 2, \pi];
B = \sin(A);
C = \value(B);
\print(A, B, C);
```

Результат выполнения:
in: $SPACE = R64[x];$
$A = [\pi/2, \pi];$
$B = \sin(A);$
$C = value(B);$
$print(A, B, C);$
out: $A = [\pi/2, \pi];$
$B = \mathbf{s}in([\pi/2, \pi]);$
$C = [1, 0];$

2.9. Создание случайных элементов

Mathpar может создавать случайные числа, полиномы и матрицы. Это удобно, когда Вам нужно создать некоторый произвольный сложный объект или требуется получить много случайных объектов.

2.9.1. Создание случайных чисел

Для того чтобы получить случайное число, необходимо выполнить команду $\backslash$**randomNumber**(k), где в аргументе k указывается количество двоичных разрядов в записи случайного числа. Это соответствует примерно $0.3k$ десятичным цифрам.

Пример.

```
SPACE = Z[x, y, z];
a = \randomNumber(10);
b = \randomNumber(100);
\print(a, b);
```

Результат выполнения:
$SPACE = Z[]$;
$a = 944$;
$b = 850800798881527094755736477974$.

2.9.2. Создание случайных полиномов

Для того чтобы создать случайный полином от s переменных, необходимо выполнить команду $\backslash$**randomPolynom**$(d1, d2, \ldots, ds, dens, bits)$, где $dens$ — плотность полинома, а $bits$ — количество двоичных разрядов в записи случайного числа, $d1, d2, \ldots, ds$ означают старшие степени переменных. Если $dens = 100$, то будет получен полином, у которого все коэффициенты отличны от нуля, всего $(d1+1)(d2+1)..(ds+1)$ членов. Если $dens < 100$, то $dens\%$ будут ненулевые, а $(100-dens)\%$ нулевых.

Пример.

```
SPACE = Z[x, y, z];
f = \randomPolynom(4, 4, 10, 5);
```

```
g = \randomPolynom(4, 4, 10, 5);
h = f + g;
\print(f, g, h);
```

Результат выполнения:
$f = y^3x^3$;
$g = 10yx^3 + 2y$;
$h = y^3x^3 + 10yx^3 + 2y$;

2.9.3. Создание случайных матриц

Для того чтобы получить случайную числовую матрицу, необходимо выполнить команду $\backslash$**randomMatrix**$(m, n, dens, bits)$, где m — количество строк в матрице, n — количество столбцов матрицы, $dens$ — это плотность матрицы в процентах, $bits$ — число двоичных разрядов в записи числовых коэффициентов.

Для того чтобы получить случайную полиномиальную матрицу, необходимо выполнить команду $\backslash$**randomMatrix**$(m, n, dens, d1, d2, \ldots, ds, polDens, polBits))$, где m — количество строк в матрице, n — количество столбцов матрицы, $dens$ — это плотность матрицы, $d1, d2, \ldots, ds$ — наибольшие степени переменных полиномов, $polDens$ — плотность полиномов, $polBits$ — количество двоичных разрядов в записи коэффициентов полиномов.

Пример.

```
SPACE = Z[x, y, z];
matr_n = \randomMatrix(4, 4, 100, 5);
matr_p = \randomMatrix(2, 2, 100, 2, 2, 25, 2);
\print(matr_n, matr_p);
```

Результат выполнения

$$matr_n = \begin{pmatrix} 22 & 2 & 10 & 28 \\ 23 & 28 & 1 & 19 \\ 30 & 24 & 19 & 12 \\ 27 & 22 & 22 & 17 \end{pmatrix};$$

$$matr_p = \begin{pmatrix} 6z^3x + 7z^3 + 5z^2 + 3y & 7z^4x + 2z^4 + 7zyx + 5x \\ z^4yx + 2zy + 7y + 7x + 4 & 7z^2x + 7zx + z + 6x \end{pmatrix}.$$

2.10. Контрольные задания

В Mathpar вычислите:

- $\ln 5,\ \sin 5,\ \cos 3,\ \cot 7,\ \arctan 1,\ sh\ 0,\ arcch\ 0.5,\ \exp 8, 12!,\ \sqrt{100},$
- $\sqrt{\sin^2(5x-1)+\exp x}/\cos(2x)$ при $x = 0.1,\ x = 1,$
- $\log_3 8,\ \sqrt[3]{50},$
- значение функции $f = \sin(\cos(x+\tan(y)))$ при $x = 0.2$ и $y = 1$.
- Создайте два случайных вектора равной длины. Найдите их сумму и произведение.

Глава 3

Построение 2D и 3D графиков

3.1. Построение графиков функций

Mathpar позволяет строить табличные графики (*tablePlot*), графики функций, которые заданы явно (*plot*) или параметрически (*paramPlot*). Можно строить несколько разных графиков в одной системе координат(*showPlots*).

Окружение для построения графиков задается командой **set2D**(). Если у команды **set2D**() нет параметров, то границы для графиков расчитываются автоматически, а для явных функций выбирается интервал $[0, 1]$ по оси абсцисс. Наименования осей координат будет X и Y, соответственно. Заголовок у графика будет отсутствовать.

Если команда **set2D**() пользователем не задавалась, то автоматически устанавливается **set2D**() без параметров в начале сеанса данного пользователя.

Существует 2 формата – полный и сокращенный.

Полный формат предполагает 3 группы параметров, каждая из которых пишется в квадратных скобках:
set2D$([x0, x1, y0, y1], ['xTitle', 'yTitle', 'title'], [0, 1, 12, 3, 5])$.

Первая кваадратная скобка определяет границы графика. Эта скобка должна обязательно присутствовать. В первой скобке можно указывать только 2 числа - границы по оси абсцисс, при этом

границы по вертикальной оси расчитываются автоматически.

Вторая кваадратная скобка это надписи к осям координат и подпись ко всему рисунку. Если в этой скобке 2 аргумента – то это первые два – подписи к осям, а если только один аргумент – то это название к рисунку.

Третья кваадратная скобка содержит 5 чисел:
1) 1 – означает: установить режим черно-белый (0 - цветной)
2) 1 – означает: установить равный масштаб по обеим осям (0-золотое сечение)
3) это размер шрифта для подписей
4) это толщина линий графиков
5) это толщина координатных осей

Есть для этой скобки и 3 сокращенных варианта:['ES'],['BW'],['ESBW']. Они, соответственно, устанавливают в значение 1 или первый параметр, или второй параметр, или оба.

Любая из 2х последних скобок может отсутствовать, могут отсутствовать и обе.

Существует 7 сокращеных вариантов для этой команды:
1) **set2D**();
2) **set2D**$(x0, x1)$;
3) **set2D**$(x0, x1, 'title')$;
4) **set2D**$(x0, x1, y0, y1)$;
5) **set2D**$(x0, x1, y0, y1, 'title')$;
6) **set2D**$(x0, x1, 'title', 'nameOX', 'nameOY')$;
7) **set2D**$(x0, x1, y0, y1, 'title', 'nameOX', 'nameOY')$.

Числа $x0$ и $x1$ ($x0 < x1$) задают интервал по оси OX. Числа $y0$ и $y1$ ($x0 < x1$) задают интервал по оси OY. Если эти параметры не заданы, то расчитываются автоматически. $nameOX$ — подпись на оси OX, $nameOY$ — подпись на оси OY, $title$ — заголовок графика.

Кроме этого, разрешается задать еще один или два ключа, которые должны стоять последними в списке параметров: BW и ES. BW указывает на построение черо-белого графика. ES указывает на равенство масштаба шкалы x масштабу шкалы . Всего имеется $7 * 4 = 28$ разных способов задания параметров окружения.

Характер линии, которая изображается на графике каждой из функций ($plot, tablePlot, paramPlot$) может быть разный: сплошная

линия, пунктирная линия и линия, которая оканчивается стрелкой. Для этого предназначены параметры: '*dash*' (пунктир), '*arrow*' (стрелки) и их сочетание '*dashAndArrow*', которые должны стоять в конце списка параметров этих функций.

Например, **plot**(x^2+1,'*dash*').

Если нескольким отдельным графикам присвоены имена, например, P=**plot**(x^2); Q=**tablePlot**([[1, 2], [3, 4]]); в этом случае их можно изобразить вместе с помощью команды **showPlots**([P, Q]).

У этой команды есть дополнительные опции 'noAxes' — не изображать оси координат и 'lattice' — изображать сетку. Например, **showPlots**([P, Q],'*lattice*').

Полученный график можно загрузить с сайта. Для этого под полем ввода нужно кликнуть на кнопку Загрузить, и файл с графиком будет загружен на компьютер пользователя.

3.1.1. Явное задание функции

Для построения графика функции $f = f(x)$ используется команда **plot**(f). Другие варианты команд:
1) **plot**(f, [$x0, x1$]), где [$x0, x1$] — интервал по оси OX;
2) **plot**(f, [$x0, x1$],'*options*'), где [$x0, x1$] — интервал по оси OX, 'options' — принимает следующие значения:
1)'dash' — график будет изображен пунктиром;
2)'arrow' — последняя точка графика будет нарисована со стрелкой;
3)'dashAndArrow' — график будет изображен пунктиром и последняя точка графика будет нарисована со стрелкой.
3) **plot**(f,'*options*').

Можно строить графики функций, содержащих параметры. Эти параметры необходимо определить как переменные в задании окружения (см. пример 3). Параметры могут принимать значения из отрезка $[0; 1]$. Сначала график строится для значений параметров, равных единице. Эти значения можно менять. Для этого надо выбрать имя параметра и передвинуть бегунок до нужного значения, затем нажать на кнопку Построить.

Пример 1.

```
SPACE = R64[x, y, z];
\set2D(-10, 10, -10, 10);
f = x^2 + \tg(x^2 - 1);
```

```
p = \plot(f);
```

Результат выполнения:
in: $f = x^2 + \mathrm{tg}(x^2 - 1)$;
out: рис. **??**.

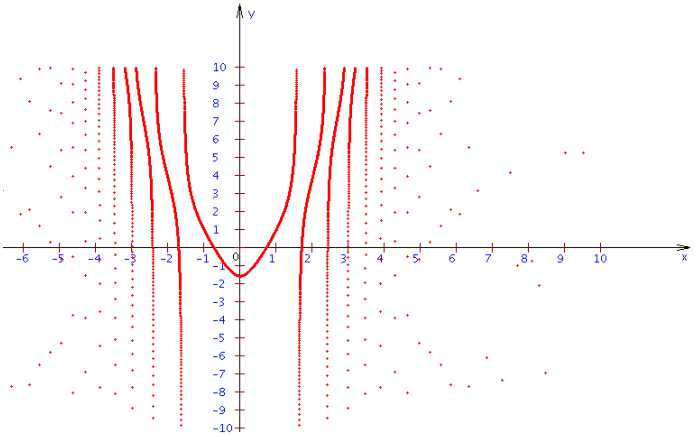


Рисунок 3.1: График функции $f = x^2 + \mathrm{tg}(x^2 - 1)$

Чтобы получить графики нескольких функций на одном рисунке нужно список этих функций заключить в квадратные скобки, как в следующем примере.

Пример 2.

```
SPACE = R64[x, y, z];
\set2D(-10, 10, -10, 10);
f = \sin(x);
p = \plot([f, \tg(x)]);
```

Результат выполнения:
in: $f = sin(x)$;
out: рис. **??**.

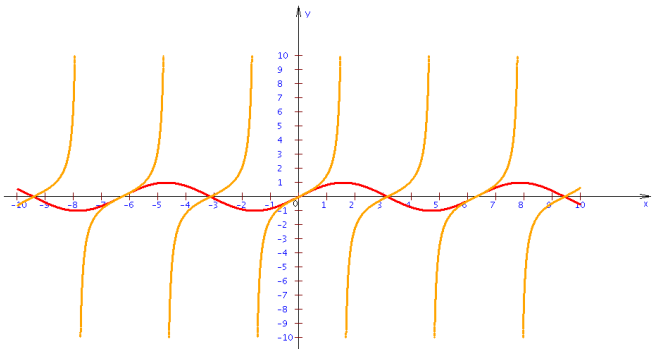


Рисунок 3.2: Графики функций $f = \sin(x)$ и $g = \mathrm{tg}(x)$

Пример 3.

```
SPACE = R64[x, y, z];
\set2D(-10, 10, 0, 2);
f = \unitBox(x,3);
p = \plot(f);
```

Пример 4.

```
SPACE = R64[x, a, b, c];
\set2D(0, 2\pi, 0, 2);
\plot(a\sin(bx) + c);
```

Пример 5.

```
SPACE = R64[x, y, z];
\set2D(-10, 10, -10, 10,'a','b','title');
f = x^2;
p = \plot(f);
```

Пример 6.

```
SPACE = R64[x, y, z];
\set2D(-10, 10, -10, 10);
f = x;
p = \plot(f,'dash');
```

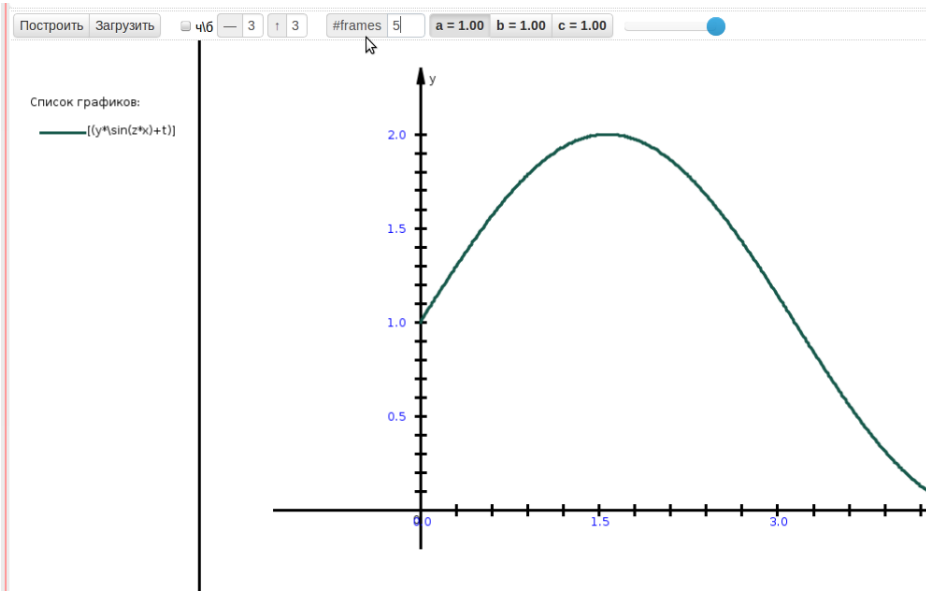
Пример 7.

```
SPACE = R64[x, y, z];
f = x;
p = \plot(f,[-5,5],'arrow');
```

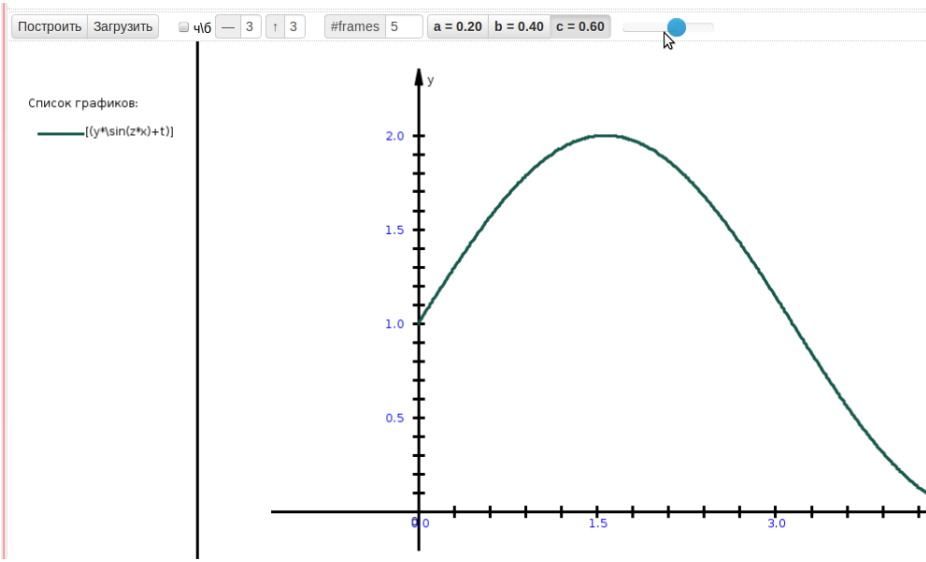
Пример 8.

```
SPACE = R64[x, y, z];
\set2D(-10, 10, -10, 10);
\plot([x,-x],'arrow');
```

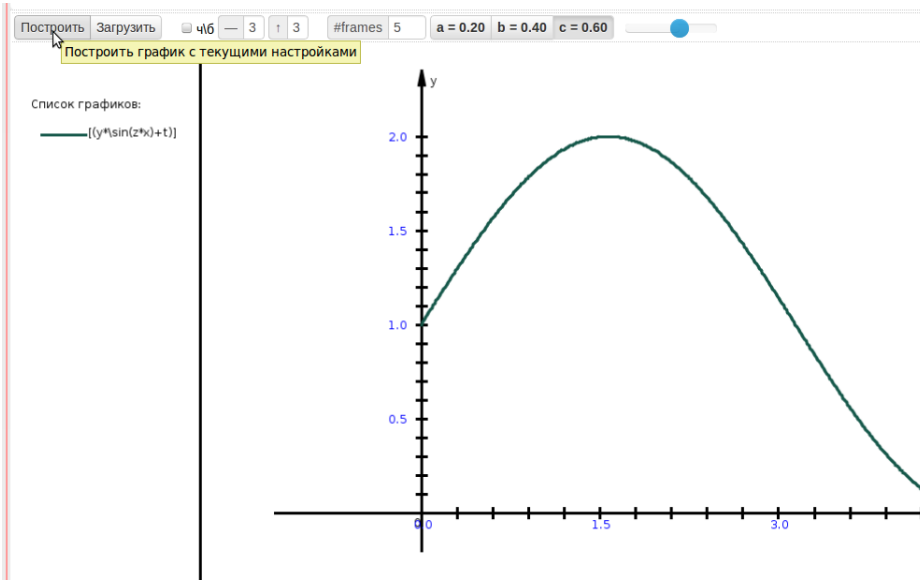
Для построения графика функции во времени с изменением параметров необходимо указать количество кадров - (например установим: frames = 5) (см. Рис.1.)



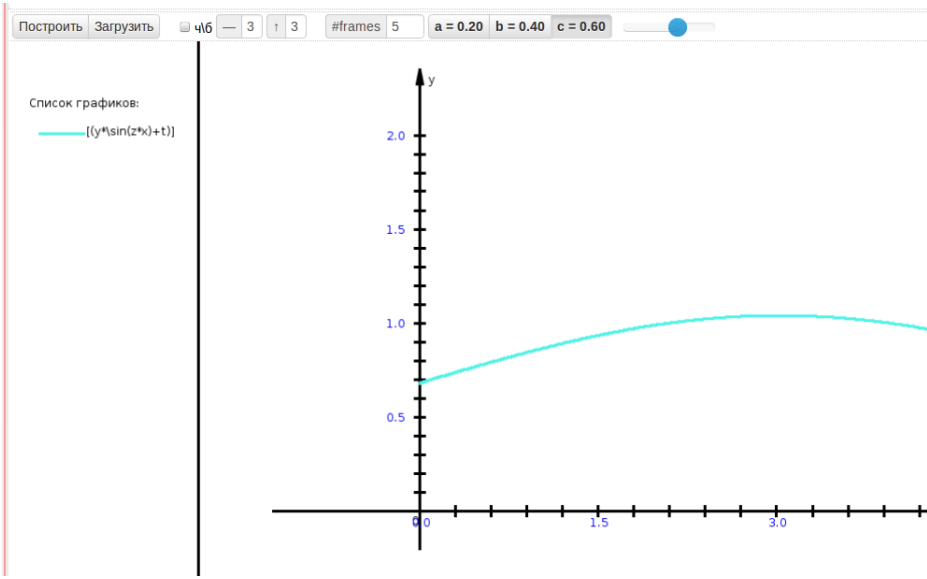
Для изменения параметров используйте ползунок - (например установим: a = 0.2, b = 0.4, c = 0.6) (см. Рис.2.)



Для построения графика нажимаем кнопку - 'Построить' (см. Рис.3.)



В результате получаем график.(см. Рис.4.)



3.1.2. Функции, заданные параметрически

Для построения графиков функций, которые заданы параметрически, необходимо выполнить команду $\backslash\mathbf{paramPlot}([f, g], [t0, t1])$, где $f = x(t)$, $g = y(t)$ — функции, заданная параметрически, $[t0, t1]$ — интервал значений для изменения параметра. Другой вариант команды: $\backslash\mathbf{paramPlot}([f, g], [t0, t1], 'options')$, где $[t0, t1]$ — интервал значений для изменения параметра, $'options'$ — принимает следующие значения:
1)$'dash'$ — график будет изображен пунктиром;
2)$'arrow'$ — последняя точка графика будет нарисована со стрелкой;
3)$'dashAndArrow'$ — график будет изображен пунктиром и последняя точка графика будет нарисована со стрелкой.

Пример 1.

```
SPACE = R64[x, y, z];
g = \sin(x);
k = \cos(x);
```

```
f = \paramPlot([g, k], [0, 2\pi]);
```

Результат выполнения:
in: $g = sin(x); k = cos(x);$
out: рис. **??**.

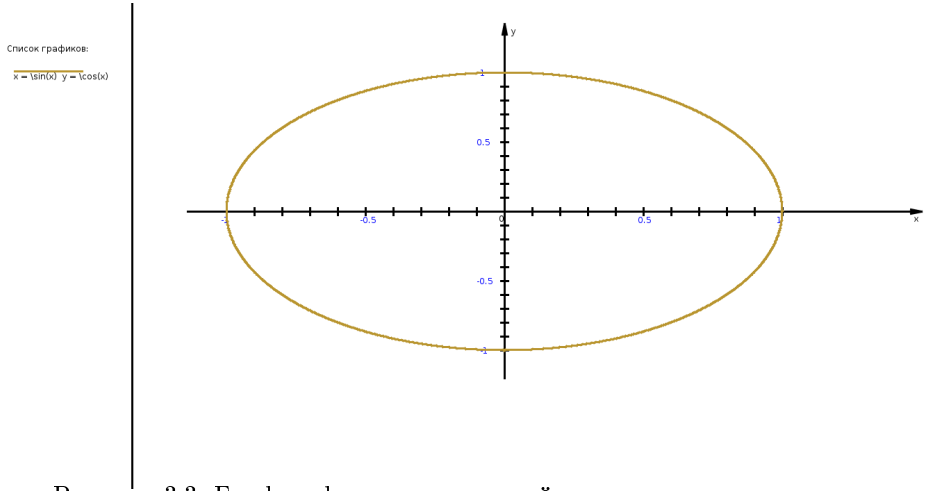
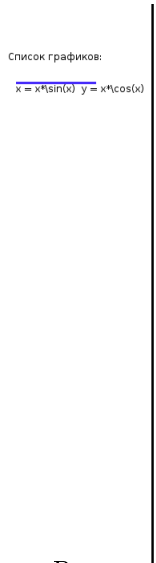


Рисунок 3.3: График функции, заданной параметрически

Пример 2.

```
SPACE = R64[x, y, z];
g = x\sin(x);
k = x\cos(x);
f = \paramPlot([g, k], [0, 5\pi]);
```

Результат выполнения:
in: $g = x\sin(x); k = x\cos(x);$
out: рис. ??.



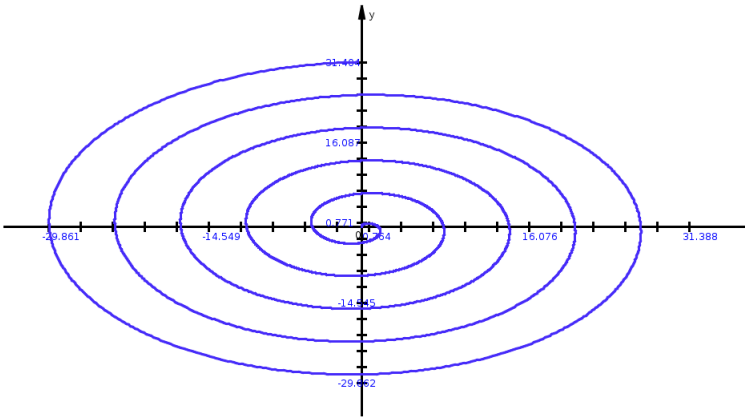


Рисунок 3.4: График функции, заданной параметрически

Пример 3.

```
SPACE = R64[x, y, z];
g = 2\cos(x)+\cos(2x);
k = 2\sin(x)-\sin(2x);
f = \paramPlot([g, k], [0, 2\pi]);
```

Результат выполнения:
in: $g = 2\cos(x) + \cos(2x); k = 2\sin(x) - \sin(2x);$
out: рис. ??.

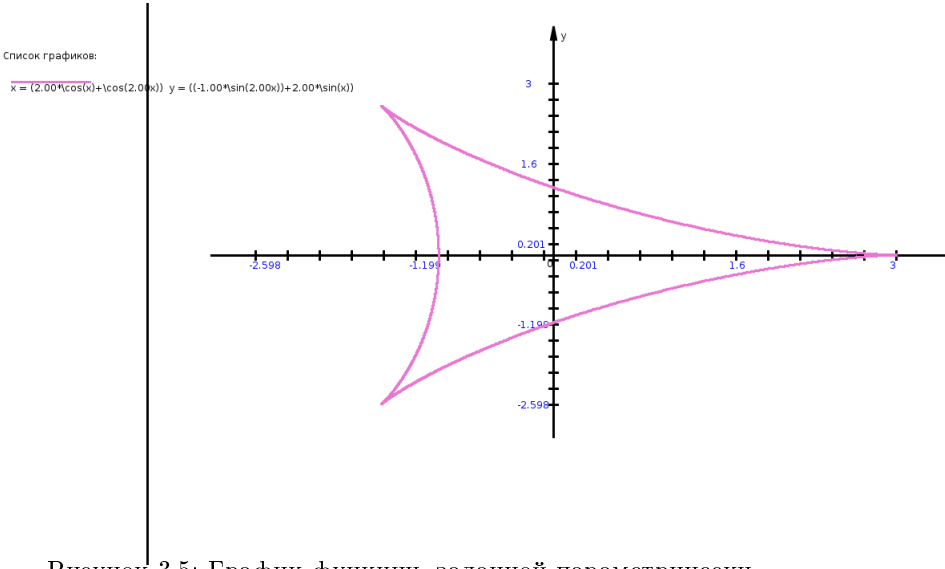


Рисунок 3.5: График функции, заданной параметрически

Пример 4.

```
SPACE = R64[x, y, z];
g = 2\sin(x)^3;
k = 2\cos(x)^3;
f = \paramPlot([g, k], [0, 2\pi]);
```

Результат выполнения:
in: $g = 2\sin(x)^3; k = 2\cos^3(x)$
out: рис. ??.

Пример 5.

```
SPACE = R64[x, y, z];
g = (1+\cos(x))\cos(x);
k = (1+\cos(x))\sin(x);
f = \paramPlot([g, k], [0, 2\pi]);
```

Результат выполнения:
in: $g = (1 + \cos(x))\cos(x); k = (1 + \cos(x))\sin(x);$
out: рис. ??.

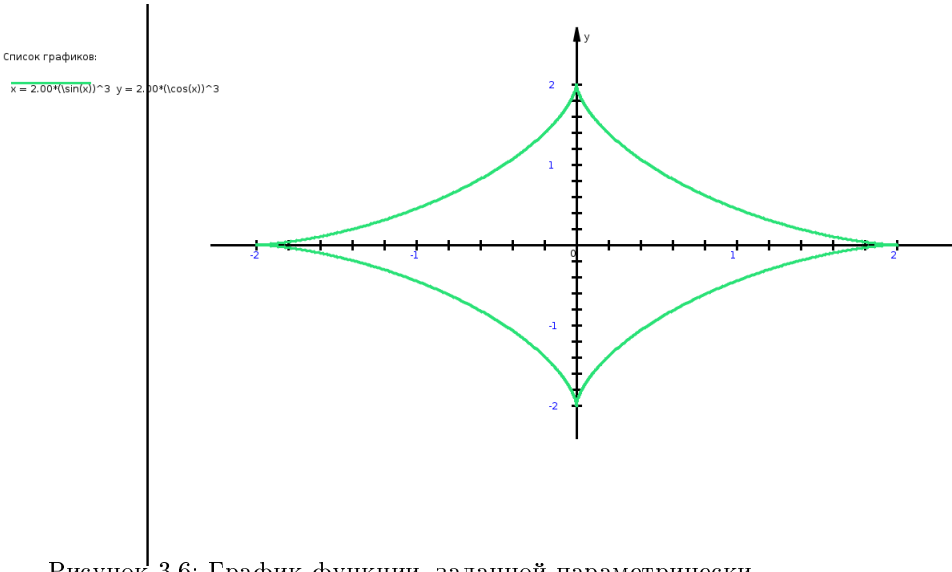


Рисунок 3.6: График функции, заданной параметрически

Пример 6.

