Введение

Тема: Использование системы компьютерной алгебры при работе с выражениями и уравнениями.

Цель:

1. Познакомиться с основными командами системы компьютерной алгебры Maxima для работы с выражениями и уравнениями:
   - Упрощение алгебраических выражений.
   - Раскрытие скобок и приведение подобных слагаемых в выражениях.
   - Разложение алгебраического выражения на множители.
   - Разложение рациональной дроби на простейшие дроби.
   - Решение уравнений.

Примечания:

1. Каждое задание лабораторной работы надо выполнять в отдельном файле.
2. Формат имени файла: "ФИО студента, номер группы/подгруппы, тема 5, ЛР (задание …).wxmx".

Требования к отчету по работе:

1. Прикрепить файлы, созданные в программе Maxima, в Moodle.
2. Выложить отчёт с кратким описанием выполненных заданий на сайт со своим портфолио.

Ход лабораторной работы

Задание 5.0

- Откройте новый файл.
- Сохраните файл. Формат имени файла: "ФИО студента, номер группы/подгруппы, тема 5, ЛР (задание 5.0).wxmx"

  Рассмотрим понятие «флага».

- Добавьте заголовок: Работа с выражениями. Введение
- Итог:

  □ **Работа с выражениями. Введение**

Флаг «numer»

- Добавьте раздел: Флаг «numer»
- Итог:

  □ ***1 Флаг «numer»***

В программе Maxima имеются два режима работы с подстановкой чисел в выражения. В одном из них число только подставляется в выражение, во втором происходит вычисление. В исходном состоянии программа находится во втором режиме, и при написанных выше действиях получится выражение, а не его значение.

Флаг «**numer**» – это флаг численных вычислений. Он влияет на представление чисел, которыми вы оперируете.

- ✓ Если выбрано значение этого флага по умолчанию (значение false), то при делении одного целого числа на другое целое число будет получена обыкновенная дробь.
- ✓ Если переключить значение этого флага (в значение true), то результат будет показан в виде десятичной дроби с точкой.

- Выполните действия и проанализируйте результаты.

```
(%i1)   3/4;
(%o1)   3/4
```

```
(%i2)   3.0/4;
(%o2)   0.75
```

При вводе числа в привычном виде (без плавающей точки) получили дробь. При вводе числа с плавающей точкой – получили значение выражения.

Так как не удобно каждое число вручную преобразовывать в формат с плавающей точкой, то можно, во-первых, воспользоваться флагом «numer».

```
(%i3)   3/4,numer;
(%o3)   0.75
```

Примечание: также можно использовать **функцию float** или одноимённый **флаг float**.

```
(%i4)   3/4,float;
(%o4)   0.75
```

```
(%i5)   float(3/4);
(%o5)   0.75
```

Результат будет одинаковым.

Использование рассмотренных флагов не зависит от правильности дроби

- Выполните действия и устно проанализируйте результат.

```
(%i6)   7/4;
(%o6)   7
        -
        4
```

```
(%i7)   7.0/4;
(%o7)   1.75
```

```
(%i8)   7/4,numer;
(%o8)   1.75
```

```
(%i9)   7/4,float;
(%o9)   1.75
```

```
(%i10)  float(7/4);
(%o10)  1.75
```

Значение флага можно переключать, написав команду. Это можно сделать двумя способами:

- ✓ 1 способ. «Вручную» написать команду.
- ✓ 2 способ. Выполнить в главном меню «Численные расчёты» команду «Toggle Numeric Output» (или «Переключить флаг numer», или «Переключить численное вычисление»).

- Выполните переключение флага при помощи команды «Toggle Numeric Output» главного меню «Численные расчёты».

```
(%i11)  if numer#false then numer:false else numer:true;
(%o11)  true
```

Значение флага numer стало равно true. Значит для всех выражений будет происходить вычисление.

- Выполните действия и проверьте полученные результаты.

```
(%i12)  3/4;
(%o12)  0.75
```

```
(%i13)  7/4;
(%o13)  1.75
```

Если выполнить команду «Toggle Numeric Output» ещё раз, то будет выполнено обратное переключение.

- Выполните действия и проверьте полученные результаты.

```
(%i14)  if numer#false then numer:false else numer:true;
(%o14)  false
```

```
(%i15)  3/4;
(%o15)  3/4
```

```
(%i16)  7/4;
(%o16)  7/4
```

Приведении конечной десятичной записи чисел к рациональной. Флаг «keepfloat»

- Добавьте раздел: Приведение конечной десятичной записи чисел к рациональной. Флаг «keepfloat»
- Итог:

□ ***2 Приведение конечной десятичной записи чисел к рациональной. Флаг «keepfloat»***

Функция rat(число) приводит число, записанное в виде конечной десятичной дроби, к рациональному числу, записанному обыкновенной дробью.

- Выполните действия и проверьте результат. Фразы добавлены при помощи «примечаний».

```
Рассмотрим дробь

-->     31/64;
(%o17)  31/64

Вычислим дробь

-->     31/64,numer;
(%o18)  0.484375

Приведём десятичную дробь к рациональной форме

-->     rat(%);
rat: replaced 0.484375 by 31/64 = 0.484375
(%o19)/R/   31/64
```

Конечная десятичная запись считается по определению приблизительной, так как при вычислениях самой Maxima такая запись может возникнуть исключительно при применении приближенных методов либо при ручном указании о переводе числа в десятичную запись из математической, в результате чего результат тоже, вероятнее всего, окажется приблизительным. Эта приблизительность учитывается и при переводе в рациональные числа, а ее уровень, то есть мера, на которую рациональное число при переводе может отклониться от конечной десятичной записи, регулируется переменной **ratepsilon**, равной по умолчанию 2.0e-8, т.е. 0.00000002.

Если необходимо, можно оставлять десятичную запись чисел как есть, установив в true значение флага keepfloat (по умолчанию он равен false).

- Добавьте подраздел «2.1 Пример 1.». Итог:

□ **2.1 Пример 1.**

- Выполните следующие действия по порядку.

    1) Узнаем текущее значение переменной ratepsilon.
    2) Преобразуем 31/64 к канонической форме, применяя функцию rat.
    3) Преобразуем 0.484375 к канонической форме, применяя функцию rat.
    4) Переключим флаг keepfloat в значение true.
    5) Применим функцию rat к 31/64.
    6) Применим функцию rat к 0.484375.
    7) Переключим флаг keepfloat в значение false.

- Проверьте свои команды и полученный результат.

```
(%i20) ratepsilon;
(%o20) 2.0 10^-15

(%i21) rat(31/64);
(%o21)/R/ 31/64

(%i22) rat(0.484375);
rat: replaced 0.484375 by 31/64 = 0.484375
(%o22)/R/ 31/64

(%i23) if keepfloat#false then keepfloat:false else keepfloat:true;
(%o23) true

(%i24) rat(31/64);
(%o24)/R/ 31/64

(%i25) rat(0.484375);
(%o25)/R/ 0.484375

(%i26) if keepfloat#false then keepfloat:false else keepfloat:true;
(%o26) false
```

- Обратите внимание, как работает переключение флага. Для этого сравните между собой результаты 3 и 6 действий.

- Добавьте подраздел «2.2 Пример 2.».
- Выполните следующие действия по порядку.

