Введение

Тема: Основы использования и работы в прикладной компьютерной программе (системе компьютерной алгебры) Maxima.

Цель:

1. Познакомиться с основными командами системы компьютерной алгебры Maxima:
   - Работа с переменными и арифметическими операциями
   - Работа со встроенными математическими и с пользовательскими функциями
   - Работа со списками

Примечания:

1. Каждое задание лабораторной работы надо выполнять в отдельном файле.
2. Формат имени файла: "ФИО студента, номер группы/подгруппы, тема 3, ЛР (задание …).wxmx".

Требования к отчету по работе:

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

Понятие «компьютерной алгебры»

Компьютерные программы по математике делят на:

1. Программы, предназначенные для вычислений, которые работают с числовыми выражениями (например, Excel). Их результаты, обычно, бывают приближенными, они оперируют с вещественными числами, представленными в виде бесконечных десятичных дробей, поэтому при вычислениях происходит их округление.
2. Программы, которые наряду с математическими вычислениями и построением графиков, проводят символьные преобразования, вычисления в символьном виде, например, производных или первообразных заданной функции, находят корни различных уравнений и систем уравнений и т.д. такие программы называют – «системы компьютерной алгебры» (например, Mathematica, MatLab, Maxima и т.п.).

Компьютерная алгебра – область математики, лежащая на стыке алгебры и вычислительных методов. Для нее, как и для любой области, лежащей на стыке различных наук, трудно определить четкие границы. Часто говорят, что к компьютерной алгебре относятся вопросы, слишком алгебраические, чтобы содержаться в учебниках по вычислительной математике и слишком вычислительные, чтобы содержаться в учебниках по алгебре. При этом ответ на вопрос о том, относится ли конкретная задача к компьютерной алгебре, часто зависит от склонностей специалиста.

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».

В чем основные отличия символьных вычислений от численных и почему возник термин компьютерная алгебра?

Когда говорим о вычислительных методах, то считаем, что все вычисления выполняются в поле вещественных или комплексных чисел. В действительности же всякая программа для ЭВМ имеет дело

только с конечным набором рациональных чисел, поскольку только такие числа представляются в компьютере. Для записи целого числа отводится обычно 16 или 32 двоичных символа (бита), для вещественного – 32 или 64 бита. Это множество не замкнуто относительно арифметических операций, что может выражаться в различных переполнениях, например, при умножении достаточно больших чисел или при делении на маленькое число. Еще более существенной особенностью вычислительной математики является то, что арифметические операции над этими числами, выполняемые компьютером, отличаются от арифметических операций в поле рациональных чисел, более того, для компьютерных операций не выполняются основные аксиомы поля (ассоциативности, дистрибутивности). Эти особенности компьютерных вычислений выражаются в терминах погрешности или точности вычислений. Оценка погрешности представляет одну из основных проблем вычислительной математики. Каждую задачу требуется решить с использованием имеющихся ресурсов ЭВМ, за обозримое время, с заданной точностью.

Набор объектов, применяемых в символьных вычислениях, весьма разнообразен, в частности, в них используется значительно большее множество рациональных чисел. Это множество все равно остается конечным, но ограничения на допустимые размеры числа (количество знаков в его записи) связаны обычно с размерами оперативной памяти ЭВМ, что позволяет пользоваться практически любыми рациональными числами, операции над которыми выполняются за приемлемое время. При этом компьютерные операции над рациональными числами совпадают с соответствующими операциями в поле рациональных чисел. Таким образом, снимается одна из основных проблем вычислительных методов – оценка погрешности вычислений.

В компьютерной алгебре практически не применяются вещественные и комплексные числа, зато широко используется алгебраические числа. Алгебраическое число задается своим минимальным многочленом, а иногда для его задания требуется указать интервал на прямой или область в комплексной плоскости, где содержится единственный корень данного многочлена. Многочлены играют в символьных вычислениях исключительно важную роль. На использовании полиномиальной арифметики основаны теоретические методы аналитической механики, они используются во многих областях математики, физики и других наук. Кроме того, в компьютерной алгебре рассматриваются такие объекты, как дифференциальные поля (функциональные поля), допускающие показательные, логарифмические, тригонометрические функции, матричные кольца (элементы матрицы принадлежат кольцам достаточно общего вида) и другие. Даже при арифметических операциях над такими объектами происходит разбухание информации, для записи промежуточных результатов вычислений требуется значительный объем памяти ЭВМ.

Ограничения на алгоритмы решаемых компьютерной алгеброй задач накладываются имеющимися ресурсами ЭВМ и обозримостью времени счета. Однако ограничения по времени счета и по используемой памяти в символьных вычислениях существенно более обременительны, чем в вычислительных методах.

В научных исследованиях и технических расчетах специалистам приходится гораздо больше заниматься преобразованиями формул, чем собственно численным счетом, однако с появлением ЭВМ основное внимание уделялось автоматизации последнего, хотя ЭВМ начали применяться для решения таких задач символьных вычислений, как, например, символьное дифференцирование, еще в 50-х годах прошлого века. Активная разработка систем компьютерной алгебры началась в конце 60-х годов. С тех пор создано значительное количество различных систем, получивших различную степень распространения; некоторые системы продолжают развиваться, другие отмирают, постоянно появляются новые.

Под системами компьютерной алгебры (или системами символьных вычислений) подразумевают такие программные продукты, как Maple, Maxima, Reduce, Derive и другие.

Основы работы в системе компьютерной алгебры Maxima

Основными **преимуществами программы** являются:

1. Возможность свободного использования (Maxima относится к классу свободных программ и распространяется на основе лицензии GNU).

   GNU (General Public License, Универсальная общедоступная лицензия GNU или Открытое лицензионное соглашение GNU) – наиболее популярная лицензия на свободное программное обеспечение, созданная в 1988 г. Её также сокращённо называют GNU GPL или даже просто GPL.

