

# Лабораторная работа №5

Компьютерный практикум по статистическому анализу данных

---

Зиязетдинов Алмаз Радикович

2025

Российский университет дружбы народов имени Патриса Лумумбы, Москва, Россия

## Цель лабораторной работы

---

- Освоить синтаксис языка Julia для построения графиков.

## Выполнение лабораторной работы

---

# Основные пакеты для работы с графиками в Julia

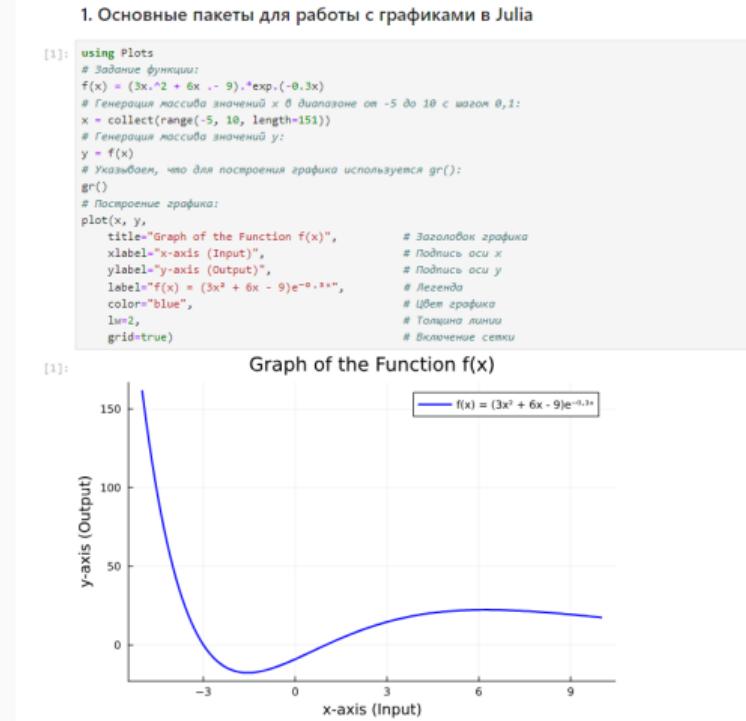


Рис. 1: График функции, построенный при помощи gr()

# Основные пакеты для работы с графиками в Julia

```
[2]: pyplot()
# Построение графика:
plot(x, y,
      title="график функции f(x)",
      xlabel="Ось x (входные данные)",
      ylabel="Ось y (выходные данные)",
      label="f(x) = (3x2 + 6x - 9)e-0.3x",
      color="blue",
      lw=2,
      grid=true)
```

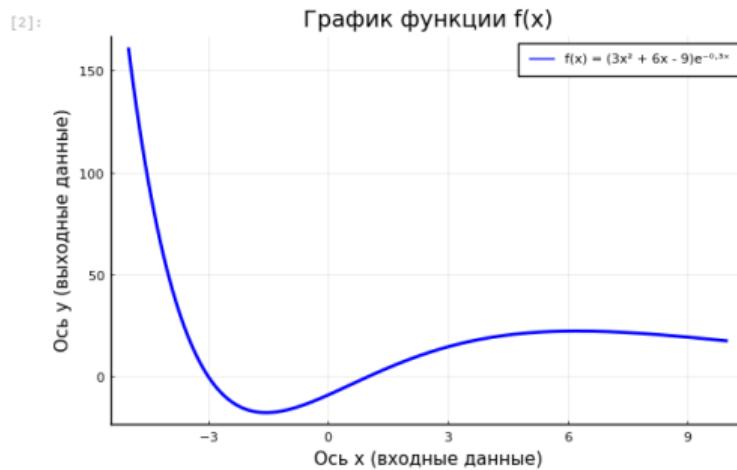


Рис. 2: График функции, построенный при помощи pyplot()

# Опции при построении графика

## 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 y (значение функции)",        # Подпись оси y
      label="sin(x)",                         # Легенда
      lw=2,                                    # Толщина линии
      color="blue",                           # Цвет линии
      grid=True)                             # Включение сетки
```

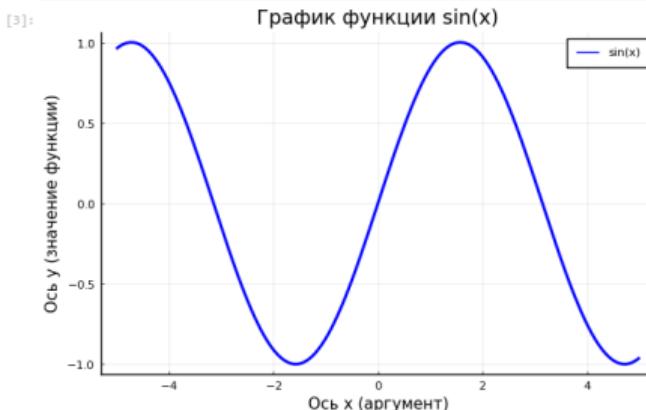


Рис. 3: График функции  $\sin(x)$

# Опции при построении графика

