

Лабораторная работа №5

Компьютерный практикум по статистическому анализу данных

Еюбоглу Тимур

Содержание

1 Цель работы	6
2 Выполнение лабораторной работы	7
2.1 Основные пакеты для работы с графиками в Julia	7
2.2 Опции при построении графика	9
2.3 Точечный график	13
2.4 Аппроксимация данных	16
2.5 Две оси ординат	18
2.6 Полярные координаты	19
2.7 Параметрический график	20
2.8 График поверхности	22
2.9 Линии уровня	26
2.10 Векторные поля	29
2.11 Анимация	31
2.12 Гипоциклоида	33
2.13 Errorbars	38
2.14 Использование пакета Distributions	42
2.15 Подграфики	45
2.16 Самостоятельное выполнение	50
3 Вывод	61
4 Список литературы. Библиография	62

Список иллюстраций

2.1 График функции, построенный при помощи gr()	8
2.2 График функции, построенный при помощи pyplot()	9
2.3 График функции $\sin(x)$	10
2.4 График функции разложения исходной функции в ряд Тейлора	11
2.5 Графики исходной функции и её разложения в ряд Тейлора	12
2.6 Вид графиков после добавления опций при их построении	13
2.7 График десяти случайных значений на плоскости (простой точечный график)	14
2.8 График пятидесяти случайных значений на плоскости с различными опциями отображения (точечный график с кодированием значения размером точки)	15
2.9 График пятидесяти случайных значений в пространстве с различными опциями отображения (3-мерный точечный график с кодированием значения размером точки)	16
2.10 Пример функции	17
2.11 Пример аппроксимации исходной функции полиномом 5-й степени	18
2.12 Пример отдельно построенной траектории	19
2.13 График функции, заданной в полярных координатах	20
2.14 Параметрический график кривой на плоскости	21
2.15 Параметрический график кривой в пространстве	22
2.16 График поверхности (использована функция surface())	23
2.17 График поверхности (использована функция plot())	24
2.18 Сглаженный график поверхности	25
2.19 График поверхности с изменённым углом зрения	26
2.20 График поверхности, заданной функцией $g(x, y) = (3x + y^2) \sin(x) + \cos(y) $	27
2.21 Линии уровня	28
2.22 Линии уровня с заполнением	29
2.23 График функции $h(x, y) = x^3 - 3x + y^2$	30
2.24 Линии уровня для функции $h(x, y) = x^3 - 3x + y^2$	31
2.25 Статичный график поверхности	32
2.26 Анимированный график поверхности	33
2.27 Большая окружность гипоциклоиды	34
2.28 Половина пути гипоциклоиды	35
2.29 Малая окружность гипоциклоиды	36
2.30 Малая окружность гипоциклоиды с добавлением радиуса	37
2.31 Малая окружность гипоциклоиды с добавлением радиуса	37

2.32 График исходных значений	38
2.33 График исходных значений с отклонениями	39
2.34 Поворот графика	40
2.35 Заполнение цветом	40
2.36 График ошибок по двум осям	41
2.37 График асимметричных ошибок по двум осям	42
2.38 Гистограмма, построенная по массиву случайных чисел	43
2.39 Гистограмма нормального распределения	44
2.40 Гистограмма распределения людей по возрастам	45
2.41 Серия из 4-х графиков в ряд	46
2.42 Серия из 4-х графиков в сетке	47
2.43 Объединение нескольких графиков в одной сетке	48
2.44 Разнообразные варианты представления данных	49
2.45 Демонстрация применения сложного макета для построения гра- фиков	50
2.46 Решение задания №1	51
2.47 Решение задания №2	52
2.48 Решение задания №3	53
2.49 Решение задания №4	54
2.50 Решение задания №5	55
2.51 Решение задания №5	55
2.52 Решение задания №6	56
2.53 Решение задания №7	57
2.54 Решение задания №8	57
2.55 Решение задания №9	58
2.56 Решение задания №10	59
2.57 Решение задания №11	60

Список таблиц

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 $f(x) = (3x^2 + 6x - 9)e^{-0,3x}$ разными способами (рис. 2.1 - (рис. 2.2):

1. Основные пакеты для работы с графиками в Julia

```
[1]: using Plots  
# Задание функции:  
f(x) = (3x.^2 + 6x .- 9).*exp.(-0.3x)  
# Генерация массива значений x в диапазоне от -5 до 10 с шагом 0,1:  
x = collect(range(-5, 10, length=151))  
# Генерация массива значений y:  
y = f(x)  
# Указываем, что для построения графика используется gr():  
gr()  
# Построение графика:  
plot(x, y,  
      title="Graph of the Function f(x)",           # Заголовок графика  
      xlabel="x-axis (Input)",                      # Подпись оси x  
      ylabel="y-axis (Output)",                     # Подпись оси y  
      label="f(x) = (3x^2 + 6x - 9)e^{-0.3x}",    # Легенда  
      color="blue",                                # Цвет графика  
      lw=2,                                         # Толщина линии  
      grid=true)                                    # Включение сетки
```

[1]:

Graph of the Function f(x)

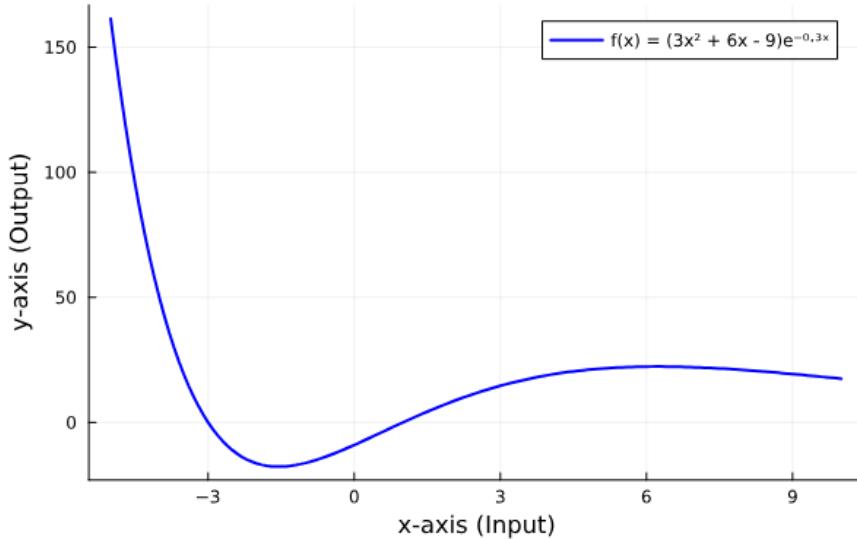


Рис. 2.1: График функции, построенный при помощи gr()

```
[2]: pyplot()
# Построение графика:
plot(x, y,
      title="График функции f(x)",           # Заголовок графика
      xlabel="Ось x (входные данные)",        # Подпись оси x
      ylabel="Ось у (выходные данные)",       # Подпись оси у
      label="f(x) = (3x2 + 6x - 9)e-0.3x",   # Легенда
      color="blue",                          # Цвет графика
      lw=2,                                 # Толщина линии
      grid=true)                           # Включение сетки
```

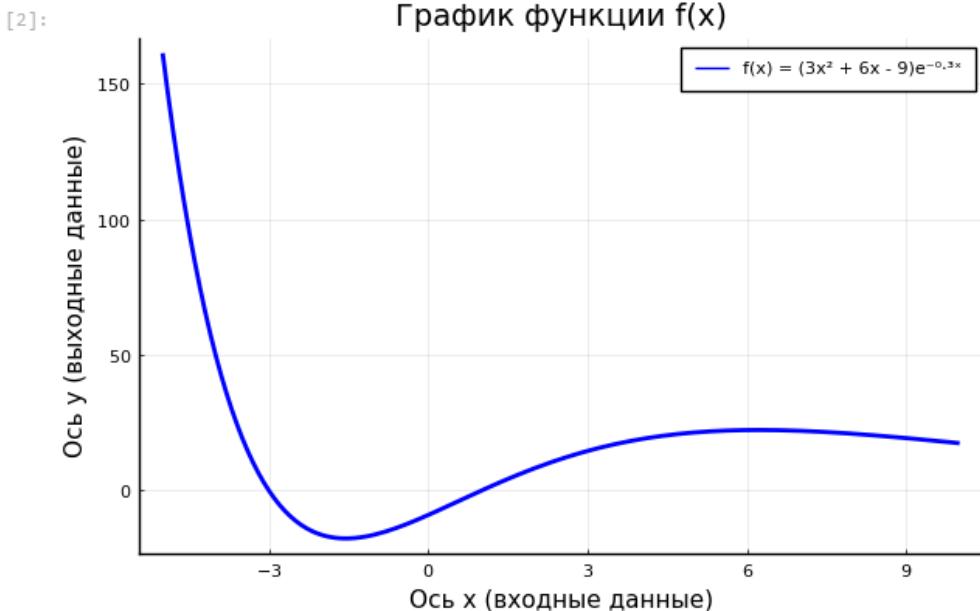


Рис. 2.2: График функции, построенный при помощи pyplot()

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. 2.3 - рис. 2.5):

▼ 2. Опции при построении графика

```
[3]: # Указываем, что для построения графика используется pyplot():
pyplot()
# Задание функции sin(x):
sin_theor(x) = sin(x)
# Построение графика функции sin(x):
plot(sin_theor,
     title="График функции sin(x)",           # Заголовок графика
     xlabel="Ось x (аргумент)",                 # Подпись оси x
     ylabel="Ось у (значение функции)",          # Подпись оси у
     label="sin(x)",                           # Легенда
     lw=2,                                     # Толщина линии
     color="blue",                             # Цвет линии
     grid=true)                                # Включение сетки
```

[3]:



Рис. 2.3: График функции $\sin(x)$

```
[4]: # Задание функции разложения исходной функции в ряд Тейлора:
sin_taylor(x) = [(-1)^i * x^(2*i+1) / factorial(2*i+1) for i in 0:4] |> sum
# Построение графика функции sin_taylor(x):
plot(sin_taylor,
      title="График разложения sin(x) в ряд Тейлора", # Заголовок графика
      xlabel="Ось x (аргумент)", # Подпись оси x
      ylabel="Ось у (значение функции)", # Подпись оси у
      label="Ряд Тейлора (первые 5 членов)", # Легенда
      lw=2, # Толщина линии
      color="red", # Цвет линии
      grid=true) # Включение сетки
```

[4]: График разложения $\sin(x)$ в ряд Тейлора

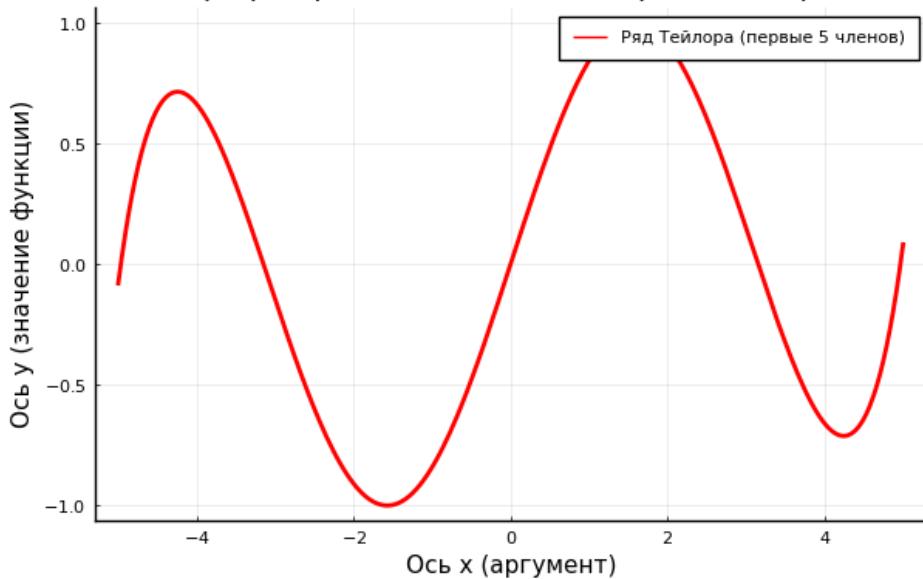


Рис. 2.4: График функции разложения исходной функции в ряд Тейлора

```
[5]: # построение двух функций на одном графике:
# Построение графика теоретической функции sin(x):
plot(sin_theor,
      label="sin(x)",
      lw=2,
      color="blue") # Легенда для sin(x)
# Толщина линии
# Цвет линии sin(x)

# Добавление графика разложения в ряд Тейлора:
plot!(sin_taylor,
      label="Ряд Тейлора (5 членов)", # Легенда для ряда Тейлора
      lw=2, # Толщина линии
      color="red") # Цвет линии для ряда Тейлора

# Добавление оформлений:
title!("Сравнение sin(x) и его ряда Тейлора") # Название графика
xlabel!("Ось x (аргумент)") # Подпись оси x
ylabel!("Ось y (значение функции)") # Подпись оси y
```

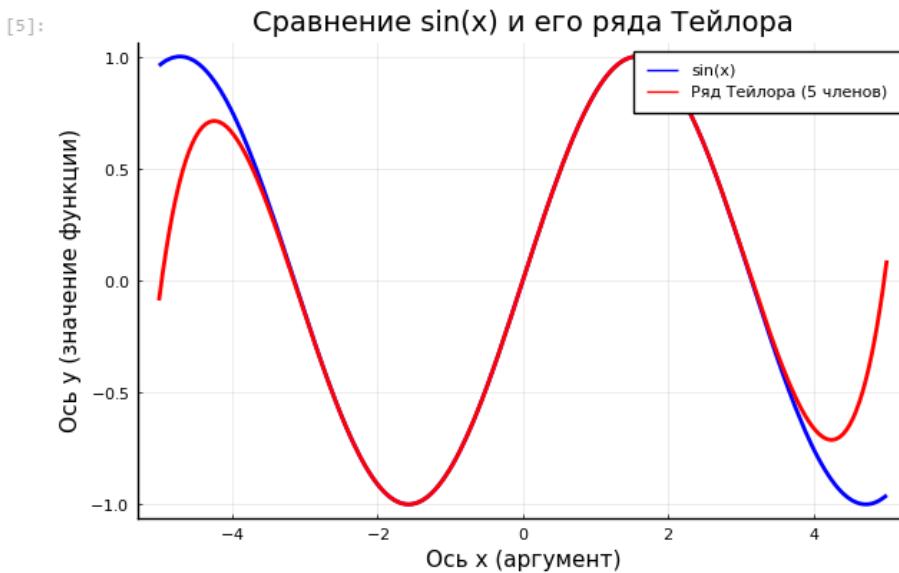


Рис. 2.5: Графики исходной функции и её разложения в ряд Тейлора

Затем добавим различные опции для отображения на графике (рис. 2.6):

```
[6]: 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, заданных 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))
```

