Отчёт по лабораторной работе №5  
Компьютерный практикум по статистическому анализу данных

Построение графиков

Выполнила: Коннова Татьяна Алексеевна,  
НПИбд-01-22, 1132221814

Содержание

# 1 Цель работы

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

# 2 Выполнение лабораторной работы

## 2.1 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 (рис. 1 - (рис. 2):

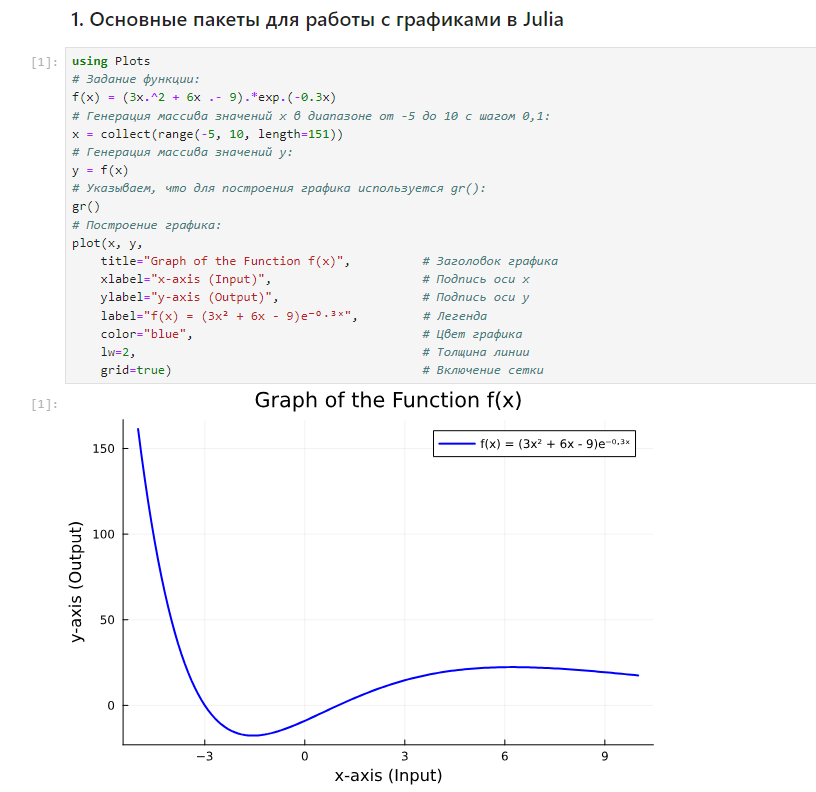


Рис. 1: График функции, построенный при помощи gr()

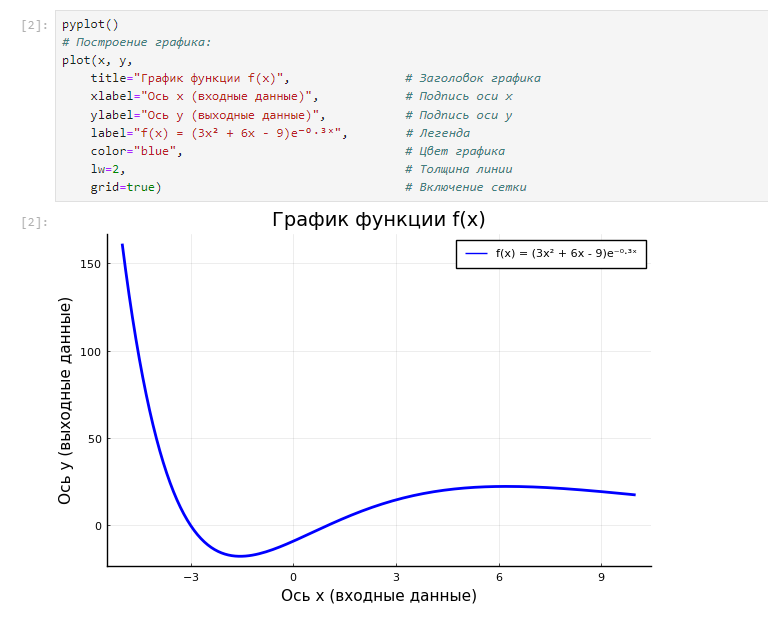


Рис. 2: График функции, построенный при помощи pyplot()

## 2.2 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 (рис. 3 - рис. 5):

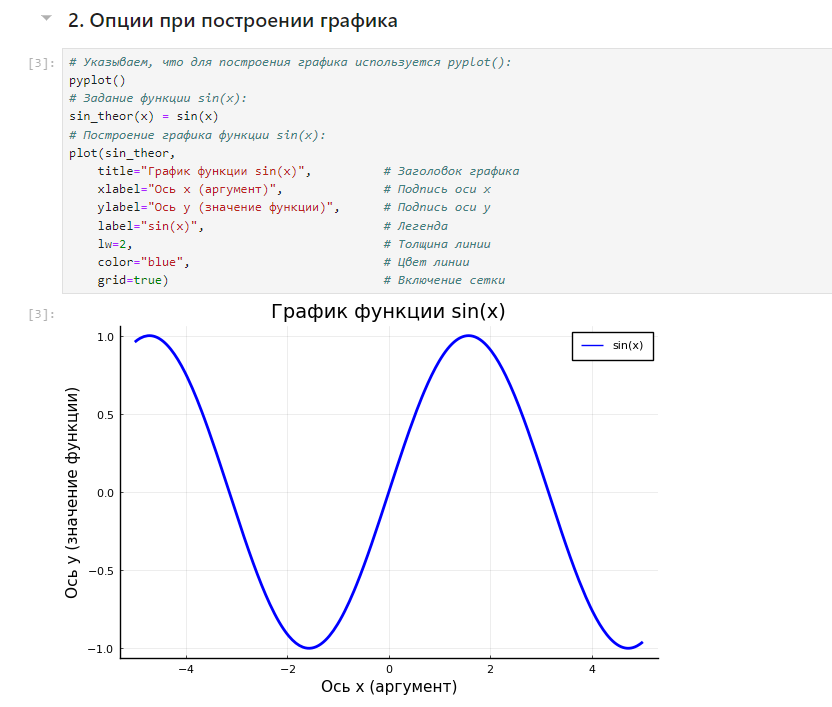


Рис. 3: График функции sin(x)

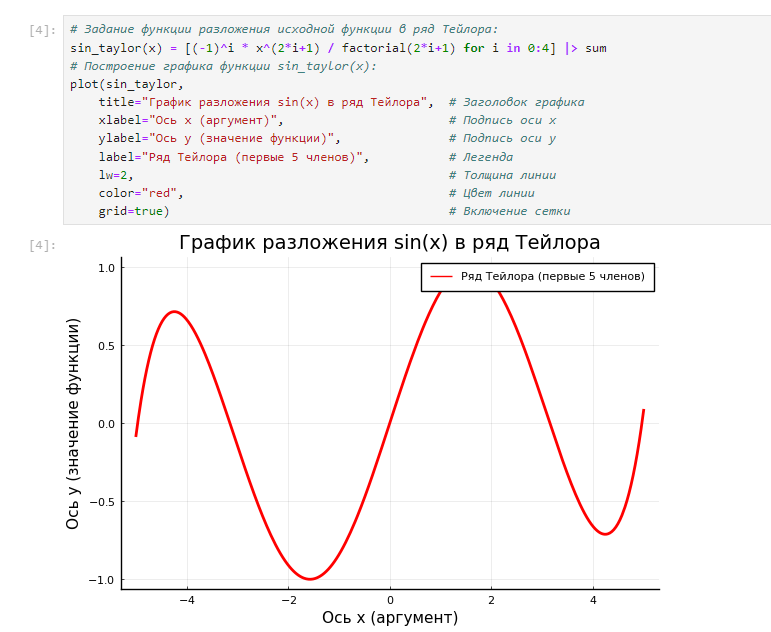


Рис. 4: График функции разложения исходной функции в ряд Тейлора

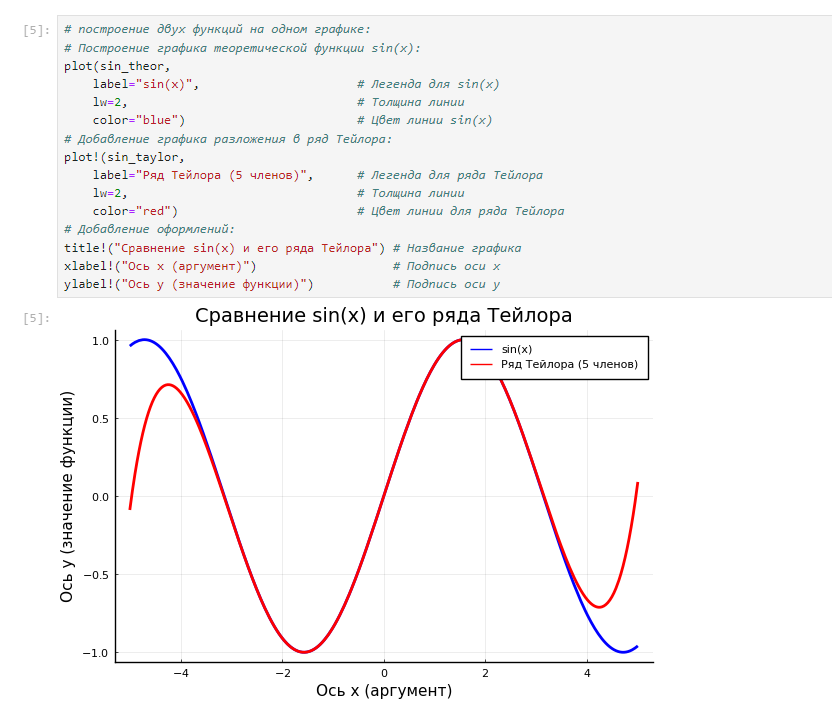


Рис. 5: Графики исходной функции и её разложения в ряд Тейлора

Затем добавим различные опции для отображения на графике (рис. 6):

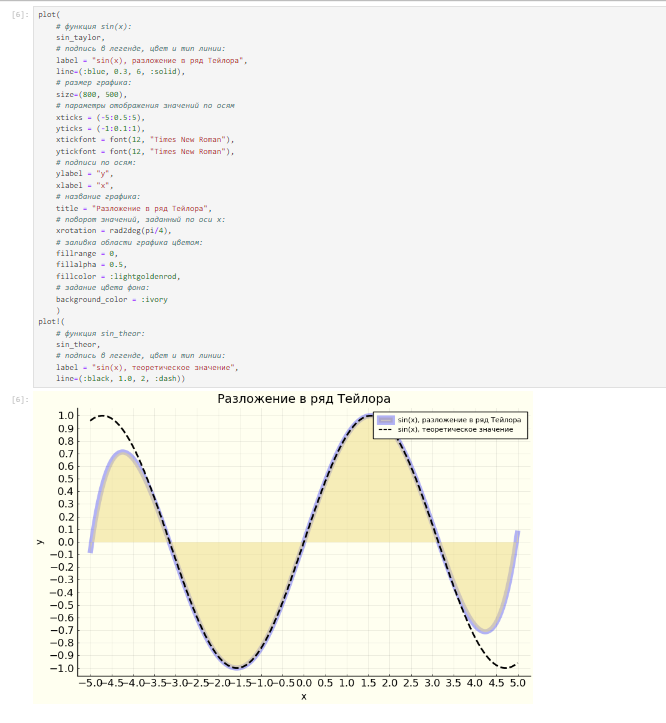


Рис. 6: Вид графиков после добавления опций при их построении

## 2.3 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 (рис. 7):

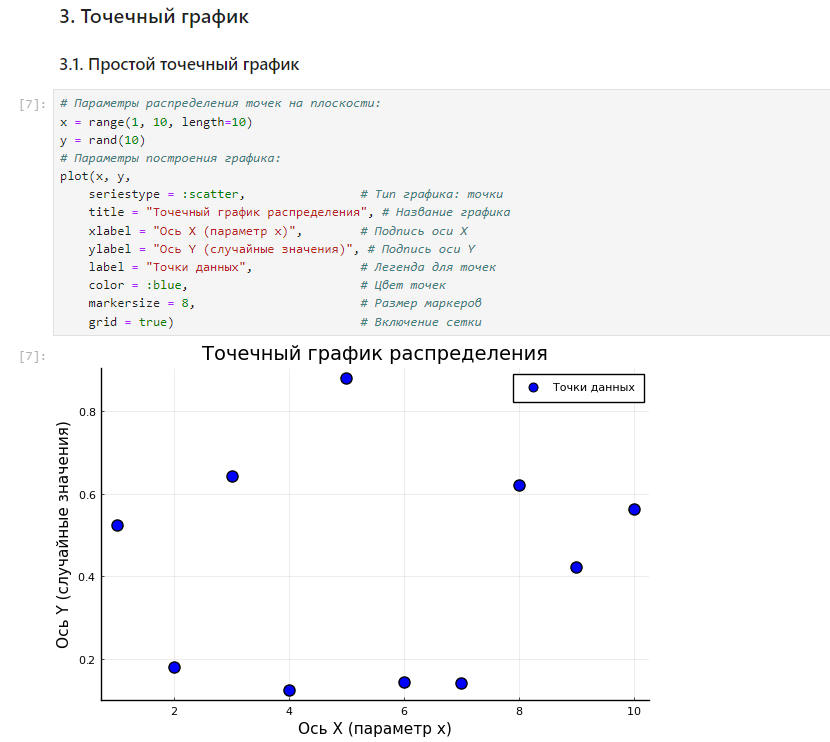


Рис. 7: График десяти случайных значений на плоскости (простой точечный график)

