Отчет по лабораторной работе 5

Петрушов Дмитрий, 1032212287

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](#fig:001)] - (рис. [[2](#fig:002)]):

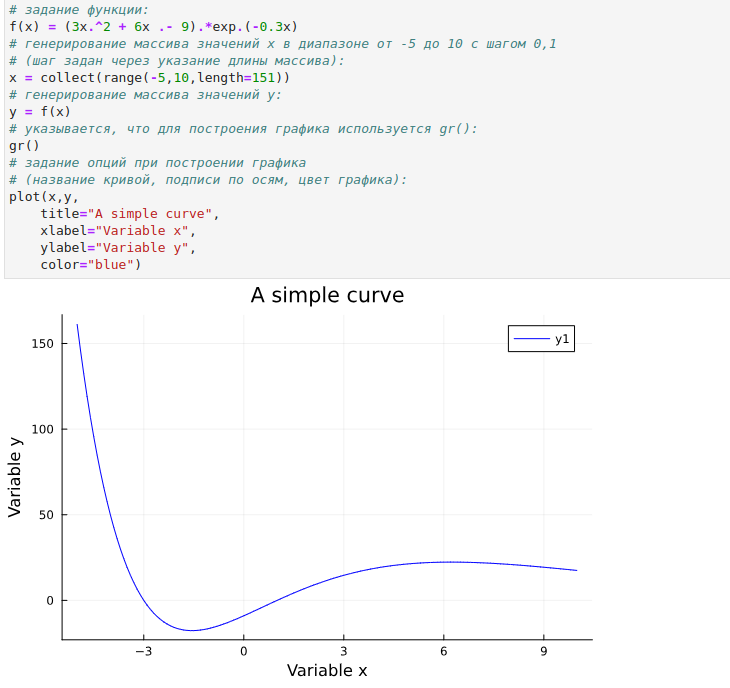


Figure 1: График функции, построенный при помощи gr()

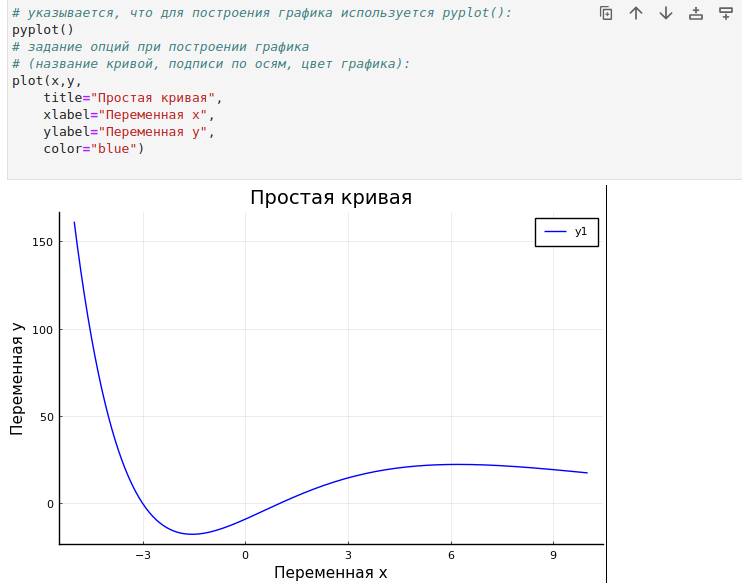


Figure 2: График функции, построенный при помощи pyplot()

## 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](#fig:003)] - рис. [[5](#fig:005)]):

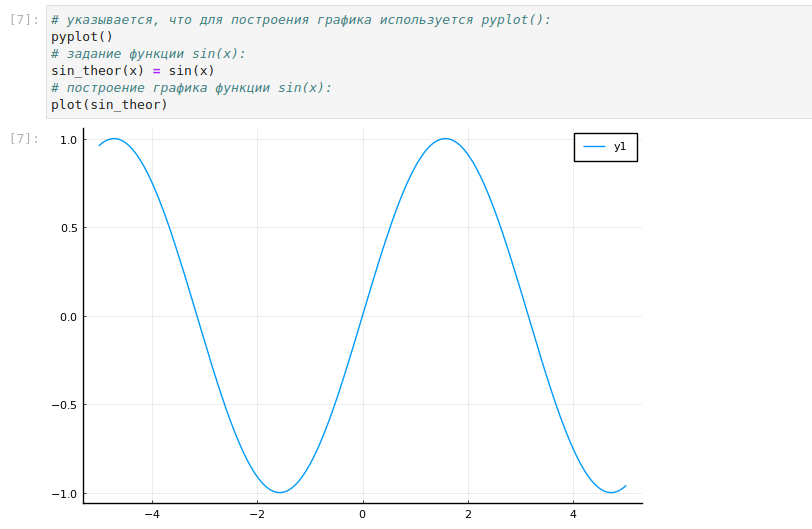


Figure 3: График функции sin(x)