   Эта лицензия предоставляет пользователям компьютерных программ следующие права:

   1) свободу запуска программы, с любой целью;
   2) свободу изучения того, как программа работает, и её модификации;
   3) свободу распространения копий;
   4) свободу улучшения программы, и выпуска улучшений в публичный доступ;

2. Возможность функционирования под управлением различных ОС (в частности Linux и Windows);
3. Небольшой размер программы (дистрибутив занимает порядка 23 мегабайт, в установленном виде со всеми расширениями потребуется около 80 мегабайт);
4. Maxima имеет удобный графический интерфейс (wxMaxima) на русском языке, а также есть возможность работать в режиме командной строки.
5. Maxima дает возможность решать широкий класс задач.

   Например:

   - Операции с полиномами (манипуляция рациональными и степенными выражениями, вычисление корней и т.п.)
   - Вычисления с элементарными функциями, в том числе с логарифмами, экспоненциальными функциями, тригонометрическими функциями
   - Вычисления со специальными функциями, в том числе эллиптическими функциями и интегралами
   - Вычисление пределов и производных
   - Аналитическое вычисление определённых и неопределённых интегралов
   - Решение интегральных уравнений
   - Решение алгебраических уравнений и их систем
   - Операции со степенными рядами и рядами Фурье
   - Операции с матрицами и списками, большая библиотека функций для решения задач линейной алгебры
   - Операции с тензорами
   - Теория чисел, теория групп, абстрактная алгебра
   - И другие (при установке дополнительных пакетов)

Недостаток: отсутствие справки на русском языке, но в сети Интернет присутствует большое количество статей с примерами использования Maxima.

В Maxima 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 разрабатываемым версиям, с чётным – к стабильным версиям.

Установить последнюю версию программы на свой компьютер можно с ее сайта в сети Интернет: http://maxima.sourceforge.net/. Русская локализация сайта: http://maxima.sourceforge.net/ru/.

Открытие/запуск программы

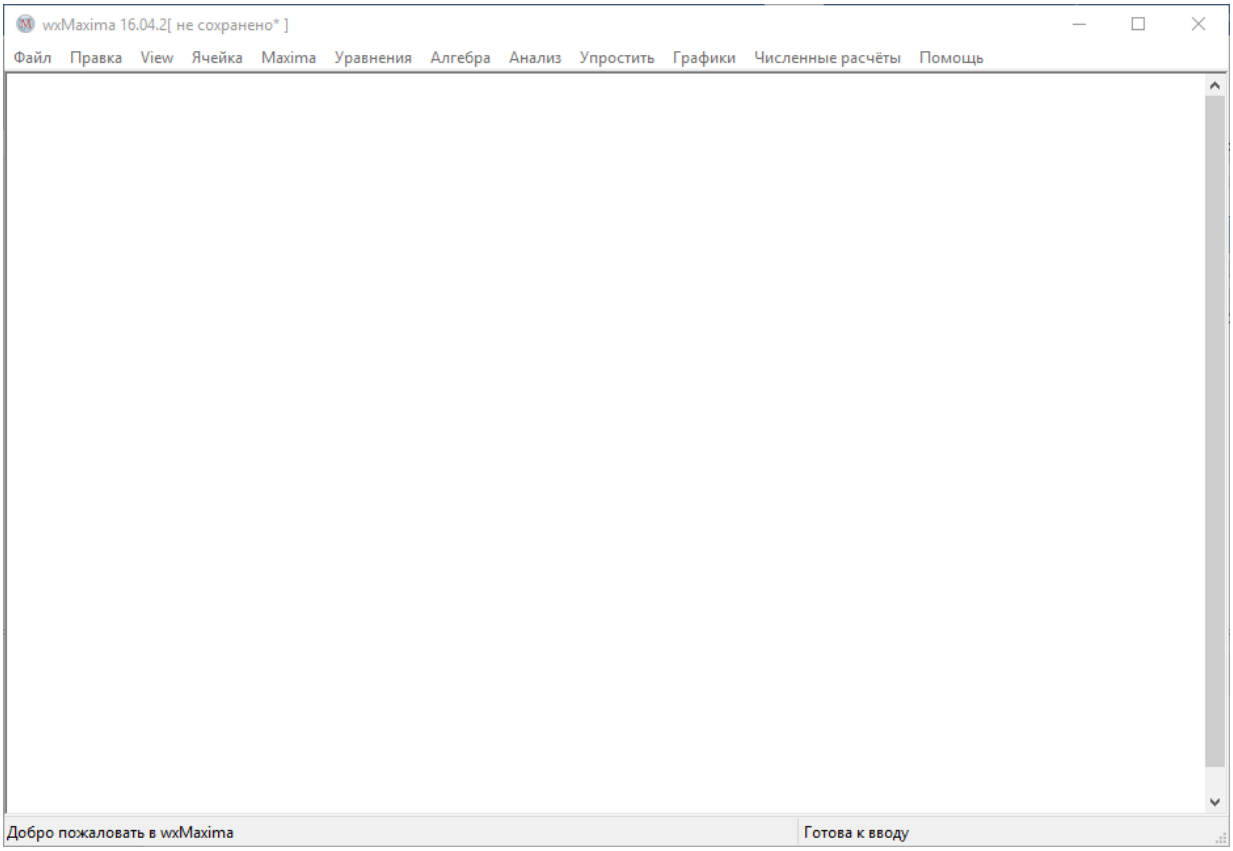
1 способ. При помощи ярлыка на рабочем столе.
2 способ. Кнопка «Пуск» – набрать «wxMaxima» – выбрать «wxMaxima, Классическое приложение».

Интерфейс программы

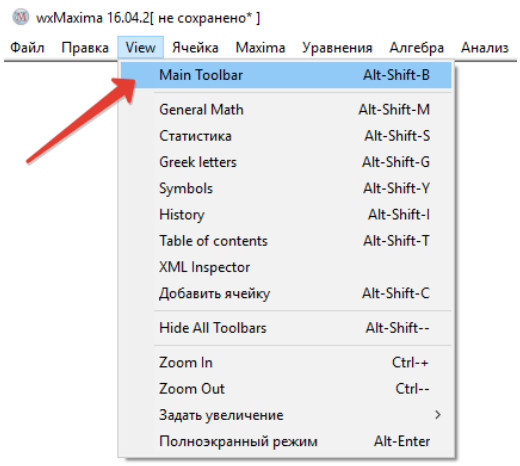
После запуска Maxima появляется окно программы, в верхней графической части окна интерфейса указано, какая загружена версия.