Для точечного графика можно задать различные опции, например размер маркера, его тип, цвет и и т.п. (рис. 8):

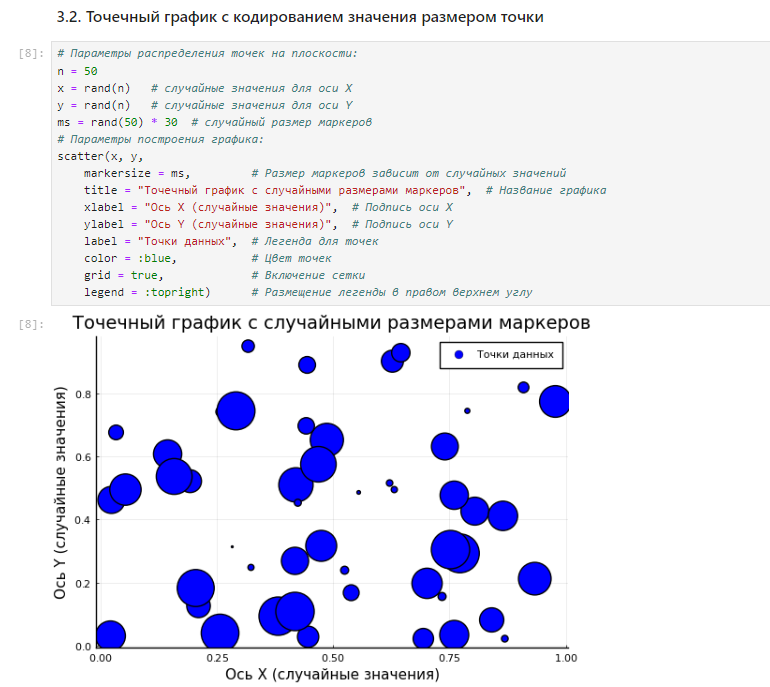


Рис. 8: График пятидесяти случайных значений на плоскости с различными опциями отображения (точечный график с кодированием значения размером точки)

Также можно строить и 3-мерные точечные графики (рис. 9):

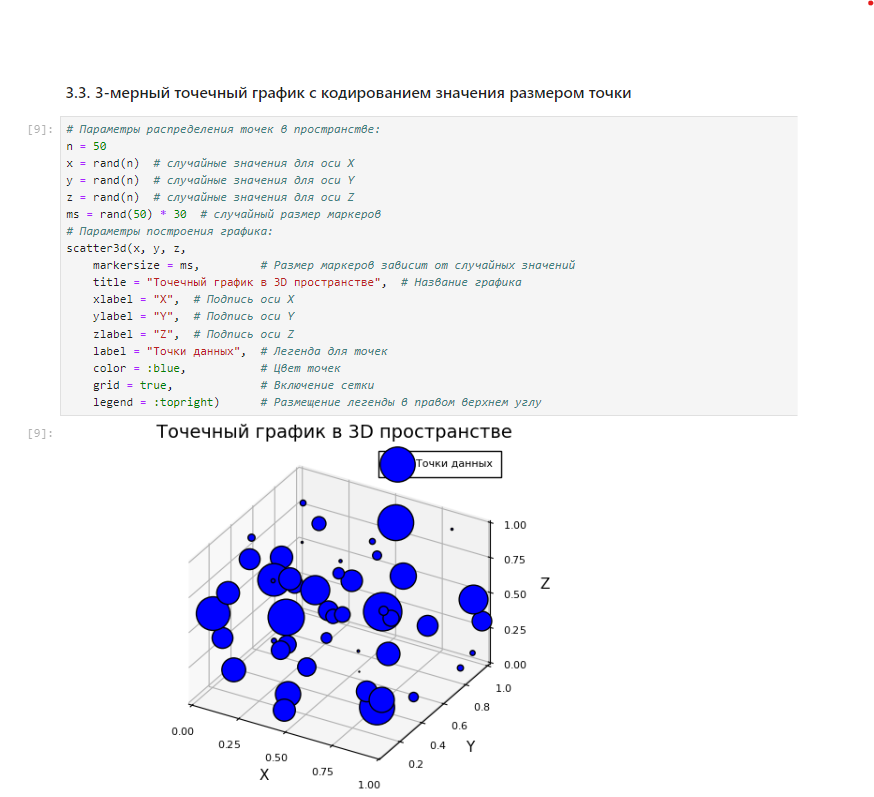


Рис. 9: График пятидесяти случайных значений в пространстве с различными опциями отображения (3-мерный точечный график с кодированием значения размером точки)

## 2.4 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 (рис. 10):

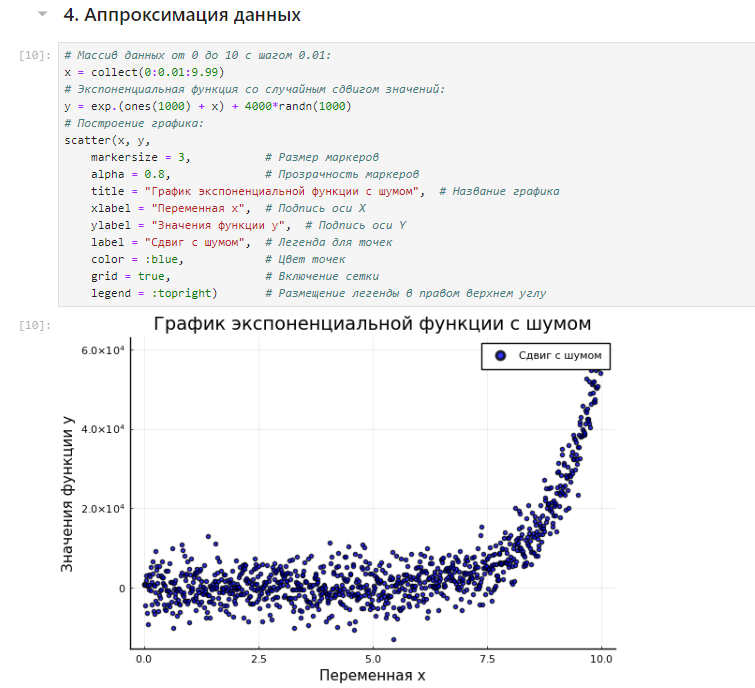


Рис. 10: Пример функции

Аппроксимируем полученную функцию полиномом 5-й степени (рис. 11):

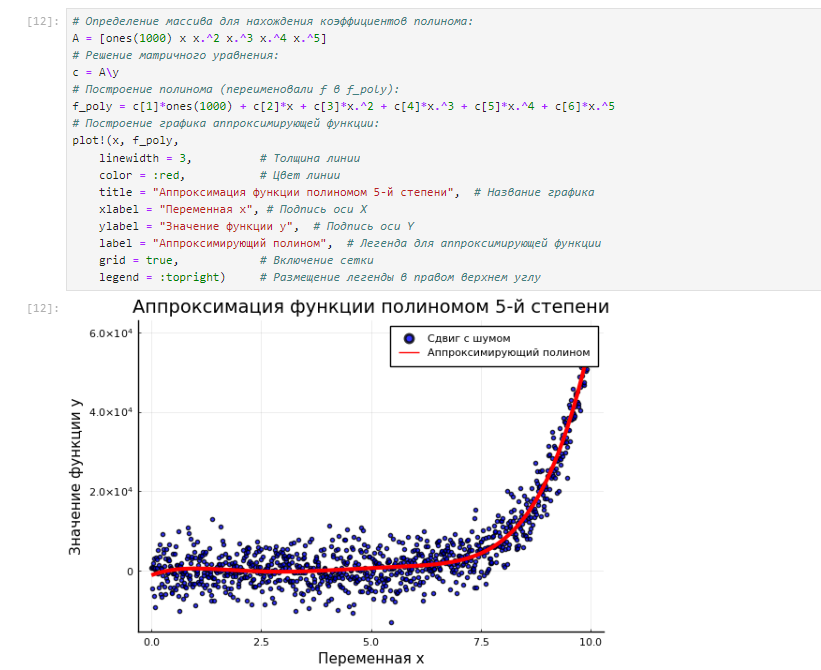


Рис. 11: Пример аппроксимации исходной функции полиномом 5-й степени

## 2.5 Две оси ординат

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

Пример первой траектории (рис. 12):

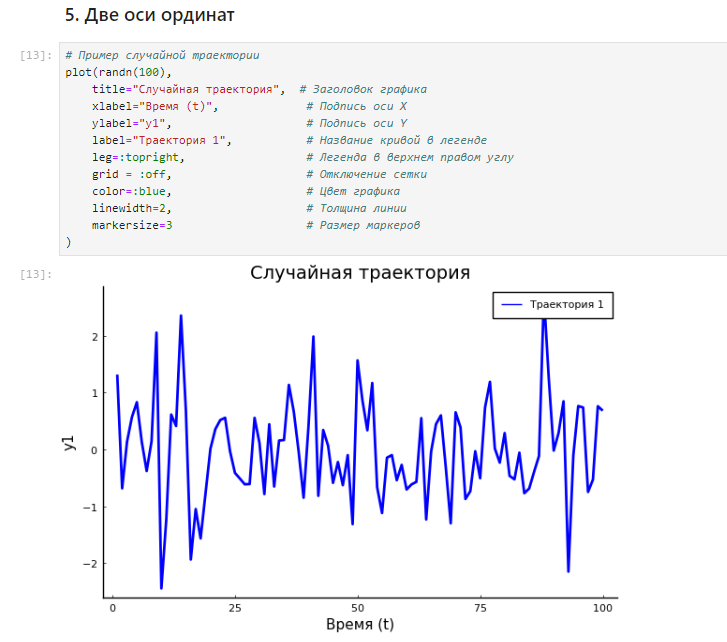


Рис. 12: Пример отдельно построенной траектории

## 2.6 Полярные координаты

Приведём пример построения графика функции в полярных координатах (рис. 13):

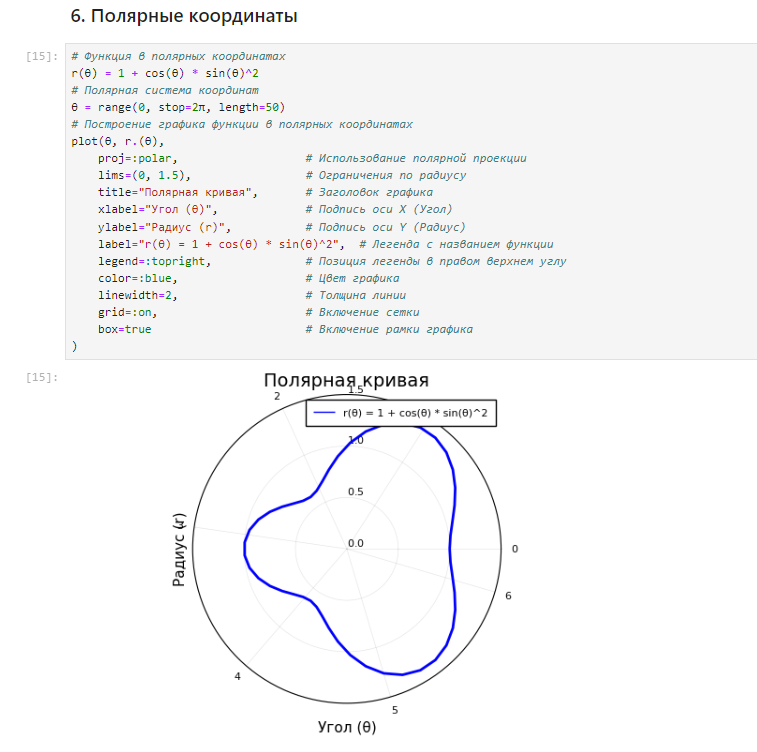


Рис. 13: График функции, заданной в полярных координатах

## 2.7 Параметрический график

Приведём пример построения графика параметрически заданной кривой на плоскости (рис. 14):