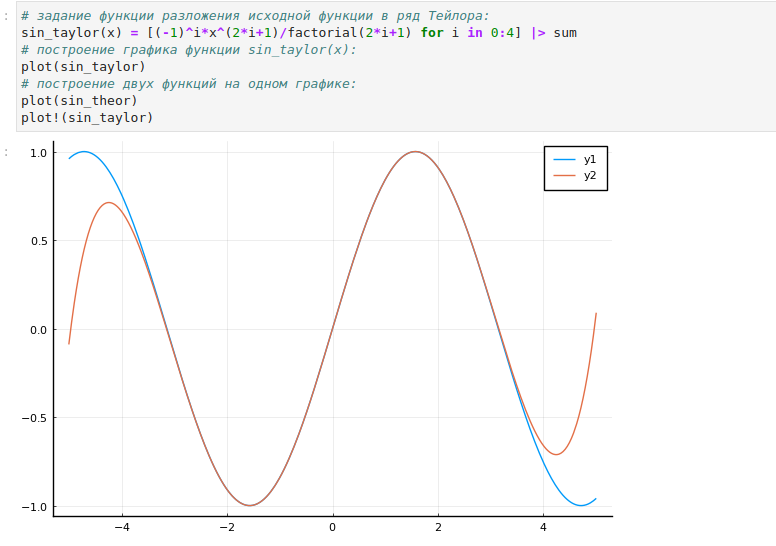


Figure 4: График функции разложения исходной функции в ряд Тейлора



Figure 5: Графики исходной функции и её разложения в ряд Тейлора

Затем добавим различные опции для отображения на графике (рис. [[6](#fig:006)]):

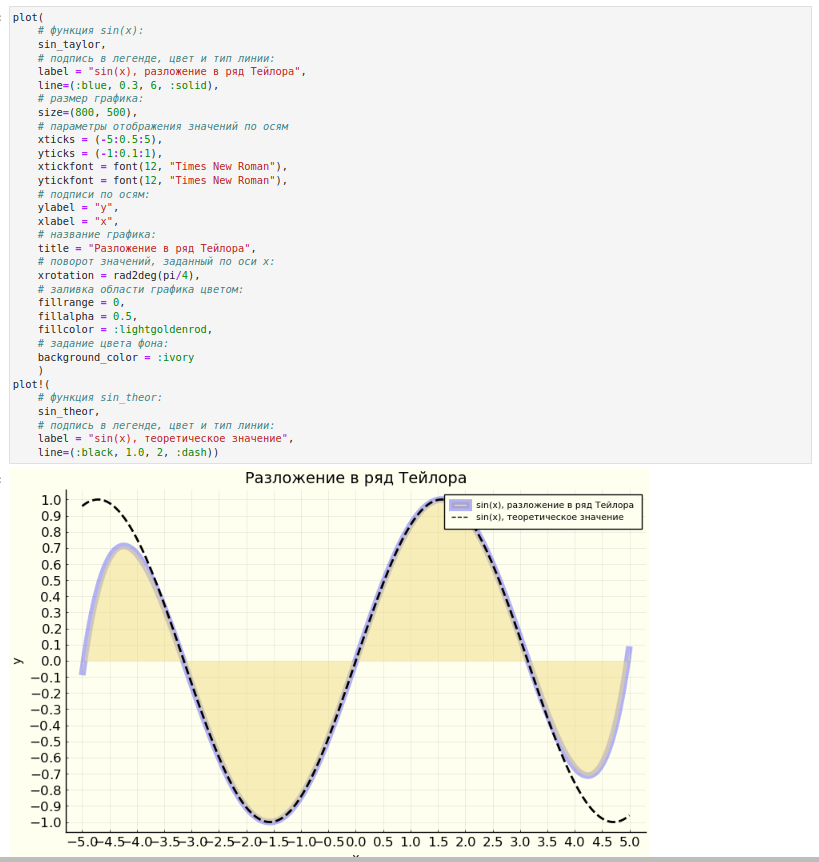


Figure 6: Вид графиков после добавления опций при их построении

## 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](#fig:007)]):

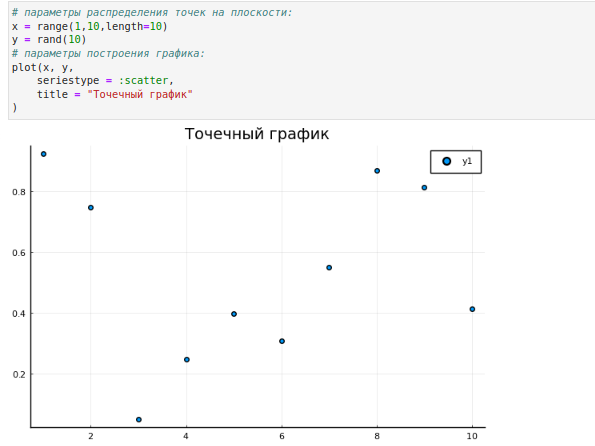


Figure 7: График десяти случайных значений на плоскости (простой точечный график)

Для точечного графика можно задать различные опции, например размер маркера, его тип, цвет и и т.п. (рис. [[8](#fig:008)]):

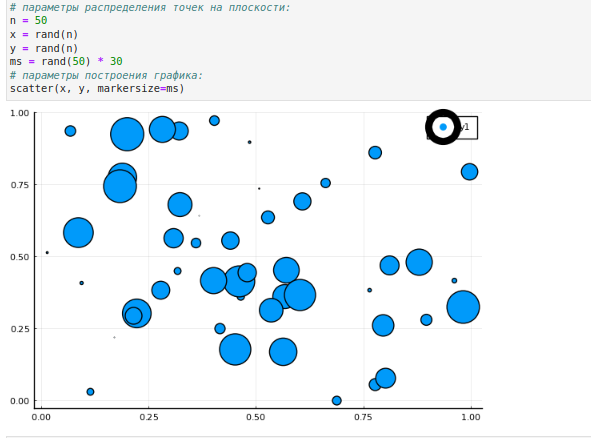


Figure 8: График пятидесяти случайных значений на плоскости с различными опциями отображения (точечный график с кодированием значения размером точки)

Также можно строить и 3-мерные точечные графики (рис. [[9](#fig:009)]):