    1) Изменим текущее значение переменной ratepsilon на 2e-3.
    2) Преобразуем 31/64 к канонической форме, применяя функцию rat.
    3) Преобразуем 0.484375 к канонической форме, применяя функцию rat.
    4) Переключим флаг keepfloat в значение true.
    5) Применим функцию rat к 31/64.
    6) Применим функцию rat к 0.484375.
    7) Переключим флаг keepfloat в значение false.

- Проверьте свои команды и полученный результат.

```
(%i27)  ratepsilon:2e-3;
(ratepsilon)  0.002

(%i28)  rat(31/64);
(%o28)/R/  31
           --
           64

(%i29)  rat(0.484375);
rat: replaced 0.484375 by 15/31 = 0.4838709677419355
(%o29)/R/  15
           --
           31

(%i30)  if keepfloat#false then keepfloat:false else keepfloat:true;
(%o30)  true

(%i31)  rat(31/64);
(%o31)/R/  31
           --
           64

(%i32)  rat(0.484375);
(%o32)/R/  0.484375

(%i33)  if keepfloat#false then keepfloat:false else keepfloat:true;
(%o33)  false
```

- Добавьте подраздел «2.3 Пример 3.».
- Выполните следующие действия по порядку.

  1) Изменим текущее значение переменной ratepsilon на 2e-2.
  2) Преобразуем 31/64 к канонической форме, применяя функцию rat.
  3) Преобразуем 0.484375 к канонической форме, применяя функцию rat.
  4) Переключим флаг keepfloat в значение true.
  5) Применим функцию rat к 31/64.
  6) Применим функцию rat к 0.484375.
  7) Переключим флаг keepfloat в значение false.

- Проверьте свои команды и полученный результат.

```
(%i34) ratepsilon:2e-2;
(ratepsilon) 0.02

(%i35) rat(31/64);
(%o35)/R/ 31/64

(%i36) rat(0.484375);
rat: replaced 0.484375 by 15/31 = 0.4838709677419355
(%o36)/R/ 15/31

(%i37) if keepfloat#false then keepfloat:false else keepfloat:true;
(%o37) true

(%i38) rat(31/64);
(%o38)/R/ 31/64

(%i39) rat(0.484375);
(%o39)/R/ 0.484375

(%i40) if keepfloat#false then keepfloat:false else keepfloat:true;
(%o40) false
```

- Добавьте подраздел «2.4 Пример 4.».
- Выполните следующие действия по порядку.

  1) Изменим текущее значение переменной ratepsilon на 2e-1.
  2) Преобразуем 31/64 к канонической форме, применяя функцию rat.
  3) Преобразуем 0.484375 к канонической форме, применяя функцию rat.
  4) Переключим флаг keepfloat в значение true.
  5) Применим функцию rat к 31/64.
  6) Применим функцию rat к 0.484375.
  7) Переключим флаг keepfloat в значение false.

- Проверьте свои команды и полученный результат.

```
(%i41) ratepsilon:2e-1;
(ratepsilon) 0.2
```

```
(%i42) rat(31/64);
(%o42)/R/  31/64
```

```
(%i43) rat(0.484375);
rat: replaced 0.484375 by 1/2 = 0.5
(%o43)/R/  1/2
```

```
(%i44) if keepfloat#false then keepfloat:false else keepfloat:true;
(%o44) true
```

```
(%i45) rat(31/64);
(%o45)/R/  31/64
```

```
(%i46) rat(0.484375);
(%o46)/R/ 0.484375
```

```
(%i47) if keepfloat#false then keepfloat:false else keepfloat:true;
(%o47) false
```

- Сравните между собой результаты третьих действий, полученные в примерах 1 – 4. Результат приближения будет отличаться.

- Выполните аналогичные действия, изменяя значение переменной ratepsilon на
  - ✓ 1e-4.
  - ✓ 1e-3.
  - ✓ 1e-2.
  - ✓ 1e-1.

- Сравните между собой результаты третьих действий, полученные в решённых примерах

Флаги используют и для других функций. Частично мы будем их использовать при изучении дисциплины.

Задание 5.1

- Откройте новый файл.
- Сохраните файл. Формат имени файла: "ФИО студента, номер группы/подгруппы, тема 5, ЛР (задание 5.1).wxmx"

Рассмотрим возможности Maxima по упрощению и прочим преобразованиям выражений. В частности, рассмотрим:

- ✓ автоматическое раскрытие скобок
- ✓ вынесение за скобки,
- ✓ упрощение арифметических действий над некоторыми элементами,
- ✓ упрощение выражений с участием степенных, показательных и логарифмических функций,

✓ обработку тригонометрических выражений.

Большинство рассматриваемых функций предназначено для преобразования рациональных выражений. В математике под рациональным выражением понимают выражение, состоящее только из арифметически операндов и возведения в натуральную степень. Если элементы такого выражения содержат неарифметические или нестепенные функции, то такие элементы с точки зрения рационального выражения считаются атомарными, то есть неделимыми/непреобразуемыми.

Рациональные функции с математической точки зрения рассматриваются как расширение многочленов (полиномов).

В Maxima имена всех функций по обработке рациональных выражений содержат буквосочетание rat (от слова rational).

- Добавьте заголовок: Работа с выражениями.
- Добавьте раздел: Приведение рациональных выражений к канонической форме
- Итог:

Работа с выражениями

1 Приведение рациональных выражений к канонической форме

Функция rat - Преобразование рационального выражения к канонической форме (Canonical Rational Expression, CRE)

Приведение к канонической форме: раскрытие скобок, затем приведение к общему знаменателю, суммирование, сокращение. Также все числа приводятся в конечной десятичной записи к рациональным.

В общем виде: **rat (выражение)**

- Выполните действия и проверьте результат.

```
(%i1)   (x-1)^2/(x^2+x)+1/(x+1)+0.25;
```

$$(\%o1)\quad \frac{(x-1)^2}{x^2+x}+\frac{1}{x+1}+0.25$$

```
(%i2)   rat(%);
rat: replaced 0.25 by 1/4 = 0.25
```

$$(\%o2)/R/\quad \frac{5x^2-3x+4}{4x^2+4x}$$

То есть Maxima выполнила следующие действия:

$$\frac{(x-1)^2}{x^2+x}+\frac{1}{x+1}+0.25=\frac{x^2-2x+1}{x(x+1)}+\frac{1}{x+1}+\frac{1}{4}=\frac{4(x^2-2x+1)}{4x(x+1)}+\frac{4x}{4x(x+1)}+\frac{x(x+1)}{4x(x+1)}$$
$$=\frac{4x^2-8x+4+4x+x^2+x}{4x(x+1)}=\frac{5x^2-3x+4}{4x^2+4x}$$

Важно: атомарные элементы, то есть символы и числа, в канонической форме рационального выражения в Maxima имеют другое внутреннее представление. Надо иметь ввиду, что если каноническая форма рационального выражения используется в других рациональных выражениях, то последние также автоматически приводятся к канонической форме

- Выполните действия и проверьте результат. Обратите внимание, что результат НЕ оставлен двумя дробями, а также приведён к канонической форме.

```
(%i3)    (%)-1/x;
(%o3)/R/   (5 x-7)/(4 x+4)
```

$$\frac{5x-7}{4x+4}$$

Автоматическое приведение к канонической форме удобно, когда необходимо пошагово проделать большое количество рациональных преобразований. Можно 1 раз вызвать rat(), а затем ссылаться на предыдущие ячейки и автоматически получать результат в компактной и удобной к восприятию канонической форме.