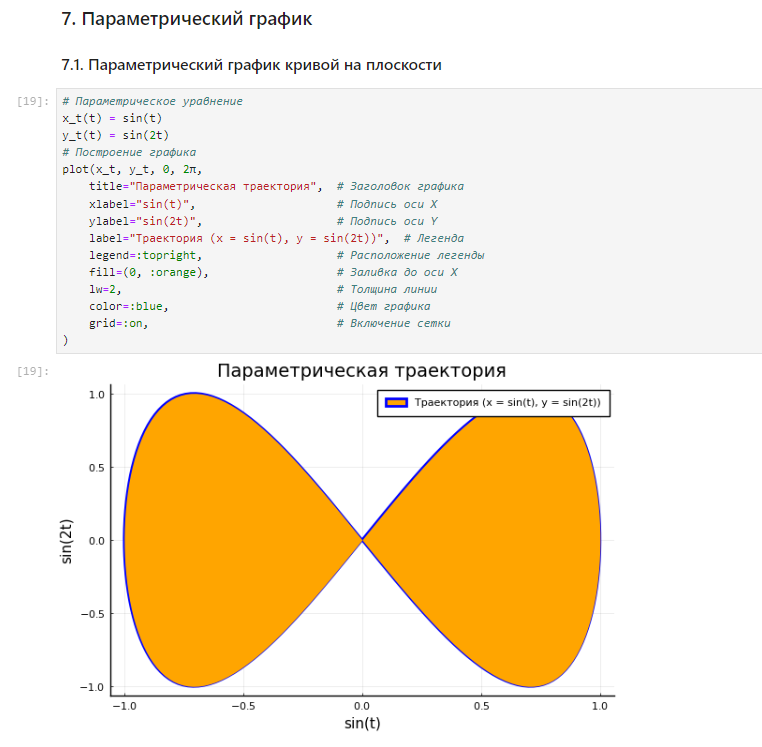


Рис. 14: Параметрический график кривой на плоскости

Далее приведём пример построения графика параметрически заданной кривой в пространстве (рис. 15):

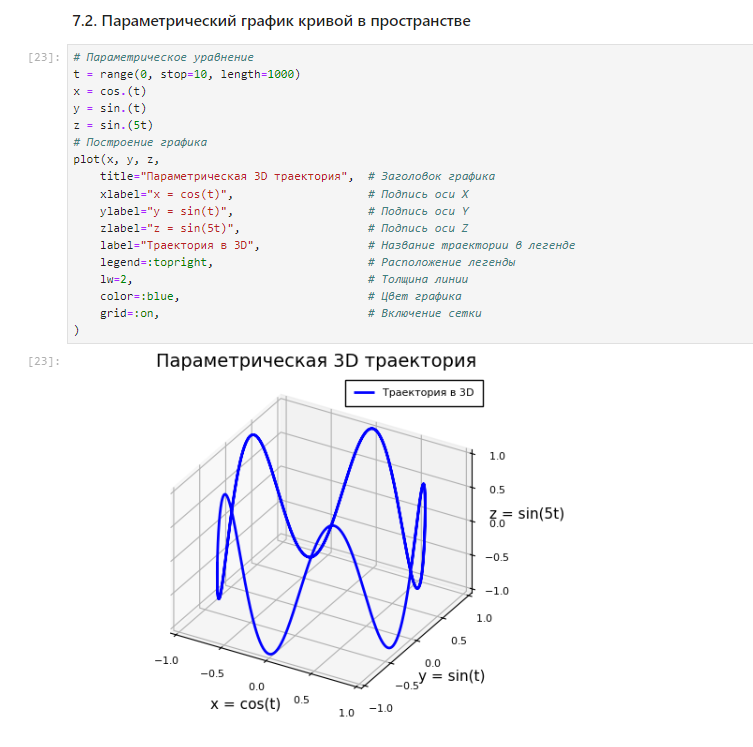


Рис. 15: Параметрический график кривой в пространстве

## 2.8 График поверхности

Для построения поверхности, заданной уравнением f(x, y) = x2 + y2, можно воспользоваться функцией surface() (рис. 16):

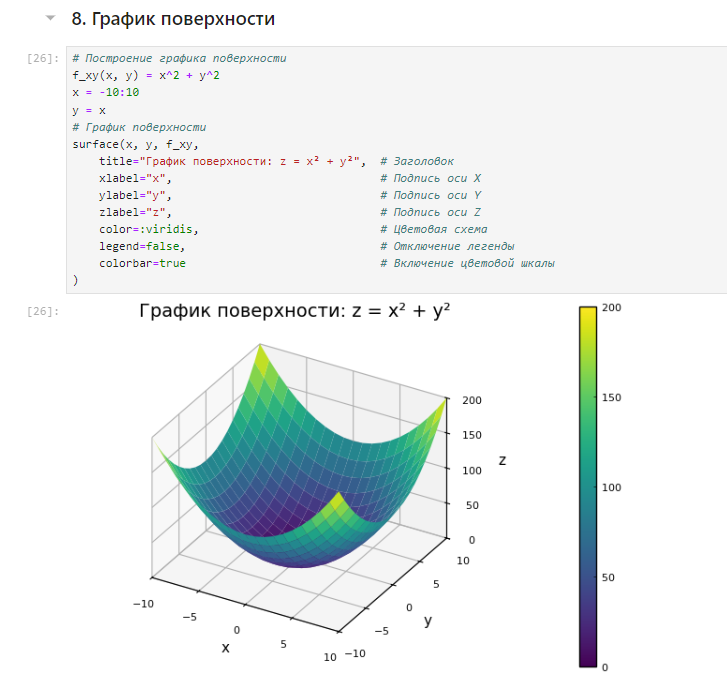


Рис. 16: График поверхности (использована функция surface())

Также можно воспользоваться функцией plot() с заданными параметрами (рис. 17):

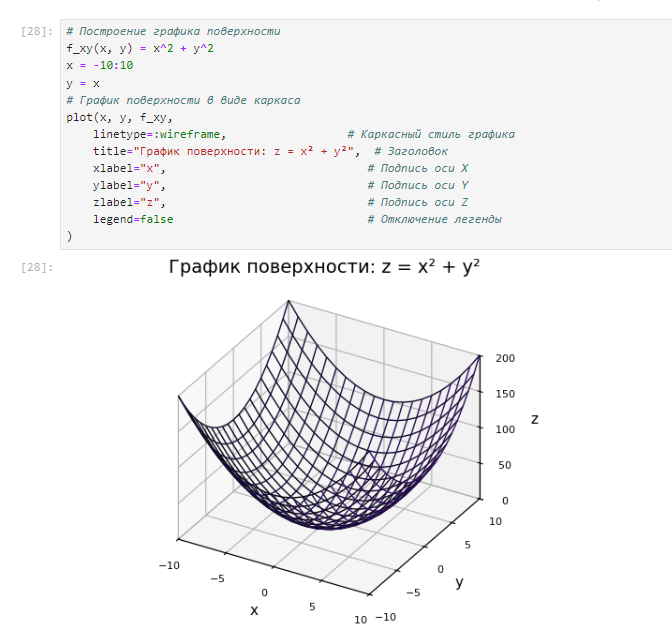


Рис. 17: График поверхности (использована функция plot())

Можно задать параметры сглаживания (рис. 18):

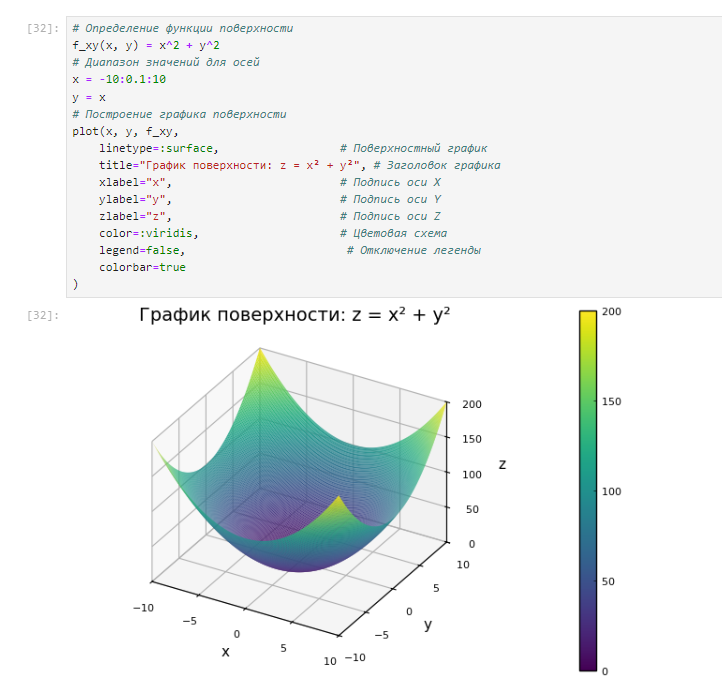


Рис. 18: Сглаженный график поверхности

Можно задать определённый угол зрения (рис. 19):

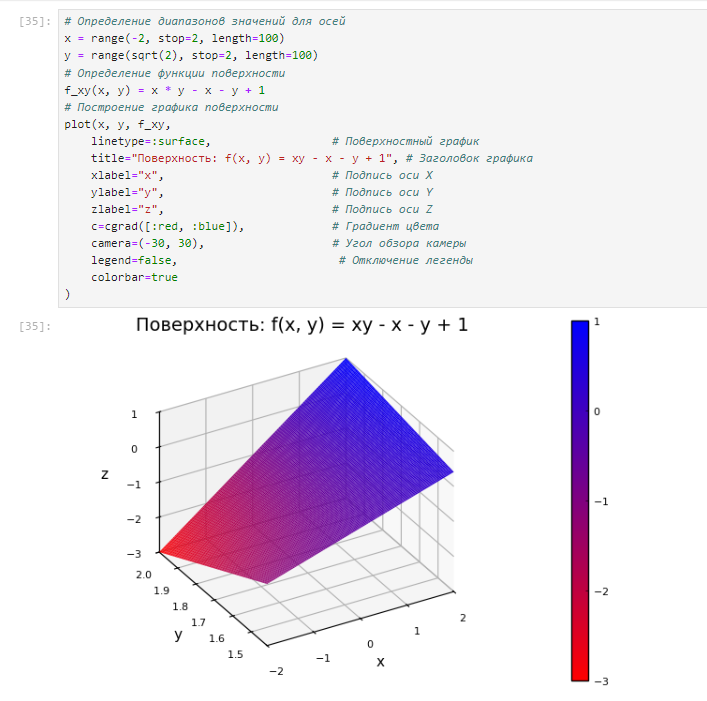


Рис. 19: График поверхности с изменённым углом зрения

## 2.9 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)| (рис. 20):

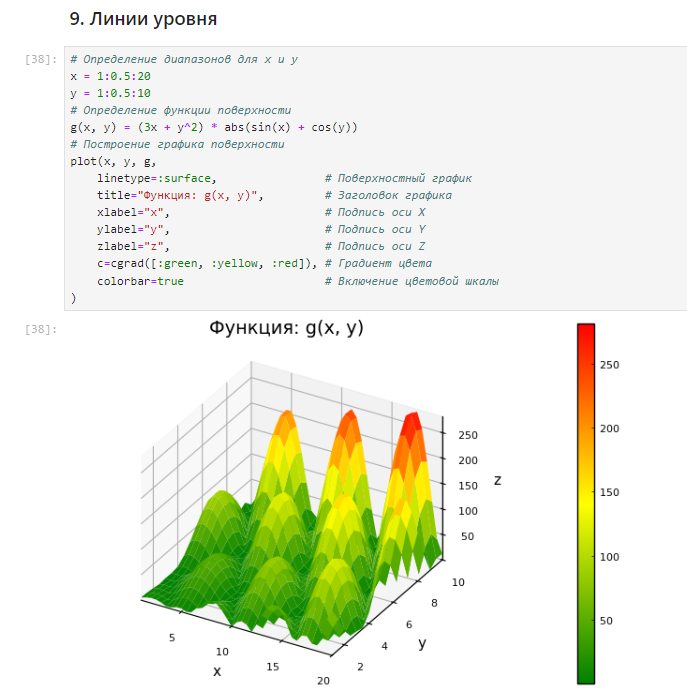


Рис. 20: График поверхности, заданной функцией g(x, y) = (3x + y2)| sin(x) + cos(y)|

Линии уровня можно построить, используя проекцию значений исходной функции на плоскость (рис. 21):

