

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ
ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«ЕЛЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ им. И. А. БУНИНА»

ЦЕНТР СВОБОДНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Т. Н. Губина, Е. В. Андропова

**РЕШЕНИЕ ДИФФЕРЕНЦИАЛЬНЫХ
УРАВНЕНИЙ В СИСТЕМЕ КОМПЬЮТЕРНОЙ
МАТЕМАТИКИ МАХИМА**

Учебное пособие

Елец — 2009

УДК 519.62+519.63+004.94

ББК 32.973.26-018

Г 93

*Печатается по решению редакционно-издательского совета
Елецкого государственного университета имени И.А. Бунина
от 27. 05. 2009 г., протокол № 2*

Рецензенты:

О.Н. Масина, кандидат физико-математических наук, доцент
(Елецкий государственный университет им. И.А. Бунина);

А. В. Якушин, кандидат педагогических наук, доцент
(Тульский государственный педагогический университет им. Л.Н. Толстого)

Т. Н. Губина, Е. В. Андропова

Г 93 Решение дифференциальных уравнений в системе компьютерной математики *Math*: учебное пособие. – Елец: ЕГУ им. И.А. Бунина, 2009. – 99 с.

Учебное пособие может быть использовано в рамках дисциплин математический анализ, дифференциальные уравнения, пакеты прикладных программ и др. на разных специальностях в учреждениях высшего профессионального образования, если государственным образовательным стандартом предусмотрено изучение раздела «Дифференциальные уравнения», а также в рамках курсов по выбору. Оно также может быть полезным для знакомства с системами компьютерной математики в профильных классах общеобразовательных учреждений с углубленным изучением математики и информатики.

УДК 519.62+519.63+004.94

ББК 22.1+22.18 Р30

© Губина Т.Н., Андропова Е.В., 2009

© ЕГУ им. И.А. Бунина, 2009

Содержание

Предисловие	4
Глава 1. Основы работы в системе компьютерной математики Maxima	
1.1. О системе Maxima.....	7
1.2. Установка Maxima на персональный компьютер.....	7
1.3. Интерфейс основного окна Maxima.....	8
1.4. Работа с ячейками в Maxima.....	10
1.5. Работа со справочной системой Maxima.....	14
1.6. Функции и команды системы Maxima.....	16
1.7. Управление процессом вычислений в Maxima.....	22
1.8. Простейшие преобразования выражений.....	25
1.9. Решение алгебраических уравнений и их систем.....	28
1.10. Графические возможности.....	31
Глава 2. Численные методы решения дифференциальных уравнений	
2.1. Общие сведения о дифференциальных уравнениях.....	45
2.2. Численные методы решения задачи Коши для обыкновенного дифференциального уравнения первого порядка.....	49
2.2.1. Метод Эйлера.....	50
2.2.2. Метод Эйлера-Коши.....	52
2.2.3. Метод Рунге-Кутты 4 порядка точности.....	53
2.3. Решение краевых задач для обыкновенных дифференциальных уравнений методом конечных разностей.....	54
2.4. Метод сеток для решения дифференциальных уравнений в частных производных.....	57
Глава 3. Нахождение решений дифференциальных уравнений в системе Maxima	
3.1. Встроенные функции для нахождения решений дифференциальных уравнений.....	63
3.2. Решение дифференциальных уравнений и их систем в символьном виде.....	66
3.3. Построение траекторий и поля направлений дифференциальных уравнений.....	75
3.4. Реализация численных методов решения задачи Коши для обыкновенных дифференциальных уравнений.....	85
3.4.1. Метод Эйлера.....	85
3.4.2. Метод Эйлера-Коши.....	88
3.4.3. Метод Рунге-Кутты.....	89
3.5. Реализация конечно-разностного метода решения краевой задачи для обыкновенных дифференциальных уравнений.....	91
3.6. Реализация метода сеток для дифференциальных уравнений в частных производных.....	93
Задания для самостоятельного решения	95
Литература	98

Предисловие

Теория дифференциальных уравнений является одним из самых больших разделов современной математики. Одной из основных особенностей дифференциальных уравнений является непосредственная связь теории дифференциальных уравнений с приложениями. Изучая какие-либо физические явления, исследователь, прежде всего, создает его математическую идеализацию или математическую модель, записывает основные законы, управляющие этим явлением, в математической форме. Очень часто эти законы можно выразить в виде дифференциальных уравнений. Такими оказываются модели различных явлений механики сплошной среды, химических реакций, электрических и магнитных явлений и др. Исследуя полученные дифференциальные уравнения вместе с дополнительными условиями, которые, как правило, задаются в виде начальных и граничных условий, математик получает сведения о происходящем явлении, иногда может узнать его прошлое и будущее [1].

Для составления математической модели в виде дифференциальных уравнений нужно, как правило, знать только локальные связи и не нужна информация обо всем физическом явлении в целом. Математическая модель дает возможность изучать явление в целом, предсказать его развитие, делать качественные оценки измерений, происходящих в нем с течением времени. На основе анализа дифференциальных уравнений были открыты электромагнитные волны.

Можно сказать, что необходимость решать дифференциальные уравнения для нужд механики, то есть находить траектории движений, в свою очередь, явилась толчком для создания Ньютоном нового исчисления. Через обыкновенные дифференциальные уравнения шли приложения нового исчисления к задачам геометрии и механики.

Учитывая современное развитие компьютерной техники и интенсивное развитие нового направления — **компьютерной математики** — получили широкое распространение и спрос комплексы программ, называемые системами компьютерной математики.

Компьютерная математика — новое направление в науке и образовании, возникшее на стыке фундаментальной математики, информационных и компьютерных технологий.

Система компьютерной математики (СКМ) — это комплекс программ, который обеспечивает автоматизированную, технологически единую и замкнутую обработку задач математической направленности при задании условия на специально предусмотренном языке.

Современные системы компьютерной математики представляют собой программы с многооконным графическим интерфейсом, развитой системой помощи, что облегчает их освоение и использование.

Основными тенденциями развития СКМ являются рост математических возможностей, особенно в сфере аналитических и символьных вычислений, существенное расширение средств визуализации всех этапов вычислений, широкое применение 2D- и 3D-графики, интеграция различных систем друг с другом

и другими программными средствами, широкий доступ в Internet, организация совместной работы над образовательными и научными проектами в Internet, использование средств анимации и обработки изображений, средств мультимедиа и др.

Существенным обстоятельством, которое до недавнего времени препятствовало широкому использованию СКМ в образовании, является дороговизна профессионального научного математического обеспечения. Однако в последнее время многие фирмы, разрабатывающие и распространяющие такие программы, представляют (через Internet - <http://www.softline.ru>) для свободного использования предыдущие версии своих программ, широко используют систему скидок для учебных заведений, бесплатно распространяют демонстрационные или пробные версии программ [5].

Кроме того, появляются бесплатные аналоги систем компьютерной математики, например, Maxima, Scilab, Octave и др.

В настоящем учебном пособии рассматриваются возможности системы компьютерной математики Maxima для нахождения решений дифференциальных уравнений.

Почему именно Maxima?

Во-первых, система Maxima — это некоммерческий проект с открытым кодом. Maxima относится к классу программных продуктов, которые распространяются на основе лицензии GNU GPL (General Public License).

Во-вторых, Maxima — программа для решения математических задач как в численном, так и в символьном виде. Спектр ее возможностей очень широк: действия по преобразованию выражений, работа с частями выражений, решение задач линейной алгебры, математического анализа, комбинаторики, теории чисел, тензорного анализа, статистических задач, построение графиков функций на плоскости и в пространстве в различных системах координат и т.д.

В-третьих, в настоящее время у системы Maxima есть мощный, эффективный и «дружественный» кроссплатформенный графический интерфейс, который называется WxMaxima (<http://wxmaxima.sourceforge.net>).

Авторами книги уже на протяжении десяти лет изучаются системы компьютерной математики такие как Mathematica, Maple, MathCad. Поэтому, зная возможности этих программных продуктов, в частности для нахождения решений дифференциальных уравнений, хотелось изучить вопрос, связанный с организацией вычислений в символьном виде в системах компьютерной математики, распространяемых свободно.

Настоящее пособие рассказывает о возможностях организации процесса поиска решений дифференциальных уравнений на базе системы Maxima, содержит в себе общие сведения по организации работы в системе.

Пособие состоит из 3 глав. Первая глава знакомит читателей с графическим интерфейсом wxMaxima системы Maxima, особенностями работы в ней, синтаксисом языка системы. Начинается рассмотрение системы с того, где можно найти дистрибутив системы и как его установить. Во второй главе рассматриваются общие вопросы теории дифференциальных уравнений, численные методы их решения. Третья глава посвящена встроенным функциям системы

компьютерной математики Maxima для нахождения решений обыкновенных дифференциальных уравнений 1 и 2 порядка в символьном виде. Также в третьей главе показана реализация в системе Maxima численных методов решения дифференциальных уравнений. В конце пособия приведены задания для самостоятельного решения.

Мы надеемся, что пособием заинтересуется широкий круг пользователей и оно станет их помощником в освоении нового инструмента для решения математических задач.

Т.Н. Губина, Е.В. Андропова
Елец, июль 2009

1.1. О системе Maxima

В рамках проекта создания искусственного интеллекта в 1967 году в Массачусетском технологическом институте была инициирована разработка первой системы компьютерной алгебры Macsyma. Программа в течение многих лет использовалась и развивалась в университетах Северной Америки, где появилось множество вариантов системы. Maxima является одним из таких вариантов, созданным профессором Вильямом Шелтером (William Schelter) в 1982 году. В 1998 году он получил официальное разрешение Министерства энергетики США на выпуск Maxima под лицензией GPL. А начиная с 2001 года Maxima развивается как свободный международный проект, базирующийся на [SourceForge](http://sourceforge.net) [2].

В настоящее время Maxima — это система компьютерной математики, которая предназначена для выполнения математических расчетов (как в символьном, так и в численном виде) таких как:

- упрощение выражений;
- графическая визуализация вычислений;
- решение уравнений и их систем;
- решение обыкновенных дифференциальных уравнений и их систем;
- решение задач линейной алгебры;
- решение задач дифференциального и интегрального исчисления;
- решение задач теории чисел и комбинаторных уравнений и др.

В системе имеется большое количество встроенных команд и функций, а также возможность создавать новые функции пользователя. Система имеет свой собственный язык. Она также имеет встроенный язык программирования высокого уровня, что говорит о возможности решения новых задач и возможности создания отдельных модулей и подключения их к системе для решения определенного круга задач.

1.2. Установка Maxima на персональный компьютер

Свободно распространяемую версию дистрибутива Maxima, документацию на английском языке, типы и виды интерфейсов системы можно посмотреть и скачать с сайта программы <http://maxima.sourceforge.net>. На период написания пособия последняя версия дистрибутива — Maxima 5.18.1.

Сама по себе Maxima — консольная программа и все математические формулы «отрисовывает» обычными текстовыми символами.

Система является многоплатформенной, имеет небольшой размер дистрибутива (≈21,5 Мб), легко устанавливается, имеет несколько графических русифицированных интерфейсов: xMaxima, wxMaxima, TexMacS.

Наиболее дружелюбным, простым и удобным в работе графическим интерфейсом в настоящее время является интерфейс wxMaxima. Поэтому в дальнейшем будем использовать именно этот интерфейс.

Установка Maxima под управлением Windows

Полученный после скачивания файл, например maxima-5.18.1.exe (размер файла около 21,5 мегабайт), является исполняемым. Для начала установки программы достаточно нажать на него два раза левой кнопкой мыши. Сразу появится окно выбора локализации (выбираем русский язык).

В следующем окне выбираем «Далее», внимательно читаем лицензионное соглашение, выбираем «я принимаю условия соглашения» и снова выбираем «Далее» (два раза).

В появившемся окне выбираем путь установки программы (можно оставить его без изменения).

Переходим к выбору устанавливаемых компонент. Из всего перечисленного для нас «лишними» являются Пакеты поддержки языков Maxima.

При установке желательно установить и графический интерфейс xMaxima, поскольку на нем базируется интерфейс wxMaxima и при решении некоторых задач он необходим, например, при выполнении графических построений.

В следующих окнах предлагается выбрать место размещения ярлыка для запуска программы (в меню «Пуск», на рабочий стол и т.д.). Завершающим этапом будет окно с предложением начать установку. По окончании установки выбираем «Далее» и «Завершить».

Таким образом, установка программы закончена.

Установка Maxima под управлением Linux

Maxima входит в состав многих дистрибутивов Linux, например, таких как AltLinux, Mandriva, Ubuntu, Fedora и др. В некоторых случаях может понадобиться доустановка с репозитория дистрибутива с помощью систем yum или synaptic.

Для установки в других дистрибутивах Linux необходимо использовать подходящий пакет системы Maxima, который можно скачать с сайта <http://maxima.sourceforge.net>.

Теперь можно приступить к работе с системой.

Учебное пособие ориентировано на работу с системой Maxima, установленную под управлением Linux. Заметим, что все рассматриваемые команды активны и в системе, установленной под управлением Windows.

Для начала познакомимся с интерфейсом основного окна программы.

1.3. Интерфейс основного окна Maxima

После запуска системы Maxima 5.18.1 с графическим интерфейсом wxMaxima появляется рабочее окно программы (Рис. 1).

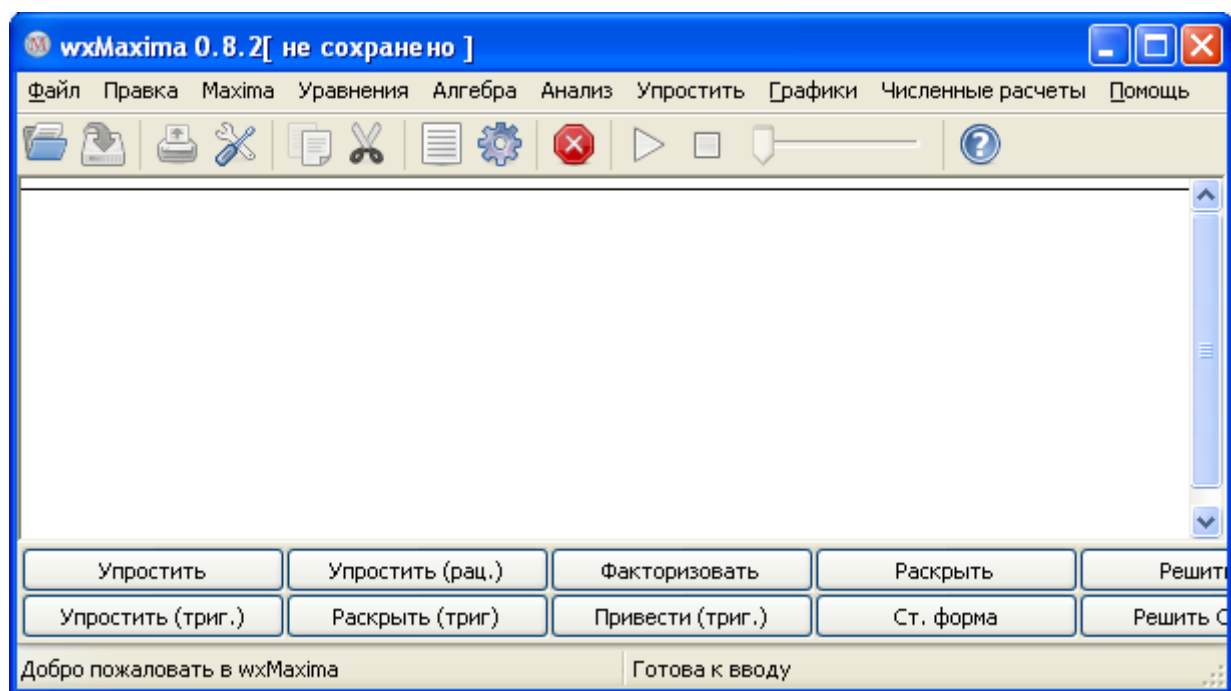


Рис. 1. Вид рабочего окна системы Maxima

Структура окна, как видно из рисунка, имеет стандартный вид:


- строка заголовка, в которой располагается название программы и информация о том, сохранен ли рабочий документ (если документ сохранен, то прописывается его имя);
- панель меню программы – доступ к основным функциям и настройкам программы. В ней находятся функции для решения большого количества типовых математических задач, разделенные по группам: уравнения, алгебра, анализ, упростить, графики, численные вычисления. Заметим, что ввод команд через диалоговые окна упрощает работу с программой для начинающих пользователей;
- панель инструментов — на ней находятся кнопки для создания нового документа, быстрого сохранения документа, вызова окна справки, создания ячеек ввода, прерывания вычислений, кнопки для работы с буфером обмена и др.;
- рабочая область — непосредственно сам документ, в котором формируются ячейки ввода и выводятся результаты выполненных команд;
- полосы прокрутки;
- панель с кнопками — набор кнопок для быстрого вызова некоторых команд: упростить, решить уравнение или систему, построить график и др.;
- строка состояния.

В системе **Maхiтa команда** — это любая комбинация математических выражений и встроенных функций. Каждая команда завершается символом «;», причем в случае его отсутствия система сама добавит этот символ.

1.4. Работа с ячейками в Maхiтa

После того, как система загрузилась, можно приступить к вычислениям. Для этого следует добавить так называемую ячейку ввода, в которую вводится команда системе выполнить какое-либо действие.

Систему можно использовать в качестве мощного калькулятора для нахождения значений числовых выражений. Например, для того, чтобы найти значение произведения 120 и 1243, надо:

- на панели инструментов нажать кнопку *Insert input cell* (или  нажать на клавиатуре клавишу Enter). В результате в рабочей области будет сформирована ячейка ввода (Рис.2).

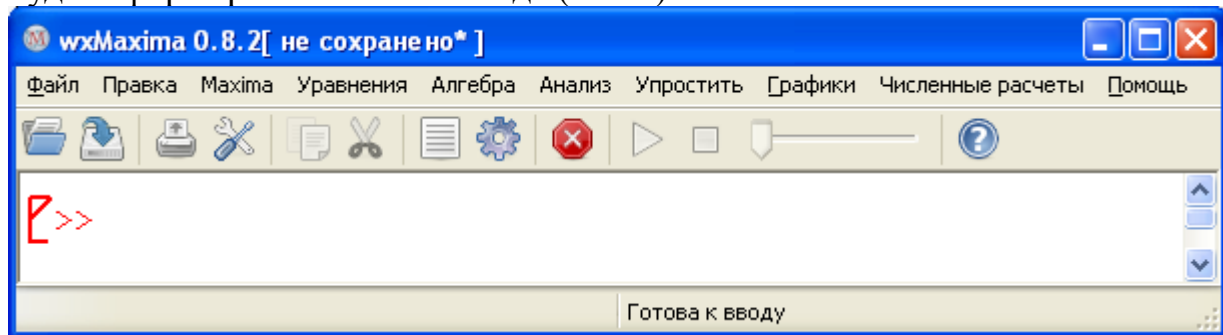


Рис.2. Формирование новой ячейки ввода

- далее с клавиатуры вводим команду: $120*1243$ и нажимаем комбинацию клавиш Ctrl+Enter (Рис.3).

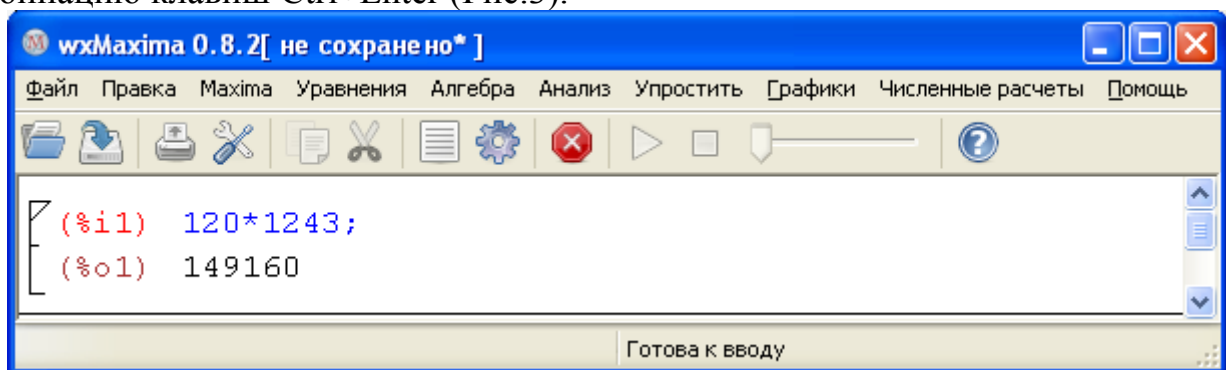


Рис.3. Выполнение вычислений в системе Maхiтa

Таким образом, в документе были сформированы две строки: (%i1) — ячейка ввода и для нее (%o1) — ячейка вывода. Каждая ячейка имеет свою метку — заключенное в скобки имя ячейки. Ячейки, в которых размещаются входные данные (формулы, команды, выражения) называют **ячейками ввода**. Они обозначаются %iChislo, где Chislo — номер ячейки ввода (i — сокращенно от английского слова *input* — ввод). Ячейки, в которых размеща-

ются выходные данные (списки значений, выражения) называют **ячейками вывода**. Они обозначаются `%oChislo`, где `Chislo` — номер ячейки вывода (о — сокращенно от английского слова *output* — вывод).

Почему же имена ячеек начинаются с символа `%`? Разработчики системы Maxima посчитали удобным начинать имена всех встроенных служебных имен: констант, переменных, зарезервированных слов, с этого символа. Сделано это для того, чтобы избежать возможных накладок с пользовательскими именами.

В системе Maxima предусмотрена возможность ввода сразу нескольких команд в одной строке. Для этого одна команда от другой отделяется символом «;». При этом формируется одна строка ввода и столько строк вывода, сколько команд было задано.

```
(%i2) 4+7; 25/5; 3^6;  
(%o2) 11  
(%o3) 5  
(%o4) 729
```

Для обозначения конца ввода команды можно вместо точки с запятой использовать знак `$`. Это бывает удобно в том случае, если вывод результата вычисления на экран не нужен; тогда его можно «заглушить». Заглушенный результат при этом все равно будет вычисляться. Например,

```
(%i11) a:6$ b:7$ a+b;  
(%o13) 13
```

Как видим, были записаны три команды в одной ячейке ввода и сформирована одна ячейка вывода. Здесь же была использована команда присваивания значений переменным `a` и `b`. Она задается в виде:

имя_переменной: значение

Имена функций и переменных в системе Maxima чувствительны к регистру, то есть **прописные и строчные буквы** в них различаются.

Задание команды в ячейке ввода и формирование ячейки вывода при нажатии комбинации клавиш `Ctrl+Enter`, называют **отдельной сессией работы с системой Maxima**.

Рассмотрим **основные приемы работы** с отдельными сессиями работы в Maxima.

1. Задание команды для выполнения математических расчетов.

Как уже понятно, для задания команды системе нужно в строке ввода задать само выражение и оценить его, закончив ввод нажатием комбинации клавиш `Ctrl+Enter`. В результате образуется ячейка ввода и соответствующая ей ячейка вывода.

2. Сворачивание и разворачивание отдельных сессий.

Если мы наведем курсор мыши на левый верхний угол квадратной скобки и нажмем левую кнопку мыши, то произойдет сворачивание части документа, относящейся к этой команде (Рис.4).

```
[ (%i1) 15+78-34;
```

Рис.4. Вид свернутой ячейки

Повторный щелчок мыши вернет ячейку в начальное состояние.

3. Добавление ячеек в документ.

При работе с системой можно столкнуться с трудностью добавления команд в заданное место документа. Остановимся на этом подробнее. Например, у нас имеются две ячейки ввода (Рис. 5).

```
[ (%i2) a+b;
[ (%o2) b + a

[ (%i3) c+d;
[ (%o3) d + c
```

Рис.5. Ячейки в документе, между которыми требуется вставить дополнительно ячейку

Для добавления новой ячейки между ячейками (%i2) и (%i3) щелчком левой кнопкой мыши между ячейками так, чтобы появилась горизонтальная черта. После чего нажимаем на клавиатуре клавишу Enter. Ячейка ввода добавлена (Рис.6).

```
[ (%i2) a+b;
[ (%o2) b + a

[ >> 3+8

[ (%i3) c+d;
[ (%o3) d + c
```

Рис.6. Добавление новой ячейки ввода

4. Переоценить значение отдельно взятой ячейки.

Если нужно переоценить значение введенного выражения (или выражения после внесенных в него исправлений), то для этого достаточно установить курсор в ячейке ввода и нажать комбинацию клавиш Ctrl+Enter.

5. Переоценить ячейки всего документа в целом.

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 *Правка*.

6. Удалить ячейки ввода из документа.

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

В системе Maxima можно добавлять в документ текстовые комментарии (Рис.7). Для этого выбираем пункт меню *Правка*→*Cell*→*New Text Cell* (или клавиша F6), после чего с клавиатуры набираем текст.

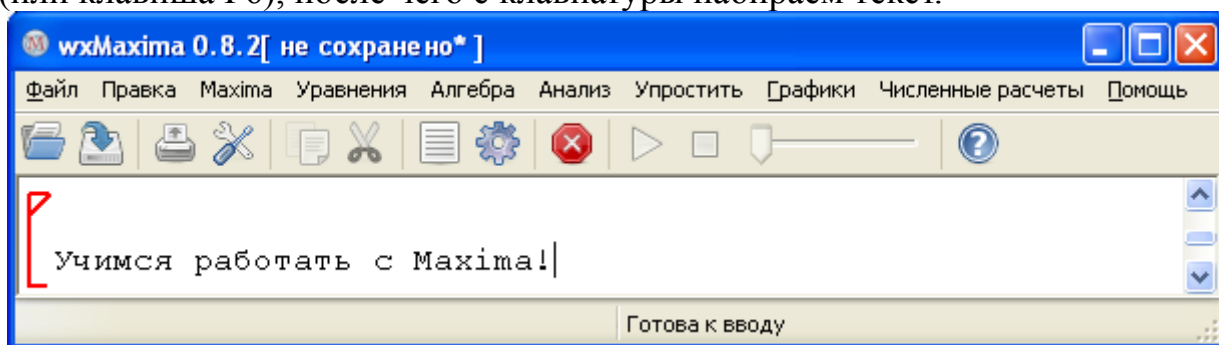


Рис.7. Вид текстовой ячейки в системе Maxima

Кроме того, в документе для оформления текста можно применять различные стили (Рис.8). Для этого можно воспользоваться пунктом меню *Правка*→*Cell*→*New Section Cell* (или Ctrl+F6), или *Правка*→*Cell*→*New Title Cell* (или Ctrl+Shift+F6).

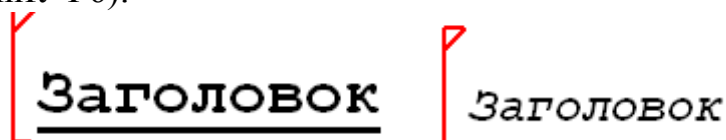


Рис.8. Стили оформления текстовых ячеек

Сохранение документа выполняется обычным способом с использованием пункта меню *Файл*.

С помощью кнопки *Настроить wxMaxima* на панели инструментов (или пункт меню *Правка*→*Настройка*) можно изменять конфигурацию графического интерфейса системы, применяемую по умолчанию. После нажатия на кнопке появляется окно с настройками, которое имеет две вкладки: *Параметры* и *Стиль*. На вкладке *Параметры* можно выбрать язык, вид панели инструментов и ряд дополнительных опций: проверять баланс скобок в тексте, копировать в буфер при выделении и др. На вкладке *Стиль* можно выбрать применяемый по умолчанию шрифт и его размер, а также стиль оформления (цвет и начертание) различных данных, используемых в документе: переменных, констант, чисел, имен функций и др.

В системе Maxima есть возможность представлять выражения в формате LaTeX. Для этого можно воспользоваться возможностью копирования выражения в формате LaTeX в буфер обмена (пункт меню *Правка*→*Сору LaTeX*), либо возможностью вывода выражения в формате LaTeX в документ (пункт меню *Maxima*→*Display TeX form*). Например, зададим выражение и получим его представление на языке пакета LaTeX:

```
(%i1) x^2+5*x-18*y;
(%o1) -18 y+x2+5 x
(%i2) tex(%)$
$$-18\,y+x^2+5\,x$$
```

1.5. Работа со справочной системой Maxima

В системе Maxima встроена справочная система на английском языке. Она включает в себя документацию по организации работы в системе, а также информацию по встроенным командам системы с большим количеством примеров их использования для решения математических задач.

Работать со справочной системой можно несколькими способами.

Первый — вызов примеров использования команды по имени. Для этого выбираем пункт меню *Помощь*→*Examples*. В диалоговое окно вводим имя команды, например, `desolve` (Рис.9).