Если каноническая форма не нужна, то есть надо оставить общий вид, тогда применяют функцию **ratdisrep(выражение)**. Кроме того, каноническая форма автоматически «отменяется» и в случае любых преобразований, не являющихся рациональными.

- Выполните действия и проверьте результат. Обратите внимание, что в одном случае результат приведён к канонической форме, а в другом – нет.

```
(%i4)    log(exp(%o3))+1/x;
(%o4)
```

$$\frac{5x-7}{4x+4}+\frac{1}{x}$$

```
(%i5)    rat(%);
(%o5)/R/
```

$$\frac{5x^2-3x+4}{4x^2+4x}$$

```
(%i6)    ratdisrep(%o4);
(%o6)
```

$$\frac{5x-7}{4x+4}+\frac{1}{x}$$

- Добавьте раздел: Представление выражения в виде суммы простейших дробей
- Итог:

2 Представление выражения в виде суммы простейших дробей

Если необходимо представить выражение в виде суммы простейших дробей, то эту задачу решает функция **partfrac(выражение, имя переменной)**. В общем виде **выражение** – это то выражение, которое надо преобразовать в сумму простейших дробей. **Имя переменной** – это та переменная, относительно которой реализуется данное преобразование.

- Выполните действия и проверьте результат. Обратите внимание, что можно обращаться к последнему результату или писать само выражение. НО: обязательно надо указывать имя переменной.

```
(%i7) (5*x^2-3*x+4)/(4*x^2+4*x);
```

(%o7) $\frac{5\,x^2-3\,x+4}{4\,x^2+4\,x}$

```
(%i8) partfrac(%,x);
```

(%o8) $-\frac{3}{x+1}+\frac{1}{x}+\frac{5}{4}$

```
(%i9) partfrac((5*x^2-3*x+4)/(4*x^2+4*x),x);
```

(%o9) $-\frac{3}{x+1}+\frac{1}{x}+\frac{5}{4}$

```
(%i10) partfrac((5*x^2-3*x+4)/(4*x^2+4*x));
Maxima encountered a Lisp error:
 invalid number of arguments: 1
Automatically continuing.
To enable the Lisp debugger set *debugger-hook* to nil.
```

- Добавьте раздел: Раскрытие скобок
- Итог:

3 Раскрытие скобок

За раскрытие скобок отвечает функция ratexpand(выражение).
При переводе с английского языка одно из значений слова «expand» и есть «раскрыть скобки».

Чтобы применить функцию, надо сначала набрать выражение, а потом подставить его в функцию. Можно сразу набирать выражение в скобках указанной функции.

Для этой функции также действует опция «keepfloat». Также есть опция «ratdenomdivide». По умолчанию она установлена в true. Это приводит к тому, что каждая дробь, в которой числитель является суммой, распадается на сумму дробей с одинаковым знаменателем.

- Придумайте своё выражение, в котором будет сумма в числителе (лучше брать не только числа, но и буквенные выражения; например, с x, x^2, x^3 и так далее).
- Примените функцию ratexpand.
- Проверьте, что будет показана сумма дробей с одинаковым знаменателем.

Если значение флага «ratdenomdivide» установить в false, то все дроби с одинаковым знаменателем будут объединены в единую дробь.

- Затем придумайте выражение с несколькими дробями с одинаковым знаменателем.
- Установите флаг «ratdenomdivide» в false.
- Примените функцию ratexpand.
- Проверьте, что будет показана единая дробь с одинаковым знаменателем. А числители исходных дробей будут складываться/вычитаться. Такой результат будет очень похож на результат функции rat().

В функции ratexpand() и в числите, и в знаменателе все скобки будут раскрыты. Пользователю не видно, но внутри программы выражение остаётся в общем виде. А значит и результаты дальнейших рациональных действий с выражением не будут автоматически «канонизироваться».

В функции rat(), слагаемые с одинаковыми переменными будут сгруппированы. Причём одна из них будет вынесена за скобки (такая форма записи называется «рекурсивной»/recursive).

- Выполните действия. Сравните результаты функций ratexpand() и rat().

Примечание 1: нумерация команд может не совпадать.
Примечание 2: в современных версия maxima вместо «%e» нужно набирать «e».

(%i30) `((x+1)^2*(%e^x-1))/((x+2)*(2*x+1));`

(%o30) $\dfrac{(x+1)^2(\%e^x-1)}{(x+2)(2x+1)}$

(%i31) `ratexpand(%),ratdenomdivide: false;`

(%o31) $\dfrac{x^2\%e^x+2x\%e^x+\%e^x-x^2-2x-1}{2x^2+5x+2}$

(%i32) `rat(%);`

(%o32) $\dfrac{(x^2+2x+1)\%e^x-x^2-2x-1}{2x^2+5x+2}$

(%i33) `ratexpand(%);`

(%o33) $\dfrac{x^2\%e^x}{2x^2+5x+2}+\dfrac{2x\%e^x}{2x^2+5x+2}+\dfrac{\%e^x}{2x^2+5x+2}-\dfrac{x^2}{2x^2+5x+2}-\dfrac{2x}{2x^2+5x+2}-\dfrac{1}{2x^2+5x+2}$

Кроме функции ratexpand(), есть функция expand(). Первая их них раскрывает только рациональное выражение «верхнего уровня», а все подвыражения, не являющиеся рациональными, обрабатываются как атомарные (то есть она их не раскрывает). Вторая же функция раскрывает скобки на всех уровнях вложенности.

- Выполните действия и проверьте результат. Примечание: нумерация команд может не совпадать.

(%i37) `ratexpand((a+b)^((2-x)*(2+x)+x^2));`

(%o37) $(b+a)^{x^2+(2-x)(x+2)}$

(%i38) `expand((a+b)^((2-x)*(2+x)+x^2));`

(%o38) $b^4+4ab^3+6a^2b^2+4a^3b+a^4$

- Обратите внимание, что первая функция не проанализировала, что показатель степени равен числу 4 (проверьте, раскрыв скобки и приведя подобные слагаемые). Вторая функция и упростила показатель степени, и потом раскрыла скобки в выражении $(a + b)^4$.

  Примечания (сравнение ratexpand() и expand()):
  - Во-первых, функция ratexpand() раскрывает только рациональное выражение «верхнего уровня», а все подвыражения, не являющиеся рациональными, обрабатываются как атомарные (то есть она их не раскрывает). Функция expand() раскрывает скобки на всех уровнях вложенности.
  - Во-вторых, ratexpand() приводит дроби-слагаемые к общему знаменателю. Функция expand() дроби-слагаемые к общему знаменателю НЕ приводит.
  - В-третьих, на функцию ratexpand() действует переключатель ratdenomdivide. На функцию expand() переключатель ratdenomdivide НЕ действует.

- В-четвертых, функция ratexpand() преобразовывает к рациональным числам конечную десятичную запись (в зависимости от флага keepfloat). А функция eaxpand() НЕ преобразовывает к рациональным числам конечную десятичную запись (вне зависимости от флага keepfloat).
- В-пятых, функция expand() имеет несколько вариаций (в виде отдельных функций с похожими названиями *expand*(), которые раскрывают скобки по-разному.