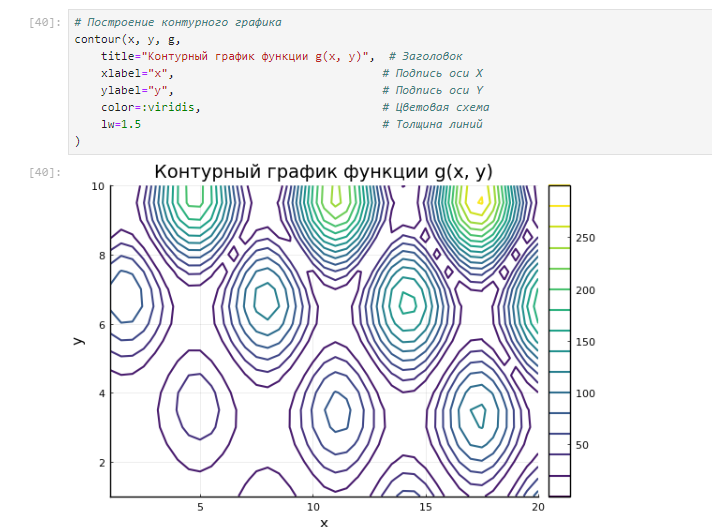


Рис. 21: Линии уровня

Можно дополнительно добавить заливку цветом (рис. 22):

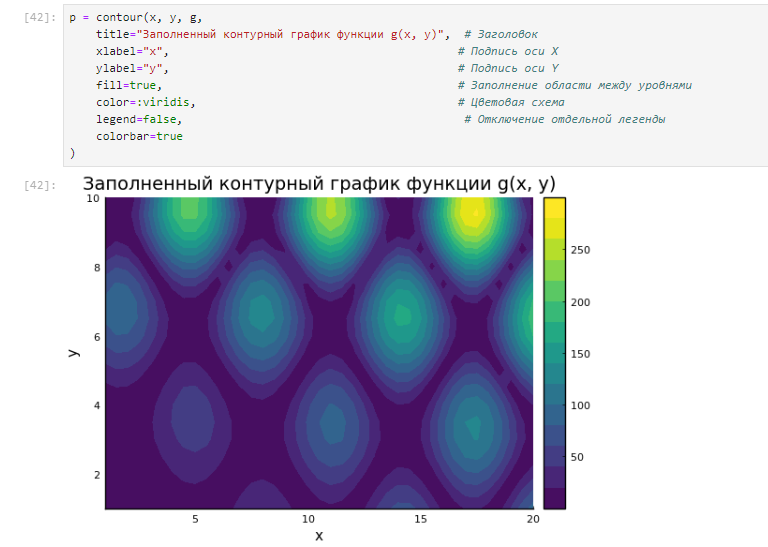


Рис. 22: Линии уровня с заполнением

## 2.10 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 (рис. 23) и линии уровня (рис. 24):

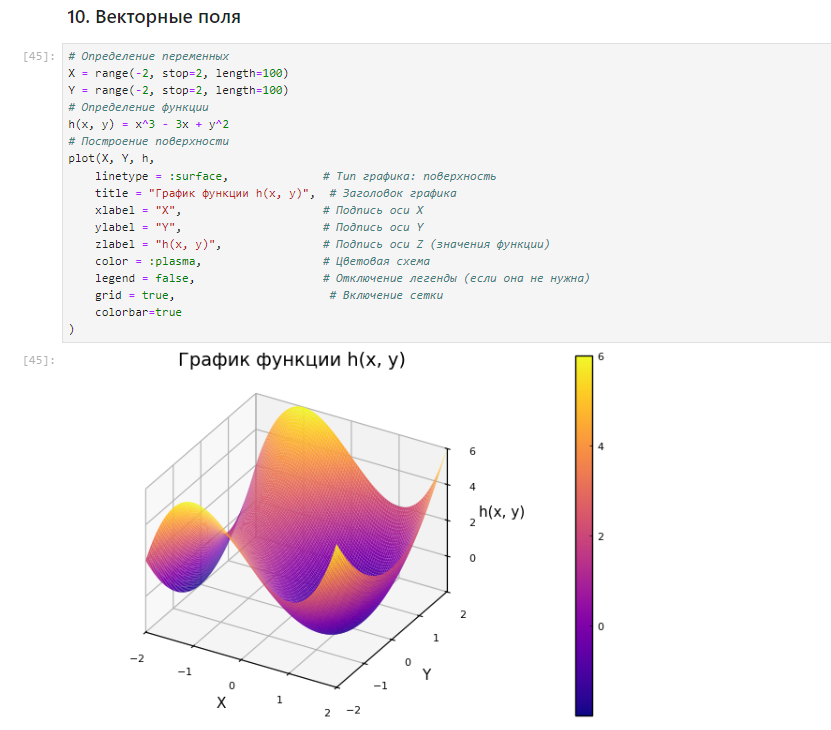


Рис. 23: График функции h(x, y) = x3 − 3x + y2

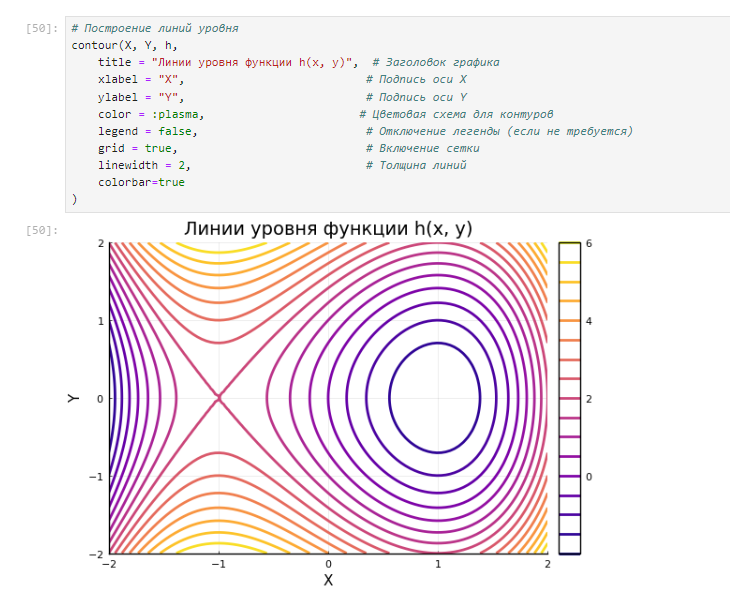


Рис. 24: Линии уровня для функции h(x, y) = x3 − 3x + y2

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.

## 2.11 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 (рис. 25):



Рис. 25: Статичный график поверхности

Добавляем анимацию (рис. 26):

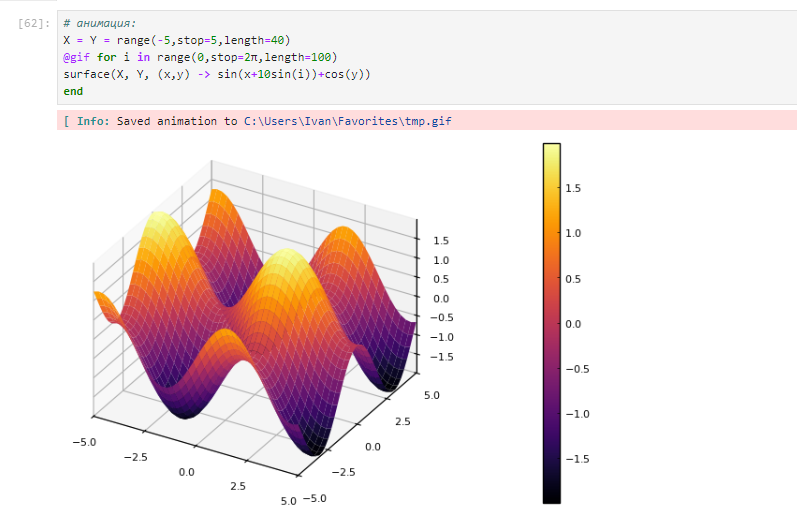


Рис. 26: Анимированный график поверхности

## 2.12 Гипоциклоида

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

Построим большую окружность (рис. 27):

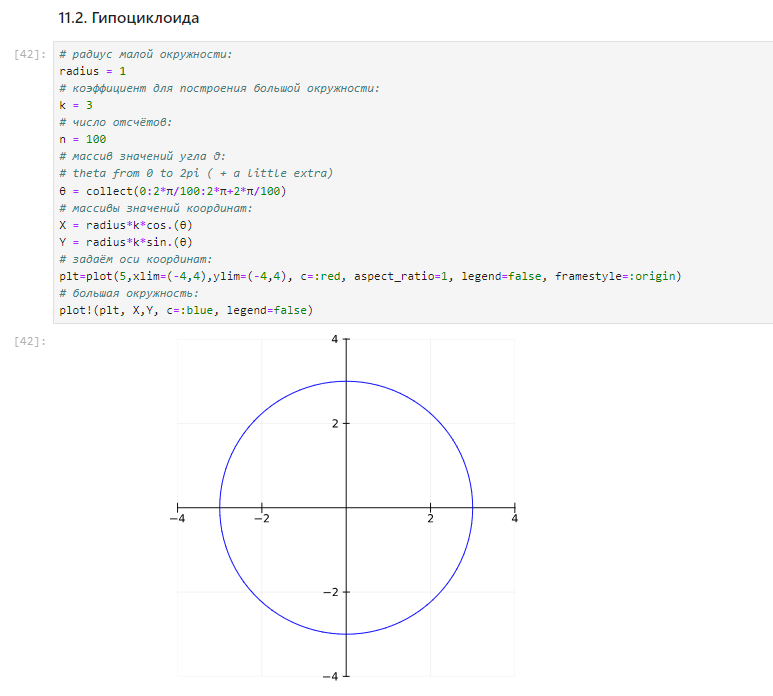


Рис. 27: Большая окружность гипоциклоиды

Для частичного построения гипоциклоиды будем менять параметр t (рис. 28):

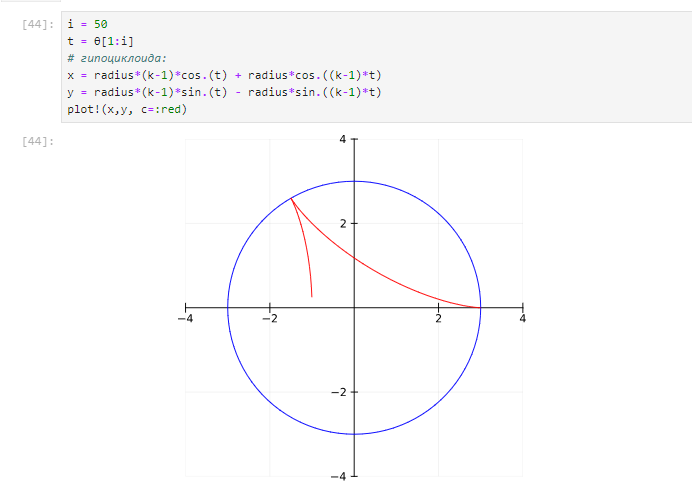


Рис. 28: Половина пути гипоциклоиды

Добавляем малую окружность гипоциклоиды (рис. 29):

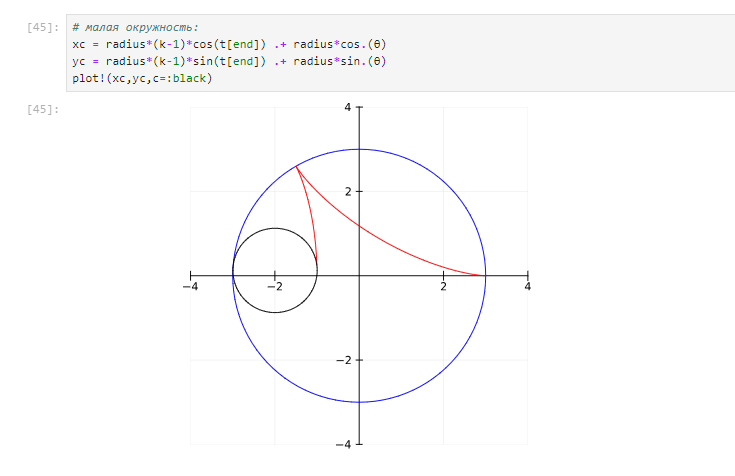


Рис. 29: Малая окружность гипоциклоиды

Добавим радиус для малой окружности (рис. 30):