В верхней части окна есть главное меню. Затем – рабочее поле. Внизу – строка состояния.

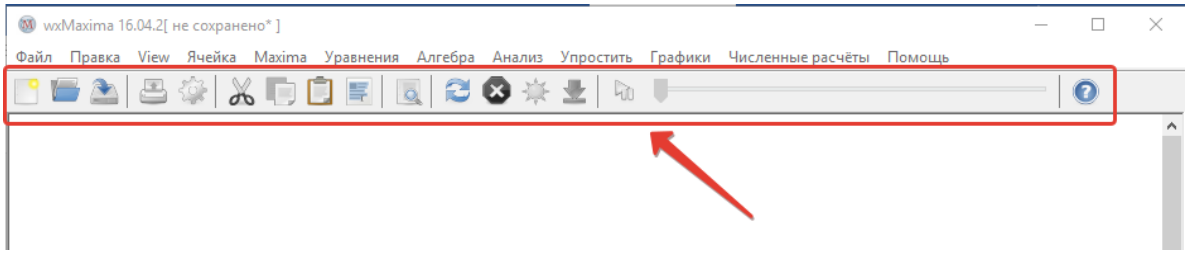
В главном меню находятся группы функций: Уравнения, Алгебра, Анализ, Упростить, Графики, Численные расчеты. Этих функций достаточно для решения большого количества типовых математических задач.



Для открытия главной панели инструментов надо нажать: «View (Вид) – Main Toolbar (Главная панель инструментов)».



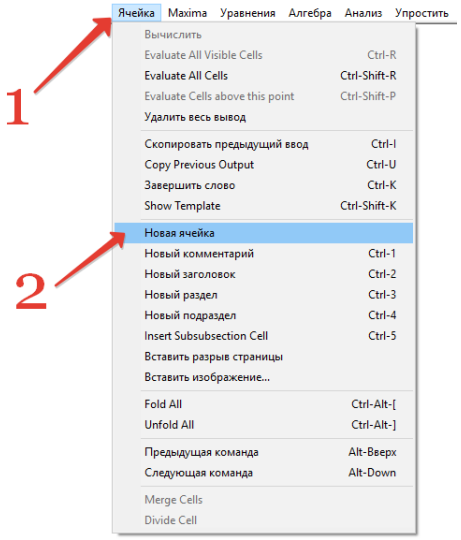
Итог:



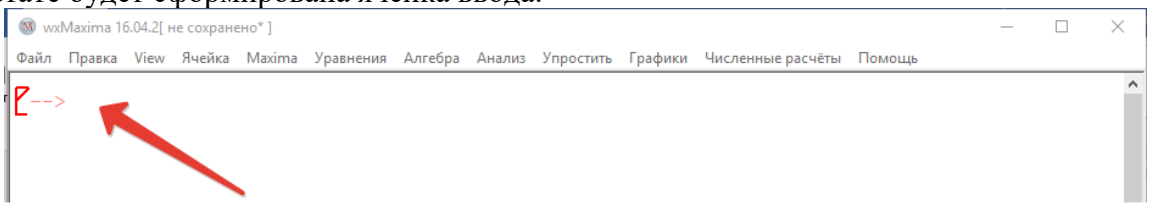
Ввод простейших команд

В разных версиях программы **Ввод команды** начинается с:

- 1 вариант. Нажатия клавиши «Enter».
- 2 вариант. Начинать печатать команду
- 2 вариант. Выполнения команды «Ячейка – Новая ячейка»



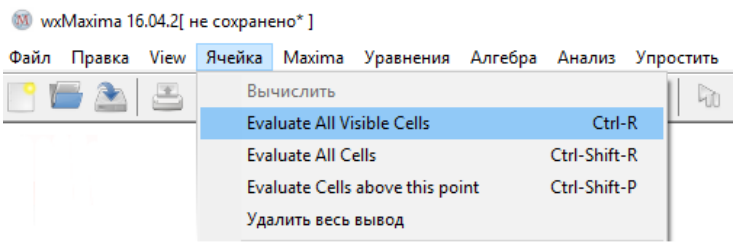
В результате будет сформирована ячейка ввода.



Разделителем команд является символ « ; » (в ранних версиях Maxima и некоторых ее оболочках наличие точки с запятой после каждой команды строго обязательно, поэтому рекомендуется добавлять « ; » после каждой команды).

После ввода команды для ее обработки и вывода результата необходимо:

- Либо нажать сочетание клавиш «Shift + Enter».
- Либо выполнить команду главного меню «Ячейка – Evaluate All Visible Cells».



После ввода **каждой команде присваивается порядковый номер** (%i1), (%i2), (%i3) и т.д. Результаты вычислений имеют соответственно порядковый номер (%o1), (%o2) и т. д. Где «i» –сокращение от англ. input (ввод), а «o» – англ. output (вывод).

Это 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).

Если вместо точки с запятой завершить ввод символом $, то это позволяет вычислить **результат** введённой команды, но **не выводить** его **на экран**.

Если нужно **отобразить выражение** (а **не вычислить** его), то перед ним необходимо поставить знак « ‘ » (одиночная кавычка). Примечание: одиночная кавычка расположена на той же клавише, что и русская буква «э». Набирать кавычку надо в английском режиме. Не путайте с апострофом!

Но этот метод не работает, когда выражение имеет явное значение, например, выражение sin(π) заменяется на значение, равное нулю.

Если поставить перед выражением две одиночных кавычки последовательно, применённые к выражению во входной строке, то это приводит к замещению входной строки результатом вычисления вводимого выражения.

Последняя выполненная команда может обозначаться знаком « % ». В этом случае номер команды указывать не нужно.

