Федеральное агентство по образованию

Н.А. Стахин

ОСНОВЫ РАБОТЫ С СИСТЕМОЙ АНАЛИТИЧЕСКИХ (СИМВОЛЬНЫХ) ВЫЧИСЛЕНИЙ MAXIMA

(ПО для решения задач аналитических (символьных) вычислений)

Учебное пособие

Москва 2008

Стахин Н.А. Основы работы с системой аналитических (символьных) вычислений Maxima. (ПО для решения задач аналитических (символьных) вычислений): Учебное пособие. – Москва: 2008. — 86 с.

Настоящее учебное пособие содержит описание основных приемов работы с компьютерной программой для выполнения алгебраических вычислений, символьных преобразований и построения разнообразных графиков — Maxima. Система Maxima распространяется под лицензией GPL и доступна как пользователям ОС Linux, так и пользователям MS Windows.

Учебное пособие является составной частью проекта внедрения свободного программного обеспечения в образовательные учреждения, ориентировано на учителей школ и преподавателей педагогических вузов, учащихся и студентов и предназначено как для самообразования преподавателей математики и информатики школ пилотных регионов, так и сопровождения курсов повышения квалификации при изучении свободного программного обеспечения.

Оглавление

Предисловие

Большинство компьютерных программ, предназначенных для вычислений, умеет работать только с числовыми выражениями. Да и сами создатели первых компьютеров вряд ли изначально предполагали, что настанет такое время, когда проектируемые ими ЦЭВМ (цифровые электронно–вычислительные машины), равно как и различные «компьютеры» (англ. *computer* означает "*вычислитель*"), освоив арифметику, смогут освоить и алгебру.

Поясним, чем алгебра отличается от арифметики. «Арифметика», как известно, происходит от греческого слова ἀριθμός, означающего «число». В типичных задачах арифметики по известным или данным числам находится неизвестное число. Все операции в арифметике совершаются с численными величинами. В алгебре неизвестную величину обозначают буквой (символом) и для нахождения этой величины совершают различные математические (символьные) преобразования, в которых известные величины также обозначены буквами.

Рассматриваемая в настоящем пособии компьютерная программа Maxima представляет собой свободно распространяемую систему компьютерной алгебры, рассчитанную на широкий круг пользователей.

Данное пособие в первую очередь предназначено для первичного знакомства с системой компьютерной алгебры Maxima и может использоваться как при очном обучении, так и в системе дополнительного образования. Пособие содержит большое число различных примеров по разным разделам математики, имеет набор заданий для самостоятельной работы. Наилучшим вариантом обучения будет такой, когда читатель самостоятельно с пособием в руках сначала практически повторит на компьютере рассматриваемые примеры, а затем будет использовать приобретенные им навыки в дальнейшей работе.

Отметим, что наряду с документацией в качестве достоверного источника о дополнительных свойствах (флагах) различных функций в пособии были использованы описания Тихона Тарнавского [7,8]. Все приведенные в пособии решения задач и примеров были подвергнуты реальной проверке на компьютере с установленным на нем дистрибутивом от компании ALTLinux, уровень сложности решаемых задач соответствует вузу. Список использованных источников имеется в конце пособия.

Если по какой-либо причине на вашем компьютере не установлен описываемый в пособии графический интерфейс программы wxMaxima, то в качестве руководства в работе с программой Maxima из терминала мы рекомендуем имеющийся в Интернете перевод статьи Роберта Додиера [6].

Со всеми трудностями, замечаниями и пожеланиями публикации других необходимых в работе разделов компьютерной математики просим обращаться по адресу электронной почты spo_method_support@armd.ru.

Введение

Программа Maxima распространяется под лицензией GPL и доступна как пользователям ОС Linux, так и пользователям MS Windows. К сожалению, русская версия программы не имеет даже простой справки на русском языке, а немногочисленные статьи, посвященные изучению этой программы, имеющиеся в некотором количестве в Интернете, – не всегда доступны и, зачастую, рассчитаны на уже компьютеризированного пользователя.

Предлагаемое пособие в первую очередь предназначено для первичного знакомства с системой компьютерной алгебры Maxima и может использоваться как при очном обучении, так и в системе дополнительного образования.

Большинство компьютерных программ, как уже отмечалось, предназначено для вычислений с числовыми выражениями. Как правило, их результаты бывают приближенными, ведь при операциях с вещественными числами происходит их округление. Системы *компьютерной математики*, избавлены от подобных недостатков. Они способны использовать в процессе вычислений математические теоремы и факты. Так, известное тригонометрическое тождество гласит, что $\sin^2 x+\cos^2 x=1$ для любого x. Ни один калькулятор не способен применить это тождество в процессе преобразований, в то время как такие программы, как *Mathematica* или *Maxima*, предназначенные для символьных вычислений, легко справляются с подобными задачами.

Там, где необходимо выполнить вычисления точно, либо осуществить аналитическое преобразование, например, упростить сложное математическое выражение, вычислить в символьном виде производную или первообразную заданной функции, разложить ее в ряд Тейлора, найти корни уравнения и т. д., применяются *системы компьютерной алгебры* (системы символьных вычислений). Отметим также особую роль подобных систем в техническом и математическом образовании. Они позволяют проверить результаты громоздких математических расчетов и наглядно представить сложные математические объекты.

В отличие от коммерческой программы *Mathematica* программа *Maxima* распространяется под лицензией GNU, что позволяет рекомендовать ее широкому кругу пользователей. У каждой из этих двух программ есть свои сильные и слабые стороны. Удобный графический интерфейс является несомненным достоинством программы *Mathematica*, в то время как *Maxima* зачастую дает математически более строгие ответы. Так, например, при вычислении первообразной функции x^n *Maxima* просит уточнить значение n, так как при $n=-1$ результатом является функция $\ln x$, а при других n первообразная равна $x^{n+1}/(n+1)$. Программа *Mathematica* не уточняет n и для такой функции всегда в качестве первообразной выдает значение $x^{n+1}/(n+1)$, хотя если в качестве функции задать $1/x$, то получим верный результат – $\ln x$. Другими представителями систем такого рода являются программы *Mathcad*, *Maple*, *Axiom, GAP, FreeMat, Octave, Scilab, YACAS* и другие.

1. О происхождении Maxima

Maxima среди прочих аналогичных программ обладает наиболее широкими возможностями по части символьных вычислений; и вполне способна поспорить в этой области с коммерческими *Mathematica* и *Maple*. Система аналитических вычислений Maxima идеально подходит в качестве объекта для изучения как при обучении школьников старших классов, так и студентов вузов, ее могут использовать и профессиональные математики для проведения сложных расчетов и исследований.

По происхождению Maxima принадлежит к древнейшему роду среди программ этого вида деятельности – она обладает, пожалуй, одной из длиннейших историй среди всех сколь-нибудь распространённых сегодня программ. Жизнеописание Максимы берёт своё начало в 60-х годах, когда появился продукт под названием *Macsyma*, в котором реализовывались, как принято сейчас говорить, «передовые идеи» в области компьютерной математики. Позже эти идеи легли в основу обоих уже упомянутых лидеров проприетарного рынка математического софта – *Mathematica* и *Maple*.

Проект *Macsyma* был основан Энергетическим Управлением США (Departament of Energy, DOE) в 60-х годах. Разрабатывать его начали в легендарном Массачусетском Технологическом Институте (Massachusets Institute of Technology, MIT), на языке, который заслуженно считался тогда наиболее подходящим для невычислительных задач из всех существовавших на тот момент. Этим языком был Lisp, единственный из языков того времени доживший до наших дней и даже сейчас соперничающий по распространённости в живых проектах со многими ультрасовременными языками.

Естественно, изначально *Macsyma* была закрытым коммерческим проектом. Доступность проекта OpenSource-сообществу стала возможной благодаря профессору Техасского Университета Вильяму Шелтеру (William Schelter), который добился от DOE получения кода Macsyma и его публикации под лицензией GPL под именем Maxima. Он же долгое время разрабатывал как саму Максиму, так и один из диалектов лиспа – GCL (GNU Common Lisp) – на котором разрабатывалась Максима после её «освобождения». К величайшему сожалению, Вильям Шелтер умер в 2001 году. Но, как это часто бывает в мире открытого ПО, жизнь проекта не закончилась вместе с жизнью его основателя. Сейчас над проектом работает большое число математиков и программистов во главе с Джеймсом Эмундсоном (James Amundson). Теперь Maxima работает не только с GCL, но и с CLisp и CMUCL, полностью отвечающими стандарту ANSI Common Lisp (в отличие от GCL, в котором пока есть незначительные отклонения от стандарта).

В *Максиме* сейчас принят такой же принцип нумерации версий, как и в ядре Linux: номер состоит из трёх чисел, разделённых точками, причём номера с нечётным средним числом соответствуют так называемым development-версиям (разрабатываемым), с чётным – stable (стабильным). Стабильность

одной ветки и статус «в разработке» другой здесь означает не столько стабильность или нестабильность работы программы, сколько стабилизацию самого процесса разработки: в development-ветке новая младшая версия может иметь новые функции и новые интерфейсы, в стабильной же младшие версии будут содержать только исправления ошибок. Конечно, из этого следует несколько больший риск столкновения с ошибками в разрабатываемой версии, но риск этот весьма мал и в достаточной степени будет окупаться теми нововведениями, которые изначально в стабильной версии будут недоступны.

2. Загрузка и интерфейсы Maxima

Версия, существующая на начало лета 2008 года – 5.15.0, доступна для загрузки с русской версии сайта Maxima http://maxima.sourceforge.net/ru/. С каждой новой версией в Maxima появляются новые функциональные возможности и виды решаемых задач. Пакет Maxima либо входит в Linux-дистрибутив, и при отсутствии программы на компьютере, ее просто нужно доустановить из дистрибутива, либо пакет доступен для скачивания с упомянутого сайта проекта, либо Максиму можно собрать из исходников, лежащих на том же сайте.

Попробуем загрузить ту Максиму, что находится на нашем компьютере и приобрести минимальный начальный опыт, например, следующим образом. Найдем на рабочем столе или в меню задач Терминал (Konsole) и запустим эту программу, после получения приглашения введем *maxima*. Тем самым мы пытаемся загрузить Максиму из Консоли (её консольный вариант). Maxima тут же напишет нам номер своей версии и название сайта для своего последующего обновления. Добавим 2*2 после выражения в скобках, у нас получится (%i1) 2*2 (мы желаем узнать, знает ли об этом Максима?). Нажав *Enter*, мы замечаем, что курсор как всегда переместился на строчку ниже, но Максима не сказала нам ни "Да", ни "Нет". В отчаянии мы стучим пальцем по стрелке "Вверх", но курсор вверх не передвигается, однако снова сама собой появляется запись 2*2 (тут же замечаем для себя на будущее, что Maxima запомнила введенный нами текст) (рис. 1).

Рассуждая о том, что нам теперь делать, вспоминаем, что очень часто программисты используют знак **;** (точку с запятой), чтобы один оператор отделить от другого. Добавим **;** к нашему тексту, получим 2*2; и снова упорно надавим *Enter*. Maxima тут же отвечает, но сама возмущена – пишет (`Incorrect syntax`: ...) – и указывает нам на нашу некорректность в записи команды в две строчки (рис. 1).

Мы, конечно же, прежде всего, вынуждены "зарубить на носу", что точку с запятой надо ставить обязательно, если мы хотим получить ответ от Максимы. На наше новое немного измененное предложение (%i1) 2*12; (мы попробовали сдвинуть курсор влево и вставили 1 перед двойкой и исправления оказались доступны!). На это повторное предложение: "перемножь эти два числа"

Максима ответила (%o1) 24, но записала это по-своему: номер ответа (%o1) поставила на экране слева, а сам ответ 24 – на самой середине экрана (рис. 1).

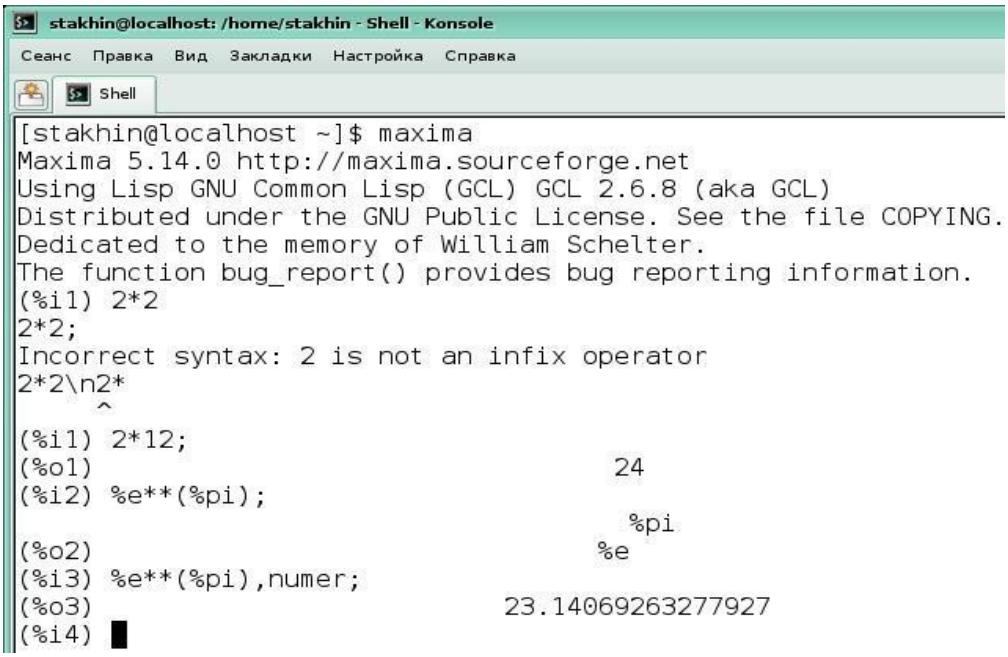


Рис. 1. Начало работы в консольной версии Максимы

Ответ на наше более сложное задание: вычислить e^{π}, которое мы ввели в виде одной строки %e**(%pi), Максима также получила мгновенно, но ответ вывела в 2 строки (в верхней строке Максима записала показатель степени %pi, а ниже и немного левее — основание степенного выражения %e). Из записи ответа видно, что Максима понимает математику, но предпочитает записывать ответ не в виде числа (зачастую, как мы знаем, приближенного), а в символьном виде (точно), однако в консольном варианте других возможностей – кроме как выводить результат в несколько строк – у Максимы нет. Когда мы повторили наше задание и добавили в новом задании слово numer, Максима вывела нам численный ответ в виде числа с большой точностью – 16 знаков (считать, конечно же, Максима умеет и численно, и весьма точно).

На рис. 2 на наш запрос вида
(%i1) 'integrate(exp(-x^2/2),x,–2,2);
оболочка *xmaxima* вывела ответ в виде 8-ми строк.

Трудно согласиться с тем, что красивый знак интеграла *xmaxima* записывает таким корявым способом – в виде набора обычных текстовых символов.

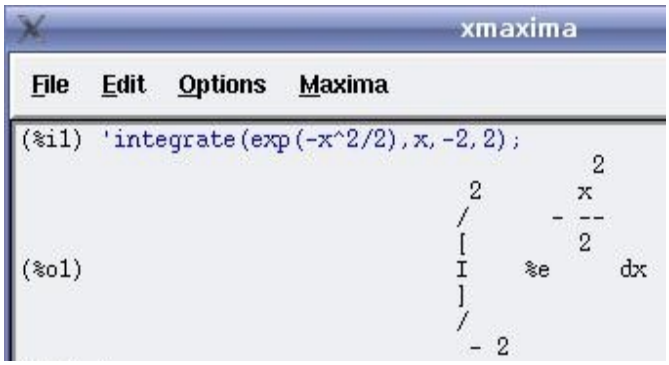


Рис. 2. Интерфейс xMaxima

Возможности Максимы в консольной версии, как видим, — весьма ограниченные. Без дополнительного графического интерфейса Максиму можно использовать лишь ввиду безысходности, так как голая консоль, или простой интерфейс *xmaxima* (рис. 2) довольно-таки бедны визуальными возможностями:

все математические формулы «рисуются» обычными текстовыми символами – в несколько строк дисплея.

Однако за счёт отсутствия сложной графической надстройки понижаются требования к компьютеру (железу). *Максима* в консольном варианте способна работать даже на таких компьютерах, которые сегодня и за компьютеры уже никто не считает. Самостоятельная оболочка, *xmaxima*, ненамного более требовательна к ресурсам, чем консольный интерфейс, оснащена системой меню и позволяет встраивать графические объекты прямо в документ в момент их создания (по желанию пользователя), но математические знаки имитируются в ней, также как и в консольной версии, текстовыми символами.

Ещё два интерфейса, *emaxima* и *imaxima*, реализованы как библиотеки к редактору *emacs* и будут полезны тем, кто набирает свои работы в формате LaTeX. Четвёртый графический интерфейс к Максиме – это интерфейс к редактору TeXmacs. TeXmacs используется в первую очередь для работы с текстами научной тематики. Для интеграции Максимы с редактором TeXmacs сначала необходимо запустить TeXmacs, затем для подключения Максимы – на панели инструментов программы TeXmacs необходимо нажать на кнопку с изображением монитора и выбрать интерактивную сессию Maxima (рис. 3). Выбор пункта *Maxima* позволит начать сеанс работы с этой программой. И закончить … , если *Maxima* плохо интегрирована в редактор TeXmacs (рис. 4).

Впрочем, упомянутые интерфейсы в дистрибутив Alt-Linux не входят. В меню рабочего стола KDE Образование | Математика, Прочие | Научные и математические, Образование | Разработка, входит Графическая оболочка wxMaxima, которую мы и рассмотрим подробнее.

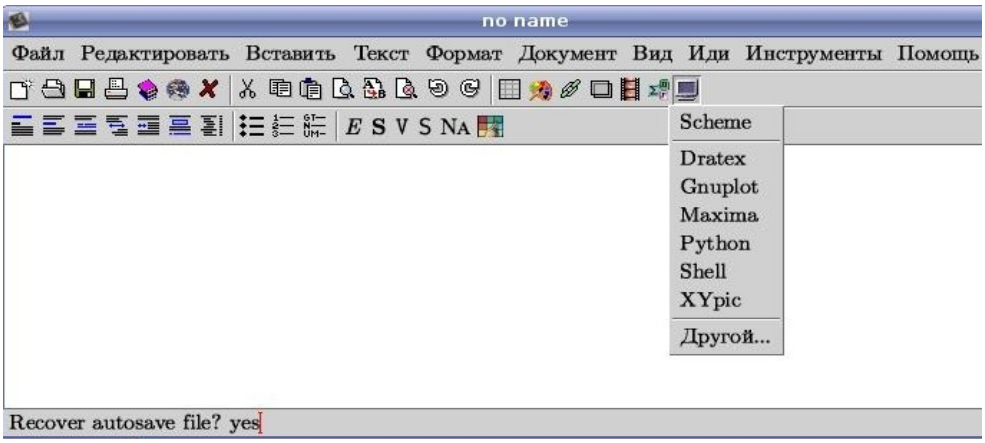


Рис. 3. Запуск Maxima в редакторе TeXmacs

Графическая оболочка wxMaxima (входящая в комплект поставки дистрибутива Alt-Linux) предоставляет пользователю удобный и понятный интерфейс на русском языке и графическое окно для результатов расчета (рис. 5). И, хотя в русскоязычной версии *wxMaxima* встроенной справки на русском языке пока нет, в сети Интернет непрерывно растет количество статей с примерами использования *Maxima*.

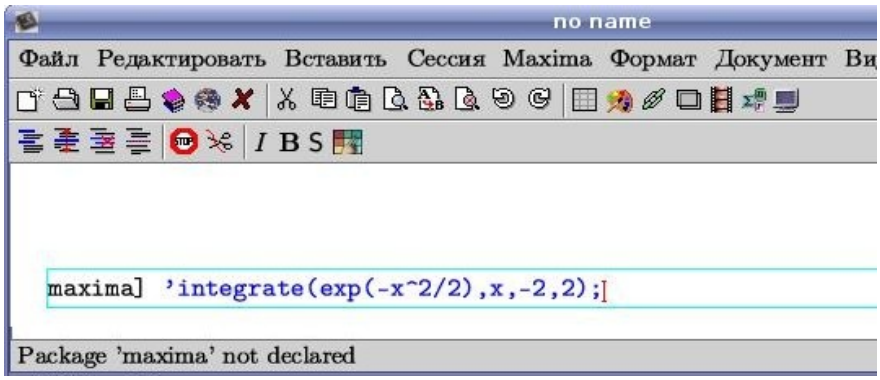


Рис. 4. Maxima не интегрирована в TeXmacs

На рис. 5 на аналогичный запрос вида (%i1) ' integrate (exp(–x^2/2),x,–2,2); графический интерфейс wxMaxima выводит красивый и привычный вид интеграла, правильное положение дробной черты, да и величина символов, используемых при записи показателя степени, явно меньше (то есть, величина символов масштабируется) по сравнению с величиной символов обычного текста.

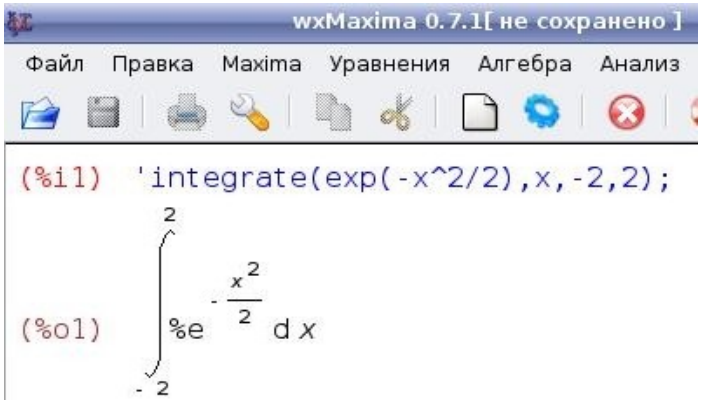


Рис. 5. Графический интерфейс wxMaxima

3. Ввод простейших команд в wxMaxima

После запуска *wxMaxima* появляется окно программы, показанное на рис. 6.

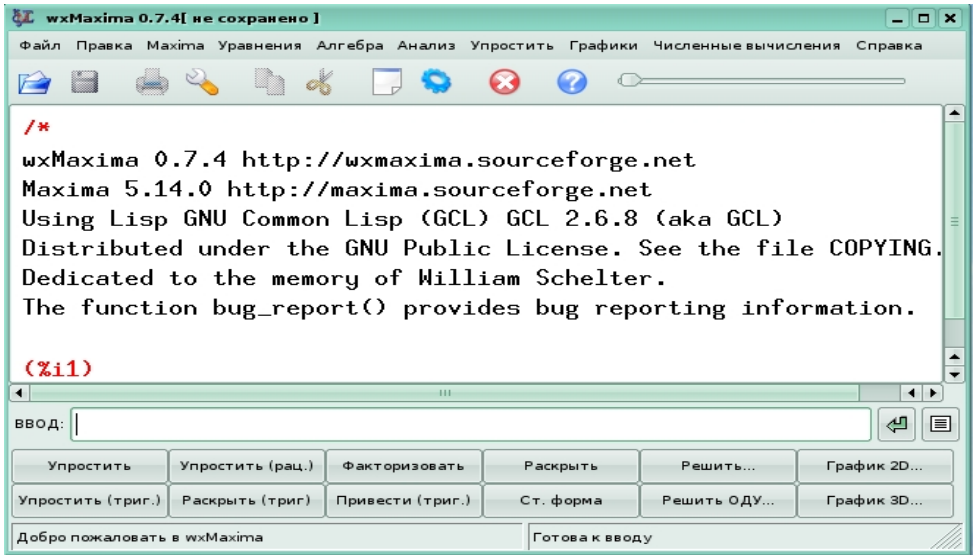


Рис. 6. Запуск графического интерфейса wxMaxima

В верхней графической части окна интерфейса *Maxima* рассказывает, что загружена версия 5.14.0, что она распространяется по лицензии GNU, с какого сайта доступна и кто её родитель. В нижнем окне в поле ВВОД: *Maxima*

приготовилась воспринимать команды. Разделителем команд является символ **;** (точка с запятой). После ввода команды необходимо нажать клавишу *Enter* для ее обработки и вывода результата.

В ранних версиях *Maxima* и некоторых ее оболочках (например, *xMaxima*), и в консольной версии наличие точки с запятой после каждой команды строго обязательно. Поэтому настоятельно рекомендуем при использовании *Максимы* не забывать добавлять точку с запятой **;** после каждой команды.

В случае, когда выражение надо отобразить, а не вычислить, перед ним необходимо поставить знак **‘** (одинарная кавычка). Но этот метод не работает, когда выражение имеет явное значение,

```
(%i3) 'integrate(x*x+3,x);
(%o3) ∫ x^2 + 3 dx
(%i4) integrate(x^2+3,x);
(%o4) x^3/3 + 3x
(%i5) 'sin(%pi);
(%o5) 0
```

например, выражение sin(π) Максима рассматривает как нуль и при наличии апострофа.

Трудно предусмотреть многообразие возможных вариантов использования Максимы для расчета или преобразования выражений. В сложных случаях, можно попытаться получить справку на английском языке. Для вызова справки достаточно в поле ВВОД написать ? и нажать *Enter*.

3.1. Обозначение команд и результатов вычислений

После ввода каждой команде присваивается порядковый номер. На приведенном ниже рисунке введенные команды имеют номера 1–3 и обозначаются соответственно (%i1), (%i2), (%i3). Результаты вычислений имеют соответственно порядковый номер (%o1), (%o2) и т. д. Где "i" – сокращение от англ. input (ввод), а "o" – англ. output (вывод).

```
(%i1) x:3;
(%o1) 3
(%i2) y:10;
(%o2) 10
(%i3) x/y;
(%o3) 3/10
```

Этот механизм позволяет при дальнейшей записи команд сослаться на ранее записанные, например (%i1)+(%i2) будет означать добавление к выражению первой команды выражения второй с последующим вычислением результата. Также можно использовать и номера результатов вычислений, например, таким образом (%o1)*(%o2).

Для последней выполненной команды в Maxima есть специальное обозначение – %.

Пример: Вычислить значение производной функции

$$Y(x) = x^2 + \frac{1}{x}$$

в точке x=1.

```
(%i9)  diff(x^2+1/x,x);
```

(%o9) $2x - \frac{1}{x^2}$

```
(%i10)  (%o9),x=1;
(%o10)  1
```

Команда (%i9) была выполнена, и был получен результат (%o9). Поэтому следующая команда (%i10) сослалась на уже полученный результат, но уточнила значение переменной x, поэтому команда получала вид (%i10) (%o9), x=1.

3.2. Ввод числовой информации

Правила ввода чисел в *Maxima* точно такие, как и для многих других подобных программ. Целая и дробная часть десятичных дробей разделяются символом *точка*. Перед отрицательными числами ставится знак *минус*. Числитель и знаменатель обыкновенных дробей разделяется при помощи символа / (*прямой слэш*).

Обратите внимание, что если в результате выполнения операции получается некоторое символьное выражение, а необходимо получить конкретное числовое значение в виде десятичной дроби, то решить эту задачу позволит применение оператора *numer*. В частности он позволяет перейти от обыкновенных дробей к десятичным.

```
(%i11)  3/7+5/3;
```

(%o11) $\frac{44}{21}$

```
(%i12)  3/7+5/3,numer;
(%o12)  2.095238095238095
```

Здесь *Maxima* прежде всего действовала по умолчанию. Она сложила дроби 3/7 и 5/3 по правилам арифметики точно: нашла общий знаменатель, привела дроби к общему знаменателю и сложила числители. В итоге она получила 44/21. Лишь после того, как мы попросили её получить численный ответ, она вывела приближенный, с точностью 16 знаков численный ответ 2,095238095238095.