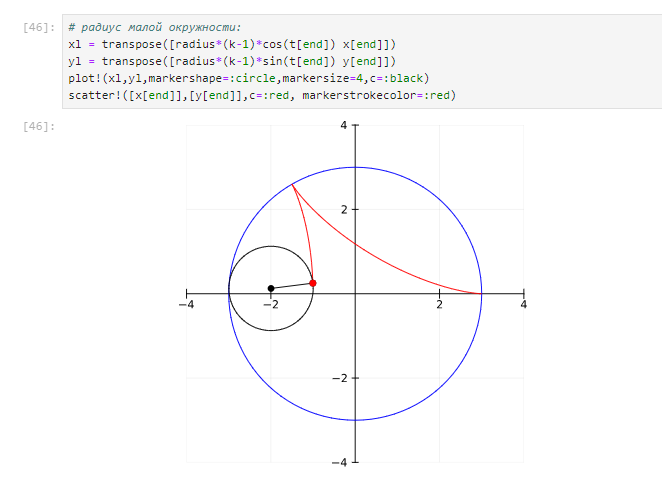


Рис. 30: Малая окружность гипоциклоиды с добавлением радиуса

В конце сделаем анимацию получившегося изображения (рис. 31):

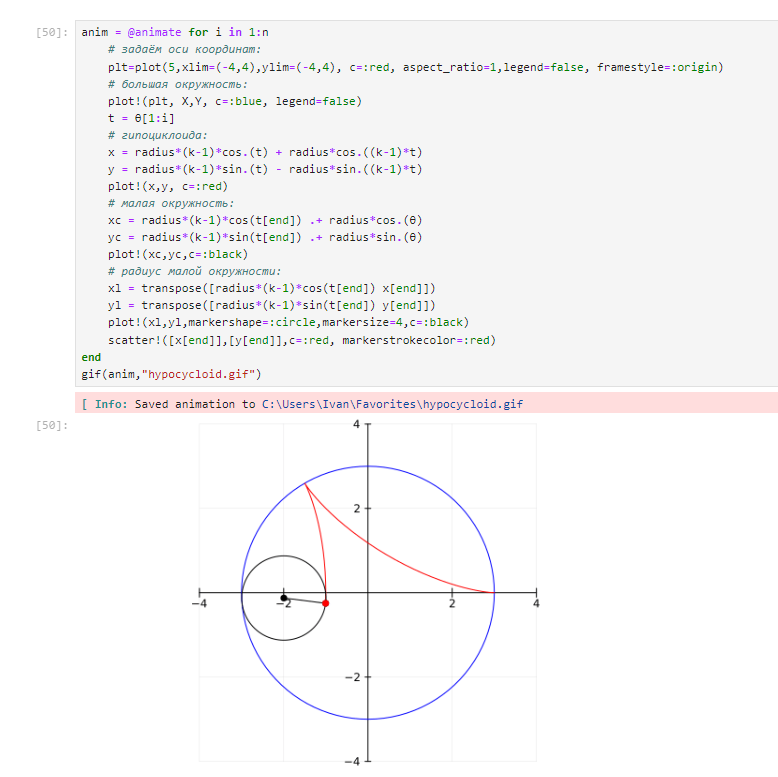


Рис. 31: Малая окружность гипоциклоиды с добавлением радиуса

## 2.13 Errorbars

В исследованиях часто требуется изобразить графики погрешностей измерения.

Построим график исходных значений (рис. 32):

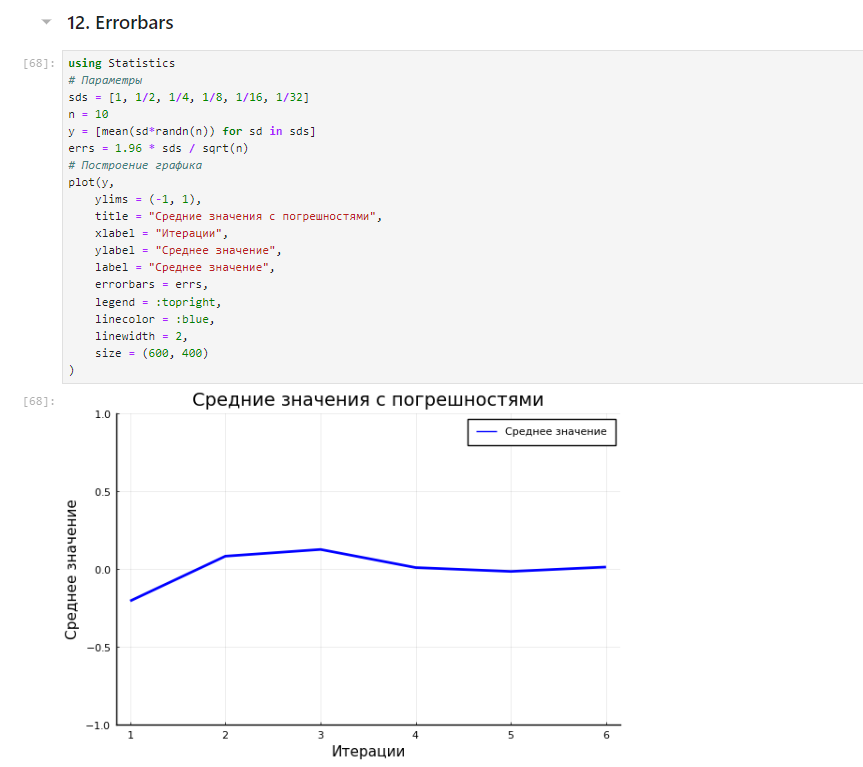


Рис. 32: График исходных значений

Построим график отклонений от исходных значений (рис. 33):

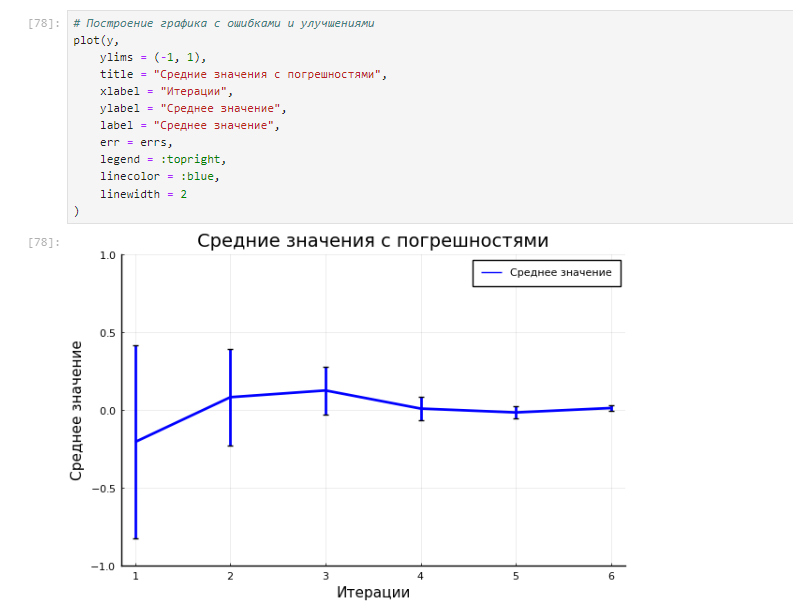


Рис. 33: График исходных значений с отклонениями

Повернём график (рис. 34):

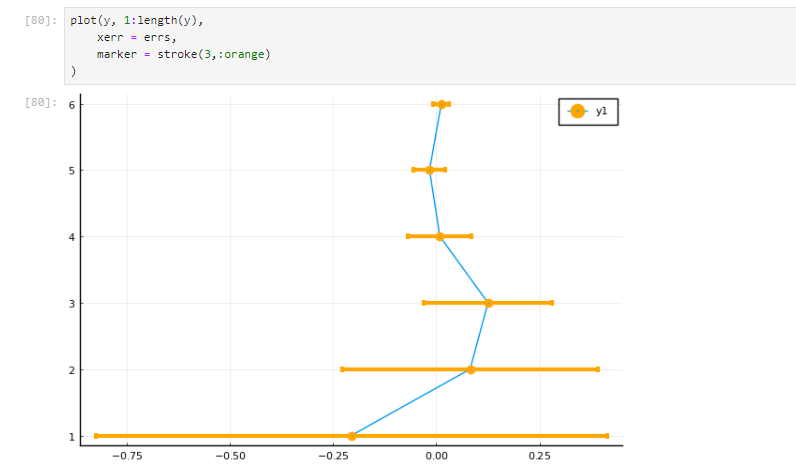


Рис. 34: Поворот графика

Заполним область цветом (рис. 35):

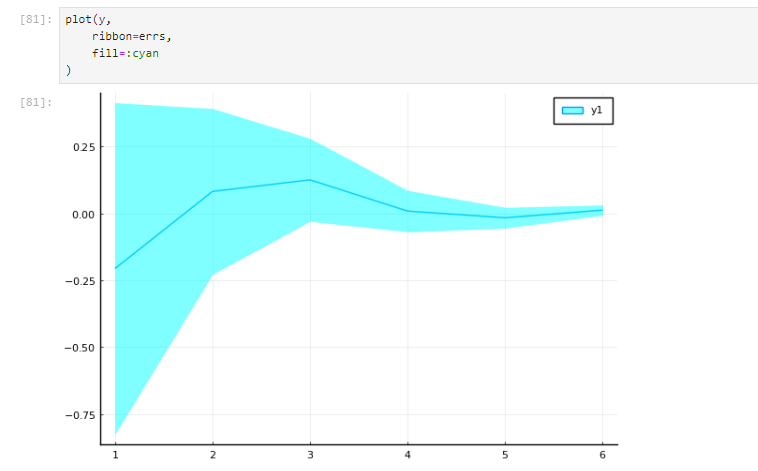


Рис. 35: Заполнение цветом

Можно построить график ошибок по двум осям (рис. 36):

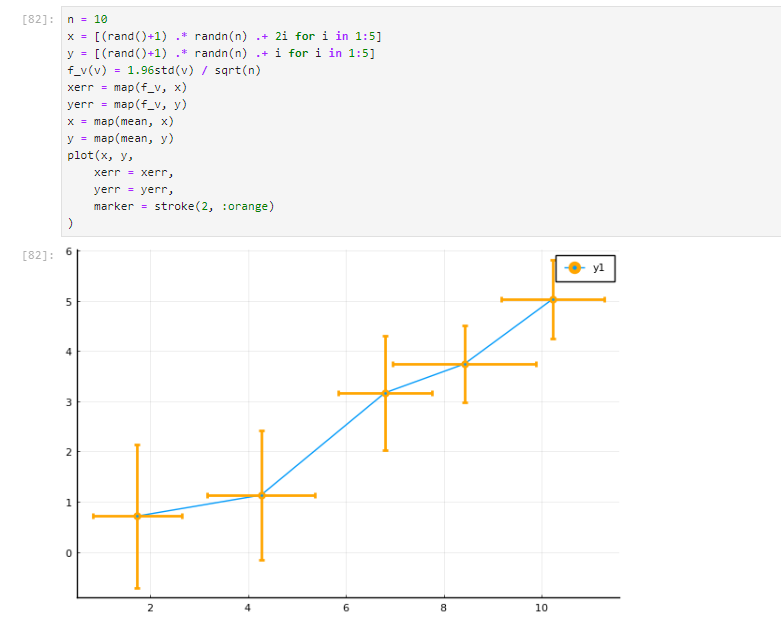


Рис. 36: График ошибок по двум осям

Можно построить график асимметричных ошибок по двум осям (рис. 37):

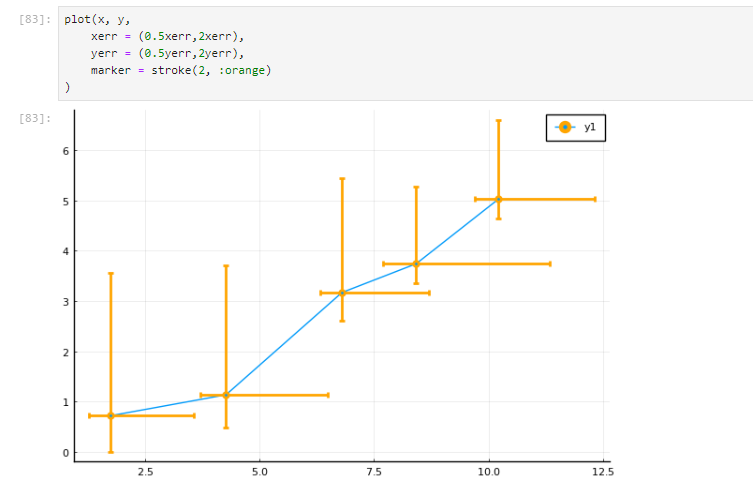


Рис. 37: График асимметричных ошибок по двум осям

## 2.14 Использование пакета Distributions

Строим гистограмму (рис. 38):