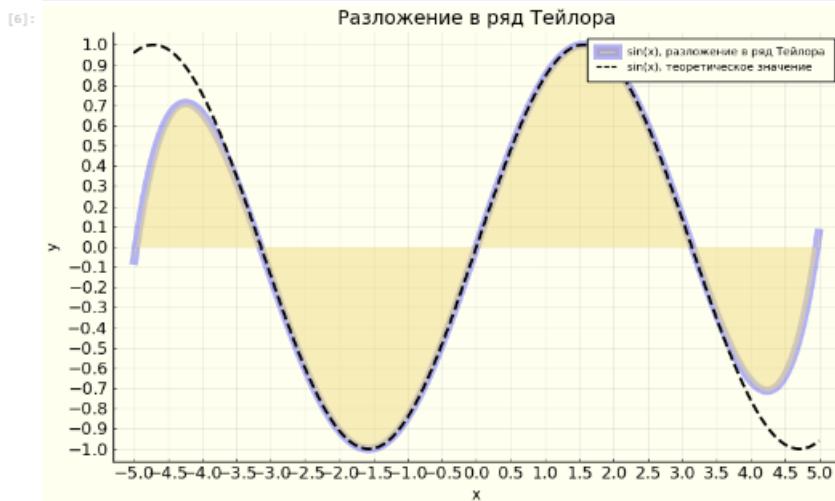


Рис. 2.6: Вид графиков после добавления опций при их построении

2.3 Точечный график

Как и при построении обычного графика для точечного графика необходимо задать массив значений \mathbb{X} , посчитать или задать значения y , задать опции построения графика (рис. 2.7):

3. Точечный график

3.1. Простой точечный график

```
[7]: # Параметры распределения точек на плоскости:  
x = range(1, 10, length=10)  
y = rand(10)  
# Параметры построения графика:  
plot(x, y,  
      seriestype = :scatter, # Тип графика: точки  
      title = "Точечный график распределения", # Название графика  
      xlabel = "Ось X (параметр x)", # Подпись оси X  
      ylabel = "Ось Y (случайные значения)", # Подпись оси Y  
      label = "Точки данных", # Легенда для точек  
      color = :blue, # Цвет точек  
      markersize = 8, # Размер маркеров  
      grid = true) # Включение сетки
```



Рис. 2.7: 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. 2.8):

3.2. Точечный график с кодированием значения размером точки

```
[8]: # Параметры распределения точек на плоскости:  
n = 50  
x = rand(n) # случайные значения для оси X  
y = rand(n) # случайные значения для оси Y  
ms = rand(50) * 30 # случайный размер маркеров  
# Параметры построения графика:  
scatter(x, y,  
        markersize = ms, # Размер маркеров зависит от случайных значений  
        title = "Точечный график с случайными размерами маркеров", # Название графика  
        xlabel = "Ось X (случайные значения)", # Подпись оси X  
        ylabel = "Ось Y (случайные значения)", # Подпись оси Y  
        label = "Точки данных", # Легенда для точек  
        color = :blue, # Цвет точек  
        grid = true, # Включение сетки  
        legend = :topright) # Размещение легенды в правом верхнем углу
```

[8]: Точечный график с случайными размерами маркеров

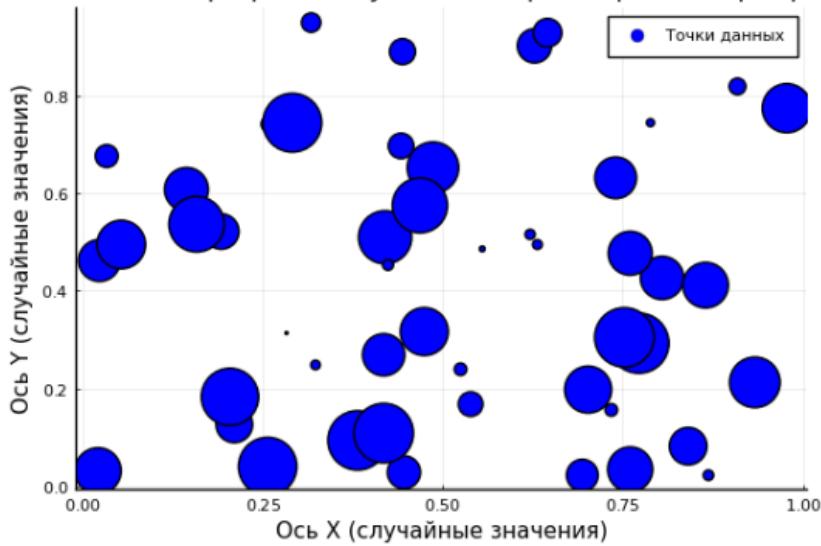


Рис. 2.8: График пятидесяти случайных значений на плоскости с различными опциями отображения (точечный график с кодированием значения размером точки)

Также можно строить и 3-мерные точечные графики (рис. 2.9):

3.3. 3-мерный точечный график с кодированием значения размером точки

```
[9]: # Параметры распределения точек в пространстве:  
n = 50  
x = rand(n) # случайные значения для оси X  
y = rand(n) # случайные значения для оси Y  
z = rand(n) # случайные значения для оси Z  
ms = rand(50) * 30 # случайный размер маркеров  
# Параметры построения графика:  
scatter3d(x, y, z,  
          markersize = ms,           # Размер маркеров зависит от случайных значений  
          title = "Точечный график в 3D пространстве", # Название графика  
          xlabel = "X",             # Подпись оси X  
          ylabel = "Y",             # Подпись оси Y  
          zlabel = "Z",             # Подпись оси Z  
          label = "Точки данных", # Легенда для точек  
          color = :blue,            # Цвет точек  
          grid = true,              # Включение сетки  
          legend = :topright)      # Размещение легенды в правом верхнем углу
```

[9]: Точечный график в 3D пространстве

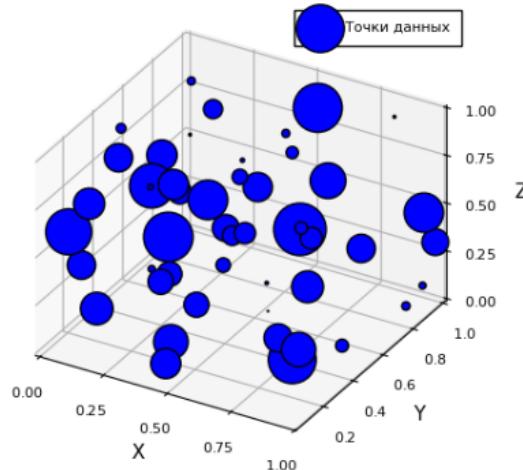


Рис. 2.9: График пятидесяти случайных значений в пространстве с различными опциями отображения (3-мерный точечный график с кодированием значения размером точки)

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. 2.10):

▼ 4. Аппроксимация данных

```
[10]: # Массив данных от 0 до 10 с шагом 0.01:  
x = collect(0:0.01:9.99)  
# Экспоненциальная функция со случайным сдвигом значений:  
y = exp.(ones(1000) + x) + 4000*randn(1000)  
# Построение графика:  
scatter(x, y,  
    markersize = 3,           # Размер маркеров  
    alpha = 0.8,             # Прозрачность маркеров  
    title = "График экспоненциальной функции с шумом", # Название графика  
    xlabel = "Переменная x", # Подпись оси X  
    ylabel = "Значения функции y", # Подпись оси Y  
    label = "Сдвиг с шумом", # Легенда для точек  
    color = :blue,            # Цвет точек  
    grid = true,              # Включение сетки  
    legend = :topright)       # Размещение легенды в правом верхнем углу
```

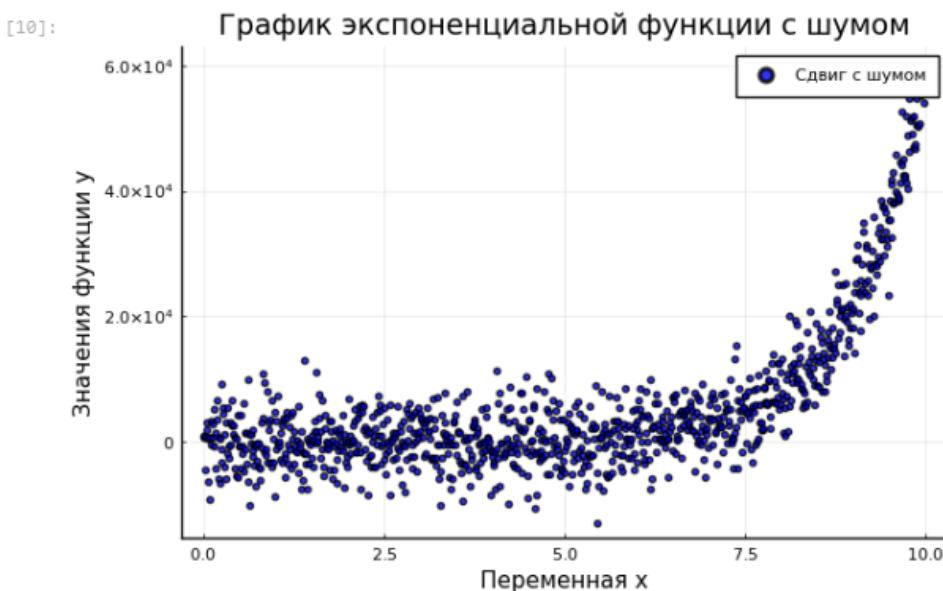


Рис. 2.10: Пример функции

Аппроксимируем полученную функцию полиномом 5-й степени (рис. 2.11):

```
[12]: # Определение массива для нахождения коэффициентов полинома:
A = [ones(1000) x x.^2 x.^3 x.^4 x.^5]
# Решение матричного уравнения:
c = A\y
# Построение полинома (переименовали f в f_poly):
f_poly = c[1]*ones(1000) + c[2]*x + c[3]*x.^2 + c[4]*x.^3 + c[5]*x.^4 + c[6]*x.^5
# Построение графика аппроксимирующей функции:
plot!(x, f_poly,
    linewidth = 3,           # Толщина линии
    color = :red,            # Цвет линии
    title = "Аппроксимация функции полиномом 5-й степени", # Название графика
    xlabel = "Переменная x", # Подпись оси X
    ylabel = "Значение функции y", # Подпись оси Y
    label = "Аппроксимирующий полином", # Легенда для аппроксимирующей функции
    grid = true,              # Включение сетки
    legend = :topright)      # Размещение легенды в правом верхнем углу
```

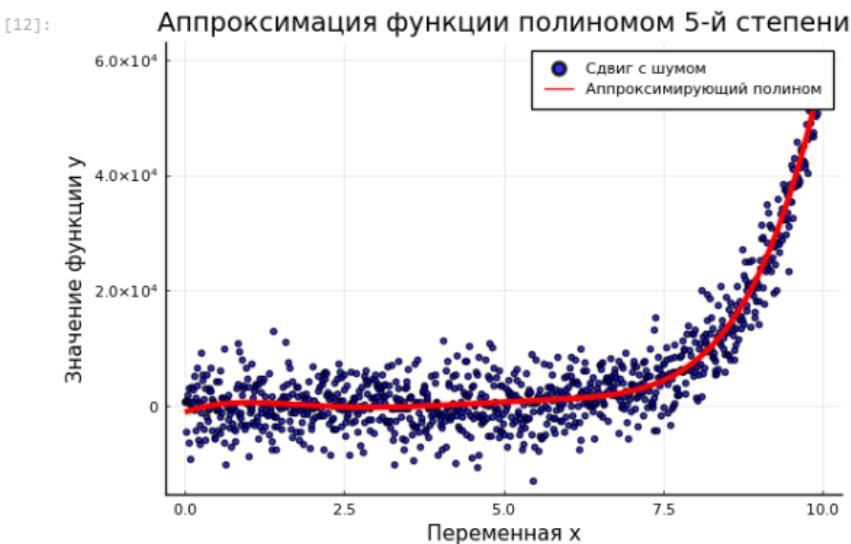


Рис. 2.11: Пример аппроксимации исходной функции полиномом 5-й степени

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. 2.12):

5. Две оси ординат

```
[13]: # Пример случайной траектории
plot(randn(100),
      title="Случайная траектория", # Заголовок графика
      xlabel="Время (t)",          # Подпись оси X
      ylabel="y1",                 # Подпись оси Y
      label="Траектория 1",       # Название кривой в легенде
      leg=:topright,              # Легенда в верхнем правом углу
      grid = :off,                # Отключение сетки
      color=:blue,                # Цвет графика
      linewidth=2,                # Толщина линии
      markersize=3                # Размер маркеров
)
```

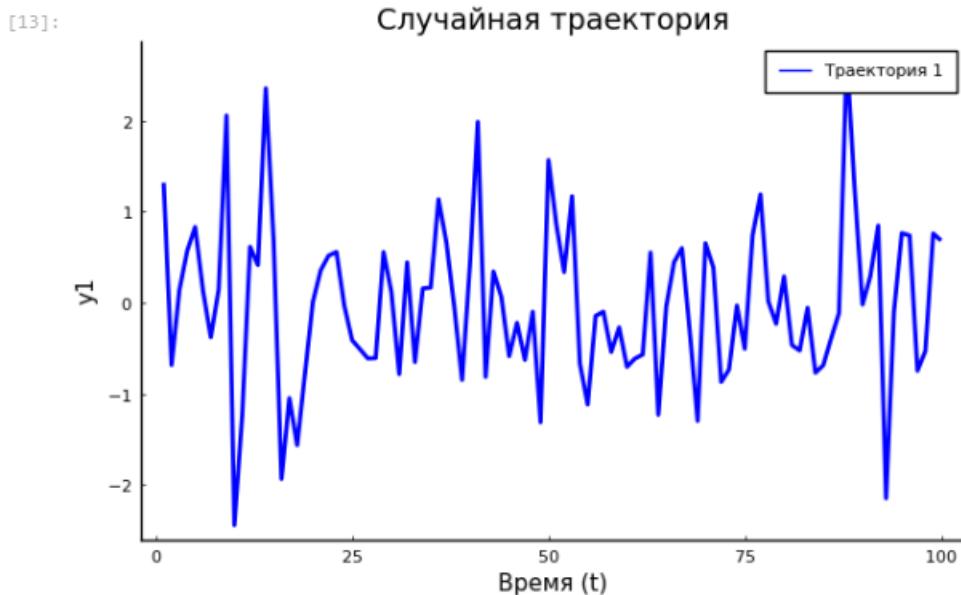


Рис. 2.12: Пример отдельно построенной траектории

2.6 Полярные координаты

Приведём пример построения графика функции в полярных координатах (рис. 2.13):

6. Полярные координаты

```
[15]: # Функция в полярных координатах
r(theta) = 1 + cos(theta) * sin(theta)^2
# Полярная система координат
theta = range(0, stop=2pi, length=50)
# Построение графика функции в полярных координатах
plot(theta, r.(theta),
      proj=:polar,                      # Использование полярной проекции
      lims=(0, 1.5),                     # Ограничения по радиусу
      title="Полярная кривая",          # Заголовок графика
      xlabel="Угол (θ)",                # Подпись оси X (Угол)
      ylabel="Радиус (r)",              # Подпись оси Y (Радиус)
      label="r(θ) = 1 + cos(θ) * sin(θ)^2", # Легенда с названием функции
      legend=:topright,                 # Позиция легенды в правом верхнем углу
      color=:blue,                      # Цвет графика
      linewidth=2,                      # Толщина линии
      grid=:on,                         # Включение сетки
      box=true                          # Включение рамки графика
)
```