3.3. Константы

В *Maxima* для удобства вычислений есть ряд встроенных констант, самые распространенные из них показаны в следующей таблице (табл. 1):

Таблица 1

Названия констант и их обозначение в Maxima

Название	Обозначение
π (число Пи)	**%pi**
e (экспонента)	**%e**
Мнимая единица ($\sqrt{(-1)}$)	**%i**
$+\infty$ (плюс бесконечность)	**inf**
$-\infty$ (минус бесконечность)	**minf**
Истина	**true**
Ложь	**false**
Комплексная бесконечность	**infinity**
слева (в отношении пределов)	**minus**
справа (в отношении пределов)	**plus**
Золотое сечение (1 + $\sqrt{(5)}$)/2	**%phi**

3.4. Арифметические операции

Обозначения арифметических операций в *Maxima* ничем не отличаются от классического представления, используются математические знаки: + – * /.

```
(%i2)  4**2;
(%o2)  16
(%i3)  4^2;
(%o3)  16
(%i4)  6!;
(%o4)  720
(%i5)  4^^1/2;
(%o5)  2
```

Возведение в степень можно обозначать тремя способами: ^ , ^^ , **. Извлечение корня степени *n* записывают, как степень ^^(1/*n*).

Напомним еще одну встроенную в *Maxima* полезную операцию – нахождение факториала числа. Эта операция обозначается восклицательным знаком.

Например, 6!=1· 2· 3· 4· 5· 6=120.

Для увеличения приоритета операции, как и в математике, при записи команд для *Maxima* используют круглые () скобки.

3.5. Переменные

Для хранения результатов промежуточных расчетов применяются переменные. Заметим, что при вводе названий переменных, функций и констант важен регистр букв, так переменные *x* и *X* – это две разные переменные.

Присваивание значения переменной осуществляется с использованием символа: (двоеточие), например x: 5;.

Если необходимо удалить значение переменной (очистить ее), то применяется метод kill:

kill(x) – удалить значение переменной x;

kill(all) – удалить значения всех используемых ранее переменных.

```
(%i1) x:3;y:5;
(%o1) 3
(%o2) 5
(%i3) kill(all);
(%o0) done
(%i1) x/y;
(%o1) x/y
```

И кроме того, метод kill начинает новую нумерацию для исполняемых команд (обратите внимание, что ответом на команду (%i3), приведенную выше, оказался ответ с номером ноль (%o0) *done*, и далее нумерация команд продолжилась с единицы).

3.6. Математические функции

В *Maxima* имеется достаточно большой набор встроенных математических функций. Вот некоторые из них (табл.2). Следует иметь ввиду, что некоторые названия функций отличаются от названий, используемых в отечественной литературе: Вместо tg – tan, вместо ctg – cot, вместо arcsin – asin, вместо arccos – acos, вместо arctg – atan, вместо arcctg – acot, вместо ln – log, вместо cosec – csc.

Таблица 2

Встроенные математические функции Maxima и примеры их использования

Функция	Обозначение
тригонометрические	sin (синус), cos (косинус), tan (тангенс), cot (котангенс)
обратные тригонометрические	asin (арксинус), acos (арккосинус), atan (арктангенс), acot (арккотангенс)
секонс, косеконс	sec x = 1/cos x, (секонс), csc x = 1/sin x, (косеконс)
гиперболические	sinh (гиперболический синус), cosh (гиперболический косинус), tanh (гиперболический тангенс), coth (гиперболический котангенс), sech (гиперболический секанс), cosh (гиперболический косеканс);

натуральный логарифм;
квадратный корень;
модуль;
остаток от деления;
	log();
	sqrt();
	abs();
	mod();

минимальный из списка	min(x1, ... ,xN)
максимальный из списка	max(x1, ... ,xN)
знак аргумента	=pos (x>0); =zero (x=0) sign(x); =neg (x<0); =pnz – (не определен)

```
(%i1)  min(5,1,3);
(%o1)  1

(%i2)  max(3,4,6,1);
(%o2)  6

(%i3)  sign(5);
(%o3)  pos

(%i4)  sign(-2);
(%o4)  neg

(%i5)  sign(y);
(%o5)  pnz

(%i6)  x:0;sign(x);
(%o6)  0
(%o7)  zero
```

3.7. Правило записи функций

Для записи функции необходимо указать ее название, а затем, в круглых скобках записать через запятую значения аргументов. Если значением аргумента является список, то он заключается в квадратные скобки, а элементы списка также разделяются запятыми.

Пример:

```
sin(x);
integrate(sin(x),x,-5,5); plot2d([sin(x)+3,cos(x)],[x,-%pi, %pi],[y,-5,5]);
```

3.8. Пользовательские функции

Пользователь может задать собственные функции. Для этого сначала указывается название функции, в скобках перечисляются названия аргументов,

после знаков := (двоеточие и равно) следует описание функции. После задания пользовательская функция вызывается точно так, как и встроенные функции *Maxima*.

Пример:

```
(%i1)  f(x):=x^2;
(%o1)  f(x) := x^2

(%i2)  f(3+7);
(%o2)  100
```

Нужно помнить, что не следует использовать для функций названия, зарезервированные для встроенных функций *Maxima* (записанные выше в табл. 2).

3.9. Перевод сложных выражений в линейную форму записи

Одним из самых сложных занятий для начинающих пользователей системы *Maxima* является запись сложных выражений, содержащих степени, дроби и другие конструкции, в линейной форме (в текстовой форме записи, при помощи ASCII символов, в одну строку).

Для облегчения данного процесса нелишне дать несколько рекомендаций:

1. Не забывайте ставить знак умножения! В графическом окне Maxima по правилам математики удвоенное значение переменной *x* записывает в виде 2*x*, но в окне ВВОД: команда для *Maxima* должна выглядеть как 2*x.

2. В случае сомнения всегда лучше поставить «лишние», дополнительные *скобки* (). Числитель и знаменатель выражения всегда необходимо заключать в скобки.

А также при возведении в степень основание и степень лучше всегда брать в скобки.

Примеры:

Математическая запись	**Команда для Максимы**
$\frac{x+2}{y-7}$	(x+2)/(y–7);
$(x+3)^{2y}$	(x+3)**(2*y);

3. Функция не существует отдельно от своих аргументов (если таковые имеются). Поэтому, например, при возведении в степень можно взять всю функцию с аргументами в скобки, а потом уже возводить полученную конструкцию в нужную степень: (sin(x))**2; Очень часто начинающие пользователи пытаются возвести в степень только название функции, забывая про аргументы: sin**2(x) – это неправильно!

4. Также помните, что несколько аргументов функции записываются в скобках, через запятую, **например,** min(x1,x2,x3,xN);

5. Недопустима запись функции sin(2*x) в виде sin*2*x или sin2x. Запомните, как действует *Maxima* при записи скобок: как только вы пытаетесь написать открывающую скобку, она тут же пишет вторую – парную ей – закрывающую скобку. Поэтому при записи функций напишите название функции, затем поставьте после нее пустые скобки и только потом в этих скобках напишите все ее аргументы, разделяя их запятыми. Никаких конструкций между названием функции и открывающейся скобкой быть не должно!

6. В случае записи сложного выражения разбейте его на несколько простых составляющих, введите их по отдельности, а затем объедините, используя рассмотренные ранее обозначения введенных команд.

Пример: необходимо ввести следующее выражение:

$$\frac{\frac{3}{7} - x}{(y - 5x + 2)^{\frac{3}{x - 7}}}$$

Разделим это выражение на три составные части: числитель, выражение в скобках и степень. Запишем каждую составную часть и объединим их в выражение.

```
(%i1)  3/7-x;
(%o1)  3/7 - x
(%i2)  y-5*x+2;
(%o2)  y - 5 x + 2
(%i3)  3/(x-7);
(%o3)  3/(x - 7)
(%i5)  (%o1)/(%o2)**(%o3);
                                3
                               --- - x
                                7
(%o5)  ------------------------------
                          3/(x - 7)
                (y - 5 x + 2)
```

Отметим, что строку с ошибочной записью команды для *Максимы* можно удалить с графического экрана (клавишей *Del*), если выделить целиком и команду, и соответствующий ответ (строк с командой (%i4) и с ответом (%o4) здесь нет – они удалены).

Отметим также, что запись команды для Максимы (%o1)/(%o2)**(%o3) в строке (%i5) мы могли написать "более точно", если бы поставили дополнительные скобки для знаменателя: (%o1)/((%o2)**(%o3)). Но *Maxima* и

без этих "лишних скобок" правильно нас поняла и написала правильно, так как понимает принятое в математике *старшинство операций*: сначала вычисляются функции, потом выполняется возведение в степень, затем операции деление и умножение и только потом сложение и вычитание.

4. Решение задач элементарной математики

К таким задачам можно отнести вычисление и преобразование арифметических выражений, построение графиков функций, решение уравнений и систем алгебраических уравнений.

4.1. Maxima упростит выражение

Рассмотрим возможности *Maxima* по упрощению и прочим преобразованиям выражений. В частности, речь пойдет об автоматическом раскрытии скобок и вынесении за скобки; об упрощении как арифметических действий над некоторыми элементами, так и выражений с участием степенных, показательных и логарифмических функций; а также об обработке тригонометрических выражений. Все эти функции призваны облегчать читаемость математических формул и повышать простоту их восприятия, а посему стоит уделить этому уроку достаточно внимания: при верном использовании данные манипуляции позволят сэкономить в процессе работы значительное количество времени.

Существенная часть интересующих нас прежде всего функций предназначена для преобразования рациональных выражений. Математики рациональным называют выражение, состоящее только из арифметических операторов и возведения в натуральную степень; естественно, элементы такого выражения могут содержать и неарифметические и нестепенные функции — тогда такие элементы с точки зрения рационального выражения считаются атомарными, т.е. неделимыми и непреобразуемыми.

Функции, работающие с рациональными выражениями, описаны в разделе документации «*Polynomials*»; потому как рациональные функции с математической точки зрения рассматриваются как расширение многочленов (полиномов) — примерно так же, как рациональные числа считаются расширением целых (многочлены, кстати, тоже иногда называют целыми функциями; хотя общий математический смысл этого термина несколько шире).

Имена всех функций *Максимы* по обработке рациональных выражений содержат буквосочетание `rat`, но не от слова *крыса*, а от слова *rational*. И начнем мы знакомство с ними с функции, которая так и называется: `rat(выражение)`. Эта функция преобразовывает рациональное выражение к так называемой *канонической форме* (*Canonical Rational Expression*, *CRE*). То есть раскрывает все скобки, затем приводит все к общему знаменателю, суммирует и сокращает; кроме того, приводит все числа в конечной десятичной записи к рациональным.

```
(%i9)  (x-1)^2/(x^2+x)+1/(x+1)+0.25;
```

$$(\%o9)\quad \frac{(x-1)^2}{x^2+x}+\frac{1}{x+1}+0.25$$

```
(%i10)  rat(%o9);
`rat' replaced 0.25 by 1/4 = 0.25
```

$$(\%o10)\quad \frac{5x^2-3x+4}{4x^2+4x}$$

Тут надо заметить, что атомарные элементы, т.е. символы и числа, в канонической форме рационального выражения в *Maxima* имеют другое внутреннее представление. Следует иметь ввиду, что если каноническая форма рационального выражения используется в других рациональных выражениях, то последние также автоматически приводятся к канонической форме:

```
(%i11)  (%)-1/x;
```

$$(\%o11)\quad \frac{5x-7}{4x+4}$$

Это может быть достаточно удобно, если вам нужно пошагово проделать большое количество рациональных преобразований: вы можете, один раз вызвав `rat();` ссылаться на предыдущие ячейки и, благодаря этому, далее автоматически видеть на каждом шаге итоговое выражение в канонической, а значит, достаточно компактной и удобной к восприятию форме.

```
(%i26)  log(exp(%o11))+1/x;
```

$$(\%o26)\quad \frac{5x-7}{4x+4}+\frac{1}{x}$$

```
(%i27)  rat(%);
```

$$(\%o27)\quad \frac{5x^2-3x+4}{4x^2+4x}$$

```
(%i28)  ratdisrep(%o26);
```

$$(\%o28)\quad \frac{5x-7}{4x+4}+\frac{1}{x}$$

Здесь, хотя (`%11`) и было выражением в канонической форме, выражение (%26) уже не находится в канонической форме ввиду того, что были преобразования log(exp()). Выражение (%27) получило каноническую форму в результате воздействия функции rat(); а выражение (%28) после применения функции ratdisrep( ); имеет общий вид.

Во многих случаях для получения наиболее простого результата требуется записать выражение в виде суммы простейших дробей. Такую задачу решает

функция `partfrac()`, ей только нужно указать имя переменной, относительно которой она сделает таки преобразования.

```
(%i29)  (5*x^2-3*x+4)/(4*x^2+4*x);
                        2
                     5 x  - 3 x + 4
(%o29)               ---------------
                        2
                     4 x  + 4 x
(%i30)  partfrac((5*x^2-3*x+4)/(4*x^2+4*x),x);
                      3     1     5
(%o30)              - ----- + - + -
                      x + 1   x   4
```

Несколько слов о приведении конечной десятичной записи чисел к рациональной. Конечная десятичная запись считается по определению приблизительной, что и понятно, т.к. 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 такое положение вещей вас не устраивает, вы можете убедить *Maxima* оставлять десятичную запись чисел как есть, установив в `true` значение флага `keepfloat` (по умолчанию он равен `false`).

4.2. Раскрытие скобок

Следующая функция раскрывает скобки в рациональном выражении и называется *ratexpand*(); (одно из значений слова *expand* и есть «раскрыть скобки»). Здесь также действует опция `keepfloat`.

Кроме нее, есть еще одна 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. Если же сбросить эту опцию в `false`, тогда все дроби с одинаковым знаменателем будут, напротив, объединены в одну дробь с числителем в виде суммы числителей изначальных дробей. То есть внешне результат будет в этом случае выглядеть почти так же, как и у функции `rat()`; к тому же единственная видимая пользователю разница проявляется только в рациональных выражениях от нескольких переменных (или различных иррациональных выражений).

Заключается эта разница в том, что после `ratexpand();` и в числителе, и в знаменателе дроби все скобки будут раскрыты, в случае же `rat();` слагаемые, где присутствуют, скажем, две переменных, будут сгруппированы,

и одна из них будет вынесена за скобки (в документации такая форма записи называется «рекурсивной» (*recursive*):

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

$$(\%o30)\quad \frac{(x+1)^2\left(\%e^x-1\right)}{(x+2)(2x+1)}$$

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

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

```
(%i32) rat(%);
```

$$(\%o32)\quad \frac{\left(x^2+2x+1\right)\%e^x-x^2-2x-1}{2x^2+5x+2}$$

```
(%i33) ratexpand(%);
```

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

Кроме того, разница, конечно, заключается и во внутреннем представлении: с точки зрения программы, после ratexpand(); выражение будет по-прежнему общего вида. Соответственно и все результаты дальнейших рациональных действий с выражением не будут автоматически «канонизироваться».

Следующая функция, собирает воедино дроби с одинаковыми знаменателями; зовут ее `combine();`

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

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

4.3. Снова раскрытие скобок

Помимо `ratexpand();` есть также и функция «просто» `expand();`. Различий между ними несколько, наиболее принципиальные таковы. Во-первых, `ratexpand();` раскрывает только рациональное выражение «верхнего уровня», все же подвыражения, не являющиеся рациональными, обрабатываются как атомарные, то есть внутрь них она не залезает; `expand();` же раскрывает скобки на всех уровнях вложенности.

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

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

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

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

На этом примере видно: функция `ratexpand();` "не знает" о том, что показатель степени равен 4 (если раскрыть скобки), а функция `expand();` не

только "знает" об этом, но и не желает записывать результат в виде $(b+a)^4$, а максимально раскрывает скобки:

Во-вторых, `ratexpand();` приводит дроби-слагаемые к общему знаменателю, а `expand();` этого не делает; в-третьих, на функцию expand не действует переключатель `ratdenomdivide`. И в-четвертых, `expand();` не преобразовывает к рациональным числам конечную десятичную запись — опять-таки, вне зависимости от флага `keepfloat`. Кроме всего сказанного функция `expand();`, в отличие от своего рационального сородича, имеет несколько вариаций — в виде отдельных функций с похожими названиями `*expand*();`, которые раскрывают скобки несколько по-разному.

В противоположность функциям `*expand*();`, раскрывающим скобки, можно также записать анализируемое выражение как произведение сомножителей, то есть максимально повыносить все за скобки. Делается это с помощью функции `factor();`

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

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

Если функции `factor();` передать целое число, она разложит его на простые множители; если же передать рациональное число, на множители будут разложены его числитель и знаменатель:

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

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

```
(%i59) (3*5^3*11*31*41*101*251*601*1801*4051*8101*268501)/(2^100-1);
(%o59) 1
(%i60) factor(123456789/987654321);
```

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

Если многочлен не может быть представлен в виде произведения нескольких сомножителей, его можно попытаться преобразовать в сумму таких произведений с помощью функции `factorsum();`

В следующем примере используется много переменных x, y, z, v, u, t, w и не удается вынести за скобки общий множитель, поэтому функция factor(); с поставленной ей задачей — записать результат в виде сомножителей — не справилась, но функция factorsum(); решила задачу и записала выражение в виде суммы произведений.

Функция `factorsum();` умеет раскладывать на множители только независимые слагаемые, то есть такие, которые не содержат одинаковых переменных. Если мы раскроем скобки в выражении, содержащем в двух разных местах один и тот же символ, то так как коэффициенты при этом символе после раскрытия сгруппируются, `factorsum();` не сможет понять,

```
(%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
```

каким именно образом разгруппировать их обратно. Нужно заметить, что функции `factor();` и `factorsum();` хотя и не имеют в имени приставки `rat`, все же ведут себя в смысле разбора передаваемых им выражений не как `expand();` и сопутствующие, а как `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)
```

4.4. А если нужно сделать еще проще

Функция `ratsimp(выражение);` упрощает выражение за счет рациональных преобразований, но, в отличие от остальных функций по обработке рациональных выражений, работает в том числе и «вглубь», то есть иррациональные части выражения не рассматриваются как атомарные, а упрощаются, в том числе и все рациональные элементы внутри них:

```
(%i9)  %e^((x^3+1)/(x+1));
                 x^3 + 1
                 -------
                 x + 1
(%o9)  %e

(%i10)  ratsimp(%);
                 x^2 - x + 1
(%o10)  %e
```

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.

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

```
(%i11)  (sqrt((x-a)^3)-(x+a)*sqrt(x-a))/sqrt((x-a)*(x+a));
```

$$(\%o11)\quad \frac{(x-a)^{3/2}-\sqrt{x-a}\,(x+a)}{\sqrt{(x-a)(x+a)}}$$

```
(%i12)  rat(%o11);
```

$$(\%o12)\quad \frac{\sqrt{x-a}^{3}+(-x-a)\sqrt{x-a}}{\sqrt{(x-a)(x+a)}}$$

```
(%i13)  ratexpand(%o11);
```

$$(\%o13)\quad \frac{(x-a)^{3/2}}{\sqrt{(x-a)(x+a)}}-\frac{x\sqrt{x-a}}{\sqrt{(x-a)(x+a)}}-\frac{a\sqrt{x-a}}{\sqrt{(x-a)(x+a)}}$$

```
(%i14)  ratsimp(%o11);
```

$$(\%o14)\quad -\frac{2a\sqrt{x-a}}{\sqrt{x^2-a^2}}$$

А для более серьезных упрощений существует расширенный вариант — `fullratsimp(выражение).`

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

$$(\%o1)\quad \frac{\left(x^{a/2}-1\right)^2\left(x^{a/2}+1\right)^2}{x^a-1}$$

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

$$(\%o2)\quad \frac{x^{2a}-2x^a+1}{x^a-1}$$

```
(%i3)  ratsimp(%);
```

$$(\%o3)\quad x^a-1$$

```
(%i4)  fullratsimp(%o1);
```

$$(\%o4)\quad x^a-1$$

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.

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, экспоненциальных функций и степенных с нецелыми рациональными показателями, то есть корней (радикалов). Например, выражение (%o11) из предыдущего примера в этом разделе `radcan();` сможет упростить сильнее, чем `ratsimp();` или `fullratsimp();`

```
(%i12)  (sqrt((x-a)^3)-(x+a)*sqrt(x-a))/sqrt((x-a)*(x+a));
```

$$(\%o12)\quad \frac{(x-a)^{3/2}-\sqrt{x-a}\,(x+a)}{\sqrt{(x-a)(x+a)}}$$

```
(%i13)  ratsimp(%);
```

$$(\%o13)\quad -\frac{2a\sqrt{x-a}}{\sqrt{x^2-a^2}}$$

```
(%i14)  fullratsimp(%o12);
```

$$(\%o14)\quad -\frac{2a\sqrt{x-a}}{\sqrt{x^2-a^2}}$$

```
(%i15)  radcan(%o12);
```

$$(\%o15)\quad -\frac{2a}{\sqrt{x+a}}$$

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 независимо от значения ключа. 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(x^2)` будет раскрываться в `x` — как для действительных, так и для комплексных чисел; при значении `true` (по умолчанию), `sqrt(x^2)` для действительных чисел превращается в `|x|`, а для комплексных не преобразуется; а при значении `false`, `sqrt(x_2)` не будет упрощаться вообще.

Следующие две функции 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`.

```
(%i21)  factcomb((n+1)*n!);factcomb(n!/n);
```

$$(\%o21)\quad (n+1)!$$

$$(\%o22)\quad (n-1)!$$

```
(%i25)  minfactorial(n!/n);
```

$$(\%o25)\quad \frac{n!}{n}$$

```
(%i26)  minfactorial(n!/(n-2)!);
```

$$(\%o26)\quad (n-1)n$$

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

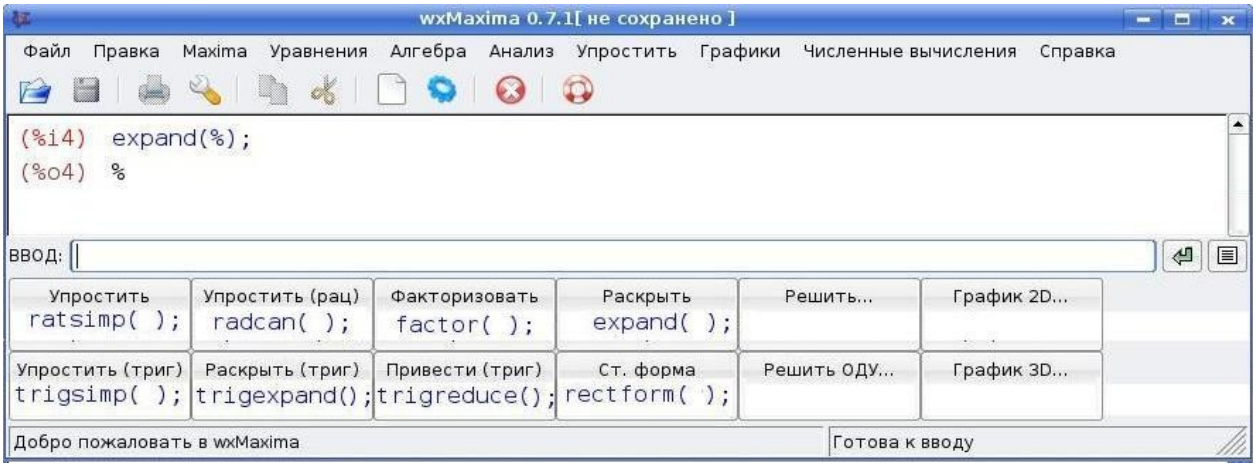


Рис. 7. Функции и соответствующие им кнопки интерфейса *wxMaxima*

Если после записи в поле ВВОД какого-либо выражения щелкнуть мышью по соответствующей кнопке Упростить, Упростить (рац), Факторизовать, Раскрыть, то введенное выражение будет рассматриваться как аргумент функции, математические названия которых показанные в виде второй строки на рис. 7, "спрятаны" под кнопками. Если поле ВВОД было не заполнено, то в качестве аргумента функции будет использоваться результат последней выполненной команды (%). В примере на рис. 7 поле ВВОД было пустое, после щелчка по кнопке Раскрыть *Maxima* выполнила команду `(%i4) expand(%)` и записала в ответ ссылку на предыдущий результат `(%o4) %`, которого в данном случае не было.

Аналогично под кнопками Упростить (триг), Раскрыть (триг), Привести (триг) "прячутся" функции trigsimp(); trigexpand(); trigreduce();, которые работают с тригонометрическими выражениями и о которых говорится в следующем разделе.

4.5. А эти функции имеют дело с углами …

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.

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

Продолжение таблицы 3

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) $cos(3x)$

5. Операторы и функции

На самом деле в Максиме нет четкого разграничения между операторами и функциями. Более того, каждый оператор — это на самом деле функция:

```
(%i1)  "+"(1,2,3);
(%o1)  6

(%i2)  "*"("+"(a,b),"/"(c,d));
(%o2)  (b + a) c
       -------
         d
```

Здесь в первом задании аргументами функции "сложить" являются три числа (1,2,3), которые записаны в круглых скобках и перечислены через запятую, во втором задании первым сомножителем функции "умножить" является сумма чисел (a,b), а вторым — частное двух чисел (c,d). Имена функций–операторов записаны в кавычках лишь потому, что содержат символы, нестандартные для имен функций. Это похоже на работу в командной оболочке UNIX, где, если в имя файла входят управляющие символы, вы должны взять это имя в кавычки.

Итак, все встроенные операторы *Максимы* являются функциями; более того, вы можете наделить любую (в том числе свою собственную) функцию определенными свойствами, которые фактически превратят ее в оператор. Так как разделение на функции и операторы в *Maxima* достаточно условно, то в этом разделе речь пойдет не только о некоторых операторах, но и о нескольких функциях, которые по природе своих действий сходны с операторами. Наиболее привычные операторы уже упоминались + − * / ^ или **

(возведение в степень) и функцию `sqrt(x)` (квадратный корень). Несколько примеров с произведением матриц, которое обозначается точкой (.)

```
(%i5) matrix([1,2,4],[0,a,a],[0,0,b]);
```

$$(\%o5)\ \begin{bmatrix} 1 & 2 & 4 \\ 0 & a & a \\ 0 & 0 & b \end{bmatrix}$$

```
(%i6) matrix([1,2],[0,a],[0,b]);
```

$$(\%o6)\ \begin{bmatrix} 1 & 2 \\ 0 & a \\ 0 & b \end{bmatrix}$$

```
(%i7) %i5.%o6;
```

$$(\%o7)\ \begin{bmatrix} 1 & 4b+2a+2 \\ 0 & ab+a^2 \\ 0 & b^2 \end{bmatrix}$$

В документации утверждается, что сама точка при этом должна быть отделена пробелами от обоих своих операндов — дабы не спутать ее с точкой десятичной. Но, как отмечается в [7], в некоторых дистрибутивх можно писать и без пробелов. В случае если заданные матрицы не могут быть перемножены из-за несовпадающих размерностей, *Maxima* выдаст сообщение об ошибке:

```
(%i7) %i5.%o6;
```

$$(\%o7)\ \begin{bmatrix} 1 & 4b+2a+2 \\ 0 & ab+a^2 \\ 0 & b^2 \end{bmatrix}$$

```
(%i8) %i6.%o5;
incompatible dimensions - cannot multiply
 -- an error.  To debug this try debugmode(true);
```

Восклицательный знак, стоящий после своего аргумента (т. е. постфиксный оператор), традиционно называется факториал. Не менее традиционно двумя восклицательными знаками обозначен полуфакториал — *произведение всех четных (для четного операнда) или нечетных чисел, меньших либо равных данному*. Функции `abs(x)` и `signum(x)` возвращают, как опять же нетрудно догадаться, модуль и знак числа. А функции `max(x1,...,xn)` и `min(x1,...,xn)` — соответственно максимальное и минимальное из заданных чисел.

Тут стоит остановиться на нескольких моментах. Во-первых, все функции и операторы *Maxima* работают не только с действительными, но и комплексными числами. Сами комплексные числа записываются в Максиме в

алгебраической форме, с мнимой единицей, обозначенной через `%i`; то есть в виде `a+b*%i`, где `a` и `b` — соответственно действительная и мнимая части числа. Поэтому факториал задан в наиболее общем виде и представляет собой, по сути, гамма-функцию (точнее, `x! = gamma(x+1)`), то есть определен на множестве всех комплексных чисел, кроме отрицательных целых. При этом факториал от натурального числа (и нуля) автоматически упрощается до натурального же числа:

```
(%i9)  0!;5!;10!;
(%o9)  1
(%o10)  120
(%o11)  3628800
```

Точно так же и модуль определен для всех комплексных чисел (`|a+b*i|=sqrt(a^2+b^2)`). Минимум, максимум и знак определены, естественным образом, только для действительных чисел, так как комплексные числа общего вида, как известно, между собой несравнимы.

Второй важный момент: когда некоторая встроенная функция или оператор *Maxima* не может получить для переданного выражения однозначный результат, *Maxima* пытается максимально упростить это выражение.