- Добавьте раздел: Собирание дробей с одинаковым знаменателем.
- Итог:

□ *4 Собирание дробей с одинаковым знаменателем*

- Выполните действия и проверьте результат. Примечание: нумерация команд может не совпадать.

```
(%i34) combine(%);
```

(%o34) $\dfrac{x^2 \%e^x + 2x\%e^x + \%e^x - x^2 - 2x - 1}{2x^2 + 5x + 2}$

- Добавьте раздел: Вынесение за скобки, разложение на множители.
- Познакомьтесь с теорией и выполните **все** действия. Примечание: нумерация команд может не совпадать.

Для записи анализируемого выражения в виде произведения сомножителей, то есть максимального вынесения за скобки, используется функция factor().

```
(%i39) factor(x^24-1);
```

(%o39) $(x-1)(x+1)(x^2+1)(x^2-x+1)(x^2+x+1)(x^4+1)(x^4-x^2+1)(x^8-x^4+1)$

Если этой функции передать целое число, то она разложит его на простые множители. Если функции передать рациональное число, то на множители будут разложены его числитель и знаменатель.

```
(%i58) factor(2^100-1);
```

(%o58) $3\,5^3\,11\,31\,41\,101\,251\,601\,1801\,4051\,8101\,268501$

```
(%i59) (3*5^3*11*31*41*101*251*601*1801*4051*8101*268501)/(2^100-1);
```

(%o59) 1

```
(%i60) factor(123456789/987654321);
```

(%o60) $\dfrac{3607\,3803}{17^2\,379721}$

Если многочлен не может быть представлен в виде произведения нескольких сомножителей, его можно попытаться преобразовать в сумму таких произведений. Для этого используется функция factorsum().

Например, в следующем примере используется много переменных «x, y, z, v, u, t, w» и за скобки общий множитель не вынести. То есть функция factor() не справится с поставленной задачей. Функция factorsum() решит задачу и запишет выражение в виде суммы произведений.

Функция factorsum() умеет раскладывать на множители только независимые слагаемые, то есть такие, которые не сожержат одинаковых переменных. Если раскрыть скобки в выражении, содержащем в двух разных местах один и тот же символ, то, так как коэффициенты при этом символе после раскрытия сгруппируются, то factorsum() не сможет понять каким именно образом разгруппировать их обратно.

```
(%i5) factor(4*y*z+4*x*z+y^2+x*y-v*w-u*w+t*w);
(%o5) 4 y z + 4 x z + y^2 + x y - v w - u w + t w

(%i6) factorsum(%);
(%o6) (y + x)(4 z + y) - (v + u - t)w
```

Не смотря на то, что функции factorsum() и factor() не имеют приставки rat, но с выражениями работают как функция ratexpand(). То есть на любой нерациональной функции останавливаются и внутрь не идут.

```
(%i7) factor(log(x^2+2*x+1));
(%o7) log(x^2 + 2 x + 1)

(%i8) log(factor(x^2+2*x+1));
(%o8) 2 log(x + 1)
```

Задание 5.2

- Откройте новый файл.
- Сохраните файл. Формат имени файла: "ФИО студента, номер группы/подгруппы, тема 5, ЛР (задание 5.2).wxmx"

- Добавьте раздел: Упрощение выражений, дополнительные функции.
- Познакомьтесь с теорией и выполните **все** действия. Примечание: нумерация команд может не совпадать.
- Не забывайте добавлять подразделы, названия которых должны соответствовать выполняемым командам (выполняемым действиям).

Функция ratsimp(выражение) упрощает выражение за счёт рациональных преобразований. В отличие от остальных функций по обработку рациональных выражений, работает в том числе «вглубь». То есть иррациональные части выражения не рассматриваются как атомарные, а упрощаются, в том числе и все рациональные элементы внутри них.

```
(%i1) e^((x^3+1)/(x+1));
```

(%o1) $e^{\frac{x^3+1}{x+1}}$

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

(%o2) e^{x^2-x+1}

На ratsimp() действуют те же флаги, что и на rat(), и ratexpand, и keepfloat, и ratfac. Но отличается она от rat() или ratexpand() не только умением работать «в глубину», но и некоторыми дополнительными рациональными преобразованиями, которые не поддерживаются этими двумя функциями.

Сравните результаты полученные при помощи разных функций, упрощающих выражения.

```
(%i3) w:(sqrt((x-a)^3)-(x+a)*sqrt(x-a))/sqrt((x-a)*(x+a));
```

$$(w)\quad \frac{(x-a)^{3/2}-\sqrt{x-a}\,(x+a)}{\sqrt{(x-a)(x+a)}}$$

```
(%i4) rat(w);
```

$$(\%o4)/R/\quad \frac{\sqrt{x-a}^{3}+(-x-a)\sqrt{x-a}}{\sqrt{(x-a)(x+a)}}$$

```
(%i5) ratexpand(w);
```

$$(\%o5)\quad \frac{(x-a)^{3/2}}{\sqrt{(x-a)(x+a)}}-\frac{x\sqrt{x-a}}{\sqrt{(x-a)(x+a)}}-\frac{a\sqrt{x-a}}{\sqrt{(x-a)(x+a)}}$$

```
(%i6) ratsimp(w);
```

$$(\%o6)\quad -\frac{2\,a\sqrt{x-a}}{\sqrt{x^2-a^2}}$$

Кроме непосредственно функции ratsimp(), есть еще и дополнительный переключатель — ratsimpexpons. По умолчанию он установлен в false; если же назначить ему значение true — это приведет к автоматическому упрощению показателей степени.

```
(%i7) x^((a^2+a+1/4)/(2*a+1));
```

$$(\%o7)\quad x^{\frac{a^2+a+\frac{1}{4}}{2\,a+1}}$$

```
(%i8) %,ratsimpexpons:true;
```

$$(\%o8)\quad x^{\frac{2\,a+1}{4}}$$

<u>Примечание</u>:

$$\frac{a^2+a+\frac{1}{4}}{2a+1}=\frac{\frac{4a^2}{4}+\frac{4a}{4}+\frac{1}{4}}{2a+1}=\frac{\frac{4a^2+4a+1}{4}}{2a+1}=\frac{4a^2+4a+1}{4(2a+1)}=\frac{4a^2+4a+1}{8a+4}=$$

$$=[\text{разделим многочлен на многочлен}]=$$

$$=\frac{(8\mathrm{a}+4)(\frac{1}{2}a+\frac{1}{4})}{8\mathrm{a}+4}=\left(\frac{1}{2}a+\frac{1}{4}\right)=\frac{2a+1}{4}$$

Функция ratsimp() – это уже достаточно мощный, и в то же время весьма быстрый, механизм упрощения; но, конечно, не достаточный: ведь те действия, которые можно упростить в разнообразных математических выражениях, не ограничиваются рациональными. Поэтому все же основной плюс этой функции – это скорость.

Также для более серьезных упрощений существует расширенный вариант – fullratsimp(выражение). Эта функция последовательно применяет к переданному выражению функцию ratsimp(), а также некоторые нерациональные преобразования – и повторяет эти действия в цикле до тех пор, пока выражение не перестанет в процессе них изменяться. За счет этого функция работает несколько медленнее, чем ratsimp(), зато дает более надежный результат – к некоторым выражениям, которые она может упростить с ходу, ratsimp() пришлось бы применять несколько раз, а иногда та и вообще не справилась бы с задачей.

