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

Построение графиков

Выполнил: Зиязетдинов Алмаз Радикович,
НПИбд-01-22, 1132222010

Содержание

1 Цель работы

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

2 Выполнение лабораторной работы

2.1 Основные пакеты для работы с графиками в Julia

Julia поддерживает несколько пакетов для работы с графиками. Использование того или иного пакета зависит от целей, преследуемых пользователем при построении. Стандартным для Julia является пакет Plots.jl.

Рассмотрим построение графика функции f(x) = (3x2 + 6x − 9)e−0,3x разными способами (рис. 1 - (рис. 2):

1. Основные пакеты для работы с графиками в Julia

[1]:
```julia
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) = (3x² + 6x - 9)e⁻⁰·³ˣ",         # Легенда
    color="blue",                                # Цвет графика
    lw=2,                                        # Толщина линии
    grid=true)                                   # Включение сетки
```

[1]:

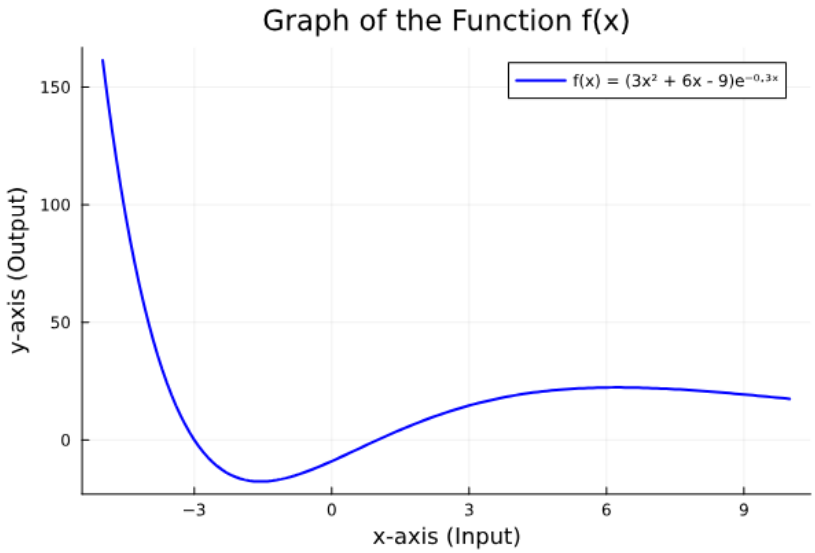


Рис. 1: График функции, построенный при помощи gr()

[3]:
```
using Plots  # Подключаем Plots

# Задание функции (поэлементные операции для векторов)
f(x) = (3x.^2 + 6x .- 9) .* exp.(-0.3x)
# Массив x от -5 до 10 с шагом 0.1 (151 точка)
x = collect(range(-5, 10, length=151))

# Массив y
y = f(x)
gr()  # Включаем бэкенд gr() — работает без Python

# Построение графика с опциями
plot(x, y,
     title="График функции f(x)",
     xlabel="x - ось",
     ylabel="y - ось",
     label="f(x) = (3x² + 6x - 9)·exp(-0.3x)",
     color=:blue,
     linewidth=2)
```

[3]:

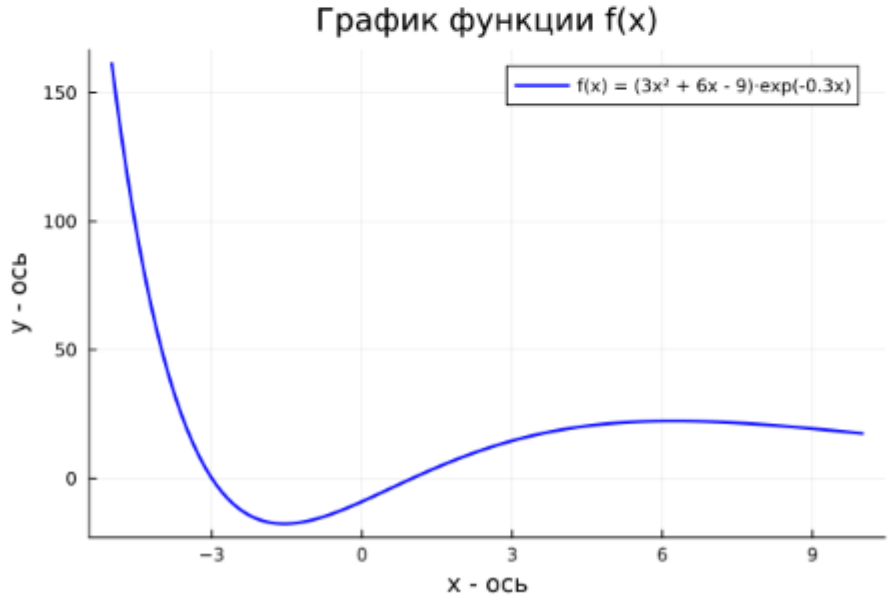


Рис. 2: График функции, построенный при помощи pyplot()

2.2 Опции при построении графика

На примере графика функции sin(x) и графика разложения этой функции в ряд Тейлора рассмотрим дополнительные возможности пакетов для работы с графикой (рис. 3 - рис. 5):

Рис. 3: График функции sin(x)

[12]:
```
# 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
plot(sin_taylor,
title="разложение исходной функции в ряд Тейлора",
xlabel="Ось x",
ylabel="Ось y",
color="red")
```

[12]:

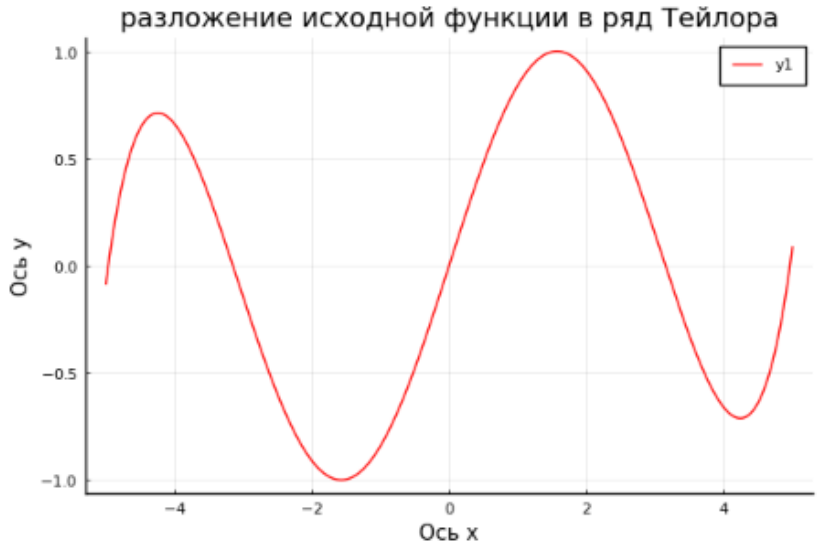


Рис. 4: График функции разложения исходной функции в ряд Тейлора

20]:

```
# построение двух функций на одном графике:
 plot(sin_theor,
label1="sin(x)",
color="blue")
 plot!(sin_taylor,
label1="Ряд Тейлора (5 членов)",
color="red"
)
title!("Сравнение sin(x) и его ряда ")
xlabel!("Ось x")
ylabel!("Ось y")
```

20]:

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

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

```
        s += (-1)^i * x^(2*i+1) / factorial(2*i+1)
    end
    return s
end


x = range(-5, 5, length=200)


plot(sin_taylor,                    # функция sin(x) в виде ряда Тейлора
     label = "sin(x), разложение в ряд Тейлора",
     line = (:blue, 0.3, 6, :solid),
     size = (800, 500),
     xticks = (-5:0.5:5),
     yticks = (-1:0.1:1),
     xtickfont = font(12, "Times New Roman"),
     ytickfont = font(12, "Times New Roman"),
     ylabel = "y",
     xlabel = "x",
     title = "Разложение в ряд Тейлора",
     xrotation = rad2deg(pi/4),
     fillrange = 0,
     fillalpha = 0.5,
     fillcolor = :lightgoldenrod,
     background_color = :ivory)

# === ДОБАВЛЯЕМ ТЕОРЕТИЧЕСКУЮ КРИВУЮ ===
plot!(sin_theor,
      label = "sin(x), теоретическое значение",
      line = (:black, 1.0, 2, :dash))
```

:7]:

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

2.3 Точечный график

Как и при построении обычного графика для точечного графика необходимо задать массив значений , посчитать или задать значения y, задать опции построения графика (рис. 7):

```
[29]: # параметры распределения точек на плоскости:
x = range(1,10,length=10)
y = rand(10)
# параметры построения графика:
plot(x, y,
   ylabel = "Ось y",
    xlabel = "Ось x",
   label= "Точки данных",
seriestype = :scatter,
title = "Точечный график распределения"
)
```

[29]:

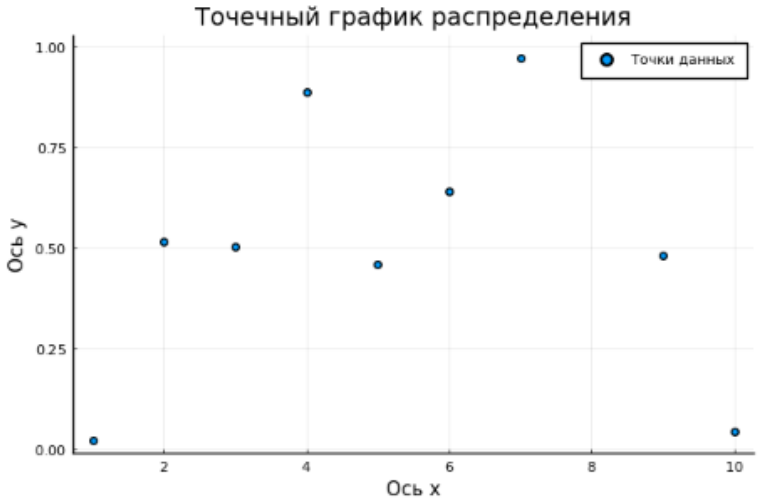


Рис. 7: График десяти случайных значений на плоскости (простой точечный график)

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

[32]:
```
n = 50
x = rand(n)
y = rand(n)
ms = rand(50) * 30
scatter(x, y,
   title="Точечный график с случайными размерами маркеров",
    ylabel = "Ось y",
    xlabel = "Ось x",
   label= "Точки данных",

   markersize=ms)
```