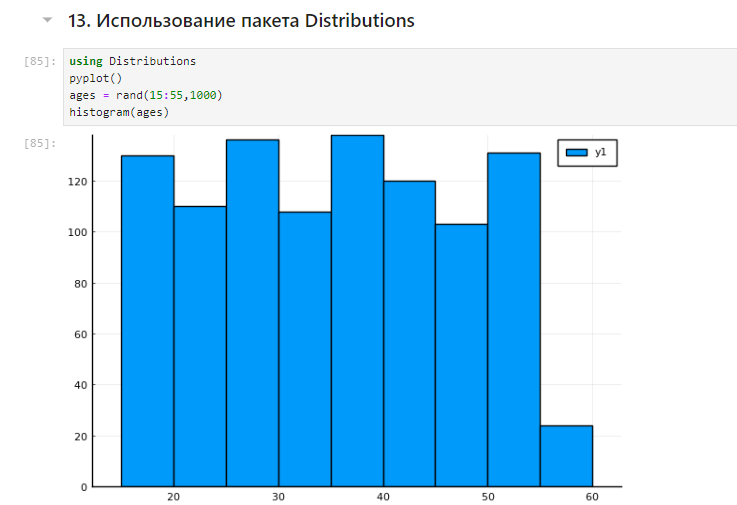


Рис. 38: Гистограмма, построенная по массиву случайных чисел

Задаём нормальное распределение и строим гистограмму (рис. 39):

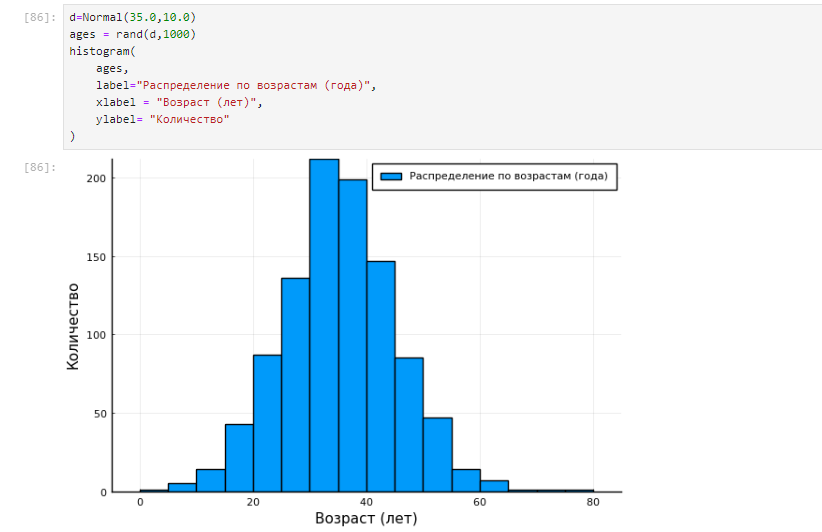


Рис. 39: Гистограмма нормального распределения

Далее применим для построения нескольких гистограмм распределения людей по возрастам на одном графике plotly() (рис. 40):

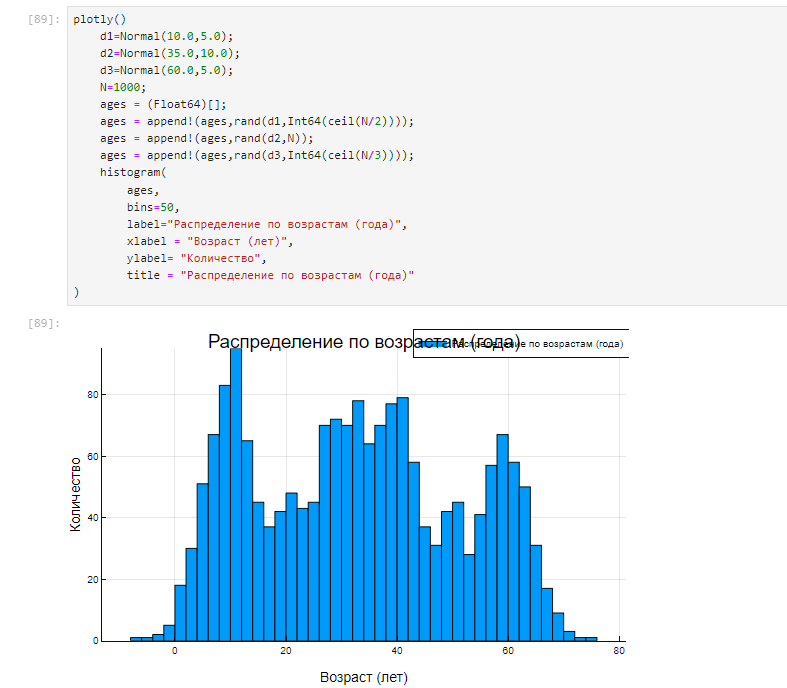


Рис. 40: Гистограмма распределения людей по возрастам

## 2.15 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 (рис. 41):

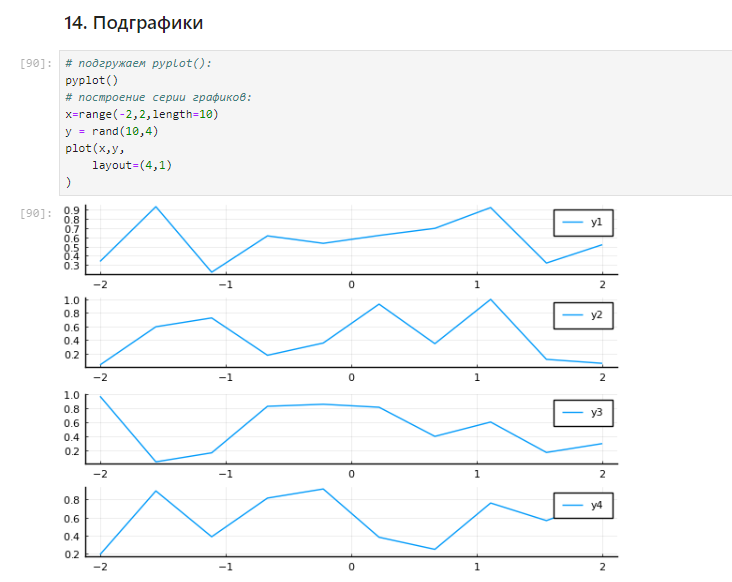


Рис. 41: Серия из 4-х графиков в ряд

Для автоматического вычисления сетки необходимо передать layout целое число (рис. 42):

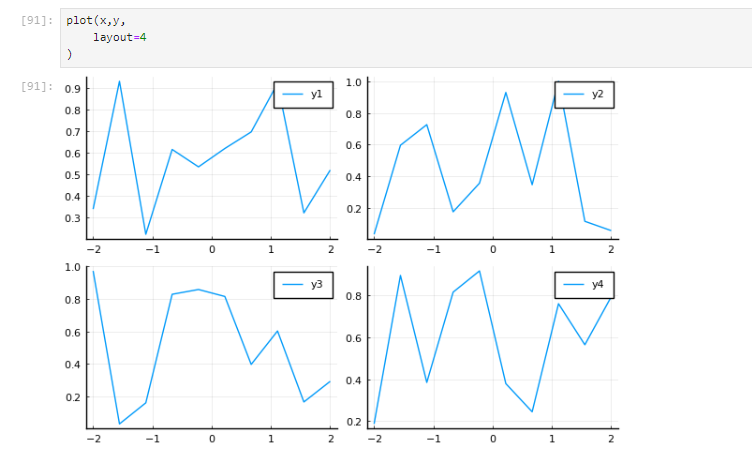


Рис. 42: Серия из 4-х графиков в сетке

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 (рис. 43):



Рис. 43: 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 (рис. 44):

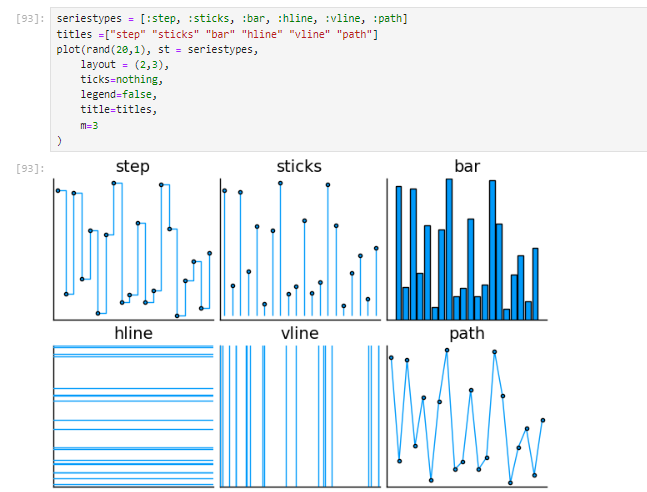


Рис. 44: Разнообразные варианты представления данных

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 (рис. 45):

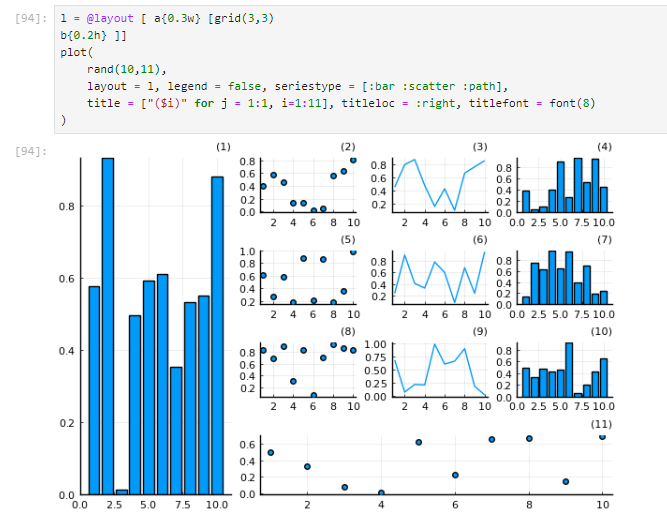


Рис. 45: Демонстрация применения сложного макета для построения графиков

## 2.16 Самостоятельное выполнение

Выполнение задания №1 (рис. 46):

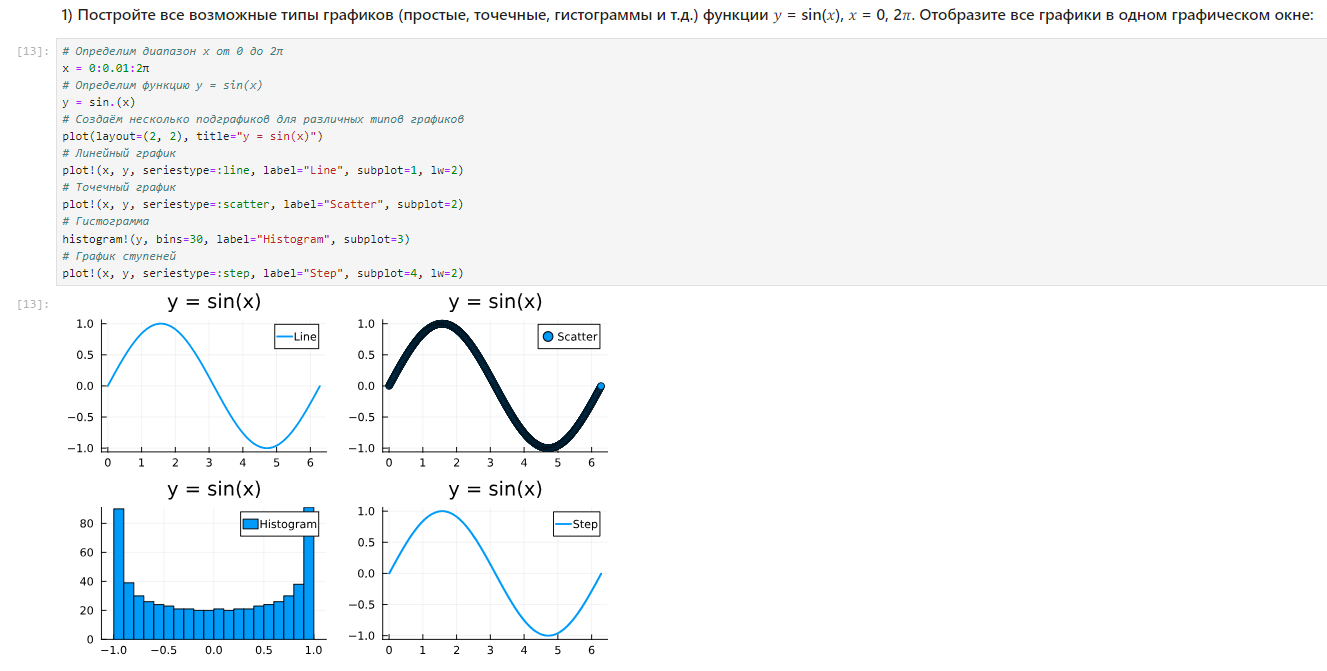


Рис. 46: Решение задания №1

Выполнение задания №2 (рис. 47):