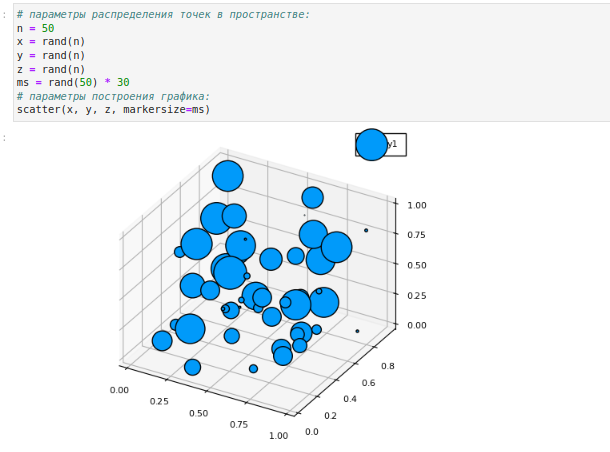


Figure 9: График пятидесяти случайных значений в пространстве с различными опциями отображения (3-мерный точечный график с кодированием значения размером точки)

## 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](#fig:010)]):

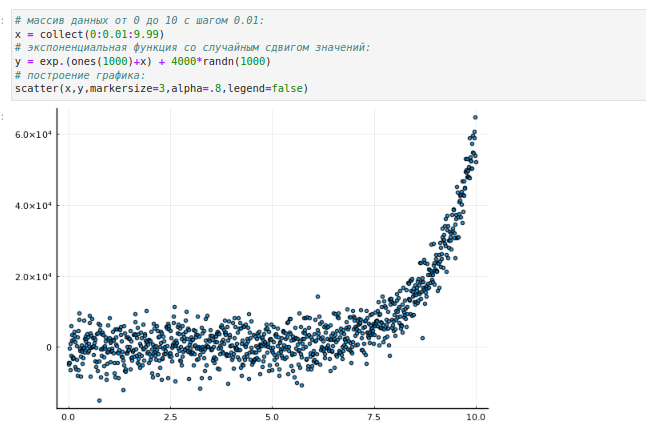


Figure 10: Пример функции

Аппроксимируем полученную функцию полиномом 5-й степени (рис. [[11](#fig:011)]):

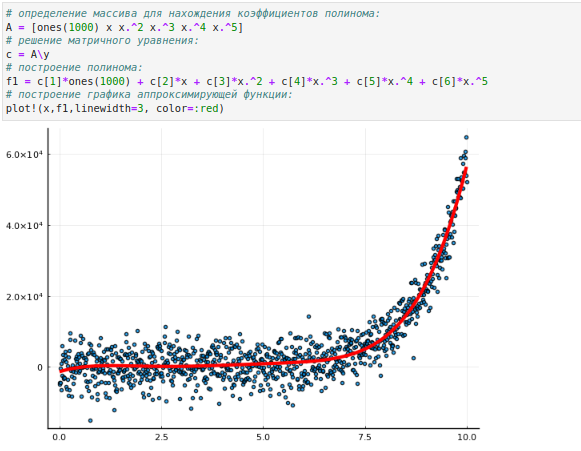


Figure 11: Пример аппроксимации исходной функции полиномом 5-й степени

## 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](#fig:012)]):

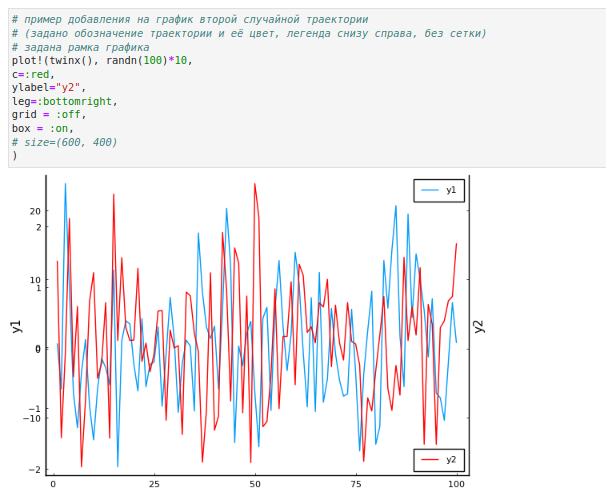


Figure 12: Пример отдельно построенной траектории

## 2.6 Полярные координаты

Приведём пример построения графика функции в полярных координатах (рис. [[13](#fig:013)]):

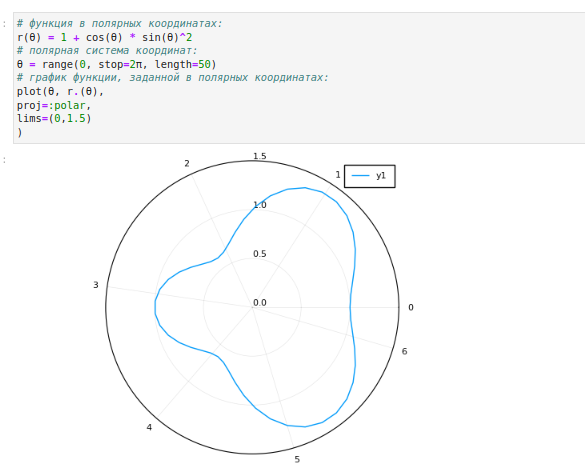


Figure 13: График функции, заданной в полярных координатах

## 2.7 Параметрический график

Приведём пример построения графика параметрически заданной кривой на плоскости (рис. [[14](#fig:014)]):