```
(%i2)  abs(-x);
(%o2)  |x|
(%i3)  signum(-x);
(%o3)  - signum(x)
(%i4)  put(trylevel,3,maxmin)$
(%i5)  max(x,-x);
(%o5)  |x|
(%i6)  max(x,x+1,x-abs(a));
(%o6)  x + 1
(%i7)  max(x,2*x,3*x);
(%o7)  max(3x , x)
```

Подобные упрощения, равно как и «раскрытие» факториалов и арифметических операторов, не считаются вычислениями, а следовательно оператор блокировки вычислений их не предотвращает:

Здесь сначала сработал оператор присвоения значений **:** (двоеточие), поэтому оператор блокировки вычислений **‘** (апостроф) не заблокировал оператор определения знака числа, и поэтому *Maxima* в качестве знака числа записала –1 (потому что $-x = -2$).

Для определения функции следует указать ее имя, аргумент (или аргументы), заключенный в круглые скобки, добавить два символы := и записать, как вычислять саму функцию, например,

```
(%i1)  f(x,y):=x*sin(y)+cos(y)/x;f(1,y);f(x,0);
```

$$(\%o1)\quad f(x\ ,\ y) := x\ sin(y) + \frac{cos(y)}{x}$$

$$(\%o2)\quad sin(y) + cos(y)$$

$$(\%o3)\quad \frac{1}{x}$$

Здесь следует остановиться на одном моменте. Если апострофом предварен вызов функции, то блокируется вычисление самой функции, но не ее аргументов. Если же поставить апостроф перед выражением, заключенным в скобки, то невычисленным останется все это выражение целиком, т. е. и все входящие в него функции, и все аргументы этих функций см. (%o11).

```
(%i6)  f(x,y):=x*sin(y)+cos(y)/x;y:6;x:3;
```

$$(\%o6)\quad f(x\ ,\ y) := x\ sin(y) + \frac{cos(y)}{x}$$

```
(%o7)  6
(%o8)  3
(%i9)  'f(x,y);
```

$$(\%o9)\quad f(3\ ,\ 6)$$

```
(%i10)  f(x,y);
```

$$(\%o10)\quad 3\ sin(6) + \frac{cos(6)}{3}$$

```
(%i11)  '(f(x,y));
```

$$(\%o11)\quad f(x\ ,\ y)$$

В *Maxima* оператор присвоения значений рассматривается как оператор именования, другим оператором именования является использованный здесь оператор задания функции. Обозначается он через :=, но аналогии здесь прослеживаются не с языками Pascal или Algol, а с другими обозначениями самой *Максимы*: с одной стороны определение функции можно воспринимать как уравнение (которое обозначается знаком =), а с другой — оно родственно назначению имени некоторому выражению (то есть :). То есть определение функции можно в какой-то мере считать симбиозом этих двух выражений — и оттого вполне логично, что оно обозначается обоими их символами. (В продолжение этой аналогии можно добавить, что в *Maxima* есть и расширенные варианты операторов присвоения и назначения функции, обозначаемые соответственно через :: и ::=.)

В противовес блокировке вычислений, можно также принудительно вычислить любое выражение — для этого имеется оператор, состоящий из двух апострофов:

```
(%i12)  ''%;
```

$$(\%o12)\quad 3\ sin(6) + \frac{cos(6)}{3}$$

В терминологии *Maxima* невычисленная форма выражения называется «*noun form*», вычисленная — «*verb form*». Сохраняя лингвистические параллели, на русский это можно перевести как «*несовершённая форма*» и «*совершённая форма*».

Если говорить о ячейках ввода-вывода, то значение ячейки ввода в *Maxima* закономерно сохраняется до его вычисления (т. е. в несовершённой форме), а значение ячейки вывода — после (т. е. в совершённой); другими словами, тут имеется естественный порядок «ввод → вычисление → вывод».

```
(%i13) %i11;
(%o13) 'f(x, y)
(%i14) ''%i11;
(%o14) f(x, y)
(%i15) ''(''%i11);
(%o15) 3 sin(6) + cos(6)/3
```

Оператор, принудительного вычисления, обозначенный двумя апострофами, является синонимом к функции `ev(`*`выражение`*`)`. Сама функция `ev` предоставляет гораздо более широкие возможности, нежели простое принудительное вычисление заданного выражения: она может принимать произвольное число аргументов, первый из которых — вычисляемое выражение, а остальные — специальные опции, которые как раз и влияют на то, как именно будет производиться вычисление.

Опция функции `ev`, одноименная этому переключателю, позволяет включить упрощение для данного конкретного вычисления — вне зависимости от того, включено или выключено оно глобально:

```
(%i1) simp:false$
(%i2) (1/2+2/3)*(3/4+4/5);122!/112!;sqrt(x^2);
(%o2) (1/2 + 2/3)(3/4 + 4/5)
(%o3) 122!/112!
(%o4) sqrt(x^2)
(%i5) %o2,simp;%o3,simp;%o4,simp;
(%o5) 217/120
(%o6) 500127116029962220800
(%o7) |x|
```

Точно так же, как двойной апостроф, — сокращение для `ev` без дополнительных опций есть еще более упрощенная запись функции `ev` с

опциями: в этом случае вместо имени функции и скобок вообще ничего писать не нужно; т. е. «`ev(`*`выражение`*`, `*`опц1`*`, `*`опц2`*`, …)`» можно записать просто как «*`выражение`*`, `*`опц1`*`, `*`опц2`*`, …»

Первая из таких опций связана с автоупрощением. Глобально автоупрощение регулируется переключателем `simp` (от «*simplification*» — упрощение), и по умолчанию оно включено; в любой момент его можно выключить, установив значение переключателя в `false`. Следует отметить еще, что вызов `kill(all);` не восстанавливает принятые по умолчанию значения переключателей; т. е. если мы, к примеру, изменили значение переключателя `simp`, как в примере выше, то для того чтобы вернуться к изначальному порядку вещей, установленному сразу после запуска *Maxima*, нам нужно не только сделать `kill(all);`, но и вручную назначить `simp: true`.

Опция `float` позволяет преобразовать все рациональные числа в конечную десятичную запись; опция `numer` включает опцию `float` и, кроме того, приводит к десятичному виду многие математические функции от числовых аргументов:

```
(%i1) cos(1);
(%o1) cos(1)
(%i2) %,numer;
(%o2) 0.54030230586814
```

Опция `noeval` блокирует сам этап вычисления как таковой; т. е. ее можно использовать для того чтобы применить к выражению другие опции функции `ev`, не перевычисляя выражение.

При этом опять-таки нужно иметь ввиду разницу между вычислением и упрощением:

```
(%i4) cos(%pi),noeval;
(%o4) - 1
(%i5) simp:false$ev(cos(%pi));
(%o6) cos(%pi)
```

Таким образом, мы можем принудительно упростить выражение, не перевычисляя его.

О других константных опциях и переключателях функции ev можно узнать, если вызвать из окна ВВОД справку, введя `? ev`.

Кроме константных значений есть еще несколько видов опций. Первая из них — это какое-либо имя специальной функции, которая занимается упрощением или преобразованием математических выражений. Будучи упомянута по имени в качестве опции, такая функция просто применяется к вычисляемому выражению. Например, выражение `fullratsimp` — это то же

самое, что и `fullratsimp(ev(выражение));`. Полный список таких функций можно найти через справку, введя `? evfun`.

Если в качестве опции ввести имя любой другой функции, не имеющей свойства `evfun`, то все несовершённые вхождения этой функции будут заменены совершёнными, т.е. принудительно вычислены.

Также в качестве опции можно задать назначение символа или функции; все такие назначения действуют локально в пределах вычисляемого выражения, и все подстановки производятся параллельно:

```
(%i2) x+y+z,x:1,z:a,y:x*z;
(%o2) x z + a + 1
```

Опция подстановки символа допустима не только в виде оператора присвоения, но и в виде равенства; сделано это, в частности, для того, чтобы в качестве подстановок можно было использовать решения, найденные функцией `solve`:

```
(%i1) ev(y^x,solve(x^3+3*x^2+3*x+1));
(%o1) 1/y
```

Здесь функция solve(); нашла решение $x = -1$ уравнения $x^3+3x^2+3x+1=0$ и подставила его в качестве показателя степени.

6. Графики функций

Как уже упоминалось, количество различных функций в *Maxima* разработчики постарались свести к минимуму, а широту размаха каждой конкретной функции, соответственно, к максимуму. Соблюдается эта тенденция и в функциях построения графиков: основных таких функций всего две, с очевидными, как всегда, названиями — `plot2d` и `plot3d` (одно из значений слова *plot* — *график*, а аббревиатуры 2d и 3d переводятся как *двумерный* и *трехмерный*). Кроме того, в графическом интерфейсе имеются эти же функции, но с дополнительным префиксом wx – обязаны графическому интерфейсу *wxMaxima*.

6.1. Степенная функция

Кратко о возможностях. Начнем с `plot2d`. Пусть мы выбрали программу *wxMaxima* в списке программ, загрузили её и, желая нарисовать двумерный график, щёлкнули по кнопке `График 2D...` В появившемся окне Выражение(ния) запишем сразу 4 функции: x, abs(x), x^2, x^3, перечислив их через запятую (рисовать, так рисовать, коль предлагают).

Изменим граничные значения переменных x и y на относительно небольшие (рис. 8) из: –1.5 к: 1.5 (чтобы график выглядел покрупнее) и, пока не разбираясь с форматом и опциями, щелкнем ОК (или нажмём *Enter*).

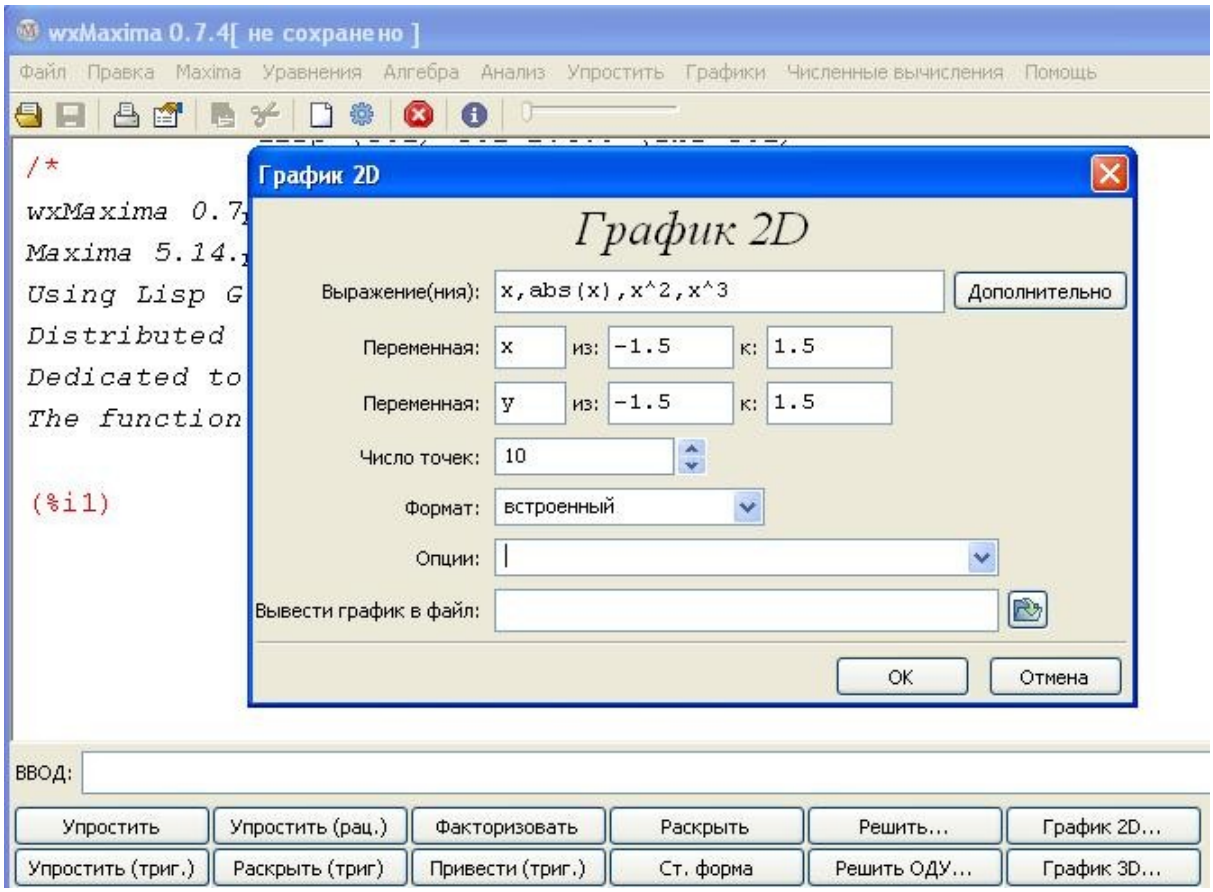


Рис. 8. Окна–формы интерфейса wxMaxima для рисования графиков

В верхнем графическом окне wxMaxima появились графики (рис. 9). Кроме того, в графической части окна *wxMaxima* появилась команда wxplot2d([x,abs(x),x^2,x^3], [x,–1.5,1.5], [y,–1.5,1.5])$. Что касается графика, то он "слишком правильный" и никак не масштабируется – таковы возможности "встроенного" формата.

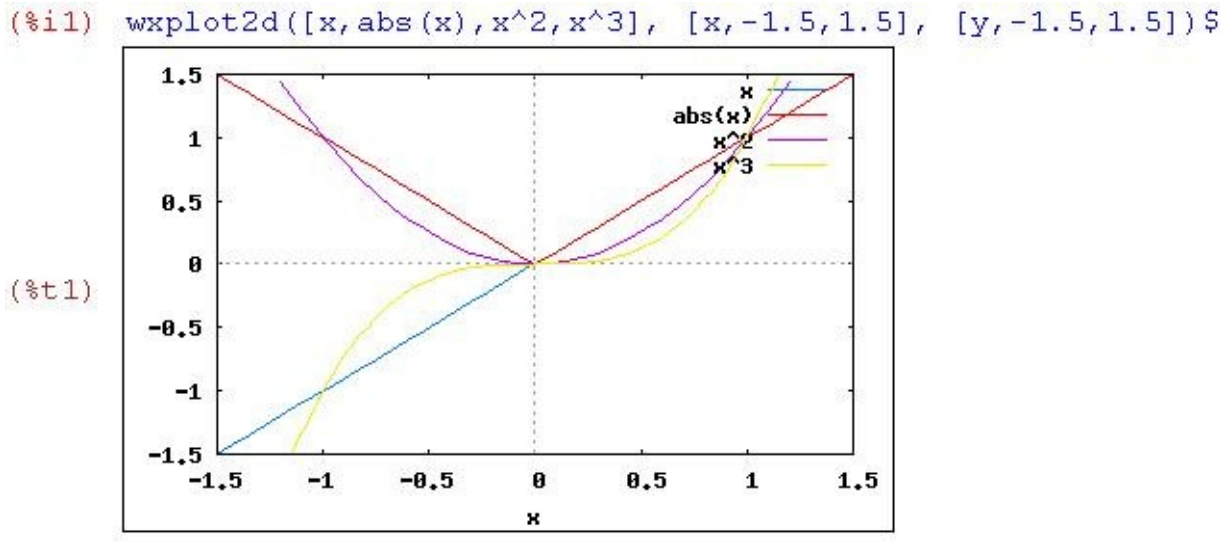


Рис. 9. Графики $y=x$, $y=\mathrm{abs}(x)$, $y=x^2$, $y=x^3$ во встроенном формате интерфейса *wxMaxima*

Построенный график абстрактно правилен в том отношении, что масштаб по оси *y* (ординат) в общем случае может не совпадать с масштабом по оси *x* (абсцисс). Но в данном случае различие масштабов совсем даже не требуется, коль скоро нас интересует график вида $y = x$. Нас скорее интересует один и тот же масштаб, нежели различный.

Попробуем изменить формат и опции графика. Мы можем перенести команду из графического окна назад в окно команд ВВОД – для этого нужно щелкнуть по команде и после её выделения нажать клавишу F5. А можно "вспомнить" предыдущие команды, нажимая на клавишу "Вверх" (ведь *Максима* запоминает введенные и исполненные команды). Однако, если команда достаточно длинная, то она может не уместиться в строке ВВОД.

В этом случае нужно нажать на кнопке «Многострочный ввод»  , (на которую указывает стрелка на рис. 10), правее кнопки «Ввести команду»  – декоративного изображения клавиши *Enter* и далее можно будет записывать команды неограниченной длины в отдельном окне с названием Ввод wxMaxima.

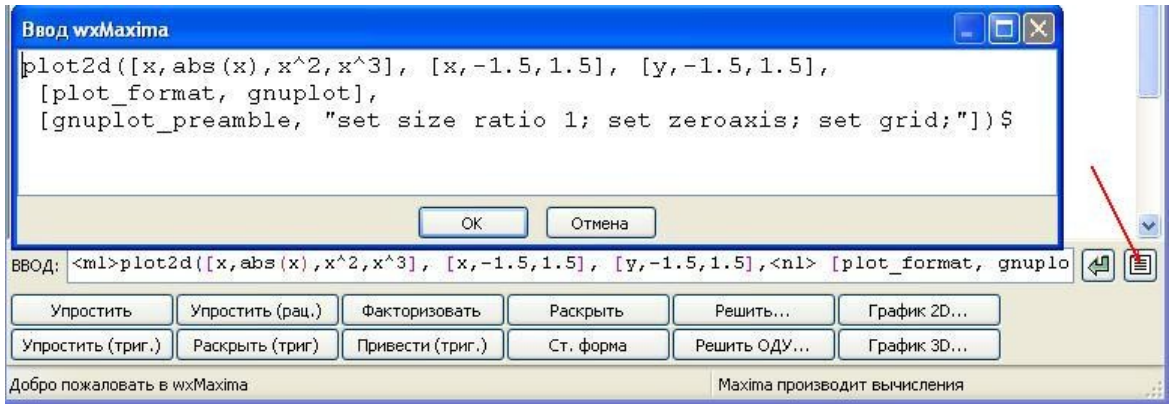


Рис. 10. Вызов окна Ввод wxMaxima для написания длинных команд

Заметим, что если просто оставить в окне ВВОД начальную запись x,abs(x),x^2,x^3, то после нажатия на кнопку График 2D... снова вернутся значения переменных *x* и *y*, которые были приняты по умолчанию (Переменная x из: –5 к: 5; Переменная y из: 0 к: 0).

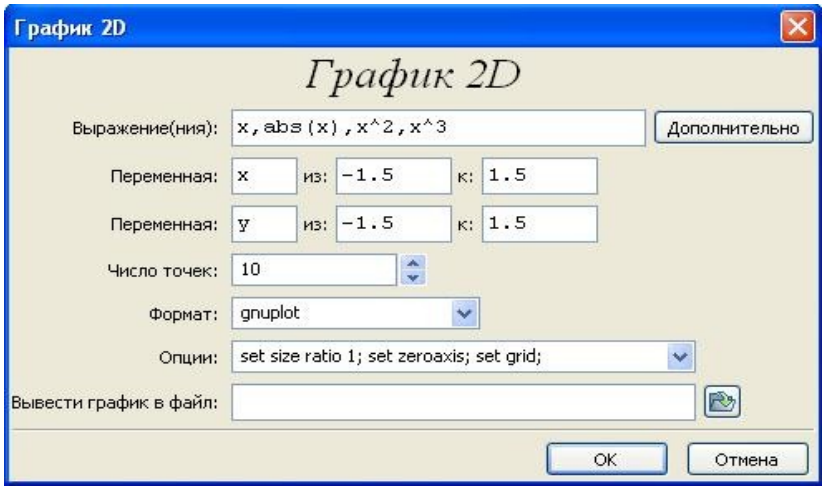


Рис. 11. Окно–форма *График 2D* для ввода опций двумерных графиков

Но если скопировать в буфер обмена (даже не планируя дальнейшей вставки) названия функций (через контекстное меню или просто, нажав, например, Ctrl+C) или даже если только щелкнуть в графическом окне на текст исполненной команды, то после щелчка по кнопке График 2D... снова вернутся и названия функций и предыдущие, уже вводимые нами, настройки по *x* и *y* из: –1.5 к: 1.5.

Выберем Формат: gnuplot, частично выберем, а частично напишем сами с клавиатуры в окне Опции: set size ratio 1; set zeroaxis; set grid; (перечисляя их через точку с запятой) и щёлкнем на кнопку ОК или нажмем клавишу *Enter*.

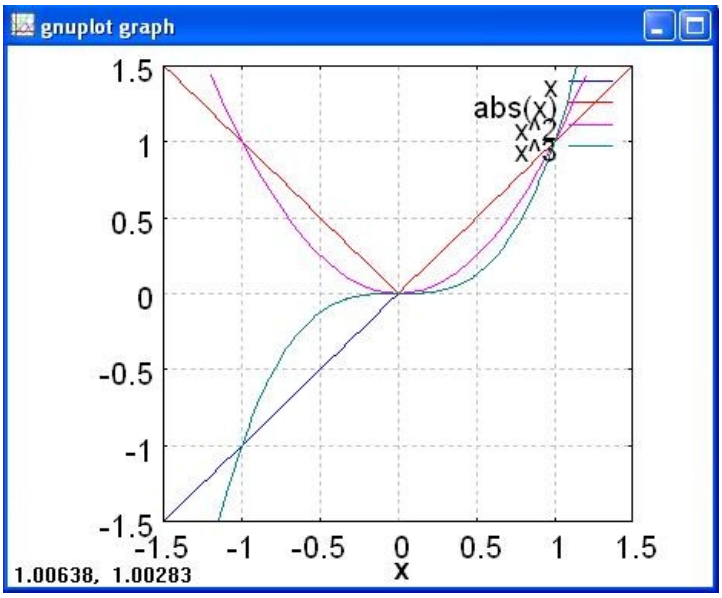


Рис. 12. Графики $y=x$, $y=\text{abs}(x)$, $y=x^2$, $y=x^3$ в формате gnuplot интерфейса wxMaxima

В формате gnuplot график *Максима* нарисует в отдельном окне, и мы можем масштабировать его (изменять размеры за счет изменения размеров окна). При движении мышки внизу слева отображаются координаты положения указателя мышки — сейчас он находится в точке пересечения всех графиков (1,1). Опция set zeroaxis; проводит оси через начало координат, опция set grid; прорисовывает сетку, опция set size ratio 1; выравнивает масштабы по осям координат, чтобы круг на мониторе выглядел круглым, а не в виде овала. Отметим, что последнее обстоятельство связано с тем, что разрешение монитора по горизонтали и по вертикали разное (пиксель не является "круглым").

Чтобы нарисовать график в формате openmath, команда для Максимы будет такой.

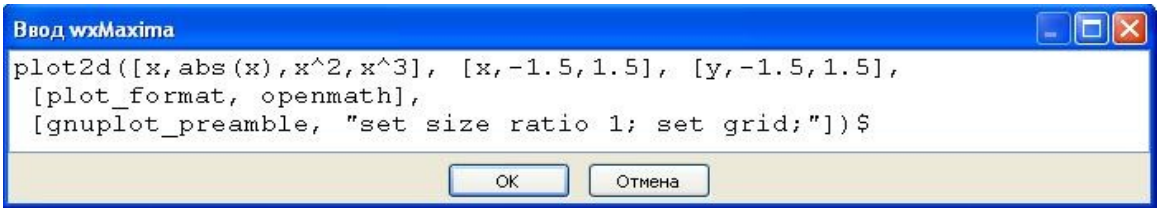


Рис. 13. Команда для Maxima для рисования графиков в формате openmath

График будет нарисован в отдельном окне.

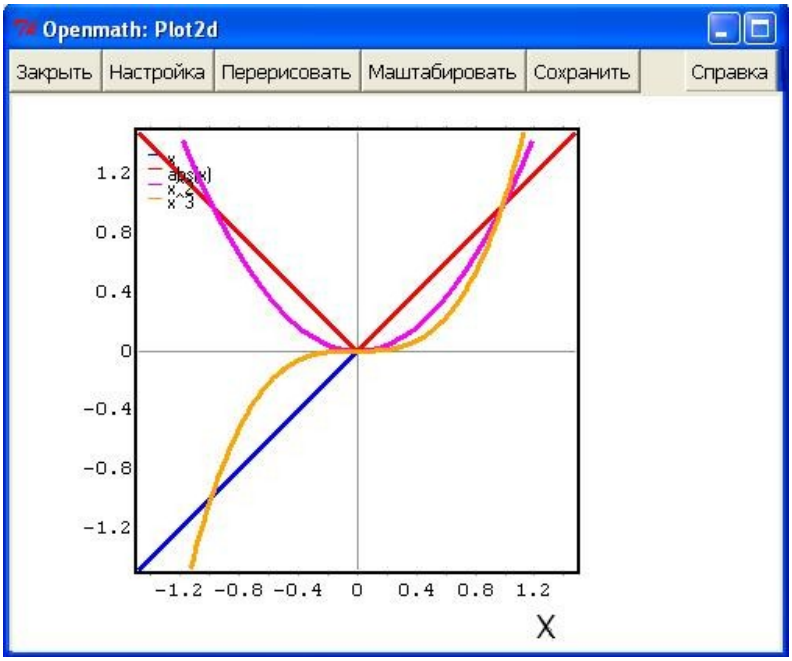


Рис. 14. Графики $y=x$, $y=\mathrm{abs}(x)$, $y=x^2$, $y=x^3$ в формате openmath интерфейса wxMaxima

И может видоизменяться в интерактивном режиме, в том числе: график можно масштабировать не только за счет изменения общих размеров окна, но и с помощью кнопки меню Масштабировать: после щелчка мышью на графике (размеры увеличатся), и после щелчка мышью при нажатой клавише Shift (размеры уменьшатся); график можно сохранить (кнопка Сохранить) в виде графического файла в формате *.ps, можно изменить толщину линий (кнопка Настройка), перерисовать (кнопка Перерисовать) график после изменения его параметров.

Отметим, что аргументами функции `plot2d` служат не отдельные переменные–параметры, а списки [для записи которых используются квадратные скобки]. Это связано с тем, что `plot2d` может принимать еще и дополнительные аргументы — в таком случае они перечисляются следом за таким списком, что исключает всякую путаницу.

По умолчанию, построением графиков занимается `gnuplot`, но кроме него есть разрабатываемый вместе с *Maxima* и идущий в ее же пакете `openmath`. Gnuplot необходимо установить (вручную либо автоматически — как зависимость *Maxima*) из пакета `gnuplot-nox`, либо просто `gnuplot`, а для работы `openmath` нужен командный интерпретатор `wish`, входящий обычно в пакет `tk`; и, начиная с версии 5.10.0, еще и пакет *xMaxima*.

6.2. Тригонометрические функции

Синусоиду $\sin(x)$ полезно сравнить с аналогичными ей синусоидами $\sin(2x)$ и $2\sin(x)$ (рис. 14).

Можно даже не переживать от того, что встроенный формат отошлёт к функции wxplot2d интерфейса *wxMaxima*, в котором масштабы по разным осям будут различными, так как в данном случае различие масштабов некритично: по оси абсцисс отложены какие-то углы, а по оси ординат - синусы этих углов.