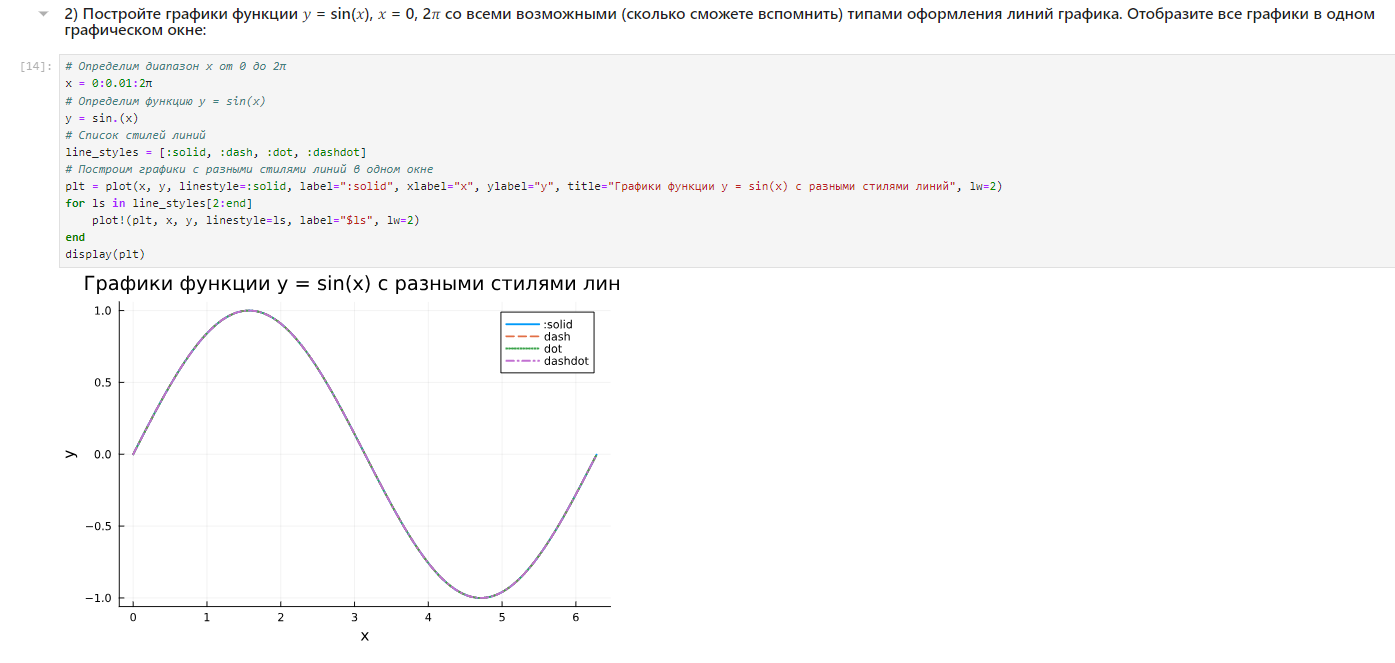


Рис. 47: Решение задания №2

Выполнение задания №3 (рис. 48):

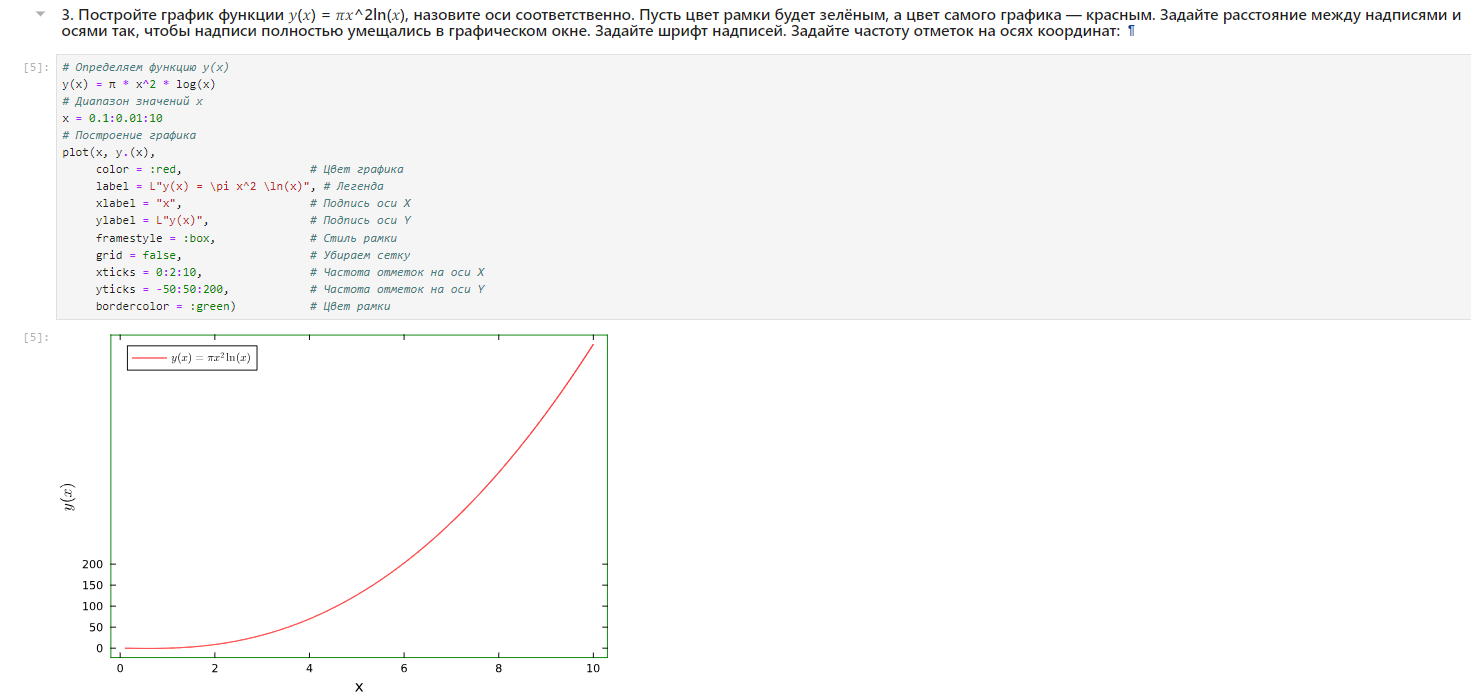


Рис. 48: Решение задания №3

Выполнение задания №4 (рис. 49):

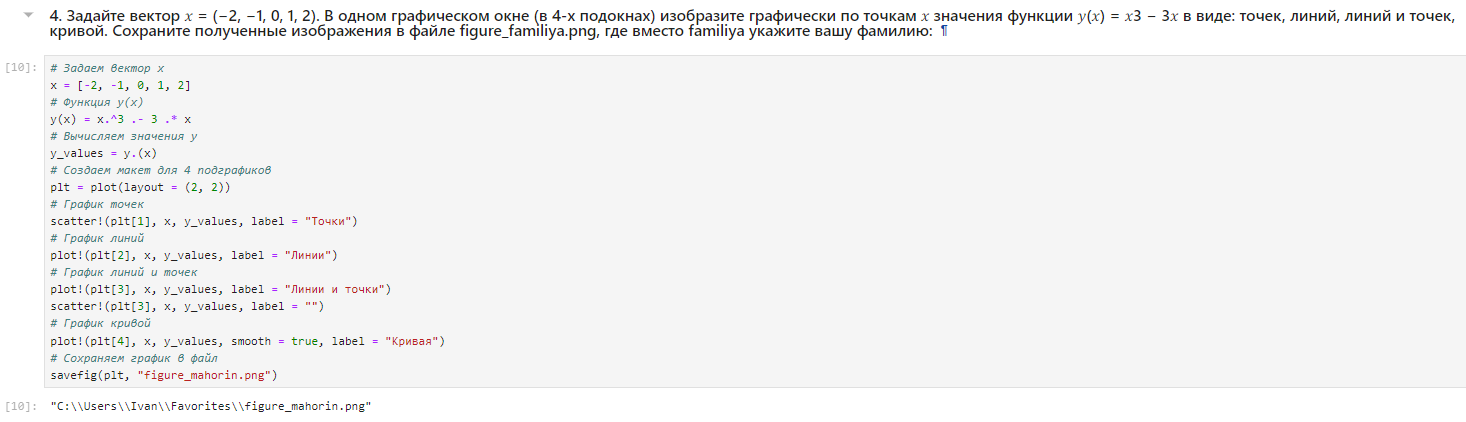


Рис. 49: Решение задания №4

Выполнение задания №5 (рис. 50 - рис. 51):

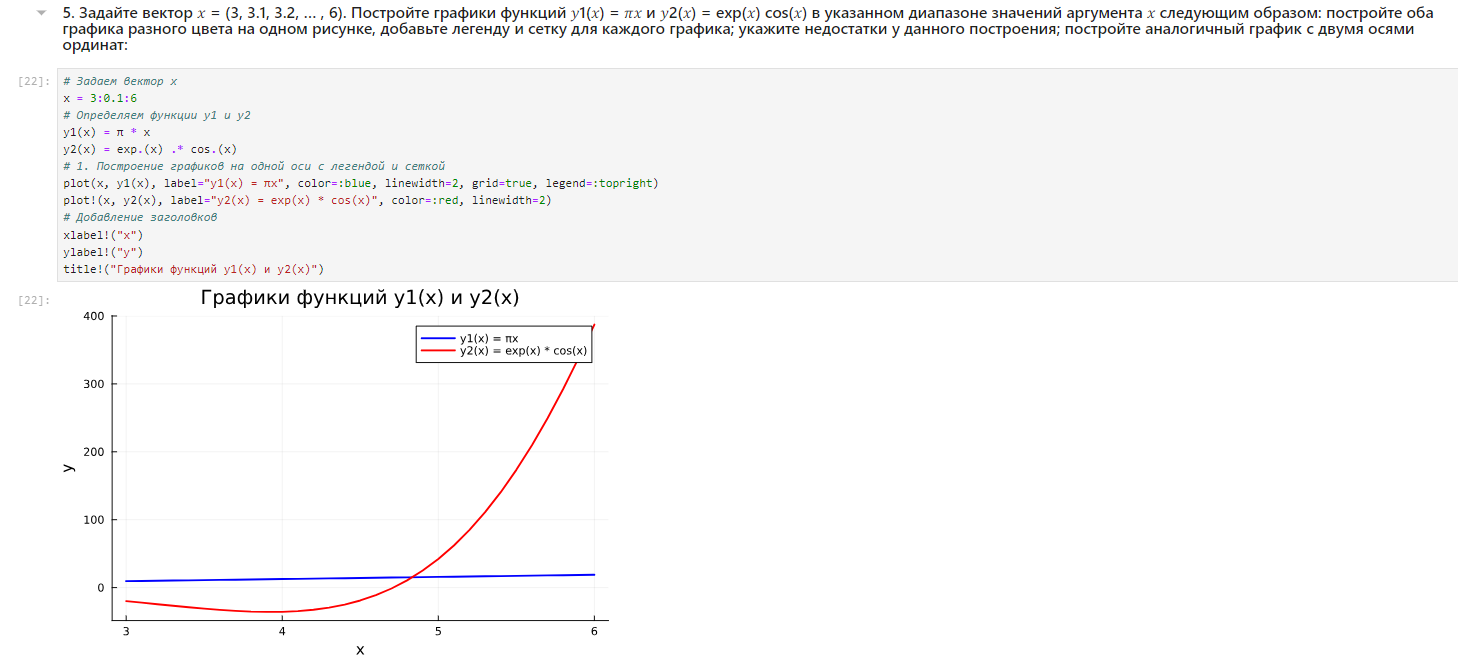


Рис. 50: Решение задания №5

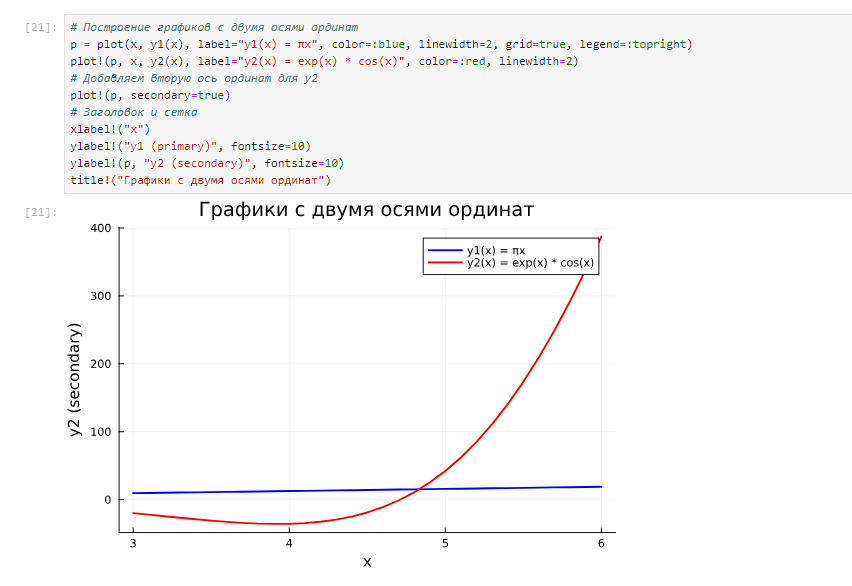


Рис. 51: Решение задания №5

Выполнение задания №6 (рис. 52):