[15]:

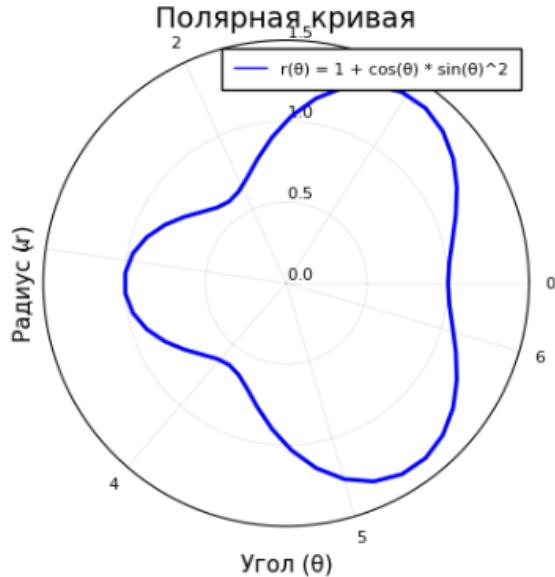


Рис. 2.13: График функции, заданной в полярных координатах

2.7 Параметрический график

Приведём пример построения графика параметрически заданной кривой на плоскости (рис. 2.14):

7. Параметрический график

7.1. Параметрический график кривой на плоскости

```
[19]: # Параметрическое уравнение
x_t(t) = sin(t)
y_t(t) = sin(2t)
# Построение графика
plot(x_t, y_t, 0, 2π,
      title="Параметрическая траектория", # Заголовок графика
      xlabel="sin(t)", # Подпись оси X
      ylabel="sin(2t)", # Подпись оси Y
      label="Траектория (x = sin(t), y = sin(2t))", # Легенда
      legend=:topright, # Расположение легенды
      fill=(0, :orange), # Заливка до оси X
      lw=2, # Толщина линии
      color=:blue, # Цвет графика
      grid=:on, # Включение сетки
)
```

[19]:

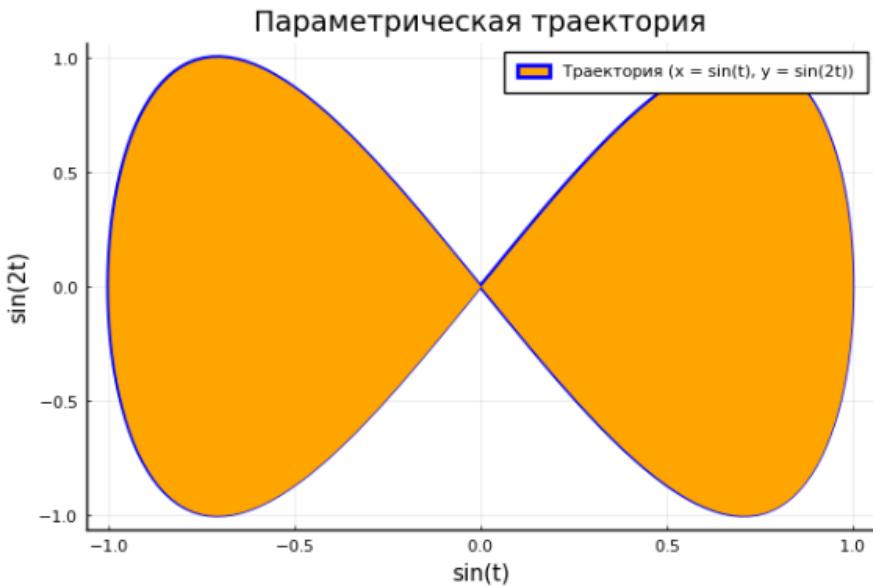


Рис. 2.14: Параметрический график кривой на плоскости

Далее приведём пример построения графика параметрически заданной кривой в пространстве (рис. 2.15):

7.2. Параметрический график кривой в пространстве

```
[23]: # Параметрическое уравнение
t = range(0, stop=10, length=1000)
x = cos.(t)
y = sin.(t)
z = sin.(5t)
# Построение графика
plot(x, y, z,
      title="Параметрическая 3D траектория", # Заголовок графика
      xlabel="x = cos(t)", # Подпись оси X
      ylabel="y = sin(t)", # Подпись оси Y
      zlabel="z = sin(5t)", # Подпись оси Z
      label="Траектория в 3D", # Название траектории в легенде
      legend=:topright, # Расположение легенды
      lw=2, # Толщина линии
      color=:blue, # Цвет графика
      grid=:on, # Включение сетки
      )
```

[23]: Параметрическая 3D траектория

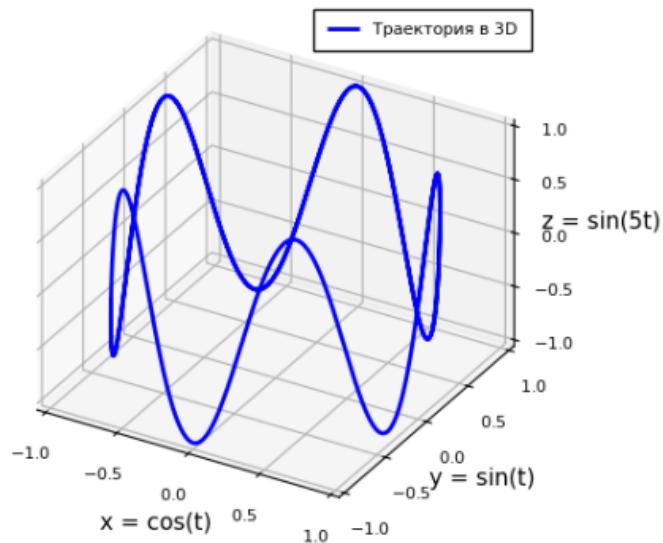


Рис. 2.15: Параметрический график кривой в пространстве

2.8 График поверхности

Для построения поверхности, заданной уравнением $f(x, y) = x^2 + y^2$, можно воспользоваться функцией `surface()` (рис. 2.16):

▼ 8. График поверхности

```
[26]: # Построение графика поверхности
f_xy(x, y) = x^2 + y^2
x = -10:10
y = x
# График поверхности
surface(x, y, f_xy,
        title="График поверхности: z = x2 + y2", # Заголовок
        xlabel="x", # Подпись оси X
        ylabel="y", # Подпись оси Y
        zlabel="z", # Подпись оси Z
        color=:viridis, # Цветовая схема
        legend=false, # Отключение легенды
        colorbar=true # Включение цветовой шкалы
)
```

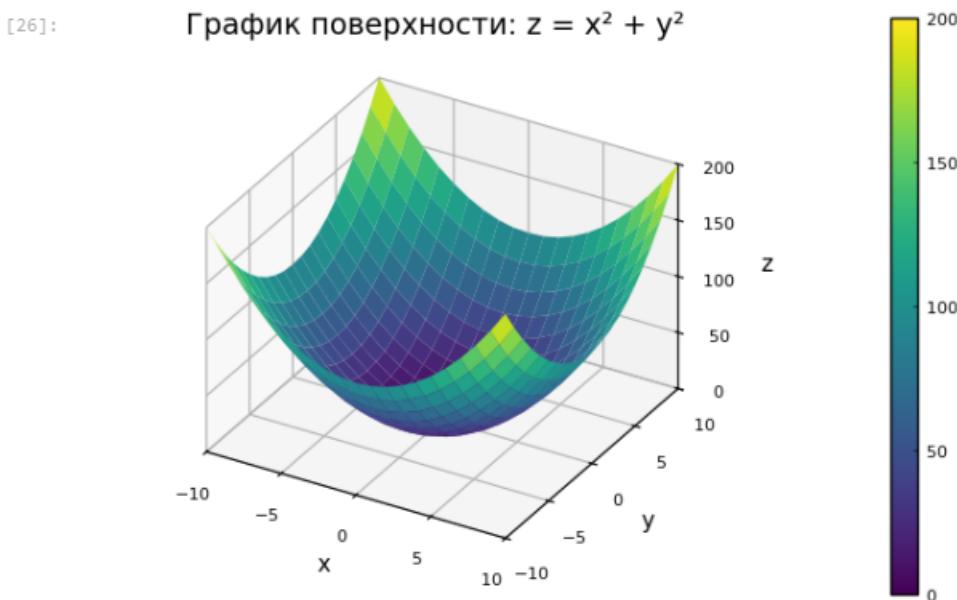


Рис. 2.16: График поверхности (использована функция `surface()`)

Также можно воспользоваться функцией `plot()` с заданными параметрами (рис. 2.17):

```
[28]: # Построение графика поверхности
f_xy(x, y) = x^2 + y^2
x = -10:10
y = x
# График поверхности в виде каркаса
plot(x, y, f_xy,
      linetype=:wireframe,           # Каркасный стиль графика
      title="График поверхности: z = x^2 + y^2", # Заголовок
      xlabel="x",                   # Подпись оси X
      ylabel="y",                   # Подпись оси Y
      zlabel="z",                   # Подпись оси Z
      legend=false                 # Отключение легенды
)
```

[28]: График поверхности: $z = x^2 + y^2$

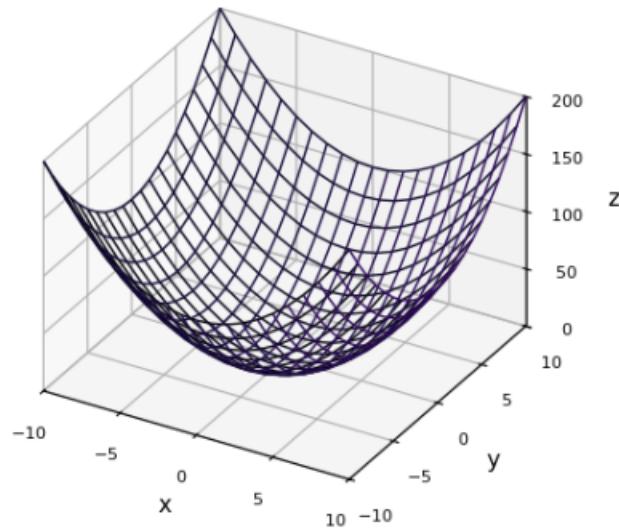


Рис. 2.17: График поверхности (использована функция plot())

Можно задать параметры сглаживания (рис. 2.18):

```
[32]: # Определение функции поверхности
f_xy(x, y) = x^2 + y^2
# Диапазон значений для осей
x = -10:0.1:10
y = x
# Построение графика поверхности
plot(x, y, f_xy,
      linetype=:surface,           # Поверхностный график
      title="График поверхности: z = x^2 + y^2", # Заголовок графика
      xlabel="x",                  # Подпись оси X
      ylabel="y",                  # Подпись оси Y
      zlabel="z",                  # Подпись оси Z
      color=:viridis,             # Цветовая схема
      legend=false,                # Отключение легенды
      colorbar=true
)
```

[32]: График поверхности: $z = x^2 + y^2$

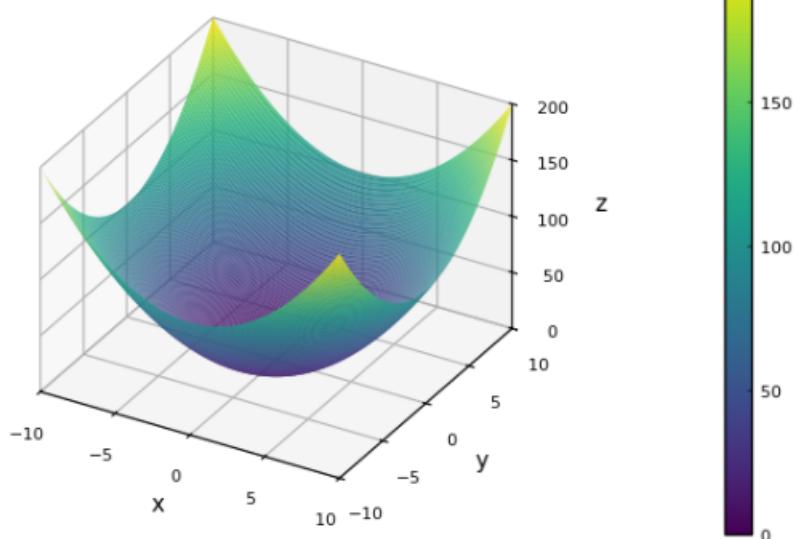


Рис. 2.18: Сглаженный график поверхности

Можно задать определённый угол зрения (рис. 2.19):

```
[35]: # Определение диапазонов значений для осей
x = range(-2, stop=2, length=100)
y = range(sqrt(2), stop=2, length=100)
# Определение функции поверхности
f_xy(x, y) = x * y - x - y + 1
# Построение графика поверхности
plot(x, y, f_xy,
      linetype=:surface,                      # Поверхностный график
      title="Поверхность: f(x, y) = xy - x - y + 1", # Заголовок графика
      xlabel="x",                             # Подпись оси X
      ylabel="y",                             # Подпись оси Y
      zlabel="z",                             # Подпись оси Z
      c=cgrad([:red, :blue]),                 # Градиент цвета
      camera=(-30, 30),                      # Угол обзора камеры
      legend=false,                          # Отключение легенды
      colorbar=true
)
```

[35]: Поверхность: $f(x, y) = xy - x - y + 1$

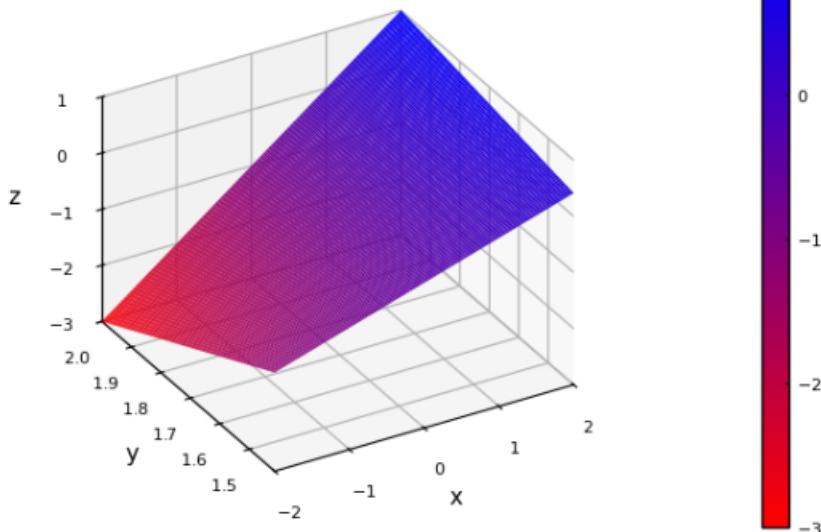


Рис. 2.19: График поверхности с изменённым углом зрения

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

ние 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. 2.20):