Задание 3.1

- Откройте новый файл.
- Сохраните файл. Формат имени файла: "ФИО студента, номер группы/подгруппы, тема 3, ЛР (задание 3.1).wxmx"

- **Постепенно** введите следующие выражения (то есть после каждой команды нажимайте сочетание клавиш «Shift + Enter» или выполняйте соответствующую команду главного меню).

```
aa:1024;
bb:19;
sqrt(aa)+bb;
’(sqrt(aa)+bb);
”%;
%i1+%i2;
(%i1)+(%i2);
(%o1)+(%o2);
```

- Проверьте результат – на экране будет следующее:

```
(%i1)   aa:1024;
(aa)    1024

(%i2)   bb:19;
(bb)    19

(%i3)   sqrt(aa)+bb;
(%o3)   51

(%i4)   '(sqrt(aa)+bb);
(%o4)   bb+√aa

(%i5)   ''%;
(%o5)   51

(%i6)   %i1+%i2;
(%o6)   (bb:19)+aa:1024

(%i7)   (%i1)+(%i2);
(%o7)   (bb:19)+aa:1024

(%i8)   (%o1)+(%o2);
(%o8)   1043
```

- Проанализируйте команды. Комментарии к командам:

№	Команда	Комментарий
1	aa:1024;	Переменной «aa» присваивается значение 1024
2	bb:19;	Переменной «bb» присваивается значение 19
3	sqrt(aa)+bb;	К значению sqrt(aa)=1024 прибавляется зачение bb=19
4	’(sqrt(aa)+bb);	Вывод на экран выражения sqrt(aa)+bb Благодаря одиночной кавычке, происходит вывод выражения на экран.
5	”%;	Вывод на экран значения выражения sqrt(aa)+bb. Именно это выражение является последней выполненной командой. Благодаря двум одиночным кавычкам, происходит вычисление этого выражения.
6	%i1+%i2;	Вывод на экран суммы первой команды и второй команды.
7	(%i1)+(%i2);	Вывод на экран суммы первой команды и второй команды.
8	(%o1)+(%o2);	Вывод на экран суммы: (значение первой команды) + (значение второй команды).

Примечание:
Например, в третьей команде вместо « sqrt(aa)+bb; » можно было написать «sqrt(aa)+ (%o2); ».

Можно **вводить несколько команд в одной строке**. Каждая команда отделяется от другой символом «точка с запятой». При этом будет сформировано столько строк вывода, сколько команд было задано в строке ввода.

- Введите одной строкой:
  4+7; 25/5; 3^6;

- Проверьте результат – на экране будет следующее:

```
(%i11)   4+7; 25/5; 3^6;
(%o9)    11
(%o10)   5
(%o11)   729
```

В данном случае одна строка ввода (%i11), состоящая из трех команд. Но при этом три строки вывода, каждая из которых соответствует введенным командам.

При вводе нескольких команд можно **блокировать вывод отдельных команд**. В этом случае в конце команды вводят не точку с запятой, а знак доллара.

- Введите одной строкой:
  a: 6$ b:7$ a+b;

- Проверьте результат – на экране будет следующее:

```
(%i14)   a: 6$ b:7$ a+b;
(%o14)   13
```

В данном случае нет вывода (%o12) и вывода (%o13). Сразу организован вывод (%o14).

Если необходимо **добавить команду** между уже введенными командами, то необходимо выполнить следующий порядок действий:

1. Нажать левой кнопкой мыши между двумя набранными командами – появится черная горизонтальная линия.
2. Добавить ячейку ввода: «Ячейка – Новая ячейка».
3. Вводить новую команду.

Если необходимо **пересчитать значение команды** (после внесенных изменений), то необходимо поставить курсор в соответствующей строке ввода и нажать сочетание клавиш Ctrl + Enter.

- Добавьте ячейку после строки вывода %o8.
- Увеличьте последний результат в 2 раза.
- Сохраните файл.

Особенностью графического интерфейса системы wxMaxima является то, что при открытии ранее сохраненного документа в рабочем окне выводятся только команды, все же **ячейки с результатами не отображаются**. **Для** их **вывода** можно воспользоваться командой «Evaluate all cells» пункта меню «Ячейка».

- Закройте файл.
- Откройте его и выполните показ результатов вычислений: «Ячейка – Evaluate all cells».

Для удаления ячейки необходимо ее выделить и, например, нажать на клавиатуре клавишу Delete.

- Наберите любую команду в конце. Затем удалите её.

В системе Maxima можно **добавлять** в документ **текстовые комментарии**. Для этого выбираем пункт меню «Ячейка – Новый комментарий» («Cell – New Text Cell (Insert Text Cell)»), после чего с клавиатуры набираем текст.

- Добавьте в конце документа текстовый комментарий со своими ФИО.

Также можно **добавлять заголовки** («Ячейка – Новый заголовок» / «Cell – Insert Title Cell»), **пункты меню (секция, раздел)** («Ячейка – Новый раздел» / «Cell – Insert Section Cell»), **пункты подменю (подсекция, подраздел)** («Ячейка – Новый подраздел» / «Cell – Insert Subsection Cell») и так далее.

- Добавьте в конце документа заголовок с названием факультета (института).
- Добавьте в конце документа раздел с названием специальности (направления, профиля).
- Добавьте в конце документа подраздел с номером группы / подгруппы.

- Сохраните файл.

Используемые обозначения для ввода команд

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 слеш).

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.

Преобразование к форме с плавающей точкой осуществляет также функция **float**.

- Откройте новый файл.
- Сохраните файл. Формат имени файла: "ФИО студента, номер группы/подгруппы, тема 3, ЛР (задание 3.2).wxmx"

- Выполните команды:

```
(%i1) 2/3+5/6;
(%o1) 3/2
(%i2) 2/3+5/6,numer;
(%o2) 1.5
```

Константы

В Maxima для удобства вычислений есть ряд встроенных констант:

Название	Обозначение
π (число Пи)	%pi
e (экспонента)	%e
+∞ (плюс бесконечность)	inf
-∞ (минус бесконечность)	minf
Комплексная бесконечность	infinity
Мнимая единица ($\sqrt{-1}$)	%i
Истина	true
Ложь	false

- Выполните команды:

(%i3) ?pi;
(%o3) 3.141592653589793

(%i4) sin(pi)
(%o4) sin(π)

(%i5) sin(%pi)
(%o5) 0

(%i6) log(e)
(%o6) log(e)

(%i7) log(%e)
(%o7) 1

(%i8) log(1900),numer;
(%o8) 7.549609165154532

Арифметические операции

Для обозначения арифметических операций в Maxima используются математические знаки: «+» - сложение, «–» - вычитание, «*» - умножение, «/»- деление.