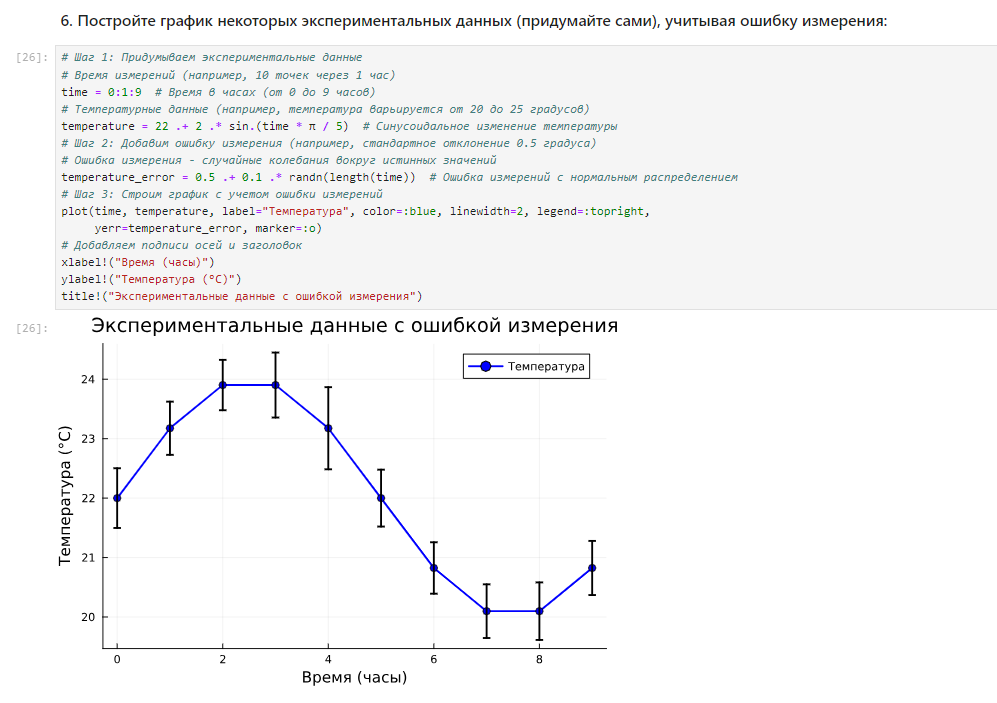


Рис. 52: Решение задания №6

Выполнение задания №7 (рис. 53):

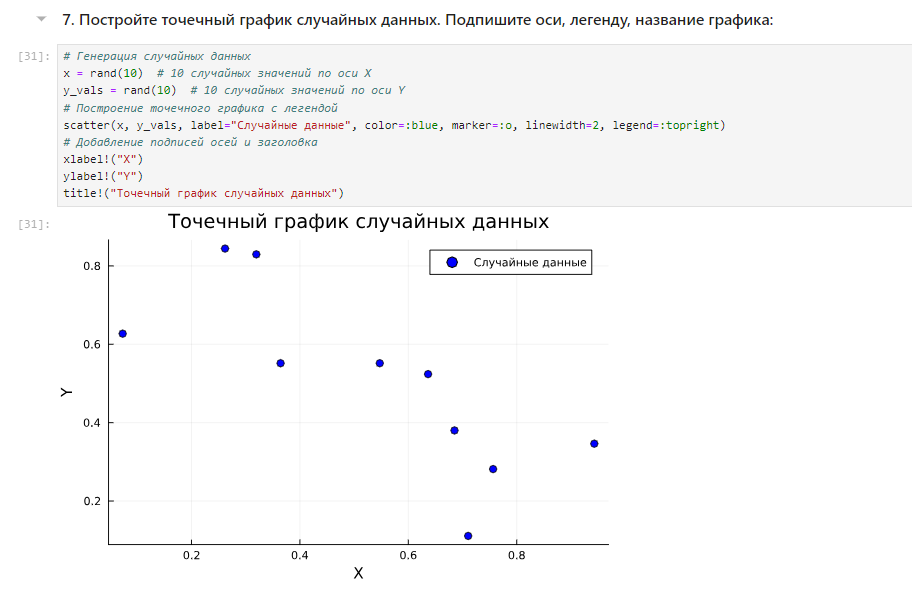


Рис. 53: Решение задания №7

Выполнение задания №8 (рис. 54):



Рис. 54: Решение задания №8

Выполнение задания №9 (рис. 55):

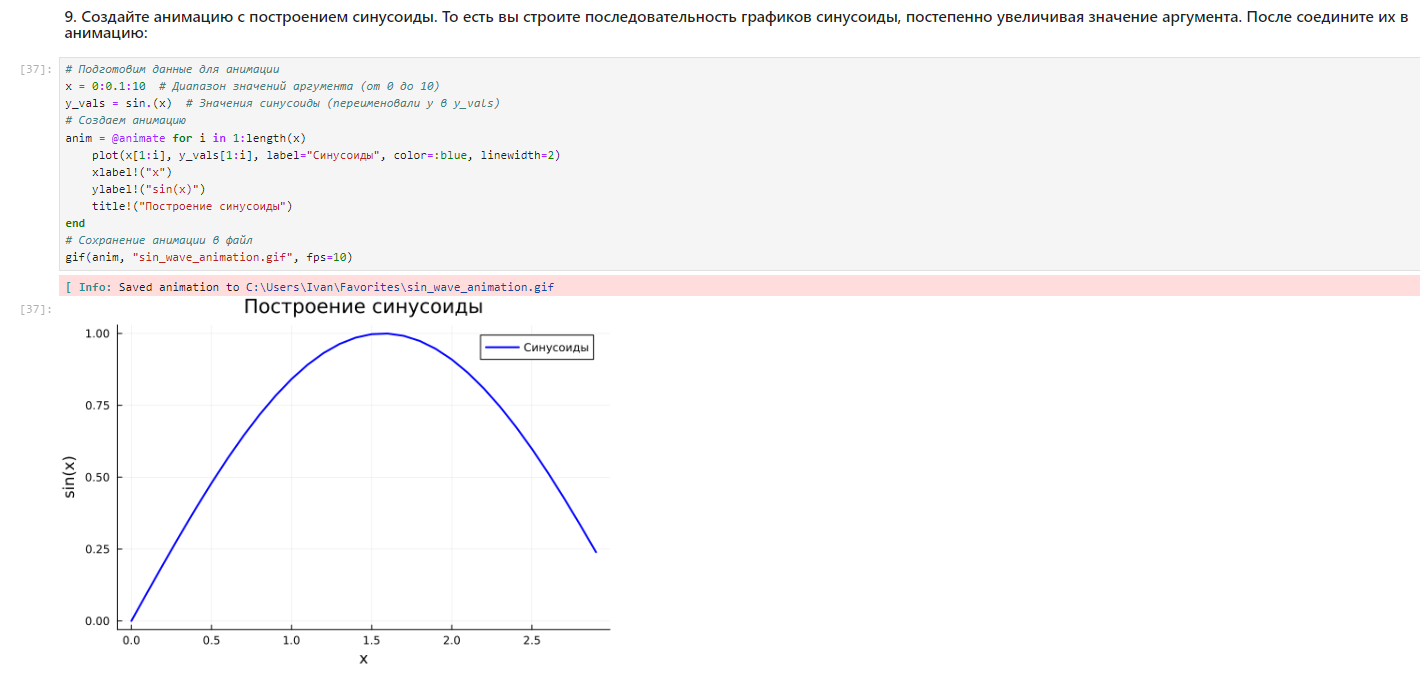


Рис. 55: Решение задания №9

Выполнение задания №10 (рис. 56):

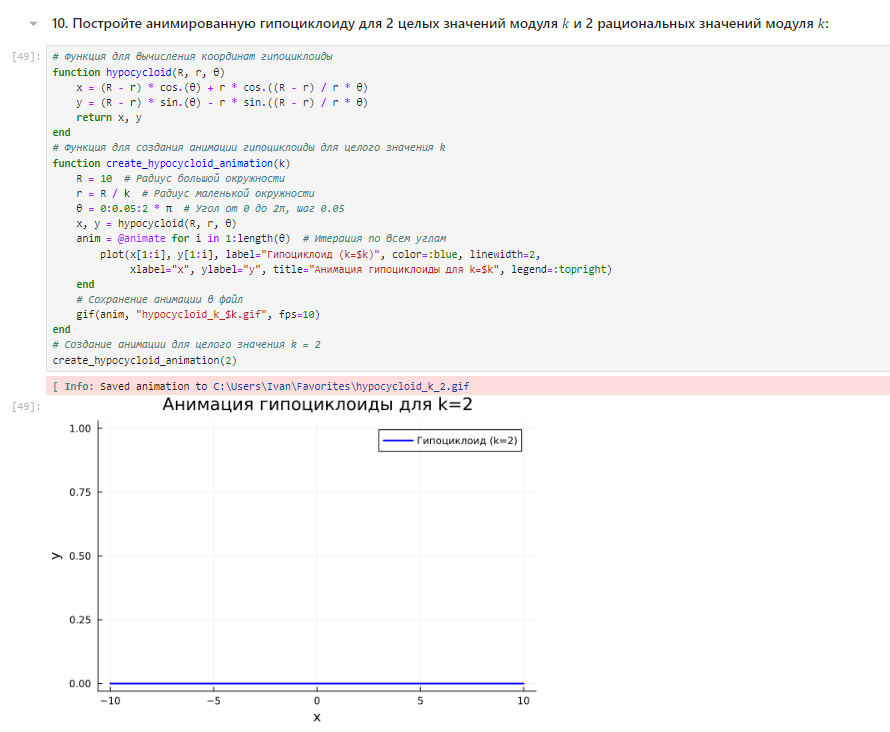


Рис. 56: Решение задания №10

Выполнение задания №11 (рис. 57):

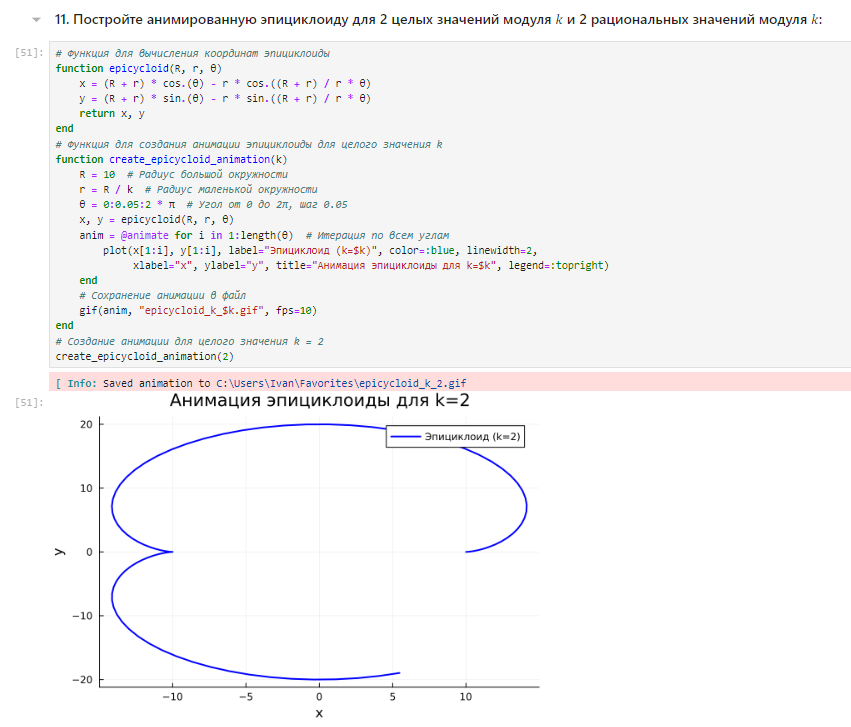


Рис. 57: Решение задания №11

# 3 Вывод

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

# 4 Список литературы. Библиография

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