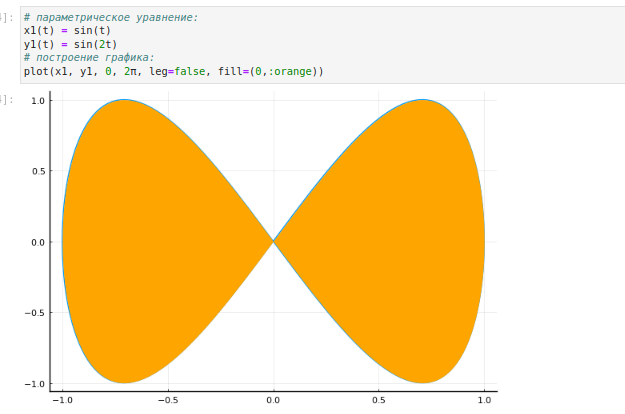


Figure 14: Параметрический график кривой на плоскости

Далее приведём пример построения графика параметрически заданной кривой в пространстве (рис. [[15](#fig:015)]):

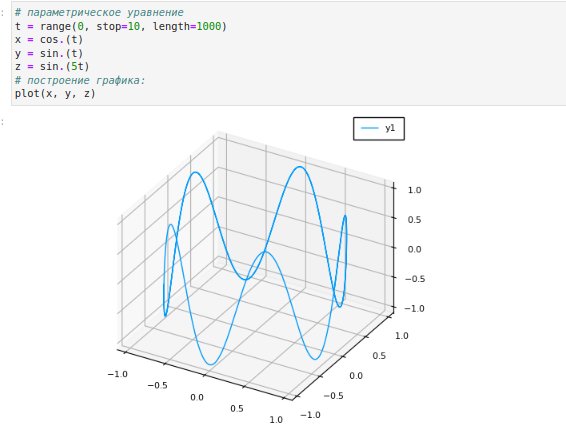


Figure 15: Параметрический график кривой в пространстве

## 2.8 График поверхности

Для построения поверхности, заданной уравнением f(x, y) = x2 + y2, можно воспользоваться функцией surface() (рис. [[16](#fig:016)]):

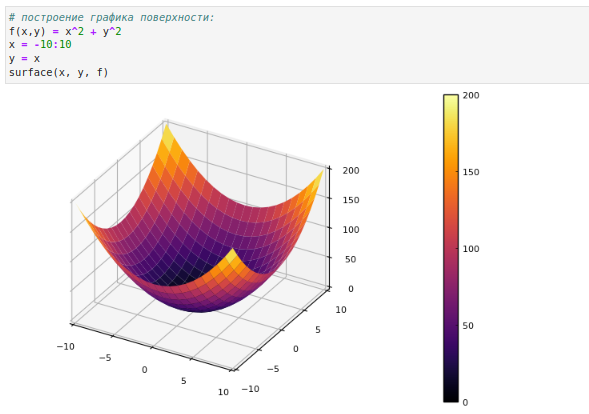


Figure 16: График поверхности (использована функция surface())

Также можно воспользоваться функцией plot() с заданными параметрами (рис. [[17](#fig:017)]):

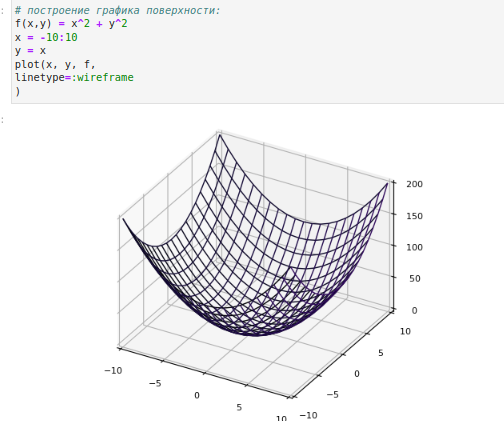


Figure 17: График поверхности (использована функция plot())

Можно задать параметры сглаживания (рис. [[18](#fig:018)]):

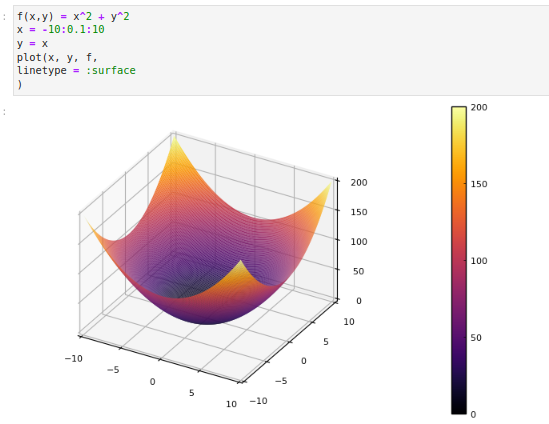


Figure 18: Сглаженный график поверхности

Можно задать определённый угол зрения (рис. [[19](#fig:019)]):

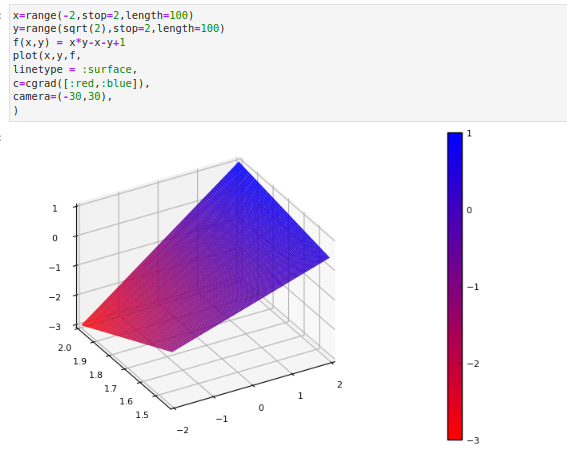


Figure 19: График поверхности с изменённым углом зрения

## 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](#fig:020)]):

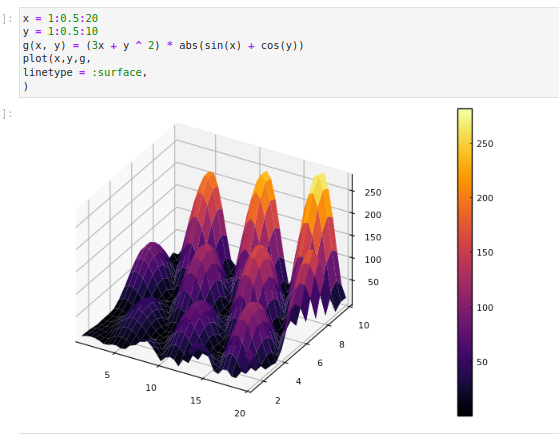


Figure 20: График поверхности, заданной функцией g(x, y) = (3x + y2)| sin(x) + cos(y)|

