

Лабораторная работа №5

Построение графиков

Чемоданова Ангелина Александровна

Содержание

1 Введение	5
1.1 Цели и задачи	5
2 Выполнение лабораторной работы	6
2.1 Основные пакеты для работы с графиками в Julia	6
2.2 Опции при построении графика	9
2.3 Точечный график	12
2.4 Аппроксимация данных	14
2.5 Две оси ординат	15
2.6 Полярные координаты	15
2.7 Параметрический график	16
2.8 График поверхности	17
2.9 Линии уровня	19
2.10 Векторные поля	21
2.11 Анимация	22
2.12 Гипоциклоида	23
2.13 Errorbars	26
2.14 Использование пакета Distributions	30
2.15 Подграфики	31
2.16 Самостоятельное выполнение	34
3 Выводы	44
Список литературы	45

Список иллюстраций

2.1 Задание функции	6
2.2 Задание функции	7
2.3 График функции, построенный при помощи gr()	7
2.4 График функции, построенный при помощи pyplot()	8
2.5 График функции, построенный при помощи unicodeplots()	8
2.6 График функции $\sin(x)$	9
2.7 График функции разложения исходной функции в ряд Тейлора . .	10
2.8 Графики исходной функции и её разложения в ряд Тейлора . . .	10
2.9 Добавления опций для графика	11
2.10 Вид графиков после добавления опций при их построении	11
2.11 График десяти случайных значений на плоскости (простой точечный график)	12
2.12 График пятидесяти случайных значений на плоскости с различными опциями отображения (точечный график с кодированием значения размером точки)	13
2.13 График пятидесяти случайных значений в пространстве с различными опциями отображения (3-мерный точечный график с кодированием значения размером точки)	13
2.14 Пример функции	14
2.15 Пример аппроксимации исходной функции полиномом 5-й степени	14
2.16 Пример отдельно построенной траектории	15
2.17 График функции, заданной в полярных координатах	16
2.18 Параметрический график кривой на плоскости	16
2.19 Параметрический график кривой в пространстве	17
2.20 График поверхности (использована функция surface())	17
2.21 График поверхности (использована функция plot())	18
2.22 Сглаженный график поверхности	18
2.23 График поверхности с изменённым углом зрения	19
2.24 График поверхности, заданной функцией $g(x, y) = (3x + y^2) \sin(x) + \cos(y) $	20
2.25 Линии уровня	20
2.26 Линии уровня с заполнением	21
2.27 График функции $h(x, y) = x^3 - 3x + y^2$	22
2.28 Линии уровня для функции $h(x, y) = x^3 - 3x + y^2$	22
2.29 Статичный график поверхности	23
2.30 Анимированный график поверхности	23
2.31 Построение большой окружности гипоциклоиды	24

2.32 Большая окружность гипоциклоиды	24
2.33 Половина пути гипоциклоиды	25
2.34 Малая окружность гипоциклоиды	25
2.35 Малая окружность гипоциклоиды с добавлением радиуса	26
2.36 Зададим исходные значения	26
2.37 График исходных значений	27
2.38 График исходных значений с отклонениями	27
2.39 Поворот графика	28
2.40 Заполнение цветом	28
2.41 График ошибок по двум осям	29
2.42 График асимметричных ошибок по двум осям	29
2.43 Гистограмма, построенная по массиву случайных чисел	30
2.44 Гистограмма нормального распределения	30
2.45 Серия из 4-х графиков в ряд	31
2.46 Серия из 4-х графиков в сетке	31
2.47 Серия из 4-х графиков разной высоты в ряд	32
2.48 Объединение нескольких графиков в одной сетке	32
2.49 Объединение нескольких графиков в одной сетке	33
2.50 Разнообразные варианты представления данных	33
2.51 Демонстрация применения сложного макета для построения графиков	34
2.52 Решение задания №1	34
2.53 Решение задания №2	35
2.54 Решение задания №3	35
2.55 Решение задания №4	36
2.56 Решение задания №5	36
2.57 Решение задания №5	37
2.58 Решение задания №6	37
2.59 Решение задания №7	38
2.60 Решение задания №8	38
2.61 Решение задания №9	39
2.62 Решение задания №10	39
2.63 Решение задания №10	40
2.64 Решение задания №10	40
2.65 Решение задания №10	40
2.66 Решение задания №10	41
2.67 Решение задания №10	41
2.68 Решение задания №11	41
2.69 Решение задания №11	42
2.70 Решение задания №11	42
2.71 Решение задания №11	42
2.72 Решение задания №11	43
2.73 Решение задания №11	43

1 Введение

1.1 Цели и задачи

Цель работы

Основная цель работы – освоить синтаксис языка Julia для построения графиков[1].

Задание

1. Используя Jupyter Lab, повторите примеры. При этом дополните графики обозначениями осей координат, легендой с названиями траекторий, названиями графиков и т.п.
2. Выполните задания для самостоятельной работы[2].

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 $f(x) = (3x^2 + 6x - 9)e^{-0.3x}$ разными способами (рис. 2.1 - рис. 2.5):

```
# подключаем для использования Plots:
using Plots
✓ 0.0s

# задание функции:
f(x) = (3x.^2 + 6x .- 9).*exp.(-0.3x)
✓ 0.0s
f (generic function with 2 methods)

# генерирование массива значений x в диапазоне от -5 до 10 с шагом 0,1
# (шаг задан через указание длины массива):
x = collect(range(-5,10,length=151))
✓ 0.0s
151-element Vector{Float64}:
-5.0
-4.9
-4.8
-4.7
-4.6
-4.5
-4.4
-4.3
-4.2
-4.1
⋮
9.2
9.3
9.4
9.5
9.6
9.7
9.8
9.9
10.0
```

Рис. 2.1: Задание функции

```

# генерирование массива значений y:
y = f(x)
✓ 0.6s

151-element Vector{Float64}:
161.34080653217032
146.26477799394003
132.19219298833204
119.06942359634911
106.8453557470588
95.47128188475011
84.9007968362764
75.08969810741056
65.9958924347995
57.579293095755176
⋮
18.9951255200995375
18.811475185747092
18.625664977689375
18.437877278854756
18.248288762126195
18.057878179740368
17.86438705526336
17.670399177629
17.475260997120245

# указывается, что для построения графика используется gr():
gr()
✓ 0.4s

Plots.GRBackend()

```

Рис. 2.2: Задание функции

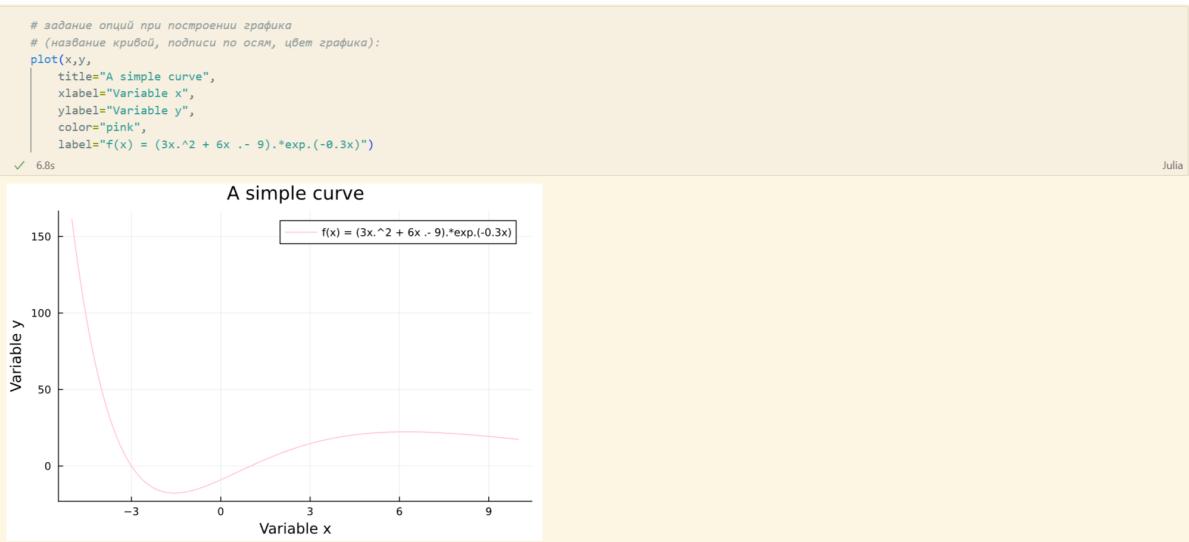


Рис. 2.3: График функции, построенный при помощи gr()

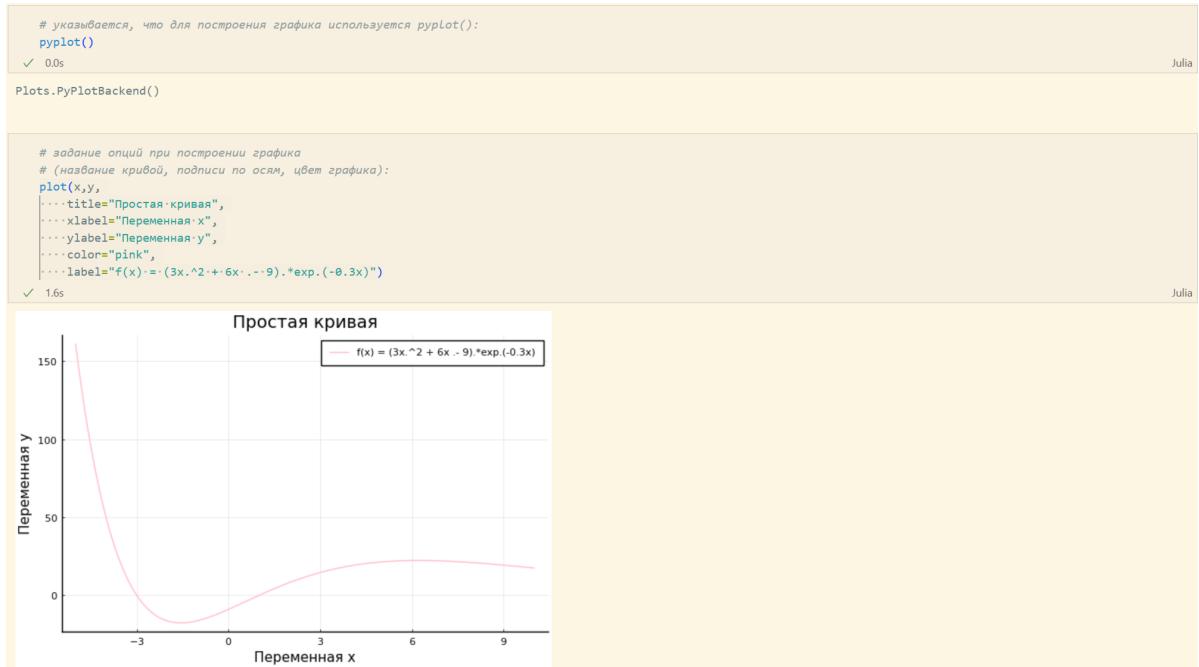


Рис. 2.4: График функции, построенный при помощи pyplot()

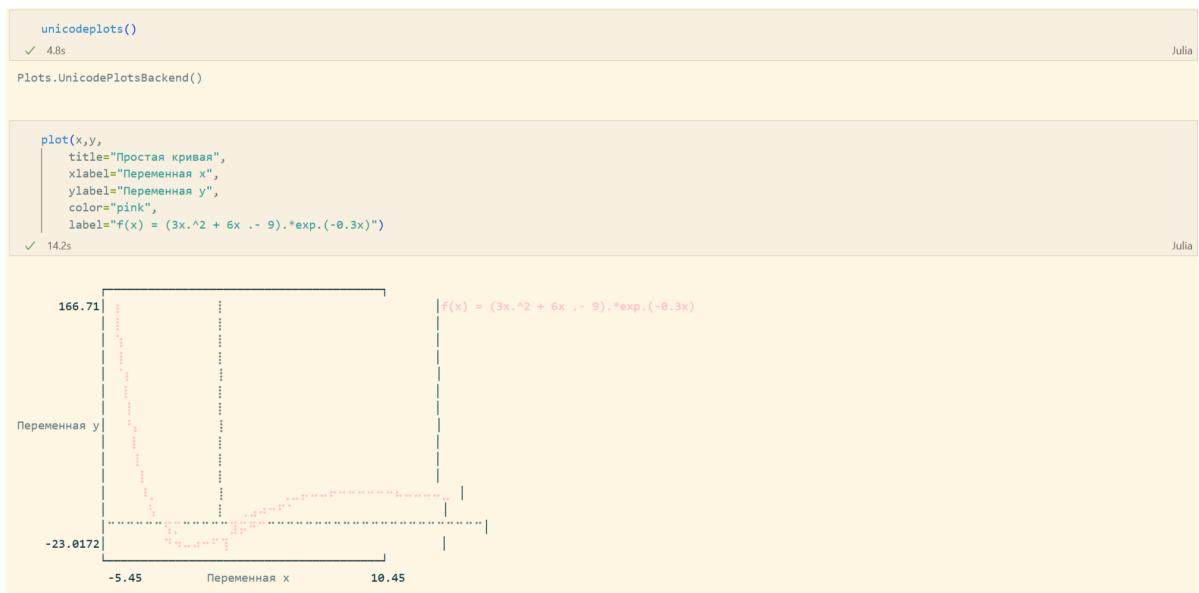


Рис. 2.5: График функции, построенный при помощи unicodeplots()

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. 2.6 - рис. 2.8):