Возведение в степень можно обозначать тремя способами: ^ , ^^ , **.

Извлечение корня степени n записывают, как степень ^^(1/n).

Также в Maxima встроена еще одна полезная операция – нахождение факториала числа. Эта операция обозначается восклицательным знаком.

Например, 6!=1*2*3*4*5*6=120.

Для нахождения полуфакториала числа используют команду !!

Полуфакториал – это произведение всех четных (для четного операнда) или нечетных чисел (для нечетного операнда), меньших либо равных данному числу.

Например, 10!! = 2*4*6*8*10, а 9!! = 1*3*5*7*9

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

- Придумайте несколько своих команд, использующих все рассмотренные арифметические операции. Действия можно и нужно группировать (то есть в одном примере может быть несколько действий)
  - ✓ Сложение « + »
  - ✓ Вычитание « – »
  - ✓ Умножение « * »
  - ✓ Деление « / »
  - ✓ Возведение в степень « ^ »
  - ✓ Возведение в степень « ^^ »
  - ✓ Возведение в степень « ** »
  - ✓ Извлечение корня степени n « ^^(1/n) »
  - ✓ Нахождение факториала числа « ! »
  - ✓ Нахождение полуфакториала числа « !! »
  - ✓ Использование круглых скобок.

- Выполните придуманные команды в программе Maxima.
- Сохраните файл.

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 – «переменной x присвоено значение 5»
- ✓ b: a^2+3 – «переменная b будет иметь значение равное a^2+3».

- Выполните команды:
  - ✓ x: 5
  - ✓ b: a^2+3

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.
Кроме того, kill начинает новую нумерацию для исполняемых команд.

- Выполните команды:
  - ✓ kill(x)
  - ✓ kill(all)

Работа с функциями

Математические функции

В Maxima имеется достаточно большой набор встроенных математических функций. Для записи функции необходимо указать ее название, а затем, в круглых скобках записать через запятую значения аргументов.

Например, sin(x);

Следует иметь в виду, что некоторые названия функций отличаются от названий, используемых в математике:

Функции	**Обозначение**	
Тригонометрические [1]	sin(x)	(синус)
	cos(x)	(косинус)
	tan(x)	(тангенс)
	cot(x)	(котангенс)
	sec(x)	(секанс, $\frac{1}{\cos(x)}$)
	csc(x)	(косеканс, $\frac{1}{\sin(x)}$)
	Нет отдельного обозначения	версинус
	Нет отдельного обозначения	коверсинус
	Нет отдельного обозначения	гаверсинус

[1] Смотри Приложение 1

Функции	Обозначение	
	Нет отдельного обозначения	экссеканс
	Нет отдельного обозначения	экскосеканс
Обратные тригонометрические	asin(x)	(арксинус)
	acos(x)	(арккосинус)
	atan(x)	(арктангенс)
	acot(x)	(арккотангенс)
Гиперболические	sinh(x)	(гиперболический синус)
	cosh(x)	(гиперболический косинус)
	tanh(x)	(гиперболический тангенс)
	coth(x)	(гиперболический котангенс)
	sech(x)	(гиперболический секанс)
	csch(x)	(гиперболический косеканс)
Натуральный логарифм	log(x)	
Остаток от деления переменной «x» на переменную «y»	mod(x,y)	
Квадратный корень	sqrt(x)	
Модуль	abs(x)	
Минимальный элемент из списка	min(x1,x2,…,xn)	
Максимальный элемент из списка	max(x1, x2,…,xn)	
Экспонента	exp(x)	

Для отдельных функций необходимо приписывать к названию функции « , numer »
Например:

- ✓ log(1),numer;
- ✓ exp(1),numer;

- Откройте новый файл.
- Сохраните файл. Формат имени файла: "ФИО студента, номер группы/подгруппы, тема 3, ЛР (задание 3.3).wxmx"

- Найдите значение всех тригонометрических функций для угла 60° (угол надо брать в радианах). Примечание: рассмотреть в том числе функции, указанные в приложении 1.
- Найдите значение всех обратных тригонометрических функций для значения (самостоятельно выберите нужные, известные вам, значения для каждой обратной тригонометрической функции).
- Найдите значение всех гиперболических функций для угла 60° (угол надо брать в радианах).
- Найдите значение натурального логарифма от суммы (номер вашего дома + номер квартиры + дата рождения + месяц рождения + год рождения). Сумму задайте отдельной переменной.
- Найдите остаток от деления вашего года рождения на номер дома, в котором вы живёте.
- Возведите в шестую степень дату (число) вашего рождения. А затем найдите значение квадратного корня от полученного значения.
- Найдите модуль любого отрицательного числа.
- Выполните действия:

```
max(1/5, 1/7, 4/8, 20/18, 3/4, 5/7);
min(1/5, 1/7, 4/8, 20/18, 3/4, 5/7);
```

- Существует функция, возвращающая знак числа: signum(x). Выполните действия, при x = дате вашего рождения:

```
signum(-5);signum(7+x^2);
```

- Вычислите значение экспоненты от введённого « x ».

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.

- Выполните действия, проверьте и проанализируйте результат. Примечание: нумерация команд может быть другой.

```
(%i1) f(x):=x^2;
(%o1) f(x):=x^2
(%i2) f(3+sin(%pi/2));
(%o2) 16
```

Зарезервированные слова

Зарезервированные слова, использование которых в качестве имён переменных вызывает синтаксическую ошибку: and, at, diff, do, else, elseif, for, from, if, in, integrate, limit, next, or, product, step, sum, then, thru, unless, while.

- Сохраните файл.

Списки. Работа со списками.