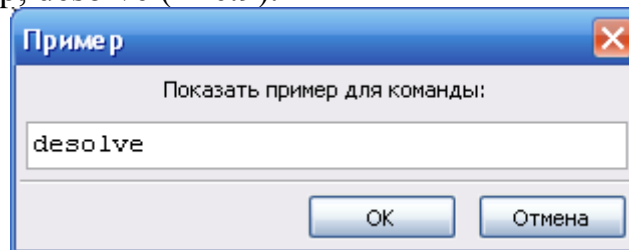


Рис.9. Вызов команды из справки по имени

После нажатия на кнопке *Ok* в документе формируется ячейка ввода и ниже пример использования введенной команды для решения конкретной задачи. Заметим, что команда `desolve` используется для нахождения решения задачи Коши для обыкновенных дифференциальных уравнений и их систем. Подробнее эта команда рассматривается во второй главе пособия.

```

(%i1) example(desolve);
(%i2) eqn1:'diff(f(x),x) = sin(x)+'diff(g(x),x)
(%o2)  $\frac{d}{dx}f(x) = \frac{d}{dx}g(x) + \sin(x)$ 
(%i3) eqn2:'diff(g(x),x,2) = 'diff(f(x),x) - cos(x)
(%o3)  $\frac{d^2}{dx^2}g(x) = \frac{d}{dx}f(x) - \cos(x)$ 
(%i4) atvalue('diff(g(x),x),x = 0,a)
(%o4) a
(%i5) atvalue(f(x),x = 0,1)
(%o5) 1
(%i6) desolve([eqn1,eqn2],[f(x),g(x)])
(%o6) [ f(x)=a %ex - a + 1 , g(x)=cos(x)+a %ex - a + g(0)- 1 ]
(%i7) ev([eqn1,eqn2],%,diff)
(%o7) [ a %ex = a %ex , a %ex - cos(x) = a %ex - cos(x) ]
(%o7) done

```

Второй способ — вызов документации по Maxima (Рис.10). В этом случае требуется выбрать пункт меню *Правка*→*Maxima help* (или клавиша F1). Откроется окно справочной системы.

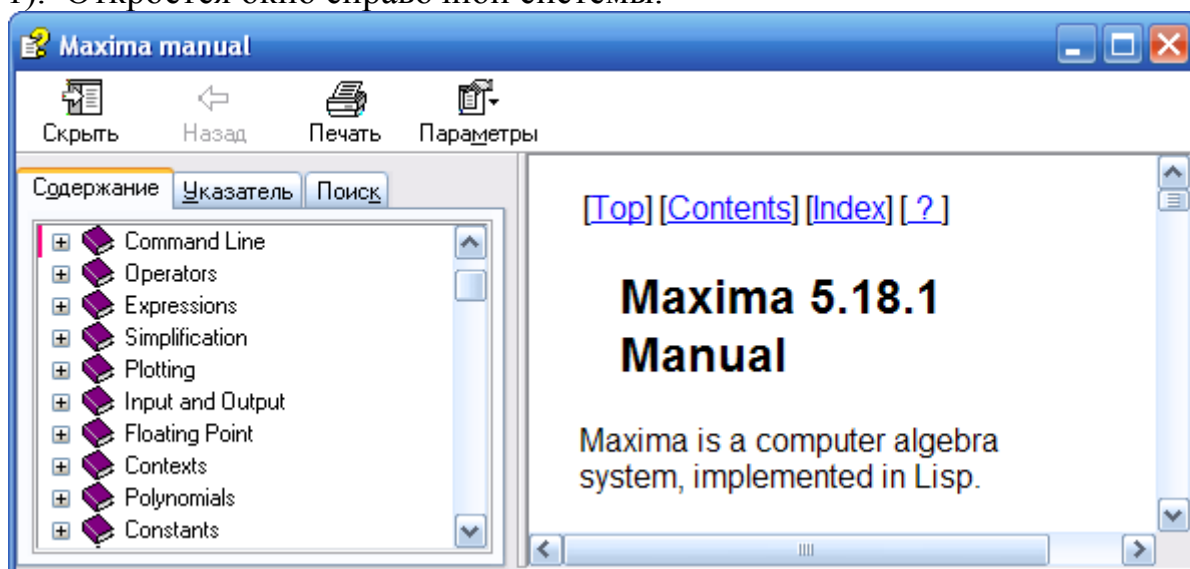


Рис.10. Окно документации по системе Maxima

Осуществлять поиск интересующих команд можно на закладке *Содержание* по разделам, например, работа с выражениями, многочленами, построение графиков функций и др., можно осуществлять поиск по имени команды на закладке *Указатель* или *Поиск*. Например, найдем информацию о функции \sin . Для этого перейдем на закладку *Указатель*, в текстовое поле введем \sin и сразу же у нас появится список, в котором содержится эта функция (Рис.11). Остается только открыть в правом окне справку по функции \sin .

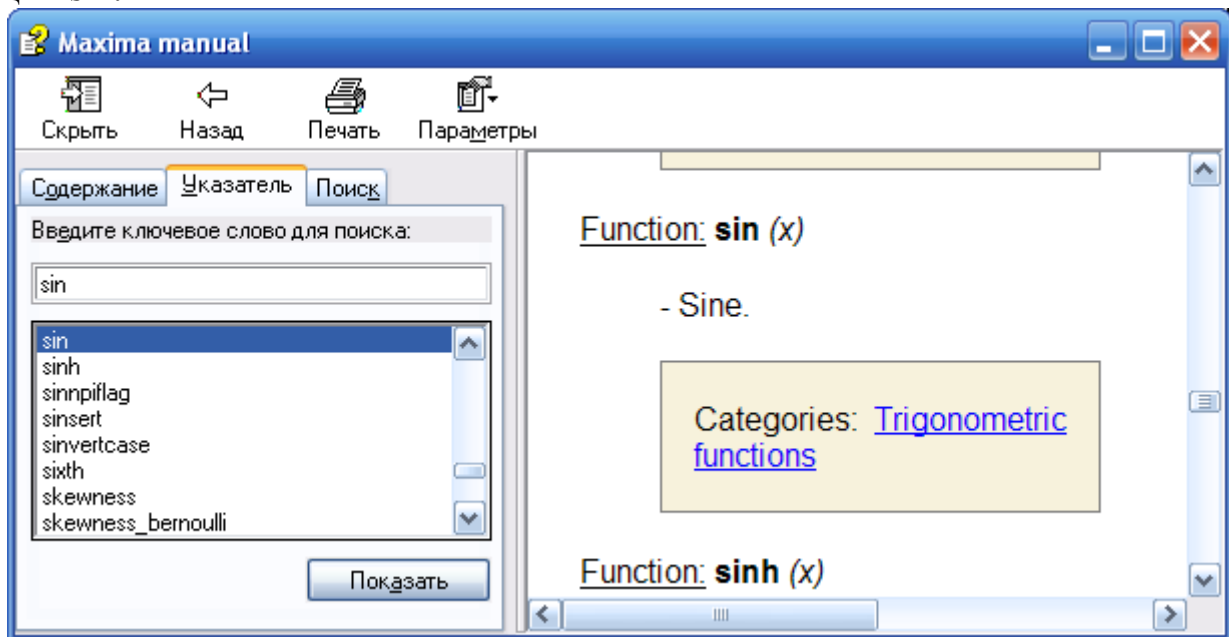


Рис.11. Работа с указателем в окне справочной системы

1.6. Функции и команды системы Maxima

В системе Maxima имеется множество встроенных функций. Как было показано выше, для каждой встроенной функции можно получить описание в документации, содержащейся в справочной системе. Вызвать справку можно с помощью функциональной клавиши F1. Также в Maxima есть специальная функция, которая выдает информацию из документации по конкретным словам. Сокращенная версия вызова этой функции: `?? name` (Рис.12). Здесь `??` — это имя оператора, и аргумент нужно отделять от него пробелом. Оператор `??` выдает список тех разделов помощи и имен функций, которые содержат заданный текст, после чего предлагают ввести номер того раздела или описания той функции, которые требуется посмотреть:


```
(%i1) ??sin;
0: Introduction to string processing
1: absint (Functions and Variables for Fourier series)
2: asin (Functions and Variables for Trigonometric)
3: asinh (Functions and Variables for Trigonometric)
4: fourintsin (Functions and Variables for Fourier series)
5: foursin (Functions and Variables for Fourier series)
6: maxpsinegint (Gamma and factorial Functions)
7: maxpsiposint (Gamma and factorial Functions)
8: poisint (Poisson series)
9: sin (Functions and Variables for Trigonometric)
10: sinh (Functions and Variables for Trigonometric)
11: sinnpiflag (Functions and Variables for Fourier series)
12: sinert (Functions and Variables for strings)
13: sinvertcase (Functions and Variables for strings)
Enter space-separated numbers, 'all' or 'none': B
```

Рис.12. Вызов справки по интересующей команде системы Maxima

Заметим, что в системе Maxima нет четкого разграничения между операторами и функциями. Более того, **каждый оператор — это на самом деле функция**.

Все функции и операторы Maxima работают не только с действительными, но и комплексными числами. Сами комплексные числа записываются 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.

Рассмотрим **синтаксис базовых функций** системы Maxima.

1. Арифметические операторы: +, -, *, /, ^.

Пример:

```
(%i2) x*y/5-x^2+6;
```

```
(%o2)  $\frac{xy}{5} - x^2 + 6$ 
```

2. Операторы сравнения: <, <=, >=, >.

Пример:

```
(%i1) y(x):=if x<0 then x^2 else x^(1/2);
```

```
(%o1) y(x):=if x<0 then x^2 else x1/2
```

```
(%i2) y(5);y(-4);
```

```
(%o2)  $\sqrt{5}$ 
```

```
(%o3) 16
```

3. Логические операторы: and, or, not.

Пример:

```
(%i1) y(x):=if (x<0 or x>=6) then x^2 else x^(1/2);
```

```
(%o1) y(x):=if x<0 or x>=6 then x^2 else x^(1/2)
```

```
(%i2) y(2);y(10);
```

```
(%o2)  $\sqrt{2}$ 
```

```
(%o3) 100
```

4. Функция нахождения факториала числа: !

Факториал задан в наиболее общем виде и представляет собой, по сути, гамма-функцию (точнее, $x! = \text{gamma}(x+1)$), то есть определен на множестве всех комплексных чисел, кроме отрицательных целых. Факториал от натурального числа (и нуля) автоматически упрощается до натурального же числа.

Пример:

```
(%i1) 15!;
```

```
(%o1) 1307674368000
```

5. Функция нахождения полуфакториала числа: !! (произведение всех четных (для четного операнда) или нечетных чисел, меньших либо равных данному).

Пример:

```
(%i1) 45!!;
```

```
(%o1) 25373791335626257947657609375
```

6. Функция отрицания синтаксического равенства: #

Запись $a\#b$ эквивалентна $\text{not } a=b$.

Пример:

```
(%i1) a#b; not a=b;
```

```
(%o1) a # b
```

```
(%o2) true
```

7. Функция нахождения модуля числа x: abs(x)

Модуль определен для всех комплексных чисел.

Пример:

```
(%i1) abs(-5);abs(5+7*%i);
```

```
(%o1) 5
```

```
(%o2)  $\sqrt{74}$ 
```

8. Функция, возвращающая знак числа x: signum(x)

Пример:

```
(%i1) signum(-5);signum(7+x^2);
(%o1) - 1
(%o2) 1
```

9. Функции, возвращающие наибольшее и наименьшее значения из заданных действительных чисел: $\max(x_1, \dots, x_n)$ и $\min(x_1, \dots, x_n)$.

Пример:

```
(%i1) 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);
(%o1) 10
      9
(%o2) 1
      7
```

10. Некоторые встроенные математические функции:

$\text{sqrt}(x)$	Квадратный корень из x
$\text{acos}(x)$	Арккосинус аргумента x
$\text{acosh}(x)$	Гиперболический арккосинус аргумента x
$\text{acot}(x)$	Арккотангенс аргумента x
$\text{acoth}(x)$	Гиперболический арккотангенс аргумента x
$\text{acsc}(x)$	Арккосеканс аргумента x
$\text{acsch}(x)$	Гиперболический арккосеканс аргумента x
$\text{asec}(x)$	Арксеканс аргумента x
$\text{asech}(x)$	Гиперболический арксеканс аргумента x
$\text{asin}(x)$	Арсинус аргумента x
$\text{asinh}(x)$	Гиперболический арксинус аргумента x
$\text{atan}(x)$	Арктангенс аргумента x
$\text{atanh}(x)$	Гиперболический арктангенс аргумента x
$\text{cosh}(x)$	Гиперболический косинус аргумента x
$\text{coth}(x)$	Гиперболический котангенс аргумента x
$\text{csc}(x)$	Косеканс аргумента x
$\text{csch}(x)$	Гиперболический косеканс аргумента x

$\sec(x)$	Секанс аргумента x
$\operatorname{sech}(x)$	Гиперболический секанс аргумента x
$\sin(x)$	Синус аргумента x
$\sinh(x)$	Гиперболический синус аргумента x
$\tan(x)$	Тангенс аргумента x
$\tanh(x)$	Гиперболический тангенс аргумента x
$\log(x)$	Натуральный логарифм x
$\exp(x)$	Экспонента x

В языке системы Maxima заложены основные исполнимые операторы, которые есть в любом языке программирования. Рассмотрим их.

1. Операторы присваивания значений (именования выражений).

1. Оператор «:=» (оператор задания значения переменной).
2. Оператор «:=» (оператор задания функции пользователя).
3. Расширенные варианты операторов присваивания и задания функции, обозначаемые соответственно через :: и ::=.

Использование оператора задания функции пользователя значительно облегчает работу с ней, поскольку к ней можно обращаться по имени и легко и удобно вычислять значения функции в заданных точках.

Пример: найдем значение функции $f(x, y) = \cos x + \sin y$ в точке $(\pi, 2\pi)$.

```
(%i1) f(x, y) := cos(x) + sin(y);
(%o1) f(x, y) := cos(x) + sin(y)
(%i2) f(%pi, 2*%pi);
(%o2) -1
```

2. Условный оператор

1. Неполная форма условного оператора

```
(%i1) y:if a<0 then 6;
(%o1) if a<0 then 6
```

2. Полная форма условного оператора

```
(%i1) y:if a<0 then 6 else 9;
(%o1) if a<0 then 6 else 9
```

3. Вложение условных операторов друг в друга

```
(%i1) y:if a<0 then 5 else if a=0 then 8 else 9;
(%o1) if a<0 then 5 else if a=0 then 8 else 9
```

Оператор цикла

Оператор цикла может задаваться несколькими способами. Способ задания зависит от того, известно ли заранее сколько раз необходимо выполнить тело цикла.

Пример: задание цикла для вывода значений переменной a в диапазоне от -3 до 10 с шагом 5:

```
(%i1) for a:-3 thru 10 step 5 do display(a);
a = - 3
a = 2
a = 7
(%o1) done
```

Пример: цикл для нахождения суммы всех натуральных чисел до числа 50 включительно:

```
(%i1) s:0$ for i:1 while i<=50 do s:s+i;
(%o2) done
(%i3) s;
(%o3) 1275
```

Следующей важной возможностью системы Maxima является **работа со списками и массивами**.

Для формирования списков используется команда `makelist`. Например, с помощью команды

```
(%i4) x:makelist((k+1)/2, k, 1, 10);
(%o4) [1, 3/2, 2, 5/2, 3, 7/2, 4, 9/2, 5, 11/2]
```

мы сформировали список с именем x , состоящий из десяти элементов, значения которых находятся по формуле $\frac{k+1}{2}$.

Для формирования массивов используется команда `array`. Например с помощью команды,

```
(%i1) array(a, 10, 5);
(%o1) a
```

мы сформировали двумерный массив A , состоящий из 10 строк и 5 столбцов. Для заполнения массива элементами воспользуемся циклом с параметром. Например,

```
(%i2) for i:1 thru 10 do for j:1 thru 5 do
      (a[i,j]:i+j);
(%o2) done
```

Для вывода элементов массива на экран можно воспользоваться командой:

```
--> for i:1 thru 10 do for j:1 thru 5 do
      display(a[i, j]);
```

Массив можно формировать и без предварительного объявления. В следующем примере мы сформировали одномерный массив x , состоящий из 5 элементов, значения которых вычисляются по формуле $x(i)=\sin i$.

```
(%i2) for i:1 thru 5 do
      (arraymake(x, [i]), x[i]:sin(i), display (x[i]));
x1=sin(1)
x2=sin(2)
x3=sin(3)
x4=sin(4)
x5=sin(5)
```

Неудобство работы с массивами заключается в том, что вывод значений элементов массива осуществляется в столбец. Гораздо удобнее, если значения массива (двумерного) выводятся в виде матрицы. Для этих целей можно воспользоваться командой `genmatrix`. Например, для формирования двумерного массива (матрицы) следует задать команду в следующем виде:

```
(%i3) a:genmatrix(lambda([i, j], i^2+j^2), 6, 6) $
```

Выведем полученный массив:

```
(%i4) a;
(%o4) 
$$\begin{bmatrix} 2 & 5 & 10 & 17 & 26 & 37 \\ 5 & 8 & 13 & 20 & 29 & 40 \\ 10 & 13 & 18 & 25 & 34 & 45 \\ 17 & 20 & 25 & 32 & 41 & 52 \\ 26 & 29 & 34 & 41 & 50 & 61 \\ 37 & 40 & 45 & 52 & 61 & 72 \end{bmatrix}$$

```

1.7. Управление процессом вычислений в Maxima

Система компьютерной математики Maxima относится к системам символьной математики. Поэтому (по умолчанию) система выдает результат в символьном виде. То есть, если не задавать специальную команду, система

никогда не представит полученные в ходе вычислений результаты в виде приближенного вещественного числа. Например, если мы введем в ячейку ввода команду $\sqrt{2}$, то получим:

```
(%i1) sqrt(2);  
(%o1)  $\sqrt{2}$ 
```

Если же возникает необходимость представить полученный в ходе расчетов результат в виде вещественного числа, то в этом случае требуется дать специальную команду системе. Например, можно поступить так: если требуется получить приближенное значение $\sqrt{2}$, то выбираем пункт меню *Численные расчеты* → *To float* (в число с одинарной точностью) (или *To BigFloat* (в число с двойной точностью)). Результат будет выглядеть так:

```
(%i2) float(%), numer;  
(%o2) 1.414213562373095
```

Знак «%» в Maxima используется для обращения к результату, полученному в последней сессии работы. Это бывает удобно, если нет необходимости вводить переменные пользователя и в дальнейшем использовать полученные значения.

Для управления процессом вычислений предусмотрена возможность так называемой «**блокировки вычислений**». Выполняется блокировка с помощью одинарного знака апострофа. Ее суть:

- если перед именем функции или переменной поставить знак апострофа, то блокируется вычисление самой функции (но не ее аргументов) или переменной;
- если поставить апостроф перед выражением, заключенным в скобки, то невычисленными останется все это выражение целиком, т. е. и все входящие в него функции, и все аргументы этих функций.

Например, зададим функцию $f(x)$ и сравним результаты, полученные при попытке вычисления значения функции в точке $x=0$.

```
(%i1) f(x):=tan(x);  
(%o1) f(x):=tan(x)  
(%i2) 'f(0); f(0);  
(%o2) f(0)  
(%o3) 0
```

Как видим, знак апострофа заблокировал попытку вычисления значения функции в первом случае.

Другой пример:

```
(%i4) integrate(cos(x), x);
(%o4) sin(x)
(%i5) 'integrate(cos(x), x);
(%o5)  $\int \cos(x) dx$ 
```

В противовес блокировке вычислений с помощью двух знаков апострофа наоборот можно заставить систему выполнять вычисления — «**принудительное вычисление**». Например,

```
(%i1) a(n):=integrate(cos(%pi*x/2)*cos(x), x, -2,2)
/(2*%pi*x);
```

$$\int_{-2}^2 \cos\left(\frac{\pi x}{2}\right) \cos(x) dx$$

К

```
(%o1) a(n):=
$$\frac{\int_{-2}^2 \cos\left(\frac{\pi x}{2}\right) \cos(x) dx}{2 \pi x}$$

```

ак видно, система отказалась вычислять интеграл, хотя мы не давали команду заблокировать вычисления. Если же мы поставим двойной апостроф перед командой, то получим следующий результат:

```
(%i2) a(n):=' '(integrate(cos(%pi*x/2)*cos(x), x, -2,2)
/(2*%pi*x));
```

```
(%o2) a(n):=
$$\frac{4 \sin(2)}{\pi(\pi^2 - 4)x}$$

```

Обратим внимание на то, что в системе Maxima по умолчанию все углы измеряются в радианах. Поэтому если требуется работать с углами в градусах, для этого потребуется вспомнить формулу перевода из радиан в градусы.

В терминологии Maxima невычисленная форма выражения называется «*noun form*», вычисленная — «*verb form*».

Следующим важным моментом при работе в системах компьютерной математики является умение выполнять подстановку значений переменных или частей выражений в функции, выражения. Рассмотрим некоторые возможности системы, предусмотренные для этих целей.

Например, требуется в выражение $\cos x + 4 \sin x - \sqrt{x}$ вместо переменной x подставить конкретное значение, например, π .

1 способ.

```
(%i1) cos(x)+4*sin(x)-sqrt(x),x=%pi;
(%o1)  $-\sqrt{\pi} - 1$ 
```


2 способ.

```
(%i2) subst(%pi,x, cos(x)+4*sin(x)-sqrt(x));
(%o2)  $-\sqrt{\pi} - 1$ 
```

Таким образом, команда subst позволяет выполнять подстановку в выражение значений каких-либо переменных. На самом деле, команд подстановки значений в выражение или функцию в Maxima несколько.

1.8. Простейшие преобразования выражений

По умолчанию в системе Maxima является активной функция автоупрощения, т.е. система старается упростить вводимое выражение сама без какой-либо команды.

Пример. Пусть требуется найти значение следующего числового выражения: $\frac{\frac{1}{2} + \frac{1}{4} - \frac{4}{5}}{\frac{1}{2} + \frac{4}{5}} * 7$.

Зададим выражение по правилам языка системы Maxima.

```
(%i14) (1/2+1/4-4/5)/(1/2+4/5)*7;
(%o14)  $-\frac{7}{26}$ 
```

Как видим, система в ответ вывела значение выражения, хотя мы не задали никакой команды.

Как же заставить систему вывести не результат, а само выражение? Для этого функцию упрощения надо отключить с помощью команды simp:false\$. Тогда получим:

```
(%i15) simp:false$
(%i16) (1/2+1/4-4/5)/(1/2+4/5)*7;
(%o16)  $\frac{\frac{1}{2} + \frac{1}{4} - \frac{4}{5}}{\frac{1}{2} + \frac{4}{5}} 7$ 
```

Для того чтобы активировать функцию упрощения, надо задать команду simp:true\$. Функция автоупрощения может работать как с числовыми, так и с некоторыми не числовыми выражениями. Например,

```
(%i1) (x^3)^4*x^7/x^16;
(%o1)  $x^3$ 
(%i2) -2*x^2*c^5*d^8*(-1/2*c^6*d*x);
(%o2)  $c^{11} d^9 x^3$ 
```

При вводе мы можем обращаться к любой из предыдущих ячеек по ее имени, подставляя его в любые выражения. Кроме того, последняя ячейка вывода обозначается через %, а последняя ячейка ввода — через _. Это позволяет обращаться к последнему результату, не отвлекаясь на то, каков его номер. Но такими обращениями к ячейкам злоупотреблять не надо, поскольку при переоценивании всего документа или его отдельных ячеек ввода может произойти разногласие между номерами ячеек.

Пример. Найти значение выражения $\frac{72}{4} + 5 + \frac{2}{8} - \frac{5}{6} \cdot \frac{9}{5}$ и увеличить полученный результат в 5 раз.

```
(%i19) 72/4+5+2/8-(5/6)*(9/5);
(%o19) 87
         4

(%i20) %*5;
(%o20) 435
         4
```

Желательно вместо имен ячеек использовать переменные и присваивать их имена любым выражениям. В этом случае в виде значения переменной может выступать любое математическое выражение.

Значения имен переменных сохраняются на протяжении всей работы с документом. Напомним, что если необходимо снять определение с переменной, то это можно сделать с помощью функции `kill(name)`, где `name` — имя уничтожаемого выражения; причем это может быть как имя, назначенное вами, так и любая ячейка ввода или вывода. Точно так же можно очистить всю память и освободить все имена, введя команду `kill(all)` (или выбрать меню *Maxima*->*Очистить память* (Clear Memory)). В этом случае очистятся в том числе и все ячейки ввода-вывода, и их нумерация опять начнется с единицы.

Функция автоупрощения далеко не всегда способна упростить выражение. В дополнение к ней имеется целый ряд команд, которые предназначены для работы с выражениями: рациональными и иррациональными. Рассмотрим некоторые из них.

rat(выражение) — преобразовывает рациональное выражение к канонической форме: раскрывает все скобки, затем приводит все к общему знаменателю, суммирует и сокращает; приводит все числа в конечной десятичной записи к рациональным. Каноническая форма автоматически «отменяется» в случае любых преобразований, не являющихся рациональными

ratsimp(выражение) — упрощает выражение за счет рациональных преобразований. 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

fullratsimp(выражение) — функция упрощения рационального выражения методом последовательного применения к переданному выражению функции ratsimp(). За счет этого функция работает несколько медленнее, чем ratsimp(), зато дает более надежный результат.

expand(выражение) — раскрывает скобки в выражении на всех уровнях вложенности. В отличие от функции ratexpand(), не приводит дробно-слагаемые к общему знаменателю.

radcan(выражение) — функция упрощения логарифмических, экспоненциальных функций и степенных с нецелыми рациональными показателями, то есть корней (радикалов).

Часто при попытке упрощения выражения в Maxima может происходить на самом деле только его усложнение. Увеличение результата может происходить из-за того, что неизвестно, какие значения могут принимать переменные, входящие в выражение. Чтобы этого избежать, следует накладывать ограничения на значения, которые может принимать переменная. Делается это с помощью функции assume(условие). Поэтому в некоторых случаях наилучшего результата можно добиться, комбинируя radcan() с ratsimp() или fullratsimp().

Пример. Упростить выражение
$$\frac{\sqrt{ab}a^{1/4}}{(b+a)\left(\frac{b^2}{a}\right)^{1/4}} - \frac{a^2+b^2}{a^2-b^2}.$$

Если применить к нашему выражению команду упростить рационально, то получим:

```
(%i1)
sqrt(a*b)*a^(1/4)/((a+b)*(b^2/a)^(1/4)) - (a^2+b^2)/(a^2-b^2);
```

```
(%o1) 
$$\frac{\sqrt{a}\sqrt{ab}}{(b+a)\sqrt{|b|}} - \frac{b^2+a^2}{a^2-b^2}$$

```

```
(%i2) ratsimp(%o1);
```

```
(%o2) 
$$\frac{\sqrt{a}b\sqrt{ab}}{b^2\sqrt{|b|} - a^2\sqrt{|b|}} - \frac{a^{3/2}\sqrt{ab}}{b^2\sqrt{|b|} - a^2\sqrt{|b|}} + \frac{b^2}{b^2 - a^2} + \frac{a^2}{b^2 - a^2}$$

```

Применим функцию assume(условие) и наложим с ее помощью на некоторые переменные, входящие в выражение, ограничения на их значения:

```
(%i1) assume (b>0);
```

```
(%o1) [ b > 0 ]
```

```
(%i2) sqrt(a*b)*a^(1/4)/((a+b)*(b^2/a)^(1/4))
      -(a^2+b^2)/(a^2-b^2);
```

```
(%o2) 
$$\frac{a}{b+a} - \frac{b^2+a^2}{a^2-b^2}$$

```

```
(%i3) ratsimp(%o2);
```

```
(%o3) 
$$\frac{b}{b-a}$$

```

Как видим, получили компактный результат.

1.9. Решение алгебраических уравнений и их систем

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

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

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

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

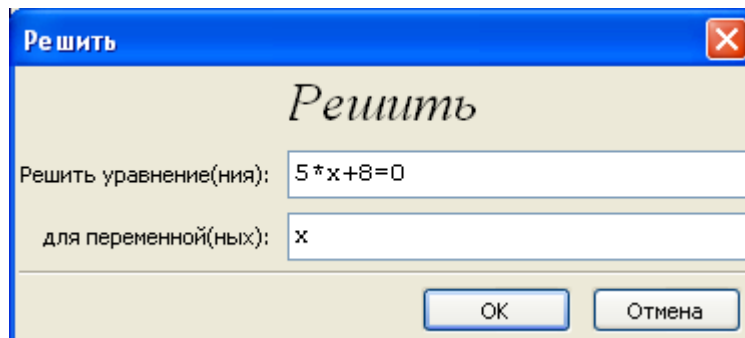


Рис. 13. Диалоговое окно для решения уравнений

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

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

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

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

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

```
(%i1) eq:x^3+1=0;
(%o1) x3 + 1 = 0
(%i2) resh:solve(eq, x);
(%o2) [ x = - $\frac{\sqrt{3} \%i - 1}{2}$ , x =  $\frac{\sqrt{3} \%i + 1}{2}$ , x = - 1 ]
```

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

```
(%i3) expand(ev(eq, resh[1]));
      expand(ev(eq, resh[2]));
      expand(ev(eq, resh[3]));
(%o3) 0 = 0
(%o4) 0 = 0
(%o5) 0 = 0
```

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

```
(%i8) eq:(1+2*x)^3=13.5*(1+x^5);
(%o8) (2 x + 1)3 = 13.5 (x5 + 1)
(%i12) allroots(eq);
(%o12) [ x = 0.82967499021294, x = -1.015755543828121, x =
0.96596251521964 %i - 0.40695972319241, x = -0.96596251521964
%i - 0.40695972319241, x = 1.0 ]
```

С помощью команды `solve` можно находить решение систем линейных алгебраических уравнений. Например, система линейных уравнений

$$\begin{cases} x + 2y + 3z + 4k + 5m = 13 \\ 2x + y + 2z + 3k + 4m = 10 \\ 2x + 2y + z + 2k + 3m = 11 \\ 2x + 2y + 2z + k + 2m = 6 \\ 2x + 2y + 2z + 2k + m = 3 \end{cases}$$

может быть решена следующим образом:

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

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

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

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

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

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

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

2. Находим решение системы.

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

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

3. Выполним проверку найденного решения:

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

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

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

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

Для решения систем нелинейных уравнений можно воспользоваться командой `algsys`. Например, найдем решение системы уравнений

$$\begin{cases} x^2 + 16y = 9 \\ 25x + 9y^2 = 16 \end{cases}$$

. Воспользуемся пунктом меню *Уравнения* → *Solve algebraic system*.