```
[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
```

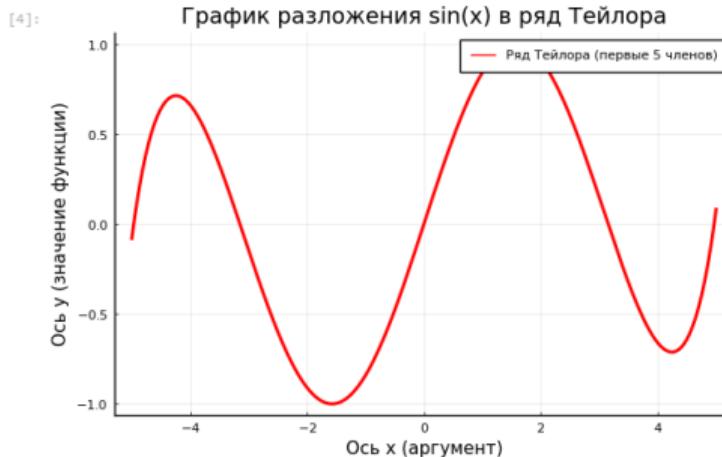


Рис. 4: График функции разложения исходной функции в ряд Тейлора

# Опции при построении графика

```
[5]: # построение двух функций на одном графике:  
# Построение графика теоретической функции sin(x):  
plot(sin_theor,  
     label="sin(x)",  
     lw=2,  
     color="blue")  
# Добавление графика разложения в ряд Тейлора:  
plot(sin_taylor,  
     label="Ряд Тейлора (5 членов)",  
     lw=2,  
     color="red")  
# 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
```

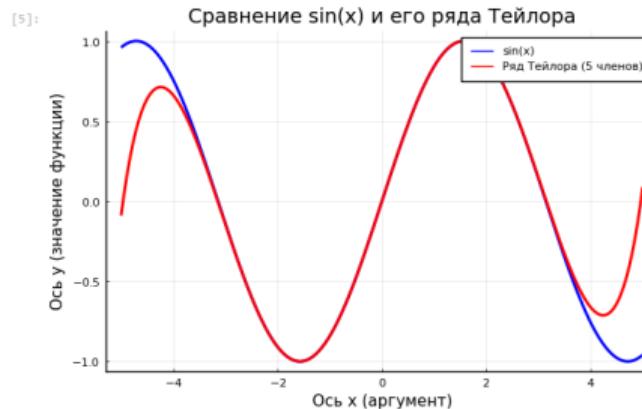


Рис. 5: Графики исходной функции и её разложения в ряд Тейлора

# Опции при построении графика

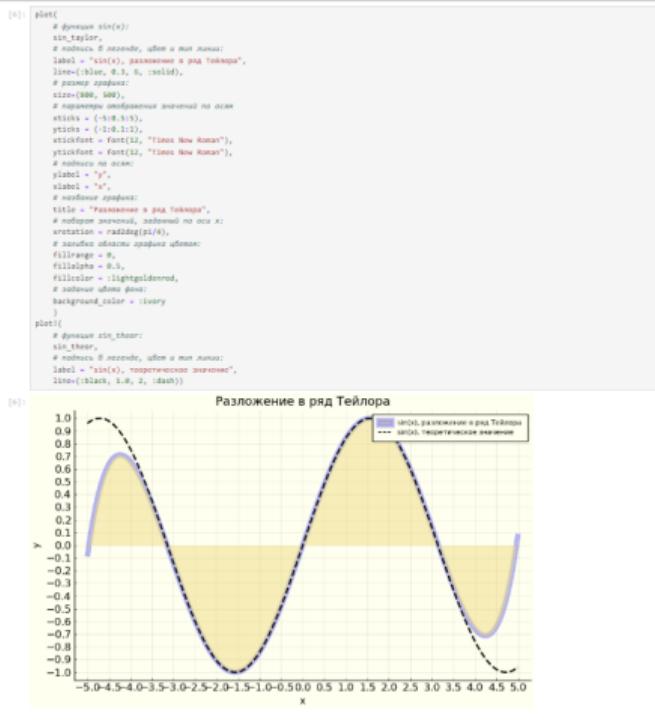


Рис. 6: Вид графиков после добавления опций при их построении

# Точечный график

## 3. Точечный график

### 3.1. Простой точечный график

