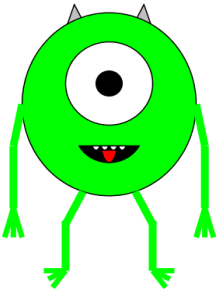
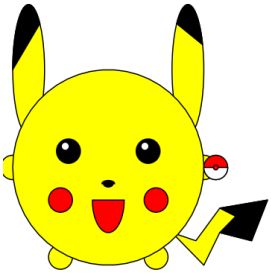
Т.М. Банникова, А.К. Кощеева



ГРАФИКА MAPLE НА ПЛОСКОСТИ



Ижевск
2025

Министерство науки и высшего образования Российской Федерации
ФГБОУ ВО «Удмуртский государственный университет»
Институт математики, информационных технологий и физики
Кафедра алгебры и топологии

Т.М. Банникова, А.К. Кощеева

ГРАФИКА MAPLE НА ПЛОСКОСТИ

Учебно-методическое пособие

Ижевск
2025

УДК 004.92(075.8)
ББК 32.972.131.2я73
Б232

Рекомендовано к изданию Учебно-методическим советом УдГУ

Рецензент: канд. физ.-мат. наук, ст. науч. сотрудник ФТИ УдмФИЦ УрО РАН **О.М. Немцова**

Банникова Т.М., Кощеева А.К.

Б232 Графика Maple на плоскости : учеб.-метод. пособие / Т.М. Банникова, А.К. Кощеева. – Ижевск : Удмуртский университет, 2025. – 8,2 Мб. – Текст : электронный.

Настоящее учебно-методическое пособие предназначено для бакалавров 1–2 курсов направлений подготовки 01.03.01 Математика, 01.03.02 Прикладная математика и информатика, 02.03.01 Математика и компьютерные науки, очной формы обучения, изучающих дисциплины «Компьютерная геометрия», «Компьютерная графика», «Компьютерная анимация», «Компьютерная геометрия и геометрическое моделирование», «Математические пакеты в геометрии». Пособие может быть использовано при организации практических и самостоятельных работ студентов по этим дисциплинам. Учебно-методическое пособие также может быть полезно студентам различных специальностей высших учебных заведений при изучении тем, связанных компьютерной графикой.

Минимальные системные требования:
Celeron 1600 Mhz; 128 Мб RAM; Windows XP/7/8 и выше;
разрешение экрана 1024×768 или выше; программа для просмотра pdf.

Банникова Татьяна Михайловна, Кощеева Анна Константиновна
Графика Maple на плоскости
Учебно-методическое пособие

Подписано к использованию 30.12.2025
Объем электронного издания 8,2 Мб
Издательский центр «Удмуртский университет»
426034, г. Ижевск, ул. Ломоносова, д. 4Б, каб. 021
Тел. : +7(3412)916-364 E-mail: editorial@udsu.ru

Введение

Пространственное представление человека всегда вначале мысленно создает некую объемную модель объекта, которая является основой для преобразования ее в ортогональные проекции. Идеология двухмерного проектирования заключается в выполнении изображений на основе воображаемого человеком трехмерного объекта с помощью набора различных линий и функций. Каждая проекция детали строится отдельно в проекционной связи и, в данном случае, автоматизируется лишь сам процесс получения изображения и проставления размеров.

Необходимость развития пространственного представления студентов требует такого подхода, при котором на начальном этапе изучения программных продуктов желательно использование двухмерных систем либо их двухмерных модулей.

Освоение студентами вузов компьютерной техники и программных графических продуктов позволяет:

– повысить уровень подготовки кадров для различных отраслей промышленности;

– ускорить процесс выполнения и улучшить качество учебных графических работ;

– использовать полученные знания и умения для разработки курсовых и дипломных работ.

Одним из пакетов, в котором возможно изучение предметов, связанных с компьютерной графикой, является студенческий пакет Maple. Студенты имеют возможность использовать его бесплатные версии.

Цель пособия – помощь в формировании у студентов необходимых знаний, умений и навыков по компьютерной геометрии.

Задачи:

• приобретение теоретических знаний об основных элементах и периферийных устройствах, определяющих эффективность использования компьютера при работе с графическим материалом;

• приобретение базовых основ создания графических изображений. Растровая, векторная, фрактальная графика. Основные представления о цветовых моделях (RGB, CMYK и т. д.);

• знакомство с современными стандартами компьютерной графики;

• приобретение теоретических знаний о способах хранения графической информации;

• приобретение прикладных знаний в области использования векторной графики в практической деятельности;

• приобретение прикладных знаний в области верстки изданий различного характера.

В результате изучения студент должен:

знать:

– основные требования, предъявляемые к компьютеру при работе с графическими редакторами;

– реализацию аппаратно-программных модулей графической системы;

– основные приемы геометрического моделирования и решаемые им задачи;

– основы представления видеоинформации и ее машинной генерации;

– современные стандарты компьютерной графики;

– основные способы визуализации изображения: растровая и векторная графика;

– основные форматы файлов, используемые при работе с графикой;

– основные принципы создания векторных графических изображений;

– основные принципы создания растровых изображений и их редактирования;

–

уметь:

– применять интерактивную графику в информационных системах;

– создавать двумерные растровые и векторные графические изображения и их редактировать в наиболее распространенных графических редакторах;

– сохранять созданные изображения в необходимом формате;

владеть:

– методами решения задач геометрической графики.

При изучении данной темы формируются элементы следующей совокупности профессиональных компетенций:

– способен принимать научно обоснованные решения на основе математического моделирования и методов системного анализа, осуществлять проверку их корректности, адекватности и эффективности;

– способен самостоятельно изучать новые разделы фундаментальных и прикладных наук.

Одной из форм оценки уровня сформированности заявленных компетенций является проект, выполняемый в рамках самостоятельной работы.

Цель проекта. Проект является самостоятельно выполняемым заданием, предназначен для усвоения обучаемым современных графических технологий и графических инструментов, требует разработки учащимся целостного законченного проекта.

Тема проекта. Проект выполняется по темам, указанным в приложении. Тема также может быть выбрана студентом самостоятельно, если она согласована с преподавателем курса и руководителем индивидуального обучения.

Проект выполняется студентом индивидуально. Допускается выполнение работы в составе группы – два, три человека при условии увеличения объема работ в соответствующее число раз.

Предпочтителен отчет, подготовленный компьютерным способом. При подготовке отчета следует учитывать требования ГОСТ на техническую документацию.

Данное учебно-методическое пособие разработано на основе опыта преподавания перечисленных дисциплин и современных методик обучения.

Maple – это программный пакет, система компьютерной математики, предназначенная для символьных вычислений. Система Maple имеет огромный спектр средств для численных решений. Этот продукт используют в разных областях математики: при решении дифференциальных уравнений, численных методов, теоретической механики и так далее. Программа имеет собственный интегрируемый язык программирования, легко усваиваемый для любого пользователя. В этом пособии мы рассмотрим различные графические пакеты, которые предоставляет нам это приложение.

Учебно-методическое пособие написано в соответствии с требованиями государственных образовательных стандартов третьего

поколения направлений подготовки 01.03.01 Математика, 01.03.02 Прикладная математика и информатика, 02.03.01 Математика и компьютерные науки. Включает в себя всесторонне изложенный теоретический материал с разобранными на каждую тему практическими заданиями, с объяснением практического смысла каждого введенного понятия. Данное пособие может быть использовано в качестве основной литературы для проведения лекций и практических занятий. Предпосылками написания данного пособия являлась необходимость систематизировать накопленный материал при многолетнем прочтении лекций и проведении практических занятий по работе с этой программой. При написании пособия были учтены современны требования и компетенции, предъявляемые к бакалавру. Материал был подобран так, чтобы не только можно было уловить суть предмета, но и понять его назначение в современном мире.

Лекция 1. Знакомство с программой Maple. Интерфейс программы. Подготовка к работе с графикой Maple

1. Знакомство с программой Maple

В Maple на сегодня в общей сложности используется более трех тысяч команд, однако некоторые из них (наиболее важные) применяются достаточно часто и составляют костяк базового языка. Они, в основном, имеют отношение к проблемам интегрирования и дифференцирования функций, решения уравнений и т.п. Некоторые команды доступны только при подключении специальных пакетов.

Отличительной особенностью Maple является то, что он работает с большинством платформ и операционных систем, в том числе Windows, Unix, Linux, VMS. При этом сам язык Maple для различных операционных систем один и тот же, так что существенной разницы, какая из них используется, нет. Кроме того, в Maple предусмотрены самые широкие возможности для преобразования рабочих документов во всевозможные форматы — RTF, LaTex, HTML и др. Это делает его незаменимым помощником не только при выполнении вычислений, но и при оформлении документации.

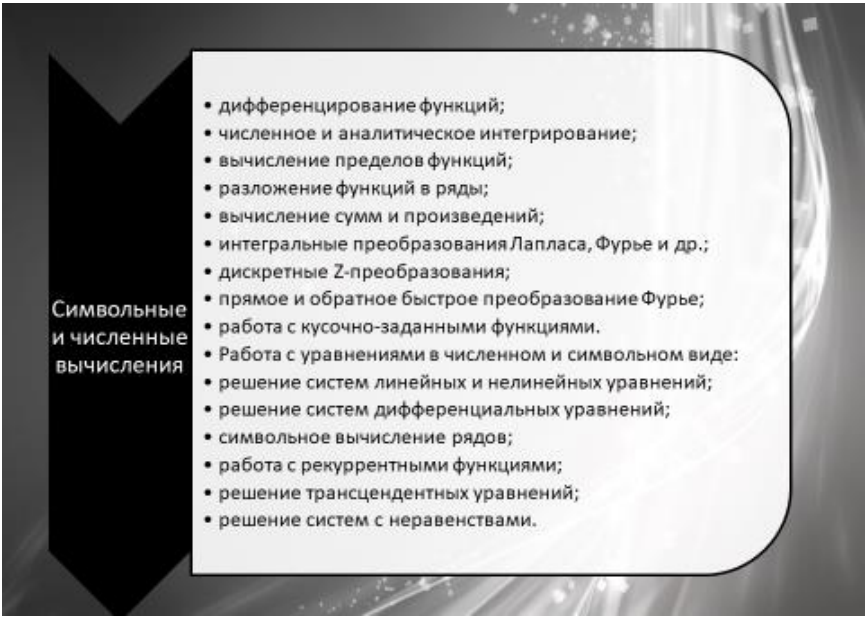
Maple — типичная интегрированная система. Она объединяет в себе:

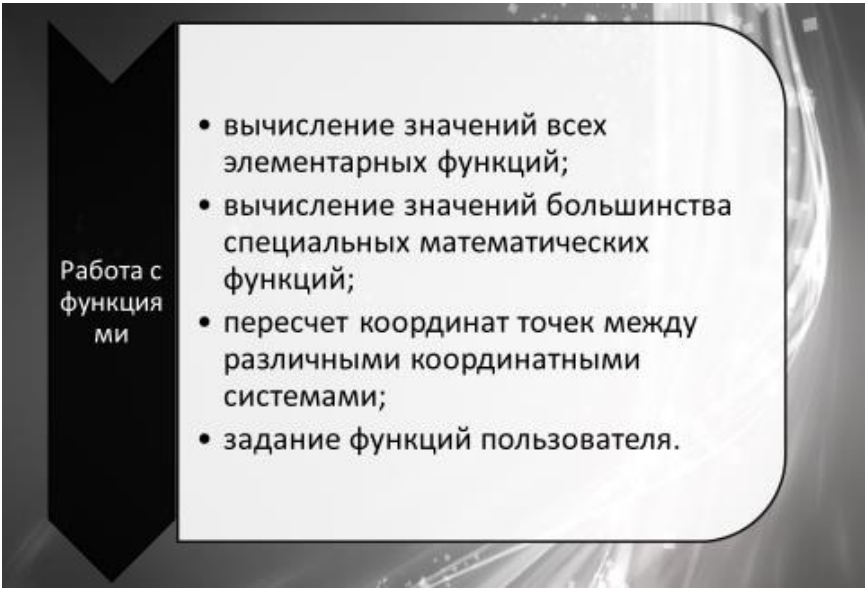
- мощный язык программирования (он же язык для интерактивного общения с системой);
- редактор для подготовки и редактирования документов и программ;
- современный многооконный пользовательский интерфейс с возможностью работы в диалоговом режиме;
- мощную справочную систему со многими тысячами примеров;
- ядро алгоритмов и правил преобразования математических выражений;
- численный и символьный процессоры;
- систему диагностики;
- библиотеки встроенных и дополнительных функций.

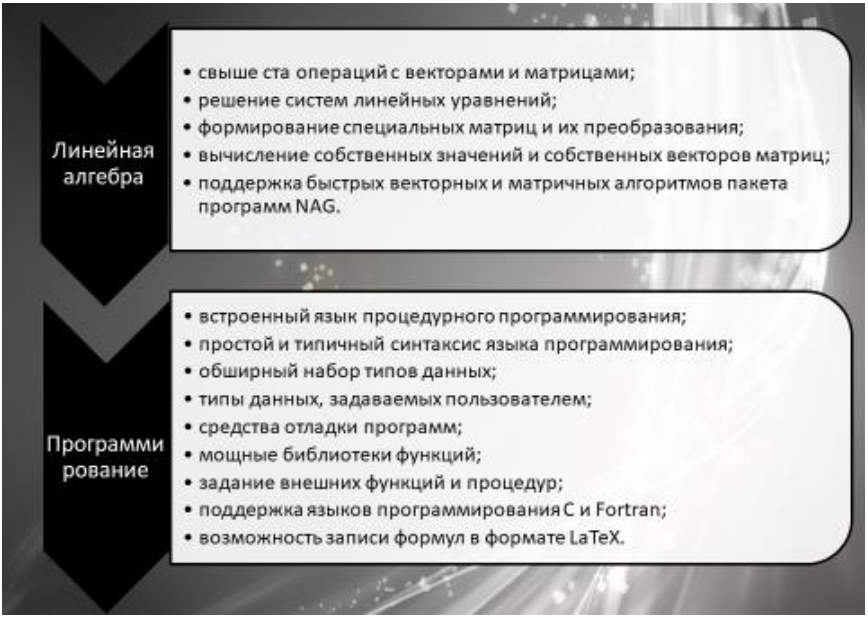
ВОЗМОЖНОСТИ:

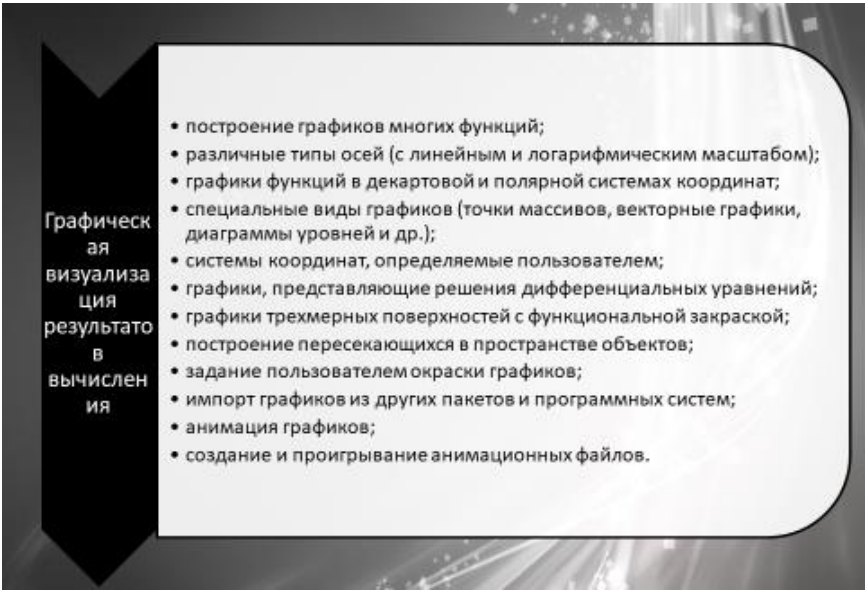
Интерфейс

- работа со многими окнами;
- вывод графиков в отдельных окнах или в окне документа;
- представление выходных и входных данных в виде естественных математических формул;
- задание текстовых комментариев различными шрифтами;
- возможность использования гиперссылок и подготовки электронных документов;
- удобное управление с помощью клавиатуры через главное меню и инструментальную панель;
- управление с помощью мыши.

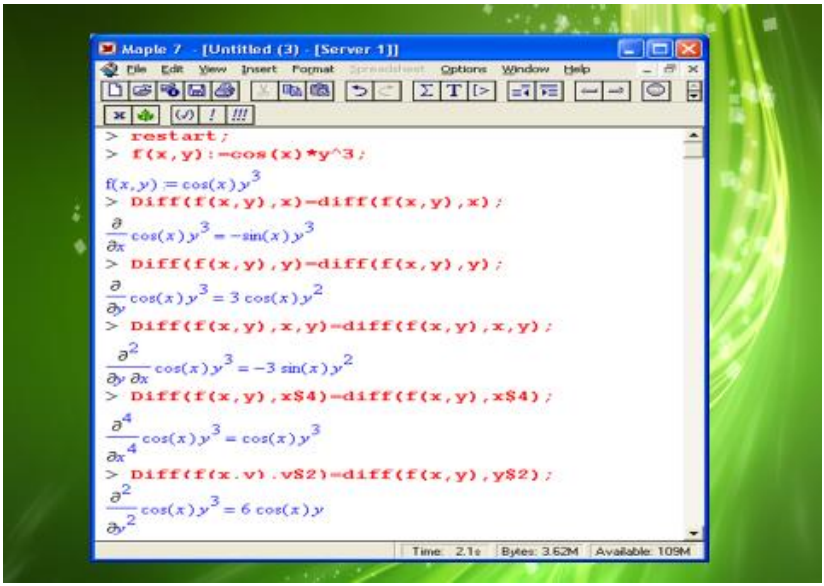

Символьные и численные вычисления
• дифференцирование функций;
• численное и аналитическое интегрирование;
• вычисление пределов функций;
• разложение функций в ряды;
• вычисление сумм и произведений;
• интегральные преобразования Лапласа, Фурье и др.;
• дискретные Z-преобразования;
• прямое и обратное быстрое преобразование Фурье;
• работа с кусочно-заданными функциями.
• Работа с уравнениями в численном и символьном виде:
• решение систем линейных и нелинейных уравнений;
• решение систем дифференциальных уравнений;
• символьное вычисление рядов;
• работа с рекуррентными функциями;
• решение трансцендентных уравнений;
• решение систем с неравенствами.

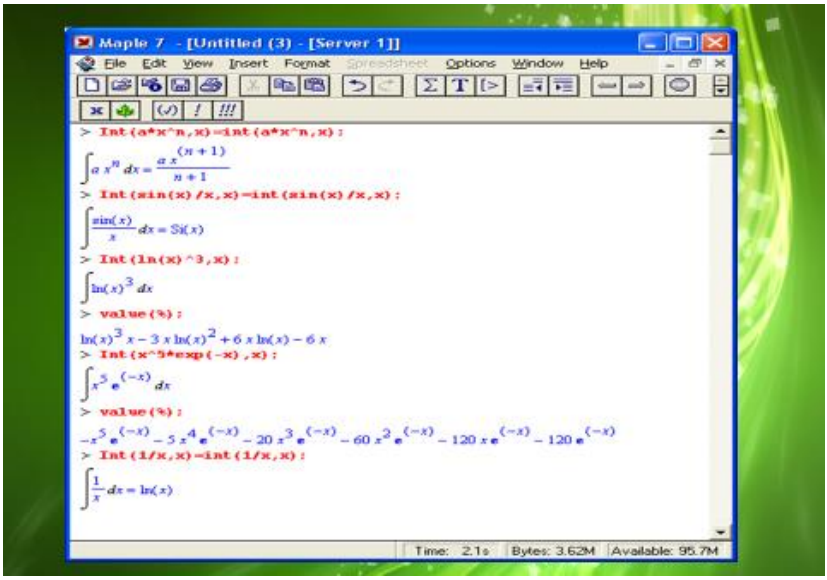

Работа с функциями
• вычисление значений всех элементарных функций;
• вычисление значений большинства специальных математических функций;
• пересчет координат точек между различными координатными системами;
• задание функций пользователя.


Линейная алгебра
• свыше ста операций с векторами и матрицами;
• решение систем линейных уравнений;
• формирование специальных матриц и их преобразования;
• вычисление собственных значений и собственных векторов матриц;
• поддержка быстрых векторных и матричных алгоритмов пакета программ NAG.
Программирование
• встроенный язык процедурного программирования;
• простой и типичный синтаксис языка программирования;
• обширный набор типов данных;
• типы данных, задаваемых пользователем;
• средства отладки программ;
• мощные библиотеки функций;
• задание внешних функций и процедур;
• поддержка языков программирования C и Fortran;
• возможность записи формул в формате LaTeX.

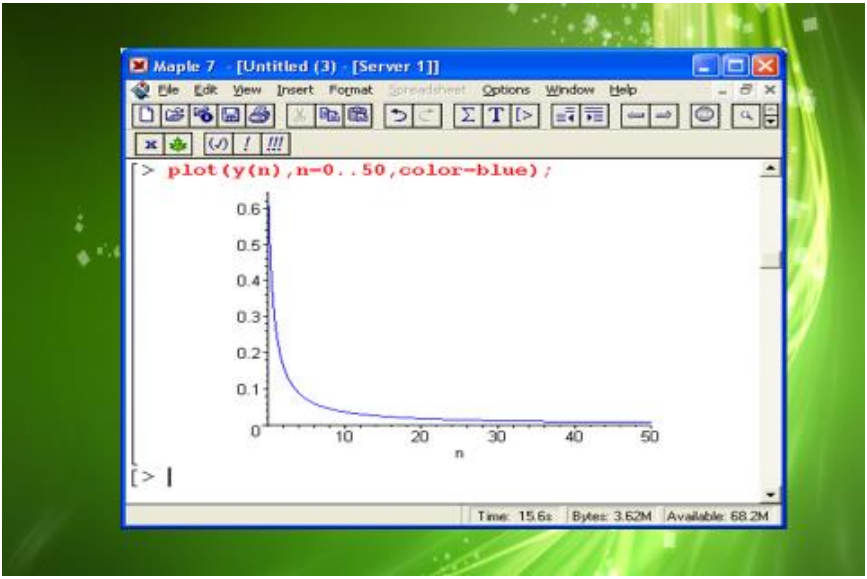

Графическая визуализация результатов вычисления
• построение графиков многих функций;
• различные типы осей (с линейным и логарифмическим масштабом);
• графики функций в декартовой и полярной системах координат;
• специальные виды графиков (точки массивов, векторные графики, диаграммы уровней и др.);
• системы координат, определяемые пользователем;
• графики, представляющие решения дифференциальных уравнений;
• графики трехмерных поверхностей с функциональной закраской;
• построение пересекающихся в пространстве объектов;
• задание пользователем окраски графиков;
• импорт графиков из других пакетов и программных систем;
• анимация графиков;
• создание и проигрывание анимационных файлов.

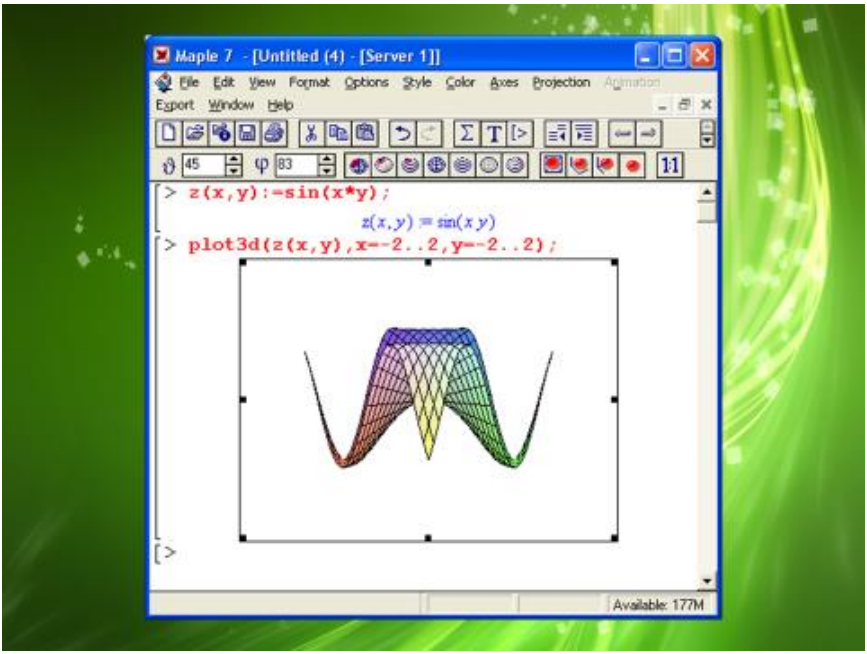
Далее рассмотрим примеры оформления и работы программы в версии 2007 года:





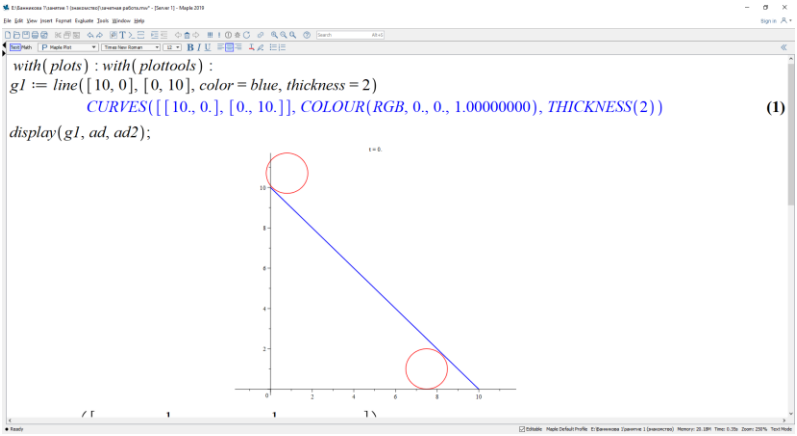
Графики в Maple 7:





2. Примеры работы программы с графикой на плоскости

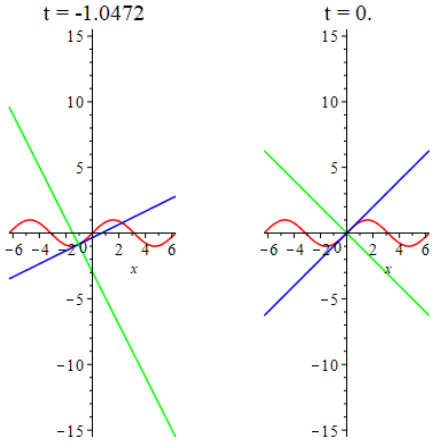
Теперь рассмотрим работу с графиками в Maple 19:



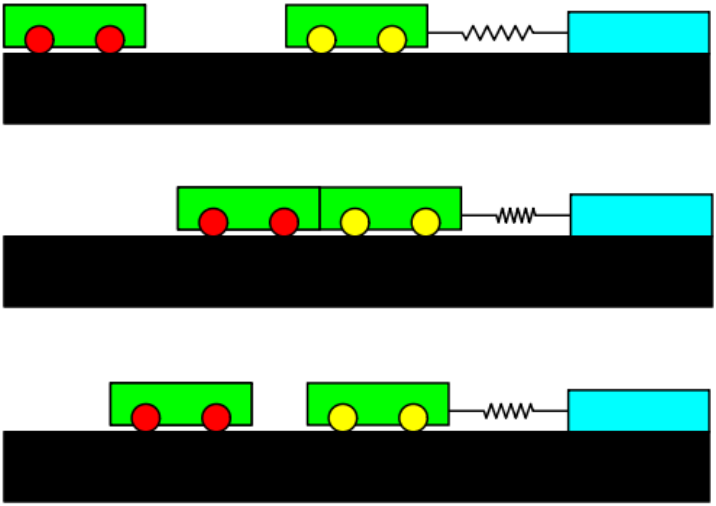
Как мы видим, изменился формат вводимых функций, они стали иметь более привычный нам вид.

С помощь графических возможностей пакета уже можно создавать обучающие программы с демонстрацией свойств фигур, физических объектов.

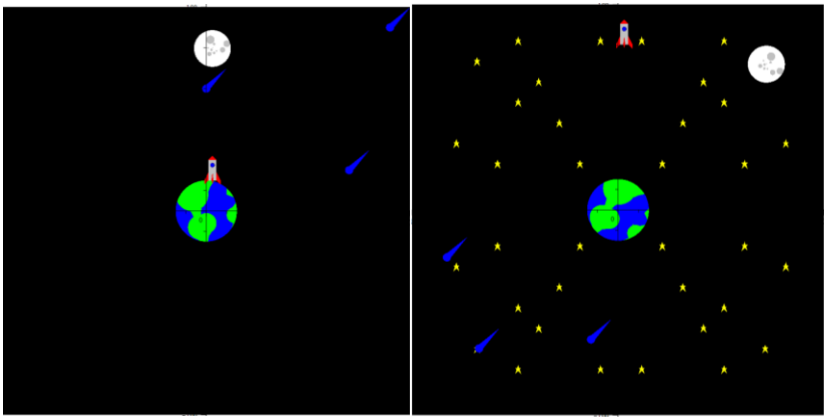
Движение касательной и нормали по синусоиде:



Сопротивление пружины под воздействием силы:



Появилась возможность создавать достаточно художественные картинки и мультипликацию:



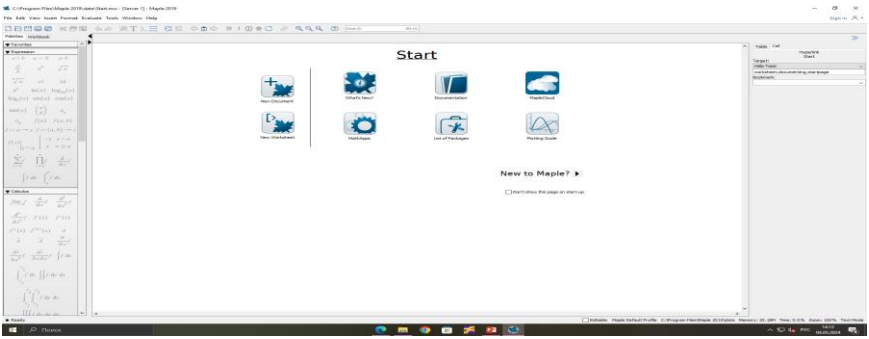
3. Интерфейс программы

Интерфейс программы приложения Maple представлен в виде графической оболочки, которая обеспечивает удобный доступ к функционалу программы. Основные элементы стартового окна включают в себя: меню с панелью инструментов, рабочую область, боковую панель. Навигация интегрирована в интерфейс программы и предлагает пользователям быстрый и легкий доступ к необходи-мым функциям. Благодаря расположению элементов управления и разумному использованию пространства, совершение операций становится простым и эффективным.

Ниже рассмотрим по отдельности каждый из элементов.

Меню программы.

В самом начале работы с приложением нас приветствует надпись “Start” в верхней части окна. Ниже расположено основное меню с различными категориями команд, где мы можем создать рабочий лист программы, в котором будем работать в дальнейшем, посмотреть обновления, ознакомиться с документацией, прочитать помощь.

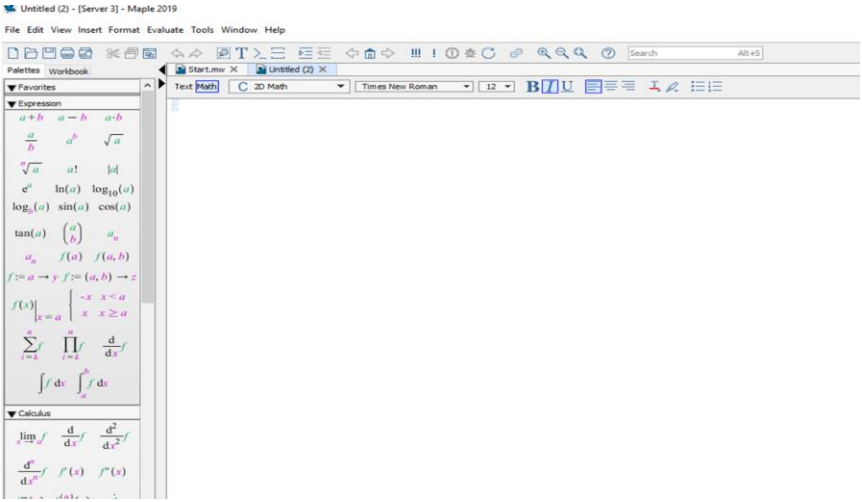


Здесь пользователь может найти все основные функции программы, такие как открытие и сохранение файлов, выполнение вычислений, настройка параметров, возможность быстро выполнить определенные действия, такие как создание нового документа, сохранение документа, изменение настроек и т. д.

Рабочая область.

После создания нового документа, перед нами открывается пустая область, где мы будем работать. Это основное окно приложения,

где пользователь может вводить и редактировать математические выражения, выполнять вычисления и строить графики. Здесь отображается результат выполнения заданных команд и вводимых пользователем формул.



Наиболее полные возможности управления предоставляет главное меню системы. Оно, как обычно, расположено непосредственно под строкой заголовка. Меню предоставляет доступ к основным операциям и параметрам пользовательского интерфейса системы. Ниже дан перечень меню, доступных при наличии открытого документа:

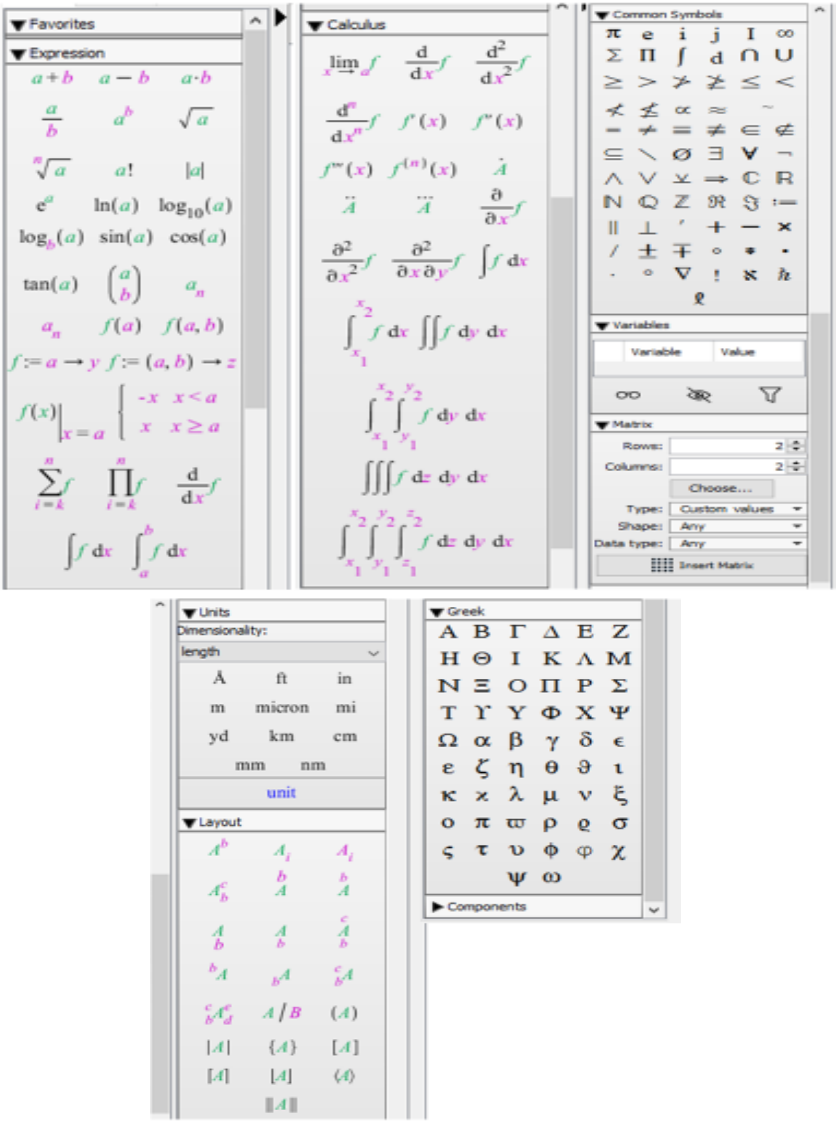
- File – работа с файлами и печатью документов;
- Edit – команды редактирование документа и операции с буфером обмена;
- View – управление видом пользовательского интерфейса;
- Insert – операции вставки;
- Format – операции задания форматов;
- Spreadsheet — операции задания таблиц;
- Options – задание параметров;
- Window – управление окнами;
- Help – работа со справочной системой.