```
wxplot2d([sin(2*x),sin(x),2*sin(x)], [x,-%pi,%pi], [y,-3,3],
 [gnuplot_preamble, "set grid;"])$
```

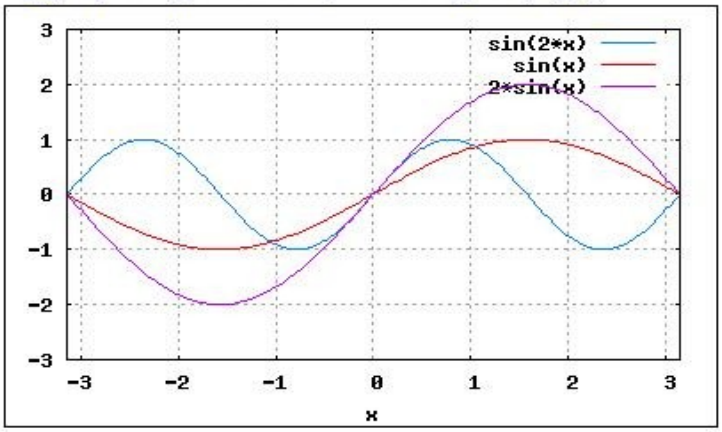


Рис. 14. Синусоиды sin(x), sin(2x) и 2sin(x)

Более того, интуитивно должно быть более преемлемо и, возможно даже, математически правильнее использовать для этих графиков именно разный масштаб. Дополнительная опция “set grid;”, вполне возможная и во встроенном формате gnuplot интерфейса wxMaxima позволяет прорисовать на графике сетку.

Функцию cos(x) полезно рассмотреть совместно с графиками x и x*cos(x), что называется "два в одном". Можно одновременно рассматривать не только косинусоиду, но и то, что произойдет, если перемножить функцию y=cos(x) с функцией y=x.

```
plot2d([x,-x,cos(x),x*cos(x)], [x,-4*%pi,4*%pi], [y,-12,12],
 [plot_format, gnuplot],
 [gnuplot_preamble, "set size ratio 1; set zeroaxis; set grid;"])$
```

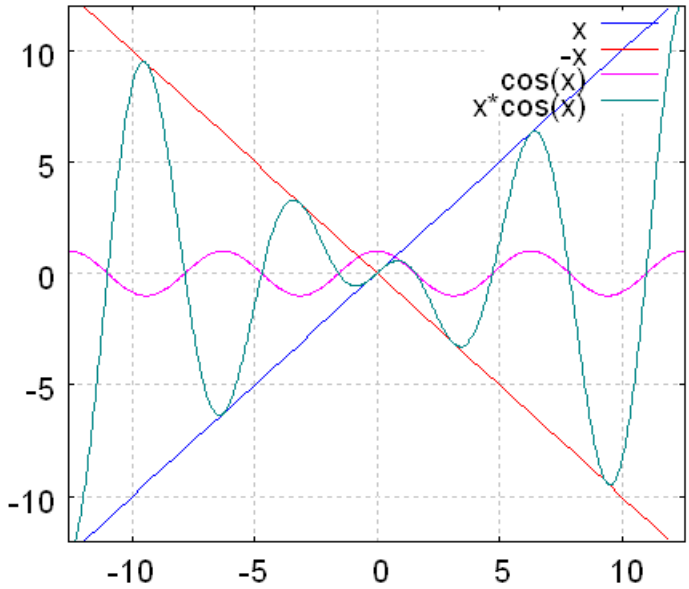


Рис. 15. Графики y=x, y=–x, y=cos(x), y=x*cos(x)

Конечно же, для этого графика лучше использовать формат `gnuplot`, позволяющий выравнивать масштабы по разным осям координат (рис. 15), что нам в данном случае необходимо для правильной интерпретации графика $y=x$. Кроме того, полезно построить другую комбинацию этих функций вида $\frac{x}{\cos(x)}$, где cos(x) находится в знаменателе (рис. 16, рис. 17).

График этой комбинации функций получился на рис. 16 неожиданным и таким интересным из-за того, что функция cos(x) имеет на заданном промежутке точки разрыва.

```
wxplot2d([x/cos(x)], [x,-2*%pi,2*%pi])$
```

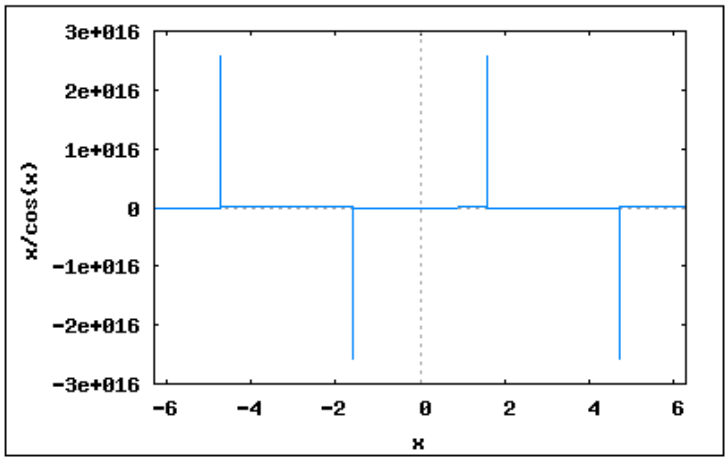


Рис. 16. Точки разрыва графика $\frac{x}{\cos(x)}$

Поскольку в команде «нарисовать график» мы не указали интервал изменения переменной y, а по умолчанию *Maxima* производит вычисления с точностью 16 знаков, то *Максима* выбрала масштаб $1*10^{16}$ и пытается втиснуть график от –3 единицы масштаба до +3 единицы выбранного масштаба. Однако, если мы укажем относительно небольшой интервал изменения переменной y, например, от –10 до +10, вид графика окажется более понятным и даже логичным (рис. 17).

```
wxplot2d([x,-x,x/cos(x)], [x,-2*%pi,2*%pi], [y,-10,10])$
```

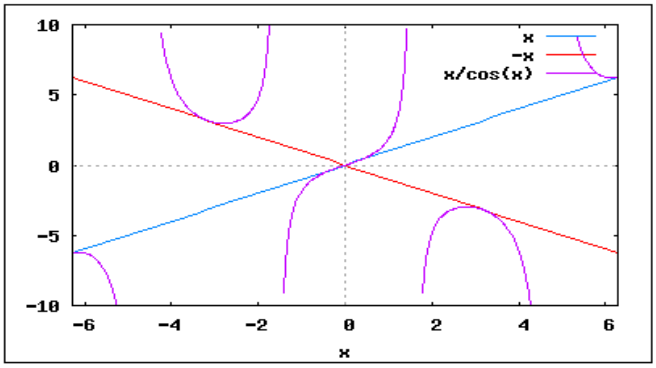


Рис. 17. Графики функций $y=x$, $y=-x$, $y=\frac{x}{\cos(x)}$

В тех точках, где cos(x)=1, отношение x/cos(x) оказывается равно x (и мы попадаем на график $y=x$, $y=-x$); в других точках, где cos(x)=0, функция x/cos(x)

терпит разрывы, одна ветвь уходит на +∞, другая возвращается с –∞.

Тангенс равен нулю в точке x=0 и обращается в бесконечность в точках $x = \frac{\pi}{2} \pm \pi$, соответственно функция **котангенс** в точке x=0 имеет разрыв, а в точках $x = \frac{\pi}{2} \pm \pi$ обращается в нуль, так как ctg(x)=1/tg(x) (рис. 18).

```
wxplot2d([tan(x),cot(x)], [x,-3*%pi/2,3*%pi/2], [y,-4,4],
 [gnuplot_preamble, "set grid;"])$
The number 0.0 isn't in the domain of cot
```

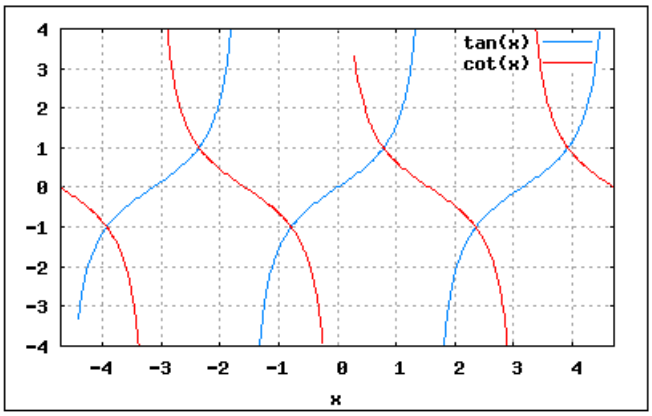


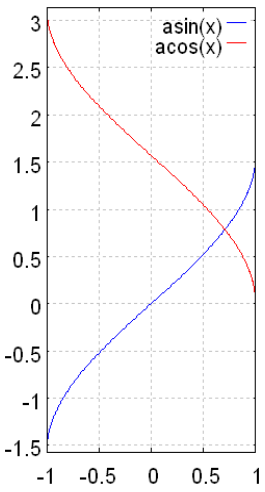
Рис. 18. Графики y=tg(x), y=ctg(x)

Максима отмечает, что в точке x=0 котангенс не определен (*The number 0.0 isn't in the domain of cot*), однако про аналогичные точки $x = \frac{\pi}{2} \pm \pi$, в которых тангенс терпит разрыв, она умалчивает (прогуливаясь по множеству точек на оси x, *Максима* в точки разрыва не попадает ввиду специфичности численных значений этих точек).

6.3. Обратные тригонометрические функции

Значения обратных тригонометрических функций Максима рисует только в первой четверти – функции должны быть однозначными.

```
plot2d([asin(x),acos(x)], [x,-1,1], [y,-%pi/2,%pi],
 [plot_format, gnuplot],
 [gnuplot_preamble, "set grid;"])$
```




```
plot2d([atan(x),acot(x)], [x,-5,5], [y,-%pi/2,%pi/2],
 [plot_format, gnuplot],
 [gnuplot_preamble, "set grid;"])$
The number 0.0 isn't in the domain of acot
```

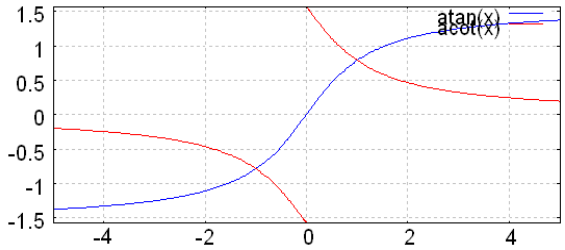


Рис. 19. Графики обратных тригонометрических функций

6.4. Экспонента и логарифм

Экспоненту и логарифм натуральный можно нарисовать и во встроенном формате интерфейса *wxMaxima*.

```
wxplot2d([exp(x),exp(-x)], [x,-3,3], [y,0,15],
 [gnuplot_preamble, "set grid;"])$
```

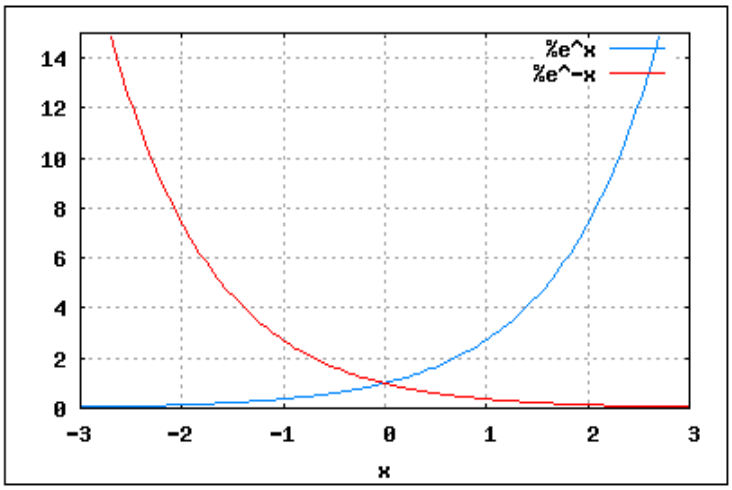


Рис. 20. Графики двух экспонент

На рис. 20 одна экспонента растет, другая – экспоненциально – убывает, логарифмы натуральные этих экспонент приведены на рис. 21.

```
wxplot2d([-log(x),log(x)], [x,0,15], [y,-3,3],
 [gnuplot_preamble, "set grid;"])$
```

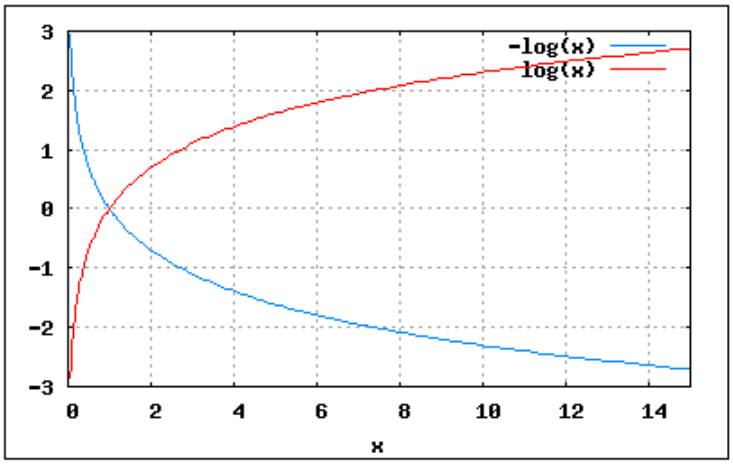


Рис. 21. Графики логарифма натурального

Рассматривая графики логарифмов, многие отметят, что они очень похожи на две экспоненты с предыдущего графика, но только их ветви смотрят в другую сторону.

Конечно же, если график с экспонентами повернуть по часовой стрелке на 90º и выровнять соответствующие масштабы этих двух графиков по осям *y* и *x*, то графики экспонент и логарифмов совпадут, так как это взаимообратные функции.

6.5. Графики параметрически заданных функций

Для построения графика используется список с ключевым словом `parametric`. В качестве наиболее простого примера обычно приводят параметрическую окружность (рис. 22).

```
plot2d([['parametric, cos(t), sin(t), [t, -%pi, %pi], [nticks, 300]]],
 [x,-1.2,1.2],[y,-1.2,1.2],[plot_format, gnuplot],
 [gnuplot_preamble, "set size ratio 1; set grid;"])$
```

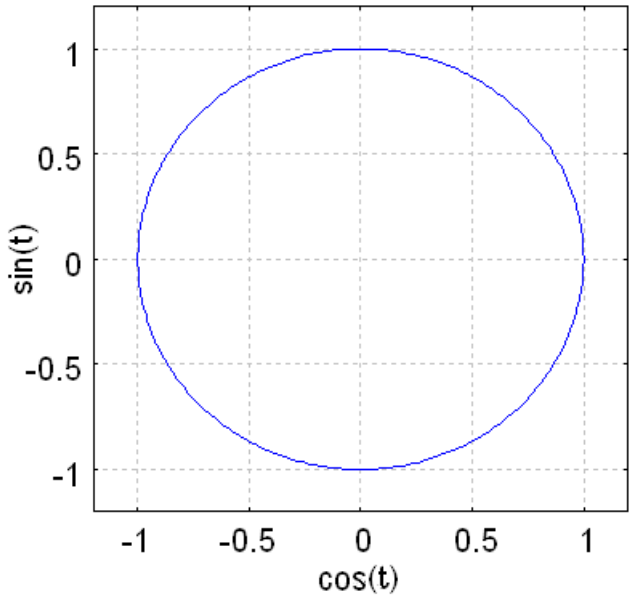


Рис. 22. График параметрической окружности: $y = \cos(x)$, $x = \sin(t)$

В общем случае для получения графика параметрической кривой записывается команда `plot2d([parametric, x-выражение, y-выражение, [переменная, начало, конец], [nticks, количество]])`, где `x-выражение` и `y-выражение` задают зависимость координат от параметра, то есть, по сути, это две функции вида $x=x(t)$, $y=y(t)$, где t — переменная параметризации. Эта же переменная должна фигурировать в следующем аргументе–списке, а параметры `начало`, `конец`, как и в двух других рассмотренных случаях, задают отрезок, в пределах которого этот параметр будет изменяться. Последний аргумент–список, с ключевым словом `nticks`, задает количество кусочков, на которые будет разбит интервал изменения параметра при построении графика.

Интерфейс *wxMaxima* достаточно удобен и не требует умения запоминать и безошибочно вводить длинный текст–вызов функции plot2d со всеми её параметрами. Достаточно лишь заполнить две вспомогательные формы для построения параметрического графика. После запуска *wxMaxima* и щелчка по кнопке График 2D... появляется окно диалога *График 2D*. А после щелчка по кнопке Дополнительно на этой форме появляется второе окно *Параметрический график*.

Для получения графика теперь достаточно лишь ответить на вопросы этих двух форм.

На рис. 23 приведены заполненные информацией окна–формы для вывода графика параметрической окружности, изображенной на рис. 22.

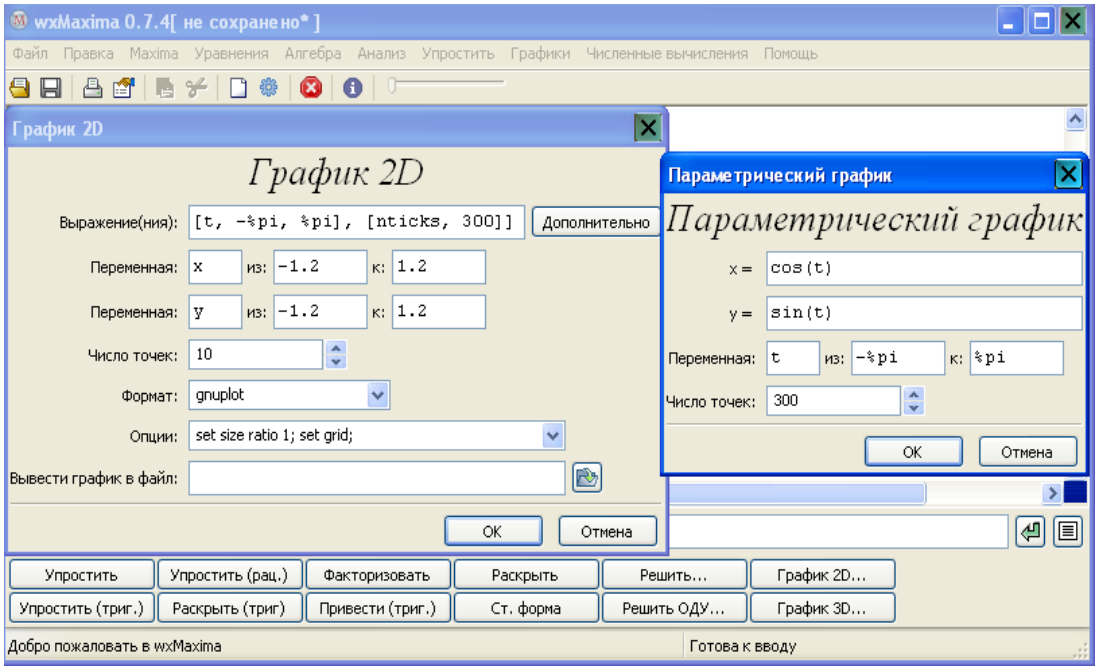


Рис. 23. Окна–формы для задания вида параметрического графика

Формы существенно упрощают технологию ввода команд для рисования параметрических графиков и позволяют не запоминать весьма сложный синтаксис функции plot2d. Так, например, на рис. 23 приведена только незначительная часть информации, на основе которой сформирована команда для Максимы в виде 3-х строчек текста, приведенная на рис. 22. Текст для команды, позволяющей совместить на одном рисунке два параметрических графика, еще более длинен и более труден для точного набора, но окна–формы *График 2D* и *Параметрический график* позволяют достаточно просто совместить на одном рисунке два параметрических графика. С этой целью сначала нужно построить первый график. Затем щелкнуть в графической части окна интерфейса *wxMaxima* на тексте–вызове `plot2d(['param...` первого графика и после выделения текста–вызова щелкнуть сначала по кнопке График 2D..., а затем по кнопке Дополнительно. После заполнения сведений о втором графике на одном рисунке появятся два графика. Например, такие (рис. 24).

```
wxplot2d([['parametric, cos(t), sin(t), [t, -%pi, %pi], [nticks, 300]],
          ['parametric, cos(t), sin(2*t),[t, -%pi, %pi], [nticks, 300]]],
          [x,-2.5,2.5], [y,-1.5,1.5], [gnuplot_preamble, "set grid;"])$
```

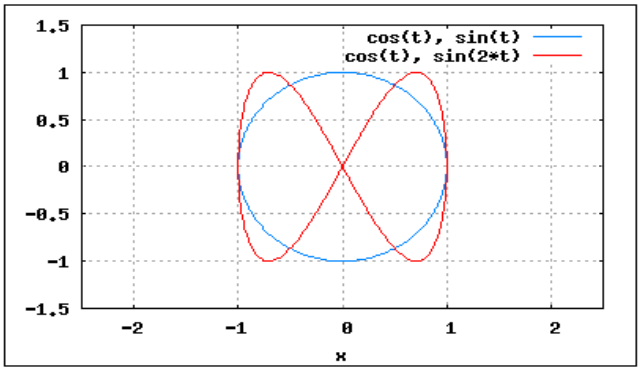


Рис. 24. Две параметрические функции, совмещенные на одном графике

Отметим, какие действия при этом осуществляет *Максима*. При выделении текста–вызова функции часть выделенного текста сначала попадает в буфер обмена, а после вызова окна–формы *График 2D* эта информация о виде первой функции и диапазоне изменения переменных из буфера обмена попадает в поле Выражение(ния). Обратите внимание на то, что на предыдущем рисунке действительно поле ввода Выражение(ния) не пустое и информация о первой функции там имеется.

Заметим попутно, что физический смысл параметрической функции состоит в визуализации процесса перемещения светящейся точки (луча) по экрану осциллографа. Если угол между положительным направлением оси oX и радиус–вектором точки обозначить через *t,* расстояние от точки до начала координат обозначить через *R*, то из формул геометрии следует, что абсцисса и ордината этой точки равны $x=R\cos(t)$ и $y=R\sin(t)$, соответственно. Если же абсцисса и ордината точки изменяются, например, по закону $x=\cos(t)$ и $y=\sin(t)$ в то время как угол t изменяется от $-\pi$ до π, то в это время точка совершает полный оборот против часовой стрелки из крайне левой точки окружности единичного радиуса в ту же самую точку. Такие устойчивые картины на экране осциллографа называются фигурами Лиссажу.

Фигуры Лиссажу́ — это замкнутые траектории, прочерчиваемые точкой, совершающей одновременно два гармонических колебания в двух взаимно перпендикулярных направлениях. Фигуры Лиссажу вписываются в прямоугольник, центр которого совпадает с началом координат, а стороны параллельны осям координат и расположены по обе стороны от них на расстояниях, равных амплитудам колебаний. Вид фигур Лиссажу зависит от соотношения между периодами (частотами), фазами и амплитудами обоих колебаний. В случае рационального отношения частот этих колебаний траектории замкнуты и именно они называются фигурами Лиссажу. На следующем рис. 25 две другие фигуры Лиссажу совмещены *Максимой* на одном и том же графике.

Отметим еще раз, что редактирование текста команды, вызывающей функцию wxplot2d, удобнее всего осуществлять через «Многостраничный ввод» в окне Ввод wxMaxima и пользоваться свойством *Максимы* запоминать исполненные ею команды. Дело в том, что при наборе длинной команды можно ошибиться. Узнав о характере ошибки и вызвав предыдущий текст (в котором имеется ошибка), необходимо будет только лишь исправить допущенные опечатки.

Чтобы "вспомнить" текст исполненной команды, используйте клавишу клавиатуры "Стрелка вверх". Чтобы запомненный текст попал в окно *Ввод wxMaxima*, сначала его нужно выделить (щелчком мыши) в графической части интерфейса *wxMaxima*, а затем тут же щелкнуть мышью по кнопке  «многострочный ввод» интерфейса *wxMaxima*. Чтобы текст ранее исполненной команды попал в окно ВВОД, служит клавиша F5, а также пункт «Выделение во ввод» меню Правка.

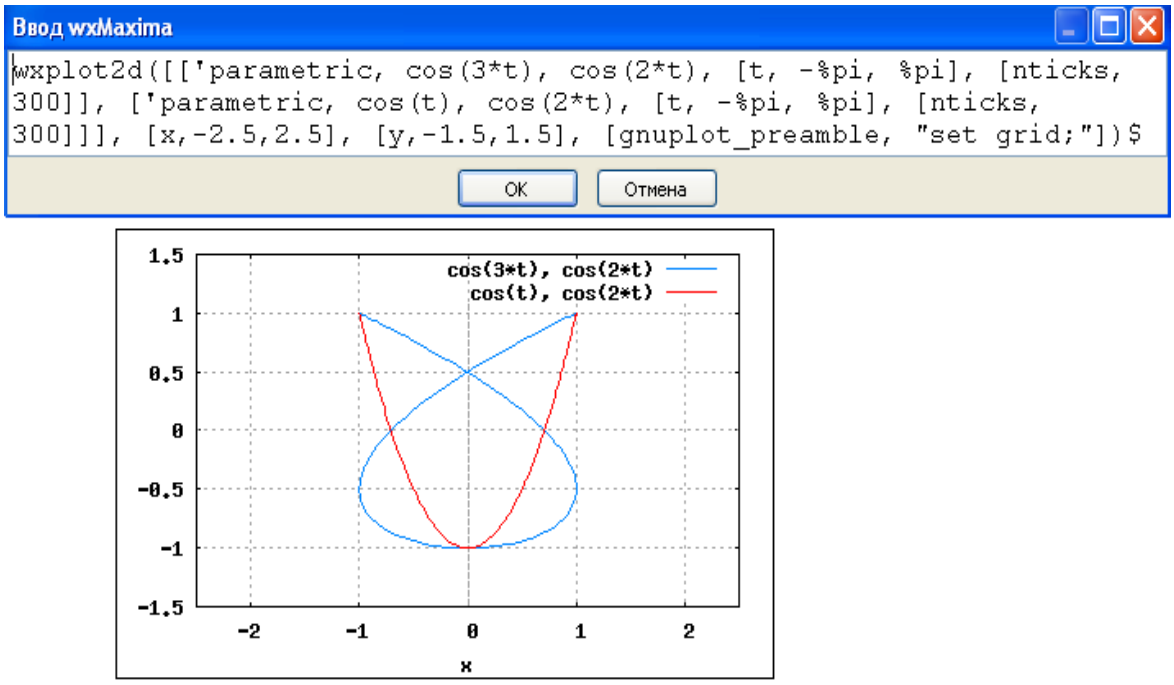


Рис. 25. Две фигуры Лиссажу на одном графике

6.6. Дискретный график

Maxima может нарисовать график по координатам отдельных дискретных точек. Для этого ей нужны два списка: один — для значений абсцисс дискретных точек, второй — для значений ординат этих точек.

Например, следующие две пары списков координат точек позволяют нарисовать шестиугольник и пятиконечную звезду (рис. 26).