Линии уровня можно построить, используя проекцию значений исходной функции на плоскость (рис. [[21](#fig:021)]):

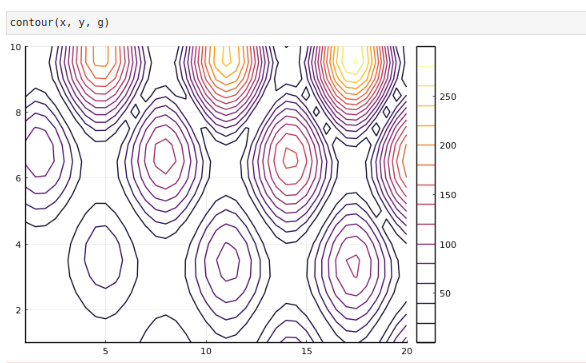


Figure 21: Линии уровня

Можно дополнительно добавить заливку цветом (рис. [[22](#fig:022)]):

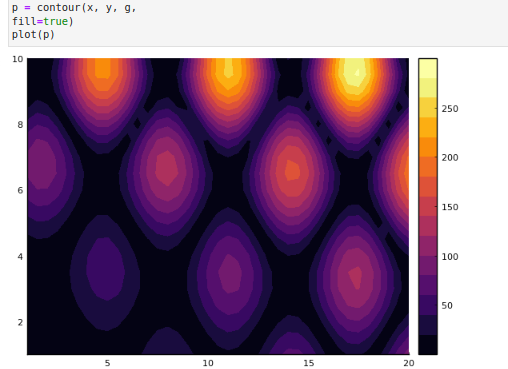


Figure 22: Линии уровня с заполнением

## 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](#fig:023)]) и линии уровня (рис. [[24](#fig:024)]):

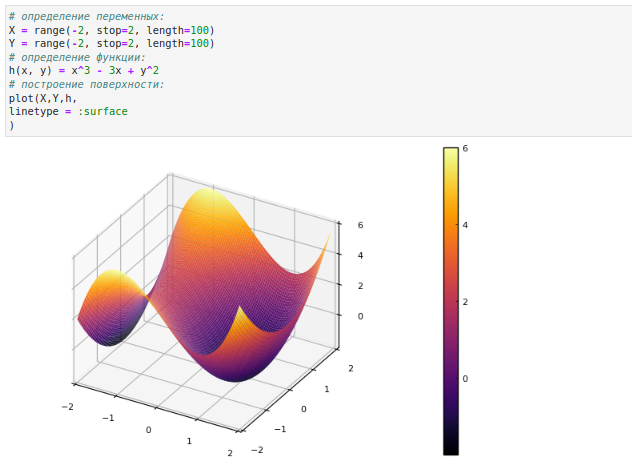


Figure 23: График функции h(x, y) = x3 - 3x + y2

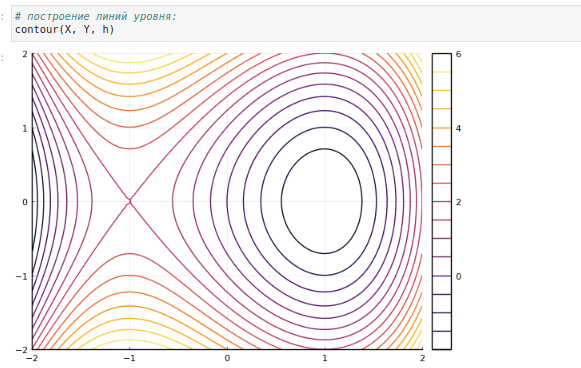


Figure 24: Линии уровня для функции h(x, y) = x3 - 3x + y2

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](#fig:025)]):

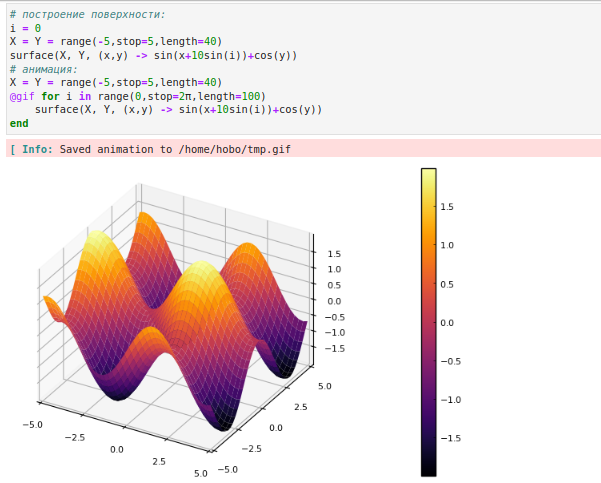


Figure 25: Статичный график поверхности

Добавляем анимацию (рис. [[26](#fig:026)]):