В диалоговом окне вводим количество уравнений системы: 2.

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

После нажатия на кнопку *OK* получим решения:

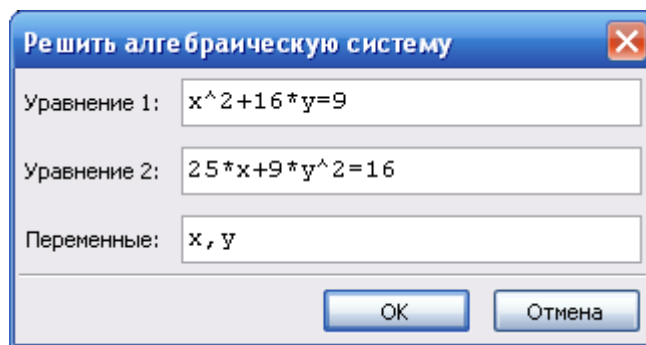


Рис.14. Ввод системы уравнений

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

1.10. Графические возможности

Графические возможности в Maxima реализованы посредством внешних программ. По умолчанию, построением графиков в Maxima занимается программа Gnuplot и разрабатываемый вместе с Maxima и идущий в ее же пакете Openmath.

Рассмотрим обзорно некоторые возможности системы для графической визуализации данных.

Для построения графиков на плоскости можно использовать команду plot2d: **plot2d(выражение, [символ, начало, конец])**, где выражение задает функцию, график которой нужно построить, символ — неизвестное, входящее в выражение, начало и конец задают отрезок оси X для построения графика, участок по оси Y выбирается автоматически, исходя из минимума и максимума функции на заданном промежутке. После вызова функции plot2d открывается окно Gnuplot graph с выполненным построением. График можно только масштабировать за счет изменения размеров окна. Также можно посмотреть координаты какой-либо точки графика функции. Чтобы построить в одной плоскости одновременно два графика (или больше), в функции plot2d следует вместо отдельного выражения указать их список.

С помощью команды `plot2d` можно строить графики параметрически заданных функций. Для этого используется список с ключевым словом `parametric`:

`plot2d([parametric, x-выражение, y-выражение, [переменная, начало, конец], [nticks, количество])`.

Здесь `x-выражение` и `y-выражение` задают зависимость координат от параметра, то есть это две функции вида $x(t)$, $y(t)$, где t — переменная параметризации. Эта же переменная прописывается в следующем списке, параметры `начало`, `конец` задают отрезок, в пределах которого этот параметр будет изменяться. Последний аргумент-список, с ключевым словом `nticks`, задает количество точек, на которые будет разбит интервал изменения параметра при построении графика.

Кроме `parametric`, функция `plot2d` может выполнять построение графиков дискретных множеств (конечных наборов точек):

`plot2d([discrete, x-список, y-список])` и `plot2d([discrete, [x, y]-список])`.

Для выполнения построений дополнительно в системе Maxima есть пакет `Draw` (загружается пакет с помощью команды `load(draw)`), в который, в частности, входит функция:

`draw2d (опции, explicit(имя_функции, независимая_переменная, min, max), опции)` — функция, предназначенная для построения графиков на плоскости с применением большого количества дополнительных опций:

- **`xrange, yrange`** — установлены по умолчанию — определяют промежутки изменения значений переменной по осям Ox и Oy . В случае необходимости, можно изменять значений вручную. Например, `xrange=[-2, 3]`;

- **`grid`** — в случае, если `grid=true`, на координатной плоскости выводятся линии сетки;

- **`title`** — позволяет выводить заголовок к графику функции. Например, `title = "Exponential function"`;

- **`xlabel, ylabel`** — позволяют выводить подписи к осям. Например, `ylabel = "Population"`;

- **`xtics, ytics`** — позволяют устанавливать цену деления по осям Ox и Oy , с которой будут наноситься метки на оси. Имеет значение по умолчанию, однако их действием можно управлять вручную. Например, можно задать, чтобы метки по оси Ox наносились на промежутке от -3 до 3 с шагом $0,2$: `xtics= [-3, 0.2, 3]`. Также можно указать, в каком виде выводить подписи к осям (см. пример 6);

- **`xaxis, yaxis`** — в случае, если значения этих опций равны `true`, координатные оси выводятся на экран;

- **xaxis_width, yaxis_width** — ширина координатных осей (по умолчанию ширина равна 1). Для изменения толщины оси необходимо изменить значение по умолчанию вручную, например, `xaxis_width=3`;

- **xaxis_type, yaxis_type** — стиль линии осей Ox и Oy . Допустимые значения: `solid` и `dots`;

- **xaxis_color, yaxis_color** — цвет координатных осей (по умолчанию — черный). Для изменения цвета оси необходимо изменить значение опции вручную, например, `xaxis_color = red`;

- **color** — позволяет изменять цвет графика. Например, `color=»red»` (дается до слова `explicit`);

- **line_width** - позволяет изменять толщину линии графика функции (значение по умолчанию — 1);

- **line_type** — позволяет изменять стиль линии графика функции. Допустимые значения: `solid` и `dots` и др.

В системе Maxima есть возможность выполнять построение различных графических примитивов, например:

polygon (`[[x1,y1], [x2,y2],...`]) — построение замкнутой ломаной линии, соединяющей точки с координатами `[x1,y1], [x2,y2], ...`. С помощью опции `fill_color` можно заливать фигуру выбранным цветом;

points (`[[x1,y1], [x2,y2],...`]) — построение точек с координатами `[x1,y1], [x2,y2], ...`. Опция `point_type` позволяет выбрать стиль точки, например, окружность: `point_type= circle`. Опция `point_size` позволяет установить размер точки, например, `point_size = 3`. Опция `key` позволяет выполнять подписи к точкам;

rectangle (`[x1,y1], [x2,y2]`) — построение прямоугольника, где `[x1,y1], [x2,y2]` — координаты противоположных углов;

bars (`[x1,h1,w1], [x2,h2,w2, ...]`) — построение столбиковых диаграмм. Здесь `x1, x2, ...` - точки, относительно которых центрируется столбик, `h1, h2, ...` - высота столбиков, `w1, w2, ...` - ширина столбиков;

ellipse (`xc, yc, a, b, ang1, ang2`) — построение эллипса с центром в точке `(xc, yc)`.

В системе Maxima есть встроенная функция для построения графиков функций, заданных неявно. Ее синтаксис:

implicit_plot (`expr, x_range, y_range`)

implicit_plot (`[expr_1, ..., expr_n], x_range, y_range`)

где `expr` – уравнение, задающее неявную функцию, `x_range` и `y_range` – промежутки изменения переменных `x` и `y`.

Для того, чтобы можно было использовать функцию `implicit_plot`, необходимо подключить пакет, содержащий эту функцию, с помощью команды `load(implicit_plot)`.

Кроме того, в системе Maxima можно воспользоваться встроенными функциями для построения графиков в различных системах координат. Например, функция

polar (radius, ang, minang, maxang)

выполняет построение графика функции $\text{radius}(\text{ang})$, заданной в полярной системе координат, аргумента ang , меняющего значения от minang до maxang .

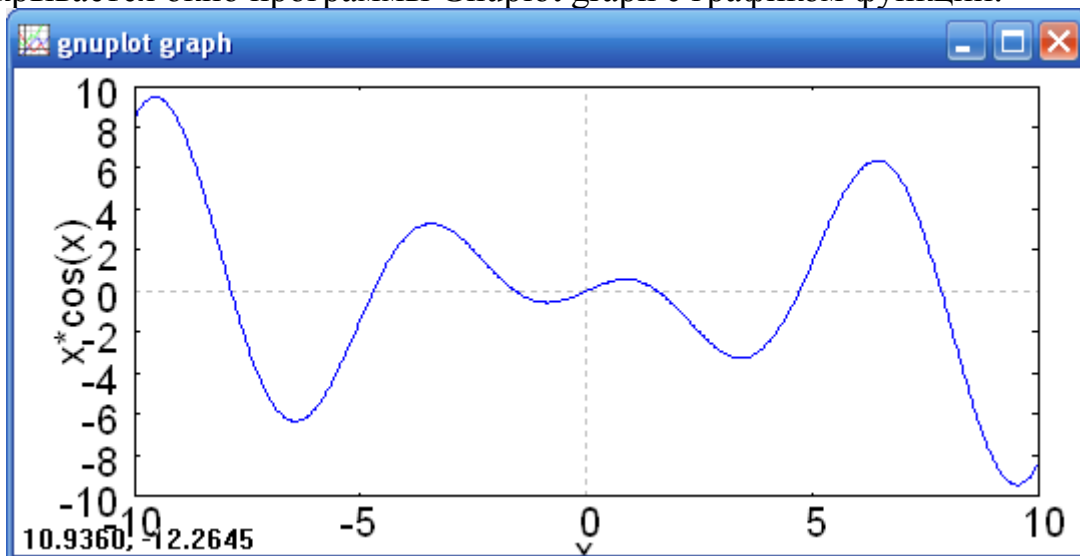
Приведем несколько примеров.

Пример 1. Построить график функции $y = x \cos x$.

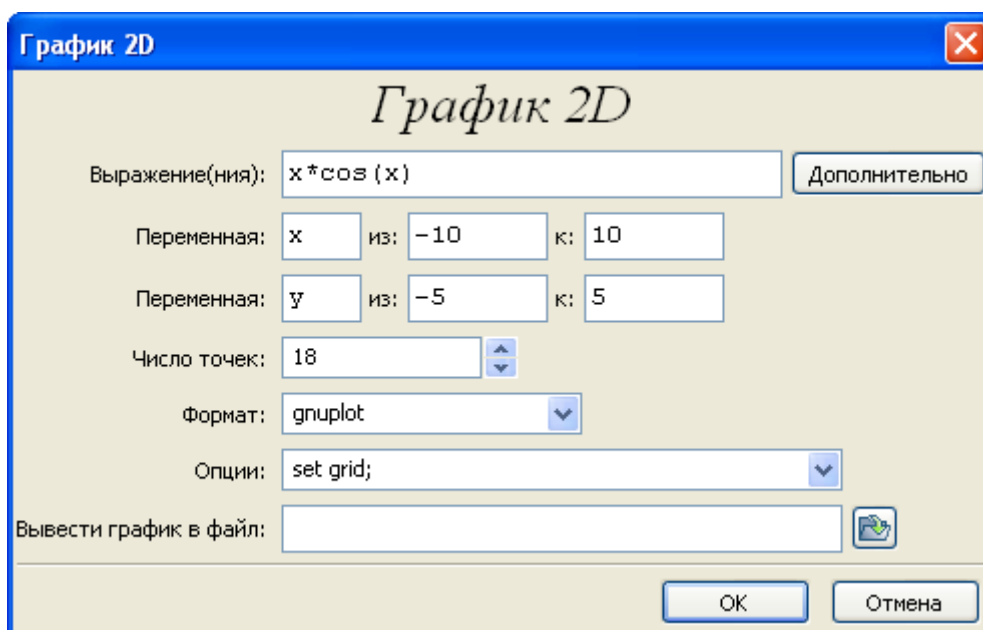
1 способ. В ячейке ввода задаем команду: `plot2d(x*cos(x),[x, -10, 10])`. После нажатия клавиш `Ctrl+Enter` формируется ячейка ввода в документе

```
(%i1) plot2d(x*cos(x), [x, -10, 10]);
```

и открывается окно программы Gnuplot graph с графиком функции:



2 способ. В нижней панели инструментов выбираем кнопку *График2d*, появляется диалоговое окно, в котором предлагается ввести выражение для графика функции, пределы изменения переменной по оси X и Y, количество точек графика, выбрать формат для построения графика функции, задать, в случае необходимости, дополнительные опции. Заметим, что здесь также можно, нажав на кнопку *Дополнительно*, задать функцию в параметрическом виде и дискретную функцию. Например, для построения графика функции $y = x \cos x$ выберем следующие параметры:



При нажатии на кнопку *Ok* получим тот же график.

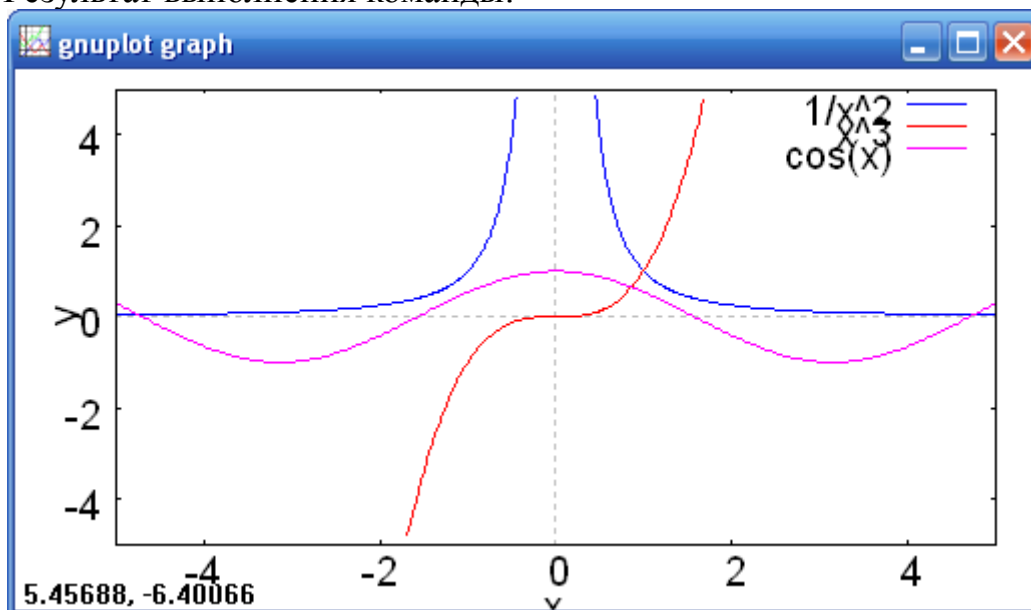
Таким образом, можно выбирать наиболее удобный способ построения графиков функций на плоскости.

Пример 2. Построить в одной координатной плоскости графики функций $y = \frac{1}{x^2}$, $y = x^3$, $y = \cos x$.

Сформируем ячейку ввода:

```
(%i8) plot2d([1/x^2, x^3, cos(x)], [x,-5,5],[y,-5,5],
            [gnuplot_preamble, "set zeroaxis;"])$
```

Результат выполнения команды:



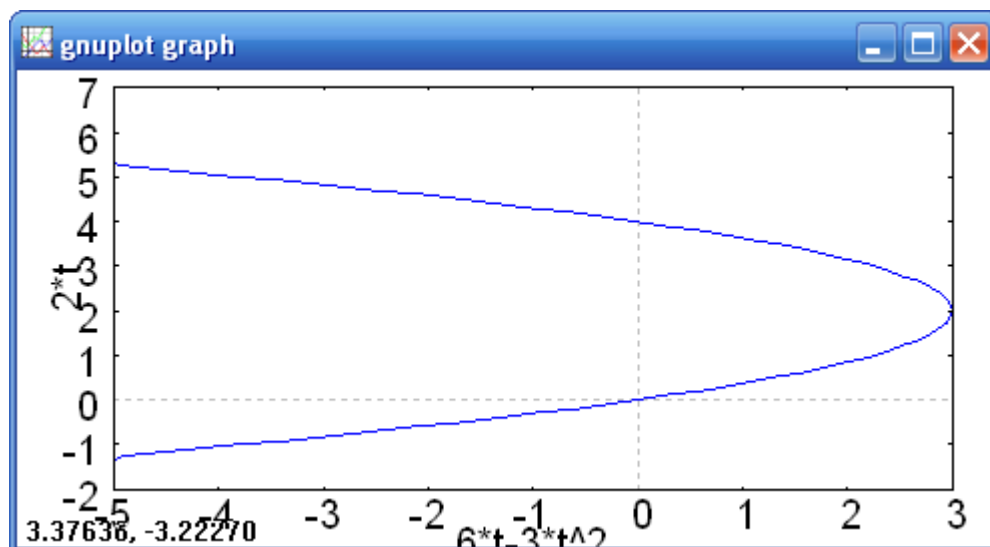
Как видим, система автоматически определяет цвет для каждого графика.

Пример 3. Построить график функции $\begin{cases} x(t) = -3t^2 + 6t \\ y(t) = 2t \end{cases}$.

Выполним построение графика параметрически заданной функции следующим образом. Вызываем диалоговое окно для построения графика функции, нажав кнопку *График2d*. В этом окне выбираем *Дополнительно* > *Параметрический график*. Открывается диалоговое окно для ввода функции. Заполняем:

Нажимаем на кнопку *Ok*. Теперь вводим пределы изменения переменных x и y в окне *График 2D*.

Нажимаем на кнопку *Ok*. В результате получаем график:



```
(%i5) plot2d(['parametric, -3*t^2+6*t, 2*t,
             [t, -6, 6], [nticks, 300]],
             [x,-5,3], [y,-2,7],
             [plot_format, gnuplot])$
```

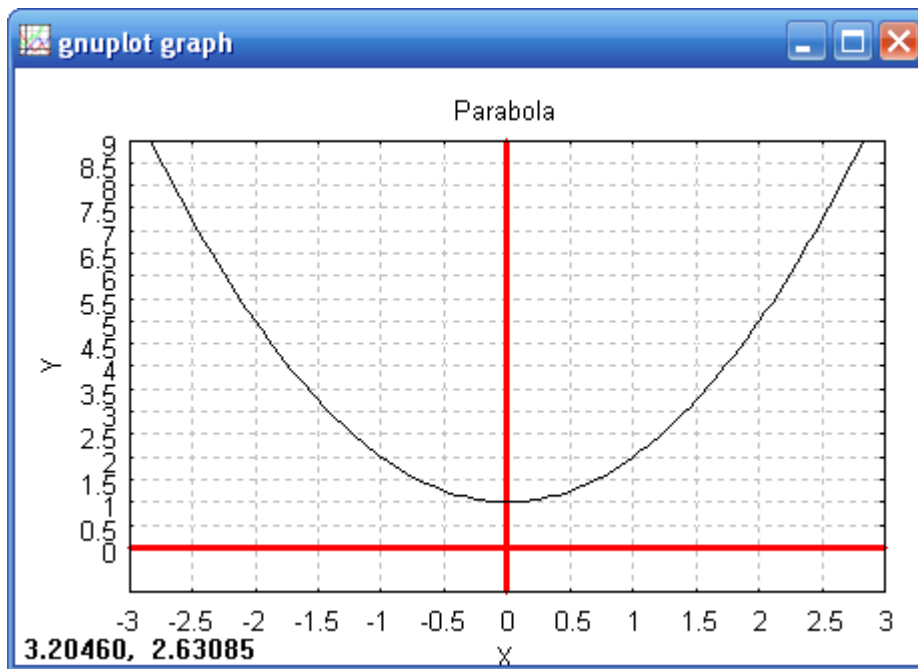
Пример 4. Построить график функции $y = x^2 + 1$ на отрезке от -3 до 3, значения y меняются от -1 до 9. Нанести линии сетки на указанном промежутке, цена деления – 0.5. Вывести координатные оси красным цветом, сделать подписи к осям Ox и Oy , подписать название графика функции – «Парабола».

Подключим пакет draw:

```
(%i1) load(draw);
(%o1)
```

```
C:/PROGRA~1/MAXIMA~1.0/share/maxima/5.15.0/share/draw/draw.lisp
```

Теперь можно воспользоваться функцией draw2d. Воспользуемся опциями этой функции: xrange, yrange — для задания промежутков изменения значений по осям Ox и Oy , xlabel, ylabel — для задания подписей к осям, title — для отображения заголовка к графику функции, xtics, ytics — для установления цены деления по осям Ox и Oy , grid — для нанесения координатной сетки, xaxis, yaxis — для вывода осей координат и их пересечения в точке (0,0), xaxis_color, yaxis_color — для задания цвета координатных осей, xaxis_width, yaxis_width — для изменения толщины линии осей Ox и Oy . Получаем:



```
(%i2) draw2d(xrange=[-3,3],xlabel="X",xtics=[-3,1/2,3],
grid=true,explicit(x^2+1,x,-3,3),
ylabel="Y",yrange=[-1,9],ytics=[0,1/2,9],
title="Parabola",xaxis=true,yaxis=true,
xaxis_color=red,yaxis_color=red,
xaxis_width=3,yaxis_width=3);
```

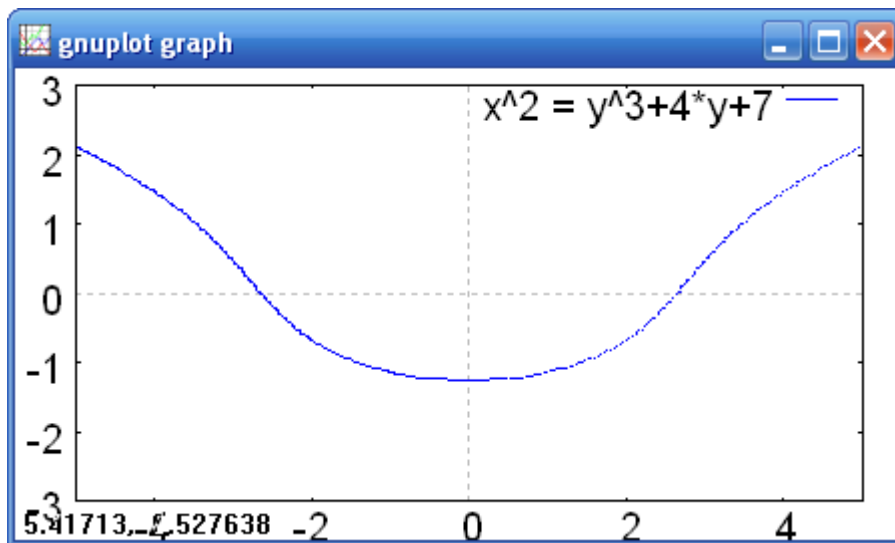
Пример 5. Построить график функции $x^2 = y^3 + 4y + 7$.

Как видим, функция задана неявно. Поэтому воспользуемся пакетом для построения графиков неявно заданных функций `implicit_plot`. Для этого выполним загрузку пакета:

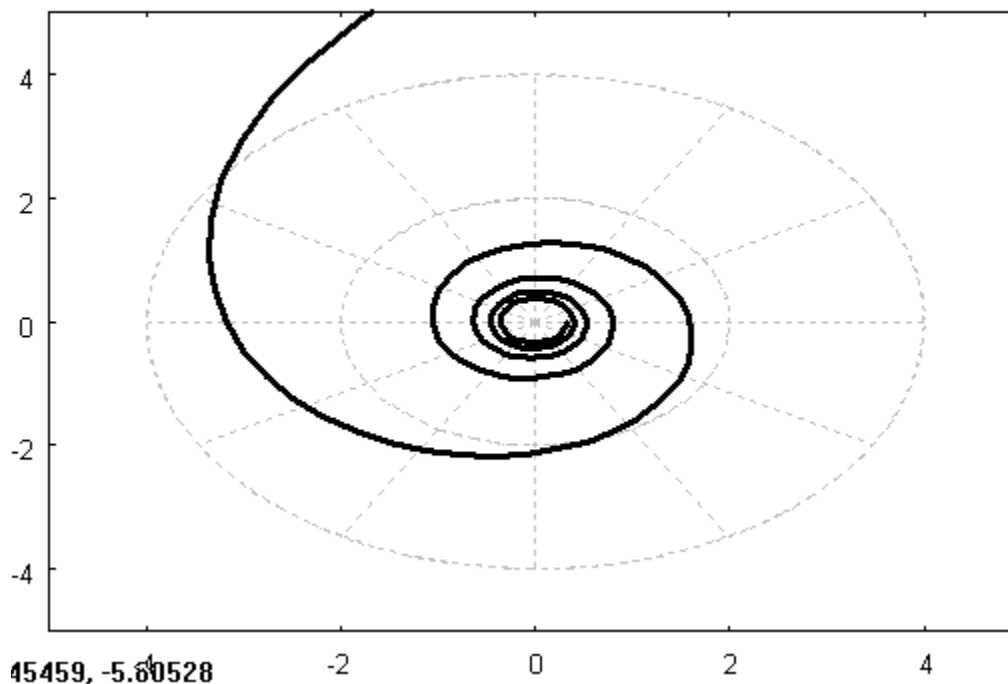
```
(%i1) load(implicit_plot)$
```

Теперь выполняем построение графика:

```
(%i2) implicit_plot(x^2=y^3+4*y+7,[x,-5,5],[y,-3,3],
[gnuplot_preamble,"set zeroaxis"]);
```



Пример 6. Построить график гиперболической спирали $r(\theta) = 10/\theta$.



```
(%i68) load(draw);
```

```
(%i69)
draw2d(user_preamble="set grid polar", nticks=200,
xrange=[-5,5],yrange=[-5,5],line_width=3,polar(10/theta,
theta,1, 10*%pi));
```

Для построения графиков поверхностей и кривых в пространстве предназначена функция `plot3d`. Функция `plot3d` имеет два варианта вызова: один для явного задания функции и один для параметрического. В обоих случаях функция принимает три аргумента.

Синтаксис для явно заданной функции:

plot3d(выражение, [переменная1, начало, конец], [переменная2, начало, конец]);

аргументы аналогичны plot2d, с той разницей, что здесь независимых переменных две.

График параметрически заданной функции строится так:

plot3d([выражение1, выражение2, выражение3], [переменная1, начало, конец], [переменная2, начало, конец]);

где выражения соответствуют, по порядку, $x(u, v)$, $y(u, v)$, $z(u, v)$.

Функция plot3d имеет ряд опций. Опция grid применима к трехмерным графикам вместо опции nticks, используемой для двумерных. Она задается в виде двух целых значений, которые для поверхностей задают размер ячеек сетки, в виде которой отображается поверхность; первое число — вдоль оси X , второе — вдоль оси Y ; либо, в случае параметрического задания, по первому и по второму параметру соответственно. Для кривых из этих параметров действует только один, но писать нужно опять же оба.

Опция, задающая формат вывода результата — plot_format. Формат может принимать одно из четырех значений, первое из которых действует по умолчанию: gnuplot, openmath и встроенный. В умолчательном варианте (значение gnuplot) данные для отображения передаются напрямую программе gnuplot, которая сама по себе имеет достаточно гибкое управление, и параметры ей можно передавать прямо из Maxima с помощью дополнительных опций функций plot2d/3d. Gnuplot генерирует статичное изображение, mgnuplot и openmath позволяют в реальном времени масштабировать и передвигать картинку, plot3d — еще и вращать линию или поверхность в разные стороны в пространстве.

Openmath предоставляет хорошую интерактивность: после того, как объект сгенерирован, его можно масштабировать и динамично вращать, разглядывая со всех сторон.

Опция преобразования системы координат transform_xy (по умолчанию она равна false).

Передавать ей нужно выражение, сгенерированное функцией make_transform([x, y, z], f1(x, y, z), f2(x, y, z), f3(x, y, z)). Кроме того, существует одно встроенное преобразование, известное как polar_xy и соответствующее make_transform([r, th, z], r*cos(th), r*sin(th), z), то есть переходу к полярной цилиндрической системе координат.

Для построения 3D графика функции в сферической системе координат используется функция

spherical(radius, azi, minazi, maxazi, zen, minzen, maxzen)

где функция radius(azi, zen) задается в сферических координатах.

Для построения 3D графика функции в цилиндрической системе координат используется функция

cylindrical (radius,z,minz,maxz,azi,minazi,maxazi)

где функция radius(z, azi) задается в цилиндрических координатах.

Пример 1. Построить график поверхности $z=6x^2+7y^3$.

```
(%i22) plot3d(6*x^2+7*y^3, [x,-3, 3], [y, -3, 3]);
```

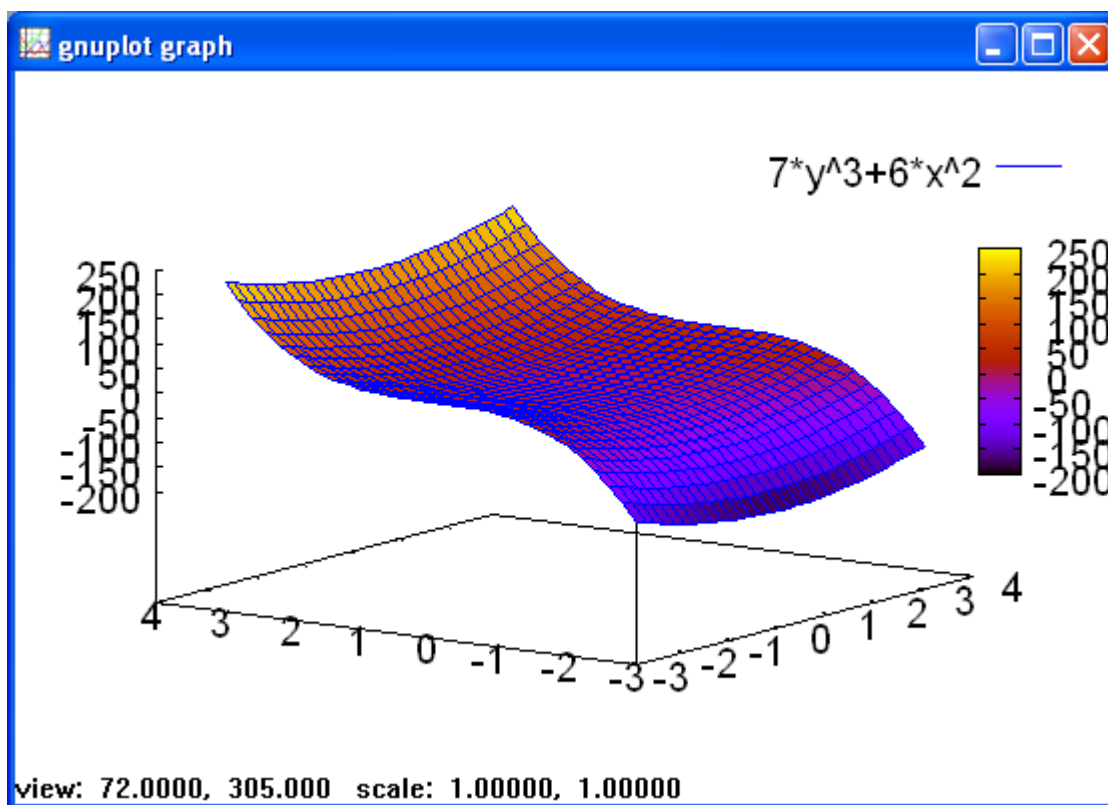


График поверхности можно вращать, удерживая левую кнопку мыши.