Настройка графического интерфейса содержит: режимы просмотра документа; графические примитивы; выделение и преобразование объектов; управление масштабом просмотра объектов; копирование объектов; упорядочение размещения объектов; группировка объектов; соединение объектов.

Оформление текста подразумевает: виды текста: простой, фигурный текст; создание, редактирование, форматирование, предназначение; размещение текста вдоль кривой; редактирование геометрической формы текста.

Боковая панель.

Слева от рабочей области расположена боковая панель с различными вкладками. В ней находятся специализированные инструменты и окна, например, для работы с символьными выражениями, графиками, решением уравнений и т. д.



Панель включает в себя тригонометрические, логарифмические, показательные и др. функции. Данная вкладка позволяет быстро переключаться между различными функциональными модулями, не вводя вручную часто используемые математические символы.

Палитры ввода математических символов

- Список палитр Palettes в меню View.

Назначение некоторых знаков в палитрах:

- SYMBOL — ввод отдельных символов (греческих букв и некоторых математических знаков);
- EXPRESSION — ввод шаблонов математических операторов и операций;
- MATRIX — ввод шаблонов матриц разных размеров;
- VECTOR — ввод шаблонов векторов разных размеров и типов (векторы-столбцы или векторы-строки).
- **ВНИМАНИЕ!** Во избежании грубых ошибок при исполнении того или иного примера рекомендуется перед этим исполнить команду **restart** которая снимает определения со всех использованных ранее переменных и позволяет начать вычисления «с чистого листа».

Maple-язык и его синтаксис. Знаки алфавита.

- Язык Maple (или Maple-язык) – это одновременно и входной язык общения с Maple и язык ее программирования.

Алфавит Maple-языка содержит :

- 26 малых латинских букв (от a до z),
- 26 больших латинских букв (от A до Z),

- 10 арабских цифр (от 0 до 9) ,
- 32 специальных символа (арифметические операторы +, -, *, /, знак возведения в степень ^ и др.).
- Имеется пять пар альтернативных символов (означающих одно и тоже):

" и ** [и (|]и |) { и (* } и *)

Maple-язык и его синтаксис. Знаки алфавита.

- Специальные одиночные и составные знаки – элементы синтаксиса языка:

- % — системная переменная, хранящая результат предшествующей операции;
- : — фиксатор выражения, предотвращающий вывод результата вычисления в ячейку вывода;
- ; — фиксатор выражения, дающий вывод результата вычисления в ячейку вывода;
- # — указатель программного комментария;
- " — ограничитель строки (например, 'string');
- := — оператор присваивания (например, x:=5);
- : ; — пустой оператор;
- :: — указатель типа переменной (например, n::integer или z: -.complex);
- \ — знак обратного деления, который имеет множественные значения в зависимости от контекста (см. справку по этому знаку - backslash).

Работа с текстом в Maple очень похожа на работу в документе Word, но некоторая ограниченность присутствует, так же можно набирать текст различными шрифтами (но разных вариантов меньше, чем в Word), различного размера, жирный, курсив, шрифтами разного цвета, работа с таблицами и др., различное выравнивание текста. Можно выделить текст знаками #.

Практическое занятие 1

Задание 1. Написать эссе на свободную тему на 150–200 слов. Оформить текст: шрифт Times New Roman, цвет синий, 1,5 интервал, с применением 14-го размера шрифта, текст должен быть отформатирован по левому краю страницы, заголовок посередине.

Задание 2. Задать формулы:

$\int_1^2 \cos(x)\,dx;$ $\begin{cases} x_1 + x_2 + x_3 + x_4 = 1 \\ 2x_1 + 3x_2 + 4x_3 + 5x_4 = 2; \\ -2x_1 + 5x_2 - x_3 + x_4 = 0 \end{cases}$

$\left|\overrightarrow{AB}\right| = \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2};$

$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{pmatrix};$ $\lim\limits_{x \to 0} \frac{\sin(x)}{x}$; $\sum_{n=1}^{\infty} a_n b_n;$

Задание 3. Нарисовать треугольник ABC и его параметры: высоту BH,

биссектрису BL и медиану BM, обозначить их на рисунке, подписать точки.

Задание 4. Создать презентацию и включить в нее материал из предыдущих заданий. На последнем слайде сделать гиперссылку на первый слайд.

Лекция 2. Оформление графика функции

В этой теме мы познакомимся с тем, как можно вводить функцию для построения ее графика и работы с ней, а также какие графические приложения Maple существуют.

Для того, чтобы не только строить график, но и производить вычисления с данной функцией, необходимо задать ее функционально:

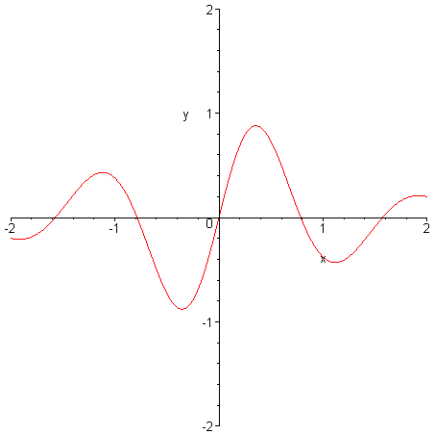
$$f:=x->sin(4*x)/(x\text{^}2+1);$$

Поставим в конце знак «*;*» нажмем кнопку Enter и программа отреагирует следующей записью, дав нам понять, что мы все правильно записали:

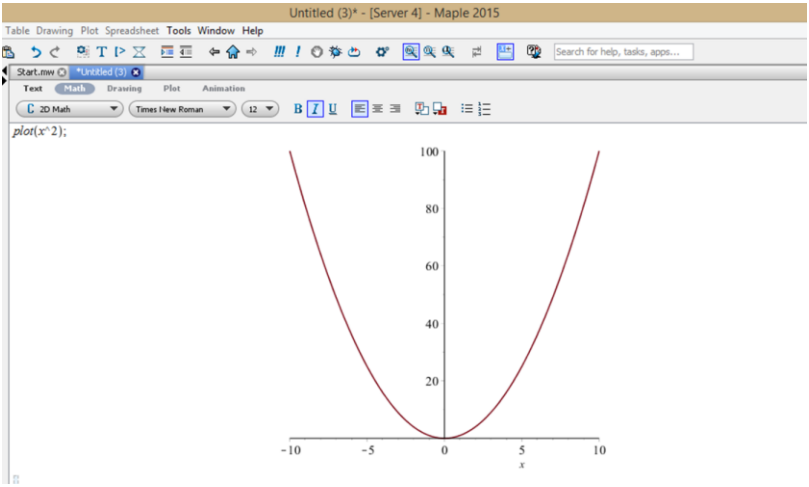
$$f := x \to \frac{\sin(4\,x)}{x^2+1}$$

Если в конце записи поставим «:», то действие будет сделано в компьютере, но на экране ничего отображаться не будет. Далее нарисуем график этой функции с помощью основной графической команды *plot*:

$$plot(f(x),x=-2..2,y=-2..2);$$



Но можно рисовать графики непосредственно вписывая функцию в команду *plot*: *plot(x^2);*



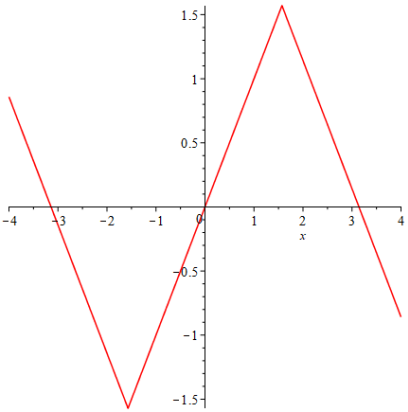
Пример задания одного и того же графика сложной функции тремя различными способами:

1 вариант: *plot*(*arcsin*(*sin*(*x*)),*x*= −4..4);

2 вариант: *h*:=*arcsin*(*sin*(*x*)):*plot*(*h*,*x*= −4..4);

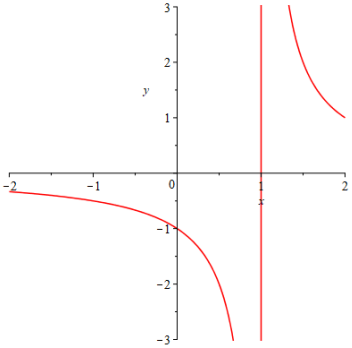
3 вариант: *f*:=*x*−>*sin*(*x*):*g*:=*arcsin*(*f*(*x*)):*plot*(*g*,*x*= −4..4);

График в каждом случае будет одинаковый:



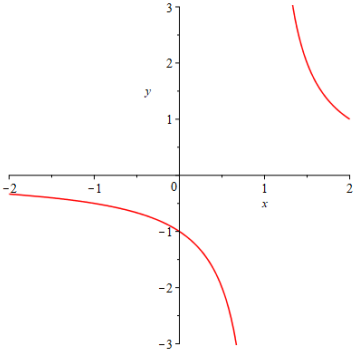
Следующая особенность программы, связана с графиками разрывных функций: по умолчанию такие графики Maple дополняет до непрерывных:

plot(*1/*(*x*−*1*), *x* = −*2* .. *2*, *y* = −*3* .. *3*);

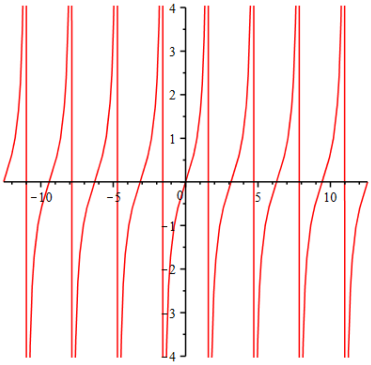


Чтобы задать правильно, добавляем команду DISCONT:

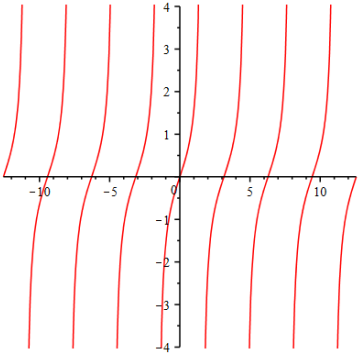
plot(*1/*(*x*−*1*), *x* = −*2* .. *2*, *y* = −*3* .. *3*, *discont*=*true*);



plot(*tan*(*x*),*x*= −*4***Pi*..*4***Pi*,*y*= −*4*..*4*);

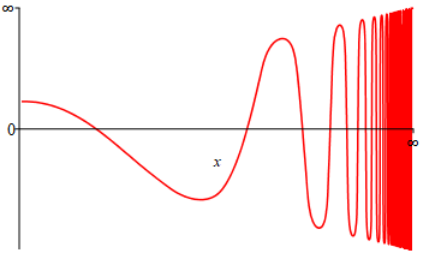


plot(*tan*(*x*),*x*= −4*Pi..4*Pi,*y*= −4..4,*discont*=*true*);

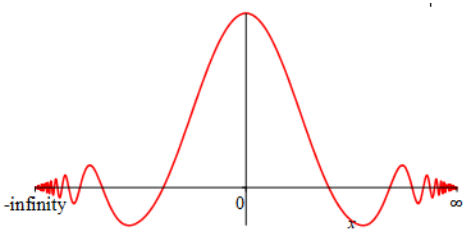


Следующая интересная особенность программы Maple состоит в том, что можно нарисовать поведение графика функции на бесконечности, как одностороннюю, так и в обе стороны:

f:=*x*−>*x***sin*(*x*):
plot(*f*(*x*),*x*=2..*infinity*);



g:=sin(x)/x:
plot(g(x),x= −infinity..infinity);



Своеобразный синтаксис стандартных функций в Maple можно увидеть в следующих таблицах:

Функция	Синтаксис Maple	Функция	Синтаксис Maple
$\sin(x)$	`sin(x)`	$\mathrm{sh}(x)$	`sinh(x)`
$\cos(x)$	`cos(x)`	$\mathrm{ch}(x)$	`cosh(x)`
$\mathrm{tg}(x)$	`tan(x)`	$\mathrm{th}(x)$	`tanh(x)`
$\sec(x)$	`sec(x)`	$\mathrm{sech}(x)$	`sech(x)`
$\mathrm{cosec}(x)$	`csc(x)`	$\mathrm{cosech}(x)$	`csch(x)`
$\mathrm{ctg}(x)$	`cot(x)`	$\mathrm{cth}(x)$	`coth(x)`

Функция	Синтаксис Maple	Функция	Синтаксис Maple
$\arcsin(x)$	`arcsin(x)`	$\mathrm{arcsh}(x)$	`arcsinh(x)`
$\arccos(x)$	`arccos(x)`	$\mathrm{arcch}(x)$	`arccosh(x)`
$\mathrm{arctg}(x)$	`arctan`	$\mathrm{arcth}(x)$	`arctanh(x)`
$\mathrm{arcsec}(x)$	`arcsec(x)`	$\mathrm{arcsech}(x)$	`arcsech(x)`
$\mathrm{arccosec}(x)$	`arccsc(x)`	$\mathrm{arccosech}(x)$	`arccsch(x)`
$\mathrm{arcctg}(x)$	`arccot(x)`	$\mathrm{arccth}(x)$	`arccoth(x)`

Функция	Синтаксис Maple	Функция	Синтаксис Maple
e^x	`exp(x)`	$\sqrt{x}$	`sqrt(x)`
$\ln(x)$	`ln(x)` **или** `log(x)`	$\lvert x\rvert$	`abs(x)`
$\log_{10}(x)$	`log10(x)`	$\mathrm{sgn}(x)$	`signum(x)`
$\log_a(x)$	`log[a](x)`	$n!$	`n!`

Познакомимся с основными опциями оформления графика функции:

axes	Определяет тип отображаемых осей координат. Эта опция может принимать следующие значения: NORMAL — обычные оси координат, пересекающиеся в точке начала координат (0,0); BOXED — график заключен в прямоугольник с нанесенными шкалами по нижней и левой вертикальной граням; FRAME — оси с точкой пересечения в левом нижнем углу рисунка; NONE — оси не отображаются
axesfont	Задает шрифт для надписей под засечками вдоль осей координат. Значение этой опции аналогично значению опции font
color	Задает цвета кривых, отображаемых на график. В качестве значения этой опции может выступать одно из зарезервированных значений цвета в Maple: aquamarine, black, blue, navy, coral, cyan, brown, gold, green, gray, grey, khaki, magenta, maroon, orange, pink, plum, red, sienna, tan, turquoise, violet, wheat, white и yellow. Можно также определить и собственный цвет, соответствующий смешению заданных частей красного, зеленого и синего цветов. Это осуществляется с помощью следующей команды macro(palegreen= COLOR(RGB,.5607,.7372,.5607)), где palegreen — имя константы нового цвета, в котором красный составляет 0.5607 части, зеленый 0.7372 и синий 0.5607. В дальнейшем это имя можно использовать для задания цвета аналогично именам встроенных цветов
coords	По умолчанию при выводе как явно заданной функции, так и параметрически заданной функции используется декартовая система координат (cartesian), т. е. задаваемое уравнение кривой рассматривается именно в этой системе координат. Данная опция меняет тип системы координат. Возможные значения: bipolar, cardiod, cassinian, elliptic, hyperbolic, invcassinian, invelliptic, logarithmic, logcosh, maxwell, parabolic, polar, rose и tangent, описание которых можно получить в справочной системе Maple с помощью команды ?coords. Здесь отметим только, что значение polar задает полярную систему координат

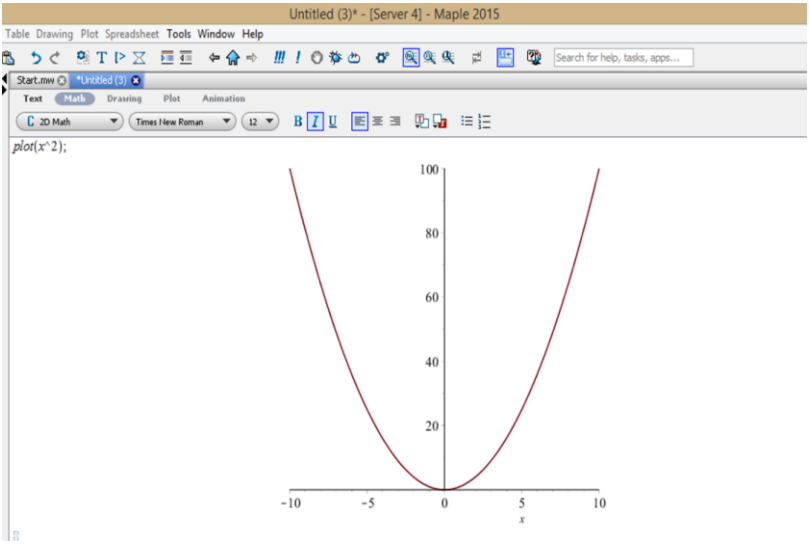
Опция	Описание
discont	Установка значения этой опции, равной true, приводит к тому, что Maple первоначально вызывает команду discont(), которая определяет промежутки непрерывности функции, а затем на них рисуются непрерывные участки графика функции. Значение по умолчанию false
filled	Установка значения данной опции равным true приводит к тому, что область, ограниченная графиком функции и горизонтальной осью *x*, закрашивается заданным в опции color цветом
font	Задает шрифт для вывода текста на рисунке. Значение опции задается в виде списка [семейство, стиль, размер]. Параметр семейство задает гарнитуру шрифта: TIMES, COURIER, HELVETICA или SYMBOL. Параметр стиль определяет стиль шрифта: для гарнитуры TIMES возможные значения ROMAN, BOLD, ITALIC или BOLDITALIC, для гарнитур COURIER и HELVETICA стиль можно опустить или задать BOLD, OBLIQUE или BOLDOBLIQUE, для шрифта SYMBOL стиль не задается. Последний параметр размер задает размер шрифта в пунктах (points) (один пункт приблизительно равен 1/72 дюйма)
labels	Задание названий осей координат в виде списка [x,y]. Параметры x и y задаются в виде строк и соответствуют отображаемым названиям горизонтальной и вертикальной осей. По умолчанию принимают значения имени независимой переменной и имени функции
labeldirections	Эта опция определяет направление отображения названий осей и задается в виде списка [x,y], элементы которого могут принимать одно из двух значений HORISONTAL или VERTICAL и определяют расположение надписей осей координат: горизонтально или вертикально. Умалчиваемое значение HORISONTAL
labelfont	Задает параметры шрифта, которым отображаются названия осей координат. Значение этой опции аналогично значению опции font
legend	Задает отображение легенды для нескольких кривых на одном графике в виде списка, в котором *i*-й строковый элемент соответствует *i*-й кривой графика
linestyle	Определяет тип линии графика. Значением этой опции является целое число n. При n=0 тип линии соответствует умалчиваемому типу для используемого устройства отображения (обычно сплошная линия), значение 1 соответствует сплошной линии, значение 2 — отображению линии точками, 3 — пунктиром и 4 — штрихпунктиром

Опция	Описание
numpoints	Определяет минимальное число вычисляемых точек, по которым строится график (значение по умолчанию равно 50)
resolution	Определяет горизонтальное разрешение дисплея в пикселах на дюйм и используется в качестве критерия для завершения адаптивного алгоритма отображения (значение по умолчанию равно 200)
sample	Определяет список значений параметров, который используется для "пробного" отображения кривой. Отключение адаптивного алгоритма вычисления точек кривой позволяет явным образом управлять отображением кривой
scaling	Задает масштаб, в котором отображается график. Если значение этой опции равно CONSTRAINED, то это соответствует заданию абсолютных значений по осям координат, т. е. одна единица измерения по оси независимой переменной равна одной единице измерения по оси значений функции. Значение по умолчанию равно UNCONSTRAINED, и это соответствует тому, что оси растягиваются таким образом, чтобы их размеры соответствовали размерам графического окна вывода
style	Задает отображение графика функции линиями (значение опции равно LINE) или точками (значение опции равно POINT). Значения этой опции, равные PATCH и PATCHNOGRID, применяются только тогда, когда выводится замкнутый многоугольник (графическая структура POLYGONS). В этом случае его внутренняя область закрашивается цветом, установленным в опции color, причем в случае значения PATCHNOGRID его граница не отображается. Если в графическом выводе нет замкнутых многоугольников, то действие этих значений данной опции соответствует значению LINE
symbol	Определяет тип символа, которым помечаются точки графика функции при style=POINT. Может принимать следующие значения: BOX для □, CROSS для +, CIRCLE для ○, POINT для • (точка) и DIAMOND для ◊
symbolize	Задает размер символа в пунктах. Его значение может быть любое натуральное число. По умолчанию используются символы размером 10 пунктов. Действие этой опции не распространяется на символ POINT
thickness	Задает толщину линии графика. Значение является целым числом и изменяется от 0 до 3, соответствуя изменению толщины линии от самой тонкой до самой жирной

Опция	Описание
tickmarks	Определяет число точек, не менее которого должно быть помечено по горизонтальной и вертикальной оси координат. Значение задается в виде списка [n,m]. Для каждой из осей можно определить список помечаемых точек
title	Определяет строку, которая выводится как заголовок рисунка. По умолчанию заголовок не выводится. В строке можно использовать специальные комбинации символов. Например, \n осуществляет перевод на новую строку, формируя тем самым многострочный заголовок
titlefont	Определяет шрифт для заголовка рисунка. Значение этой опции аналогично значению опции font
xtickmarks	Задает число точек, не менее которого должно быть помечено на горизонтальной оси. Значение этой опции может быть целым числом или списком значений координат точек горизонтальной оси, которые должны быть помечены. Список может состоять из уравнений, левые части которых определяют координаты помечаемых точек, а правые задают в обратных кавычках отображаемый текст, например, [0=`0.`,0.5=`1/2`,1=`1.`]
ytickmarks	Задает число точек, не менее которого должно быть помечено на вертикальной оси. Значение этой опции может быть целым числом или списком значений координат точек вертикальной оси, которые должны быть помечены. Список может состоять из уравнений, левые части которых определяют координаты помечаемых точек, а правые задают в обратных кавычках отображаемый текст, например, [0=`0.`,0.5=`1/2`,1=`1.`]

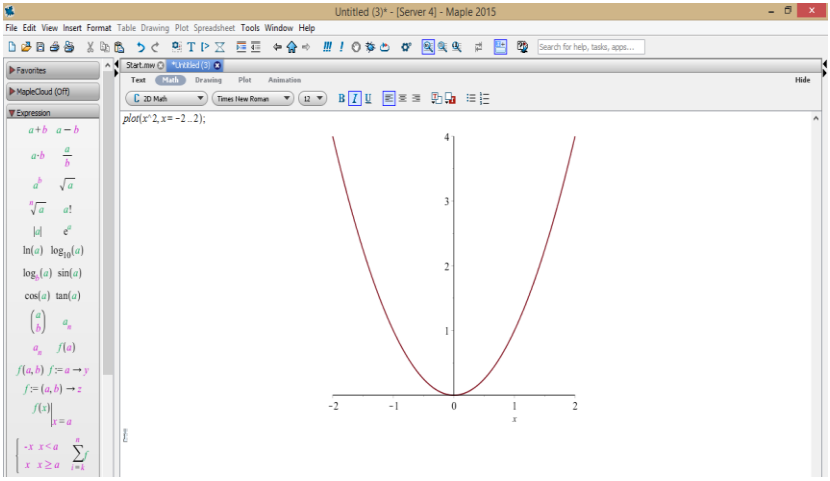
Рассмотрим примеры применения этих опций.

Возьмем знакомое нам уравнение параболы. Открываем новый файл и в самом начале пишем plot(x^2), где plot – это обязательное обращение к программе, в скобочках мы указываем нашу функцию, ставим точку с запятой и при нажатии клавиши Enter получаем результат:

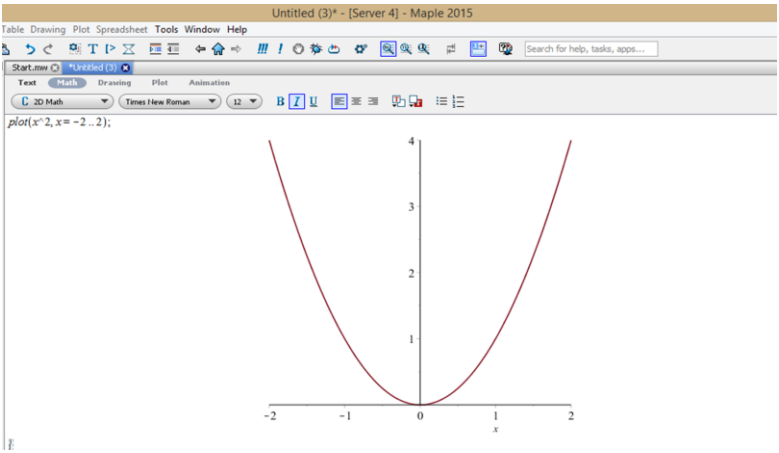


Ограничим нашу функцию в пределах x от −2 до 2. Для этого в скобочках через запятую напишем условие:

plot(*x*^2, *x* = −2 .. 2);

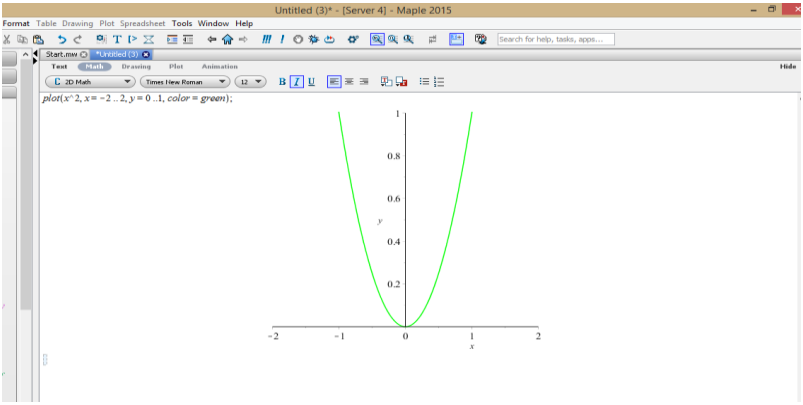


Ограничим нашу функцию ещё и по y.
Всё также в скобочках, через запятую, перечисляем условия:
plot(x^2, x = −2 .. 2, y=0..1);



Теперь добавим цвет. Цвета в Maple пишутся на английском языке и к ним идёт обращение color =... Для удобства выпишем несколько цветов, которые можно использовать: red – красный, blue – синий, purple – фиолетовый и так далее. Пусть график нашей функции будет зелёным:

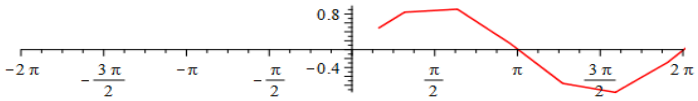
plot(x^2, x = −2 .. 2, y=0..1,color=green);



Чтобы сделать наш график более понятным для стороннего пользователя, Maple позаботился об удобстве оформления.

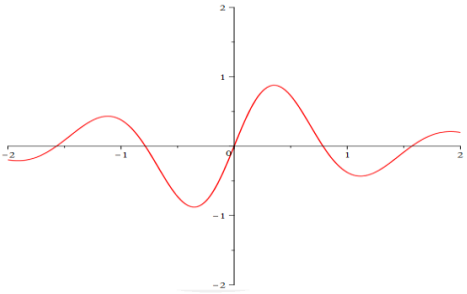
Выберем отдельные точки, благодаря которым построим график функции:

plot(*sin*(*x*), *sample* = [*1*, *2*, *3*, *4*, *5*, *6*, *7*, *8*, *9*], *adaptive* = *false*, *scaling* = *constrained*);



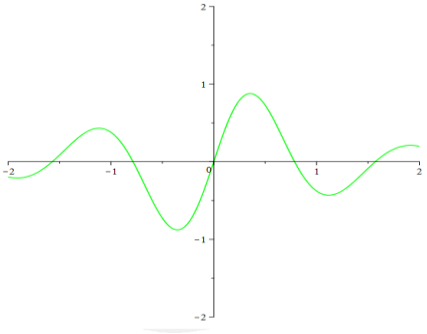
Далее для примера возьмем уже знакомую нам функцию:

f := *x*− > *sin*(*4* ∗ *x*)/(*x*^*2* + *1*): *plot*(*f* (*x*), *x* = −*2*..*2*, *y* = −*2*..*2*);



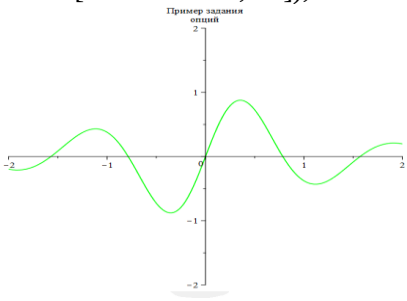
Изменим цвет:

plot(*f* (*x*), *x* = −*2*..*2*, *y* = −*2*..*2*, *color* = *green*);



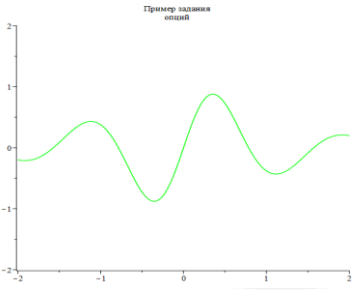
Теперь, когда мы построили график, давайте подпишем его. Добавим заголовок:

plot(*f*(*x*), *x* = −2..2, *y* = −2..2, *color* = *green*, *title* = *Пример задания\nопций*, *titlefont* = [*HELVETICA*, *12*]);

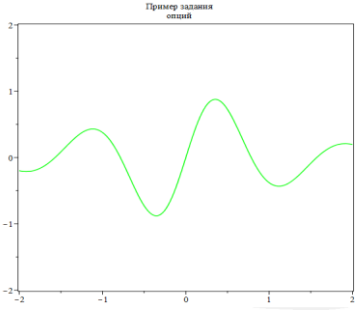


Сейчас поработаем над осями координат нашего графика:

plot(*f*(*x*), *x* = −2..2, *y* = −2..2, *color* =*green*, *title* = *Пример задания\nопций*, *titlefont* = [*HELVETICA*, *12*], *axes* = *FRAME*);

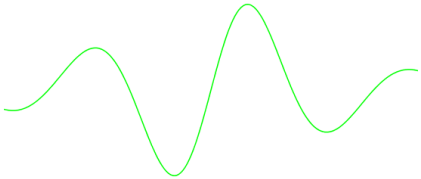


plot(*f*(*x*), *x* = −2..2, *y* = −2..2, *color* =*green*, *title* = *Пример задания\nопций*, *titlefont* = [*HELVETICA*, *12*], *axes* = *BOXED*);



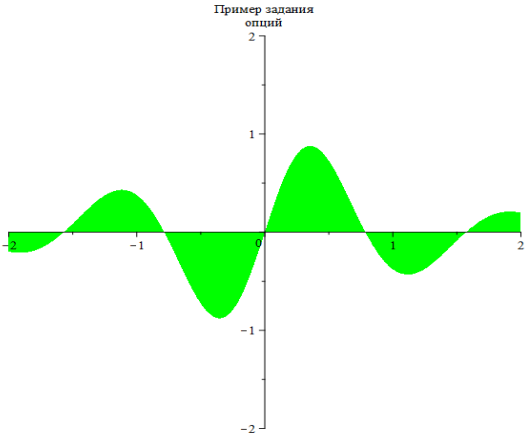
Мы также можем убрать все оси, если они нам мешают:

plot(*f*(*x*), *x* = −2..2, *y* = −2..2, *color* = *green*, *title* = *Пример задания\nопций*, *titlefont* = [*HELVETICA*, *12*], *axes* = *NONE*);



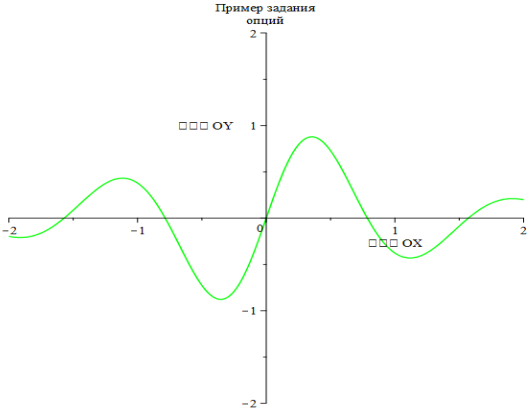
Допустим нам нужно закрасить область, ограниченной графиком и осью Ox заданным цветом. Используем функцию filled – закрашивание области, команду transparency для насыщенности закраски:

plot(*f*(*x*), *x* = −2..2, *y* = −2..2, *color* = *green*, *title* = *Пример задания\nопций*, *titlefont* = [*HELVETICA*, *12*], *axes* = *NORMAL*, *filled*, transparency = 0.5);



Теперь назовем оси, labels – задание названия осей:

plot(*f*(*x*), *x* = −2..2, *y* = −2..2, *color* = *green*, *title* = *Пример задания\nопций*, *titlefont* = [*HELVETICA*, *12*], *axes* = *NORMAL*, *labels* = [*"OX "OY "*]);



Функция textplot − подписи и надписи к графику работает только при подключении встроенного графического пакета PLOTS:

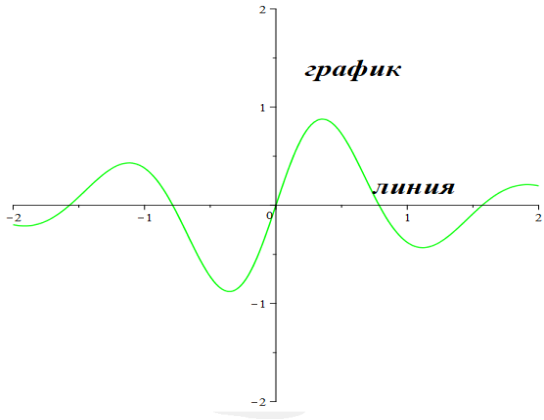
with(*plots*)*:*

r1 := *textplot*([*0.2, 1.4, ‘‘график”*], *front* = [*TIMES*, *BOLD*, *24*], *align* = *RIGHT*)*:*

r2 := *textplot*([*1.4, 0.2, ‘‘линия”*], *front* = [*TIMES*, *BOLD*, *24*], *align* = *LEFT*)*:*

r3 := *plot*(*f*(*x*), *x* = −*2..2*, *y* = −*2..2*, *color* = *green*)*:*

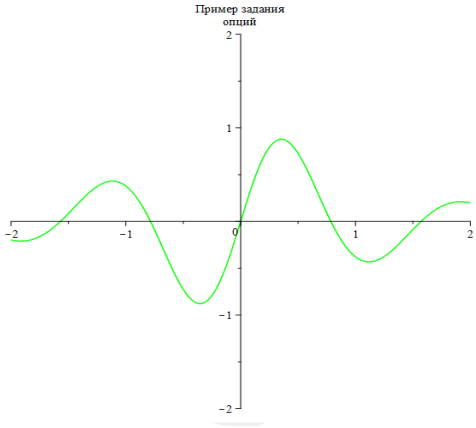
display(*r1*, *r2*, *r3*)*;*



Также у нас есть стиль линии; linestyle – тип линии графика:

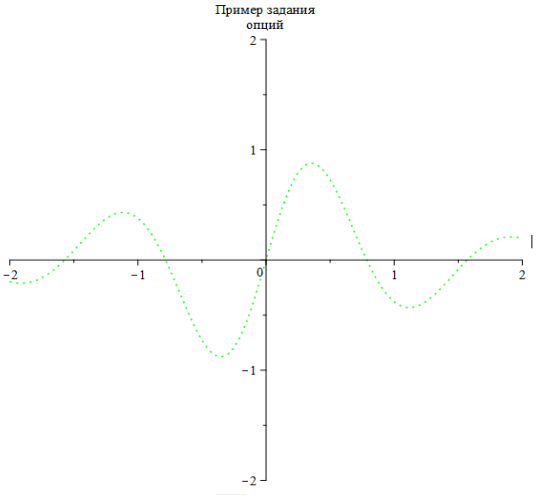
plot(f(x), x = −2..2, y = −2..2, color = green, title = Пример задания\nопций, titlefont = [HELVETICA, 12], axes = NORMAL, linestyle = 0);

Получается сплошная линия.



