Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Математика. Математический анализ

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовой работе
на тему

ВИЗУАЛИЗАЦИЯ В MAPLE. АНИМАЦИОННЫЕ ВОЗМОЖНОСТИ MAPLE ДЛЯ ВИЗУАЛИЗАЦИИ РЕШЕНИЙ

БГУИР КР 1-40 04 01

Студент: гр. 053506 Слуцкий Н. С.
Руководитель: канд. ф.-м. н., доцент Рыкова О.В.
канд. ф.-м. н., доцент Калугина М.А

Минск 2021

СОДЕРЖАНИЕ

Введение

Когда “одинокие” наборы чисел и (или) формул не раскрывают в достаточно понятной степени найденное решение поставленной математической и не только проблемы, на вооружение к нам, как к исследователям, приходят различные возможности, предоставляемые современными системами компьютерной алгебры и другими пакетами прикладных программ по графическому представлению решений в форме визуальных объектов.
В целом визуализация каких-либо данных представляет собой наглядное графическое представление массивов различной информации. Если смотреть на поверхности, то классическое построение графиков функций на уроках математики делается как раз с целью отображения (визуализации) той или иной функции. Для каких целей ? Чтобы показать, что просмотреть поведение функции (экстремумы, монотонность, знакопостоянство, выпуклость и др.) можно также не аналитически, а фактически на рисунке. Это обычно воспринимается более легко, понятно и в целом является наиболее топорным методом по-быстрому исследовать поведения функций, когда из многих нужно выбрать наиболее подходящие.
Наглядность поставленной задачи является важной частью не только для понимания процесса решения, но и для исследования его в динамике, которая, разумеется, легче воспринимается зрительно. Приятным и полезным бонусом является анимация этапов решения, которая даёт экспрессивное представление, например, о скорости явления. Нам, как студентам, как исследователям, визуализация и анимация часто позволяет находить аналогии и закономерности, систематизировать найденные решения и моделировать определённый кейс при проведении какой-то исследовательской работы. Анимация графиков может найти широкое применение при создании учебных материалов. С ее помощью можно акцентировать внимание на отдельных параметрах графиков и функций, которые их образуют, и наглядно иллюстрировать характер их поведения.

Введение в пакет “plots” для обычной и расширенной визуализации

Базовое построение классических графиков обыкновенных функций ограничивается применением и настройкой функции “plot” из глобальной области видимости в Maple. Подробнее эта функция будет рассматриваться в одном из разделов ниже. Однако в большинстве случаев возникает необходимость построить нечто более изощрённое. Не только построить, но проанимировать и так далее. На помощь приходит встроенный пакет (библиотека) функций “plots” [1]. Пакет “plots” содержит почти полсотни графических функций, существенно расширяющих возможности построения двумерных и трёхмерных графиков в Maple. Среди этих функций надо отметить прежде всего средства построения ряда графиков новых типов (например, в виде линий равного уровня, векторных полей и т. д.), а также инструменты объединения различных графиков в один.
В целом этот пакет вполне заслуживает описания в отдельной методичке. Но, учитывая ограниченный объем данной курсовой работы, автор рассмотрит лишь основные и нужные тут примеры его применения. Автор акцентирует внимание на то, что для использования приведённых функций необходим вызов пакета, например, командой “with(plots)” или доступом по ключу вида “plots[implicitplot]()”. Способы обращения к функции “display” на рисунках 1 и 2 равнозначны.

```
> with(plots):
> display(y1, y2);
```

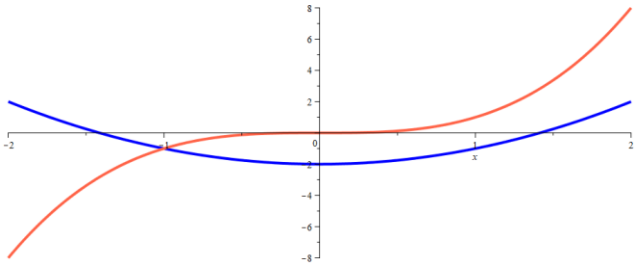


Рисунок 1. Обращение с директивой использования всего пакета

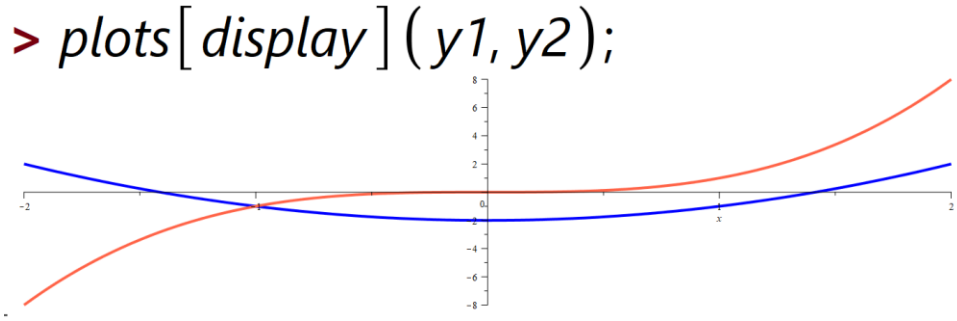


Рисунок 2. Получение одной функции из пакета по её имени

Немного о статической визуализации в системе компьютерной алгебры Maple

Maple предоставляет достаточно обширный набор инструментов визуализации. Можно создавать двухмерные и трёхмерные графики и анимации в интерактивном режиме с помощью помощника по построению графиков и контекстных меню. Maple также включает в себя большую коллекцию команд и инструментов программирования для создания и настройки сюжета. Эти команды можно использовать в интерактивном режиме или включать в программы и сценарии Maple для создания пользовательских специализированных графиков и расширенных приложений.
Базовый инструментал по созданию графических изображений с графиками функций представлен командой “plot” и некоторыми вспомогательными методами из пакета “plots”, описанного в прошлом пункте.