9. Линии уровня

```
[38]: # Определение диапазонов для x и y
x = 1:0.5:20
y = 1:0.5:10
# Определение функции поверхности
g(x, y) = (3x + y^2) * abs(sin(x) + cos(y))
# Построение графика поверхности
plot(x, y, g,
      linetype=:surface,           # Поверхностный график
      title="Функция: g(x, y)",    # Заголовок графика
      xlabel="x",                  # Подпись оси X
      ylabel="y",                  # Подпись оси Y
      zlabel="z",                  # Подпись оси Z
      c=cgrad([:green, :yellow, :red]), # Градиент цвета
      colorbar=true                # Включение цветовой шкалы
)
```

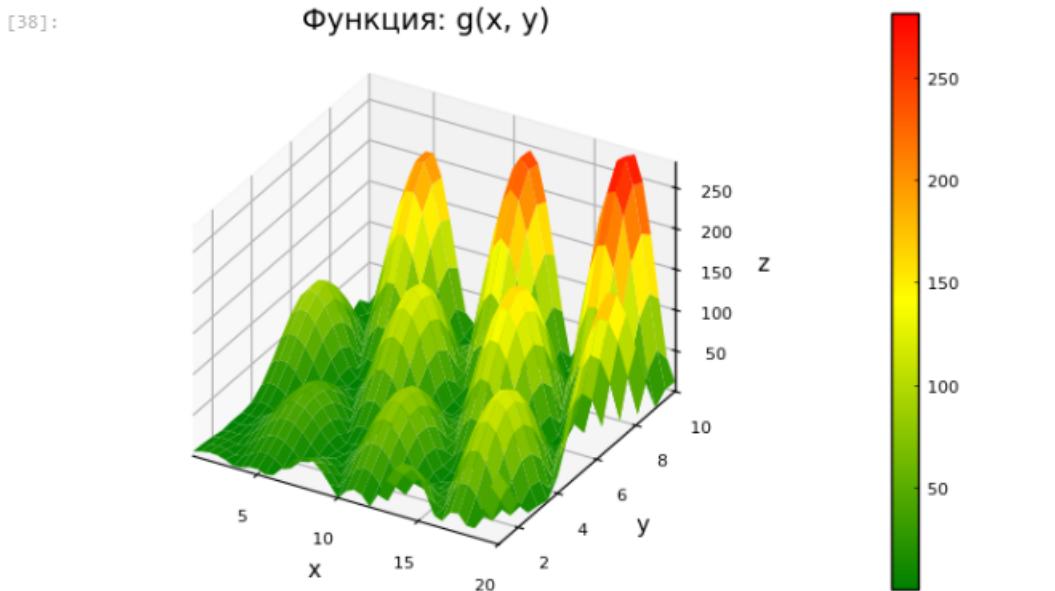


Рис. 2.20: График поверхности, заданной функцией $g(x, y) = (3x + y^2)|\sin(x) + \cos(y)|$

Линии уровня можно построить, используя проекцию значений исходной функции на плоскость (рис. 2.21):

```
[40]: # Построение контурного графика
contour(x, y, g,
        title="Контурный график функции g(x, y)", # Заголовок
        xlabel="x", # Подпись оси X
        ylabel="y", # Подпись оси Y
        color=:viridis, # Цветовая схема
        lw=1.5 # Толщина линий
)
```

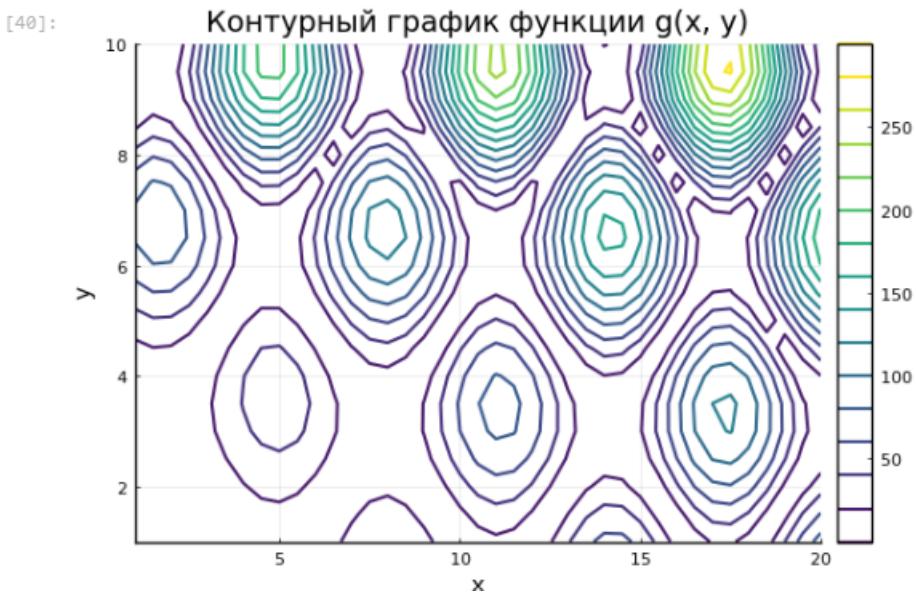


Рис. 2.21: Линии уровня

Можно дополнительно добавить заливку цветом (рис. 2.22):

```
[42]: p = contour(x, y, g,
    title="Заполненный контурный график функции g(x, y)", # Заголовок
    xlabel="x", # Подпись оси X
    ylabel="y", # Подпись оси Y
    fill=true, # Заполнение области между уровнями
    color=:viridis, # Цветовая схема
    legend=false, # Отключение отдельной легенды
    colorbar=true
)
```

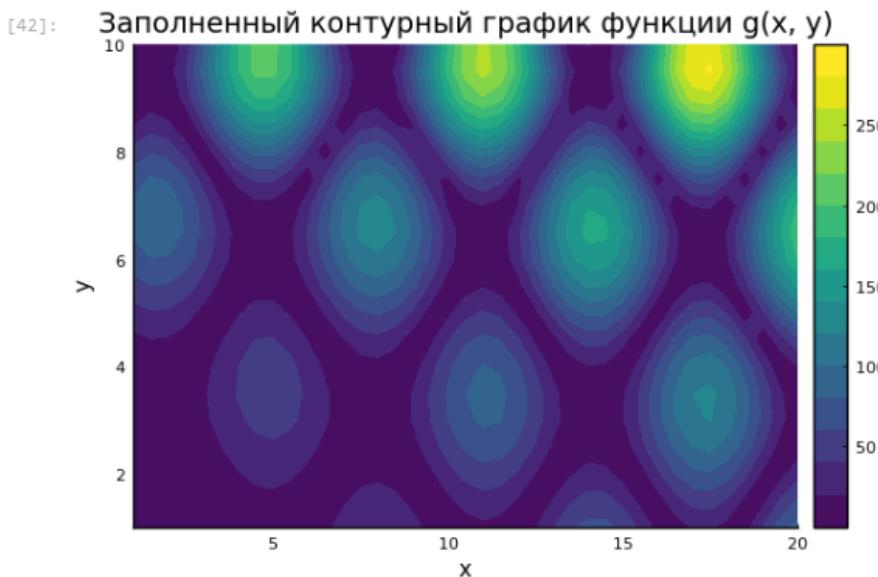


Рис. 2.22: Линии уровня с заполнением

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 $h(x, y) = x^3 - 3x + y^2$ сначала построим её график (рис. 2.23) и линии уровня (рис. 2.24):

10. Векторные поля

```
[45]: # Определение переменных
X = range(-2, stop=2, length=100)
Y = range(-2, stop=2, length=100)
# Определение функции
h(x, y) = x^3 - 3x + y^2
# Построение поверхности
plot(X, Y, h,
    linetype = :surface,           # Тип графика: поверхность
    title = "График функции h(x, y)", # Заголовок графика
    xlabel = "X",                  # Подпись оси X
    ylabel = "Y",                  # Подпись оси Y
    zlabel = "h(x, y)",            # Подпись оси Z (значения функции)
    color = :plasma,              # Цветовая схема
    legend = false,                # Отключение легенды (если она не нужна)
    grid = true,                  # Включение сетки
    colorbar=true
)
```

[45]:

График функции $h(x, y)$

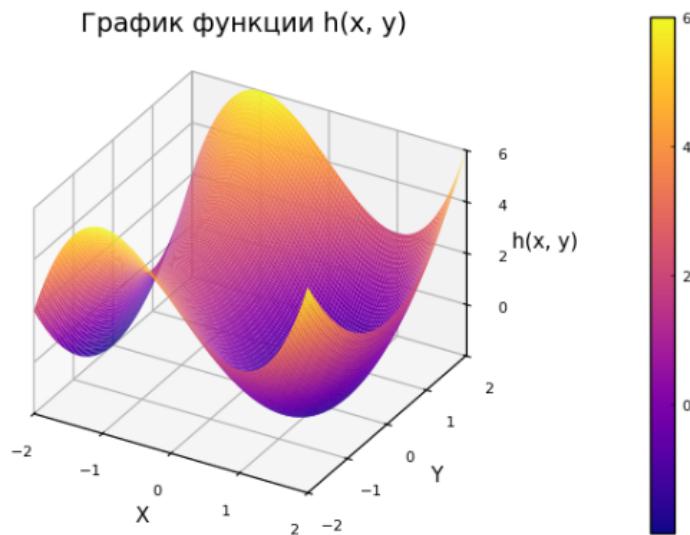


Рис. 2.23: График функции $h(x, y) = x^3 - 3x + y^2$

```
[50]: # Построение линий уровня
contour(X, Y, h,
        title = "Линии уровня функции h(x, y)", # Заголовок графика
        xlabel = "X", # Подпись оси X
        ylabel = "Y", # Подпись оси Y
        color = :plasma, # Цветовая схема для контуров
        legend = false, # Отключение легенды (если не требуется)
        grid = true, # Включение сетки
        linewidth = 2, # Толщина линий
        colorbar=true
)
```

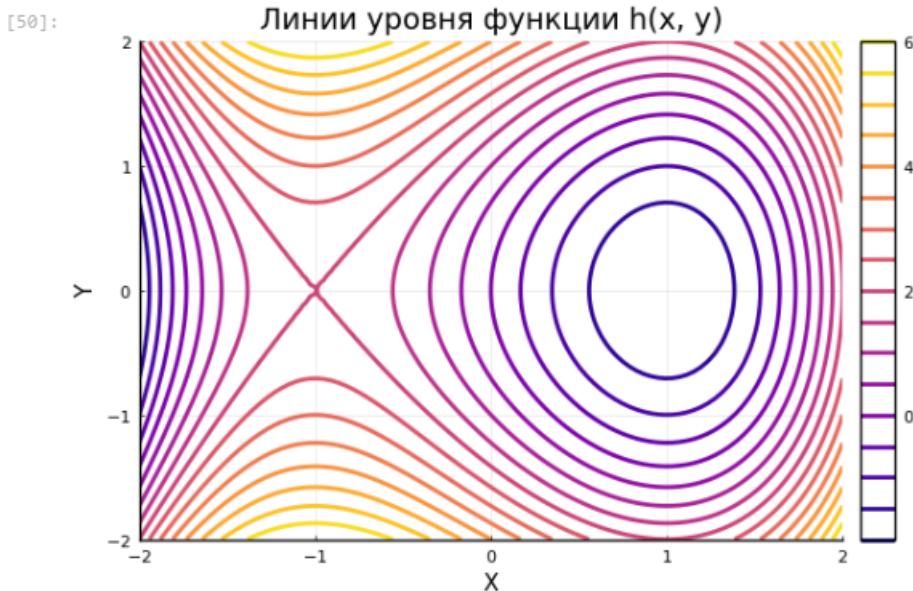


Рис. 2.24: Линии уровня для функции $h(x, y) = x^3 - 3x + y^2$

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 графиков) в одном файле.

В Julia рекомендуется использовать gif-анимацию в `pyplot()`.

Строим поверхность (рис. 2.25):

11. Анимация

11.1. Gif-анимация

```
[61]: # построение поверхности:  
i = 0  
X = Y = range(-5,stop=5,length=40)  
surface(X, Y, (x,y) -> sin(x+10sin(i))+cos(y))
```

[61]:

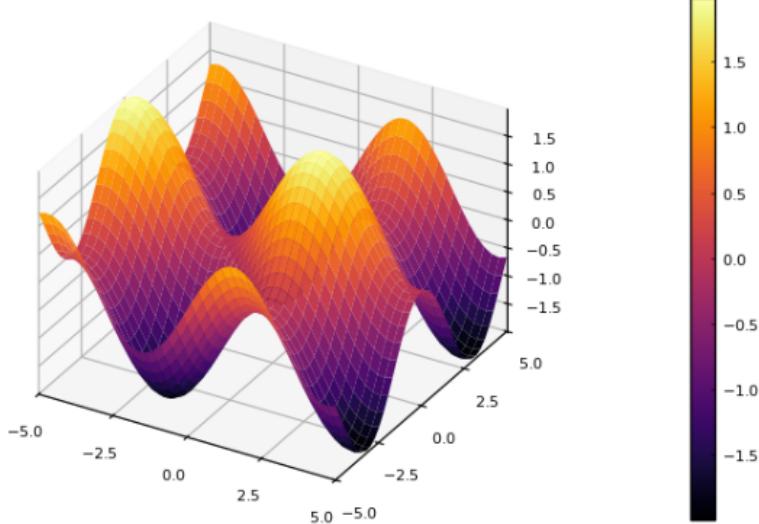


Рис. 2.25: Статичный график поверхности

Добавляем анимацию (рис. 2.26):

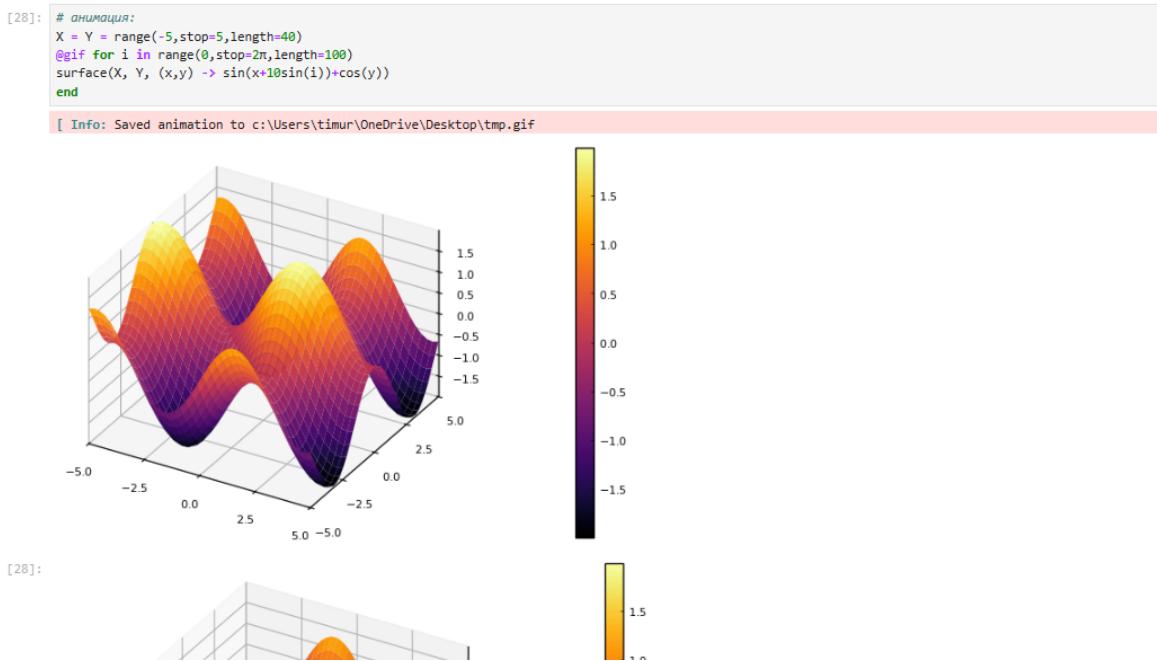


Рис. 2.26: Анимированный график поверхности

2.12 Гипоциклоида

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

Построим большую окружность (рис. 2.27):