```
[7]: # Параметры распределения точек на плоскости:  
x = range(1, 10, length=10)  
y = rrand(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
```

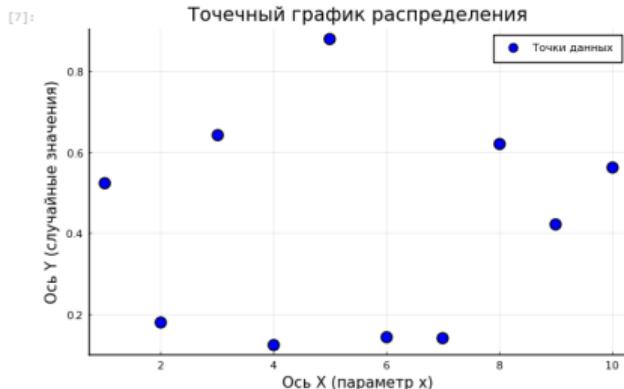


Рис. 7: График десяти случайных значений на плоскости (простой точечный график)

# Точечный график

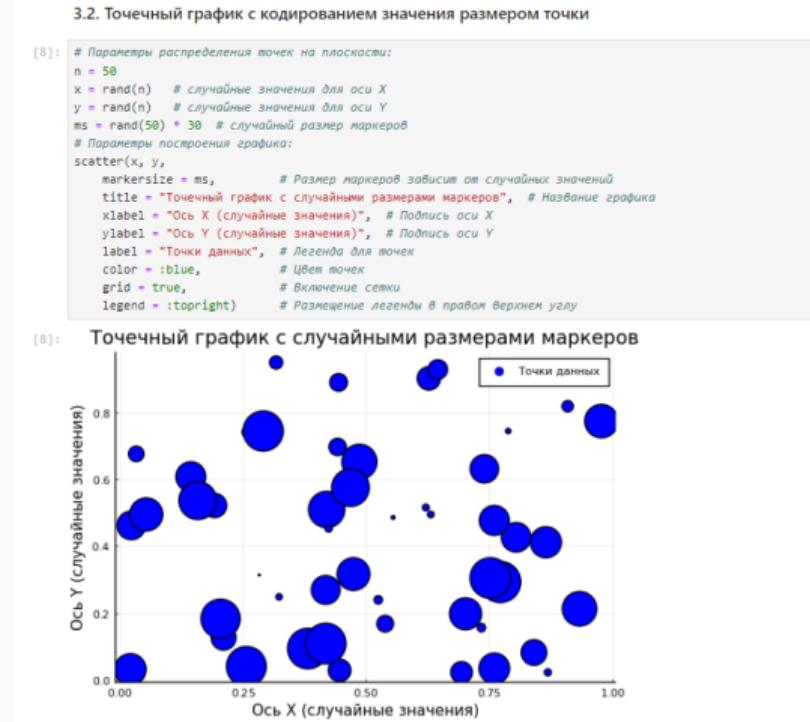


Рис. 8: График пятидесяти случайных значений на плоскости с различными опциями отображения (точечный график с кодированием значения размером точки)

# Точечный график

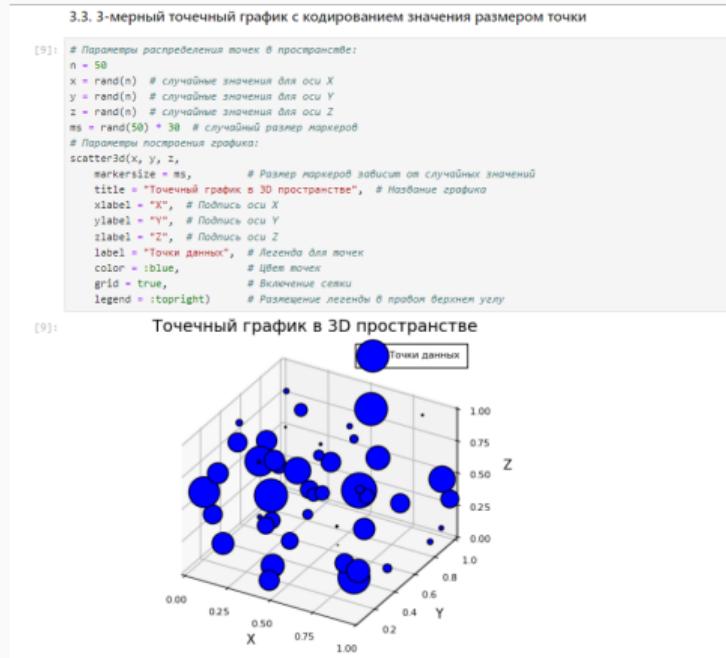


Рис. 9: График пятидесяти случайных значений в пространстве с различными опциями отображения (3-мерный точечный график с кодированием значения размером точки)

# Аппроксимация данных

## 4. Аппроксимация данных

```
[10]: # Массив данных от 0 до 10 с шагом 0.01:  
x = np.array([0.01 * i for i in range(1000)])  
# Экспоненциальная функция со случайным сдвигом значений:  
y = np.exp(np.ones(1000) * x) + 4000 * np.random(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
```

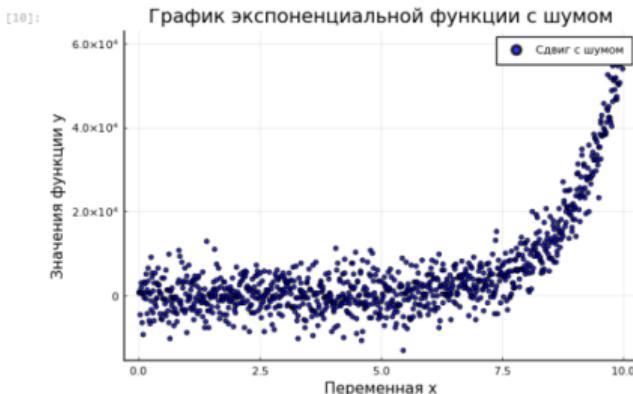


Рис. 10: Пример функции

# Аппроксимация данных