Отображение двумерных объектов

Для построения графиков функций и простейших кривых и поверхностей нет необходимости подгружать пакет “plots”. Достаточно использовать функцию “plot”, входящую в ядро Maple.
На рисунке 3 представлен пример самой базовой визуализации привычной всем тригонометрической функции на отрезке [-10 ; 10].

```
plot(sin(x), x=-10..10, thickness=5, color=orange)
```

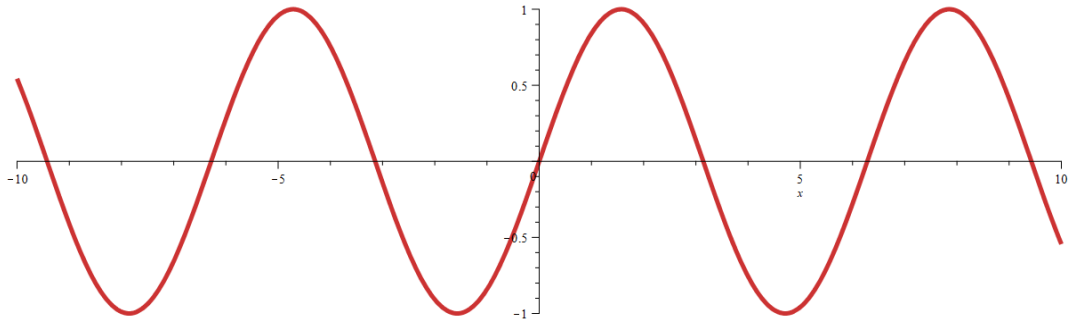


Рисунок 3. Построение графика простейшей тригонометрической функции на отрезке

Со структурой команды plot можно ознакомиться на сайте производителя[2]. Разумеется, в системе компьютерной алгебры Maple мы не ограничены в построении одиночных графиков элементарных непрерывных функций. В качестве элементарного примера можно построить кусочно-заданную функцию (рисунок 4).

```
# с помощью команды piecewise можно работать как с обычными, так и с кусочными функциями
func := piecewise(x < -Pi, 4 ·cos(2·x), x ≥ Pi, 6 ·exp(1)^(-0.4·x)) :
funcChart := plot(func, discont = true, color = blue, thickness = 3);
```

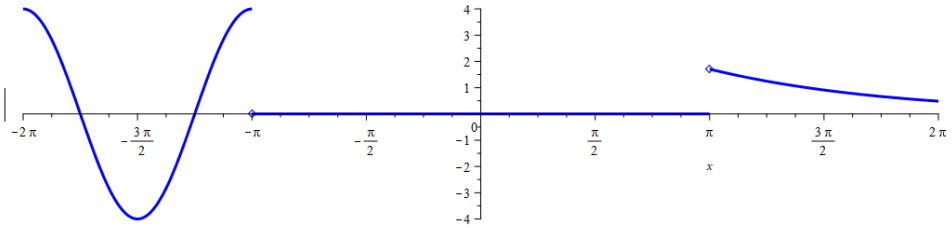


Рисунок 4. Построение графика кусочной функции

Всегда можно получить, например, графики производных и первообразных. В одной системе координат можно построить несколько графиков. Это может быть полезно, когда необходимо проследить поведение разных функций на наблюдаемых интервалах (рисунок 5).

```
> func := piecewise(x < -Pi, 4 ·cos(2·x), x ≥ Pi, 6 ·exp(1)^(-0.4·x)) :
> plot([func, diff(func, x), int(func, x) + 3], discont = true, thickness = 5);
```

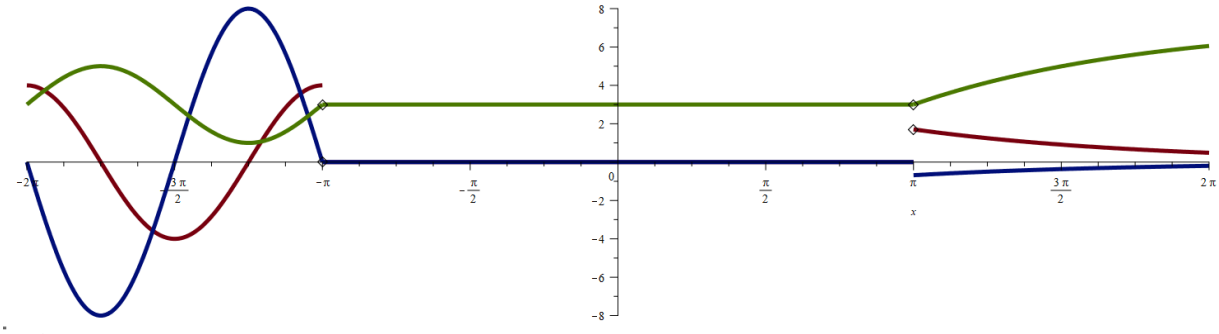


Рисунок 5. Построение нескольких графиков в одной системе координат

Выше можно было наблюдать и фиксировать самую основную базовую функцию для построения графиков и их кастомизации.

Используя дополнительные функции подключаемого пакета “plots”, можно расширять границы. Появляется возможность перехода в полярные координаты и многого другого, что часто бывает полезно (рисунок 6).

```
with(plots) :
polarplot(2 + 2 cos(4·θ − Pi/3), θ = 0..2 π, thickness = 5, color = green);
```

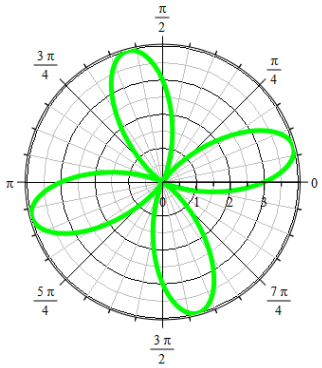


Рисунок 6. Пример построения графика функции в полярных координатах

В примерах, предоставленных автором выше, можно наблюдать, что даже статическая визуализация всего лишь двумерных сущностей является достаточно гибко настраиваемой и любые данные можно представить в наглядном, читаемом и приятном исследователю виде. Настройка цветов,

толщины линий, способа отображения графика (точечно или сплошной линией) — это лишь несколько пунктов из возможностей кастомной визуализации функций.