Обратите внимание, что при помощи функции ratsimp() результат был получен за 2 действия. А при помощи функции fullratsimp() результат был получен за 1 действие.

```
(%i9)  t:((x^(a/2)-1)^2*(x^(a/2)+1)^2)/(x^a-1);
```

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

```
(%i10) ratsimp(t);
```

(%o10) $\frac{x^{2\,a}-2\,x^a+1}{x^a-1}$

```
(%i11) ratsimp(%);
```

(%o11) x^a-1

```
(%i12) fullratsimp(t);
```

(%o12) x^a-1

И третья основная функция упрощения выражений – уже никак с предыдущими двумя не соотносящаяся – radcan(выражение). Если ratsimp() и fullratsimp() ориентированы на упрощение рациональных действий, то radcan() занимается упрощением:

- ✓ логарифмических функций,
- ✓ экспоненциальных функций
- ✓ степенных с нецелыми рациональными показателями, то есть корней (радикалов).

Например, выражение «w» radcan() сможет упростить сильнее, чем ratsimp() или fullratsimp():

```
(%i13) radcan(w);
```

(%o13) $-\frac{2\,a}{\sqrt{x+a}}$

В некоторых случаях наилучшего результата можно добиться, комбинируя radcan() с ratsimp() или fullratsimp().

С функцией radcan() смежны по действию еще два управляющих ключа.

- ✓ Один из них называется %e_to_numlog. Влияет он не на саму функцию, а на автоматическое упрощение. Если выставить его в true, то выражения вида e^(r*log(выражение)), где r — рациональное число, будут автоматически раскрываться в выражение r. Функция radcan() делает такие преобразования <u>независимо</u> от значения ключа.

```
(%i14) s:e^(3*log((x^3+1)/(x+1)));
```

(s) $e^{3\log\left(\frac{x^3+1}{x+1}\right)}$

```
(%i15) %e_to_numlog:false;
(%o15) false
(%i16) radcan(s);
```

(%o16) $e^{3\log\left(x^2-x+1\right)}$

```
(%i17) %e_to_numlog:true;
(%o17) true
(%i18) radcan(s);
```

(%o18) $e^{3\log\left(x^2-x+1\right)}$

- ✓ Второй ключ — radexpand (от radical, не путать с ratexpand) — влияет на упрощение квадратного корня из четной степени какого-либо выражения. Он, в отличие от большинства переключателей, имеет не два, а три значения: при значении all, sqrt(x2) будет раскрываться в x — как для действительных, так и для комплексных чисел; при значении true (по умолчанию), sqrt(x2) для действительных чисел превращается в |x|, а для комплексных не преобразуется; а при значении false, sqrt(x2) не будет упрощаться вообще.

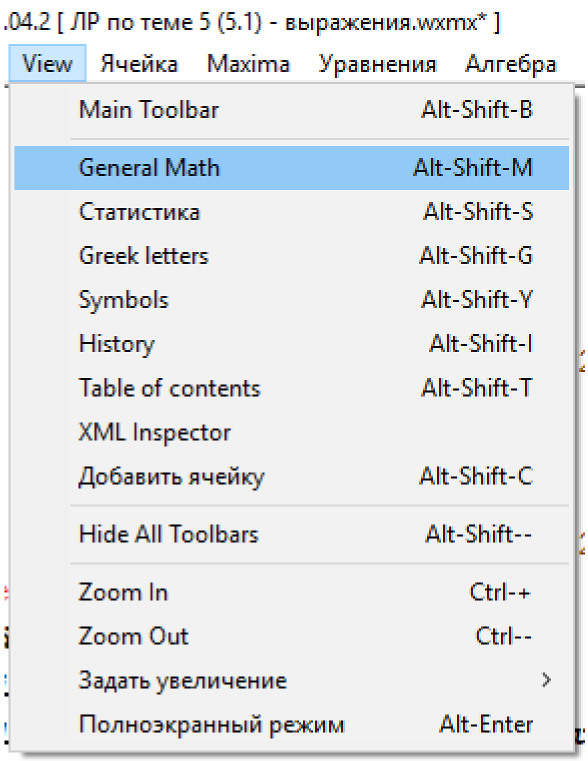
```
(%i19)  radexpand;
(%o19)  true

(%i20)  r:sqrt(x^2);
(r)     |x|

(%i21)  radexpand:false;
(radexpand) false

(%i22)  d:sqrt(x^2);
(d)     sqrt(x^2)

(%i23)  radexpand:all;
(radexpand) all

(%i24)  h:sqrt(x^2);
(h)     x
```

Следующие две функции и один флаг относятся к упрощению факториалов.

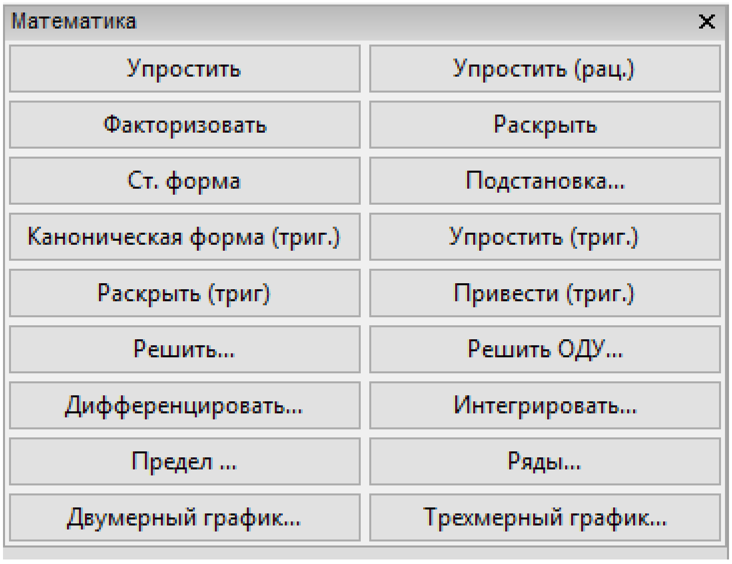
Функция factcomb(выражение) проводит упрощения вида n!*(n+1) = (n+1)! и тому подобные. Функция minfactorial, напротив, сокращает факториалы, то есть действует по принципу n!/(n–1)! = n. И флаг sumsplitfact, который изначально установлен в true, находясь в состоянии false, приводит к тому, что после того, как отработает factcomb, minfactorial вызывается автоматически.

```
(%i21)  factcomb((n+1)*n!);factcomb(n!/n);
(%o21)  (n + 1)!
(%o22)  (n - 1)!
(%i25)  minfactorial(n!/n);
(%o25)  n!/n
(%i26)  minfactorial(n!/(n-2)!);
(%o26)  (n - 1)n
```

Отметим, что интерфейс *wxMaxima* позволяет набирать имя функций ratsimp(); radcan(); factor(); expand(); в "одно касание" щелчком мыши.



Получим в левой части окна:



- Попробуйте использовать уже изученные функции, набрав их с помощью инструментов. Выражения для работы придумайте самостоятельно.

У функций, используемых для преобразования тригонометрических формул, присутствует общая для всех приставка — `trig`. Функция `trigexpand(выражение);` раскрывает скобки в тригонометрических выражениях:

```
(%i3) trigexpand(sin(2*x+y)+cos(x+2*y));
```

(%o3) $-\sin(x)\sin(2y)+\cos(x)\cos(2y)+\cos(2x)\sin(y)+\sin(2x)\cos(y)$

Эту функцию можно вызвать с более полным списком аргументов: `trigreduce(выражение, переменная),` — тогда формулы понижения степени будут применяться только по отношению к заданной переменной (переменная может быть, как и почти везде, не только отдельным символом, но и выражением).