```
[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 (переменными f и 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
```

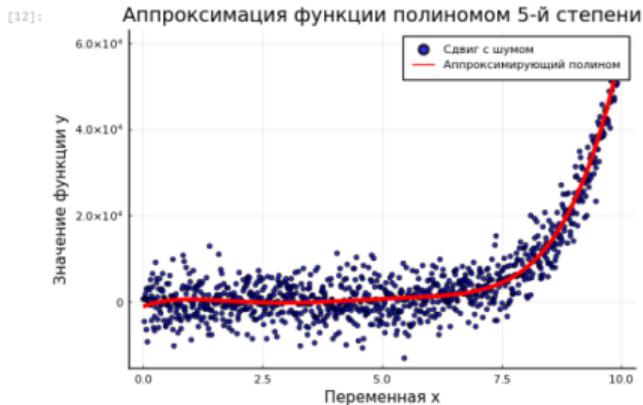


Рис. 11: Пример аппроксимации исходной функции полиномом 5-й степени

# Две оси ординат

## 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
)
```

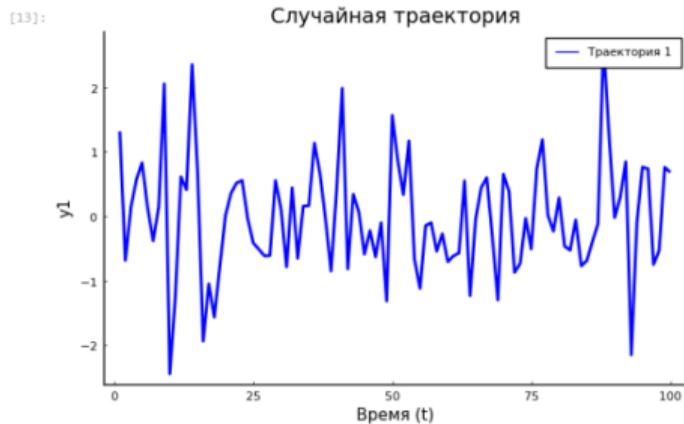


Рис. 12: Пример отдельно построенной траектории

# Полярные координаты

## 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",
      lims=(0, 1.5),
      title="Полярная кривая",
      xlabel="Угол (θ)",
      ylabel="Радиус (r)",
      label="r(θ) = 1 + cos(θ) * sin(θ)^2", # Легенда с называнием функции
      legend="topright", # Положение легенды в правом верхнем углу
      color="blue",
      linewidth=2,
      grid=on,
      box=true
)
```

```
[15]:
```

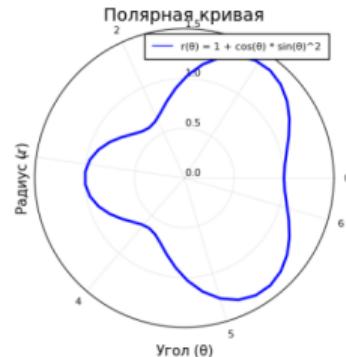


Рис. 13: График функции, заданной в полярных координатах

# Параметрический график

## 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 оси X
      lw=2, # Толщина линии
      color=blue, # Цвет графика
      grid=True, # Включение сетки
      )
```