11.2. Гипоциклоида

```
[42]: # радиус малой окружности:  
radius = 1  
# коэффициент для построения большой окружности:  
k = 3  
# число отсчётов:  
n = 100  
# массив значений угла θ:  
# theta from 0 to 2pi ( + a little extra)  
θ = collect(0:2*π/100:2*π+2*π/100)  
# массивы значений координат:  
X = radius*k*cos.(θ)  
Y = radius*k*sin.(θ)  
# задаём оси координат:  
plot=plt(xlim=(-4,4),ylim=(-4,4), c=:red, aspect_ratio=1, legend=false, framestyle=:origin)  
# большая окружность:  
plot!(plt, X,Y, c=:blue, legend=false)
```

[42]:

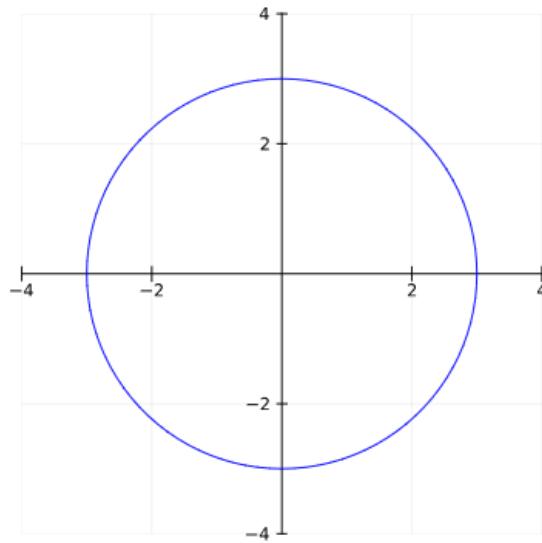


Рис. 2.27: Большая окружность гипоциклоиды

Для частичного построения гипоциклоиды будем менять параметр t (рис. 2.28):

```
[44]: i = 50
t = θ[1:i]
# гипоциклоида:
x = radius*(k-1)*cos.(t) + radius*cos.((k-1)*t)
y = radius*(k-1)*sin.(t) - radius*sin.((k-1)*t)
plot!(x,y, c=:red)
```

[44]:

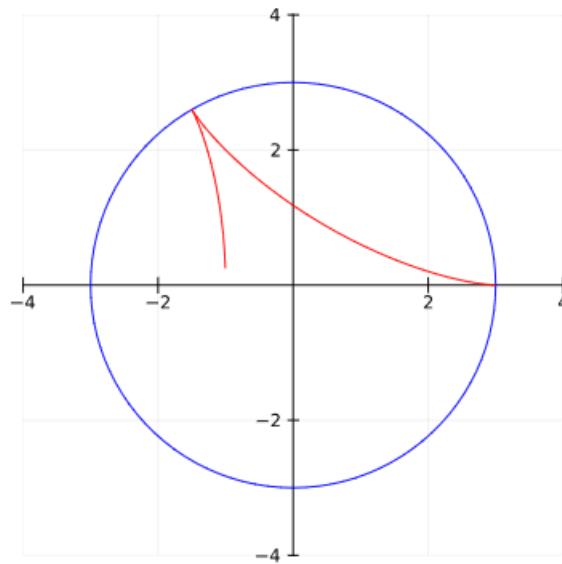


Рис. 2.28: Половина пути гипоциклоиды

Добавляем малую окружность гипоциклоиды (рис. 2.29):

```
[45]: # малая окружность:  
xc = radius*(k-1)*cos(t[end]) .+ radius*cos_(θ)  
yc = radius*(k-1)*sin(t[end]) .+ radius*sin_(θ)  
plot!(xc,yc,c=:black)
```

[45]:

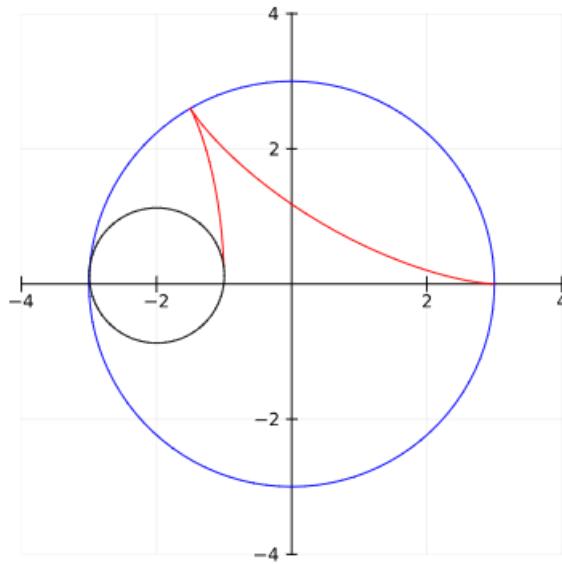


Рис. 2.29: Малая окружность гипоциклоиды

Добавим радиус для малой окружности (рис. 2.30):

```
[46]: # радиус малой окружности:
x1 = transpose([radius*(k-1)*cos(t[end]) x[end]])
y1 = transpose([radius*(k-1)*sin(t[end]) y[end]])
plot!(x1,y1,markershape=:circle,markersize=4,c=:black)
scatter!([x[end]],[y[end]],c=:red, markerstrokecolor=:red)
```

[46]:

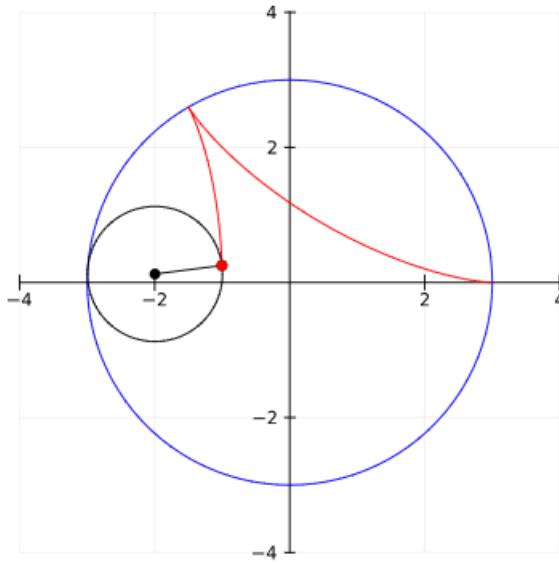


Рис. 2.30: Малая окружность гипоциклоиды с добавлением радиуса

В конце сделаем анимацию получившегося изображения (рис. 2.31):

```
anim = @animate for i in 1:n
    # задаём оси координат:
    plt=plot(5,xlim=(-4,4),ylim=(-4,4), c=:red, aspect_ratio=1,legend=false, framestyle=:origin)
    # большая окружность:
    plot!(plt, X,Y, c=:blue, legend=false)
    t = θ[1:i]
    # гипоциклоида:
    x = radius*(k-1)*cos.(t) + radius*cos.((k-1)*t)
    y = radius*(k-1)*sin.(t) - radius*sin.((k-1)*t)
    plot!(x,y, c=:red)
    # малая окружность:
    xc = radius*(k-1)*cos(t[end]) .+ radius*cos.(θ)
    yc = radius*(k-1)*sin(t[end]) .+ radius*sin.(θ)
    plot!(xc,yc,c=:black)
    # радиус малой окружности:
    x1 = transpose([radius*(k-1)*cos(t[end]) x[end]])
    y1 = transpose([radius*(k-1)*sin(t[end]) y[end]])
    plot!(x1,y1,markershape=:circle,markersize=4,c=:black)
    scatter!([x[end]],[y[end]],c=:red, markerstrokecolor=:red)
end
gif(anim, "hypocycloid.gif")
```

Рис. 2.31: Малая окружность гипоциклоиды с добавлением радиуса

2.13 Errorbars

В исследованиях часто требуется изобразить графики погрешностей измерений.

Построим график исходных значений (рис. 2.32):

▼ 12. Errorbars

```
[68]: using Statistics  
# Параметры  
sds = [1, 1/2, 1/4, 1/8, 1/16, 1/32]  
n = 10  
y = [mean(sd*randn(n)) for sd in sds]  
errs = 1.96 * sds / sqrt(n)  
# Построение графика  
plot(y,  
    ylims = (-1, 1),  
    title = "Средние значения с погрешностями",  
    xlabel = "Итерации",  
    ylabel = "Среднее значение",  
    label = "Среднее значение",  
    errorbars = errs,  
    legend = :topright,  
    linecolor = :blue,  
    linewidth = 2,  
    size = (600, 400)  
)
```

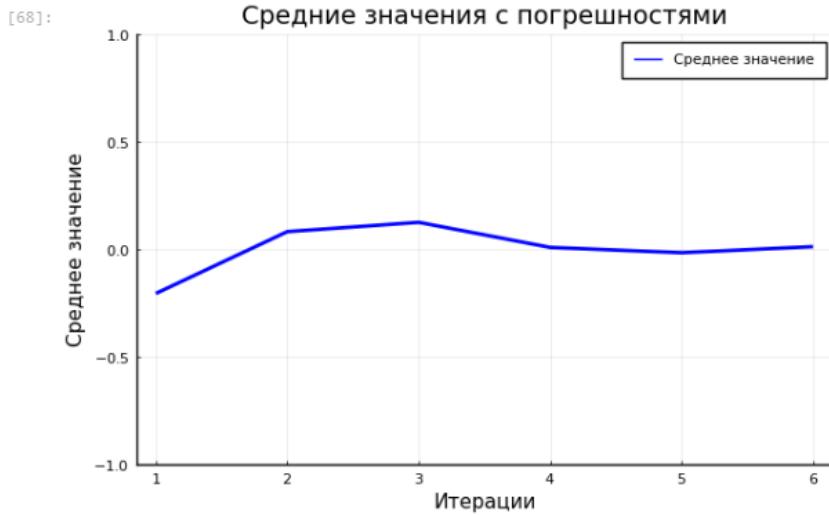


Рис. 2.32: График исходных значений

Построим график отклонений от исходных значений (рис. 2.33):

```
[78]: # Построение графика с ошибками и улучшениями
plot(y,
      ylims = (-1, 1),
      title = "Средние значения с погрешностями",
      xlabel = "Итерации",
      ylabel = "Среднее значение",
      label = "Среднее значение",
      err = errs,
      legend = :topright,
      linecolor = :blue,
      linewidth = 2
)
```

[78]: Средние значения с погрешностями

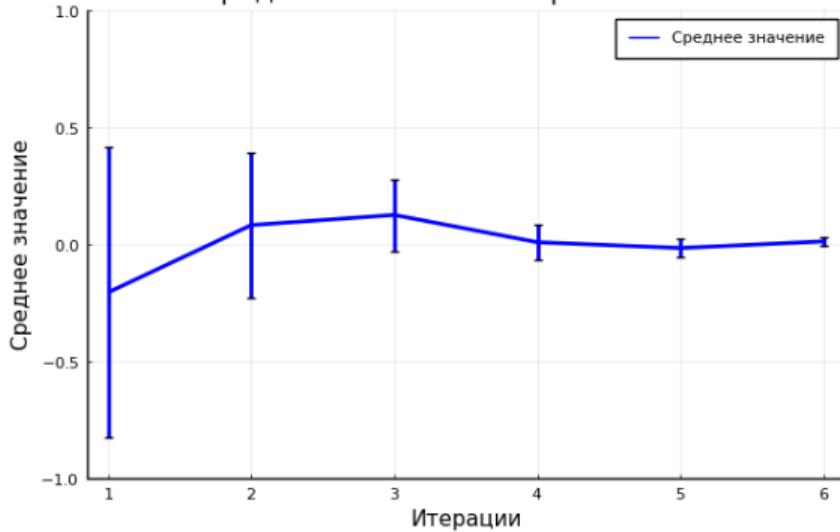


Рис. 2.33: График исходных значений с отклонениями

Повернём график (рис. 2.34):

```
[80]: plot(y, 1:length(y),
         xerr = errs,
         marker = stroke(3,:orange)
     )
```

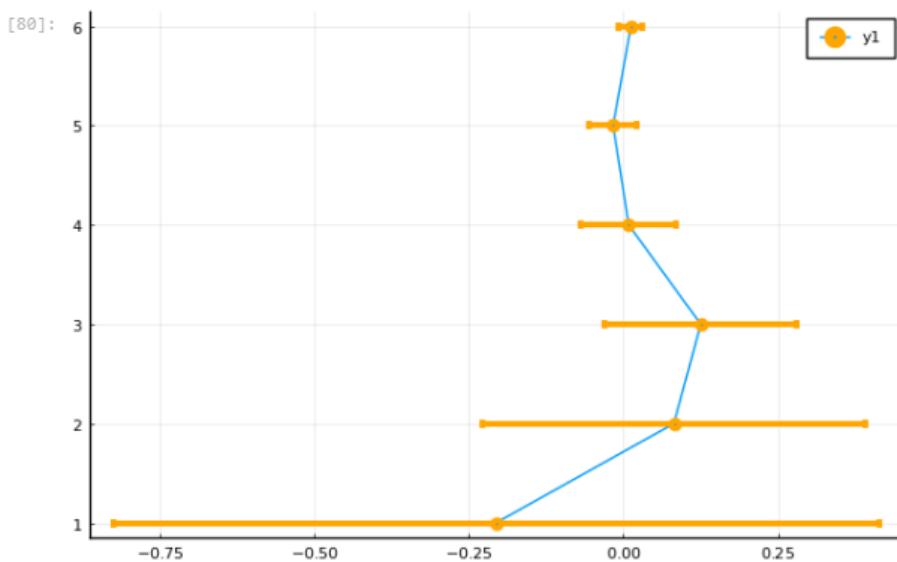


Рис. 2.34: Поворот графика

Заполним область цветом (рис. 2.35):

```
[81]: plot(y,
         ribbon=errs,
         fill=:cyan
     )
```