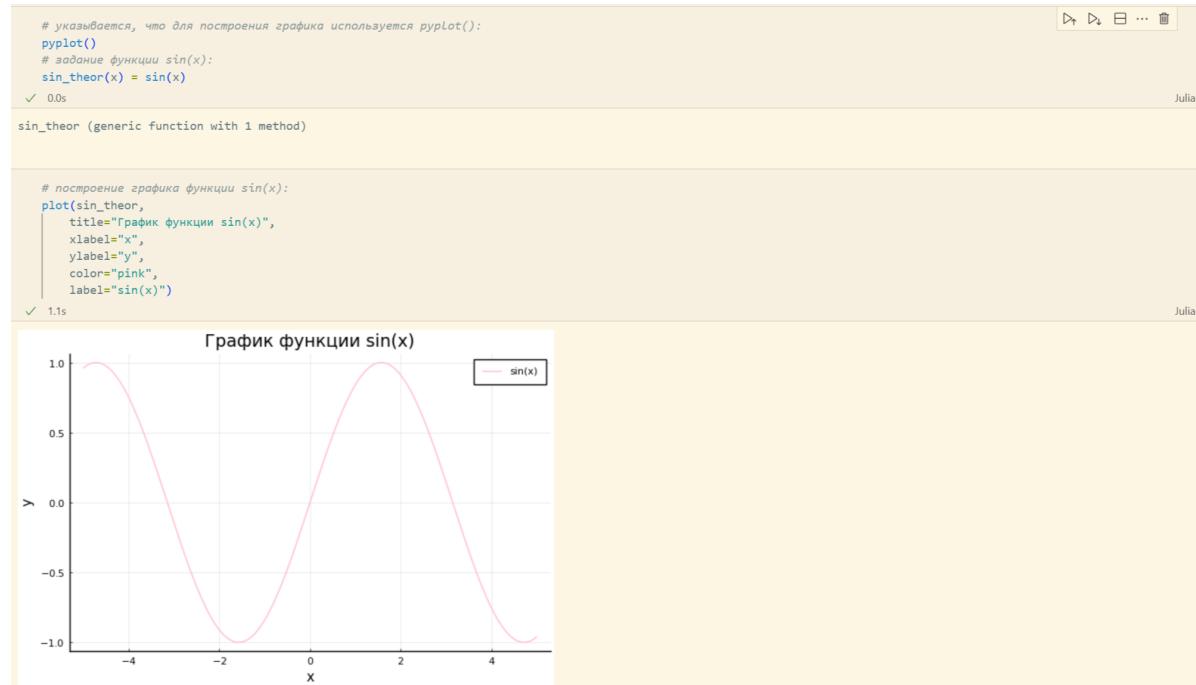


Рис. 2.6: График функции $\sin(x)$

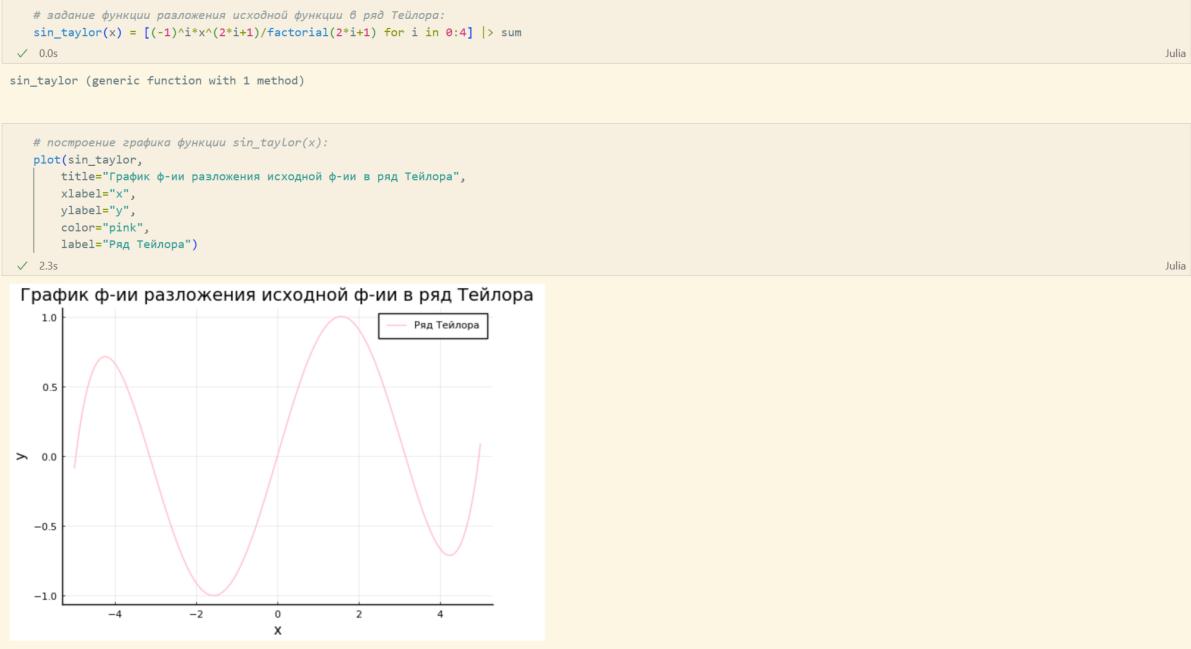


Рис. 2.7: График функции разложения исходной функции в ряд Тейлора

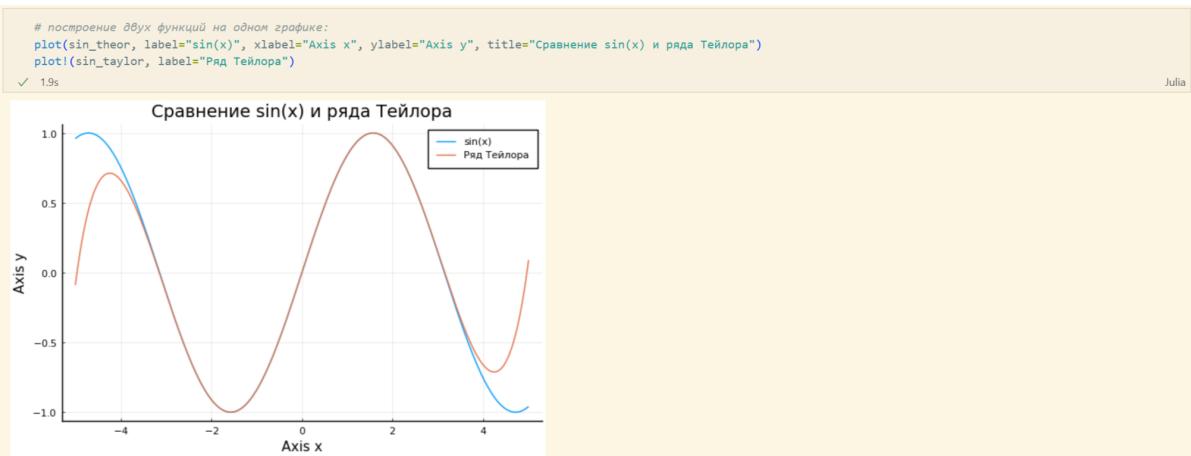


Рис. 2.8: Графики исходной функции и её разложения в ряд Тейлора

Затем добавим различные опции для отображения на графике (рис. 2.9 - рис. 2.10):

```

plot(
    # функция sin(x):
    sin_taylor,
    # подпись в легенде, цвет и тип линии:
    label = "sin(x), разложение в ряд Тейлора",
    line=(:blue, 0.3, 6, :solid),
    # размер графика:
    size=(800, 500),
    # параметры отображения значений по осям
    xticks = (-5:0.5:5),
    yticks = (-1:0.1:1),
    xtickfont = font(12, "Times New Roman"),
    ytickfont = font(12, "Times New Roman"),
    # подписи по осям:
    ylabel = "y",
    xlabel = "x",
    # назование графика:
    title = "Разложение в ряд Тейлора",
    # подбором значений, заданный по оси x:
    xrotation = rad2deg(pi/4),
    # заливка области графика цветом:
    fillrange = 0,
    fillalpha = 0.5,
    fillcolor = :lightgoldenrod,
    # задание цвета фона:
    background_color = :ivory
)
plot!(
    # функция sin_theor:
    sin_theor,
    # подпись в легенде, цвет и тип линии:
    label = "sin(x), теоретическое значение",
    line=(:black, 1.0, 2, :dash))

```

✓ 7.3s

Julia

Рис. 2.9: Добавления опций для графика

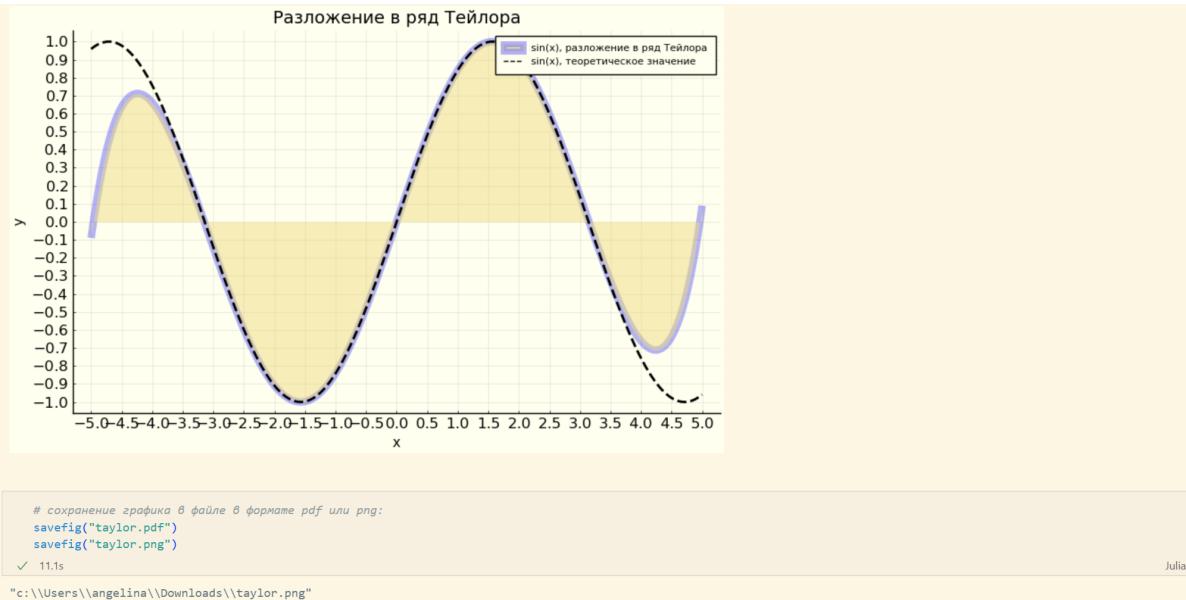


Рис. 2.10: Вид графиков после добавления опций при их построении

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. 2.11):

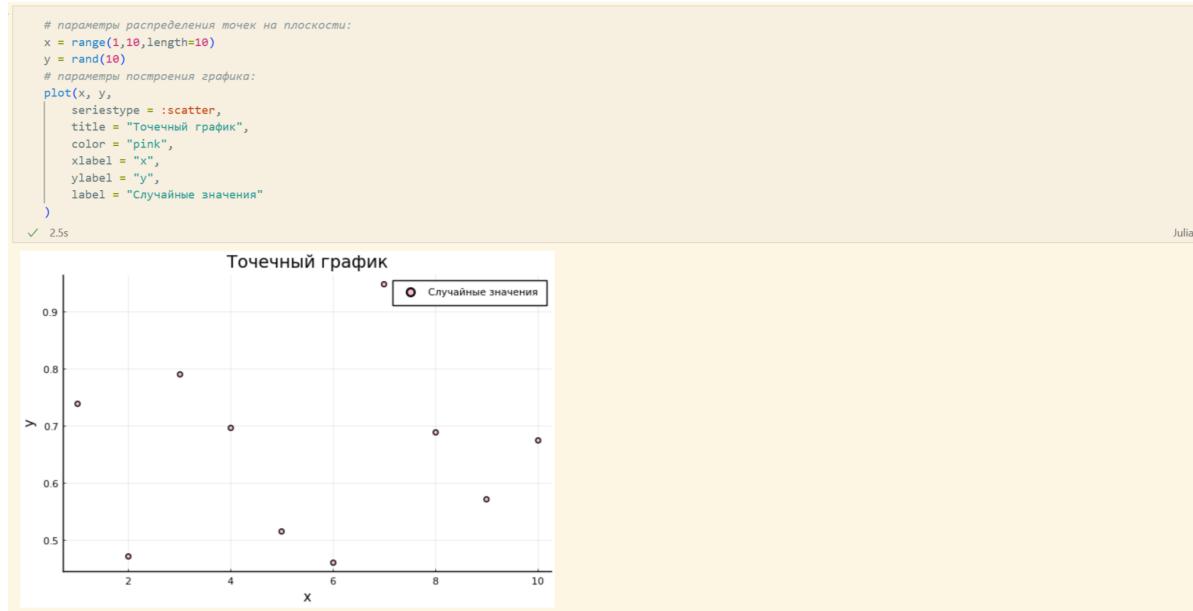


Рис. 2.11: График десяти случайных значений на плоскости (простой точечный график)

Для точечного графика можно задать различные опции, например размер маркера, его тип, цвет и т.п. (рис. 2.12):

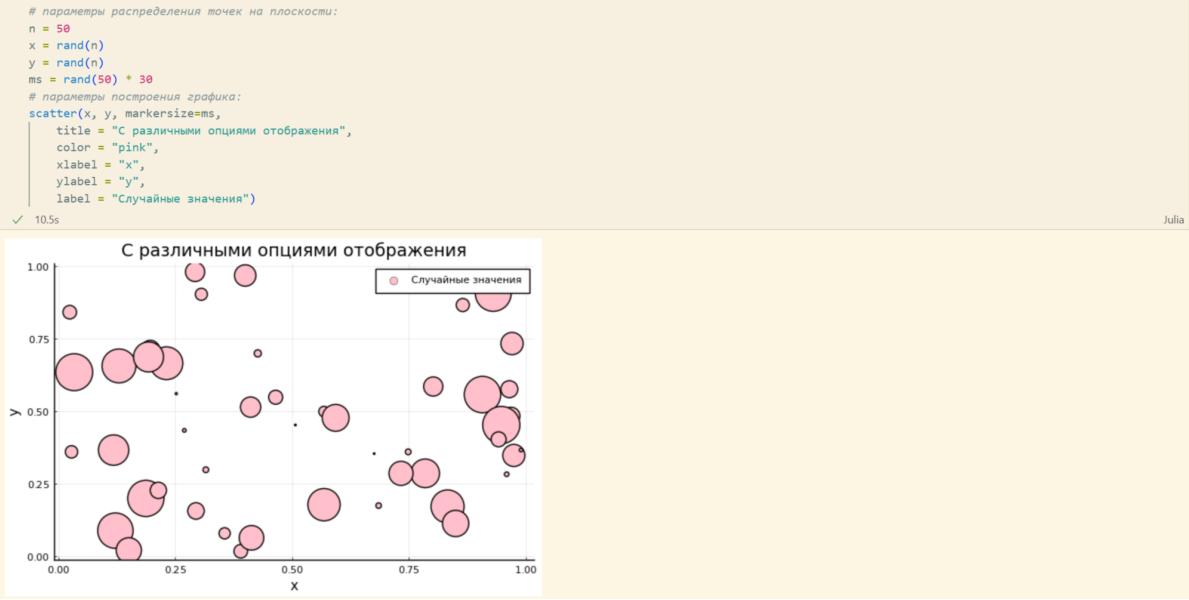


Рис. 2.12: График пятидесяти случайных значений на плоскости с различными опциями отображения (точечный график с кодированием значения размером точки)

Также можно строить и 3-мерные точечные графики (рис. 2.13):