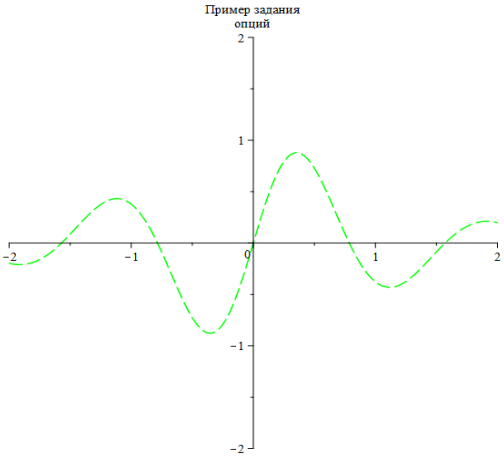
Можем нарисовать линию с помощью точек:

plot(f(x), x = −2..2, y = −2..2, color = green, title = Пример задания\nопций, titlefont =[HELVETICA, 12], axes = NORMAL, linestyle =2);



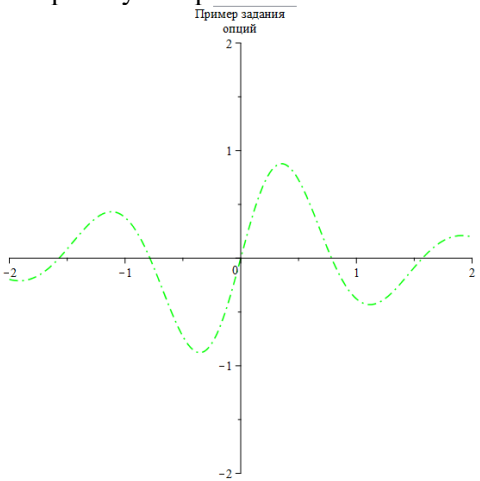
plot(*f*(*x*), *x* = −2..2, *y* = −2..2, *color* = *green*, *title* = *Пример задания\nопций*, *titlefont* =[*HELVETICA*, *12*], *axes* = *NORMAL*, *linestyle* = *3*);

Получаем пунктир.



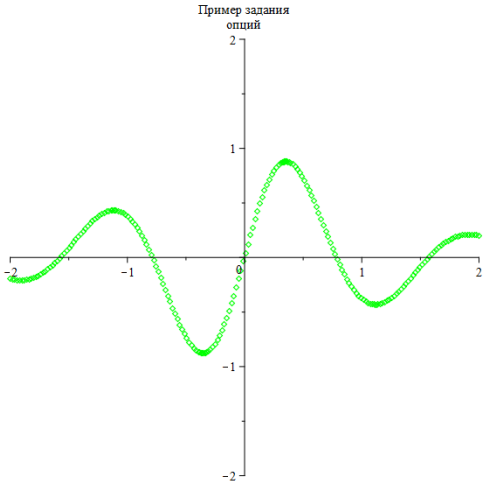
plot(*f*(*x*), *x* = −2..2, *y* = −2..2, *color* = *green*, *title* = *Пример задания\nопций*, *titlefont* =[*HELVETICA*, *12*], *axes* = *NORMAL*, *linestyle* = *4*);

Получаем штрих-пунктир.



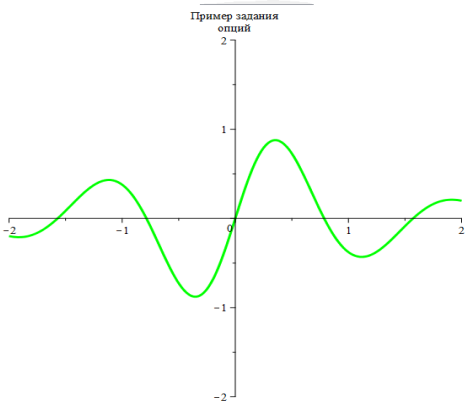
Можем задать стиль «точка»:

plot(*f*(*x*), *x* = −2..2, *y* = −2..2, *color* = *green*, *title* = *Пример задания\nопций*, *titlefont* =[*HELVETICA*, *12*], *axes* = *NORMAL*, *style* = *POINT*);



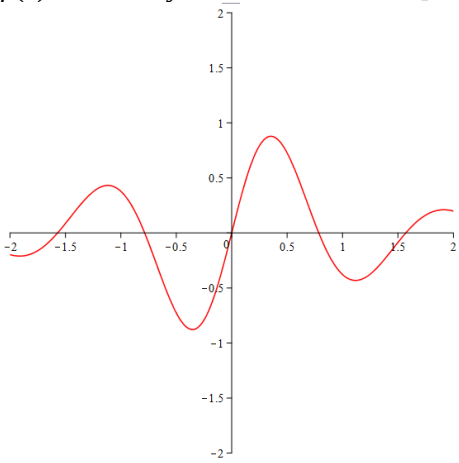
Чтобы выделить линию, мы можем указать её толщину:

plot(*f*(*x*), *x* = −2..2, *y* = −2..2, *color* = *green*, *title* = *Пример задания\nопций*, *titlefont* = [*HELVETICA*, *12*], *axes* = *NORMAL*, *thickness* = *3*);



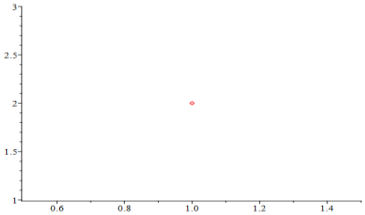
Можем задать необходимый нам масштаб, указав нужное количество чисел на осях:

$$plot(f(x), x=-2..2, y=-2..2, tickmarks=[8,8]);$$



Для выполнения практического задания нам понадобится задание точки:

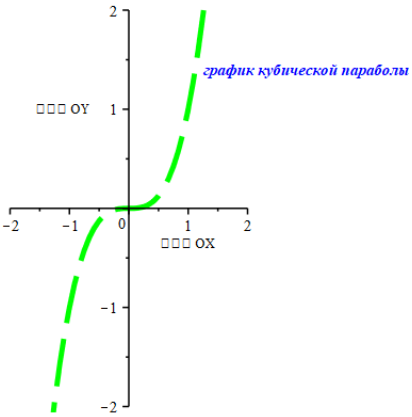
$$plot([[1, 2]], style = POINT);$$



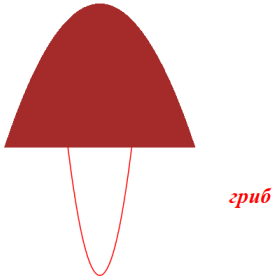
Точку можно задавать в виде различных фигур с заполнением и без, изменять ее размер, но об этом на следующей лекции.

Практическое задание 2

Задание 1. Нарисовать график y=x^3 с соответствующим оформлением.

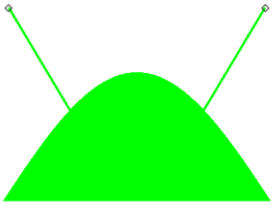


Задание 2. Нарисовать гриб, подписать. Какие простейшие функции вы использовали в работе?



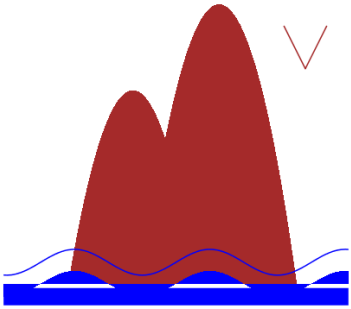
гриб

Задание 3. Нарисовать НЛО, подписать. Какие простейшие функции вы использовали в работе?



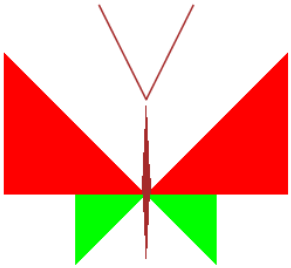
НЛО

Задание 4. Нарисовать горы, чайку, волны, подписать. Какие простейшие функции вы использовали в работе?



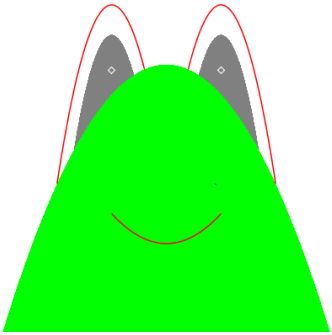
ГОРЫ – ВОЛНЫ – ЧАЙКА

Задание 5. Нарисовать бабочку, подписать. Какие простейшие функции вы использовали в работе?



бабочка

Задание 6. Нарисовать лягушку. Какие простейшие функции вы использовали в работе?



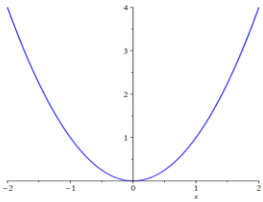
Лекция 3. Построение графиков различных функций

Для дальнейшей работы нам понадобится подключить графический пакет with(plots).

Теперь мы можем рассмотреть графики, когда функции заданы не только в явном виде, но и в других вариантах.

Вспомним оформление графиков функций, заданных явно. Зададим параболу:

plot(*x^2, x = −2 .. 2, color = blue*);



Следующими рассмотрим функции заданные параметрически.

Пусть заданы две функции x(t), y(t); при этом переменная t называется параметром.

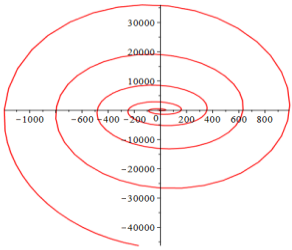
Тогда говорят, что y как функция от x задана параметрически.

Пример: $\begin{cases} x = a\ cos\ t, \\ y = a\ sin\ t, \end{cases}$ $0 \le t \le 2 \cdot Pi,$ это параметрические уравнения окружности: $x^2 + y^2 = a^2$.

Объявление параметрической функции задается практически аналогично функции в явном виде:

plot([*t^2*cos*(*t*), *t^3*sin*(*t*) + 2, t = 0 .. 36*]);

Можно заметить появление квадратных скобок, что отличает два вида функций друг от друга. Получаем рисунок:



Далее изобразим известное уравнение окружности, но со смещением по оси y:

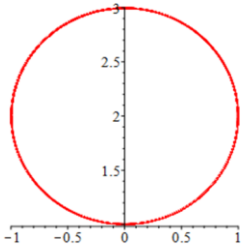
plot([*cos*(*t*), *sin*(*t*) + *2*, *t* = *0* .. *2*Pi*]);

В некоторых версиях Maple вместо обещанной окружности мы можем увидеть эллипс. Дело в том, что по умолчанию во всех графических командах используется значение UNCONSTRAINED параметра scaling, и график растягивается по осям таким образом, чтобы полностью заполнить отводимое под него пространство на рабочем листе, что приводит к несоответствию единиц измерения по горизонтальной и вертикальной осям.

Исправить данную неприятность можно с помощью команды scaling=CONSTRAINED:

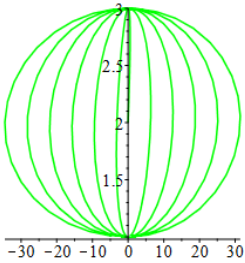
plot([*cos*(*t*), *sin*(*t*) + *2*, *t* = *0* .. *2*Pi*], *scaling* = *constrained*);

И на выходе получим симпатичную окружность:



Также, мы можем задавать цвет, всё также через запятую и специально не напишем опцию scaling:

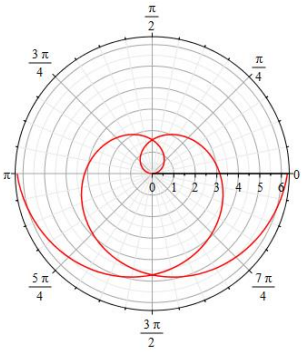
plot([*t*cos*(*t*), *sin*(*t*)+*2*, *t* = *0* .. *36*], *color* = *green*);



Функции заданные полярно.

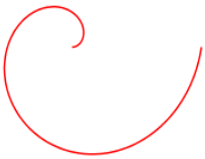
Когда функция задаётся через полярные координаты, мы обращаемся к команде polarplot. Переменную можно использовать любую (и даже оставить x), но мы сделаем стандартную φ:

polarplot(*phi, phi = −2*Pi .. 2*Pi, scaling = constrained*);



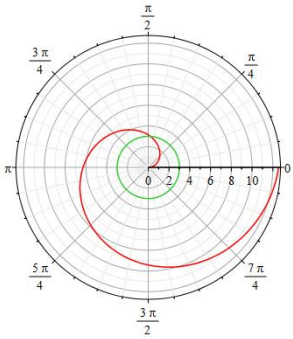
Рассмотрим известную спираль Архимеда, систему координат удалим:

polarplot (*phi, phi=0..2*Pi, scaling= constrained, axes=NONE*);

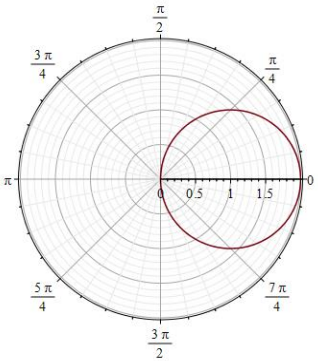


Можно задать сразу несколько графиков:

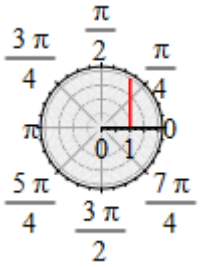
polarplot([*2*phi, 3*], *phi = 0 .. 2*Pi, scaling = constrained*);



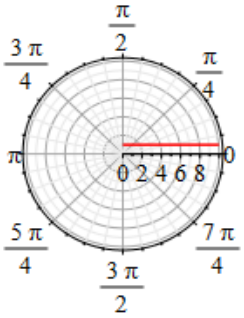
Как задать в полярной системе координат окружность?
polarplot(*2*cos*(*phi*),*phi=0..Pi*,*scaling=constrained*);
Это окружность $(x-1)^2+y^2 = 1$:



Далее зададим отрезки, сначала вертикальный:
polarplot(*1/cos*(*phi*), *phi = 0 .. (1/3)*Pi, scaling = constrained*);



А теперь горизонтальный:
polarplot(*1/sin*(*phi*), *phi =0 .1 .. (1/2)*Pi, scaling = constrained*);



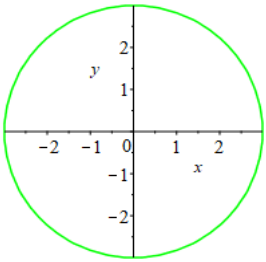
Как видим, приходиться использовать определенные уловки в виде специальных промежутков переменной.

Неявно заданные функции.

Когда у нас есть уравнение, к которому надо построить график, но выражать функцию y через привычную нам переменную x затруднительно, то мы пользуемся помощью команды implicitplot. Для того чтобы команда работала необходимо задавать интервалы на обе переменные.

Давайте реализуем уравнение окружности: $x^2 + y^2 - 9 = 0$, в программу мы запишем:

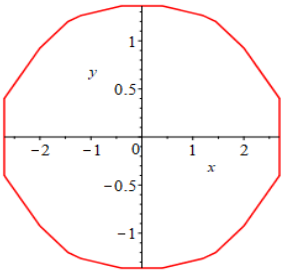
implicitplot(*x^2+y^2−9 = 0, x = −5 .. 5, y = −5 .. 5, color = green*);



Конечно, мы могли выразить уравнение и записать в кодовую строку y=sqrt(9−x^2), нам даже не нужно было бы подключать пакет with(plots), достаточно знакомого нам plot, но в этом случае получили только половину окружности.

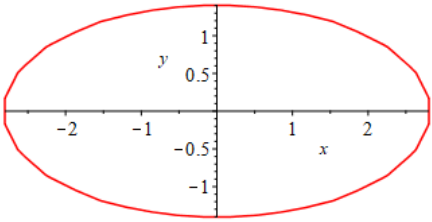
Давайте чуть усложним нашу задачу и реализуем вот такое уравнение: $x^2 + 4 \cdot y^2 = 8$ – эллипс:

implicitplot(*x^2+4*y^2 = 8, x = −10 .. 10, y = −10 .. 10*);



Для правильности графика сгладим углы командой *grid* (количество точек графика по переменным) и выравним масштаб:

implicitplot(*x^2+4*y^2 = 8, x = −10 .. 10, y = −10 .. 10, grid = [60, 60], scaling = constrained*);

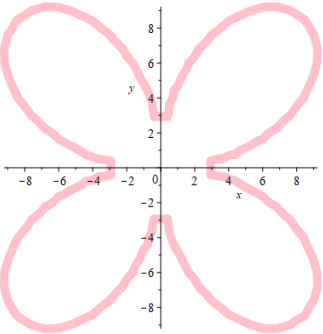


В неявном виде строятся очень необычные графики.

Как насчёт вот такого симпатичного уравнения, заданного в неявном виде:

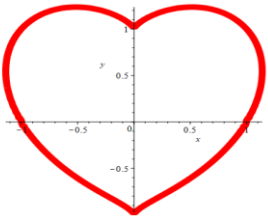
implicitplot((*x^2+y^2*)^3-(*4*144*)**x^2*y^2 = 0, x = -10 .. 10, y = -10 .. 10, thickness = 10, color = pink*);

Здесь мы добавили толщину линии, к которой обращаемся через thickness, и на выходе получаем красавицу-бабочку:



Если вам также понравилась наша бабочка, то давайте реализуем математическую валентинку:

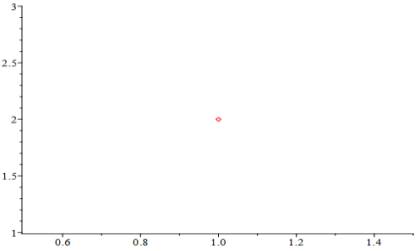
implicitplot((*x^2+y^2−1*)^3−*x^2*y^3 = 0, x = −1.25 .. 1.25, y = −1.25 .. 1.25, grid = [100, 100], thickness = 10*);



И снова опция grid поможет нам сгладить углы и очертить рисунок, делая его более симпатичным на вид.

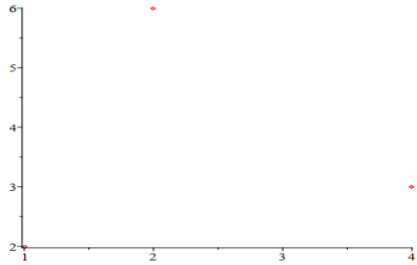
Вернемся к заданию точки:

plot([[*1, 2*]], *style = POINT*);



Можно одной командой задать сразу несколько точек. Перечисляем координаты наших точек через запятую:

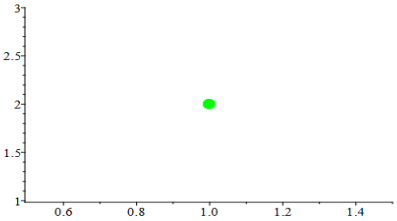
plot([[*1,2*],[*4,3*],[*2,6*]],*style=POINT*);



По умолчанию точка имеет вид ромба и пустая внутри.

Если нам нужно выделить какую-то точку, показать её более объемной, то используем solidCIRCLE (закрашенная точка, форма окружность) и укажем её размер с помощью опции symbolsize:

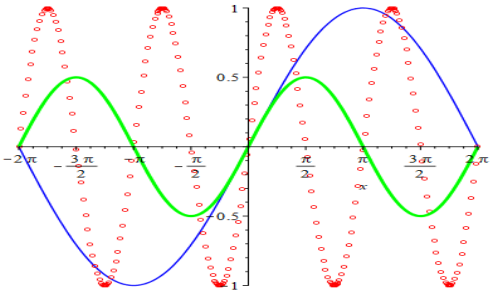
plot([[*1, 2*]], *style = POINT, symbol = solidCIRCLE, symbolsize = 26, color = green*);



Очень часто нам приходится сравнивать один график с другим. Определить какой промежуток лучше подходит для нашей математической задачи. Мы зададим сразу три графика, при этом один из них будет разрывным, созданным с помощью движения точки:

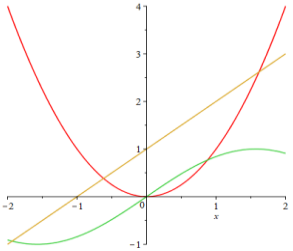
*f1 := sin((1/2)*x): f2 := sin(2*x): f3 := (1/2)*sin(x):*

plot([f1(x), f2(x), f3(x)], x, color = [blue, red, green], style = [line, point], symbol = circle, thickness = [1, 0, 3]);



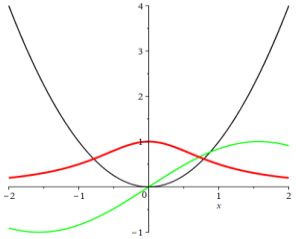
Попробуем создать другой рисунок из известных любому школьнику уравнений, цвет программа задает по умолчанию:

plot([x^2, sin(x), 1+x], x = −2 .. 2);



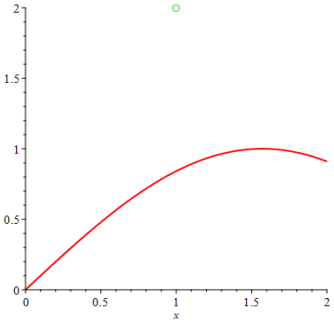
Укажем цвет и толщину линии для каждого графика:

plot([*x*^2, *sin*(*x*), *1/*(*x*^2+1)], *x* = −2 .. 2, *color* = [*black*, *green*, *red*], *thickness* = [*1*, *2*, *3*]);



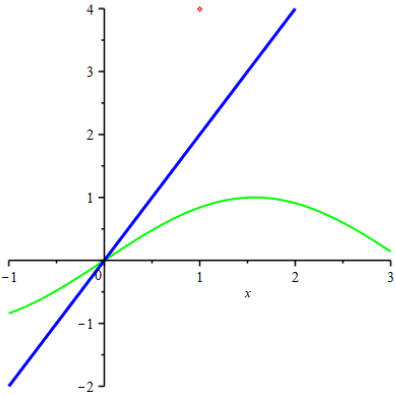
Теперь изобразим одновременно точку и функцию:

plot([*sin*(*x*), [[*1*, *2*]]], *x* = *0* .. *2*, *style* = [*LINE*, *POINT*], *thickness* = *2*, *symbol* = *CIRCLE*, *symbolsize* = *16*);



Другой вариант соединить разные графики с помощью команды *display*:

g1:=*plot*(*sin*(*x*),*x*= −*1*..*3*,*color*=*green*,*thickness*=*2*):
g2:=*plot*([[*1*,*4*]],*style*=*POINT*,*color*=*red*):
g3:=*plot*(*2***x*,*x*= −*1*..*2*,*color*=*blue*,*thickness*=*3*):
display([*g1*,*g2*,*g3*]);

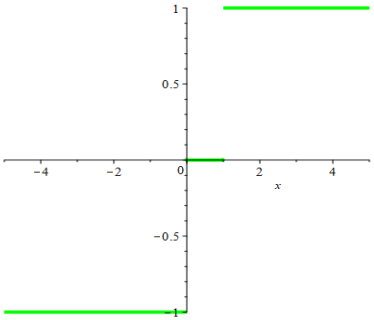


Кусочно-заданная функция.

К сожалению, в жизни не каждая функция непрерывная, даже для такого случая, как разрывная функция, Maple позаботилась о пользователях:

f:=x->piecewise(x<0, −1,x>=0 and x<=1,0,1<x,1);

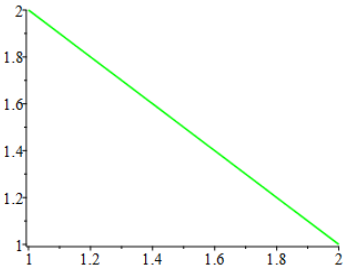
plot(f(x),x= −5..5,color=green, discont=true, thickness=4);



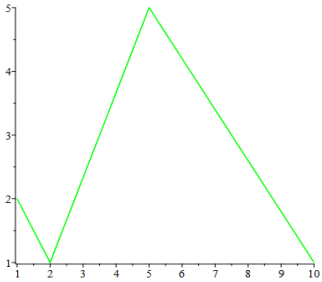
Задание отрезка, ломанной.

Давайте для примера нарисуем с вами прямую с помощью двух точек:

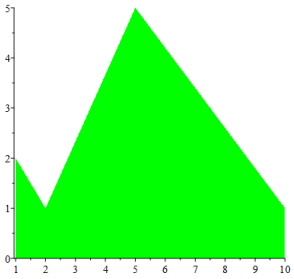
plot([[*1, 2*], [*2, 1*]], *color = green*);



Теперь зададим ломанную на 4 точках:
plot([[*1*, *2*], [*2*, *1*], [*5*, *5*], [*10*, *1*]], *color* = *green*);

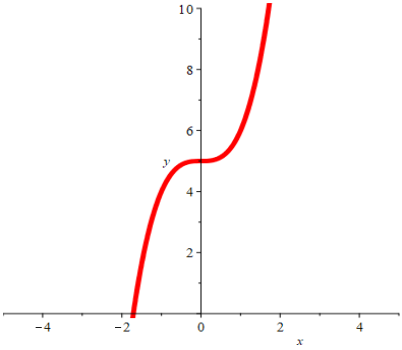


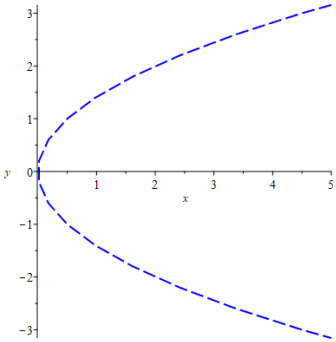
Закрасим получившуюся область:
plot([[*1*, *2*], [*2*, *1*], [*5*, *5*], [*10*, *1*]], *color* = *green*, *filled*);



Практическое занятие 3

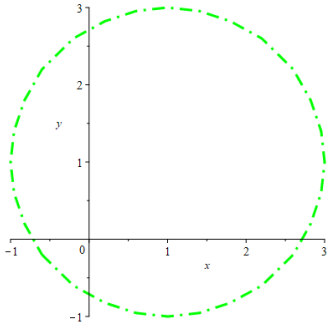
Задание 1. Нарисовать графики функций с соответствующим оформлением: $a)\ y = x^3 + 5;\ b)\ y\text{^}2\ =\ (2 * x);$

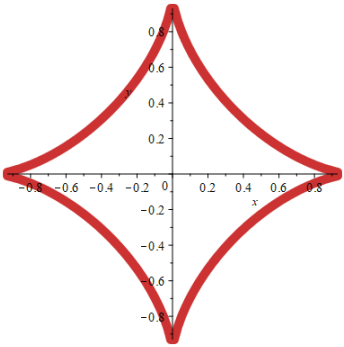




Задание 2. Нарисовать графики функций с соответствующим оформлением: $a)\ x\text{^}2 - 2 * x + y\text{^}2 - 2 * y - 2 = 0;$

$b)\ x\text{^}(2/3) + y\text{^}(2/3) = 1;$

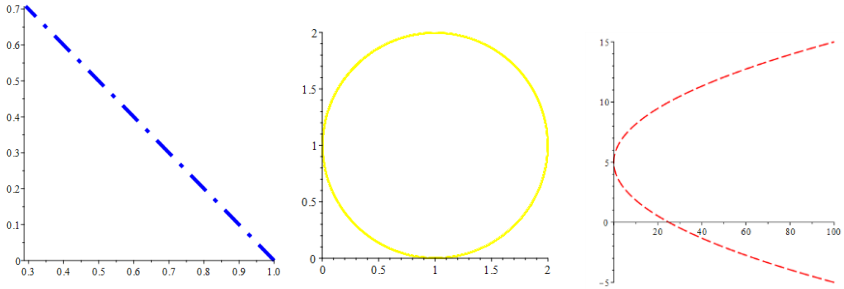




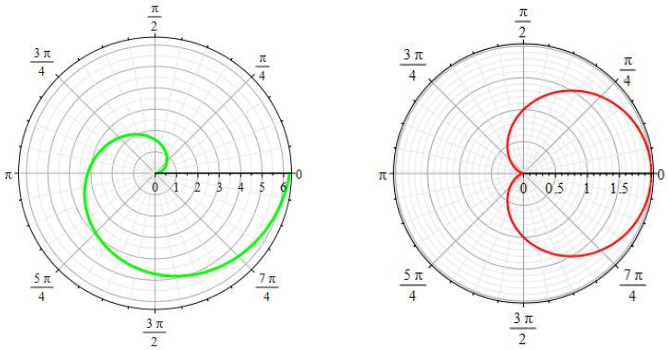
Задание 3. Нарисовать графики функций с соответствующим оформлением: $a)\ \{`x = cos^2(t), y = sin^2(x)\};$

$b)\ \{x = 1 - sin(t), y = 1 - cos(t)\};$

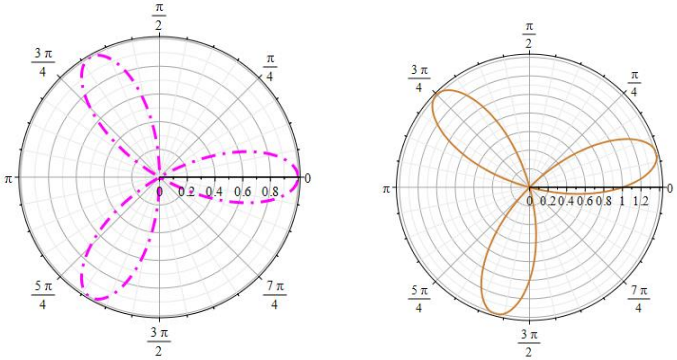
$c)\ \{x = t^2, y = t + 5\};$



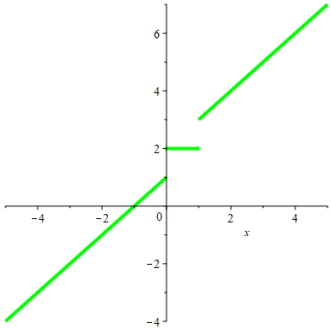
Задание 4. Нарисовать графики функций с соответствующим оформлением:$a)\ r = phi;\ \ b)\ r = 1 + \cos(phi)$;



Задание 5. Нарисовать графики функций с соответствующим оформлением: $a)\ r = cos(3 * t);\ \ b)\ r\ =\ cos(3 * t)\ +\ sin(3 * t)$;



Задание 6. Нарисовать график функции с соответствующим оформлением: $y = \begin{cases} x+1, x<0 \\ 2, \ 0 \le x \le 1; \\ x+2, \ x>1 \end{cases}$



Задание 7. Нарисовать графики функций:

1) $y = x\text{^}(2) + 1/(x)$ `трезубец` `Ньютона`

2) $y = \dfrac{1}{1+x^2}$ `кривая` `Аньези`

3) $y = \dfrac{2\,x}{1+x^2}$ `серпантин` `Ньютона`

4) $y\text{^}(2) = x\text{^}(3)$ `парабола` `Нейля`

5) $y^2 = \dfrac{x^3}{10-x}$ `циссоида`

6) $r = phi$ спираль Архимеда`

7) $r = 1 + \cos(phi)$ `кардиоида`

8) $r = e^{phi}$ `логарифмическая` `спираль`

9) $\{x = t - \sin(t)\,, y = 1 - \cos(t)\}$ `циклоида`

10) $\{x = ln(tg\left(\dfrac{t}{2}\right) + cos(t), y = sin(t)\}$ `трактриса`

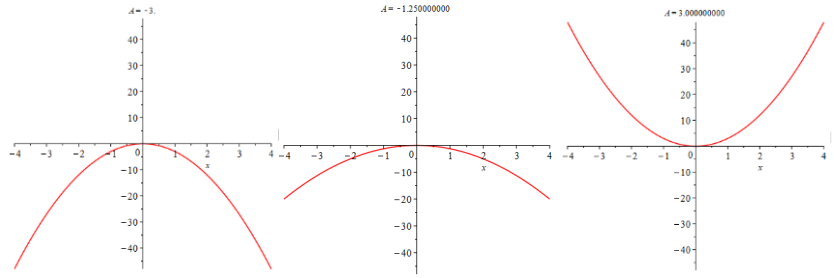
Лекция 4. Анимация графика функции

Помимо статических картинок в приложении Maple возможно создавать динамический рисунок. Для этого мы воспользуемся функцией animatic. Напишем явно заданную функцию:

with(*plots*)*:*

animate(*plot*, [*A*x^2,x= −4..4*], *A= −3..3*)*;*

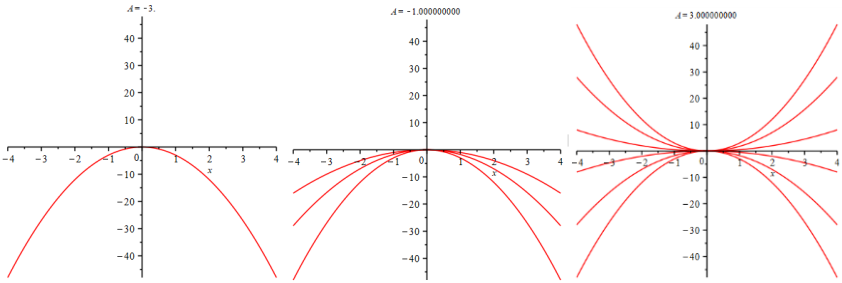
Теперь остаётся лишь нажать на картинку и в верхней панели появится проигрыватель. Нажимаем кнопку воспроизведения, и ветви параболы плавно поднимутся вверх:



В анимации есть возможность зацикливания, обратного хода, задание количества кадров.

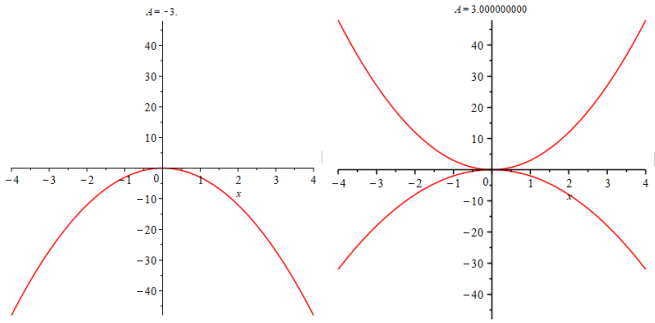
С помощью команды trace мы можем задавать, сколько и какие промежуточные кадры оставлять на экране:

animate(*plot*, [*A*x^2,x= −4..4*], *A= −3..3*, *trace=5*)*;*



Команда trace – это промежуточные кадры, которые помогают рассмотреть картинку внимательнее. Попробуйте написать:

animate(*plot*, [*A*x^2,x= −4..4*], *A= −3..3*, *trace=*[*5,25*])*;*



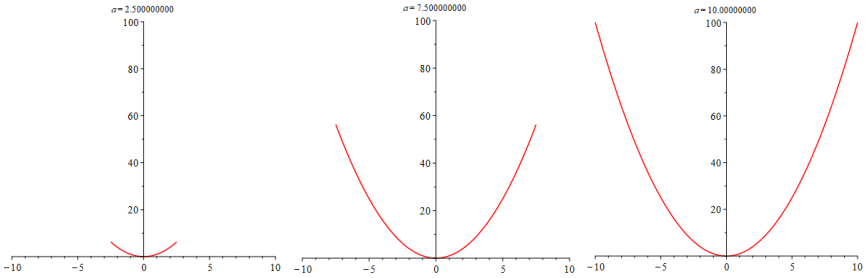
А теперь прибавим нашей анимации немного скорости, для этого воспользуемся командой frames:

animate(*plot*,[*A*x^2,x= −4..4],A= −3..3,*frames=10*, *color=blue*);

И ветви параболы поднимаются в разы быстрее.

Другой вариант анимации для параболы, так называемый «эффект рисования»:

animate(*plot*, [*x^2*, *x* = −*a* .. *a*], *a* = *0* .. *10*);



Здесь «рисуем» обе веточки сразу.

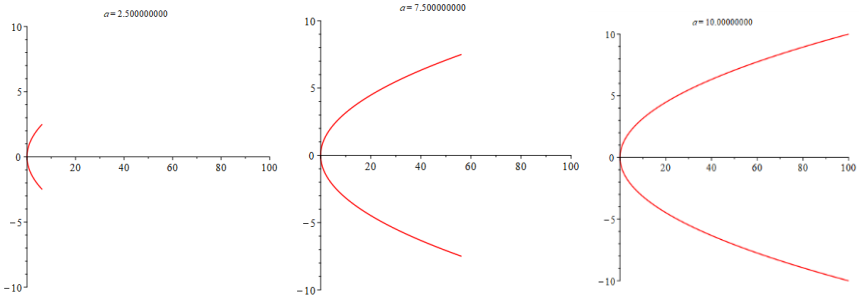
Возможен другой вариант оформления этой же анимации, по типу параметрического задания функции:

animate(*plot*, [[*x*, *x^2*, *x* = −*a* .. *a*]], *a* = *0* .. *10*);

Анимацию получаем такую же.