Можно выполнить построение поверхности в виде каркаса.

```
(%i68) load(draw);
```

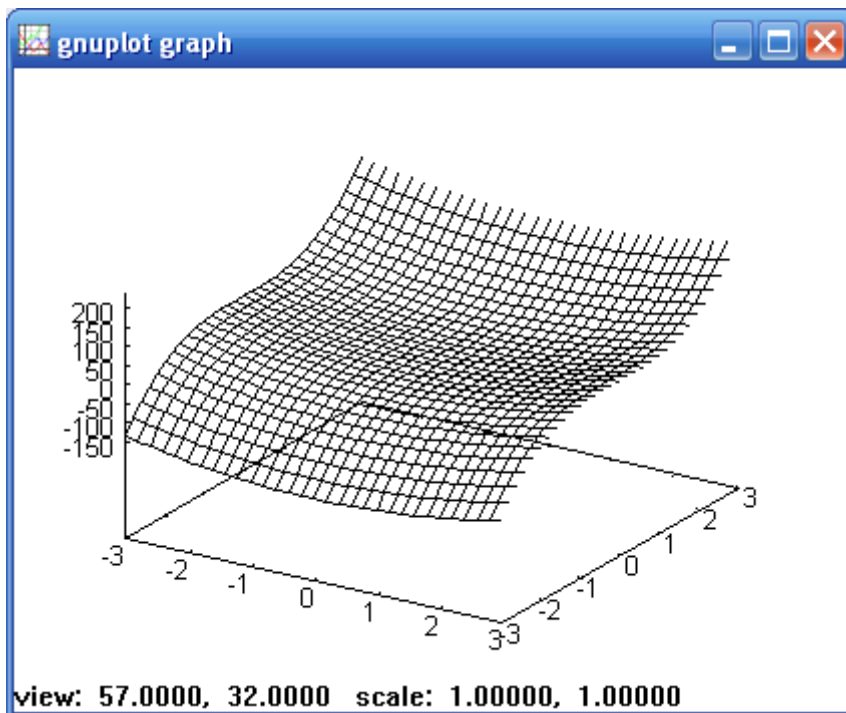
```
(%i82)
```

```
draw(gr3d(explicit(6*x^2+7*y^3, x, -3, 3, y, -3, 3)));
[gr3d(explicit)];
```

или

```
(%i84) draw3d(explicit(6*x^2+7*y^3, x, -3, 3, y, -3, 3));
[gr3d(explicit)];
```

Результат выполнения будет один и тот же.

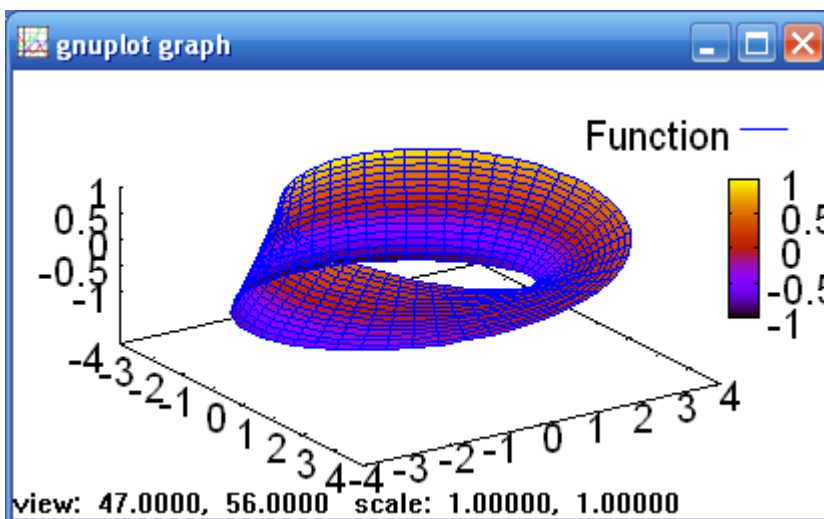


Пример 2. Построить лист Мебиуса

$$u(x, y) = \cos x (3 + y \cos \frac{x}{2}), v(x, y) = \sin x (3 + y \cos \frac{x}{2}), w(x, y) = y \sin \frac{x}{2}.$$

Используем синтаксис функции plot3d в случае параметрически заданной функции:

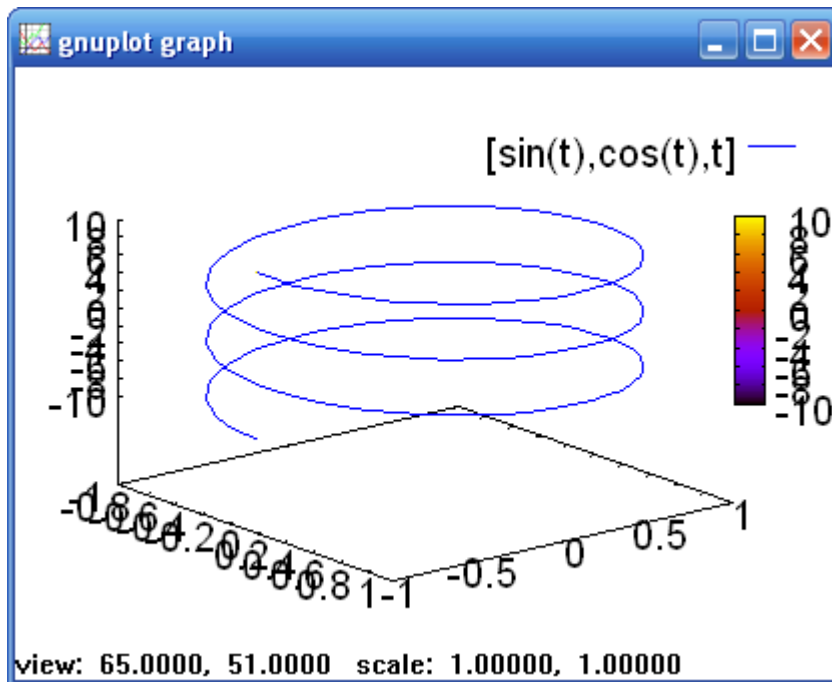
```
(%i1)
plot3d([cos(x)*(3+y*cos(x/2)), sin(x)*(3+y*cos(x/2)),
y*sin(x/2)], [x, -%pi, %pi], [y, -1, 1],
['grid, 50, 15]);
```



С помощью параметрической формы можно строить и пространственные кривые. Для этого просто нужно задать второй (фиктивный) параметр, чтобы Maxima не ругалась на неправильный синтаксис вызова функции.

Пример 3. Построить кривую в пространстве $x=\sin t, y=\cos t$.

```
(%i1) plot3d([sin(t), cos(t), t], [t, -3*%pi, 3*%pi],
             [v, 0,1], [grid, 150, 150]);
```

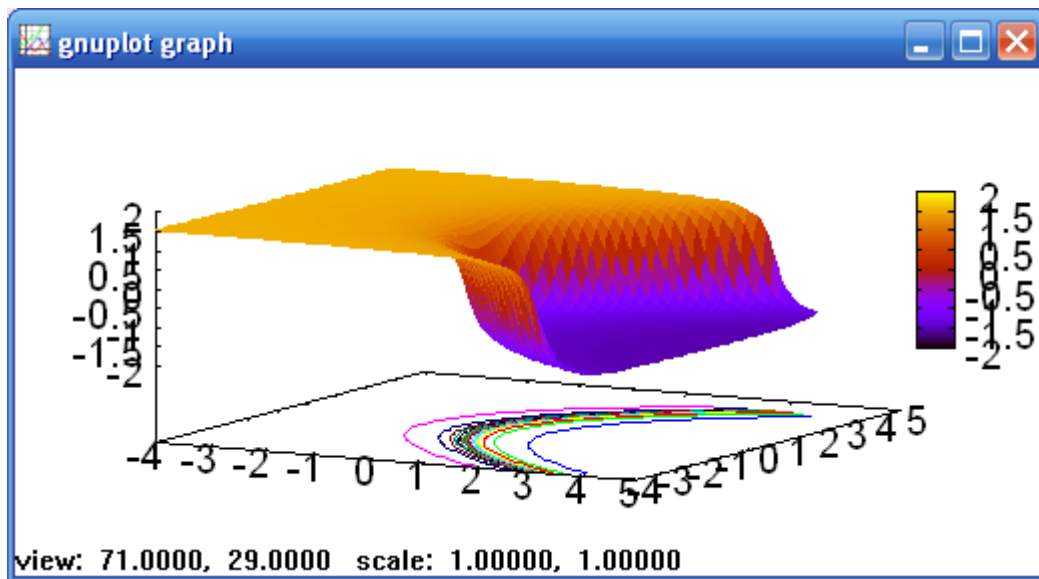


В системе Maxima можно выполнять построение графиков поверхностей и проекцию контуров графика на нижнюю плоскость.

Пример 6. Построить график поверхности $z=\arctg(-5x+y^2)$ и проекцию его контуров на нижнюю плоскость.

```
(%i1)
my_preamble:"set pm3d at s; unset surface; set contour;
set cntrparam levels 20; unset key"$

(%i2) plot3d(atan(-5*x+y^2), [x, -4, 4], [y,-4,4],
             [grid, 50,50],[gnuplot_pm3d, true],
             [gnuplot_preamble,my_preamble]);
```



На этом все возможности системы не исчерпываются. Более подробную информацию можно найти в справочной системе.

2.1. Общие сведения о дифференциальных уравнениях

При изучении физических явлений часто не удается непосредственно найти закон, связывающий независимые переменные и искомую функцию, но можно установить связь между этой функцией и ее производными, выражаемую дифференциальным уравнением.

Как известно, *дифференциальным уравнением* называется уравнение, связывающее независимую переменную x , искомую функцию $y = y(x)$ и ее производные $y'(x), \dots, y^{(n)}(x)$, т. е. уравнение вида

$$F(x, y(x), y'(x), \dots, y^{(n)}(x)) = 0. \tag{1.1}$$

Здесь F - известная функция, x - независимое переменное, $y(x)$ - неизвестная функция.

Если искомая функция $y = y(x)$ есть функция одной переменной, то дифференциальное уравнение называется *обыкновенным*.

Порядком дифференциального уравнения называется наивысший порядок входящих в него производных.

Решением дифференциального уравнения n -го порядка на интервале (a, b) называется функция $y = y(x)$, определенная на (a, b) вместе со своими производными до n -го порядка включительно, и такая, что подстановка функции $y = y(x)$ в дифференциальное уравнение превращает последнее в тождество по x на (a, b) .

График решения дифференциального уравнения называется *интегральной кривой* этого уравнения.

Дифференциальное уравнение (1.1) имеет бесконечно много решений. Множество всех решений уравнения (1.1) называется *общим решением уравнения* (1.1). Всякое отдельно взятое решение называется его *частным решением*.

Задача нахождения решения $y = y(x)$ уравнения (1), удовлетворяющего начальным условиям $y|_{x=x_0} = y_0, y'|_{x=x_0} = y'_0, \dots, y^{(n-1)}|_{x=x_0} = y_0^{(n-1)}$, называется *задачей Коши* для уравнения (1.1).

Теорема существования и единственности решения задачи Коши. Если в уравнении (1.1) функция $f(x, y, y', y'', \dots, y^{(n-1)})$

А) непрерывна по всем своим аргументам $x, y, y', y'', \dots, y^{(n-1)}$ в некоторой области D их изменения;

Б) имеет ограниченные в области D частные производные $\frac{\partial f}{\partial y}, \frac{\partial f}{\partial y'}, \frac{\partial f}{\partial y''}, \dots, \frac{\partial f}{\partial y^{(n-1)}}$ по аргументам $x, y, y', y'', \dots, y^{(n-1)}$, то найдется интервал $x_0 - h < x < x_0 + h$,

на котором существует единственное решение $y = y(x)$, удовлетворяющее условиям $y|_{x=x_0} = y_0, y'|_{x=x_0} = y_0', \dots, y^{(n-1)}|_{x=x_0} = y_0^{(n-1)}$, где значения $x = x_0, y = y_0, y' = y_0', \dots, y^{(n-1)} = y_0^{(n-1)}$ содержатся в области D .

Решение дифференциального уравнения, в каждой точке которого его касается другое решение, отличное от рассматриваемого решения в сколь угодно малой окрестности этой точки, называется *особым решением* дифференциального уравнения. В каждой точке особого решения нарушается единственность.

Существует несколько классов дифференциальных уравнений и для каждого из них существуют различные методы решения.

Различают следующие обыкновенные дифференциальные уравнения первого порядка:

- *Уравнения с разделяющимися переменными и приводящиеся к ним.*

Уравнение вида $g_1(y)f_1(x)dx = g_2(y)f_2(x)dy$, в котором коэффициенты при дифференциалах распадаются на множители, зависящие только от x и только от y , называется уравнением с разделяющимися переменными. Уравнение вида $g(y)dy = f(x)dx$ называется уравнением с разделенными переменными.

- *Однородные уравнения и приводящиеся к ним.*

Дифференциальное уравнение вида $y' = f(x, y)$, если $f(x, y)$ есть однородная функция своих аргументов нулевого измерения, называется однородным дифференциальным уравнением. Его можно представить в виде $y' = f(y/x)$. С помощью замены $u = \frac{y}{x}$ его можно привести к уравнению с разделяющимися переменными.

- *Уравнения в полных дифференциалах.*

Дифференциальное уравнение вида $M(x, y)dx + N(x, y)dy = 0$ называется уравнением в полных дифференциалах, если его левая часть есть полный дифференциал от некоторой функции $F(x, y)$, т.е. $Mdx + Ndy \in dF \in \frac{\partial F}{\partial x} dx + \frac{\partial F}{\partial y} dy$.

- *Линейные уравнения первого порядка.*

Линейным дифференциальным уравнением первого порядка называется уравнение, в которое y и y' входят линейно, то есть в первой степени. Оно имеет вид $y' + p(x)y = q(x)$, где $p(x)$ и $q(x)$ - заданные функции от x , непрерывные в той области, в которой требуется проинтегрировать уравнение.

Уравнение Бернулли - уравнение, которое можно записать в виде $y' + p(x)y = q(x)y^\alpha$, $\alpha \neq 0, \alpha \neq 1$. С помощью замены переменной $z = \frac{1}{y^{n-1}}$ уравнение Бернулли приводится к линейному уравнению.

- *Уравнения первого порядка, не разрешенные относительно производной.*

Эти уравнения имеют вид $F(x, y, y') = 0$, где F - заданная функция трех аргументов, F нелинейна по y' . Это уравнение при определенных условиях эквивалентно нескольким (и даже бесконечному множеству) уравнений вида $y' = f_i(x, y)$, ($i = 1, 2, \dots$) по числу корней уравнения относительно y' . К такому классу уравнений относится уравнение Лагранжа $y = xf(y') + g(y')$ и Клеро $y = xy' + g(y')$.

- *Уравнение Риккати.*

Дифференциальное уравнение первого порядка вида $y' + a(x)y^2 + b(x)y + c(x) = 0$, где $a(x), b(x), c(x)$ - известные функции, называется уравнением Риккати.

Различают следующие обыкновенные дифференциальные уравнения высших порядков:

- *Уравнения, допускающие понижение порядка.*

К ним относятся уравнения вида $y^{(n)} = f(x), F(y, y', y'', \dots, y^{(n)}) = 0$ и др.

- *Линейные однородные уравнения с постоянными коэффициентами.*

Дифференциальное уравнение вида $a_0 y^{(n)} + a_1 y^{(n-1)} + \dots + a_n y = 0$, где a_0, a_1, \dots, a_n - вещественные постоянные, $a_0 \neq 0$, называется линейным однородным дифференциальным уравнением с постоянными коэффициентами.

- *Линейные неоднородные уравнения с постоянными коэффициентами.*

Дифференциальное уравнение вида $a_0 y^{(n)} + a_1 y^{(n-1)} + \dots + a_n y = f(x)$, где a_0, a_1, \dots, a_n - вещественные постоянные, $a_0 \neq 0$, называется линейным неоднородным дифференциальным уравнением с постоянными коэффициентами.

- *Уравнения Эйлера.*

Линейные уравнения вида $a_0 x^n y^{(n)} + a_1 x^{n-1} y^{(n-1)} + \dots + a_{n-1} x y' + a_n y = 0$, где все a_i - постоянные, называются уравнениями Эйлера. С помощью замены $x = e^t$ его можно свести к линейному однородному уравнению с постоянными коэффициентами.

- *Линейные дифференциальные уравнения с переменными коэффициентами.*

Линейным дифференциальным уравнением n -го порядка с переменными коэффициентами называется уравнение вида $a_0(x)y^{(n)} + a_1(x)y^{(n-1)} + \dots + a_{n-1}(x)y' + a_n(x)y = f(x)$, где f, a_i - известные функции.

Дифференциальные уравнения в частных производных — это уравнения, содержащие неизвестные функции от нескольких переменных и их частные производные. Уравнения в частных производных имеют следующий вид:

$$F = \left(x_1, x_2, \dots, x_m, z, \frac{\partial z}{\partial x_1}, \frac{\partial z}{\partial x_2}, \dots, \frac{\partial z}{\partial x_m}, \frac{\partial^2 z}{\partial x_1^2}, \frac{\partial^2 z}{\partial x_1 \partial x_2}, \frac{\partial^2 z}{\partial x_2^2}, \dots, \frac{\partial^n z}{\partial x_m^n} \right),$$

где x_1, x_2, \dots, x_m — независимые переменные, а z — функция этих переменных.

Дифференциальные уравнения в частных производных классифицируются по разным признакам, причем для каждого класса, также как и для обыкновенных дифференциальных уравнений, существуют общие методы решения уравнений. Приведем основные методы классификации уравнений [4, с.10-12].

1. Порядок уравнения.
2. Число независимых переменных.
3. Линейность.
4. Однородность.
5. Виды коэффициентов (постоянные и переменные).

Все линейные уравнения с частными производными второго порядка вида

$$A u_{xx} + B u_{xy} + C u_{yy} + D u_x + E u_y + F u = G$$

относятся к одному из трех типов:

1. *Параболический тип.* Уравнения параболического типа описывают процессы теплопроводности и диффузии и определяются условием: $B^2 - 4AC = 0$.
2. *Гиперболический тип.* Уравнения гиперболического типа описывают колебательные системы и волновые движения и определяются условием $B^2 - 4AC > 0$.
3. *Эллиптический тип.* Уравнения эллиптического типа описывают установившиеся процессы и определяются условием $B^2 - 4AC < 0$.

В случае переменных коэффициентов тип уравнения может меняться от точки к точке.

Задача Коши — одна из основных задач теории дифференциальных уравнений (обыкновенных и с частными производными); состоит в отыскании решения (интеграла) дифференциального уравнения, удовлетворяющего так называемым начальным условиям (начальным данным).

Задача Коши обычно возникает при анализе процессов, определяемых дифференциальным законом и начальным состоянием, математическим выражением которых и являются уравнение и начальное условие (откуда терминология и выбор обозначений: начальные данные задаются при $t=0$, а решение отыскивается при $t>0$).

От краевых задач задача Коши отличается тем, что область, в которой должно быть определено искомое решение, здесь заранее не указывается. Тем не менее, задачу Коши можно рассматривать как одну из краевых задач.

Основные вопросы, которые связаны с задачей Коши, таковы:

Существует ли (хотя бы локально) решение задачи Коши?

Если решение существует, то какова область его существования?

Является ли решение единственным?

Если решение единственно, то будет ли оно корректным, то есть непрерывным (в каком-либо смысле) относительно начальных данных?

Говорят, что задача Коши имеет единственное решение, если она имеет решение $y=f(x)$ и никакое другое решение не отвечает интегральной кривой, которая в сколь угодно малой выколотой окрестности точки (x_0, y_0) имеет поле направлений, совпадающее с полем направлений $y=f(x)$. Точка (x_0, y_0) задаёт начальные условия.

В теории дифференциальных уравнений, начальные и граничные условия — дополнение к основному дифференциальному уравнению (обыкновенному или в частных производных), задающее его поведение в начальный момент времени или на границе рассматриваемой области соответственно.

Дифференциальные уравнения широко используются в практике математических вычислений. Они являются основой при решении задач моделирования — особенно в математической физике.

Решение дифференциальных уравнений может быть получено в символьном (аналитическом) или численном виде. *Под аналитическим решением* понимают такие решения, в которых неизвестная функция выражена через независимые переменные и параметры в виде формул, бесконечных рядов, интегралов. *Под численным решением* понимают решения, полученные численно после приближенной замены исходного уравнения другим, более простым уравнением [4, с.301-302].

Главное преимущество численных решений состоит в том, что их можно получить даже в том случае, когда аналитические решения получить невозможно.

2.2. Численные методы решения задачи Коши для обыкновенного дифференциального уравнения первого порядка

Рассмотрим постановку задачи Коши для системы обыкновенных дифференциальных уравнений (ОДУ) вида

$$\frac{dy}{dx} = f(x, y), \quad (2.1)$$

где: y - искомая вектор-функция; x — независимая переменная; $y(x) = (y_1(x), \dots, y_m(x))$; $f(x) = (f_1, \dots, f_m)$, m — порядок системы; $y_1(x), \dots, y_m(x)$ - координаты; $x \geq 0$; $y(0) = y^0$.

Систему (1) можно переписать в развернутом виде

$$\frac{d y_i}{d x} = f_i(x, y_1, \dots, y_m), \quad (2.2)$$

где: $i=1, \dots, m$; $y_i(0) = y_i^0$.

Если $i=1$, то мы получаем обыкновенное дифференциальное уравнение первого порядка:

$$\frac{d y}{d x} = f(x, y), \quad (2.3)$$

При этом решение задачи Коши для уравнения (2.3) заключается в нахождении интегральной кривой, проходящей через заданную точку и удовлетворяющую заданному начальному условию $y(a) = y_a$. Задача состоит в том, чтобы найти искомую функцию y , удовлетворяющую (2.3) и заданным начальным условиям.

Построение численных алгоритмов решения уравнения (2.1) опирается на дискретизацию задачи. Введем в области расчета $x \in [a, b]$ дискретный набор точек $x_i = a + hi, i=0, 1, \dots, N, h=(b-a)/N$, в которых будем вычислять приближенное решение. Точки x_i называются узлами интегрирования или узлами сетки, расстояние h - шагом интегрирования или шагом сетки. Сеточной областью (сеткой) называется совокупность всех узлов.

Для характеристики точности численного метода определяется погрешность приближенного решения по формуле:

$$\delta = \max_i |y_i - y(x_i)|, \quad (2.4)$$

где $y(x_i)$ - значение точного решения в узле сетки.

Существует два класса методов для решения задачи (2.1):

- 1) семейство одношаговых методов (Рунге-Кутты);
- 2) семейство многошаговых (m-шаговых) методов.

Численный метод называется *явным*, если вычисление решения в следующей точке y_{i+1} осуществляется по явной формуле. Метод называется *одношаговым*, если вычисление решения в следующей точке y_{i+1} производится с использованием только одного предыдущего значения y_i .

В дальнейшем будем рассматривать численные методы решения задачи Коши на примере уравнения первого порядка:

$$\begin{cases} \frac{dy}{dx} = a(x, y), a \leq x \leq b \\ y(a) = y_a \end{cases} \quad (2.5)$$

2.2.1. Метод Эйлера

Простейшим численным методом решения задачи Коши (2.5) для обыкновенного дифференциального уравнения является метод Эйлера, который еще называют методом ломаных Эйлера.

По оси x введем равномерную сетку с шагом $h > 0$, т.е. рассмотрим систему точек $x_i = \{x_i = i \cdot h, i = 0, 1, 2, \dots\}$. Обозначим через $y(x)$ точное решение задачи (2.5), а через $y_i = y(x_i)$ — приближенные значения функций y в заданной системе точек.

Заменяя в уравнении (2.5) производную в окрестности каждого i -го узла сетки разностным отношением, приходим к уравнению:

$$\frac{y_{i+1} - y_i}{h} = f(x_i, y_i), \quad i = 0, 1, 2, \dots, N-1, \quad y_0 = y_a. \quad (2.6)$$

Алгебраические соотношения между компонентами сеточной функции, которыми заменяются исходные дифференциальные уравнения в окрестности каждого узла сетки, называются *разностными уравнениями*. Поэтому уравнение (2.6) — разностное уравнение.

В окончательной форме значения y_{i+1} можно определить по явной формуле

$$y_{i+1} = y_i + h \cdot f(x_i, y_i). \quad (2.7)$$

Геометрическая интерпретация метода Эйлера: интегральная кривая $y(x)$ на отрезке $[a; b]$ приближается к ломаной, наклон которой определяется наклоном интегральной кривой уравнения в точке $[x_i; y_i]$ (рис.1).

Метод Эйлера относится к *явным одношаговым* методам. Вследствие систематического накопления ошибок метод используется редко или используется только для оценки вида интегральной кривой. Метод Эйлера называют методом Рунге-Кутты первого порядка точности.

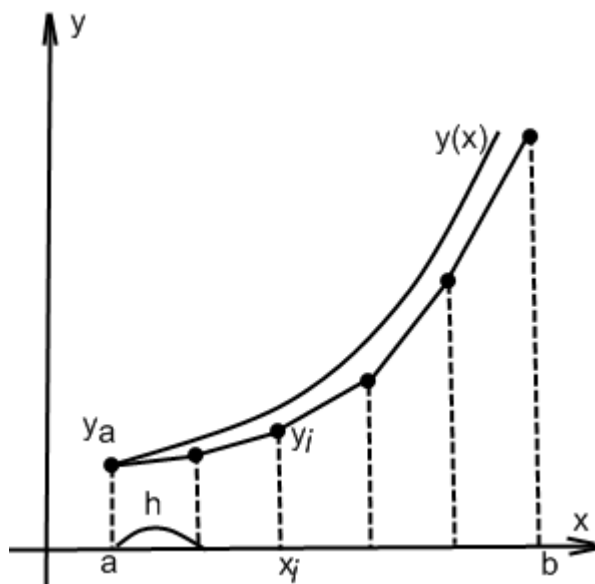


Рис.1. Геометрическая интерпретация метода Эйлера

ПРИМЕР 1. Решение задачи Коши методом Эйлера. Применяя метод Эйлера, найти решение задачи Коши: $\begin{cases} y' = y - x \\ y(0) = 1.5 \end{cases}$ в трех последователь-

ных точках $x_1=0.2$, $x_2=0.4$, $x_3=0.6$. Найти точное решение задачи и найти величину абсолютной погрешности в указанных точках.

Решение:

Возьмем шаг $h = 0.2$. Используя расчетную формулу Эйлера, найдем приближенное решение задачи Коши:

$$y_1 = y_0 + 0.2(y_0 - x_0) = 1.5 + 0.2 \cdot 1.5 = 1.8$$

$$y_2 = y_1 + 0.2(y_1 - x_1) = 1.8 + 0.2(1.8 - 0.2) = 2.12$$

$$y_3 = y_2 + 0.2(y_2 - x_2) = 2.12 + 0.2(2.12 - 0.4) = 2.464$$

Таким образом, получили численное решение задачи:

x_i	0	0.2	0.4	0.6
y_i	1.5	1.8	2.12	2.464

Графиком приближенного решения является ломаная, последовательно соединяющая точки (x_i, y_i) .

В этой задаче легко находится точное решение, например, методом вариации постоянной: $y(t) = 0.5e^t + t + 1$. Вычислим значения точного решения в указанных точках.

t_i	0	0.2	0.4	0.6
$y(t_i)$	1.5	1.811	2.146	2.511

Абсолютную погрешность вычислим так: $r_i = |y(t_i) - y_i|$. Тогда $r_1 = 0.011$, $r_2 = 0.026$, $r_3 = 0.047$. Таким образом, максимальная величина погрешности равна $R \approx 0.05$.

2.2.2. Метод Эйлера-Коши

Отличительная особенность метода Эйлера-Коши от метода Эйлера заключается в том, что значение правой части уравнения вычисляется не только в точках сетки (шаг h), но и также в середине отрезков (шаг $\frac{h}{2}$) (промежуточных точках).

Предположим, что приближенное значение y_i решения задачи в точке $x = x_i$ уже известно, y_{i+1} вычисляются по следующим формулам:

$$y_{i+1} = y_i + \Delta y_i, \Delta y_i = \Delta y_{i1} + \Delta y_{i2},$$

$$\Delta y_{i1} = \frac{h}{2} f(x_i, y_i), \Delta y_{i2} = \frac{h}{2} f(x_i + h, y_i + h f(x_i, y_i))$$

Отсюда вычисляют

$$y_{i+1} = y_i + h \frac{f(x_i, y_i) + f(x_{i+1}, y_i + h f(x_i, y_i))}{2}. \quad (2.8)$$

Геометрическая интерпретация метода Эйлера-Коши: определяется направление интегральной кривой в исходной точке (x_i, y_i) и во вспомогательной точке (x_{i+1}, y_{i+1}^o) , $y_{i+1}^o = y_i + h f(x_i, y_i)$, а в качестве окончательного выбирается среднее из этих направлений.