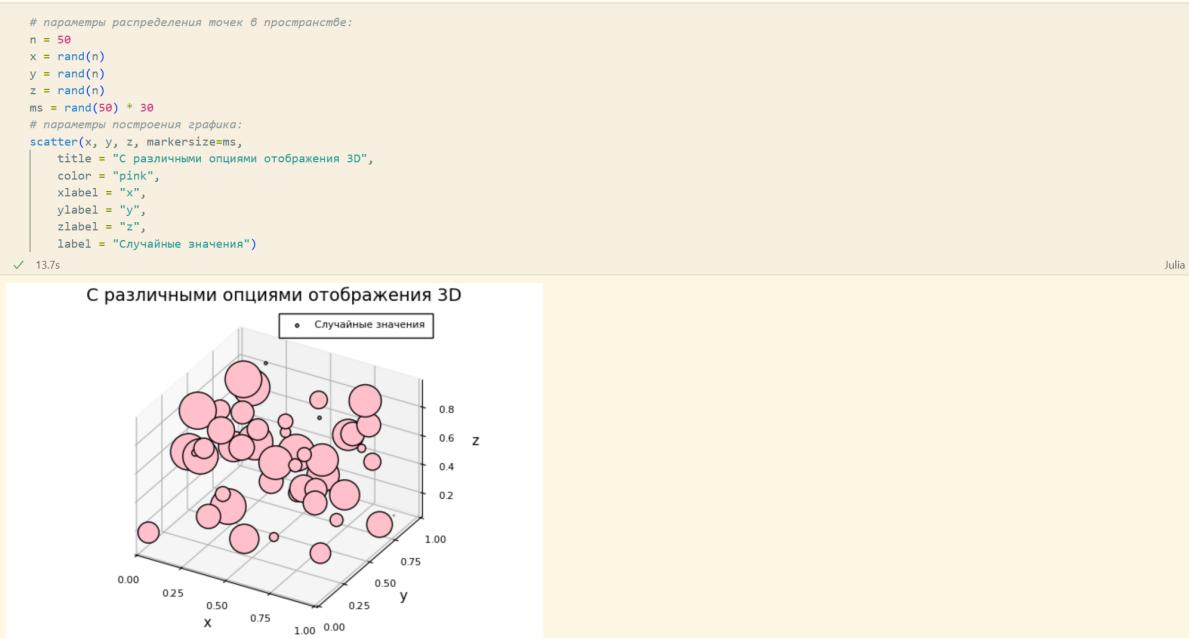


Рис. 2.13: График пятидесяти случайных значений в пространстве с различными опциями отображения (3-мерный точечный график с кодированием значения размером точки)

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. 2.14):

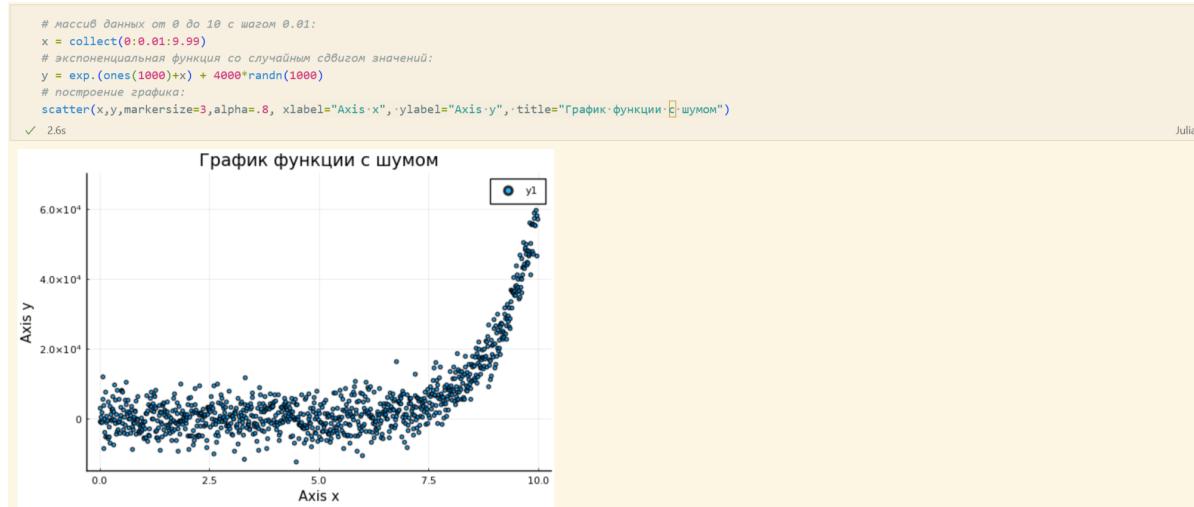


Рис. 2.14: Пример функции

Аппроксимируем полученную функцию полиномом 5-й степени (рис. 2.15):

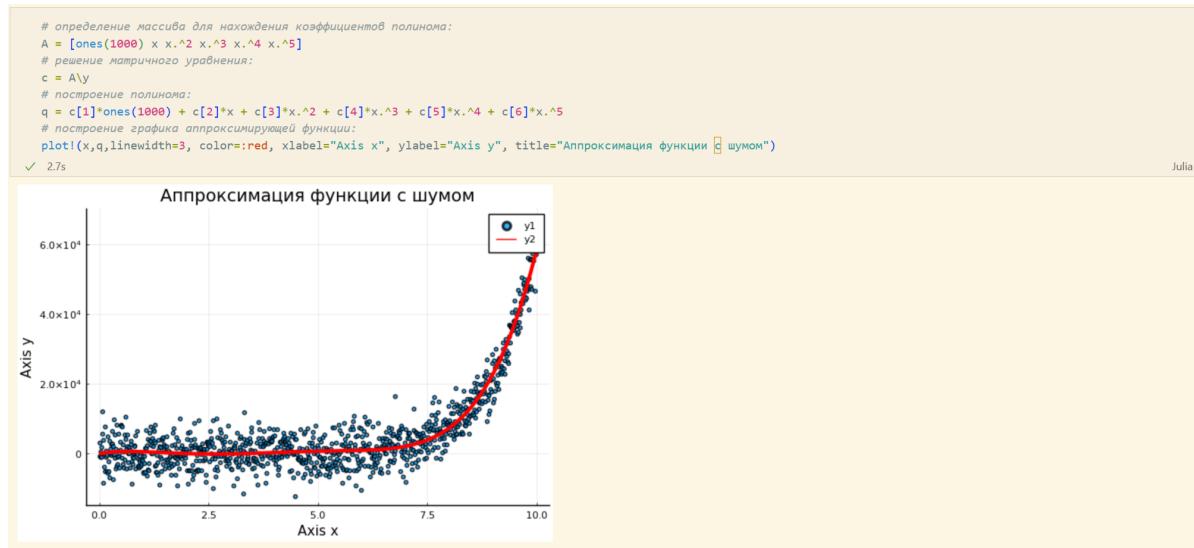


Рис. 2.15: Пример аппроксимации исходной функции полиномом 5-й степени

2.5 Две оси ординат

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

Пример траектории (рис. 2.16):



Рис. 2.16: Пример отдельно построенной траектории

2.6 Полярные координаты

Приведём пример построения графика функции в полярных координатах (рис. 2.17):

```

# функция в полярных координатах:
r(θ) = 1 + cos(θ) * sin(θ)^2
# полярная система координат:
θ = range(θ, stop=2π, length=50)
# график функции, заданной в полярных координатах:
plot(θ, r(θ),
      proj=:polar,
      lims=(θ, 1.5),
      xlabel="Угол",
      ylabel="Радиус",
      title="Полярная кривая",
      label="1 + cos(θ) * sin(θ)^2")

```

Julia



Рис. 2.17: График функции, заданной в полярных координатах

2.7 Параметрический график

Приведём пример построения графика параметрически заданной кривой на плоскости (рис. 2.18):

```

# параметрическое уравнение:
x_1(t) = sin(t)
y_1(t) = sin(2t)
# построение графика:
plot(x_1, y_1, 2π, leg=false, fill=(0,:orange), xlabel="sin(t)", ylabel="sin(2t)", title="Параметрическая траектория", label="Траектория(x=sin(t), y=sin(2t))")

```

Julia

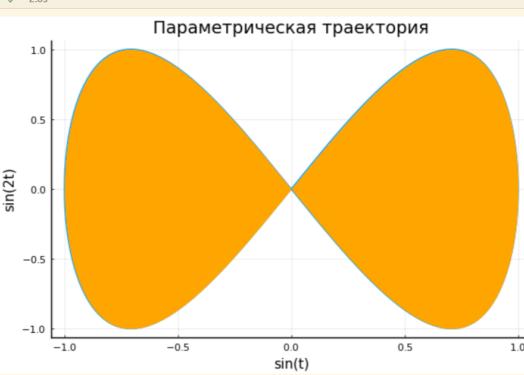


Рис. 2.18: Параметрический график кривой на плоскости

Далее приведём пример построения графика параметрически заданной

кривой в пространстве (рис. 2.19):

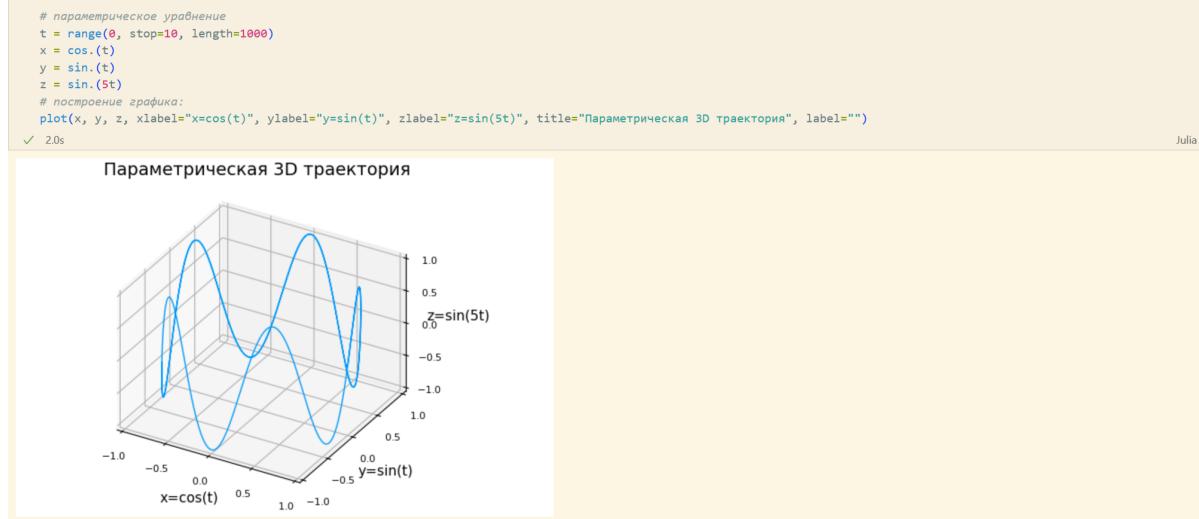


Рис. 2.19: Параметрический график кривой в пространстве

2.8 График поверхности

Для построения поверхности, заданной уравнением $f(x, y) = x^2 + y^2$, можно воспользоваться функцией `surface()` (рис. 2.20):

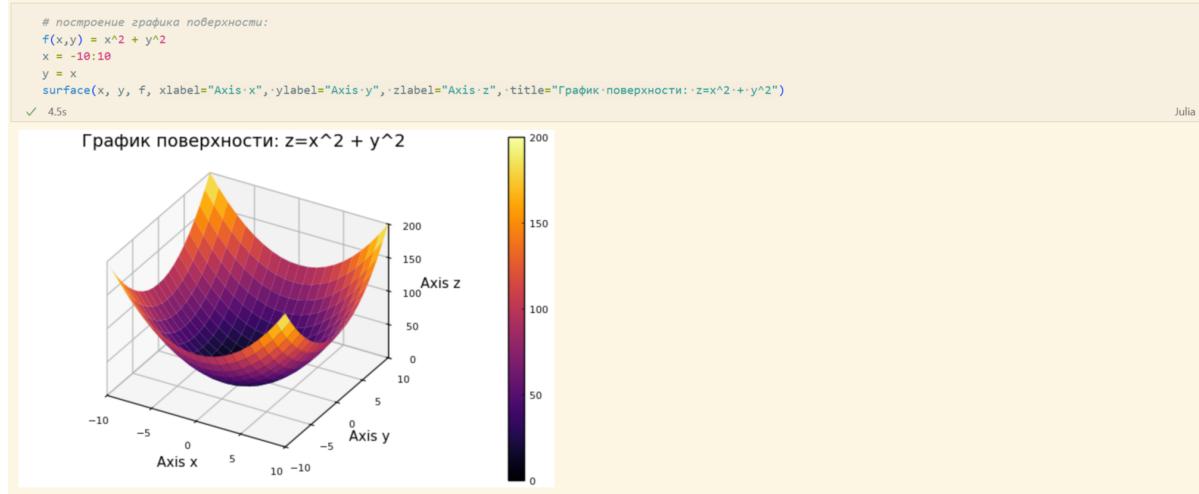


Рис. 2.20: График поверхности (использована функция `surface()`)

Также можно воспользоваться функцией `plot()` с заданными параметрами (рис. 2.21):

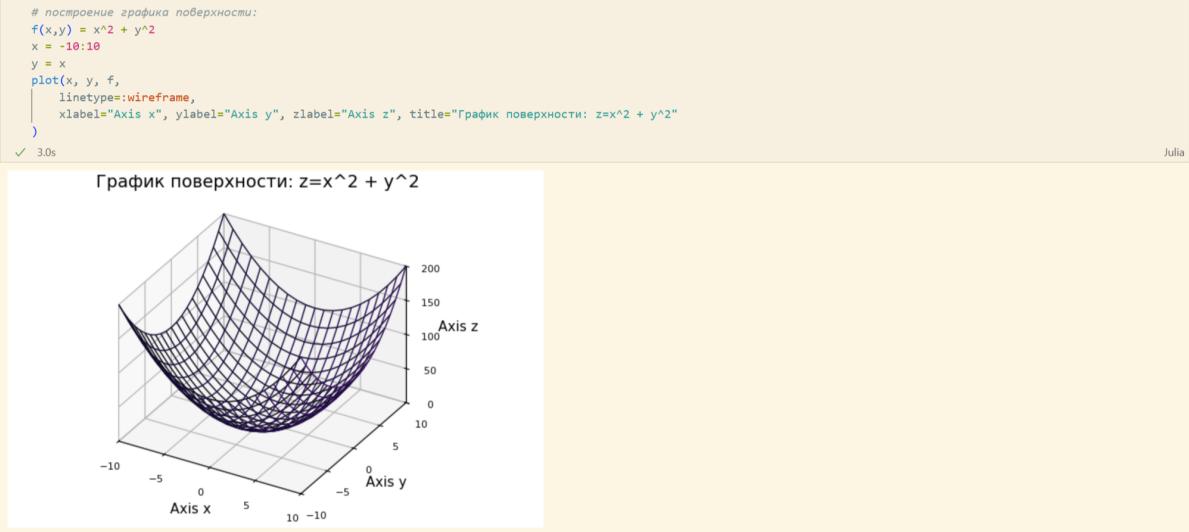


Рис. 2.21: График поверхности (использована функция plot())

Можно задать параметры сглаживания (рис. 2.22):