[32]:

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

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

```
# параметры распределения точек в пространстве:
n = 50
x = rand(n)
y = rand(n)
z = rand(n)
ms = rand(50) * 30
# параметры построения графика:
scatter(x, y, z,
    title="Точечный график в 3D",
    ylabel = "Ось y",
    xlabel = "Ось x",
   zlabel = "Ось z",
   label= "Точки данных",

   markersize=ms)
```

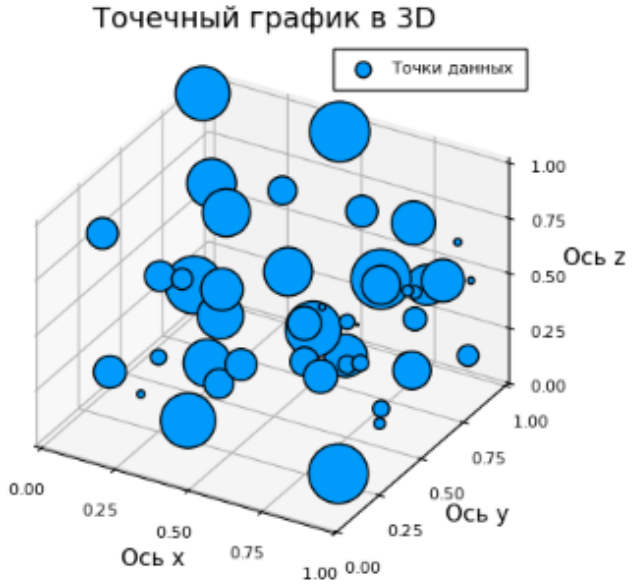


Рис. 9: График пятидесяти случайных значений в пространстве с различными опциями отображения (3-мерный точечный график с кодированием значения размером точки)

2.4 Аппроксимация данных

Аппроксимация — научный метод, состоящий в замене объектов их более простыми аналогами, сходными по своим свойствам.

Для демонстрации зададим искусственно некоторую функцию, в данном случае похожую по поведению на экспоненту (рис. 10):

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
```

[10]:

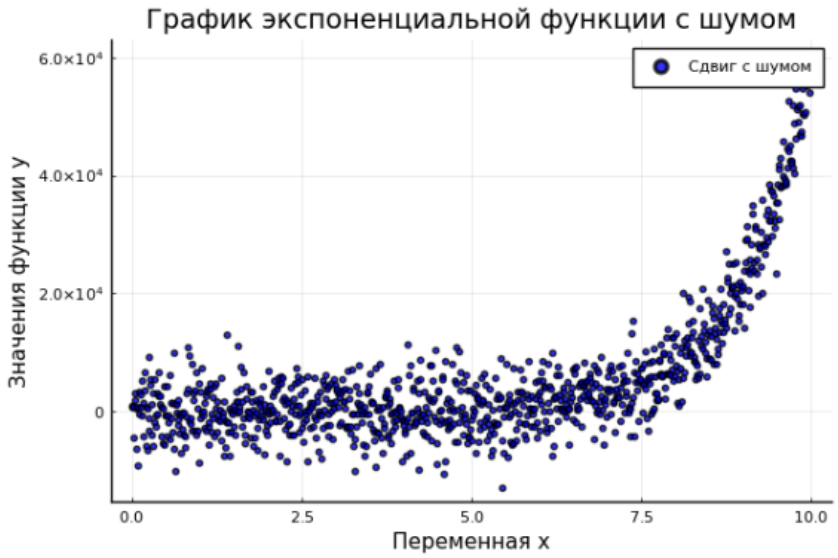


Рис. 10: Пример функции

Аппроксимируем полученную функцию полиномом 5-й степени (рис. 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
```

[12]:

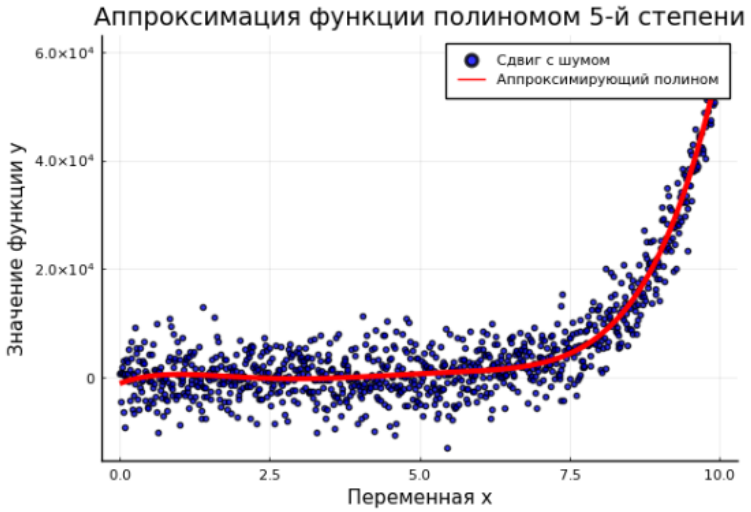


Рис. 11: Пример аппроксимации исходной функции полиномом 5-й степени

2.5 Две оси ординат

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

Пример первой траектории (рис. 12):

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
)
```

[13]:

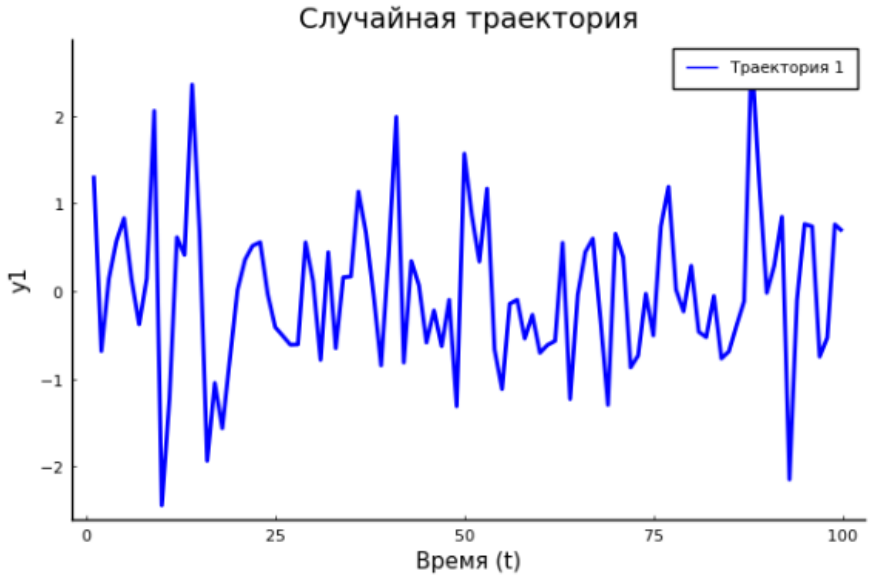


Рис. 12: Пример отдельно построенной траектории

2.6 Полярные координаты

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

6. Полярные координаты

[15]:
```
# Функция в полярных координатах
r(θ) = 1 + cos(θ) * sin(θ)^2
# Полярная система координат
θ = range(0, stop=2π, length=50)
# Построение графика функции в полярных координатах
plot(θ, r.(θ),
    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]:

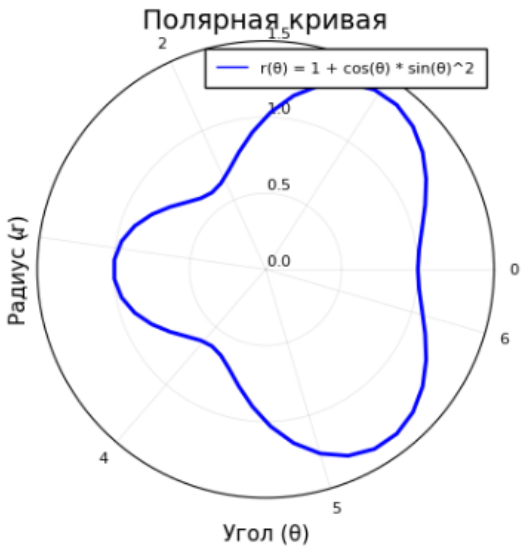


Рис. 13: График функции, заданной в полярных координатах

2.7 Параметрический график

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

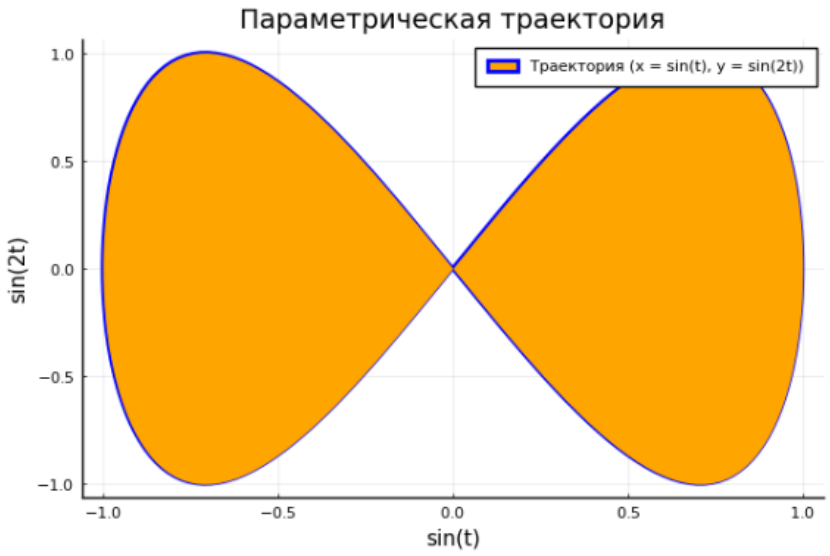


Рис. 14: Параметрический график кривой на плоскости

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

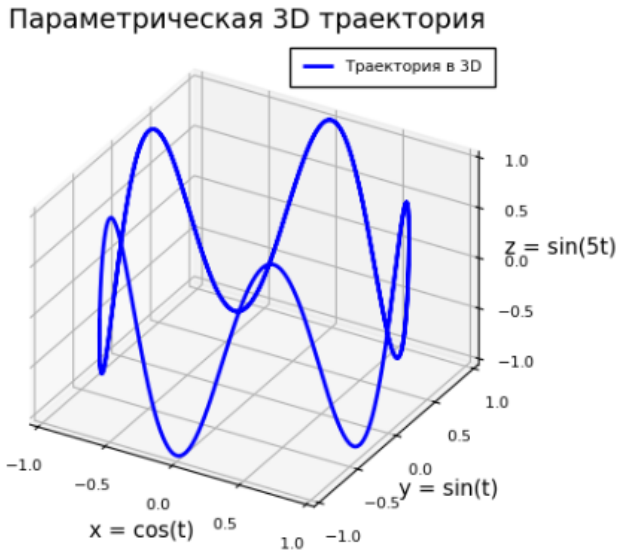


Рис. 15: Параметрический график кривой в пространстве

2.8 График поверхности

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

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
)
```

[26]:

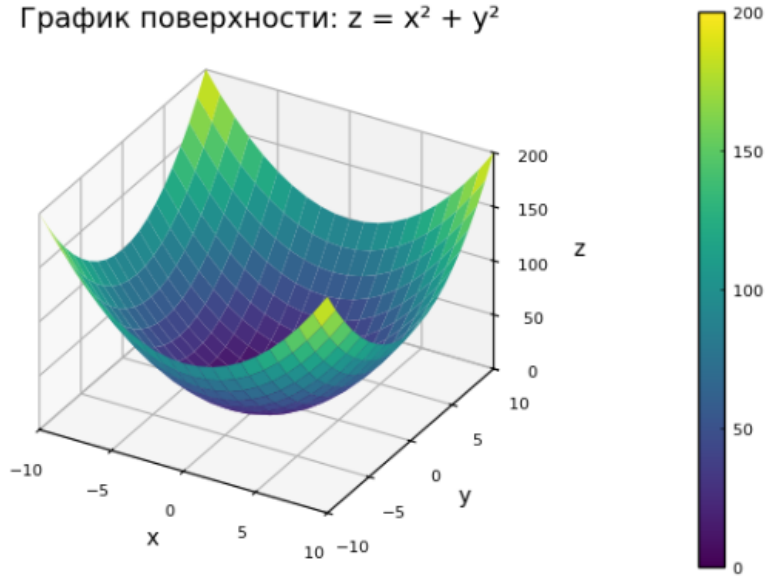


Рис. 16: График поверхности (использована функция surface())

Также можно воспользоваться функцией plot() с заданными параметрами (рис. 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 = x² + y²",  # Заголовок
    xlabel="x",                             # Подпись оси X
    ylabel="y",                             # Подпись оси Y
    zlabel="z",                             # Подпись оси Z
    legend=false                            # Отключение легенды
)
```

[28]:

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

Можно задать параметры сглаживания (рис. 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 = x² + y²", # 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]:

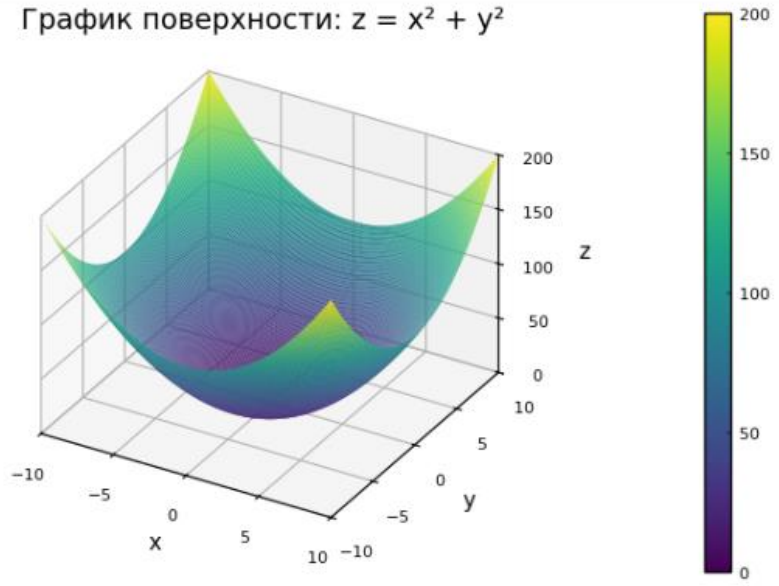


Рис. 18: Сглаженный график поверхности

Можно задать определённый угол зрения (рис. 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]:

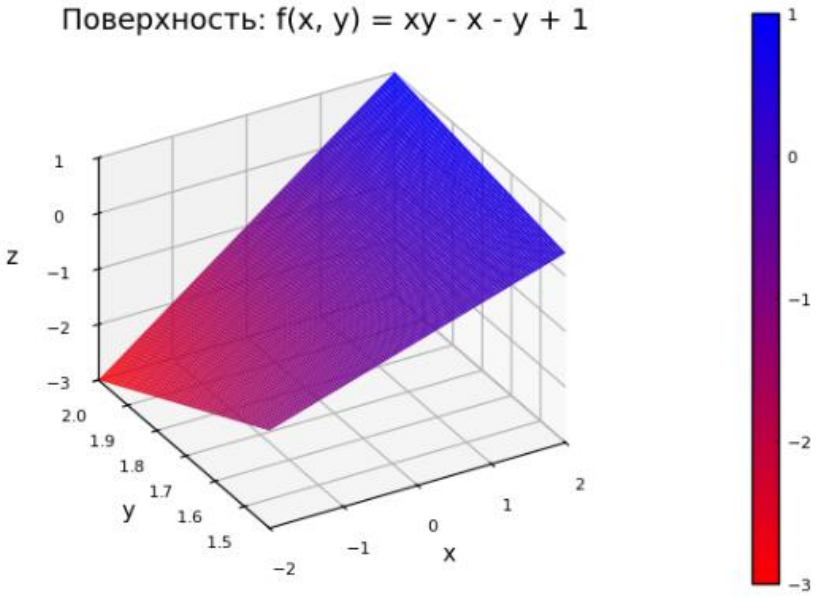


Рис. 19: График поверхности с изменённым углом зрения

2.9 Линии уровня

Линией уровня некоторой функции от двух переменных называется множество точек на координатной плоскости, в которых функция принимает одинаковые значения. Линий уровня бесконечно много, и через каждую точку области определения можно провести линию уровня.

С помощью линий уровня можно определить наибольшее и наименьшее значение исходной функции от двух переменных. Каждая из этих линий соответствует определённому значению высоты.

Поверхности уровня представляют собой непересекающиеся пространственные поверхности.

Рассмотрим поверхность, заданную функцией g(x, y) = (3x + y2)| sin(x) + cos(y)| (рис. 20):

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
)
```

[38]:

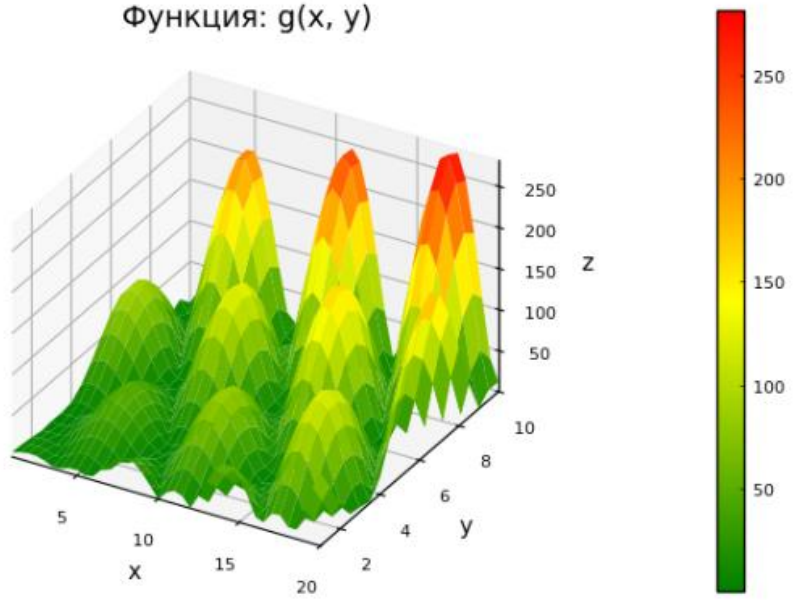


Рис. 20: График поверхности, заданной функцией g(x, y) = (3x + y2)| sin(x) + cos(y)|

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

[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
)
```

[40]:

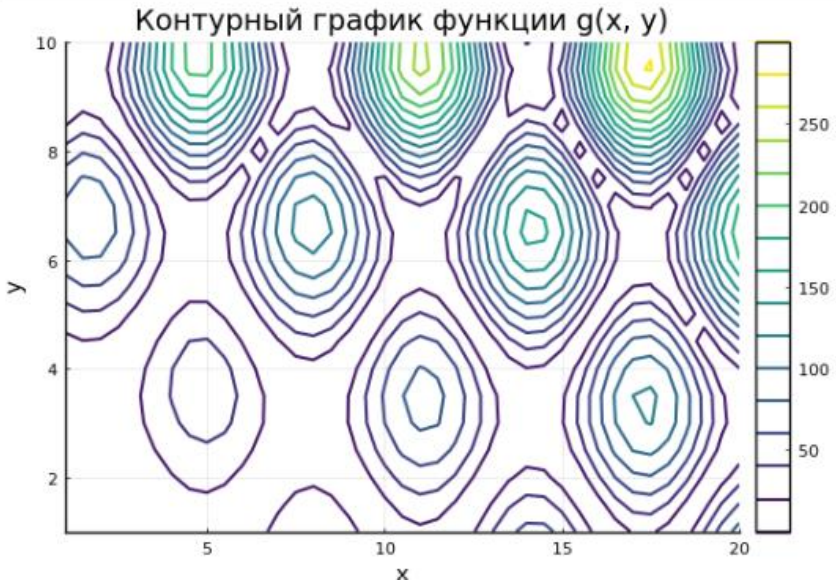


Рис. 21: Линии уровня

Можно дополнительно добавить заливку цветом (рис. 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
)
```

[42]:

Рис. 22: Линии уровня с заполнением

2.10 Векторные поля

Если каждой точке некоторой области пространства поставлен в соответствие вектор с началом в данной точке, то говорят, что в этой области задано векторное поле.

Векторные поля задают векторными функциями.

Для функции h(x, y) = x3 − 3x + y2 сначала построим её график (рис. 23) и линии уровня (рис. 24):

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]:

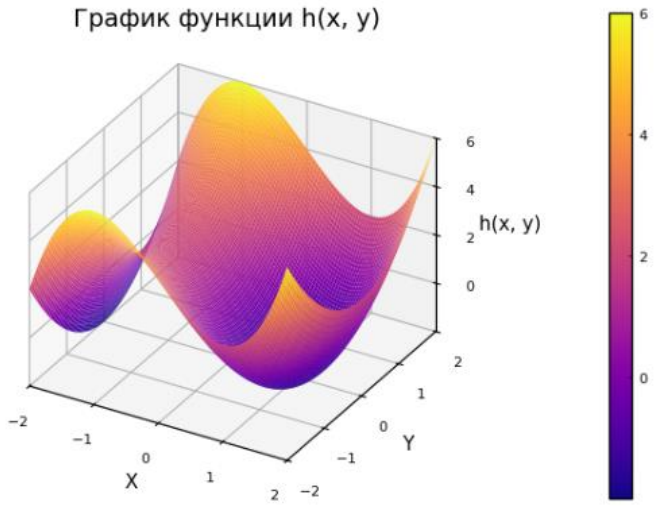


Рис. 23: График функции h(x, y) = x3 − 3x + y2

[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
)
```