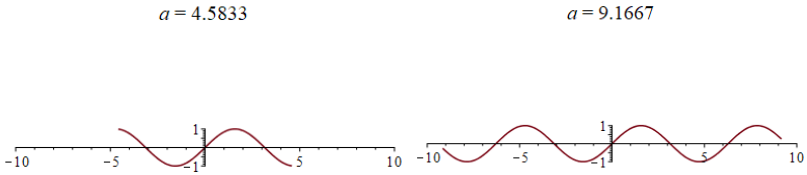
Для явно заданной функции есть возможность смены осей местами. Для этого достаточно поменять в последнем варианте задания анимации x и y местами:

animate(*plot*, [[*x^2*, *x*, *x* = −*a* .. *a*]], *a* = *0* .. *10*);

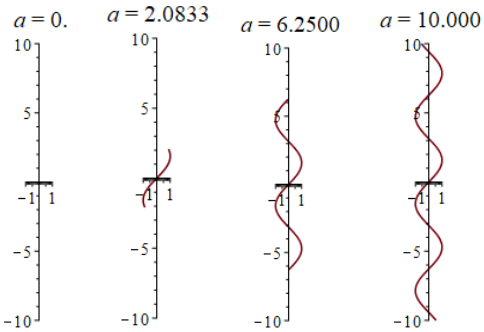


Другой пример:

animate(*plot*, [[*x*, *sin*(*x*), *x* = −*a* .. *a*]], *a* = *0* .. *10*, *scaling* = *constrained*);

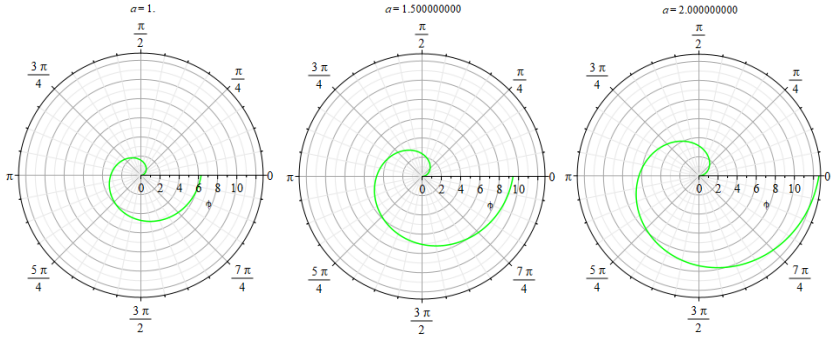


animate(*plot*, [[*sin*(*x*), *x*, *x* = −*a* .. *a*]], *a* = *0* .. *10*, *scaling* = *constrained*);

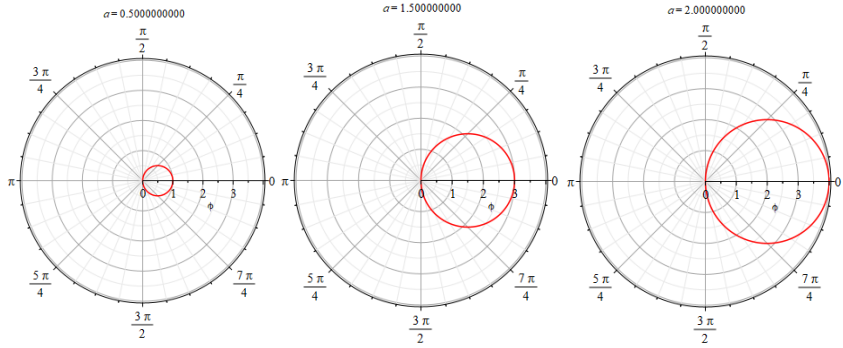


Рассмотрим полярно заданную функцию, уже знакомую нам спираль Архимеда:

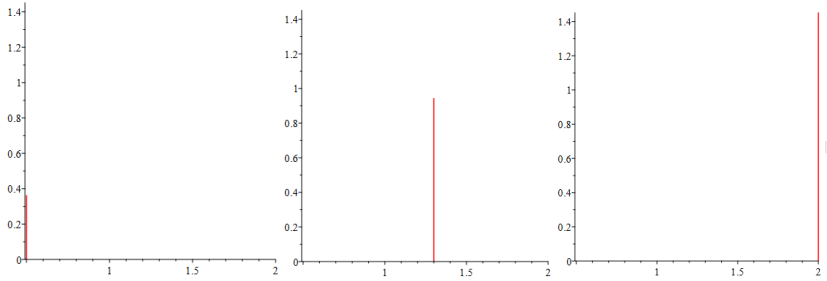
animate(*polarplot*, [[*a***phi*, *phi*, *phi* = *0* .. *2***Pi*], *color* = *green*], *a* = *1* .. *2*, *scaling* = *constrained*);



Анимируем те же кривые, которые задавали ранее.
Окружность:
animate(*polarplot*, [[*2*a*cos*(*phi*), *phi*, *phi* = *0* .. *2*Pi*]], *a* = *1/2* .. *2*);

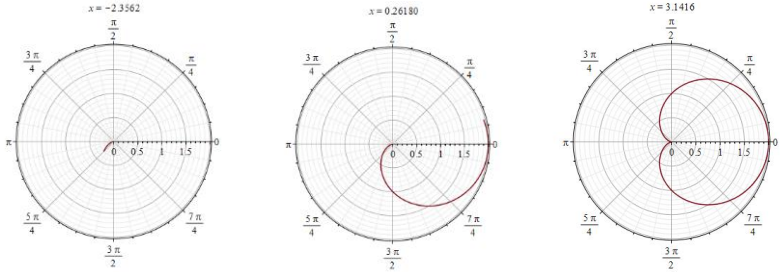


Отрезок:
animate(*a/cos*(*phi*), *phi* = *0* .. *0.2*Pi*, *a* = *1/2* .. *2*, *coords* = *polar*);



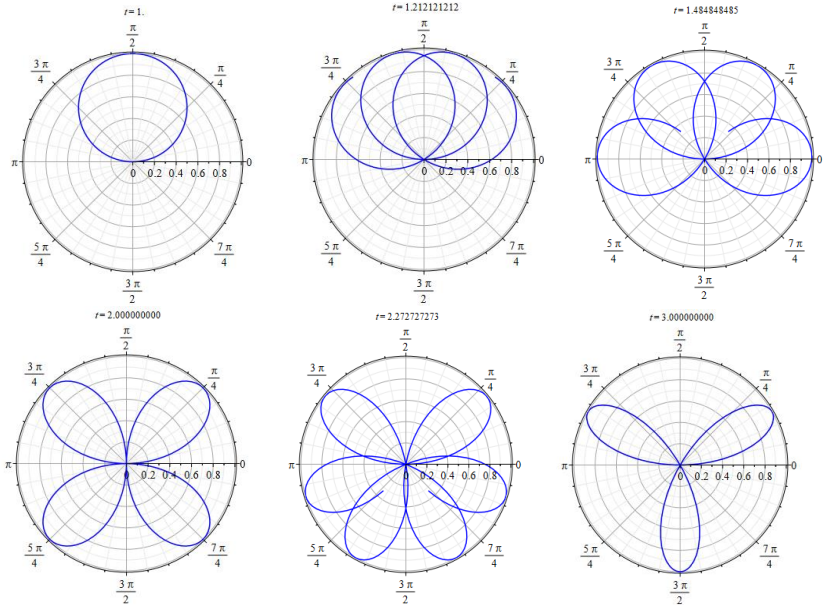
Кардиоида:

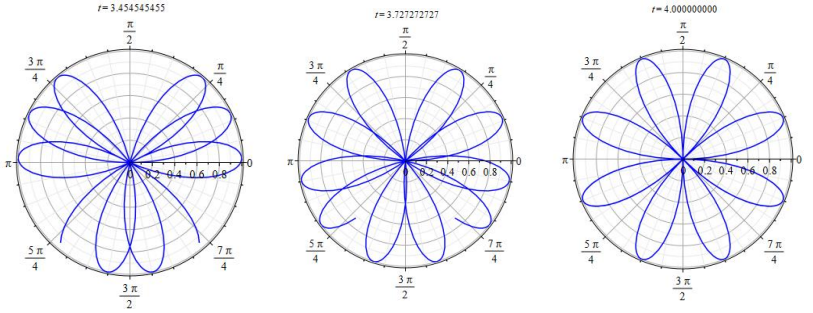
animate (*polarplot*, [*1+cos*(*phi*), *phi=x.. −Pi*], *x= −Pi..Pi*);



Команда *numpoints* задает количество точек, по которым строится график:

animate(*polarplot*, [[*sin*(*x*t*), *x*, *x = −4 .. 4*], *color = blue*], *t = 1 .. 4*, *numpoints = 100*, *frames = 100*);



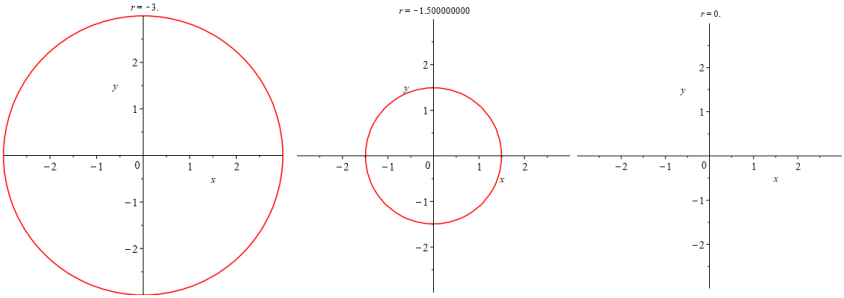


animate(*polarplot*, [[*10*sin*(*10*x*) + *sin*(*t*), *x*, *x* = *0* .. *2*Pi*], *color* = *yellow*, *thickness* = *5*], *t* = *0* .. *6*Pi*, *frames* = *240*, *axes* = *none*);

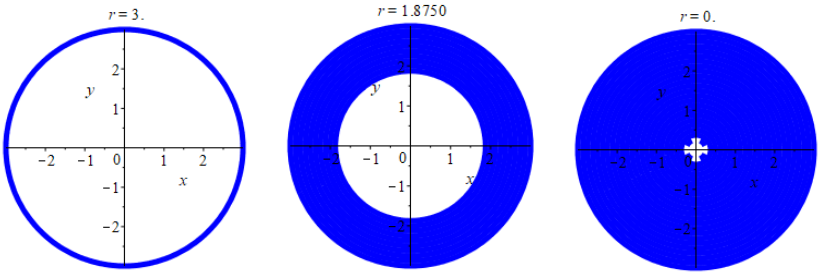
Теперь посмотрим, как будет себя вести неявно заданная функция в режиме анимации:

animate(*implicitplot*, [*x^2+y^2=r^2*, *x*= −*3..3*, *y*= −*3..3*], *r*= −*3..3*, *scaling*=*constrained*);

Если нажать на картинку, то мы увидим, как наша окружность быстро сужается.

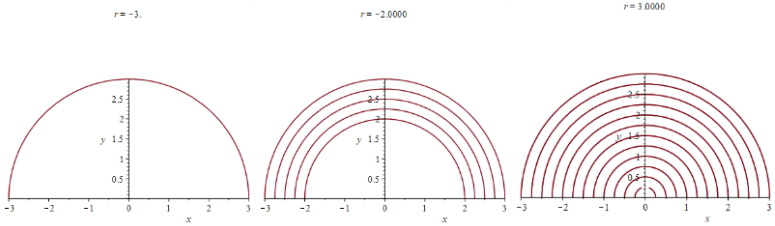


animate(*implicitplot*, [*x*^2 + *y*^2 = *r*^2, *x* = −3 .. 3, *y* = −3 .. 3, *color* = *blue*], *r* = 3 .. 0, *scaling* = *constrained*, *thickness* = 5, *trace* = 25);



Закраска окружности происходит за счет наложения кадров.

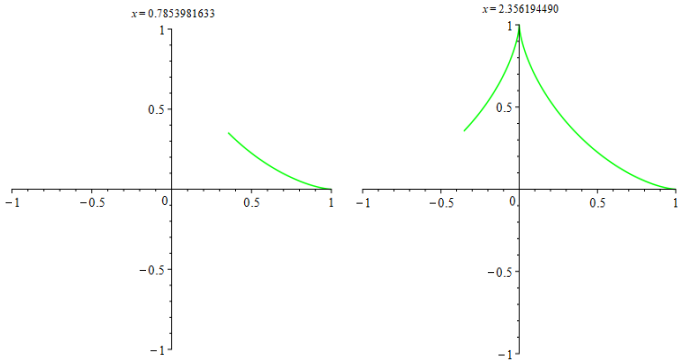
animate(*implicitplot*, [*x*^2 + *y*^2 = *r*^2, *x* = −3 .. 3, *y* = 0 .. 3], *r* = −3 .. 3, *scaling* = *constrained*, *trace* = 25, *grid* = 10000);

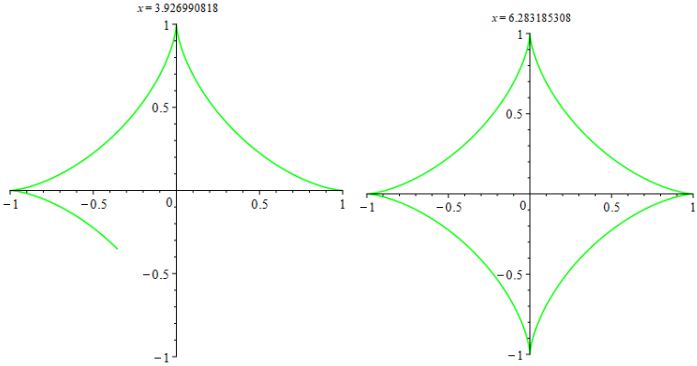


Тоже самое проделаем с параметрически заданной функцией:

animate(*plot*,[[*cos*(*t*)^3,*sin*(*t*)^3,*t*=0..*x*],*color*=*green*],*x*=0..2*Pi*, *scaling*=*constrained*);

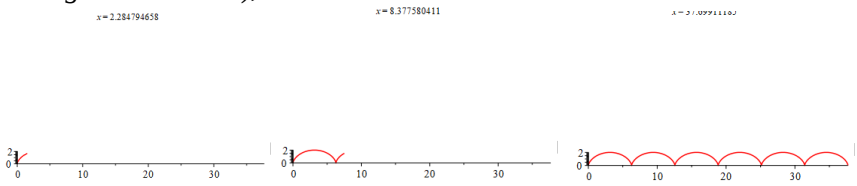
Здесь опять есть «эффект рисования». На выходе мы получаем вот такую красивую звездочку, называемую астроидой:





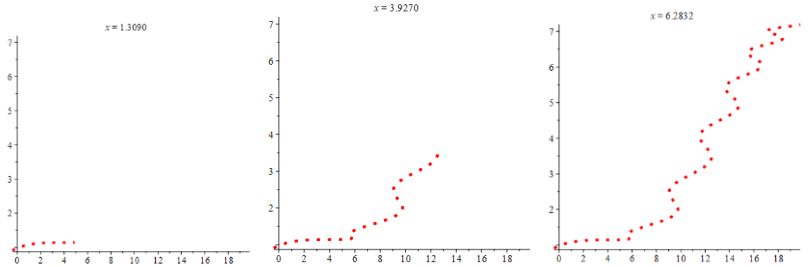
Следующий график называется циклоидой. Циклоида демонстрирует траекторию точки колеса автомобиля при движении по прямой.

animate(*plot*,[[*t*−*sin*(*t*),1−*cos*(*t*),*t*=0..*x*]],*x*=0..12*Pi, *frames*=100, *scaling*=*constrained*);

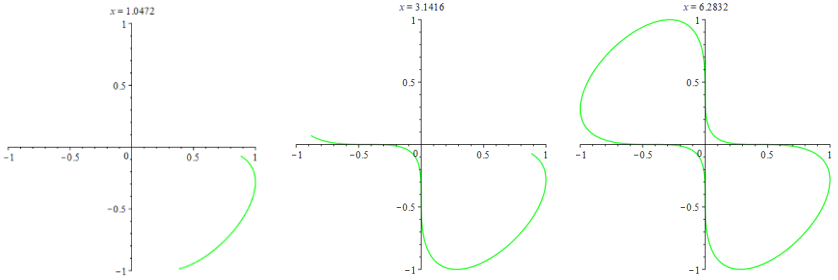


Нарисуем график точками:

animate(*plot*, [[*cos*(*t*^2 − 2) + 3*t, *t* + *sin*(*t* + 2), *t* = 0 .. *x*], *color* = *red*], *x* = 0 .. 2*Pi, *linestyle* = 2, *thickness* = 4);



animate(*plot*, [[*sin*(*t* − 5)^3, *cos*(*t* + 2)^3, *t* = 0 .. *x*], *color* = *green*], *x* = 0 .. 2*Pi, scaling* = *constrained*);

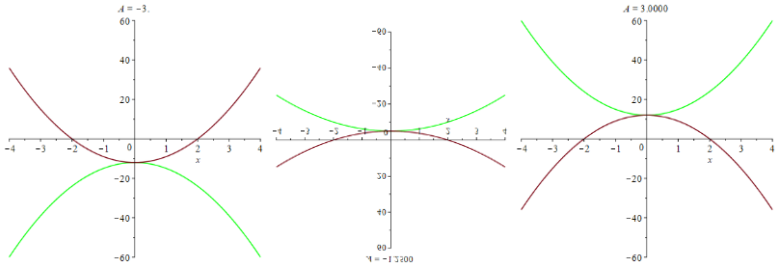


С помощью display мы можем обьединять несколько функций на одну картинку:

g1 := *animate*(*plot*, [*A**(*x*^2 + 4), *x* = −4 .. 4, *color* = *green*], *A* = −3 .. 3);

g2 := *animate*(*plot*, [*A**(−*x*^2 + 4), *x* = −4 .. 4], *A* = −3 .. 3);

display(*g1*, *g2*);

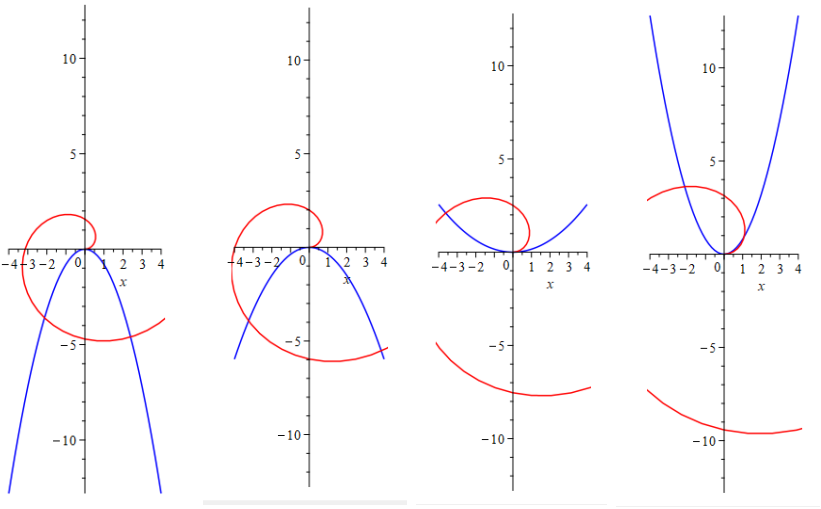


Еще пример:

g1:=*animate*(0.2*a*x^2,x= −4..4,a= −4..4,color=blue):

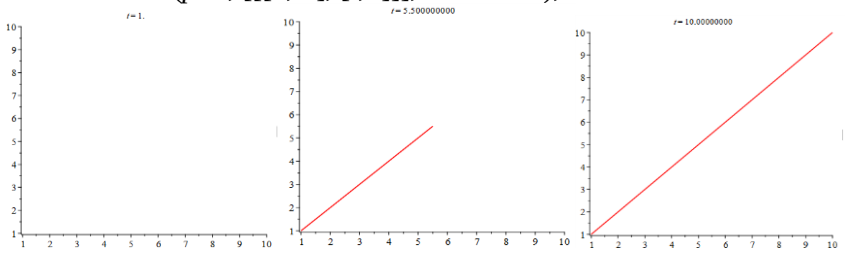
g2:=*animate*(a*phi,phi=0..2*Pi,a=1..2,coords=polar,scaling=constrained):

display(*g1*,*g2*);

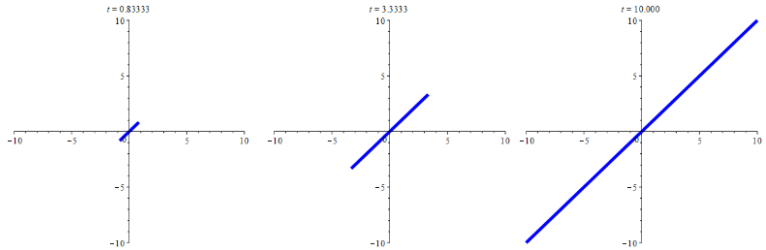


Анимация отрезка:

animate(*plot*, [[[*1*, *1*], [*t*, *t*]]], *t* = *1* .. *10*);

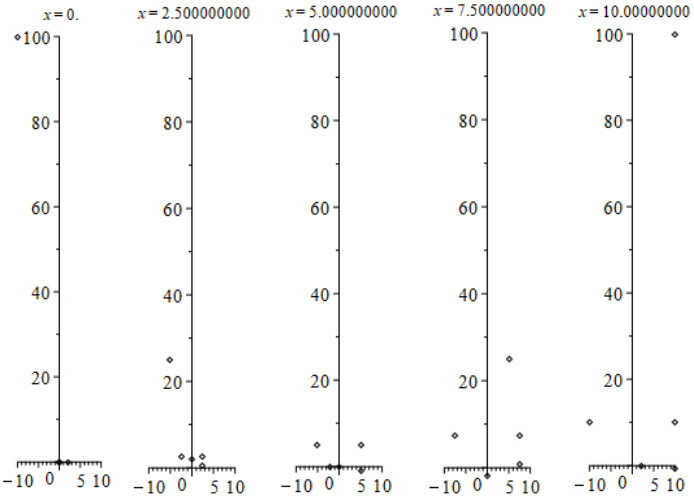


animate(*plot*, [[[−*t*, −*t*], [*t*, *t*]], *color* = *blue*, *thickness* = *5*], *t* = *0* .. *10*);



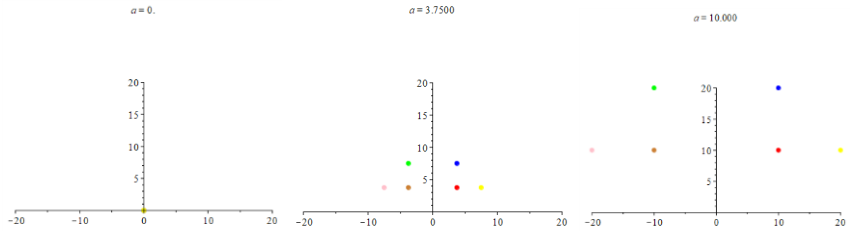
Так же можно задавать анимацию точек:

g1 := animate(pointplot, [[a, a]], a = 0 .. 10):
g2 := animate(pointplot, [[−a, a]], a = 0 .. 10):
*g3 := animate(pointplot, [[2*cos(p), 2*sin(p)]], p = 0 .. 2*Pi):*
g5 := animate(pointplot, [[[x, sin(x)]]], x = 0 .. 10):
g4 := animate(pointplot, [[x, x^2]], x = −10 .. 10):
display(g1, g2, g3, g4, g5, scaling = constrained);



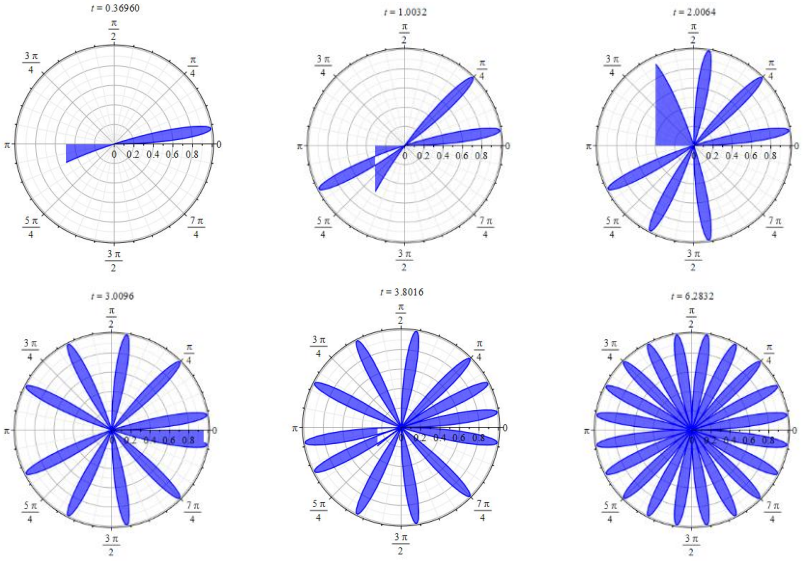
Еще вариант:

g5 := animate(pointplot, [[a, a]], a = 0 .. 10, symbol = solidCIRCLE, symbolsize = 15, color = red);
*g6 := animate(pointplot, [[a, 2*a]], a = 0 .. 10, symbol = solidCIRCLE, symbolsize = 15, color = blue);*
*g7 := animate(pointplot, [[−a, 2*a]], a = 0 .. 10, symbol = solidCIRCLE, symbolsize = 15, color = green);*
g8 := animate(pointplot, [[−a, a]], a = 0 .. 10, symbol = solidCIRCLE, symbolsize = 15, color = gold);
*g9 := animate(pointplot, [[−2*a, a]], a = 0 .. 10, symbol = solidCIRCLE, symbolsize = 15, color = pink);*
*g10 := animate(pointplot, [[2*a, a]], a = 0 .. 10, symbol = solidCIRCLE, symbolsize = 15, color = yellow);*
display(g5, g6, g7, g8, g9, g10, scaling = constrained);

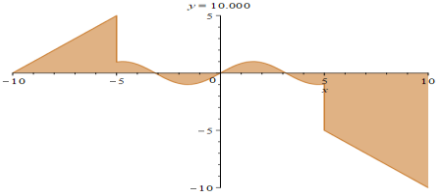


С помощью команды filled мы можем закрашивать наши функции в процессе анимации:

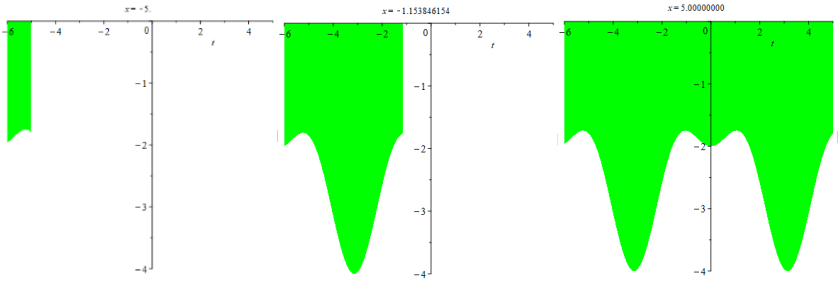
animate (*polarplot*, ([*sin*(*10*x*), *x=0..t*, *filled*, *color=blue*], *t=0..2*Pi*, *frames=120*));



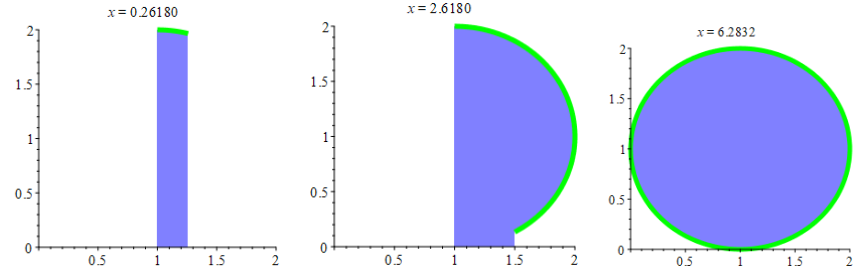
f:=*piecewise* (*x*<−*5*,*x*+*10*,*x*<*5 and x*>−*5*, *sin*(*x*),*x*>*5*, −*x*):
animate (*plot*, [*f*, *x*= −*10..y*, *color=gold*,*filled*], *y*= −*10..10*);



animate (*plot*, [*sin*(*t*)^2+*cos*(*t*) −3, *t*= −6..*x*,*filled*,*color*=*green*], *x*= −5..5, *frames*=40);



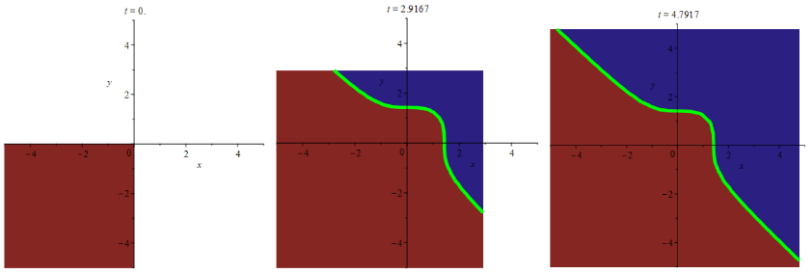
animate(*plot*, [[*1*+*sin*(*t*), *1*+*cos*(*t*), *t*=*0*..*x*],*color*=*green*, *thickness* = 5,*filled* = [*color* = "Blue", *transparency* = *0.5*]], *x*=*0*..*2***Pi*);



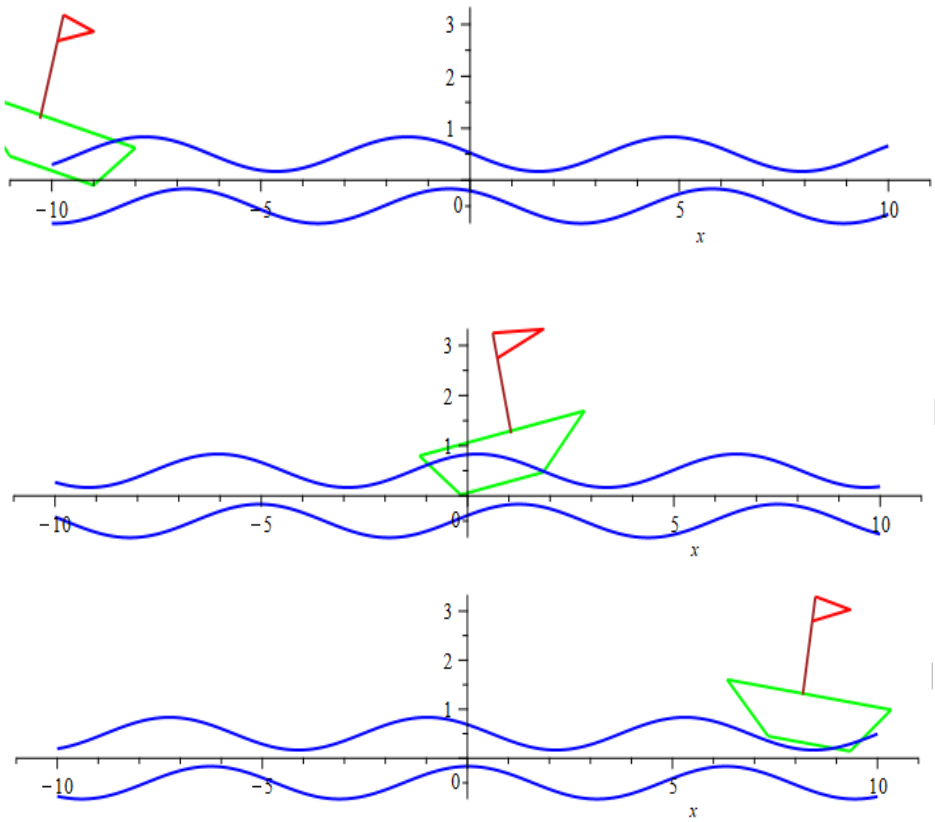
Так как мы потратили уже много памяти компьютера, то можем ее «обнулить» командой restart:

restart: with (*plots*)*:*

animate (*implicitplot*, [*x*^3+*y*^3=3, *x*= −5..*t*, *y*= −5..*t*, *filled*, *color*= *green*, *thickness* = 5],*t*=*0*..*5*);

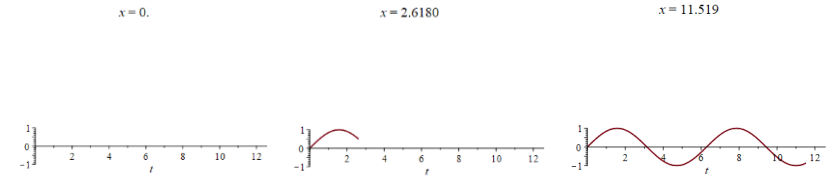


Теперь мы научились с вами анимировать графики, а значит совсем скоро мы сможем попробовать себя в мультипликационном деле и создать короткий мультик, например:

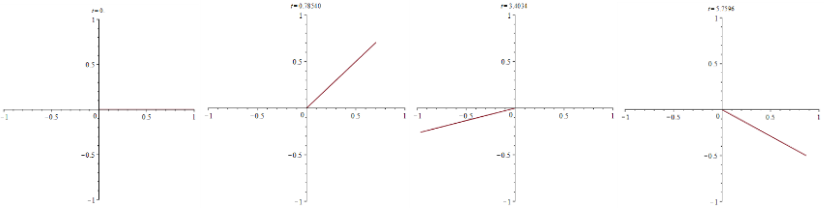


Практическое задание 4

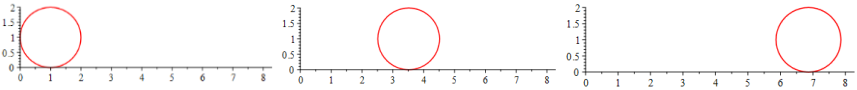
Задание 1. «Нарисовать» синусоиду с эффектом рисования:



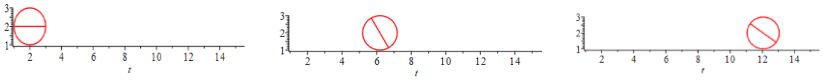
Задание 2. Нарисовать вращающийся отрезок:



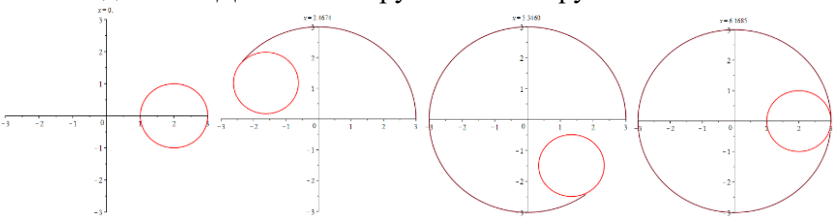
Задание 3. Окружность «катится»:



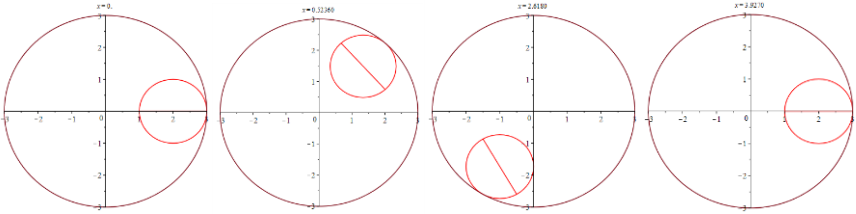
Задание 4. Окружность с диаметром «катится»



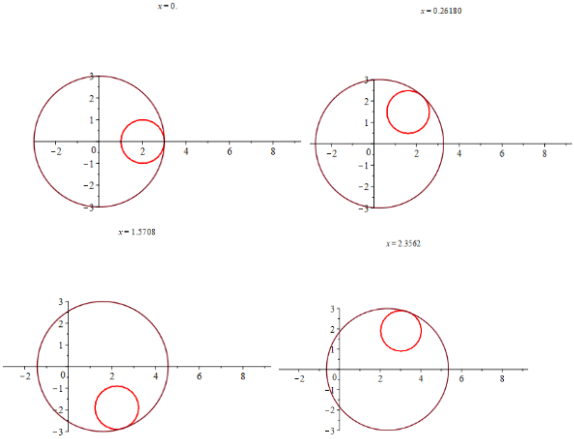
Задание 5. Движение окружности в окружности:



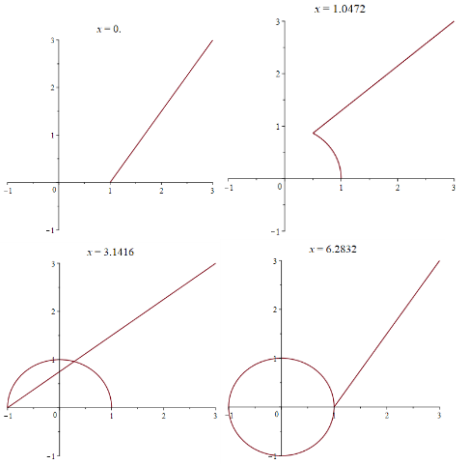
Задание 6. Движение окружности с диаметром в окружности:



Задание 7. Движение двух окружностей:



Задание 8. Ручкой рисуем окружность:



Лекции 5, 6. Построение геометрических фигур по встроенным опциям

Подключим знакомый нам пакет графики и новый пакет графических встроенных опций:

with(*plots*)*:*

with(*plottools*)*:*

Когда в конце записи ставим «;», то можно увидеть список тех самых опций:

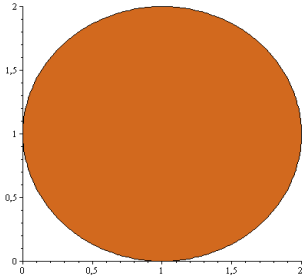
[*annulus, arc, arrow, circle, cone, cuboid, curve, cutin, cutout, cylinder, disk, dodecahedron, ellipse, ellipticArc, exportplot, extrude, getdata, hemisphere, hexahedron, homothety, hyperbola, icosahedron, importplot, line, octahedron, parallelepiped, pieslice, point, polygon, polygonbyname, prism, project, rectangle, reflect, rotate, scale, sector, semitorus, sphere, stellate, tetrahedron, torus, transform, translate*]

Ознакомимся с геометрическими фигурами этого пакета.