Списки – базовые строительные блоки для Maxima. Чтобы создать список необходимо в квадратных скобках записать все его элементы через запятую.

Список может состоять из одного элемента: в квадратных скобках указывается один элемент.

Список может быть пустым: указываются пустые квадратные скобки.

- Откройте новый файл.
- Сохраните файл. Формат имени файла: "ФИО студента, номер группы/подгруппы, тема 3, ЛР (задание 3.4).wxmx"

- Постепенно выполните действия. И проверьте результаты. Примечание: нумерация команд может быть другой.

```
(%i1) a1:[1,6,-3,s,2*t];
(%o1) [1,6,-3,s,2 t]
(%i2) a2:[4];
(%o2) [4]
(%i3) b:[];
(%o3) []
```

- Элементом списка может и другой список. Постепенно выполните действия. И проверьте результаты. Примечание: нумерация команд может быть другой.

```
(%i4) d:[1,2,[6,-7],[s,t,q]];
(%o4) [1,2,[6,-7],[s,t,q]]

(%i5) r:[a1,a2,3,8];
(%o5) [[1,6,-3,s,2 t],[4],3,8]
```

- Ссылка на элемент списка. Чтобы вывести на экран один из элементов списка нужно записать имя списка, а затем в квадратных скобках указать номер интересующего элемента. Постепенно выполните действия. И проверьте результаты. Примечание: нумерация команд может быть другой.

```
(%i6) d[1];
(%o6) 1
(%i7) r[2];
(%o7) [4]
(%i8) r[1];
(%o8) [1,6,-3,s,2 t]
```

- Пример работы со списками. Постепенно выполните действия. И проанализируйте устно результат.

list1:[1,2,3,x,x+y];
list2:[];
list3:[3];
list4:[1,2,[3,4],[5,6,7]];
list4[1];
list4[3];
list4[3][2];

Функции для элементарных операций со списками

Функция **length** возвращает число элементов списка (при этом элементы списка сами могут быть достаточно сложными конструкциями):

- Постепенно выполните действия. И проанализируйте устно результат.

length(list4);
length(list3);

Примечание: list4:[1,2,[3,4],[5,6,7]]; и list3:[3];

Функция **copylist(expr)** возвращает копию списка expr:

- Постепенно выполните действия. И проанализируйте устно результат.

list1:[1,2,3,x,x+y];
list2:copylist(list1);

Функция **makelist** создаёт список, каждый элемент которого генерируется из некоторого выражения. Возможны два варианта вызова этой функции:

- ✓ **makelist(expr,i,i0,i1)** – возвращает список, j-й элемент которого равен ev(expr,i = j), при этом индекс j меняется от i0до i1.

  То есть общий вид функции имеет вид:

  makelist(выражение, переменная, начальное значение, конечное значение)

  - Выражение – это некоторое выражение (функция), в которое подставляются значения переменной.
  - Переменная – имя аргумента в выражении.
  - Начальное значение – первое подставляемое значение.
  - Конечное значение – последнее подставляемое значение.

- Выполните действие. И проанализируйте устно результат.

makelist(2*x,x,1,5);
[2,4,6,8,10]

To есть вместо переменной «x» в выражение «2*x» подставляются значения 1, 2, 3, 4, 5

- ✓ **makelist(expr,x,list)** – возвращает список, j-й элемент которого равен ev(expr,x = list[j]), при этом индекс j меняется от 1 до length(list).
  В данном случае значения переменной берутся из списка list – от первого значения до последнего значения.

- Постепенно выполните действия. И проверьте результаты. Примечание: нумерация команд может быть другой.

```
(%i1) makelist(concat(x,i),i,1,6);
(%o1) [x1,x2,x3,x4,x5,x6]

(%i2) list:[1,2,3,4,5,6,7];
(%o2) [1,2,3,4,5,6,7]

(%i3) makelist(exp(i),i,list);
(%o3) [e,e^2,e^3,e^4,e^5,e^6,e^7]
```

Во многом аналогичные действия выполняет функция **create_list(form,x1,list1,...,xn,listn)**.

Эта функция строит список путём вычисления выражения form, зависящего от x1, к каждому элементу списка list1(аналогично form, зависящая и от x2, применяется к list2и т.д.).

- Постепенно выполните действия. И проверьте результаты. Примечание: нумерация команд может быть другой.

```
(%i1) create_list(x^i,i,[1,3,7]);
(%o1) [x,x^3,x^7]

(%i2) create_list([i,j],i,[a,b],j,[e,f,h]);
(%o2) [[a,e],[a,f],[a,h],[b,e],[b,f],[b,h]]
```

Функция **append** позволяет склеивать списки. При вызове **append (list_1, \dots, list_n)** возвращается один список, в котором за элементами list1 следуют элементы list2 и т.д. вплоть до listn.

- Постепенно выполните действия. И проверьте результаты. Примечание: нумерация команд может быть другой.

```
(%i1) append([1],[2,3],[4,5,6,7]);
(%o1) [1,2,3,4,5,6,7]
```

Создать новый список, компонуя элементы двух списков поочерёдно в порядке следования, позволяет функция **join(k,m)**. Новый список содержит k1, затем m1, затем k2, m2 и т.д.

- Постепенно выполните действия. И проверьте результаты. Примечание: нумерация команд может быть другой.

```
(%i1) join([1,2,3],[10,20,30]);
(%o1) [1,10,2,20,3,30]

(%i2) join([1,2,3],[10,20,30,40]);
(%o2) [1,10,2,20,3,30]
```

Длина полученного списка ограничивается минимальной длиной списков k и m.

Функция **cons(expr,list)** создаёт новый список, первым элементом которого будет expr, а остальные – элементы списка list.

Функция **endcons(expr,list)** также создаёт новый список, первые элементы которого – элементы списка list, а последний – новый элемент expr.

- Постепенно выполните действия. И проверьте результаты. Примечание: нумерация команд может быть другой.

```
(%i1) cons(x,[1,2,3]);
(%o1) [x,1,2,3]

(%i2) endcons(x,[1,2,3]);
(%o2) [1,2,3,x]
```

Функция **reverse** меняет порядок элементов в списке на обратный.