[50]:

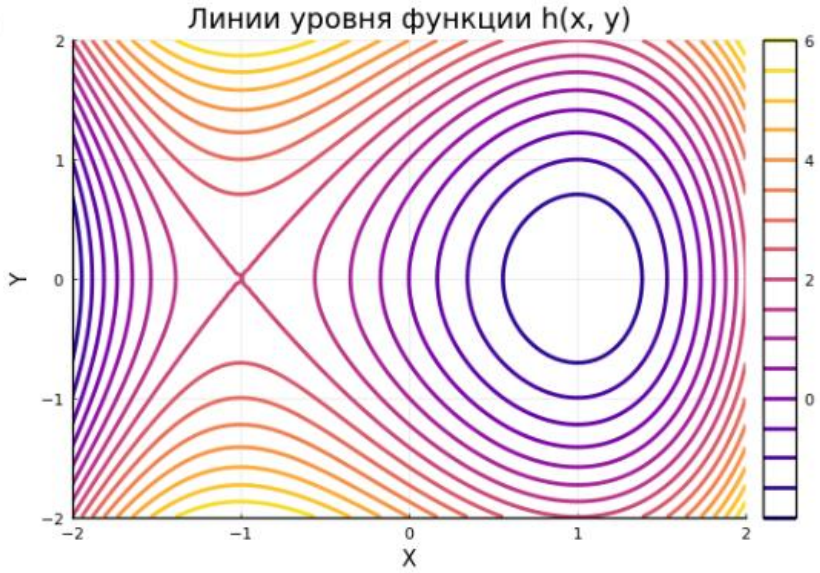


Рис. 24: Линии уровня для функции h(x, y) = x3 − 3x + y2

Векторное поле можно охарактеризовать векторными линиями. Каждая точка векторной линии является началом вектора поля, который лежит на касательной в данной точке.

Для нахождения векторной линии требуется решить дифференциальное уравнение.

2.11 Анимация

Технически анимированное изображение представляет собой несколько наложенных изображений (или построенных в разных точках графиках) в одном файле.

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

Строим поверхность (рис. 25):

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]:

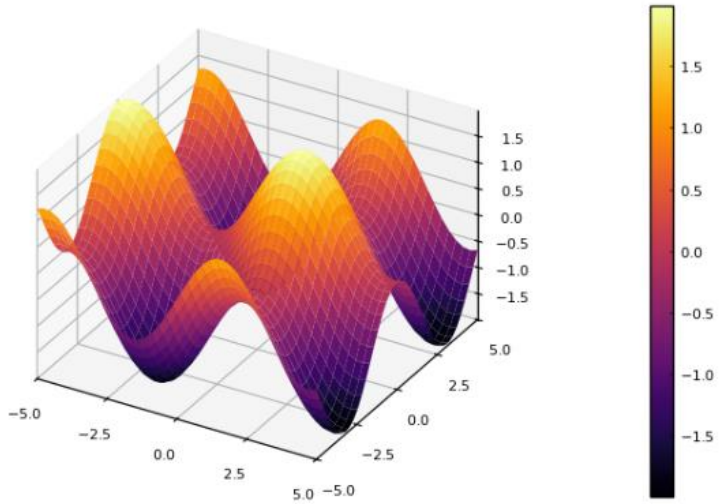


Рис. 25: Статичный график поверхности

Добавляем анимацию (рис. 26):

[62]:
```
# анимация:
X = Y = range(-5,stop=5,length=40)
@gif for i in range(0,stop=2π,length=100)
surface(X, Y, (x,y) -> sin(x+10sin(i))+cos(y))
end
```

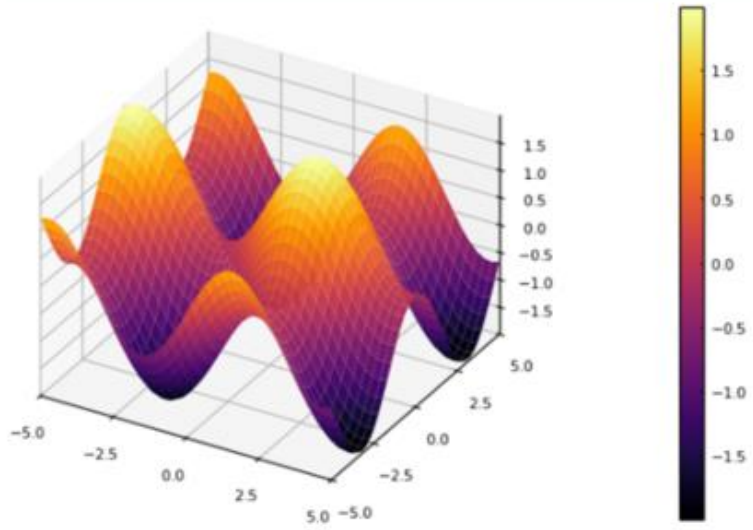


Рис. 26: Анимированный график поверхности

2.12 Гипоциклоида

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

Построим большую окружность (рис. 27):

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:
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)
```

[42]:

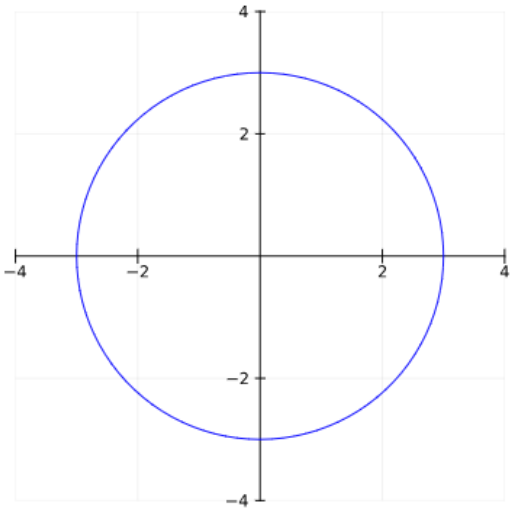


Рис. 27: Большая окружность гипоциклоиды

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

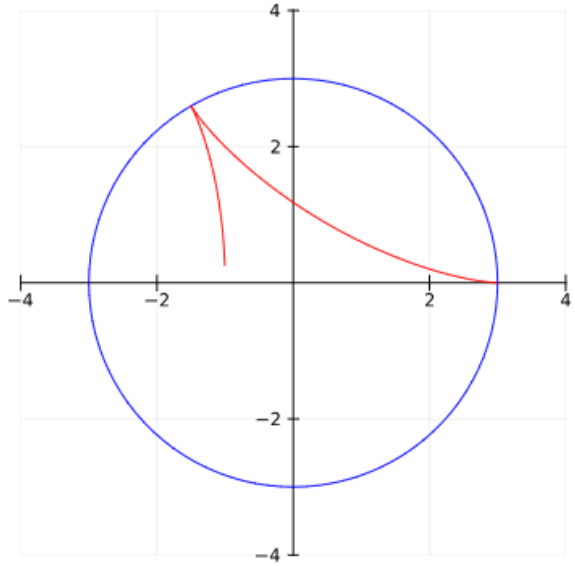


Рис. 28: Половина пути гипоциклоиды

Добавляем малую окружность гипоциклоиды (рис. 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]:

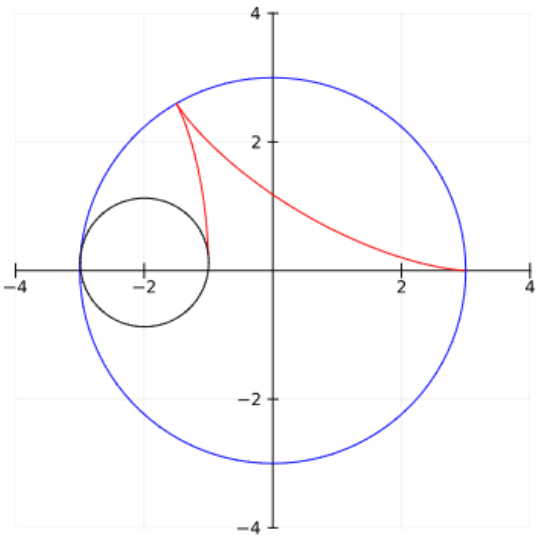


Рис. 29: Малая окружность гипоциклоиды

Добавим радиус для малой окружности (рис. 30):

```
[46]: # радиус малой окружности:
      xl = transpose([radius*(k-1)*cos(t[end]) x[end]])
      yl = transpose([radius*(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)
```

[46]:

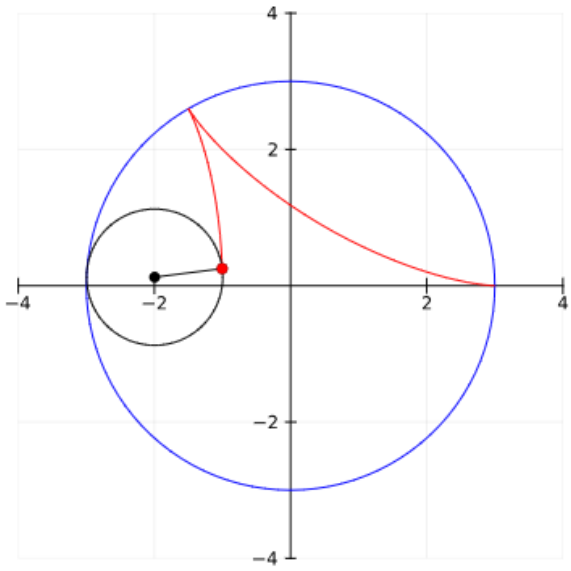


Рис. 30: Малая окружность гипоциклоиды с добавлением радиуса

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