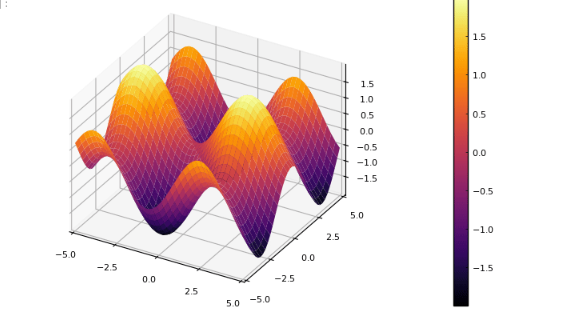


Figure 26: Анимированный график поверхности

## 2.12 Гипоциклоида

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

Построим большую окружность (рис. [[27](#fig:027)]):

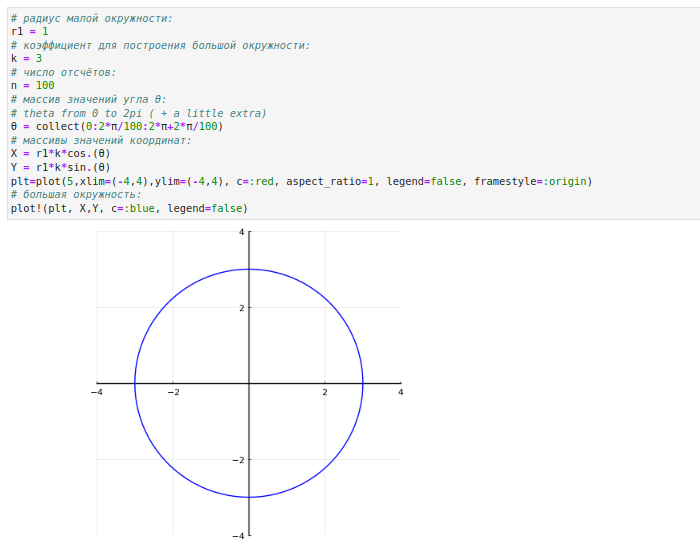


Figure 27: Большая окружность гипоциклоиды

Для частичного построения гипоциклоиды будем менять параметр t (рис. [[28](#fig:028)]):

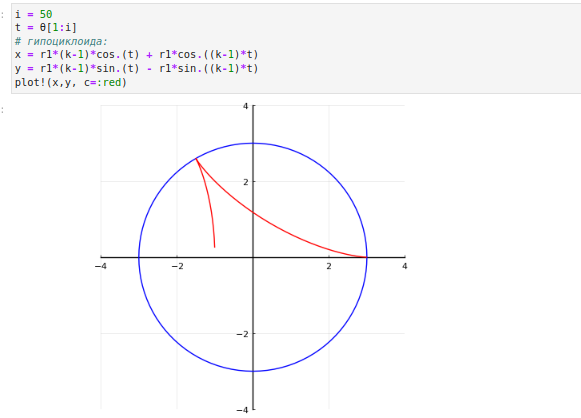


Figure 28: Половина пути гипоциклоиды

Добавляем малую окружность гипоциклоиды (рис. [[29](#fig:029)]):

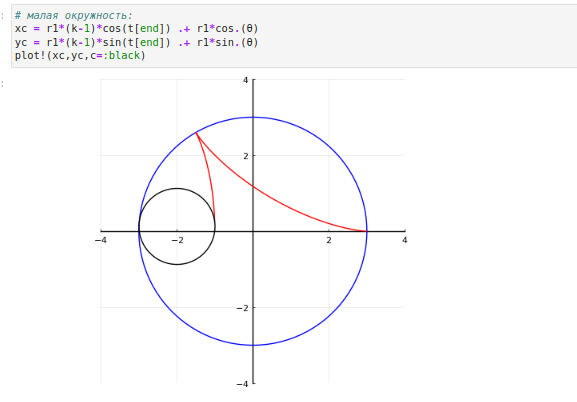


Figure 29: Малая окружность гипоциклоиды

Добавим радиус для малой окружности (рис. [[30](#fig:030)]):

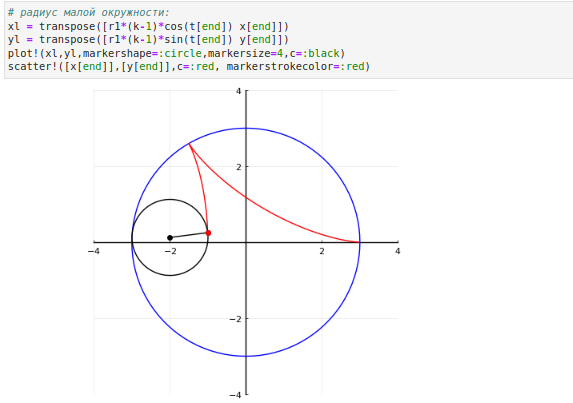


Figure 30: Малая окружность гипоциклоиды с добавлением радиуса

В конце сделаем анимацию получившегося изображения (рис. [[31](#fig:031)]):