- Постепенно выполните действия. И проверьте результаты. Примечание: нумерация команд может быть другой.

```
(%i5) list1:[1,2,3,x];
(%o5) [1,2,3,x]

(%i6) list2:reverse(list1);
(%o6) [x,3,2,1]
```

Функция **member(expr1,expr2)** возвращает true, если expr1 является элементом списка expr2, и false в противном случае.

- Постепенно выполните действия. И проверьте результаты. Примечание: нумерация команд может быть другой.

```
(%i1) member (8, [8, 8.0, 8b0]);
(%o1) true

(%i2) member (8, [8.0, 8b0]);
(%o2) false

(%i3) member (b, [[a, b], [b, c]]);
(%o3) false

(%i4) member ([b, c], [[a, b], [b, c]]);
(%o4) true
```

Функция **rest(expr)** выделяет остаток после удаления первого элемента списка expr. Можно удалить первые n элементов, используя вызов rest(expr,n). Примечание: список expr при этом не изменяется.

Функция **last(expr)** выделяет последний элемент списка expr (аналогично first – первый элемент списка). Примечание: список expr при этом не изменяется.

- Постепенно выполните действия. И проверьте результаты. Примечание: нумерация команд может быть другой.

```
(%i1) list1:[1,2,3,4,a,b];
```

(%o1) [1,2,3,4,a,b]

(%i2) rest(list1);
(%o2) [2,3,4,a,b]

(%i3) rest(%);
(%o3) [3,4,a,b]

(%i4) last(list1);
(%o4) b

(%i5) rest(list1,3);
(%o5) [4,a,b]

Суммирование и перемножение списков (как и прочих выражений) осуществляется функциями sum и product.

Функция **sum(expr,i,in, ik)** суммирует значения выражения expr при изменении индекса i от in до ik.

Функция **product(expr,i,in,ik)** перемножает значения выражения expr при изменении индекса i от in до ik.

- Постепенно выполните действия. И проверьте результаты. Примечание: нумерация команд может быть другой.

(%i1) product (x + i*(i+1)/2, i, 1, 4);
(%o1) (x + 1) (x + 3) (x + 6) (x + 10)

Примечание:

- Вместо «i» в выражение «x + i*(i+1)/2» подставили 1 и получили «x + 1*(1+1)/2», что равно x + 1.
- Вместо «i» в выражение «x + i*(i+1)/2» подставили 2 и получили «x + 2*(2+1)/2», что равно x + 3.
- Вместо «i» в выражение «x + i*(i+1)/2» подставили 3 и получили «x + 3*(3+1)/2», что равно x + 6.
- Вместо «i» в выражение «x + i*(i+1)/2» подставили 4 и получили «x + 4*(4+1)/2», что равно x + 10.
- Перемножили полученные выражения.
- То есть получили выражение (x + 1) (x + 3) (x + 6) (x + 10)

(%i2) sum (x + i*(i+1)/2, i, 1, 4);
(%o2) 4x + 20

Примечание:

- Вместо «i» в выражение «x + i*(i+1)/2» подставили 1 и получили «x + 1*(1+1)/2», что равно x + 1.
- Вместо «i» в выражение «x + i*(i+1)/2» подставили 2 и получили «x + 2*(2+1)/2», что равно x + 3.
- Вместо «i» в выражение «x + i*(i+1)/2» подставили 3 и получили «x + 3*(3+1)/2», что равно x + 6.
- Вместо «i» в выражение «x + i*(i+1)/2» подставили 4 и получили «x + 4*(4+1)/2», что равно x + 10.
- Сложили полученные выражения.
- То есть получили выражение (x + 1) + (x + 3) + (x + 6) + (x + 10). Упростив выражение, получим 4x + 20

(%i3) product (i^2, i, 1, 4);
(%o3) 576

Примечание:

- Вместо «i» в выражение «i^2» подставили 1 и получили «1^2», что равно 1.
- Вместо «i» в выражение «i^2» подставили 2 и получили «2^2», что равно 4.
- Вместо «i» в выражение «i^2» подставили 3 и получили «3^2», что равно 9.
- Вместо «i» в выражение «i^2» подставили 4 и получили «4^2», что равно 16.
- Перемножили полученные выражения.
- То есть получили выражение 1 * 4 * 9 * 16, что равно 576

(%i4) sum (i^2, i, 1, 4);
(%o4) 30

Примечание:

- Вместо «i» в выражение «i^2» подставили 1 и получили «1^2», что равно 1.
- Вместо «i» в выражение «i^2» подставили 2 и получили «2^2», что равно 4.
- Вместо «i» в выражение «i^2» подставили 3 и получили «3^2», что равно 9.
- Вместо «i» в выражение «i^2» подставили 4 и получили «4^2», что равно 16.
- Сложили полученные выражения.
- То есть получили выражение 1 + 4 + 9 + 16, что равно 30

Функции, оперирующие с элементами списков

Функция **map(f,expr1,...,exprn)** позволяет применить функцию (оператор, символ операции) f к частям выражений expr1, expr2,...,exprn. При использовании со списками применяет f к каждому элементу списка. Следует обратить внимание, что f – именно имя функции (без указания переменных, от которых она зависит).

- Постепенно выполните действия. И проверьте результаты. Примечание: нумерация команд может быть другой.

(%i1) map(ratsimp, x/(x^2+x)+(y^2+y)/y);

(%o1) $y+\frac{1}{x+1}+1$

Примечание: функция ratsimp упрощает выражение за счет рациональных преобразований.

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

(%i2) map("=",[a,b],[-0.5,3]);
(%o2) [a = −0.5,b = 3]

(%i3) map(exp,[0,1,2,3,4,5]);
(%o3) [1,e,e^2,e^3,e^4,e^5]

Функция f может быть и заданной пользователем, например:

(%i5) f(x):=x^2;
(%o5) f (x) := x^2

```
(%i6) map(f,[1,2,3,4,5]);
(%o6) [1,4,9,16,25]
```

Функция **apply** применяет заданную функцию ко всему списку (список становится списком аргументов функции; при вызове (F,[x1,...,xn] вычисляется выражение F(arg1,...,argn)). Следует учитывать, что apply не распознаёт ординарные функции и функции от массива.