Метод Эйлера-Коши называют методом Рунге-Кутта второго порядка точности.

ПРИМЕР 2. Решение задачи Коши методом Эйлера-Коши. Применяя метод Эйлера-Коши, найти решение задачи Коши: $\begin{cases} y' = y - x \\ y(0) = 1.5 \end{cases}$ в трех последовательных точках $x_1 = 0.2$, $x_2 = 0.4$, $x_3 = 0.6$.

Решение:

Возьмем шаг $h = 0.2$. Используя расчетную формулу Эйлера-Коши (2.8), найдем приближенное решение задачи Коши:

$$y_1^0 = y_0 + hf(x_0, y_0) = 1.5 + 0.2(1.5 - 0) = 1.8$$

$$y_1 = y_0 + h \frac{y_0 - x_0 + f(x_1, y_1^0)}{2} = 1.5 + 0.2 \frac{1.5 - 0 + 1.8 - 0.2}{2}$$

$$y_1 = 1.5 + 0.2 \frac{1.5 + 1.6}{2} = 1.5 + 0.31 = 1.81$$

$$y_2^0 = y_1 + h(y_1 - x_1) = 1.81 + 0.2(1.81 - 0.2) = 2.132$$

$$y_2 = y_1 + h \frac{y_1 - x_1 + f(x_2, y_2^0)}{2}$$

$$y_2 = 1.81 + 0.2 \frac{(1.81 - 0.2 + 2.132 - 0.4)}{2} = 1.81 + 0.3342 = 2.1442$$

$$y_3^0 = y_2 + h(y_2 - x_2) = 2.1442 + 0.2(2.1442 - 0.4) = 2.49304$$

$$y_3 = y_2 + h \frac{y_2 - x_2 + f(x_3, y_3^0)}{2}$$

$$y_3 = 2.1442 + 0.2 \frac{2.1442 - 0.4 + 2.49304 - 0.6}{2} = 2.1442 + 0.363724 = 2.507924$$

Таким образом, получили численное решение задачи Коши:

x_i	0	0.2	0.4	0.6
y_i	1.5	1.81	2.1442	2.507924

Графиком приближенного решения является ломаная, последовательно соединяющая точки (x_i, y_i) .

Как видим, решения задачи Коши, полученные методом Эйлера и методом Эйлера-Коши очень близки.

2.2.3. Метод Рунге-Кутта

В вычислительной практике наиболее часто используется метод Рунге-Кутта четвертого порядка (классический метод Рунге-Кутта), поскольку позволяет наиболее точно находить решения обыкновенного дифференциального уравнения.

В этом методе значения y_{i+1} находятся по следующим формулам:

$$y_{i+1} = y_i + \Delta y_i; \Delta y_i = h(k_1 + 2k_2 + 2k_3 + k_4)/6, \text{ где } i = 0, 1, \dots$$

$$k_1 = f(x_i, y_i); \quad k_2 = f\left(x_i + \frac{h}{2}, y_i + \frac{h}{2} \cdot k_1\right); \quad (2.9)$$

$$k_3 = f(x_i + \frac{h}{2}, y_i + \frac{h}{2} \cdot k_2); \quad k_4 = f(x_i + h, y_i + h \cdot k_3).$$

ПРИМЕР 3. Решение задачи Коши методом Рунге-Кутты 4 порядка. Применяя метод Рунге-Кутты, найти решение задачи Коши: $\begin{cases} y' = y - x \\ y(0) = 1.5 \end{cases}$ в трех последовательных точках $x_1 = 0.2$, $x_2 = 0.4$, $x_3 = 0.6$.

Решение:

Возьмем шаг $h = 0.2$. Используя расчетные формулы Рунге-Кутты (2.9), найдем приближенное решение задачи Коши:

$$k_1 = f(x_0, y_0) = y_0 - x_0 = 1.5 - 0 = 1.5$$

$$k_2 = f(x_0 + \frac{h}{2}, y_0 + \frac{h}{2} k_1) = f(0 + 0.2/2, 1.5 + 0.2/2 \cdot 1.5) = f(0.1, 1.65) = 1.65 - 0.1 = 1.55$$

$$k_3 = f(0.1, 1.5 + 0.1 \cdot 1.55) = f(0.1, 1.655) = 1.655 - 0.1 = 1.555$$

$$k_4 = f(0.2, 1.5 + 0.2 \cdot 1.555) = f(0.2, 1.811) = 1.811 - 0.2 = 1.611$$

$$y_1 = y_0 + h(k_1 + 2k_2 + 2k_3 + k_4)/6 = 1.5 + 0.2(1.5 + 2 \cdot 1.55 + 2 \cdot 1.555 + 1.611)/6 = 1.8107$$

$$k_1 = f(x_1, y_1) = y_1 - x_1 = 1.8107 - 0.2 = 1.6107$$

$$k_2 = f(x_1 + \frac{h}{2}, y_1 + \frac{h}{2} k_1) = f(0.2 + 0.1, 1.8107 + 0.1 \cdot 1.6107) = f(0.3, 1.97177) = 1.67177$$

$$k_3 = f(0.3, 1.8107 + 0.1 \cdot 1.67177) = f(0.3, 1.977877) = 1.977877 - 0.3 = 1.677877$$

$$k_4 = f(0.4, 1.8107 + 0.2 \cdot 1.677877) = f(0.4, 2.1462754) = 1.7462754$$

$$y_2 = y_1 + h(k_1 + 2k_2 + 2k_3 + k_4)/6$$

$$y_2 = 1.8107 + 0.2(1.6107 + 2 \cdot 1.67177 + 2 \cdot 1.677877 + 1.7462754)/6 = 2.14590898$$

Аналогично находим $y_3 = 2.511053228172$

Таким образом, получили численное решение задачи Коши:

x_i	0	0.2	0.4	0.6
y_i	1.5	1.8107	2.14590898	2.511053228172

Графиком приближенного решения является ломаная, последовательно соединяющая точки (x_i, y_i) .

2.3. Решение краевых задач для линейных дифференциальных уравнений второго порядка методом конечных разностей

Линейное дифференциальное уравнение второго порядка имеет вид:

$$y'' + p(x)y' + q(x)y = f(x), \quad (3.1)$$

где $p(x), q(x)$ и $f(x)$ — некоторые непрерывные на $[a, b]$ функции. Краевая задача для линейного дифференциального уравнения состоит в нахождении его решения $y = y(x)$, удовлетворяющего двухточечным линейным краевым условиям

$$\begin{cases} \alpha_1 y(a) + \alpha_2 y'(a) = A, \\ \beta_1 y(b) + \beta_2 y'(b) = B, \end{cases} \quad (3.2)$$

где $\alpha_1, \alpha_2, \beta_1, \beta_2, A, B$ — постоянные и $|\alpha_1| + |\alpha_2| \neq 0$, $|\beta_1| + |\beta_2| \neq 0$.

При решении этой задачи методом конечных разностей отрезок $[a, b]$ разбивают на равные части с шагом h , где $h = \frac{b-a}{n}$. Точки разбиения имеют абсциссы

$$x_k = x_1 + (k-1) \cdot h, \quad k=1, 2, \dots, n+1, \quad x_1 = a, \quad x_{n+1} = b.$$

Значения в точках деления x_k искомой функции и её производных $y' = y'(x)$, $y'' = y''(x)$ обозначим соответственно через $y_k = y(x_k)$, $y'_k = y'(x_k)$, $y''_k = y''(x_k)$. Заменяя производные правыми одно-сторонними конечно-разностными отношениями для внутренних точек x_k отрезка $[a, b]$, приближенно будем иметь

$$\begin{aligned} y'_k &= \frac{y_{k+1} - y_k}{h}, \\ y''_k &= \frac{y_{k+2} - 2y_{k+1} + y_k}{h^2}. \end{aligned} \quad (3.3)$$

Для концевых точек $x_1 = a$ и $x_{n+1} = b$ полагаем

$$y'_1 = \frac{y_2 - y_1}{h} \quad \text{и} \quad y'_{n+1} = \frac{y_{n+1} - y_n}{h}. \quad (3.4)$$

Используя формулы (3.3), дифференциальное уравнение (3.1) при $x = x_k$ ($k = 2, 3, \dots, n$) приближенно можно заменить системой линейных уравнений

$$\frac{y_{k+2} - 2y_{k+1} + y_k}{h^2} + p(x_k) \frac{y_{k+1} - y_k}{h} + q(x_k) y_k = f(x_k), \quad k=1, 2, \dots, n-1.$$

В силу формул (3.4) краевые условия (3.2) дополнительно дают ещё два уравнения

$$\begin{aligned} \alpha_1 y_1 + \alpha_2 \frac{y_2 - y_1}{h} &= A, \\ \beta_1 y_{n+1} + \beta_2 \frac{y_{n+1} - y_n}{h} &= B. \end{aligned}$$

Таким образом получаем систему $n+1$ линейных уравнений с $n+1$ неизвестными $y_1, y_2, \dots, y_n, y_{n+1}$, представляющими собой значения искомой функции $y = y(x)$,

$$\left\{ \begin{aligned} \frac{y_{k+2} - 2y_{k+1} + y_k}{h^2} + p(x_k) \frac{y_{k+1} - y_k}{h} + q(x_k) y_k &= f(x_k), \\ \alpha_1 y_1 + \alpha_2 \frac{y_2 - y_1}{h} &= A, \\ \beta_1 y_{n+1} + \beta_2 \frac{y_{n+1} - y_n}{h} &= B. \end{aligned} \right.$$

Обозначим $p(x_k) = p_k$, $q(x_k) = q_k$, $f(x_k) = f_k$. Выполнив алгебраические преобразования над уравнениями, можно привести систему к следующему виду:

$$\begin{cases} (h^2 q_k - h p_k + 1) y_k + (h p_k - 2) y_{k+1} + y_{k+2} = h^2 f_k, \\ (\alpha_1 h - \alpha_2) y_1 + \alpha_2 y_2 = h A, \\ -\beta_2 y_n + (\beta_1 h + \beta_2) y_{n+1} = h B. \end{cases} \quad (3.5)$$

Решив систему, получим таблицу значений искомой функции $y(x)$.

ПРИМЕР 4. Найти решение уравнения $y'' + 2y' + \frac{y}{x} = 5$ на $[0,4; 0,7]$

($n=3$) с начальными условиями

$$y(0,4)=7, \quad y(0,7)-2y'(0,7)=3.$$

Решение:

Из условия задачи и (1)-(2) следует:

$$p(x)=2, q(x)=\frac{1}{x}, f(x)=5, \alpha_1=1, \alpha_2=0, \beta_1=1, \beta_2=-2, A=7, B=3, a=0.4, b=0.7.$$

Разобьём отрезок $[a, b]$ на равные части с шагом $h=0.1$, $n=3$. Точки разбиения имеют абсциссы $x_1=0.4, x_2=0.5, x_3=0.6, x_4=0.7$.

Построим систему (3.5) линейных алгебраических уравнений, где неизвестными являются y_1, \dots, y_4 .

$$\begin{cases} (h^2 q_k - h p_k + 1) y_k + (h p_k - 2) y_{k+1} + y_{k+2} = h^2 f_k, \\ (\alpha_1 h - \alpha_2) y_1 + \alpha_2 y_2 = h A, \\ -\beta_2 y_n + (\beta_1 h + \beta_2) y_{n+1} = h B. \end{cases}$$

Для коэффициентов основной матрицы системы для $n-1$ уравнений введем обозначения: $a_{k,k} = h^2 q_k - h p_k + 1$, $a_{k,k+1} = h p_k - 2$, $a_{k,k+2} = 1$, $k=1,2$. Для коэффициентов последних двух уравнений (n -го и $(n+1)$ -го) введем обозначения: $a_{n,1} = \alpha_1 h - \alpha_2$, $a_{n,2} = \alpha_2$, $a_{n+1,n} = -\beta_2$, $a_{n+1,n+1} = \beta_1 h + \beta_2$.

Для матрицы свободных членов введем обозначения:

$$b_k = h^2 f_k, \quad b_n = h A, \quad b_{n+1} = h B, \quad k=1,2, \quad n=3, \quad h=0.1$$

Остальные коэффициенты системы равны нулю.

Составим развернутую систему (3.5) для нашей задачи:

$$\begin{aligned} (h^2 q(0.4) - h p(0.4) + 1) y_1 + (h p(0.4) - 2) y_2 + y_3 &= h^2 f(0.4) \\ (h^2 q(0.5) - h p(0.5) + 1) y_2 + (h p(0.5) - 2) y_3 + y_4 &= h^2 f(0.5) \\ (\alpha_1 h - \alpha_2) y_1 + \alpha_2 y_2 &= h A \\ -\beta_2 y_3 + (\beta_1 h + \beta_2) y_4 &= h B \end{aligned}$$

Представим систему в матричном виде:

$$\begin{pmatrix} (h^2 q(0.4) - h p(0.4) + 1) & (h p(0.4) - 2) & 1 & 0 \\ 0 & (h^2 q(0.5) - h p(0.5) + 1) & (h p(0.5) - 2) & 1 \\ (\alpha_1 h - \alpha_2) & \alpha_2 & 0 & 0 \\ 0 & 0 & -\beta_2 & (\beta_1 h + \beta_2) \end{pmatrix} = \begin{pmatrix} h^2 f(0.4) \\ h^2 f(0.5) \\ h A \\ h B \end{pmatrix}$$

Подставим значения переменных в систему:

$$\begin{pmatrix} (0.01 \frac{1}{0.4} - 0.1 \cdot 2 + 1) & (0.1 \cdot 2 - 2) & 1 & 0 \\ 0 & (0.01 \frac{1}{0.5} - 0.1 \cdot 2 + 1) & (0.1 \cdot 2 - 2) & 1 \\ (0.1 - 0) & 0 & 0 & 0 \\ 0 & 0 & 2 & (0.1 - 2) \end{pmatrix} = \begin{pmatrix} 0.01 \cdot 5 \\ 0.01 \cdot 5 \\ 0.1 \cdot 7 \\ 0.1 \cdot 3 \end{pmatrix}$$

После упрощения получим:

$$\begin{pmatrix} 0.825 & -1.8 & 1 & 0 \\ 0 & 0.82 & -1.8 & 1 \\ 0.1 & 0 & 0 & 0 \\ 0 & 0 & 2 & -1.9 \end{pmatrix} = \begin{pmatrix} 0.05 \\ 0.05 \\ 0.7 \\ 0.3 \end{pmatrix}$$

Решая полученную систему, найдем приближенное решение дифференциального уравнения: $(0.4, 7)$, $(0.5, 7.75)$, $(0.6, 8.22)$, $(0.7, 8.5)$.

2.4. Метод сеток для решения дифференциальных уравнений в частных производных

Для решения дифференциального уравнения методом конечных разностей (сеток) сначала область, на которой ищется решение, заменяется дискретным множеством точек (разностной сеткой). В этом методе, как правило, используются регулярные сетки, шаг которых либо постоянен, либо меняется по несложному закону.

Пусть в качестве области изменения функции задан прямоугольник. Оси x и y разбиваются на отрезки, которые являются шагами сетки по соответствующим направлениям. Через точки деления проводятся прямые, параллельные осям координат. Совокупность точек пересечения (узлов) этих прямых и образует сетку в заданной двумерной области. Узлы, расстояние между которыми равно шагу сетки по одной из осей, называются *соседними*.

Способ построения сетки не меняется и в том случае, если задана область произвольной формы. Узлы сетки, попавшие внутрь области, называются *внутренними узлами*. Точки пересечения прямых, образующих сетку, с границей области называются *граничными узлами*. Для двумерной области произвольной формы сетка в общем случае всегда является нерегулярной, причем особенности геометрии учитываются только в околограничных точках (рис.1).

На рис.1 внутренние точки области обозначены кружками, граничные — крестиками. Решение уравнения в частных производных ищется во внутренних точках области, в граничных точках области оно задается граничными условиями.

Прежде чем приступить к решению дифференциального уравнения, оно само и граничные условия заменяются разностными аналогами. Вспомним ряд Тейлора для функции $f(x)$:

$$f(x+h) = f(x) + f'(x)h + \frac{f''(x)}{2!}h^2 + \dots$$

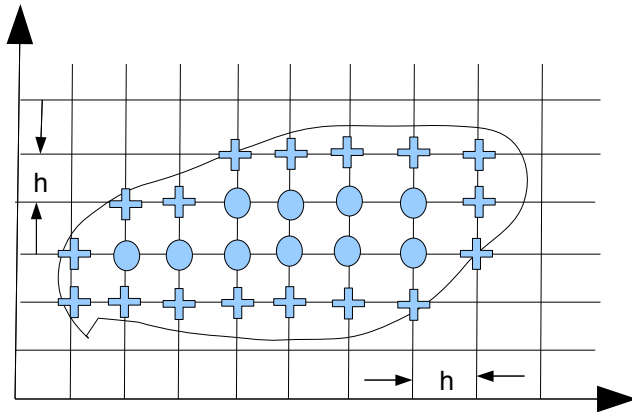


Рис.1. Сетка для произвольной области

Если оборвать этот ряд на втором члене, то получим

$$f(x+h) = f(x) + f'(x)h, \text{ или } f'(x) \approx \frac{f(x+h) - f(x)}{h}.$$

Выражение, стоящее в правой части, называется правой разностной производной. Она аппроксимирует первую производную $f'(x)$ в точке x .

В разложении Тейлора функции $f(x)$ можно заменить h на $-h$ и получить левую разностную производную

$$f'(x) \approx \frac{f(x) - f(x-h)}{h}.$$

Вычитая $f(x-h) \approx f(x) - f'(x)h$ из $f(x+h) \approx f(x) + f'(x)h$, получаем центральную разностную производную

$$f'(x) \approx \frac{1}{2h} [f(x+h) - f(x-h)].$$

Если в ряде Тейлора оставить третий член ряда, то можно получить центральную разностную производную для аппроксимации $f''(x)$:

$$f''(x) \approx \frac{1}{h^2} [f(x+h) - 2f(x) + f(x-h)].$$

Если исходить из разложения Тейлора функции двух переменных

$$u(x+h, y) = u(x, y) + u_x(x, y)h + u_{xx}(x, y)\frac{h^2}{2!} + \dots,$$

$$u(x-h, y) = u(x, y) - u_x(x, y)h + u_{xx}(x, y)\frac{h^2}{2!} - \dots,$$

то можно получить следующие аппроксимации частных производных:

$$u_x(x, y) \approx \frac{u(x+h, y) - u(x, y)}{h},$$

$$u_{xx}(x, y) \approx \frac{1}{h^2}[u(x+h, y) - 2u(x, y) + u(x-h, y)],$$

$$u_y(x, y) \approx \frac{u(x, y+k) - u(x, y)}{k},$$

$$u_{yy}(x, y) \approx \frac{1}{k^2}[u(x, y+k) - 2u(x, y) + u(x, y-k)].$$

Разностные операторы, соответствующие дифференциальному уравнению, записываются во внутренних узлах сетки. Разностные операторы, соответствующие граничным условиям, записываются в граничных узлах сетки. В результате получаем систему алгебраических уравнений, число которых пропорционально числу внутренних узлов сеточной области. Для получения численного решения требуется решить эту систему уравнений.

Удобно, если предполагается использование ЭВМ для реализации вычислений, перейти к следующим обозначениям:

$$u(x, y) = u_{ij}, \quad u(x, y+k) = u_{i+1, j}, \quad u(x, y-k) = u_{i-1, j},$$

$$u(x-h, y) = u_{i, j-1}, \quad u(x+h, y) = u_{i, j+1},$$

$$u_x(x, y) = \frac{1}{2h}[u_{i, j+1} - u_{i, j-1}], \quad u_y(x, y) = \frac{1}{2k}[u_{i+1, j} - u_{i-1, j}],$$

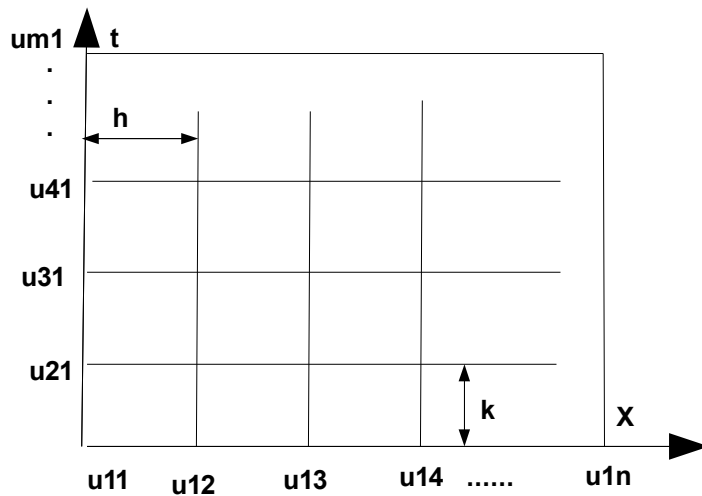
$$u_{xx}(x, y) = \frac{1}{h^2}[u_{i, j+1} - 2u_{i, j} + u_{i, j-1}], \quad u_{yy}(x, y) = \frac{1}{k^2}[u_{i+1, j} - 2u_{i, j} + u_{i-1, j}]$$

ПРИМЕР 1. Рассмотрим задачу теплопроводности в стержне, начальная температура которого равна нулю. Пусть температура левого конца фиксирована, а на правом конце происходит теплообмен с окружающей средой, так что тепловой поток пропорционален разности температур конца стержня и среды. Пусть температура среды определяется функцией $g(t)$. Другими словами, решим задачу:

$$\begin{cases} u_t = u_{xx}, & 0 < x < 1, 0 < t < \infty, \\ u(0, t) = 1, \\ u_x(1, t) = -[u(1, t) - g(t)], & 0 < t < \infty, \\ u(x, 0) = 0, & 0 \leq x \leq 1. \end{cases}$$

Решение:

Построим в плоскости прямоугольную сетку (рис.2), узлы которой определяются формулами: $x_j = jh$, $j = 0, 1, 2, \dots, n$, $t_i = ik$, $i = 0, 1, 2, \dots, m$.



Значения u_{ij} на левой и нижней сторонах сетки известны из граничных и начальных условий. Наша задача состоит в отыскании остальных значений u_{ij} .

Для решения задачи заменим частные производные в уравнении теплопроводности их конечно-разностными аппроксимациями

$$u_t(x, y) \approx \frac{u(x, t+k) - u(x, t)}{k} = \frac{1}{k} [u_{i+1, j} - u_{i, j}],$$

$$u_{xx}(x, t) \approx \frac{1}{h^2} [u(x+h, t) - 2u(x, t) + u(x-h, t)] = \frac{1}{h^2} [u_{i, j+1} - 2u_{i, j} + u_{i, j-1}].$$

Подставим эти выражения в наше уравнение $u_t = u_{xx}$ и разрешим получившееся уравнение относительно значений функции на верхнем временном слое. Имеем:

$$\frac{1}{k} [u_{i+1, j} - u_{i, j}] = \frac{1}{h^2} [u_{i, j+1} - 2u_{i, j} + u_{i, j-1}].$$

Отсюда

$$u_{i+1, j} = \frac{k}{h^2} [u_{i, j+1} - 2u_{i, j} + u_{i, j-1}] + u_{i, j}. \quad (1)$$

Полученная формула выражает решение в данный момент времени через решение в предыдущий момент времени (индекс i относится к временной переменной).

Аппроксимируем производную в граничном условии $u_x(1, t) = -[u(1, t) - g(t)]$ на правом конце, заменив $u_x(1, t)$ левой разностной

производной, поскольку правая разностная производная требует значений функции за пределами сетки:

$$\frac{1}{h}[u_{i,n}-u_{i,n-1}]=-[u_{i,n}-g_i] \quad , \quad g_i=g(ik) \quad .$$

Отсюда находим:

$$u_{i,n}=\frac{u_{i,n-1}-hg_i}{1+h} \quad . \quad (2)$$

Таким образом, для решения поставленной задачи мы будем использовать *явную схему бегущего счета*: заменяя частные производные по времени и пространственной переменной конечно-разностными производными, мы получаем явные выражения для $u_{i,j}$ через значения функции u в предыдущие моменты времени.

Шаг 1. Находим решение на сеточном слое $t=\Delta t$, используя явную формулу (1).

Шаг 2. Величину $u_{2,n}$ находим по формуле (2).

Выполнив шаги 1-2, получаем решение для $t=\Delta t$. Повторив шаги 1-2, получаем решение при $t=2\Delta t$ и т. д.

Недостаток явной схемы: если шаг по времени оказывается достаточно большим по сравнению с шагом по x , погрешности округления могут стать настолько большими, что полученное решение теряет смысл. Отношение шагов по t и x зависит от уравнения и граничных условий. Для применимости явной схемы должно выполняться условие $kh^2 \leq 0,5$. В противном случае метод будет численно не устойчив.

ПРИМЕР 2. Решим задачу:

$$\begin{cases} u_{xx}=\frac{1}{a^2}u_{tt}, & 0 \leq x \leq m, \quad 0 \leq t \leq n \\ u(x,0)=\sin\frac{\pi x}{50}, & u_t(x,0)=0 \\ u(0,t)=u(m,t)=0 \end{cases} \quad .$$

Решение:

Положим $m=10$, $n=20$, $h=1$ — шаг изменения пространственной переменной. Заменяем частные производные в волновом уравнении их конечно-разностными аппроксимациями

$$u_{xx}(x,y)=\frac{1}{h^2}[u_{i,j+1}-2u_{i,j}+u_{i,j-1}] \quad , \quad u_{yy}(x,y)=\frac{1}{k^2}[u_{i+1,j}-2u_{i,j}+u_{i-1,j}] \quad .$$

Получим: $\frac{1}{h^2}[u_{i,j+1}-2u_{i,j}+u_{i,j-1}]=\frac{1}{a^2} \frac{1}{k^2}[u_{i+1,j}-2u_{i,j}+u_{i-1,j}]$. Отсюда

$$u_{i+1,j}=\frac{a^2 k^2}{h^2}[u_{i,j+1}-2u_{i,j}+u_{i,j-1}]+2u_{i,j}-u_{i-1,j} \quad .$$

Получили явную разностную схему, которая будет устойчивой, если $\frac{a^2 k^2}{h^2} \leq 0,5$. Отсюда $k \leq \sqrt{0,5} \frac{h}{a}$. Выберем $a=1, k=0,1$.

Построим алгоритм решения задачи:

1 шаг. Вводим сетку: $m=100$, $n=200$, $h=1$. Создаем нулевой массив значений $U(i, j)$ размера $m \times n$.

2 шаг. Задаем значения $a=1, k=0,1$.

3 шаг. Заполняем первую и вторую строки массива U начальными условиями $u(x, 0) = \sin \frac{\pi x}{50}$, $u_t(x, 0) = 0$ (нулевой начальной скорости соответствует совпадение значений (смещений) в первом и втором столбцах).

4 шаг. Заполняем первый и последний столбец массива U граничными условиями $u(0, t) = u(m, t) = 0$ (на концах струны смещение равно нулю в любой момент времени).

5 шаг. Находим решение, используя разностную схему

$$u_{i+1, j} = \frac{a^2 k^2}{h^2} [u_{i, j+1} - 2u_{i, j} + u_{i, j-1}] + 2u_{i, j} - u_{i-1, j}.$$

Также можно использовать и неявные разностные схемы. В этом случае частные производные заменяются конечно-разностными аппроксимациями, но $u_{i+1, j}$ не выражаются в явном виде через значения на предыдущих слоях. Для определения $u_{i+1, j}$ на каждом временном шаге необходимо решать систему уравнений. При использовании неявных схем можно вести вычисления с достаточно большим шагом.

Преимущество неявных схем перед явными в том, что в неявных схемах шаг сетки можно сделать достаточно большим, не опасаясь, что ошибки округления «разрушат» решение.

3.1. Встроенные функции для нахождения решений дифференциальных уравнений

Перейдем к более подробному рассмотрению функций и команд, с помощью которых можно находить решения дифференциальных уравнений.

Система компьютерной математики Maxima может с успехом решать следующие виды дифференциальных уравнений первого порядка: с разделяющимися переменными, линейные, нелинейные уравнения, однородные, неоднородные. Виды уравнений второго порядка: с постоянными коэффициентами, линейные однородные с непостоянными коэффициентами, которые могут быть преобразованы к уравнению с постоянным коэффициентом, уравнение Эйлера, уравнения, разрешимые методом вариации постоянных, и уравнения, которые допускают понижение порядка.

Рассмотрим команды системы Maxima для нахождения решений дифференциальных уравнений и их систем в символьном виде:

desolve (eqn, x) — ищет частные решения линейных дифференциальных уравнений первого и второго порядков

desolve ([eqn_1, ..., eqn_n], [x_1, ..., x_n]) (сокращенно от слов «differential equation solve») — ищет частные решения систем линейных дифференциальных уравнений первого и второго порядков.

Эта функция принимает два аргумента, первый из которых – уравнение либо список уравнений, а второй – соответственно одна переменная или список переменных.

Если не заданы значения функций и/или их производных в нуле, то в найденном решении они просто отображаются в виде $f(0)$ или $\frac{d}{dx} f(x)|_{x=0}$. Задать эти значения позволяет функция **atvalue(выражение, переменная=точка, значение)**, то есть, в данном случае **atvalue(f(x),x=0, значение)** или **atvalue('diff(f(x),x)=0, значение)**. Производные в уравнениях и системах, решаемых с помощью этой функции, должны быть записаны в виде 'diff(f(x),x).