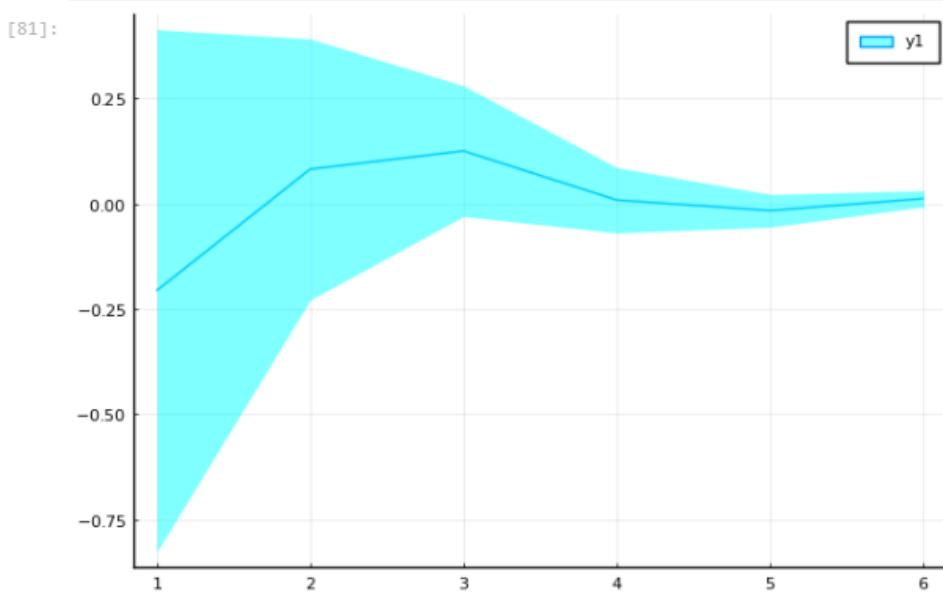


Рис. 2.35: Заполнение цветом

Можно построить график ошибок по двум осям (рис. 2.36):

```
[82]: n = 10
x = [(rand() + 1) .* randn(n) .+ 2i for i in 1:5]
y = [(rand() + 1) .* randn(n) .+ i for i in 1:5]
f_v(v) = 1.96std(v) / sqrt(n)
xerr = map(f_v, x)
yerr = map(f_v, y)
x = map(mean, x)
y = map(mean, y)
plot(x, y,
      xerr = xerr,
      yerr = yerr,
      marker = stroke(2, :orange))
)
```

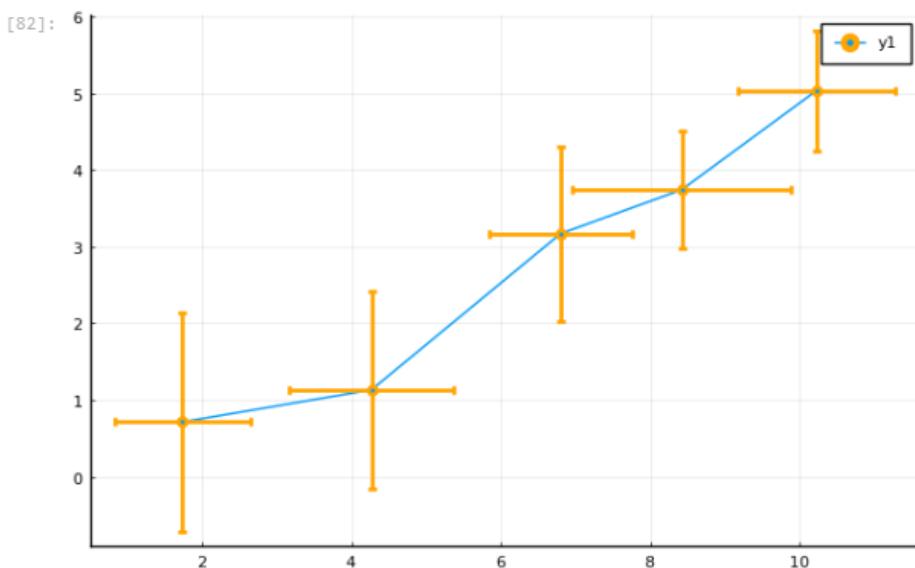


Рис. 2.36: График ошибок по двум осям

Можно построить график асимметричных ошибок по двум осям (рис. 2.37):

```
[83]: plot(x, y,
         xerr = (0.5xerr,2xerr),
         yerr = (0.5yerr,2yerr),
         marker = stroke(2, :orange)
     )
```

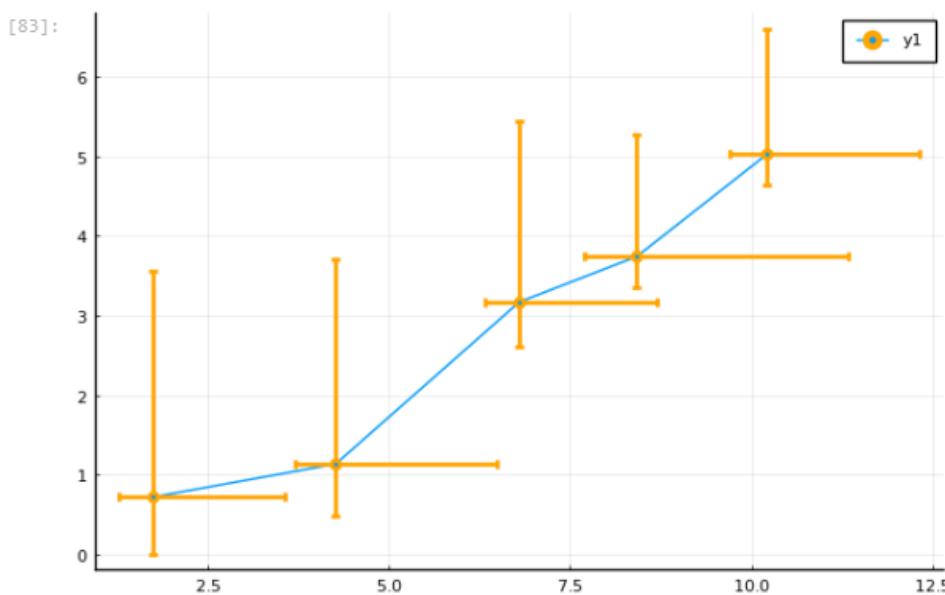


Рис. 2.37: График асимметричных ошибок по двум осям

2.14 Использование пакета Distributions

Строим гистограмму (рис. 2.38):

▼ 13. Использование пакета Distributions

```
[85]: using Distributions  
pyplot()  
ages = rand(15:55,1000)  
histogram(ages)
```

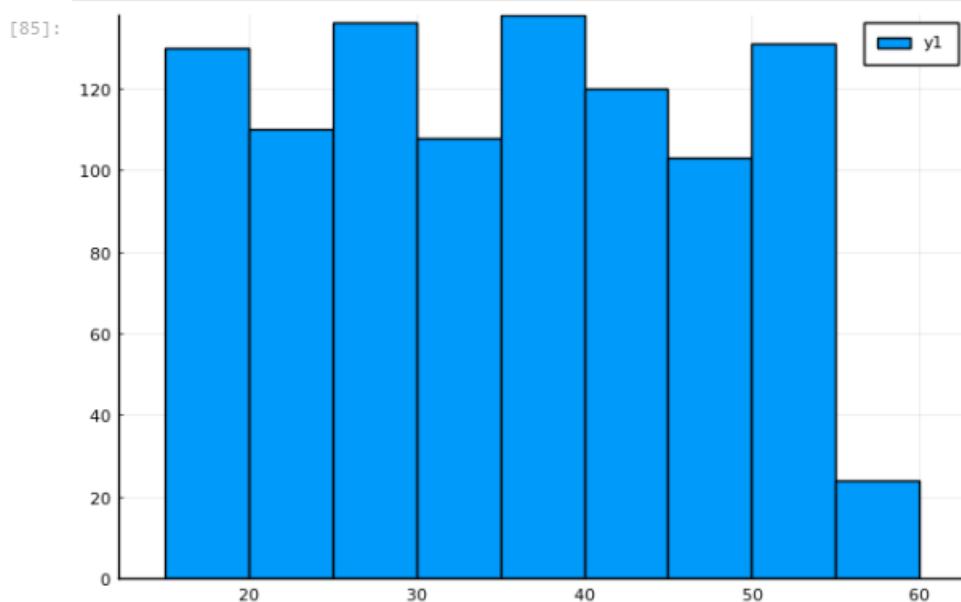


Рис. 2.38: Гистограмма, построенная по массиву случайных чисел

Задаём нормальное распределение и строим гистограмму (рис. 2.39):

```
[86]: d=Normal(35.0,10.0)
ages = rand(d,1000)
histogram(
    ages,
    label="Распределение по возрастам (года)",
    xlabel = "Возраст (лет)",
    ylabel= "Количество"
)
```

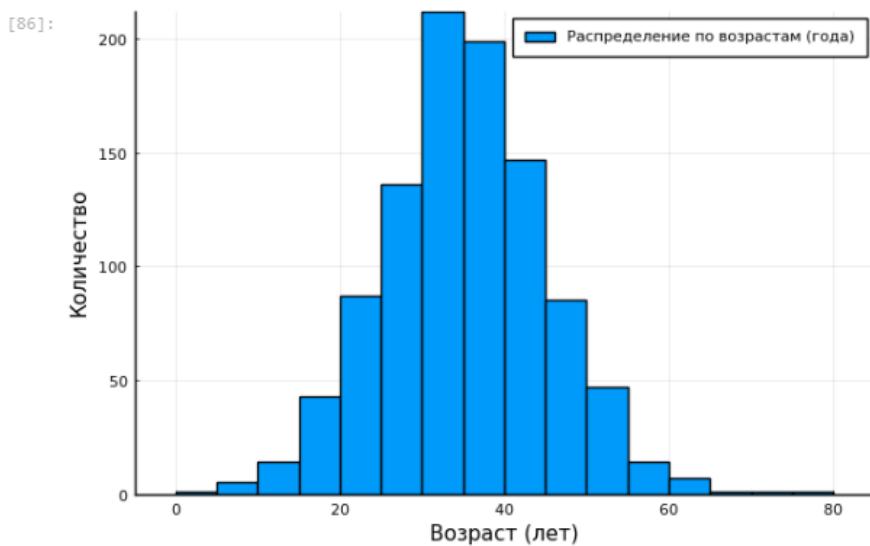


Рис. 2.39: Гистограмма нормального распределения

Далее применим для построения нескольких гистограмм распределения людей по возрастам на одном графике plotly() (рис. 2.40):

```
[89]: plotly()
d1=Normal(10.0,5.0);
d2=Normal(35.0,10.0);
d3=Normal(60.0,5.0);
N=1000;
ages = (Float64)[];
ages = append!(ages,rand(d1,Int64(ceil(N/2))));
ages = append!(ages,rand(d2,N));
ages = append!(ages,rand(d3,Int64(ceil(N/3))));
histogram(
    ages,
    bins=50,
    label="Распределение по возрастам (года)",
    xlabel = "Возраст (лет)",
    ylabel= "Количество",
    title = "Распределение по возрастам (года"
)
)
```

[89]:

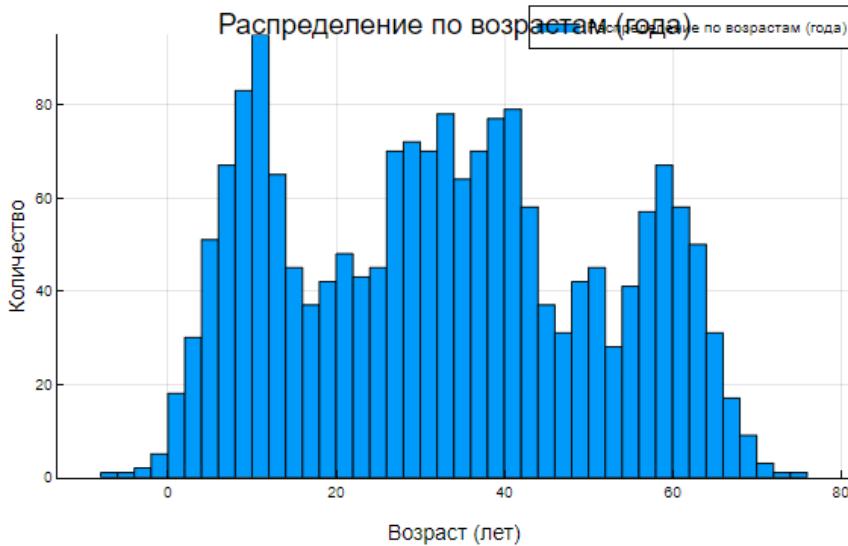


Рис. 2.40: Гистограмма распределения людей по возрастам

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. 2.41):

14. Подграфики

```
[90]: # подгружаем pyplot():
pyplot()
# построение серии графиков:
x=range(-2,2,length=10)
y = rand(10,4)
plot(x,y,
      layout=(4,1)
)
```

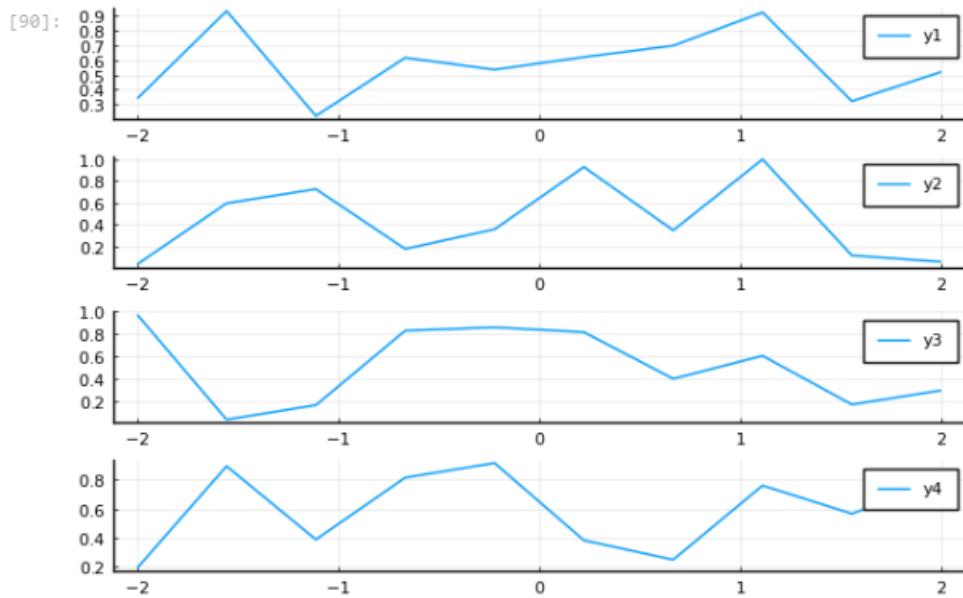


Рис. 2.41: Серия из 4-х графиков в ряд

Для автоматического вычисления сетки необходимо передать layout целое число (рис. 2.42):

```
[91]: plot(x,y,  
         layout=4  
)
```