```
xx:[1,2,4,5,4,2,1]$
yy:[0,1.3,1.3,0,-1.3,-1.3,0]$
  wxplot2d([discrete,xx,yy])$
```

```
uu:[-2.5,2.5,-2,0,2,-2.5]$
vv:[1,1,-2.2,2.2,-2.2,1]$
  wxplot2d([discrete,uu,vv])$
```

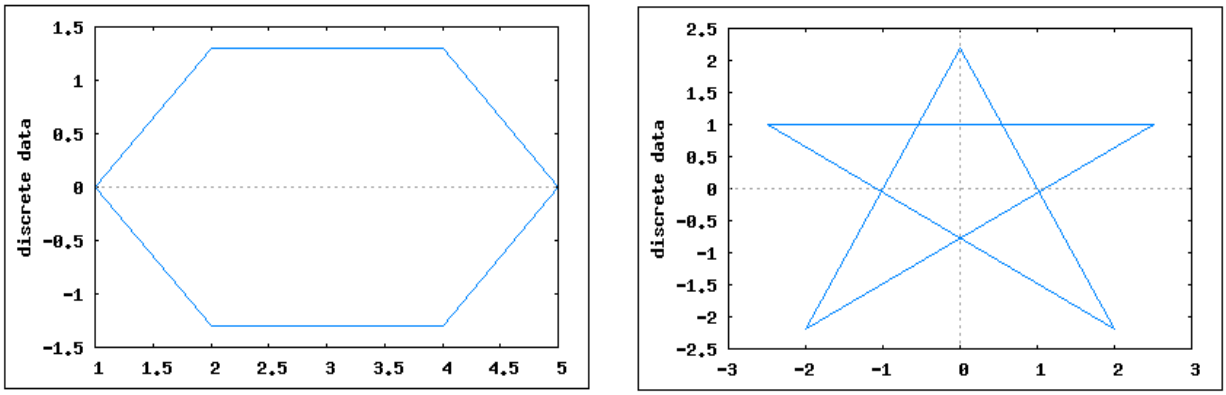


Рис. 26. Шестиугольник и звезда нарисованы по координатам вершин

Соответствующие массивы значений (координаты *x* точек, и координаты *y* точек) могут быть введены через вспомогательные окна–формы *График 2D* и *Дискретный график* (последнее вызывается щелчком по кнопке Дополнительно).

На рис. 27 представлен интерфейс для ввода координат чисел для получения дискретного графика в виде пятиконечной звезды, приведенной на рис. 27.

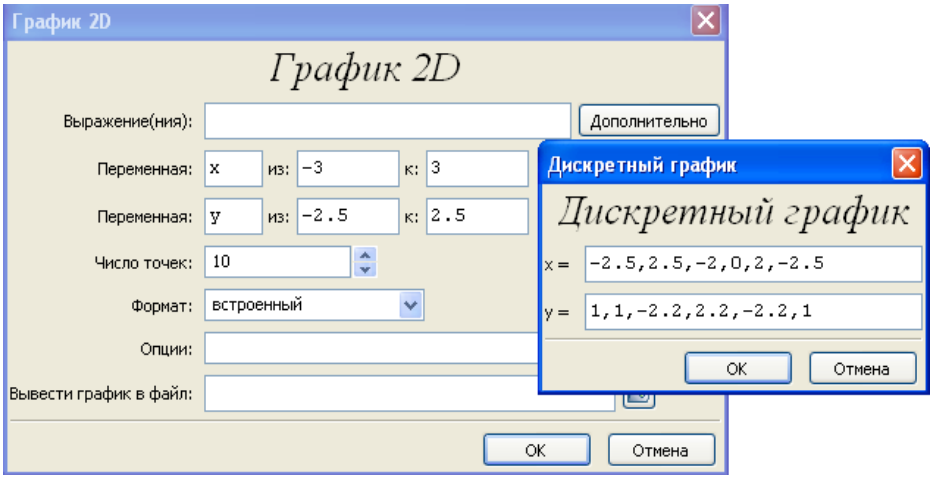


Рис. 27. Интерфейс ввода координат точек для дискретного графика

Массивы чисел, если необходимо, Maxima может генерировать автоматически. Для этой цели в Maxima имеется функция makelist(*expr*, *i*, *i*_0, *i*_1); генерирующая список значений, для которого *expr* – это вычисляемое выражение, зависящее от переменной *i*, принимающей значения от *i*_0, до *i*_1.

В следующем примере (рис. 27) массив чисел [1, 2, ..., 15] создан автоматически функцией makelist(x,x,1,15) (где в качестве выражения взята величина *x*). Этот массив далее используется в качестве координат *x* точек, массив data значений координат *y* точек также получен автоматически, но как массив случайных чисел, с этой целью использовалась функция random(*x*);

```
(%i17)  data:makelist(random(5*x),x,1,15);
(%o17)  [1,5,4,7,2,28,18,20,21,22,41,16,7,14,56]

(%i18)  wxplot2d([discrete,makelist(x,x,1,15),data])$
```

(%t18)

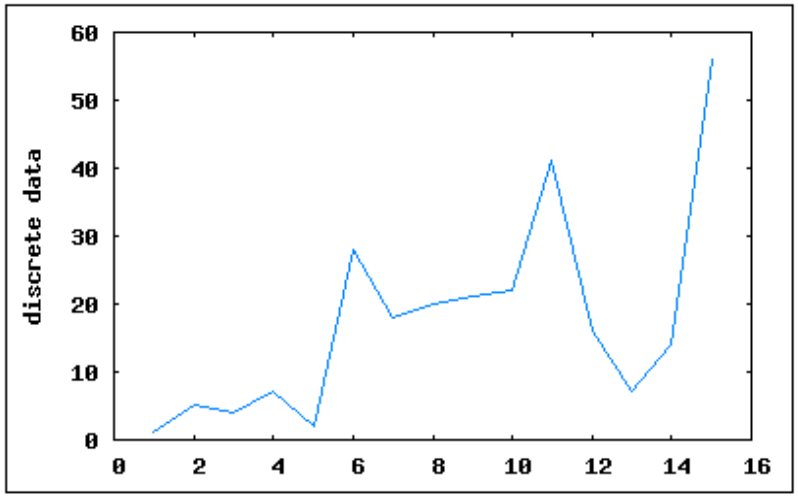


Рис. 28. Дискретный график координат случайных чисел

В следующем примере мы поступили наоборот: абсциссы точек графика рассчитали с помощью функции makelist();, а абсциссы оставили нерассчитанными, но описали функцию $f(x)$, которая будет рассчитывать эти значения.

```
(%i4) x:makelist(z,z,-2,6);
(%o4) [-2,-1,0,1,2,3,4,5,6]
(%i5) f(x):=x^3/6-x^2;
```

$$(\%o5)\quad f(x):=\frac{x^3}{6}-x^2$$

По рассчитанным координатам был построен дискретный график, на котором нанесена сетка, а точки не соединены линией.

Команда получилась такой: wxplot2d([discrete, x, f(x)], [gnuplot_preamble, "set grid;"], [style, points])$

```
wxplot2d([discrete,x,f(x)],
 [gnuplot_preamble, "set grid;"],[style, points])$
```

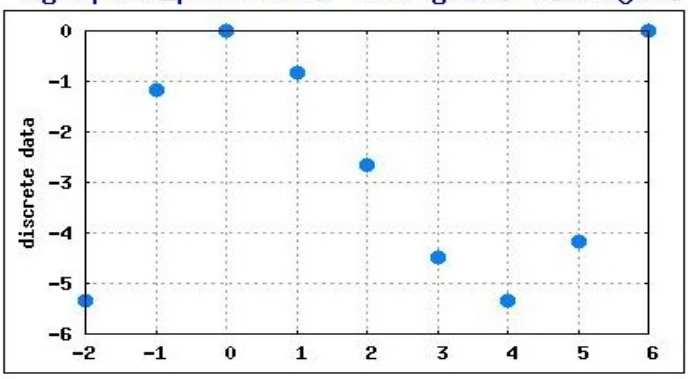


Рис. 29. Дискретный график, точки не соединены линиями

На рис. 30 совмещены два параметрических графика и три дискретных графика. Графики нарисованы для того, чтобы пояснить геометрический смысл параметра параметрической функции как величины полярного угла.

Первый график — это параметрическая окружность единичного радиуса: x=cos(t), y=sin(t), параметр t которой (он же угол в полярной системе координат) изменяется от $-\pi$ до π, пятый — discrete5 — график: [discrete, [[.707,.707]]],[style, points] – это отдельная "светящаяся" точка с координатами $(\frac{\sqrt{2}}{2},\frac{\sqrt{2}}{2})$, в эту "светящуюся" точку направлен радиус–вектор discrete4 (он же гипотенуза прямоугольного треугольника, при ней два катета) – помечен как четвертый график – [discrete, [[0,0],[.707,0],[.707,.707],[0,0]]], величина полярного угла на рисунке показана для гипотенузы (т. е. радиус–вектора) с помощью параметрической дуги: x=cos(t)/2, y=sin(t)/2, для которой угол t параметрически изменяется от 0 до π/4 – второй график, на конце дуги имеется стрелка (треугольник), которую реализует discrete3 – третий график [discrete, [[.35,.35],[.36,.31],[.4,.34],[.35,.35]]].

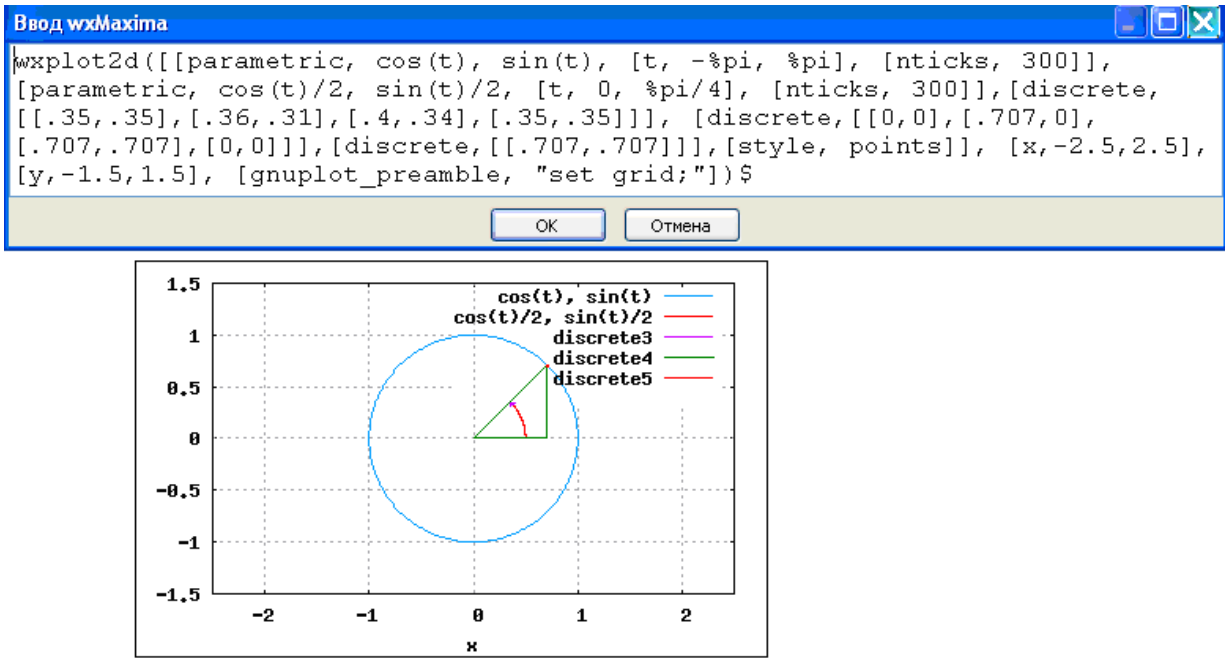


Рис. 30. Графики, поясняющие смысл параметра *t* параметрической функции

6.7. Графики в полярной системе координат

Если использовать две окружности с одинаковыми радиусами и вращать одну вокруг другой (рис. 31), то получится *кардиоида* (греч. кардиа – сердце) — по мнению математиков, получаемая кривая отдаленно напоминает сердце (рис. 32).

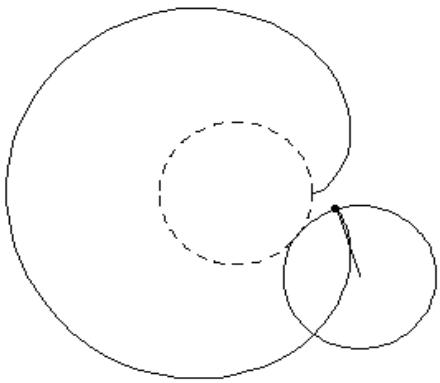


Рис. 31. Кардиоиду рисует выделенная точка окружности движущейся по другой окружности

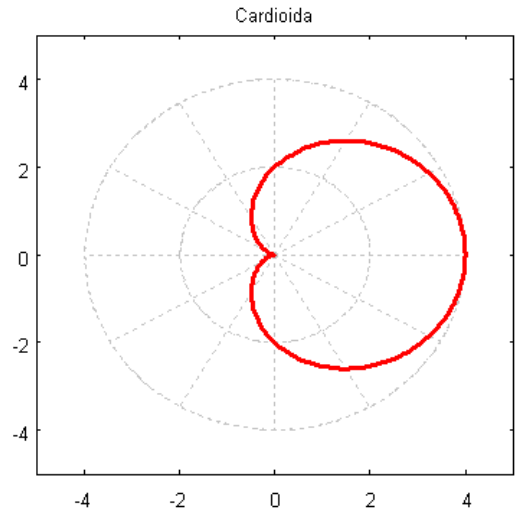


Рис. 32. График кардиоиды в полярной системе координат

В прямоугольной декартовой системе координат уравнение кардиоиды имеет сложный вид $(x^2 + y^2 - ax)^2 = a^2(x^2 + y^2)$. Но в полярной системе координат ($x = \rho\cos(t)$, $y = \rho\sin(t)$) уравнение кардиоиды имеет простой вид $\rho = a + a\cos(t)$,

где ρ – расстояние от точки кривой до начала координат, t– полярный угол, a – диаметр окружности.

Команда для рисования графика получается достаточно сложной, поэтому мы приводим ее отдельно (рис. 33).

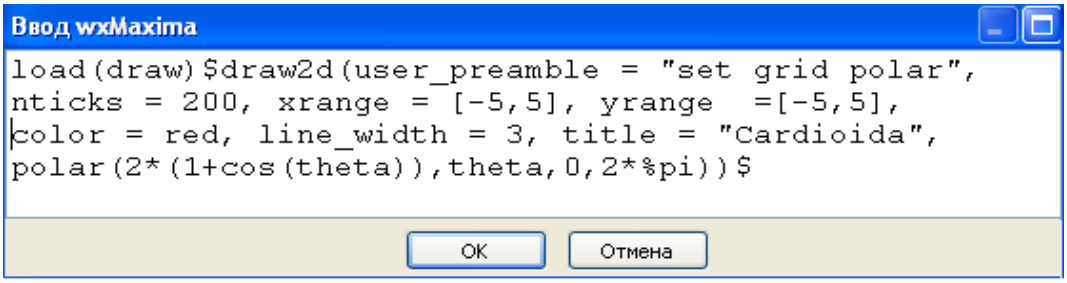


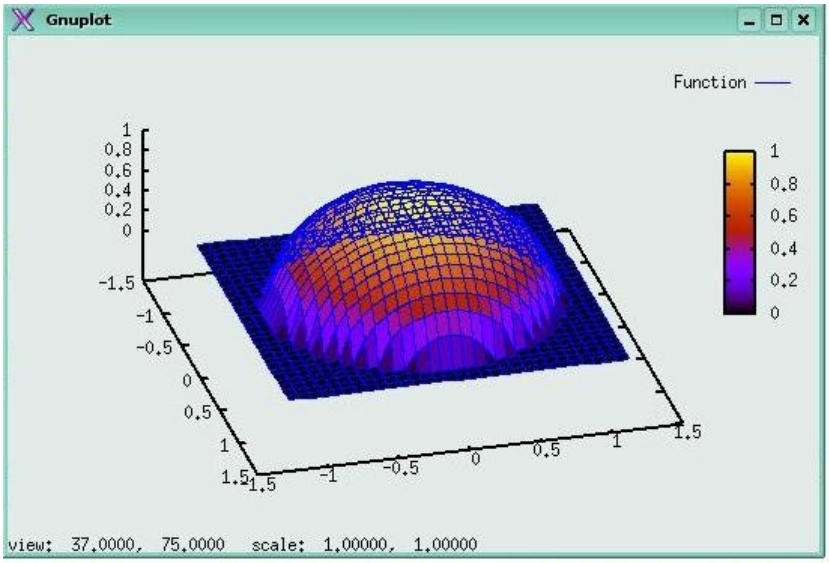
Рис. 33. Команды для *Maxima*, для вывода кардиоиды в полярной системе координат

Отметим, что в Maxima графики в полярной системе координат рисует функция draw2d(); но, прежде чем пользоваться этой функцией, нужно попросить Maxima дополнительно загрузить этот модуль оператором load(draw).

Непосредственно построением графика занимается функция polar(ρ,theta, theta_0, theta_M). Функция draw2d(); использует эту функцию (см. рис. 32).

7. Трехмерные графики

Многие трехмерные графики могут быть построены естественным образом в привычной прямоугольной декартовой системе координат, как поверхность вида $z = f(x,y)$ (рис. 33–37).




```
plot3d([x,y,sqrt(1-x^2-y^2)], [x,-1.2,1.2], [y,-1.2,1.2]);
```

Рис. 34. График функции $z = \pm c\sqrt{1-\frac{x^2}{a^2}-\frac{y^2}{b^2}}$ (эллипсоид)

Эллипсоид. Каноническое уравнение эллипсоида в декартовой системе координат имеет вид

$\frac{x^2}{a^2}+\frac{y^2}{b^2}+\frac{z^2}{c^2}=1$ откуда находим $z=\pm c\sqrt{1-\frac{x^2}{a^2}-\frac{y^2}{b^2}}$.

Поскольку построение нескольких поверхностей на одном графике Максима не практикует, а функция должна быть однозначной, мы можем построить только верхнюю часть эллипсоида $z>0$ или его нижнюю часть $z<0$.

Отметим, что в данном конкретном случае полуоси эллипсоида совпадают, и фактически нарисована верхняя часть сферы, но если мы попытаемся взять другие численные значения для полуосей a, b, c, все равно график визуально окажется таким же, потому что Максима выберет по осям координат другой масштаб и различие окажется незаметным.

Кроме того отметим, что окно Gnuplot, содержащее график, легко масштабируется и его можно сделать любой ширины и высоты, а изображенную поверхность можно поворачивать и разглядывать и с любого бока, и сверху, и снизу.

Однополостный гиперболоид.. Каноническое уравнение однополостного гиперболоида в декартовой системе координат имеет вид

$\frac{x^2}{a^2}+\frac{y^2}{b^2}-\frac{z^2}{c^2}=1$ откуда следует $z=\pm c\sqrt{\frac{x^2}{a^2}+\frac{y^2}{b^2}-1}$.

График нижней части для $z<0$ однополостного гиперболоида приведен на рис. 35. Верхняя часть представляет собой зеркальное отражение нарисованной части относительно горизонтальной плоскости $z=0$ (нечто вроде воронки с отверстием).

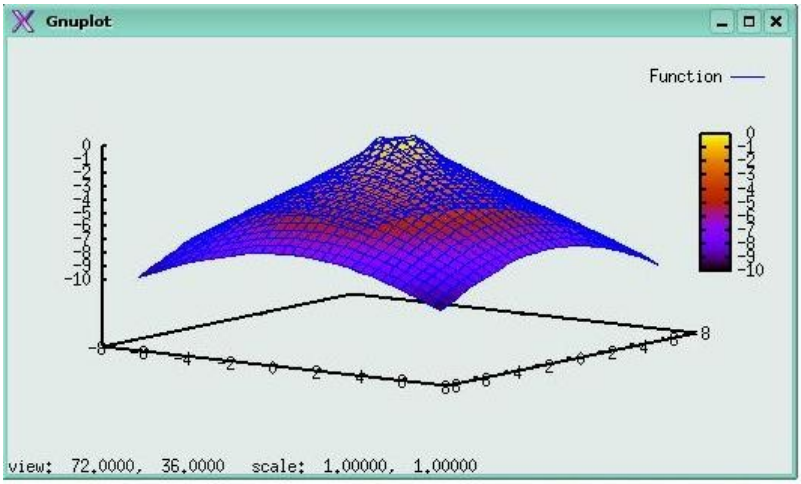


Рис. 35. График функции $z=-c\sqrt{\frac{x^2}{a^2}+\frac{y^2}{b^2}-1}$ (однополостный гиперболоид)

Двуполостный гиперболоид.. Каноническое уравнение двуполостного гиперболоида в декартовой системе координат имеет вид

$\frac{x^2}{a^2}-\frac{y^2}{b^2}-\frac{z^2}{c^2}=1$ откуда получаем $z=\pm c\sqrt{\frac{x^2}{a^2}-\frac{y^2}{b^2}-1}$.

График нижней части $z<0$ двуполостного гиперболоида приведен на рис. 35. Две не связанные отдельные полости гиперболоида расположены слева и справа относительно середины рисунка. Чтобы представить как выглядят полости нужно на нарисованную часть мысленно положить сверху ее отражение относительно горизонтальной плоскости в вертикальном направлении – точно такую же, верхнюю часть (для $z>0$) двуполостного гиперболоида.

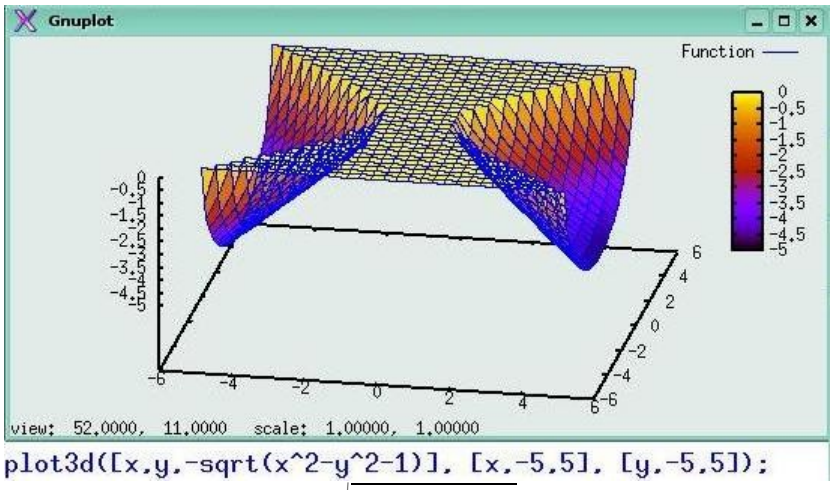


Рис. 36. График функции $z=-c\sqrt{\frac{x^2}{a^2}-\frac{y^2}{b^2}-1}$ (двуполостный гиперболоид)

Эллиптический параболоид приведен на рис. 37.

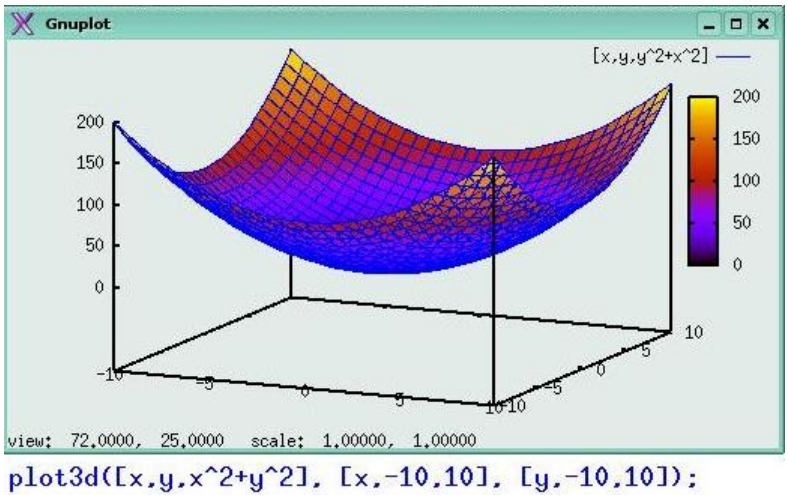


Рис. 37. График функции $z=+c\,(\frac{x^2}{a^2}+\frac{y^2}{b^2})$ (эллиптический параболоид)

Сечение параболоида горизонтальной плоскостью представляет собой эллипс, а сечение вертикальной плоскостью – параболу.

Гиперболический параболоид приведен на рис. 38. Поверхность имеет вид «седла» – прекрасную иллюстрацию отсутствия локального экстремума.

Отметим, что в вызове функции plot3d() в рассмотренных случаях мы использовали список вида [x,y,z(x,y)], но функция plot3d() допускает и более простой вызов вида plot3d(выражение, [переменная1, начало, конец],

[переменная2, начало, конец]); если выражение в явном виде зависит от переменной1 и переменной2. В этом случае удобно пользоваться также кнопкой График 3D... или пунктом меню Графики | График 3D.

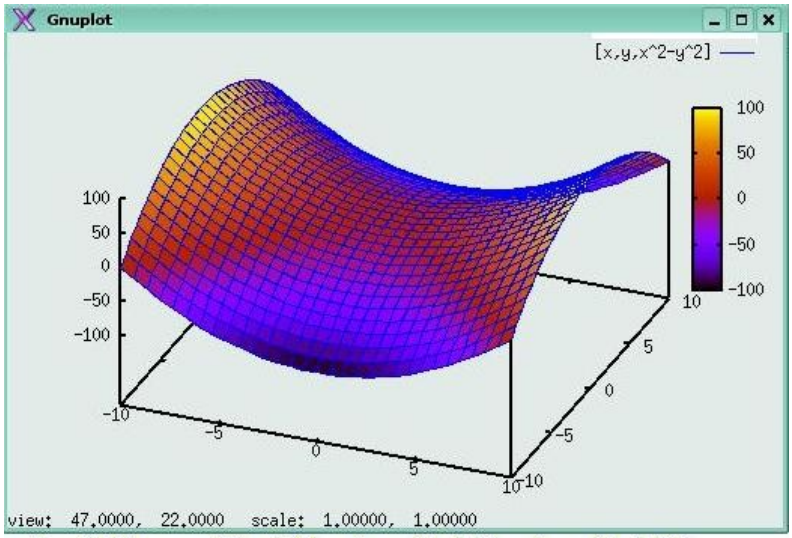


Рис. 38. График функции $z = +c\left(\frac{x^2}{a^2} - \frac{y^2}{b^2}\right)$ (гиперболический параболоид)

7.1. Трехмерные параметрические графики

Если координаты x, y, z зависят от параметров u, v: ($x = X(u,v)$, $y = Y(u,v)$, $z = Z(u,v)$), то в вызове функции plot3d() нужно использовать только списки вида plot3d([выражение1, выражение2, выражение3], [переменная1, начало, конец], [переменная2, начало, конец]), где выражения представляют собой упорядоченные зависимости $X(u,v)$, $Y(u,v)$, $Z(u,v)$.

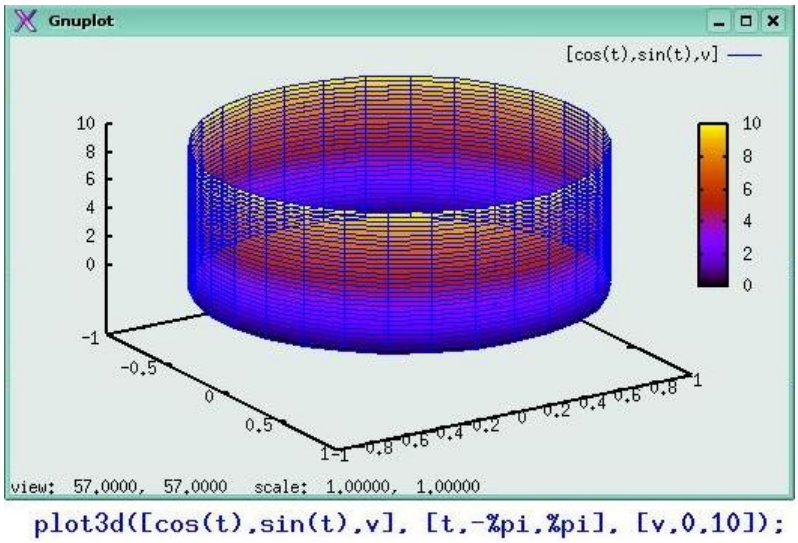


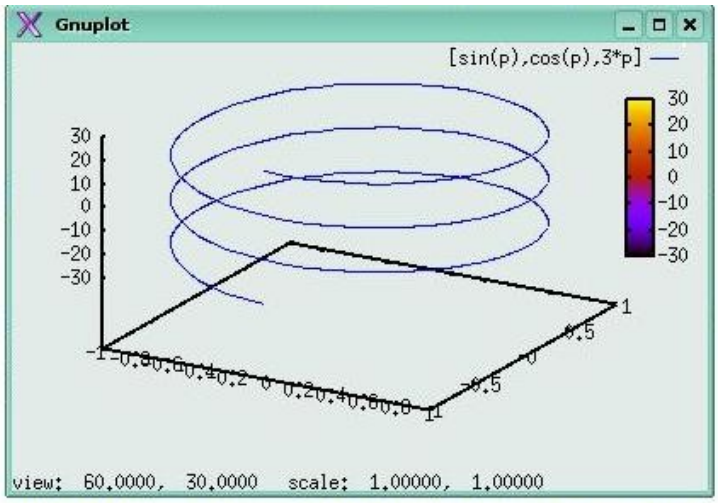
Рис. 39. Параметрический круговой цилиндр

Простейшей параметрической поверхностью после плоскости является круговой цилиндр вдоль оси Z. Однако, если мы попробует записать вызов функции `plot3d()` «слишком просто» в виде. `plot3d([sin(t),cos(t),z],`

`[t,-%pi,%pi],[z,-1.2,1.2]);`, то вместо цилиндра Максима нарисует пространственную окружность, а переменную z изменять от –1.2 до 1.2 не будет.

«Обмануть» Максиму в данном случае нетрудно, достаточно вместо стандартной декартовой переменной z ввести любую другую переменную. Хотя она нам как таковая и не нужна в данном случае, но мы вынуждены поддержать стандарт записи списка из трех выражений (рис. 38).

Наличие фиктивной переменной позволяет нарисовать пространственную кривую. Чтобы винтовая линия на графике (рис. 39) получилась достаточно гладкой, была добавлена опция [grid, 150,150], которая использовалась ранее для рисования сетки.