```
SPACE = R64[x, y, z];
g = \sin(x)(\exp(\cos(x))-2\cos(4x)+\sin(x/12)^5);
k = \cos(x)(\exp(\cos(x))-2\cos(4x)+\sin(x/12)^5);
f = \paramPlot([g, k], [0, 12\pi]);
```

Результат выполнения:
in: $g = \sin(x)(\exp(\cos(x)) - 2\cos(4x) + \sin^5(x/12))$;
$k = \cos(x)(\exp(\cos(x)) - 2\cos(4x) + \sin^5(x/12))$;
out: рис. **??**.

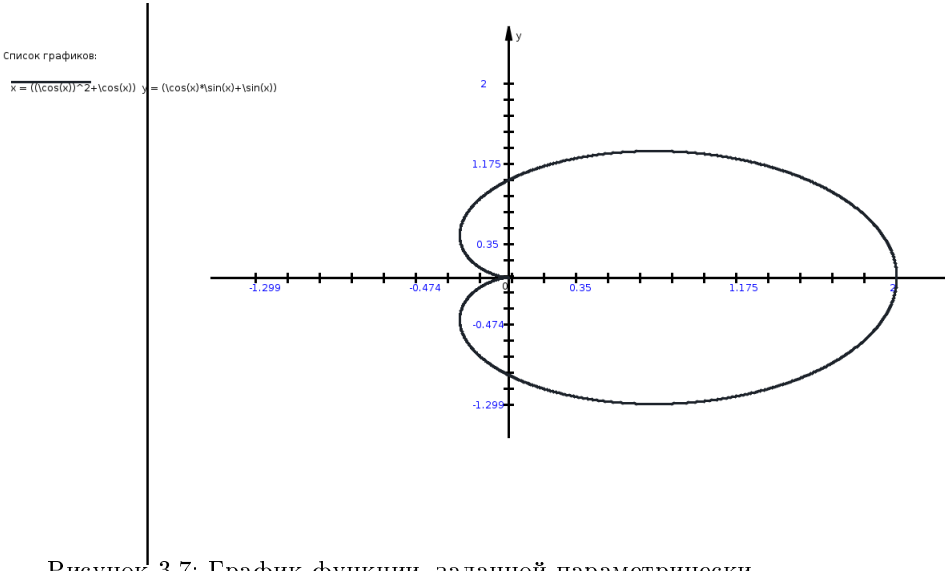


Рисунок 3.7: График функции, заданной параметрически

Пример 7.

```
SPACE = R64[x, y, z];
\set2D('','','','','x(t)','y(t)','paramPlot');
g = \sin(x)(\exp(\cos(x))-2\cos(4x)+\sin(x/12)^5);
k = \cos(x)(\exp(\cos(x))-2\cos(4x)+\sin(x/12)^5);
f = \paramPlot([g, k], [0, 12\pi]);
```

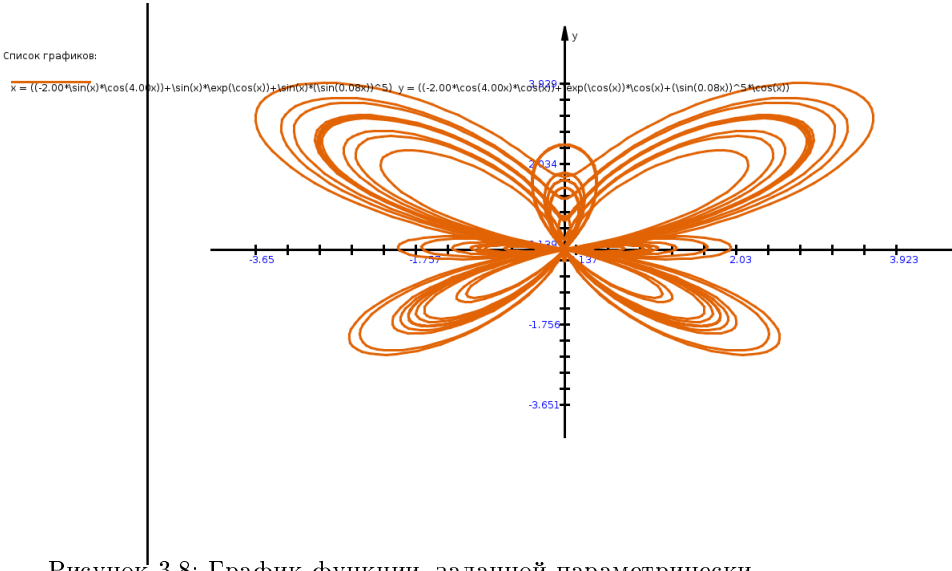


Рисунок 3.8: График функции, заданной параметрически

Пример 8.

```
SPACE = R64[x, y, z];
g = \sin(x);
k = \cos(x);
f = \paramPlot([g, k], [0, 2\pi],'dashAndArrow');
```

3.1.3. Функции, которые заданы таблицей значений

Для построения графиков функций, заданных табличными значениями, необходимо выполнить команду: **tablePlot**($[[x_1, \ldots, x_n], [y_{11}, \ldots, a_{1n}], \ldots, [y_{k1}, \ldots, a_{kn}]]$) Другой вариант команды: **tablePlot**($[[x_1, \ldots, x_n], [y_{11}, \ldots, a_{1n}], \ldots, [y_{k1}, \ldots, a_{kn}]], '$*options*$'$,где $'$*options*$'$ — принимает следующие значения:
1)$'$*dash*$'$ — график будет изображен пунктиром;
2)$'$*arrow*$'$ — последняя точка графика будет нарисована со стрелкой;

3)*'dashAndArrow'* — график будет изображен пунктиром и последняя точка графика будет нарисована со стрелкой.

Пример 1.

```
SPACE = R64[x, y, z];
 \tablePlot(
   [
     [0, 1, 2, 3, 4, 5],
     [0, 1, 4, 9, 16, 25],
     [0, -1, -2, -3, -4, -5],
     [0, 4, 8, 12, 16, 20]
   ]);
```

Результат выполнения:

in:

out: рис. ??.

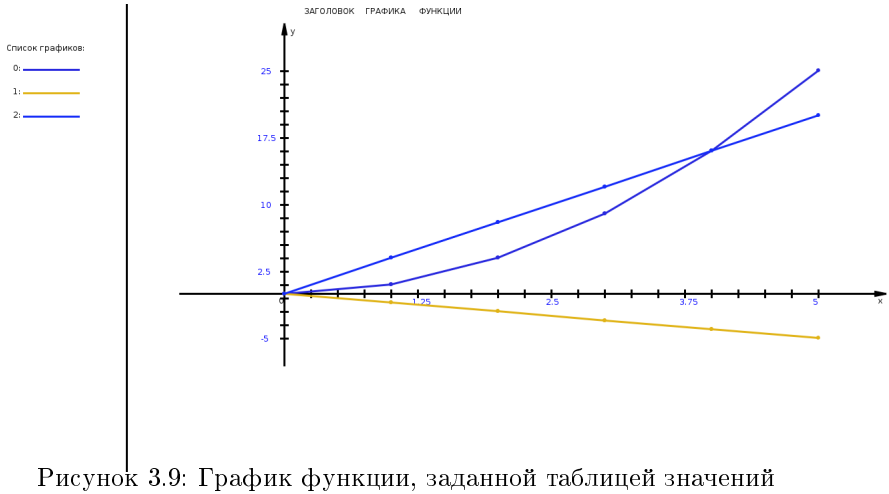


Рисунок 3.9: График функции, заданной таблицей значений

Пример 2.

```
SPACE=R64[x];
\set2D(-1,5,-10,10);
"Пусть задана табличная функция:"
A=[[0, 1, 2, 3,  4, 5],    [3, 0, 4, 10, 5, 10]]; t=\table(A);
```

```
"Аппроксимируем эту функцию полиномом 4-й степени:" p=\approx(t,4);
"Построим график полинома:" P=\plot(p,[1,5]);
"Построим график табличной функции:" T=\tablePlot(t);
"Построим оба графика в одной системе координат:" \showPlots([P,T]);
\print(p);
```

Результат выполнения:

in:

out: $0.54x^4 - 5.64x^3 + 18.38x^2 - 17.28x + 3.17$

рис. ??.

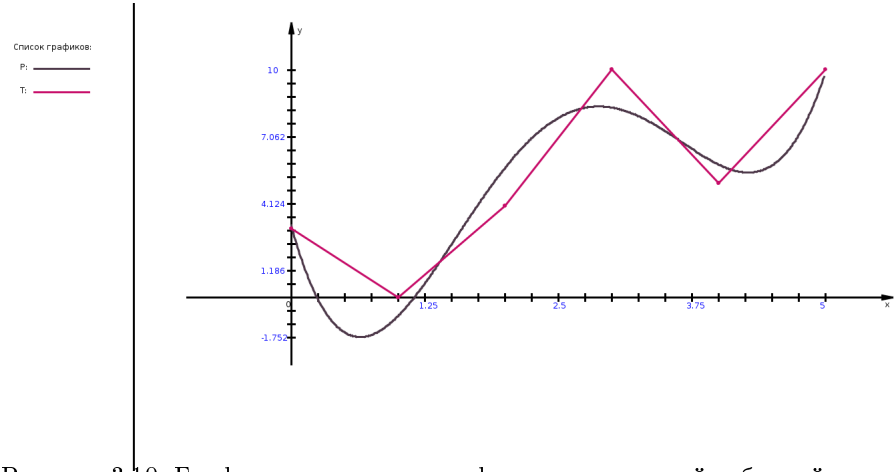


Рисунок 3.10: График аппроксимации функции, заданной таблицей значений

Пример 3.

```
SPACE = R64[x, y, z];
\set2D(-10, 10, -10, 10, '','', 'Header of Graphics');
 \tablePlot(
   [[-3, -6, -6, -3, 3, 6, 6, 3, -3],
    [6, 3, -3, -6, -6, -3, 3, 6, 6]]);
```

Пример 4.

```
SPACE = R64[x, y, z];
 \tablePlot(
   [[-3, -6, -6, -3, 3, 6, 6, 3, -3],
    [6, 3, -3, -6, -6, -3, 3, 6, 6]],'arrow');
```

Пример 5.

```
SPACE = R64[x, y, z];
 \tablePlot(
   [[-3, -6, -6, -3, 3, 6, 6, 3, -3],
    [6, 3, -3, -6, -6, -3, 3, 6, 6]], 'dash');
```

Пример 6.

```
SPACE = R64[x, y, z];
 \tablePlot(
   [[-3, -6, -6, -3, 3, 6, 6, 3, -3],
    [6, 3, -3, -6, -6, -3, 3, 6, 6]], 'dashAndArrow');
```

3.1.4. Функции, которые заданы таблицей значений по точкам

Для построения графиков функций по точкам, заданных табличными значениями, необходимо выполнить команду: **\pointsPlot**$([[x_1,\ldots,x_n],[y_1,\ldots,y_n]],[s_1,\ldots,s_n],[kv_1,\ldots,kv_n],[kg_1,\ldots,kg_n])$, где s_n — подпись точки, kv_n — коэффициент поворота вокруг точки (принимает значения от 0 до 7, и означает поворот на (kv_n * 45) градусов), kg_n — коэффициент смещения вдоль оси OX (если отрицательный то смещение происходит влево). Сокращенные варианты команды: **\pointsPlot**$([[x_1,\ldots,x_n],[y_1,\ldots,y_n]],[s_1,\ldots,s_n])$ или **\pointsPlot**$([[x_1,\ldots,x_n],[y_1,\ldots,y_n]],[s_1,\ldots,s_n],[kv_1,\ldots,kv_n])$ или **\pointsPlot**$([[x_1,\ldots,x_n],[y_1,\ldots,y_n]],[s_1,\ldots,s_n],[kv_1,\ldots,kv_n],[kg_1,\ldots,kg_n])$.

Пример 1.

```
\set2D(-10, 10, -10, 10);
\pointsPlot(
   [[0, 1, 2],
     [0, 1, 4]],['a','b','c']);
```

Пример 2.

```
\pointsPlot(
  [
    [0, 1, 2],
    [0, 1, 4]],['a','b','c'],[0,2,4]);
```

Пример 3.

```
\pointsPlot(
  [
    [0, 1, 2],
    [0, 1, 4]],['a','b','c'],[0,2,4],[0,-5,5]);
```

Пример 4.

```
\pointsPlot(
  [
    [0, 1, 2],
    [0, 1, 4]]);
```

Пример 5.

```
SPACE = R64[x, y, z];
f1=\tablePlot([[1, 1], [1, 5]]);
f2=\tablePlot([[1, 5], [1, 1]]);
f3=\tablePlot([[5, 5], [1, 5]]);
f4=\tablePlot([[1, 5], [5, 5]]);
f5=\pointsPlot([[1, 1, 5, 5],[1, 5, 5, 1]],['A','B','C','D'],[6,0,0,2
\showPlots([f1, f2, f3, f4, f5]);
```

Пример 6.

```
SPACE = R64[x, y, z];
f1=\tablePlot([[1, 1], [1, 5]]);
f2=\tablePlot([[1, 5], [1, 1]]);
f3=\tablePlot([[5, 5], [1, 5]]);
f4=\tablePlot([[1, 5], [5, 5]]);
f5=\pointsPlot([[1, 1, 5, 5],[1, 5, 5, 1]],['A','B','C','D'],[6,0,0,2
\showPlots([f1, f2, f3, f4, f5],'noAxes');
```

3.1.5. Построение разных графиков функций в одной системе координат

Для построения графиков функций, заданных разными способами, необходимо сначала построить график каждой функции, а затем выполнить команду **showPlots**($[f_1, f_2, \ldots, f_n]$). Другие варианты команды: **showPlots**($[f1, f2, f3, f4]$,$'noAxes'$), где 'noAxes' — параметр, указывающий на изображение графика без осей. или **showPlots**($[f1, f2, f3, f4]$,$'lattice'$), где 'lattice' — параметр, указывающий на изображение графика с решеткой.

Пример 1.

```
SPACE = R64[x, y, z];
\set2D(-20, 20, -20, 20);
f1 = \plot(\tg(x));
f2 = \tablePlot([[0, 1, 4, 9, 16, 25], [0, 1, 2, 3, 4, 5]]);
f3 = \paramPlot([\sin(x), \cos(x)], [-10, 10]);
f4=\tablePlot([[0, 1, 4, 9, 16, 25], [0, -1, -2, -3, -4, -5]]);
\showPlots([f1, f2, f3, f4]);
```

Результат выполнения:
in:
out: рис. **??**.

Пример 2.

```
p1=\tablePlot([[-1, -3, 3, 3, -3, -3],[4, 3, 3, -3 , -3, 3]]);
p2=\tablePlot([[5, 5, 3, 3, 5, -1],[4, -2, -3, 3, 4, 4]]);
p3=\tablePlot([[-3, -1, -1],[-3, -2, 4]], 'dash');
p4=\tablePlot([[-1, 5],[-2, -2]], 'dash');
\showPlots([p1,p2,p3,p4], 'noAxes');
```

Пример 3.

```
p1=\tablePlot([[-1, -3, 3, 3, -3, -3],[4, 3, 3, -3, -3, 3]]);
p1p=\pointsPlot([[-1, -3, 3, 3, -3],[4, 3, 3, -3 , -3]],
['F','B','C','D','A'],[0,0,0,4,4]);
p2=\tablePlot([[5, 5, 3, 3, 5, -1],[4, -2, -3, 3, 4, 4]]);
p3=\tablePlot([[-3, -1, -1],[-3, -2, 4]],'dash');
p4=\tablePlot([[-1, 5],[-2, -2]],'dash');
p2p=\pointsPlot([[5, 5, -1 ],[4, -2, -2]],['G','H','E'],[0,4,4]);
\showPlots([p1,p2,p3,p4,p1p,p2p], 'noAxes');
```

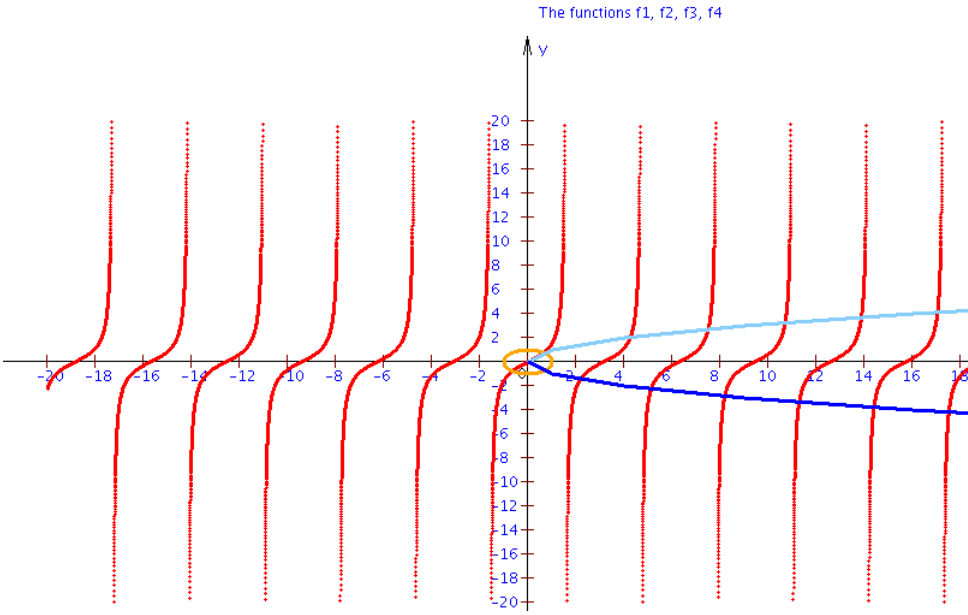


Рисунок 3.11: Графики функций, заданных разными способами

3.1.6. Построение графов

Для построения графов необходимо выполнить команду **plotGraph**($[[a_{11},\ldots,a_{1n}],\ldots,[a_{n1},\ldots,a_{nn}]],[[x_1,\ldots,x_n],[y_1,\ldots,y_n]]$), где $[[a_{11},\ldots,a_{1n}],\ldots,[a_{n1},\ldots,a_{nn}]]$ — матрица смежности, $[[x_1,\ldots,x_n],[y_1,\ldots,y_n]]$ — матрица координат.

Пример 1.

```
\plotGraph([[0,1,1,0,1,0],[1,0,0,1,1,0],[1,0,0,0,1,1],[0,1,0,0,0,0],
[1,1,1,0,0,1],[0,0,1,0,1,0]],[[3,2,4,1,3,5],[3,2,2,1,1,1]]);
```

Результат выполнения:
in:
out: рис. **??**.

Кроме того, можно выполнить команду лишь с первым параметром **plotGraph**($[[a_{11},\ldots,a_{1n}],\ldots,[a_{n1},\ldots,a_{nn}]]$), где $[[a_{11},\ldots,a_{1n}],\ldots,[a_{n1},\ldots,a_{nn}]]$ — матрица смежности.

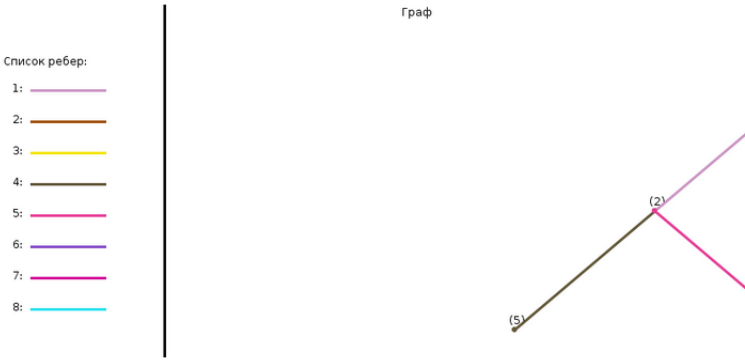


Рисунок 3.12: Граф

Пример 2.

```
\plotGraph([[0,1,1,0,1,0],[1,0,0,1,1,0],[1,0,0,0,1,1],[0,1,0,0,0,0],
[1,1,1,0,0,1],[0,0,1,0,1,0]]);
```

Результат выполнения:

in:
out: рис. ??.

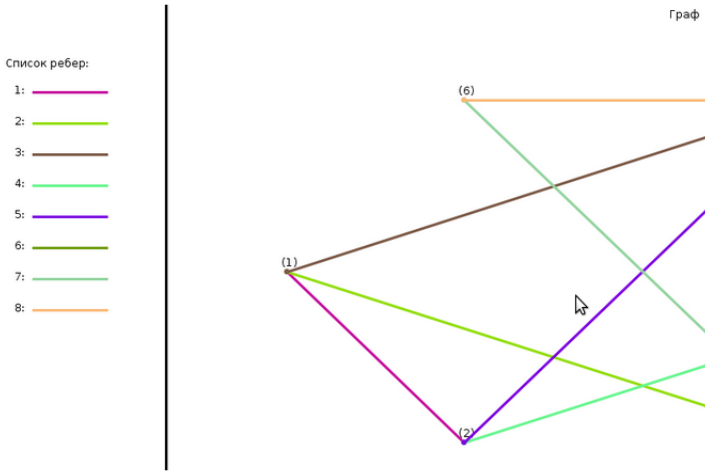


Рисунок 3.13: Граф

Можно выполнить команду с одним числовым параметром **plotGraph**(N), где N — количество вершин в графе.

Пример 3.

```
\plotGraph(6);
```

Результат выполнения:
in:
out: рис. **??**.

3.1.7. Текст на графиках

Для того, чтобы делать любые виды надписей используется функция **textPlot**()

Для задания одной надписи записывают в кваратных скобках следуюшие параметры: ['str',sizeText,xCor,yCor,alpha]

где str - это текст; sizeText - размер шрифта; xCor, yCor - координаты на экране первой буквы текста, alpha - угол наклона текста (по умолчанию, если это параметр не указан, то он равен нулю).

В одной команде можно определить сколько угоно надписей, разделяя их запятыми:
textPlot([], [], [], ...[]).

3.2. Построение 3D графиков функций

Окружение для построения 3D графиков задается командой **set3D**()

Существует несколько вариантов для этой команды:
1) **set3D**($x0, x1, y0, y1, z0, z1$);
2) **set3D**($x0, x1, y0, y1, z0, z1, gridSize$);
3) **set3D**($x0, x1, y0, y1, z0, z1, gridSize, framesNumber$);
4) **set3D**($x0, x1, y0, y1, z0, z1, gridSize, framesNumber, [a1, a2, ...]$);

Числа $x0$ и $x1$ ($x0 < x1$) задают интервал по оси OX. Числа $y0$ и $y1$ ($y0 < y1$) задают интервал по оси OY. Числа $z0$ и $z1$ ($z0 < z1$) задают интервал по оси OZ. $gridSize$ отвечает за размер сетки параллелепипеда в котором строится график. $framesNumber$ отвечает за количество кадров при построении графика с параметрами, изменение которых можно наблюдать в виде изменения кадров. $a1$ и $a2$ отвечают за конечное значение параметров функции. Эти значения будут выставлены в ползунки, которые появляются под текстом

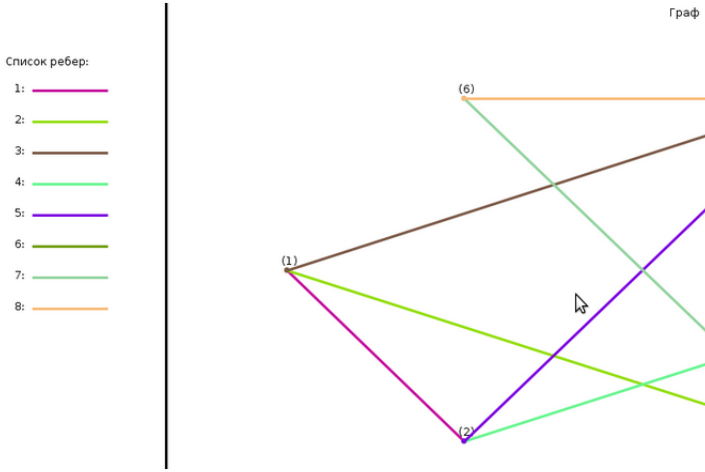


Рисунок 3.14: Граф

з запросом пользователя. При изменении параметров во время изменения кадров, их значения будут меняться в диапазоне от 1.0 до $a1$ для первого параметра.

3.2.1. Явное задание функции. Построение на сервере

Mathpar позволяет строить 3D графики функций, которые заданы явно.

Для построения 3D графика функции $f = f(x, y)$ используется команда $\backslash$**plot3d**$(f, [x0, x1, y0, y1])$, где $[x0, x1]$ — интервал по оси OX, $[y0, y1]$ — интервал по оси OY.

Кроме того, полученные графики можно вращать и масштабировать: увеличивать либо уменьшать.

Перемещение мыши с нажатой левой кнопкой приводит к вращению системы координат графика. После остановки происходит перерисовка графика в новой повернутой системе координат.

Перемещение мыши с нажатой левой кнопкой и нажатой клавишей *Shift* приводит к изменению масштаба изображения. После остановки перемещения происходит перерисовка графика в новом масштабе.

Пример.