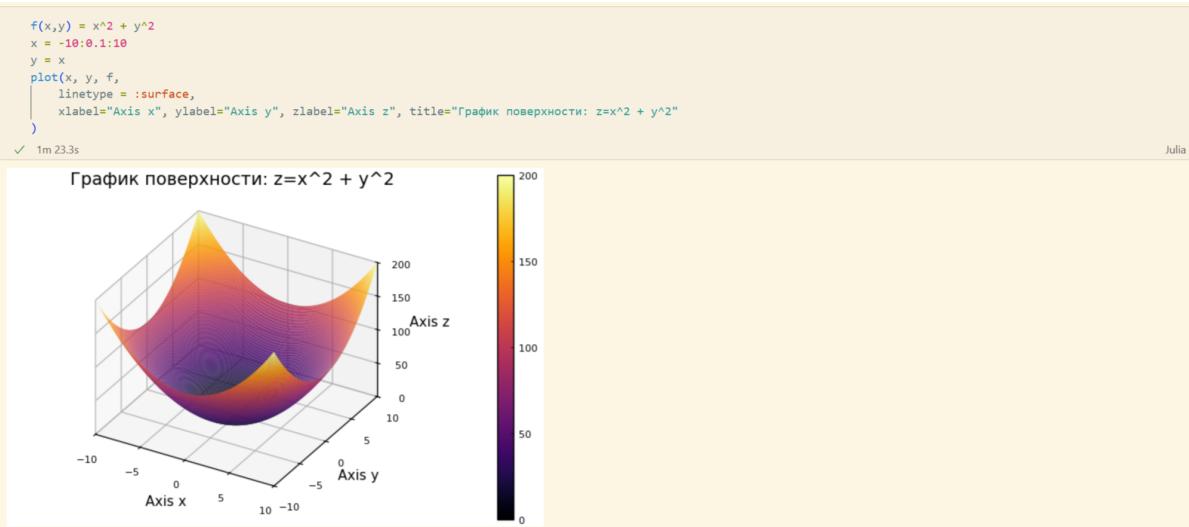


Рис. 2.22: Сглаженный график поверхности

Можно задать определённый угол зрения (рис. 2.23):

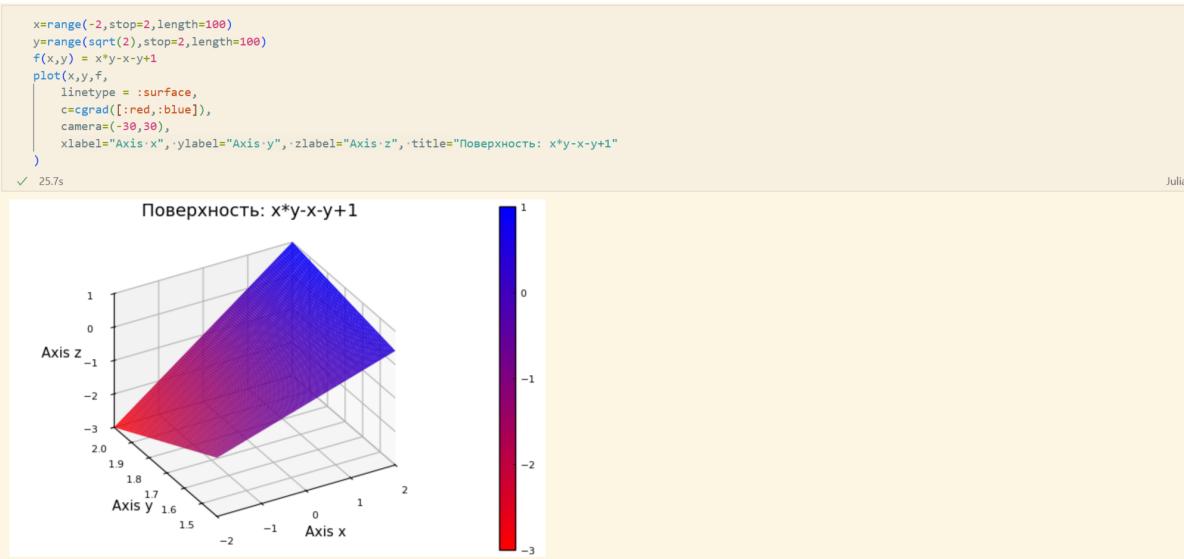


Рис. 2.23: График поверхности с изменённым углом зрения

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 $g(x, y) = (3x + y^2)|\sin(x) + \cos(y)|$ (рис. 2.24):

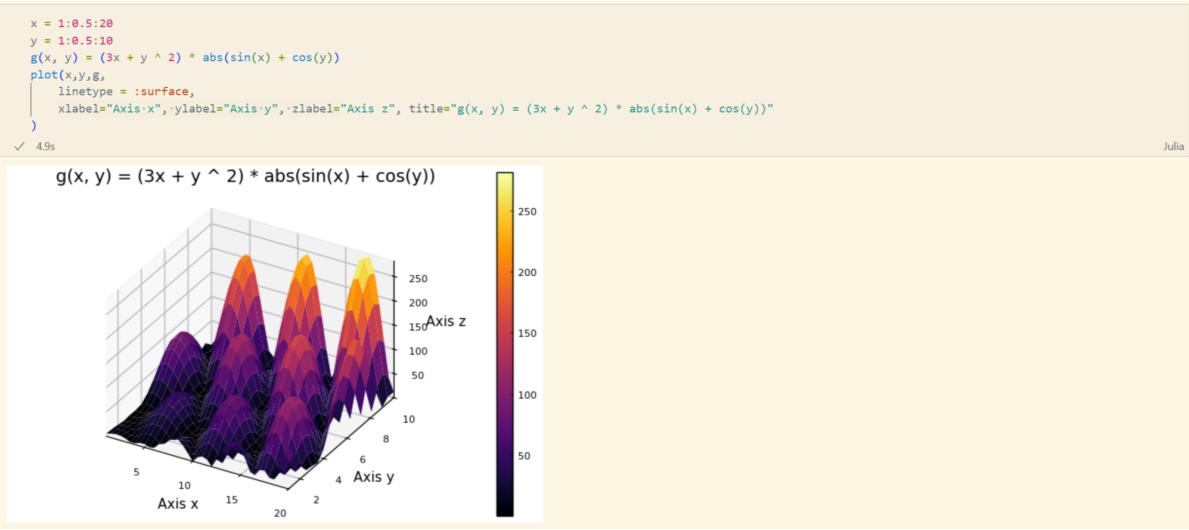


Рис. 2.24: График поверхности, заданной функцией $g(x, y) = (3x + y^2)|\sin(x) + \cos(y)|$

Линии уровня можно построить, используя проекцию значений исходной функции на плоскость (рис. 2.25):

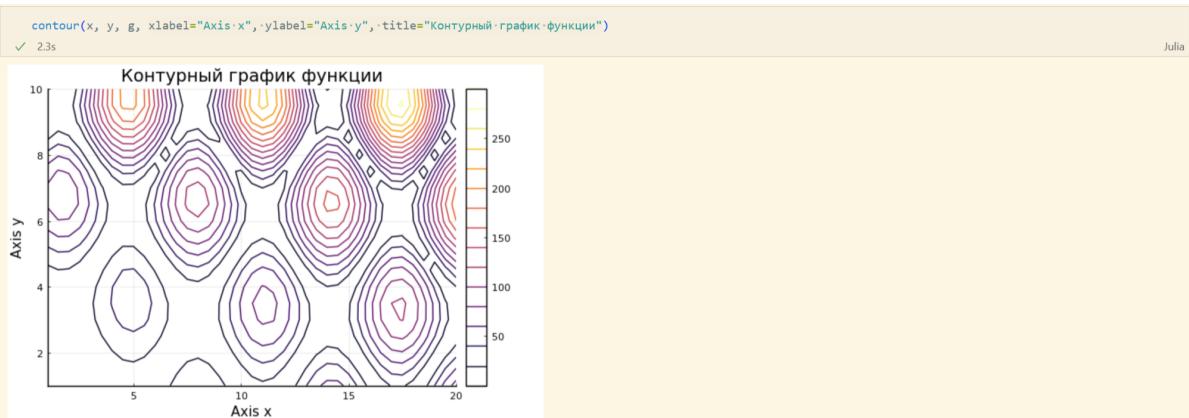


Рис. 2.25: Линии уровня

Можно дополнительно добавить заливку цветом (рис. 2.26):

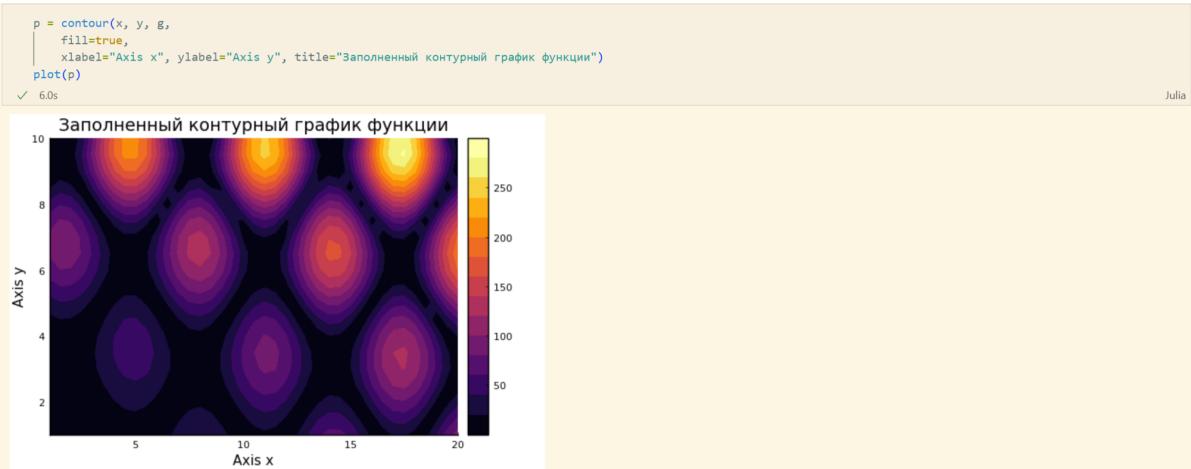


Рис. 2.26: Линии уровня с заполнением

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 $h(x, y) = x^3 - 3x + y^2$ сначала построим её график (рис. 2.27) и линии уровня (рис. 2.28):

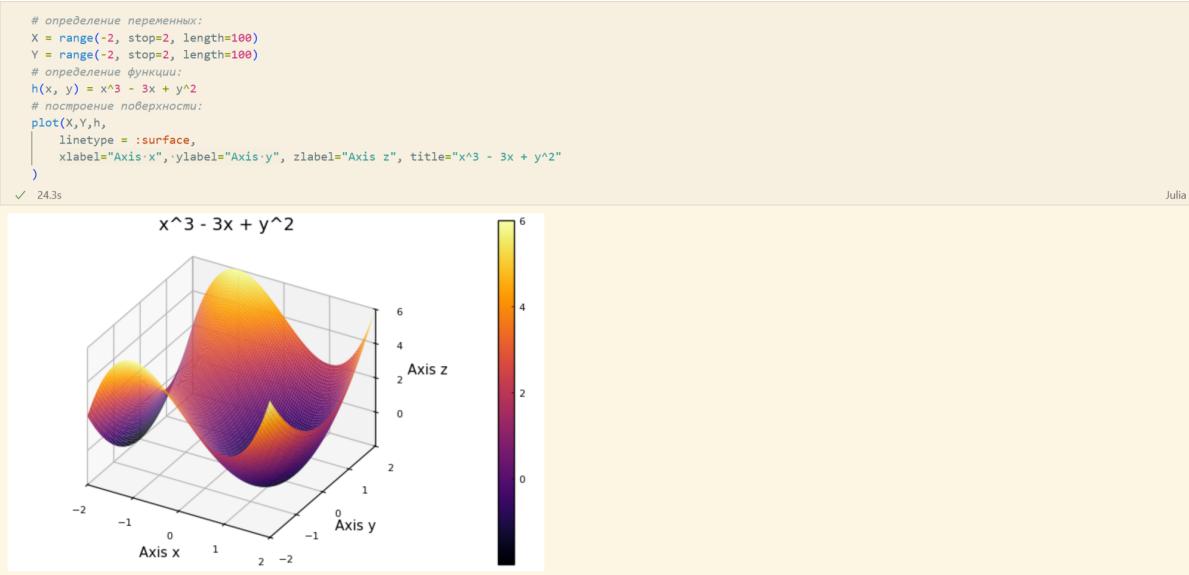


Рис. 2.27: График функции $h(x, y) = x^3 - 3x + y^2$

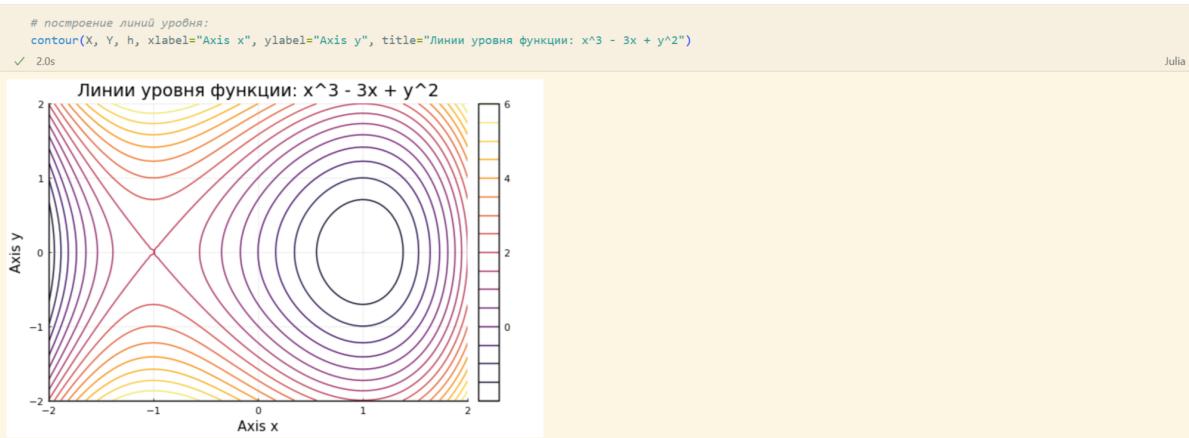


Рис. 2.28: Линии уровня для функции $h(x, y) = x^3 - 3x + y^2$

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. 2.29):

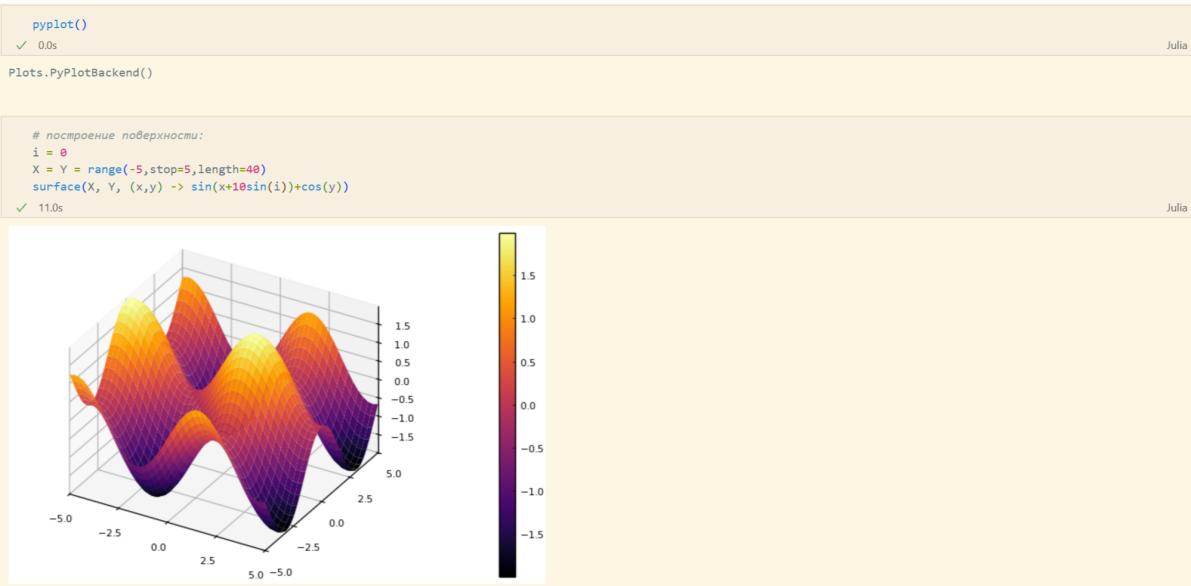


Рис. 2.29: Статичный график поверхности

Добавляем анимацию (рис. 2.30):