Отображение трёхмерных объектов

Перейдём в пространство R^3. Имея функцию вида z = f (x, y) или неявно заданную функцию F (x, y, z) = 0, мы можем, построив график этой функции, получить трёхмерную сущность. Если для функции одной переменной (или двух, если задана неявно – F (x, y) = 0) y = f (x) графиком в общем случае являлась какая-либо кривая в плоскости XoY пространства R^2, то для функции двух переменных это в общем случае будет какая-то поверхность в пространстве R^3. Система компьютерной алгебры позволяет строить эти поверхности для вышеописанных функций. Пример построения находится на рисунке 7.

```
plot3d(x^2 - y^2, x=-1..1, y=-1..1);
```

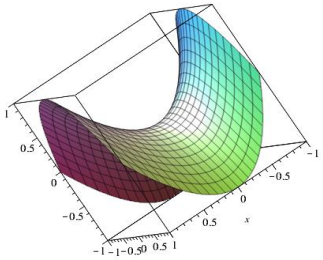


Рисунок 7. Использование команды plot3d

Автор хочет обратить внимание на то, что в данном случае система компьютерной алгебры Maple не просто выдаёт статическое изображение (или, иначе говоря, отпечаток), содержащее проекцию поверхности на плоскость экрана монитора с произвольного “местоположения”, но выдаёт интерактивный фрейм, где пользователи имеют возможность “крутить” пространство и наблюдать поверхность с разных сторон, что, безусловно, удобно для более детального рассмотрения поведения функции в каких-нибудь интересных для исследования областях. Путём комбинирования

графиков, гибкой настройки цветов можно получать подобные относительно необычные формы (рисунок 8).

```
c1 := [cos(x)-2 cos(0.4 y), sin(x)-2 sin(0.4 y), y]:
c2 := [cos(x) + 2 cos(0.4 y), sin(x) + 2 sin(0.4 y), y]:
c3 := [cos(x) + 2 sin(0.4 y), sin(x)-2 cos(0.4 y), y]:
c4 := [cos(x)-2 sin(0.4 y), sin(x) + 2 cos(0.4 y), y] :
plot3d({c1, c2, c3, c4}, x = 0..2 π, y = 0..10, grid = [25, 15], color = sin(x));
```

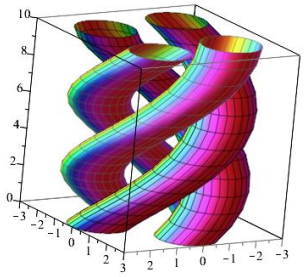


Рисунок 8. Пример построения относительно сложной комбинации поверхностей с элементами их кастомизации

Панель инструментов для настройки отображения

Необходимо также добавить ко всему вышесказанному, что пользовательская настройка отображения графиков не ограничивается опциональным массивом “options” в аргументах функции “plot” или её подобной. Уже после компиляции и рендера объекта с графикой у пользователя есть возможность кастомизировать некоторые составляющие отображения. Это такие параметры, как:

- масштаб,
- стиль линий,
- сетка,
- угол поворота к плоскости экрана,
- другое.

Панель для настройки вышеописанных параметров представлена на рисунке 9.

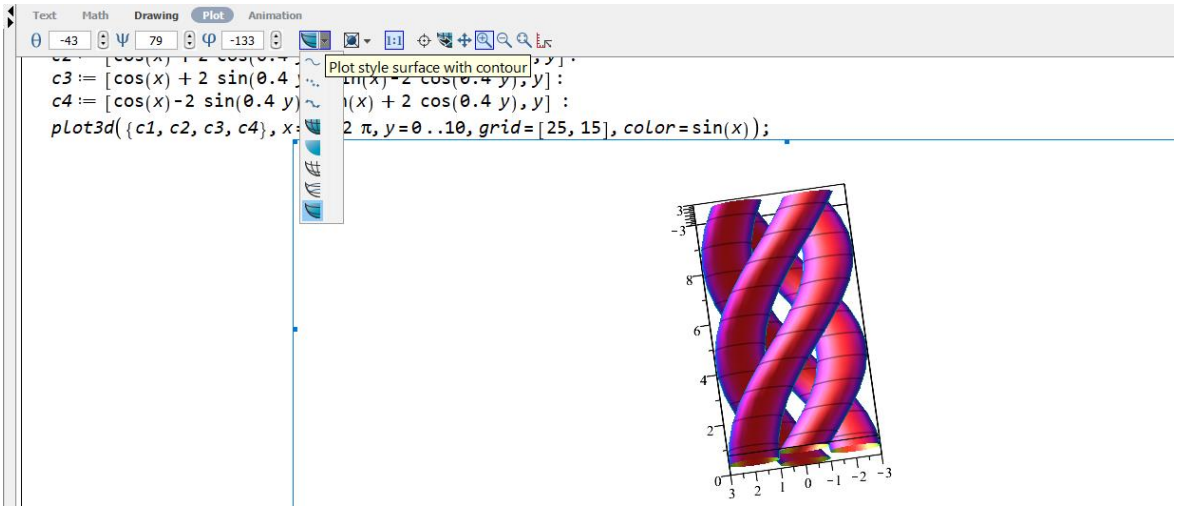


Рисунок 9. Дополнительные возможности настройки отображения (см. верхнюю часть картинки)

Анимация в системе компьютерной алгебры Maple

Визуализация графических построений и результатов моделирования каких-либо математических явлений и объектов значимо повышается при использовании “оживляющих средств” — анимации изображений. Анимация — это визуальное изображение изменений свойств одного или нескольких объектов. Пакет “plots” имеет три простые функции для создания анимированных графиков.

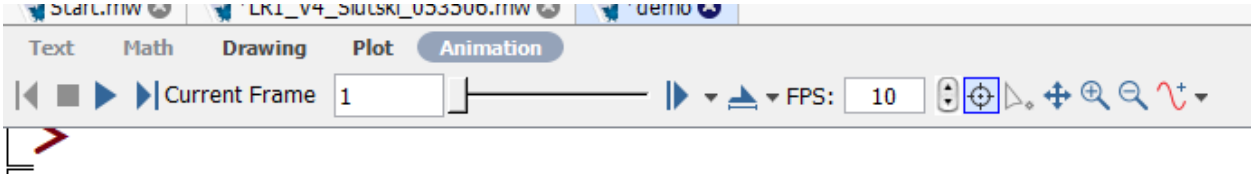
Панель инструментов анимации

Но для начала рассмотрим в целом компонент пользовательского интерфейса системы компьютерной алгебры Maple, который предназначен для работы с анимацией.

При создании анимации неотъемлемым помощником является панель инструментов (рисунок 10). Она позволяет:

- задать количество кадров в секунду;
- задать режим повтора проигрывания анимации;
- выбрать участок для более детального наблюдения;
- поставить анимацию на паузу, продолжить или начать сначала;

- покадрово просмотреть анимацию;
- пронаблюдать значения функции в точках путём наведения курсора мыши на них;
- и другое.