Если функция **desolve** не может найти решения, то она возвращает значение **false**.

ode2(eqn, dvar, ivar) — предназначена для решения обыкновенных линейных дифференциальных уравнений первого и второго порядка. Решение ищется в общем виде. Принимает три аргумента: eqn – само дифференциальное уравнение, зависимая переменная dvar, и независимая переменная ivar. Данная функция может возвращать решение в явном и неявном виде.

Здесь уже зависимая переменная указывается в списке параметров функции явно, поэтому обозначения вида $y(x)$ не нужны. Функция и переменная обозначаются одиночными буквами.

Как и для обычных уравнений и систем, мы можем проверить решение с помощью подстановки, но только надо еще задать принудительные вычисления производных, так как в уравнениях они фигурируют в несовершенной форме.

Произвольная константа для уравнений первого порядка обозначается через `%c`. В уравнениях второго порядка константы обозначаются как `%k1` и `%k2`.

Если функция `ode2` не может получить решение по какой-либо причине, то она возвращает значение `false`, при этом возможен вывод сообщения об ошибке.

В дополнение к функции `ode2` существуют три функции для поиска частных решений на основе полученных общих. То есть, эти функции, получая конкретные условия относительно значения функции-решения в заданной точке, находят исходя из этих значений соответствующие им величины интегральных констант. Перечислим их.

ic1(solution, xval, yval) — предназначена для обработки решения дифференциального уравнения первого порядка с начальными условиями. Принимает три аргумента: первый – само решение, в том виде, в котором его находит функция `ode2`, второй – начальное значение независимой переменной в форме $x = x_0$, третий – значение функции `yval` при указанном значении `xval` в виде $y = y_0$. Возвращает частное решение, проходящее через точку с заданными координатами (`xval`, `yval`).

ic2(solution, xval, yval, dval) — предназначена для обработки решения дифференциального уравнения второго порядка с начальными условиями. Принимает четыре аргумента: `solution` - общее решение уравнения, найденное с помощью функции `ode2`, `xval` — начальное значение независимой переменной в форме $x = x_0$, `yval` — начальное значение зависимой переменной в форме $y = y_0$, и `dval` — начальное значение для первой производной зависимой переменной относительно независимой переменной, в форме $(y, x) = dy_0$.

bc2(solution, xval1, yval1, xval2, yval2) — предназначена для нахождения решения граничной задачи для дифференциального уравнения второго порядка. Здесь `solution` — общее решение уравнения, найденное с помощью функции `ode2`, `xval1` — значение независимой переменной в первой граничной точке, задается в виде $x=x_1$, `yval1` — соответственно значение зависимой переменной в точке x , задается также в виде $y=y_1$, `xval2`, `yval2` – вторая граничная точка, задается в той же форме, что и первая точка.

Функция `plotdf` позволяет строить траектории и поле направлений дифференциального уравнения первого порядка или автономной системы двух обыкновенных дифференциальных уравнений первого порядка. Эта функция входит в дополнительный пакет, поэтому для ее использования следует первоначально его загрузить командой `load(plotdf)`. Функция `Plotdf` требует установки модуля `Openmath`, который входит в пакет `xMaxima`.

Для построения поля направлений дифференциального уравнения первого порядка оно должно быть записано в виде $\frac{dy}{dx} = f(x, y)$. В случае авто-

номной системы она должна быть записана в виде:

$$\begin{cases} \frac{dx}{dt} = G(x, y) \\ \frac{dy}{dt} = F(x, y) \end{cases}$$

Синтаксис функции:

`plotdf(dydx, ...options...)`

`plotdf(dvdu, [u,v], ...options...)`

`plotdf([dxdt,dydt], ...options...)`

`plotdf([dudt,dvdt], [u,v], ...options...)`

Здесь `dydx`, `dxdt` и `dydt` — выражения, которые зависят от x и y , `dvdu`, `dudt` и `dvdt` — выражения, которые зависят от u и v . В дополнение к этим двум переменным, могут использоваться опции. Они могут выбираться с помощью пунктов меню, а также задаваться с помощью специальных команд.

В системе Maxima есть дополнительный пакет `contrib_ode`, который позволяет находить решение нелинейных дифференциальных уравнений первого и второго порядка.

Для того чтобы можно было использовать команды этого пакета, его необходимо сначала подключить с помощью команды `load(contrib_ode)`.

Заметим, что возможности пакета `contrib_ode` расширяются с каждой новой версией системы **Maxima**.

Синтаксис команды: **`contrib_ode (eqn, y, x)`**

Для корректного выполнения поиска решения с использованием команды `contrib_ode` ей нужно передать три аргумента:

1. Само дифференциальное уравнение, которое требуется решить (в правой части дифференциального уравнения должен стоять ноль)
2. Имя зависимой переменной — искомой функции
3. Имя независимой переменной — аргумента функции.

Если дифференциальное уравнение система может решить, то она возвращает решение или список решений, причем каждое решение может быть представлено в одном из следующих видов:

1. Искомая функция выражена в явном виде.

2. Искомая функция выражена в неявном виде.
3. Решение получено в параметрическом виде с параметром t .
4. Дифференциальное уравнение преобразуется к другому дифференциальному уравнению относительно функции u .

В случае, если решение не может быть получено, команда `contrib_ode` возвращает сообщение об ошибке и значение «false».

3.2. Решение дифференциальных уравнений и их систем в символьном виде

Рассмотрим ряд примеров на нахождение решений дифференциальных уравнений с помощью встроенных функций системы.

Пример 1. Найти частное решение дифференциального уравнения с разделенными переменными $\sin x dx - dy = 0$ при начальном условии $y(0) = 4$.

Решение. Для решения уравнения в системе Maxima выделим производную функции y явно, поделив обе части уравнения на dx : $\sin x - y' = 0$.

Зададим уравнение в строке ввода и обозначим его `eqn`.

```
(%i7) eqn:sin(x)'diff(y(x),x);
```

```
(%o7) sin(x) =  $\frac{d}{dx}y(x)$ 
```

Зададим начальное условие $y(0) = 4$:

```
(%i8) atvalue(y(x),x=0,4);
```

```
(%o8) 4
```

Теперь задаем команду для нахождения искомого частного решения:

```
(%i9) desolve(eqn,y(x));
```

```
(%o9) y(x) = 5 - cos(x)
```

В том случае, если не задать предварительно начальные условия, то система выдаст ответ в виде:

```
(%i2) desolve(eqn,y(x));
```

```
(%o2) y(x) = -cos(x) + y(0) + 1
```

Как видно, если мы подставим условие $y(0) = 4$, то решения совпадут.

Пример 2. Найти частное решение линейного дифференциального уравнения первого порядка $5y' - 8y = x$ при условии $y(0) = 1$.

Решение. Зададим уравнение под именем `eqn` и начальное условие (заметим, что знак умножения нужно прописывать каждый раз явно):

```
(%i42) eqn:5*'diff(y(x),x)-8*y(x)=x;atvalue(y(x),x=0,1);
```

```
(%o42) 5\left(\frac{d}{dx}y(x)\right)-8y(x)=x
```

```
(%o43) 1
```

Теперь воспользуемся командой `desolve` для поиска частного решения:

```
(%i44) desolve(eqn,y(x));
```

```
(%o44) y(x)=\frac{69e^{\frac{8x}{5}}}{64}-\frac{x}{8}-\frac{5}{64}
```

Пример 3. Найти частное решение системы линейных дифференциальных уравнений $\begin{cases} f'(x) = g'(x) + \sin x \\ g'(x) = 5f'(x) - \cos x \end{cases}$ при условиях $f(0)=1, g(0)=7$.

Решение. Зададим каждое из уравнений системы.

```
(%i51) eq1:'diff(f(x),x)='diff(g(x),x)+sin(x);
```

```
(%o51) \frac{d}{dx}f(x)=\frac{d}{dx}g(x)+sin(x)
```

```
(%i52) eq2:'diff(g(x),x)=5*'diff(f(x),x)-cos(x);
```

```
(%o52) \frac{d}{dx}g(x)=5\left(\frac{d}{dx}f(x)\right)-cos(x)
```

Зададим начальные условия:

```
(%i56) atvalue(f(x),x=0,1);atvalue(g(x),x=0,7);
```

```
(%o56) 1
```

```
(%o57) 7
```

Ищем частное решение:

```
(%i58) desolve([eq1,eq2],[f(x),g(x)]);
```

```
(%o58) [f(x)=\frac{sin(x)}{4}+\frac{cos(x)}{4}+\frac{3}{4},g(x)=\frac{sin(x)}{4}+\frac{5cos(x)}{4}+\frac{23}{4}]
```

Для нахождения общего решения дифференциального уравнения в системе Maxima используется команда `ode2`. Рассмотрим ряд примеров.

Пример 4. Найти общее решение линейного дифференциального уравнения первого порядка $x \frac{dy}{dx} + 3y = x \sin x$.

1 способ. Зададим уравнение в ячейке ввода.

```
(%i1) eq1:x*'diff(y(x),x)+3*y(x)=x*sin(x);
```

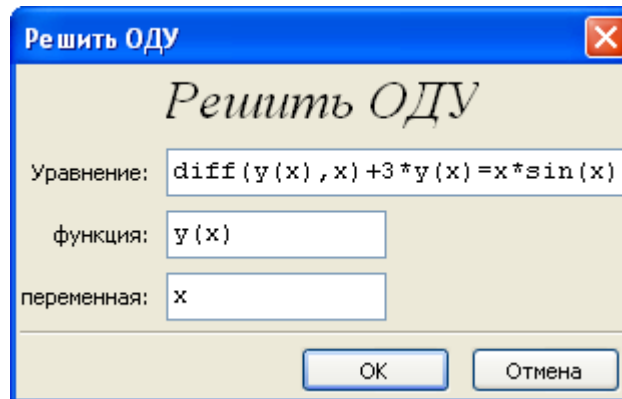
```
(%o1) x  $\left( \frac{d}{dx} y(x) \right) + 3 y(x) = x \sin(x)$ 
```

Теперь решим уравнение. Для этого в ячейке ввода задаем команду `ode2(eq1, y(x), x)` и оцениваем. В результате получим:

```
(%i2) ode2(eq1, y(x), x);
```

```
(%o2) y(x) =  $\frac{(3x^2 - 6)\sin(x) + (6x - x^3)\cos(x) + c}{x^3}$ 
```

2 способ. В нижней части окна программы на панели инструментов нажимаем на кнопке *Решить ОДУ*. Появляется диалоговое окно, в котором надо задать само дифференциальное уравнение, имя искомой функции и имя независимой переменной:



После нажатия на кнопке *ОК* получаем решение:

```
(%i4) ode2(x*'diff(y(x),x)+3*y(x)=x*sin(x), y(x), x);
```

```
(%o4) y(x) =  $\frac{(3x^2 - 6)\sin(x) + (6x - x^3)\cos(x) + c}{x^3}$ 
```

Как видим, решения совпали.

В некоторых случаях система может выдавать решения в неявном виде — в виде общего интеграла.

Пример 5. Найти общее решение дифференциального уравнения $\sin x = -\frac{1}{\sqrt{y}} y'$.

Решение. Задаем дифференциальное уравнение

```
(%i31) eqn:sin(x)=-1/sqrt(y(x))*'diff(y(x),x);
```

```
(%o31) sin(x) =  $-\frac{\frac{d}{dx} y(x)}{\sqrt{y(x)}}$ 
```

Решаем уравнение:

```
(%i37) ode2(eqn, y(x), x);
```

```
(%o37) -2*sqrt(y(x)) = %c - cos(x)
```

Пример 6. Найти общее решение уравнения Бернулли $y' + 2e^x y = 2e^x \sqrt{y}$.
Решение. Задаем уравнение и обозначаем его eqn.

```
(%i58) eqn: 'diff(y, x) + 2*exp(x)*y = 2*exp(x)*sqrt(y);
```

```
(%o58)  $\frac{d}{dx}y + 2e^x y = 2e^x \sqrt{y}$ 
```

Находим решение:

```
(%i59) ode2(eqn, y, x);
```

```
(%o59) -log(sqrt(y) - 1) = %e^x + %c
```

В том случае, если система не может решить дифференциальное уравнение, она возвращает значение false или выдает сообщение об ошибке. Рассмотрим пример.

Пример 7. Найти общее решение уравнения Клеро $y = x y' - y'^2$.

Решение. Зададим уравнение и воспользуемся командой ode2 для его решения.

```
(%i38) eqn: y = x*'diff(y, x) - ('diff(y, x))^2;
```

```
(%o38)  $y = x \left( \frac{d}{dx}y \right) - \left( \frac{d}{dx}y \right)^2$ 
```

```
(%i39) ode2(%, y, x);
```

```
(%t39)  $y = x \left( \frac{d}{dx}y \right) - \left( \frac{d}{dx}y \right)^2$ 
```

Maxima encountered a Lisp error:

Error in SYMBOL-NAME [or a callee]: "first or
Automatically continuing.

To reenale the Lisp debugger set *debugger-ho

Рассмотрим ряд примеров на нахождение решений обыкновенных дифференциальных уравнений первого порядка с начальными условиями. Для этого используется команда ic1.

Пример 8. Найти общее решение дифференциального уравнения $2x^2 y' = x^2 + y^2$ и найти его частное решение, удовлетворяющее начальному условию $y(1) = 0$.

Решение. Найдем общее решение дифференциального уравнения и обозначим его rez.

```
(%i85) rez:ode2(2*x^2*'diff(y,x)=x^2+y^2,y,x);
```

```
(%o85) %c x = %e  $-\frac{2x}{y-x}$ 
```

Теперь воспользуемся условием $y(1)=0$.

```
(%i86) ic1(rez,x=1,y=0);
```

```
(%o86) %e2 x = %e  $-\frac{2x}{y-x}$ 
```

Как видно, система нашла решение в неявном виде. Разрешим полученное уравнение относительно y и тем самым найдем решение в явном виде.

```
(%i87) solve(%,y);
```

```
(%o87) [ y =  $\frac{x \log(x)}{\log(x)+2}$  ]
```

Пример 9. Найти частное решение дифференциального уравнения $(x+xy)dy+(y+xy)dx=0$ при $y(1)=1$.

Решение. Аналогично предыдущему примеру, ищем общее решение дифференциального уравнения.

```
(%i103) rez:ode2((x+x*y)*'diff(y,x)=x*y-y,y,x);
```

```
(%o103) log(y)+y = -log(x)+x + %c
```

Далее находим частное решение.

```
(%i104) ic1(rez,x=1,y=1);
```

```
(%o104) log(y)+y = x - log(x)
```

Пример 10. Решить задачу Коши для дифференциального уравнения $xy' = y(1 + \ln y - \ln x)$, $y(1)=e^2$.

Решение. Находим решение задачи Коши с помощью команды `ic1`.

```
(%i52) rez:ode2(x*'diff(y,x)=y*(1+log(y)-log(x)),y,x);
ic1(rez,x=1,y=%e^2);
```

```
(%o52) x = %c(log(y)-log(x))
```

```
(%o53) x =  $\frac{\log(y)-\log(x)}{2}$ 
```

Для упрощения полученного результата разрешим его относительно y .

```
(%i54) solve(%,y);
```

```
(%o54) [ y = x %e2x ]
```

Для нахождения решений дифференциальных уравнений второго порядка используются те же команды, что и для уравнений первого порядка. Однако при решении начальной и граничной задач используются другие команды. Рассмотрим на примерах.

Пример 11. Найти общее решение дифференциального уравнения второго порядка $15y' - 26y = y''$.

Решение. Зададим исходное уравнение:

```
(%i1) eqn: 15*'diff(y(x), x) - 26*y(x) = 'diff(y(x), x, 2);
```

```
(%o1) 15  $\left(\frac{d}{dx}y(x)\right) - 26y(x) = \frac{d^2}{dx^2}y(x)$ 
```

Для поиска общего решения используем команду ode2.

```
(%i2) ode2(eqn, y(x), x);
```

```
(%o2) y(x) = %k1 %e13x + %k2 %e2x
```

Выполним проверку найденного решения. Для этого зададим команду в следующем виде:

```
(%i16) eqn, %, diff;
```

```
(%o16) 15  $\left(13 %k1 %e^{13x} + 2 %k2 %e^{2x}\right) - 26 \left(%k1 %e^{13x} + %k2 %e^{2x}\right) =$   

 $169 %k1 %e^{13x} + 4 %k2 %e^{2x}$ 
```

Из полученного ответа не видно, правильно ли было получено решение. Упростим полученный результат.

```
(%i17) expand(%);
```

```
(%o17) 169 %k1 %e13x + 4 %k2 %e2x = 169 %k1 %e13x + 4 %k2 %e2x
```

Отсюда видно, что левая часть уравнения eqn равна его правой части, следовательно, решение было найдено правильно.

Пример 12. Найти общее решение дифференциального уравнения второго порядка $y'' + 6y' + 9y = 0$.

Решение. Зададим исходное уравнение:

```
(%i5) eqn: 'diff(y(x), x, 2) + 6*'diff(y(x), x) + 9*y(x) = 0;
```

```
(%o5)  $\frac{d^2}{dx^2}y(x) + 6\left(\frac{d}{dx}y(x)\right) + 9y(x) = 0$ 
```

Находим его решение.

```
(%i6) ode2(eqn, y(x), x);
(%o6) y(x)=(%k2 x + %k1)%e-3 x
```

Пример 13. Найти общее решение дифференциального уравнения второго порядка $y'' - 2y' + 10y = 0$.

Решение. Зададим исходное уравнение:

```
(%i7) eqn: 'diff(y(x), x, 2) - 2*'diff(y(x), x) + 10*y(x) = 0;
(%o7)  $\frac{d^2}{dx^2}y(x) - 2\left(\frac{d}{dx}y(x)\right) + 10y(x) = 0$ 
```

Находим его решение.

```
(%i8) ode2(eqn, y(x), x);
(%o8) y(x)=%ex(%k1 sin(3 x) + %k2 cos(3 x))
```

Пример 14. Решить задачу Коши для дифференциального уравнения $y'' - 7y' + 6y = (x-2)e^x$, $y(0)=1$, $y'(0)=3$.

Решение. Зададим уравнение и обозначим его eqn.

```
(%i9) eqn: 'diff(y, x, 2) - 7*'diff(y, x) + 6*y = (x-2)*exp(x);
(%o9)  $\frac{d^2}{dx^2}y - 7\left(\frac{d}{dx}y\right) + 6y = (x-2)e^x$ 
```

Воспользуемся командой ic2 для нахождения решения начальной задачи для дифференциального уравнения второго порядка.

```
(%i10) ic2(ode2(eqn, y, x), x=0, y=1, 'diff(y, x)=3);
(%o10)  $y = \frac{41 e^{6x}}{125} - \frac{(25x^2 - 90x - 18)e^x}{250} + \frac{3e^x}{5}$ 
```

Попробуем упростить полученное решение, применив команду раскрытия выражения expand.

```
(%i11) expand(%);
(%o11)  $y = \frac{41 e^{6x}}{125} - \frac{x^2 e^x}{10} + \frac{9x e^x}{25} + \frac{84 e^x}{125}$ 
```

Выполним проверку найденного решения.


```
(%i12) eqn, %, diff;
```

$$\begin{aligned}
 (\%o12) \quad & -7 \left(\frac{246 e^{6x}}{125} - \frac{x^2 e^x}{10} + \frac{4 x e^x}{25} + \frac{129 e^x}{125} \right) + \frac{1476 e^{6x}}{125} + 6 \\
 & \left(\frac{41 e^{6x}}{125} - \frac{x^2 e^x}{10} + \frac{9 x e^x}{25} + \frac{84 e^x}{125} \right) - \frac{x^2 e^x}{10} - \frac{x e^x}{25} + \frac{149 e^x}{125} = (x - 2) e^x
 \end{aligned}$$

Упростим результат после подстановки решения в исходное уравнение.

```
(%i13) expand(%);
```

$$(\%o13) \quad x e^x - 2 e^x = x e^x - 2 e^x$$

Таким образом, решение найдено правильно.

Пример 15. Найти решение уравнения $y'' + y = x$, удовлетворяющее краевым условиям $y(0) = 0$, $y(4) = 1$.

Решение. Зададим исходное уравнение.

```
(%i29) eqn: 'diff(y, x, 2) + y = x;
```

$$(\%o29) \quad \frac{d^2}{dx^2} y + y = x$$

Найдем общее решение уравнения.

```
(%i35) ode2(eqn, y, x);
```

$$(\%o35) \quad y = \%k1 \sin(x) + \%k2 \cos(x) + x$$

Воспользуемся командой bc2 для решения краевой задачи.

```
(%i1) load('contrib_ode) §
```

```
(%i2) eqn: 'diff(y, x)^2 - x*y*'diff(y, x) = 0;
```

$$(\%o2) \quad \left(\frac{d}{dx} y \right)^2 - x y \left(\frac{d}{dx} y \right) = 0$$

```
(%i36) bc2(%, x=0, y=0, x=4, y=1);
```

$$(\%o36) \quad y = x - \frac{3 \sin(x)}{\sin(4)}$$

Пример 16. Решить дифференциальное уравнение: $y'^2 - x y y' = 0$.

Подключим пакет contrib_ode и зададим дифференциальное уравнение eqn. Найдем решение дифференциального уравнения и запоемнм его под именем ans. Убедимся в правильности найденного решения. Для этого воспользуемся командой ode_check и выполним проверку найденных решений.

```
(%i3) ans:contrib_ode(eqn,y,x);
(%t3)  $\left(\frac{d}{dx}y\right)^2 - xy\left(\frac{d}{dx}y\right) = 0$ 
first order equation not linear in y'
(%o3) [ y = %c , y = %c %e $\frac{x^2}{2}$  ]
(%i4) ode_check(eqn,ans[1]);
(%o4) 0
(%i5) ode_check(eqn,ans[2]);
(%o5) 0
```

Пример 17. Решить дифференциальное уравнение: $y'^2 - xy' - y = 0$.

```
(%i1) load('contrib_ode)$
(%i2) eqn:'diff(y,x)^2-x*'diff(y,x)-y=0;
(%o2)  $\left(\frac{d}{dx}y\right)^2 - x\left(\frac{d}{dx}y\right) - y = 0$ 
(%i3) contrib_ode(eqn,y,x);
(%t3)  $\left(\frac{d}{dx}y\right)^2 - x\left(\frac{d}{dx}y\right) - y = 0$ 
first order equation not linear in y'
(%o3) [ [ x = %e $-\frac{\log(\%t)}{2}$   $\left(\frac{3 \log(\%t)}{2} + \%c\right)$  , y = %t2 - %t x ] ]
```

Как видим, решение получено в параметрическом виде. С помощью команды `method` можно посмотреть метод, примененный для нахождения решения дифференциального уравнения:

Пример 18. Решить дифференциальное уравнение: $y' = (x + y)^2$.

```
(%i2) eqn: 'diff(y, x) = (x+y)^2;
(%o2)  $\frac{d}{dx}y = (y+x)^2$ 

(%i3) contrib_ode(eqn, y, x);
(%o3) [ [ x = %c - atan(sqrt(%t)), y = -x - sqrt(%t) ], [ x = atan(sqrt(%t)) + %c,
y = sqrt(%t) - x ] ]

(%i4) method;
(%o4) lagrange
```

3.3. Построение траекторий и поля направлений дифференциальных уравнений

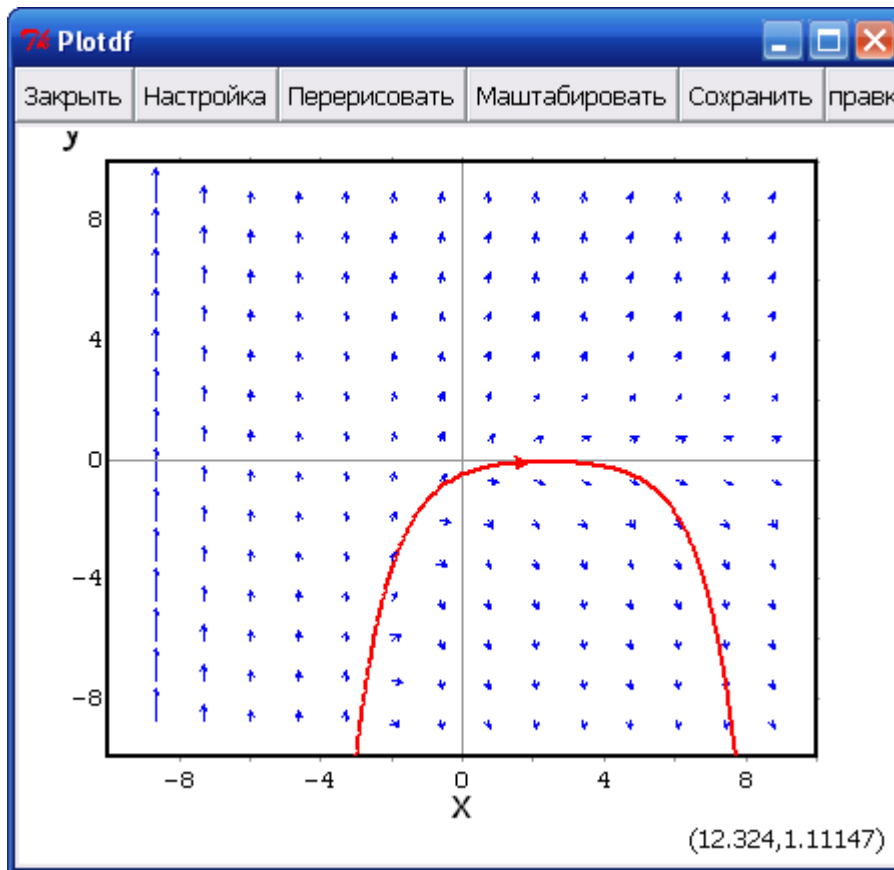
Пример 1. Построить интегральную кривую дифференциального уравнения $\frac{dy}{dx} = e^{-x} + y$, проходящую через точку $(2, -0.1)$, и поле направлений.

Подключим пакет plotdf с помощью команды load(plotdf)\$. Зададим команду для построения интегральной кривой уравнения и поля направлений. В результате у нас будут образованы две ячейки ввода с именами (%i1) и (%i2).

```
(%i1) load(plotdf)$
(%i2) plotdf(exp(-x)+y, [trajectory_at, 2, -0.1])$
```

После оценивания ячейки ввода %i2 откроется окно программы OpenMath с результатом выполнения команды.

Как видно, окно имеет несколько пунктов меню для работы с графическим изображением. Меню Масштабировать позволяет увеличивать и уменьшать график. Меню Настройка позволяет изменять некоторые установки: решаемое уравнение, начальная точка и направление вычисляемой траектории, шаг интервала интегрирования и др. С помощью меню Сохранить может быть сохранен построенный график в PostScript файл.



Пример 2. Построить интегральную кривую и поле направлений системы дифференциальных уравнений

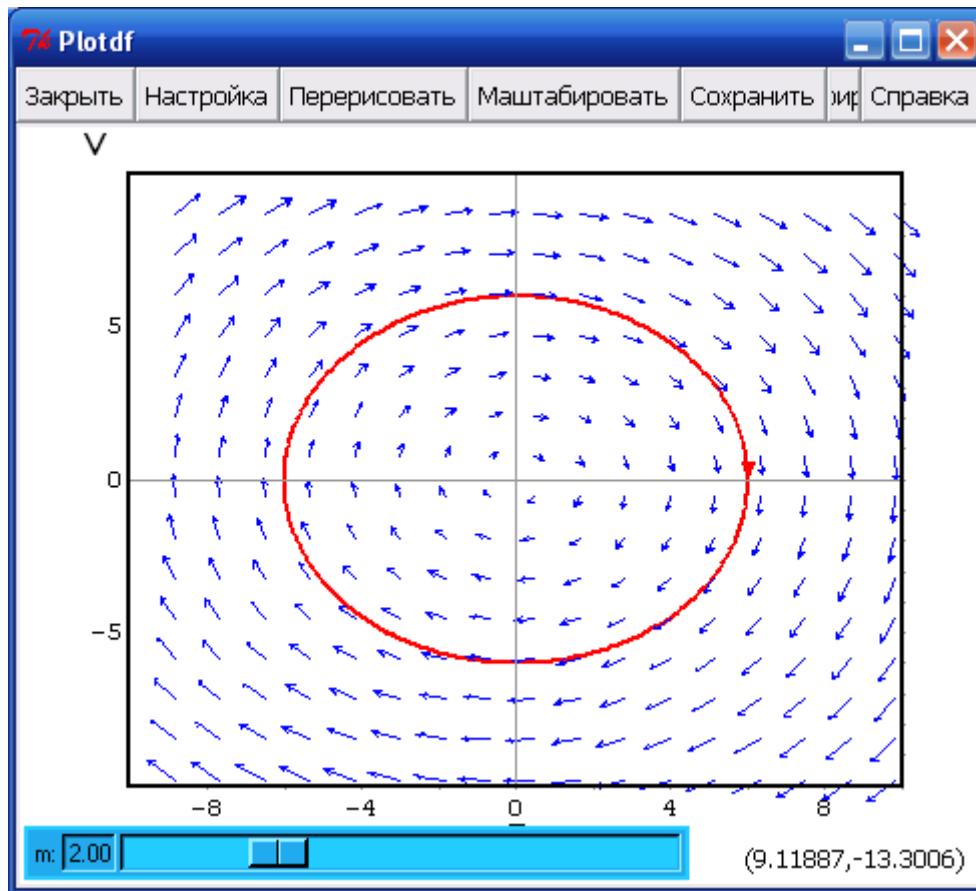
$$\begin{cases} \frac{dz}{dt} = v \\ \frac{dv}{dt} = \frac{-2*z}{m} \end{cases}$$

Построим интегральную кривую, определенную этими двумя уравнениями и проходящую через точку $(z, v) = (6, 0)$. Установим начальные значения параметра m второго уравнения системы равным 2. Воспользуемся опцией `slider`, которая позволит нам изменять значения параметра m в интерактивном режиме с помощью бегунка в левом нижнем углу графического окна.