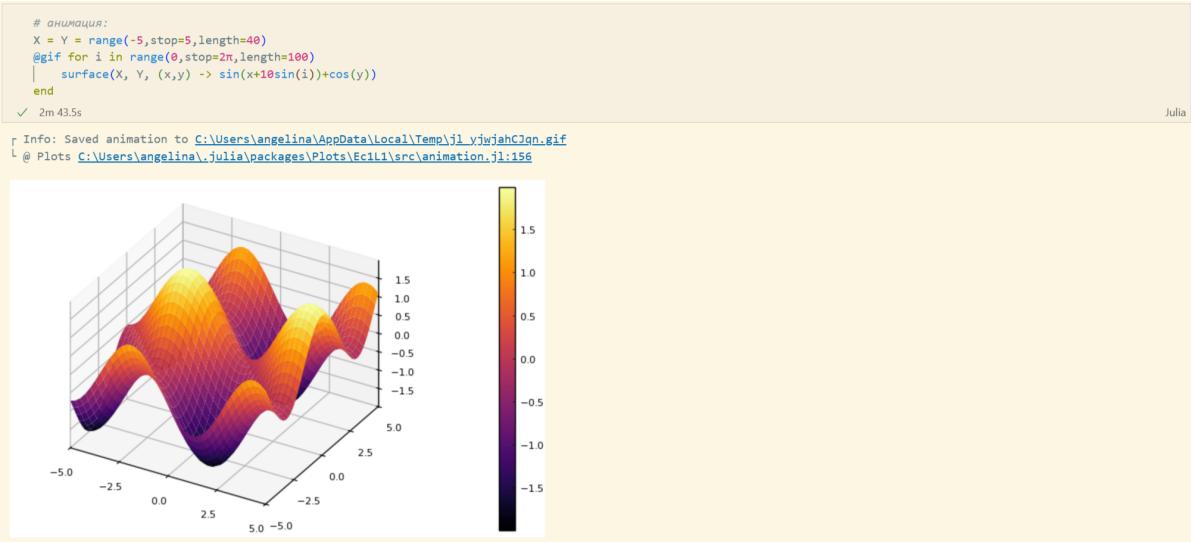


Рис. 2.30: Анимированный график поверхности

2.12 Гипоциклоида

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

Построим большую окружность (рис. 2.31 - рис. 2.32):

```
# радиус малой окружности:
r_1 = 1
# коэффициент для построения большой окружности:
k = 3
# число отсчётов:
n = 100
✓ 0.0s
100

# массив значений угла θ:
# theta from 0 to 2pi ( + a little extra)
θ = collect(0:2π/100:2π+2π/100)
# массивы значений координат:
X = r_1*k*cos.(θ)
Y = r_1*k*sin.(θ)
✓ 0.0s
101-element Vector{Float64}:
0.0
0.18837155858794014
0.37599970069529128
0.5621439437571739
0.7460696614945643
0.9270509831248421
1.104373658054034
1.2773378746952182
1.445261822305146
1.60748038493699
;
-1.445261022305146
-1.2773378746952186
-1.1043736580540335
-0.9270509831248404
-0.7460696614945634
-0.562143943757174
-0.375999700695291133
-0.1883715585879398
1.929747179611964e-15
Julia
```

Рис. 2.31: Построение большой окружности гипоциклоиды

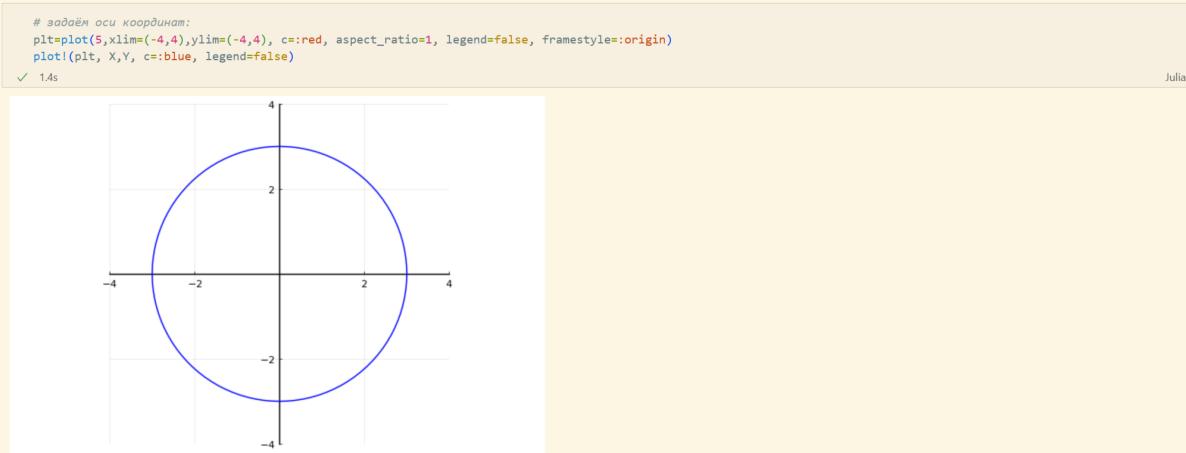


Рис. 2.32: Большая окружность гипоциклоиды

Для частичного построения гипоциклоиды будем менять параметр t (рис. 2.33):

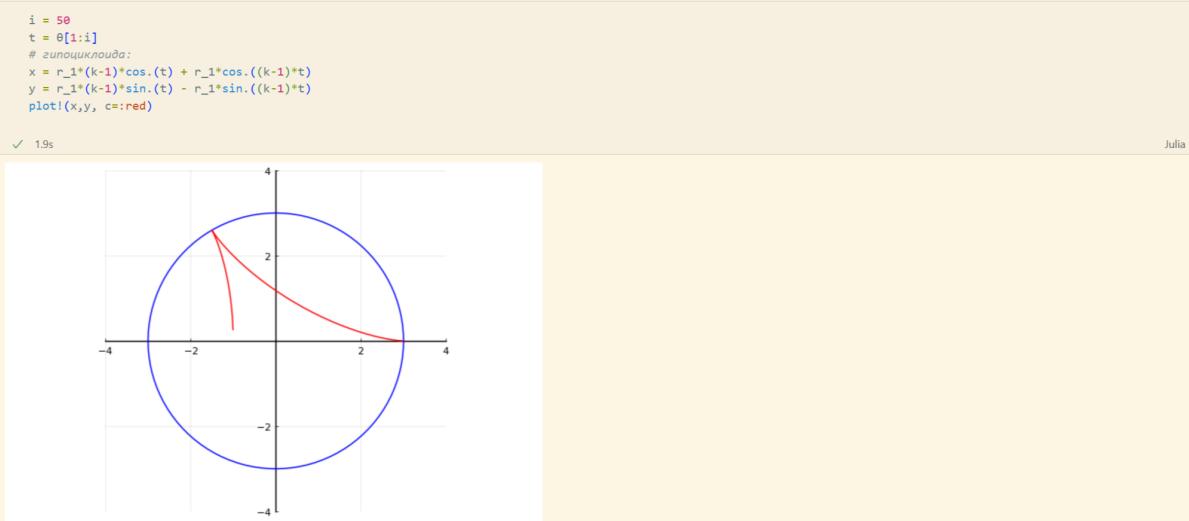


Рис. 2.33: Половина пути гипотрохойды

Добавляем малую окружность гипотрохойды (рис. 2.34):

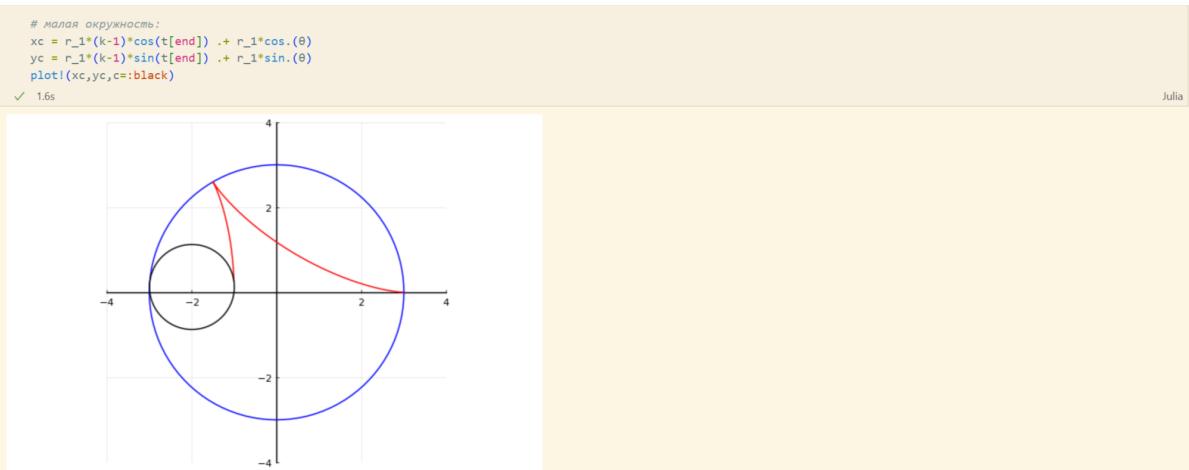


Рис. 2.34: Малая окружность гипотрохойды

Добавим радиус для малой окружности (рис. 2.35):

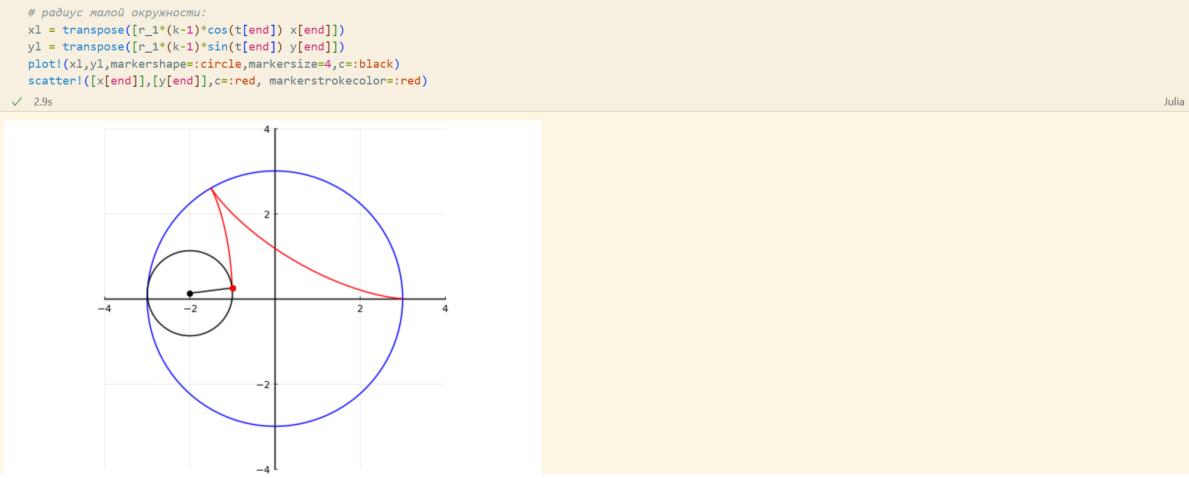


Рис. 2.35: Малая окружность гипотрохоиды с добавлением радиуса

2.13 Errorbars

В исследованиях часто требуется изобразить графики погрешностей измерения.

Построим график исходных значений (рис. 2.36 - рис. 2.37):

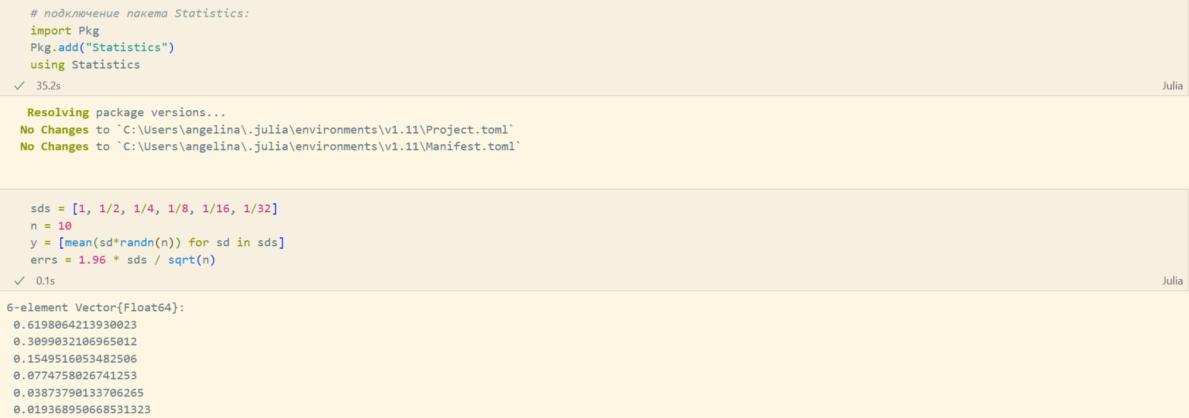


Рис. 2.36: Зададим исходные значения

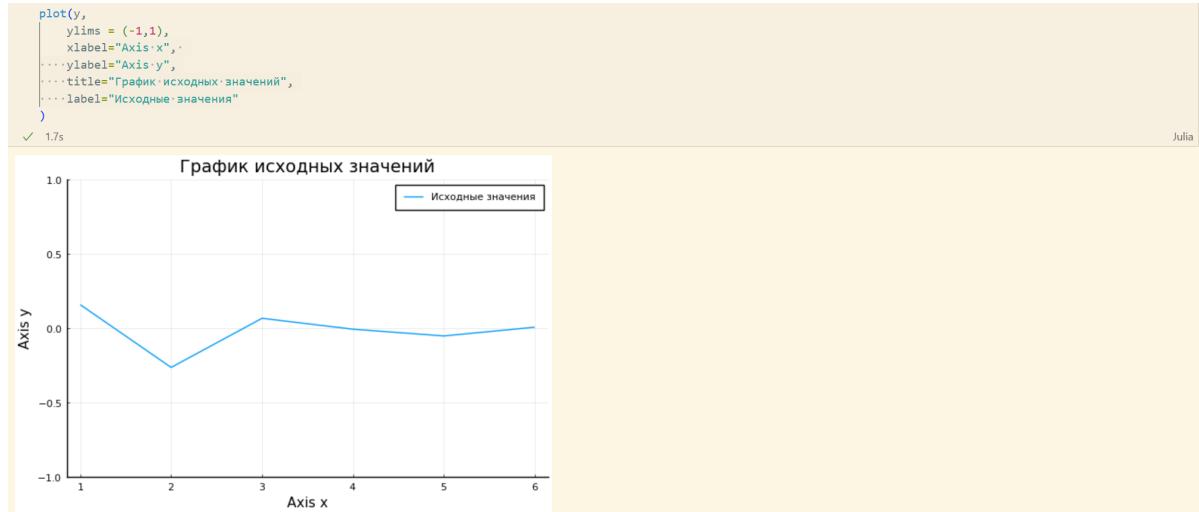


Рис. 2.37: График исходных значений

Построим график отклонений от исходных значений (рис. 2.38):