- Постепенно выполните действия. И проверьте результаты. Примечание: нумерация команд может быть другой.

```
(%i1) L : [1, 5, -10.2, 4, 3];
(%o1) [1,5,−10.2,4,3]

(%i2) apply(max,L);
(%o2) 5

(%i3) apply(min,L);
(%o3) −10.2
```

Чтобы найти максимальный или минимальный элемент набора чисел, надо вызвать функции **max** или **min**. Однако, обе функции в качестве аргумента ожидают несколько чисел, а не список, составленный из чисел. Применять подобные функции к спискам и позволяет функция apply.

- Сохраните файл.

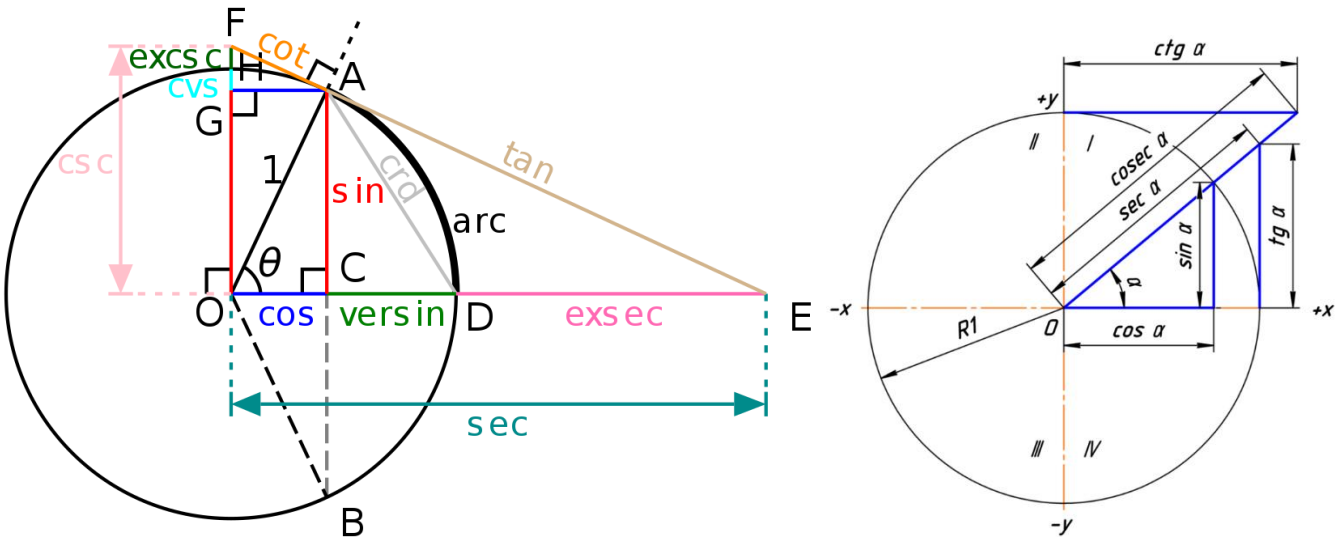
Использованные источники

1. http://dic.academic.ru/dic.nsf/ruwiki/308929
2. Основы работы с системой компьютерной алгебры Maxima: Учебно-методическое пособие / М.С. Малакаев, Л.Р. Секаева, О.Н. Тюленева. Казань: Казанский университет, 2012. – 57с.
3. Компьютерная математика с Maxima: Руководство для школьников и студентов / Е.А.Чичкарёв – М. : ALT Linux, 2012. – 384 с. : ил. – (Библиотека ALT Linux).
4. Губина Т.Н., Андропова Е.В. Решение дифференциальных уравнений в системе компьютерной математики Maxima: учебное пособие. – Елец: ЕГУ им. И.А. Бунина, 2009. – 99 с.

Тригонометрические функции

- Синус угла (sin) – это отношение длины противоположного этому углу катета к гипотенузе.
- Косинус (cos) – это отношение прилежащего этому углу катета к гипотенузе.
- Тангенс: $tg(x) = \frac{\sin(x)}{\cos(x)}$ (отношение длины противоположного углу катета к прилежащему катету). В американской и английской литературе обозначается tan(x).
- Котангенс: $ctg(x) = \frac{\cos(x)}{\sin(x)}$ (отношение длины прилежащего к углу катета к противоположному катету). В американской и английской литературе обозначается cot(x).
- Секанс: $sec(x) = \frac{1}{\cos(x)}$ (отношение длины гипотенузы к прилежащему к углу катету)
- Косеканс: $cosec(x) = \frac{1}{\sin(x)}$ (отношение длины гипотенузы к противоположному катету). В американской и английской литературе обозначается csc(x).
- Версинус:$versin(x) = 1 - \cos(x)$. Также называется синус версус, синус-верзус, «стрелка дуги». Это расстояние от центральной точки дуги, измеряемой удвоенным данным углом, до центральной точки хорды (chord, crd), стягивающей дугу. Иногда используют обозначения vers (x), sir vers (x)
- Коверсинус: $vercos(x) = 1 - \sin(x)$. Также называется косинус-верзус, косинус версус. Иногда используют обозначения cvs (x), cos vers (x)
- Гаверсинус: $haversin(x) = \frac{versin(x)}{2}$. Иногда используют обозначение hav(x).
- Гаверкосинус: $havercos(x) = \frac{vercos(x)}{2}$. Иногда используют обозначение hac(x).
- Экссеканс: $exsec(x) = \sec(x) - 1$. Также называют эксеканс.
- Экскосеканс: $excsc(x) = cosec(x) - 1$. Это дополнительная функция к экссекансу.

Геометрическое представление тригонометрических функций:



Смотри дополнительно на сайте:
https://ru.wikipedia.org/wiki/Редко_используемые_тригонометрические_функции

Источники:

- https://planetcalc.ru/307/
- https://ru.wikipedia.org