```
f = x^2 / 20 + y^2 / 20;
\plot3d(f, [-20, 20, -20, 20]);
```

```
\plot3d([x / 20 + y^2 / 20, x^2 / 20 + y / 20], [-20, 20, -20, 20]);
```

```
SPACE = R64[x, y, a, b];
f = ax^2 / 20 + by^2 / 20;
\plot3d(f, [-20, 20, -20, 20]);
```

Для построения графика функции во времени с изменением параметров необходимо указать количество кадров - (например установим: frames = 5) (см. Рис.1.)

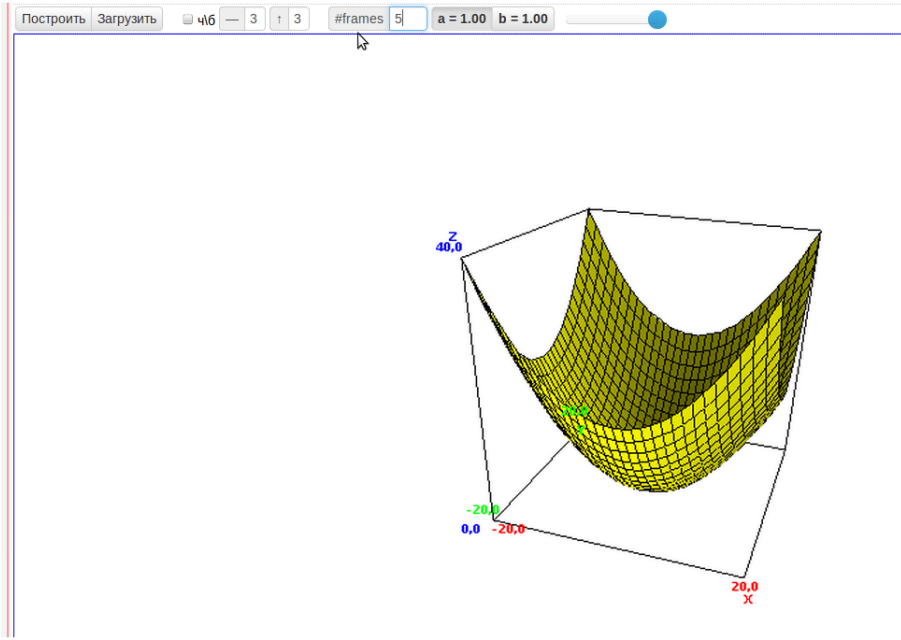


Рис. 1.

Для изменения параметров используйте ползунок - (например установим: a = 0.7, b = 0.24) (см. Рис.2.)

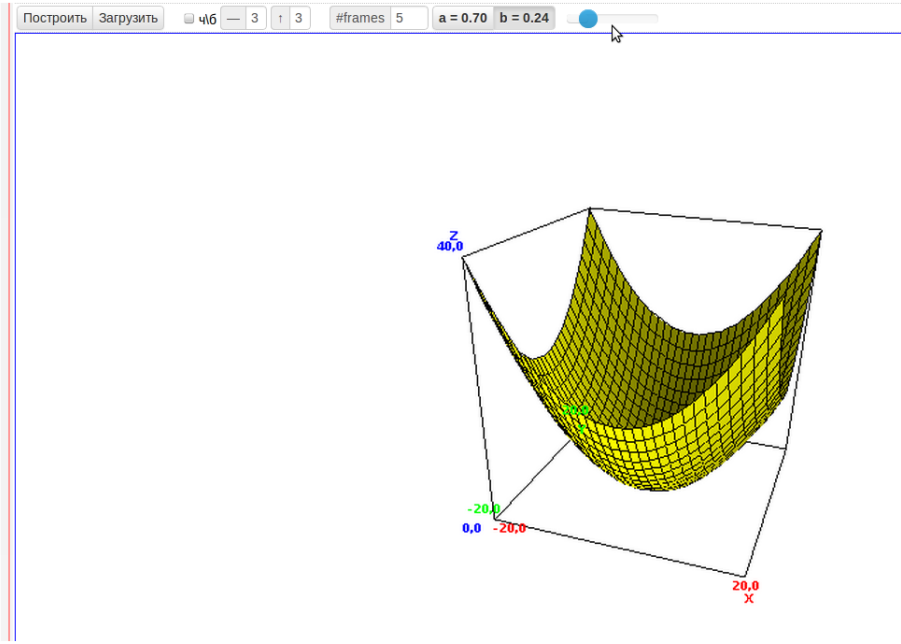


Рис. 2.

Для построения графика нажимаем кнопку - 'Построить' (см. Рис.3.)

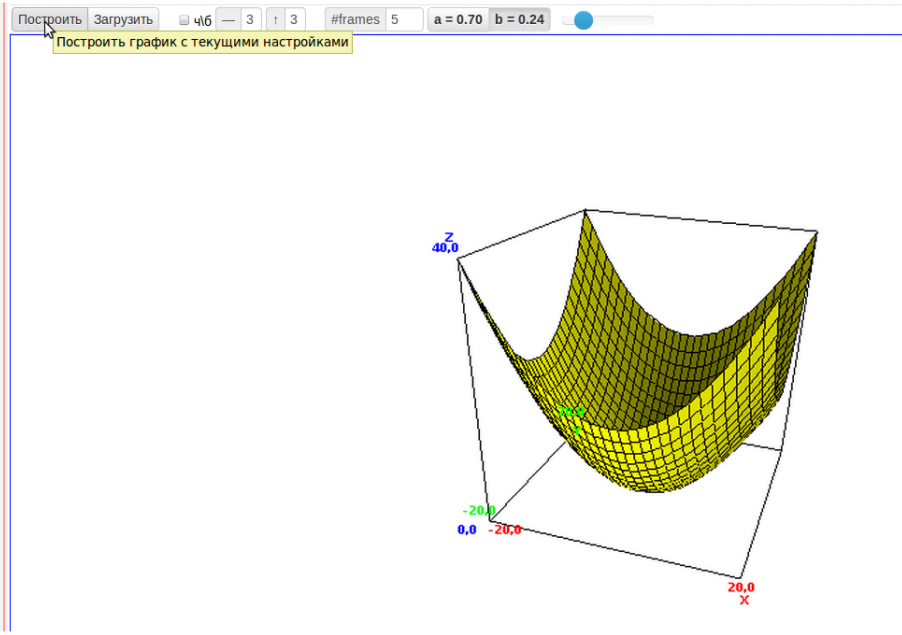


Рис. 3.

В результате получаем график.(см. Рис.4.)

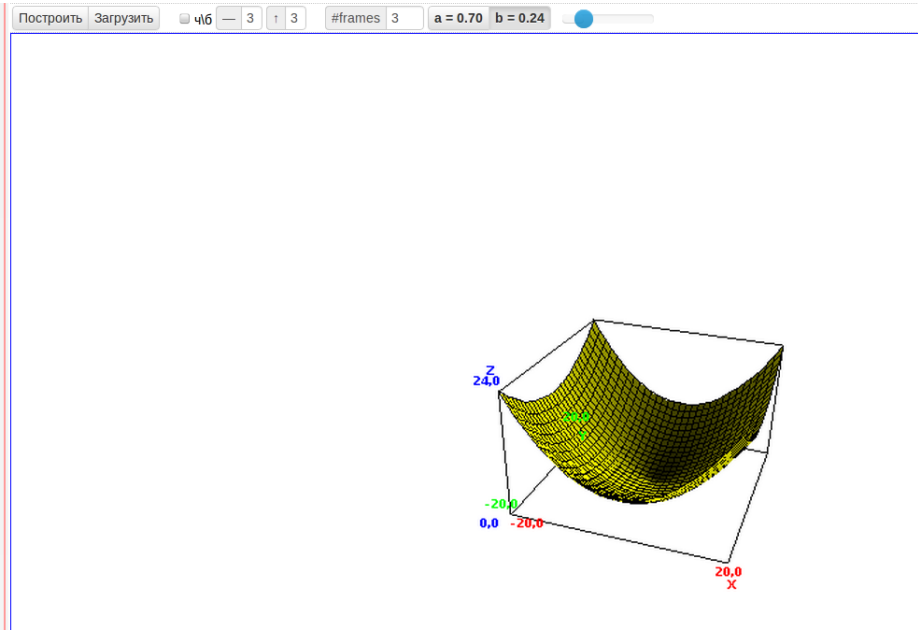


Рис. 4.

Результат выполнения:
in: $f = x^2/20 + y^2/20$;
$plot3d(f, [-20, 20, -20, 20])$;
$plot3d([x/20 + y^2/20, x^2/20 + y/20], [-20, 20, -20, 20])$;
out: рис. **??**.

3.2.2. Явное задание функции. Построение на стороне пользователя

Также графики функций, которые заданы явно можно построить командой $\backslash$**explicitPlot3d**$(f, xMin, xMax, yMin, yMax, zMin, zMax)$, где числа $xMin, xMax, yMin, yMax, zMin, zMax$ задают область в пространстве, имеющую форму параллелепипеда, в которой изображается явная функция.

Допускаются, кроме того, следующий набор аргументов: $(f, xMin, xMax, yMin, yMax, zMin, zMax, gridSize)$, где $gridSize$

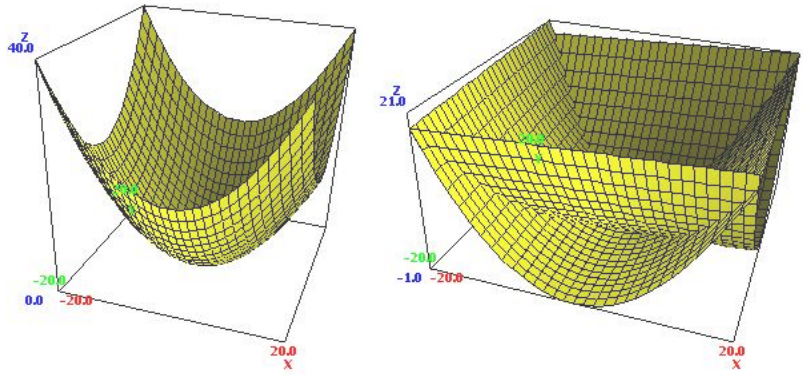


Рисунок 3.15: Построение 3D графиков функций

отвечает за размер сетки параллелепипеда в котором строится график.

Можно задавать только одну функцию, следующим образом: **explicitPlot3d**(f), в этом случае предполагается, что будет изображена функция f в кубе $20 \times 20 \times 20$, центр которого располагается в начале координат.

Используя **explicitPlot3d**() вы можете вращать систему координат перемещая указатель мышки с нажатой левой клавишей. Также вы можете сдвигать начало системы координат перемещая указатель мышки с нажатой правой клавишей.

Для построения графика функции во времени с изменением параметров необходимо указать сначала количество кадров. Затем следует выставить конечное значение параметров используя ползунки. Для построения графика необходимо нажать кнопку - 'Построить'. Значение количества кадров и параметры на ползунках можно указать через **set3D**().

Пример.

```
SPACE = R64[x, y];
f = (x^2+y^2)/20;
ePl=\explicitPlot3d(f, -10, 10, -10, 10, -10, 10, 40);

SPACE = R64[x, y, a];
```

```
\set3D(-10, 10, -10, 10, -10, 10, 40, 25, [0.2]);
f = (ax^2+y^2)/20;
ePl=\explicitPlot3d(f);
```

3.2.3. Построение 3D графиков функций, которые заданы параметрически. Построение на сервере

Mathpar позволяет строить 3D графики, которые заданы параметрически

Для построения графика необходимо передать 3 функции $f(x,y)$, $g(x,y)$ и $h(x,y)$, используя команду **paramPlot3d**($[[f],[g],[h]],[x0,x1,y0,y1]$), где $[x0,x1]$ — интервал по оси OX, $[y0,y1]$ — интервал по оси OY.

Сфера

```
SPACE=R64[u,v];
\paramPlot3d([[\cos(u)\cos(v)], [\sin(u)\cos(v)], [\sin(v)]],
[-\pi, \pi, -\pi/2, \pi/2]);
```

Тор

```
SPACE=R64[u,v];
\paramPlot3d([[\cos(u)(\cos(v)+3)],[\sin(u)(\cos(v)+3)],[\sin(v)]],
[-\pi, \pi, -\pi, \pi]);
```

Спираль

```
SPACE=R64[u,v];
\paramPlot3d([[\cos(u)(\cos(v)+3)],[\sin(u)(\cos(v)+3)],[\sin(v)+u]],
[-2\pi, 2\pi, -\pi, \pi]);
```

Логарифмическая спираль

```
SPACE=R64[u,v];
\paramPlot3d([[u\cos(u)(\cos(v)+1)],[u\sin(u)(\cos(v)+1)],[u\sin(v)]]
[0, 3\pi, -\pi, \pi]);
```

"Морская раковина"

```
SPACE=R64[u,v];
\paramPlot3d([[u\cos(u)(\cos(v)+1)],[u\sin(u)(\cos(v)+1)],
[u\sin(v)-(((u+3)/8)\pi)^2-20]], [0, 8\pi, -\pi, \pi]);
```

Трилистник

```
SPACE=R64[u,v];
\paramPlot3d([[\cos(u)\cos(v)+3\cos(u)(1.5+\sin(1.5u/2))],
[\sin(u)\cos(v)+3\sin(u)(1.5+\sin(1.5u/2))],[\sin(v)+2\cos(1.5u)]],
[-2\pi, 2\pi, -\pi, \pi]);
```

Поверхность Дини

```
SPACE=R64[u,v];
\paramPlot3d([[\cos(u)\sin(v)],[\sin(u)\sin(v)],
[\cos(v)+\lg(\tg(v/2))+0.2u-4]], [0, 4\pi, 0.0001, 2]);
```

Лента Мёбиуса

```
SPACE=R64[u,v];
\paramPlot3d([[(1+v/2\cos(u/2))\cos(u)],[(1+v/2\cos(u/2))\sin(u)],
[v/2\sin(u/2)]], [0, 2\pi, -1, 1]);
```

Куб

```
SPACE=R64[u,v];
\paramPlot3d([[u,v,5,u,v,-5],[v,5,u,v,-5,u],[5,u,v,-5,u,v]],
[-5, 5, -5, 5]);
```

Цилиндр

```
SPACE=R64[u,v];
\paramPlot3d([[5\cos(u)],[5\sin(u)],[v]], [-5, 5, -5, 5]);
```

Конус

```
SPACE=R64[u,v];
\paramPlot3d([[\cos(u) * (5 * (1 - v/6))],[\sin(u) * (5 * (1 - v/6))]
[-6, 6, 0, 6]);
```

Усеченный конус

```
SPACE=R64[u,v];
\paramPlot3d([[\cos(u) * (5 * (1 - v/6) + 1 * v/6)],
[\sin(u) * (5 * (1 - v/6) + 1 * v/6) ],[v]], [-5, 5, -5, 5]);
```

Песочные часы

```
SPACE=R64[u,v];
\paramPlot3d([[\cos(u) * (5 * (0.5 - v/6) + 0.01*v/6)],
[\sin(u) * (5 * (0.5 - v/6) + 0.01*v/6)],[v]], [0, 2\pi, 0, 2\pi]);
```

3.2.4. Построение 3D графиков функций, которые заданы параметрически. Построение на стороне пользователя

Также графики функций, которые заданы параметрически можно построить командой **\parametricPlot3d**$(f, g, h, uMin, uMax, vMin, vMax, gridSize)$, где указаны 3 функции $f(u, v)$, $g(u, v)$ и $h(u, v)$ для осей OX, OY, OZ Числа $uMin, uMax, vMin, vMax$ задают диапазон для параметров функций f, g и h. $gridSize$ отвечает за размер сетки параллелепипеда в котором строится график.

Вы можете вращать систему координат перемещая указатель мышки с нажатой левой клавишей. Также вы можете сдвигать начало системы координат перемещая указатель мышки с нажатой правой клавишей.

Для построения графика функции во времени с изменением параметров необходимо указать сначала количество кадров. Затем следует выставить конечное значение параметров используя ползунки. Для построения графика необходимо нажать кнопку - 'Построить'. Значение количества кадров и параметры на ползунках можно указать через **\set3D**().

Тор с параметром

```
SPACE = R64[u, v, a];
X=\cos(u)*(3+\cos(v));
Y=\sin(u)*(3+\cos(v));
Z=a*\sin(v);
\parametricPlot3d(X, Y, Z, 0, 7, 0, 7, 64);
```

Спираль с параметром

```
SPACE = R64[u, v, b];
X=\cos(u)*(\cos(v)+2);
Y=\sin(u)*(\cos(v)+4b);
Z=\sin(v)+u/2+1;
\parametricPlot3d(X,Y,Z, -6.3, 6.3, -3.15, 3.15, 64);
```

3.2.5. Построение 3D графиков функций, которые заданы неявно

Mathpar позволяет строить 3D графики неявно заданных функций.

Для построения графика функции $f(x, y, z) = 0$ используется команда

\implicitPlot3d$(f, xMin, xMax, yMin, yMax, zMin, zMax)$,

где числа $xMin, xMax, yMin, yMax, zMin, zMax$ задают область в пространстве, имеющую форму параллелепипеда, в которой изображается неявная функция.

Можно задавать только одну функцию, следующим образом:

\implicitPlot3d(f),

в этом случае предполагается, что будет изображена функция f в кубе $20\times20\times20$, центр которого распологается в начале координат.

Вы можете вращать систему координат перемещая указатель мышки с нажатой левой клавишей. Вы можете сдвигать начало системы координат перемещая указатель мышки с нажатой правой клавишей.

Можно, дополнительно, указывать координаты источника света, цвет и сетку. По умолчанию принимается сетка из 50 точек на кажом ребре параллелепипеда.

Цвет в формате RGB (красный, зеленый, голубой) задается числом

$R * 256 * 256 + G * 256 + B$,

где каждая буква обозначает неотрицательное целое число не превосходящее 255. Например, $255*256*256$ – красный цвет, а $255*256*256+255*256$ – желтый (красный+зеленый). Допускаются, кроме того, следующие наборы аргументов:

$(f, xMin, xMax, yMin, yMax, zMin, zMax, gridSize)$,

$(f, xMin, xMax, yMin, yMax, zMin, zMax, lightX, lightY, lightZ, gridSize)$,

$(f, xMin, xMax, yMin, yMax, zMin, zMax, lightX, lightY, lightZ, color, gridS$

Для построения графика функции во времени с изменением параметров необходимо указать сначала количество кадров. Затем следует виставить конечное значение параметров используя ползунки. Для построения графика необходимо нажать кнопку - 'Построить'. Значение количества кадров и параметры на ползунках можна указать через **set3D**().

Example.

```
SPACE = R64[x, y, z];
f = -x^2+2y^2+3z^2-25;
\implicitPlot3d( f, -10, 10, -10, 10, -10, 10);
```

Гиперболоид

```
SPACE = R64[x, y, z ];
\implicitPlot3d( x^2+ y^2+ z^2-25 , -7, 7, -7, 7, -7, 7,10, 10, 10, 2
```

Красная сфера

```
SPACE = R64[x, y, z];
f =  \sin(xyz/100) ;
\implicitPlot3d( f , -9,9,-9,9,-9,9, 10,10,4, 255*256*256+255*256, 50
```

Желтая поверхность с центральной симметрией.

```
SPACE = R64[x, y, z];
f = ((x+2)^2+ (y-2)^2 -1)((x-2)^2+ (y+2)^2 -1)((x+2)^2+ (y+2)^2 -1) (
\implicitPlot3d( f, -10, 10, -10, 10, -10, 10 );
```

Органные трубы.

3.2.6. Построение разных 3D графиков функций в одной системе координат

Для построения графиков функций, заданных разными способами, необходимо выполнить команду **showPlots3D**(f, g), где f и g - команды для постройки других графиков функции

Параметры окружения можно задать с помощью **set3D**().

Вы можете вращать систему координат перемещая указатель мышки с нажатой левой клавишей. Также вы можете сдвигать начало системы координат перемещая указатель мышки с нажатой правой клавишей.

Для построения разных графиков функции в одной системе координат во времени с изменением параметров необходимо указать сначала количество кадров. Затем следует выставить конечное значение параметров используя ползунки. Для построения графика необходимо нажать кнопку - 'Построить'. Значение количества кадров и параметры на ползунках можно указать через **set3D**().

```
SPACE = R64[x, y, z];
\set3D(-5,5,-5,5,-10,10,40);
f = -x^2+2y^2+3z^2-25;
g = (x^2+y^2)/20;
\showPlots3D(\implicitPlot3d(f), \explicitPlot3d(g));
```

3.3. Геометрия

paintElement'$operator1; operator2; ...operatork;$', - это инструмент для построения рисунков для школьной геометрии.

Операторы задаются так:
operator(arg$_1$: $type_1, ..arg_n : type_n = default$) : $returnType$
$arg_1, ..arg_n$ − .
$type_1, ..type_n$ − , , .
$= default$ − .
$returnType$ − .

У операторов могут быть дополнительный аргумент label: string, который определяет подпись фигуры, и требование отобразить на рисунке:
• label не задан - фигура используется как промежуточная и НЕ рисуется.
• label = %% - фигура рисуется, но без подписи.
• label = %text% - фигура подписывается текстом.

Подпись фигуры: operator (arg$_1, ..arg_n$).$display$(%$text$%);

Строки отделяются знаками процента, а не кавычками, как в языке mfthpar.

3.3.1. Пример (построить окружность)

Circle(radius: r1, center: Point = Point(x1, y1)).
Можно применять разные способы построения окружности:
• Circle(r1, Point(x1, y1));
• Circle(r1); - центр автоматически будет в x1=0, y1=0.
• R = r1; P = Point(x1, y1).display(“% O1 %); C = Circle(R, P);

3.3.2. Операторы

Point(x: number1, y: number2) - конструктор точки.
• x, y - позиция точки.

```
\paintElement('a = Point(1, 1).display(%A%);');
```

Line(point1: Point, point2: Point) - конструктор линии.
• point1, point2 - точки линии. Порядок не имеет значения.

```
\paintElement('l = Line(Point(1, 1), Point(2, 2)).display(%L%);');
```

Polygon(point1, point2, ..., pointN: Point[]) - конструктор полигона, фигуры являющейся списком точек соединенных отрезками. Последняя точка соединяется с первой.
• point1, point2, ..., pointN – Вершины полигона. Длинна списка вершин не ограничена.

```
\paintElement(pl = Polygon(Point(1, 1), Point(3, 3), Point(0, 2)).dis
```

Rectangle(width: number1, height: number2, bottomLeft: Point = Point(1, 1)) - конструктор горизонтального прямоугольника.
• width - ширина прямоугольника.
• height - высота прямоугольника.
• bottomLeft - нижняя левая точка прямоугольника.

```
\paintElement(pl = Rectangle(1, 2,Point(4,0)).display();');
```

Square(size: number, bottomLeft: Point = Point(x1, y1)) - конструктор квадрата.
• size - длинна стороны квадрата.
• bottomLeft - нижняя левая точка квадрата.

```
\paintElement(pl = Square(2,Point(4,0)).display();');
```

Triangle(point1: Point, point2: Point, point3: Point) - конструктор треугольника по трем точкам. Порядок не имеет значения.
• point1, point2, point3 - точки треугольника.

```
\paintElement(pl = Triangle(Point(1, 1), Point(3, 3), Point(0, 2)).di
```

Circle(radius: number1, center: Point = Point(x1, y1)) - конструктор круга.
• radius - радиус круга.
• center - центр круга.

```
\paintElement('c = Circle(3).display();');
```

Ellipse(width: number1, height: number2, center: Point = Point(x1, y1)) - конструктор элипса.
• width – горизонтальная полуось эллипса.
• height - вертикальная полуось эллипса.
• center - центр эллипса.

```
\paintElement('a = Ellipse(2, 3).display();');
```

normal(point: Point1, line: Line1):Point - построить перпендикуляр из точки на прямую. Возвращает точку пересечения перпендикуляра и прямой.
• point - точка из которой восстанавливается перпендикуляр.
• line – прямая к которой восстанавливается перпендикуляр.

```
\paintElement('a = Point(1, 1).display(%A%); l = Line(Point(1,3),
  Point(3,1)).display(%L%); n = normal(a, l).display(%N%);');
```

median(point: Point, line: Line): Point - построить медиану из точки на отрезок. Возвращает середину отрезка.
• point - точка из которой построить медиану.
• line – отрезок, на который опирается медиана.

```
\paintElement('a = Point(1, 1).display(%A%); l = Line(Point(1,3), Poi
```

Text(text: %string1%, leftBottom: Point1, fontSize: number = 10) - написать текст, начиная его в определенной точке. • text - собственно текст.
• leftBottom - нижняя левая (начальная) точка текста.
• fontSize - размер шрифта текста.

```
\paintElement('a = Text(%This is text%, Point(1, 1)).display();');
```

middle(line: Line): Point - найти точку середины отрезка.
• line - отрезок, середину которого нужно найти.

```
\paintElement(l = Line(Point(1,3), Point(3,1)).display(%L%); m = midd
```

incircle(triangle: Triangle): Circle - построить окружность вписанную в труегольник.
• triangle - треугольник в который надо вписать окружность

```
\paintElement(tr = Triangle(Point(1,3), Point(3,1), Point(1,1)).displ
```

Circumcircle(triangle: Triangle): Circle - построить окружность описанную вокруг треугольника. • triangle - треугольник вокруг которого надо описать окружность

```
\paintElement(tr = Triangle(Point(1,3), Point(3,1), Point(1,1)).displ
```

lineCircleCross(line: Line, circle: Circle): Point[] - найти и вернуть точки пересечения линии и окружности.
• line - линия которая может пересекать окружность.
• circle – окружность, которая может пересекать линию.

```
\paintElement('l = Line(Point(1,3), Point(3,1)).display(); c = Circle
```

circlesCross(circle1: Circle, circle2: Circle): Point[] - найти и вернуть точки пересечения двух окружностей.
• circle1, circle2 – окружности, которые могут перескаться.

```
\paintElement('c1 = Circle(2).display();
c2 = Circle(3, Point(2,2)).display(); p = circleCross(c1, c2);');
```

Конец раздела о графиках.

3.4. Контрольные задания

В Mathpar постройте графики функций

- $f(x) = x^2 + 2y$,
- $f(x) = \sqrt{\sin^2(5x - 1) + \exp x}$,
- $f(x, y) = \sin(\cos(x + \tan(y)))$.

Глава 4

Выбор окружения для математических объектов

4.1. Окружение

Прежде чем будет задан любой математический объект, число, функция или символ, должно быть ясно определено «окружение» — пространство, в котором будут определяться объекты. В этой главе описываются способы задания окружения. Перемещение из некоторого окружения в текущее, как правило, должно выполнятся явно, с помощью функции **toNewRing**. В некоторых случаях такое преобразование к текущему окружению происходит автоматически.

Для выбора окружения задается *алгебраическое пространство переменных*. Оно определяется именами переменных и числовыми пространствами, в которых эти переменные принимают значения. Порядок переменных в списке переменных задает линейный порядок на этих переменных. Слева направо располагаются переменные, упорядоченные по старшинству от младших к старшим.

По умолчанию определено пространство R64$[x, y, z, t]$ четырех переменных, самая младшая — x, самая старшая — t.

В любой момент пользователь может сменить окружение, задав новое алгебраическое пространство переменных с помощью команды установки «SPACE=». Например, для задач вычислительной математики может быть достаточно пространства типа R64$[x]$ или Q$[x]$. Команда установки: «SPACE=R64[x];» или «SPACE=Q[x];»,

соответственно.

Если имя переменной начинается с символа \ и заглавной буквы (верхний регистр), то такая переменная обозначает элемент алгебры, у которой *операция умножения некоммутативная*, для всех остальных переменных *операция умножения коммутативная*.

4.2. Числовые множества

Определены следующие числовые множества:

Z — множество целых чисел Z,

Zp — конечное поле из p=MOD элементов Z/pZ, MOD — постоянная,

Zp32 — конечное поле из p=MOD32 элементов Z/pZ, MOD32 меньше 2^{31},

Z64 — кольцо целых чисел z таких, что $-2^{63} \leqslant z < 2^{63}$,

Q — множество рациональных чисел,

R — множество чисел с плавающей точкой для хранения приближенных действительных чисел с произвольной мантиссой,

R64 — множество чисел с плавающей точкой для хранения приближенных действительных чисел с двойной точностью (со стандартной 52-разрядной мантиссой и отдельным 11-разрядным полем для хранения порядка),

R128 — стандартные 64-битные числа с плавающей точкой для хранения приближенных действительных чисел со стандартной 52-разрядной мантиссой и отдельным 64-разрядным полем для хранения порядка,

C — комплексный класс, образованный из класса R,

C64 — комплексный класс, образованный из класса R64,

C128 — комплексный класс, образованный из класса R128,

CZ — комплексный класс, образованный из класса Z,

CZp — комплексный класс, образованный из класса Zp,

CZp32 — комплексный класс, образованный из класса Zp32,

CZ64 — комплексный класс, образованный из класса Z64,

CQ — комплексный класс, образованный из класса Q.

Примеры простых полиномиальных колец:

SPACE = Z [x, y, z];

SPACE = R64 [u, v];

SPACE = C [x].

4.3. Определение нескольких числовых множеств

Разрешается устанавливать алгебраические пространства из нескольких числовых множеств, например, пространство «C[z]R[x, y]Z[n, m]» позволяет работать с пятью именами переменных, определенных в множествах C, R и Z, соответственно. Первое множество считается основным и к нему будут приводится, при необходимости, все остальные переменные. В данном случае это C.

Его можно рассматривать как кольцо полиномов пяти переменных над C, при этом оно обладает дополнительными свойствами. Если полином не содержит переменной z, то это полином с коэффициентами из R. Если полином не содержит переменных z, x, y, то это полином с коэффициентами из Z.

Примеры:

SPACE=Z[x, y]Z[u];

SPACE=R64[u, v]Z[a, b];

SPACE=C[x]R[y, z];

Кольцо «Z[x, y, z]Z[u, v, w]», в котором шесть переменных разделены на две группы, можно использовать для задач в которых строятся полиномы, у которых коэффициенты являются полиномами или функциями других переменных. Например, характеристический полином для матрицы над кольцом Z$[x,\ y,\ z]$ будет получен как полином с неизвестной u, коэффициенты которого лежат в кольце Z$[x,\ y,\ z]$.

4.4. Идемпотентные алгебры. Тропическая математика.

Кроме классических числовых алгебр с операциями «+, -, *» и операцией «/» для полей, будут доступны пользователю и идемпотентные алгебры. Для числового множества R64, можно будет использовать алгебры $R64MaxPlus$, $R64MinPlus$, $R64MaxMin$, $R64MinMax$, $R64MaxMult$, $R64MinMult$. Для числового множества R, можно будет использовать алгебры $RMaxPlus$, $RMinPlus$, $RMaxMin$, $R64MinMax$, $RMaxMult$, $RMinMult$. Для числового множества Z, можно будет использовать алгебры $ZMaxPlus$,

ZMinPlus, *ZMaxMin*, *ZMinMax*, *ZMaxMult*, *ZMinMult*.

Пример.

```
SPACE=ZMaxPlus[x, y];
a=2; b=9; c=a+b; d=a*b; \print(c, d)
```

Результат выполнения:
$c = 9$;
$d = 11$.

4.5. Константы

Можно установить или заменить следующие постоянные.

FLOATPOS — число десятичных знаков после запятой, которые выводятся на печать. По умолчанию принимается значение 2.

MachineEpsilonR — машинное эпсилон для чисел типа R. По умолчанию принимается значение 10^{-29}. Число, модуль которого меньше 10^{-29}, считаестя машинным нулем. Для установки нового значения 10^{-30} нужно ввести команду «MachineEpsilonR64=30».

MachineEpsilonR64 — машинное эпсилон для чисел типа R64. По умолчанию принимается значение 2^{-36}. Число, модуль которого меньше 2^{-36}, считается машинным нулем. Отметим, что числа R64 имеет 52 разряда в мантиссе, Для установки нового значения 2^{-48} нужно ввести команду «MachineEpsilonR64=48».

Постоянная MachineEpsilonR (и MachineEpsilonR64) используется при факторизации полиномов с коеффициентами типа R (или R64). Каждый коэффициент такого полинома будет предварительно делиться на число *MachineEpsilonR* (или *MachineEpsilonR*64) и округляется до целого значения.

ACCURACY определяет число точных десятичных позиций после запятой для чисел типа R и C в операциях умножения и деления. По умолчанию ACCURACY имеет значение $MachineEpsilonR*10^{-5}$. Если n<m, то команда «MachineEpsilonR=n/m» установит одновременно MachineEpsilonR=10^{-n} и ACCURACY=10^{-m}.

MOD32 — модуль для простого поля, не превосходящий 2^{31} (по умолчанию принимается значение 268435399). Простое число MOD32 – характеристика конечного поля. Константа MOD32 используется в том случае, когда вычисления происходят в конечном поле Zp32 и она должна быть меньше числа 2^{31}.

MOD — модуль типа Z для простого поля (по умолчанию принимается значение 268435399). Простое число MOD — это характеристика конечного поля, но в отличие от MOD32 у него нет ограничения на абсолютное значение. Константа MOD используется в том случае, когда вычисления происходят в конечном поле Zp.

RADIAN может принимать значения 1 или 0. Если RADIAN = 1, то углы измеряются в радианах, иначе — в градусах. По умолчанию RADIAN=1.

STEPBYSTEP может принимать значения 1 или 0. Если STEPBYSTEP = 1, то будут выводиться промежуточные результаты вычислений. По умолчанию STEPBYSTEP = 0.

EXPAND может принимать значения 1 или 0. Если EXPAND = 1, то во входном выражении будут раскрываться все скобки. По умолчанию EXPAND = 1.

SUBSTITUTION может принимать значения 1 или 0. Если SUBSTITUTION = 1, то во входном выражении будут подставляться вместо имен выражений их значения, если они были определены раньше. По умолчанию SUBSTITUTION = 1.

Пример.