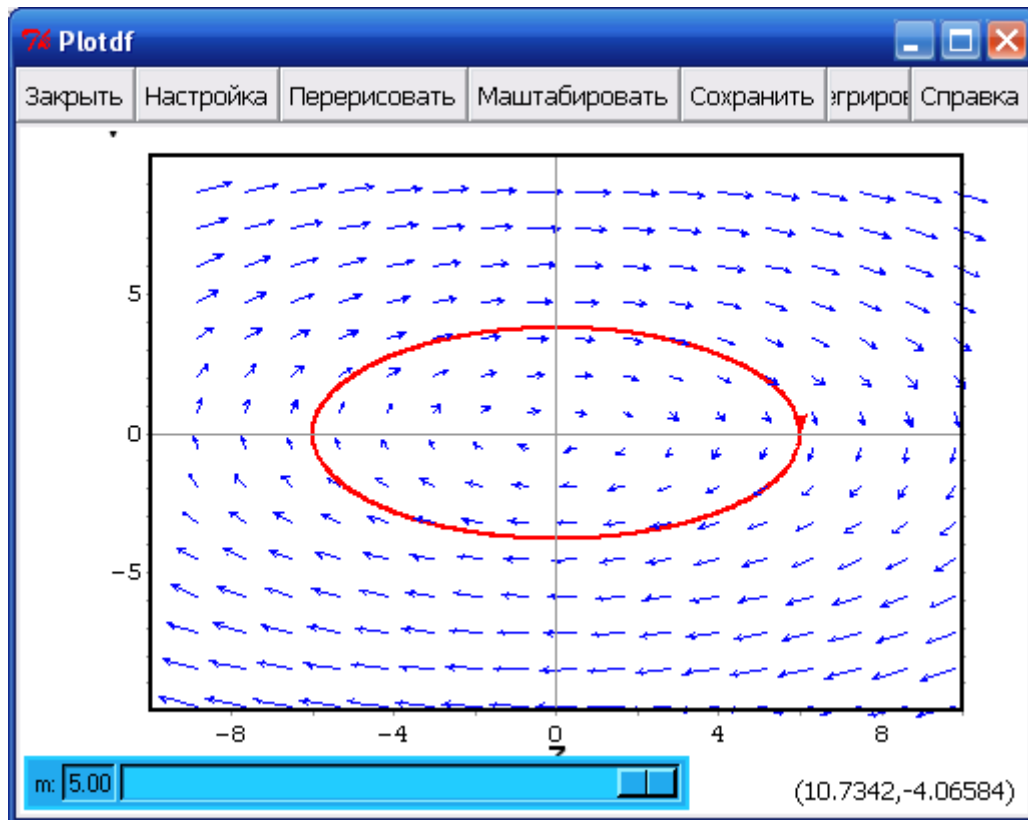
Вводим две команды:

```
(%i1) load(plotdf) $
(%i2) plotdf([v, -2*z/m], [z, v], [parameters, "m=2"],
[slicers, "m=1:5"], [trajectory_at, 6, 0]) $
```

В результате получаем:



Как видно из построений, начальное значение параметра m установлено равным 2. Передвигая бегунок полосы прокрутки, мы можем менять исходную систему, а соответственно и фазовый портрет системы:



Пример 3. При каких значениях параметра a особая точка системы

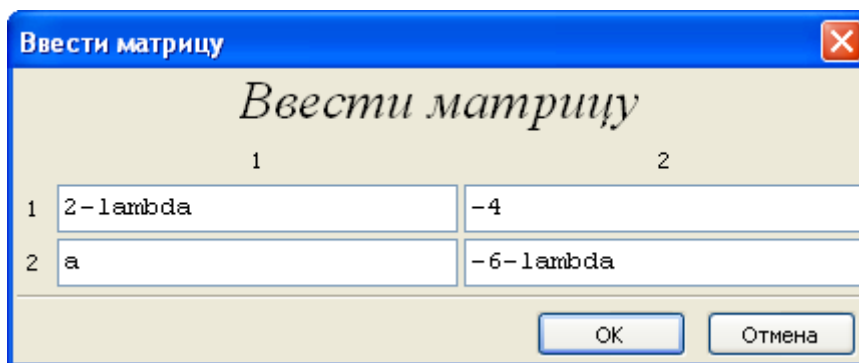
$$\begin{cases} \frac{dx}{dt} = 2x - 4y \\ \frac{dy}{dt} = ax - 6y \end{cases}$$

только одна и является седлом? узлом? фокусом? Дать чертеж

траекторий при $a=8$.

Составим характеристическое уравнение и найдем его корни. Для этого зададим матрицу. Выберем пункт меню *Алгебра* \rightarrow *Enter Matrix* (*Ввести матрицу*). Появится диалоговое окно, запрашивающее размерность матрицы.

После задания размерности нажимаем *ОК* и в следующем диалоговом окне вводим элементы матрицы:



При нажатии на кнопке *OK* создаются ячейки ввода и вывода с матрицей.

```
(%i1) matrix(
      [2-lambda,-4],
      [a,-6-lambda]
    );
(%o1) 
$$\begin{bmatrix} 2 - \lambda & -4 \\ a & -\lambda - 6 \end{bmatrix}$$

```

Вычислим определитель матрицы и найдем собственные значения матрицы.

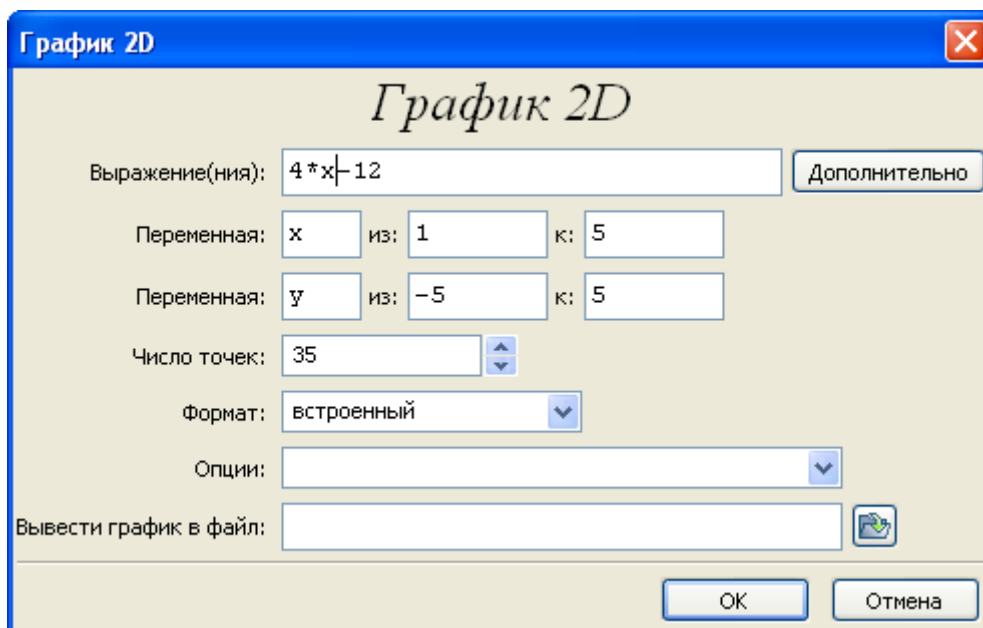
```
(%i2) d:determinant(%);
(%o2) (-lambda - 6)(2 - lambda) + 4 a

(%i3) solve([%=0], [lambda]);
(%o3) [ lambda = -2*sqrt(4 - a) - 2 , lambda = 2*sqrt(4 - a) - 2 ]
```

Вычислим произведение собственных значений матрицы $\lambda_1 \lambda_2$:

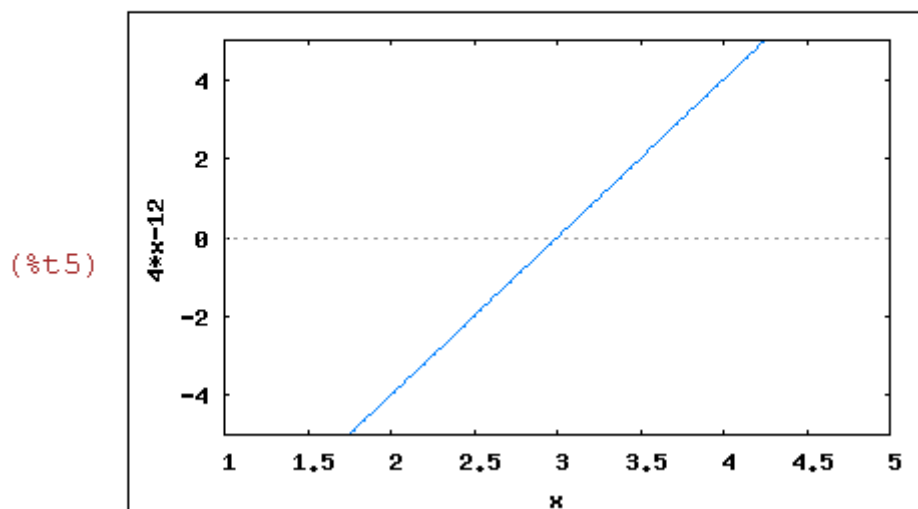
```
(%i4) expand((-2*sqrt(4-a)-2)*(2*sqrt(4-a)-2));
(%o4) 4 a - 12
```

Так как система Maxima не умеет решать неравенства в символьном виде, то определим графически знак полученного произведения $\lambda_1 \lambda_2$ при различных значениях параметра a . Для этого воспользуемся пунктом меню *Графики->Plot 2d* (*График 2D*), заполним значения полей в появившемся диалоговом окне следующим образом (при выборе формата *Встроенный* график будет располагаться не в отдельном окне, а в самом документе):



В результате получим следующее построение:

```
(%i5) wxplot2d([4*x-12], [x, 1, 5], [y, -5, 5],
               [nticks, 50])$
```



Из графика видно, что $\lambda_1 \lambda_2 = 4a - 12 = \begin{cases} < 0 \text{ при } a < 3, \\ = 0 \text{ при } a = 3 \\ > 0 \text{ при } a > 3 \end{cases}$. Остается посмотреть

на знак дискриминанта и определить, при каких значениях a собственные значения являются действительными, а при каких комплексными, числами. Вычислим дискриминант выражения d , предварительно представив его в виде квадратного трехчлена:


```
(%i7) expand(d);
```

```
(%o7) lambda^2 + 4 lambda + 4 a - 12
```

```
(%i8) dis:16-4*(4*a-12);
```

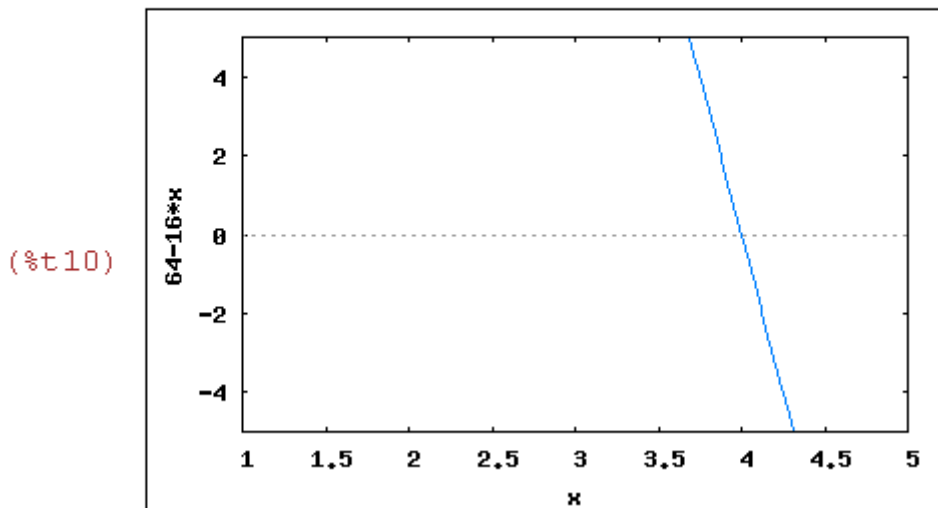
```
(%o8) 16 - 4(4 a - 12)
```

```
(%i9) radcan(%);
```

```
(%o9) 64 - 16 a
```

Определим знак дискриминанта при различных значениях параметра a .

```
(%i10) wxplot2d([64-16*x], [x,1,5], [y,-5,5],
[nticks,29])$
```



Видим, что $dis = 64 - 16a = \begin{cases} < 0 \text{ при } a > 4, \lambda_1, \lambda_2 - \text{комплексные} \\ = 0 \text{ при } a = 4, \lambda_1 = \lambda_2 = -2 \\ > 0 \text{ при } a < 4, \lambda_1, \lambda_2 - \text{вещественные различные} \end{cases}$.

Тогда, согласно классификации особых точек [см., например, 1, с.112-114], получаем:

1. При $a < 3$ имеем $\lambda_1 \lambda_2 < 0$, особая точка – седло.
2. При $a = 3$ имеем $\lambda_1 = 0, \lambda_2 = -4$, особых точек много.
3. При $3 < a \leq 4$ корни вещественные, $\lambda_1 \lambda_2 > 0$, особая точка устойчивый узел (при $a = 4$ — вырожденный узел).
4. При $a > 4$ корни комплексные, особая точка устойчивый фокус.

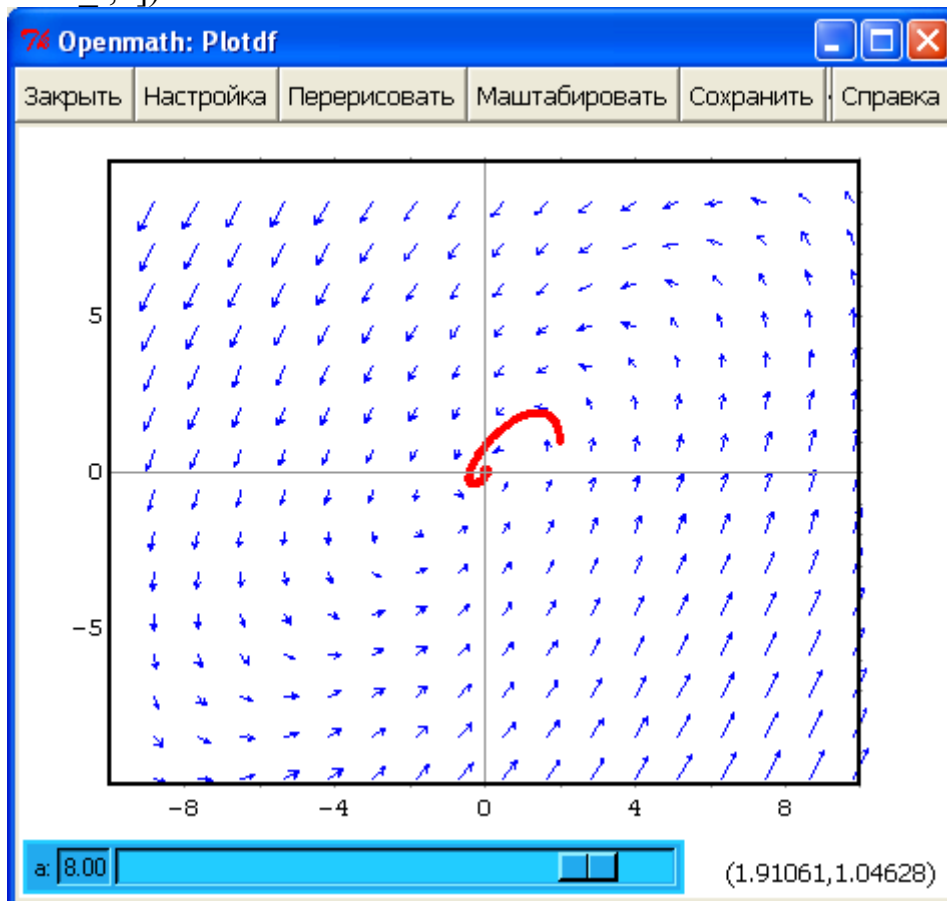
Подтвердим рассуждения построениями. Построим поле направлений для исходной системы уравнений, положив, что значение параметра a меняется от -8 до 10, тем самым мы сможем, изменяя положение бегунка полосы прокрутки, просмотреть все возможные варианты особых точек. За началь-

ное значение параметра a возьмем значение равное 8 (согласно условию задачи). Задаем команду для построения поля направлений:

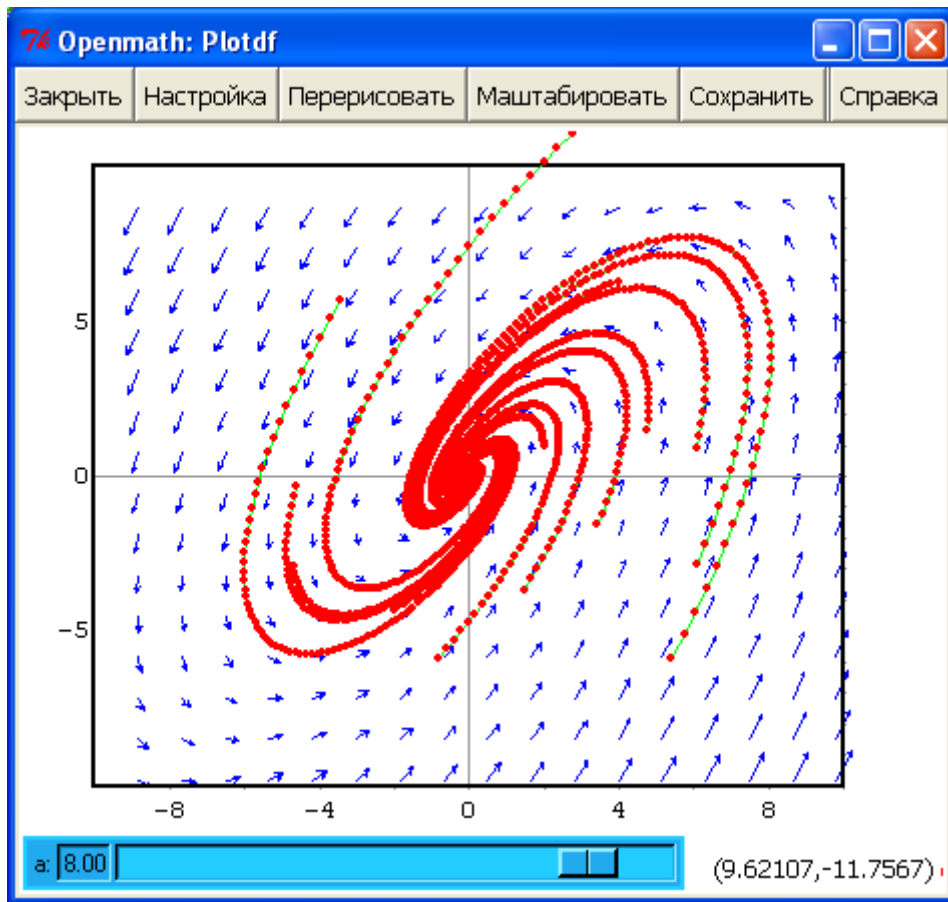
```
(%i12) load("plotdf")$

(%i21) plotdf([2*x-4*y, a*x-6*y], [x, y],
[parameters, "a=8"], [trajectory_at, 2, 1],
[tstep, 0.01], [x, -10, 10], [y, -10, 10],
[direction, forward], [nsteps, 300],
[sliders, "a=-8:10"], [versus_t, 1])$
```

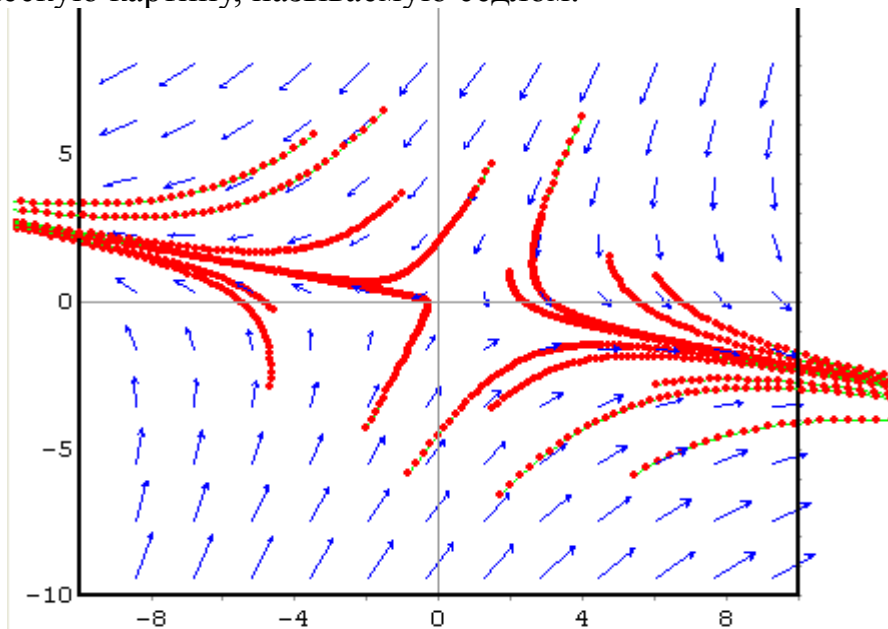
В результате у нас откроются два графических окна – одно с изображением поля направлений и траекторией, проходящей через точку (2,1), второе – с изображением значений функций $x[t]$ и $y[t]$ параметра t (чтобы это окно убрать, необходимо из строки для построения поля направлений удалить команду `[versus_t, 1]`).



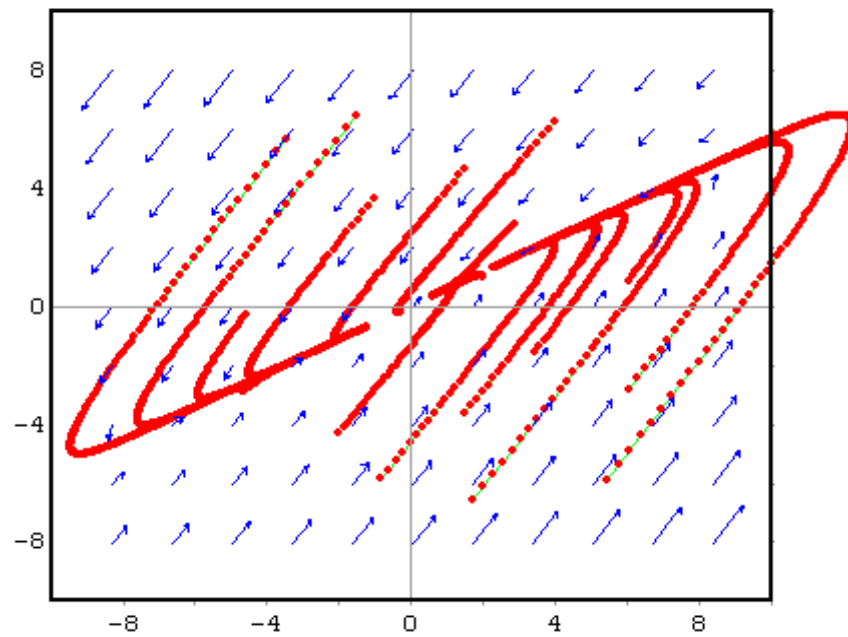
Для большей наглядности добавим еще несколько траекторий в окне поля направлений (щелкая левой кнопкой мыши в поле направлений) и тем самым убедимся в том, что у нас особая точка является устойчивым фокусом.



Передвинем бегунок полосы прокрутки в позицию меньше 3. Получим геометрическую картину, называемую седлом.

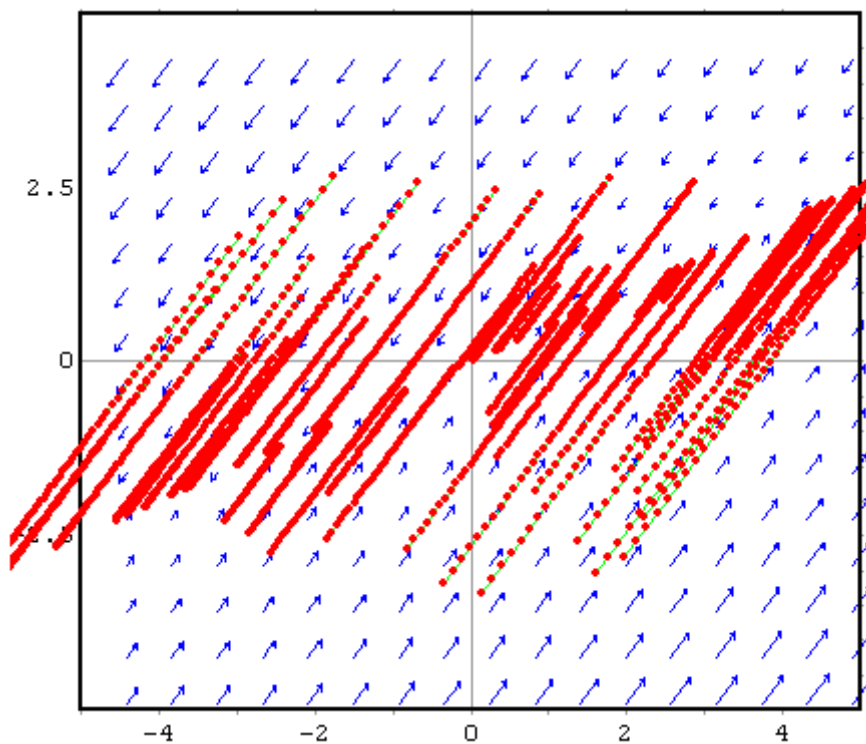


При a , изменяющемся от 3 до 4, имеем устойчивый узел.



Для значения параметра $a=3$ изменим команду `plotdf` следующим образом.

```
(%i17) plotdf([2*x-4*y, a*x-6*y], [x, y],
[parameters, "a=3"],
[tstep, 0.01], [x, -5, 5], [y, -5, 5],
[direction, forward], [nsteps, 300],
[sliders, "a=1:5"])$
```



Как видно из построений, система дифференциальных уравнений действительно имеет множество особых точек.

3.4. Реализация численных методов решения задачи Коши для обыкновенных дифференциальных уравнений

Во второй главе пособия были рассмотрены численные методы решения обыкновенного дифференциального уравнения первого порядка и приведены примеры решения одной задачи Коши методами Эйлера, Эйлера-Коши, Рунге-Кутта.

Постановка задачи (4.1): Найти решение задачи Коши: $\begin{cases} y' = y - x \\ y(0) = 1.5 \end{cases}$ на отрезке $[0, 1]$ с шагом $h = 0.2$. Найти точное решение задачи и найти величину абсолютной погрешности в указанных точках.

Решим поставленную задачу с помощью системы Maxima и сравним полученные результаты.