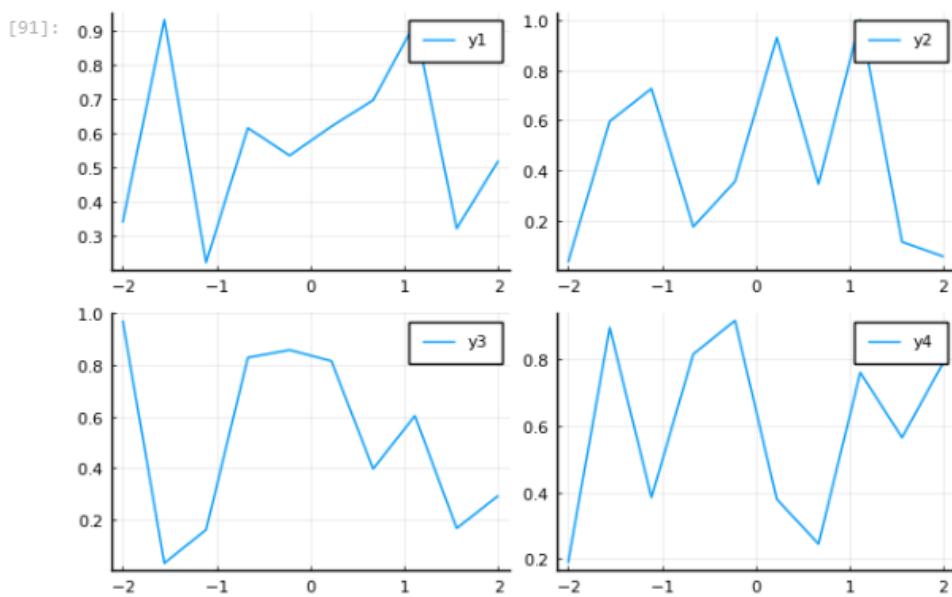


Рис. 2.42: Серия из 4-х графиков в сетке

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. 2.43):

```
[92]: # график в виде линий:
p1 = plot(x,y)
# график в виде точек:
p2 = scatter(x,y)
# график в виде линий с оформлением:
p3 = plot(x,y[:,1:2], xlabel="Labelled plot of two
↪ columns", lw=2, title="Wide lines")
# 4 гистограммы:
p4 = histogram(x,y)
plot(
    p1,p2,p3,p4,
    layout=(2,2),
    legend=false,
    size=(800,600),
    background_color = :ivory
)
```

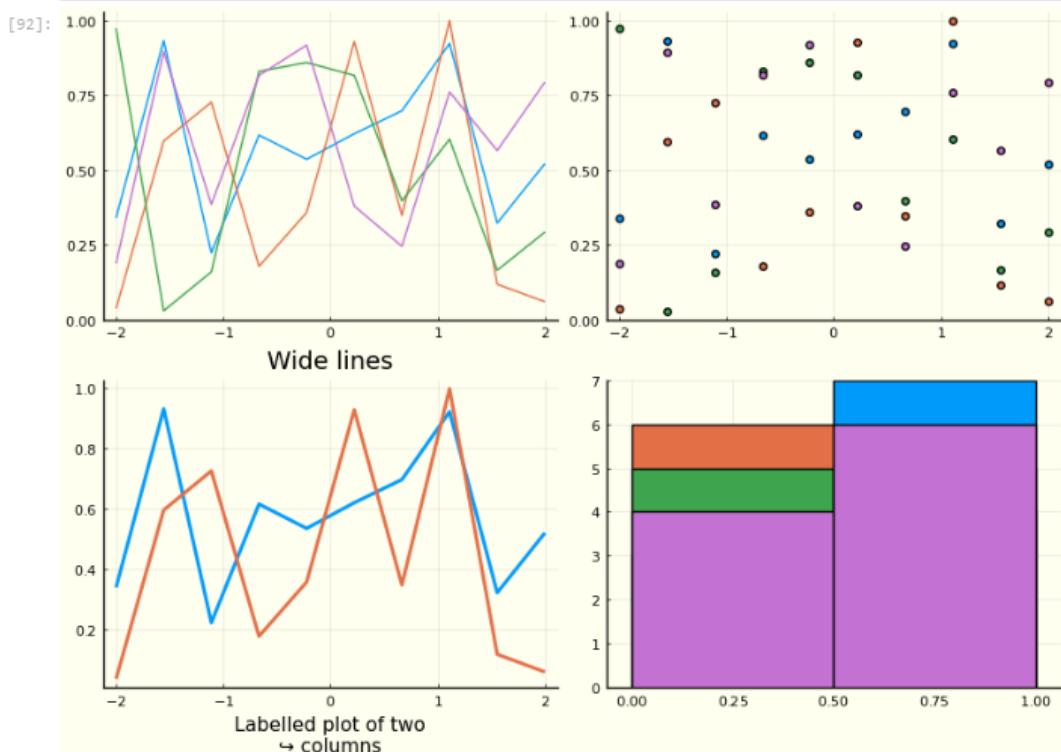


Рис. 2.43: Объединение нескольких графиков в одной сетке

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. 2.44):

```
[93]: seriestypes = [:step, :sticks, :bar, :hline, :vline, :path]
titles =["step" "sticks" "bar" "hline" "vline" "path"]
plot(rand(20,1), st = seriestypes,
      layout = (2,3),
      ticks=nothing,
      legend=false,
      title=titles,
      M=3
    )
```

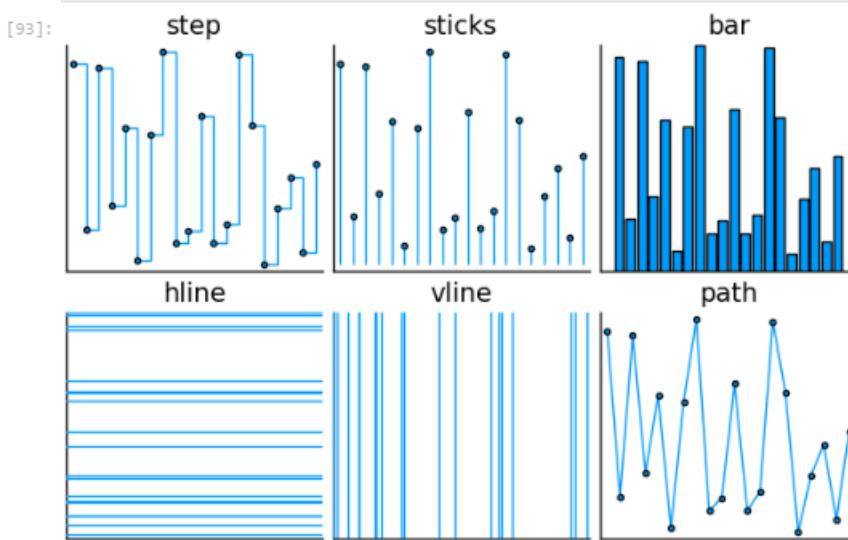


Рис. 2.44: Разнообразные варианты представления данных

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. 2.45):

```
[94]: l = @layout [ a{0.3w} [grid(3,3)
b{0.2h} ]]
plot(
rand(10,11),
layout = 1, legend = false, seriestype = [:bar :scatter :path],
title = ["\"$(i)\"" for j = 1:1, i=1:11], titleloc = :right, titlefont = font(8)
)
```

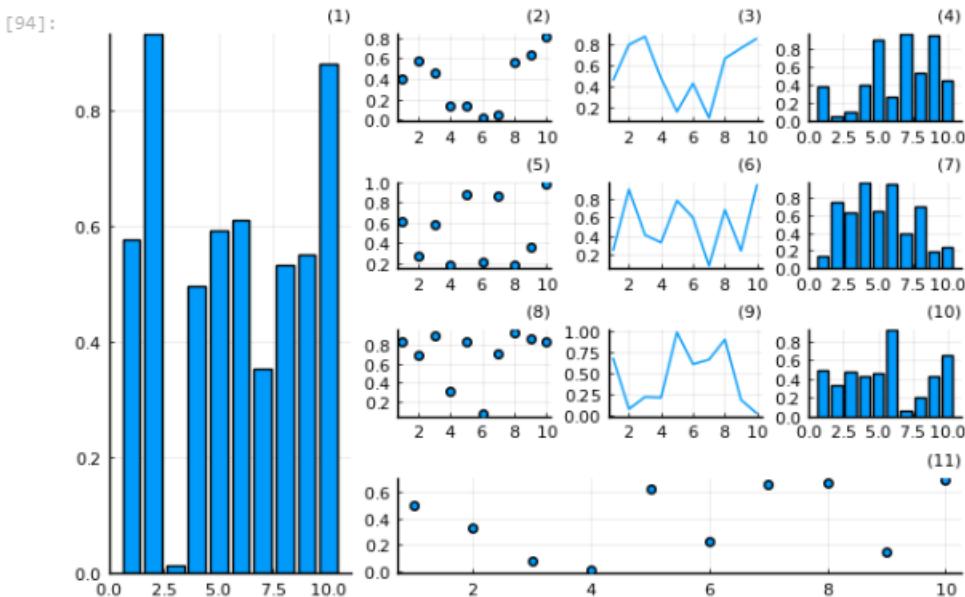


Рис. 2.45: Демонстрация применения сложного макета для построения графиков

2.16 Самостоятельное выполнение

Выполнение задания №1 (рис. 2.46):

1) Постройте все возможные типы графиков (простые, точечные, гистограммы и т.д.) функции $y = \sin(x)$, $x \in [0, 2\pi]$. Отобразите все графики в одном графическом окне:

```
[6@]: # Определим диапазон x от 0 до 2π
x = 0:0.01:2π
# Определим функцию y = sin(x)
y = sin.(x)
# Создаём несколько подграфиков для различных типов графиков
plot(layout=(2, 2), title="y = sin(x)")
# Линейный график
plot!(x, y, seriestype=:line, label="Line", subplot=1, lw=2)
# Точечный график
plot!(x, y, seriestype=:scatter, label="Scatter", subplot=2)
# Гистограмма
histogram!(y, bins=30, label="Histogram", subplot=3)
# График ступеней
plot!(x, y, seriestype=:step, label="Step", subplot=4, lw=2)
```

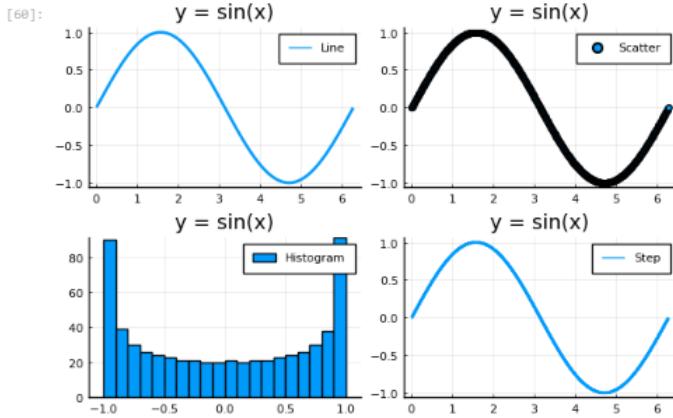


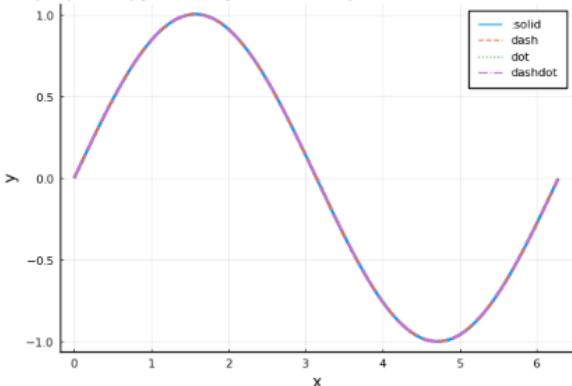
Рис. 2.46: Решение задания №1

Выполнение задания №2 (рис. 2.47):

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:

```
[61]: # Определим диапазон x от 0 до 2π
x = 0:0.01:2π
# Определим функцию y = sin(x)
y = sin.(x)
# Список стилей линий
line_styles = [:solid, :dash, :dot, :dashdot]
# Построим графики с разными стилями линий в одном окне
plt = plot(x, y, linestyle=:solid, label="solid", xlabel="x", ylabel="y", title="Графики функции y = sin(x) с разными стилями линий", lw=2)
for ls in line_styles[2:end]
    plot!(plt, x, y, linestyle=ls, label="$ls", lw=2)
end
display(plt)
```

Графики функции $y = \sin(x)$ с разными стилями линий



Графики функции $y = \sin(x)$ с разными стилями линий

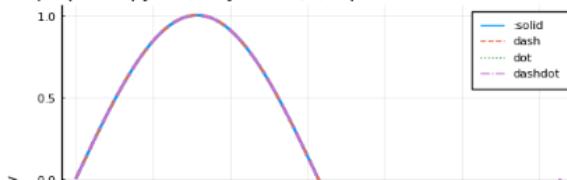


Рис. 2.47: Решение задания №2

Выполнение задания №3 (рис. 2.48):

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: 10

```
[5]: # Определяем функцию y(x)
y(x) = pi * x^2 * log(x)
# Диапазон значений x
x = 0.1:0.01:10
# Построение графика
plot(x, y(x),
    color = :red,
    label = "y(x) = \pi x^2 \ln(x)", # Легенда
    xlabel = "x", # Подпись оси X
    ylabel = "y(x)", # Подпись оси Y
    fontstyle = :bold,
    grid = false, # Убираем сетку
    xticks = 0:2:10, # Частота отметок на оси X
    yticks = -50:50:200, # Частота отметок на оси Y
    bordercolor = :green)
```

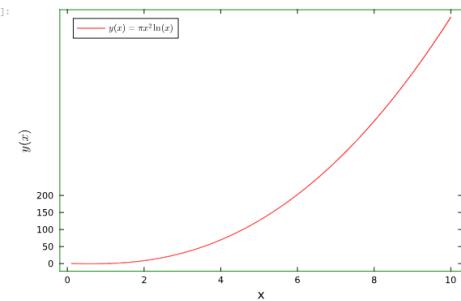


Рис. 2.48: Решение задания №3

Выполнение задания №4 (рис. 2.49):

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:

```
68]: # Задаем вектор x
x = [-2, -1, 0, 1, 2]
# Функция y(x)
y_4(x) = x.^3 - 3.*x
# Вычисляем значения y
y_values = y_4.(x)
# Создаем макет для 4 подграфиков
plt = plot(layout = (2, 2))
# График точек
scatter!(plt[1], x, y_values, label = "Точки")
# График линий
plot!(plt[2], x, y_values, label = "Линии")
# График линий и точек
plot!(plt[3], x, y_values, label = "Линии и точки")
scatter!(plt[3], x, y_values, label = "")
# График кривой
plot!(plt[4], x, y_values, smooth = true, label = "Кривая")
# Сохраняем график в файл
savefig(plt, "figure_eyüboğlu.png")
```

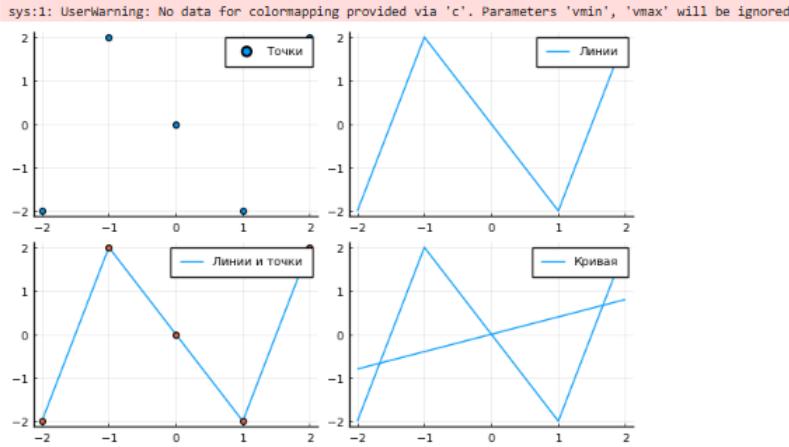


Рис. 2.49: Решение задания №4

Выполнение задания №5 (рис. 2.50 - рис. 2.51):