Круг задается по координатам центра и радиусу:

g1 := *disk*([*1, 1*], *1, color = "Chocolate"*)*:*

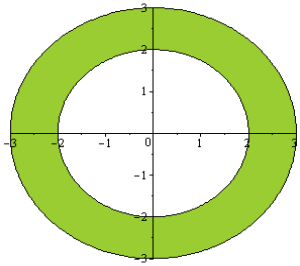
PLOT(*g1*)*;*



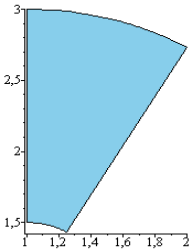
Следующая фигура кольцо, оно по умолчанию находится в центре координат, а задаются только два радиуса:

g2 := *annulus*(*2..3, color= "YellowGreen"*)*:*

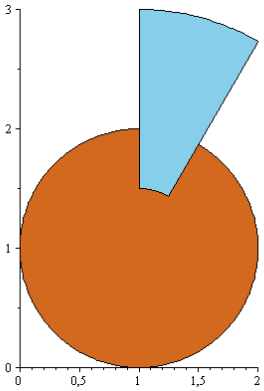
PLOT(*g2*)*;*



Часть круга или кольца:
a1 := *sector*([*1,1*], *0.5 ..2, Pi/3..Pi/2, color = "SkyBlue"*):
PLOT(*a1*);

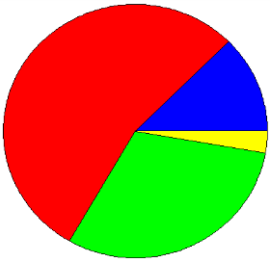


Соединим круг и сегмент:
display(*a1,g1, scaling = constrained*);



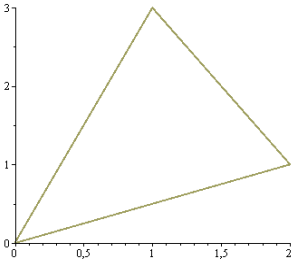
Можно создать полноценную круговую диаграмму аналогичным образом:

```
c := sector([1, 1], 5, 0..(1/4)*Pi, color = blue):
d := sector([1, 1], 5, (1/4)*Pi .. 4*Pi*(1/3), color = red):
e := sector([1, 1], 5, 4*Pi*(1/3) .. 35*Pi*(1/18), color = green):
f:= sector([1, 1], 5, 35*Pi*(1/18) .. 2*Pi, color = yellow):
display(c, d, e, f, axes = none, scaling = constrained);
```



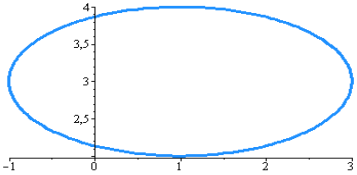
Следующая линия ломаная:

```
g3 := curve([[0, 0], [1, 3], [2, 1], [0, 0]], thickness = 2,
 color = khaki):
PLOT(g3);
```



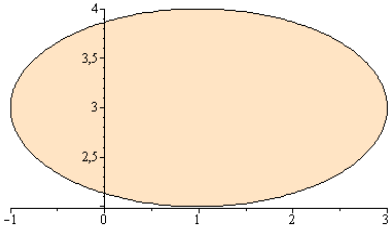
Эллипс или по-простому овал:

```
g4 := ellipse([1, 3], 2, 1, thickness = 3, color = "DodgerBlue"):
PLOT(g4, SCALING(CONSTRAINED));
```



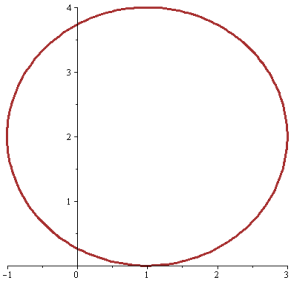
Овал, в отличие от ломаной, можно закрасить:

```
g5 := ellipse([1, 3], 2, 1, filled, color = "Bisque"):
PLOT(g5, SCALING(CONSTRAINED));
```



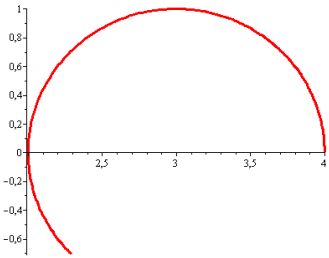
Окружность. Ее тоже нельзя закрасить, но можно задать толщину:

```
g6 := circle([1,2], 2, thickness = 3, color = brown):
PLOT(g6);
```



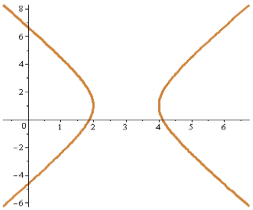
А теперь удалим часть линии:

```
a := arc([3,0], 1, 0..(5*Pi/4)):
display(a, color = red, thickness = 3, scaling = constrained);
```



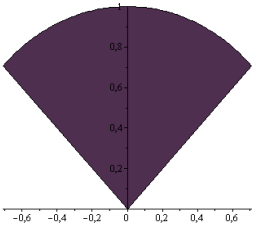
Гипербола:

g7 := *hyperbola*([*3*, *1*], *1*, *2*, *thickness* = *3*, *color* = *gold*):
PLOT(*g7*);



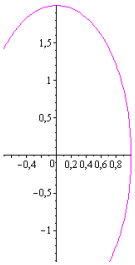
Сектор:

g8 := *pieslice*([*0*, *0*], *1*, (*1*/*4*)*Pi .. *3*Pi*(*1*/*4*), *color* = *violet*):
PLOT(*g8*);



Часть эллипса:

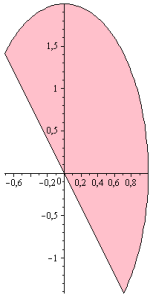
g9 := *ellipticArc*([*0*, *0*], *1*, *2*, −(*1*/*4*)*Pi .. *3*Pi*(*1*/*4*),
color = *magenta*):
PLOT(*g9*, *SCALING*(*CONSTRAINED*));



Закрасим получившуюся область:

g10 := *ellipticArc*([*0, 0*], *1, 2, −*(*1/4*)**Pi .. 3*Pi**(*1/4*), *filled, color = pink*):

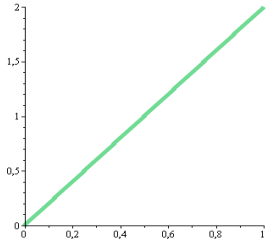
PLOT(*g10*);



Отрезок задается по координатам концов:

g11 := *line*([*0, 0*], [*1, 2*], *thickness = 5, color = aquamarine*):

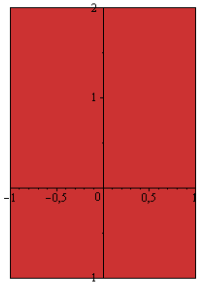
PLOT(*g11*);



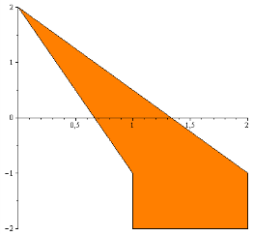
Прямоугольник по координатам двух противоположных вершин:

g12 := *rectangle*([*−1, −1*], [*1, 2*], *color = orange*):

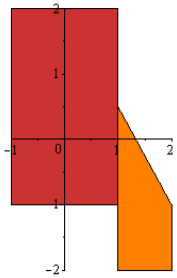
PLOT(*g12*, *SCALING*(*CONSTRAINED*));



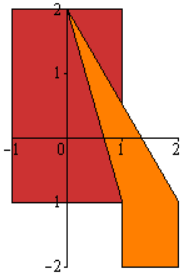
Самая универсальная фигура многоугольник:
g13 := *polygonplot*([[*0, 2*], [*2, −1*], [*2, −2*], [*1, −2*], [*1, −1*]], *color* = *coral*): *display*(*g13*);



Теперь поиграемся с фигурами:
display(*g12, g13*);

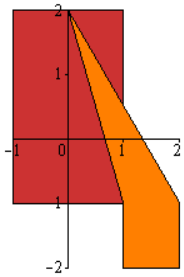


display(*g13, g12*);

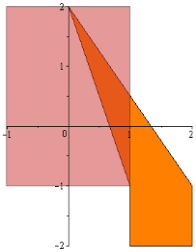


Становится понятным, что фигуры накладываются «с конца». Можно получить наложение «с начала», поставив дополнительные скобки:

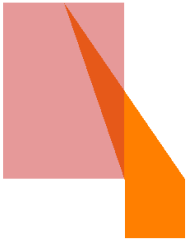
display([*g12, g13*]);



Можно задать прозрачность:
g121 := *rectangle*([−*1*, −*1*], [*1*, *2*], *color* = *orange*, *transparency*= *.5*)*; display*(*g121*, *g13*)*;*

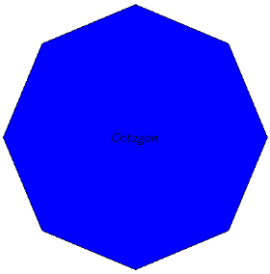


Можно удалить оси и убрать черный контур фигур:
display(*g121*, *g13*, *style* = *polygon*, *axes* = *NONE*)*;*

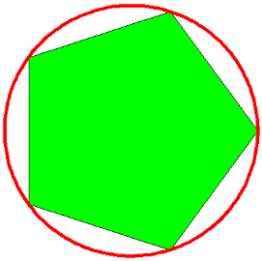


Можно задать правильный многоугольник через цикл (команда seq) и вводимую функцию:

ngon := *n* −> [*seq*([*cos*(*2*Pi*i/n*), *sin*(*2*Pi*i/n*)], *i* = *1* .. *n*)]*:*
display([*polygonplot*(*ngon*(*8*), *color* = *blue*),
textplot([*0*, *0*, *Octagon*])], *axes* = *none*)*;*



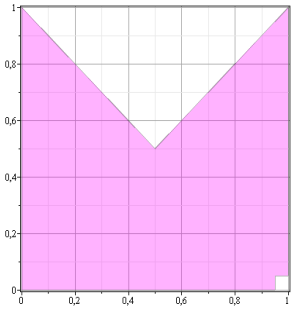
s1 := *display*([*polygonplot*(*ngon*(*5*), *color* = *green*)], *axes* = *none*);
s2 := *circle*([*0*, *0*], *1*, *thickness* = *5*, *color* = *red*);
display(*s1*, *s2*);



Многоугольники можно рисовать при помощи матриц:

one_poly := *Matrix*([[*0*, *0*], [*0*, *1*], [*0.5*, *0.5*], [*1*, *1*], [*1*, *0.05*], [*0.95*, *0.05*], [*0.95*, *0*]], *datatype* = *float*):

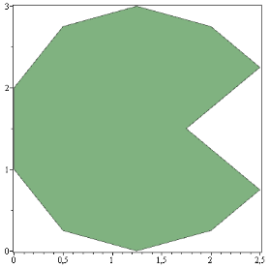
polygonplot(*one_poly*, *axes* = *boxed*, *colour* = *"Magenta"*, *transparency* = *0.7*, *gridlines*);



Мы можем воссоздать многоугольник выписывая координаты точек отдельно от опции:

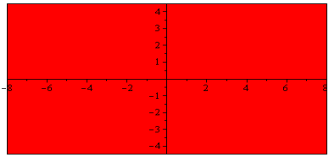
poly := [[*0, 1*], [*0, 2*], [*0.5, 2.75*], [*1.25, 3*], [*2, 2.75*], [*2.5, 2.25*], [*1.75, 1.5*], [*2.5, 0.75*], [*2, 0.25*], [*1.25, 0*], [*0.5, 0.25*]]*:*

polygonplot(*poly, axes = boxed, color = "DarkGreen", transparency = 0.5*)*;*



Еще одно задание прямоугольника:

display(*polygonbyname*(*"rectangle", inbox* = [*16, 9*], *color = "Red"*), *scaling = constrained*)*;*



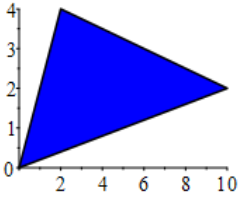
Если вам пресытились часто используемые цвета, то введите в рабочее окно эти две функции и сможете увидеть все доступные оттенки:

with(*ColorTools*)*:*

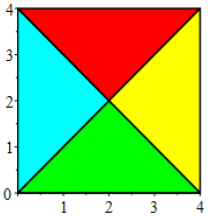
Palette(*GetColorNames*())*;*

Практические занятия 5, 6.

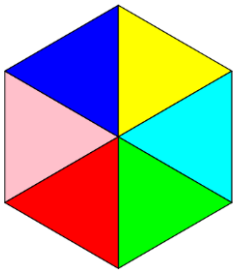
Задание 1. Нарисовать треугольник:



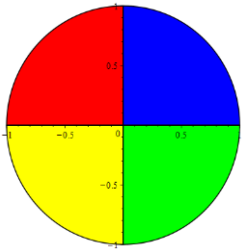
Задание 2. Из четырех треугольников составить цветной квадрат:



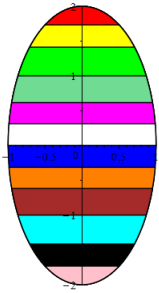
Задание 3. Из шести треугольников составить цветной правильный шестиугольник:



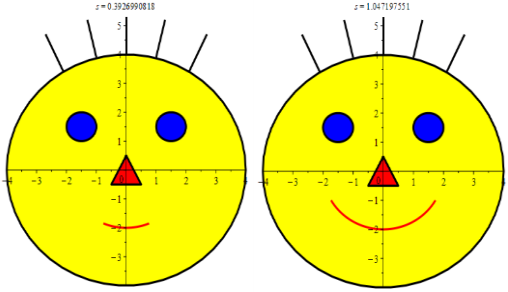
Задание 4. Из четырех секторов составить цветной круг:



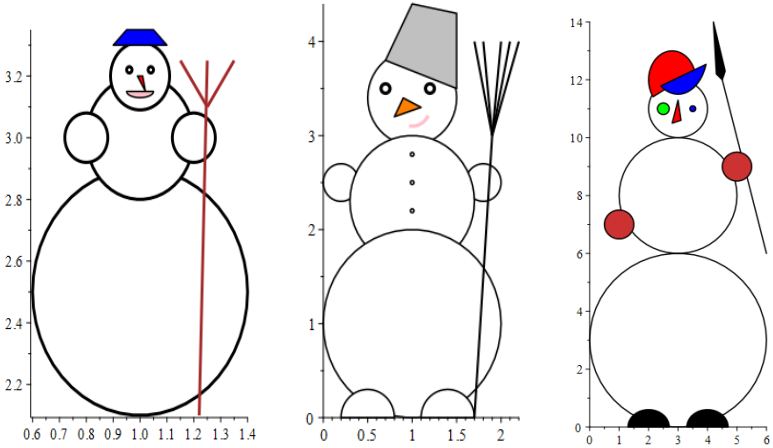
Задание 5. Нарисуйте полосатый эллипс. Как получить такую закраску?



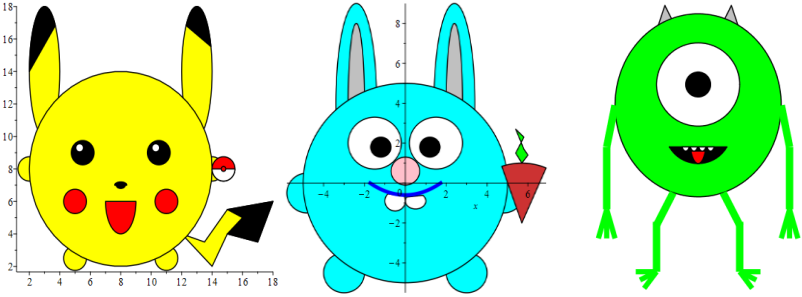
Задание 6. Нарисуйте рожицу. Задайте анимацию ее улыбки:



Задание 7. Нарисуйте снеговика. Можно свой вариант:



Задание 8. Если у вас есть желание, то можете нарисовать мультяшного героя. Например:



Лекция 7. Анимация встроенных фигур

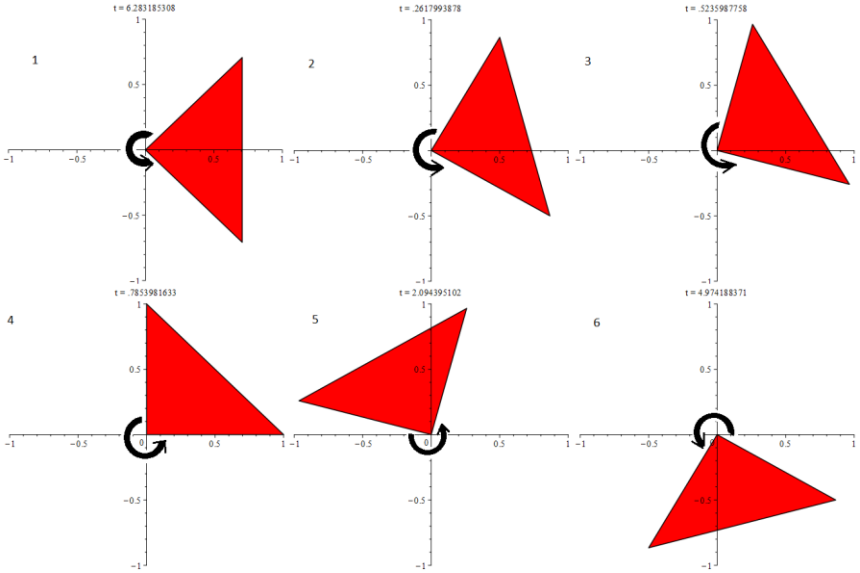
Попробуем объединить изученные ранее нами две темы: анимация и встроенные фигуры. Набираем в рабочее окно нужные нам для работы встроенные пакеты:

with(*plots*)*: with*(*plottools*)*:*

Не все встроенные фигуры можно анимировать командой *animate*. Этой команде лучше всего подчиняется многоугольник.

Анимируем цветной квадрат, состоящий из четырех маленьких треугольников, задавая анимацию каждого треугольника по отдельности:

animate(*polygonplot*,[[[*cos*(*Pi/4+t*),*sin*(*Pi/4+t*)],[*0,0*],[*cos*(*Pi/4+t*),*sin*(*Pi/4 + t*)]],*color=red*],*t=0..2*Pi*)*;*



А теперь добавим ещё три, но других цвета (обратите внимание на чередование знаков внутри функции):

Желтый:

g1:=animate(*polygonplot*,[[[*cos*(−(*3/4*)**Pi+t/4*),*sin*(−(*3/4*)**Pi+ t/4*)],[*0,0*],[*cos*(−(*1/4*)**Pi+t/4*), *sin*((*1/4*)**Pi+t/4*)]],*color=yellow*], *t=0..2*Pi*)*:*

Синий:

*g2:=animate(polygonplot,[[[cos((3/4)*Pi+t/4),sin((3/4)*Pi+t/4)],[0,0],[cos(−(3/4)*Pi+t/4),sin(−(3/4)*Pi+t/4)]],color=blue],t=0..2*Pi):*

Зеленый:

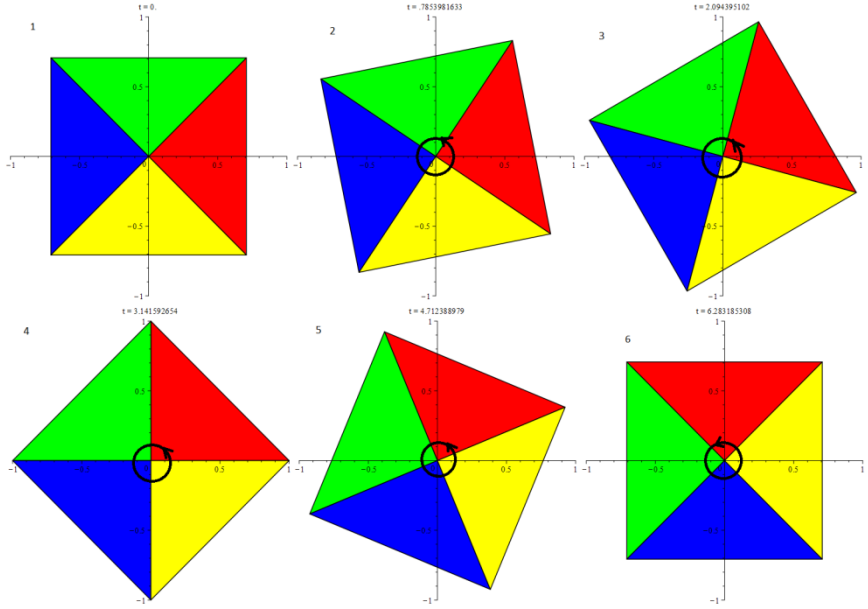
*g3:=animate(polygonplot,[[[cos(Pi/4+t/4),sin(Pi/4+t/4)],[0,0],[cos((3/4)*Pi+t/4),sin((3/4)*Pi+t/4)]], color=green],t=0..2*Pi):*

Красный:

*g4:=animate(polygonplot,[[[cos(−Pi/4+t/4),sin(−Pi/4+t/4)],[0,0],[cos(Pi/4+t/4),sin(Pi/4+t/4)]],color=red],t=0..2*Pi):*

И наконец-то запускаем:

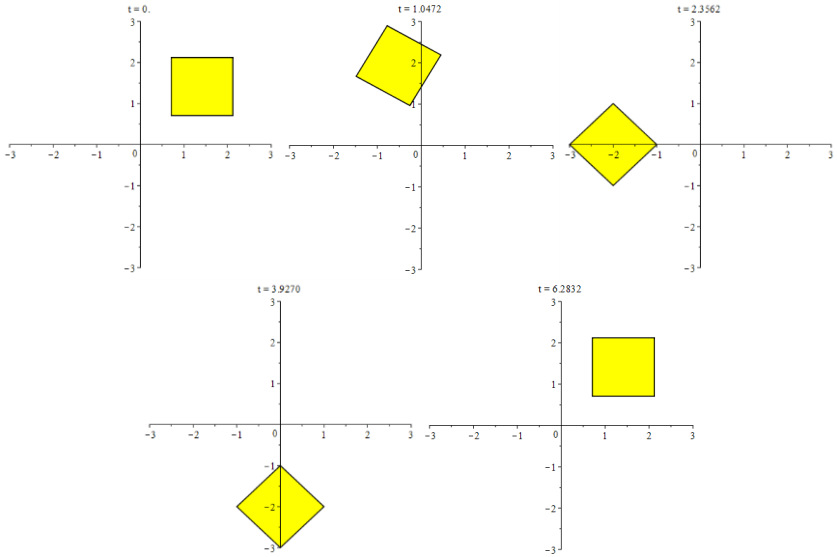
display(g1,g2,g3,g4);



Вот такая красота у нас вышла. Если вам нужен единый квадрат, то дело обстоит куда серьёзнее, ведь здесь не будет раздельных функций и нужно быть как можно внимательнее при наборе текста:

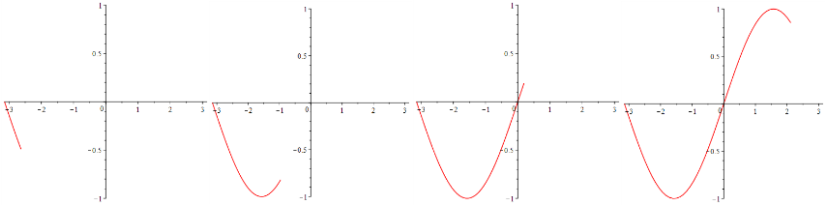
*animate(polygonplot,[[[2*cos(Pi/4+t)+cos(Pi/4+t),2*sin(Pi/4+t)+sin(Pi/4+t)],[2*cos(Pi/4+t)+cos(3*Pi/4+t),2*sin(Pi/4+t)+sin(3*Pi/4+t)],[2*cos(Pi/4+t)+cos(5*Pi/4+t),2*sin(Pi/4+t)+sin(5*Pi/4+t)],*

[*2*cos(Pi/4+t)+cos(−Pi/4+t),2*sin(Pi/4+t)+sin(−Pi/4+t)*]]],
*t=0..2*Pi, color=yellow*);



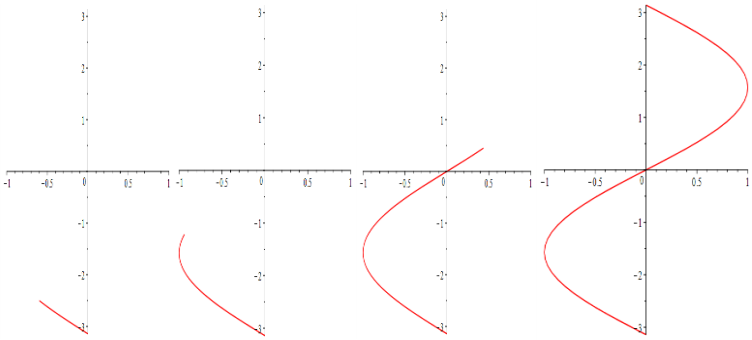
Следующий анимационный график – это график кривой линии *curve*. Нарисуем синусоиду с эффектом «рисования»:

animatecurve(*sin*(*x*), *x* = −*Pi* .. *Pi*, *frames* = *50*);



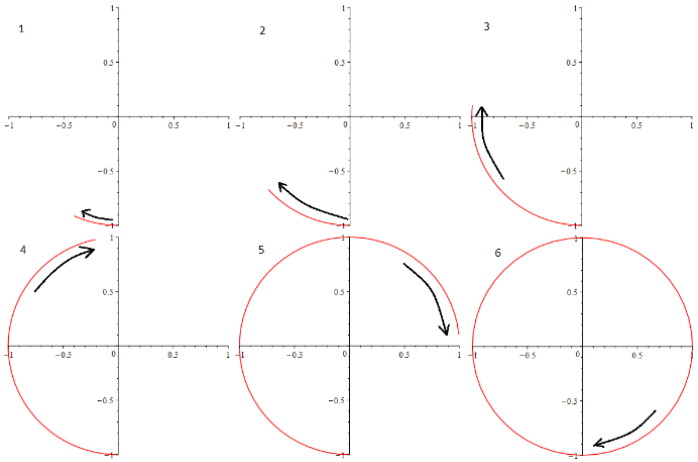
Кстати, если нам нужно изобразить график по оси у, то после указания нашей функции, просто, через запятую пишем нужную ось:

animatecurve([*sin*(*x*), *x*, *x* = −*Pi* .. *Pi*], *frames* = *50*);



А теперь зададим окружность:

animatecurve([*sin*(*x*), *cos*(*x*), *x* = −*Pi* .. *Pi*]);

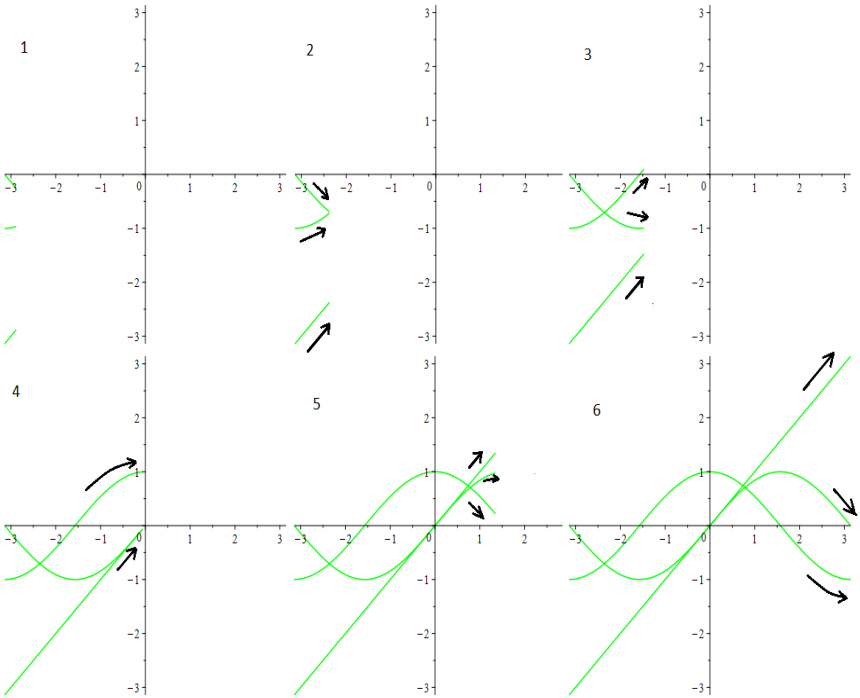


Давайте добавим дополнительные параметры, чтобы посмотреть, как поведёт себя график:

animatecurve([*sin*(*x*), *cos*(*x*), *x* = −*Pi* .. *Pi*], *view* = [−*1..2*, −*2..2*], *scaling* = *constrained*);

Теперь изобразим на координатной плоскости несколько функций одновременно:

animatecurve({*x*, *cos*(*x*), *sin*(*x*)}, *x* = −*Pi* .. *Pi*, *color* = *green*, *frames* = *50*);



Давайте выделим каждое уравнение отдельным цветом:

animatecurve({*x*, *cos*(*x*), *sin*(*x*)}, *x* = −*Pi* .. *Pi*, *color* = [*blue*, *red*, *green*], *frames* = 50);

Заметьте, что цвет мы указываем через запятую в специальном месте кода.

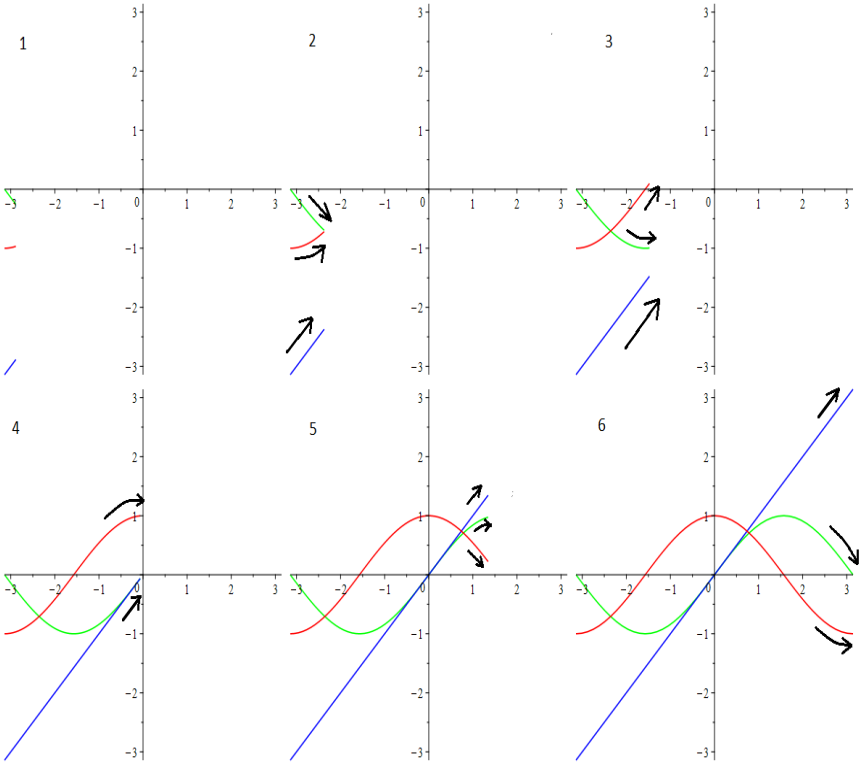
Или иначе: каждое уравнение обозначим через отдельную переменную:

q1 := *animatecurve*(*sin*(*x*), *x* = −*Pi* .. *Pi*, *color* = *green*, *frames* = 50);

q2 := *animatecurve*(*x*, *x* = −*Pi* .. *Pi*, *color* = *blue*, *frames* = 50);

q3 := *animatecurve*(*cos*(*x*), *x* = −*Pi* .. *Pi*, *frames* = 50);

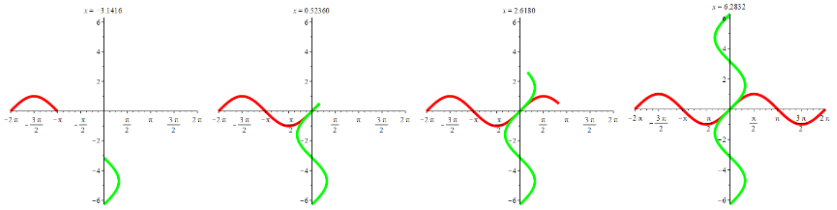
display(*q1*, *q2*, *q3*);


1
2
3
4
5
6

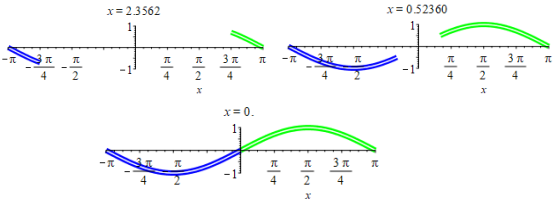
Практическое занятие 7

Задание 1. Нарисовать вращающийся треугольник

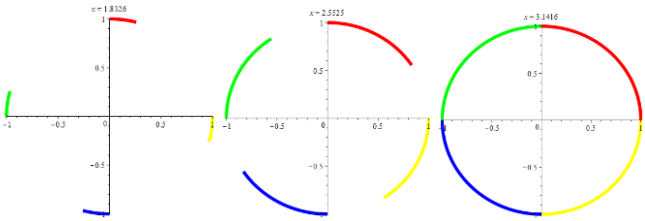
Задание 2. Нарисовать одновременную анимацию двух синусоид



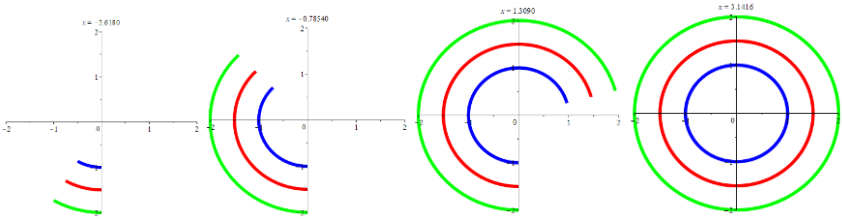
Задание 3. Нарисовать встречную анимацию двух синусоид



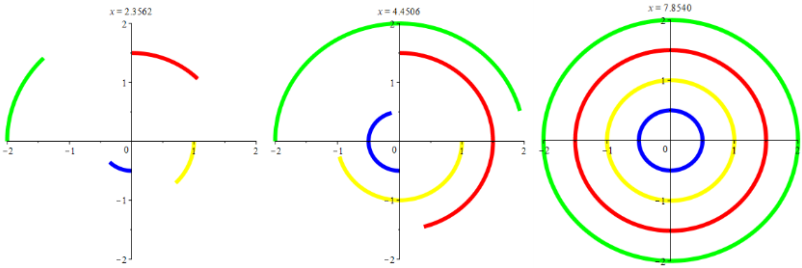
Задание 4. Нарисовать анимацию окружности



Задание 5. Нарисовать параллельную анимацию окружностей



Задание 6. Нарисовать анимацию окружностей со сдвигом



Лекции 8, 9. Встроенная анимация

Для изучения темы в рабочее окно введём новую функцию:

with(*plots*);

with(*plottools*);