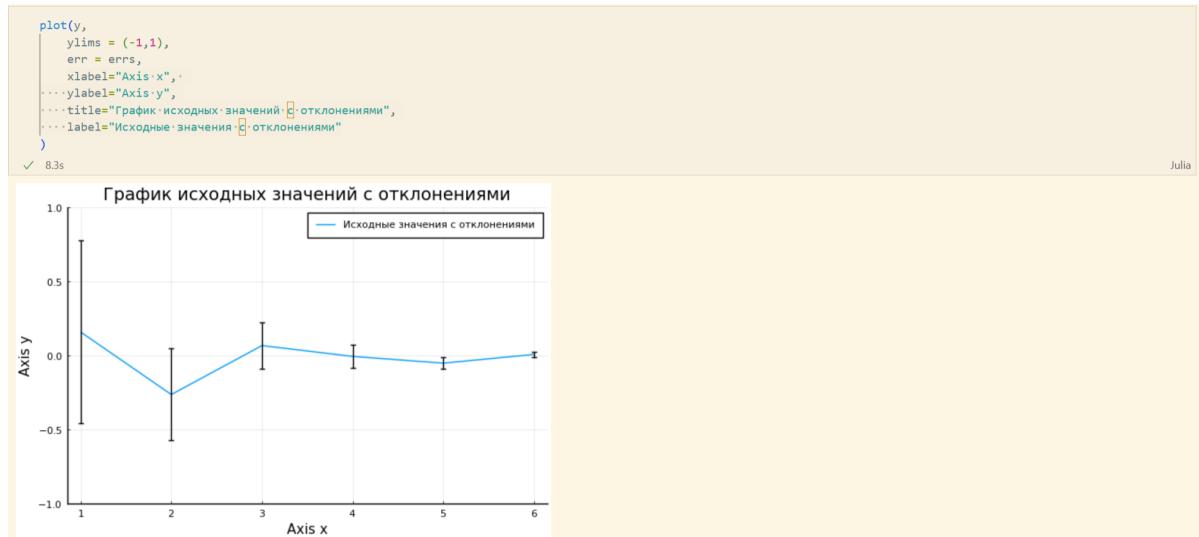


Рис. 2.38: График исходных значений с отклонениями

Повернём график (рис. 2.39):

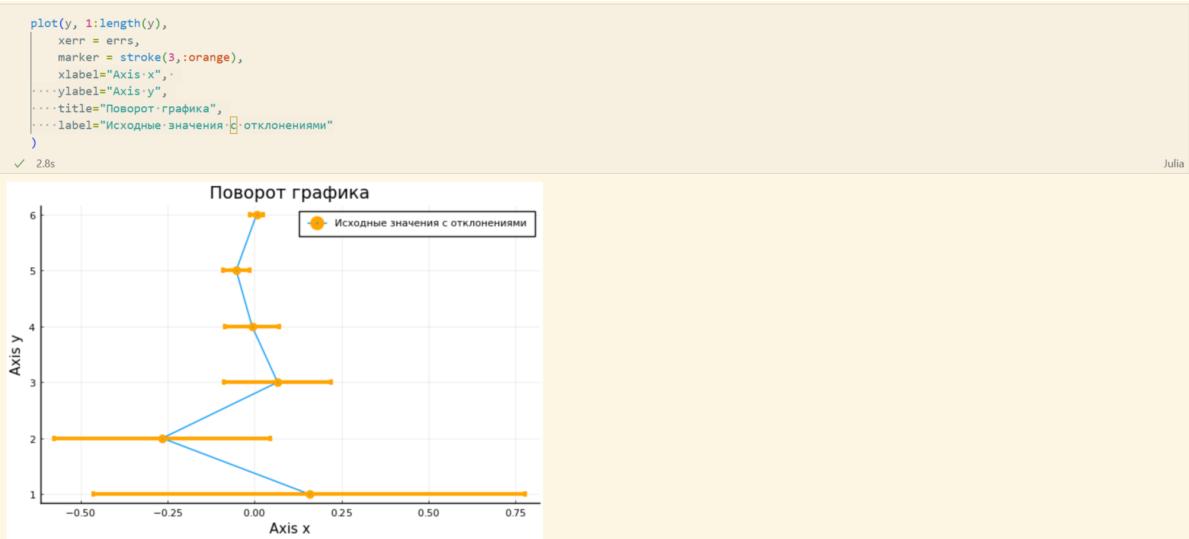


Рис. 2.39: Поворот графика

Заполним область цветом (рис. 2.40):

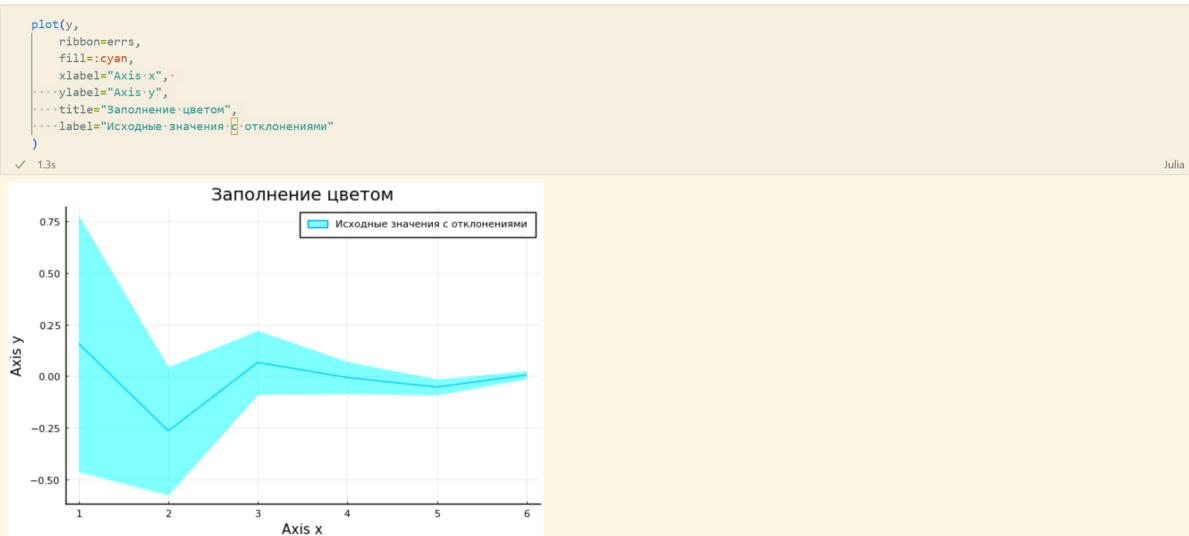


Рис. 2.40: Заполнение цветом

Можно построить график ошибок по двум осям (рис. 2.41):



Рис. 2.41: График ошибок по двум осям

Можно построить график асимметричных ошибок по двум осям (рис. 2.42):

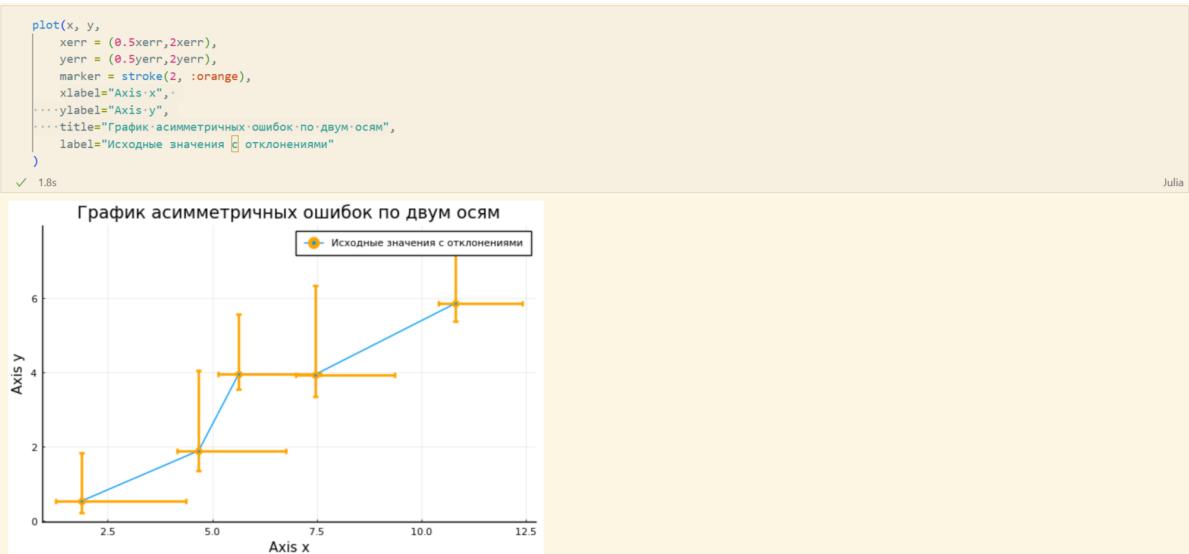


Рис. 2.42: График асимметричных ошибок по двум осям

2.14 Использование пакета Distributions

Строим гистограмму (рис. 2.43):



Рис. 2.43: Гистограмма, построенная по массиву случайных чисел

Задаём нормальное распределение и строим гистограмму (рис. 2.44):

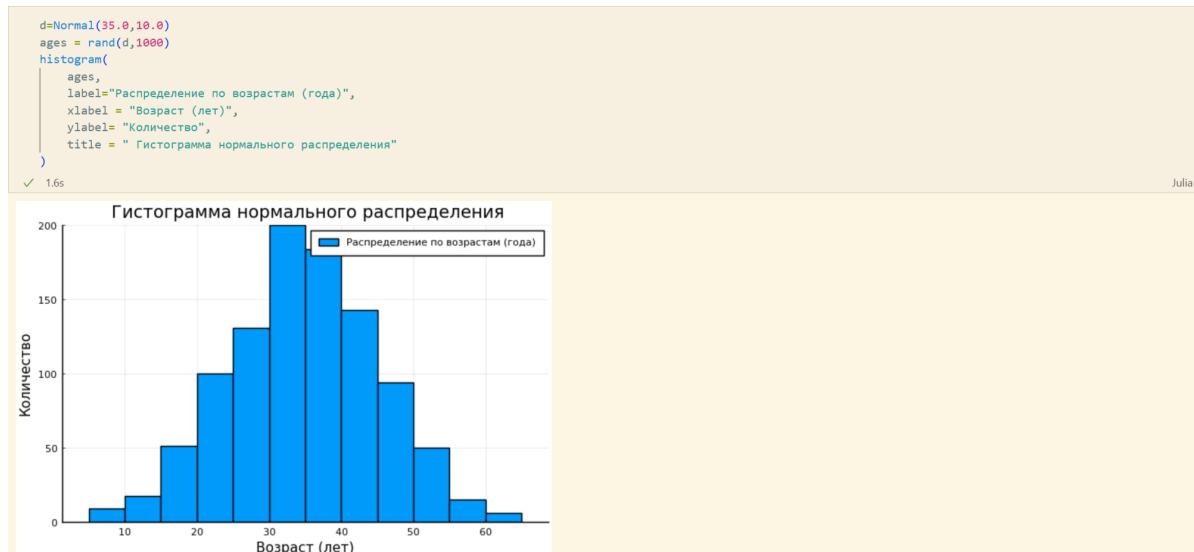


Рис. 2.44: Гистограмма нормального распределения

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. 2.41):

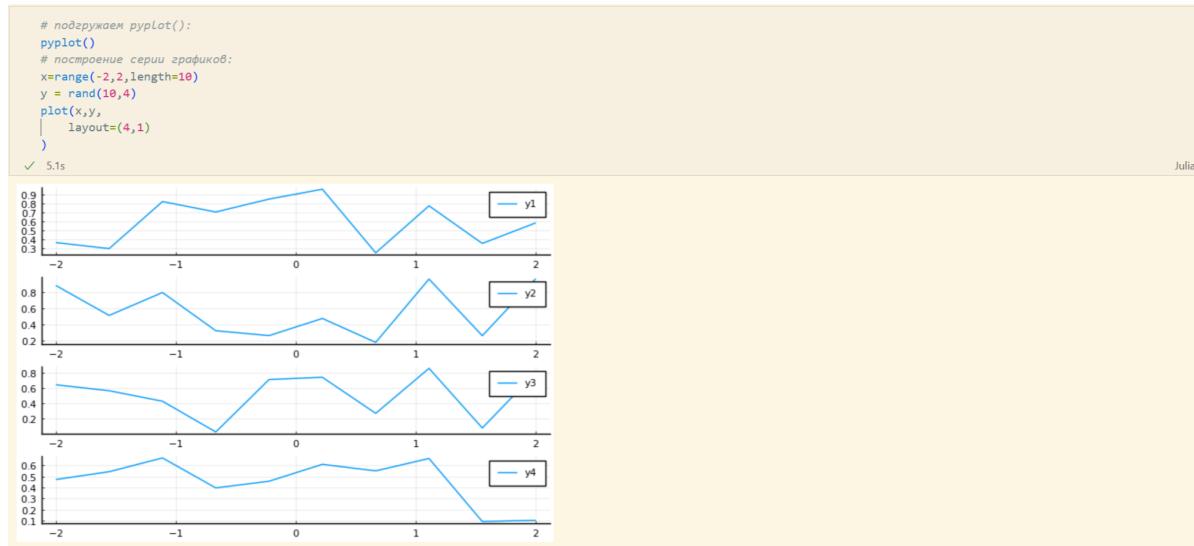


Рис. 2.45: Серия из 4-х графиков в ряд

Для автоматического вычисления сетки необходимо передать `layout` целое число (рис. 2.46):



Рис. 2.46: 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 (рис. 2.47).