```
SPACE=Zp32[x, y];
MOD32=7;
f1=37x+42y+55;
f2=2f1;
\print(f1, f2 );
```

Результат выполнения:
$f1 = 2x - 1$;
$f2 = 4x + 5$.

4.6. Контрольные задания

В Mathpar вычислите

- значение функции $f(x) = \sqrt{\sin^2(5x-1) + e^x}$ в R при $x = 7$,
- значение функции $f(x) = x^3 + 10x$ в Z/(11)Z при $x = 7$.

Глава 5

Функции одной и нескольких переменных

5.1. Математические функции

Приняты следующие обозначения для элементарных функций и констант.

5.1.1. Константы

\i — мнимая единица,
\e — основание натурального логарифма,
\pi — число π, т. е. отношение длины окружности к диаметру,
\infty — знак бесконечности.

5.1.2. Функции одного аргумента

\ln — натуральный логарифм,
\lg — десятичный логарифм,
\sin — синус,
\cos — косинус,
\tg — тангенс,
\ctg —котангенс,
\arcsin — арксинус,
\arccos — арккосинус,

\arctg — арктангенс,
\arcctg — арккотангенс,
\sh — синус гиперболический,
\ch — косинус гиперболический,
\th — тангенс гиперболический,
\cth — котангенс гиперболический,
\arcsh — арксинус гиперболический,
\arcch — арккосинус гиперболический,
\arcth — арктангенс гиперболический,
\arccth — арккотангенс гиперболический,
\exp — экспонента,
\sqrt — корень квадратный,
\abs — абсолютное значение для действительных чисел, модуль для комплексного числа;
\sign — знак числа. Возвращает 1, 0, -1, когда число положительное, ноль или отрицательное соответственно;
\unitStep$(x, a)$ — это функция, которая при $x \geqslant 0$ принимает значение 1, а при $x < 0$ принимает значение 0;
\fact — факториал. Определен для целых положительных чисел. Можно писать в привычном виде, например, «5!».

5.1.3. Функции двух аргументов

^ — степень;
\log — логарифм от функции по указанному основанию;
\rootOf(x, n) — корень степени n из x;
\Gamma — функция Гамма;
\Gamma2 — функция Гамма 2;
\binomial — число сочетаний.

Примеры.

```
SPACE = R64[x, y];
f1 = \sin(x);
f2 = \sin(\cos(x + \tg(y)));
f3 = \sin(x^2) + y;
\print(f1, f2, f3);
```

Результат выполнения:
$f1 = sin(x)$;

$f2 = sin(cos(x + tg(y)))$;
$f3 = sin(x^2) + y$.

5.2. Вычисление значений функции в точке

Для вычисления значения функции в точке необходимо выполнить команду $\backslash$**value**$(f, [var1, var2, \ldots, varn])$, где f — функция, а $var1$, $var2, \ldots,$ $varn$ — значения соответствующих переменных.

Для тригонометрических функций мерой угла считается радиан или градус. Указание меры угла определяется константой RADIAN. Если не указывать угловую меру, то угловой мерой выбирается радиан. Чтобы поменять угловую меру с радиан на градусы, нужно выполнить команду «RADIAN=0;». Если же нужно поменять угловую меру с градусов на радианы, то нужно выполнить команду «RADIAN=1;».

Если аргументами являются целые числа $15k$ и $18k$ градусов или $\pi k/12$ и $\pi k/10$ радиан ($k \in$ Z), то значениями тригонометрических функций являются алгебраические числа.

Примеры.

```
SPACE = R[x, y];
f = \sin(x^2 + \tg(y^3 + x));
g = \value(f, [1, 2]);
\print(g);
```

Результат выполнения:
in: $SPACE = R[x, y]$;
$f = sin(x^2 + tg(y^3 + x))$;
$g = value(f,\ [1,\ 2])$; $print(g)$;
out: $g = 0.52$;

```
SPACE = Z[x];
RADIAN = 0;
f = \sin(x);
g = \value(f, 15);
\print(g);
```

Результат выполнения:
in: $SPACE = Z[x]$;

$RADIAN = 0;$
$f = sin(x);$
$g = value(f, 15);$
$print(g);$
out: $g = (\sqrt{6} - (\sqrt{2}))/(4);$

```
SPACE = Z[x];
RADIAN = 0;
f = \sin(x);
g = \value(f, 225);
\print(g);
```

Результат выполнения:
$g = (-1 \cdot \sqrt{2})/(2);$

```
SPACE = Z[x];
RADIAN = 0;
f = \cos(x);
g = \value(f, 54);
\print(g);
```

Результат выполнения:
$g = \sqrt{(5 - \sqrt{5})/(8)};$

```
SPACE = Z[x];
RADIAN = 0;
f = \tg(x);
g = \value(f, 126);
\print(g);
```

Результат выполнения:
$g = (-1 \cdot \sqrt{(1 + 2 \cdot \sqrt{5}/(5))});$

```
SPACE = Z[x];
RADIAN = 0;
f = \sin(x);
g = \value(f, 216);
\print(g);
```

Результат выполнения:
$g = (-1 \cdot \sqrt{(5 - \sqrt{5})/(8)});$

```
SPACE = Z[x];
RADIAN = 0;
f = \cos(x);
g = \value(f, 108);
\print(g);
```

Результат выполнения:
$g = (1 - \sqrt{5})/(4)$.

5.3. Подстановка выражений в функции

Для вычисления композиции функций нужно «подставлять» в функцию вместо ее аргументов другие функции. Для этого необходимо выполнить команду $\backslash\mathbf{value}(f, [func1, func2, \ldots, funcn])$, где f — данная функция, $func1, func2, \ldots, funcn$ — функции, которые подставляются вместо соответствующих переменных.

Пример.

```
SPACE = Z[x, y];
f = x + y;
g = f^2;
r = \value(g, [x^2, y^2]);
\print(r);
```

Результат выполнения:
in: $g = y^2 + 2yx + x^2$;
$f = y + x$;
out: $r = y^4 + 2y^2x^2 + x^4$.

5.4. Вычисление предела функции в точке

Для вычисления предела функции в точке необходимо выполнить команду $\backslash\mathbf{lim}(f, var)$, где f — это функция, а var — точка, возможно бесконечная, в которой требуется найти предел, конечный или бесконечный.

Примеры.

```
SPACE = R64[x];
f = \sin(x) / x;
g = \lim(f, 0);
\print(g);
```

Результат выполнения:

in: $SPACE = R64[x]$;
$f = sin(x)/x$;
$g = lim(f, 0)$;
$print(g)$;
out: $g = 1.00$;

```
SPACE = R64[x];
f = (x^2 - 2x + 2) / (x^2 + x - 2);
g = \lim(f, 1);
\print(g);
```

Результат выполнения:

in: $SPACE = R64[x]$;
$f = (x^2 - 2x + 2)/(x^2 + x - 2)$;
$g = lim(f, 1)$;
$print(g)$;
out: $g = \infty$;

```
SPACE = R64[x];
f = \sin(x + 3) / (x^2 + 6x + 9);
g = \lim(f, -3);
\print(g);
```

Результат выполнения:

in: $SPACE = R64[x]$;
$f = \sin(x + 3)/(x^2 + 6x + 9)$;
$g = lim(f, -3)$;
$print(g)$;
out: $g = \infty$;

```
SPACE = R64[x];
f = (1 + 1 / x)^x;
g=\lim(f, \infty);
\print(g);
```

Результат выполнения:

in: $SPACE = R64[x]$;

$f = (1 + 1/x)^x$;
$g = lim(f, \infty)$;
$print(g)$;
out: $g = 2.72$.

5.5. Дифференцирование функций

Для вычисления производной функции f по переменной y из кольца $Z[x, y, z]$ необходимо выполнить команду $\backslash\mathbf{D}(f, y)$. Вычисление третьей производной по y можно выполнить $\backslash\mathbf{D}(f, [y\hat{}3])$. Если необходимо найти производную функции f один раз по первой переменной из текущего кольца (в данном случае x) можно записать $\backslash\mathbf{D}(f)$ или $\backslash\mathbf{D}(f, x)$.

Для нахождения смешанной производной первого порядка от функции f существует команда $\backslash\mathbf{D}(f, [x, y])$, для нахождения производной высших порядков нужно использовать команду $\backslash\mathbf{D}(f, [x\hat{}k, z\hat{}m, y\hat{}n])$, где k, m, n указывают, какого порядка по соответствующей переменной вычисляется производная.

Примеры.

```
SPACE=Z[x, y];
f = \sin(x^2 + \tg(y^3 + x));
h= \D(f, y);
\print(h);
```

Результат выполнения:
Результат выполнения:
in: $SPACE = Z[x, y]$;
$f = sin(x^2 + tg(y^3 + x))$;
$h = D(f, y)$;
$print(h)$;
out: $h = 3y^2cos(x^2 + tg(y^3 + x))/(cos(y^3 + x))^2$;

```
SPACE = Z[x, y];
f = \sin(x^2 + \tg(y^3 + x));
h = \D(f);
\print(h);
```

Результат выполнения:
in: $SPACE = Z[x, y]$;
$f = sin(x^2 + tg(y^3 + x))$;

$h = D(f)$;
$print(h)$;
out: $h = (2x\cos(x^2 + \mathrm{tg}(y^3 + x))(\cos(y^3 + x))^2 + \cos(x^2 + \mathrm{tg}(y^3 + x)))/(\cos(y^3 + x))^2$;

```
SPACE = Z[x, y, z];
f = x^8y^4z^9;
g = \D(f, [x^2, y^2, z^2]);
\print(g);
```

Результат выполнения:
in: $SPACE = Z[x, y, z]$;
$f = x^8y^4z^9$;
$g = D(f, [x^2, y^2, z^2])$;
$print(g)$;
out: $g = 48384z^7y^2x^6$.

5.6. Интегрирование композиций элементарных функций

Символьное интегрирование композиций элементарных функций выполняется командой $\backslash\mathbf{int}(f(x))dx$.

Примеры.

```
SPACE = Z[x, y, z];
l1 = \int(x^6yz + 3x^2y - 2z) d x;
dl1 = \D(l1,x);
l2 = \int(x^6yz + 3x^2y - 2z) d y;
dl2 = \D(l2,y);
l3 = \int(x^6yz + 3x^2y - 2z) d z;
dl3 = \D(l3,z);
\print(l1, dl1,l2, dl2,l3, dl3);
```

Результат выполнения:
in: $SPACE = Z[x, y, z]$;
$l1 = \int(x^6yz + 3x^2y - 2z)dx$;
$dl1 = D(l1, x)$;
$l2 = \int(x^6yz + 3x^2y - 2z)dy$;
$dl2 = D(l2, y)$;

$l3 = \int(x^6yz + 3x^2y - 2z)dz;$
$dl3 = D(l3, z);$
$print(l1, dl1, l2, dl2, l3, dl3);$
out: $l1 = (1/7)zyx^7 - 2zx + yx^3;$
$dl1 = x^6yz + 3x^2y - 2z.$ $l2 = (1/2)zy^2x^6 - 2zy + (3/2)y^2x^2;$
$dl2 = x^6yz + 3x^2y - 2z.$ $l3 = (1/2)z^2yx^6 - z^2 + 3zyx^2;$
$dl3 = x^6yz + 3x^2y - 2z.$

```
SPACE = R[x];
l = \int(1/(x^2-5x+6)) d x;
dl = \D(l,x);
\print(l, dl);
```

Результат выполнения:
in: $SPACE = Q[x, y, z];$
$l = \int(1/(x^2 - 5x + 6))dx;$
$dl = D(l, x);$
$print(l, dl);$
out: $l = (\ln(x - 3) - \ln(x - 2));$
$dl = (1/(x - 3) - 1/(x - 2)).$

```
SPACE = Q[x];
l = \int(\exp(x)+\exp(-x)) d x;
dl = \D(l,x);
\print(l, dl);
```

Результат выполнения:
in: $SPACE = Q[x, y, z];$
$l = \int(\exp(x) + \exp(-x))dx;$
$dl = D(l, x);$
$print(l, dl);$
out: $l = (\exp(x) - ((\exp(x))^{-1}));$
$dl = (\exp(x) + \exp(-x)).$

```
SPACE = Q[x];
l = \int(x*\exp(x^2)) d x;
dl = \D(l,x);
\print(l, dl);
```

Результат выполнения:
in: $SPACE = Q[x, y, z];$
$l = \int(x * \exp(x^2))dx;$

$dl = D(l, x)$;
$print(l, dl)$;
out: $l = (\exp(x^2)/2)$;
$dl = (x * \exp(x^2))$.

```
SPACE = Q[x];
l = \int((x*\ln(x)*\exp(x)+\exp(x))/x) d x;
dl = \D(l,x);
\print(l, dl);
```

Результат выполнения:
in: $SPACE = Q[x, y, z]$;
$l = \int((x * \ln(x) * \exp(x) + \exp(x))/x)dx$;
$dl = D(l, x)$;
$print(l, dl)$;
out: $l = (\ln(x) * \exp(x))$;
$dl = ((x * \ln(x) * \exp(x) + \exp(x))/x)$.

```
SPACE = R64[x];
l = \int((\ln(x+3)+\ln(x+2)+\ln(x+1))) d x;
dl = \D(l,x);
\print(l, dl);
```

Результат выполнения:
in: $SPACE = R64[x, y, z]$;
$l = \int((\ln(x + 3) + \ln(x + 2) + \ln(x + 1)))dx$;
$dl = D(l, x)$;
$print(l, dl)$;
out: $l = (((x * \ln(x + 3) + 3.00 * \ln(x + 3) + x * \ln(x + 2) + 2.00 * \ln(x + 2) + x * \ln(x + 1) + \ln(x + 1)) - 3x))$;
$dl = ((\ln(x + 3) + \ln(x + 2) + \ln(x + 1)))$.

```
SPACE = Q[x];
l = \int((2x^2+1)^3) d x;
dl = \D(l,x);
m=\factor(dl);
\print(l, m);
```

Результат выполнения:
in: $SPACE = Q[x, y, z]$;
$l = \int((2x^2 + 1)^3)dx$;
$dl = D(l, x)$;

$m = factor(dl)$;
$print(l, m)$;
out: $l = (8/7)x^7 + (12/5)x^5 + 2x^3 + x$;
$m = (2x^2 + 1)^3$.

5.7. Упрощение композиции функций

Для разложения любой тригонометрической или логарифмической функции с помощью тождеств:
$sin(x)cos(y) \pm cos(x)sin(y) = sin(x \pm y)$,
$cos(x)cos(y) \pm sin(x)sin(y) = cos(x \mp y)$,
$sin^2(x) + cos^2(x) = 1$,
$cos^2(x) - sin^2(x) = cos(2x)$,
$ln(a) + ln(b) = ln(ab)$,
$ln(a) - ln(b) = ln(\frac{a}{b})$,
используется команда **Expand**$(f(x))$

Примеры.

```
SPACE=Q[x, y, z];
g=\ln(x^2*4x);
f=\Expand(g);
\print(f);
```

Результат выполнения:
in: $SPACE = Q[x, y, z]$;
$g = \ln(x^2 * 4x)$;
$f = Expand(g)$;
$print(f)$;
out: $f = \ln(x^2) + \ln(4x)$;

```
SPACE=Q[x, y, z];
g=\sin(x^2+4x+2\pi);
f=\Expand(g);
\print(f);
```

Результат выполнения:
in: $SPACE = Q[x, y, z]$;
$g = \sin(x^2 + 4x + 2\pi)$;
$f = Expand(g)$;
$print(f)$;

out: $f = (\sin(x^2)(\cos(4x)\cos(2) - \sin(4x)\sin(2)) + \cos(x^2)(\sin(4x)\cos(2) + \cos(4x)\sin(2)))$;

```
SPACE=Q[x, y, z];
g=\cos(\sin(x)+\cos(y));
f=\Expand(g);
\print(f);
```

Результат выполнения:
in: $SPACE = Q[x, y, z]$;
$g = \cos(\sin(x) + \cos(y))$;
$f = Expand(g)$;
$print(f)$;
out: $f = (\cos(\cos(y))\cos(\sin(x)) - \sin(\cos(y))\sin(\sin(x)))$;

Для разложения на множители выражений при помощи описанных выше тригонометрических и логарифмических тождеств, а также следующих тождеств:
$ln(a)^k = k \cdot ln(a)$,
$e^{iz} + e^{-iz} = 2\cos(z)$,
$e^{iz} - e^{-iz} = 2i\sin(z)$,
$\ln(1 + iz) - \ln(1 - iz) = 2i \cdot arctg(z)$,
$\ln(1 - iz) - \ln(1 + iz) = 2i \cdot arcctg(z)$,
$e^z + e^{-z} = 2ch(z)$
$e^z - e^{-z} = 2i \cdot sh(z)$,
используется команда **Factor**$(f(x))$.

Примеры.

```
SPACE=Q[x, y, z];
g=\log_{2}(x)+\log_{2}(y)-\log_{2}(xz)+\lg(y)+\lg(y)-\lg(z);
f=\Factor(g);
\print(f);
```

Результат выполнения:
in: $SPACE = Q[x, y, z]$;
$g = \log_2(x) + \log_2(y) - \log_2(xz) + \lg(y) + \lg(y) - \lg(z)$;
$f = Factor(g)$;
$print(f)$;
out: $f = \lg(y^2/z) + \log_2(y/z)$;

```
SPACE=Q[x, y, z];
```

```
g=16\sin(x/48)\cos(x/48)\cos(x/24)\cos(x/12)\cos(x/6);
f=\Factor(g);
\print(f);
```

Результат выполнения:
in: $SPACE = Q[x, y, z]$;
$g = 16\sin(\frac{x}{48})\cos(\frac{x}{48})\cos(\frac{x}{24})\cos(\frac{x}{12})\cos(\frac{x}{6})$;
$f = Factor(g)$;
$print(f)$;
out: $f = \sin(0.33x)$;

```
SPACE=C64[x, y, z];
g=\ln(1-\ix) - \ln(1+\ix) + \e^(\ix) - 2\e^(-\ix) + \sin(x)^2 - \cos(
f=\Factor(g);
\print(f);
```

Результат выполнения:
in: $SPACE = C64[x, y, z]$;
$g = \ln(1-ix) - \ln(1+ix) + \exp(ix) - 2\exp(-ix) + \sin(x)^2 - \cos(x)^2$;
$f = Factor(g)$;
$print(f)$;
out: $f = (-1.00\cos(2x)) + 2.00i(\sin(x)) + (-1.00\exp(-ix)) + (2.00i(\mathrm{arcctg}(x)))$;

Комбинируя команды **Factor**$(f(x))$ и **Expand**$(f(x))$, можно упрощать более сложные выражения.

```
SPACE=R64[x, y, z];
g=(\sin(x+y) + \sin(x-y))\cos(x) + (\sin(x+y) + \sin(x-y))\sin(y);
f=\Expand(g);
u=\Factor(f);
\print(f,u);
```

Результат выполнения:
in: $SPACE = R64[x, y, z]$;
$g = (\sin(x+y) + \sin(x-y))\cos(x) + (\sin(x+y) + \sin(x-y))\sin(y)$;
$f = Expand(g)$;
$u = Factor(g)$;
$print(f, u)$;
out: $f = 2.00\cos(y)\sin(x)\cos(x) + 2.00\sin(y)\cos(y)\sin(x)$;
$u = \sin(x)\sin(2.00y) + \cos(y)\sin(2.00x)$;

5.8. Arithmetic-geometric mean

Given two non-negative numbers x and y, one can define their arithmetic, geometric and harmonic means as $\frac{x+y}{2}$, $\sqrt{xy}$, and $\frac{2xy}{x+y}$, respectively. Moreover, $\backslash\mathbf{AGM}(x,y)$ denotes the arithmetic-geometric mean of x and y. $\backslash\mathbf{GHM}(x,y)$ denotes the geometric-harmonic mean of x and y. At last, $\backslash\mathbf{MAGM}(x,y)$ denotes the modified arithmetic-geometric mean of x and y. Every mean is a symmetric homogeneous function in their two variables x and y. In contrast to well-known means, $\backslash\mathbf{AGM}(x,y)$, $\backslash\mathbf{GHM}(x,y)$, and $\backslash\mathbf{MAGM}(x,y)$ are calculated iteratively.

The arithmetic-geometric mean $\backslash\mathbf{AGM}(x,y)$ is equal to the limit of both sequences x_n and y_n, where $x_0 = x$, $y_0 = y$, $x_{n+1} = \frac{1}{2}(x_n + y_n)$, and $y_{n+1} = \sqrt{x_n y_n}$.

In the same way, the geometric-harmonic mean $\backslash\mathbf{GHM}(x,y)$ is equal to the limit of both sequences x_n and y_n, where $x_0 = x$, $y_0 = y$, $x_{n+1} = \sqrt{x_n y_n}$, and $y_{n+1} = \frac{2x_n y_n}{x_n+y_n}$.

The modified arithmetic-geometric mean $\backslash\mathbf{MAGM}(x,y)$ is equal to the limit of the sequence x_n, where $x_0 = x$, $y_0 = y$, $z_0 = 0$, $x_{n+1} = \frac{x_n+y_n}{2}$, $y_{n+1} = z_n + \sqrt{(x_n - z_n)(y_n - z_n)}$, and $z_{n+1} = z_n - \sqrt{(x_n - z_n)(y_n - z_n)}$.

Examples.

```
SPACE=R64[];
FLOATPOS=5;
agm=\AGM(1,5);
ghm=\GHM(1,5);
p=agm*ghm;
magm=\MAGM(1,5);
\print(agm, ghm, p, magm);
```

Results:

$$agm = 2.60401$$

$$ghm = 1.92012$$

$$p = 5$$

$$magm = 2.61051$$

5.9. The complete elliptic integrals of the first and second kind

Let us use the parameter $0 \leq k \leq 1$.
The complete elliptic integral of the first kind $K(k)$ is defined as

$$K(k) = \int_0^1 \frac{dt}{\sqrt{(1-t^2)(1-k^2t^2)}}$$

It can be computed in terms of the arithmetic-geometric mean:

$$K(k) = \frac{\pi}{2AGM(1, \sqrt{1-k^2})}$$

On the other hand, for $k < 1$, it can be computed in terms of the geometric-harmonic mean:

$$K(k) = \frac{\pi}{2} GHM(1, \frac{1}{\sqrt{1-k^2}})$$

The complete elliptic integral of the second kind $E(k)$ is defined as

$$E(k) = \int_0^1 \sqrt{\frac{1-k^2t^2}{1-t^2}} dt$$

It can be computed in terms of the modified arithmetic-geometric mean:

$$E(k) = K(k) MAGM(1, 1-k^2)$$

See also: S. Adlaj (2012) An eloquent formula for the perimeter of an ellipse. Notices of the American Mathematical Society. 59(8), 1094-1099. DOI:10.1090/noti879

5.10. The period of a simple gravity pendulum

A point mass suspended from a pivot with a massless cord. The length of the pendulum equals $L = 1$ metre. It swings under gravitational acceleration $g = 9.80665$ metres per second squared. The

maximum angle that the pendulum swings away from vertical, called the amplitude, equals $\theta_0 = 2.0944$, that is, $\frac{2}{3}\pi$ radians.

Find the period T of the pendulum using the arithmetic-geometric mean

$$T = \frac{2\pi}{AGM(1, \cos(\theta_0/2))}\sqrt{\frac{L}{g}}$$

```
SPACE = R64[];
FLOATPOS = 4;
\theta_0=2.0944;
w=\cos(\theta_0/2);
Ts = 2*\pi*\sqrt{L/g}/\AGM(1,w);
 \print(Ts);
L = 1; g = 9.80665;
T=\value(Ts);   \print(T);
```

Results:
$Ts = 2.7458 * \pi * (L/g)^{(1/2)}$
$T = 2.7546$

5.11. Контрольные задания

В Mathpar

- подставьте в функцию $f(x) = \sqrt{\sin^2(5x-1) + e^x}$ вместо x выражение $x + y$, а вместо y — 5,
- найдите $\lim(x^3 + 10x)/x^2$ при $x \to 0$,
- найдите производную функции $f(x) = \sqrt{\sin^2(5x-1) + e^x}$,
- найдите $\int_0^2 (x^3 + 10x)dx$.

Глава 6

Ряды

Ряд задается в виде f=**sum** _{i=k}^{ \infty} $F(i, x, y, \ldots, z)$, где i — индекс суммирования, k — начальное значение i, $F(i, x, y, \ldots, z)$ — функция многих переменных, т.е. F может зависеть и от i.

Над рядами определены такие арифметические операции как сложение, вычитание и умножение.

Пусть f и g — ряды.

Для сложения двух рядов необходимо выполнить команду **seriesAdd**(f, g).

Для разности двух рядов необходимо выполнить команду **seriesSubtract**(f, g).

Для умножения двух рядов необходимо выполнить команду **seriesMultiply**(f, g).

Для разложения функции в ряд Тейлора с определенным количеством членов ряда необходимо выполнить команду **teilor**$(f, point, num)$, где f — функция, $point$ — точка, num — общее количество членов ряда.

Примеры.

```
SPACE=R[x, y];
f=\sum_{i=2}^{\infty} (2x^i y b i);
g=\sum_{i=4}^{\infty} (x^i a\sin(a i x));
h=\seriesAdd(f, g);
\print(f, g, h);
```

Результат выполнения:

$$f = \sum_{i=2}^{\infty} 2(x)^i ybi;$$

$$g = \sum_{i=4}^{\infty} (x)^i a \sin(aix);$$

$$h = \sum_{i=4}^{\infty} (2(x)^i ybi + (x)^i a \sin(aix)) + \sum_{i=2}^{3} (2x^i ybi);$$

```
SPACE=R[x, y];
f=\sum_{i=1}^{\infty} (x^i y i\cos(b));
g=\sum_{i=2}^{\infty} (5x^i a\cos(axi));
h=\seriesSubtract(f, g);
\print(f, g, h);
```

Результат выполнения:

$$f = \sum_{i=1}^{\infty} (x)^i yi \cos(b);$$

$$g = \sum_{i=2}^{\infty} 5(x)^i a \cos(axi);$$

$$h = \sum_{i=2}^{\infty} ((x)^i yi \cos(b) - 5(x)^i a \cos(axi)) + xy \cos(b);$$

```
SPACE=R[x, y];
f=\sum_{i=0}^{\infty} (2x^i y b i);
g=\sum_{i=2}^{\infty} (5y^i x^2 b i\cos(a_1 x));
h=\seriesSubtract(f, g);
\print(f, g, h);
```

Результат выполнения:

$$f = \sum_{i=0}^{\infty} 2(x)^i ybi;$$

$$g = \sum_{i=2}^{\infty} 5y^i x^2 bi \cos(a_1 x);$$

$$h = \sum_{i=2}^{\infty}(2x^i ybi - 5y^i x^2 bi\cos(a_1 * x)) + \sum_{i=0}^{1}(2x^i ybi);$$

```
SPACE = R[x, y]; \clean();
f = \sum_{a = 6}^{\infty} (x a a_0);
g = \sum_{a = 9}^{\infty} (56x^4 a \cos(a_1 x));
h = \seriesMultiply(f, g);
\print(f, g, h);
```

Результат выполнения:

$$f = \sum_{a=6}^{\infty} xaa_0;$$

$$g = \sum_{a=9}^{\infty} 56.00x^4 a\cos(a_1 x);$$

$$h = \sum_{a_2=6}^{\infty}\sum_{a=9}^{\infty} xa_2 a_0 \cdot 56.00x^4 a\cos(a_1 x);$$

```
SPACE=R[x, y];
f = \sum_{a = 6}^{\infty} (\sin(xa)\cos(g y) a_0);
g = \sum_{a = 9}^{\infty} (6x^2 a \sin(a x y^2));
h = \seriesMultiply(f, g);
\print(f, g, h);
```

Результат выполнения:

$$f = \sum_{a=6}^{\infty} \sin(xa)\cos(\sum_{a=9}^{\infty} 56.00x^4 a\cos(a_1 x)y)a_0;$$

$$g = \sum_{a=9}^{\infty} 6.00x^2 a\sin(axy^2);$$

$$h = \sum_{a_2=6}^{\infty}\sum_{a=9}^{\infty} \sin(xa_2)\cos(\sum_{a_2=9}^{\infty} 56.00x^4 a_2\cos(a_1 x)y)a_0 6.00x^2 a\sin(axy^2);$$

```
SPACE = R[x];
FLOATPOS = 15;
a = \teilor(\sin(x), 0, 7);
c = \value(a);
\print(a, c);
```

Результат выполнения:
$a = ((-x^7)/(7!) + x^5/(5!) + (-x^3)/(3!) + x/(1!))$;
$c = (-0.000198412698412x^7 + 0.008333333333333x^5 - 0.166666666666666x^3 + x)$.

6.1. Контрольные задания

В Mathpar

- разложите функцию $f(x) = \sin^2(5x - 1)$ в ряд Тейлора,
- создайте два случайных ряда с полиномиальными членами, используя генерацию полиномов. Найдите сумму, разность и произведение полученных рядов.

Глава 7

Решение дифференциальных уравнений и их систем

7.1. Решение дифференциальных уравнений первого порядка

Для нахождения решения дифференциального уравнения необходимо:

1. Задать пространство переменных ($SPACE$).
2. Задать уравнение и получить решение в функции (**solveDE**).

Примеры.

Уравнение с разделяющимися переменными:

```
SPACE=Q[x,y];
\solveDE(\d(y,x) = (2/y)\sin(x)\cos(x));
```

Линейное однородное уравнение первого порядка:

```
SPACE=Q[x,y];
\solveDE(x\d(y,x) = y-x\exp(y/x));
```

Уравнение в полных дифференциалах:

```
SPACE=Q[x,y];
\solveDE((3x^2-3y^2+4x)\d(x)-(6xy+4y)\d(y) = 0);
```

7.2. Решение дифференциальных уравнений

Для решения дифференциального уравнения с постоянными коэффициентами необходимо выполнить следующие шаги.

1. Задать пространство переменных ($SPACE$).
2. Задать уравнение (**systLDE**).
3. Задать начальные условия (**initCond**).
4. Получить решение (**solveLDE**).

Примеры.