```
# Анимация — исправлено: пробел в "for i" → "fori"
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 = r*(k-1)*cos.(t) + r*cos.((k-1)*t)
    y = r*(k-1)*sin.(t) - r*sin.((k-1)*t)
    plot!(x, y, c=:red)

    # Малая окружность
    xc = r*(k-1)*cos(t[end]) .+ r*cos.(θ)
    yc = r*(k-1)*sin(t[end]) .+ r*sin.(θ)
    plot!(xc, yc, c=:black)

    # Радиус малой окружности
    xl = transpose([r*(k-1)*cos(t[end]) x[end]])
    yl = transpose([r*(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

# Сохраняем анимацию
gif(anim, "hypocycloid_full.gif", fps=30)
```

```
[ Info: Saved animation to C:\Users\Олеся\hypocycloid_full.gif
```

[19]:

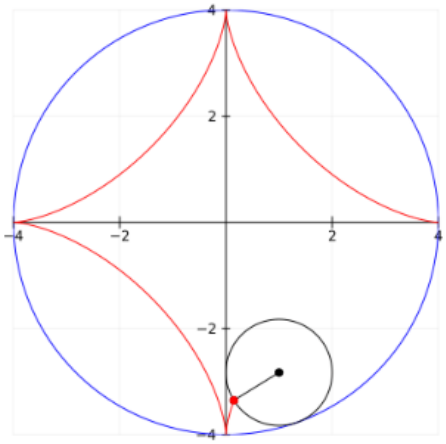


Рис. 31: Малая окружность гипоциклоиды с добавлением радиуса

2.13 Errorbars

В исследованиях часто требуется изобразить графики погрешностей измерения.

Построим график исходных значений (рис. 32):

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)
)
```

[68]:

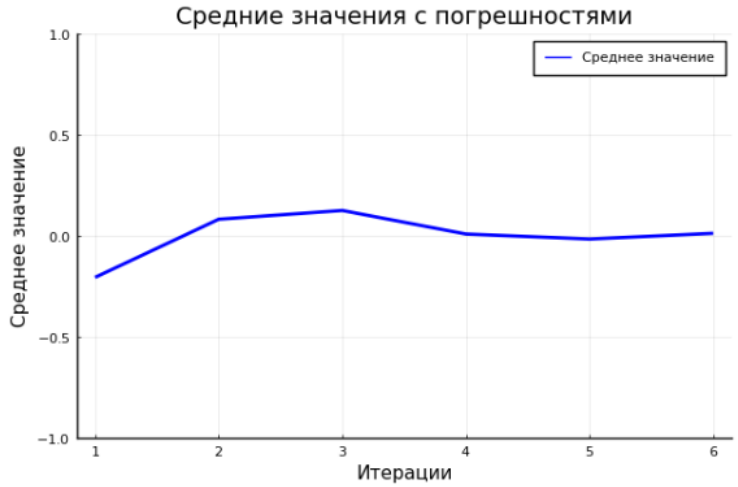


Рис. 32: График исходных значений

Построим график отклонений от исходных значений (рис. 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]:

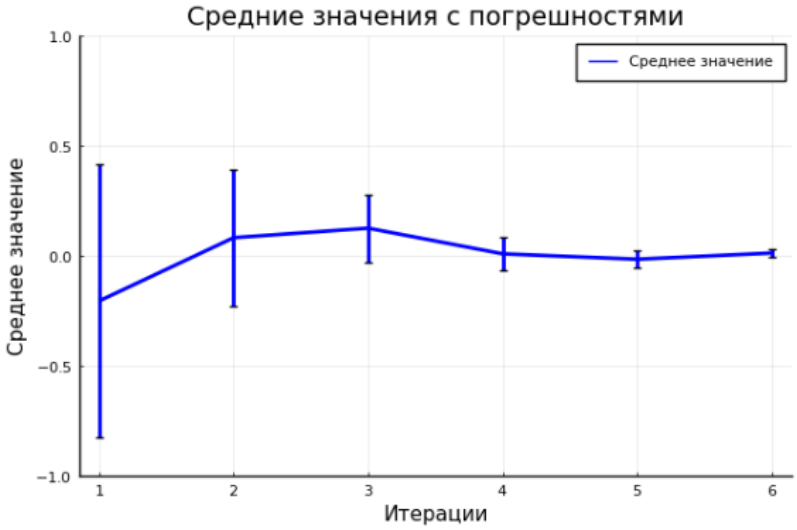


Рис. 33: График исходных значений с отклонениями

Повернём график (рис. 34):

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

[80]:

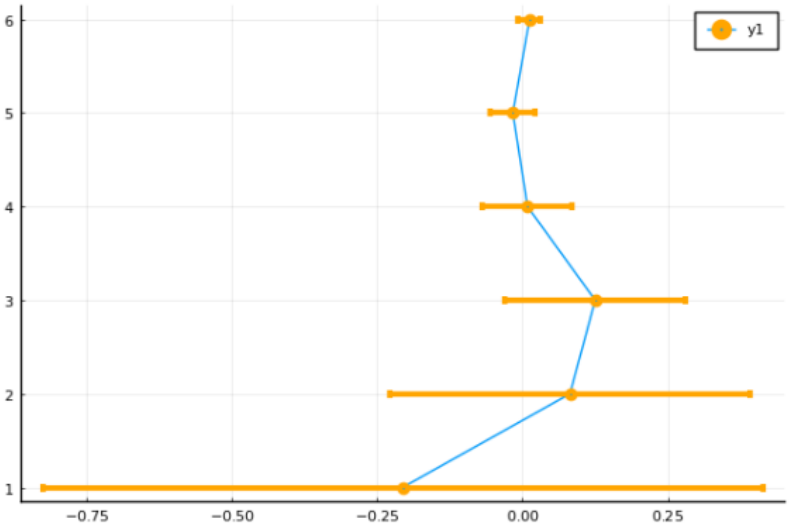


Рис. 34: Поворот графика

Заполним область цветом (рис. 35):