- *Рисунок 10. Панель инструментов для работы с анимацией в Maple 16*

Обзор базовых команд для анимации из пакета "plots"

Визуализация графических построений и результатов моделирования различных объектов и явлений повышается при использовании средств «оживления» (анимации) изображений. Пакет “plots” имеет две простые функции для создания анимированных графиков.

Первая из этих функций служит для создания анимации графиков, представляющих функцию одной переменной F(x): “animatecurve".

Эта функция просто позволяет наблюдать медленное построение графика. Формат ее применения подобен используемому в функции plot. При вызове данной функции вначале строится пустой шаблон графика. Если активизировать шаблон мышью, то в строке главного меню появляется меню Animation. Меню Animation содержит команды управления анимацией. Такое же подменю появляется и в контекстном. Указанное подменю содержит те же команды, что и вышеописанная панель инструментов анимации.

При исполнении команды “Play” происходит построение кривой (или нескольких кривых). В зависимости от выбора
команд “Faster” или “Slower” построение идет быстро или медленно. Команда “Next” выполняет один шаг анимации -построение очередного фрагмента кривой. Переключатель “Backward/Forward” позволяет задать направление построения кривой - от начала к концу или от конца к началу. Построение может быть непрерывным или циклическим в зависимости от

состояния позиции “Continiuus/Singlecycle” в подменю управления анимацией.

При циклической анимации число циклов задается параметром “frames”. Пример постепенной отрисовки заданной линии представлен на рисунках 11, 12.

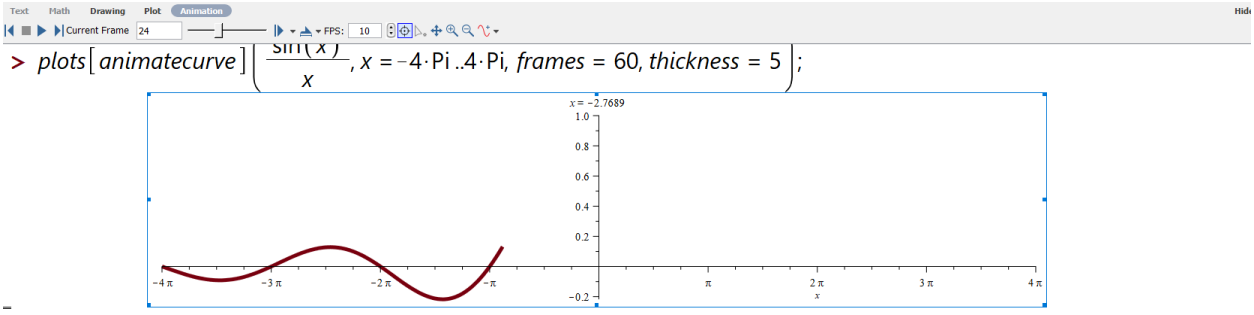


Рисунок 11

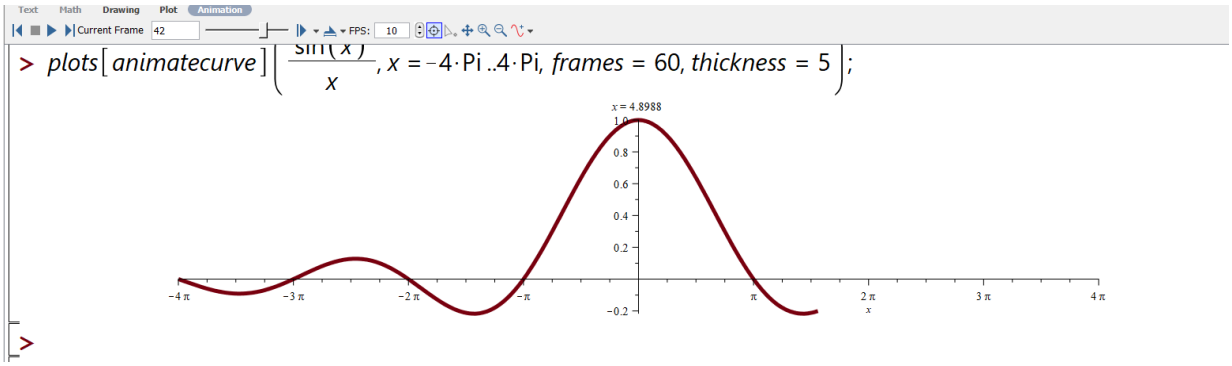


Рисунок 12

Более обширные возможности анимации двумерных графиков обеспечивает функция “animate”: animate(F, x, t).

В ней параметр x задает пределы изменения переменной x, а параметр t — пределы изменения дополнительной переменной t. Суть анимации при использовании данной функции заключается в построении серии кадров (как в мультфильме), причем каждый кадр связан со значением изменяемой во времени переменной t.

Структура команды “animate” следующая: animate (plotcommand, plotargs, t = a..b, options), где параметры представляют собой:

1. функцию для построения графика (plot, pointplot, plot3d…)
2. аргументы для функции для построения графика
3. t – имя и диапазон параметра, который изменяется в функции построения графика

Пример команды и нескольких кадров из построения анимации из тестовой функции представлены на рисунках 13 – 15.