Функция имеет несколько управляющих флагов, первый из которых опять же является тезкой самой функции. Он приводит к повторному раскрытию всех синусов-косинусов, то есть фактически равнозначен повторному вызову самой функции:

```
(%i4)%,trigexpand: true;
```

(%o4) $\cos(x)\left(\cos(y)^2-\sin(y)^2\right)-2\sin(x)\cos(y)\sin(y)+\left(\cos(x)^2-\sin(x)^2\right)\sin(y)+2\cos(x)\sin(x)\cos(y)$

Второй флаг — `halfangles` — управляет раскрытием формул половинных углов. Оба эти флага по умолчанию сброшены. А следующие два флага — `trigexpandplus` и `trigexpandtimes` — отвечают соответственно за применение формул сумм углов и кратных углов. То есть в примере выше сначала сработал флаг `trigexpandplus`, а затем — `trigexpandtimes`. Эти флаги по умолчанию установлены, что и видно из примера.

Кроме всего уже упомянутого, есть еще флаги `trigsign` и `triginverses`. Первый принимает традиционные два значения (по умолчанию — `true`) и регулирует вынос знака за пределы тригонометрической функции, то есть, к примеру, `sin(-x)` упростится до `-sin(x)`, а `cos(-x)` — до `cos(x)`. Флаг `triginverses` — трехзначный, и умолчательное его значение равно `all`. Он отвечает за обработку сочетаний вида `sin(asin(x))` или `atan(tan(x))`. Значение `all` позволяет раскрывать эти сочетания в обоих направлениях (при этом часть корней будет теряться); значение `true` оставляет разрешенным раскрытие только вида `sin(asin(x))`, то есть блокирует вариант с потерями периодических значений; а случай `false` запрещает оба направления преобразований.

Функция, обратная `trigexpand();` называется `trigreduce();`

```
(%i5) trigreduce(%);
```

(%o5) $\frac{\cos(2y+x)}{2}-\frac{\cos(2y-x)}{2}+\frac{\sin(y+2x)}{2}-\frac{\sin(y-2x)}{2}+\cos(x)\cos(2y)+\cos(2x)\sin(y)$

```
(%i6) trigreduce(%);
```

(%o6) $\cos(2y+x)+\sin(y+2x)$

— здесь, в полном соответствии со значением слова `reduce`, действуют формулы понижения степени.

Например, применив дважды эту функцию к результату предыдущего примера, мы получим его в исходном виде.

Третья функция занимается уже упрощением, и зовут ее, соответственно, trigsimp(выражение); Она старается упростить любое тригонометрическое выражение, используя известные формулы, такие как $\sin^2(x)+\cos^2(x)=1$ и тому подобные. Для наилучшего результата ее можно комбинировать с `trigreduce(); ratsimp(); / fullratsimp();` и `radcan();` Эти возможности *Maxima* по преобразованию и упрощению разнообразных выражений далеко не исчерпаны, для справок мы поместили описания ряда полезных функций в табл. 3.

Придумайте самостоятельно примеры, чтобы проанализировать работу флагов.

Таблица 3

Функции *Maxima* для преобразования выражений

Имя функции	Что делает?	Пример
assume	вводит ограничения	(%i1) sqrt(x^2); (%o1) $\lvert x\rvert$ (%i2) assume(x<0); (%o2) $[x<0]$ (%i3) sqrt(x^2); (%o3) $-x$
forget	отменяет ограничения	(%i4) forget(x<0); (%o4) $[x<0]$ (%i5) sqrt(x^2); (%o5) $\lvert x\rvert$
divide	делит один многочлен на другой; первый результат – частное; второй – остаток от деления	(%i6) divide(x^3-2,x-1); (%o6) $[x^2+x+1,\ -1]$
factor	раскладывает на множители	(%i7) factor(a*x^2+a*x+a); (%o7) $a\left(x^2+x+1\right)$ (%i8) factor(x^2+2*x+1); (%o8) $(x+1)^2$
expand	раскрывает скобки	(%i9) expand((2+3*x)*(3*y+5*x)); (%o9) $9xy+6y+15x^2+10x$
gcd	находит наибольший общий делитель многочленов	(%i10) gcd(x^3-1,x^2-1,(x-1)^2); (%o10) $x-1$
ratsimp	упрощает выражение	(%i11) a/(5*x)+b/x-c/x; (%o11) $-\frac{c}{x}+\frac{b}{x}+\frac{a}{5x}$ (%i12) ratsimp(%o11); (%o12) $-\frac{5c-5b-a}{5x}$
partfrac	преобразует в простые дроби по заданной переменной	(%i15) -x/(x^3+4*x^2+5*x+2); (%o15) $-\frac{x}{x^3+4x^2+5x+2}$ (%i16) partfrac((%o15),x); (%o16) $\frac{2}{x+2}-\frac{2}{x+1}+\frac{1}{(x+1)^2}$

Имя функции	Что делает?	Пример
trigexpand	раскрывает скобки в тригонометрическом выражении	(%i39) trigexpand(cos(3*x)); (%o39) $\cos(x)^3-3\cos(x)\sin(x)^2$
trigsimp	упрощает тригонометрическое выражение	(%i40) trigsimp((%o39)); (%o40) $4\cos(x)^3-3\cos(x)$
trigreduce	приводит к сумме элементов, содержащих **sin** или **cos**	(%i41) trigreduce((%o40)); (%o41) $4\left(\frac{\cos(3x)}{4}+\frac{3\cos(x)}{4}\right)-3\cos(x)$ (%i44) trigsimp((%o41)); (%o44) $\cos(3x)$

Задание 5.3

- Откройте новый файл.
- Сохраните файл. Формат имени файла: "ФИО студента, номер группы/подгруппы, тема 5, ЛР (задание 5.3).wxmx"

- Добавьте раздел: Решение уравнений.
- Познакомьтесь с теорией и выполните **все** действия. Примечание: нумерация команд может не совпадать.
- <u>Не забывайте</u> добавлять <u>подразделы</u>, названия которых должны соответствовать выполняемым командам (выполняемым действиям).

В система Maxima для решения линейных и нелинейных уравнений используется встроенная функция solve, имеющая следующий синтаксис:

solve *(expr, x)* – решает алгебраическое уравнение expr относительно переменной *x*

solve *(expr)* – решает алгебраическое уравнение expr относительно неизвестной переменной, входящей в уравнение.

Например, решим линейное уравнение $5x + 8 = 0$. Для этого воспользуемся кнопкой *Решить* на панели инструментов, при нажатии на которую появляется диалоговое окно *Решить* (Рис.13). Вводим исходное уравнение и нажимаем *OK*.