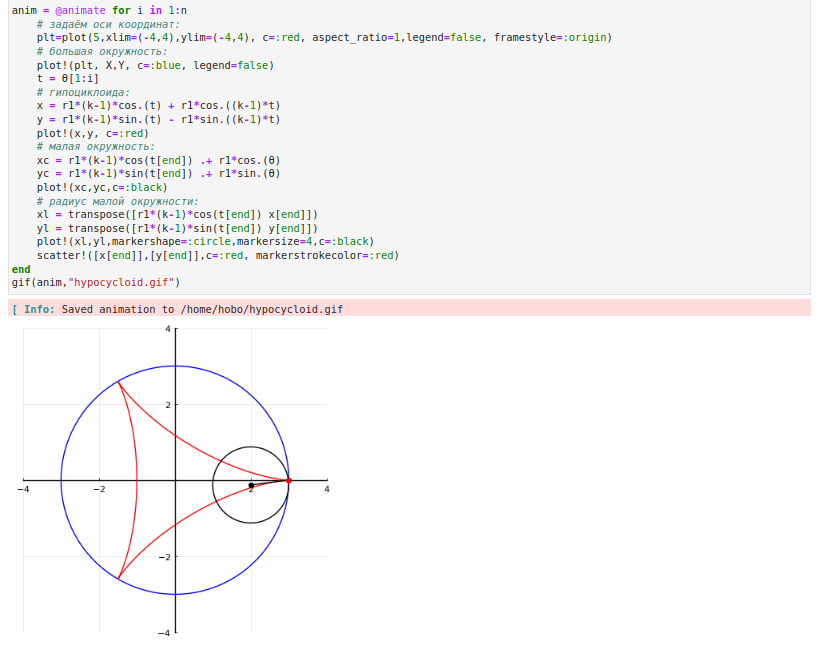


Figure 31: Малая окружность гипоциклоиды с добавлением радиуса

## 2.13 Errorbars

В исследованиях часто требуется изобразить графики погрешностей измерения.

Построим график исходных значений (рис. [[32](#fig:032)]):

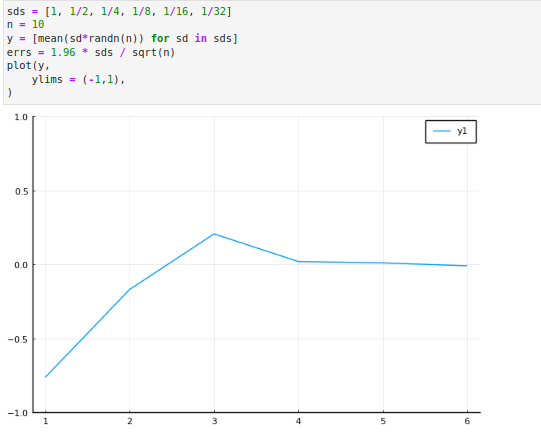


Figure 32: График исходных значений

Построим график отклонений от исходных значений (рис. [[33](#fig:033)]):

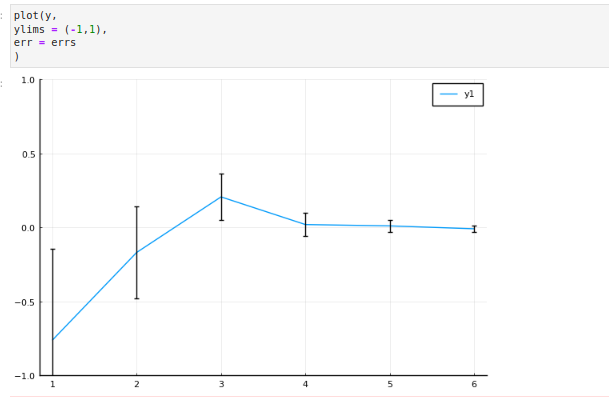


Figure 33: График исходных значений с отклонениями

Повернём график (рис. [[34](#fig:034)]):

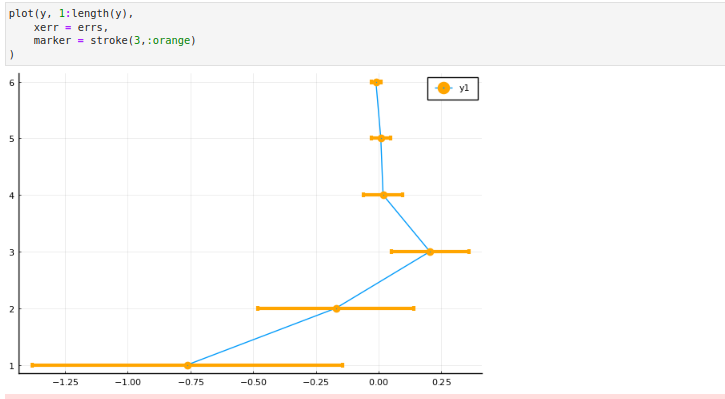


Figure 34: Поворот графика

Заполним область цветом (рис. [[35](#fig:035)]):

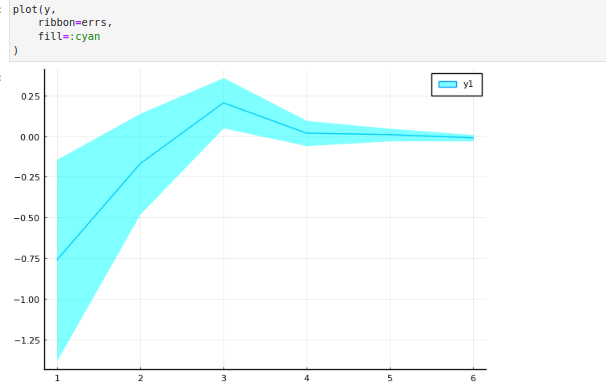


Figure 35: Заполнение цветом

Можно построить график ошибок по двум осям (рис. [[36](#fig:036)]):