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

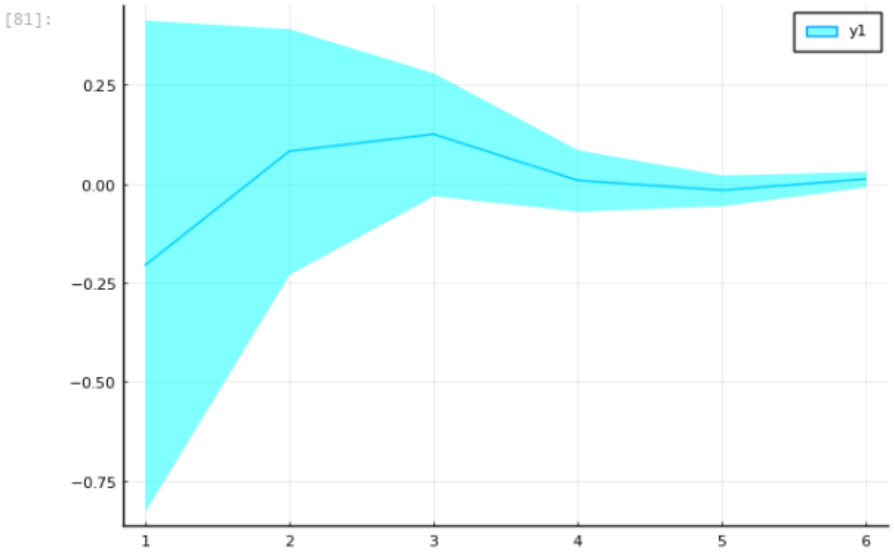


Рис. 35: Заполнение цветом

Можно построить график ошибок по двум осям (рис. 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)
      )
```

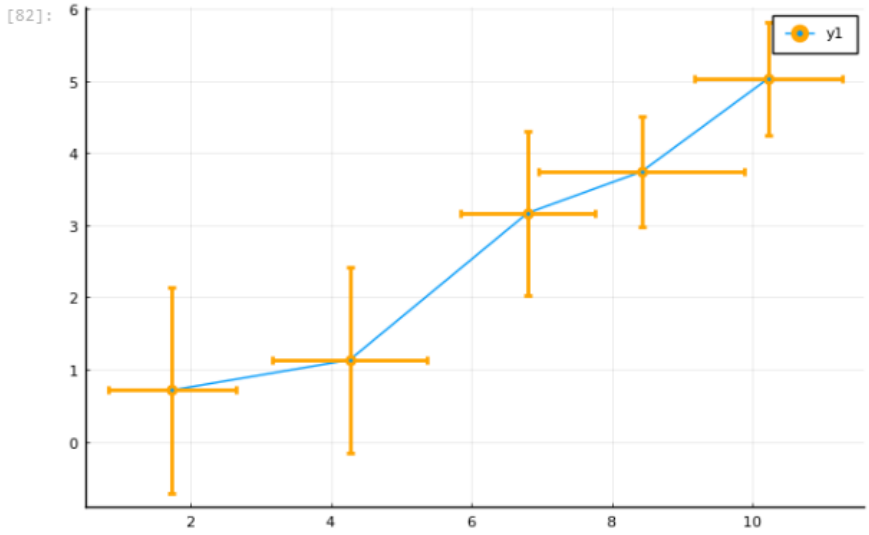


Рис. 36: График ошибок по двум осям

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

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

[83]:

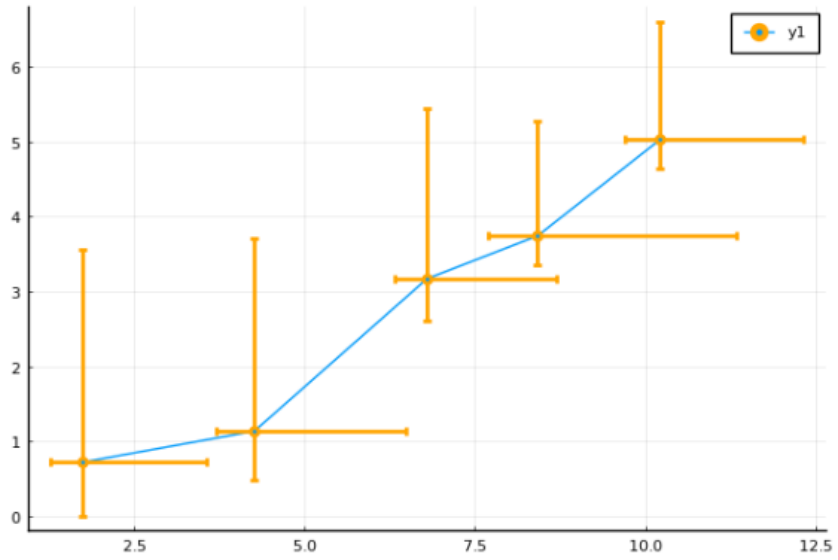


Рис. 37: График асимметричных ошибок по двум осям

2.14 Использование пакета Distributions

Строим гистограмму (рис. 38):

13. Использование пакета Distributions

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

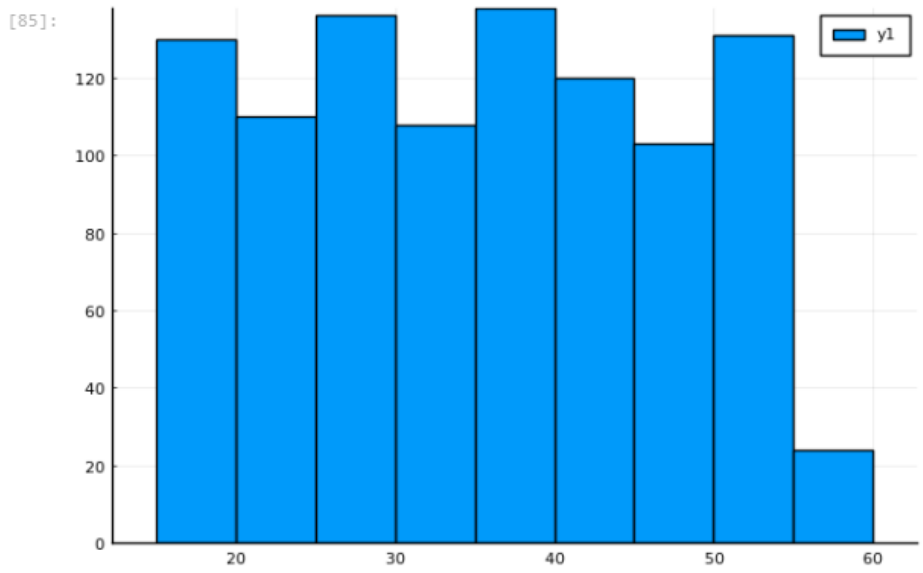


Рис. 38: Гистограмма, построенная по массиву случайных чисел

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

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

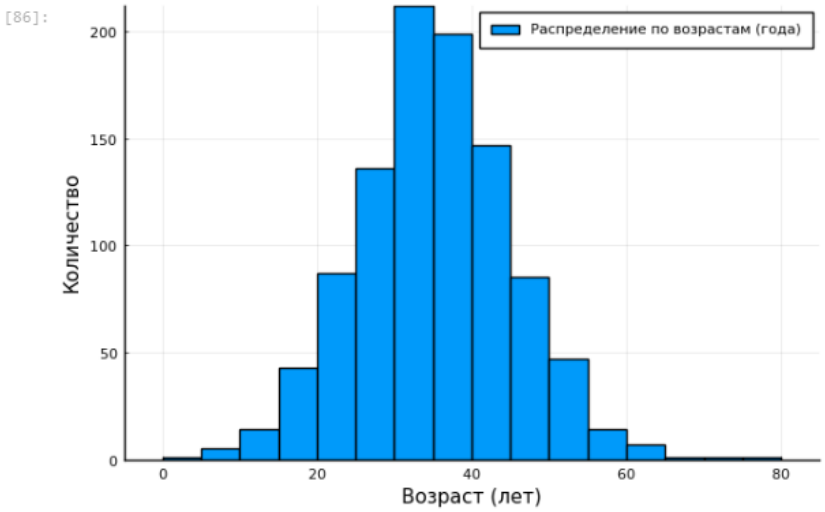


Рис. 39: Гистограмма нормального распределения

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

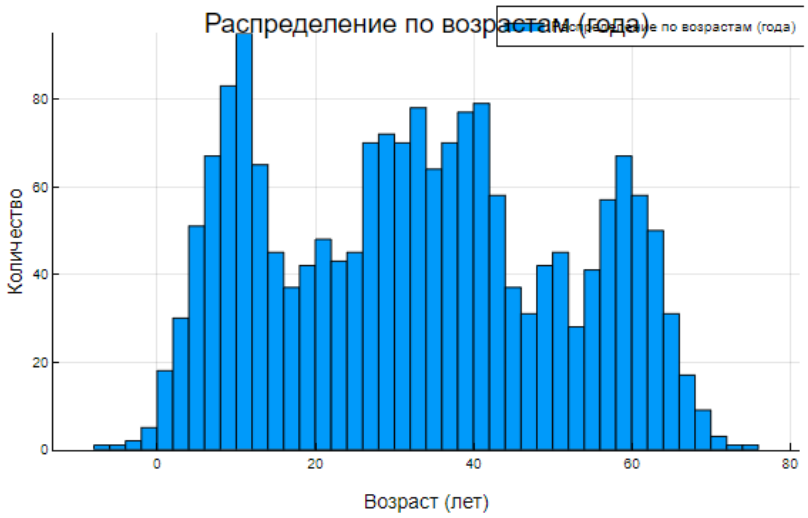


Рис. 40: Гистограмма распределения людей по возрастам

2.15 Подграфики

Определим макет расположения графиков. Команда layout принимает кортеж layout = (N, M), который строит сетку графиков NxM. Например, если задать layout = (4,1) на графике четыре серии, то получим четыре ряда графиков (рис. 41):

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

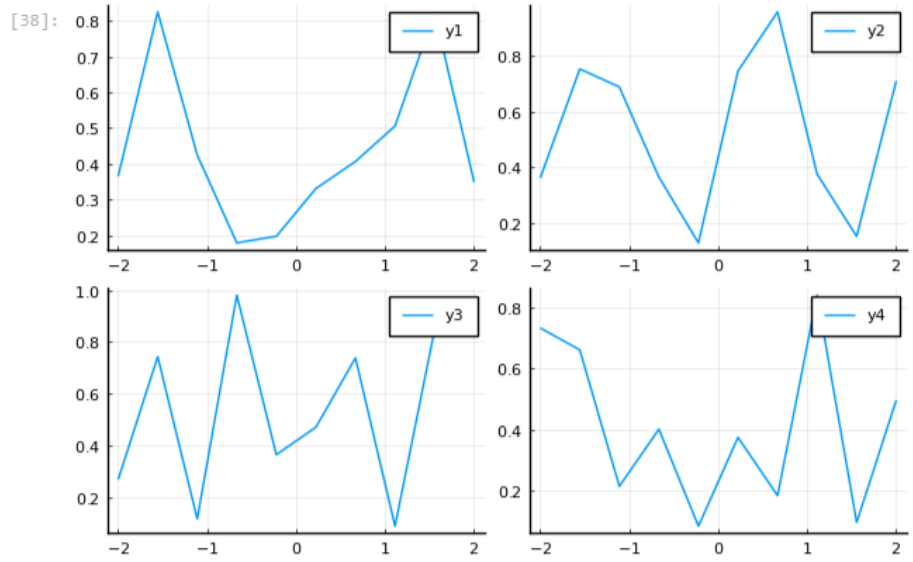


Рис. 41: Серия из 4-х графиков в ряд

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

```
[40]: plot(x,y,
      size=(600,300),
      layout = grid(4,1,heights=[0.2,0.3,0.4,0.1])
      )
```

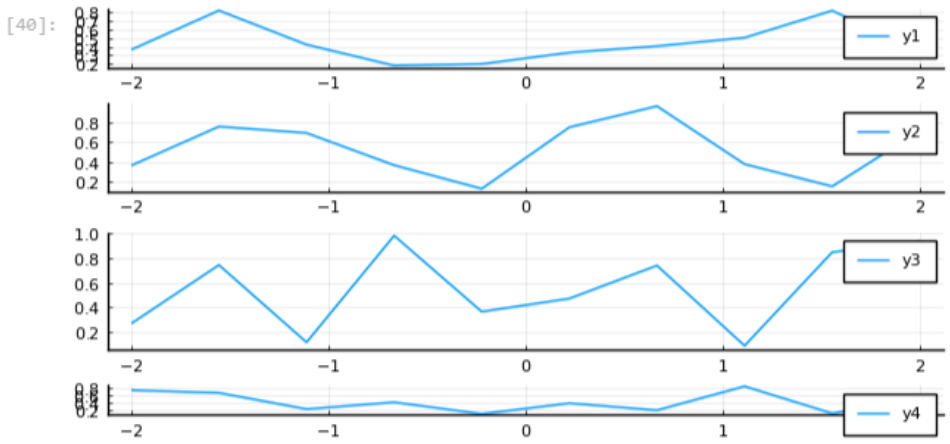


Рис. 42: Серия из 4-х графиков в сетке

Аргумент heights принимает в качестве входных данных массив с долями желаемых высот. Если в сумме дроби не составляют 1,0, то некоторые подзаголовки могут отображаться неправильно.

Можно сгенерировать отдельные графики и объединить их в один, например, в сетке 2 × 2 (рис. 43):

[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
)
```

[92]:

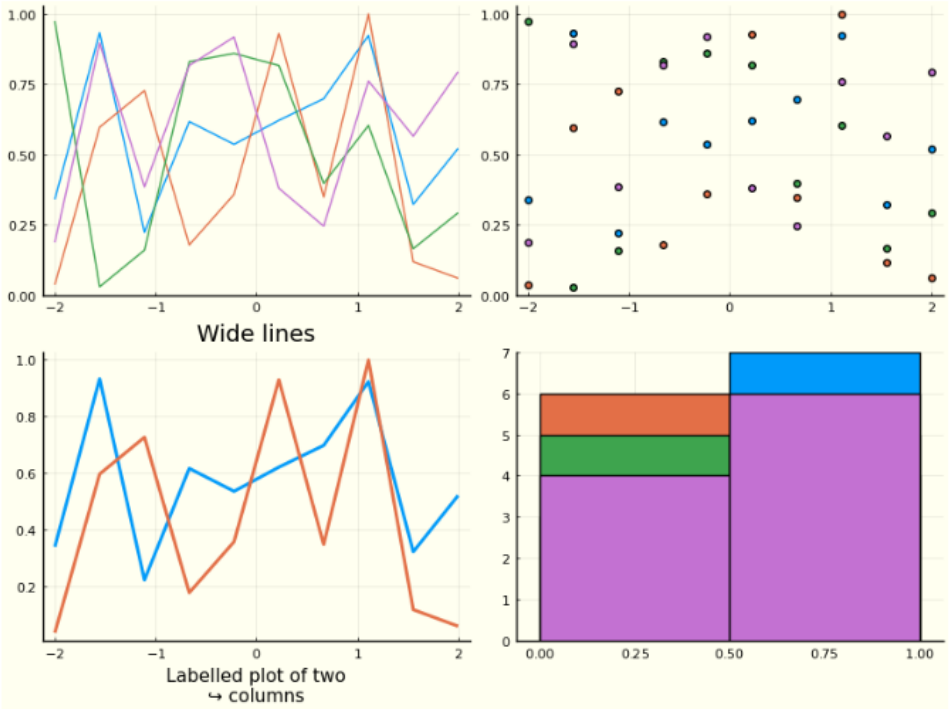


Рис. 43: Объединение нескольких графиков в одной сетке

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

Разнообразные варианты представления данных (рис. 44):

```
[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
)
```

[93]:

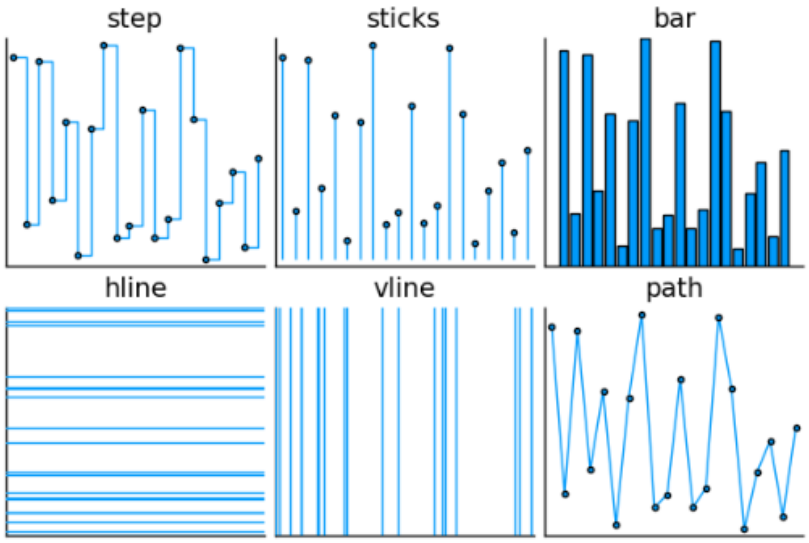


Рис. 44: Разнообразные варианты представления данных

Применение макроса [**layout?**] наиболее простой способ определения сложных макетов. Точные размеры могут быть заданы с помощью фигурных скобок, в противном случае пространство будет поровну разделено между графиками (рис. 45):

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

[94]:

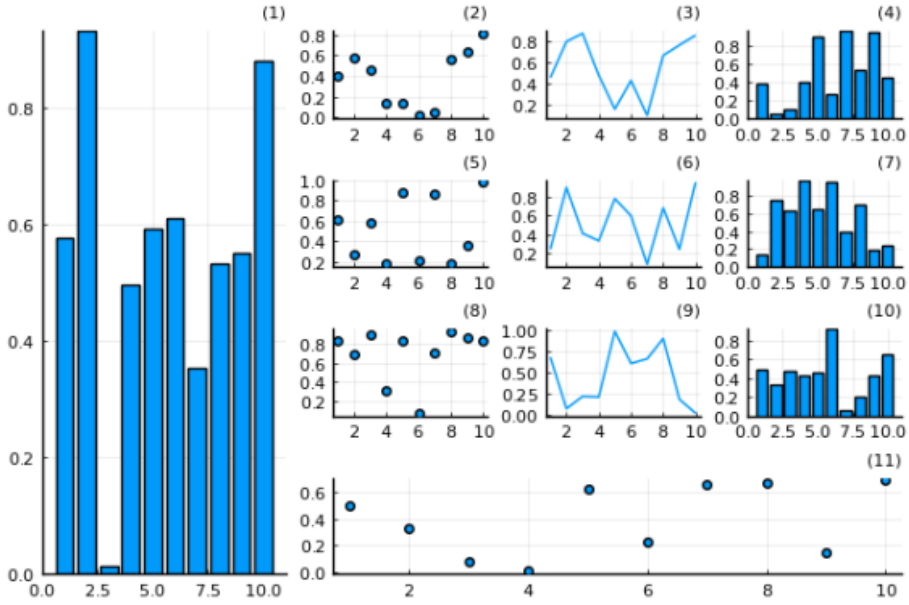


Рис. 45: Демонстрация применения сложного макета для построения графиков

2.16 Самостоятельное выполнение

Выполнение задания №1 (рис. 46):

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

```
[13]: # 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)
```

[13]:

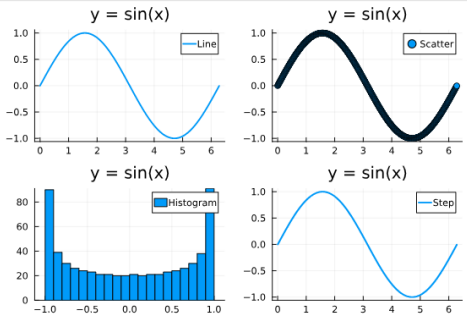


Рис. 46: Решение задания №1

Выполнение задания №2 (рис. 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:

[14]:
```
# 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)
```

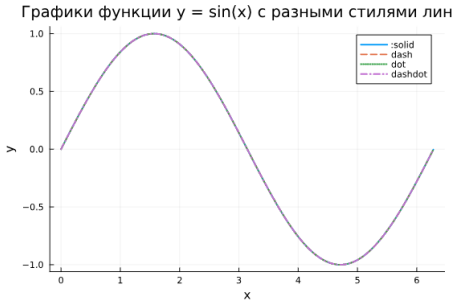


Рис. 47: Решение задания №2

Выполнение задания №3 (рис. 48):

[51]:
```
x = 0:0.01:0.2π
y = sin.(x)

plot(x, y, line=(:solid, 2, :blue), label="Сплошная")
plot!(x, y, line=(:dash, 2, :red), label="Пунктир")
plot!(x, y, line=(:dot, 2, :green), label="Точечная")
plot!(x, y, line=(:dashdot, 2, :purple), label="Штрих-пунктир")
title!("Типы линий")
```

[51]:

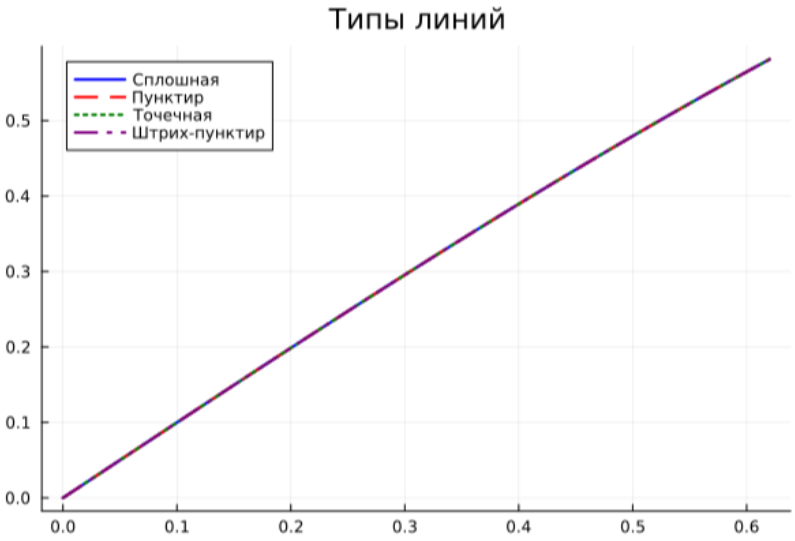


Рис. 48: Решение задания №3

Выполнение задания №4 (рис. 49):

```
[52]: x = range(0.1, 5, length=100)  # x > 0 для ln(x)
      y = π .* x.^2 .* log.(x)

      plot(x, y,
           color=:red,
           linewidth=2,
           label="y = πx² ln(x)",
           xlabel="x",
           ylabel="y",
           title="График функции y(x) = πx² ln(x)")
```

[52]:

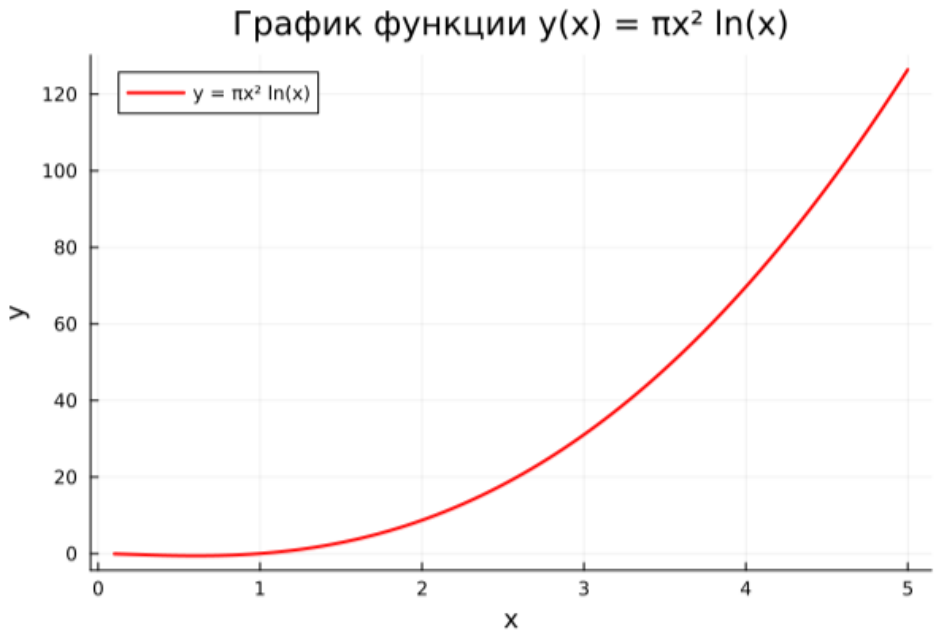


Рис. 49: Решение задания №4

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

5. Задайте вектор x = (3, 3.1, 3.2, ... , 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:

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

[22]:

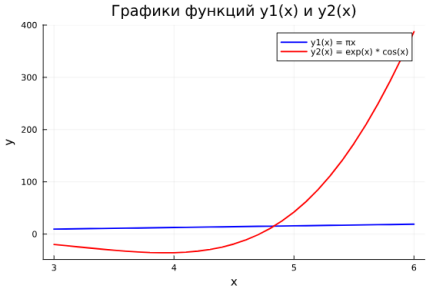


Рис. 50: Решение задания №5

[21]:
```
# Построение графиков с двумя осями ординат
p = plot(x, y1(x), label="y1(x) = πx", 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")
```

[21]:

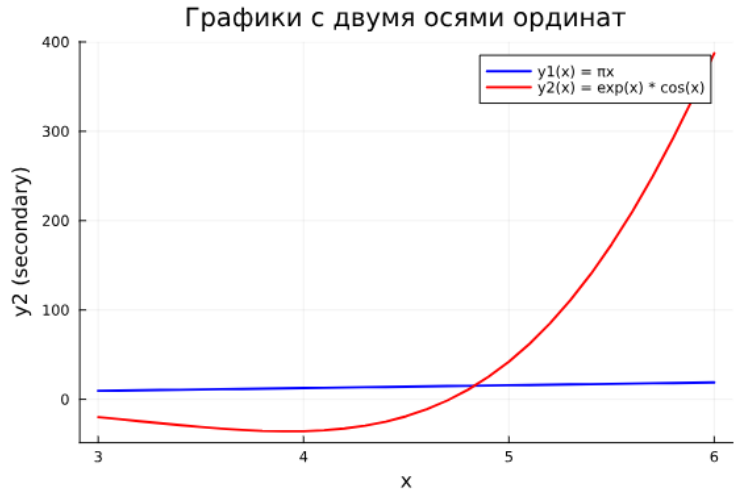


Рис. 51: Решение задания №5

Выполнение задания №6 (рис. 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]:

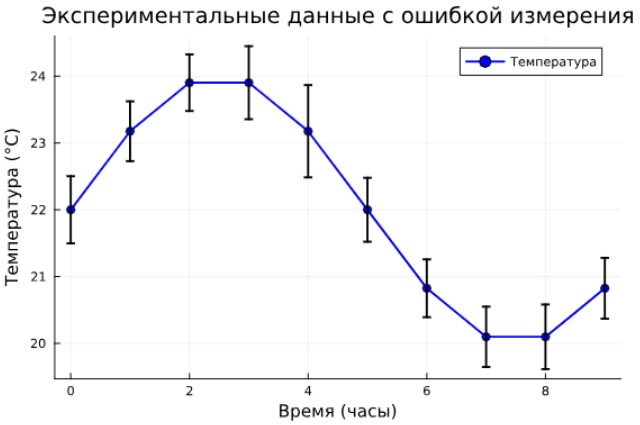


Рис. 52: Решение задания №6

Выполнение задания №7 (рис. 53):

[58]:
```
# Задание 7
using Random
Random.seed!(42)
data = randn(50)

scatter(1:50, data, label="Случайные данные", color=:green, markersize=4)
title!("Точечный график случайных данных")
xlabel!("Номер точки")
ylabel!("Значение")
```

[58]:

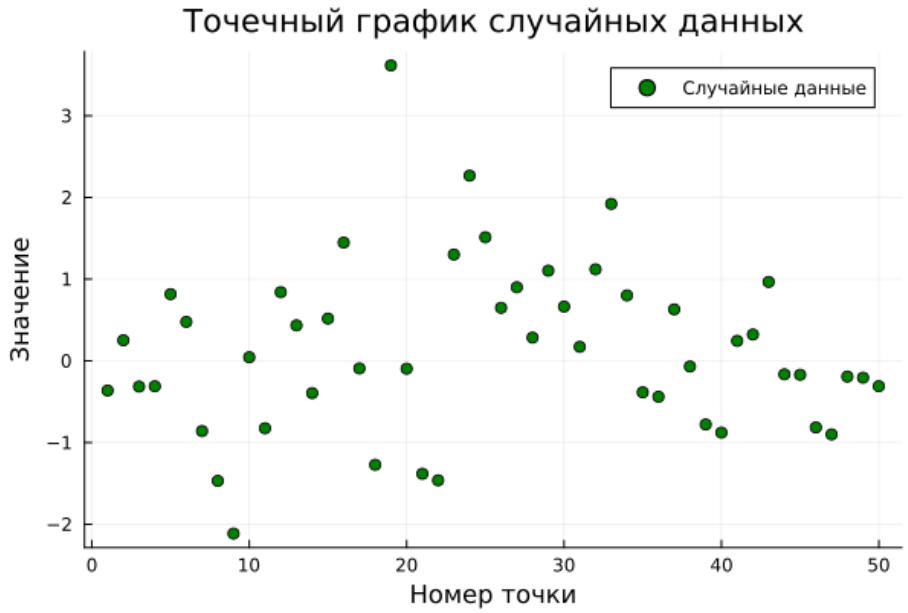


Рис. 53: Решение задания №7

Выполнение задания №8 (рис. 54):

9]:
```
# Задание 8
x3 = randn(100)
y3 = randn(100)
z3 = randn(100)

scatter(x3, y3, z3, label="3D точки", color=:purple, markersize=3)
title!("3D точечный график")
xlabel!("x")
ylabel!("y")
zlabel!("z")
```

9]:

Рис. 54: Решение задания №8

Выполнение задания №9 (рис. 55):

```
60]: # Задание 9
x = range(0, 2π, length=200)
k_values = [1, 2, 3, 5, 8]

plot(x, sin.(x), label="k=1", color=:black, linewidth=2)
for k in k_values[2:end]
    plot!(x, sin.(k*x), label="k=$k", linewidth=1.5)
end
title!("Последовательность синусоид T_k = sin(kx)")
xlabel!("x")
ylabel!("T_k")
```

60]:

Рис. 55: Решение задания №9

Выполнение задания №10 (рис. 56):

[61]:
```
# Задание 10: k = 2, 3, 4, 5, ...
r = 1.0
θ = range(0, 2π, length=200)

anim = @animate for k in 2:5
    t = range(0, 2π, length=200)
    x = r*(k-1)*cos.(t) + r*cos.((k-1)*t)
    y = r*(k-1)*sin.(t) - r*sin.((k-1)*t)

    plot(x, y, title="Гипоциклоида, k = $k", aspect_ratio=1, legend=false)
end

gif(anim, "hypocycloid_k2to5.gif", fps=1)
```

```
[ Info: Saved animation to C:\Users\Олеся\hypocycloid_k2to5.gif
```

[61]:

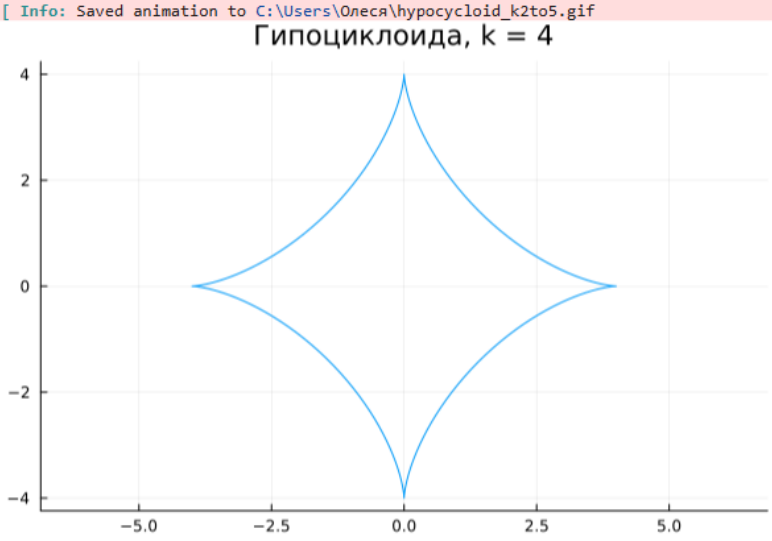


Рис. 56: Решение задания №10

Выполнение задания №11 (рис. 57):

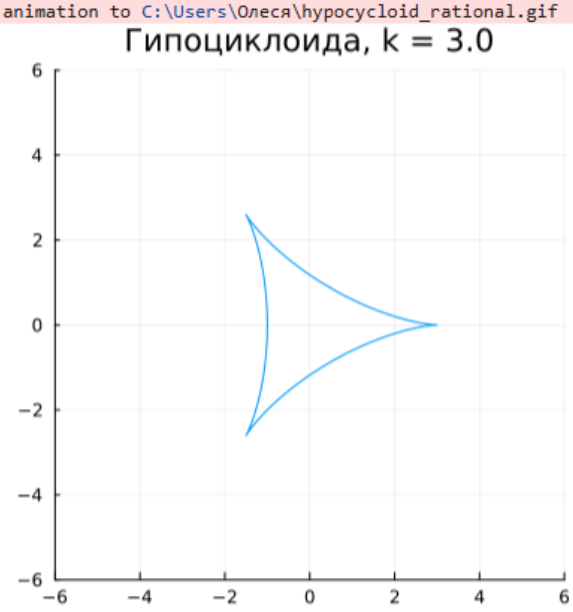
[62]:
```julia
# Задание 11: k = 2.0, 2.5, 3.0, 3.5, ...
anim_rational = @animate for k in [2.0, 2.5, 3.0, 3.5, 4.0]
    t = range(0, 4π, length=400)  # больше точек для дробных k
    x = r*(k-1)*cos.(t) + r*cos.((k-1)*t)
    y = r*(k-1)*sin.(t) - r*sin.((k-1)*t)

    plot(x, y, title="Гипоциклоида, k = $k", aspect_ratio=1, legend=false, xlim=(-6,6), ylim=(-6,6))
end

gif(anim_rational, "hypocycloid_rational.gif", fps=1)
```

```
[ Info: Saved animation to C:\Users\Олеся\hypocycloid_rational.gif
```

[62]:



[ ]:

Рис. 57: Решение задания №11

3 Вывод

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

4 Список литературы. Библиография

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