```
SPACE=R64[t];
g=\systLDE(\d(y, t, 3)+3\d(y, t, 2)+3\d(y, t)+y=1);
f=\initCond(\d(y, t, 0, 0)=0, \d(y, t, 0, 1)=0,
                \d(y, t, 0, 2)=0);
h=\solveLDE(g, f);
\print(h);
```

Результат выполнения:

in: $SPACE = R64[t]$;

$g = y_t^{'''} + 3y_t^{''} + 3y'_t + y = 1$;

$$f = \begin{cases} y_{t=0} &= 0, \\ y'_{t=0} &= 0, \\ y''_{t=0} &= 0, \end{cases}$$

$h = solveLDE(g, f)$;

$print(h)$;

out: $h = (1.00 + (-t^2)e^{-t}/2.00) - (te^{-t} + e^{-t})$;

```
SPACE=R64[t];
g=\systLDE(\d(y, t, 2)-2\d(y, t)+y=\exp(t));
f=\initCond(\d(y, t, 0, 0)=1, \d(y, t, 0, 1)=2);
h=\solveLDE(g, f);
\print(h);
```

Результат выполнения:

in: $SPACE = R64[t]$;

$g = y_t^{''} - 2y'_t + y = e^t$;

$$f = \begin{cases} y_{t=0} &= 1, \\ y'_{t=0} &= 2, \end{cases}$$

$h = solveLDE(g, f)$;
$print(h)$;
out: $h = e^t t^2/2.00 + te^t + e^t$;

```
SPACE=R64[t];
g=\systLDE(\d(y, t, 2)+\d(y, t)-12y=3);
f=\initCond(\d(y, t, 0, 0)=1, \d(y, t, 0, 1)=0);
h=\solveLDE(g, f);
\print(h);
```

Результат выполнения:
in: $SPACE = R64[t]$;
$g = y''_t + y'_t - 12y = 3$;
$f = \begin{cases} y_{t=0} = 1, \\ y'_{t=0} = 0, \end{cases}$
$h = solveLDE(g, f)$;
$print(h)$;
out: $h = 1.11e^{-1.62t} + 2.89e^{0.62t} - 3.00$;

```
SPACE=R64[t];
g=\systLDE(\d(y, t)-2y=0);
f=\initCond(\d(y, t, 0, 0)=1);
h=\solveLDE(g, f);
\print(h);
```

Результат выполнения:
in: $SPACE = R64[t]$;
$g = y'_t - 2y = 0$;
$f = y_{t=0} = 1$;
$h = solveLDE(g, f)$;
$print(h)$;
out: $h = e^{2t}$;

```
SPACE=R64[t];
g=\systLDE(\d(y, t, 2)-4y=4t);
f=\initCond(\d(y, t, 0, 0)=a, \d(y, t, 0, 1)=b);
h=\solveLDE(g, f);
\print(h);
```

Результат выполнения:
in: $SPACE = R64[t]$;
$g = y_t'' - 4y = 4t$;
$f = \begin{cases} y_{t=0} = a, \\ y_{t=0}' = b, \end{cases}$
$h = solveLDE(g, f)$;
$print(h)$;
out: $h = (-8.00 + (-2.00b) + 2.00a)/4.00e^{-t} + (8.00 + 2.00b + 2.00a)/4.00e^{t} - 4.00t$.

```
SPACE=R64[t];
g=\systLDE(\d(y,t,2)-4\d(y,t)+5y=0);
f=\initCond(\d(y, t, 0, 0)=0, \d(y, t, 0, 1)=1);
h=\solveLDE(g, f);
\print(h);
```

Результат выполнения:
in: $SPACE = R64[t]$;
$g = y_t'' - 4y_t' + 5y = 0$;
$f = \begin{cases} y_{t=0} = 0, \\ y_{t=0}' = 1, \end{cases}$
$h = solveLDE(g, f)$;
$print(h)$;
out: $h = 0.53i(e^{(2.05-0.95i)t}) - 0.53i(e^{(2.05+0.95i)t})$.

```
SPACE=R64[t];
g=\systLDE(\d(y,t,2)-\d(y,t)-6y=2);
f=\initCond(\d(y, t, 0, 0)=1, \d(y, t, 0, 1)=0);
h=\solveLDE(g, f);
\print(h);
```

Результат выполнения:
in: $SPACE = R64[t]$;
$g = y_t'' - y_t' - 6y = 2$;
$f = \begin{cases} y_{t=0} = 1, \\ y_{t=0}' = 0, \end{cases}$
$h = solveLDE(g, f)$;
$print(h)$;
out: $h = 0.53e^{3.00t} + 0.80e^{-2.00t} - 0.33$.

```
SPACE=R64[t];
g=\systLDE(\d(y,t,2)-9y=2-t);
f=\initCond(\d(y, t, 0, 0)=0, \d(y, t, 0, 1)=1);
h=\solveLDE(g, f);
\print(h);
```

Результат выполнения:

in: $SPACE = R64[t]$;

$g = y_t'' - 9y = 2 - t$;

$f = \begin{cases} y_{t=0} & = & 0, \\ y'_{t=1} & = & b, \end{cases}$

$h = solveLDE(g, f)$;

$print(h)$;

out: $h = 0.26e^{3.00t} + 0.04e^{-3.00t} + 0.11t - 0.22$.

7.3. Решение систем дифференциальных уравнений

Для решения системы дифференциальных уравнений с постоянными коэффициентами необходимо выполнить следующие шаги.

1. Задать пространство переменных ($SPACE$).
2. Задать систему уравнений (**systLDE**).
3. Задать начальные условия (**initCond**).
4. Получить решение (**solveLDE**).

Примеры.

```
SPACE=R64[t];
g=\systLDE(3\d(x, t)+2x+\d(y, t)=1, \d(x, t)+4\d(y, t)+3y=0);
f=\initCond(\d(x, t, 0, 0)=0, \d(x, t, 0, 1)=0,
            \d(y, t, 0, 0)=0, \d(y, t, 0, 1)=0);
h=\solveLDE(g, f);
```

Результат выполнения:

in: $SPACE = R64[t]$;

$g = \begin{cases} 3x'_t + 2x + y'_t & = & 1, \\ x'_t + 4y'_t + 3y & = & 0, \end{cases}$

$f = \begin{cases} x_{t=0} & = & 0, \\ x'_{t=0} & = & 0, \\ y_{t=0} & = & 0, \\ y'_{t=0} & = & 0, \end{cases}$

$h = solveLDE(g, f)$;
out: $h = [0.50 + (-0.30e^{-0.55t}) + (-0.20e^{-t}), (-0.20e^{-0.55t}) + 0.20e^{-t}]$;

В следующем примере опция STEPBYSTEP=1; дает вывод всех промежуточных выкладок, которые необходимы для решения этой системы дифференциальных уравнений. Отметим, что при этом не следует использовать команду $print()$.

```
SPACE=R64[t];
STEPBYSTEP=1;
g=\systLDE(3\d(x, t)+2x+\d(y, t)=1, \d(x, t)+4\d(y, t)+3y=0);
f=\initCond(\d(x, t, 0, 0)=0, \d(x, t, 0, 1)=0,
            \d(y, t, 0, 0)=0, \d(y, t, 0, 1)=0);
h=\solveLDE(g, f);
```

Результат выполнения:
in: $SPACE = R64[t]$;
$STEPBYSTEP = 1$;

$$g = \begin{cases} 3x'_t + 2x + y'_t &= 1, \\ x'_t + 4y'_t + 3y &= 0, \end{cases}$$

$$f = \begin{cases} x_{t=0} &= 0, \\ x'_{t=0} &= 0, \\ y_{t=0} &= 0, \\ y'_{t=0} &= 0, \end{cases}$$

$h = solveLDE(g, f)$;
out: $h = [0.50 + (-0.30e^{-0.55t}) + (-0.20e^{-t}), (-0.20e^{-0.55t}) + 0.20e^{-t}]$;

Решение системы дифференциальных уравнений с точностью e.

```
SPACE=R[t];
e=0.00000001;
g=\systLDE(3\d(x, t)+2x+\d(y, t)=1, \d(x, t)+4\d(y, t)+3y=0);
f=\initCond(\d(x, t, 0, 0)=0, \d(x, t, 0, 1)=0,
            \d(y, t, 0, 0)=0, \d(y, t, 0, 1)=0);
h=\solveLDE(g, f, e);
```

Результат выполнения:
in: $SPACE = R[t]$;
$e = 0.00000001$;

$$g = \begin{cases} 3x'_t + 2x + y'_t &= 1, \\ x'_t + 4y'_t + 3y &= 0, \end{cases}$$

$$f = \begin{cases} x_{t=0} & = & 0, \\ x'_{t=0} & = & 0, \\ y_{t=0} & = & 0, \\ y'_{t=0} & = & 0, \end{cases}$$

$h = solveLDE(g, f, e);$

out: $h = [0.50 + (-0.30e^{-0.55t}) + (-0.20e^{-t}), (-0.20e^{-0.55t}) + 0.20e^{-t}];$

Вывод графика решения системы дифференциальных уравнений с точностью e.

```
SPACE=R[t];
e=0.00000001;
g=\systLDE(3\d(x, t)+2x+\d(y, t)=1, \d(x, t)+4\d(y, t)+3y=0);
f=\initCond(\d(x, t, 0, 0)=0, \d(x, t, 0, 1)=0,
            \d(y, t, 0, 0)=0, \d(y, t, 0, 1)=0);
h=\solveLDE(g, f, e);
p=\plot(h,[-10,10,-10,10]);
```

Результат выполнения:

in: $SPACE = R[t];$

$e = 0.00000001;$

$$g = \begin{cases} 3x'_t + 2x + y'_t & = & 1, \\ x'_t + 4y'_t + 3y & = & 0, \end{cases}$$

$$f = \begin{cases} x_{t=0} & = & 0, \\ x'_{t=0} & = & 0, \\ y_{t=0} & = & 0, \\ y'_{t=0} & = & 0, \end{cases}$$

$h = solveLDE(g, f, e); \qquad p = plot(h, [-10, 10, -10, 10]);$

out: $h = [0.50 + (-0.30e^{-0.55t}) + (-0.20e^{-t}), (-0.20e^{-0.55t}) + 0.20e^{-t}];$

Вывод графика решения системы дифференциальных уравнений

```
SPACE=R[t];
g=\systLDE(3\d(x, t)+2x+\d(y, t)=1, \d(x, t)+4\d(y, t)+3y=0);
f=\initCond(\d(x, t, 0, 0)=a, \d(x, t, 0, 1)=b,
            \d(y, t, 0, 0)=c, \d(y, t, 0, 1)=d);
h=\solveLDE(g, f, 1);
p=\plot(h,[-10,10,-10,10]);
```

Результат выполнения:

in: $SPACE = R[t];$

$$g = \begin{cases} 3x'_t + 2x + y'_t &= 1, \\ x'_t + 4y'_t + 3y &= 0, \end{cases}$$

$$f = \begin{cases} x_{t=0} &= a, \\ x'_{t=0} &= b, \\ y_{t=0} &= c, \\ y'_{t=0} &= d, \end{cases}$$

$h = solveLDE(g, f, 1); \qquad p = plot(h, [-10, 10, -10, 10]);$

out: $h = [0.50 + (-0.30e^{-0.55t}) + (-0.20e^{-t}), (-0.20e^{-0.55t}) + 0.20e^{-t}];$

```
SPACE=R64[t];
g=\systLDE(\d(x, t)+x-2y=0, \d(y, t)+x+4y=0);
f=\initCond(\d(x, t, 0, 0)=1, \d(y, t, 0, 0)=1);
h=\solveLDE(g, f);
```

Результат выполнения:

in: $SPACE = R64[t];$

$$g = \begin{cases} x'_t + x - 2y &= 0, \\ y'_t + x + 4y &= 0, \end{cases}$$

$$f = \begin{cases} x_{t=0} &= 1, \\ y_{t=0} &= 1, \end{cases}$$

$h = solveLDE(g, f);$

out: $h = [(4.0e^{-2.0t} + (-3.0)e^{-3.0t}, (-2.0)e^{-2.0t} + 3.0e{-}3.0t];$

```
SPACE=R64[t];
g=\systLDE(\d(x, t)+2x+2y=10\exp(2t), \d(y, t)-2x+y=7\exp(2t));
f=\initCond(\d(x, t, 0, 0)=1, \d(y, t, 0, 0)=3);
h=\solveLDE(g, f);
```

Результат выполнения:

in: $SPACE = R64[t];$

$$g = \begin{cases} x'_t + 2x + 2y &= 10e^{2t}, \\ y'_t - 2x + y &= 7e^{2t}, \end{cases}$$

$$f = \begin{cases} x_{t=0} &= 1, \\ y_{t=0} &= 1, \end{cases}$$

$h = solveLDE(g, f);$

out: $h = [e^{2.0t}, 3.0e^{2.0t}];$

```
SPACE=R64[t];
g=\systLDE(\d(x, t)-y+z=0, -x-y+\d(y, t)=0, -x-z+\d(z, t)=0);
```

```
f= \initCond(\d(x, t, 0, 0)=1, \d(y, t, 0, 0)=2,
             \d(z, t, 0, 0)=3);
h= \solveLDE(g, f);
\print(h);
```

Результат выполнения:

in: $SPACE = R64[t]$;

$$g = \begin{cases} x'_t - y + z &= 0, \\ -x - y + y'_t &= 0, \\ -x - z + z'_t &= 0, \end{cases}$$

$$f = \begin{cases} x_{t=0} &= 1, \\ y_{t=0} &= 2, \\ z_{t=0} &= 3, \end{cases}$$

$h = solveLDE(g, f)$;

$print(h)$;

out: $h = [(-2.00) + 5.00e^t + (-1.00e^t)t, 2.00 + (-1.00e^t), (-2.00) + 4.00e^t + (-1.00e^t)t]$;

```
SPACE=R64[t];
g=\systLDE(\d(x, t, 2)+\d(x, t)-\d(y, t)=1,
           \d(x, t)+x-\d(y, t, 2)=1+4\exp(t));
f=\initCond(\d(x, t, 0, 0)=1, \d(x, t, 0, 1)=2,
            \d(y, t, 0, 0)=0, \d(y, t, 0, 1)=1);
h= \solveLDE(g, f);
\print(h);
```

Результат выполнения:

in: $SPACE = R64[t]$;

$$g = \begin{cases} x''_t + x'_t - y'_t &= 1, \\ x'_t + x - y''_t &= 1 + 4e^t, \end{cases}$$

$$f = \begin{cases} x_{t=0} &= 1, \\ x'_{t=0} &= 2, \\ y_{t=0} &= 0, \\ y'_{t=0} &= 1, \end{cases}$$

$h = solveLDE(g, f)$;

$print(h)$;

out: $h = [1.00+2.00e^t+(-1.00e^t)t+(-2.00e^{-t})+(-1.00e^{-t})t, (-2.00)+(-1.00t) + 3.00e^t + (-2.00e^t)t + (-1.00e^{-t})]$;

```
SPACE=R[t];
g=\systLDE(3\d(x, t)+2x+\d(y, t)=1, \d(x, t)+4\d(y, t)+3y=0);
f=\initCond(\d(x, t, 0, 0)=a, \d(x, t, 0, 1)=b,
            \d(y, t, 0, 0)=c, \d(y, t, 0, 1)=d);
h=\solveLDE(g, f, 1);
p=\plot(h,[-10,10,-10,10]);
```

Результат выполнения:

in: $SPACE = R[t]$;

$$g = \begin{cases} 3x'_t + 2x + y'_t &= 1, \\ x'_t + 4y'_t + 3y &= 0, \end{cases}$$

$$f = \begin{cases} x_{t=0} &= a, \\ x'_{t=0} &= b, \\ y_{t=0} &= c, \\ y'_{t=0} &= d, \end{cases}$$

$h = solveLDE(g, f, 1);$ $p = plot(h, [-10, 10, -10, 10]);$

out: $h = [0.50 + (-0.30e^{-0.55t}) + (-0.20e^{-t}), (-0.20e^{-0.55t}) + 0.20e^{-t}];$

```
SPACE=R64[t];
g=\systLDE(\d(x, t)+3x-4y=9\exp(2t),
           2x+\d(y, t)-3y=3\exp(2t));
f=\initCond(\d(x, t, 0, 0)=2, \d(y, t, 0, 0)=0);
h=\solveLDE(g, f);
\print(h);
```

Результат выполнения:

in: $SPACE = R64[t]$;

$$g = \begin{cases} x'_t + 3x - 4y &= 9e^{2t}, \\ 2x + y'_t - 3y &= 3e^{2t}, \end{cases}$$

$$f = \begin{cases} x_{t=0} &= 2, \\ y_{t=0} &= 0, \end{cases}$$

$h = solveLDE(g, f);$

$print(h);$

out: $h = [e^t + e^{2.00t}, e^t + (-1.00e^{2.00t})];$

```
SPACE=R64[t];
g=\systLDE(\d(x, t)-x-2y=0, \d(y, t)-2x-y=1);
f=\initCond(\d(x, t, 0, 0)=0, \d(y, t, 0, 0)=5);
h=\solveLDE(g, f);
\print(h);
```

Результат выполнения:

in: $SPACE = R64[t]$;

$g = \begin{cases} x'_t - x - 2y &= 0, \\ y'_t - 2x - y &= 1, \end{cases}$

$f = \begin{cases} x_{t=0} &= 0, \\ y_{t=0} &= 5, \end{cases}$

$h = solveLDE(g, f)$;

$print(h)$;

out: $h = [(-0.67) + 0.17e^{3.00t} + 0.50e^{-t}, 0.33 + (-0.50e^{-t}) + 0.17e^{3.00t}]$;

```
SPACE=R64[t];
g=\systLDE(\d(x, t, 2)+\d(y, t)+y=\exp(t)-t,
           \d(x, t)-x+2\d(y, t, 2)-y=-\exp(-t));
f=\initCond(\d(x, t, 0, 0)=1, \d(x, t, 0, 1)=2,
            \d(y, t, 0, 0)=0, \d(y, t, 0, 1)=0);
h=\solveLDE(g, f);
\print(h);
```

Результат выполнения:

in: $SPACE = R64[t]$;

$g = \begin{cases} x''_t + y'_t + y &= e^t - t, \\ x'_t - x + 2y''_t - y &= -e^{-t}, \end{cases}$

$f = \begin{cases} x_{t=0} &= 1, \\ x'_{t=0} &= 2, \\ y_{t=0} &= 0, \\ y'_{t=0} &= 0, \end{cases}$

$h = solveLDE(g, f)$;

$print(h)$;

out: $h = [1.00t + e^t, 1.00 + (-1.00t) + (-1.00e^{-t})]$;

```
SPACE=R64[t];
g=\systLDE(\d(x, t, 2)+\d(y, t)=\sh(t)-\sin(t)-t,
           \d(y, t, 2)+\d(x, t)=\ch(t)-\cos(t));
f=\initCond(\d(x, t, 0, 0)=0, \d(x, t, 0, 1)=2,
            \d(y, t, 0, 0)=1, \d(y, t, 0, 1)=0);
h=\solveLDE(g, f);
\print(h);
```

Результат выполнения:

in: $SPACE = R64[t]$;

$$g = \begin{cases} x''_t + y'_t &= sh(t) - sin(t) - t, \\ y''_t + x'_t &= \mathrm{ch}(t) - cos(t), \end{cases}$$

$$f = \begin{cases} x_{t=0} &= 0, \\ x'_{t=0} &= 2, \\ y_{t=0} &= 1, \\ y'_{t=0} &= 0, \end{cases}$$

$h = solveLDE(g, f)$;

$print(h)$;

out: $h = [1.00t + 0.50e^t + (-0.50e^{-t}), (-1.00t^2)/2.00 + 0.50e^{1.00\mathbf{it}} + (0.50e^{-1.00\mathbf{it}})]$;

```
SPACE=R64[t];
g=\systLDE(\d(x, t, 2)-\d(x, t)+\d(y, t)=\exp(-t)+\cos(t),
           \d(x, t)-\d(y, t, 2)-\d(y, t)=2\exp(t)+\sin(t));
f=\initCond(\d(x, t, 0, 0)=2, \d(x, t, 0, 1)=1,
            \d(y, t, 0, 0)=0, \d(y, t, 0, 1)=1);
h=\solveLDE(g, f);
\print(h);
```

Результат выполнения:

in: $SPACE = R64[t]$;

$$g = \begin{cases} x''_t - x'_t + y'_t &= e^{-t} + cos(t), \\ x'_t - y''_t - y'_t &= 2e^t + sin(t), \end{cases}$$

$$f = \begin{cases} x_{t=0} &= 2, \\ x'_{t=0} &= 1, \\ y_{t=0} &= 0, \\ y'_{t=0} &= 1, \end{cases}$$

$h = solveLDE(g, f)$;

$print(h)$;

out: $h = [0.50e^{1.00\mathbf{it}} + 0.50e^{-1.00\mathbf{it}} + (-1.00e^{-t}), 0.50\mathbf{i}(e^{1.00\mathbf{it}}) + (-0.50\mathbf{i}(e^{-1.00\mathbf{it}})) + (2.00e^t)]$.

```
SPACE=R64[t];
g=\systLDE(\d(x, t)-y+z=0, -x-y+\d(y, t)=0, -x-z+\d(z, t)=0);
f= \initCond(\d(x, t, 0, 0)=a, \d(y, t, 0, 0)=b,
             \d(z, t, 0, 0)=c);
h= \solveLDE(g, f);
\print(h);
```

Результат выполнения:

in: $SPACE = R64[t]$;

$$g = \begin{cases} {x'}_t - y + z &= 0, \\ -x - y + {y'}_t &= 0, \\ -x - z + {z'}_t &= 0, \end{cases}$$

$$f = \begin{cases} x_{t=0} &= a, \\ y_{t=0} &= b, \\ z_{t=0} &= c, \end{cases}$$

$h = solveLDE(g, f)$;

$print(h)$;

out: $h = [b - a - c + (-b + a + 2.00c)e^t + (b - c)e^t t, -b + c + a + (b - c)e^t, -c - a + b) + (c + a)e^t + (-c + b)e^t t]$;

```
SPACE=R64[t];
g=\systLDE(\d(y, t)+y-3x=0, -x-y+\d(x, t)=\exp(t));
f= \initCond(\d(x, t, 0, 0)=0, \d(y, t, 0, 0)=0);
h= \solveLDE(g, f);
\print(h);
```

Результат выполнения:

in: $SPACE = R64[t]$;

$$g = \begin{cases} {y'}_t - y - 3x &= 0, \\ -x - y + {x'}_t &= e^t, \end{cases}$$

$$f = \begin{cases} x_{t=0} &= 0, \\ y_{t=0} &= 0, \end{cases}$$

$h = solveLDE(g, f)$;

$print(h)$;

out: $h = [-e^t + 0.25e^{-2.00t} + 0.75e^{2.00t}, -0.08e^{-2.00t} + 0.75e^{2.00t} - 0.67e^t]$;

```
SPACE=R64[t];
g=\systLDE(\d(y, t)+\d(x, t)-x=\exp(t), 2\d(y, t)+\d(x, t)+2x=\cos(t)
f= \initCond(\d(x, t, 0, 0)=0, \d(y, t, 0, 0)=0);
h= \solveLDE(g, f);
\print(h);
```

Результат выполнения:

in: $SPACE = R64[t]$;

$$g = \begin{cases} {y'}_t + {x'}_t - x &= e^t, \\ 2{y'}_t + {x'}_t + 2x &= cos(t), \end{cases}$$

$f = \left\{ \begin{array}{rcl} x_{t=0} & = & 0, \\ y_{t=0} & = & 0, \end{array} \right.$

$h = solveLDE(g, f)$;

$print(h)$;

out: $h = [(0.12 + 0.03i)e^{it} + (0.12 - 0.03i)e^{-it} - 0.67e^{t} + 0.43e^{4.00t}, -0.32e^{4.00t} - 0.50 + e^{t} + (-0.09 - 0.15i)e^{it} + (-0.09 + 0.15i)e^{-it}]$;

```
SPACE=R64[t];
g=\systLDE(\d(y, t)-y+x=1.5t^2, \d(x, t)+2x+4y=4t+1);
f= \initCond(\d(x, t, 0, 0)=0, \d(y, t, 0, 0)=0);
h= \solveLDE(g, f);
\print(h);
```

Результат выполнения:

in: $SPACE = R64[t]$;

$g = \left\{ \begin{array}{rcl} y'_t - y + x & = & 1.5t^2, \\ x'_t + 2x + 4y & = & 4t + 1, \end{array} \right.$

$f = \left\{ \begin{array}{rcl} x_{t=0} & = & 0, \\ y_{t=0} & = & 0, \end{array} \right.$

$h = solveLDE(g, f)$;

$print(h)$;

out: $h = [t + t^2, -0.5t^2]$;

```
SPACE=R64[t];
g=\systLDE(\d(y, t)+y-x-z=0, \d(x, t)-y+x-z=0, \d(z, t)-y-x-z=0);
f= \initCond(\d(x, t, 0, 0)=0, \d(y, t, 0, 0)=1,
             \d(z, t, 0, 0)=0);
h= \solveLDE(g, f);
\print(h);
```

Результат выполнения:

in: $SPACE = R64[t]$;

$g = \left\{ \begin{array}{rcl} y'_t + y - x - z & = & 0, \\ x'_t - y + x - z & = & 0, \\ z'_t - y - x - z & = & 0, \end{array} \right.$

$f = \left\{ \begin{array}{rcl} x_{t=0} & = & 0, \\ y_{t=0} & = & 1, \\ z_{t=0} & = & 0, \end{array} \right.$

$h = solveLDE(g, f)$;

$print(h)$;
out: $h = [0.33e^{-t}+0.17e^{2.00t}+0.5e^{-2.00t}, 0.33e^{2.00t}-0.33e^{-t}, 0.17e^{2.00t}+0.33e^{-t}-0.5e^{-2.00t}]$;

```
SPACE=R64[t];
g=\systLDE(\d(y, t)-x+z=0, \d(x, t)+2y-x=0, \d(z, t)-2y+x=0);
f= \initCond(\d(x, t, 0, 0)=0, \d(y, t, 0, 0)=1,
             \d(z, t, 0, 0)=0);
h= \solveLDE(g, f);
\print(h);
```

Результат выполнения:
in: $SPACE = R64[t]$;

$$g = \begin{cases} {y'}_t - x + z &= 0, \\ {x'}_t + 2y - x &= 0, \\ {z'}_t - 2y + x &= 0, \end{cases}$$

$$f = \begin{cases} x_{t=0} &= 0, \\ y_{t=0} &= 1, \\ z_{t=0} &= 0, \end{cases}$$

$h = solveLDE(g, f)$;
$print(h)$;
out: $h = [-0.52ie^{(0.5+1.94i)t}+0.52ie^{(0.5-1.94i)t}, (0.5+0.13i)e^{(0.5+1.94i)t}+(0.5-0.13i)e^{(0.5-1.94i)t}, 0.52ie^{(0.5+1.94i)t}-0.52ie^{(0.5-1.94i)t}]$;