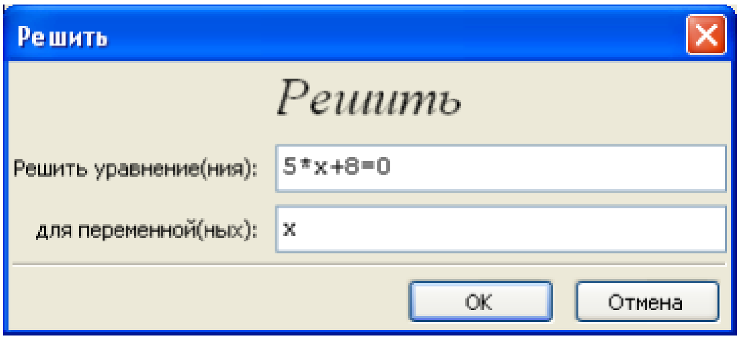


Рис. 13. Диалоговое окно для решения уравнений

В результате в рабочем документе сформируется команда для решения уравнения и выведется найденное решение:

```
(%i2) solve([5*x+8=0], [x]);
```

(%o2) $[x = -\frac{8}{5}]$

Команду для решения уравнений можно задавать таким образом, чтобы можно было легко выполнять проверку найденных решений. Для этого целесообразно воспользоваться командой подстановки ev.

Например, решим алгебраическое уравнение $x^3+1=0$ и выполним проверку найденных решений.

```
(%i1) eq:x^3+1=0;
```

(%o1) $x^3+1=0$

```
(%i2) resh:solve(eq, x);
```

(%o2) $[x=-\frac{\sqrt{3}\,\%i-1}{2}, x=\frac{\sqrt{3}\,\%i+1}{2}, x=-1]$

В результате получили три корня. Под именем resh у нас хранится список значений — корней уравнения. Элементы списка заключены в квадратные скобки и отделены один от другого запятой. К каждому такому элементу списка можно обратиться по его номеру. Воспользуемся этим при проверке решений: подставим поочередно каждый из корней в исходное уравнение.

```
(%i3) expand(ev(eq, resh[1]));
      expand(ev(eq, resh[2]));
      expand(ev(eq, resh[3]));
```

(%o3) $0=0$

(%o4) $0=0$

(%o5) $0=0$

С помощью команды allroots (expr) можно найти все приближенные решения алгебраического уравнения. Данную команду можно использовать в том случае, если команда solve не смогла найти решение уравнения или решение получается слишком громоздким, как, например, для следующего уравнения: $(1+2x)^3=13.5(1+x^5)$.

```
(%i8) eq:(1+2*x)^3=13.5*(1+x^5);
```

(%o8) $(2x+1)^3=13.5\left(x^5+1\right)$

```
(%i12) allroots(eq);
```

(%o12) [x = 0.82967499021294 , x = − 1.015755543828121 , x = 0.96596251521964 %i − 0.40695972319241 , x = − 0.96596251521964 %i − 0.40695972319241 , x = 1.0]

С помощью команды solve можно находить решение систем линейных алгебраических уравнений. Например, система линейных уравнений $\begin{cases} x+2y+3z+4k+5m=13 \\ 2x+y+2z+3k+4m=10 \\ 2x+2y+z+2k+3m=11 \\ 2x+2y+2z+k+2m=6 \\ 2x+2y+2z+2k+m=3 \end{cases}$ может быть решена следующим образом:

1. Сохраним каждое из уравнений системы под именами eq1, eq2, eq3, eq4, eq5.

```
(%i1) eq1:x+2*y+3*z+4*k+5*m=13;eq2:2*x+y+2*z+3*k+4*m=10;
      eq3:2*x+2*y+z+2*k+3*m=11;eq4:2*x +2*y+2*z+k+2*m=6;
      eq5:2*x+2*y+2*z+2*k+m=3;
```

(%o1) $3z+2y+x+5m+4k=13$

(%o2) $2z+y+2x+4m+3k=10$

(%o3) $z+2y+2x+3m+2k=11$

(%o4) $2z+2y+2x+2m+k=6$

(%o5) $2z+2y+2x+m+2k=3$

2. Находим решение системы.

```
(%i6) solve([eq1,eq2,eq3,eq4,eq5],[x,y,z,m,k]);
```

(%o6) $[[x=0, y=2, z=-2, m=3, k=0]]$

3. Выполним проверку найденного решения:

```
(%i7) ev([eq1,eq2,eq3,eq4,eq5],[%]);
```

(%o7) $[13=13, 10=10, 11=11, 6=6, 3=3]$

Таким образом, при подстановке полученного решения в каждое из уравнений системы получены верные равенства.

Функция solve системы Maxima может решать и системы линейных уравнений в случае, если решение не единственно. Тогда она прибегает к обозначениям вида %r_number чтобы показать, что неизвестная переменная является свободной и может принимать любые значения.

Для решения систем нелинейных уравнений можно воспользоваться командой algsys. Например, найдем решение системы уравнений $\begin{cases} x^2+16y=9 \\ 25x+9y^2=16 \end{cases}$. Воспользуемся пунктом меню *Уравнения→Solve algebraic system*.

В диалоговом окне вводим количество уравнений системы: 2.

В следующем диалоговом окне вводим сами уравнения и искомые переменные (рис.14).

После нажатия на кнопку *OK* получим решения:

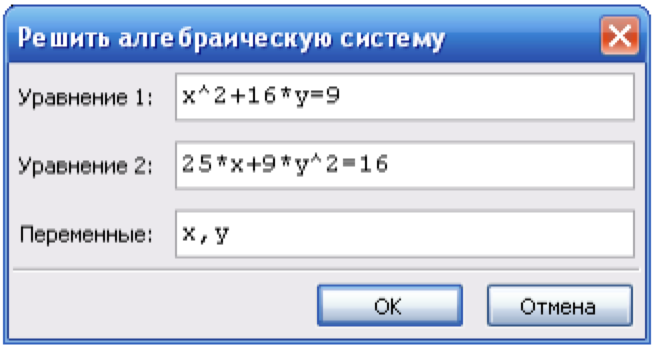


Рис.14. Ввод системы уравнений

```
(%i1) algsys([x^2+16*y=9, 25*x+9*y^2=16], [x,y]);
(%o1) [[x=0.53317586429032,y=0.54473272198613],
[x=-9.743068391866913,y=-5.370461538461538],[x
=7.128208840616651 %i+4.604946339770822,y=
2.412864405204793-4.103127401214957 %i],[x=
4.604946339770822-7.128208840616651 %i,y=
4.103127401214957 %i+2.412864405204793]]
```

Примеры: Решение одного уравнения с одним неизвестным

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

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

Решение одного уравнения в символьном виде:

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

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

Решение системы уравнений в символьном виде:

```
(%i10) solve([x*y/(x+y)=a,x*z/(x+z)=b,y*z/(y+z)=c], [x,y,z]);
```

(%o10) $$[[x=0,y=0,z=0],[x=\frac{2\,a\,b\,c}{(b+a)\,c-a\,b},y=\frac{2\,a\,b\,c}{(b-a)\,c+a\,b},z=-\frac{2\,a\,b\,c}{(b-a)\,c-a\,b}]]$$

В последнем примере решений несколько, и Maxima выдаёт результат в виде списка. Функция solve применима и для решения тригонометрических уравнений. При этом в случае множества решений у тригонометрических уравнений выдается соответствующее сообщение только и одно из решений. Пример:

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

(%o13) $$`solve'isusingarc-trigfunctionstogetasolution.Somesolutionswillbelost.[x=0]$$

Также Maxima позволяет находить комплексные корни

```
(%i18) solve([x^2+x+1], [x]);
```

(%o18) $$[x=-\frac{\sqrt{3}\,i+1}{2},x=\frac{\sqrt{3}\,i-1}{2}]$$

Приложение 1

Бесконечные периодические дроби и их перевод в дроби обыкновенные

Вспомним, как организована бесконечная периодическая дробь:

$$3\frac{5}{11}=3,454545...=3,\left(45\right)$$

Дробь может содержать целую часть (у нас – 3), цифры после запятой, которые не повторяются (у нас их нет), и повторяющуюся цифру или группу цифр (у нас 45), которая называется периодом.