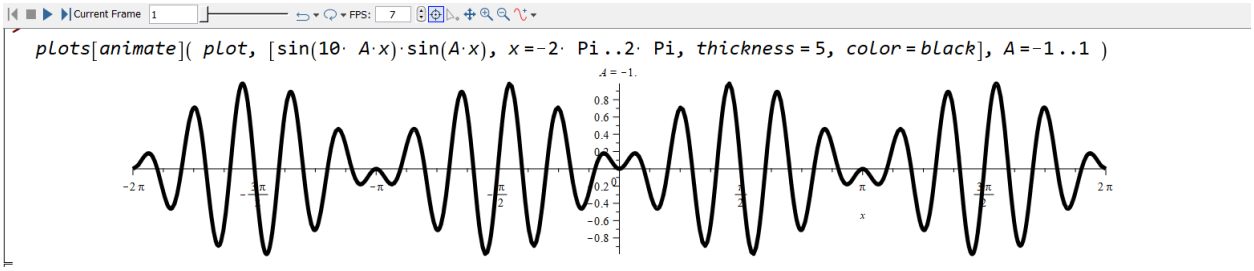


Рисунок 13

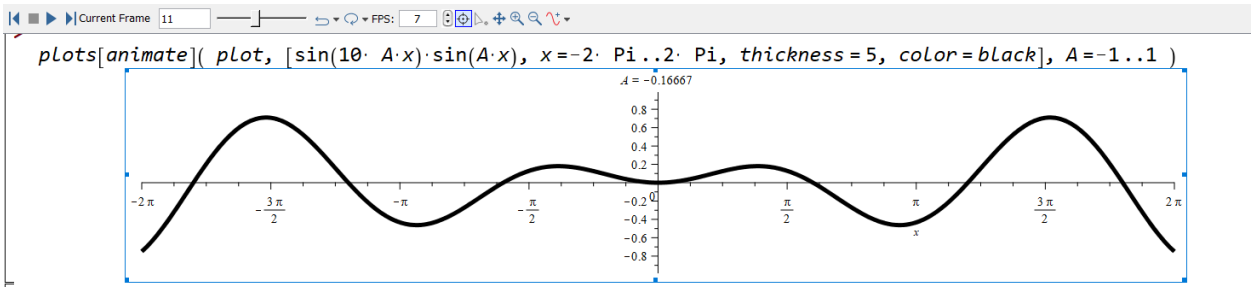


Рисунок 14

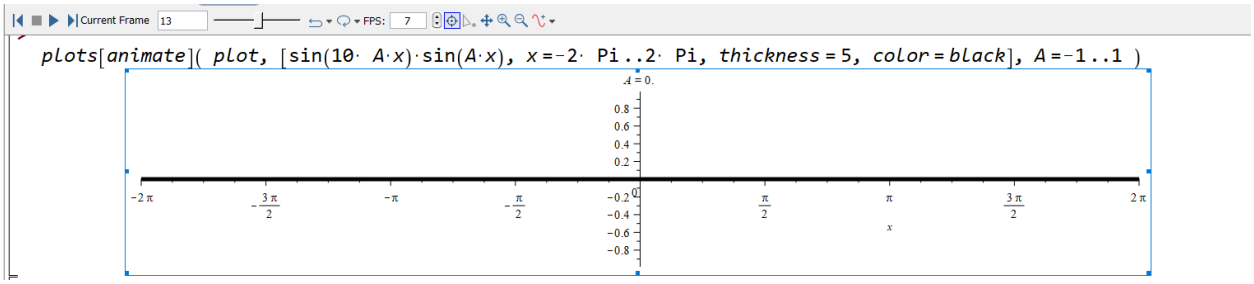


Рисунок 15

Таким же образом осуществляется и оживление трёхмерных фигур. Для этого используется функция “animate3d”. На рисунке 16 показано построение анимированного графика.

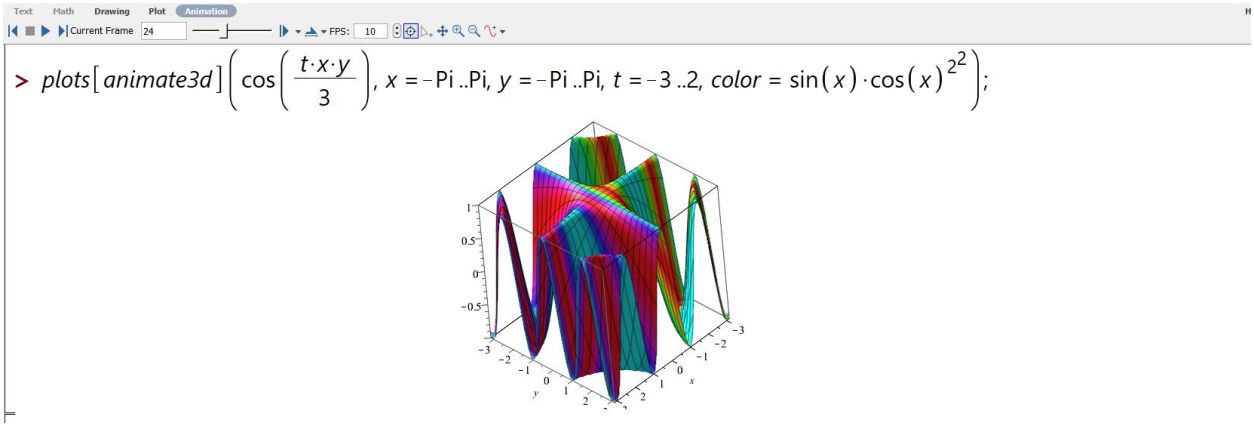


Рисунок 16. Подготовка анимационного фрейма с трёхмерным графиком

Еще один способ получения анимационных фреймов — это создание ряда графических объектов a_1, a_2, a_3,…, a_n, и их последовательный вывод с помощью функций “display” или “display3d”:
“display (a1, a2, a3, insequence = true)” .Здесь ключевым моментом является задействование параметра “insequence” . Именно он обеспечивает вывод одного за другим ряда графических объектов a1, a2, a3 и так далее. При этом объекты просто появляются по одному и каждый предшествующий объект стирается перед появлением следующего. На рисунках 17, 18 представлен пример смены цвета графика одной и той же функции, чтобы создать мигание. Использована функция display с только что описанным аргументом.