```
SPACE=R64[t];
g=\systLDE(\d(y, t, 2)+2x=0, \d(x, t, 2)-2y=0);
f=\initCond(\d(x, t, 0, 0)=0, \d(x, t, 0, 1)=0,
            \d(y, t, 0, 0)=0, \d(y, t, 0, 1)=1);
h=\solveLDE(g, f);
\print(h);
```

Результат выполнения:
in: $SPACE = R64[t]$;

$$g = \begin{cases} y''_t + 2x &= 0, \\ x''_t - 2y &= 0, \end{cases}$$

$$f = \begin{cases} x_{t=0} &= 0, \\ x'_{t=0} &= 0, \\ y_{t=0} &= 0, \\ y'_{t=0} &= 1, \end{cases}$$

$h = solveLDE(g, f)$;
$print(h)$;
out: $h = [(0.13 - 0.13i)e^{(1+i)t} + (0.13 + 0.12i)e^{(1-i)t} + (-0.12 - 0.13i)e^{(-1+i)t} + (-0.13 + 0.13i)e^{(-1-i)t}, (-0.13 - 0.13i)e^{(1+i)t} + (-0.13 + 0.13i)e^{(1-i)t} + (0.13 - 0.13i)e^{(-1+i)t} + (0.13 + 0.13i)e^{(-1-i)t}]$;

```
SPACE=R64[t];
g=\systLDE(\d(x, t)-8y+x=0, \d(y, t)-x-y=0);
f= \initCond(\d(x, t, 0, 0)=a, \d(y, t, 0, 0)=b);
h= \solveLDE(g, f);
\print(h);
```

Результат выполнения:
in: $SPACE = R64[t]$;

$$g = \begin{cases} x'_t - 8y + x &= 0, \\ y'_t - x - y &= 0, \end{cases}$$

$$f = \begin{cases} x_{t=0} &= a, \\ y_{t=0} &= b, \end{cases}$$

$h = solveLDE(g, f)$;
$print(h)$;
out: $h = [((4b + a)/3)e^{3.00t} + ((-4 * b + 2a)/3)e^{-3.00t}, ((-a + 2b)/6)e^{-3.00t} + ((a + 4b)/6)e^{3.00t}]$;

```
SPACE=R64[t];
g=\systLDE(\d(x, t)+3x-4y=9(\exp(t))^2, \d(y, t)+2x-3y=3(\exp(t))^2);
f= \initCond(\d(x, t, 0, 0)=2, \d(y, t, 0, 0)=0);
h= \solveLDE(g, f);
\print(h);
```

Результат выполнения:
in: $SPACE = R64[t]$;

$$g = \begin{cases} x'_t + 3x - 4y &= 9(e^t)^2, \\ y'_t + 2x - 3y &= 3(e^t)^2, \end{cases}$$

$$f = \begin{cases} x_{t=0} &= 2, \\ y_{t=0} &= 0, \end{cases}$$

$h = solveLDE(g, f)$;
$print(h)$;
out: $h = [e^t + e^{2.00t}, e^t - e^{2.00t}]$;

```
SPACE=R64[t];
g=\systLDE(\d(x, t, 2)+\d(y, t)=\sh(t)-\sin(t)-t, \d(y, t, 2)-\d(x, t
f=\initCond(\d(x, t, 0, 0)=2, \d(x, t, 0, 1)=0,
            \d(y, t, 0, 0)=0, \d(y, t, 0, 1)=1);
h=\solveLDE(g, f);
\print(h);
```

Результат выполнения:

in: $SPACE = R64[t]$;

$g = \begin{cases} x''_t + y'_t &= sh(t) - sin(t) - t, \\ y''_t - x'_t &= ch(t) - cos(t), \end{cases}$

$f = \begin{cases} x_{t=0} &= 2, \\ x'_{t=0} &= 0, \\ y_{t=0} &= 0, \\ y'_{t=0} &= 1, \end{cases}$

$h = solveLDE(g, f)$;

$print(h)$;

out: $h = [1 - t + (0.5 - 0.5i)e^{it} + (0.5 + 0.5i)e^{-it}, -1 - 0.5t^2 - 0.5ie^{it} + 0.5ie^{-it} + 0.5e^t + 0.5e^{-t}]$;

```
SPACE=R64[t];
g=\systLDE(\d(x, t)+5y-4x=0, \d(y, t)-x=0);
f= \initCond(\d(x, t, 0, 0)=0, \d(y, t, 0, 0)=1);
h= \solveLDE(g, f);
\print(h);
```

Результат выполнения:

in: $SPACE = R64[t]$;

$g = \begin{cases} x'_t + 5y - 4x &= 0, \\ y'_t - x &= 0, \end{cases}$

$f = \begin{cases} x_{t=0} &= 0, \\ y_{t=0} &= 1, \end{cases}$

$h = solveLDE(g, f)$;

$print(h)$;

out: $h = [(0.5 + i)e^{(2+i)t} + (0.5 - i)e^{(2-i)t}, 2.5ie^{(2+i)t} - 2.5ie^{(2-i)t}]$;

7.4. Прямое и обратное преобразование Лапласа

```
SPACE=R64[t];
L=\laplaceTransform(\exp(3t));
\print(L);
```

Результат выполнения:
in: $SPACE = R64[t]$;
$L = laplaceTransform(exp(3t))$.
$print(L)$;
out: $L = \frac{1.0}{t-3.0}$

```
SPACE=R64[t];
L=\inverseLaplaceTransform(1/(t-3));
\print(L);
```

Результат выполнения:
in: $SPACE = R64[t]$;
$L = inverseLaplaceTransform(1/(t-3))$.
$print(L)$;
out: $L = exp(3t)$

7.5. Расчет характеристик динамических объектов и систем

Для нахождения передаточной функции объекта необходимо выполнить следующие шаги:

1. Задать пространство переменных ($SPACE$).
2. Задать уравнение входа - x.
3. Задать уравнение выхода - y.
4. Получить решение (**solveWFDS**).

Примеры.

```
SPACE = R64[t];
f = \d(y,t,2)+2\d(y,t);
g = 3x;
h = \solveTransferFunction(g,f);
```

```
\print(h);
\set2D(-10, 10, -10, 10,'p','W(p)','Передаточная функция');
p=\plot(h);
```

Результат выполнения:
in: $SPACE = R64[t]$;
$f = y_t^{''} + 2y_t^{''}$;
$g = 3x$;
$h = solveTransferFunction(g, f)$;
$print(h)$;
$set2D(-10, 10, -10, 10,' p',' W(p)',' ')$;
$p = plot(h)$;
out: $h = [3.0/(p^2 + 2.0p)]$;

Для нахождения временных характеристик объекта необходимо выполнить следующие шаги:

1. Задать пространство переменных ($SPACE$).
2. Задать уравнение входа - x.
3. Задать уравнение выхода - y.
4. Получить решение (**solveTPDS**).

```
SPACE = R64[t];
f = \d(y,t,2)+2\d(y,t);
g = 3x;
h = \solveTimeResponse(g,f);
\print(h);
\set2D(-10, 10, -10, 10,'p','k(p),h(p)','Временные характеристики');
p=\plot(h);
```

Результат выполнения:
in: $SPACE = R64[t]$;
$f = y_t^{''} + 2y_t^{''}$;
$g = 3x$;
$h = solveTimeResponse(g, f)$;
$print(h)$;
$set2D(-10, 10, -10, 10,' p',' k(p), h(p)',' ')$;
$p = plot(h)$;
out: $h = [(1.5exp(2.0p) * 3.0 + (-1.5) * 3.0), ((-0.75) + (-1.5p) + 0.75exp(2p))]$;

Для нахождения частотных характеристик объекта необходимо выполнить следующие шаги:

1. Задать пространство переменных ($SPACE$).
2. Задать уравнение входа - x.
3. Задать уравнение выхода - y.
4. Получить решение (**solveCHDS**).

```
SPACE = R64[t];
f = \d(y,t,2)+2\d(y,t);
g = 3x;
h = \solveFrequenceResponse(g,f);
SPACE = R64[j,p];
\print(h);
```

Результат выполнения:

in: $SPACE = R64[t]$;
$f = y_t'' + 2y_t''$;
$g = 3x$;
$h = solveFrequenceResponse(g, f)$;
$SPACE = R64[j, p]$;
$print(h)$;

out: $h = [3.0/(p^2j^2+2.0pj), sqrt9.0/(p^4 + 4.0p^2), arctg(sqrt2/p), 20.0lg(sqrt9.0/(p^4$

7.6. Контрольные задания

В Mathpar решите

- дифференциальное уравнение $y'' + 3y' - y = e^t$ с начальными условиями $y(0) = 1,\ y'(0) = 1$,
- систему дифференциальных уравнений

$$x'' + y' = \cos t + t^2,$$

$$y'' + x' = \sin t,$$

с начальными условиями $x(0) = y(0) = 0,\ x(1) = 2,\ y(1) = 1$.

Глава 8

Полиномиальные вычисления

8.1. Вычисление значения полинома в точке

Для вычисления значения функции в точке необходимо выполнить команду $\backslash$**value**$(f, [var1, var2, \ldots, varn])$, где f — это полином, в который на позиции переменных кольца подставляем соответствующие значения $var1, var2, \ldots, varn$.

Пример.

```
SPACE=R[x, y];
f=x^2+5x(y^3+x);
g=\value(f, [1, 2]);
\print(g);
```

Результат выполнения:
in: $SPACE = R[x, y]$;
$f = x^2 + 5x(y^3 + x)$;
$g = value(f, [1, 2])$;
$print(g)$;
out: $g = 46.00$.

8.2. Приведение полиномов к стандартному виду и разложение полиномов на множители

Для приведения полинома к стандартному виду необходимо выполнить команду **expand**(f), где f — это полином.

Для разложения полинома на множители необходимо выполнить команду **factor**(f), где f — это полином.

Пример.

```
SPACE=Q[x, y];
f= (y^3+x)^2(x+1)^3;
 g=\expand(f);
h=\factor(g);
\print(g,h);
```

Результат выполнения:

in: $SPACE = Q[x, y]$;
$f = (y^3 + x)^2(x + 1)^3$;
$g = expand(f)$;
$h = factor(g)$;
$print(g, h)$;

out: $g = y^6x^3 + 3y^6x^2 + 3y^6x + y^6 + 2y^3x^4 + 6y^3x^3 + 6y^3x^2 + 2y^3x + x^5 + 3x^4 + 3x^3 + x^2$;
$h = (x + 1)^3(y^3 + x)^2$;

8.3. Суммирование полинома по переменным. Геометрические прогрессии

Для суммирования полинома по переменным необходимо выполнить команду **SumOfPol**$(f, [x, y], [x1, x2, y1, y2])$, где f — полином, x, y — переменные по которым ведется суммирование, $x1, x2$ — интервал суммирования по x, $y1, y2$ — интервал суммирования по y.

Если интервалы суммирования для всех переменных совпадают, то можно записать **SumOfPol**$(f, [x, y], [x1, x2])$, где $x1, x2$ — интервал суммирования по x и y.

Пример.

```
SPACE=R[x, y, z];
f=x^2z+xy+y^3xz;
res=\SumOfPol(f, [x, y], [2, 4, -2, 3]);
\print(res);
```

Результат выполнения:
in: $SPACE = R[x, y, z]$;
$f = x^2z + xy + y^3xz$;
$res = SumOfPol(f, [x, y], [2, 4, -2, 3])$;
$print(res)$;
out: $res = 417.00z + 27.00$.

Для преобразования полинома с помощью формулы суммы геометрической прогрессии необходимо выполнить команду **SearchOfProgression**(f). Данная команда ищет геометрическую прогрессию с наибольшим числом членов среди мономов полинома, затем делает это еще раз для оставшихся членов и так далее. Найденные прогрессии записываются в виде $S_n = b_1(q^n - 1)/(q - 1)$, где S_n — сумма первых n членов, b_1 — первый член геометрической прогрессии, q — знаменатель прогрессии.

Пример.

```
SPACE=R[x, y, z];
f = x^3 + x^4 + x^5 + x^6 + x^7 + x^8 + x^9 + x^{10} + x^{11} + x^{12
g = x + x^5 + x^9 + x^13 + xyz + 7x^2y^2z^2 + 7x^3y^3z^3 + 100xy + x
f1 = \SearchOfProgression(f);
g1 = \SearchOfProgression(g);
\print(f1, g1);
```

Результат выполнения:
in: $SPACE = R[x, y, z]$;
$f = x^3 + x^4 + x^5 + x^6 + x^7 + x^8 + x^9 + x^10 + x^11 + x^12 + x^13$;
$g = x+x^5+x^9+x^13+xyz+7x^2y^2z^2+7x^3y^3z^3+100xy+x+x^2+x^3+x^4$;
$f1 = SearchOfProgression(f)$;
$g1 = SearchOfProgression(g)$;
$print(f1, g1)$;
out: $f1 = (x^{14} - x^3)/(x - 1)$;
$g1 = ((100.00yx + x^{13} + x^9 + x) + (z^4y^4x^4 - zyx)/(zyx - 1) + (x^6 - x)/(x - 1) + 6.00z^3y^3x^3 + 6.00z^2y^2x^2)$.

8.4. Вычисление базисов Гребнера

Для вычисления базиса Гребнера полиномиального идеала $[p_1, p_2, \ldots, p_N]$ над рациональными числами можно воспользоваться командой $\backslash\mathbf{groebnerB}(p_1, p_2, \ldots, p_N)$ или командой $\backslash\mathbf{groebner}(p_1, p_2, \ldots, p_N)$. Команда $\backslash\mathbf{groebnerB}()$ вычисляет базиса Гребнера, используя алгоритм Бухбергера, а команда $\backslash\mathbf{groebner}()$ использует матричный вариант алгоритма, предложенный Фужером. Используется обратное лексикографическое упорядочение переменных. Порядок на переменных определяется в команде SPACE.

Примеры.

```
SPACE = Q[x, y, z];
b = \groebnerB(x^4y^3 + 2xy^2 + 3x + 1, x^3y^2 + x^2, x^4y + z^2 + xy
\print(b);
```

Результат выполнения:

in: $SPACE = Q[x, y, z]$;
$b = groebnerB(x^4y^3 + 2xy^2 + 3x + 1, x^3y^2 + x^2, x^4y + z^2 + xy^4 + 3)$;
$print(b)$;

out: $b = [z^2 - x^4 + 3x^2 + (-10)x + 9, y + (-9)x^4 + (-3)x^3 - x^2 + (-81)x + 27, x^5 + 9x^2 + (-6)x + 1]$;

```
SPACE = Z[x, y, z];
b = \groebner(x^4y^3 + 2xy^2 + 3x + 1, x^3y^2 + x^2, x^4y + z^2 + xy^
\print(b);
```

Результат выполнения:

in: $SPACE = Q[x, y, z]$;
$b = groebner(x^4y^3 + 2xy^2 + 3x + 1, x^3y^2 + x^2, x^4y + z^2 + xy^4 + 3)$;
$print(b)$;

out: $b = [z^2 - x^4 + 3x^2 + (-10)x + 9, x^5 + 9x^2 + (-6)x + 1, y + (-9)x^4 + (-3)x^3 - x^2 + (-81)x + 27]$.

8.5. Вычисления в факторкольце по идеалу

Функция $\backslash\mathbf{reduceByGB}(f, [g_1, \ldots, g_N])$ редуцирует полином p с помощью данного множества полиномов $g_1, \ldots, g_N$.

```
SPACE = Q[x, y, z];
p = \reduceByGB(5y^2 + 3x^2 + z^2, [y + x, 5z^2 + 5z]);
```

Результат выполнения:
in: $SPACE = Q[x, y, z]$;
$p = \backslash reduceByGB(5y^2 + 3x^2 + z^2, [y + x, 5z^2 + 5z])$;
out: $-z + 8x^2$;

В случае, когда второй аргумент не является редуцированным базисом Гребнера, результат зависит от расположения полиномов в массиве: при наличии нескольких потенциальных редукторов выбирается первый из них.

```
SPACE = Q[x, y];
NotGB1 = [x + y, x^2 + y^2];
imForNotGBset1 = \reduceByGB(x^2 + y^2, NotGB1);
NotGB2 = [x^2 + y^2, x + y];
imForNotGBset2 = \reduceByGB(x^2 + y^2, NotGB2);
GB = \groebner(x+y, x^2+y^2);
imForGB  = \reduceByGB(x^2 + y^2, GB);
\print(GB, imForNotGBset1, imForNotGBset2, imForGB);
```

Результат выполнения:
in: $SPACE = Q[x, y]$;
$NotGB1 = [x + y, x2 + y2]$;
$imForNotGBset1 = reduceByGB(x2 + y2, NotGB1)$;
$NotGB2 = [x2 + y2, x + y]$;
$imForNotGBset2 = reduceByGB(x2 + y2, NotGB2)$;
$GB = groebner(x + y, x2 + y2)$;
$imForGB = reduceByGB(x2 + y2, GB)$;
$print(GB, imForNotGBset1, imForNotGBset2, imForGB)$;
out: $GB = [y + x, x^2]$;
$imForNotGBset1 = 2x^2$;
$imForNotGBset2 = 0$;
$imForGB = 0$;

8.6. Решение систем нелинейных алгебраических уравнений

Для решения системы нелинейных алгребраических уравнений вида:

$p_1 = 0$,
$p_2 = 0$,
...
$p_N = 0$,
use the command **solveNAE**$(p_1, p_2, \ldots, p_N)$.

Перед нахождением корней вычисляется базис Гребнера системы. Если базис содержит уравнения от одной переменной, они решаются, и корни подставляются в оставшиеся уравнения. Корни вычисляются численно. Ответом является вектор решений, в котором каждый элемент в свою очередь является вектором с элементами, соответствующими одному решению. Переменные в решении перечисляются в том же порядке, в котором они указаны при объявлении SPACE.

```
SPACE = R[x, y];
\solveNAE(x^2 + y^2 - 4, y - x^2);

SPACE = R[a, b, c];
S = \solveNAE(a + b + c, a b + a c + b c, a b c - 1);
```

8.7. Другие полиномиальные функции

Для полиномов от нескольких переменных (f,g) можно вычислять НОД, НОК, результант (как определитель их матрицы Сильвестра), дискриминант:

GCD(f,g),
LCM(f,g),
resultant(f,g),
discriminant(f).

При этом главной переменной является старшая (последняя) переменная, которая определена в операторе окружения SPACE.

Пример.

```
PACE=Z[q,r,s,x];
p=x^4+q*x^2+r*x+s;
\discriminant(p);
```

Результат выполнения:
in: $256s^3 - 128s^2q^2 + 144sr^2q + 16sq^4 - 27r^4 - 4r^2q^3$

out: $256s^3 - 128s^2q^2 + 144sr^2q + 16sq^4 - 27r^4 - 4r^2q^3$

8.8. Контрольные задания

В Mathpar вычислите

- $f(1) + f(2) + f(3) + f(4) + f(5)$ для $f = -3x^3 - x^2 + x + 2$,
- базис Гребнера полиномиального идеала для полиномов $x^2 + xy$, $4xy^3 - 2xy - 4$, $y^2 - x$, $x^2y^2 + x + y - 6$.

Глава 9

Матричные функции

9.1. Вычисление транспонированной матрицы

Для вычисления транспонированной матрицы для матрицы A необходимо выполнить команду **transpose**(A) или **A^{T}**.

Пример.

```
SPACE=Z[x];
A=[[1, 2], [4, 5]];
B=A^{T};
\print(B);
```

Результат выполнения:
in: $SPACE = Z[x]$;
$A = \begin{pmatrix} 1 & 2 \\ 4 & 5 \end{pmatrix}$;
$B = A^T$;
$print(B)$;
out: $B = \begin{pmatrix} 1 & 4 \\ 2 & 5 \end{pmatrix}$.

9.2. Получение размеров матрицы и вектора

Вы можете получить число строк, количество столбцов или оба размера матрицы. Для этого нужно выполнить одну из команд **rowNumb**(A), **colNumb**(A) или **size**(A). Эти же команды можно использовать и для вектора-строки, и для вектора-столбца.

Пример.

```
SPACE=Z[x];
A=[[1, 2,1], [4, 5,9]];
r=\rowNumb(A); c=\colNumb(A); s = \size(A);
\print(r,c,s);
```

Результат выполнения:

in: $SPACE = Z[x]$;

$A = \begin{pmatrix} 1 & 2 & 1 \\ 4 & 5 & 9 \end{pmatrix}$;

$r = rowNumb(A); c = colNumb(A); s = size(A)$;

$print(r, c, s)$;

out: $r = 2 \quad c = 3 \quad s = [2, 3]$

9.3. Вычисление обратной и присоединенной матрицы

9.3.1. Вычисление обратной матрицы

Для вычисления обратной матрицы для матрицы A необходимо выполнить команду **inverse**(A) или $\mathbf{A\hat{}\{(-1)\}}$.

Примеры.

```
SPACE=Q[x];
A=[[1, 4], [4, 5]];
B=\inverse(A);
\print(B);
```

Результат выполнения:

in: $SPACE = Q[x]$;

$A = \begin{pmatrix} 1 & 2 \\ 4 & 5 \end{pmatrix}$;

$B = inverse(A)$;
$print(B)$;
out: $B = \begin{pmatrix} (-5)/11 & 4/11 \\ 4/11 & (-1)/11 \end{pmatrix}$;

```
SPACE=Q[x, y];
A=[[x+y, x], [y, \cos(x)]];
B=\inverse(A);
\print(B);
```

Результат выполнения:
in: $SPACE = Q[x, y]$;
$A = \begin{pmatrix} x+y & x \\ y & cos(x) \end{pmatrix}$;
$B = inverse(A)$;
$print(B)$;
out: $B = \begin{pmatrix} \frac{\cos(x)}{y\cos(x)+x\cos(x)+(-yx)} & \frac{-x}{y\cos(x)+x\cos(x)+(-yx)} \\ \frac{-y}{(y\cos(x)+x\cos(x)+(-yx)} & y+\frac{x}{(y\cos(x)+x\cos(x)+(-yx)} \end{pmatrix}$.

9.3.2. Вычисление присоединенной матрицы

Для вычисления присоединенной матрицы для заданной матрицы A необходимо выполнить команду **adjoint**(A) или **A^{\star}**.

Примеры.

```
SPACE=Z[x];
A=[[1, 4], [4, 5]];
B=\adjoint(A);
\print(B);
```

Результат выполнения:
in: $SPACE = Z[x]$;
$A = \begin{pmatrix} 1 & 2 \\ 4 & 5 \end{pmatrix}$;
$B = adjoint(A)$;
$print(B)$;
out: $B = \begin{pmatrix} 5 & -4 \\ -4 & 1 \end{pmatrix}$;

```
SPACE=Z[x, y];
A=[[\cos(y), \sin(x)], [\sin(y), \cos(x)]];
B=\adjoint(A);
\print(B);
```

Результат выполнения:
in: $SPACE = Z[x, y]$;
$A = \begin{pmatrix} cos(y) & sin(x) \\ sin(y) & cos(x) \end{pmatrix}$;
$B = adjoint(A)$;
$print(B)$;
out: $B = \begin{pmatrix} cos(x) & -sin(x) \\ -sin(y) & cos(y) \end{pmatrix}$.

9.4. Вычисление ранга и определителя матрицы

Для вычисления ранга матрицы A необходимо выполнить команду **rank**(A), а для вычисления определителя матрицы A – команду **det**(A).

Примеры.

```
SPACE=Z[x];
A=[[1, 4], [4, 5]];
B=\det(A); r=\rank(A);
\print(B,r);
```

Результат выполнения:
in: $SPACE = Z[x]$;
$A = \begin{pmatrix} 1 & 4 \\ 4 & 5 \end{pmatrix}$;
$B = det(A); r = rank(A)$;
$print(B, r)$;
out: $B = -11$; $r = 2$;

```
SPACE=Z[x, y];
A=[[x+y, \sin(x)], [y, \cos(x)]];
B=\det(A);
\print(B);
```

Результат выполнения:
in: $SPACE = Z[x, y]$;
$A = \begin{pmatrix} x+y & sin(x) \\ y & cos(x) \end{pmatrix}$;
$B = det(A)$;
$print(B)$;
out: $B = y \cdot cos(x) + x \cdot cos(x) - y \cdot sin(x)$.

9.5. Вычисление сопряженной матрицы

Для вычисления сопряженной матрицы необходимо выполнить команду **conjugate**(A) или **A^{\ast}**.

Пример.

```
SPACE=C[x];
A=[[1+\i, 2-\i], [-3, -2\i]];
B=A^{\ast};
\print(B);
```

Результат выполнения:
in: $SPACE = C[x]$;
$A = \begin{pmatrix} 1+i & 2-i \\ -3 & -2i \end{pmatrix}$;
$B = A^*$;
$print(B)$;
out: $B = \begin{pmatrix} 1-1.0i & -3 \\ 2+1.0i & 2.0i \end{pmatrix}$.

9.6. Вычисление SVD-разложения

Для вычисления SVD- разложения матрицы необходимо выполнить команду **SVD**(A). В результате будут вычислены три матрицы $[U, D, V]$. Матрицы U, V - унитарные, матрица D - диагональная: $A = UDV$.

Пример.

```
SPACE=R64[];
A=[[2,3,4], [1,3,3], [2,4,3]];
B=\SVD(A);
\print(B);
```

9.7. Вычисление обобщенной обратной матрицы

Для вычисления обобщенной обратной матрицы Мурра-Пенроуза необходимо выполнить команду **genInverse**(A) или **A**^{+}.

Пример.

```
SPACE=Z[x];
A=[[1, 4, 5], [2, 4, 5]];
B=A^{+};
\print(B);
```

Результат выполнения:
in: $SPACE = Z[x]$;
$A = \begin{pmatrix} 1 & 4 & 5 \\ 2 & 4 & 5 \end{pmatrix}$;
$B = A^{+}$;
$print(B)$;
out: $B = \begin{pmatrix} -1 & 1 & 0 \\ 8/41 & (-4)/41 & 0 \\ 10/41 & (-5)/41 & 0 \end{pmatrix}$.

9.8. Вычисление ядра оператора и эшелонной формы

9.8.1. Вычисление эшелонной формы матрицы

Для вычисления эшелонной формы матрицы A необходимо выполнить команду **toEchelonForm**(A).

Примеры.

```
SPACE=Z[x];
A=[[1, 4], [4, 5]];
B=\toEchelonForm(A);
\print(B);
```

Результат выполнения:
in: $SPACE = Z[x]$;

$A = \begin{pmatrix} 1 & 4 \\ 4 & 5 \end{pmatrix}$;
$B = toEchelonForm(A)$;
$print(B)$;
out: $B = \begin{pmatrix} -11 & 0 \\ 0 & -11 \end{pmatrix}$;

```
SPACE=Z[x, y];
A=[[\cos(y), \sin(x)], [\sin(y), \cos(x)]];
B=\toEchelonForm(A);
\print(B);
```

Результат выполнения:
in: $SPACE = Z[x, y]$;
$A = \begin{pmatrix} cos(y) & sin(x) \\ sin(y) & cos(x) \end{pmatrix}$;
$B = toEchelonForm(A)$;
$print(B)$;
out: $B = \begin{pmatrix} cos(y)cos(x) - sin(x)sin(y) & 0 \\ 0 & cos(y)cos(x) - sin(x)sin(y) \end{pmatrix}$.

9.8.2. Вычисление ядра оператора

Для вычисления ядра оператора матрицы A необходимо выполнить команду **kernel**(A).

Примеры.

```
SPACE=Z[x];
A=[[1, 4], [4, 16]];
B=\kernel(A);
\print(B);
```

Результат выполнения:
in: $SPACE = Z[x]$;
$A = \begin{pmatrix} 1 & 4 \\ 4 & 16 \end{pmatrix}$;
$B = kernel(A)$;
$print(B)$;
out: $B = \begin{pmatrix} 0 & 4 \\ 0 & -1 \end{pmatrix}$;

```
SPACE=Z[x, y];
A=[[x+y, x], [(x+y)x, x^2]];
B=\kernel(A);
\print(B);
```

Результат выполнения:
in: $SPACE = Z[x, y]$;
$A = \begin{pmatrix} x+y & x \\ (x+y)x & x^2 \end{pmatrix}$;
$B = kernel(A)$;
$print(B)$;
out: $B = \begin{pmatrix} 0 & x \\ 0 & -y-x \end{pmatrix}$.

9.9. Вычисление характеристического полинома матрицы

Для вычисления характеристического полинома матрицы A, элементы которой из $R[x_1, \ldots, x_m]$, необходимо задать кольцо полиномов $R[x_1, \ldots, x_m]R[t]$ или $R[t, x_1, \ldots, x_m]$, в котором переменная t — это переменная, по которой строится полином, и выполнить команду **charPolynom**(A). Например,если элементы исходной матрицы из кольца Z[x,y], то можно указать Z[x,y]Z[t] или Z[t,x,y].

Примеры.