Данная функция поможет нам при дальнейшей работе, чтобы использовать новые опции. Начнём:

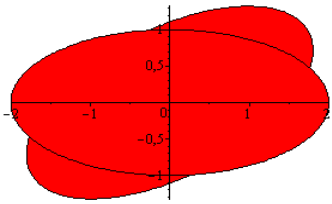
Rotate (Поворот)

Иногда на графике нам требуется посмотреть на одну и ту же фигуру под разным углом, желательно, чтобы изображен был как исходный рисунок, так и перевернутый под углом. Возьмём знакомый нам эллипс или по-простому овал:

s1 := *ellipse*([*0, 0*], *2, 1, filled, color = red*);

s11 := *display*(*rotate*(*s1*, (*1/6*)*Pi*));

display(*s1, s11, scaling = constrained*);

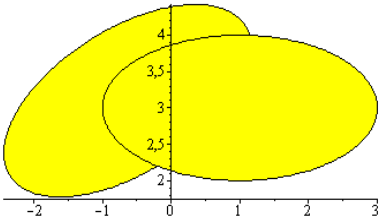


Под s1 мы рисуем эллипс, а затем создаем дополнительную функцию s11. В скобках, после указания опции rotate, мы берем нашу фигуру s1 и через запятую указываем на какой угол нам стоит её повернуть. Давайте ещё раз:

s2 := *ellipse*([*1, 3*], *2, 1, filled, color = yellow*);

s22 := *display*(*rotate*(*s2*, (*1/6*)*Pi*));

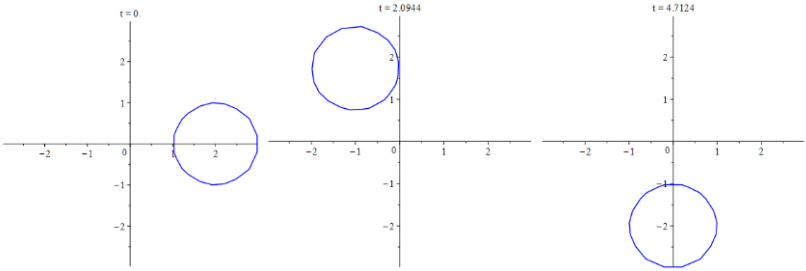
display(*s2, s22, scaling = constrained*);



Теперь воспользуемся с знакомыми нами ранее функциями и объединим их с нашей темой:

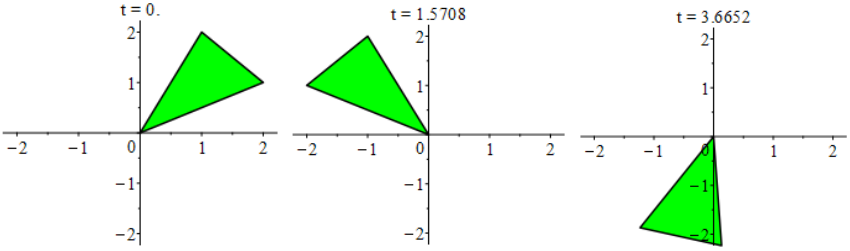
s2 := implicitplot((x−2)^2+y^2 = 1, x = −2 .. 5, y = −5 .. 5, color = blue);

*animate(rotate, [s2, t], t = 0 .. 4*Pi);*



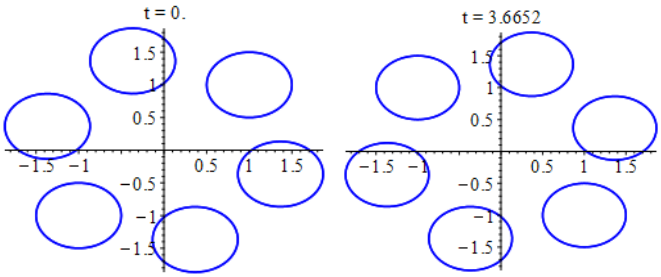
А теперь подобным образом анимируем треугольник:

*s3 := polygonplot([[[0, 0], [2, 1], [1, 2], [0, 0]]], color = green); animate(rotate, [s3, t], t = 0 .. 4*Pi);*



Ну что же, мы вспомнили опцию анимации, даже построение фигуры по точкам. Теперь научимся зацикливать наши фигуры, чтобы при воспроизведении рисунок делал одно и то же движение несколько раз:

```
c := circle([1, 1], .5, color = blue);
r1 := seq(rotate(c, (1/3)*Pi*i), i = 1 .. 6);
r2 := display(r1, r1); display(r2);
```

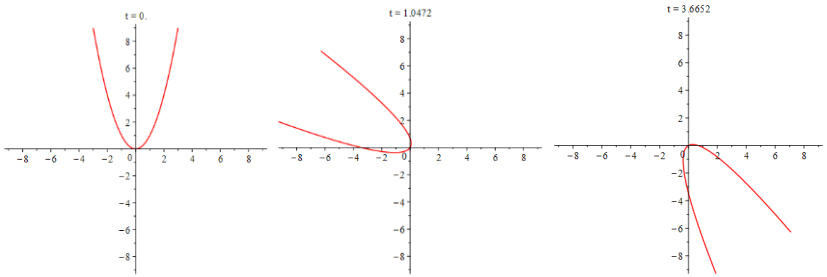


Немного напоминает кабинки из-под колеса обозрения, не так ли? Давайте заставим наши окружности двигаться по одной траектории, и всё что нам для этого нужно – приписать маленькую строчку кода:

```
animate(rotate, [r2, t], t = 0 .. 4*Pi);
```

Ну а теперь вернёмся к знакомой параболе и заставим двигаться её по траектории окружности, не сдвигая центра.

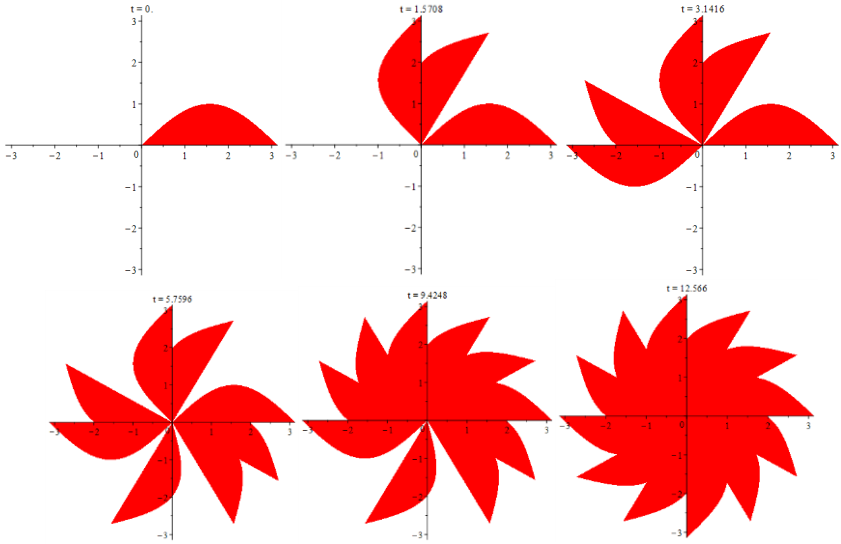
```
s4 := plot(x^2, x = −3 .. 3);
animate(rotate, [s4, t], t = 0 .. 4*Pi);
```



Сейчас, зная этот трюк, мы попробуем создать цветок, для этого возьмем график синуса, анимируем и зафиксируем некоторые её положения на плоскости:

a1 := plot(sin(x), x = 0 .. Pi, filled = true,color=red);

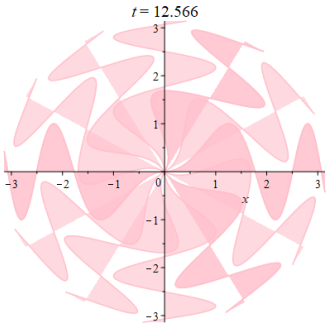
*animate(rotate, [a1, t], t = 0 .. 4*Pi, trace = 15);*



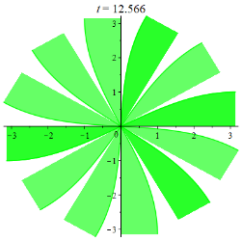
Теперь мы можем создавать различные варианты этого приема:

*a1 := plot(sin(x^2), x = 0 .. 2*Pi, color = pink, filled = true);*

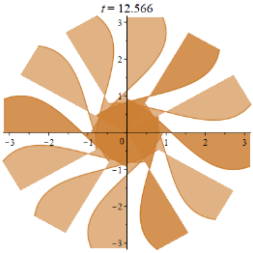
*animate(rotate, [a1, t], t = 0 .. 4*Pi, trace = 15);*



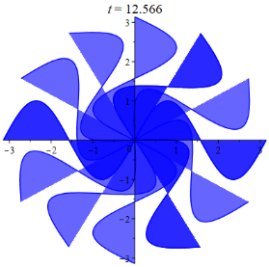
```
a1 := plot(sin(x/2), x = 0 .. Pi, filled, color = green);
animate(rotate, [a1, t], t = 0 .. 4*Pi, trace = 15);
```




```
a1 := plot(sin(x + 2), x = 0 .. Pi, filled, color = gold);
animate(rotate, [a1, t], t = 0 .. 4*Pi, trace = 15);
```

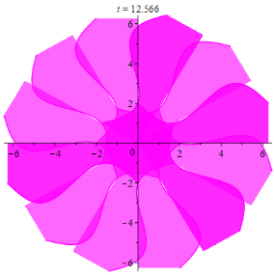



```
a1 := plot(sin(2*x), x = 0 .. Pi, filled, color = blue);
animate(rotate, [a1, t], t = 0 .. 4*Pi, trace = 15);
```



*a1 := plot(sin(2 + x), x = 0 .. 2*Pi, color = yellow, filled = true); animate(rotate, [a1, t], t = 0 .. 4*Pi, trace = 15,axes=none);*

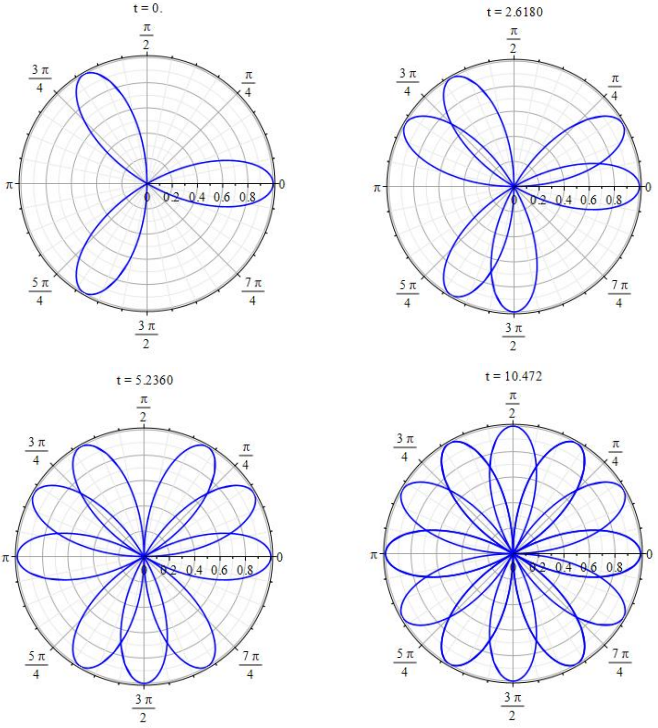
*a1 := plot(2 − sin(x), x = 0 .. 2*Pi, color = magenta, filled = true); animate(rotate, [a1, t], t = 0 .. 4*Pi, trace = 15);*



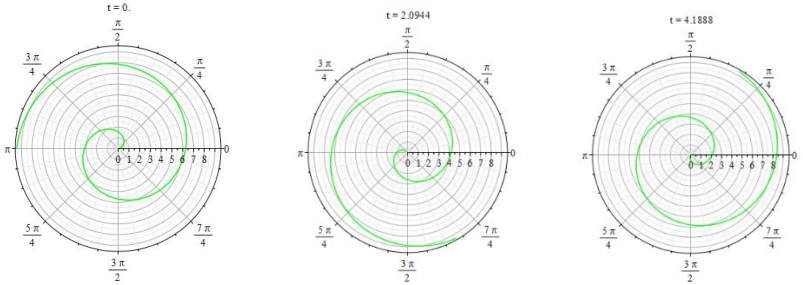
Попробуйте и вы сделать подобный цветок, изменив цвет или формулу изначального графика.

Такой же трюк сработает и в полярных координатах:

*a3 := polarplot(cos(3*x), x = 0 .. 5*Pi, color = blue); animate(rotate, [a3, t], t = 0 .. 4*Pi, trace = 5);*

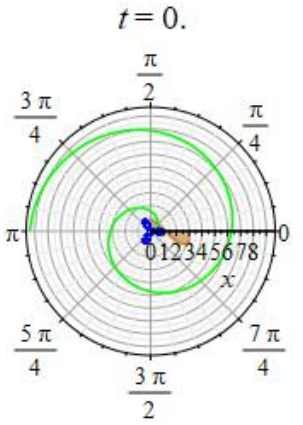


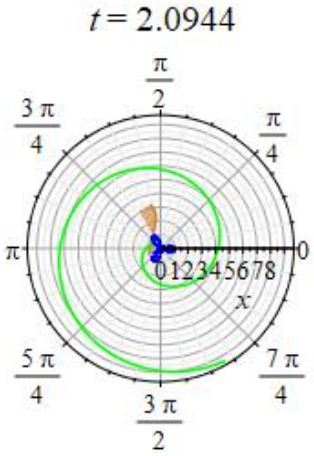
*a2 := polarplot(x, x = 0 .. 3*Pi, color = green);*
*animate(rotate, [a2, t], t = 0 .. 4*Pi);*



Повторим материал прошлого параграфа и объединим наши три фигуры:

a := *display*(*a1*, *a3*, *s3*); *animate*(*rotate*, [*a*, *t*], *t* = 0 .. 4*Pi);





Посмотрим как происходит вращение надписи:

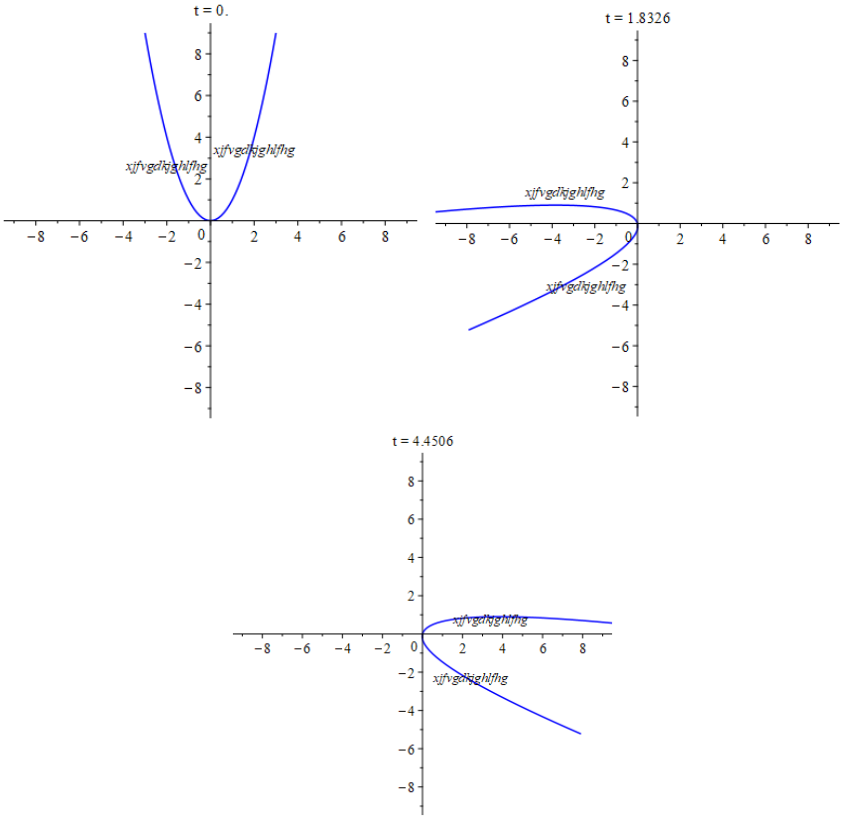
t1 := *plot*(*x^2*, *x* = −3 .. 3, *color* = *blue*):

t2 := *textplot*([2, 3, 'xjfvgdkjghlfhg'], *align* = *ABOVE*):

t3 := *textplot*([−2, 3, 'xjfvgdkjghlfhg'], *align* = *BELOW*):

t4 := *display*(*t1*, *t2*, *t3*):

animate(*rotate*, [*t4*, *t*], *t* = 0 .. 2*Pi);

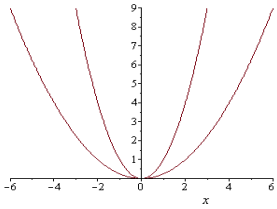


Видим, что надпись, в отличие от графика не переворачивается.

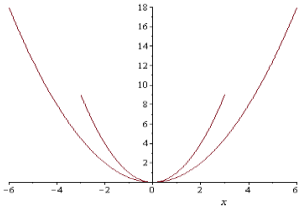
Опция scale (масштабирование)

Иногда нам требуется изменить размер графика, его длину, ширину и так далее. Для этого совсем необязательно создавать новый график или переписывать старый, достаточно воспользоваться опцией scale. Начнём знакомство с параболы:

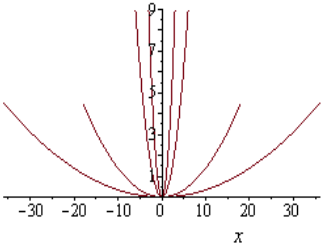
```
with(plots); with(plottools);
s1 := plot(x^2, x = −3 .. 3);
s2 := scale(s1, 2, 1);
display(s1, s2);
```



a3 := *scale*(*s1*, *2*, *2*); *display*(*s1*, *a3*);

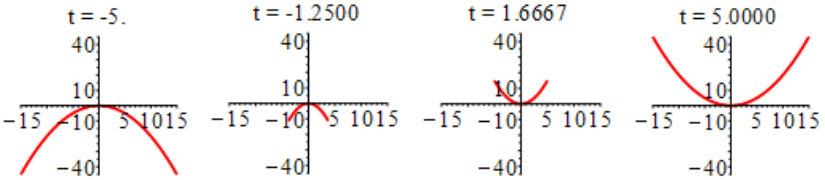


Теперь соеденим эти графики в одном рисунке:
s3 := *display*(*s1*, *s2*); *s4* := *scale*(*s3*, *6*, *1/2*); *display*(*s4*, *s3*);



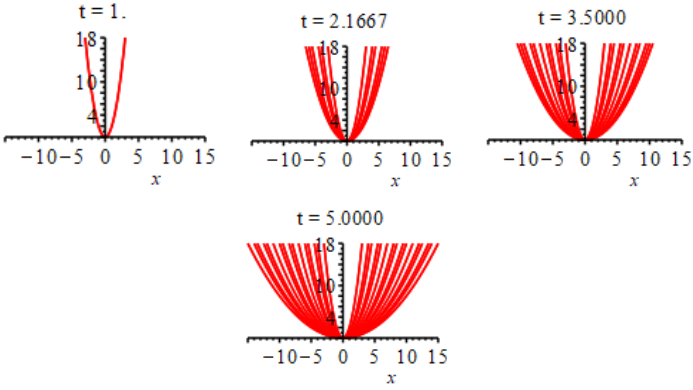
Сейчас добавим параметр, обратите внимание, что функция s4 повторяется дважды.

s4 := *animate*(*scale*, [*s1*, *t*, *t*], *t* = −*5* .. *5*);
display(*s4*, *s4*);



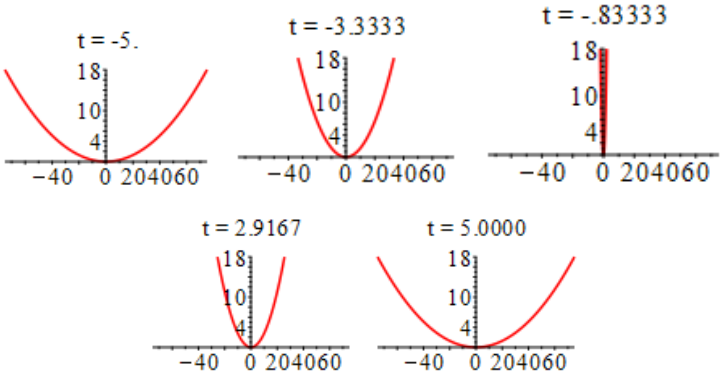
Если вам нужно зафиксировать некоторые положения этой параболы, то нужно добавить лишь функцию trace

```
s4 := animate(scale, [s1, t, 2], t = 1 .. 5, trace = 15);
display(s4, s4);
```



Заставим наш график поворачиватся по оси x, с условием, что мы находимся в трехмерном пространстве:

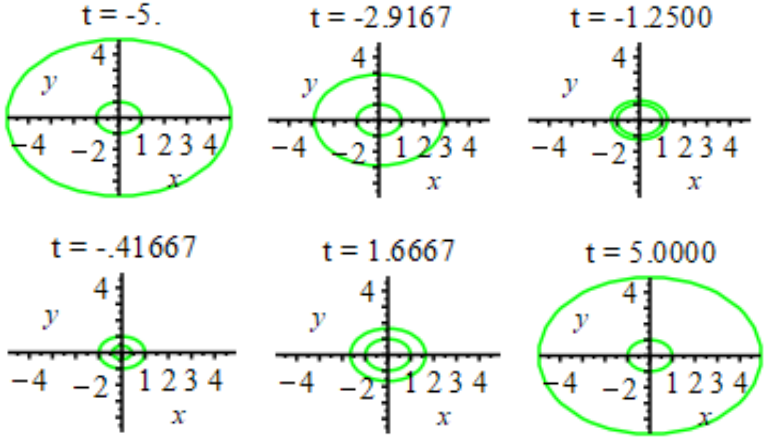
```
animate(scale, [s1, t^2, 2], t = −5 .. 5);
```



Попробуем проделать аналогичные действия с окружностью:

```
b1 := implicitplot(x^2+y^2 = 1, x = −3 .. 3, y = −3 .. 3,
 color = green);
b2 := animate(scale, [b1, t, t], t = −5 .. 5);
```

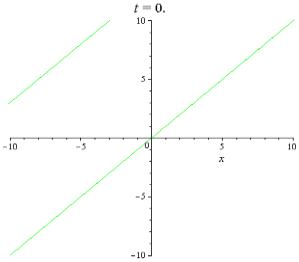
А если мы воспроизведём две эти функции одновременно, то на выходе получим вот такую забавную анимацию:



Опция translate (параллельный перенос)

Очень часто нам требуется перенести объект из одного места в другое. Сдвинуть линию на пару сантиметров вниз, перенести центр окружности влево, перенести вершину параболы правее изначального положения. Конечно, можно просто пересчитать координаты и перестроить уравнение заново, но что делать, если нам нужно увидеть два графика одновременно?

Для этого на помощь приходит опция translate.



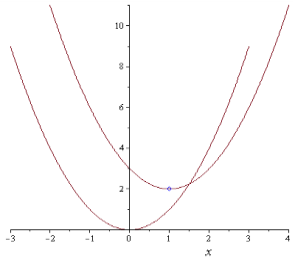
И снова вернёмся к нашей любимой параболе:

with(*plots*); *with*(*plottools*);

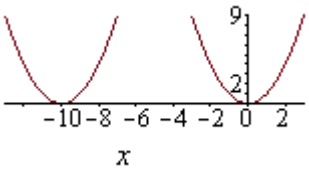
s1 := *plot*(*x*^2, *x* = −3 .. 3); *s2* := *translate*(*s1*, 1, 2);

a := *plot*([[1, 2]],

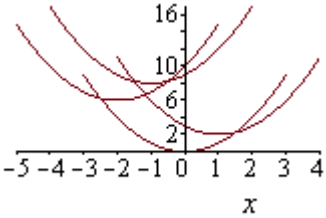
style = point, color = blue, symbol = DIAMOND,
symbolsize = 15);
display(s1, s2, a);



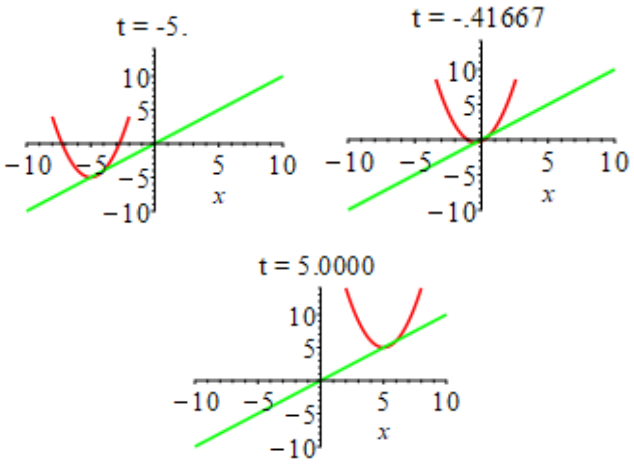
a2 := translate(s1, −10, 0); display(s1, a2);



Добавим третью и четвертую параболу:
s3 := display(s1, s2);
s4 := translate(s3, −2, 6);
display(s4, s3);

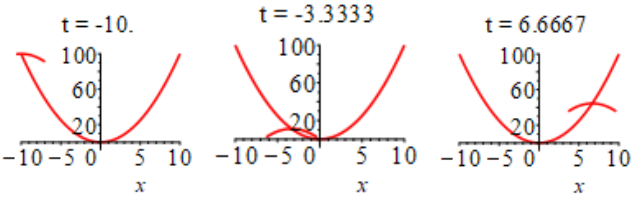


Перенос параболы вдоль прямой y = x:
s4 := animate(translate, [s1, t, t], t = −5 .. 5):
s5 := plot(x, x = −10 .. 10, color = green):
display(s4, s5);



Перемещение параболы с отрицательным коэффициентом по другой параболе:

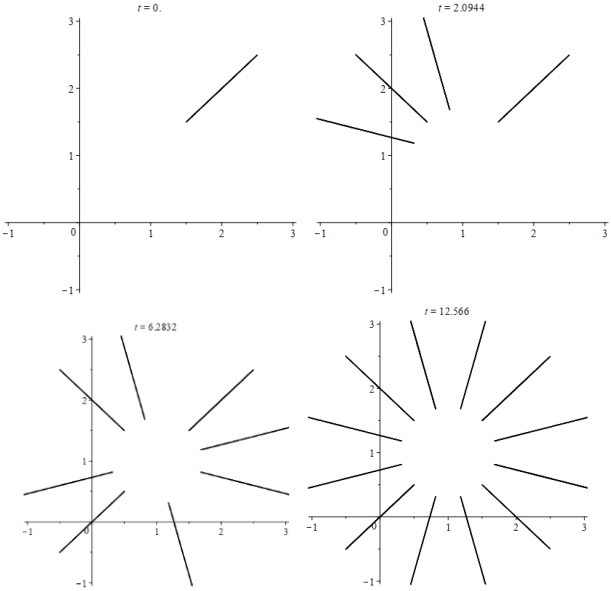
```
a1 := plot(−x^2, x = −3 .. 3):
s4 := animate(translate, [a1, t, t^2], t = −10 .. 10):
s5 := plot(x^2, x = −10 .. 10):
display(s4, s5);
```



Соединим обе команды rotate и translate:

```
s3 := polygonplot([[[0.5, 0.5], [1.5, 1.5]]]):
s4 := animate(rotate, [s3, t], t = 0 .. 4*Pi, trace = 15):
s5 := translate(s4, 1, 1):
```

display(*s5*, *s5*);



Так мы получаем вращение в любой точке плоскости.

Опция homothety (подобие)

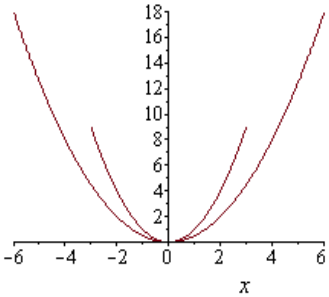
Когда нам нужная точно такая же функция или объект, но в другом месте, на помощь приходит данная функция.

with(*plots*); *with*(*plottools*);

s1 := *plot*(*x*^2, *x* = −3 .. 3);

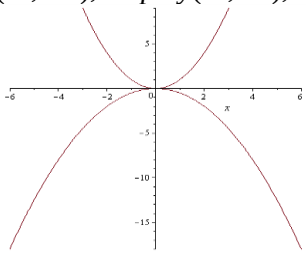
s2 := *homothety*(*s1*, 2);

display(*s1*, *s2*);



Добавив знак минус координате y, мы отразим её вниз:

a3 := *homothety*(*s1*, −*2*); *display*(*s1*, *a3*);

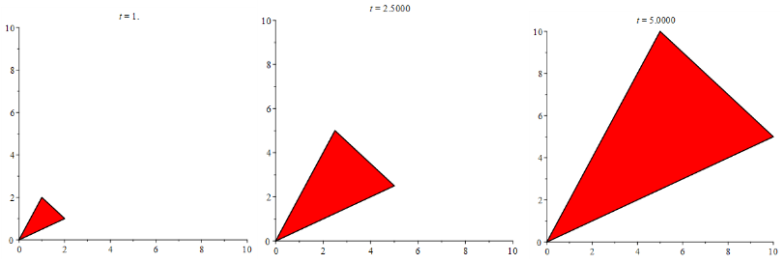


Такой трюк можно провернуть не только с обычной функцией, но и с объектом:

b1 := *polygonplot*([[*0*, *0*], [*1*, *2*], [*2*, *1*]], *color* = *red*);

b2 := *animate*(*homothety*, [*b1*, *t*], *t* = *1* .. *5*);

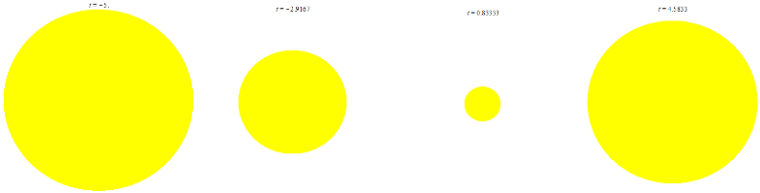
display(*b2*, *b2*);



s1 := *display*(*disk*([*0*, *0*], *1*, *color* = *yellow*, *style* = *polygon*));

s2 := *animate*(*homothety*, [*s1*, *t*], *t* = −*5* .. *5*);

display(*s2*, *s1*, *axes* = *NONE*);



Опция project (проекция)

Не секрет, что в математике мы очень часто проецируем какой-нибудь объект на прямую, данная функция применяется во многих областях.

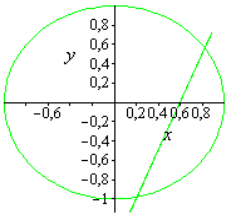
В начертательной геометрии, черчении, географии (специальные виды проектирования на плоскость, сферу и другие поверхности используются для составления карт), астрономии (устройство астролябии, с помощью которой можно измерять углы между направлениями на поверхности Земли и фиксировать координаты космических тел на небесной сфере), фотографии (используется для отображения сферических панорам).

with(*plots*); *with*(*plottools*);

s1 := *implicitplot*(*x^2+y^2 = 1, x = −3 .. 3, y = −3 .. 3, color = green*);

s2 := *project*(*s1*, [[*1, 1*], [*3, 6*]]);

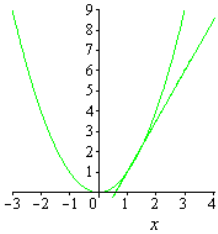
display(*s1, s2*);



d1 := *plot*(*x^2, x = −3 .. 3, color = green*);

d2 := *project*(*d1*, [[*1, 1*], [*3, 6*]]);

display(*d1, d2*);



Опция reflect (симметрия относительно точки)

Симметрия окружает нас повсюду, любому студенту физмата ясно как день, что природа полна примерами симметрии: форма листьев, цветов, бабочки и многое другое. Неудивительно, что такое многообразие помогает нам создать красивый эстетический образ в дизайне, архитектуре или в декоративно-прикладном искусстве.

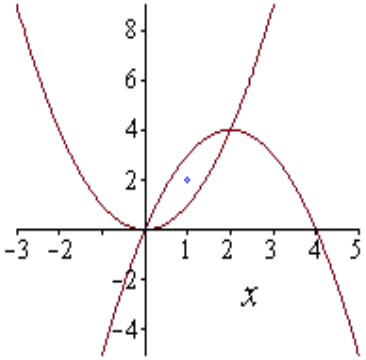
with(*plots*); *with*(*plottools*);

s1 := *plot*(*x^2, x = −3 .. 3*);

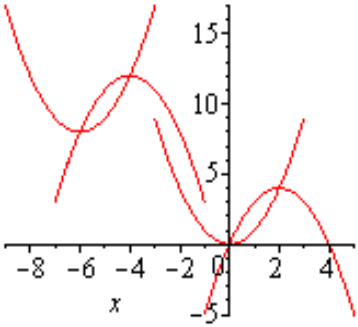
s2 := *reflect*(*s1*, [*1*, *2*]);

a := *plot*([[*1*, *2*]], *style* = *point*, *color* = *blue*, *symbol* = *DIAMOND*, *symbolsize* = *15*);

display(*s1*, *s2*, *a*);



s3 := *display*(*s1*, *s2*); *s4* := *reflect*(*s3*, [−*2*, *6*]); *display*(*s4*, *s3*);



#Анимация вращения параболы

b1 := *animate*(*rotate*, [*s5*, *t*], *t* = *0* .. *4*Pi*):

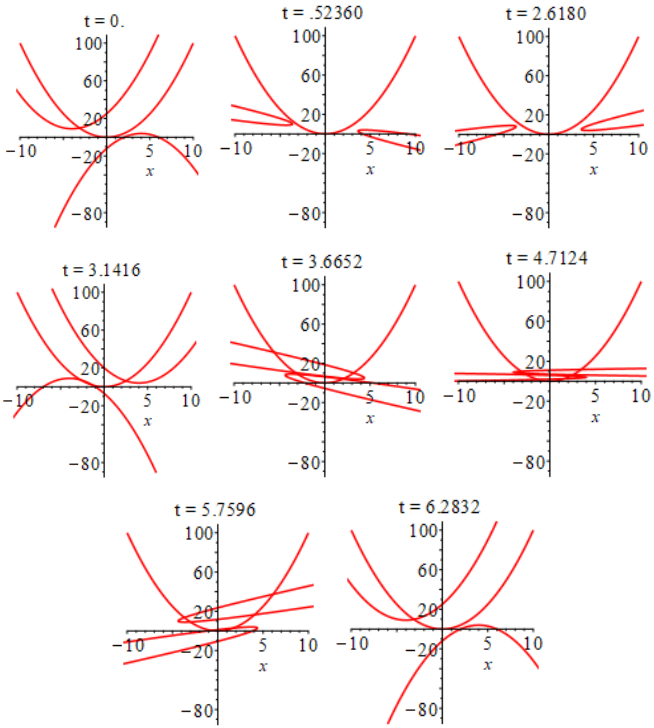
#Дублирование вращающейся параболы

b2 := *reflect*(*b1*, [*2*, *2*]):

#Перемещение сдублированной параболы

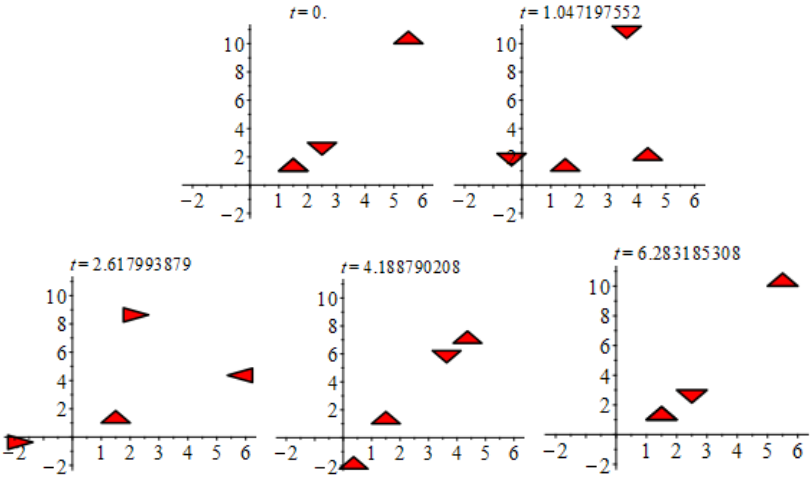
b3 := *translate*(*b1*, −*4*, *9*):

display(*s5*, *b2*, *b3*);



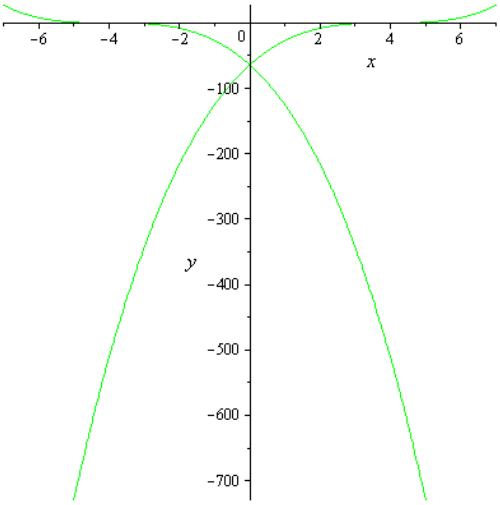
«Самолетики летают»:

```
f1 := polygonplot([[1, 1], [1.5, 1+(1/2)*sqrt(3)], [2, 1]],
color = red);
f2 := animate(rotate, [f1, t], t = 0 .. 4*Pi);
f3 := reflect(f2, [2, 2]);
f4 := translate(f2, 4, 9);
display(f1, f3, f2, f4);
```



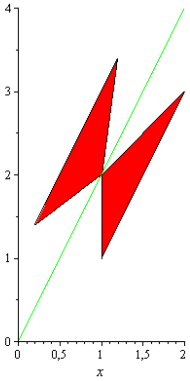
В заключении рассмотрим симметрию относительно прямой:

```
r1 := plot((x−4)^3, x = −5 .. 7, y = −10 .. 10, color = green);
r2 := reflect(r1, [[0, 0], [0, 1]]);
display(r1, r2);
```



Здесь прямой выступает ось ОУ, а далее наклонная прямая:

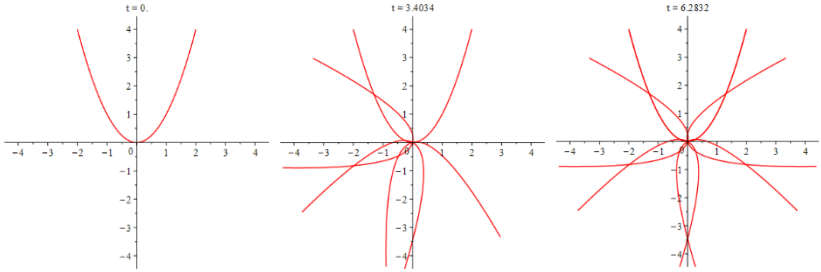
```
r1 := polygonplot([[1, 1], [1, 2], [2, 3]], color = red);
r2 := reflect(r1, [[0, 0], [1, 2]]);
r3 := plot(2*x, x = 0 .. 2, color = green);
display(r1, r2, r3, scaling = constrained);
```



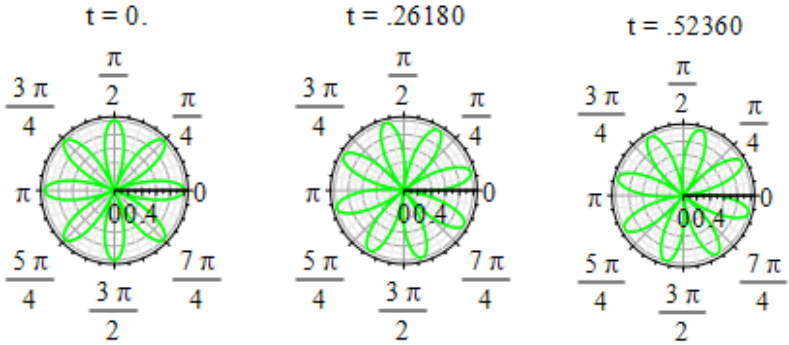
Теперь мы освоили основные приемы работы с графикой в Maple, а значит, сможем попробовать себя в мультипликационном деле и создать короткий мультик, попробуйте!