```
plot3d([sin(p),cos(p),3*p], [p,-3*%pi,3*%pi],
       [u,0,1],[grid,150,150]);
```

Рис. 40. Пространственная кривая

Отметим, что нарисовать красивый пространственный график не всегда просто, даже если использовать дополнительные опции (рис. 41).

```
(%i14) wxplot3d([cos(th)*sin(ph),sin(th)*sin(ph),cos(ph)],
       [th,0,2*%pi], [ph,-%pi/2,%pi/2],
       [gnuplot_preamble, "set mapping spherical"])$
```

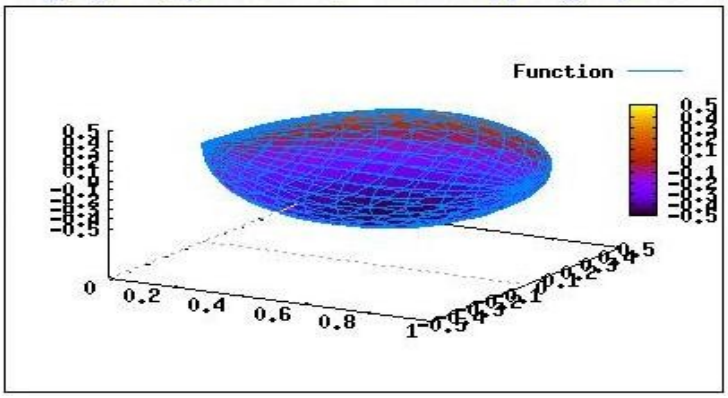


Рис. 41. Рисунок сферы во встроенном формате

Рисунок сферы на рис. 41 во встроенном формате плохо отождествляется со сферой, хотя если смотреть по масштабу сетки вдоль осей графиков, наличию опции "set mapping spherical" это должно быть сферой. Кроме того заметим, что во встроенном формате функция wxplot3d() результат выводит не в отдельное окно, а в обычное графическое окно интерфейса wxMaxima, что не позволяет поворачивать график и рассматривать его с разных сторон..

Заметим также, что кроме самой программы Maxima и ее модуля openmath на компьютере должен быть установлен также графический интерфейс xMaxima. Так, например, при отсутствии графического интерфейса xMaxima рисунок этой же самой сферы будет выведен в отдельное окно (как и в случае непараметрических графиков – окно Gnuplot) и может масштабироваться и рассматриваться с разных точек зрения, но не быть сферой (рис. 42).

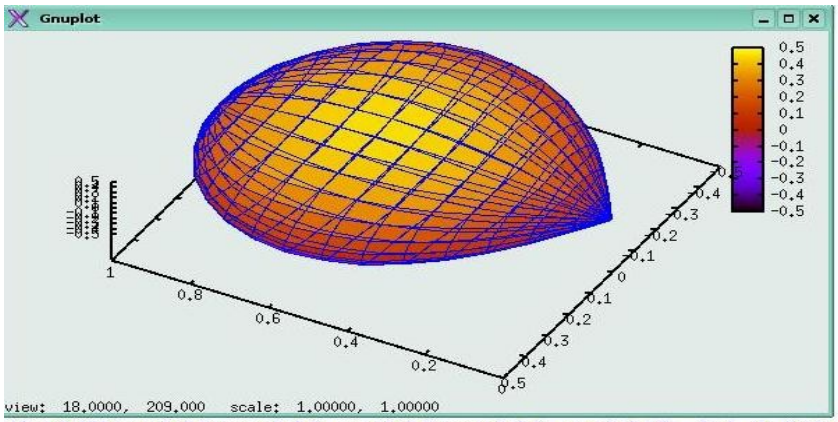


Рис. 42. Рисунок сферы в окне Gnuplot

При наличии установленного графического интерфейса xMaxima параметрические графики могут быть выведены в Окно 3D графики Вильяма Шелтера, имеющего свое собственное меню (рис. 43). На рис. 43 кроме непосредственно рисунка сферы 1) представлен также текст вызова функции plot3d() – 2), в котором присутствует опция [plot_format, openmath].

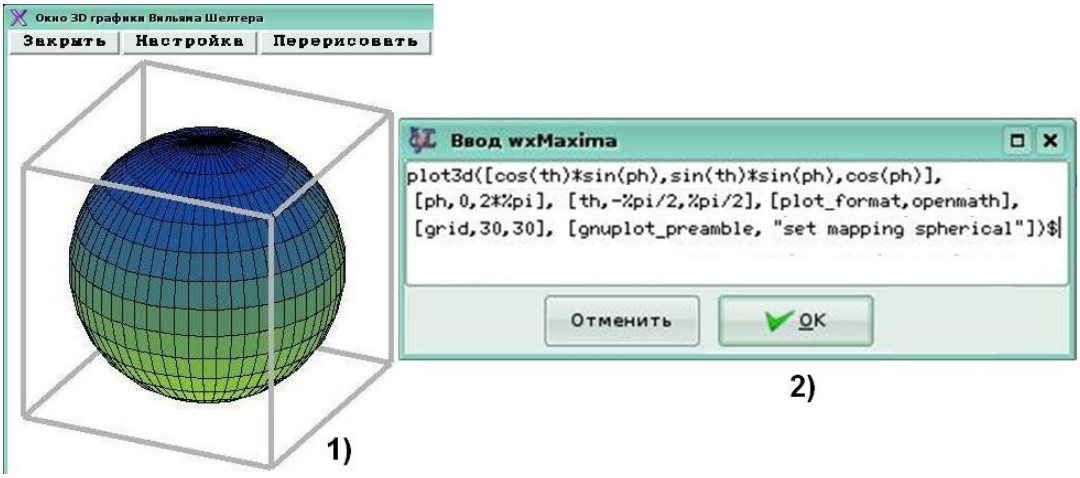


Рис. 43. Рисунок сферы в окне Вильяма Шелтера

Окно 3D графики Вильяма Шелтера позволяет не только изменять размеры самого окна, но и передвигать рисунок по окну (при нажатой правой кнопке мыши), масштабировать рисунок (щелчком мыши увеличивать размер, а щелчком мыши при нажатой кнопке *Shift* – уменьшать размеры рисунка), вращать рисунок и изменять его настройки. Если после этих манипуляций "Перерисовать" график, получится очень симпатично.

На следующем рисунке (рис. 44) представлен график односторонней поверхности – лист Мёбиуса,

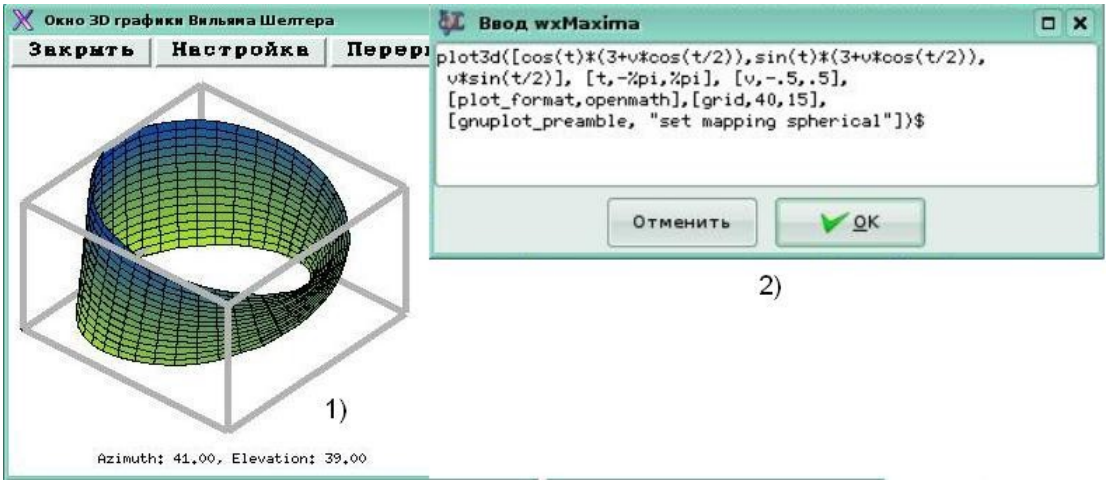


Рис. 44. Односторонняя поверхность, лист Мёбиуса

и приведен текст вызова функции plot3d(), которая выводит график в Окно 3D графики Вильяма Шелтера.

8. Решение уравнений

Уравнения и системы уравнений решаются в *Maxima* одной и той же функцией `solve`. Интерфейс *wxMaxima* позволяет упростить процедуру использования функции `solve`: после нажатия на кнопку `Решить...` появится дополнительное окно *Решить*, в котором конкретизируется и вид уравнения, и имя переменной, относительно которой нужно решить уравнение (рис. 45).

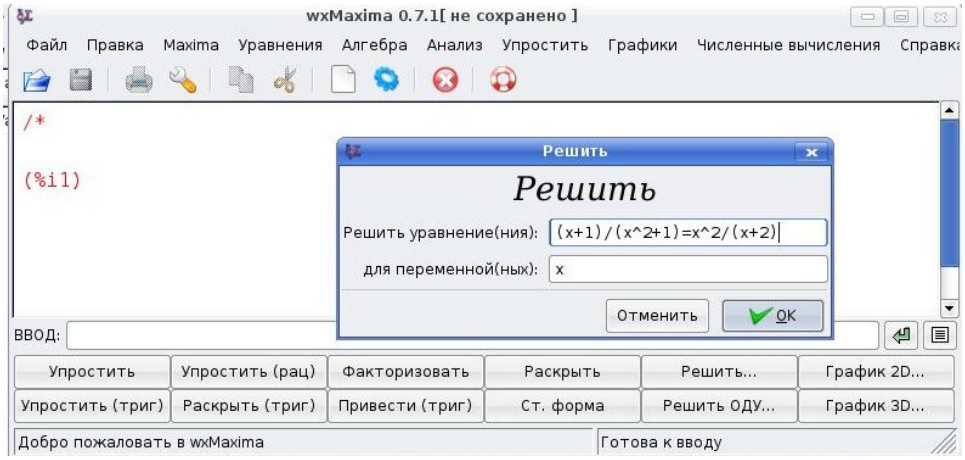


Рис. 45. Интерфейс *wxMaxima* для решения уравнения

Впрочем, интерфейс *wxMaxima* можно не использовать, можно просто написать команду для *Максимы* вида solve([(x+1)/(x^2+1)=x^2/(x+2)], [x]);

Но прежде чем рассмотреть результаты решения подробнее, нужно сказать пару слов о списках, или векторах, в *Maxima*; поскольку именно в виде списков `solve` возвращает корни, да и принимает параметры в случае решения системы уравнений, а не одного уравнения.

Синтаксис списков в *Maxima* прост; это перечисление элементов в квадратных скобках: `[элемент1, элемент2 ,…, элементN]`. Особенность — не в синтаксисе. Основное достоинство списков *Maxima* в том, что их элементами могут быть совершенно любые выражения: символы, арифметические выражения, вызовы функций, присвоения, уравнения, другие списки… Поэтому списки и во встроенных функциях широко применяются.

Функция `solve` в своем простейшем варианте, для решения одиночного уравнения, в качестве аргументов никаких списков не принимает (а принимает либо уравнение и символ, относительно которого его надо решать, либо только уравнение, если символ в нем всего один). А возвращает список, состоящий из всех корней заданного уравнения. В рассматриваемом случае получаем

```
(%i1) eq:(x+1)/(x^2+1)=x^2/(x+2);
```

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

```
(%i2) solve(eq);
```

$$(\%o2)\quad \left[x=-\frac{\sqrt{5}-1}{2},\ x=\frac{\sqrt{5}+1}{2},\ x=-\frac{\sqrt{7}\,\%i+1}{2},\ x=\frac{\sqrt{7}\,\%i-1}{2}\right]$$

Здесь мы сначала записали уравнение (команда %i1). Потом попросили решить его с помощью функции `solve` (команда %i2).

Как видно из полученного решения, функция `solve` нашла не только действительные корни, но и все комплексные корни уравнения и записала их в виде списка из четырех элементов.

К конкретному элементу полученного списка можно обратиться с помощью тех же квадратных скобок, указав в них номер элемента после имени списка. Вот так, например, мы можем осуществить проверку решения, подставив первый корень из выданного списка в исходное уравнение.

```
(%i3) eq,%[1];
```

$$(\%o3)\quad \frac{1-\frac{\sqrt{5}-1}{2}}{\frac{\left(\sqrt{5}-1\right)^2}{4}+1}=\frac{\left(\sqrt{5}-1\right)^2}{4\left(2-\frac{\sqrt{5}-1}{2}\right)}$$

```
(%i4) ratsimp(%);
```

$$(\%o4)\quad \frac{\sqrt{5}-3}{\sqrt{5}-5}=\frac{\sqrt{5}-3}{\sqrt{5}-5}$$

Здесь в равенство (eq), переданное в качестве дополнительного параметра функции ev, подставляется значение переменной, указанной в последнем (%) результате (%o2) в списке под номером один. Точно таким же образом можно обратиться ко второму, третьему и четвертому элементу списка:

```
(%i5)  eq,%o2[2],ratsimp;eq,%o2[3],ratsimp;eq,%o2[4],ratsimp;
                    sqrt(5) + 3     sqrt(5) + 3
(%o5)               ----------- = -----------
                    sqrt(5) + 5     sqrt(5) + 5
(%o6)                   - 1 = - 1
(%o7)                   - 1 = - 1
```

Вообще говоря, в качестве первого аргумента функции `solve` можно задавать не только уравнение, а вообще любое выражение. При этом «корни выражения» (не являющегося уравнением) ищутся в том самом смысле, в каком эта фраза понимается в математике: корни выражения — это те значения переменной, на которых выражение обращается в ноль. Возможность такой записи позволяет, к примеру, легко найти критические точки любой непрерывной функции (а заодно и вычислить значения функции в этих точках).

8.1. Поиск экстремума

В следующем примере мы сначала конкретизировали вид функции (команда %i1), затем попросили *Максиму* решить уравнение, указав в качестве аргумента функции `solve` выражение вида (diff(*f*, *x*)), которое представляет собой производную функции *f* по переменной *x*.

В качестве решения (%o2) уравнения $f'(x) = 0$ функция `solve` вывела список из двух значений $[\, x = -2,\ x = 0 \,]$. Команда (%i3) использована для того, чтобы вычислить значение функции в критических точках.

```
(%i1)  f:(1+x-x^2)/(1+x+x^2);
                      2
                   - x  + x + 1
(%o1)              ------------
                     2
                    x  + x + 1
(%i2)  solve(diff(f,x));
(%o2)  [ x = - 2 , x = 0 ]
(%i3)  f,%[1];f,%th(2)[2];
                    5
(%o3)               - -
                    3
(%o4)  1
```

В этом примере есть еще два важных момента. Первый — функция `%th()`. Она, как видно из контекста, вызывается как `%th(n)` и возвращает *n*-ю ячейку вывода. Ею удобно пользоваться, так же как и обозначениями `%` и `_`, чтобы не обращать внимания на номера ячеек результата, и кроме того, применимо в командных файлах Maxima, которые могут загружаться, в том числе прямо из интерактивной сессии (с помощью функции `load`). И второй момент: здесь

проиллюстрировано, что в Maxima операция индексирования списка доступна не только по отношению к именам переменных, но и к вызовам функций; иными словами, если функция возвращает список значений, мы можем выбрать одно конкретное из них, написав его номер в квадратных скобках прямо после вызова функции.

8.2. Решение систем уравнений

Попросим *Maxima* решить систему из двух уравнений: пусть требуется найти точки пересечения окржности $x^2 + y^2 = 2$ и прямой $x + y = 1$.

Для записи команды для *Maxima* можно использовать следующий вариант: solve([уравнение1, уравнение2, …], [переменная1, переменная2, …]), но можно использовать сокращенный вариант, аналогично варианту для одиночного уравнения.

Совместим графики окружности и прямой, чтобы убедиться в том, что решение существует. Из графика рис. 46 видим, что решением исследуемой системы уравнений являются координаты 2 точек.

```
wxplot2d([1-x, ['parametric, sqrt(2)*sin(t), sqrt(2)*cos(t),
  [t, -%pi, %pi], [nticks, 300]]], [x,-2,2], [y,-2,2],
  [plot_format, gnuplot], [gnuplot_preamble, "set size
  ratio 1; set zeroaxis;"])$
```

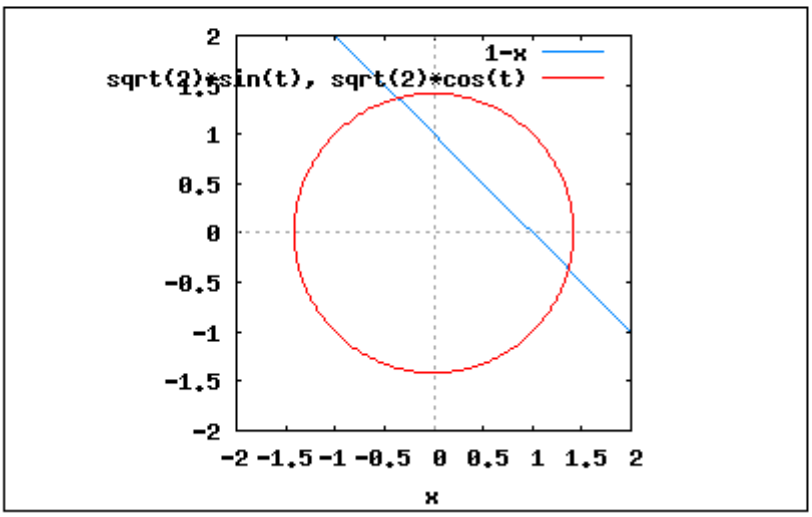


Рис. 46. Пересечение окружности и прямой

В нашем случае количество уравнений и количество неизвестных равны, поэтому список неизвестных можно не писать, а использовать обращение вида: `solve([уравнение1, уравнение2, …])`

Не следует, конечно, забывать, что квадратные скобки используются для указания списка, иначе Maxima проинтерпретирует вызов за вариант с одним уравнением.

Получим в итоге.

```
(%i1) solve([x^2+y^2=2,x+y=1]);
```

$$(\%o1)\quad \left[\left[y=-\frac{\sqrt{3}-1}{2},\ x=\frac{\sqrt{3}+1}{2}\right],\left[y=\frac{\sqrt{3}+1}{2},\ x=-\frac{\sqrt{3}-1}{2}\right]\right]$$

Здесь в качестве решения возвратился список из двух списков, каждый из которых соответствует одному решению системы (координатам точек пересечения). В качестве подстановок можно использовать как такие списки целиком (например, в данном контексте, %o1[1]), так и отдельные их элементы (например, %o1[1][1]).

Рассмотрим другой случай, когда уравнений меньше, чем неизвестных, `solve` поступит точно так же, как и в случае одного уравнения с несколькими символами: но все неуказанные в списке как переменные Maxima будет воспринимать как параметры:

```
(%i2) solve([x^2+y^2=a^2,x+y=2*a+1],[x,y]);
```

$$(\%o2)\quad \left[\left[x=-\frac{\sqrt{-2a^2-4a-1}-2a-1}{2},\ y=\frac{\sqrt{-2a^2-4a-1}+2a+1}{2}\right],\right.$$
$$\left.\left[x=\frac{\sqrt{-2a^2-4a-1}+2a+1}{2},\ y=-\frac{\sqrt{-2a^2-4a-1}-2a-1}{2}\right]\right]$$

В этом примере Максиме нужно было найти точки пересечения окружности радиуса a с пересекающей ее прямой $x+y=2a+1$ и в общем случае произвольного a Максима нашла 2 точки пересечения.

Если же `solve` не сможет найти точных решений, как это она сделала в данном случае, то она может поступить как функция `integrate` и вернуть уравнение или систему уравнений в некотором упрощенном виде, а может и самостоятельно попытаться решить систему численно.

Попробуем найти точки пересечения двух кривых второго порядка: гиперболы $4x^2 - y^2 = 12$ с гиперболой $xy - x = 2$. Прежде всего мы записали (команда %i13) систему двух рассматриваемых уравнений. Затем дали команду решить систему уравнений совместно (команда %i14) и получили в качестве решения список, в котором имеются 4 (численных) приближенных решения, два из них – это действительные координаты точек пересечения (рис. 47) и два других – комплексные корни, содержащие мнимую единицу %i (корень квадратный из –1).

```
(%i13) eq2:[4*x^2-y^2=12,x*y-x=2];
(%o13) [4*x^2-y^2=12,x*y-x=2]
(%i14) solve(eq2);
(%o14) [[y=2,x=2],[y=-0.15356757100197,x=-1.733751846381093],
[y=3.608003221870287*%i+0.076783785237878,x=-0.5202594388652*%i-
0.13312403573587],[y=0.076783785237878-3.608003221870287*%i,x=
0.5202594388652*%i-0.13312403573587]]
```

```
wxplot2d([sqrt(4*x^2-12), -sqrt(4*x^2-12), (2+x)/x],
   [x,-5,5], [y,-10,10])$
```

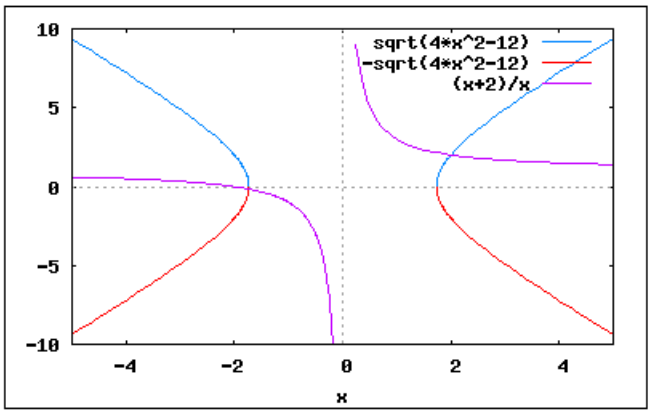


Рис. 47. Пересечение двух гипербол

Но если нам все же нужны точные значения корней (в аналитической записи), либо если они не найдены даже в числах, можно попробовать решить уравнения по очереди, выражая одно неизвестное через другое. Для данной пары уравнений решим второе из уравнений eq2 относительно *y*, подставим найденное решение в первое уравнение и решим его относительно *x*.

```
(%i15) solve(eq2[2],y);
```

$$(\%o15)\quad [\, y = \frac{x+2}{x}\,]$$

```
(%i16) eq2[1],%;
```

$$(\%o16)\quad 4x^2 - \frac{(x+2)^2}{x^2} = 12$$

```
(%i17) solve(%);
```

В результате решения получим 4 различных корня, в том числе 2 корня комплексные. После подстановки найденного решения в уравнение %o15, можно будет найти значение второй неизвестной. Точное решение уравнения %o16 имеет вид

```
(%i17) solve(%);
```

$$(\%o17)\quad [\, x = \frac{7\left(\frac{\sqrt{3}\,\%i}{2} - \frac{1}{2}\right)}{36\left(3^{-\frac{3}{2}}\sqrt{139} - \frac{8}{27}\right)^{1/3}} + \left(3^{-\frac{3}{2}}\sqrt{139} - \frac{8}{27}\right)^{1/3}\left(-\frac{\sqrt{3}\,\%i}{2} - \frac{1}{2}\right) - \frac{2}{3},\ x = \left(3^{-\frac{3}{2}}\sqrt{139} - \frac{8}{27}\right)^{1/3}\left(\frac{\sqrt{3}\,\%i}{2} - \frac{1}{2}\right) + \frac{7\left(-\frac{\sqrt{3}\,\%i}{2} - \frac{1}{2}\right)}{36\left(3^{-\frac{3}{2}}\sqrt{139} - \frac{8}{27}\right)^{1/3}} - \frac{2}{3},\ x = \left(3^{-\frac{3}{2}}\sqrt{139} - \frac{8}{27}\right)^{1/3} + \frac{7}{36\left(3^{-\frac{3}{2}}\sqrt{139} - \frac{8}{27}\right)^{1/3}} - \frac{2}{3},\ x = 2\,]$$

Запишем точное решение для первого и последнего корней из полученного списка, в итоге получим точные (хотя и немного страшные на вид) аналитические выражения.

```
(%i18) %o15[1],%[1];%o15[1],%th(2)[4];
```

$$(\%o18)\quad y=\frac{\frac{7\left(\frac{\sqrt{3}\,\%i}{2}-\frac{1}{2}\right)}{36\left(\frac{\sqrt{139}}{24\sqrt{3}}-\frac{8}{27}\right)^{1/3}}+\left(\frac{\sqrt{139}}{24\sqrt{3}}-\frac{8}{27}\right)^{1/3}\left(-\frac{\sqrt{3}\,\%i}{2}-\frac{1}{2}\right)+\frac{4}{3}}{\frac{7\left(\frac{\sqrt{3}\,\%i}{2}-\frac{1}{2}\right)}{36\left(\frac{\sqrt{139}}{24\sqrt{3}}-\frac{8}{27}\right)^{1/3}}+\left(\frac{\sqrt{139}}{24\sqrt{3}}-\frac{8}{27}\right)^{1/3}\left(-\frac{\sqrt{3}\,\%i}{2}-\frac{1}{2}\right)-\frac{2}{3}}$$

$$(\%o19)\quad y=2$$

Функция `solve` имеет довольно большое количество различных переключателей, из которых может пригодиться в первую очередь один, называемый: `solveradcan`. По умолчанию в *Максима* выставлено `false`, а если выставить этот флаг в `true`, мы заставим `solve`, помимо его поведения, применять `radcan` — функцию по упрощению показательных, логарифмических и степенных (с рациональными степенями) функций. Это делает работу функции `solve` более медленной (потому по умолчанию этот режим и выключен), но в некоторых случаях может помочь разрешить проблемы, которые без этого ключа приведут к невозможности найти точное решение [8].

9. Решение задач математического анализа

9.1. Нахождение производной

Функция diff(); позволяет найти производные как первого, так и более высоких порядков. При наличии у функции нескольких переменных можно найти частную производную по одной из них.

Синтаксис: diff(функция, переменная, порядок производной);

Пример: найти первую производную функции

$$y(x)=\frac{e^x}{x^2}.$$

Сначала введем функцию: f(x):=exp(x)/x^2; (обратите внимание, что далее в строке %i4, в отличие от присвоения значения переменной, используется комбинация символов ":=" (двоеточие и равно)), а затем дается команда найти ее производную по переменной x.

Для вычисления производной вводится команда: diff(y(x),x,1); или diff(y(x),x);. В случае первой производной ее порядок можно не указывать.