5. Задайте вектор $x = (3, 3.1, 3.2, \dots, 6)$. Постройте графики функций $y_1(x) = \pi x$ и $y_2(x) = \exp(x) \cos(x)$ 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.

```
[22]: # Задаем вектор x
x = 3:0.1:6
# Определяем функции y1 и y2
y1(x) = pi * x
y2(x) = exp(x).*cos(x)
# 1. Построение графиков на одной оси с легендой и сеткой
plot(x, y1(x), label="y1(x) = \pi x", color=blue, linewidth=2, grid=true, legend=:topright)
plot!(x, y2(x), label="y2(x) = \exp(x) * \cos(x)", color=red, linewidth=2)
# Добавление заголовка
xlabel!("x")
ylabel!("y")
title!("Графики функций y1(x) и y2(x)")
```

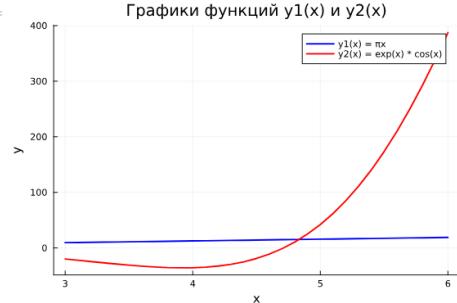


Рис. 2.50: Решение задания №5

```
[21]: # Построение графиков с двумя осями ординат
p = plot(x, y1(x), label="y1(x) = \pi x", color=blue, linewidth=2, grid=true, legend=:topright)
plot!(p, x, y2(x), label="y2(x) = \exp(x) * \cos(x)", color=red, linewidth=2)
# Добавляем вторую ось ординат для y2
plot!(p, secondary=true)
# Заголовок и сетка
xlabel!("x")
ylabel!("y1 (primary)", fontsize=10)
ylabel!(p, "y2 (secondary)", fontsize=10)
title!("Графики с двумя осями ординат")
```

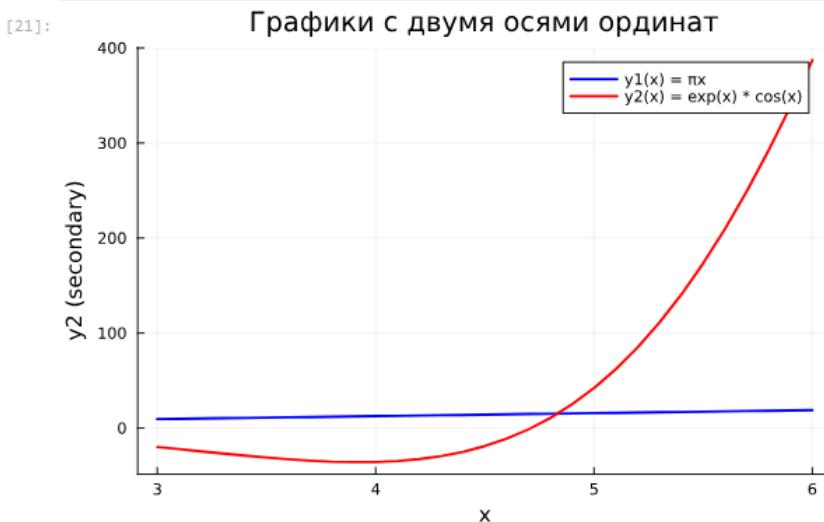


Рис. 2.51: Решение задания №5

Выполнение задания №6 (рис. 2.52):

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

```
[26]: # Шаг 1: Придумываем экспериментальные данные
# Время измерений (например, 10 точек через 1 час)
time = 0:1:9 # Время в часах (от 0 до 9 часов)
# Температурные данные (например, температура варьируется от 20 до 25 градусов)
temperature = 22.+2.*sin.(time * π / 5) # Синусоидальное изменение температуры
# Шаг 2: Добавим ошибку измерения (например, стандартное отклонение 0.5 градуса)
# Ошибка измерения - случайные колебания вокруг истинных значений
temperature_error = 0.5 + 0.1 .* randn(length(time)) # Ошибка измерений с нормальным распределением
# Шаг 3: Строим график с учетом ошибки измерений
plot(time, temperature, label="Температура", color=:blue, linewidth=2, legend=:topright,
      yerr=temperature_error, marker=:o)
# Добавляем подписи осей и заголовок
xlabel!("Время (часы)")
ylabel!("Температура (°C)")
title!("Экспериментальные данные с ошибкой измерения")
```

[26]: Экспериментальные данные с ошибкой измерения

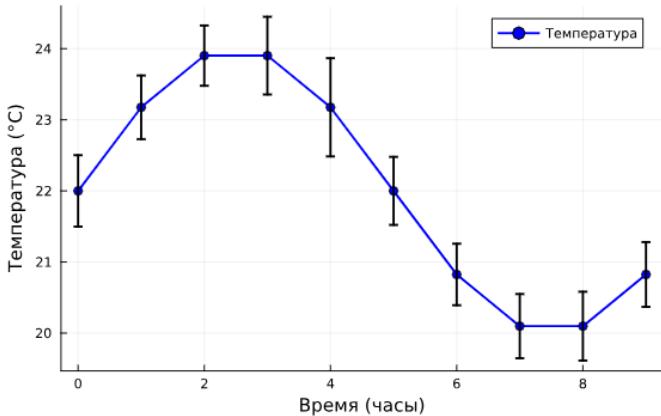


Рис. 2.52: Решение задания №6

Выполнение задания №7 (рис. 2.53):

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

```
[4]: # Генерация случайных данных
x = rand(10) # 10 случайных значений по оси X
y_vals = rand(10) # 10 случайных значений по оси Y
# Построение точечного графика с легендой
scatter(x, y_vals, label="Случайные данные", color=:blue, marker=:o, linewidth=2, legend=:topright)
# Добавление подписей осей и заголовка
xlabel!("X")
ylabel!("Y")
title!("Точечный график случайных данных")
```

[4]: Точечный график случайных данных

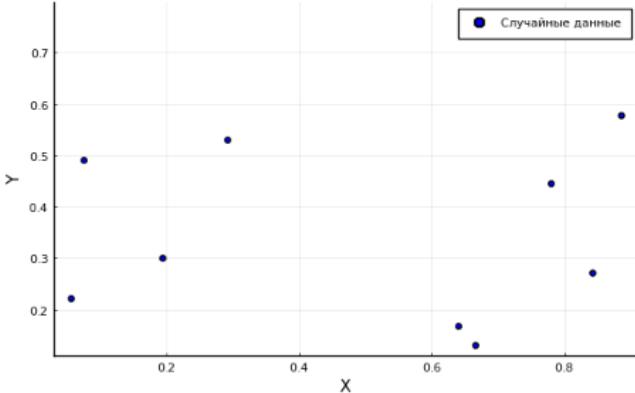


Рис. 2.53: Решение задания №7

Выполнение задания №8 (рис. 2.54):

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

```
[77]: # Генерация случайных данных для 3D графика
x_vals = rand(10) # 10 случайных значений по оси X
y_vals = rand(10) # 10 случайных значений по оси Y
z_vals = rand(10) # 10 случайных значений по оси Z
# Построение 3-мерного точечного графика
scatter3d(x_vals, y_vals, z_vals, label="Случайные данные", color=:blue, marker=:o, linewidth=2)
# Добавление подписей осей и заголовка
xlabel!("Ось X")
ylabel!("Ось Y")
zlabel!("Ось Z")
title!("3-мерный точечный график случайных данных")
```

[77]: 3-мерный точечный график случайных данных

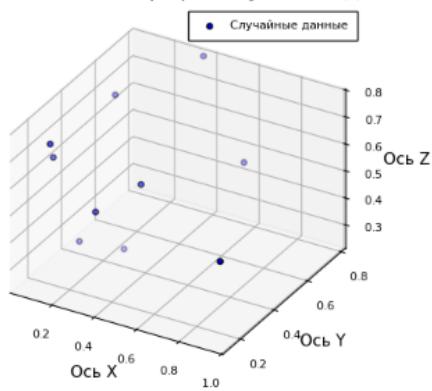


Рис. 2.54: Решение задания №8

Выполнение задания №9 (рис. 2.55):

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

```
# Подготовим данные для анимации
x = 0:0.1:10 # Диапазон значений аргумента (от 0 до 10)
y_vals = sin.(x) # Значения синусоиды (переименовали у в y_vals)
# Создаем анимацию
anim = @animate for i in 1:length(x)
    plot(x[1:i], y_vals[1:i], label="Синусоиды", color=:blue, linewidth=2)
    xlabel!("x")
    ylabel!("sin(x)")
    title!("Построение синусоиды")
end
# Сохранение анимации в файл
gif(anim, "sin_wave_animation.gif", fps=10)
```

[Info: Saved animation to c:/Users/timur/OneDrive/Desktop/sin_wave_animation.gif

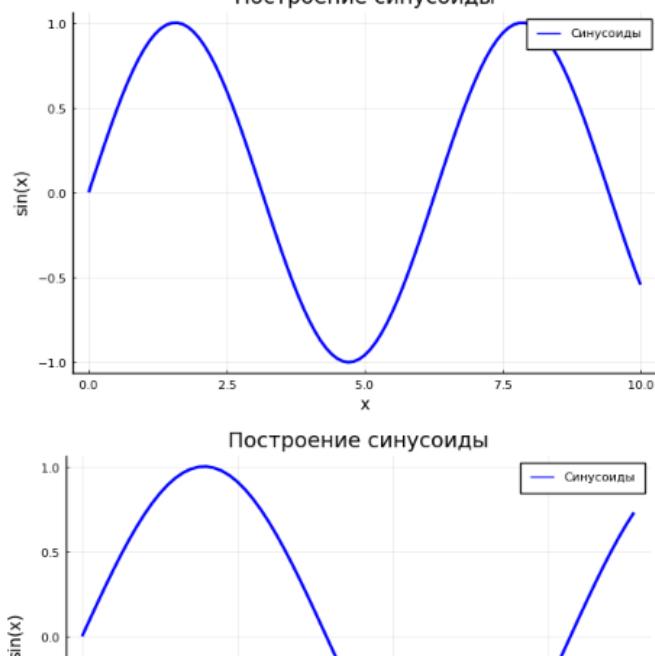


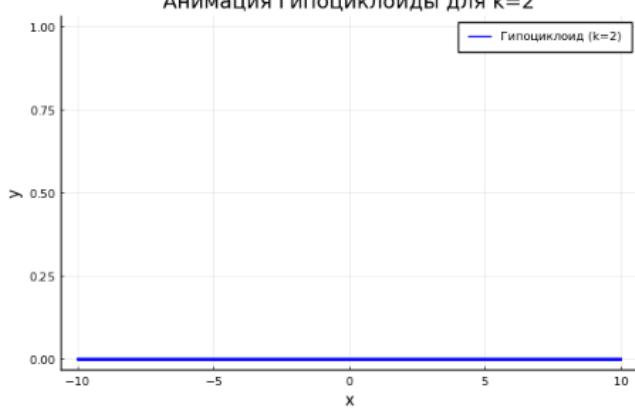
Рис. 2.55: Решение задания №9

Выполнение задания №10 (рис. 2.56):

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

```
]# Функция для вычисления координат гипоциклоиды
function hypocycloid(R, r, theta)
    x = (R - r) * cos(theta) + r * cos((R - r) / r * theta)
    y = (R - r) * sin(theta) - r * sin((R - r) / r * theta)
    return x, y
end
# Функция для создания анимации гипоциклоиды для целого значения k
function create_hypocycloid_animation(k)
    R = 10 # Радиус большой окружности
    r = R / k # Радиус маленькой окружности
    theta = 0:0.05:2 * pi # Узел от 0 до 2pi, шаг 0.05
    x, y = hypocycloid(R, r, theta)
    anim = @animate for i in 1:length(theta) # Итерация по всем углам
        plot(x[1:i], y[1:i], label="Гипоциклоид (k=$k)", color=:blue, linewidth=2,
            xlabel="x", ylabel="y", title="Анимация гипоциклоиды для k=$k", legend=:topright)
    end
    # Сохранение анимации в файл
    gif(anim, "hypocycloid_k_$.gif", fps=10)
end
# Создание анимации для целого значения k = 2
create_hypocycloid_animation(2)
```

[Info: Saved animation to c:\Users\timur\OneDrive\Desktop\hypocycloid_k_2.gif



```
]# Анимация гипоциклоиды для k=2
plot(x[1:i], y[1:i], label="Гипоциклоид (k=2)", color=:blue, linewidth=2,
```

Рис. 2.56: Решение задания №10

Выполнение задания №11 (рис. 2.57):

```
[80]: # Функция для вычисления координат эпициклоиды
function epicycloid(R, r, theta)
    x = (R + r) * cos(theta) - r * cos((R + r) / r * theta)
    y = (R + r) * sin(theta) - r * sin((R + r) / r * theta)
    return x, y
end
# Функция для создания анимации эпициклоиды для целого значения k
function create_epicycloid_animation(k)
    R = 10 # Радиус большой окружности
    r = R / k # Радиус маленькой окружности
    theta = 0:0.05:2 * pi # Угол от 0 до 2π, шаг 0.05
    x, y = epicycloid(R, r, theta)
    anim = @animate for i in 1:length(theta) # Итерация по всем углам
        plot(x[1:i], y[1:i], label="Эпициклоид (k=$k)", color=:blue, linewidth=2,
             xlabel="x", ylabel="y", title="Анимация эпициклоиды для k=$k", legend=:topright)
    end
    # Сохранение анимации в файл
    gif(anim, "epicycloid_k_$.gif", fps=10)
end
# Создание анимации для целого значения k = 2
create_epicycloid_animation(2)

[ Info: Saved animation to c:\Users\timur\OneDrive\Desktop\epicycloid_k_2.gif
```

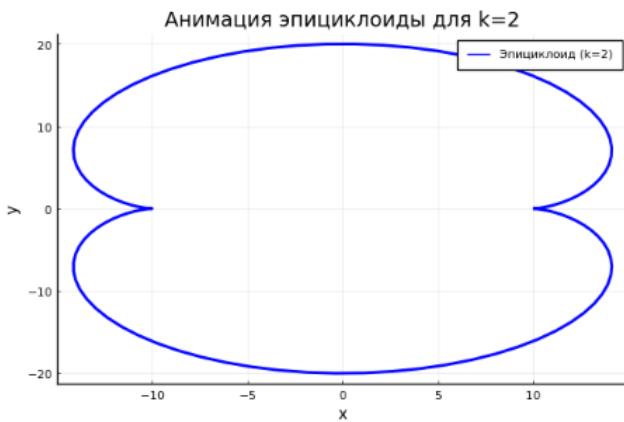


Рис. 2.57: Решение задания №11

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/>