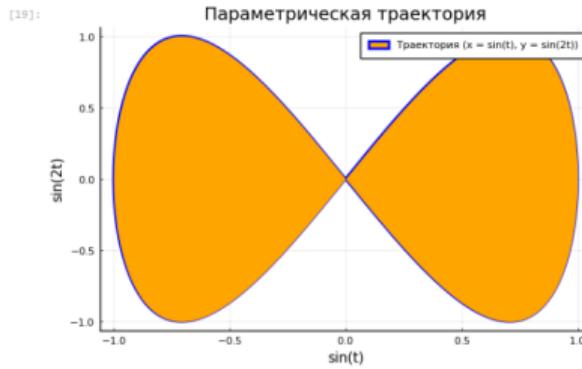


Рис. 14: Параметрический график кривой на плоскости

# Параметрический график

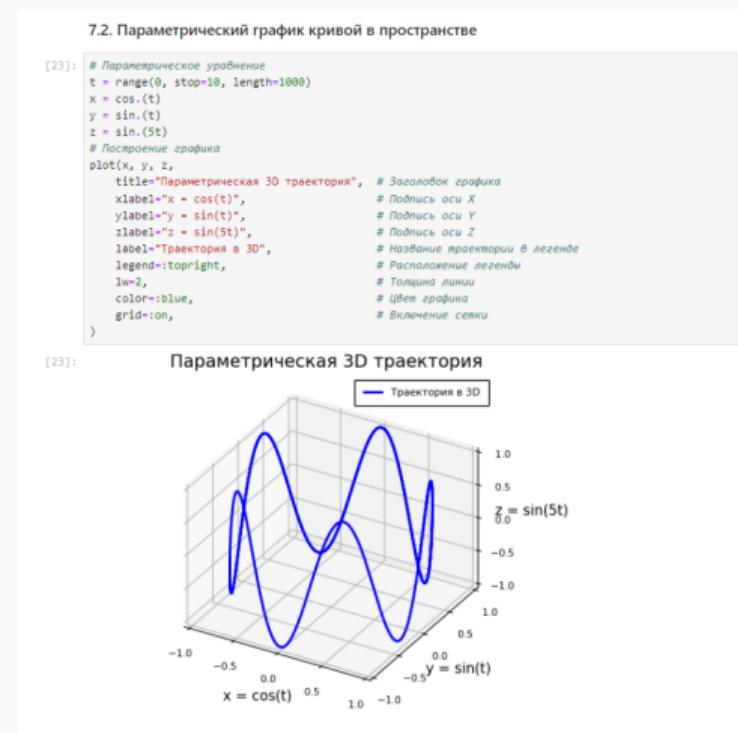


Рис. 15: Параметрический график кривой в пространстве

# График поверхности

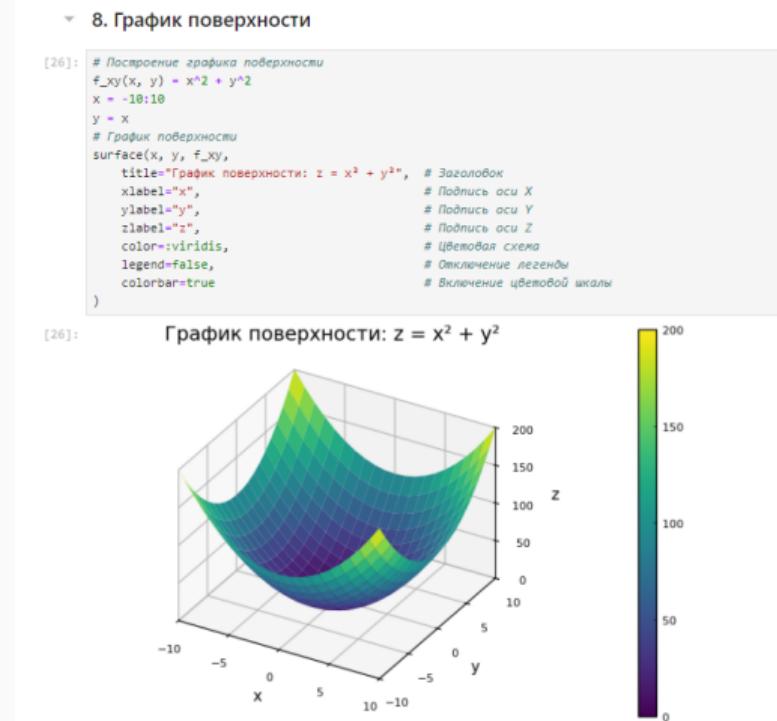


Рис. 16: График поверхности (использована функция surface())

# График поверхности

```
[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,
    line_type=: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:  $z = x^2 + y^2$

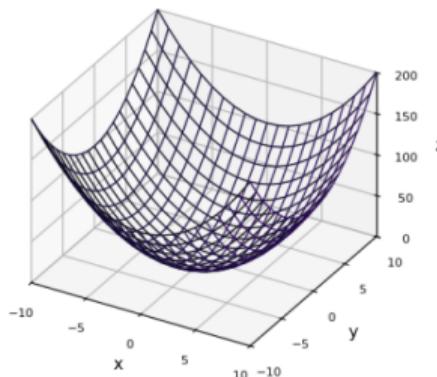


Рис. 17: График поверхности (использована функция `plot()`)

# График поверхности

```
[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:  $z = x^2 + y^2$

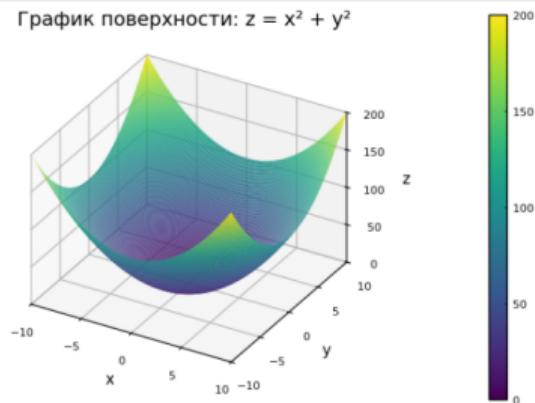


Рис. 18: Сглаженный график поверхности

# График поверхности

```
[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
      cgrad=[:red, :blue]),        # Градиент цвета
      camera=(-30, 30),           # Угол обзора камеры
      legend=false,                # Отключение легенды
      colorbar=true)
```