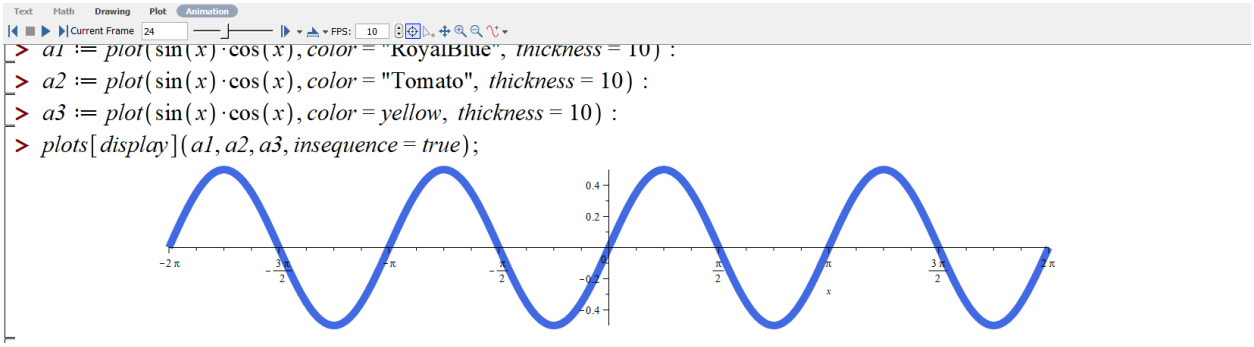


Рисунок 17

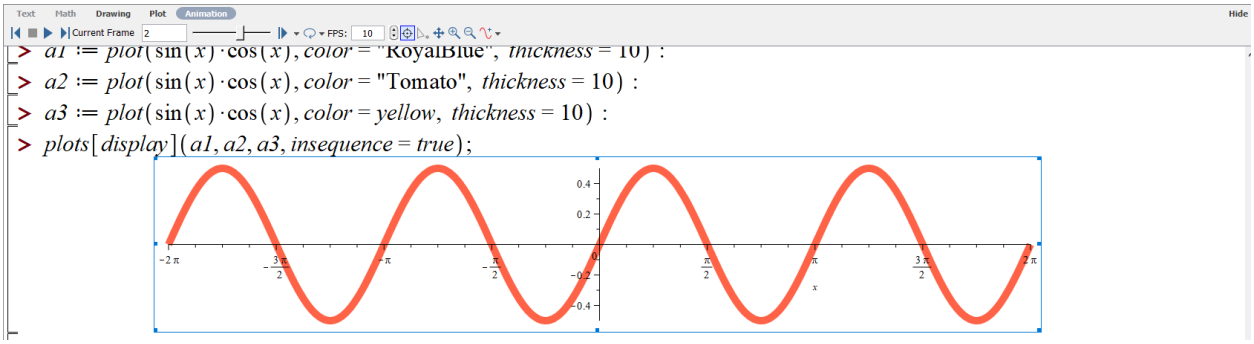


Рисунок 18

Введение в пакет “plottools” для создания графических примитивов

Инструментальный пакет графики “plottools” служит для создания графических примитивов, строящих элементарные геометрические объекты на плоскости и в пространстве: отрезки прямых и дуг, окружности, конусы, кубики и так далее. Его применение позволяет разнообразить графические построения и строить много графиков специального назначения.
В пакет входит некоторое количество геометрических примитивов, таблица которых представлена на рисунке 19.

arc	**arrow**	**circle**	**cone**	**cuboid**
curve	**cutIn**	**cutout**	**cylinder**	**disk**
dodecahedron	**ellipse**	**ellipticArc**	**hemisphere**	**hexahedron**
hyperbola	**icosahedron**	**line**	**octahedron**	**pieslice**
point tetrahedron	**polygontorus**	**rectangle**	**semitorus**	**sphere**

Рисунок 19. Графические примитивы пакета plottools

Большинство примитивов пакета “plottools” имеет достаточно простой синтаксис. Например, Конус строится примитивом “cone(c, r, h…)”, где c – список с координатами центра, r – радиус основания, h - высота. На месте многоточия могут стоять обычные параметры, задающие цвет дуги, толщину ее линии и так далее.

Все формы записи графических примитивов и их синтаксис можно найти в справочной системе на официальном сайте MapleSoft[3]. В необходимых случаях стоит проверить синтаксис того или иного примитива с помощью справки по пакету “plottools”. Нарисованные примитивы можно отобразить командой “display”. Пример на рисунке 20.

```
> arc1 := plottools[arc]([0, 0], 0.9, Pi/2..Pi, thickness = 5, color = "RoyalBlue"):
> line1 := plottools[line]([-1, 2], [1, 1], thickness = 5, color = black):
> circle1 := plottools[circle]([0, 1], 0.7, thickness = 5, color = "Tomato"):
> plots[display](arc1, line1, circle1);
```

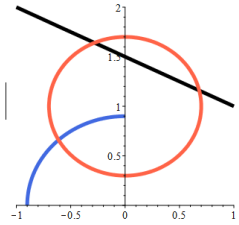


Рисунок 20

Пакет также предоставляет некоторые функции манипулирования над примитивами. Это такие операции с фигурами, как “rotate”, “transform”, “scale”. Подобные операции также можно производить, например, в каскадных таблицах стилей CSS, что широко используется при вёрстке интерфейсов приложений в программировании. На рисунке 21 (часть скриншота с официального сайта Maple) представлен полный перечень подобных команд.

• The commands to alter or examine plot structures are:

extrude	getdata	homothety	project
reflect	rotate	scale	stellate
transform	translate	triangulate	

Рисунок 21

Простой пример использования функции “rotate” можно видеть на рисунке 22. Тут уже сразу применена анимация касательно поворота — и можно наблюдать статическую синусоиду (повёрнутую на 90 градусов) и вращающуюся. Представлен один из кадров.

```
with(plottools):
with(plots):
p := rotate(plot(sin(x), x, x = 0..2π, color = "RoyalBlue", thickness = 5), Pi/2):
r := animate(rotate, [p, Pi·i/3], i = 0..10):
display(p, r, scaling = constrained)
```

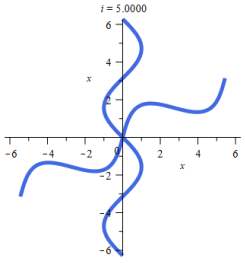


Рисунок 22

Так же, как и двумерные примитивы и операции с ними, в пакете присутствуют и трёхмерные примитивы. Они были упомянуты в таблице в рисунке 21. На рисунке 23 можно наблюдать команду для построения трёхмерной звёздчатой фигуры.

```
plots[display](plottools[stellate](dodecahedron( )), scaling = constrained)
```

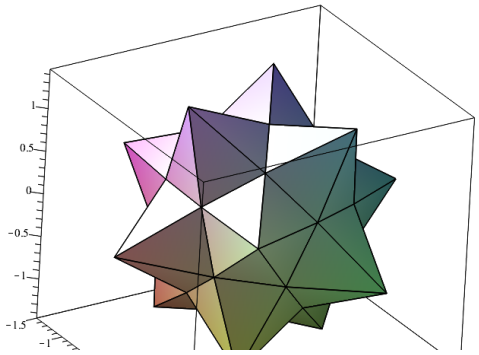


Рисунок 23

Практические примеры

Итак, за предыдущие несколько глав автор рассмотрел по сути весь необходимый функционал, на котором базируется визуализация чего-либо и последующая анимация этих данных. В этой главе будут рассмотрены не просто учебные примеры с элементарным вращением какой-нибудь простой кривой, а более интересные, более практические и визуально привлекательные вещи.