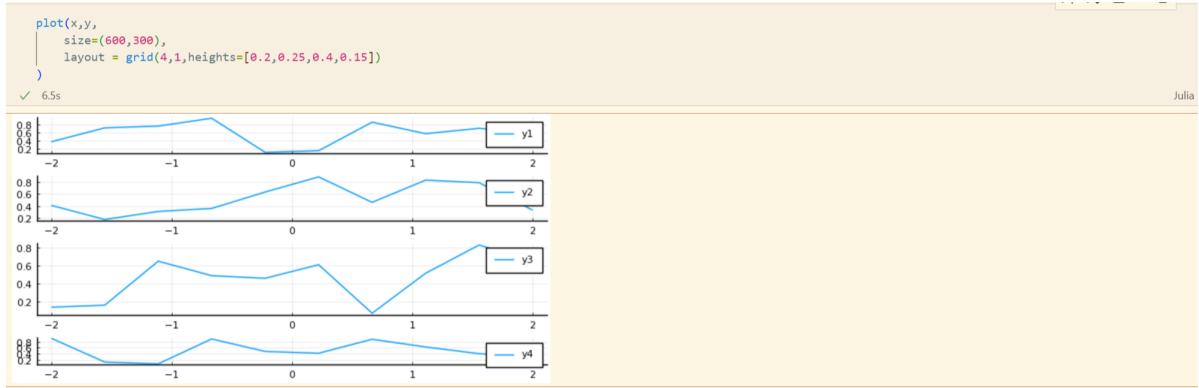


Рис. 2.47: Серия из 4-х графиков разной высоты в ряд

Можно сгенерировать отдельные графики и объединить их в один, например, в сетке 2×2 (рис. 2.48 - рис. 2.49):



Рис. 2.48: Объединение нескольких графиков в одной сетке

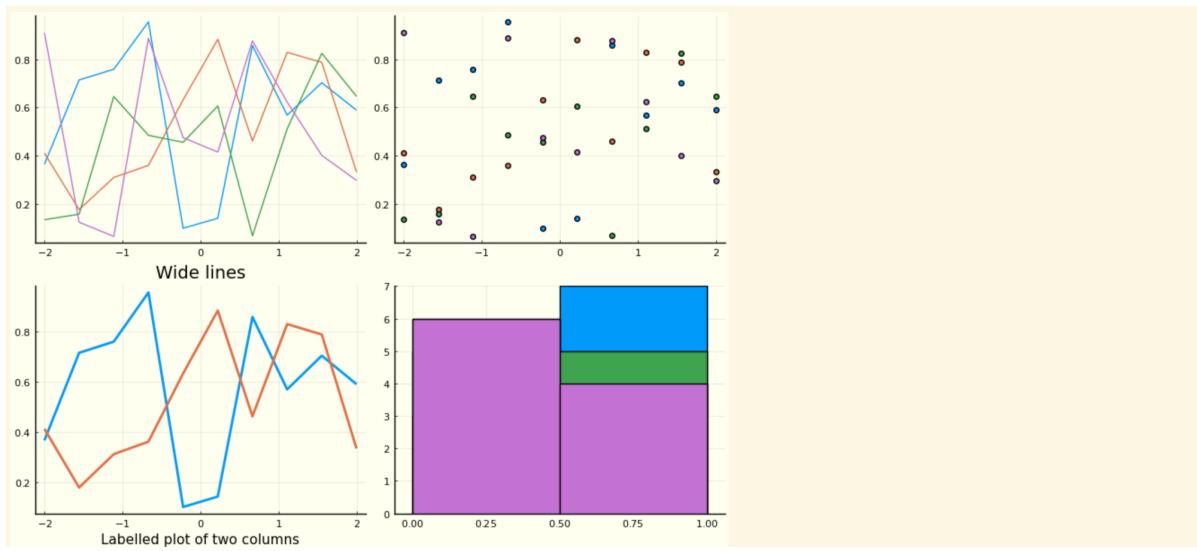


Рис. 2.49: Объединение нескольких графиков в одной сетке

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. 2.50):



Рис. 2.50: Разнообразные варианты представления данных

Применение макроса 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. 2.51):



Рис. 2.51: Демонстрация применения сложного макета для построения графиков

2.16 Самостоятельное выполнение

Выполнение задания №1 (рис. 2.52):

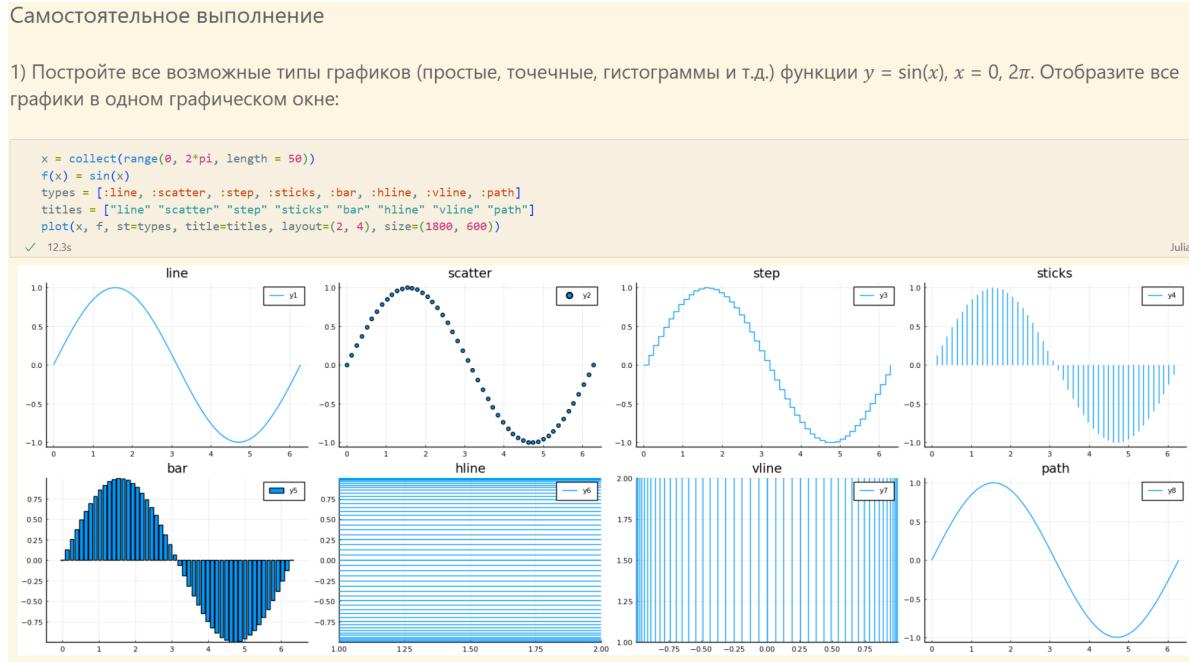


Рис. 2.52: Решение задания №1

Выполнение задания №2 (рис. 2.53):

2) Постройте графики функции $y = \sin(x)$, $x = 0, 2\pi$ со всеми возможными (сколько сможете вспомнить) типами оформления линий графика. Отобразите все графики в одном графическом окне:

```
titles = ["solid" "dash" "dot" "dashdot"]
p1 = Plots.plot(x, f, linestyle:solid)
p2 = Plots.plot(x, f, linestyle:dash)
p3 = Plots.plot(x, f, linestyle:dot)
p4 = Plots.plot(x, f, linestyle:dashdot)
Plots.plot(p1, p2, p3, p4, title=titles, layout=(2, 2))

```

✓ 28.4s

Julia

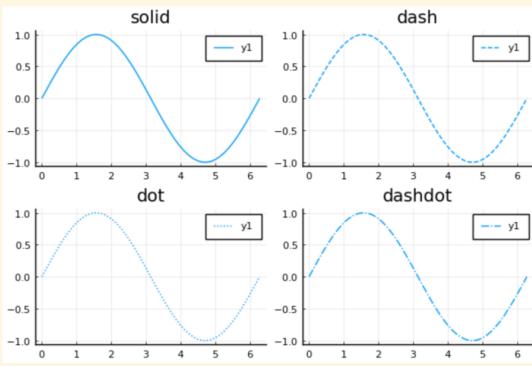


Рис. 2.53: Решение задания №2

Выполнение задания №3 (рис. 2.54):

3. Постройте график функции $y(x) = \pi x^2 \ln(x)$, назовите оси соответственно. Пусть цвет рамки будет зелёным, а цвет самого графика — красным. Задайте расстояние между надписями и осями так, чтобы надписи полностью умещались в графическом окне. Задайте шрифт надписей. Задайте частоту отметок на осях координат:

```
f(x) = pi * x^2 * log(x)
x = 0.1:0.01:6
plot(x, f.(x),
    color = :red,                      # Цвет графика
    label = "y(x) = pi*x^2*ln(x)",     # Легенда
    xlabel = "x",                        # Подпись оси X
    ylabel = "y(x) = pi*x^2*ln(x)",     # Подпись оси Y
    framestyle = :box,                  # Стиль рамки
    xticks = 0:0.5:6,                   # Частота отметок на оси X
    yticks = -50:50:200,                # Частота отметок на оси Y
    bordercolor = :green)               # Цвет рамки

```

✓ 6.1s

Julia

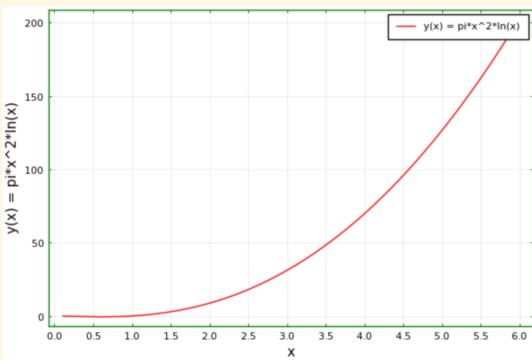


Рис. 2.54: Решение задания №3

Выполнение задания №4 (рис. 2.55):

4. Задайте вектор $x = (-2, -1, 0, 1, 2)$. В одном графическом окне (в 4-х подокнах) изобразите графически по точкам x значения функции $y(x) = x^3 - 3x$ в виде: точек, линий, линий и точек, кривой. Сохраните полученные изображения в файле figure_familiya.png, где вместо familiya укажите вашу фамилию:

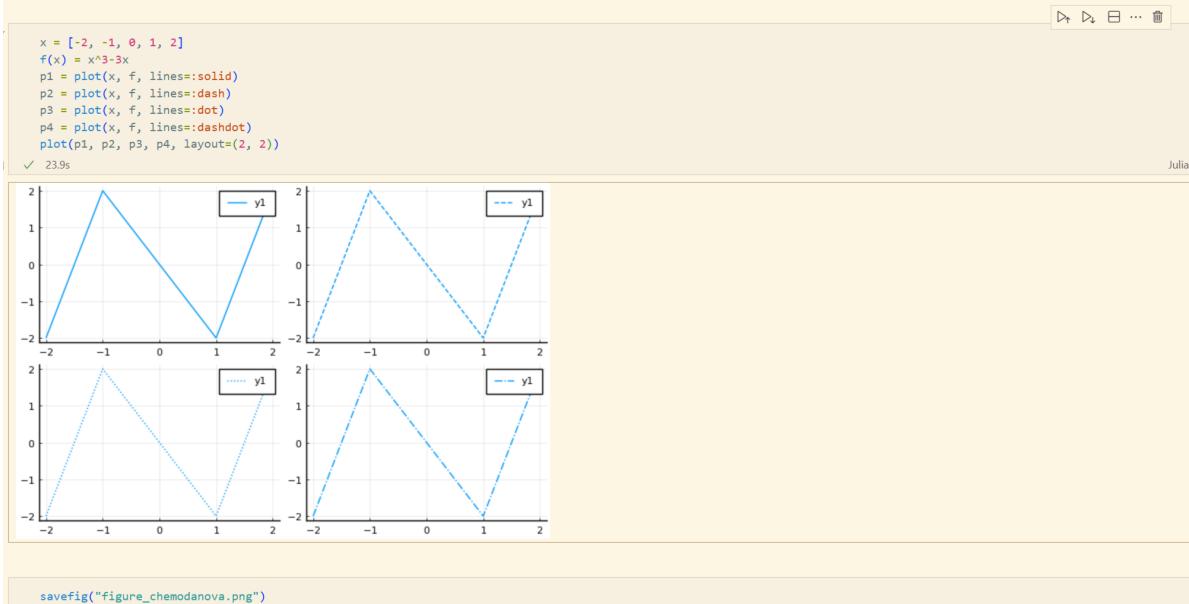


Рис. 2.55: Решение задания №4

Выполнение задания №5 (рис. 2.56 - рис. 2.57):

5. Задайте вектор $x = (3, 3.1, 3.2, \dots, 6)$. Постройте графики функций $y1(x) = \pi x$ и $y2(x) = \exp(x) \cos(x)$ в указанном диапазоне значений аргумента x следующим образом: постройте оба графика разного цвета на одном рисунке, добавьте легенду и сетку для каждого графика; укажите недостатки у данного построения; постройте аналогичный график с двумя осями ординат:

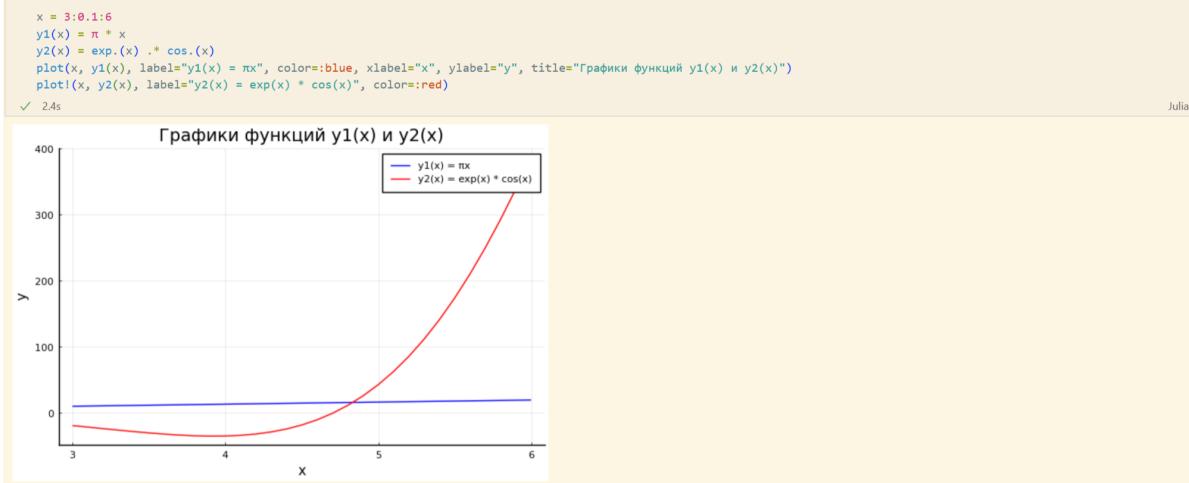


Рис. 2.56: Решение задания №5

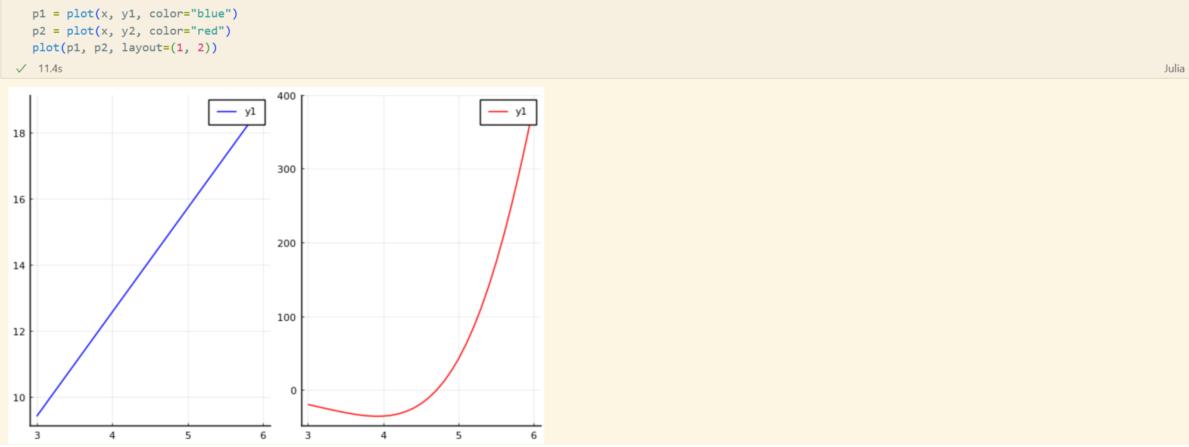


Рис. 2.57: Решение задания №5

Выполнение задания №6 (рис. 2.58):

