Отчёт по лабораторной работе №5

дисциплина: Компьютерный практикум по статистическому анализу данных

Студент: Кузнецова София Вадимовна

Содержание

# Цель работы

Основная цель работы — освоить синтаксис языка Julia для построения графиков.

# Выполнение лабораторной работы

## Основные пакеты для работы с графиками в Julia

Julia поддерживает несколько пакетов для работы с графиками. Использование того или иного пакета зависит от целей, преследуемых пользователем при построении. Стандартным для Julia является пакет Plots.jl. Рассмотрим построение графика функции f(x) = (3x2 + 6x - 9)e-0,3x разными способами:

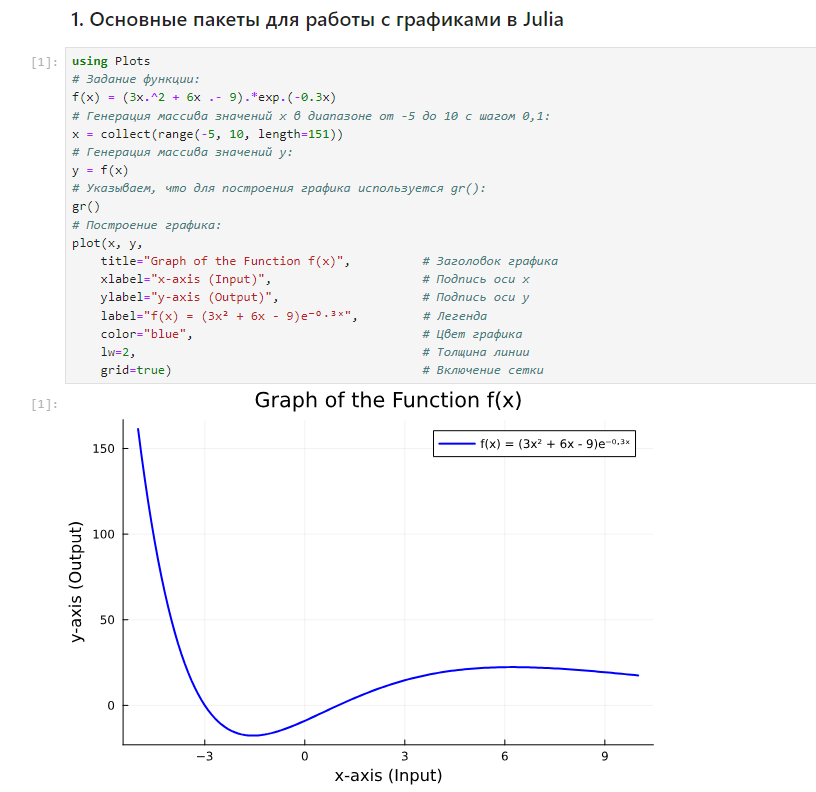


График функции, построенный при помощи gr()

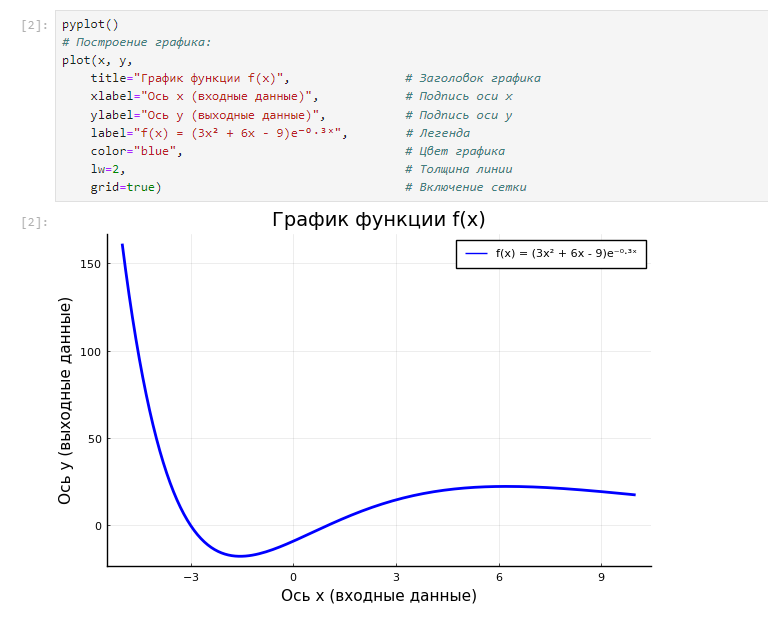


График функции, построенный при помощи pyplot()

## Опции при построении графика

На примере графика функции sin(x) и графика разложения этой функции в ряд Тейлора рассмотрим дополнительные возможности пакетов для работы с графикой:

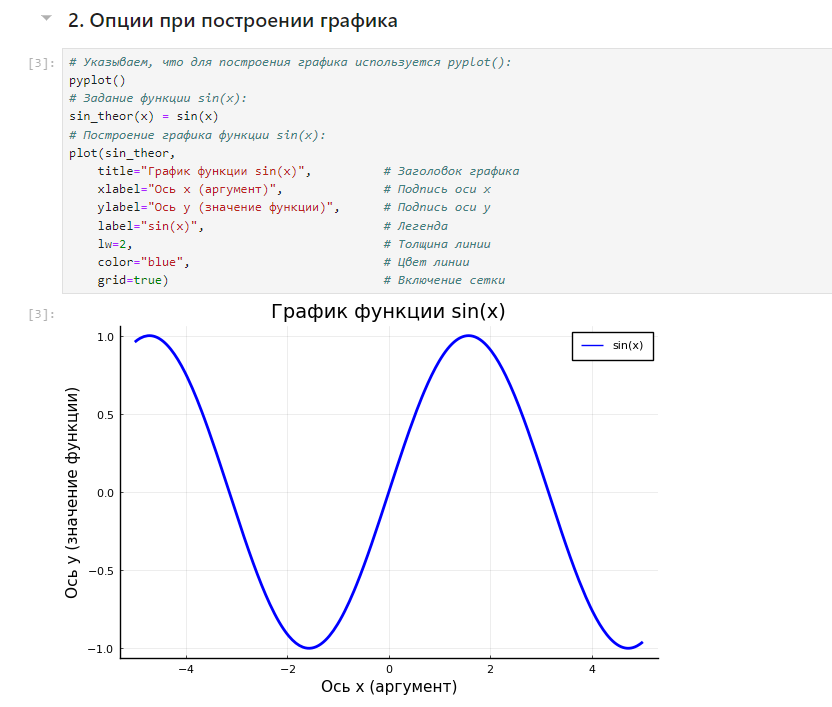


График функции sin(x)

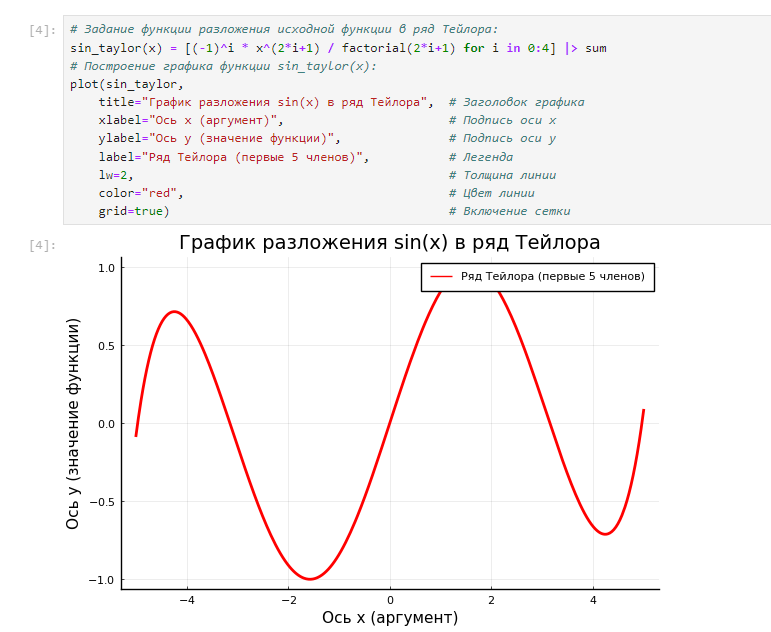
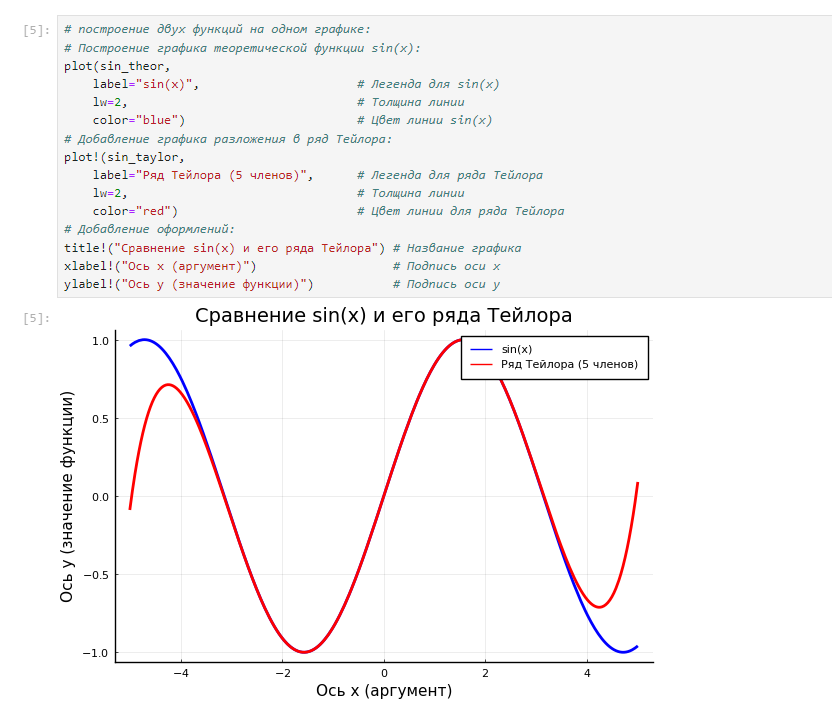
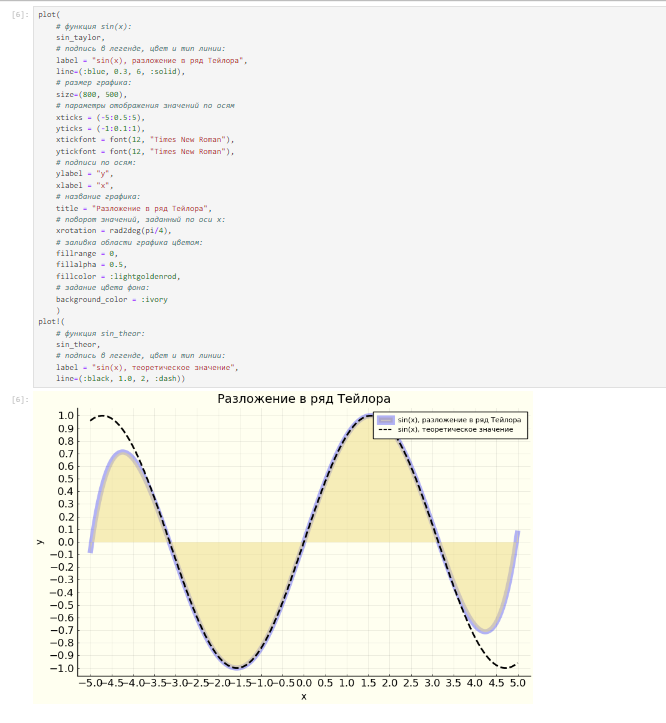


График функции разложения исходной функции в ряд Тейлора



Графики исходной функции и её разложения в ряд Тейлора

Затем добавим различные опции для отображения на графике:



Вид графиков после добавления опций при их построении

## Точечный график

Как и при построении обычного графика для точечного графика необходимо задать массив значений x, посчитать или задать значения y, задать опции построения графика:

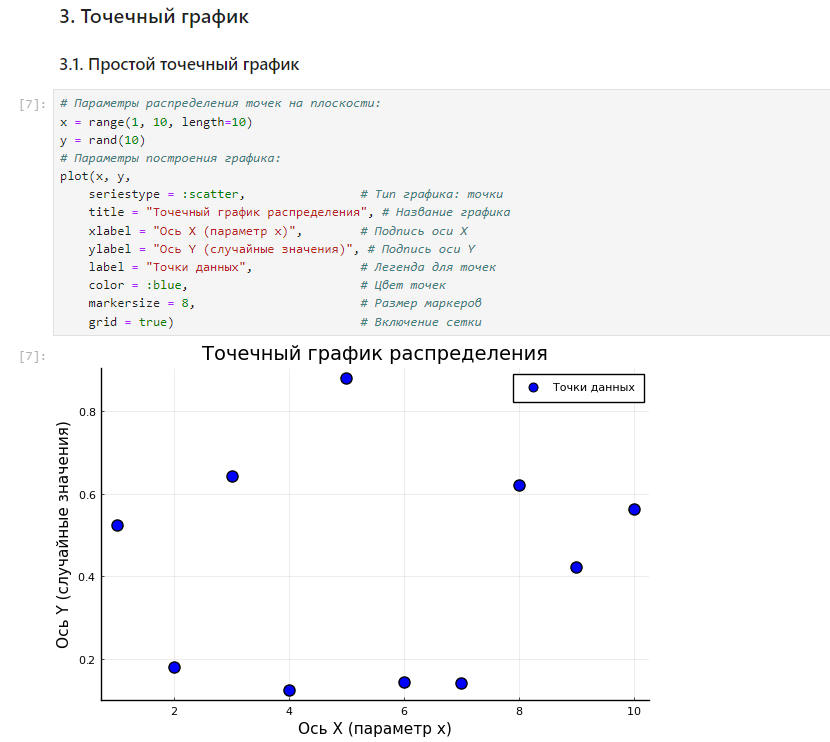


График десяти случайных значений на плоскости (простой точечный график)

Для точечного графика можно задать различные опции, например размер маркера, его тип, цвет и и т.п.:

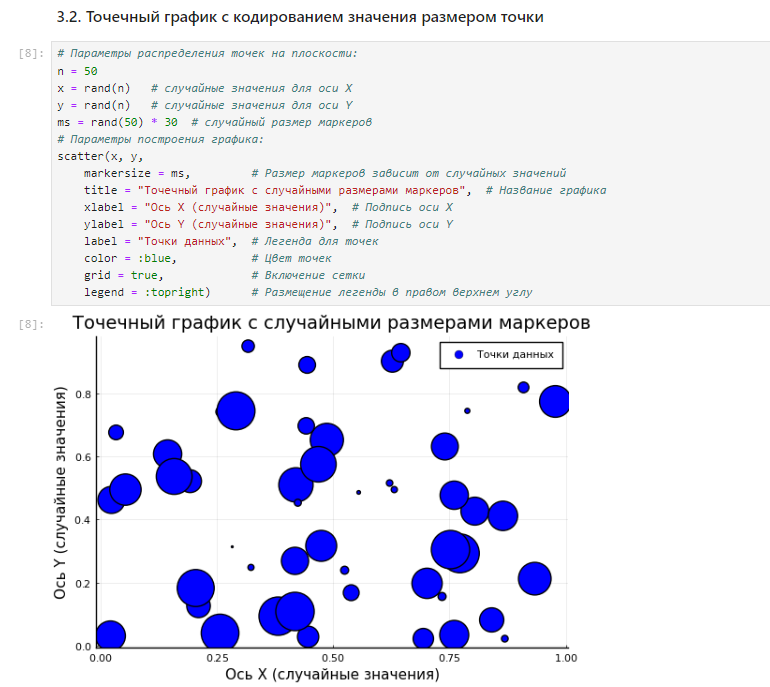


График пятидесяти случайных значений на плоскости с различными опциями отображения (точечный график с кодированием значения размером точки)

Также можно строить и 3-мерные точечные графики:

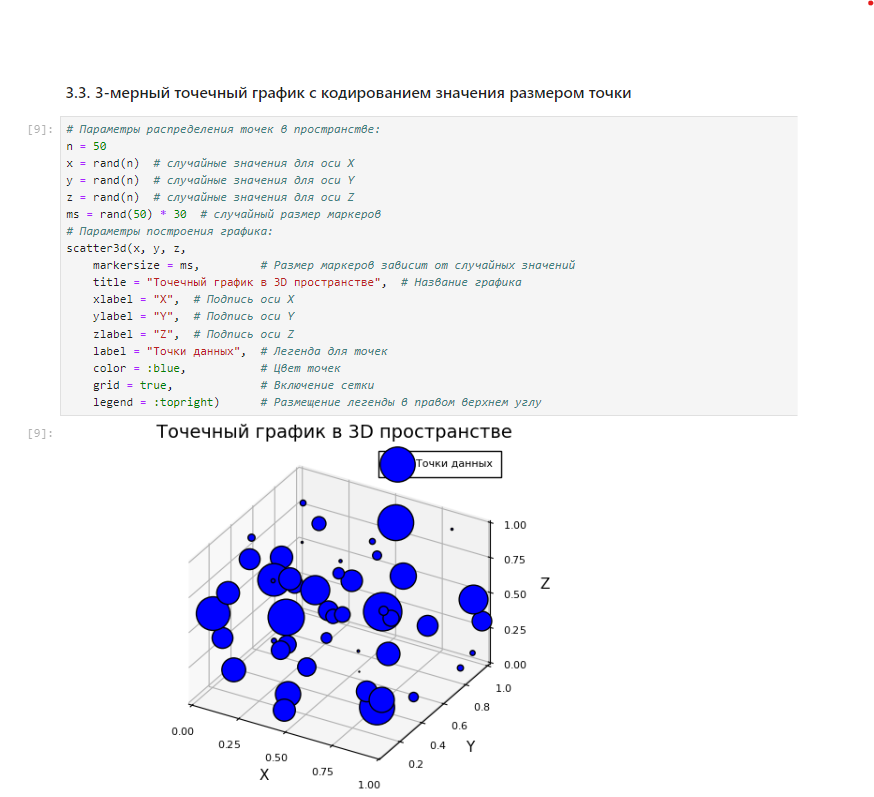
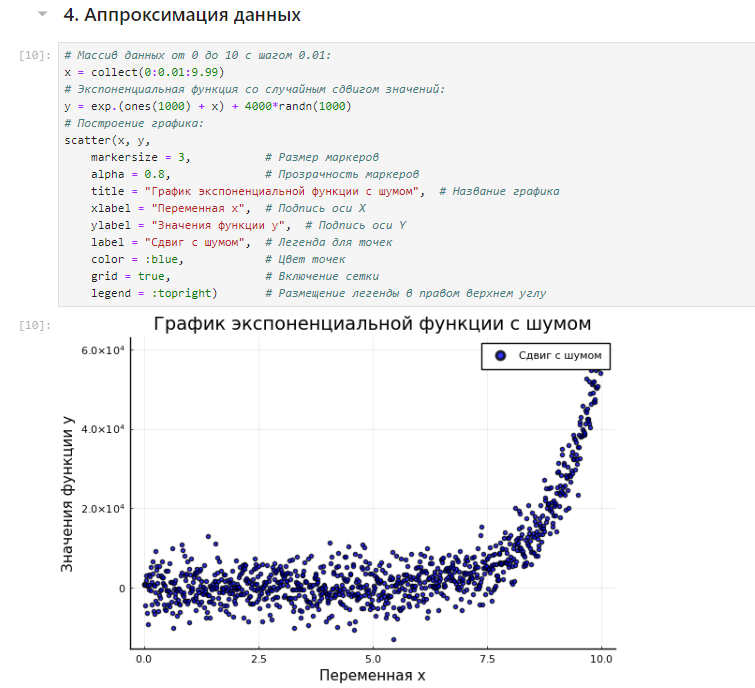


График пятидесяти случайных значений в пространстве с различными опциями отображения (3-мерный точечный график с кодированием значения размером точки)

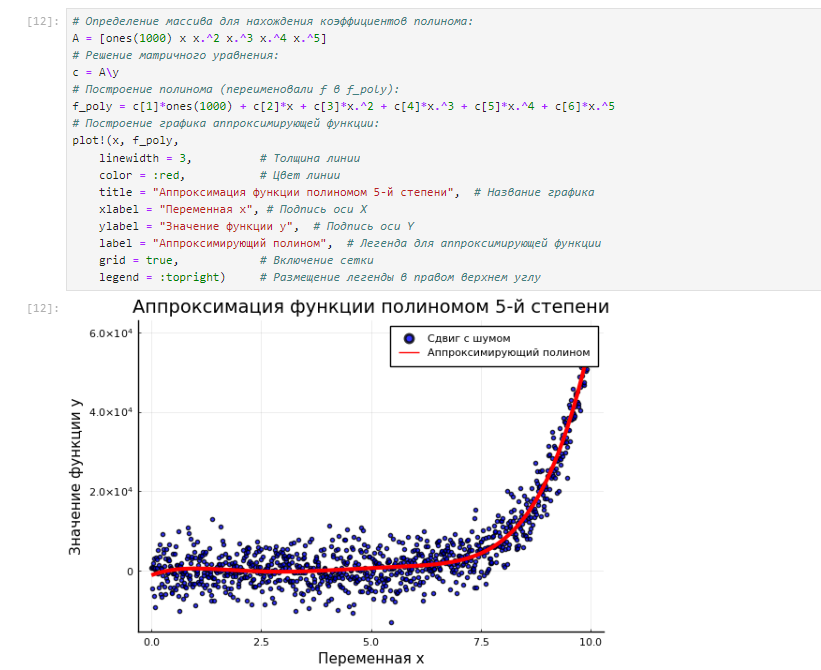
## Аппроксимация данных

Аппроксимация — научный метод, состоящий в замене объектов их более простыми аналогами, сходными по своим свойствам. Для демонстрации зададим искусственно некоторую функцию, в данном случае похожую по поведению на экспоненту:



Пример функции

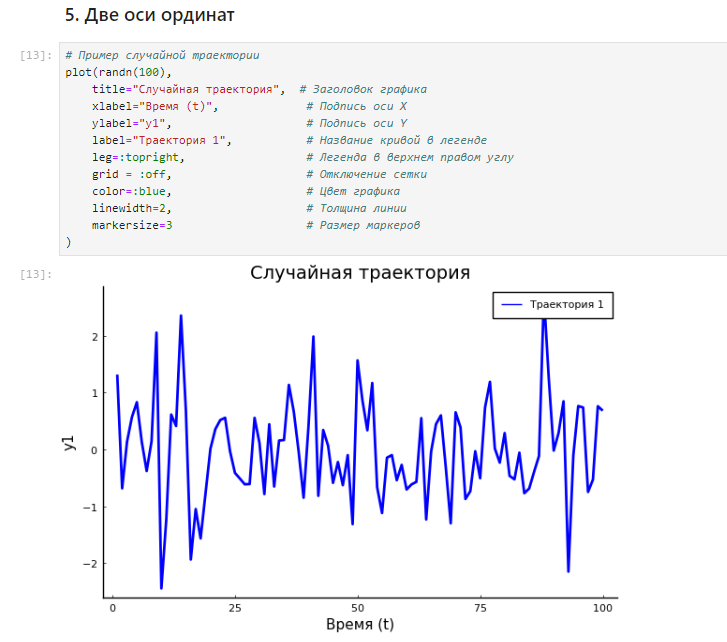
Аппроксимируем полученную функцию полиномом 5-й степени:



Пример аппроксимации исходной функции полиномом 5-й степени

## Две оси ординат

Иногда требуется на один график вывести несколько траекторий с существенными отличиями в значениях по оси ординат. Пример первой траектории:



Пример отдельно построенной траектории

## Полярные координаты

Приведём пример построения графика функции в полярных координатах:

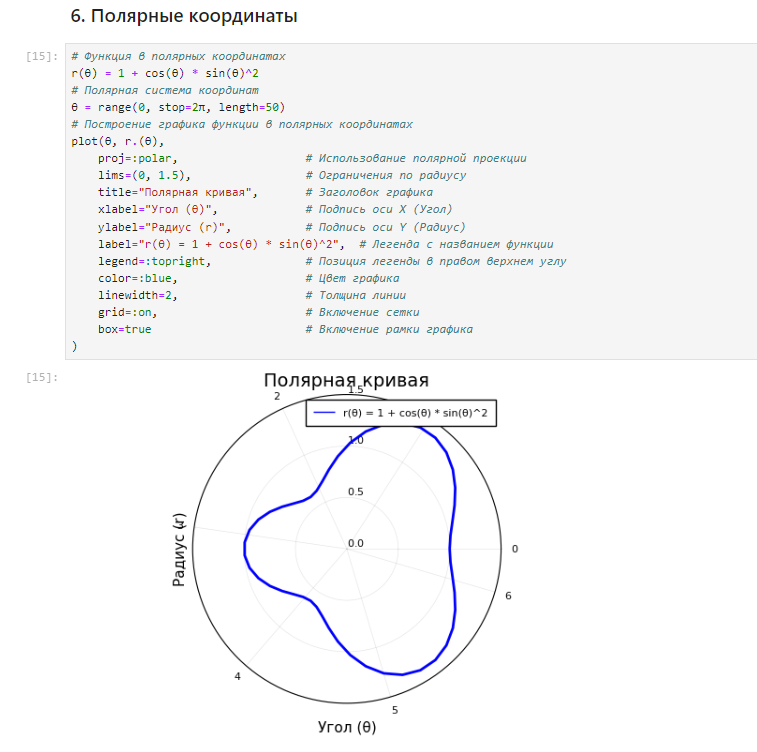
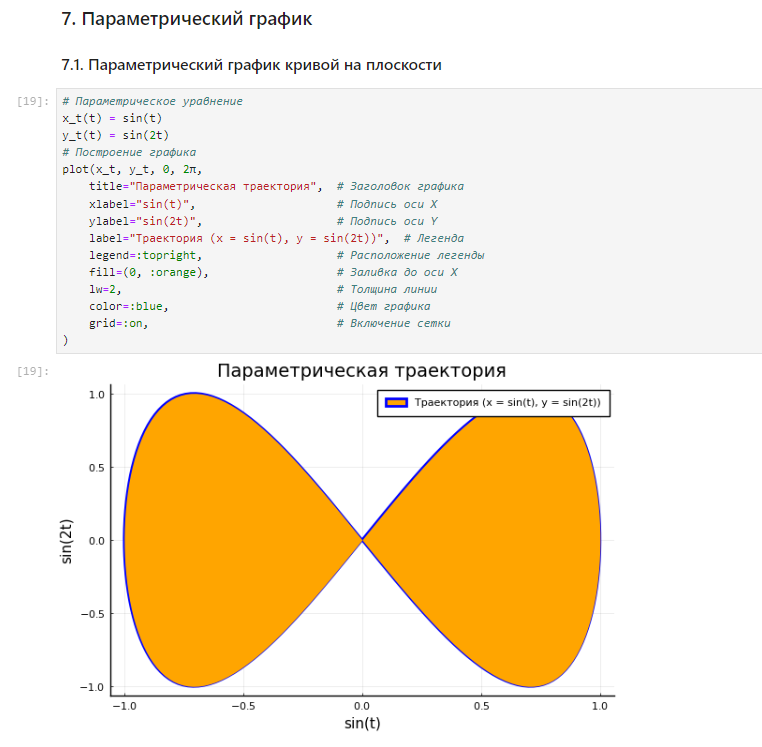


График функции, заданной в полярных координатах

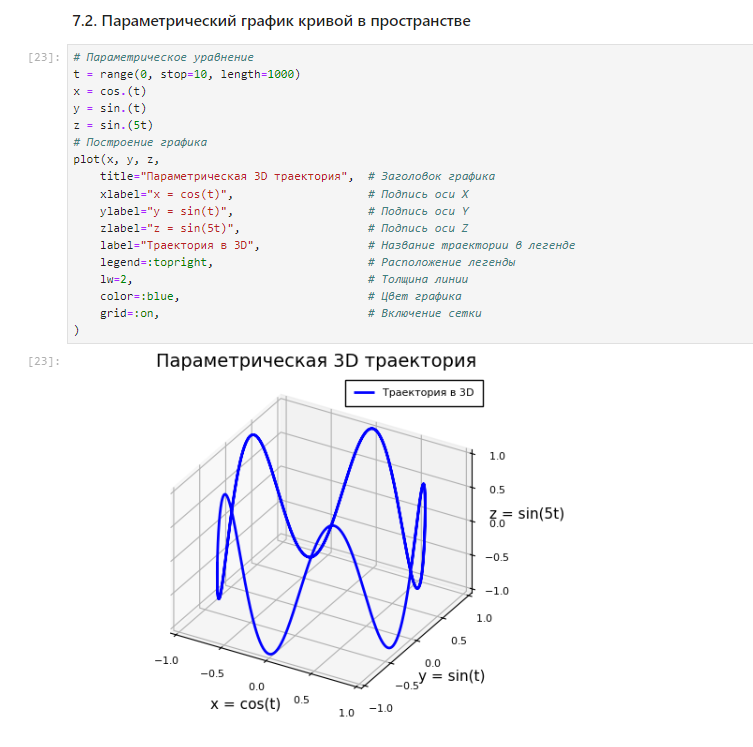
## Параметрический график

Приведём пример построения графика параметрически заданной кривой на плоскости:



Параметрический график кривой на плоскости

Далее приведём пример построения графика параметрически заданной кривой в пространстве:



Параметрический график кривой в пространстве

## График поверхности

Для построения поверхности, заданной уравнением f(x, y) = x2 + y2, можно воспользоваться функцией surface():

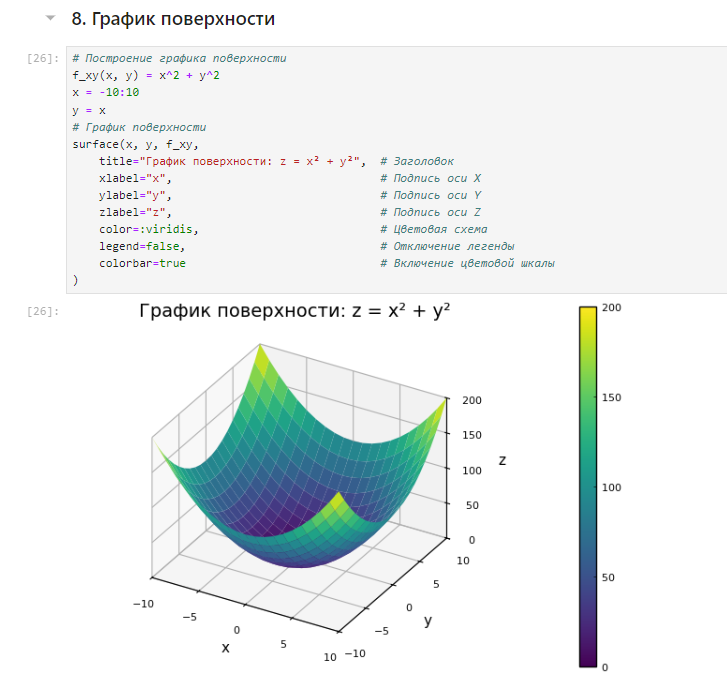


График поверхности (использована функция surface())

Также можно воспользоваться функцией plot() с заданными параметрами:

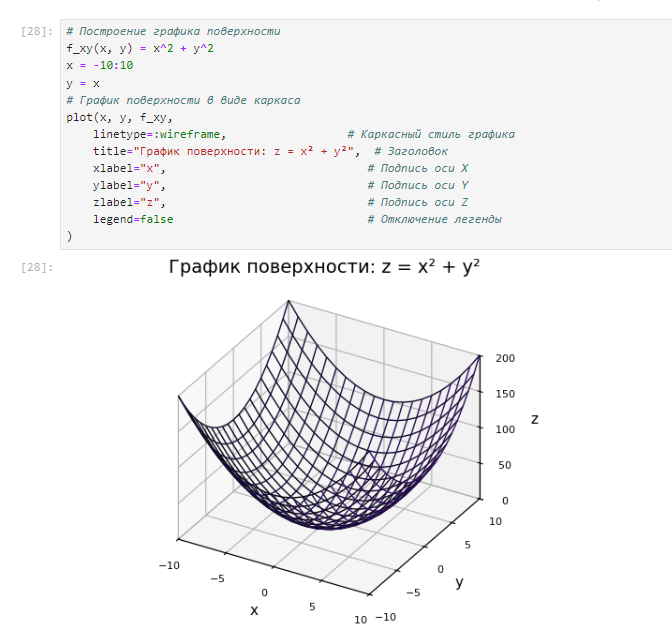
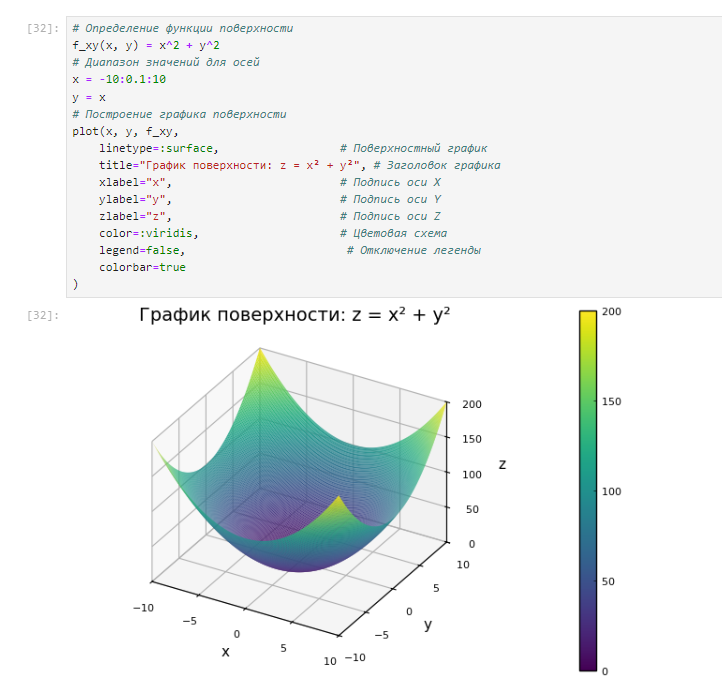


График поверхности (использована функция plot())

Можно задать параметры сглаживания:



Сглаженный график поверхности

Можно задать определённый угол зрения:

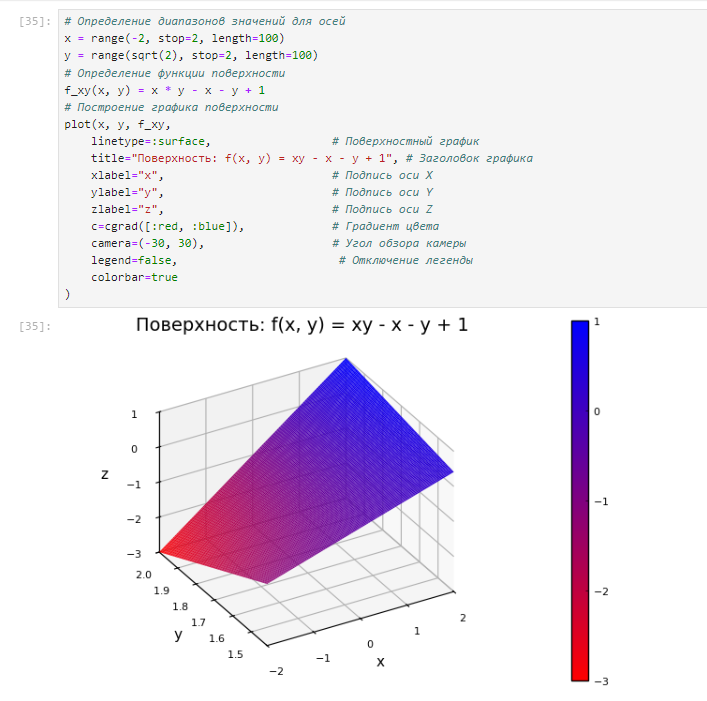


График поверхности с изменённым углом зрения

## Линии уровня

Линией уровня некоторой функции от двух переменных называется множество точек на координатной плоскости, в которых функция принимает одинаковые значения. Линий уровня бесконечно много, и через каждую точку области определения можно провести линию уровня. С помощью линий уровня можно определить наибольшее и наименьшее значение исходной функции от двух переменных. Каждая из этих линий соответствует определённому значению высоты. Поверхности уровня представляют собой непересекающиеся пространственные поверхности. Рассмотрим поверхность, заданную функцией g(x, y) = (3x + y2)| sin(x) + cos(y)|:

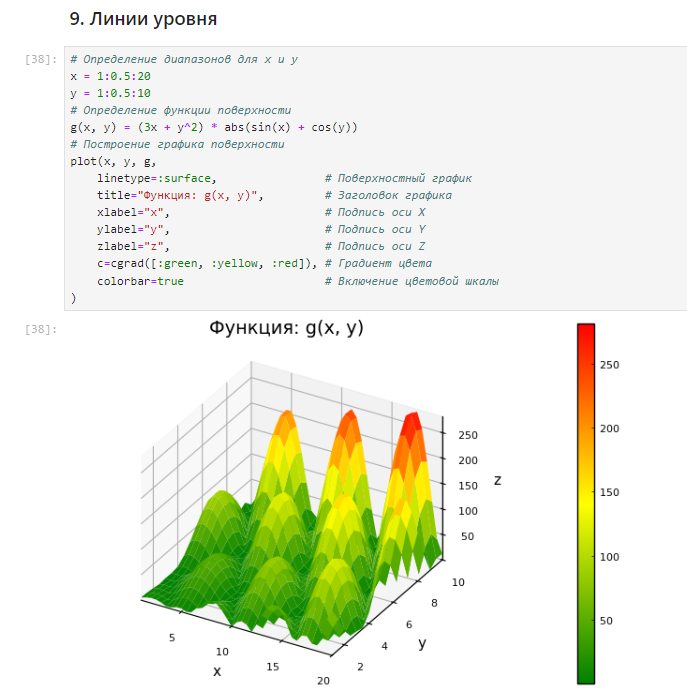
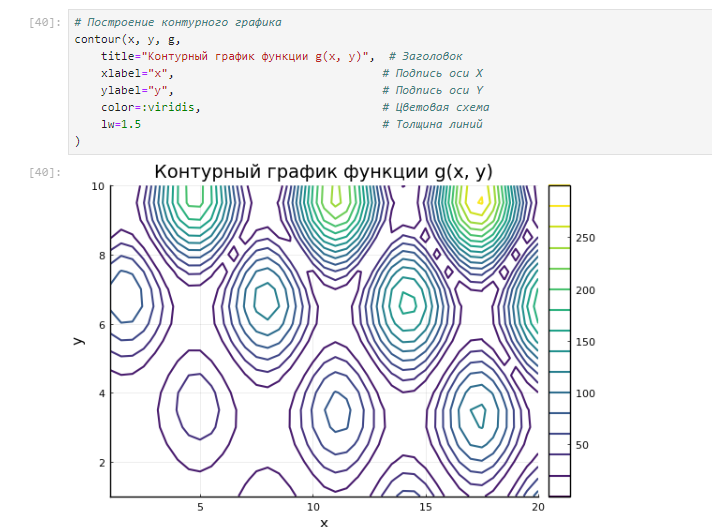


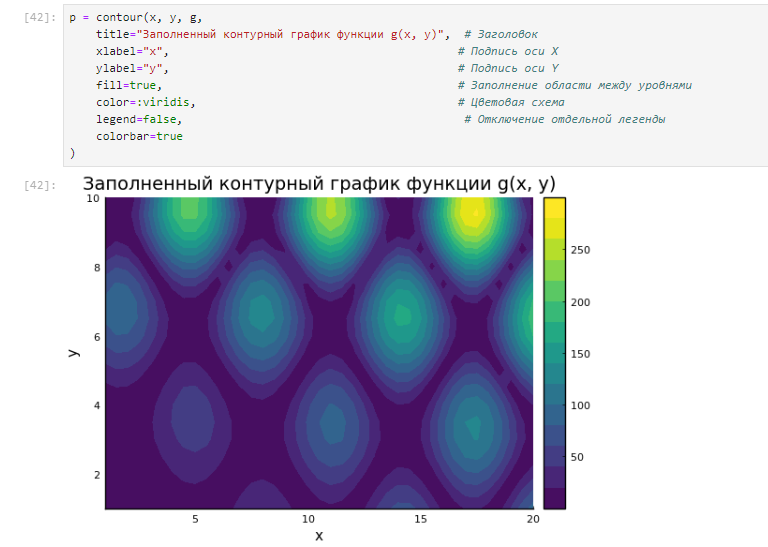
График поверхности, заданной функцией g(x, y) = (3x + y2)| sin(x) + cos(y)|

Линии уровня можно построить, используя проекцию значений исходной функции на плоскость:



Линии уровня

Можно дополнительно добавить заливку цветом:



Линии уровня с заполнением

## Векторные поля

Если каждой точке некоторой области пространства поставлен в соответствие вектор с началом в данной точке, то говорят, что в этой области задано векторное поле. Векторные поля задают векторными функциями. Для функции h(x, y) = x3 - 3x + y2 сначала построим её график и линии уровня:

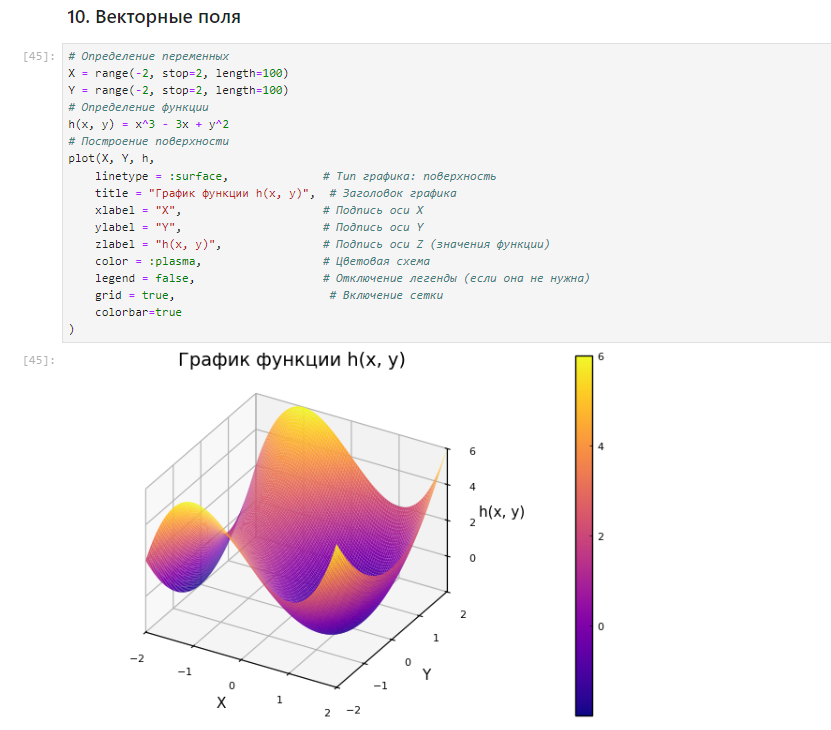
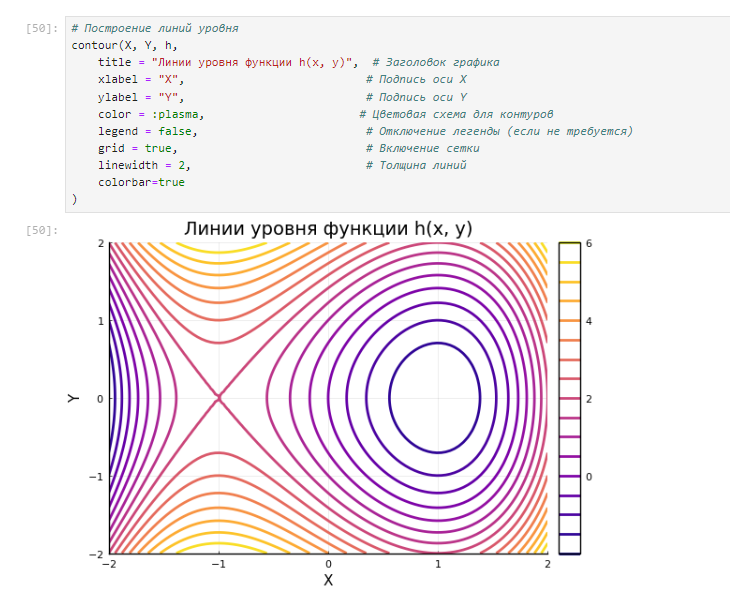


График функции h(x, y) = x3 - 3x + y2



Линии уровня для функции h(x, y) = x3 - 3x + y2

Векторное поле можно охарактеризовать векторными линиями. Каждая точка векторной линии является началом вектора поля, который лежит на касательной в данной точке. Для нахождения векторной линии требуется решить дифференциальное уравнение.

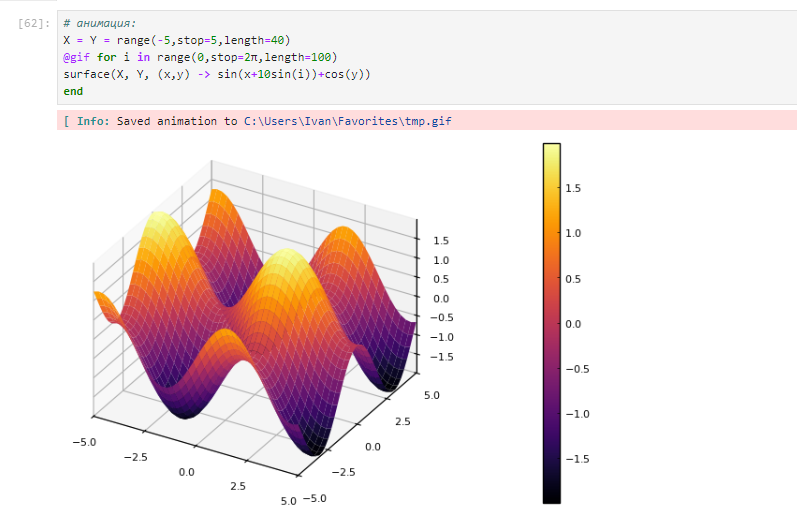
## Анимация

Технически анимированное изображение представляет собой несколько наложенных изображений (или построенных в разных точках графиках) в одном файле. В Julia рекомендуется использовать gif-анимацию в pyplot(). Строим поверхность:



Статичный график поверхности

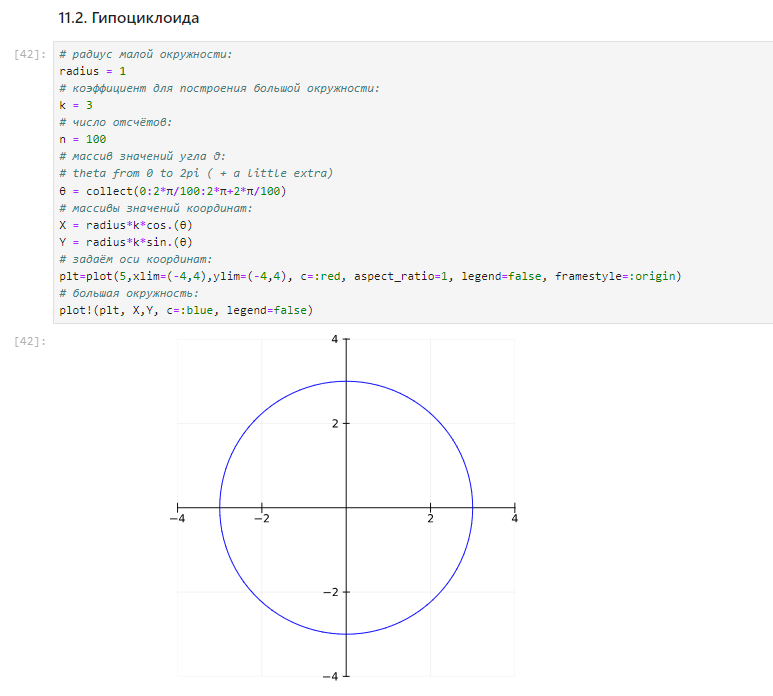
Добавляем анимацию:



Анимированный график поверхности

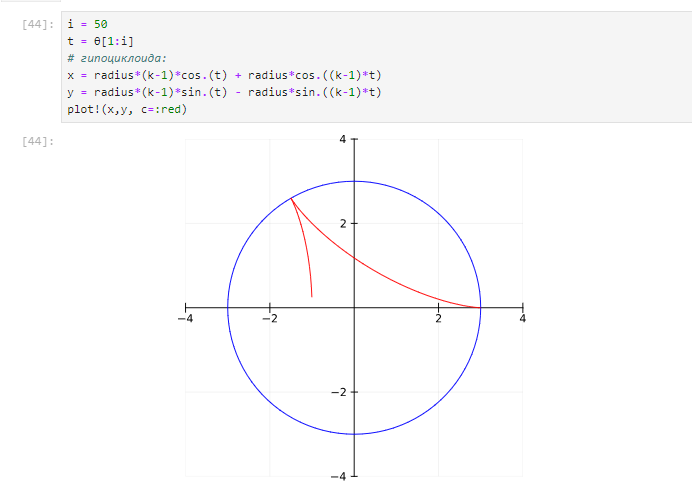
## Гипоциклоида

Гипоциклоида — плоская кривая, образуемая точкой окружности, катящейся по внутренней стороне другой окружности без скольжения. Построим большую окружность:



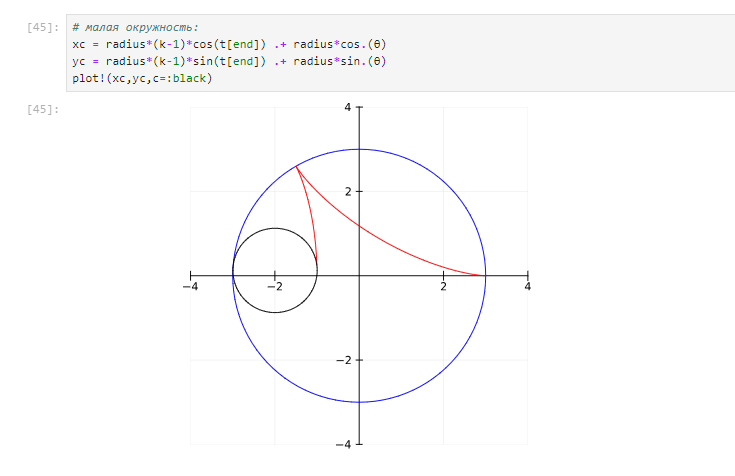
Большая окружность гипоциклоиды

Для частичного построения гипоциклоиды будем менять параметр t:



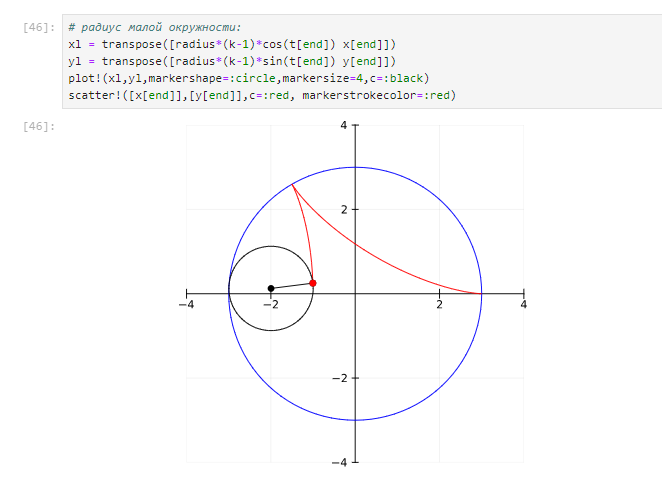
Половина пути гипоциклоиды

Добавляем малую окружность гипоциклоиды:



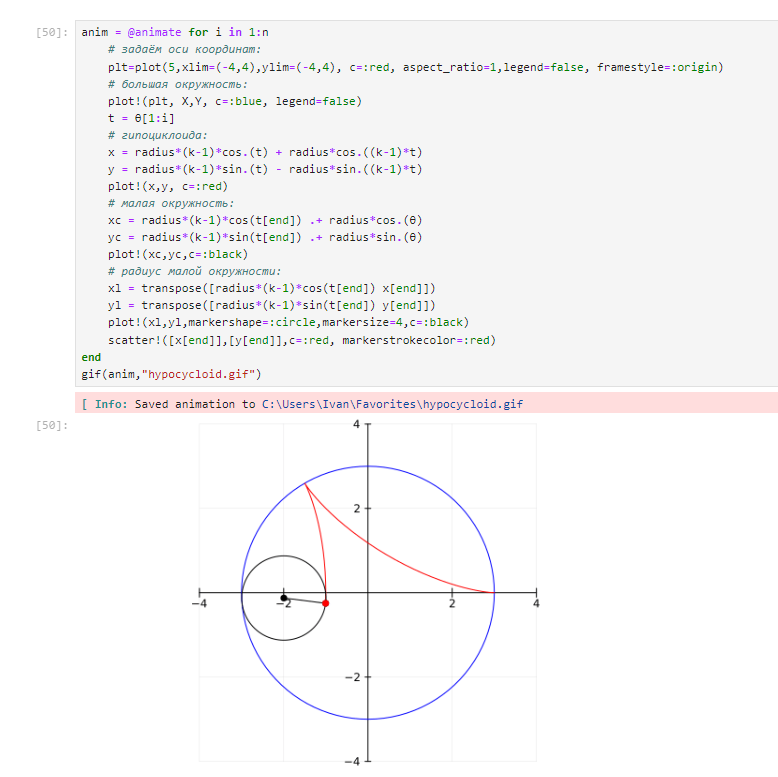
Малая окружность гипоциклоиды

Добавим радиус для малой окружности:



Малая окружность гипоциклоиды с добавлением радиуса

В конце сделаем анимацию получившегося изображения:



Малая окружность гипоциклоиды с добавлением радиуса

## Errorbars

В исследованиях часто требуется изобразить графики погрешностей измерения.

Построим график исходных значений:

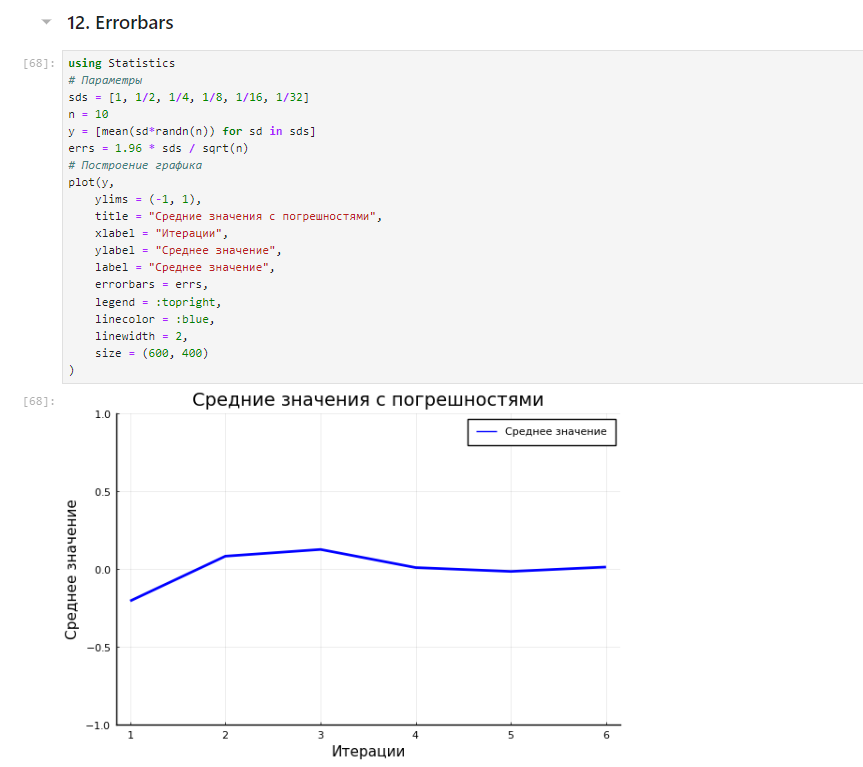


График исходных значений

Построим график отклонений от исходных значений:

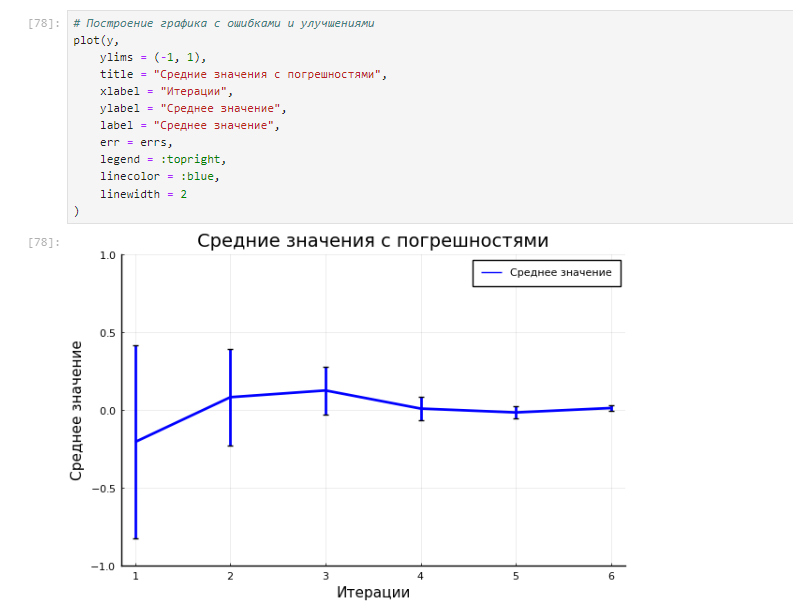
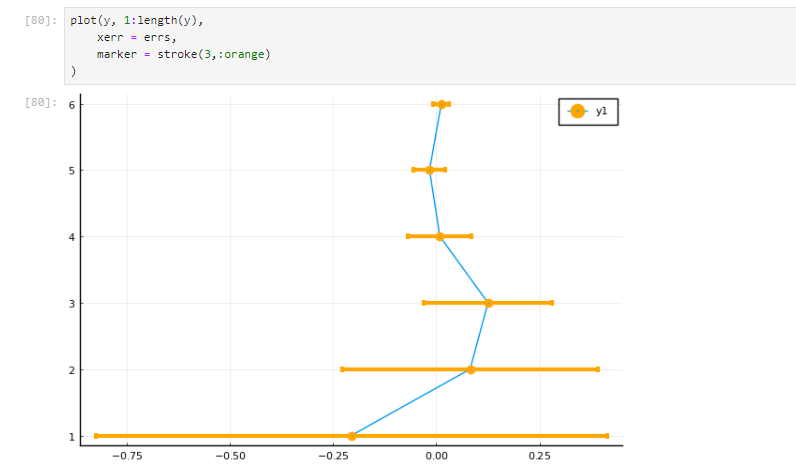


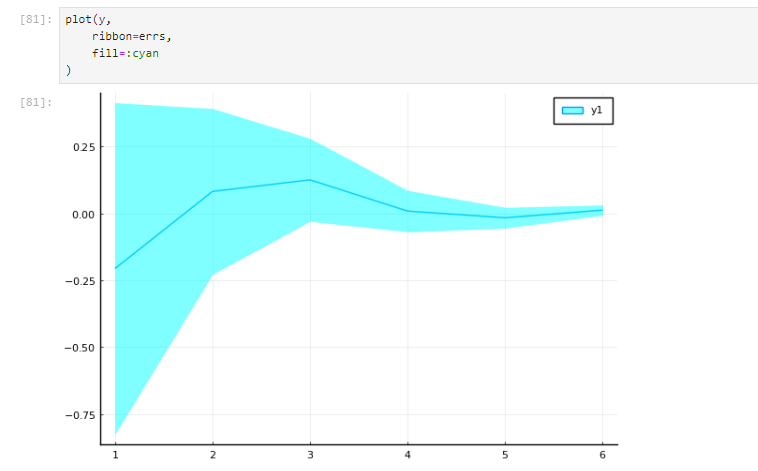
График исходных значений с отклонениями

Повернём график:



Поворот графика

Заполним область цветом:



Заполнение цветом

Можно построить график ошибок по двум осям:

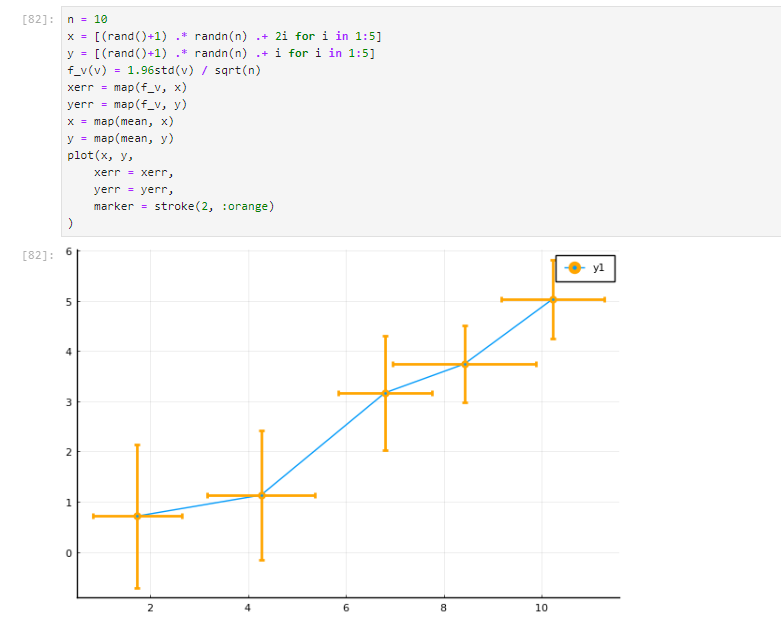


График ошибок по двум осям

Можно построить график асимметричных ошибок по двум осям:

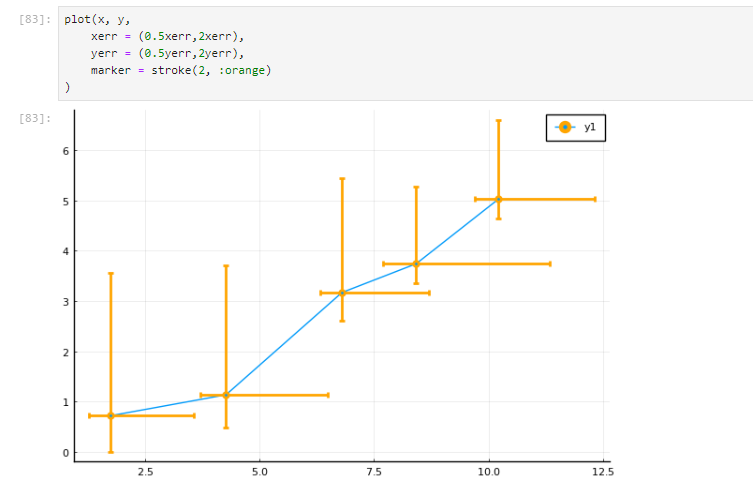
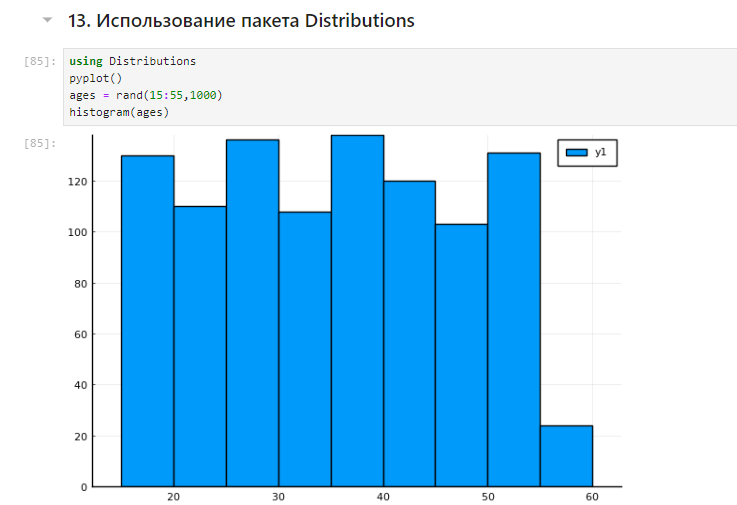


График асимметричных ошибок по двум осям

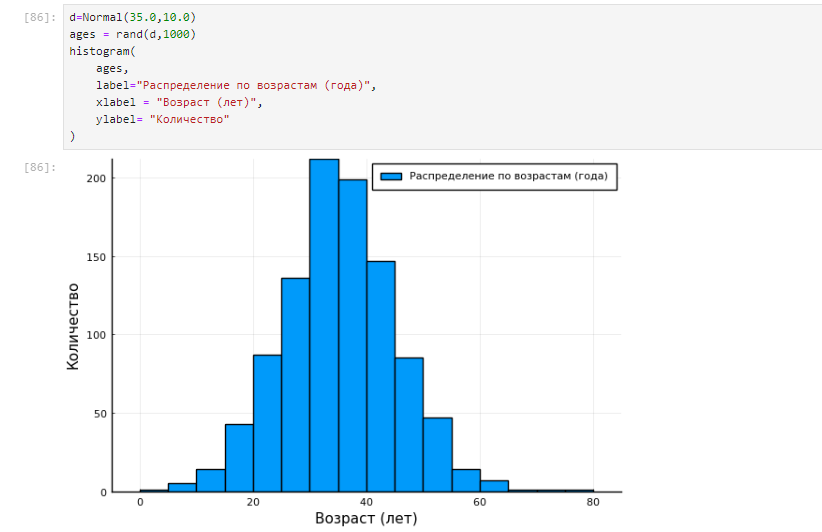
## Использование пакета Distributions

Строим гистограмму:



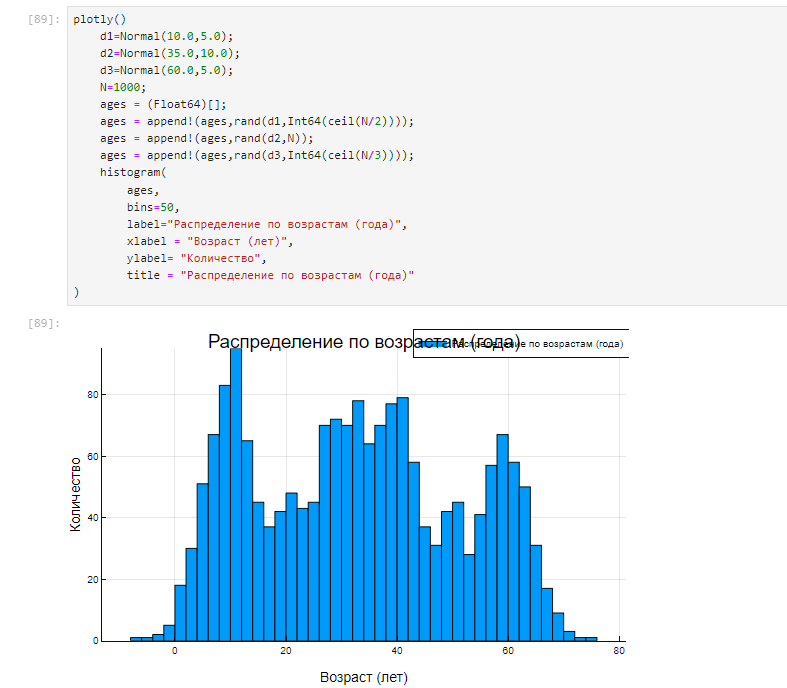
Гистограмма, построенная по массиву случайных чисел

Задаём нормальное распределение и строим гистограмму:



Гистограмма нормального распределения

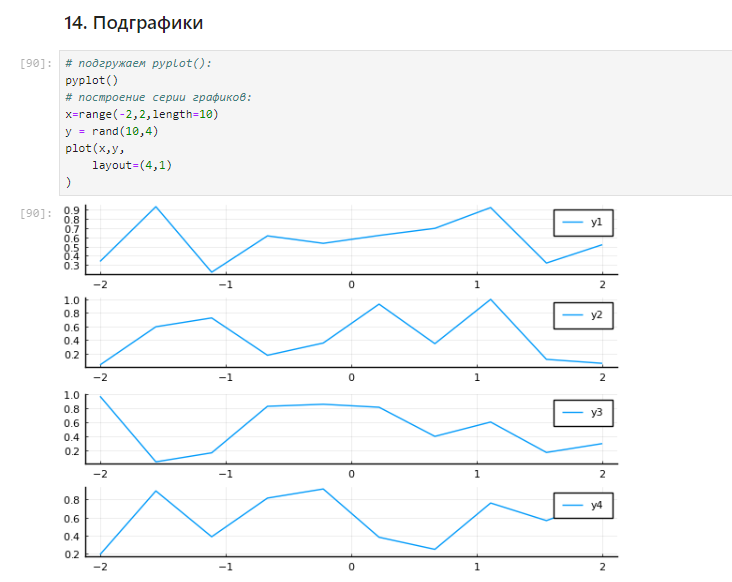
Далее применим для построения нескольких гистограмм распределения людей по возрастам на одном графике plotly():



Гистограмма распределения людей по возрастам

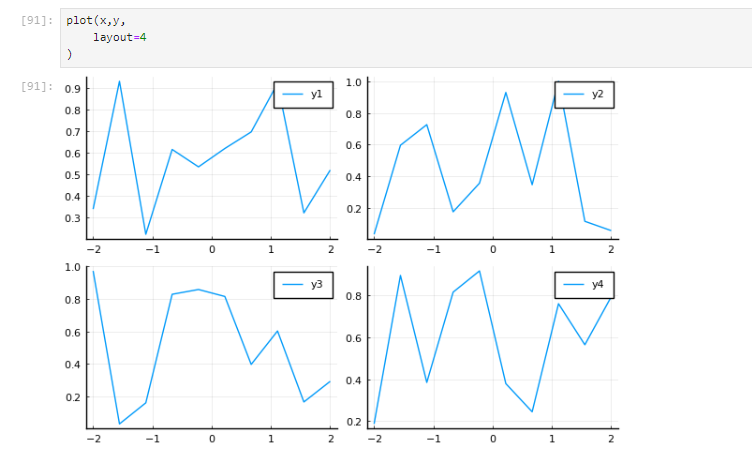
## Подграфики

Определим макет расположения графиков. Команда layout принимает кортеж layout = (N, M), который строит сетку графиков NxM. Например, если задать layout = (4,1) на графике четыре серии, то получим четыре ряда графиков:



Серия из 4-х графиков в ряд

Для автоматического вычисления сетки необходимо передать layout целое число:



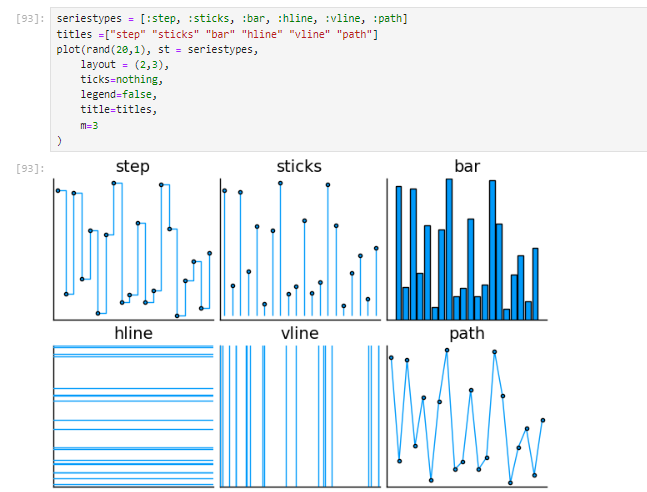
Серия из 4-х графиков в сетке

Аргумент heights принимает в качестве входных данных массив с долями желаемых высот. Если в сумме дроби не составляют 1,0, то некоторые подзаголовки могут отображаться неправильно. Можно сгенерировать отдельные графики и объединить их в один, например, в сетке 2 × 2:



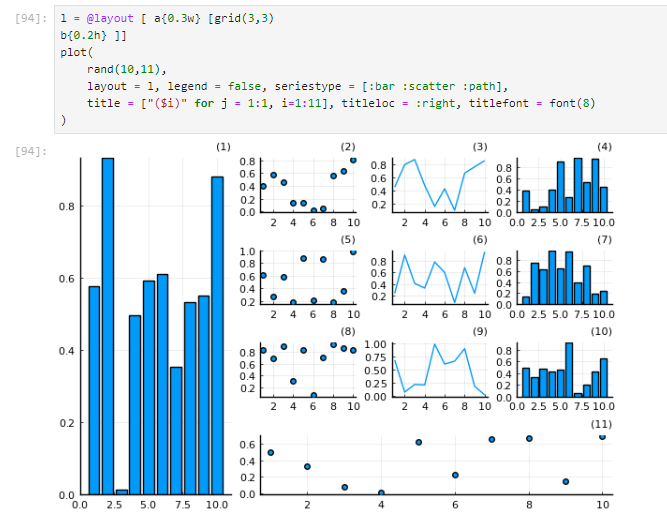
Объединение нескольких графиков в одной сетке

Обратите внимание, что атрибуты на отдельных графиках применяются к отдельным графикам, в то время как атрибуты в последнем вызове plot применяются ко всем графикам. Разнообразные варианты представления данных:



Разнообразные варианты представления данных

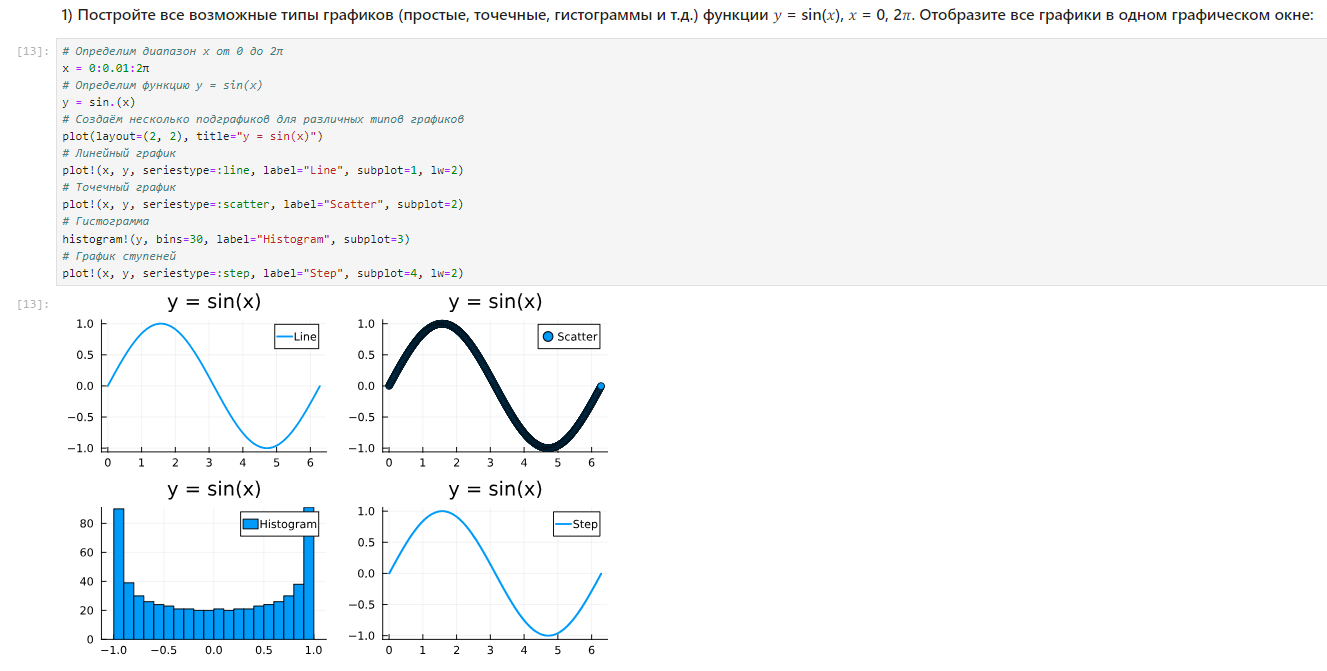
Применение макроса @layout наиболее простой способ определения сложных макетов. Точные размеры могут быть заданы с помощью фигурных скобок, в противном случае пространство будет поровну разделено между графиками:



Демонстрация применения сложного макета для построения графиков

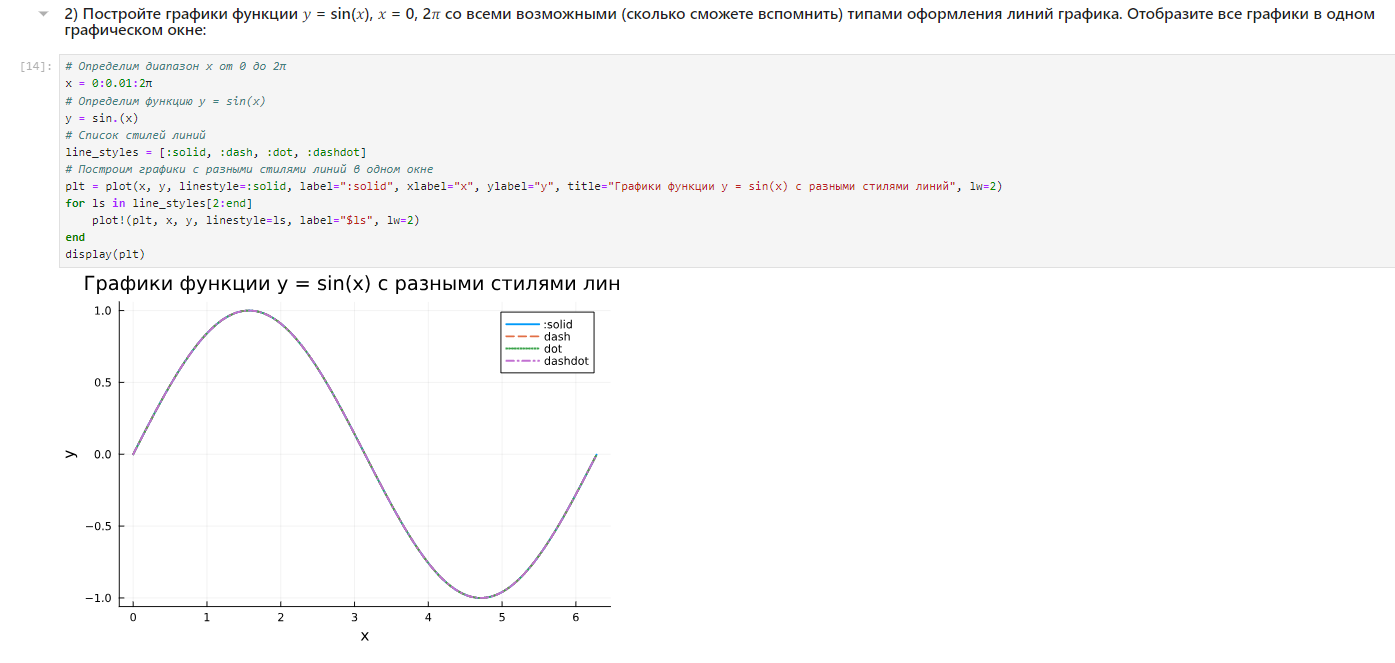
## Самостоятельное выполнение

Выполнение задания №1:



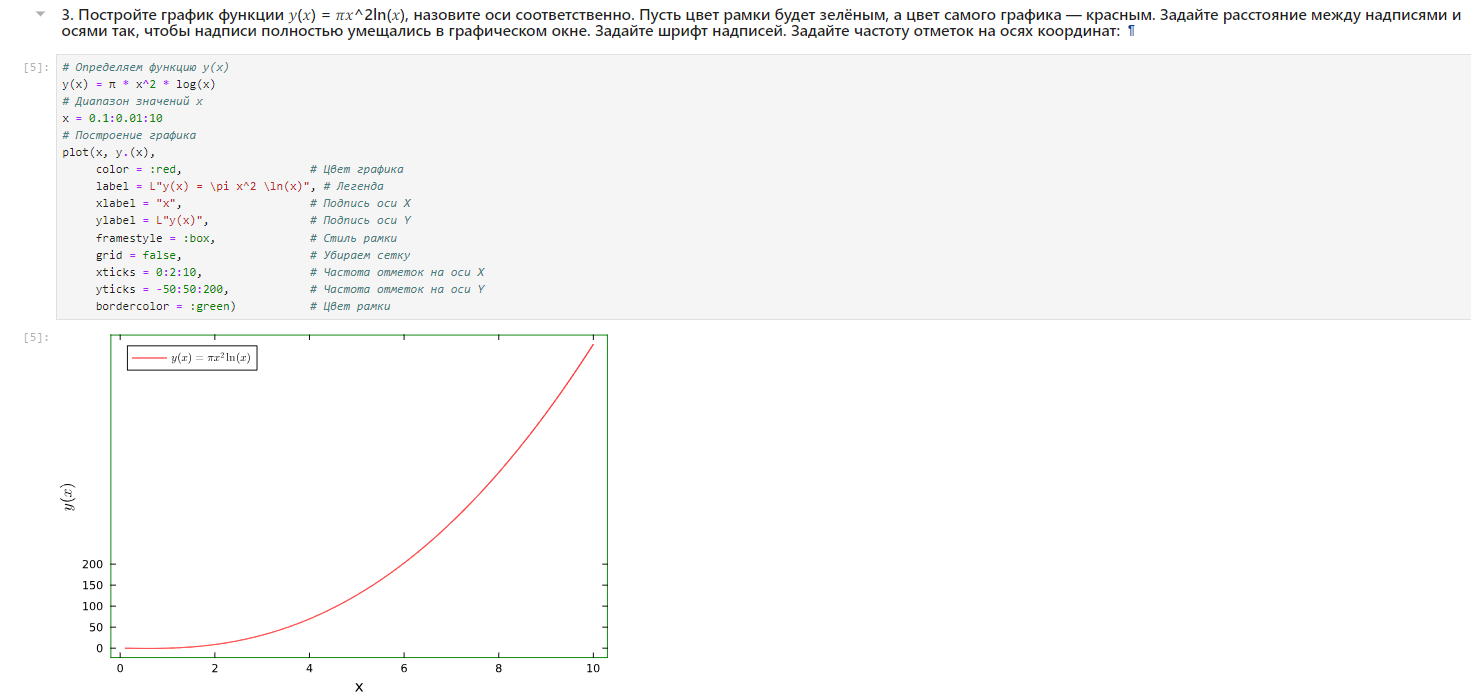
Решение задания №1

Выполнение задания №2:



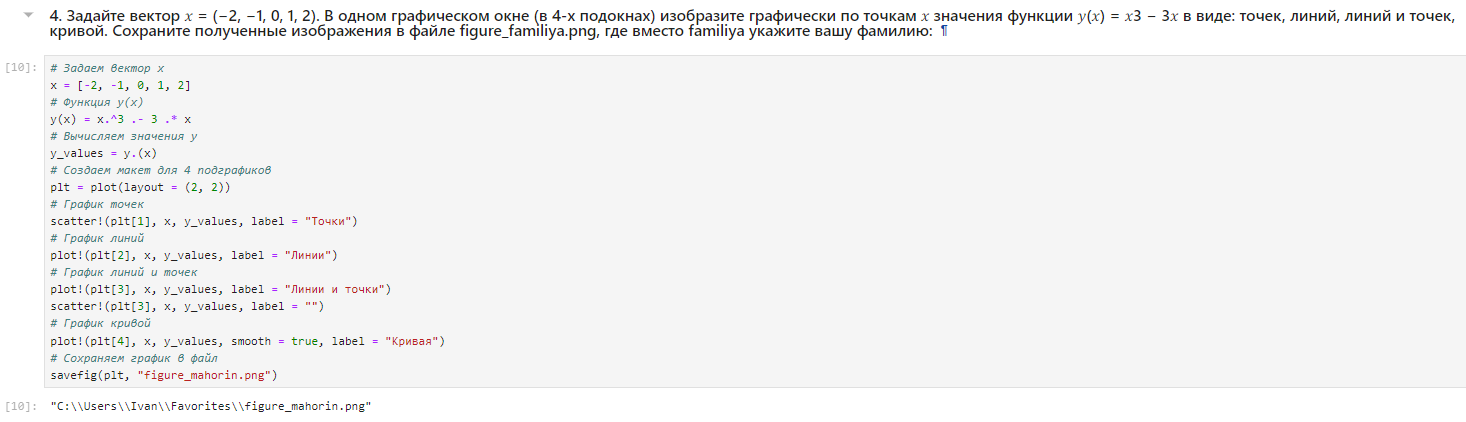
Решение задания №2

Выполнение задания №3:



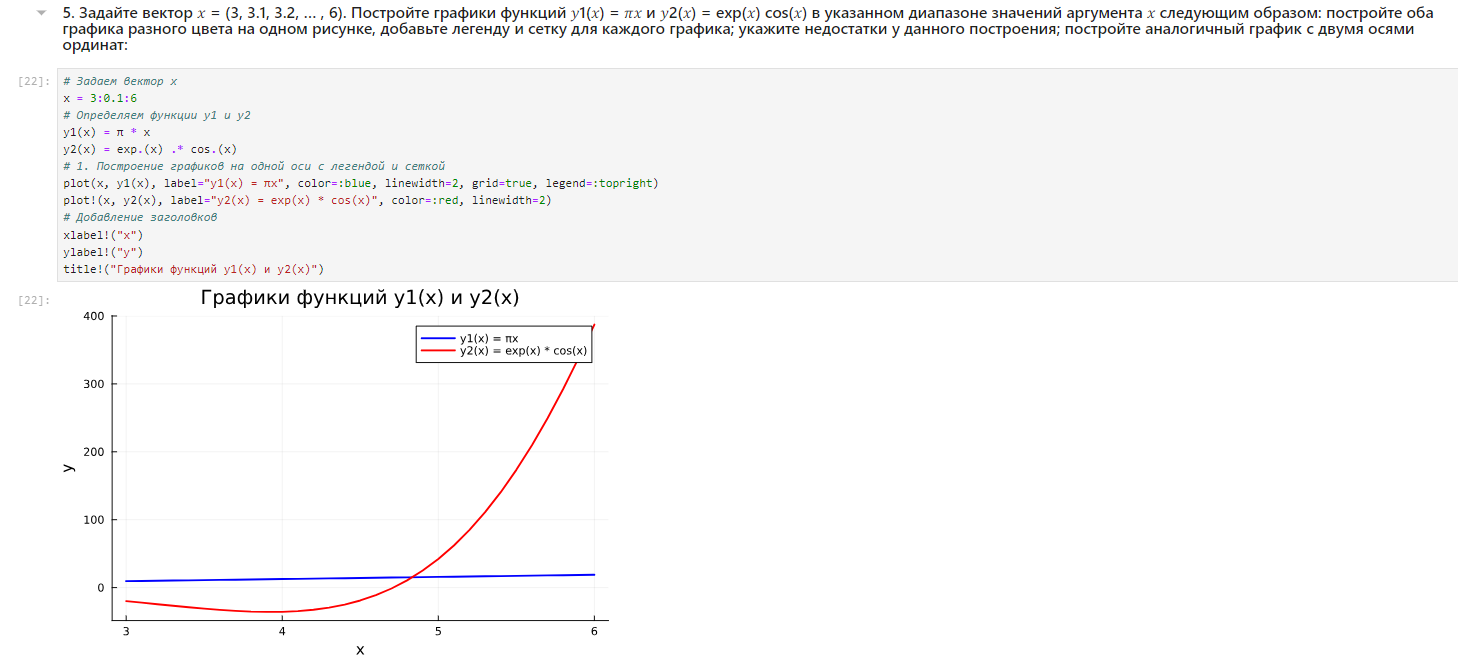
Решение задания №3

Выполнение задания №4:



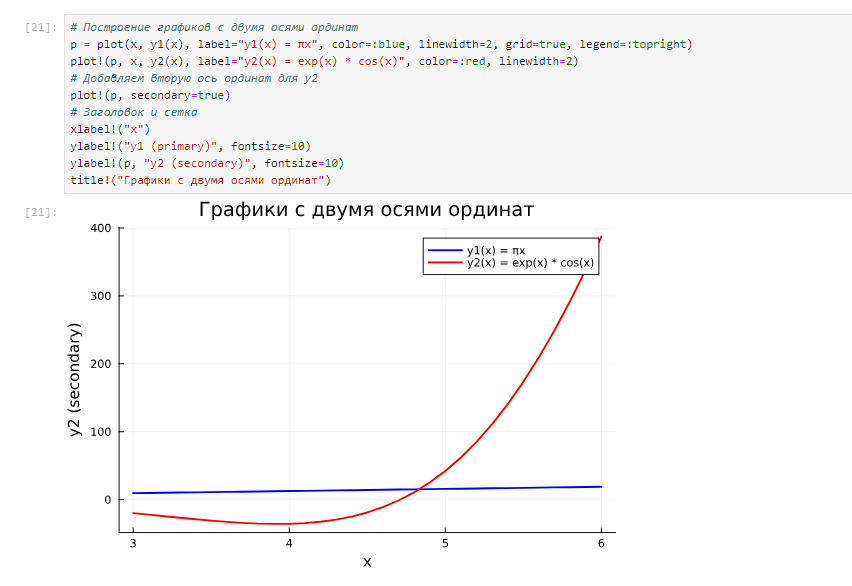
Решение задания №4

Выполнение задания №5:



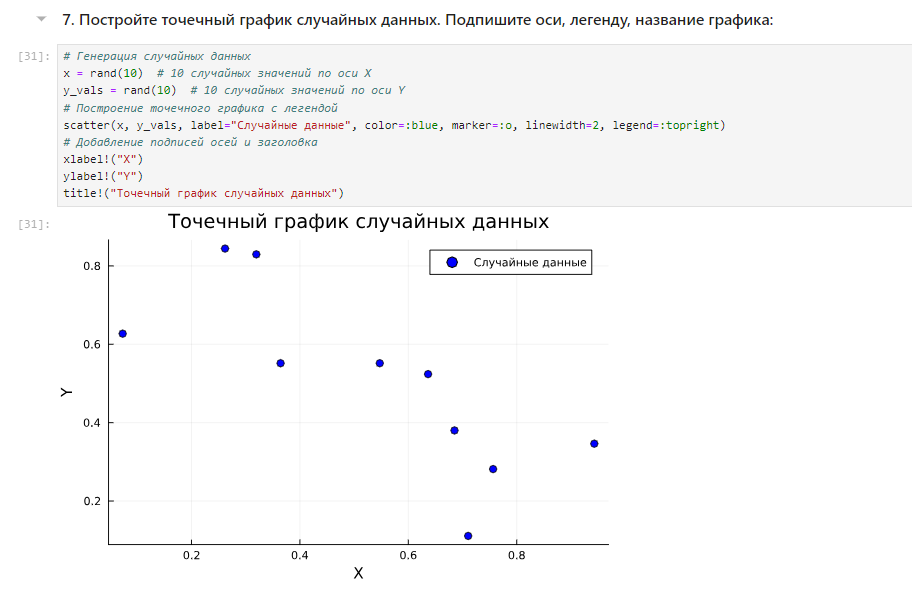
Решение задания №5

Выполнение задания №6:



Решение задания №6

Выполнение задания №7:



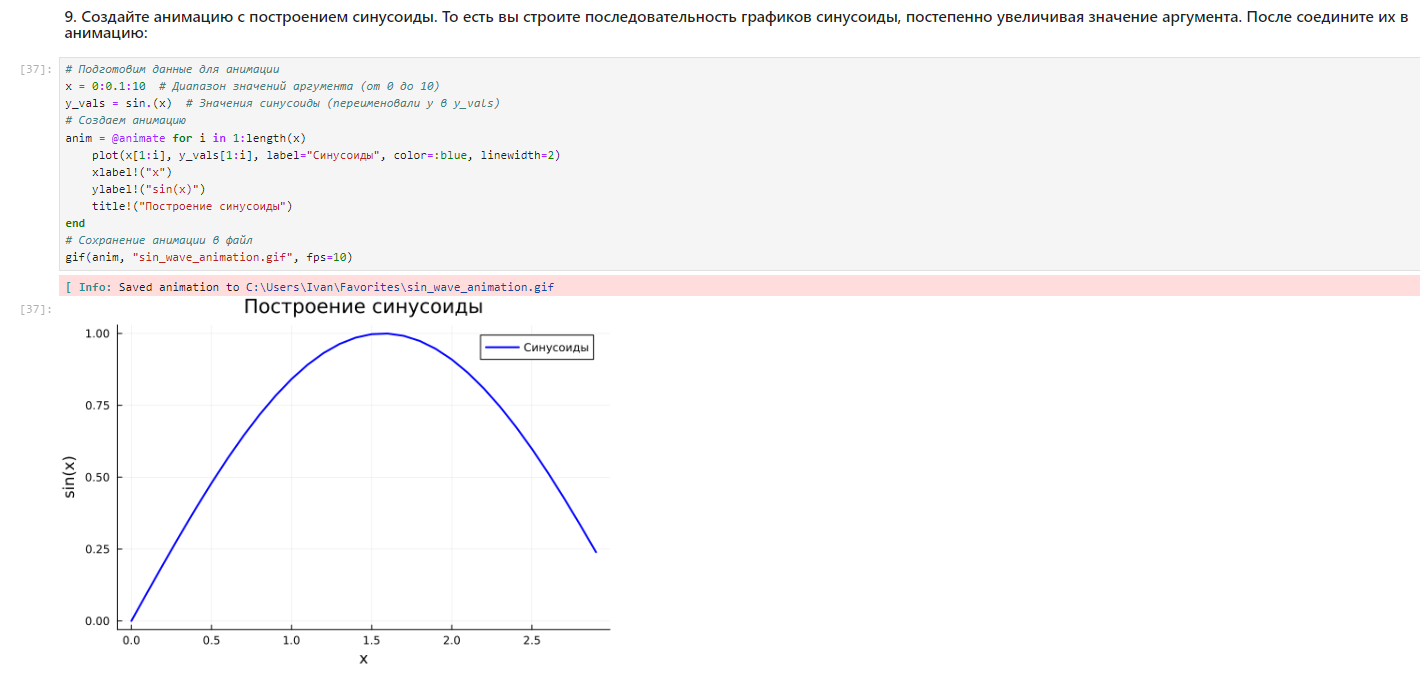
Решение задания №7

Выполнение задания №8:



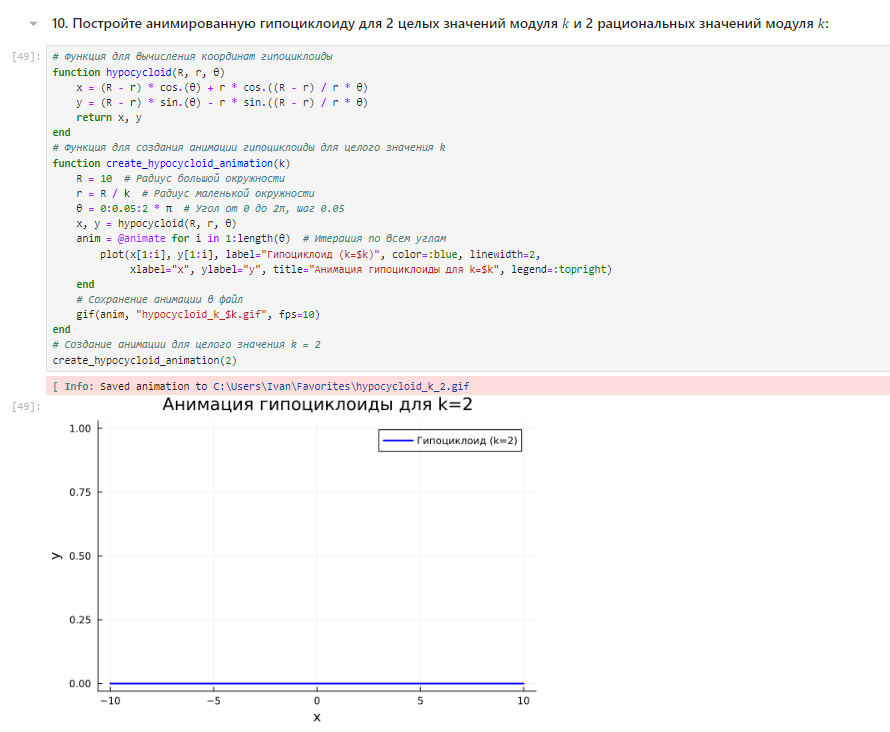
Решение задания №8

Выполнение задания №9:



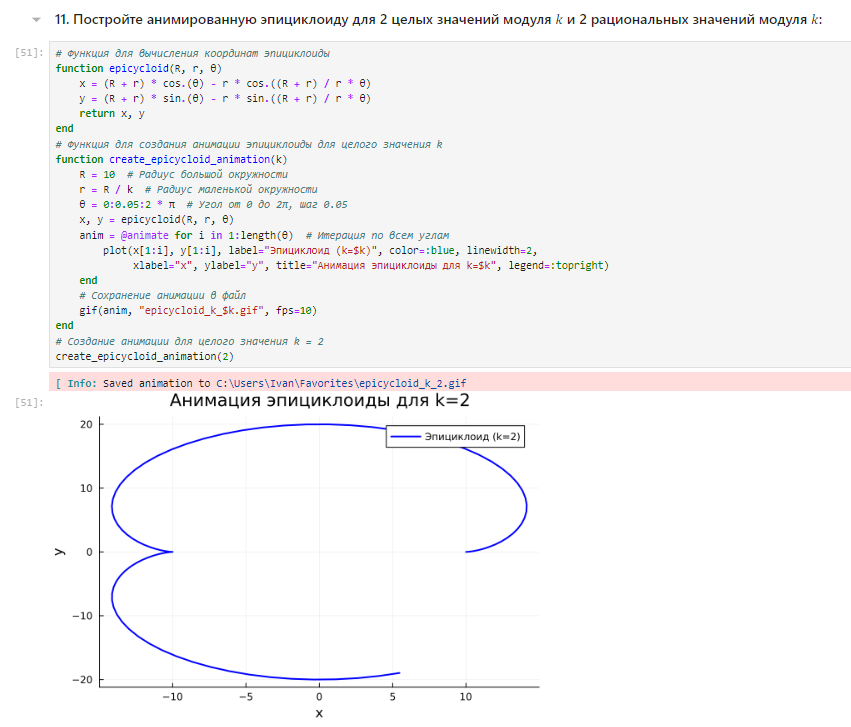
Решение задания №9

Выполнение задания №10:



Решение задания №10

Выполнение задания №11:



Решение задания №11

# Вывод

В ходе выполнения лабораторной работы был освоен синтаксис языка Julia для построения графиков.

# Список литературы. Библиография

[1] Julia Documentation: https://docs.julialang.org/en/v1/