```
(%i4)  f(x):=exp(x)/x^2;
```

$$(\%o4)\quad f(x) := \frac{exp(x)}{x^2}$$

```
(%i5)  diff(f(x),x);
```

$$(\%o5)\quad \frac{\%e^x}{x^2} - \frac{2\,\%e^x}{x^3}$$

При вычислении кратных производных по нескольким переменным после указания функции перечисляются переменные дифференцирования с указанием соответствующих кратностей,

например,

diff(x^8*y^5, x, 4, y, 2); — дифференцирует функцию x^8y^5 по переменной x четыре раза и по переменной y два раза

```
(%i12)  'diff(x^8*y^5,x,4,y,2);
```

$$(\%o12)\quad \frac{d^6}{dx^4\,dy^2}\left(x^8y^5\right)$$

```
(%i13)  diff(x^8*y^5,x,4,y,2);
```

$$(\%o13)\quad 33600\,x^4y^3$$

Если функция `diff()` содержит только один аргумент, то функция `diff(выражение);` вычисляет не производную записанного выражения, а полный дифференциал этого выражения. Другими словами, запись `diff(f, x);` равнозначна математическому обозначению d*f*/d*x*, а `diff(f)` — d*f*.

```
(%i40)  diff(x^3);
```

$$(\%o40)\quad 3x^2\,del(x)$$

```
(%i41)  diff(x^3+y^2);
```

$$(\%o41)\quad 2y\,del(y) + 3x^2\,del(x)$$

Кроме того, функция `diff`() используется еще и для обозначения производных в дифференциальных уравнениях.

9.2. Интегрирование

Для нахождения неопределенного интеграла в качестве аргументов указывается функция и переменная интегрирования.

Синтаксис: integrate(функция, переменная);

Пример: вычислить интеграл от функции x^2+5x+3 по переменной x.

```
(%i16) 'integrate(x^2+5*x+3,x);
```

$$\int x^2 + 5x + 3 \, dx$$

```
(%i17) integrate(x^2+5*x+3,x);
```

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

При нахождении значения **определенного интеграла** помимо рассмотренных параметров указываются пределы интегрирования. В качестве пределов интегрирования могут фигурировать бесконечность (inf) и минус бесконечность (minf).

Синтаксис: integrate(функция, переменная, нижний предел, верхний предел);

Пример:

вычислить интеграл функции sin(x) по переменной x на отрезке от 0 до u

```
(%i20) 'integrate(sin(x),x,0,%pi);
```

$$\int_0^{\%pi} sin(x) \, dx$$

```
(%i21) integrate(sin(x),x,0,%pi);
(%o21) 2
```

В случае, когда интеграл расходится,

```
(%i22) 'integrate(1/x,x,0,inf);
```

$$\int_0^{inf} \frac{1}{x} \, dx$$

```
(%i23) integrate(1/x,x,0,inf);
Integral is divergent
 -- an error.  To debug this try debugmode(true);
```

Maxima выдает сообщение "Integral is divergent". Например, integrate(1/x, x, 0, inf); выдаст именно такое сообщение.

В некоторых случаях *Maxima* может попросить доопределить некоторую переменную, как в случае вычисления определенного интеграла, значение которого зависит от знака параметра a, являющегося верхним пределом, или интегрирования функции x^n.

В следующем примере подынтегральное выражение не зависит от знака параметра a, но значение интеграла — зависит, так как параметр a может быть записан или как верхний предел или как нижний предел.

```
(%i11) 'integrate(x^2*sqrt(a^2-x^2),x,0,a);
```

$$(\%o11)\quad \int_0^a x^2\sqrt{a^2 - x^2}\,dx$$

```
(%i12) integrate(x^2*sqrt(a^2-x^2),x,0,a);
Is  a  positive, negative, or zero? p;
```

$$(\%o12)\quad \frac{\%pi\, a^4}{16}$$

На вопрос *Maxima Is a positive, negative, or zero*? мы ответили *p* (*positive*) и получили положительное значение. В случае отрицательного знака у параметра *a* значение интеграла (%o12) будет отрицательное, а численное значение интеграла по модулю будет тем же.

В следующем примере прежде чем записать ответ Maxima уточнила, не является ли значение *n*+1 равным нулю: Is *n*+1 zero or nonzero?, после получения положительного ответа nonzero

```
(%i24) 'integrate(x^n,x);
```

$$(\%o24)\quad \int x^n\,dx$$

```
(%i25) integrate(x^n,x);
Is  n + 1  zero or nonzero? nonzero;
```

$$(\%o25)\quad \frac{x^{n+1}}{n+1}$$

Maxima, наконец, записала численное значение интеграла.

9.3. Нахождение пределов

Полноценных функций для нахождения предела существует в Maxima одна. Но зато какая! Она может принимать три различных варианта списка аргументов, и кроме того, на ее действие влияют еще и три флага.

Зовут эту функции вполне соответственно ее действию: `limit`; и в самом стандартном варианте ее вызов выглядит как `limit(выражение, переменная, значение_переменной)`, то есть то, что в математической записи выглядит как $\lim_{x\to a} f(x)$, в контексте Maxima запишется как `limit(f(x), x, a)`.

Синтаксис: limit(функция, переменная, значение_аргумента); или limit(функция, переменная, значение_аргумента, слева/справа);

Предел слева обозначается minus, а справа – plus. Много различных примеров собрано в табл. 4

Таблица 4

Пределы и их вычисление в Maxima

Предел функции	Вычисление предела в Maxima
$\lim_{x \to \infty} x^2$	(%i14) limit(x^2,x,inf); (%o14) inf
$\lim_{x \to 4+} atan\left(\frac{1}{x-4}\right)$	(%i15) limit(atan(1/(x-4)),x,4,plus); (%o15) %pi/2
$\lim_{x \to 4-} atan\left(\frac{1}{x-4}\right)$	(%i16) limit(atan(1/(x-4)),x,4,minus); (%o16) -%pi/2
$\lim_{x \to 1} \frac{x^2-1}{2x^2-x-1}$	(%i18) limit((x^2-1)/(2*x^2-x-1),x,1); (%o18) 2/3
$\lim_{x \to 0} \frac{(mx+1)^n-(nx+1)^m}{x^2}$	(%i20) limit(((1+m*x)^n-(1+n*x)^m)/x^2,x,0); (%o20) -m(m-n)n/2
$\lim_{x \to inf} \frac{\sqrt{x}+x^{1/3}}{\sqrt{2x+1}}$	(%i29) limit((x^(1/2)+x^(1/3))/sqrt(2*x+1),x,inf); (%o29) 1/sqrt(2)
$\lim_{x \to 0+} \frac{\lvert sin(x)\rvert}{x}$	(%i32) limit((abs(sin(x)))/x,x,0,plus); (%o32) 1
$\lim_{x \to 0-} \frac{\lvert sin(x)\rvert}{x}$	(%i33) limit((abs(sin(x)))/x,x,0,minus); (%o33) - 1
$\lim_{x \to 0} sin\left(\frac{1}{x}\right)$	(%i37) limit(sin(1/x),x,0); (%o37) ind
$\lim_{x \to 0} tan\left(\frac{1}{x}\right)$	(%i38) limit(tan(1/x),x,0); (%o38) und

В двух последних представленных здесь примерах появились два новых ответа Максимы, означающие, что искомый предел не существует: `ind` (от слова *indefinite — неопределенный*) и `und` (от слова *undefined* — опять же *неопределенный*). В документации первый из этих ответов описан как *indefinite but bounded* (*не определен, но ограничен*), что дает основание предположить, что речь идет о функции, не имеющей предела, но при этом ограниченной сверху либо в окрестности предельной точки, либо на всей прямой.

Функция `limit();` в третьем варианте — `limit(выражение);` — предназначена уже не для поиска собственно пределов,

а для упрощения выражений, содержащих символы `inf` и `minf`:

```
(%i39)  limit(1/inf);
(%o39)  0
```

Выражения такого рода могут возникать, к примеру, при подстановках в формулы результатов вычисления каких-то других пределов или интегралов.

Такая способность функции `limit();` — принимать различные списки аргументов — не является в Maxima чем-то особенным; она свойственна очень многим встроенным функциям. Как и различное действие в зависимости от значений разнообразных переключателей. Это достаточно удобно: не нужно запоминать много разных имен функций (для поиска пределов, к примеру, используется исключительно функция `limit`); для вычисления производных, в том числе и частных, — функция `diff`; для нахождения интегралов, как определенных, так и неопределенных — функция integrate. Имена наиболее часто используемых функций запомнить несложно, а о дополнительных ключах или флагах в случае чего можно прочитать во встроенной справке, набрав `? имя-функции`.

Об этих самых ключах к функции limit в [8] рассказывается следующее. Первый ключ называется `lhospitallim` и задает максимальное количество применений правила Лопиталя; название ключа происходит от фамилии ученого, давшей название самому правилу, которая в оригинале пишется как *L'Hospital*. Правило Лопиталя гласит, что в случае неопределенности вида 0/0 или ∞/∞ можно продифференцировать числитель и знаменатель — и предел от этого не изменится. Ограничитель количества применений этого правила нужен для того, чтобы избежать зацикливаний, которые могут случиться для бесконечно дифференцируемых функций, у которых в данной точке равны нулю либо бесконечности все производные. По умолчанию значение `lhospitallim` равно четырем, и этого, повидимому, вполне достаточно.

```
(%i1)  limit((f(x)/f(x+a)),x,inf);
                          f(x)
(%o1)    lim   ------------------
       x -> inf f(x + a)
(%i2)  limit((f(x)/f(x+a)),x,inf),limsubst:true;
(%o2)  1
(%i3)  limit((f(x)/f(x+a)),x,inf),limsubst:false;
                          f(x)
(%o3)    lim   ------------------
       x -> inf f(x + a)
(%i4)  f(x):=sin(x)$''%i1;
(%o5)  und
```

Второй ключ к функции `limit` — это флаг `limsubst`, который, будучи выставлен в `true`, позволяет этой функции производить подстановки внутрь

неизвестных выражений. По умолчанию этот флаг равен `false`, что исключает ошибки вроде той, что появляется в строках (%i2)–(%o2), когда предел отношения двух "соседних" значений функции на бесконечности, автоматически считается равным единице для любой функции произвольного вида.

И наконец, последний дополнительный параметр — еще один флаг, по имени `tlimswitch`. По умолчанию он тоже выключен, а если его включить, функция `limit` будет, при невозможности найти предел другими способами, пытаться его найти путем разложения подпредельной функции в ряд Тейлора в окрестности заданной точки.

Что касается "неполноценных" функций, то таковая тоже имеется по имени `tlimit();` она представляет собой фактически просто-напросто вызов самой функции `limit();` с поднятым флагом `tlimswitch`, то есть пытается при необходимости разложить исследуемую функцию в ряд Тейлора вне зависимости от реального значения этого флага. Другими словами вызов `tlimit(аргументы);` полностью аналогичен записи `limit(аргументы), tlimswitch:true;` только чуть короче. И аргументы она может принимать точно такие же [8].

9.4. Разложение в ряд Тейлора

Синтаксис: taylor(функция, *x, a, n*); Параметр *n* определяет, до какой степени параметра разложения $(x–a)^n$ находить решение.

Пример: Разложить функцию sin(*x*)+*ax*+1 в ряд Тейлора по степеням *x*+1 до третьей степени.

```
(%i1)  taylor(sin(x)+a*x+1,x,-1,3);
```

$$(\%o1)\quad 1 - sin(1) - a + (a + cos(1))(x + 1) + \frac{sin(1)(x+1)^2}{2} - \frac{cos(1)(x+1)^3}{6} + \ldots$$

9.5. Нахождение суммы ряда

Синтаксис: sum(функция, переменная, индекс_начального_члена_ ряда, индекс_конечного_члена _ряда). Примеры использования sum() собраны в табл. 5.

В первом примере (табл. 5) Максима сразу записывает сумму конечного ряда, которую рассчитала с помощью функции `sum();`. Во втором примере Максима отдает предпочтение записи ряда с использованием знака суммы, но задачу с вычислением суммы бесконечно убывающей геометрической прогрессии она тут же решает, если выставить в `true` значение флага `simpsum`, который по умолчанию имеет значение `false` (команда %i33).

В любом случае, Максима не желает ни упрощать, ни суммировать до бесконечности, если не указан флаг `simpsum`. В последнем примере Максима не может получить точное значение суммы бесконечного ряда и суммировать неограниченно (до ∞) тоже отказывается, но, тем не менее, мгновенно

просуммировала ряд, когда мы количество членов ряда ограничили значениями: 10, 100 и 1000. Максима возьмется суммировать ряд, если даже мы попросим просуммировать миллион слагаемых, но в таком случае ее вычисления нам придется прервать, так как она это будет делать очень долго, да и рассчитанные ею первые 16 цифр при этом никак не изменятся.

Таблица 5

Нахождение суммы ряда

Ряд	Вычисления ряда в Maxima
$\sum_{i=1}^{7} i^2$	(%i7) sum(i^2,i,1,7); (%o7) 140
$\sum_{i=1}^{\infty} \frac{1}{i}$	(%i32) sum(1/(3^i),i,1,inf); (%o32) $\sum_{i=1}^{inf} \frac{1}{3^i}$ (%i33) %,simpsum:true; (%o33) $\frac{1}{2}$ (%i34) sum(1/(3^i),i,1,inf),simpsum; (%o34) $\frac{1}{2}$
$\sum_{i=1}^{inf} \frac{1}{i^4}$	(%i9) sum(1/(i^4),i,1,inf),simpsum; (%o9) $\frac{\%pi^4}{90}$
$\sum_{i=1}^{\infty} \frac{1}{i^i}$	(%i1) sum(1/(i^i),i,1,10),numer; (%o1) 1.291285997059043 (%i2) sum(1/(i^i),i,1,100),numer; (%o2) 1.291285997062664 (%i3) sum(1/(i^i),i,1,1000),numer; (%o3) 1.291285997062664

По умолчанию Максима производит вычисления с точностью 16 знаков, но может делать вычисления с любой заданной точностью. Количество значащих цифр в представлении числа определяется специальной переменной `FPPREC`. Увеличение ее значения приводит к возрастанию точности результата. Для вывода результата в этом случае понадобится несколько строк. Однако графический интерфейс wxMaxima не предусматривает вывод одного числа в несколько строк. Но у нас не будет ограничений если мы воспользуемся консольной версией Maxima (рис. 48).

На рис. 48 продемонстрирована возможность Максимы работать с числами произвольной длины. Maxima была запущена из терминала, “с удовольствием” она рассчитала все знаки числа 80! Кроме того, Maxima для примера вывела

первые 220 знаков числа π. Чтобы работать с числами большой длины, здесь была использована функция block(), у которой первый аргумент – переменная fpprec, ей попутно присвоено значение 220, второй аргумент – функция bfloat(), выводящая число π в экспоненциальном формате вида mbn, означающем краткую запись числа $m*10^n$.

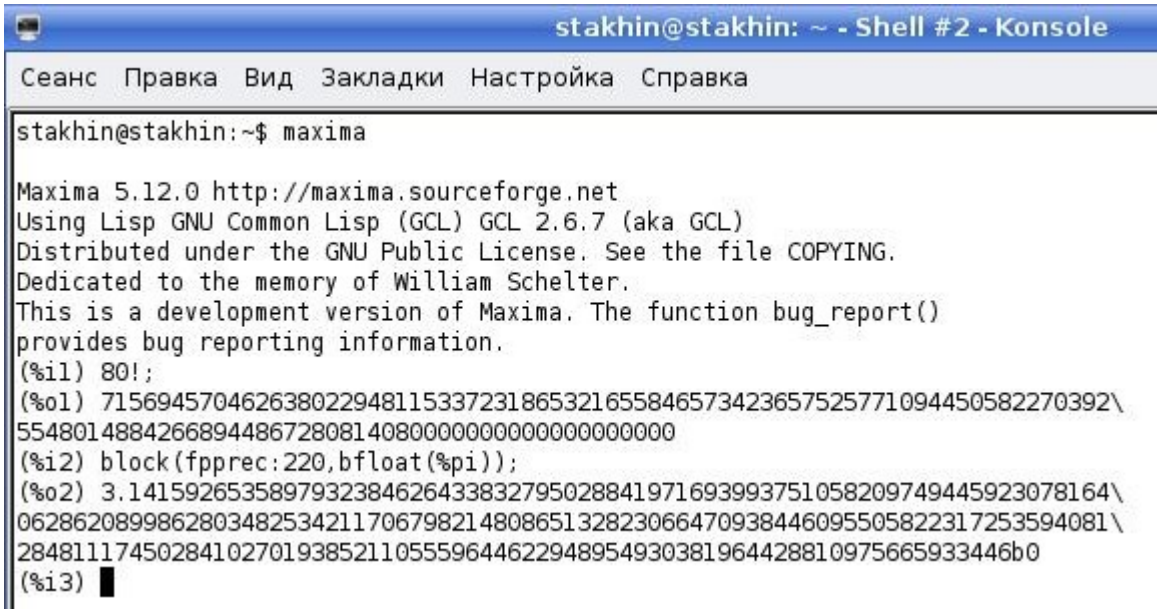


Рис. 48. Максима в консольной версии выводит 220 знаков числа π

Однако графический интерфейс wxMaxima такими возможностями не располагает. Для 50! графический интерфейс wxMaxima результат выводит, но для 80! графический интерфейс wxMaxima приводит лишь первые и последние 30 знаков факториала и указывает число пропущенных [59 digits] цифр.

```
(%i42)  50!;
(%o42)  30414093201713378043612608166064768844377641568960512000000000000

(%i43)  80!;
(%o43)  71569457046263802294811533723 1[59 digits]867280814080000000000000000000
```

Заметим, что графический интерфейс xMaxima не только обладает всеми возможностями консольной версии Maxima, но и имеет собственное меню и позволяет даже рисовать графики. Выше уже отмечались скромные возможности этого интерфейса (рис. 2), но, заметим, этот интерфейс совершенно необходим для рисования полноценных графиков. Если xMaxima не установлена, ее нужно доустановить.

Чтобы узнать, установлен ли этот интерфейс на компьютере нужно из консоли дать команду xmaxima. После загрузки xmaxima в верхней части окна появляется приглашение вида (%i1), в нижней части окна xMaxima приведены некоторые примеры использования Максимы с текстом на английском языке.

После того как была написана команда (%i1) plot2d(exp(–u)*sin(u),[u,0,2* %pi]); тут же в этом окне Максима вывела график, приведенный на рис. 49.

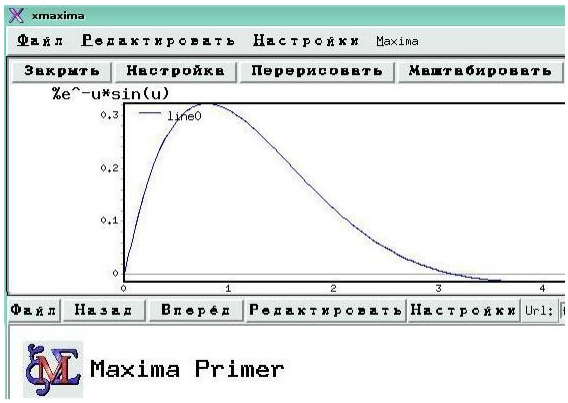


Рис. 49. График в графическом окне xMaxima

10. Решение задач линейной алгебры

В данном разделе будут рассмотрены основные операции с матрицами. Для задания матрицы используется функция matrix:

```
(%i4)  matrix([1,2],[3,4]);
                 [ 1  2 ]
(%o4)            [      ]
                 [ 3  4 ]

(%i5)  matrix([x,y,a+b],[1,2,3],[0,0,z]);
                 [ x  y  b + a ]
                 [             ]
(%o5)            [ 1  2    3   ]
                 [             ]
                 [ 0  0    z   ]
```

В этих примерах были определены две квадратные матрицы: второго и третьего порядка.

10.1. Операции с матрицами

Сложение и вычитание матриц осуществляется поэлементно. Поэлементно могут быть перемножены или поделены элементы двух матриц или поэлементно (к каждому элементу матрицы) могут быть применены функции exp(); sqrt(); (табл. 6).

Возведение в степень x^y будет записано символически (табл. 6). Поэлементное умножение обозначается символом *, поэлементное деление – символом /, поэлементное возведение в степень ^. Для сложения и вычитания матриц они должны быть одинакового размера. Ранг матрицы – rank(M) и определитель – determinant(M) определены только для квадратных матриц, транспонировать – transpose(M) можно любые матрицы, но при задании матриц нужно следить за тем, что они являются прямоугольными.

Далее мы рассмотрим основные операции с матрицами на конкретных примерах. Для этого будем использовать две матрицы: x и y.

Таблица 6

Операции с матрицами

x: matrix([5,2],[3,9]); $\begin{bmatrix} 5 & 2 \\ 3 & 9 \end{bmatrix}$	y: matrix([a,b],[%pi,%e]); $\begin{bmatrix} a & b \\ \pi & \%e \end{bmatrix}$	x+y; $\begin{bmatrix} a+5 & b+2 \\ \pi+3 & \%e+9 \end{bmatrix}$
y-x; $\begin{bmatrix} a-5 & b-2 \\ \pi-3 & \%e-9 \end{bmatrix}$	x*y; $\begin{bmatrix} 5a & 2b \\ 3\pi & 9\%e \end{bmatrix}$	y/x; $\begin{bmatrix} \frac{a}{5} & \frac{b}{2} \\ \frac{\pi}{3} & \frac{\%e}{9} \end{bmatrix}$
exp(x); $\begin{bmatrix} \%e^5 & \%e^2 \\ \%e^3 & \%e^9 \end{bmatrix}$	sqrt(y); $\begin{bmatrix} \sqrt{a} & \sqrt{b} \\ \sqrt{\pi} & \sqrt{\%e} \end{bmatrix}$	y^x; $\begin{bmatrix} a & b \\ \pi & \%e \end{bmatrix}^{\begin{bmatrix} 5 & 2 \\ 3 & 9 \end{bmatrix}}$

10.2. Умножение матриц и возведение их в степень

Умножение матриц (обозначается как пробел, точка, пробел) производится по правилу "строка умножается на столбец". Возведение матрицы в степень, обозначенное как ^ сводится к поэлементному возведению в степень компонент матрицы, но возведение в степень вида ^^ выполняется как умножение матрицы самой на себя требуемой число раз. Возведение в степень ^^(–1) означает поиск обратной матрицы.

Таблица 7

Умножение матриц

x: matrix([5,2],[3,9]); $\begin{bmatrix} 5 & 2 \\ 3 & 9 \end{bmatrix}$	y: matrix([a,b],[%pi,%e]); $\begin{bmatrix} a & b \\ \pi & \%e \end{bmatrix}$	x . x; $\begin{bmatrix} 31 & 28 \\ 42 & 87 \end{bmatrix}$ 31=5*5+2*3; 28=5*2+2*9; 42=3*5+9*3; 87=3*2+9*9
x . y; $\begin{bmatrix} 5a+2\pi & 5b+2\%e \\ 3a+9\pi & 3b+9\%e \end{bmatrix}$	z:x^^(-1); $\begin{bmatrix} \frac{3}{13} & -\frac{2}{39} \\ -\frac{1}{13} & \frac{5}{39} \end{bmatrix}$	x . z; $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
x^2; $\begin{bmatrix} 25 & 4 \\ 9 & 81 \end{bmatrix}$	invert(x); $\begin{bmatrix} \frac{3}{13} & -\frac{2}{39} \\ -\frac{1}{13} & \frac{5}{39} \end{bmatrix}$	y . x; $\begin{bmatrix} 3b+5a & 9b+2a \\ 5\pi+3\%e & 2\pi+9\%e \end{bmatrix}$

10.3. Решение систем линейных алгебраических уравнений

Системы линейных уравнений могут быть решены различными способами:
1) Например, методом Крамера система двух уравнений с двумя неизвестными может быть решена так:

```
(%i1) A: matrix([a,b],[c,d]);
```

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

```
(%i2) B: matrix([E],[F]);
```

$$\begin{bmatrix} E \\ F \end{bmatrix}$$

```
(%i3) u : determinant(matrix([E,b],[F,d]))/determinant(A);
```

$$\frac{dE-bF}{ad-bc}$$

```
(%i4) v : determinant(matrix([a,E],[c,F]))/determinant(A);
```

$$\frac{aF-cE}{ad-bc}$$

откуда следует, что решение системы уравнений

$a*u+b*v=E$;

$c*u+d*v=F$

в аналитическом виде следующее $u=(dE–bF)/(ad–bc)$; $v=(aF–cE)/(ad–bc)$.

2) Эту же самую систему уравнений вида AX = B с использованием обратной матрицы A^{-1} можно решить следующим образом: $X = A^{-1} B$

```
(%i4) B: matrix([E],[F]);
```

$$\begin{bmatrix} E \\ F \end{bmatrix}$$

```
(%i5) A^^(-1) . B;
```

$$\begin{bmatrix} \frac{dE}{ad-bc}-\frac{bF}{ad-bc} \\ \frac{aF}{ad-bc}-\frac{cE}{ad-bc} \end{bmatrix}$$

3) А с использованием функции solve() ответ получим быстрее всего.