3.4.1. Метод Эйлера

Зададим концы отрезка, на котором будем искать решение, и шаг:

```
(%i1) a:0$b:0.6$ h:0.2$
```

Найдем количество точек разбиения отрезка с шагом h :

```
(%i4) n:1+floor((b-a)/h)$
```

Сформируем два пустых одномерных массива размера $n+1$ для хранения значения координат точек $[x, y]$ искомого решения:

```
(%i5) x1:make_array(flonum, n+1)$
      y1:make_array(flonum, n+1)$
```

Зададим начальное условие:

```
(%i7) x1[0]:a$ x1[1]:a+h$y1[0]:1.5$
```

Заполним массив $x1$ значениями, начиная с 0.2 до 1 с шагом h . Для этого используем цикл с параметром.

```
(%i10) for i:2 thru n step 1 do (x1[i]:x1[i-1]+h)$
```

Используя расчетную формулу Эйлера () главы 2, заполним массив $y1$:

```
(%i11) for i:1 thru n step 1 do
      (y1[i]:float(y1[i-1]+h*(y1[i-1]-x1[i-1])))$
```

Выведем полученное решение на экран:

```

(%i12) for i:0 thru n step 1 do
      (display(x1[i]), display(y1[i]));
x1_0=0
  y1_0=1.5
  x1_1=0.2
  y1_1=1.8
  x1_2=0.4
  y1_2=2.12
  x1_3=0.6
  y1_3=2.464
  x1_4=0.8
  y1_4=2.8368000000000001
  x1_5=1.0
  y1_5=3.2441600000000001

```

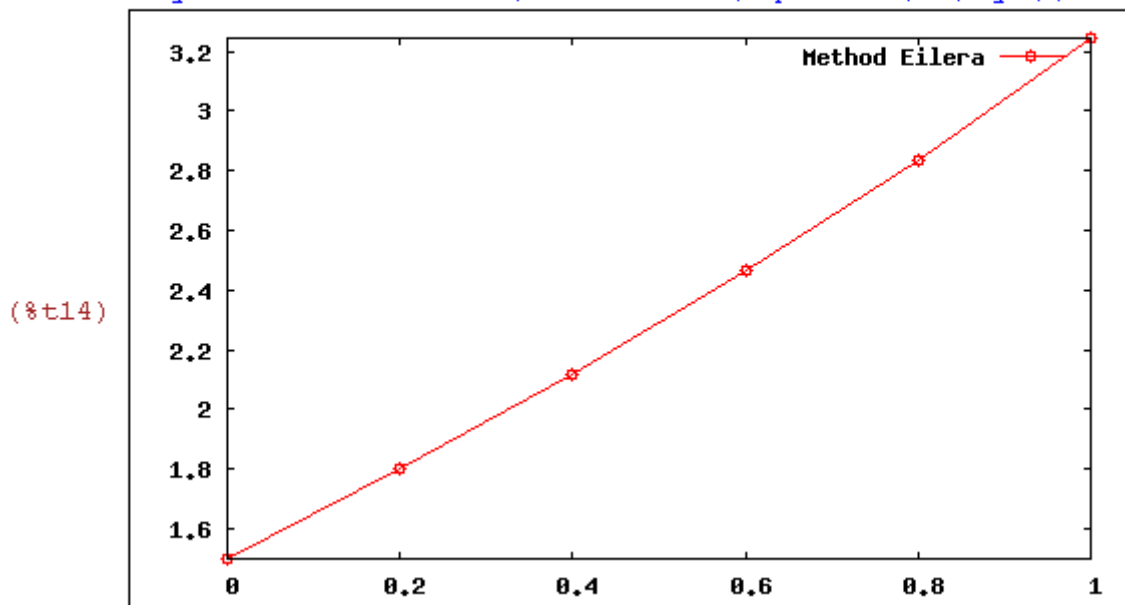
Выполним построение ломаной Эйлера (найденного решения задачи (4.1)) средствами пакета draw:

```

(%i13) load(draw)$

(%i14) wxdraw2d(point_type=circle,point_size=1,
  points_joined=true,
  key="Method Eilera", color=red, points(x1, y1))$

```



Для нахождения точного решения задачи Коши воспользуемся встроенной командой `desolve`. Дифференциальное уравнение запомним под именем `eq1`:

```
(%i15) eq1: 'diff(y(t),t)=y(t)-t;
```

```
(%o15)  $\frac{d}{dt}y(t)=y(t)-t$ 
```

Задаем начальное условие:

```
(%i16) atvalue(y(t), t=0, 1.5);
```

```
(%o16) 1.5
```

Находим точное решение задачи Коши (4.1):

```
(%i17) desolve(eq1, y(t));
```

```
rat: replaced -1.5 by -3/2 = -1.5
```

```
(%o17)  $y(t)=\frac{e^t}{2}+t+1$ 
```

Вычислим значения функции в точках отрезка $[0,1]$ с шагом $h=0.2$.

```
(%i19) for i:0 thru n do (z1[i]:%e^x1[i]/2+x1[i]+1,
display(z1[i]));
```

```
 $z1_0=\frac{3}{2}$ 
```

```
 $z1_1=1.810701379080085$ 
```

```
 $z1_2=2.145912348820635$ 
```

```
 $z1_3=2.511059400195255$ 
```

```
 $z1_4=2.912770464246234$ 
```

```
 $z1_5=3.359140914229522$ 
```

```
(%o19) done
```

Найдем величину абсолютной погрешности:

```
(%i20) for i:1 thru n do
display(abs(z1[i]-y1[i]));
```

```
|0.010701379080085|=0.010701379080085
```

```
|0.025912348820635|=0.025912348820635
```

```
|0.047059400195254|=0.047059400195254
```

```
|0.075970464246233|=0.075970464246233
```

```
|0.11498091422952|=0.11498091422952
```

```
(%o20) done
```

3.4.2. Метод Эйлера-Коши

Для решения задачи (4.1) методом Эйлера сформируем еще три пустых массива x_2 , y_2 и вспомогательный массив z .

```
(%i21) x2:make_array(flonum, n+1)$
      z:make_array(flonum, n+1)$
      y2:make_array(flonum, n+1)$
```

Зададим начальное условие:

```
(%i24) x2[0]:a$x2[1]:a+h$y2[0]:1.5$z[0]:1.5$
```

Заполним массив x_2 значениями, начиная с 0.2 до 1 с шагом h . Для этого используем цикл с параметром.

```
(%i28) for i:2 thru n step 1 do (x2[i]:x2[i-1]+h)$
```

Теперь воспользуемся расчетной формулой Эйлера-Коши и найдем решение:

```
(%i29) for i:1 thru n step 1 do (
      z[i]:float(y2[i-1]+h*(y2[i-1]-x2[i-1])),
      y2[i]:float(y2[i-1]+h*(y2[i-1]-x2[i-1]+
      z[i]-x2[i])/2))$
```

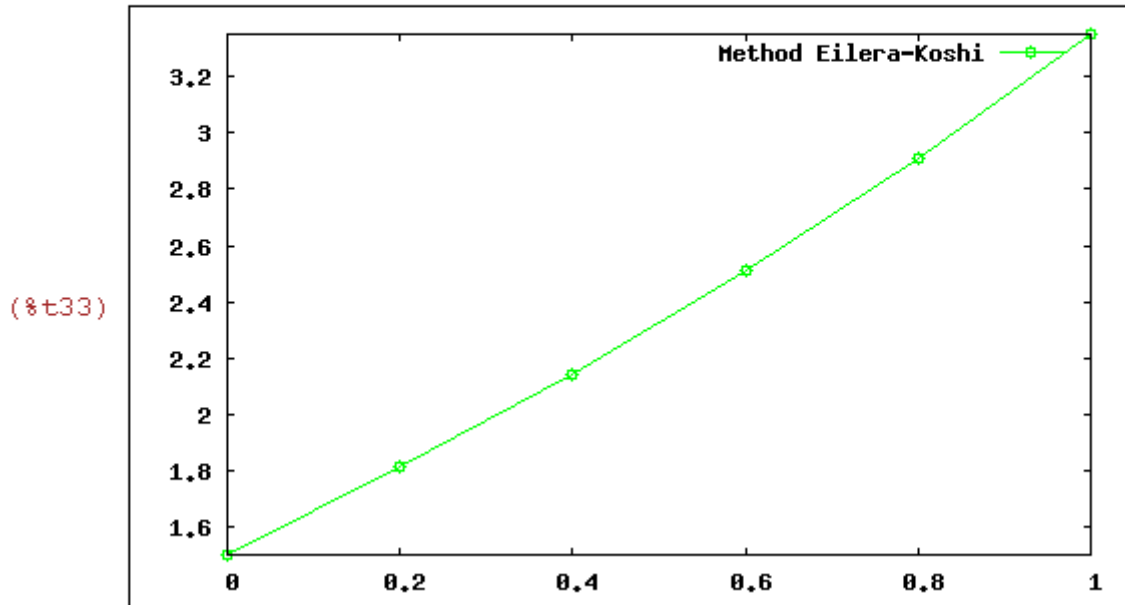
Выведем найденное решение на экран:

```
(%i31) for i:0 thru n step 1 do
      (display(x2[i]), display(y2[i]));
```

```
x2_0=0
y2_0=1.5
x2_1=0.2
y2_1=1.81
x2_2=0.4
y2_2=2.1442
x2_3=0.6
y2_3=2.507924
x2_4=0.8
y2_4=2.90766728
x2_5=1.0
y2_5=3.3513540816
(%o31) done
```


Выполним построение найденного решения задачи (4.1) средствами пакета draw:

```
(%i33) wxdraw2d(point_type =circle,point_size=1,
points_joined = true,
key="Method Eilera-Koshi",
color=green, points(x2, y2))$
```



Найдем величину абсолютной погрешности:

```
(%i32) for i:1 thru n do
      (display(abs(z1[i]-y2[i])));
|7.0137908008494065 10-4|=7.0137908008494065 10-4
|0.001712348820635|=0.001712348820635
|0.0031354001952546|=0.0031354001952546
|0.0051031842462339|=0.0051031842462339
|0.0077868326295221|=0.0077868326295221
(%o32) done
```

Как видим, метод Эйлера-Коши дает более точный результат, чем метод Эйлера. Максимальная погрешность вычислений составляет 0.7%.

3.4.3. Метод Рунге-Кутта

В системе Maxima для нахождения численного решения задачи Коши методом Рунге-Кутта (четвертого порядка точности) есть встроенная функция rk. Для того, чтобы она стала активной, требуется подключить пакет dynamics с помощью команды:

```
(%i34) load("dynamics")$
```

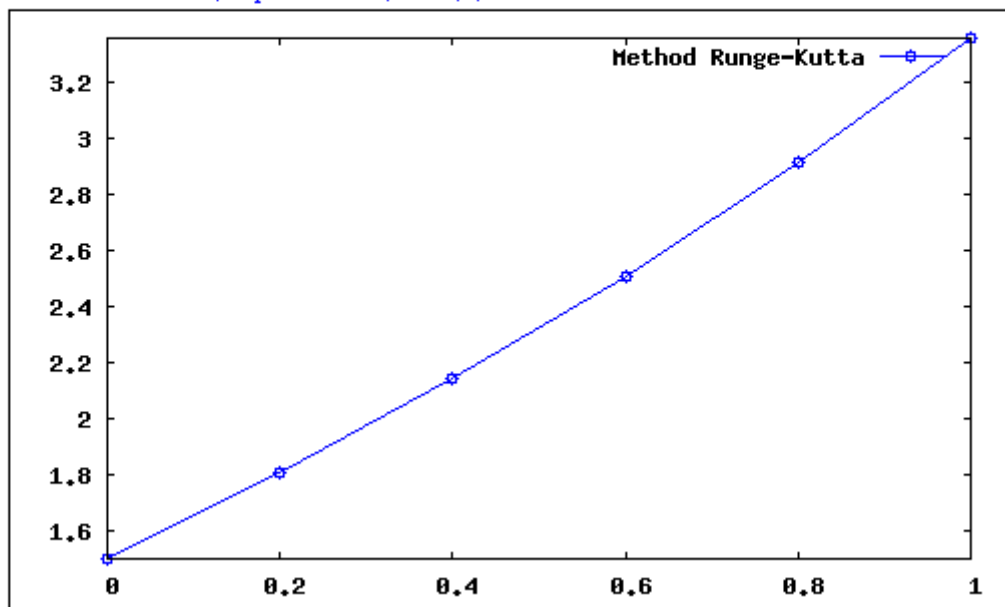
Теперь задаем команду для нахождения решения:

```
(%i37) sol: rk(y-x, y, 1.5, [x, 0, 1, 0.2]);
(%o37) [[0, 1.5], [0.2, 1.8107], [0.4, 2.14590898], [0.6,
2.511053228172], [0.8, 2.912760412889281], [1.0,
3.359125568302968]]
```

Выполним построение найденного решения задачи (4.1) средствами пакета draw:

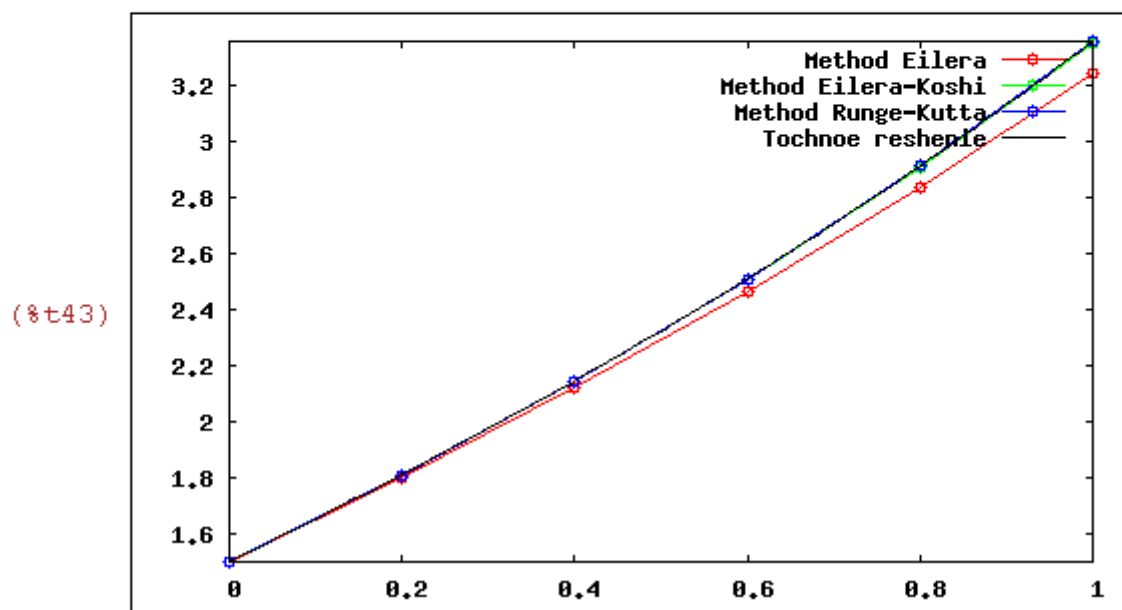
```
(%i38) wxdraw2d(point_type=circle,point_size=1,
points_joined=true,
key="Method Runge-Kutta",
color=blue, points(sol))$
```

(%t38)



Можно выполнить построение всех трех интегральных кривых в одной системе координат.

```
(%i43) wxdraw2d(point_type=circle,point_size=1,
points_joined=true,key="Method Eilera",
color=red, points(x1, y1),point_type=circle,
point_size=1,points_joined=true,
key="Method Eilera-Koshi", color=green,
points(x2, y2),point_type=circle,point_size=1,
points_joined=true,key="Method Runge-Kutta",
color=blue, points(sol),
key="Tochnoe reshenie", color=black,
explicit(exp(x)/2+x+1, x, 0, 1))$
```



Как видим, метод Эйлера дает самое плохое приближение к точному решению задачи Коши (4.1).

3.5. Реализация конечно-разностного метода решения краевой задачи для обыкновенных дифференциальных уравнений

Постановка задачи (5.1): Найти решение уравнения $y'' + 2y' + \frac{y}{x} = 5$ на $[0,4; 0,7]$ ($n=3$) с начальными условиями $y(0,4)=7$, $y(0,7)-2y'(0,7)=3$.

Решение задачи (5.1.) было рассмотрено во второй главе. Здесь же выполним решение краевой задачи конечно-разностным методом в системе Maxima.

Вводим обозначения и задаем значения переменных:

```
(%i1) p(x):=2$ q(x):=1/x$ f(x):=5$
      alfa1:1$ alfa2:0$ ac:7$ bc:3$
      beta1:1$ beta2:-2$
      a1:0.4$ b1:0.7$
```

Разбиваем отрезок $[a, b]$ на равные части с шагом $h=0.1$, $n=3$.

```
(%i12) n:3$ h:0.1$
```

Формируем список, содержащий все точки отрезка:

```
(%i14) x:makelist(a1+(k-1)*h,k,1,n+1);
(%o14) [0.4, 0.5, 0.6, 0.7]
```

Сформируем пустую квадратную матрицу a размера $n+1$:

```
(%i15) a:genmatrix(lambda([i,j],0),n+1,n+1);
```

```
(%o15) 
$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

```

Теперь заполним матрицу а по формулам:

$$a_{k,k} = h^2 q_k - h p_k + 1, \quad a_{k,k+1} = h p_k - 2, \quad a_{k,k+2} = 1, \quad k=1, n-1, \\ a_{n,1} = \alpha_1 h - \alpha_2, \quad a_{n,2} = \alpha_2, \quad a_{n+1,n} = -\beta_2, \quad a_{n+1,n+1} = \beta_1 h + \beta_2, \quad k=n, n+1$$

Для заполнения коэффициентами первых двух уравнений системы воспользуемся циклом с параметром:

```
(%i16) for i:1 thru n-1 do for j:1 thru n+1 do
  (if i=j then a[i,j]:h*h*q(x[i])-h*p(x[i])+1
  else if j=i+1 then a[i,j]:h*p(x[i])-2
  else if j=i+2 then a[i,j]:1 else 0);
(%o16) done
```

В двух последних уравнениях поменяем значения некоторых элементов с помощью оператора присваивания:

```
(%i17) a[n,1]:alfa1*h-alfa2$ a[n,2]:alfa2$
  a[n+1,n]:-beta2$ a[n+1,n+1]:beta1*h+beta2$
```

Теперь заполним столбец свободных членов:

```
(%i21) b:makelist(if k<=3 then h^2*f(x[k])
  else 0,k,1,n+1)$
  b[n]:h*ac$ b[n+1]:h*bc$
```

Выведем полученные матрицы на экран:

```
(%i28) b;
(%o28) {0.05, 0.05, 0.7, 0.3}
```

```
(%i26) a;
(%o26) 
$$\begin{bmatrix} 0.825 & -1.8 & 1 & 0 \\ 0 & 0.82 & -1.8 & 1 \\ 0.1 & 0 & 0 & 0 \\ 0 & 0 & 2 & -1.9 \end{bmatrix}$$

```

Получили систему линейных уравнений, записанную в матричном виде $ay=b$, где y — искомое решение. Найдем его матричным способом.

```
(%i29) invert(a).b;
(%o29)

$$\begin{bmatrix} 6.999999999999998 \\ 7.749999999999998 \\ 8.224999999999998 \\ 8.499999999999998 \end{bmatrix}$$

```

3.6. Реализация метода сеток для дифференциальных уравнений в частных производных

Во второй главе мы рассмотрели суть метода сеток для нахождения численных решений дифференциальных уравнений в частных производных. Выполним реализацию конечно-разностного метода в системе компьютерной математики Maxima.

Постановка задачи:

$$\begin{cases} u_{xx} = \frac{1}{a^2} u_{tt}, & 0 \leq x \leq m, \quad 0 \leq t \leq n \\ u(x, 0) = \sin \frac{\pi x}{50}, \quad u_t(x, 0) = 0 \\ u(0, t) = u(m, t) = 0 \end{cases} .$$

Решение:

Вводим сетку: $m=100$, $n=200$, $h=1$. Создаем нулевой массив значений $U(i, j)$ размера $m \times n$.

```
(%i1) kill(all)$m:100$n:200$ h:1$
(%i4) for i:1 thru m do for j:1 thru n do
      (arraymake(u, [i,j]), u[i,j]:0)$
```

Задаем значения $a=1$, $k=0,1$.

```
(%i5) a:1$ k:0.01$
```

Заполняем первую и вторую строки массива U начальными условиями $u(x, 0) = \sin \frac{\pi x}{50}$, $u_t(x, 0) = 0$ (нулевой начальной скорости соответствует совпадение значений (смещений) в первом и втором столбцах).

```
(%i7) for j:1 thru n do
      (u[1,j]:sin(%pi*j/50), u[2,j]:u[1,j])$
```

Заполняем первый и последний столбец массива U граничными условиями $u(0, t) = u(m, t) = 0$ (на концах струны смещение равно нулю в любой момент времени).

```
(%i8) for i:1 thru m do (u[i,1]:0, u[i,n]:0)$
```

Находим решение, используя разностную схему

$$u_{i+1,j} = \frac{a^2 k^2}{h^2} [u_{i,j+1} - 2u_{i,j} + u_{i,j-1}] + 2u_{i,j} - u_{i-1,j} .$$

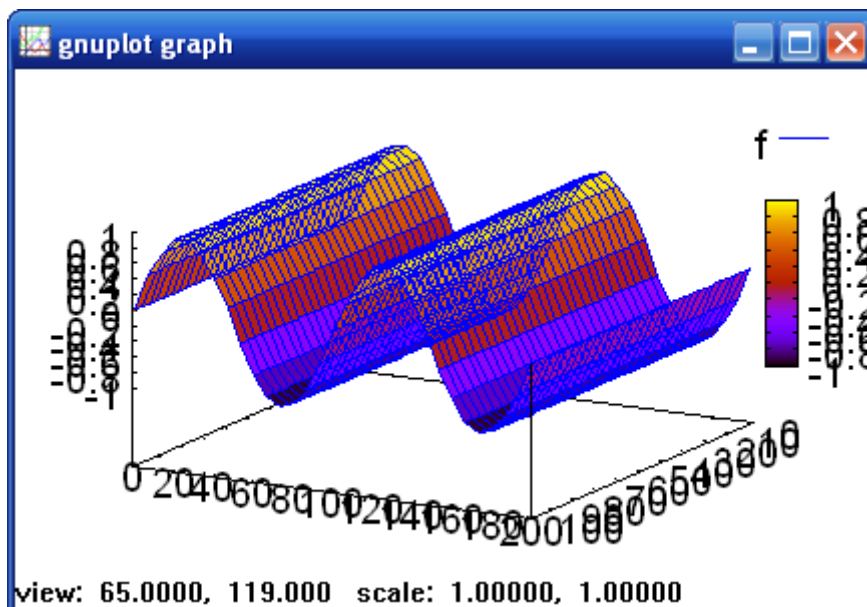
```
(%i8) for i:2 thru m-1 do for j:2 thru n-1 do
      u[i+1,j]:float((a*k/h)^2*(u[i, j+1]-
      2*u[i,j]+u[i,j-1])+2*u[i,j]-u[i-1, j]),numer$
```

Для вывода полученного решения в виде поверхности преобразуем наш массив U в функцию двух переменных:

```
(%i9) f(x,y):=float(u[round(x), round(y)])$
```

Теперь выполняем построение:

```
(%i10) plot3d(f, [x,1,m], [y,1,n], [plot_format,gnuplot])$
```



Задания для самостоятельного решения

1. Найти решение дифференциального уравнения в Математике:
 - $y'' + y' - 2y = 0, y'' - 3y' + 2y = x \cos x$
 - $y'' + 4y' + 3y = 0, y'' + 3y' - 4y = e^{-4x} + x e^{-x}$
 - $y'' - 4y' + 5y = 0, y'' + 2y' - 3y = x^2 e^x$
2. Найти решение уравнения в Математике, удовлетворяющее указанным начальным условиям. Построить график решения.
 - $y'' - 2y' + y = 0, y(2) = 1, y'(2) = -2$
 - $y'' + y = 4e^x, y(0) = 4, y'(0) = -3$
 - $y'' + 2y' + 2y = x e^{-x}, y(0) = y'(0) = 0$
3. Найти решение дифференциального уравнения в Математике:
 - $(2x + 1)y' = 4x + 2y$
 - $y' + y \operatorname{tg} x = \sec x$
 - $(xy + e^x)dx - xdy = 0$
4. Решить задачу Коши для дифференциального уравнения $y' = f(x, y)$ на отрезке $[a, b]$ при заданном начальном условии $y(a) = c$ с шагом h : методом Эйлера, методом Эйлера-Коши, методом Рунге-Кутты 4 порядка.
 - $f(x, y) = 2x^2 + xy + 3y^2$
 - $f(x, y) = 5x + 3 \cos(y + 2, 6)$
 - $f(x, y) = 5 - 2 \sin(x + 2y)^2$
5. Используя метод конечных разностей, найти решение краевой задачи для обыкновенного дифференциального уравнения второго порядка с шагом $h = 0,1$.
 - $y'' + y' - xy = 2x^2, y'(0,6) = 0,57, y(0,9) = 0,95, y'(0,9) = 3$
 - $y'' + 2y' + y/x = 5, y(0,4) = 7, y(0,7) - 2y'(0,7) = 3$
 - $y'' + xy' + 2y = x - 3, y(0,9) - 4y'(0,9) = -1, y(1,2) = 3$
6. Найти решение системы дифференциальных уравнений в Математике:
 - $\begin{cases} x' = 2x + y, \\ y' = 3x + 4y \end{cases}$
 - $\begin{cases} x' = x - y, \\ y' = y - 4x \end{cases}$
 - $\begin{cases} x' + x - 8y = 0, \\ y' - x - y = 0 \end{cases}$
7. Найти решение системы дифференциальных уравнений в Математике:
 - $\begin{cases} x' = y + 2e^t, \\ y' = x + t^2 \end{cases}$
 - $\begin{cases} x' = y - 5 \cos t, \\ y' = 2x + y \end{cases}$

$$\circ \begin{cases} x' = 3x + 2y + 4e^{5t}, \\ y' = x + 2y \end{cases}$$

8. Исследовать особые точки системы в Maxima. Начертить на фазовой плоскости траектории системы:

$$\circ \begin{cases} x' = 3x, \\ y' = 2x + y \end{cases}$$

$$\circ \begin{cases} x' = 2x - y, \\ y' = x \end{cases}$$

$$\circ \begin{cases} x' = x + 3y, \\ y' = -6x - 5y \end{cases}$$

9. Методом сеток с шагом h найти приближенное решение волнового

уравнения $\frac{\partial^2 u(x,t)}{\partial x^2} = \frac{1}{a^2} \frac{\partial^2 u(x,t)}{\partial t^2}$ (x - смещение, t - время) в пря-

моугольной области $[0, m] \times [0, n]$, используя явную разностную схему.

Граничные условия: на концах струны смещение равно 0 в любой момент времени: $u(0, t) = u(m, t) = 0$. Начальные условия:

$u(x, 0) = \varphi(x), \frac{\partial u(x, 0)}{\partial t} = 0$. Так как концы струны закреплены, то

$\varphi(0) = \varphi(m) = 0$. Шаг изменения времени τ выбирается самостоя-

тельно (так, чтобы разностная схема была устойчивой). Результат представить в виде поверхности.

$$\circ a = 1, m = 42, n = 30, h = 1, \varphi(x) = 0, 1x \sin\left(\pi \frac{x}{21}\right)$$

$$\circ a = 1, m = 36, n = 25, h = 1, \varphi(x) = \sin\left(\pi \frac{x}{9}\right) \cos(\pi x)$$

$$\circ a = 1, m = 39, n = 45, h = 1, \varphi(x) = 0, 2x \cos\left(\pi \frac{x}{2}\right)$$

10. Методом сеток с шагом $h = 1$ пространственной переменной и $\tau = 1$ временной координаты найти приближенное решение уравнения теп-

лопроводности $\frac{\partial u(x,t)}{\partial t} = c \frac{\partial^2 u(x,t)}{\partial x^2}$ (x - смещение, t - время) в

квадрате $[0, n] \times [0, n]$, используя неявную разностную схему. Гранич-

ные условия: $u(0, t) = g_1(t) = \alpha(x=0, 0 \leq t \leq n)$,

$u(n, t) = g_2(t) = \beta(x=n, 0 \leq t \leq n)$. Начальное условие:

$u(x, 0) = \sin\left(\frac{\pi x}{n}\right) + \frac{x}{n} \beta + \frac{n-x}{n} \alpha (t=0, 0 \leq x \leq n)$. Результат представить в

виде поверхности.

$$\circ n = 30, c = 5, \alpha = -1, \beta = 1$$

$$\circ n = 50, c = 3, \alpha = 2, \beta = 3$$

$$\circ n = 10, c = 4, \alpha = -1, \beta = 0$$

11. Методом сеток с шагом h найти приближенное решение уравнения диффузии $\frac{\partial u(x,t)}{\partial t} = D \frac{\partial^2 u(x,t)}{\partial x^2}$ (x - смещение, t - время) в прямоугольной области $[0,a] \times [0,b]$, используя явную разностную схему.

Граничные условия: $u(0,t) = g_1(t)$ ($x=0, 0 \leq t \leq b$),
 $u(a,t) = g_2(t)$ ($x=a, 0 \leq t \leq b$). Начальное условие:

$u(x,0) = f(x)$ ($t=0, 0 \leq x \leq a$). Шаг изменения времени τ выбирается самостоятельно (так, чтобы разностная схема была устойчивой). Результат представить в виде поверхности.

- $a=60, D=1, b=50, h=1, f(x)=\sin(x), g_1(t)=0, g_2(t)=\sin 60$
- $a=40, D=2, b=60, h=1, f(x)=\cos(x), g_1(t)=1, g_2(t)=\cos 40$
- $a=30, D=10, b=20, h=1, f(x)=\sin(x/2), g_1(t)=0, g_2(t)=\sin 15$

Список литературы

1. Додьер Р. Коротко о Maxima (пер. на русский Бешенов А), 2007. (ЭВ)
2. Житников В. Компьютеры, математика и свобода // Компьютерра, 2006 г.
3. Олейник О.А. Роль теории дифференциальных уравнений в современной математике и ее приложениях // Соросовский образовательный журнал, 1996, №4, с. 114-121 .
4. С. Фарлоу. Уравнения с частными производными для научных работников и инженеров: пер. с англ./ под редакцией С.И. Похожаева. - М.: Мир, 1985. - 384 с.
5. Сливина Н.А. Профессиональные математические пакеты в образовании // Педагогические и информационные технологии в образовании. – № 2. – <http://scholar.urc.ac.ru:8002/Teachers/methodics/journal/numero2/slivina.html>
6. Тарнавский Т. Maxima — алгебра и начала анализа // LinuxFormat, № 11, 2006 (ЭВ)
7. Тарнавский Т. Maxima — графики и управляющие конструкции // LinuxFormat, № 11, 2006 (ЭВ)
8. Тарнавский Т. Maxima — максимум свободы символьных вычислений // LinuxFormat, № 7, 2006. (ЭВ)
9. Тарнавский Т. Maxima — укротитель выражений // LinuxFormat, № 9, 2006 (ЭВ)
10. Тарнавский Т. Maxima — функции и операторы // LinuxFormat, № 8, 2006 (ЭВ)

Учебное издание

**РЕШЕНИЕ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ
В СИСТЕМЕ КОМПЬЮТЕРНОЙ МАТЕМАТИКИ MAXIMA**

Учебное пособие

Технический редактор – Н. П. Безногих
Техническое исполнение - В. М. Гришин
Компьютерная верстка – Т. Н. Губина
Переплет и обложка выполнены
в МУП "Типография" г. Ельца

Лицензия на издательскую деятельность
ИД № 06146. Дата выдачи 26.10.01.
Формат 60 x 84 /16. Гарнитура Times. Печать трафаретная.
Усл.-печ.л. 6,3 Уч.-изд.л. 6,5
Тираж 500 экз. (1-й завод 1-50 экз.). Заказ 85

Отпечатано с готового оригинал-макета на участке оперативной полиграфии
Елецкого государственного университета им. И.А.Бунина.
Государственное образовательное учреждение
высшего профессионального образования
«Елецкий государственный университет им. И.А. Бунина»
399770, г. Елец, ул. Коммунаров, 28