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

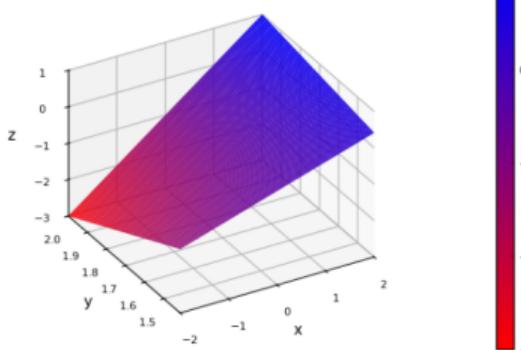


Рис. 19: График поверхности с изменённым углом зрения

# Линии уровня

## 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
      cgrad(:green, :yellow, :red), # Градиент цвета
      colorbar=true               # Включение цветовой шкалы
)
```

Функция:  $g(x, y)$

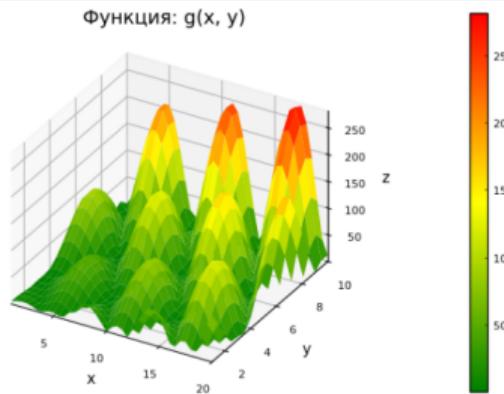


Рис. 20: График поверхности, заданной функцией  $g(x, y) = (3x + y^2)|\sin(x) + \cos(y)|$

## Линии уровня

```
[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
```

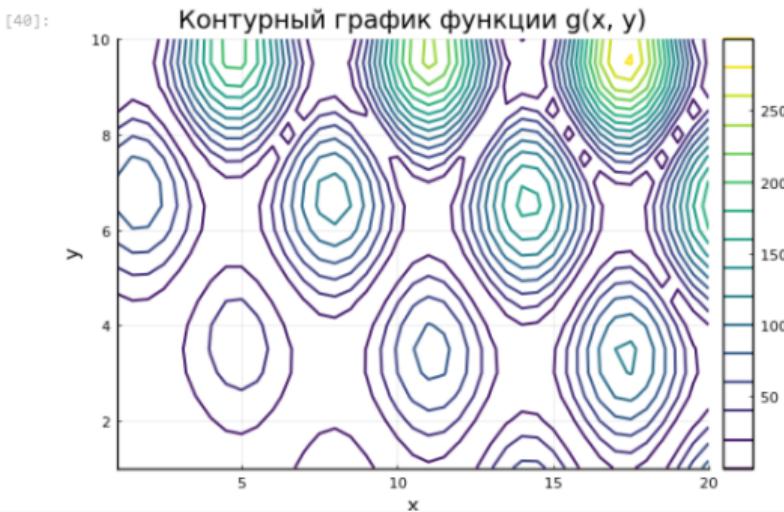


Рис. 21: Линии уровня

## Линии уровня

```
[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
)
```

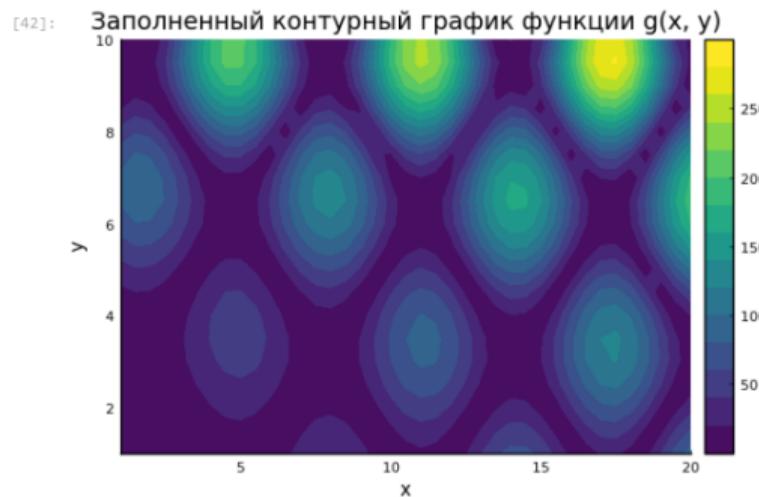


Рис. 22: Линии уровня с заполнением

# Векторные поля

## 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  $h(x, y)$

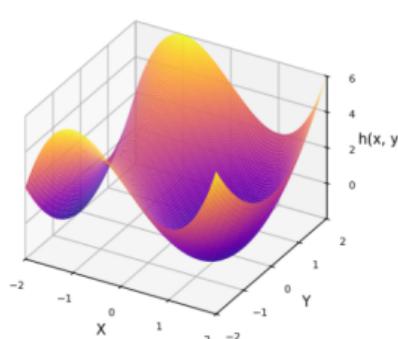


Рис. 23: График функции  $h(x, y) = x^3 - 3x + y^2$

# Векторные поля