```
(%i12) solve([a*u+b*v=E, c*u+d*v=F],[u,v]);
```

$$[[u=-\frac{dE-bF}{bc-ad}, v=\frac{cE-aF}{bc-ad}]]$$

11. Вопросы и задания для самостоятельной работы

На рис. 37 приведены греческие буквы.

Задание 1). Настройте через меню Правка | Настройка | Стиль конфигурацию wxMaxima, чтобы можно было использовать греческие буквы.

Задание 2). Запишите греческие буквы в виде матрицы рис. 50 *a).*

Вопрос 1. Как получить транспонированную матрицу *b)* с помощью функции transpose()? (Для записи греческих букв использованы следующие тексты: `\alpha,\beta,\gamma,\delta,\epsilon,\zeta,\eta,\theta, \iota, \kappa, \mu,\nu,\xi,\tau,\upsilon,\rho,\sigma, o,\pi, \phi, \chi,\psi,\omega,\Lambda,\Sigma ).`

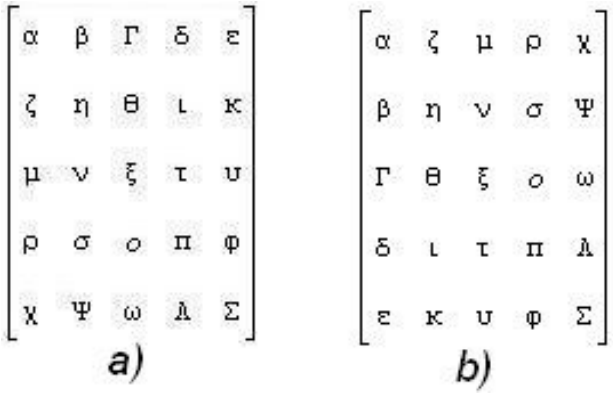


Рис. 50. Греческие буквы в интерфейсе wxMaxima

Вопрос 2. Как получить текст записи matrix(...) матрицы *b)* в окне многострочного ввода (т.е., как, пользуясь построчной записью матрицы, получить запись по столбцам)?

Задание 3. Вычислите определитель матрицы *a)* и матрицы *b).* Докажите, что они равны друг другу.

Вопрос 3. Как вычислить 20-ю цифру в записи числа π, если функция floor() вычисляет целую часть выражения, а mod(...,...) – остаток от деления? (Например: floor(10/3); возвращает 3, mod(20,3); возвращает 2).

Вопрос 4. Что больше π^e или e^π?

Вопрос 5. А на сколько (в процентах) большее из чисел (π^e или e^π) больше меньшего?

Задание 4. Вычислите $G = \sum_{n=1}^{n=N} \frac{1}{n} - \log(N)$ и найдите N_0, начиная с которого с точностью 9 знаков эта величина с ростом N не изменяется.

Задание 5. Разложите на множители числа $10^{10}-1$ и $10^{10}+1$.

Вопрос 6. Пьер Ферма считал, что все числа вида $2^{2^n}+1$, где n целое неотрицательное число – простые. Начиная с какого n Maxima опровергнет утверждение Ферма?

Задание 6. Разложите на множители полином x^3-7x+6.

Задание 7. Постройте астроиду $x = \cos^3 t$; $y = \sin^3 t$.

Задания 8 – 37. Упростите алгебраическое выражение

(Задачи 8–117, предназначенные для самостоятельного решения, взяты из учебника для вуза [10], снабжены ответами).

№	Упростите алгебраическое выражение	Ответ
8	$\left(\frac{(x^4-x^3-11x^2+9x+18)}{(x^4-3x^3-7x^2+27x-18)}\right)/\left(\frac{(x^3-9x^2+26x-24)}{(x^3-8x^2+19x-12)}\right)$	$\frac{(x+1)}{(x-1)}$
9	$\frac{(2-x)}{(x+1)}\cdot\frac{(3x^4-24x^3-3x^2+204x-252)}{(220x-70x^2-168-15x^3+10x^4-x^5)}$	$\frac{3}{(x+1)}$
10	$\frac{(x^3+2x^2+4x+8)}{(x^5+5x^4-16x-80)}\cdot\frac{(2x^4+10x^3-16x-80)}{(x^2+2x+4)}$	2
11	$\frac{(2x^4+10x^3-2x-10)}{(x^2+x+1)}\cdot\frac{(x^3+x^2+x+1)}{(x^5+5x^4-x-5)}$	2
12	$\frac{(4x^4+x^5-81x-324)}{(3x^4+10x^3-81x-270)}\cdot\frac{(3x^3+19x^2+57x+90)}{(x^4+7x^3+21x^2+63x+108)}$	1
13	$\left(\frac{(4x^5+40x^4+100x^3-80x^2-320x+256)}{(x^4+x^3-9x^2+11x-4)}\right).\left(\frac{(3x^3-3x^2)}{(x^2+8x+16)}\right)$	$12x^2$
14	$\left(\frac{(5x^4+10x^3-100x^2-330x+225)}{(x^4+x^3-7x^2-x+6)}\right)/\left(\frac{(x^2-2x-15)}{(x^2-3x+2)}\right)$	5
15	$\left(\frac{(x^3+3x^2-9x-27)}{(x^3-5x^2-15x-72)}\right).\left(\frac{(x^4-8x^3-27x+216)}{(49x^4-882x^2+3969)}\right)$	$\frac{1}{49}$
16	$\left(\frac{(7x^4-126x^2+567)}{(x^5-8x^4-27x^2+216x)}\right).\left(\frac{(x^3-5x^2-15x-72)}{(x^3+3x^2-9x-27)}\right)$	$\frac{7}{x}$
17	$\left(\frac{(x^3+6x^2+12x+8)}{(x^2+3x-4)}\right).\left(\frac{(x^4+x^3-9x^2+11x-4)}{(9x^5+36x^4+9x^3-90x^2-36x+72)}\right)$	$\frac{1}{9}$
18	$\left(\frac{(x^3-x^2-4x+4)}{(x^3-3x+2)}\right).\left(\frac{(3x-3)}{(2x-4)}\right)$	$\frac{3}{2}$
19	$\left(\frac{(x^4+2x^3-72x^2-416x-640)}{(9x^3-144x^2+180x+360)}\right).\left(\frac{(x-10)}{(x^2+8x+16)}\right)$	$\frac{1}{9}$
20	$\left(\frac{(x^4+x^3-3x^2-5x-2)}{(9x^3-351x^2+3240x+3600)}\right).\left(\frac{(x^2-40x+400)}{(x^3-3x-2)}\right)$	$\frac{1}{9}$
21	$\left(\frac{(2x^4+4x^3-4x-2)}{(x^3+x^2-x-1)}\right).\left(\frac{(x^4-7)}{(2x+2)}\right)$	x^4-7

№	Упростить алгебраическое выражение	Ответ
22	$(\frac{(4x^4+4x^3-48x^2-112x-64)}{(2x^3+4x^2-32x-64)}).(\frac{(x+4)}{(x^2+3x+2)})$	2
23	$(\frac{(4x^4+35x^3-45x^2-315x+81)}{(8x^4+166x^3+1038x^2+1674x-486)}).(\frac{(x+9)}{(x^2-6x+9)})$	$\frac{1}{(2x-6)}$
24	$(\frac{(x^4+x^3-7x^2-x+6)}{(5x^4+10x^3-100x^2-330x-225)}).(\frac{(x^3-2x^2-15x)}{(x^2-3x+2)})$	$\frac{x}{5}$
25	$(\frac{(220x-70x^2-168-15x^3+10x^4-x^5)}{(3x^4-24x^3-3x^2+204x-252)}).(\frac{(3x^2-6x^2+12)}{(x-2)})$	x^2-4
26	$(\frac{(x^2+3x+2)}{(x^2-16)}).(\frac{(2x^3+4x^2-32x-64)}{(4x^4+4x^3-48x^2-112x-64)})$	$\frac{1}{(2x-8)}$
27	$(\frac{(8x^4+166x^3+1038x^2+1674x-486)}{(4x^4-45x^2+35x^3-315x+81)}).(\frac{(x^2-9)}{(x^2+12x+27)})$	2
28	$(\frac{(4x^5+40x^4+100x^3-80x^2-320x+256)}{(x^4+x^3-9x^2+11x-4)}).(\frac{(3x^3-3x^2)}{(x^2+8x+16)})$	9
29	$(\frac{(x^3+x^2-x-1)}{(2x^4+4x^3-4x-2)}).(\frac{(2(x+1))}{(x^2+2)})$	$\frac{1}{(x^2+2)}$
30	$(\frac{(4x^5+40x^4+100x^3-80x^2-320x+256)}{(x^4+x^3-9x^2+11x-4)}).(\frac{(3x^3-3x^2)}{(x^2+8x+16)})$	9
31	$(\frac{(2x-4)}{(x-1)}).(\frac{(x^3-3x+2)}{(x^3-x^2-4x+4)})$	2
32	$(\frac{(x^3-3x-2)}{(x^2+40x+400)}).(\frac{(9x^3-351x^2+3240x+3600)}{(x^4+x^3-3x^2-5x-2)})$	9
33	$(\frac{(5x^4+10x^3-100x^2-330x-225)}{(x^4+x^3-7x^2-x+6)}).(\frac{(x^2-3x+2)}{(x^2-2x-15)})$	5
34	$(\frac{(9x^5+36x^4+9x^3-90x^2-36x+72)}{(x^4+x^3-9x^2+11x-4)}).(\frac{(x^3+3x^2-4x)}{(x^3+6x^2+12x+8)})$	$9x$
35	$(\frac{(x^2+8x+16)}{(x^2-x)}).(\frac{(x^4+x^3-9x^2+11x-4)}{(4x^5+40x^4+100x^3-80x^2-320x+256)})$	$\frac{1}{(4x)}$
36	$(\frac{(x^3+2x^2+4x+8)}{(x^5+5x^4-16x-80)}).(\frac{(3x^4+10x^3-16x-80)}{(x^2+2x+4)})$	2
37	$(\frac{(4x^5+40x^4+100x^3-80x^2-320x+256)}{(x^4+x^3-9x^2+11x-4)}).(\frac{(3x^3-3x^2)}{(x^2+8x+16)})$	x

Задания 38–67 .Раскройте скобки и приведите подобные слагаемые

№	Раскройте скобки	Ответ
38	$(x-2)(x^2+5)(x+2)$	x^4+x^2-20
39	$(x+6)(2x+3)(3x+5)$	$6x^3+55x^2+129x+90$
40	$(x-10)(x+4)^3$	$x^4+2x^3-72x^2-416x-640$
41	$2(x-1)(x+1)^3$	$2x^4+4x^3-4x-2$
42	$9(x-1)^2(x+2)^3$	$9x^5+36x^4+9x^3-90x^2-36x+72$
43	$(x-1)^3(x+4)$	$x^4+x^3-9x^2+11x-4$
44	$2(x+2)(x+6)(3x+7)$	$6x^3+62x^2-184x+168$
45	$(x+3)(x+4)(x^2+9)$	$x^4+7x^3+21x^2+63x+108$
46	$x(x-3)(3x+10)(x+3)^2$	$3x^5+19x^4+3x^3-171x^2-270x$
47	$(x-3)(x+3)(x+4)(x^2+9)$	$4x^4+x^5-81x-324$
48	$(3x+10)(x+3)^2$	$3x^3+28x^2+87x+90$
49	$2(x-2)(x+5)(x^2+2x+4)$	$2x^4+10x^3-16x-80$
50	$(x-2)(x+2)(x+5)(x^2+4)$	$x^5+5x^4-16x-80$
51	$x^2(x-5)(x+3)^2$	$x^5+x^4-21x^3-45x^2$
52	$(x^2-5)(x+3)^2$	$x^4+6x^3+4x^2-30x-45$
53	$(x+2)(2x+3)(2x^2+5)$	$4x^4+14x^3+22x^2+35x+30$
54	$2(x-2)(x+2)^2(x+5)$	$2x^4+14x^3+12x^2-56x-80$
55	$x(x-3)(x+4)(x^3+4)$	$x^6+4x^3+x^5+4x^2-48x-12x^4$
56	$(x+2)(2x-3)(x^3+4)$	$2x^5+8x^2+x^4+4x-6x^2-24$
57	$(x-7)(4x-3)(x^2+3)$	$4x^4-31x^3+33x^2-93x+63$
58	$(x-6)(x-5)(2x-3)$	$2x^3-25x^2+93x-90$
59	$2(x-4)(7x+5)(x^2-3)$	$14x^4-82x^2-46x^3+138x+120$
60	$(x-2)(x+2)^2(3x-5)$	$14x^4-82x^2-46x^3+138x+120$
61	$(x-2)(x+2)(x+3)(6x+5)$	$3x^4+x^3-22x^2-4x+40$

№	Раскройте скобки	Ответ
62	$4(x-1)(x+1)(x+3)(4x+7)$	$6x^4+23x^3-9x^2-92x-60$
63	$-(x-3)(x+4)(x^3+5)$	$-x^5-x^4+12x^3-5x^2-5x+60$
64	$-2(x-4)(x+3)(2x+5)$	$-6x^2+58x+120-4x^3$
65	$x(x+9)(x^2+7)$	$x^4+7x^2+9x^3+63x$
66	$-(x-9)(x-7)(x^2+4)$	$16x^3-67x^2+64x-x^4-252$
67	$(x+4)(x+8)(5x-4)$	$5x^3+56x^2+112x-128$

Задания 68–97. Разложите алгебраическое выражение на множители.

№	Разложите на множители	Ответ
68	x^3+2x^2+4x+8	$(x+2)(x^2+4)$
69	$6x^3+55x^2+129x+90$	$(x+6)(2x+3)(3x+5)$
70	$x^4+2x^3-72x^2-416x-640$	$(x-10)(x+4)^3$
71	$2x^4+4x^3-4x-2$	$2(x-1)(x+1)^3$
72	$9x^5+36x^4+9x^3-90x^2-36x+72$	$9(x-1)^2(x+2)^3$
73	$x^4+x^3-9x^2+11x-4$	$(x-1)^3(x+4)$
74	$6x^3+62x^2-184x+168$	$2(x+2)(x+6)(3x+7)$
75	$x^4+7x^3+21x^2+63x+108$	$(x+3)(x+4)(x^2+9)$
76	$3x^5+19x^4+3x^3-171x^2-270x$	$x(x-3)(3x+10)(x+3)^2$
77	$4x^4+x^5-81x-324$	$(x-3)(x+3)(x+4)(x^2+9)$
78	$3x^3+28x^2+87x+90$	$(3x+10)(x+3)^2$
79	$2x^4+10x^3-16x-80$	$2(x-2)(x+5)(x^2+2x+4)$
80	$x^5+5x^4-16x-80$	$(x-2)(x+2)(x+5)(x^2+4)$
81	$x^5+x^4-21x^3-45x^2$	$x^2(x-5)(x+3)^2$
82	$x^4+6x^3+4x^2-30x-45$	$(x^2-5)(x+3)^2$
83	$4x^4+14x^3+22x^2+35x+30$	$(x+2)(2x+3)(2x^2+5)$
84	$2x^4+14x^3+12x^2-56x-80$	$2(x-2)(x+2)^2(x+5)$

№	Разложите на множители	Ответ
85	$x^6+4x^3+x^5+4x^2-48x-12x^4$	$x(x-3)(x+4)(x^3+4)$
86	$2x^5+8x^2+x^4+4x-6x^2-24$	$(x+2)(2x-3)(x^3+4)$
87	$4x^4-31x^3+33x^2-93x+63$	$(x-7)(4x-3)(x^2+3)$
88	$2x^3-25x^2+93x-90$	$(x-6)(x-5)(2x-3)$
89	$14x^4-82x^2-46x^3+138x+120$	$2(x-4)(7x+5)(x^2-3)$
90	$14x^4-82x^2-46x^3+138x+120$	$(x-2)(x+2)^2(3x-5)$
91	$3x^4+x^3-22x^2-4x+40$	$(x-2)(x+2)(x+3)(6x+5)$
92	$6x^4+23x^3-9x^2-92x-60$	$4(x-1)(x+1)(x+3)(4x+7)$
93	$-x^5-x^4+12x^3-5x^2-5x+60$	$-(x-3)(x+4)(x^3+5)$
94	$-6x^2+58x+120-4x^3$	$-2(x-4)(x+3)(2x+5)$
95	$x^4+7x^2+9x^3+63x$	$x(x+9)(x^2+7)$
96	$16x^3-67x^2+64x-x^4-252$	$-(x-9)(x-7)(x^2+4)$
97	$5x^3+56x^2+112x-128$	$(x+4)(x+8)(5x-4)$

Задания 98– 107. Разложите рациональную дробь на простейшие дроби

№	Рациональная дробь	№	Рациональная дробь
98	$\frac{(5x^4+7x^3+5x-4)}{((x^2+4)(x-2)^2(x^2-1))}$	103	$\frac{(x^6+3x^3+4x+12)}{((x^2-25)(3x^2+9x))}$
99	$\frac{(3x^5+6x^3+5x-1)}{((x^2-4x+3)(x-2)^2(x^2-16))}$	104	$\frac{(3x^5+x^2+4x)}{((5x^2+6x-1)(3-x)(x+2))}$
100	$\frac{(x^5-7x^4+2x-8)}{((x^3-4x^2+5x)(x-3)^2)}$	105	$\frac{(2x^6-3x^4+9)}{((x^2-2x-15)(4x+1))}$
101	$\frac{(x^3+2x^2+3x+4)}{((x^2-x)(3-x)(x-3))}$	106	$\frac{(7x^5-5x^6+1)}{((x^2-x)x^3(x^2-9))}$
102	$\frac{(3x^4+3x+4)}{((x^2-1)(x^2-9))}$	107	$\frac{(8x^5-14x^3+34)}{(x(x^2-x)(7-x)^2)}$

№	Ответы на задание "Разложите рациональную дробь на простейшие дроби"
98	$\frac{(121x+635)}{(507(x^2+4x+1))}-\frac{1}{(18(x+1))}-\frac{5}{(6(x-1))}+\frac{989}{(1521(x-2))}+\frac{7}{(39(x-2)^2)}$
99	$\frac{1159}{(3360(x+4))}+\frac{13}{(30(x-1))}+\frac{92}{(3(x-2))}+\frac{51}{(4(x-2)^2)}-\frac{905}{(14(x-3))}+\frac{3475}{(96(x-4))}$
100	$\frac{-(111x-479)}{(10(x^2-4x+5))}-\frac{8}{(45x)}+\frac{257}{(18(x-3))}-\frac{163}{(3(x-3)^2)}+1$
101	$\frac{4}{(9x)}-\frac{5}{(2(x-1))}+\frac{19}{(18(x-3))}-\frac{29}{(3(x-3)^2)}$
102	$\frac{-119}{(24(x+3))}+\frac{1}{(4(x+1))}-\frac{5}{(8(x-1))}+\frac{16}{(3(x-3))}+3$
103	$\frac{(x^2-3x+34)}{3}-\frac{7621}{(150(x+5))}+\frac{9}{(2(x+3))}-\frac{4}{(75x)}+\frac{334}{(25(x-5))}$
104	$\frac{(22039x-2724)}{(5425(5x^2+6x-1))}-\frac{(15x-3)}{25}-\frac{20}{(7(x+2))}-\frac{75}{(31(x-3))}$
105	$\frac{(64x^3+112x^2+1092x+3887)}{128}-\frac{18409}{(29568(4x+1))}+\frac{153}{(11(x+3))}+\frac{3673}{(21(x-5))}$
106	$\frac{-5345}{(1944(x+3))}+\frac{10}{(81x)}+\frac{10}{(81x^2)}+\frac{1}{(9x^3)}+\frac{1}{(9x^4)}-\frac{3}{(8(x-1))}-\frac{1943}{(972(x-3))}$
107	$\frac{-306}{(343x)}-\frac{34}{(49x^2)}+\frac{7}{(9(x-1))}+\frac{370793}{(3087(x-7))}+\frac{64844}{(147(x-7)^2)}+8$

Задания 108–117. Графически исследуйте решение нелинейных уравнений и получите решение.

№	Решите уравнение	№	Решите уравнение
108	$\ln^2(x-1)=3\cos(2x)+1$	113	$\sqrt{(25-x^2)}=arctg(2x)$
109	$\frac{10}{(1+x^2)}=2\sin 2x+x$	114	$\frac{(x^3+2x^2+3x+4)}{((x^2-x)(3-x)^3(x^2-81))}$
110	$\frac{(10x-2)}{(3+x^2)}=2\cos 2x+\sqrt[4]{x}$	115	$arctg\,2x-\frac{(x-1)^4}{(5)}+\sin^2(5x)=0$
111	$\frac{10}{(1+x^2)}=2\cos 2x+x$	116	$\sin^2 x\cdot\sqrt{(81-x^2)}=5e^{-x^2}$
112	$\sin x\cdot\sqrt{(81-x^2)}=5\,arctg\,x$	117	$\frac{(x^2-9)}{(x^2+4)}=\sqrt{x^2+1}\,e^{x\cos x}$

12. Список сайтов и использованных источников

1. Русская версия сайта Maxima http://maxima. sourceforge.net/ru/ (редактор Алексей Бешéнов), он же на http://iais.kemsu.ru/odocs/lisp/maxima/index.htm
2. Лекция «Системы компьютерной алгебры» http://www. intuit.ru/department/se /pinform/8/ (Автор: Е.А. Роганов)
3. Системы компьютерной алгебры статья на сайте Denis Kirienko http://server.179.ru /tasks/maxima/1.html
4. Википедия о Maxima: http://ru.wikipedia.org/wiki/Maxima (дает ссылку введение в Maxima)
5. Введение в Максима (ссылка из Википедии) http://lib.custis.ru/index.php/ Maxima (очень кратко)
6. Перевод на русский язык статьи Роберта Додиера, Коротко о Maxima http://beshenov .ru/maxima/minimal-maxima.pdf
7. Основы работы в Maxima/wxMaxima. Maxima — максимум свободы символьных вычислений http://iais.kemsu.ru/odocs/lisp/maxima /maxima-tarnavsky-1.html
8. Дистанционное обучение от Mandriva.ru Основы работы в математическом пакете Maxima (Тьютор Тихон Тарнавский) http://etraining. mandriva.ru /index.php
9. http://www.pmtf.msiu.ru/chair31/students/spichkov/maxima2.pdf (Методическое пособие по изучению математического пакета Maxima) Математический практикум с применением пакета Maxima. (PDF)
10. Сборник типовых расчетов по высшей математике: Учебное пособие для технических вузов (под ред. Миносцева В.Б.) Изд. 4-е, перераб. М: 2004 – Изд–во: МГИУ – 582 с.

Глоссарий

ALT Linux(Альт Линукс) — дистрибутив GNU/Linux российской команды разработчиков ALT (аббревиатура рекурсивно расшифровывается как ALT Linux Team). Изначально ALT Linux основывался на дистрибутиве MandrakeLinux и представлял собой русскую версию MandrakeLinux. Сейчас ALT Linux является отдельной ветвью развития Linux. Дистрибутив ALT Linux отличается отличной поддержкой русского языка, притом дополнительной настройки системы для этого не нужно — русский язык настраивается во время установки системы. Кроме выпуска дистрибутивов ALT создает и поддерживает Sisyphus (Сизиф) — постоянно обновляемый репозиторий пакетов. Сизиф символизирует постоянный труд команды ALT по усовершенствованию решений, заложенных в репозиторий.

GNU General Public License – (*Универсальная общедоступная лицензия GNU* или *Открытое лицензионное соглашение GNU*) — наиболее популярная лицензия на свободное программное обеспечение, созданная в рамках проекта GNU в 1988 г. Её также сокращённо называют **GNU GPL** или даже просто **GPL**, если из контекста понятно, что речь идёт именно о данной лицензии. GPL лицензия предоставляет получателям компьютерных программ следующие права, или «свободы»: 1) свободу запуска программы, с любой целью; 2) свободу изучения того, как программа работает, и её модификации (предварительным условием для этого является доступ к исходному коду); 3) свободу распространения копий; 4) свободу улучшения программы, и выпуска улучшений в публичный доступ (предварительным условием для этого является доступ к исходному коду).

GNU/Linux (произносится «гну слэш ли́нукс») — свободная UNIX-подобная операционная система. Она основана на системных программах, разработанных в рамках проекта GNU, и на ядре Linux. Зачастую, по историческим причинам и для краткости, эту систему называют просто «Linux». В отличие от большинства других операционных систем, GNU/Linux не имеет единой «официальной» комплектации. Вместо этого GNU/Linux поставляется в большом количестве так называемых дистрибутивов. К операционной системе GNU/Linux также часто относят программы, дополняющие эту операционную систему, и прикладные программы, делающие её полноценной многофункциональной операционной средой.

А́лгебра (от арабского «*аль-джабр*», *«воссоединение»*, *«связь»*, *«завершение* — раздел математики, который можно охарактеризовать как обобщение и расширение арифметики. Слово «алгебра» также употребляется в названиях различных алгебраических систем. В более широком смысле под алгеброй понимают раздел математики, посвящённый изучению операций над элементами множества произвольной природы, обобщающих обычные операции сложения и умножения чисел.

Аргумент — элемент некоторых данных, указываемый при вызове метода.

Вычислительная математика — раздел математики, включающий круг вопросов, связанных с производством вычислений и использованием компьютеров. В более узком понимании вычислительная математика — теория численных методов решения типовых математических задач.

График функции — множество точек, у которых абсциссы являются допустимыми значениями аргумента x, а ординаты — соответствующими значениями функции y.

Графика — (греч. γραφικος — «письменный», от греч. γραφω — «пишу») вид изобразительного искусства, использующий в качестве основных изобразительных средств линии, штрихи и пятна (цвет также может

применяться, но, в отличие от живописи, здесь он играет вспомогательную роль).

Интерфейс, interface — внешний вид класса, объекта или модуля, выделяющий его существенные черты и не показывающий внутреннего устройства и секретов поведения.

Компьютерная алгебра — область математики, лежащая на стыке алгебры и вычислительных методов. Термин «**компьютерная алгебра**» возник как синоним терминов «**символьные вычисления**», «**аналитические вычисления**», «**аналитические преобразования**» и т.д. Даже в настоящее время этот термин на французском языке дословно означает «**формальные вычисления**».

Компью́терная гра́фика (также **маши́нная гра́фика**) — область деятельности, в которой компьютеры используются как для синтеза изображений, так и для обработки визуальной информации, полученной из реального мира. Также компьютерной графикой называют и результат этой деятельности.

Компью́терная програ́мма — последовательность формализованных инструкций, предназначенная для исполнения устройством управления вычислительной машины. Чаще всего образ программы оформляется в виде отдельного файла (исполняемого модуля) или группы файлов.

Кроссплатформенность — свойство программных комплексов, позволяющее им работать в разных операционных системах.

Линия — отрезок прямой, имеющий длину и направление. Линия может меняться по длине, ширине, направлению, кривизне и цвету. Линия может быть двухмерной (линия, оставленная карандашом на бумаге), или пространственной.

Математическая формула (от лат. *formula* — уменьшительное от forma — образ, вид) — всякая символическая запись (в виде выражения, равенства или неравенства), содержащая какую-либо информацию.

Око́нный интерфе́йс — способ организации полноэкранного интерфейса программы, в котором каждая интегральная часть располагается в *окне* — собственном суб–экранном пространстве, находящемся в произвольном месте «над» основным экраном.

Оператор — элемент, использующийся в процессе вычисления выражений.

Операцио́нная систе́ма, **ОС** (англ. *operating system*) — базовый комплекс компьютерных программ, обеспечивающий управление аппаратными средствами компьютера, работу с файлами, ввод и вывод данных, а также выполнение прикладных программ и утилит. При включении компьютера операционная система загружается в память раньше остальных программ и затем служит платформой и средой для их работы. Помимо вышеуказанных

функций ОС может осуществлять и другие, например, предоставление пользовательского интерфейса, сетевое взаимодействие и т. п.

Полярная система координат — система координат, ставящая в соответствие каждой точке на плоскости пару чисел ρ, φ. Основными понятиями этой системы являются точка отсчёта (*полюс*) и луч, начинающийся в этой точке (*полярная ось*). Координата ρ определяет расстояние от точки до полюса, координата φ — угол между полярной осью и отрезком, соединяющим полюс и рассматриваемую точку

Програ́ммное обеспе́чение — одна из составляющих информационных технологий, включающая компьютерные программы и данные, предназначенные для решения определённого круга задач и хранящиеся на машинных носителях. Программное обеспечение представляет собой либо данные для использования в других программах, либо алгоритм, реализованный в виде последовательности инструкций для процессора. В компьютерном жаргоне часто используется слово «софт» от английского **software**, которое, в этом смысле впервые применил Джон Тьюки (John W. Tukey) в 1958 г. В области вычислительной техники и программирования **программное обеспечение** — это совокупность всей информации, данных и программ, которые обрабатываются компьютерными системами.

Прямоугольная система координат в пространстве – образуется тремя взаимно перпендикулярными осями координат. Оси координат пересекаются в точке О, которая называется началом координат, на каждой оси выбрано положительное направление, указанное стрелками, и единица измерения отрезков на осях. Единицы измерения одинаковы для всех осей. OX — ось абсцисс, OY — ось ординат, OZ — ось аппликат.

Прямоугольная система координат на плоскости – образуется двумя взаимно перпендикулярными осями координат. Оси координат пересекаются в точке О, которая называется началом координат, на каждой оси выбрано положительное направление, указанное стрелками, и единица измерения отрезков на осях.

Прямоугольная, или **декартова система координат** — наиболее распространённая система координат на плоскости и в пространстве.

Рене Декарт – впервые ввел прямоугольную систему координат в своей работе «Рассуждение о методе» в 1637 году. Поэтому прямоугольную систему координат называют также — **Декартова система координат**.

Репозиторий — место, где хранятся и поддерживаются какие-либо данные. Чаще всего данные в репозитории хранятся в виде файлов, доступных для дальнейшего распространения по сети. Примером репозитория может служить репозиторий свободного программного обеспечения Сизиф ALT Linux.

Свободное программное обеспечение — программное обеспечение, в отношении которого пользователь обладает «четырьмя свободами»: запускать,

изучать, распространять и улучшать программу. Распространяется под свободными лицензиями, например GPL

Символ — элемент алфавита, а так же элемент данных текстового типа.

Системы компьютерной алгебры (СКА), (или системы символьных вычислений) — такие программные продукты, как Maxima, Maple, Mathematica, Reduce, MuPAD, Derive, Magma, Macsyma, Mathomatic, Axiom, GAP, FreeMat, Octave, Scilab, YACAS и другие.

Трансценде́нтное число́ (от лат. *transcendere* — переходить, превосходить) — это число, не являющееся алгебраическим, иными словами, число, не являющееся корнем многочлена с рациональными коэффициентами.

Число *e* — математическая константа, основание натурального логарифма, иррациональное и трансцендентное число. Иногда число *e* называют *числом Эйлера* или *неперовым числом*. Играет важную роль в дифференциальном и интегральном исчислении, а также многих других разделах математики, $e \approx 2.71828\ 18284\ 59\ldots$

Число π (произносится **«пи»**) — математическая константа, выражающая отношение длины окружности к длине её диаметра. Обозначается буквой греческого алфавита «пи», $\pi \approx 3.14159\ 26535\ 89793\ ...$