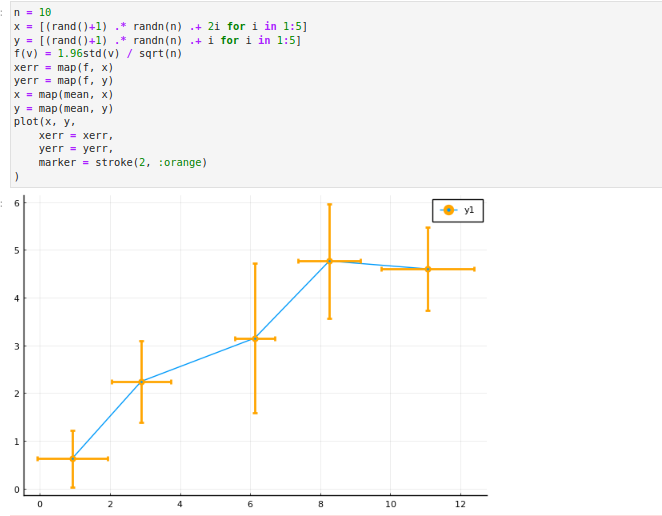


Figure 36: График ошибок по двум осям

Можно построить график асимметричных ошибок по двум осям (рис. [[37](#fig:037)]):

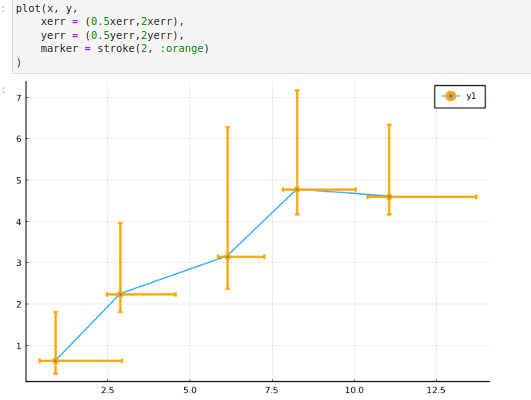


Figure 37: График асимметричных ошибок по двум осям

## 2.14 Использование пакета Distributions

Строим гистограмму.

Задаём нормальное распределение и строим гистограмму.

Далее применим для построения нескольких гистограмм распределения людей по возрастам на одном графике plotly().  
Ничего из этого не получилось сделать.

## 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. [[38](#fig:038)]):

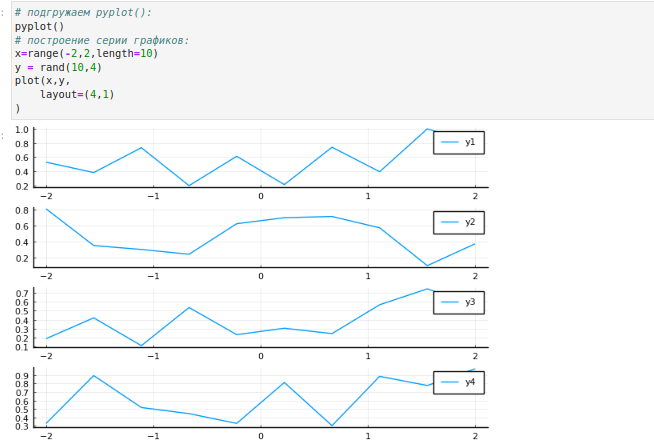


Figure 38: Серия из 4-х графиков в ряд

Для автоматического вычисления сетки необходимо передать layout целое число (рис. [[39](#fig:039)]):

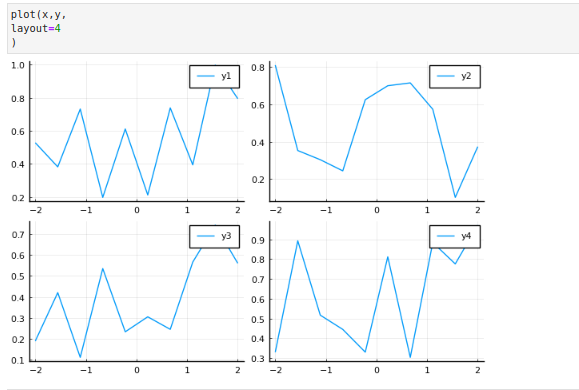


Figure 39: Серия из 4-х графиков в сетке

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. [[40](#fig:040)]):

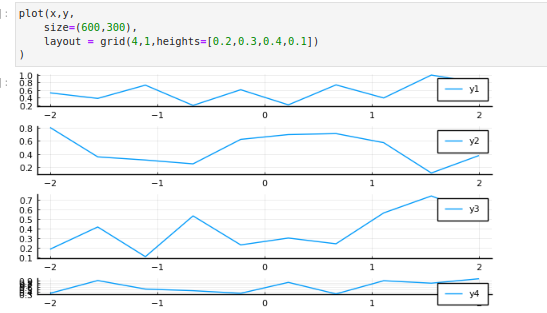


Figure 40: Объединение нескольких графиков в одной сетке

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. [[41](#fig:041)]):



Figure 41: 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. [[42](#fig:042)]):

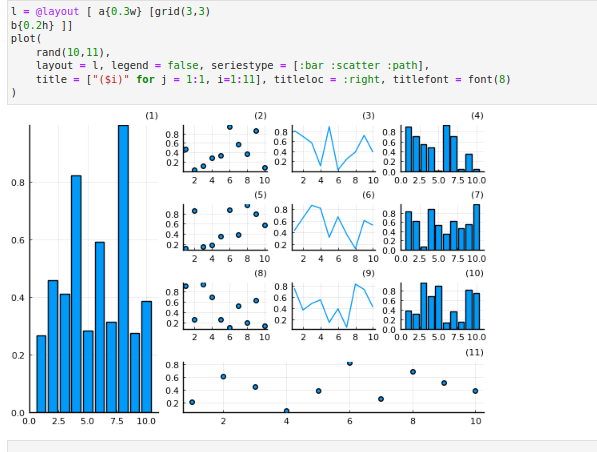


Figure 42: Демонстрация применения сложного макета для построения графиков

# 3 Вывод

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

# Список литературы