```
SPACE=Z[x];
A=[[1, 4], [4, 5]];
B=\charPolynom(A);
\print(B);
```

Результат выполнения:
in: $SPACE = Z[x]$;
$A = \begin{pmatrix} 1 & 2 \\ 4 & 5 \end{pmatrix}$;
$B = charPolynom(A)$;
$print(B)$;
out: $B = x^2 + (-6)x + (-11)$;

```
SPACE=Z[x, y]Z[t];
```

```
A=[[\cos(y), \sin(x)], [\sin(y), \cos(x)]];
B=\charPolynom(A);
\print(B);
```

Результат выполнения:
in: $SPACE = Z[x,y]Z[t]$;
$A = \begin{pmatrix} cos(y) & sin(x) \\ sin(y) & cos(x) \end{pmatrix}$;
$B = adjoint(A)$;
$print(B)$;
out: $B = t^2 + (-\cos(x) - \cos(y))t + \cos(y)\cos(x) - \sin(x)\sin(y)$.

9.10. LSU-разложение

Для вычисления LSU-разложения матрицы A, нужно выполнить команду $\backslash\mathbf{LSU}(A)$.

Результат — это три матриц $[L, S, U]$. Здесь L — нижняя треугольная матрица, U — верхняя треугольная матрица, S — матрица перестановок, умноженная на матрицу, которая является обратной к диагональной матрице. Если элементы матрицы A из коммутативной области R, то и элементы матриц L, S^{-1}, U также принадлежат области R.

Примеры.

```
SPACE=Z[x];
A=[[0, 1, 0], [4, 5, 1],[1, 1, 1]];
B=\LSU(A);
\print(B);
```

Результат выполнения:
in: $SPACE = Z[x]$;
$A = \begin{pmatrix} 0 & 1 & 0 \\ 4 & 5 & 1 \\ 1 & 1 & 1 \end{pmatrix}$;
$B = LSU(A)$
$print(B)$;
out: $B = \left[\begin{pmatrix} 4 & 0 & 0 \\ 0 & 4 & 0 \\ -1 & 1 & 3 \end{pmatrix}, \begin{pmatrix} 0 & 1/16 & 0 \\ 1/4 & 0 & 0 \\ 0 & 0 & 1/12 \end{pmatrix}, \begin{pmatrix} 4 & 5 & 1 \\ 0 & 4 & 0 \\ 0 & 0 & 3 \end{pmatrix} \right]$.

```
SPACE=Z[x];
A=[[1, 4,0,1], [4, 5,5,3],[1,2,2,2],[3,0,0,1]];
B=\LSU(A);
\print(B);
```

Результат выполнения:

in: $SPACE = Z[x]$;

$A = \begin{pmatrix} 1 & -4 & 0 & 1 \\ 4 & 5 & 5 & 3 \\ 1 & 2 & 2 & 2 \\ 3 & 0 & 0 & 1 \end{pmatrix}$;

$B = LSU(A)$

$print(B)$;

out: $B = \begin{bmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 4 & -11 & 0 & 0 \\ 1 & -2 & -12 & 0 \\ 3 & -12 & 60 & -60 \end{pmatrix} \\ \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1/(-11) & 0 & 0 \\ 0 & 0 & 1/132 & 0 \\ 0 & 0 & 0 & 1/720 \end{pmatrix} \\ \begin{pmatrix} 1 & 4 & 0 & 1 \\ 0 & -11 & 5 & -1 \\ 0 & 0 & -12 & -13 \\ 0 & 0 & 0 & -60 \end{pmatrix} \end{bmatrix}$.

```
SPACE=Z[x,y];
A=[[\cos(y),\sin(x)],[\sin(y),\cos(x)]];
B=\LSU(A);
\print(B);
```

Результат выполнения:

in: $SPACE = Z[x, y]$;

$A = \begin{pmatrix} \cos(y) & \sin(x) \\ \sin(y) & \cos(x) \end{pmatrix}$

$B = LSU(A)$

$print(B)$;

$$\text{out: } B = \left[\begin{array}{c} \begin{pmatrix} \cos(y) & 0 \\ \sin(y) & \cos(y)\cos(x) + (-\sin(x)\sin(y)) \end{pmatrix} \\ \\ \begin{pmatrix} 1/\cos(y) & 0 \\ 0 & 1/((\cos(y))^2\cos(x) + (-1\cos(y)\sin(x)\sin(y))) \end{pmatrix} \\ \\ \begin{pmatrix} \cos(y) & \sin(x) \\ 0 & \cos(y)\cos(x) + (-\sin(x)\sin(y)) \end{pmatrix} \end{array} \right].$$

9.11. Разложение Холетского

Эта разложение выполняется с помощью команды, в которой аргумент - это исходная матрица: **cholesky**(A) или **cholesky**$(A, 0)$. При этом матрица должна быть симметричной и положительно определенной, только в этом случае разложение будет правильно вычисленно.

Результат — это две нижние треугольные матрицы: $[L, S]$, при этом $A = l * L^T$ и $S * L = I$.

Для больших плотных матриц, начиная с размера 100x100, можно использовать быстрый алгоритм, в котором применяется умножение блоков по алгоритму Винограда-Штрассена: **cholesky**$(A, 1)$.

Примеры.

```
SPACE=R64[];
 A=[[3,2],[2,4]];
B=\cholesky(A);
\print(B);
```

Результат выполнения:

in: $SPACE = R64[]$;

$A = \begin{pmatrix} 3 & 2 \\ 2 & 4 \end{pmatrix}$;

$B = cholesky(A)$

$print(B)$;

$$\text{out: } B = \left[\begin{pmatrix} 1.73 & 0 \\ 1.15 & 1.63 \end{pmatrix}, \quad \begin{pmatrix} 0.58 & 0 \\ -0.41 & 0.61 \end{pmatrix} \right].$$

9.12. Разложение LSUWMdet

Для вычисления LSU разложения матрицы A, и разложения псевдо-обратной ннужно выполнить команду $\backslash$**LSUWMdet**(A).

Результат — вектор из пяти матриц $[L, S, U, W, M, [[det]]]$. Здесь L и U — нижняя и верхняя треугольные матрицы, S — усеченная взвешенная матрица перестановок, При этом $A = LSU$ и $pseudoInverse(A) = (1/det^2)WSM$. det - ненулевой, максимальный по размеру угловой минор. Если элементы матрицы A из коммутативной области, то все матрицы, кроме S, также принадлежат этой области.

Примеры.

```
SPACE=Z[x,y];
A=[[0, 1, x],[0, 5*y, x],[1+y,(x-1), 1]];
B=\LSUWMdet(A);
\print(B);
```

Результат выполнения:
in: $SPACE = Z[x, y]$;
$A = \begin{pmatrix} 0 & 1 & x \\ 0 & 5y & x \\ (1+y) & (x-1) & 1 \end{pmatrix}$;
$B = LSUMW(A)$
$print(B)$;

$$\text{out: } B = \left[\begin{array}{c} \begin{pmatrix} 1 & 0 & 0 \\ 5y & -5y^2x - 4yx + x & 0 \\ x-1 & 0 & y+1 \end{pmatrix} \\ \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & (-1/(5y^3x + 9y^2x + 3yx - x)) \\ (1/(y+1)) & 0 & 0 \end{pmatrix} \\ \begin{pmatrix} y+1 & 0 & -x^2 + x + 1 \\ 0 & 1 & x \\ 0 & 0 & -5y^2x - 4yx + x \end{pmatrix} \\ \begin{pmatrix} 0 & x^2 - x - 1 & 1 \\ 1 & -yx - x & 0 \\ 0 & y+1 & 0 \end{pmatrix} \\ \begin{pmatrix} -x+1 & 0 & 1 \\ 1 & 0 & 0 \\ -5y^2 - 5y & y+1 & 0 \end{pmatrix} \end{array} \right].$$

9.13. Разложение Брюа

Для вычисления разложения Брюа матрицы A, нужно выполнить команду **BruhatDecomposition**(A).

Результат — вектор из трёх матриц $[V, D, U]$. Здесь V и U — верхние треугольные матрицы, D — матрица перестановок, умноженная на матрицу, которая является обратной к диагональной матрице. Если элементы матрицы A из коммутативной области R, то и элементы матриц V, D^{-1}, U также принадлежат области R.

Примеры.

```
SPACE=Z[x];
A=[[1, 4,0,1], [4, 5,5,3],[1,2,2,2],[3,0,0,1]];
B=\BruhatDecomposition(A);
\print(B);
```

Результат выполнения:
in: $SPACE = Z[x]$;

$A = \begin{pmatrix} 1 & -4 & 0 & 1 \\ 4 & 5 & 5 & 3 \\ 1 & 2 & 2 & 2 \\ 3 & 0 & 0 & 1 \end{pmatrix}$;
$B = BruhatDecomposition(A)$
$print(B)$;

out: $B = \begin{bmatrix} \begin{pmatrix} -24 & 0 & 12 & 1 \\ 0 & 60 & 15 & 4 \\ 0 & 0 & 6 & 1 \\ 0 & 0 & 0 & 3 \end{pmatrix} \\ \begin{pmatrix} 0 & 0 & 1/(-144) & 0 \\ 0 & 0 & 0 & 1/(-1440) \\ 0 & 1/18 & 0 & 0 \\ 1/3 & 0 & 0 & 0 \end{pmatrix} \\ \begin{pmatrix} 3 & 0 & 0 & 1 \\ 0 & 6 & 6 & 5 \\ 0 & 0 & -24 & -16 \\ 0 & 0 & 0 & 60 \end{pmatrix} \end{bmatrix}$.

```
SPACE=Z[x,y];
A=[[\cos(y),\sin(x)],[\sin(y),\cos(x)]];
B=\BruhatDecomposition(A);
\print(B);
```

Результат выполнения:

in: $SPACE = Z[x, y]$;
$A = \begin{pmatrix} \cos(y) & \sin(x) \\ \sin(y) & \cos(x) \end{pmatrix}$
$B = BruhatDecomposition(A)$
$print(B)$;

out: $B = \begin{bmatrix} \begin{pmatrix} -\cos(y) * \cos(x) + \sin(x) * \sin(y) & \cos(y) \\ 0 & \sin(y) \end{pmatrix} \\ \begin{pmatrix} 0 & 1/(-\cos(y)\sin(y)\cos(x) + \sin(x)(\sin(y))^2) \\ 1/\sin(y) & 0 \end{pmatrix} \\ \begin{pmatrix} \sin(y) & \cos(x) \\ 0 & -\cos(y)\cos(x) + \sin(x)\sin(y) \end{pmatrix} \end{bmatrix}$.

Другие операторы.

pseudoInverse и — псевдо обратная матрица. Она в отличие от матрицы Мурра-Пенроуза, удовлетворяет только двум из четырех тождеств. Однако она быстрее вычисляется;

SVD — SVD-разложение матрицы над действительными числами. Результат — вектор из трёх матриц $[U, D, V^T]$. Здесь U, V^T — ортогональные матрицы, D — диагональная матрица.

QR — QR-разложение матрицы над действительными числами. Результат — вектор из двух матриц $[Q, R]$. Здесь Q — ортогональная матрица, R — верхняя треугольная матрица.

sylvester$p1, p2, type = 0 or 1$ — строится матрица Сильвестра по коэффициентам полиномов $p1, p2$. Кольцо Z[x,y,z,u] будет рассматриваться как кольцо Z[u][x,y,z] (кольцо от одной переменной u с коэффициентами из Z[x,y,z].) Если type=0 размер матрицы n1+n2, если type=1 размер матрицы 2 max(n1,n2).

9.14. Решение задач линейного программирования

Пусть задана целевая функция $\sum_{j=1}^{n} c_j x_j$ и условия

$$\sum_{j=1}^{n} a_{ij} x_j \leqslant b_i, \text{ где } i = 1, 2, \ldots, m,$$

$$x_j \geqslant 0, \text{ где } j = 1, 2, \ldots, n.$$

Определим $m \times n$-матрицу $A = (a_{ij})$, m-мерный вектор $b = (b_i)$, n-мерный вектор $c = (c_j)$ и n-мерный вектор $x = (x_j)$.

Тогда целевую функцию можно записать в виде $c^T x$, а условия — в виде

$$Ax \leqslant b,$$

$$x \geqslant 0.$$

Для решения задач линейного программирования нужно выполнить команду **SimplexMax** или **SimplexMin**. Результат — вектор x.

В зависимости от вида задачи возможны следующие варианты вызова команд.

1. Для решения задачи

$$c^T x \to max$$

при условиях

$$Ax \leqslant b,$$

$$x \geqslant 0,$$

используем команду $\backslash$**SimplexMax**(A, b, c).

Если целевую функцию надо минимизировать, т.е.

$$c^T x \to min,$$

то используем команду $\backslash$**SimplexMin**(A, b, c).

Пример.

Максимизировать

$$3x_1 + x_2 + 2x_3$$

при условиях

$$\begin{cases} x_1 + x_2 + 3x_3 \leqslant 30, \\ 2x_1 + 2x_2 + 5x_3 \leqslant 24, \\ 4x_1 + x_2 + 2x_3 \leqslant 36, \\ x_1, x_2, x_3 \geqslant 0. \end{cases}$$

```
SPACE = R64[];
A = [[1, 1, 3],[2, 2, 5],[4, 1, 2]];
b = [30, 24, 36];
c = [ 3,  1,  2];
x = \SimplexMax(A, b, c);
```

Результат выполнения:

in:

out: [8.0, 4.0, 0.0];

2. Для решения задачи

$$c^T x \to max$$

при условиях

$$A_1 x \leqslant b_1,$$

$$A_2x = b_2,$$

$$x \geqslant 0,$$

используем команду **SimplexMax**(A_1, A_2, b_1, b_2, c).

Если целевую функцию надо минимизировать, т.е.

$$c^T x \to min,$$

то используем команду **SimplexMin**(A_1, A_2, b_1, b_2, c).

Пример.

Максимизировать

$$7x_1 + x_3 - 4x_4$$

при условиях

$$\begin{cases} x_1 - x_2 + 2x_3 - x_4 \leqslant 6, \\ 2x_1 + x_2 - x_3 = -1, \\ x_1, x_2, x_3, x_4 \geqslant 0. \end{cases}$$

```
SPACE = R64[];
A1 = [[1, -1,  2, -1]];
A2 = [[2,  1, -1,  0]];
b1 = [ 6];
b2 = [-1];
c  = [7, 0, 1, -4];
x = \SimplexMax(A1, A2, b1, b2, c);
```

Результат выполнения:
in:
out: [0.8, 0.0, 2.6, 0.0];

3. Для решения задачи

$$c^T x \to max$$

при условиях

$$A_1x \leqslant b_1,$$

$$A_2x = b_2,$$

$$A_3x \geqslant b_3,$$

используем команду **SimplexMax**$(A_1, A_2, A_3, b_1, b_2, b_3, c)$.

Если целевую функцию надо минимизировать, т.е.

$$c^T x \to min,$$

то используем команду **SimplexMin**$(A_1, A_2, A_3, b_1, b_2, b_3, c)$.

Пример.

Максимизировать

$$x_1 + x_2$$

при условиях

$$\begin{cases} 4x_1 - x_2 \leqslant 8, \\ 2x_1 + x_2 \leqslant 10, \\ -5x_1 + 2x_2 \geqslant -2, \\ x_1, x_2 \geqslant 0. \end{cases}$$

```
SPACE = R64[];
A1 = [[ 4, -1], [2,  1]];
A3 = [[-5,  2]];
b1 = [ 8, 10];
b3 = [-2];
c  = [1, 1];
x = \SimplexMax(A1, [[]], A3, b1, [], b3, c);
```

Результат выполнения:
in:
out: [2.0, 6.0];

4. Для решения задачи общего вида, как и в предыдущем пункте,

$$c^T x \to max$$

можно обойтись четырьмя параметрами. Можно задать матрицу A, вектор b, целевую функцию c и использовать команду **SimplexMax**$(A, signs, b, c)$, где массив целых чисел $signs$ определяет знаки сравнения: -1 обозначает меньше или равно , 0 обозначает равно и 1 обозначает больше или равно. Должно быть в $signs$ столько же чисел, сколько элементов в векторе b.

Если целевую функцию надо минимизировать, т.е.

$$c^T x \to min,$$

то используем команду **SimplexMin**$(A, signs, b, c)$.

Пример.
Минимизировать

$$-2x_1 - 4x_2 - 2x_3$$

при условиях

$$\begin{cases} -2x_1 + x_2 + x_3 \leqslant 4, \\ -x_1 + x_2 + 3x_3 \leqslant 6, \\ x_1 - 3x_2 + x_3 \leqslant 2, \\ x_1, x_2, x_3 \geqslant 0. \end{cases}$$

In:

```
SPACE = R64[];
A = [[-2,  1, 1],[-1,  1, 3],[ 1, -3, 1]];
b = [ 4,  6,  2];
c = [-2, -4, -2];
signs = [-1, -1, -1];
x = \SimplexMin(A, signs, b, c);
```

Результат выполнения:
in:
out: Simplex: This problem has no finite solution.

9.15. Контрольные задания

В системе Mathpar для матриц $A \in \mathrm{Q}$, $B \in \mathrm{Z}/(17)\mathrm{Z}[x, y]$

$$A = \begin{pmatrix} 3 & 0 & 7 \\ 0 & -5 & 5 \\ 4 & 17 & 2 \end{pmatrix}, \; B = \begin{pmatrix} 0 & x & y+1 \\ x^2 & 0 & -y-x \\ x^2+y & -x & -3yx \end{pmatrix}$$

вычислите

- транспонированную матрицу,
- определитель,
- обратную матрицу,
- присоединенную матрицу,
- характеристический полином,

- разложение Брюа,
- LSU разложение,
- ядро оператора и эшелонную форму.

Глава 10

Функции теории вероятностей и математической статистики

10.1. Функции непрерывных случайных величин

Непрерывная случайная величина задается с помощью интервала (a, b) и плотности распределения вероятностей $f(x)$. Например, $a = 0; b = 2; f = 1/4 * x^2$.

Для непрерывных случайных величин определены следующие функции:

mathExpectation$(a, b, f(x))$ вычисляет математическое ожидание непрерывной случайной величины.

dispersion$(a, b, f(x))$ вычисляет дисперсию непрерывной случайной величины.

meanSquareDeviation$(a, b, f(x))$ вычисляет среднее квадратичное отклонение непрерывной случайной величины.

plotDistributionFunction$(a, b, F(x))$ строит функцию распределения непрерывной случайной величины, где F(x) — функция рас-

пределения на (a,b).

Примеры.

```
SPACE=R64[x];
a=0;
b=4;
f=1/4;
g=\mathExpectation(a,b,f);
g1=\dispersion(a,b,f);
g2=\meanSquareDeviation(a,b,f);
\print(g, g1, g2);
```

Результат выполнения:
in: $SPACE = R64[x];;$
$a = 0; b = 4; f = 0.25;$
$g = mathExpectation(a, b, f);$
$g1 = dispersion(a, b, f);$
$g2 = meanSquareDeviation(a, b, f);$
$print(g, g1, g2);$
out: $g = 2;$
$g1 = 1.33;$
$g2 = 1.15;$

```
SPACE=R64[x];
a=0;
b=4;
F=1/16*x^2;
\plotDistributionFunction(a, b, F);
```

Результат выполнения:
in: $SPACE = R64[x];$
$a = 0; b = 4; F = 1/16 * x^2$
$plotDistributionFunction(a, b, F);$
out: рис. ??.

10.2. Функции дискретных случайных величин

Дискретная случайная величина задается как матрица, имеющая две строки. В первой строке записаны значения случайной ве-

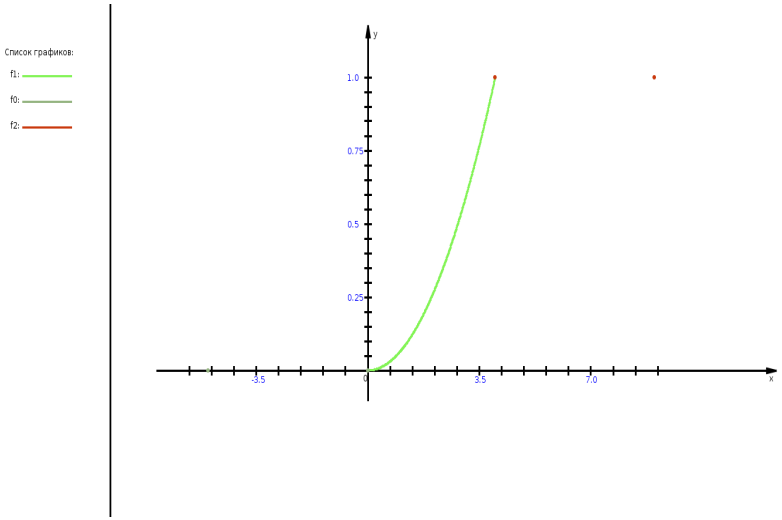


Рисунок 10.1: Функция распределения непрерывной случайной величины из примера 2

личины, во второй — соответствующие им вероятности. То есть каждый элемент второй строки является числом на отрезке [0, 1], при этом и сумма всех элементов второй строки должна быть равна 1. Например, $DRQ = ([1, 2, 3, 4, 5], [0.4, 0.1, 0.1, 0.2, 0.2])$.

Для дискретных случайных величин определены следующие функции.

mathExpectation(DRQ) вычисляет математическое ожидание дискретной случайной величины DRQ.

dispersion(DRQ) вычисляет дисперсию дискретной случайной величины DRQ.

meanSquareDeviation(DRQ) вычисляет среднее квадратичное отклонение дискретной случайной величины DRQ.

addQU($DRQ1, DRQ2$) складывает две дискретные случайные величины $DRQ1$ и $DRQ2$.

multiplyQU($DRQ1, DRQ2$) умножает две дискретные случайные величины $DRQ1$ и $DRQ2$.

covariance($DRQ1, DRQ2$) вычисляет коэффициент ковариа-

ции двух дискретных случайных величин $DRQ1$ и $DRQ2$.

correlation($DRQ1, DRQ2$) вычисляет коэффициент корреляции двух дискретных случайных величин $DRQ1$ и $DRQ2$.

plotPolygonDistribution(DRQ) строит многоугольник распределения дискретной случайной величины DRQ.

plotDistributionFunction(DRQ) строит функцию распределения дискретной случайной величины DRQ.

simplifyQU(DRQ) упрощает дискретную случайную величину DRQ.

Примеры.

```
SPACE=R64[x];
DRQ=[[1, 2], [0.2, 0.8]];
g=\mathExpectation(DRQ);
g1=\dispersion(DRQ);
g2=\meanSquareDeviation(DRQ);
\print(g, g1, g2);
```

Результат выполнения:

in: $SPACE = R64[x];;$

$DRQ = \begin{pmatrix} 1 & 2 \\ 0.2 & 0.8 \end{pmatrix};$

$g = mathExpectation(DRQ);$

$g1 = dispersion(DRQ);$

$g2 = meanSquareDeviation(DRQ);$

$print(g, g1, g2);$

out: $g = 1.8;$

$g1 = 0.16;$

$g2 = 0.39;$

```
SPACE=R64[x];
DRQ=[[7, 5, 3, 5, 1], [0.2, 0.1, 0.3, 0.1, 0.3]];
g=\simplifyQU(DRQ);
\print(g);
```

Результат выполнения:

in: $SPACE = R64[x];$

$DRQ = \begin{pmatrix} 7 & 5 & 3 & 5 & 1 \\ 0.2 & 0.1 & 0.3 & 0.1 & 0.3 \end{pmatrix};$

$g = simplifyQU(DRQ);$

$print(g)$;

out: $g = \begin{pmatrix} 1 & 3 & 5 & 7 \\ 0.3 & 0.3 & 0.2 & 0.2 \end{pmatrix}$;

```
SPACE=R64[x];
DRQ1=[[0, 1], [0.33333, 0.66666]];
DRQ2=[[1, 2], [0.25, 0.75]];
g=\addQU(DRQ1, DRQ2);
g1= \multiplyQU(DRQ1, DRQ2);
\print(g, g1);
```

Результат выполнения:

in: $SPACE = R64[x]$;

$DRQ1 = \begin{pmatrix} 0 & 1 \\ 0.33333 & 0.66666 \end{pmatrix}$;

$DRQ2 = \begin{pmatrix} 1 & 2 \\ 0.25 & 0.75 \end{pmatrix}$;

$g = addQU(DRQ1, DRQ2)$;

$g1 = multiplyQU(DRQ1, DRQ2)$;

$print(g, g1)$;

out: $g = \begin{pmatrix} 1 & 2 & 3 \\ 0.08 & 0.41 & 0.49 \end{pmatrix}$;

$g1 = \begin{pmatrix} 0 & 1 & 2 \\ 0.33 & 0.16 & 0.49 \end{pmatrix}$;

```
SPACE=R64[x];
DRQ=[[-7, -2, 0, 3, 5, 7, 9],
  [0.3, 0.05, 0.2, 0.1, 0.1, 0.2, 0.05]];
\plotPolygonDistribution(DRQ);
```

Результат выполнения:

in: $SPACE = R64[x]$;

$DRQ = \begin{pmatrix} -7 & -2 & 0 & 3 & 5 & 7 & 9 \\ 0.3 & 0.05 & 0.2 & 0.1 & 0.1 & 0.2 & 0.05 \end{pmatrix}$;

$plotPolygonDistribution(DRQ)$;

out: рис. **??**.

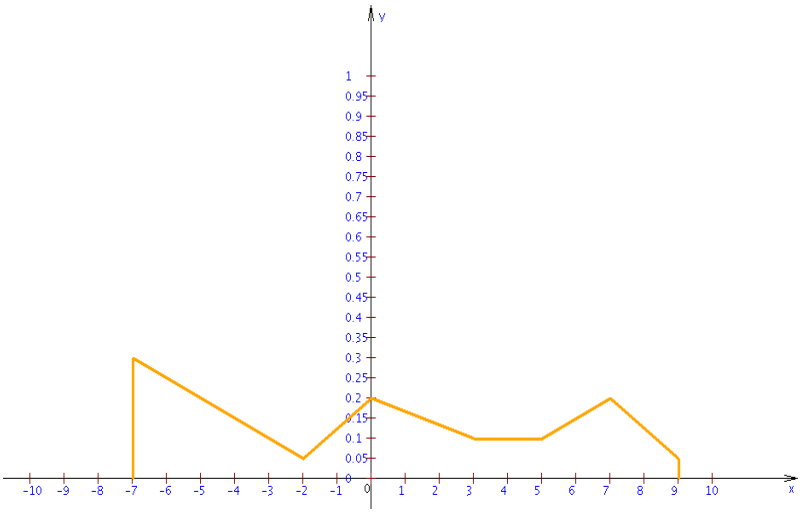


Рисунок 10.2: Многоугольник распределения дискретной случайной величины из примера 4

10.3. Функции для выборок

Выборка задается как матрица из одной строки. Например, $[1, 7, 10, 15]$.

sampleMean(S) вычисляет выборочное среднее выборки S.

sampleDispersion(S) вычисляет выборочную дисперсию выборки S.

covarianceCoefficient$(S1, S2)$ вычисляет коэффициент ковариации для 2 выборок $S1$ и $S2$.

correlationCoefficient$(S1, S2)$ вычисляет коэффициент корреляции для 2 выборок $S1$ и $S2$.

Пример.

```
SPACE=R64[x];
S1=[0, 1];
S2=[1, 2];
g=\sampleMean(S1);
g1=\sampleDispersion(S1);
```

```
g2=\covarianceCoefficient(S1, S2);
g3=\correlationCoefficient(S1, S2);
\print(g, g1, g2, g3);
```

Результат выполнения:
in: $SPACE = R64[x]$;
$S1 = [0, 1]$;
$S2 = [1, 2]$;
$g = sampleMean(S1)$;
$g1 = sampleDispersion(S1)$;
$g2 = covarianceCoefficient(S1, S2)$;
$g3 = correlationCoefficient(S1, S2)$;
$print(g, g1, g2, g3)$;
out: $g = 0.5$;
$g1 = 0.25$;
$g2 = 0.25$;
$g3 = 1.00$.

10.4. Контрольные задания

Заданы дискретные случайные величины M и K

M					K				
1	1. 2	1. 4	1. 6	1. 8	0. 9	1. 0	1. 1	1. 2	1. 3
0. 1	0. 1	0. 3	0. 4	0. 1	0. 2	0. 3	0. 2	0. 2	0. 1

В Mathpar найдите

- математическое ожидание случайных величин M и K,
- дисперсию случайных величин M и K,
- среднее квадратичное отклонение случайных величин M и K,
- сумму, произведение случайных величин M и K,
- коэффициент ковариации случайных величин M и K,
- коэффициент корреляции случайных величин M и K.
- Постройте многоугольник распределения дискретной случайной величины M и ее функции распределения.

Глава 11

Операторы управления. Процедурное программирование

11.1. Процедуры и функции

Система Mathpar позволяет создавать свои процедуры и функции. Для этого используется команда **procedure**. После команды указывается имя процедуры и в фигурных скобках описывается сама процедура.

Пример.