6. Постройте график некоторых экспериментальных данных (придумайте сами), учитывая ошибку измерения:

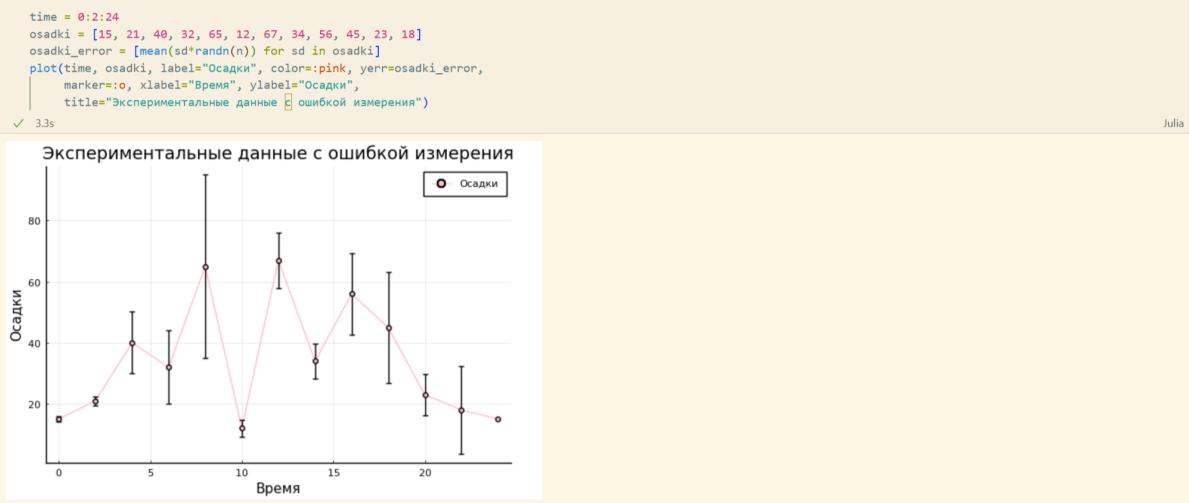


Рис. 2.58: Решение задания №6

Выполнение задания №7 (рис. 2.59):

7. Постройте точечный график случайных данных. Подпишите оси, легенду, название графика:

```
x := rand(1:10, 20)
y := rand(1:10, 20)
scatter(x, y, label="Случайные данные", color="pink", marker=:o,
| xlabel="x", ylabel="y", title="Точечный график случайных данных")
✓ 10.3s
```

Julia

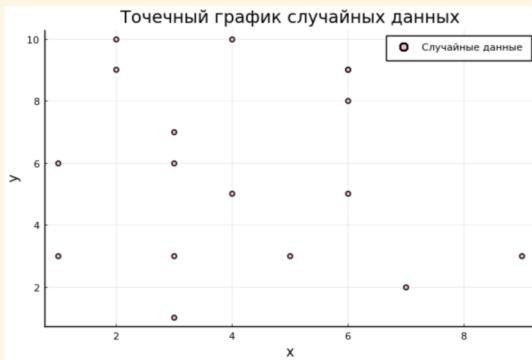


Рис. 2.59: Решение задания №7

Выполнение задания №8 (рис. 2.60):

8. Постройте 3-мерный точечный график случайных данных. Подпишите оси, легенду, название графика:

```
x = rand(1:10, 20)
y = rand(1:10, 20)
z = rand(1:10, 20)
scatter(x, y, z, label="Случайные данные", color="red", marker=:o,
| xlabel="x", ylabel="y", zlabel="z", title="3-мерный точечный график случайных данных")
✓ 2.6s
```

Julia



Рис. 2.60: Решение задания №8

Выполнение задания №9 (рис. 2.61):

9. Создайте анимацию с построением синусоиды. То есть вы строите последовательность графиков синусоиды, постепенно увеличивая значение аргумента. После соедините их в анимацию:

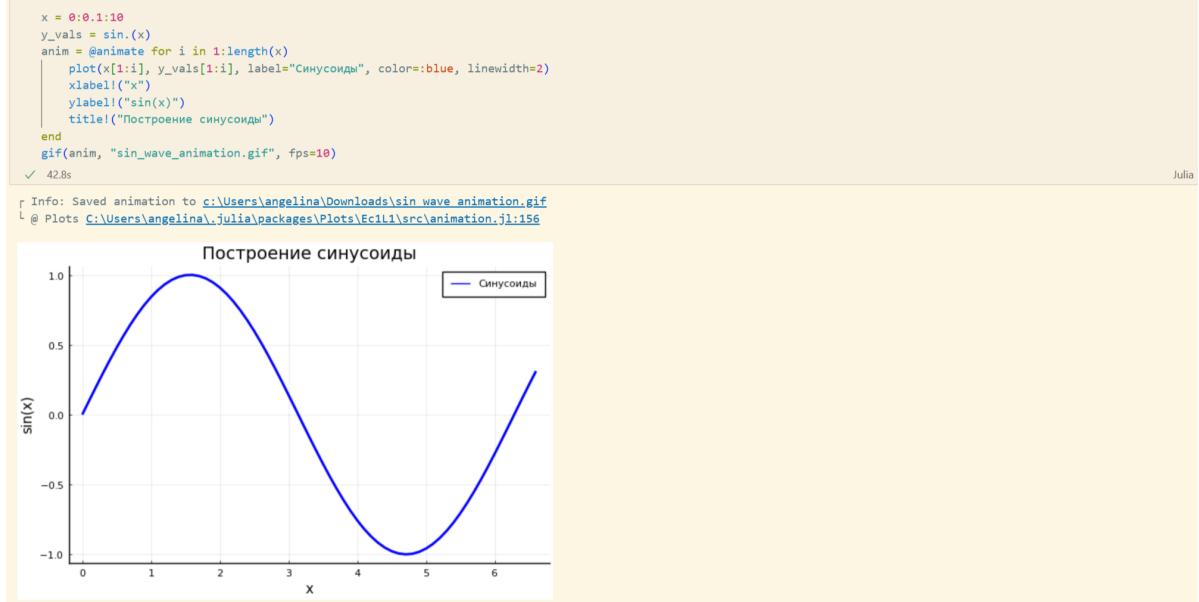


Рис. 2.61: Решение задания №9

Выполнение задания №10 (рис. 2.62 - рис. 2.67):

10. Постройте анимированную гипоциклоиду для 2 целых значений модуля k и 2 рациональных значений модуля k :



Рис. 2.62: Решение задания №10

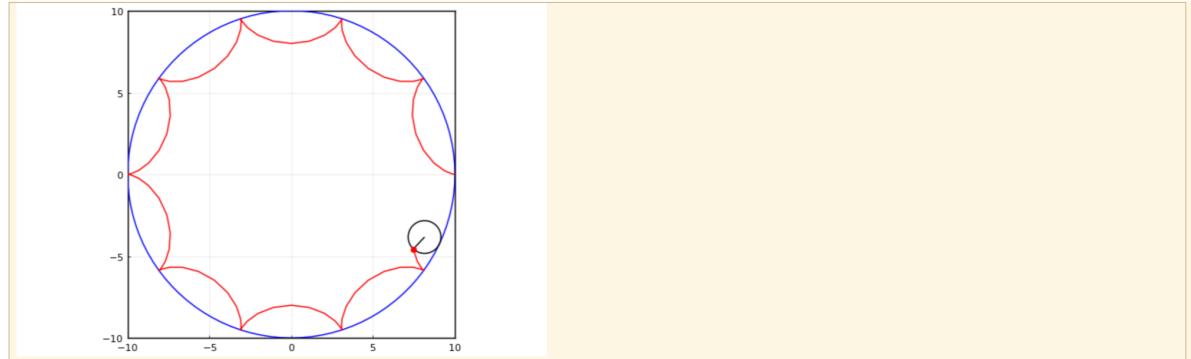


Рис. 2.63: Решение задания №10



Рис. 2.64: Решение задания №10

```

rr = 1
k = 19//3
n=100
theta = collect(0:2*pi/100:2*pi+2*pi/100)
X = rr*k*cos.(theta)
Y = rr*k*sin.(theta)
anim = @animate for i in 1:n
    plt=plot(5, xlim=(-10,10), ylim=(-10, 10), c="red", aspect_ratio=1, legend=false, framestyle="origin")
    plot!(plt, X, Y, c="blue", legend=false)
    t = theta[1:i]
    x = rr*(k-1)*cos.(t) + rr*cos.((k-1)*t)
    y = rr*(k-1)*sin.(t) - rr*sin.((k-1)*t)
    plot!(x, y, c="red")
    xc = rr*(k-1)*cos(t[end]) .+ rr*cos.(theta)
    yc = rr*(k-1)*sin(t[end]) .- rr*sin.(theta)
    plot!(xc, yc, c="black")
    xl = transpose([rr*(k-1)*cos(t[end]) x[end]])
    yl = transpose([rr*(k-1)*sin(t[end]) y[end]])
    plot!(xl, yl, markershape="circle", markersize=4, c="black")
    scatter!([x[end]], [y[end]], c="red", markerstrokecolor="red")
end
gif(anim, "hypocycloid.gif")

```

Julia

Рис. 2.65: Решение задания №10

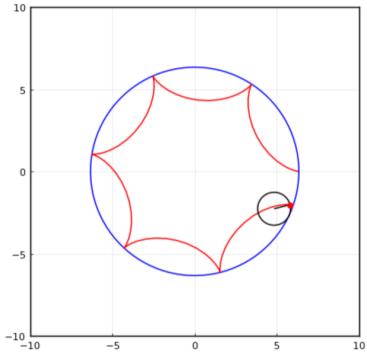


Рис. 2.66: Решение задания №10

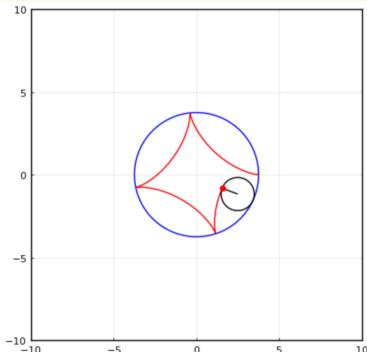


Рис. 2.67: Решение задания №10

Выполнение задания №11 (рис. 2.68 - рис. 2.73):

11. Постройте анимированную эпициклоиду для 2 целых значений модуля k и 2 рациональных значений модуля k :

```
rr = 1
k = 10
n=100
theta = collect(0:2*pi/100:2*pi+2*pi/100)
X = rr*k*cos.(theta)
Y = rr*k*sin.(theta)
anim = @animate for i in 1:n
    plt=plot(S, xlim=(-10,10), ylim=(-10, 10), c="red", aspect_ratio=1, legend=false, framestyle="origin")
    plot!(plt, X, Y, c="blue", legend=false)
    t = theta[1:i]
    x = rr*(k-1)*cos.(t) - rr*cos.((k-1)*t)
    y = rr*(k-1)*sin.(t) - rr*sin.((k-1)*t)
    plot!(x, y, c="red")
    xc = rr*(k-1)*cos(t[end]) .+ rr*cos.(theta)
    yc = rr*(k-1)*sin(t[end]) .+ rr*sin.(theta)
    plot!(xc, yc, c="black")
    x1 = transpose([rr*(k-1)*cos(t[end]) x[end]])
    y1 = transpose([rr*(k-1)*sin(t[end]) y[end]])
    plot!(x1, y1, markershape="circle", markersize=4, c="black")
    scatter!([x1], [y1], c="red", markerstrokecolor="red")
end
gif(anim, "hypocycloid.gif")
```

Julia

Рис. 2.68: Решение задания №11

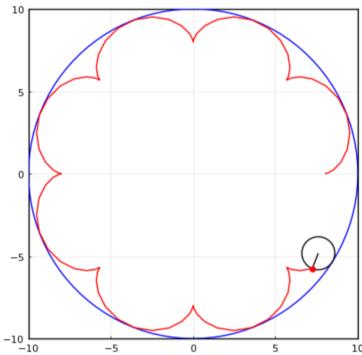


Рис. 2.69: Решение задания №11

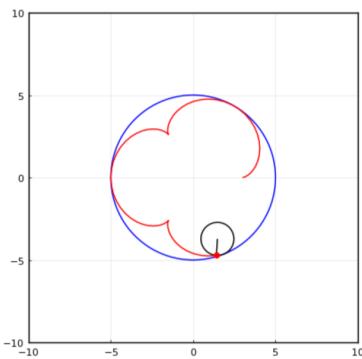


Рис. 2.70: Решение задания №11

```

rr = 1
k = 19//3
n=100
theta = collect(0:2*pi/100:2*pi+2*pi/100)
X = rr*k*cos.(theta)
Y = rr*k*sin.(theta)
anim = @animate for i in 1:n
    plt=plot(5, xlim=(-10,10), ylim=(-10, 10), c="red", aspect_ratio=1, legend=false, framestyle="origin")
    plot!(plt, X, Y, c="blue", legend=false)
    t = theta[1:i]
    x = rr*(k-1)*cos.(t) - rr*cos.((k-1)*t)
    y = rr*(k-1)*sin.(t) - rr*sin.((k-1)*t)
    plot!(x, y, c="red")
    xc = rr*(k-1)*cos(t[end]) .+ rr*cos.(theta)
    yc = rr*(k-1)*sin(t[end]) .+ rr*sin.(theta)
    plot!(xc, yc, c="black")
    xl = transpose([rr*(k-1)*cos(t[end]), x[end]])
    yl = transpose([rr*(k-1)*sin(t[end]), y[end]])
    plot!(xl, yl, makershape="circle", markersize=4, c="black")
    scatter!([(x[end]), (y[end]), c="red", markerstrokecolor="red"])
end
gif(anim, "hypocycloid.gif")
✓ 46.3s

```

Julia

Рис. 2.71: Решение задания №11



Рис. 2.72: Решение задания №11



Рис. 2.73: Решение задания №11

3 Выводы

В результате выполнения данной лабораторной работы мы освоили синтаксис языка Julia для построения графиков.

Список литературы

1. JuliaLang [Электронный ресурс]. 2025 JuliaLang.org contributors. URL: <https://julialang.org/> (дата обращения: 09.16.2025).
2. Julia 1.11 Documentation [Электронный ресурс]. 2025 JuliaLang.org contributors. URL: <https://docs.julialang.org/en/v1/> (дата обращения: 09.16.2025).