“Пакман”

В игровой индустрии существует такая небезызвестная игра, как “Пакман”. Что если у нас возникла необходимость запрограммировать поведение самого пакмана ? Необходимо математически описать поведение “рта” этого “существа”, описать его движение и поведение при “поедании” шариков. На рисунке 24 можно наблюдать один из кадров и не очень сложную функцию, выдающую такую картинку.

```
> body := proc(n)
    plots[display](plottools[pieslice]([5, 10], 1, 1/6*Pi - 1/12*n*Pi .. 11/6*Pi + 1/12*n*Pi, color = "DarkViolet"))
  end proc:
  plots[animate](body, [n], n = [0, 1, 2, 1], title = "", axes = none);
```

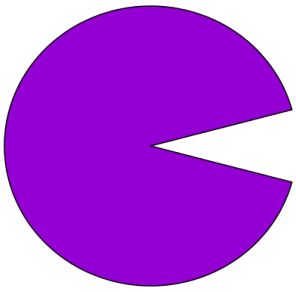


Рисунок 24. Анимация пакмана из известной игры

Автор хотел бы пояснить, что функция “piceslice” строит сектор круга. Также на рисунке показан немного необычный способ передачи параметра в функцию “animate”. Важно знать и помнить, что функция “animate” должна принять в себя функцию, которая вернёт что-то, что можно отобразить (что возвращают команды “display”, “plot”, “implicitplot” и так далее). Можно создать пользовательскую процедуру, которая возвращает отрисованный “plot”, а внутри, например, как-то дополнительно занимается какими-либо преобразованиями параметра.

“Пакман”. Усложнение

После создания “Пакмана” можно создать и “еду”. Теперь функция “frame” возвращает коллекцию графиков. Рисунок 25.

```
> frame := proc(n)
    plots[display](
      plots[pointplot]([[10 - (mod(n, 10)), 0], [15 - (mod(n, 10)), 0], [20 - (mod(n, 10)), 0]], color = "Tomato", symbol
     = solidcircle, symbolsize = 60),
      plottools[pieslice]([0, 0], 5, (1/6) * Pi - (1/12) * n * Pi .. (11/6) * Pi + (1/12) * n * Pi, color = "DarkViolet"), scaling
     = constrained
    )
  end proc:
  plots[animate](frame, [n], n = [0, 1, 2, 1], axes = none);
```

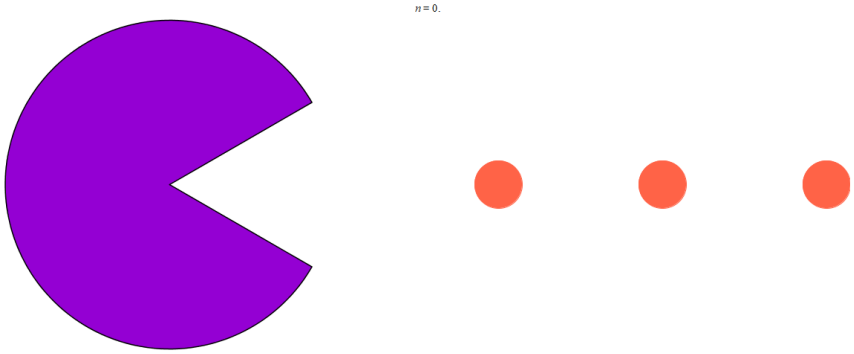


Рисунок 25

Несколько блиц-примеров анимации

```
> N := 144 : # The number of frames

  A := seq(plot([[sin(t) ^ 3, 13 * cos(t) * (1/15) - (1/3) * cos(2*t) - 2 * cos(3*t) * (1/15) - (1/15) * cos(4*t), t = 0 .. Pi * i/N], [
     -sin(t) ^ 3, 13 * cos(t) * (1/15) - (1/3) * cos(2*t) - 2 * cos(3*t) * (1/15) - (1/15) * cos(4*t), t = 0 .. Pi * i/N]], color = red,
     thickness = 5, i = 1 .. N)) :

  plots[display](A, insequence = true);
Error, (in plot) expecting a real constant as range endpoint but received (1/144)*Pi*i
```

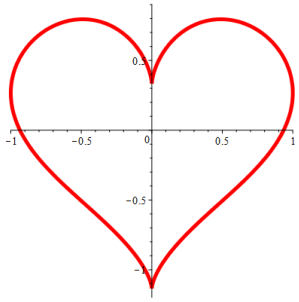


Рисунок 26. Сердце языком математики

```
> internal := plottools[arc]([0, 0], 1, 0 .. -3·Pi/2, color = "RoyalBlue", thickness = 5) :
> middle := plottools[circle]([0, 0], 2, thickness = 5, color = yellow, background = yellow, filled = true) :
> middle_animation := animate(plottools[scale], [plots[display](middle), j/100, j/100], j = 0 ..100) :
> outernal_animation := animate(rotate, [plots[display](outernal), Pi/12·i], i = 0 ..30) :
> internal_animation := animate(plottools[rotate], [plots[display](internal), -Pi/15·k], k = 0 ..24) :
> plots[display](middle_animation, outernal_animation, internal_animation);
```

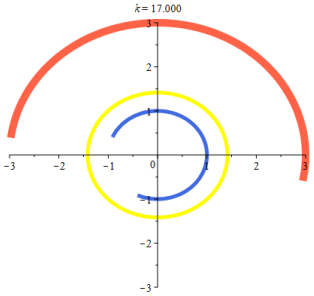


Рисунок 27. Анимация загрузки контента (например, в каком-нибудь веб-приложении)

Движение материальной точки

Пусть материальной точке из координаты (a; b) придали начальную горизонтальную скорость и отправили в “полёт”. Если не учитывать сопротивление воздуха, равняющееся kv^2 (где k – коэффициент сопротивления данного тела), то на тело действует лишь сила тяжести Земли. Соответственно достаточно легко построить модель, визуализирующую движение точки в зависимости от параметров. Рисунок 28.

```
> horizontal_speed := 10 :
> ball := plots[pointplot]([0, 100000], symbolsize = 40, symbol = solidcircle, color = "RoyalBlue") :
> ball_thrown := plots[animate](plottools[translate], [ball, time·horizontal_speed, -9.81·time^2/2], time = 0 ..100);
```