```
\procedure myProc2() {
  d = 4;
  \print(d);
}
\procedure myProc(c,  d) {
  if (c < d) {
    \return d;
  } else {
    \return d+5;
  }
}
\myProc2();
```

```
a = 10;
c = \myProc(5 + a, a);
\print(a, c);
```

Результат выполнения:
$d = 4;\ a = 10;\ c = 15.$

11.2. Операторы ветвления и циклов

Система Mathpar дает возможность использовать операторы ветвления и циклов.

if() { } **else** { } — оператор ветвления;
while() { } — оператор цикла с предусловием;
for(; ;) { } — оператор цикла с счетчиком.

Примеры.

```
a = 5; b = 1;
if (b < a) {
  b = b + a;
} else {
  \print(a, b);
}
if (b < a) {
  b = b + a;
} else {
  \print(a, b);
}
```

Результат выполнения:
$a = 5;\ b = 6;$

```
a = 0;
b = 10;
while (a < b) {
  a = a + 5;
  \print(a);
}
```

Результат выполнения:
$a = 5;\ a = 10;$

```
for (i = 3; i \le 11; i = i + 5) {
  \print(i);
}
```

Результат выполнения:
$i = 3$; $i = 8$.

11.3. Контрольные задания

В Mathpar напишите программу:

- для поиска наибольшего коэффициента матрицы,
- для вывода всех чисел от 1 до 3000, которые делятся на 252, а при делении на 101 дают в остатке 3.

Глава 12

Вычисления в идемпотентных алгебрах

12.1. Тропические алгебры

Определены следующие тропические алгебры :

ПОЛУПОЛЯ
1) На множестве целых чисел Z определены:
$ZMaxPlus$, $ZMinPlus$.
2) На множестве чисел R определены:
$RMaxPlus$, $RMinPlus$, $RMaxMult$, $RMinMult$.
3) На множестве чисел R64 определены:
$R64MaxPlus$, $R64MinPlus$, $R64MaxMult$, $R64MinMult$.

ПОЛУКОЛЬЦА
1) На множестве целых чисел Z определены:
$ZMaxMin$, $ZMinMax$, $ZMaxMult$, $ZMinMult$.
2) На множестве чисел R определены:
$RMaxMin$, $RMinMax$.
3) На множестве чисел R64 определены:
$R64MaxMin$, $R64MinMax$.

Примеры тропических алгебр:
SPACE = ZMaxPlus [x, y, z];
SPACE = R64MinMult [u, v];

SPACE = RMaxMin [u, v].

Пример простой задачи в полукольце $ZMaxMult$.

Пример 1.

```
SPACE = ZMaxMult[x, y];
a = 2; b = 9; c = a + b; d = a*b; \print(c, d)
```

Результат выполнения:

$c = 9$;

$d = 18$.

Помимо сложения и умножения доступна операция замыкания, вызываемая командой $\backslash closure(a)$, где a — элемент или матрица. Замыкание $closure(a) = 1 \oplus a \oplus a^2 \oplus \ldots$

Пример 2.

```
SPACE = R64MaxPlus[x, y];
A = [
  [0.0, -0.6, -0.62, -3.76],
[-6.29, 0.0, -0.99, -7.61],
[-2.74, -0.86, 0.0, -3.47],
[-6.31, -5.11, -2.69, 0.0]
];
B = \closure(A);
\print(B);
```

Результат выполнения:

$$B = \begin{pmatrix} 0.0 & -0.6 & -0.62 & -3.76 \\ -3.74 & 0.0 & -0.99 & -4.46 \\ -2.74 & -0.86 & 0.0 & -3.47 \\ -5.43 & -3.55 & -2.69 & 0.0 \end{pmatrix}$$

В остальных параграфах этой главы приведены примеры задач, которые решаются в тропических алгебрах, являющихся полуполями.

12.2. Решение систем линейных алгебраических уравнений

Команда $\backslash solveLAETropic(A, b)$ позволяет найти частное решение уравнения вида $Ax = b$.

Пример 3.

```
SPACE = R64MaxPlus[x, y];
A = [
  [1.00, 1.00, 0.00],
  [2.00, 0.00, 3.00],
  [3.00, 4.00, 2.00]
];
b = [8.00, 7.00, 11.00];
X = \solveLAETropic(A, b);
\print(X);
```

Результат выполнения:

$$X = \begin{pmatrix} 5.00 \\ 7.00 \\ 4.00 \end{pmatrix}$$

12.3. Решение систем линейных алгебраических неравенств

Команда $\backslash solveLAITropic(A, b)$ позволяет найти решение неравенства Ax $\leqslant$ b.

Пример 4.

```
SPACE = R64MaxPlus[x, y];
A = [
  [1.00, 1.00, 0.00],
  [2.00, 0.00, 3.00],
  [3.00, 4.00, 2.00]
];
b = [10.00, 7.00, 11.00];
X = \solveLAITropic(A, b);
\print(X);
```

Результат выполнения:

$X = [(-\infty, 5.00], (-\infty, 7.00], (-\infty, 4.00]]$

Пример 5.

```
SPACE = ZMinPlus[x, y];
A = [
  [1, 1, 0],
  [2, 0, 3],
```

```
  [3, 4, 2]
];
b = [10, 7, 11];
X = \solveLAITropic(A, b);
\print(X);
```

Результат выполнения:
$X = [[9, \infty), [9, \infty), [10, \infty)]$

12.4. Решение уравнения Беллмана

12.4.1. Однородное уравнение Беллмана

Команда $\backslash BellmanEquation(A)$ позволяет найти решение однородного уравнения Беллмана $Ax = x$.

Пример 6.

```
SPACE = R64MaxPlus[x, y]; TIMEOUT=16;
A = [
  [0.00, -2.00, -\infty, -\infty],
  [-\infty, 0.00, 3.00, -1.00],
  [-1.00, -\infty, 0.00, -4.00],
  [2.00, -\infty, -\infty, 0.00]
];
X = \BellmanEquation(A);
\print(X);
```

Результат выполнения:

$$X = \begin{pmatrix} 0.00 & -2.00 & 1.00 & -3.00 \\ 2.00 & 0.00 & 3.00 & -1.00 \\ -1.00 & -3.00 & 0.00 & -4.00 \\ 2.00 & 0.00 & 3.00 & 0.00 \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{pmatrix}, \forall v_1, v_2, v_3, v_4.$$

12.4.2. Неоднородное уравнение Беллмана

Команда $\backslash BellmanEquation(A, b)$ позволяет найти решение неоднородного уравнения Беллмана $Ax \oplus b = x$.

Пример 7.

```
SPACE = R64MaxPlus[x, y]; TIMEOUT=16;
A = [
  [0.00, -2.00, -\infty, -\infty],
  [-\infty, 0.00, 3.00, -1.00],
  [-1.00, -\infty, 0.00, -4.00],
  [2.00, -\infty, -\infty, 0.00]
];
b = [[1], [-\infty], [-\infty], [-\infty]];
X = \BellmanEquation(A, b);
\print(X);
```

Результат выполнения:

$$X = \begin{pmatrix} 0.00 & -2.00 & 1.00 & -3.00 \\ 2.00 & 0.00 & 3.00 & -1.00 \\ -1.00 & -3.00 & 0.00 & -4.00 \\ 2.00 & 0.00 & 3.00 & 0.00 \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{pmatrix} \oplus \begin{pmatrix} 1.00 \\ 3.00 \\ 0.00 \\ 3.00 \end{pmatrix}, \forall v_1, v_2, v_3, v_4.$$

12.5. Решение неравенства Беллмана

12.5.1. Однородное неравенство Беллмана

Команда $\backslash BellmanInequality(A)$ позволяет найти решение однородного неравенства Беллмана $Ax \leqslant x$.

12.5.2. Неоднородное неравенство Беллмана

Команда $\backslash BellmanInequality(A, b)$ позволяет найти решение неоднородного неравенства Беллмана $Ax \oplus b \leqslant x$.

12.6. Нахождение кратчайшего пути между вершинами графа

12.6.1. Вычисление таблицы кратчайших расстояний для всех вершин графа

Пусть A - матрица расстояний между смежными вершинами (x_{ii}=0 $\forall i$; $x_{ij} = \infty$, если нет ребра, соединяющего вершины i и j). Команда

$\backslash searchLeastDistances(A)$ позволяет найти наименьшие расстояния между всеми вершинами графа. В результате будет получена матрица кратчайших расстояний между вершинами.

Пример 8.

```
SPACE = R64MinPlus[x, y]; TIMEOUT=16;
A = [
  [0.00, 7.00, 9.00, \infty, \infty, 14.00],
  [7.00, 0.00, 10.00, 15.00, \infty, \infty],
  [9.00, 10.00, 0.00, 11.00, \infty, 2.00],
  [\infty, 15.00, 11.00, 0.00, 6.00, \infty],
  [\infty, \infty, \infty, 6.00, 0.00, 9.00],
  [14.00, \infty, 2.00, \infty, 9.00, 0.00]
];
B = \searchLeastDistances(A);
\print(B);
```

Результат выполнения:

$$B = \begin{pmatrix} 0.00 & 7.00 & 9.00 & 20.00 & 20.00 & 11.00 \\ 7.00 & 0.00 & 10.00 & 15.00 & 21.00 & 12.00 \\ 9.00 & 10.00 & 0.00 & 11.00 & 11.00 & 2.00 \\ 20.00 & 15.00 & 11.00 & 0.00 & 6.00 & 13.00 \\ 20.00 & 21.00 & 11.00 & 6.00 & 0.00 & 9.00 \\ 11.00 & 12.00 & 2.00 & 13.00 & 9.00 & 0.00 \end{pmatrix}$$

12.6.2. Нахождение кратчайшего пути между двумя вершинами графа

Пусть A - матрица расстояний между смежными вершинами (x_{ii}=0 $\forall i$; $x_{ij} = \infty$, если нет ребра, соединяющего вершины i и j). Команда $\backslash findTheShortestPath(A, i, j)$ позволяет найти кратчайший путь между вершинами i и j.

Пример 9.

```
SPACE = R64MinPlus[x, y]; TIMEOUT=16;
A = [
  [0.00, 7.00, 9.00, \infty, \infty, 14.00],
  [7.00, 0.00, 10.00, 15.00, \infty, \infty],
  [9.00, 10.00, 0.00, 11.00, \infty, 2.00],
```

```
  [\infty, 15.00, 11.00, 0.00, 6.00, \infty],
  [\infty, \infty, \infty, 6.00, 0.00, 9.00],
  [14.00, \infty, 2.00, \infty, 9.00, 0.00]
];
X = \findTheShortestPath(A, 0, 4);
\print(X);
```

Результат выполнения:
$X = [[0, 2, 5, 4]]$

Глава 13

Вычисления на суперкомпьютере

Для решения вычислительных задач, требующих большого времени вычислений или больших объемов памяти, разработаны специальные функции, которые предоставляют пользователю ресурсы суперкомпьютера. При использовании этих функций вычисления производятся не на одном процессоре, а на выделенном множестве ядер суперкомпьютера, количество которых заказывает пользователь. Имеются следующие функции, которые используют суперкомпьютер (парфункции).

1) **matMultPar1x8** — вычисление произведения двух матриц;

2) **adjointDetPar** — вычисление присоединенной матрицы;

3) **charPolPar** — вычисление характеристического полинома матрицы;

4) **polMultPar** — вычисление произведения двух полиномов;

5) **BellmanEquationPar**A — решение однородного уравнения Беллмана $Ax = x$;

6) **BellmanEquationPar**A, b — решение однородного уравнения Беллмана $Ax + b = x$;

7) **BellmanInequalityPar**A — решение однородного неравенства Беллмана $Ax \leqslant x$;

8) **BellmanInequalityPar**A, b — решение однородного неравенства Беллмана $Ax + b \leqslant x$;

До применения любой из этих функций пользователь должен

указать параметры, определяющие параллельное окружение:

$TOTALNODES$ — общее количество узлов кластера, которые выделяются для вычислений,

$PROCPERNODE$ — количество MPI-процессов, запускаемых на одном узле,

$CLUSTERTIME$ — максимальное время (в минутах) выполнения программы, после истечения которого программа принудительно завершится,

$MAXCLUSTERMEMORY$ — объем памяти, выделяемый для JVM для одного MPI-процесса (опция -Xmx).

Для задания количества ядер на одном узле пользователь должен знать, какой кластер используется и сколько доступно ему ядер на узле. По умолчанию параметры $TOTALNODES$ и $PROCPERNODE$ устанавливаются так, чтобы использовалась половина всех узлов кластера и на каждом узле было запущено по одному процессу, а $CLUSTERTIME$ двум минутам. Если на одном узле запускается K процессов, то каждому из них будет выделено $MAXCLUSTERMEMORY/K$ мегабайт памяти.

13.1. Параллельные полиномиальные вычисления

Для параллельного вычисления произведения полиномов надо использовать команду $\backslash$**polMultPar**$(p1, p2)$, где $p1$, $p2$ — входные полиномы.

Пример.

```
TOTALNODES = 2;
PROCPERNODE = 1;
A=x^2+3y;
B=x^2+3y+3z;
\polMultPar(A, B);
```

13.2. Параллельные матричные вычисления

Для параллельного вычисления произведения матриц $m1$ и $m2$ необходимо использовать команду Пример.

```
TOTALNODES = 2;
PROCPERNODE = 1;
A=[[0,1],[2,3]];
B=[[0,1],[2,3]];
\matMultPar1x8(A, B);
```

Для параллельного вычисления присоединенной матрицы для матрицы m можно использовать команду Пример.

```
TOTALNODES = 2;
PROCPERNODE = 1;
SPACE = Z[x];
A=[[0,1],[2,3]];
\adjointDetPar(A);
```

13.3. Запуск собственных параллельных программ

Mathpar позволяет загружать и исполнять собственные параллельные программы. Пакет с программой должен распологаться в корневой директории проекта mathpar. Для того, чтобы ваша программа смогла взаимодействовать с системой управления заданиями, необходимо в ваш main-метод добавить строку инициализации QueryResult queryRes=Tools.getDataFromClusterRootNode(args) (сразу после MPI.Init) и строку завершения Tools.sendFinishMessage(args) (перед MPI.Finalize), этот код будет одинаков для всех ваших программ). Также вы можете передать вашей программе любые аргументы из web-интерфейса Mathpartner. Внутри программы их можно получить, вызвав метод queryRes.getData(). Ниже приведен пример параллельной программы, которая просто выводит в стандартный поток вывода переданные ей аргументы .

```
MPI.Init(args);
QueryResult queryRes=Tools.getDataFromClusterRootNode(args);
int myRank=MPI.COMM_WORLD.getRank();
if (myRank == 0) {
    Object []ar=queryRes.getData();
```

```
        System.out.println("test...");
        for (int i=0; i<ar.length; i++){
            System.out.println(((Element)ar[i]).intValue());
        }
    }
    Tools.sendFinishMessage(args);
    MPI.Finalize();
```

Далее программу нужно скомпилировать, и папку с программой запаковать в zip-архив. Затем нужно загрузить полученный архив на сервер, воспользовавшись вкладкой "файлы"и нажав кнопку "загрузить файл". Далее вся работа будет выполняться с помощью функций mathpar.

Оперативная память делится между всеми ядрами процессора поровну. Для примера, если на узле кластера имеется 8GB памяти, то если вы запросили 4 ядра на одном процессоре, каждому будет выделено 2GB, а если одно ядро - то оно получит все 8GB.

Команда для загрузки вашего zip-архива, в котором скомпилированные java-классы, выглядит следующим образом:

uploadToCluster($FileName$), где FileName - имя zip-архива.

Чтобы просмотреть список всех ваших загруженных на кластер файлов, используется команда

showFileList().

Для запуска вашей программы используется команда

runUploadedClass($archieveName, classPath, param0, param1, ...$), где archieveName - имя загруженного zip-архива с программой, classPath - путь до класса, содержащего main-метод (с указанием пакетов), paramX - произвольные параметры, указанные через запятую, которые будут переданы в вашу программу.

Чтобы следить за работой запущенной программы, используется команда

getStatus($taskID$)

Также имеется возможность получить список всех задач текущего пользователя с описанием их состояний:

showTaskList()

Для того, чтобы получить содержимое файлов с потоком стандартного вывода/ошибок, используются команды

getOut($taskID$)

getErr(*taskID*)

Файлы задачи (файлы, содержащие поток вывода/ошибок) хранятся на кластере двое суток, zip-архивы, содержащие скомпилированные java-классы, хранятся 30 дней.

Глава 14

Список операторов

Правило образования наименований математических объектов

Заглавные и строчные буквы всюду различаются. Пользователь может давать любые имена для математических объектов. Однако эти имена не должны совпадать с операторами и константами, которые определены в системе. Кроме того, имена объектов, умножение которых не коммутативно, например, векторов и матриц, должны начинаться с заглавных латинских букв, а все остальные имена объектов должны начинаться со строчных букв. Это дает возможность сразу после ввода автоматически получать упрощенное выражение.

Приведем список основных операторов системы Mathpar.

clean — удаление всех введенных имен объектов. Если в операторе перечислены имена объектов, то удаляются только объекты с этими именами.

Инфиксные арифметические операторы

$+$ — сложение;

- — вычитание;

/ — деление;

***** — умножение (можно использовать пробел вместо знака умножения);

times — некоммутативное умножение.

Постфиксные арифметические операторы
! — факториал;
$x\hat{}\{\}$ — возведение в степень;

Инфиксные операторы сравнения.
le — меньше или равно;
$>$ — больше;
$<$ — меньше;
ge — больше или равно;
$==$ — равно;
ne — неравно.

Инфиксные логические операторы
lor — дизъюнкция;
\& — конъюнкция;
neg — отрицание.
Основные префиксные операторы
d — символ дифференцирования при записи дифференциального уравнения;

D — производная функции: **D**(f) и **D**(f, x) — первая производная по x; **D**$(f, y\hat{}3)$ — третья производная по y и т. д.;

expand — преобразование выражения в сумму с раскрытием всех скобок в выражении;

fullExpand — преобразование в сумму выражения, которое содержит логарифмические, показательные и тригонометрические функции;

extendedGCD — расширенный алгоритм вычисления наибольшего общего делителя (НОД) полиномов. В результате получается вектор, содержащий НОД и дополнительные множители аргументов;

GCD — вычисление НОД полиномов;

factor — представление выражения в виде произведения;

fullFactor — представление выражения, содержащего логарифмические и показательные функции, в виде произведения;

initCond — задание начальных условий для системы линейных дифференциальных уравнений;

LCM — вычисление наименьшего общего кратного (НОК) полиномов;

lim — предел выражения;

print — печать выражений. Аргументами выступают имена выражений, разделенные запятыми. Каждое выражение будет печататься на новой строке;

printS — печать выражений в одну строчку, для перехода на следующую строчку нужно использовать «\n»;

plot — построение графика функции, которая задана явно;

plot3D — построение графика функции двух переменных, которая задана явно;

paramPlot — построение графика функции, которая задана параметрически;

tablePlot — построение графика функции, заданной таблицей аргументов и значений;

prod — произведение (символ $\prod$);

randomPolynom — генерация случайного полинома;

randomMatrix — генерация случайной матрицы;

randomNumber — генерация случайного числа;

sequence — задание последовательности;

showPlots — построение в одной системе координат графиков функций, которые должны быть определены раньше;

solveLDE — решение систем линейных дифференциальных уравнений;

systLAE — задание систем линейных алгебраических уравнений;

systLDE — задание систем линейных дифференциальных уравнений;

sum — сумма (символ $\sum$);

time — определение процессорного времени в миллисекундах;

value — вычисление значение выражения при подстановке заданных выражений или чисел вместо переменных кольца.

Операторы процедуры, ветвления и цикла

procedure — оператор объявления процедуры;

$if(\ )\{\ \}else\{\ \}$ — оператор ветвления;

$while(\ )\{\ \}$ — оператор цикла с предусловием;

$for(\ ;\ ;\ )\{\ \}$ — оператор цикла со счетчиком.

Матрицы, их элементы и матричные операторы

[,] — задание вектора (строки);
[[,], [,]] — задание матрицы;
A_{i,j} — (i,j)-элемент матрицы A;
A_{i,?} — строка i матрицы A;
A_{?,j} — столбец j матрицы A;
\O_{n,m} — нулевая матрица размера $n \times m$;
\I_{n,m} — $n \times m$ матрица с единицами на главной диагонали;
+, -, * — сложение, вычитание, умножение;
rowNumb() — число строк матрицы (или вектора);
colNumb() — число столбцов матрицы (или вектора);
size() — оба размера матрицы (или число компонент вектора);
charPolynom() — характеристический полином;
kernel() — ядро оператора (нуль-пространство);
transpose() или **A**^{T} — транспонированная матрица;
conjugate() или **A**^{\ ast} — сопряженная матрица;
toEchelonForm() — эшелонная (ступенчатая) форма;
det() — определитель;
rank()— ранга матрицы;
inverse() или **A**^{-1} — обратная матрица;
adjoint() или **A**^{\ star} — присоединенная матрица;
genInverse() или **A** ^ {+} — обобщенная обратная матрица Мурра-Пенроуза;
closure или **A**^{\ times} — замыкание, т.е. сумма $I + A + A^2 + A^3 + \ldots$. Для классических алгебр это эквивалентно $(I - A)^{-1}$;
pseudoInverse() и — псевдо-обратная матрица. Она в отличие от матрицы Мурра-Пенроуза, удовлетворяет только двум из четырех тождеств. Однако она быстрее вычисляется;
LSU() — LSU-разложение матрицы. Результат — вектор из трёх матриц $[L, S, U]$. Здесь L — нижняя треугольная матрица, U — верхняя треугольная матрица, S — матрица перестановок, умноженная на обратную к диагональной матрицу.
LSUWMdet() — Результат — вектор из 6 матриц $[L, S, U, W, M, [[det]]]$. A=LSU, $pseudoInverse(A) = (1/det^2)WSM$, det - ненулевой максимальный по размеру угловой минор.
BruhatDecomposition() — разложение Брюа матрицы. Результат — вектор из трёх матриц $[V, D, U]$. Здесь V и U — верхние треугольные матрицы, D — матрица перестановок, умноженная на матрицу, которая является обратной к диагональной матрице.

SVD() — SVD-разложение матрицы над действительными числами. Результат — вектор из трёх матриц $[U, D, V^T]$. Здесь U, V^T — ортогональные матрицы, D — диагональная матрица.

QR() — QR-разложение матрицы над действительными числами. Результат — вектор из двух матриц $[Q, R]$. Здесь Q — ортогональная матрица, R — верхняя треугольная матрица.

sylvester($p1, p2, type = 0 or 1$) — строится матрица Сильвестра по коэффициентам полиномов $p1, p2$. Кольцо Z[x,y,z,u] будет рассматриваться как кольцо Z[u][x,y,z] (кольцо от одной переменной u с коэффициентами из Z[x,y,z].) Если type=0 размер матрицы n1+n2, если type=1 размер матрицы 2 max(n1,n2).

cholesky(A) или **cholesky**($A, 0$) — Разложение Холецкого. Матрица A должна быть симметричной и положительно определенной, только в этом случае разложение будет правильно вычислено. **cholesky**($A, 1$) можно использовать в случае больших плотных матриц, начиная с размера 100x100. Здесь мы использовали блочное умножение по алгоритму Винограда-Штрассена.

Глава 15

Примеры решения задач по физике

15.1. Передача тепла

```
"ЗАДАЧА 1"
"Кусок льда массой"
M = 10 кг;
"помещен в сосуд. Температура льда"
T = -10  \degreeC ;
"Найдите массу воды в сосуде после того, как сосуду
сообщили количество тепла равное"
q = 20000 кДж;
"Удельная теплоемкость воды"
c_v = 4.2 кДж/(кг \degreeC);
"Удельная теплоемкость льда"
c_i = 2.1 кДж/(кг \degreeC);
"Удельная теплота плавления льда"
r = 330 кДж/кг;
"Удельная теплота испарения воды"
\lambda = 2300 кДж/кг;
END


"РЕШЕНИЕ ЗАДАЧИ 1"
```

```
"Искомую массу воды обозначим через x."
SPACE = R64[x];
"Обозначим количество теплоты требуемое для нагревания льда до 0 град
q_1 = M c_i (0 - T);
"для плавления всего льда:"
q_2 = M r;
"для нагревания воды до ста градусов:"
q_3 = M c_v (100 \degreeC);
"для испарения части воды"
q_4 = (M - x)\lambda;
"Здесь мы обозначили через x массу оствшейся в сосуде воды."
"По условию задачи должно выполняться равенство,
решая которое найдем неизвестное x:"
mass  = \solve (q = q_1 + q_2 + q_3 + q_4);
mass=\value(mass);
\print(mass);
```

15.2. Кинематика

```
"ЗАДАЧА 2"
"Кинематическое уравнение движения точки по прямой (по оси x)
имеет вид $x = c_1 + c_2 t + c_3 t^3$."
"Найдите: (1) координату точки, (2) мгновенную скорость,
(3) мгновенное ускорение"
END


"РЕШЕНИЕ ЗАДАЧИ 2."
"Выбираем пространство с переменными $t, c_1, c_2, c_3$:"
SPACE = R64[t, c_1, c_2, c_3];
"Уравнение движения точки"
x = c_1 + c_2  t + c_3 t^3;
"Вычислим мгновенную сросты"
v = \D_t(x);
"Вычислим мгновенное ускорение"
a = \D_t(v);
\print(x, v, a);
```

```
"ЗАДАЧА 2А"
"Решите предыдущую задачу, при условии, что  "
"коэффициенты c1, c2, c3 в уравнении имеют следующие значения"
Coeff = [4, 2, -0.5];
"и момент времени равен "
t_0 = 2 "секунды."
END


"РЕШЕНИЕ ЗАДАЧИ 2А"
"Введем обозначение для элементов вектора  Coeff:"
cf=\elementOf(Coeff);
"Найдем числовое значение каждой функции (x, v, a)
в точке"
arg = [t_0, cf_{1}, cf_{2}, cf_{3}];
"(1) координата точки в момент времени $t_0$:"
x_0 = \value (x, arg);
"(2) мгновенная скорость точки в момент времени $t_0$:"
v_0 = \value (v, arg);
"(3) мгновенное ускорение точки в момент времени $t_0$:"
a_0 = \value (a, arg);
\print(x_0, v_0, a_0);
```

15.3. Молекулярная физика

```
"ЗАДАЧА 3"
"В центре горизонтальной трубки расположен столбик ртути длиной h"
"Часть воздуха была выкачана и концы трубки запаяны.
Трубка имеет длину l"
"Когда трубка была поставлена вертикально, столбик ртути переместился
"Ускорение свободного падения обозначим $g$, плотность ртути - $\rho$
"Какое начальное давление было в трубке?"
END
```

```
"РЕШЕНИЕ ЗАДАЧИ 3"
"Пусть в трубке было начальное давление $p_0$. Введем пространство с
SPACE=R64[p_0];
"После поворота трубки давление в нижней части трубки повысилось,
так как добавилось давление столбика ртути. Следовательно, новое давл
p_1 = p_0+\rho g h;
"Пусть s это площадь поперечного сечения трубки.
Тогда начальный обьем нижней части трубки равен:"
v_0=  (l/2-h/2) s;
" После поворота трубки обьем воздуха в нижней части трубки будет  ра
v_1=  (l/2-h/2-l_d) s;
"В соответствии с законом Бойля-Мариотта запишем и решим
уравнение относительно неизвестной $p_0$:"
initialPressure = \solve(p_0 v_0=p_1 v_1 );
\print(initialPressure );




"ЗАДАЧА 3А."
"Решите предыдущую задачу предполагая, что переменные
имеют следующие числовые значения: "
h = 0.20 m;
l = 1 m;
l_d = 0.10 m;
g = 9.8 m/s^2;
\rho = 13600 kg/m^3;
END




"РЕШЕНИЕ ЗАДАЧИ 3А"
p_1 = p_0 + \rho g h;
v_0 = (l/2 - h/2) S;
v_1 = (l/2 - h/2 - l_d) S;
\solve(p_0 v_0 = p_1 v_1);
```

15.4. Маятник

```
"ЗАДАЧА 4. Период математического маятника."
SPACE = R64[x]; FLOATPOS = 4
"Материальная точка подвешена на невесомой нити. Длина маятника равна
Она колеблется в поле силы тяжести с ускорением свободного падения $g
Максимальный угол отклонения маятника от вертикали, называемый амплит
\theta_0=  (2/3) \pi;
"Надо найти период $T$ маятника, используя арифметико-геометрическое
$$T=\frac{2\pi}{\AGM(1,\cos(\theta_0/2))}\sqrt{\frac{L}{g}}$$"
END

"РЕШЕНИЕ ЗАДАЧИ 4."
\theta_0=  (2/3) \pi;
w=\value(\cos(\theta_0/2));
 Ts = 2*\pi*\sqrt{L/g}/(\AGM(1,w)); \print(w,Ts);
L = 1;
g = 9.80665;
T= \value(Ts); \print(T);
```

The results:
$w = 0.5$
$Ts = 2.7458 * \pi * (L/g)^{(1/2)}$
$T = 2.7546$.