Практические задания 8, 9

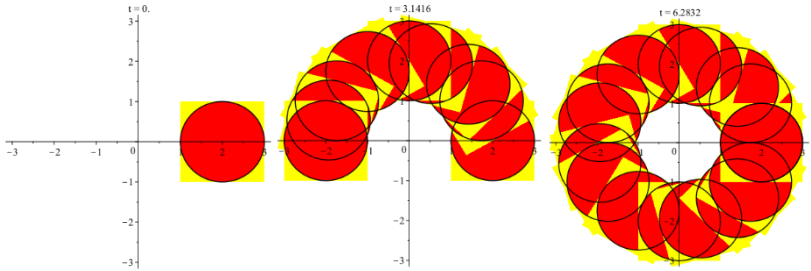
Задание 1. Нарисовать вращающуюся параболу, зафиксировать 5 положений



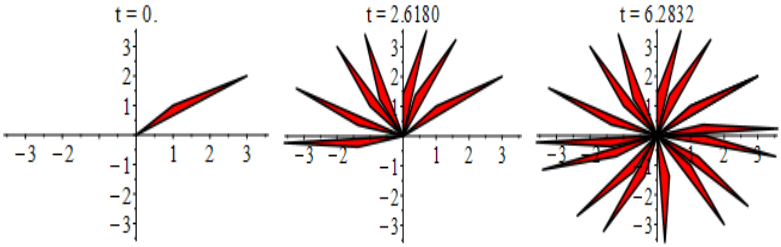
Задание 2. Нарисовать вращающийся цветок



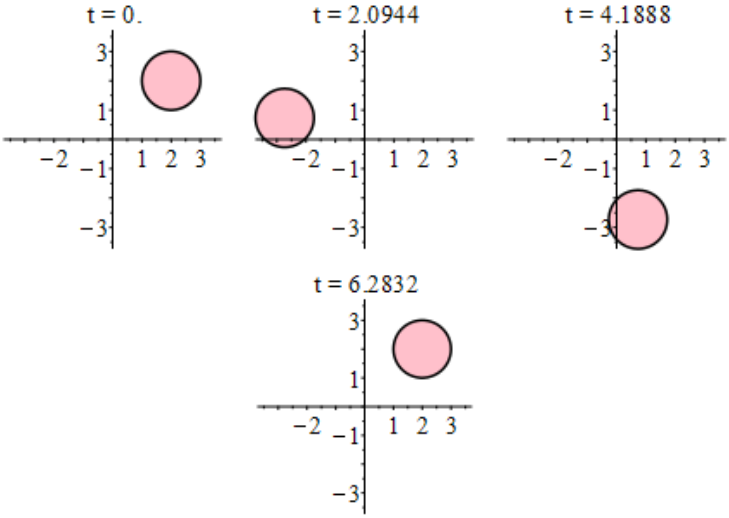
Задание 3. Нарисовать вращающийся квадрат с кругом, зафиксировать 15 положений



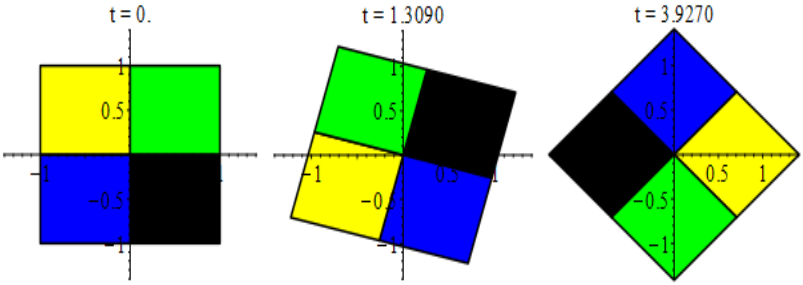
Задание 4. Нарисовать вращающийся треугольник, зафиксировать 15 положений



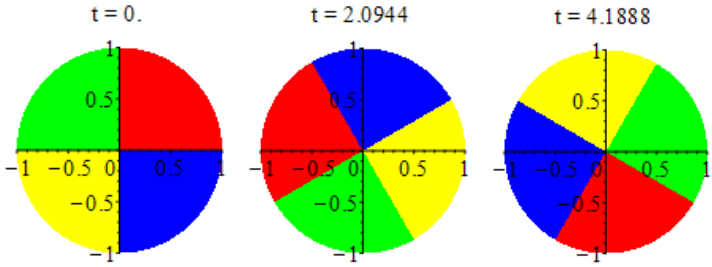
Задание 5. Нарисовать вращающийся круг



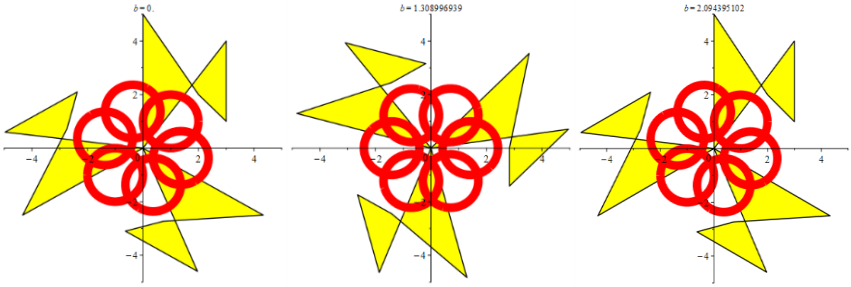
Задание 6. Нарисовать вращающийся цветной квадрат



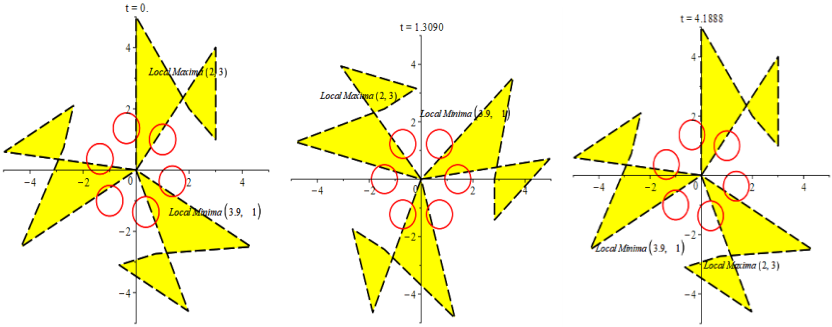
Задание 7. Нарисовать вращающийся цветной круг



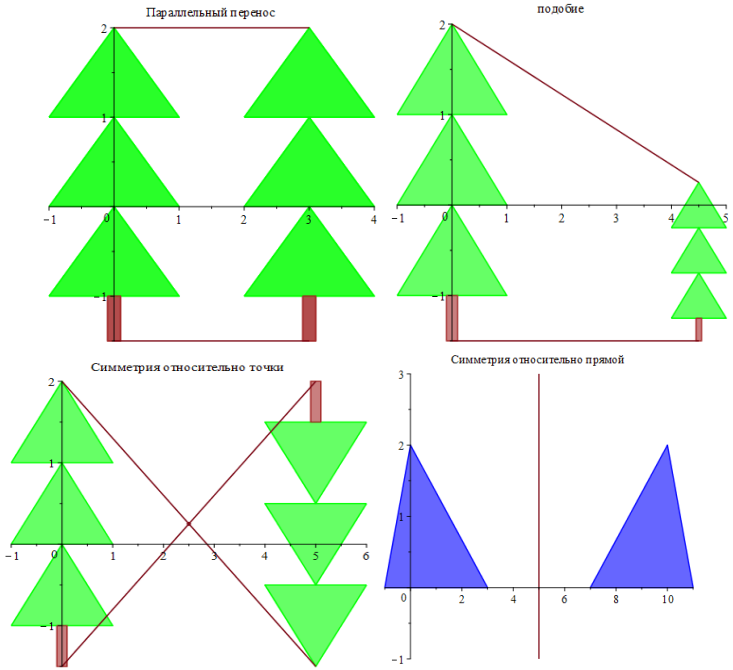
Задание 8. Нарисовать вращающуюся картинку



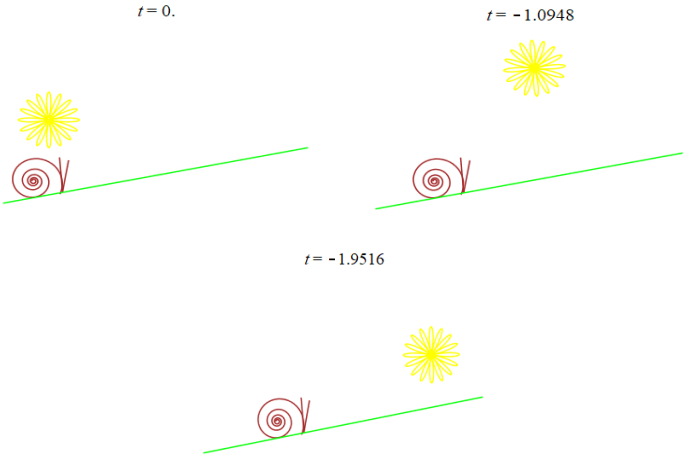
Задание 9. Нарисовать вращающуюся картинку с надписью



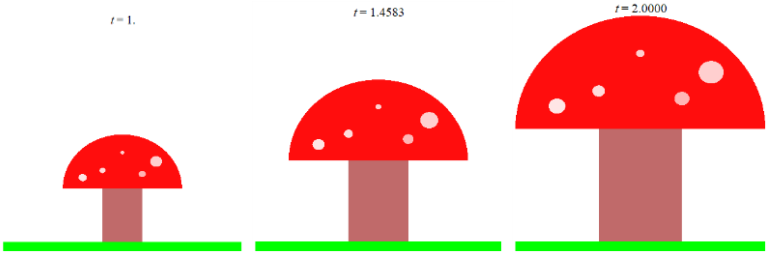
Задание 10. Нарисовать елочки с демонстрацией встроенных преобразований. Сделать надписи.



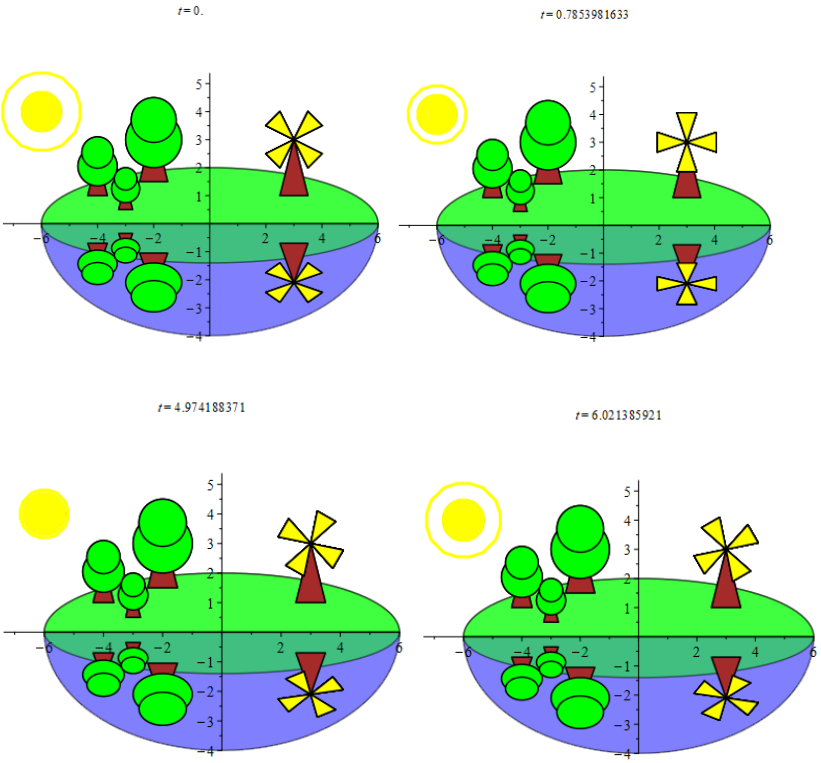
Задание 11. Нарисовать улитку в движении



Задание 12. Нарисовать рост гриба



Задание 13. Нарисовать остров с мельницей, деревьями и солнцем. Задать анимацию



Лекция 10. Смена кадров и слайдов

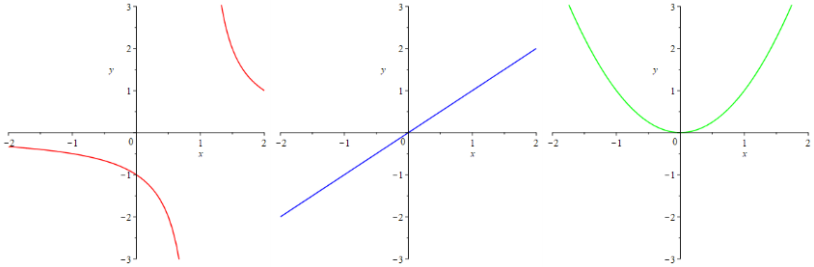
Следующая интересная графическая возможность пакета Maple, это смена кадра, команда insequence. Она позволяет быстро менять кадры, задающие разные функции:

q1 := plot(1/(x − 1), x = −2 .. 2, y = −3 .. 3, color = red, discont = true);

q2 := plot(x, x = −2 .. 2, y = −3 .. 3, color = blue);

q4 := plot(x^2, x = −2 .. 2, y = −3 .. 3, color = green);

display(q1, q2, q4, insequence = true);



Создается эффект анимации, хотя здесь ее нет.

А теперь посмотрим те же функции, добавим еще одну и сделаем добовление графиков:

q1 := plot(1/(x − 1), x = −2 .. 2, y = −3 .. 3, discont = true);

q2 := plot(x, x = −2 .. 2, y = −3 .. 3, color = blue);

q3 := plot(cos(x), x = −2 .. 2, y = −3 .. 3, color = red);

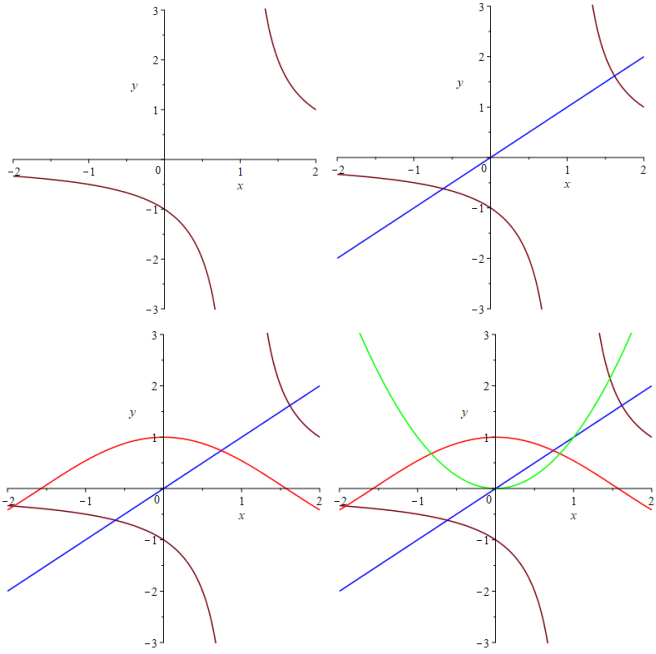
q4 := plot(x^2, x = −2 .. 2, y = −3 .. 3, color = green);

q11 := display(q1, q2);

q12 := display(q1, q2, q3);

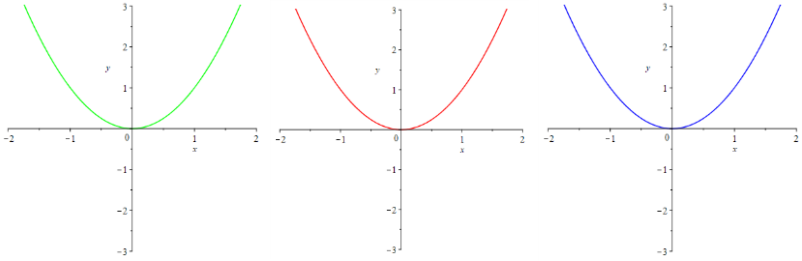
q13 := display(q1, q2, q3, q4);

display(q1, q11, q12, q13, insequence = true);



С помощью смены кадра можно получить эффект смены цвета, задавая одну и ту же функцию:

q1 := plot(x^2, x = −2 .. 2, y = −3 .. 3, color = green);
q2 := plot(x^2, x = −2 .. 2, y = −3 .. 3, color = red);
q3 := plot(x^2, x = −2 .. 2, y = −3 .. 3, color = blue);
display(q1, q2, q3, insequence = true);



Все происходит очень быстро, смена цвета едва улавливается.

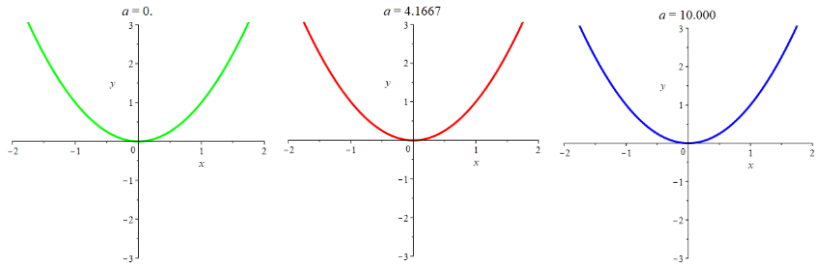
Можем замедлить переключение задавая интервал анимации через параметр:

q1 := animate(plot, [x^2, x = −2 .. 2, y = −3 .. 3, color = green, thickness = 3], a = 0 .. 10);

q2 := animate(plot, [x^2, x = −2 .. 2, y = −3 .. 3, color = red, thickness = 3], a = 0 .. 10);

q3 := animate(plot, [x^2, x = −2 .. 2, y = −3 .. 3, color = blue, thickness = 3], a = 0 .. 10);

display([q1, q2, q3], insequence);



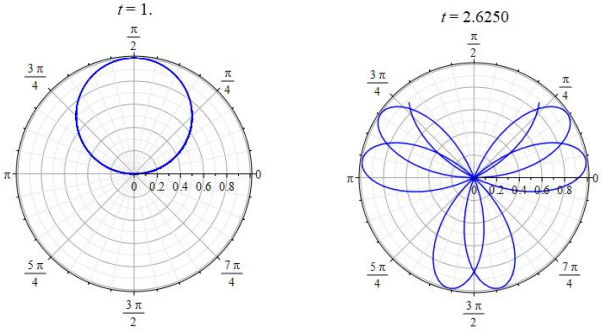
Теперь соединим две анимации, при которых одна функция плавно перейдет в другую с помощью смены кадра. Возьмем уже известные нам анимации:

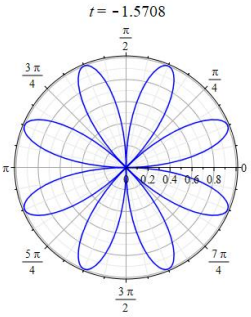
*a1 := animate(polarplot, [[sin(x*t), x, x = −4 .. 4], color = blue], t = 1 .. 4);*

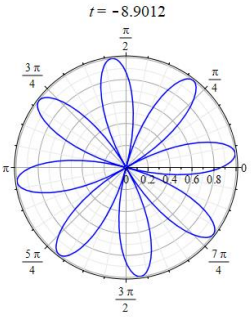
*a3 := polarplot(sin(4*x), x = 0 .. 2*Pi, color = blue);*

*a2 := animate(rotate, [a3, t], t = 0 .. −4*Pi);*

display([a1, a2], insequence = true);

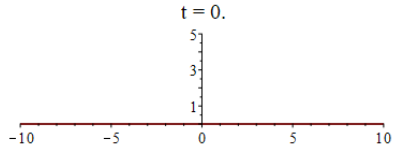


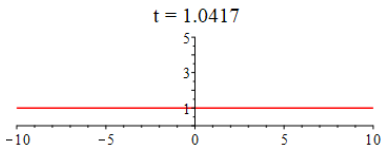


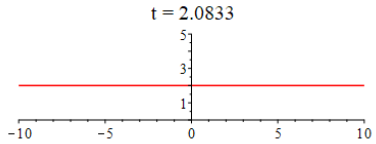


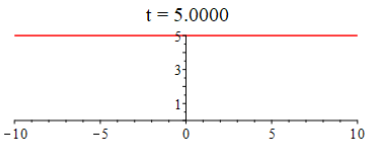
Следующая функция позволит сделать смену кадра дискретной, а не непрерывной. Как было до этого. Это функция trunc(t), задающая целую часть числа, график этой функции мы рисовали на первых занятиях.Смотрим первый пример, дискретное движение прямой:

animate(*plot*, [*trunc*(*t*)], *t = 0 .. 5, scaling = constrained*);



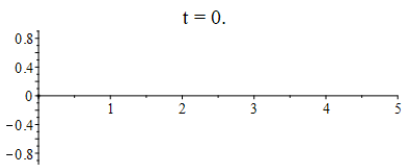


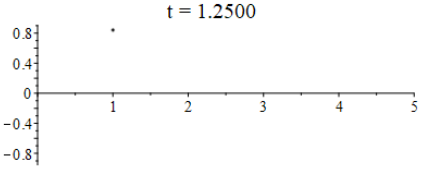


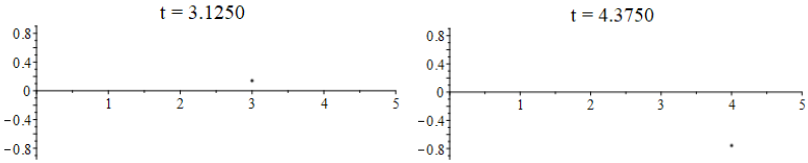


Следующий пример: точка, прыгающая по синусоидной траектории:

animate(*pointplot*, [[*trunc*(*t*), *sin*(*trunc*(*t*))]], *t = 0 .. 5, scaling = constrained*);

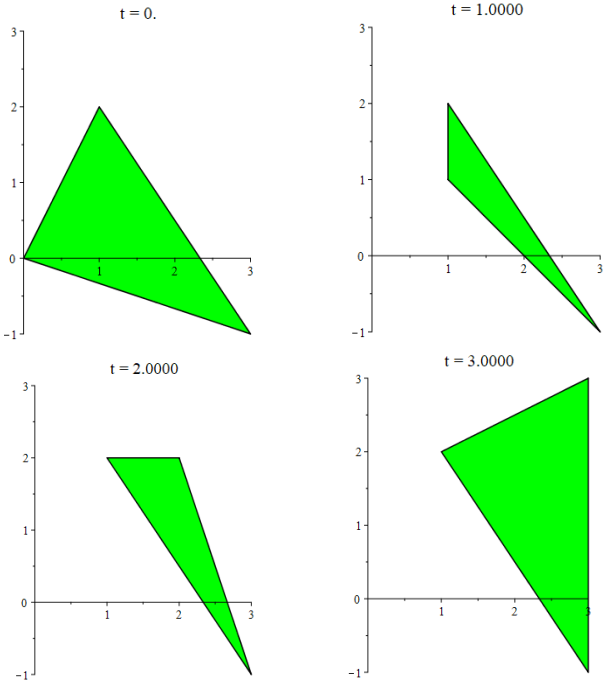






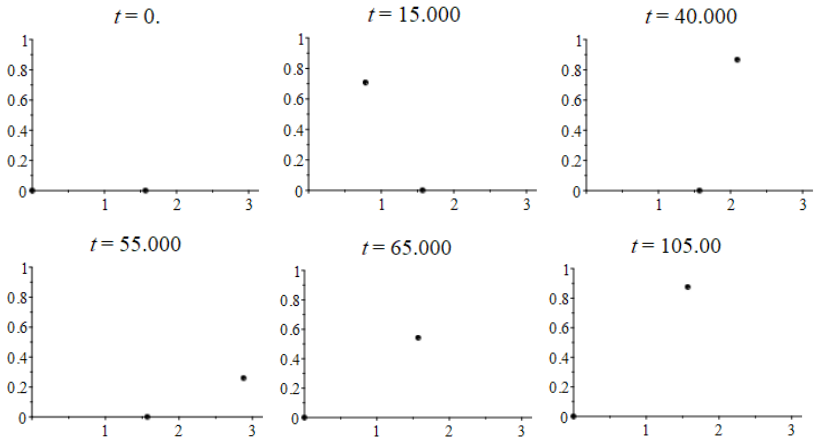
Скачкообразное изменение треугольника:

animate(*polygonplot*, [[[*trunc*(*t*), *trunc*(*t*)], [*1*, *2*], [*3*, −*1*]], *color* = *green*], *t* = *0* .. *3*, *scaling* = *constrained*);



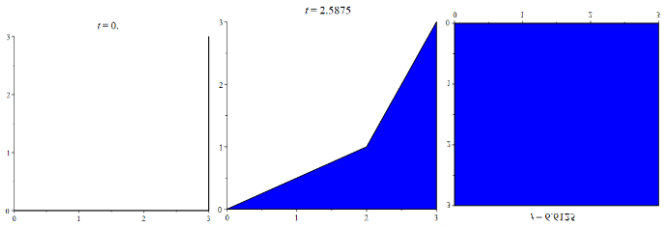
Теперь, движение одной точки по прямой меняется на движение другой точки по другой прямой:

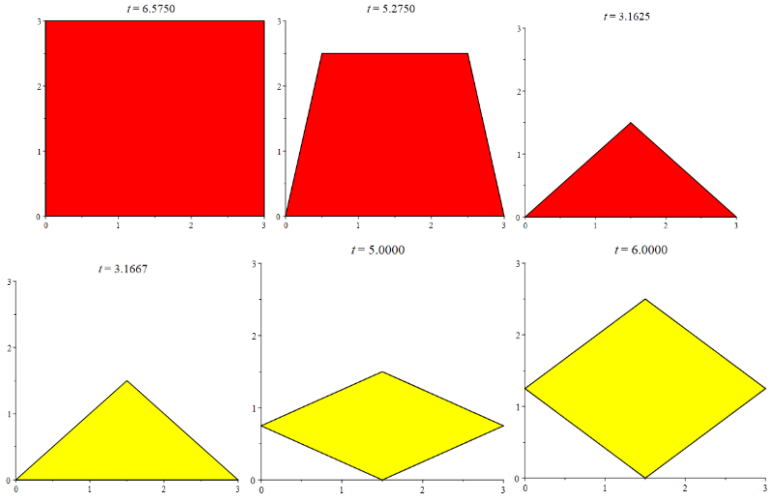
s1:=*animate*(*pointplot*,[[*Pi/2*,*trunc*(*t/62*)*(*t/120*)]]),*t*=*0..120*
,*symbol* = *solidCIRCLE*, *symbolsize* = *15*):
s2:=*animate*(*pointplot*,[[[(*1*−*trunc*(*t/62*))*Pi*(*t*)*/60*,
sin((*1*−*trunc*(*t/62*))*Pi*(*t*)*/60*)]],*t*=*0..120*):
display(*s1*,*s2*);



Соединим фнкцию trunc(t) и команду insequence, получим целый мультфильм о преобразовании многоугольников:

```
r1 := animate(polygonplot, [[[0, 0], [3, 0], [3, 3],
 [3 − trunc(t)/2, trunc(t)/2]], color = blue], t = 0 .. 6.9);
r2 := animate(polygonplot, [[[0, 0], [3, 0], [trunc(t)/2, trunc(t)/2],
[3 − trunc(t)/2, trunc(t)/2]], color = red], t = 6.9 .. 3);
r3 := animate(polygonplot, [[[0, 0.75*trunc(t/4) + trunc(t/6)/2],
[1.5, 0 + trunc(t/6)/2], [3, 0.75*trunc(t/4) + trunc(t/6)/2],
 [1.5, 1.5 + trunc(t/6)/2]], color = yellow], t = 3 .. 7);
r4 := animate(polygonplot, [[[0, 1.25], [1.5, 2 + 0.5*trunc(t/4)],
[3, 1.25], [1.5, 0.5 − 0.5*trunc(t/4)]], color = yellow], t = 0 .. 6);
display([r1, r2, r3, r4], insequence = true);
```





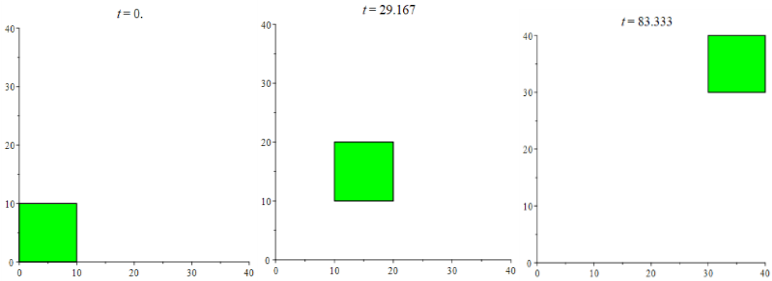
Следующий дискретный эффект анимации получим с помощью известной нам кусочно-заданной функции:

vel2 := *t* −> *rectangle*([*t*, *t*], [*t* + *10*, *t* + *10*], *color* = *green*);

vel3 := *t* −> *vel2*(*piecewise*(*t* <= *25*, *0*, *t* <= *50*, *10*, *t* <= *75*, *20*, *t* <= *100*, *30*));

disp2 := *t* −> *display*(*vel3*(*t*));

animate(*disp2*, [*t*], *t* = *0* .. *100*);

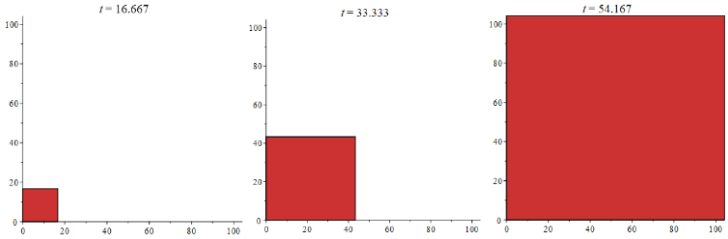


Здесь просто дискретное перемещение квадрата, а далее увеличение квадрата, при помощи аналогичной кусочно-заданной функции:

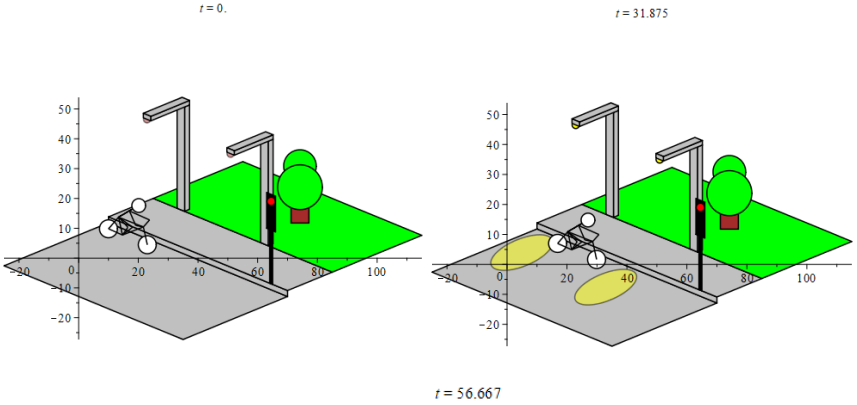
vel2 := *t* −> *rectangle*([*t*, *t*], [*0*, *0*], *color* = *orange*);

vel3 := *t* −> *vel2*(*piecewise*(*t* <= *25*, *t*, *t* <= *40*, *t* + *10*, *t* <= *55*, *t* + *50*, *t* <= *100*, −*t* + *100*));

```
disp2 := t -> display(vel3(t));
animate(disp2, [t], t = 0 .. 100);
```

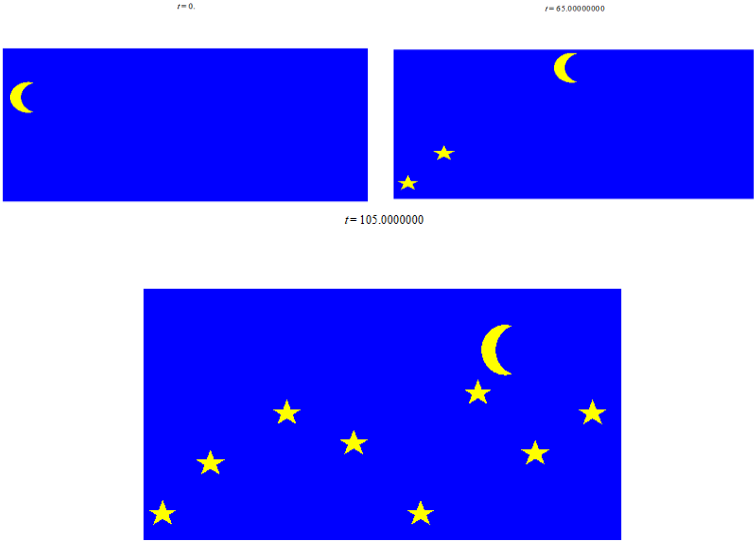


Следующий пример очень сложный, создает мультфильм, используя предложенный материал:



Практическое задание 10

Нарисовать звездное небо с последовательным появлением звезд и движение месяца:

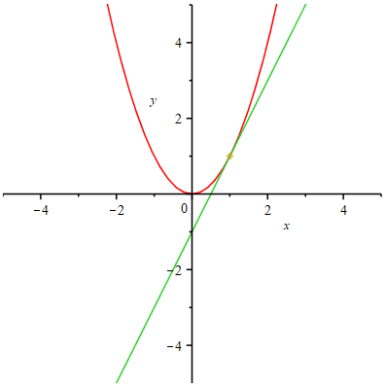


Лекция 11. Простейшее программирование в графике

1. Построение касательной к графику функции

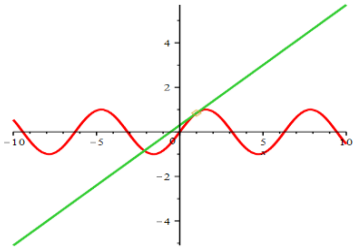
Одной из классических задач математического анализа является задача нахождения и построения касательной к графику функции. Рассмотрим эту задачу. Сначала проделаем вычисления сами «вручную» и зададим графики. Для параболы:

plot([*x*^2, 2**x* − 1, [[1, 1]]], *x* = −5 .. 5, *y* = −5 .. 5, *style* = [*line*, *line*, *point*], *symbolsize* = [15]);



Для синусоиды:

plot([*sin*(*x*), *sin*(1) + *cos*(1)*(*x* − 1), [[1, *sin*(1)]]], *x* = −10 .. 10, *style* = [*line*, *line*, *point*], *thickness* = [3, 3], *symbolsize* = [20]);



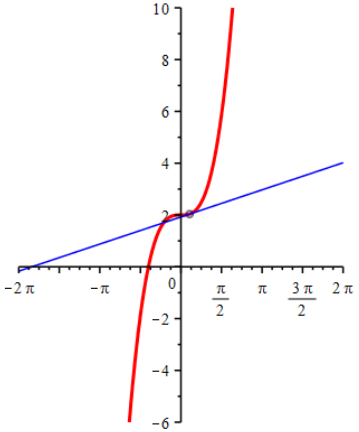
Теперь будем использовать умение программы вычислять производную для построения касательной к графику функции:

f:=*x*−>*x*^3+2:

g:=*diff*(*f*(*x*),*x*):

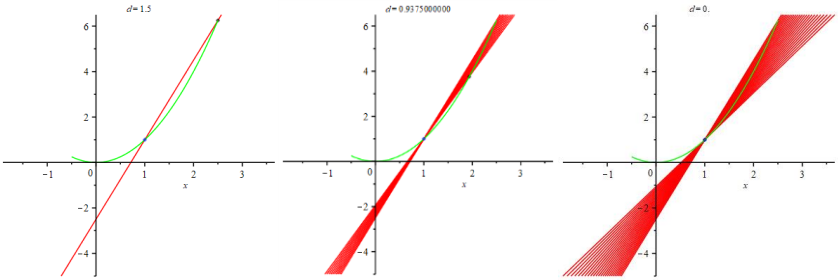
a:=1/3:

```
b:=eval(g,x=a):
g1:=plot(f(x),x= −2..2,thickness=3):
g2:=plot([[a,f(a)]],style=POINT,symbol=[CIRCLE],
symbolsize=[15],color=black):
g3:=plot(f(a)+b*(x−a),x= −2*Pi..2*Pi, color=blue):
display(g1,g2,g3,scaling=constrained);
```



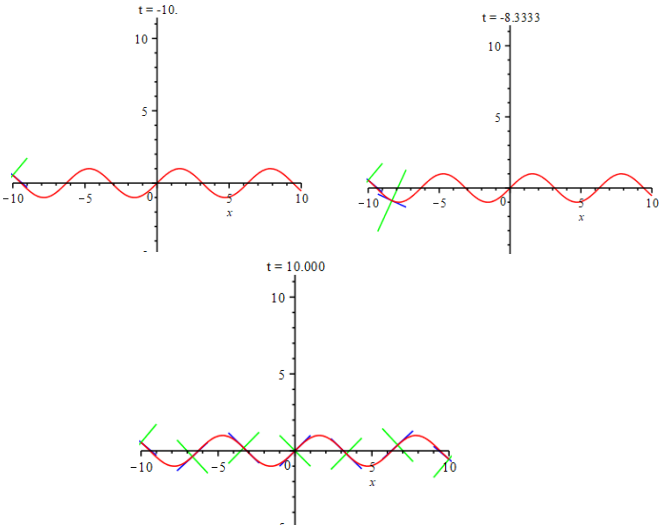
Продемонстрируем геометрический смысл производной через анимацию перехода секущей в касательную:

```
g4:=plot([[1,1]],style=POINT,color=blue):
g3:=animate(pointplot,([[1+d,(1+d)^2]]),d=1.5..0):
g1:=plot(x^2,x= −0.5..2.5,color=green):
g2:=animate(implicit-
plot,[(x−1)/d=(y−1)/((1+d)^2−1),x= −5..6.5,y= −5..6.5],d=1.5..0,trace
=24):
display(g1,g2,g3,g4);
```



Далее сделаем анимацию касательной и нормали к синусоиде:

```
a1 := plot(sin(x), scaling = constrained);
a2 := animate(plot, [cos(t)*(x − t) + sin(t), x = t − 1 .. t + 1,
 color = blue], t = −10 .. 10, trace = 6);
a3 := animate(plot, [− (x − t)/cos(t) + sin(t), x = t − 1 .. t + 1,
color = green], t = −10 .. 10, trace = 6);
display(a1, a2, a3);
```



2. Условия и циклы

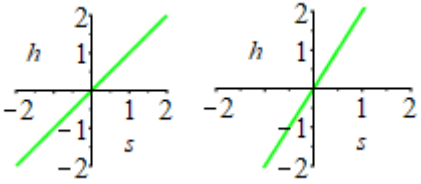
if `булево` `выражение` then `последовательность` `операторов` [elif `булево` `выражение` `последовательность` `операторов`] [else `последовательность` `операторов`] end if `оператор` `ветвления`;`операция"if"` "if" (`условие`, `операнд` 1,`операнд` 2); `операции`*`тернарного`*`условия``цикл` [for `имя`] [from `выражение`] [by `выражение`][to `выражение`] [while `булево` `выражение`] do `последовательность` `операторов` end do;

```
#Пример1
x:=5:
if x<0 then print("yes");
else print("no");
end if;
                    "no"
```

```
#Пример2
x:=2; if x>2 then print("no");
elif x<1 then print("yes");
else print("gol"):
end if;
                    "gol"
#Пример3
x:=1.3;
if x<=1 then print("no");
elif x<2 then print("yes");
else print("no"):
end if;
                    "yes"
#Пример4
x:=1.5;if x<=1 then f=0;
elif x<2 then f=1;
elif x<3 then f=2;
else f=3
end if;
                     1.5
                    f = 1
#Пример5
x:=4;
if x>0 then print("yes")end if;
                      4
                    "yes"
#Пример6
a:=1:b:=3:
c:=`if`(a<b,a,b);
                      1
#Пример7
a:=1:b:=3:
c:=`if`(a>b,cos(a),sin(b));
                    sin(3)
#Пример8
a:=4:b:=3:
c:=`if`(a<=1,cos(1),sin(2));
                    sin(2)
```

Далее рассмотрим примеры использования циклов и условий для построения графиков:

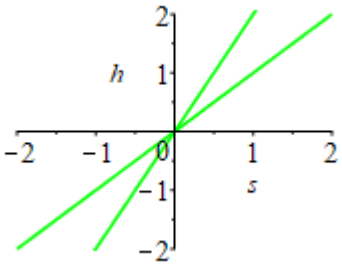
for i from 1 to 2 do plot(*i*s,s*= −*2..2,h*= −*2..2,color*=*green*)*end do;*



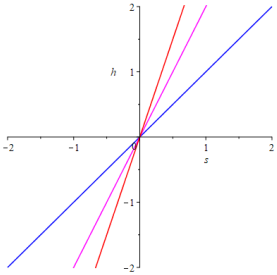
Для того, чтобы соединить полученные графики на одну картинку, необходимо перечислить результат в display:

for i from 1 to 2 do s(*i*):=*plot*(*i*s,s*= −*2..2,h*= −*2..2, color*=*green*)*end do;*

display(*s*(*1*),*s*(*2*));



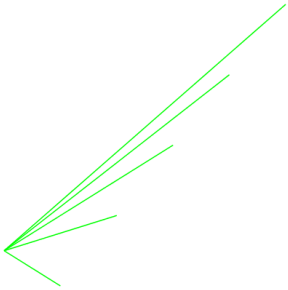
for i from 1 to 3 do s(*i*):=*plot*(*i*s,s*= −*2..2,h*= −*2..2,color =COLOR*(*RGB,i*−*1*, −*i,3*−*i*))*end do:display*(*s*(*1*),*s*(*2*),*s*(*3*));



Далее еще два примера:

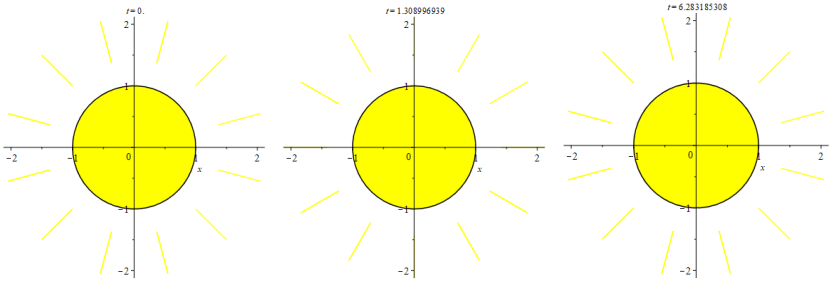
```
for n to 5 do
 s[n] := polarplot([[0, 0], [n, n*Pi/3]], color = green);
end do;
display(s[1], s[2], s[3], s[4], s[5], axes = none);
```

```
for n to 5 do
s[n] := plot([[[0, 0], [n, 2*n − 3]]], color = green);
end do;
display(s[1], s[2], s[3], s[4], s[5], axes = none);
```



А теперь с анимацией:

```
s := disk([0, 0], 1, color = yellow);
f1 := plot(x, x = 1 .. 1.5, color = yellow);
for i to 20 do
   d[i] := rotate(f1, Pi/6*i);
end do;
f2 := animate(rotate, [f1, t], t = 0 .. 2*Pi);
a := display(d[1], d[2], d[3], d[4], d[5], d[6], d[7], d[8], d[9],
d[10], d[11], d[12], d[13], d[14], d[15], d[16], d[17], d[18], d[19], d[20],
s, scaling = constrained);
animate(rotate, [a, t], t = 0 .. 2*Pi);
```



3. Процедуры

Далее познакомимся с простейшими процедурами:

Пример1

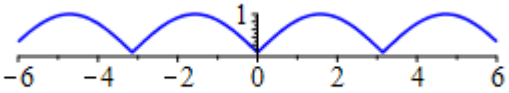
```
lc := proc(s, u, t, v) u*s + t*v; end proc;
lc(Pi, x, −i, y);
lc(1, 2, 3, 4);
```

$$Pi\,x - i\,y$$

$$14$$

Пример2

```
Y := proc(x) if 0 < sin(x) then sin(x); else −sin(x); end if; end proc;
plot(Y, −6 .. 6, color = blue, scaling = constrained);
```



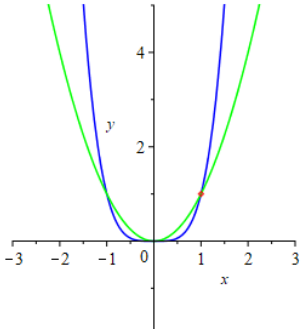
Пример3

```
restart;
ad := proc(a, b) local x, s, i; x := b; s := 0; for i to a do s := s + i; end do; s := s*x; end proc;
ad(5, 2);
```

$$30$$

Пример4

```
s := proc(f, x0) local f1; f1 := f^2; plot([f1(x), f(x), [[x0, f(x0)]]], x = −3 .. 3, y = −5 .. 5, style = [LINE, LINE, POINT], scaling = constrained, color = [blue, green, red]); end proc;
s(x −> x^2, 1);
```

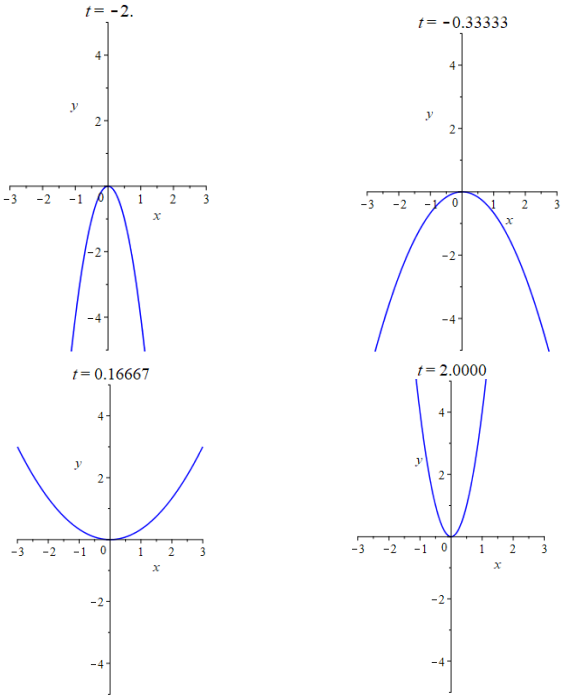


Пример5

with(*plots*)*:*

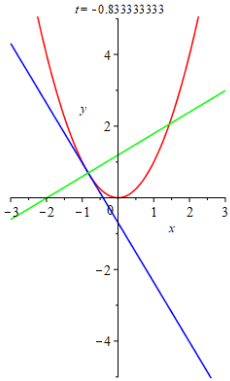
s := *proc*(*f*, *x0*) *local f1; f1* := *D*(*f*)*; plot*([*f1*(*x0*)**f*(*x*)], *x* = −*3* .. *3*, *y* = −*5* .. *5*, *scaling* = *constrained*, *color* = *blue*)*; end proc;*

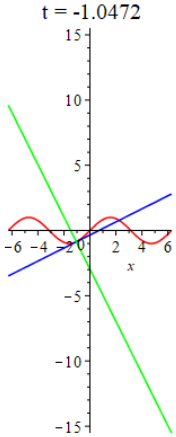
animate(*s*, [*x* −> *x*^*2*, *t*], *t* = −*2* .. *2*)*;*



А теперь пример анимации касательной и нормали через процедуру. Сначала для параболы, потом для синусоиды:

```
tangentline := proc(f, x0) local f1; f1 := D(f); plot([f(x), f1(x0)*(x − x0) + f(x0), − (x − x0)/f1(x0) + f(x0)], x = x0 − 2 .. x0 + 2, scaling = constrained); end proc;
tangentline(x −> x^2, 1);
tangentline(x −> sin(x), 1);
```

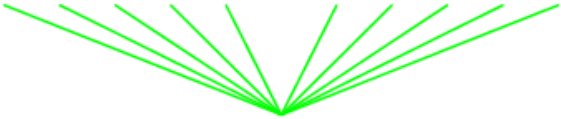




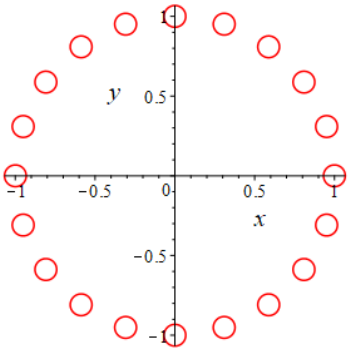
Процедура удобна тем, что задав ее один раз, мы можем получать результат для любой функции, изменяя функцию и начальную точку.

Практическое задание 11

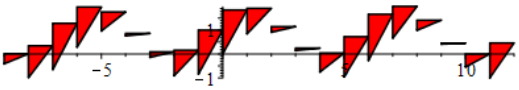
Задание 1. Нарисовать «фонтан»:



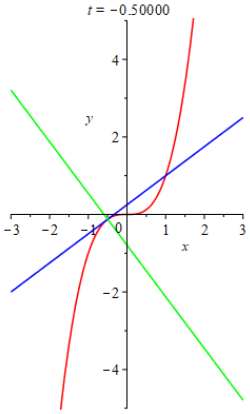
Задание 2. Нарисовать мыльные пузыри:

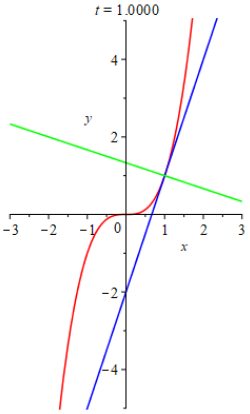


Задание 3. Нарисовать синусоиду из треугольников:



Задание 4. Задать анимацию касательной и нормали для кубической параболы:

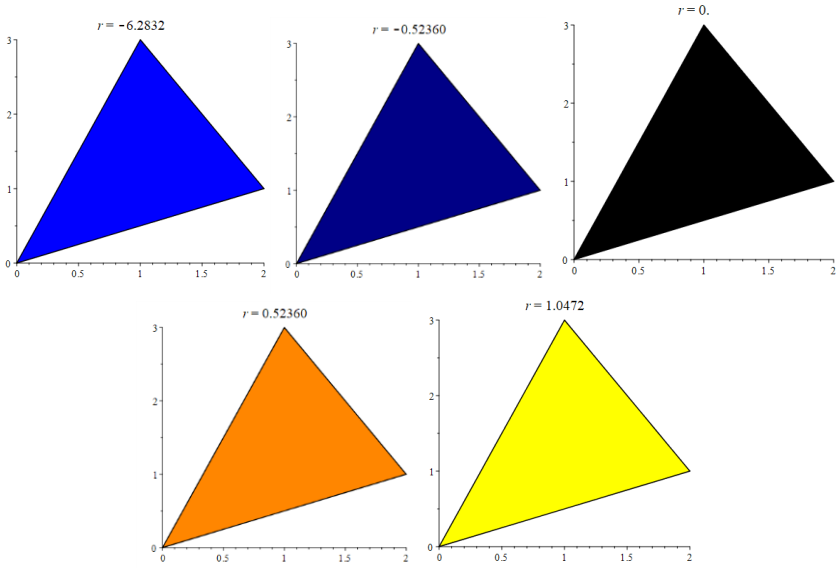




Лекция 12. Анимация цвета

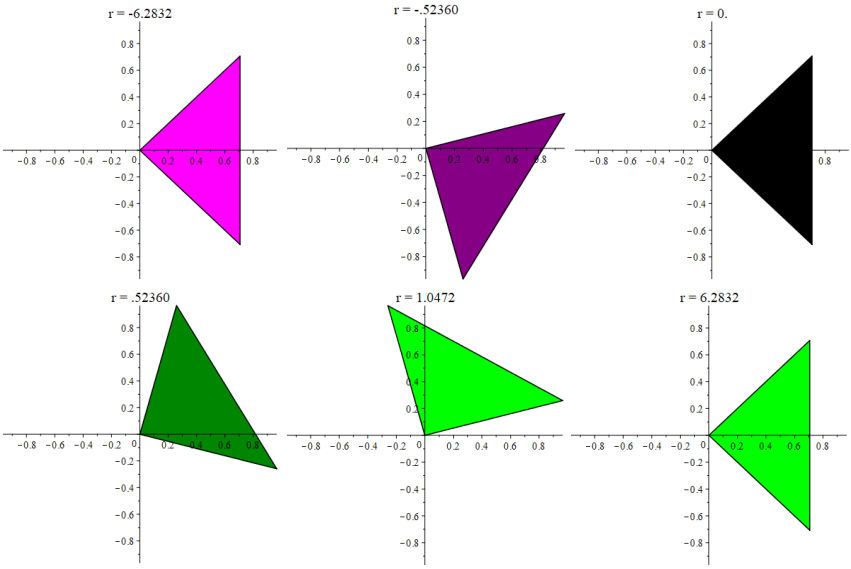
В программе существует своя цифровая система цветов, называемая RGB, где каждый цвет задается тремя числами. Это можно использовать для анимации цвета. Рассмотрим пример:

animate(*polygonplot*, [[[*2*, *1*], [*0*, *0*], [*1*, *3*]],
color = *COLOR*(*RGB*, *3*r*, *r*, −*r*)], *r* = −*2*Pi* .. *2*Pi*);



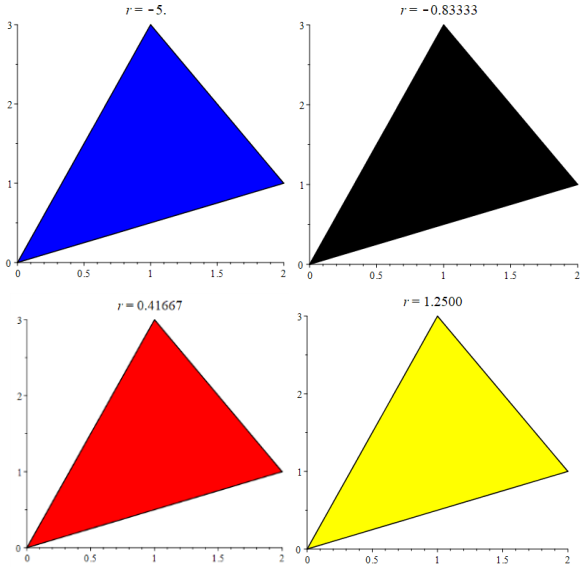
Можем сделать одновременную анимацию движения треугольника и смены его цвета:

animate(*polygonplot*, [[[*cos*(−*Pi/4* + *r*), *sin*(−*Pi/4* + *r*)], [*0*, *0*], [*cos*(*Pi/4* + *r*), *sin*(*Pi/4* + *r*)]], *color* = *COLOR*(*RGB*, −*r*, *r*, −*r*)], *r* = −*2*Pi* .. *2*Pi*);



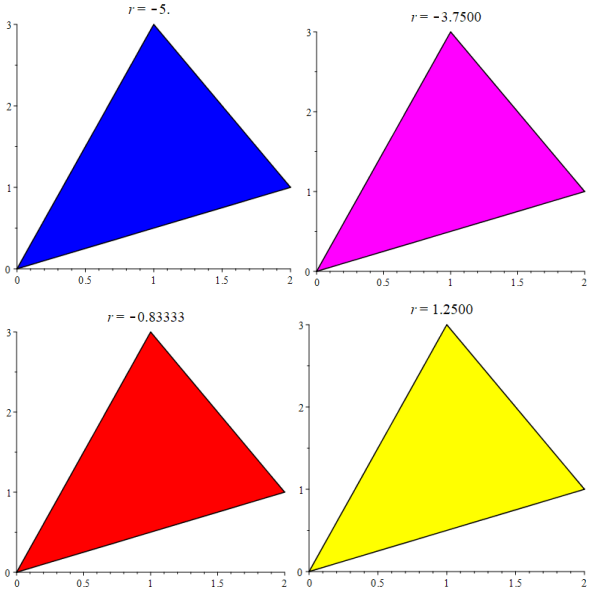
В следующих примерах пробуем разные перемены цвета:

animate(*polygonplot*, [[[*2*, *1*], [*0*, *0*], [*1*, *3*]], *color* = *COLOR*(*RGB*, *trunc*(*3***r*), *trunc*(*r*), *trunc*(−*r*))], *r* = −*5* .. *5*);



Другой вариант:

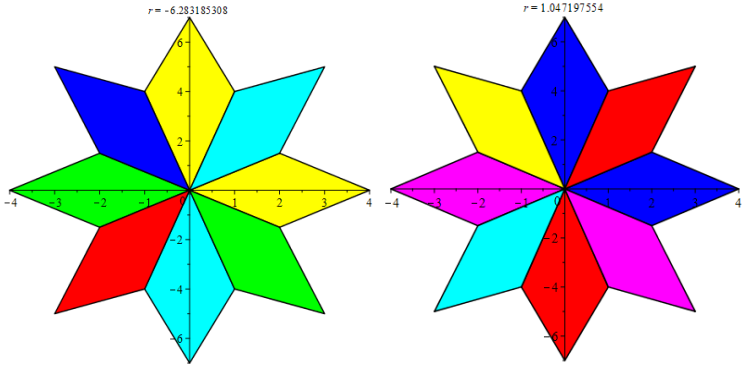
animate(*polygonplot*, [[[*2*, *1*], [*0*, *0*], [*1*, *3*]], *color* = *COLOR* (*RGB*, *trunc*(*5* + *r*), *trunc*(*r*), *trunc*(−*r*))], *r* = −*5* .. *5*);



А теперь зададим разноцветный цветок с переливами цвета:

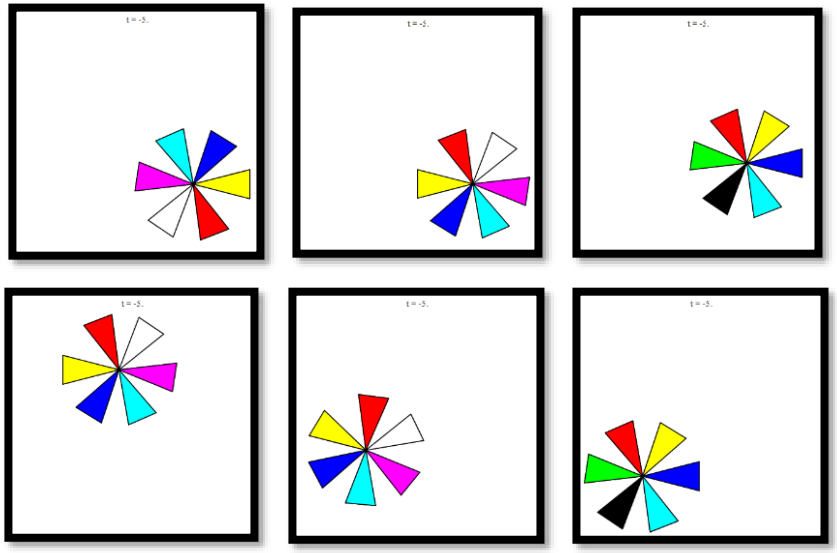
a1:=*animate*(*polygonplot*,[[[*2*,*1.5*],[*0*,*0*],[*1*,*4*],[*3*,*5*]],
color=*COLOR*(*RGB*,*r*, −*r*, −*r*)], *r*= −*2***Pi*..*2***Pi*):
a2:=*animate*(*polygonplot*,[[[−*2*,*1.5*],[*0*,*0*],[−*1*,*4*],[−*3*,*5*]],
color=*COLOR*(*RGB*,*r*,*r*, −*r*)], *r*= −*2***Pi*..*2***Pi*):
a3:=*animate*(*polygonplot*,[[[*2*, −*1.5*],[*0*,*0*],[*1*, −*4*],[*3*, −*5*]],
color=*COLOR*(*RGB*,*r*, −*r*,*r*)], *r*= −*2***Pi*..*2***Pi*):
a4:=*animate*(*polygonplot*,[[[−*2*, −*1.5*],[*0*,*0*],[−*1*, −*4*],[−*3*, −*5*]],
color=*COLOR*(*RGB*, −*r*,*r*,*r*)], *r*= −*2***Pi*..*2***Pi*):
a5:=*animate*(*polygonplot*,[[[−*1*,*4*],[*0*,*0*],[*1*,*4*],[*0*,*7*]],
color=*COLOR*(*RGB*, −*r*, −*r*,*r*)], *r*= −*2***Pi*..*2***Pi*):
a6:=*animate*(*polygonplot*,[[[*2*,*1.5*],[*0*,*0*],[*2*, −*1.5*],[*4*,*0*]],
color=*COLOR*(*RGB*, −*r*, −*r*,*r*)], *r*= −*2***Pi*..*2***Pi*):
a7:=*animate*(*polygonplot*,[[[−*2*,*1.5*],[*0*,*0*],[−*2*, −*1.5*],[−*4*,*0*]],
color=*COLOR*(*RGB*,*r*, −*r*,*r*)], *r*= −*2***Pi*..*2***Pi*):

```
a8:=animate(polygonplot,[[[ −1,−4],[0,0],[1,−4],[0,−7]],
color=COLOR(RGB,r,−r,−r)], r= −2*Pi..2*Pi):
display(a1,a2,a3,a4,a5,a6,a7,a8);
```



Практическое задание 12

Нарисовать цветик-шестицветик, он меняет цвет, вращается относительно центра и перемещается по экрану:



Список литературы

1. Аммерал Л. Принципы программирования в машинной графике: Пер. с англ./ Л. Аммерал. – Москва: Сол-Систем, 1992. – 224 с.
2. Васильев А.Н. Maple 8. Самоучитель./ А.Н. Васильев. – Москва: Издательский дом «Вильямс», 2003. –352 с.
3. Гилев В.Г. Современные пакеты прикладных программ. Основы работы в Maple [Электронный ресурс]: учебное пособие / сост. В. Г. Гилев; Пермский государственный национальный исследовательский университет. – Электронные данные. – Пермь, 2022 – 2,61 Мб; 95 с. – Режим доступа: http://www.psu.ru/files/docs/science/books/uchebnie-posobiya/Gilev-Sovremennye-Pakety-Prikladnyh-Programm-Osnovy-Raboty-V-Maple.pdf.
4. Дьяконов В.П. Maple 11.10.12/13/14 в математических расчетах: самоучитель/ В.П. Дьяконов. – Москва: ДМК-пресс, 2011. – 699–701 с.
5. Егоров А.И. Обыкновенные дифференциальные уравнения и система Maple / А.И. Егоров. – Москва: СОЛОН-ПРЕСС, 2016. – 392 с.
6. Ефремов Ю.С. Методы математической физики в пакете символьной математики Maple: уч. пособие для академического бакалавриата/ Ю.С. Ефремов. – Москва: Юрайт, 2019. – 302 с.
7. Левин В.А. Элементы линейной алгебры и аналитической геометрии на базе пакета "Mathematica": [учеб.пособие для вузов] / В.А. Левин, В.В. Калинин, Е.В. Рыбалка. – Москва: ФИЗМАТЛИТ, 2007. – 191 с.
8. Матрос Д.Ш. Элементы абстрактной и компьютерной алгебры: учеб.пособие для вузов рек. УМО по образованию по спец. пед. образования./ Д.Ш. Матрос, Г.Б. Поднебесова. – Москва: Академия, 2004. – 237 с.
9. Матросов А.В. Maple 6. Решение задач высшей математики и механики./ А.В. Матросов. – Санкт-Петербург: BHV-Санкт-Петербург, 2001. – 528 с.
10. Миронов Д. CorelDraw 11. Учебный курс. / Д. Миронов. – Санкт-Петербург: Питер, 2003. – 448 с.
11. Могилев А.В. Информатика: учеб.пособие для вузов рек. МО РФ; под ред. А. В. Могилева./ А.В. Могилев, Е.К. Хеннер, Н.И. Пак. – 2-е изд., стер. – Москва: Академия, 2008. – 325 с.

12.Петров М.Н. Компьютерная графика. Учебник для ВУЗов./ М.Н. Петров, В.П. Молочков. – Санкт-Петербург: Питер, 2003. – 736 с.

13.Порев В.Н. Компьютерная графика. / В.Н. Порев. – Санкт-Петербург: БХВ – Петербург, 2002. – 432 с.

14.Примеры выполнения лабораторных работ по алгоритмам компьютерной графики: Метод. указания. (Сост.: Хайдаров Г.Г., Алексеев С.Ю.) – Санкт-Петербург: СПбГТИ(ТУ), 2005. – 30 с.

15.Савотченко С.Е. Методы решения математических задач в Maple: учебное пособие/ С.Е. Савотченко, Т.Г. Кузьмичева. – Белгород: Белаудит, 2001. – 155 с.

16.Тимофеев Г.С. Графический дизайн. Учебный курс/ Г.С. Тимофеев, Е.В. Тимофеева. – Ростов-на-Дону: Феникс, 2002. – 320 с.

17.Шикин А.В. Компьютерная графика. Динамика, реалистические изображения./ А.В. Шикин, А.В. Боресков. – Москва: ДИАЛОГ-МИФИ, 1996. – 288 с.

18.Шикин А.В. Компьютерная графика. Полигональные модели./ А.В. Шикин, А.В. Боресков. – Москва: ДИАЛОГ-МИФИ, 2001. – 464 с.

Приложения

Приложение 1

Темы проектов

1. Отображение баз данных на карту местности. Взаимодействие графического объекта и его описания.
2. Система отображения статистических данных.
3. Фракталы (визуальная математика).
4. Начертательная и аналитическая геометрия (конструктор).
5. Интерпретатор синтаксического описания динамической картинки.
6. Формирование среды (туман, пламя, снег, салют, облака, видеоэффекты, дождь, вода, смывка и так далее) и взаимодействие ее с битовой картой.
7. Лаборатория мультипликации (взаимодействие карт, управление лентой).
8. Создатель образов (стиля) мультипликации.
9. Управление элементами поверхности (человеческое тело, лицо).
10. Построения в неэвклидовых геометриях.
11. Имитация нетрадиционных графических курсоров (например, грифель, пушок, мазок, размыв и так далее).
12. Эволюция вида растений, животных.
13. Синтез элементов ландшафта.
14. Выделение контура образа на динамической сцене и слежение за ним.
15. Обработка растровых картинок.
16. Построение объектов в проекции (прямая, обратная, стерео, рыбий глаз, цилиндрическая).
17. Синтезатор двухмерных композиций.
18. Преобразователь классических картин.
19. Карикатура.
20. Создание компьютерного ролика.

Приложение 2

Примерные варианты контрольных работ

Контрольная работа № 1
(Первый рубежный контроль)

1. Описать основные элементы интерфейса используемого пакета.

2. Каким образом можно задать цвета: желтый, оранжевый, золотой, хаки?

3. Запишите часть стихотворения 14 шрифтом, Times New Roman, синим цветом: Если мальчик любит мыло и зубной порошок, Этот мальчик очень милый поступает хорошо.

4. Нарисуйте на одной картине разным цветом графики функций: заданной явно; заданной параметрически; неявно заданной; кусочно-заданной.

5. Нарисуйте функцию y=sin(x+231) в бесконечности.

6. Задайте синий круг, зеленый прямоугольник, серую ломанную с пятью звеньями.

Контрольная работа № 2
(Второй рубежный контроль)

1. Создайте анимацию на одной картине графиков функций: заданной явно с движением вдоль оси OX; заданной параметрически вдоль оси OY; неявно заданной по диагонали относительно начала координат; кусочно-заданной по диагонали.

2. Создайте анимацию функции y=sin(x) в виде рисунка карандашом.

3. Задайте анимацию фигур произвольным образом: синий круг, зеленый прямоугольник, серая ломанная с пятью звеньями.

Примерный перечень заданий к зачету:

Индивидуальное задание: нарисовать анимационную картину на плоскости на любую тему с заданными требованиями (не менее 2-х анимаций по встроенным опциям, не менее одной анимации функции).

Содержание

ОПИСАНИЕ ФУНКЦИОНАЛЬНОСТИ ИЗДАНИЯ:

Интерфейс электронного издания (в формате pdf) можно условно разделить на 2 части.

Левая навигационная часть (закладки) включает в себя содержание книги с возможностью перехода к тексту соответствующей главы по левому щелчку компьютерной мыши.

Центральная часть отображает содержание текущего раздела. В тексте могут использоваться ссылки, позволяющие более подробно раскрыть содержание некоторых понятий.

МИНИМАЛЬНЫЕ СИСТЕМНЫЕ ТРЕБОВАНИЯ:

Celeron 1600 Mhz; 128 Мб RAM; Windows XP/7/8 и выше; разрешение экрана 1024×768 или выше; программа для просмотра pdf.

СВЕДЕНИЯ О ЛИЦАХ, ОСУЩЕСТВЛЯВШИХ ТЕХНИЧЕСКУЮ ОБРАБОТКУ И ПОДГОТОВКУ МАТЕРИАЛОВ:

Оформление электронного издания : Издательский центр «Удмуртский университет».

Компьютерная верстка: Т.В. Опарина.

Авторская редакция.

Подписано к использованию 30.12.2025
Объем электронного издания 8,2 Мб
Издательский центр «Удмуртский университет»
426034, г. Ижевск, ул. Ломоносова, д. 4Б, каб. 021
Тел. : +7(3412)916-364 E-mail: editorial@udsu.ru