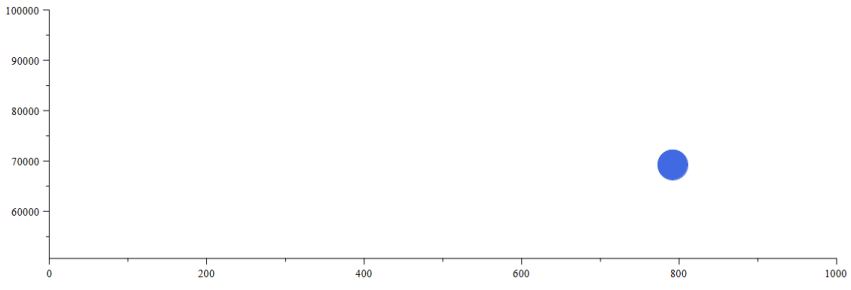


Рисунок 28. Движение материальной точки

Построение визуализации разложения функции в тригонометрический ряд Фурье

По теореме Дирихле непрерывная функция сходится к значению суммы ряда Фурье в этой точке. Команда “animate” достаточно плохо справляется с задачей построения фрейма с анимацией плавного построения всё более и более точного ряда Фурье, соответствующего функции. Интерпретация программы может занимать порядка нескольких минут времени. Однако, так как автор данной работы чуть глубже вник в вопросы визуализации и анимации, для него не является проблемой предложить относительно очевидное решение данной проблемы. Оно было описано несколькими главами ранее. Решение заключается в использовании функции “display” с параметром “insequence”. Обработка такого кода занимает порядка 10-15 секунд, но при этом в целом можно получить более наглядную картину. Когда это может понадобиться ? Тогда, когда речь идёт о вычислении порядка, при котором значение суммы частичной суммы ряда Фурье в точке отличается от фактического значения функции в точке не более, чем на заданную точность. Порой это определяется именно графически, то есть визуально. Умение строить подобные наборы кадров может значительно упростить вычисление этого минимального порядка частично суммы, особенно если речь идёт о достаточно больших значениях частичной суммы и, соответственно, достаточно малых значениях точности, что также возможно, так как Maple позволяет масштабировать графики, то есть даже большие порядки в теории можно искать по графику. Рисунок 29 демонстрирует процесс построения такой анимации. Код вычисления коэффициентов ряда Фурье оставлен выше и не располагается на скриншоте в целях экономии места.

```
> for i from 1 by 1 to 20 do
  array_plots[index1] := plot([GetFourierSumValue(expression1, index1, π,-π, π), expression1], x = -Pi..Pi, thickness = 1, color = "RoyalBlue") :
  index1 := index1 + 1 :
  end do:
> plots[display](array_plots[1], array_plots[2], array_plots[3], array_plots[4], array_plots[5], array_plots[6], array_plots[7] , array_plots[8], array_plots[9],
    array_plots[10], array_plots[11], array_plots[12], array_plots[13], array_plots[14], array_plots[15], array_plots[16], array_plots[17], array_plots[18],
    array_plots[19], array_plots[20], insequence = true);
```

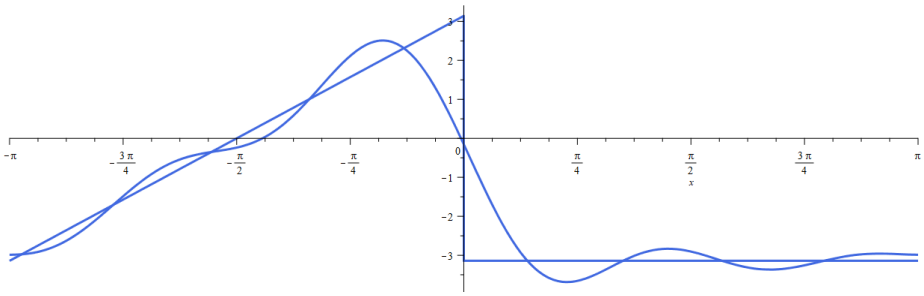


Рисунок 29. Один из кадров (который соответствует какому-то порядку частично суммы) из анимации по разложению в ряд Фурье

Заключение

В рамках заключения и выводов к данной курсовой работе автор хотел бы немного возвысить значимость данной темы. На первый взгляд кажется, что курсовая на тему анимаций является достаточно простой и не очень интересной для выполнений и исследований.
Необходимо отметить, что анимация в математике очень часто помогает показать сложные процессы более наглядно, хорошее понимание построения анимаций очень сильно “прокачивает” математическую базу. С базовыми анимациями такого нет. Но если копнуть хотя бы несколько глубже, что было проделано в данной работе, то можно извлечь большую пользу, улучшить знания по предмету, прокачать навыки работы со системой компьютерной алгебры Maple и в целом получить больше эмоций от выполнения, нежели при выполнении работы с сухими числами и данными без понимания того, как это могло бы выглядеть в природе.
В качестве итогов работы автор может отметить, что анимация помогает глубже вникнуть именно в математику, а также в дизайн, искусство. Автор разобрался во всём, что было изложено выше и считает, что это был очень полезный опыт, который поможет (и уже помог) не только в выполнении лабораторных работ по дисциплине, но и в будущем при работе с анализом данных и так далее.

Вместе с данной пояснительной запиской предлагается вниманию читателей рабочий файл Maple, где собраны все примеры, продемонстрированные в этой курсовой работе.

Список использованных источников

- [1] Документация Maple, пакет “plots”. Режим доступа: URL: https://www.maplesoft.com/support/help/maple/view.aspx?path=plots
- [2] Документация Maple, команда “plot”. Режим доступа: URL: https://www.maplesoft.com/support/help/maple/view.aspx?path=plot
- Официальная документация системы компьютерной алгебры Maple. Режим доступа: URL: : https://www.maplesoft.com/
- “Визуализация решений некоторых математических задач в Maple” – Кузнечик В.А. , Милинкевич М.И. , БГУИР 2019 г.
- “Расширенные средства графики - Боровское исследовательское учреждение” Режим доступа: URL: http://bourabai.kz/cm/le12
- [3] Документация Maple, пакет “plots”. Режим доступа: URL: https://www.maplesoft.com/support/help/maple/view.aspx?path=plottools
- “Maple. Примеры сложных программ.” Режим доступа: URL: https://www.mapleprimes.com/
- Цветовая палитра для визуализации Maple. Режим доступа: URL: https://www.maplesoft.com/support/help/maple/view.aspx?path=plot%2Fcolornames