Рассмотрим еще одну:

$5\frac{4}{65}=5,06153846153...=5,0\left(615384\right)$ – здесь целая часть 5, неповторяющаяся цифра после запятой одна – 0, и период состоит из шести цифр – 615384.

Теперь можем приступать к трансформации бесконечной дроби в обыкновенную!

Потребуется:

1. Посчитать, сколько цифр в периоде и после запятой, но до него.

2. Записать натуральным числом все цифры после запятой, включая период.
3. Записать натуральным числом все цифры после запятой до периода.
4. Записать разность этих двух натуральных чисел.

5. Разделить эту разность на число, в котором столько девяток, сколько цифр в периоде нашей дроби и столько нулей, сколько цифр до периода. Полученную дробь сократить – это дробная часть числа.

6. Не забыть про целую часть числа! Ее надо добавить к полученной дробной части.

Например, нужно представить бесконечную дробь $2,727272...=2,\left(72\right)$ в виде смешанного числа.

1. В периоде 2 цифры (72), до периода цифр нет (0).

2. Записываем период натуральным числом – 72. Записываем натуральным числом цифры до периода – 0.

3. Считаем разность этих чисел: $72-0=72$.

4. Делим эту разность на число: 99 – в нем две девятки (по числу цифр периода) и нет нулей, так как до периода в числе никаких цифр нет: $\frac{72}{99}=\frac{8}{11}$ – это дробная часть.

5. Добавляем целую часть к дробной и получаем результат: $2\frac{8}{11}$

Попробуем еще раз:

Представим в виде смешанного числа дробь $1,791666...=1,791\left(6\right)$

1. В периоде 1 цифра (6), до периода три цифры (791).

2. Записываем цифры после запятой, включая период, натуральным числом – 7916. Записываем натуральным числом цифры до периода – 791.

3. Считаем разность этих чисел: $7916-791=7125$.

4. Делим эту разность на число 9000 – в нем одна девятка (по числу цифр периода) и три нуля, так как до периода в числе три цифры: $\frac{7125}{9000}=\frac{1425}{1800}=\frac{285}{360}=\frac{57}{72}=\frac{19}{24}$ – это дробная часть.

5. Добавляем целую часть к дробной и получаем результат: $1\frac{19}{24}$.

В последний раз тренируемся:

Представим в виде обыкновенной дроби число $0,6428571428571\ldots=0,6(428571)$

1. В периоде 6 цифр (428571), до периода одна цифра (6).

2. Записываем цифры после запятой, включая период, натуральным числом – 6428571. Записываем натуральным числом цифры до периода – 6.

3. Считаем разность этих чисел: $6428571-6=6428565$.

4. Делим эту разность на число 9999990 – в нем шесть девяток (по числу цифр периода) и один ноль, так как до периода в числе одна цифра: $\frac{6428565}{9999990}=\frac{1285713}{1999998}=\frac{428571}{666666}=\frac{142857}{222222}=\frac{47619}{74074}=\frac{4329}{6734}=\frac{333}{518}=\frac{9}{14}$ – это результат, так как у числа не было целой части.

Источник:
https://lampa.io/p/преобразование-обыкновенных-и-десятичных-дробей-00000000581501436ed8d9adb1b59c5b

Цепные дроби и точность их вычисления

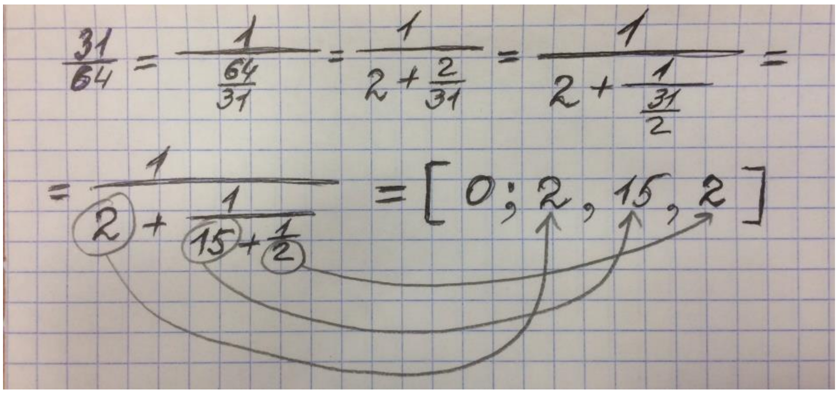
Цепная дробь (или **непрерывная дробь**) — это математическое выражение вида

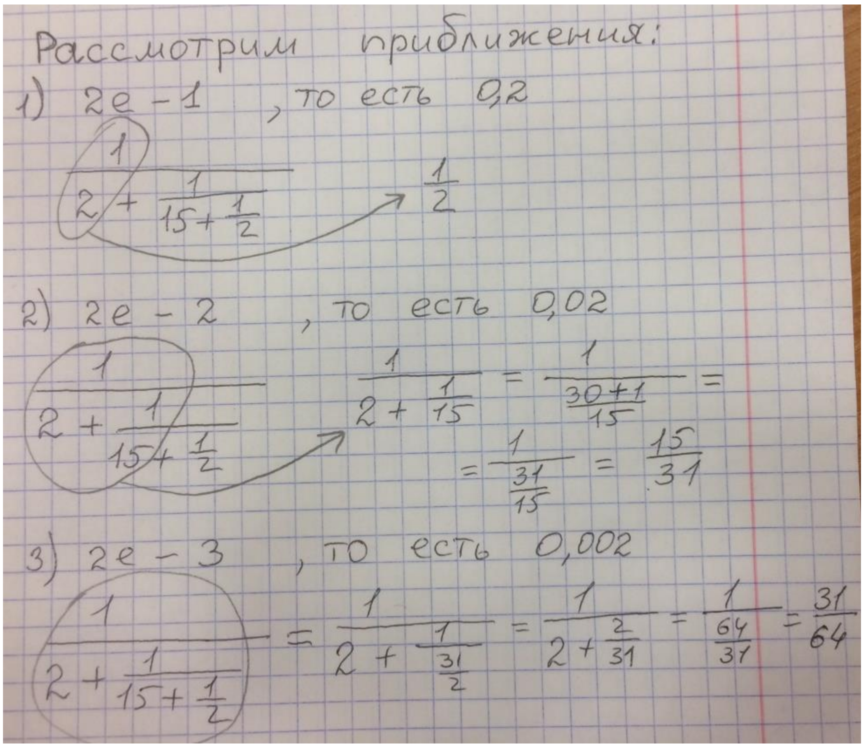
$$[a_0; a_1, a_2, a_3, \cdots] = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \ldots}}}$$

где a_0 есть целое число и все остальные a_n натуральные числа (то есть положительные целые). Любое вещественное число можно представить в виде цепной дроби (конечной или бесконечной).

- ✓ Число представляется конечной цепной дробью тогда и только тогда, когда оно рационально.
- ✓ Число представляется периодической цепной дробью тогда и только тогда, когда оно является квадратичной иррациональностью.

Например





Источники:
https://dic.academic.ru/dic.nsf/ruwiki/1187603