```
[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
)
```

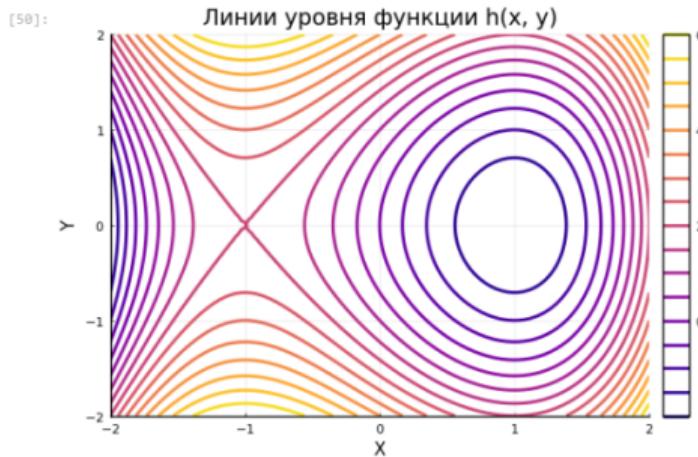


Рис. 24: Линии уровня для функции  $h(x, y) = x^3 - 3x + y^2$

## 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]:
```

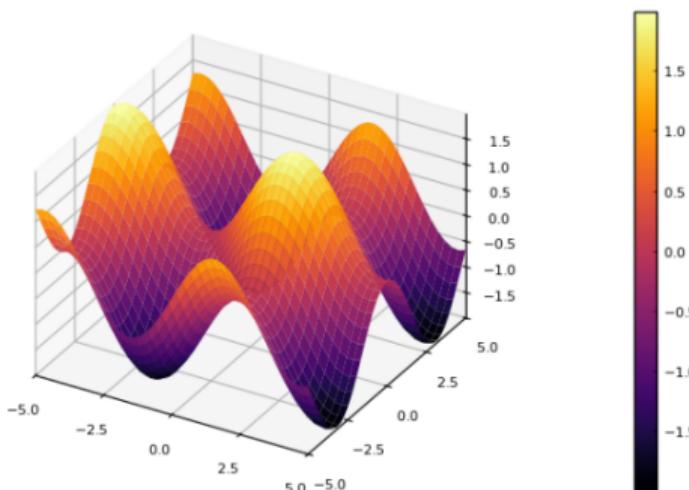


Рис. 25: Статичный график поверхности

# Анимация

```
[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
```

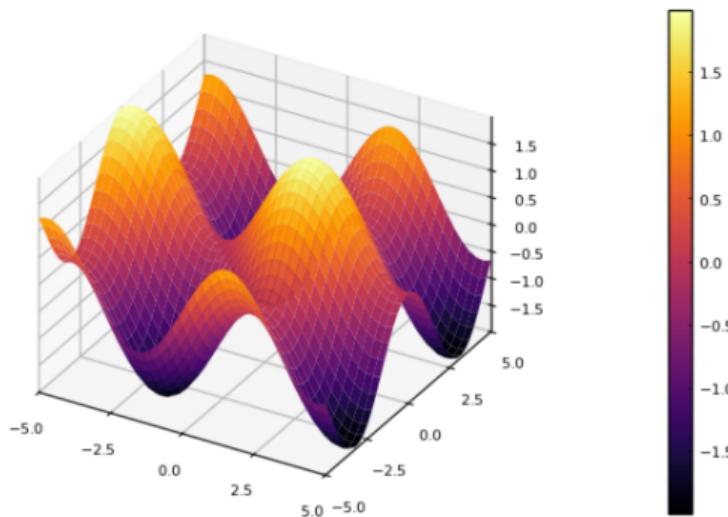


Рис. 26: Анимированный график поверхности

# Гипоциклоида

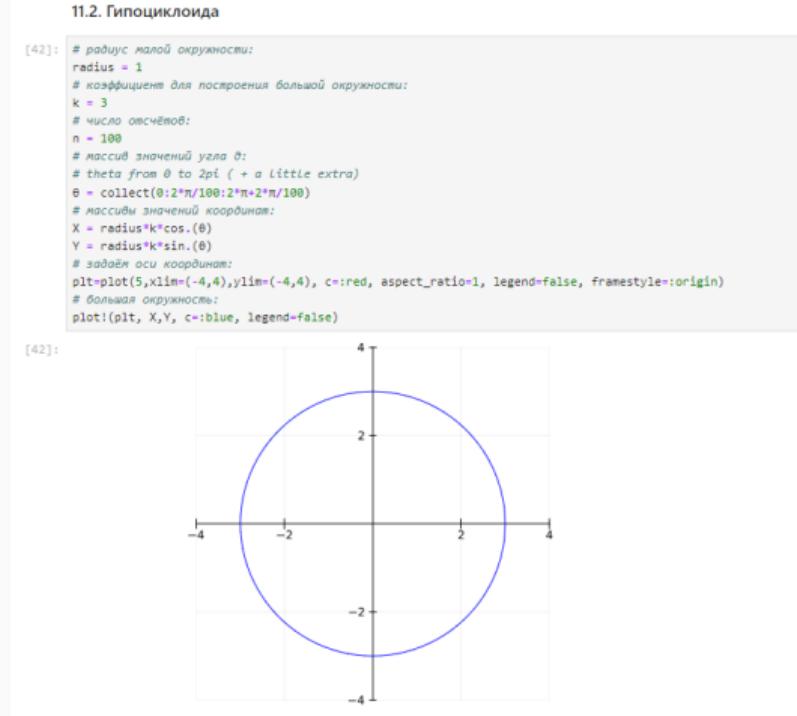


Рис. 27: Большая окружность гипоциклоиды

## Гипоциклоида

```
[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]:
```

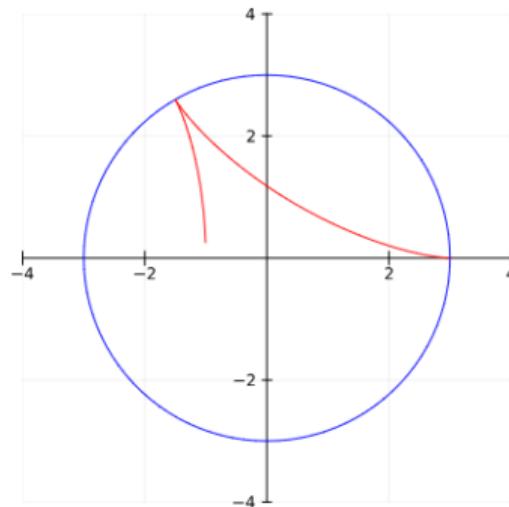


Рис. 28: Половина пути гипоциклоиды

## Гипоциклоида

```
[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]:
```

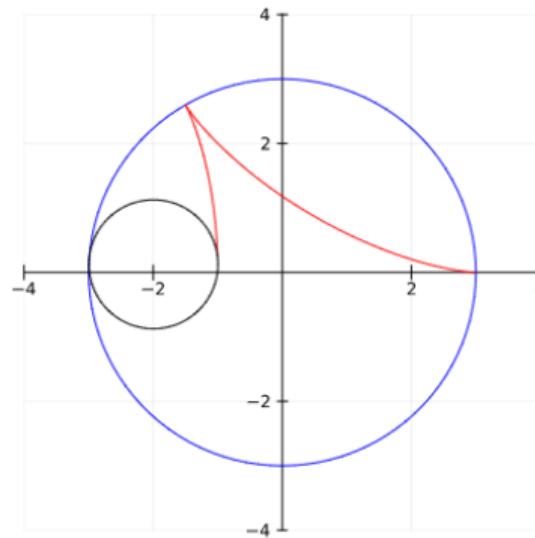


Рис. 29: Малая окружность гипоциклоиды

## Гипоциклоида

```
[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]:
```

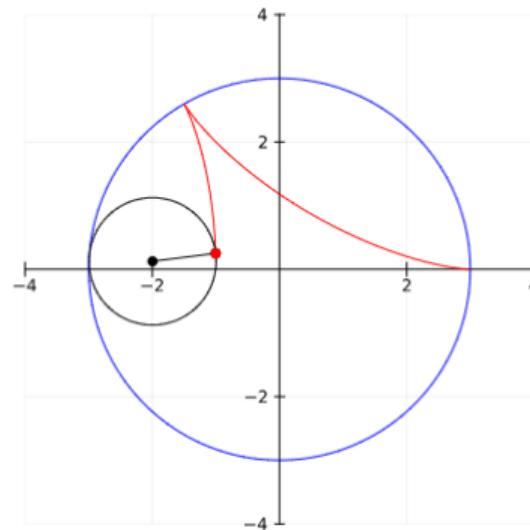


Рис. 30: Малая окружность гипоциклоиды с добавлением радиуса

# Гипоциклоида

```
[50]: anim = @animate for i in 1:n
    # добавим оси координат:
    plt=plot(5,xlim=(-4,4),ylim=(-4,4), c=:red, aspect_ratio=1,legend=false, framestyle=:origin)
    # дальняя окружность:
    plot!(plt, X,Y, c=:blue, legend=false)
    t = @t[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:
    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)
end
gif(anim,"hypocycloid.gif")
```

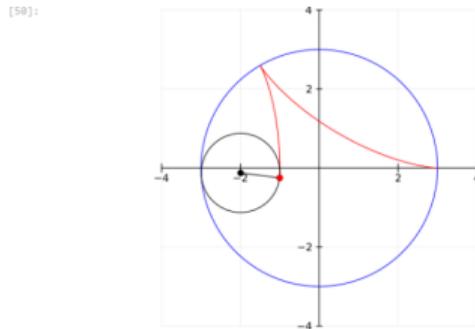


Рис. 31: Малая окружность гипоциклоиды с добавлением радиуса

# Errorbars

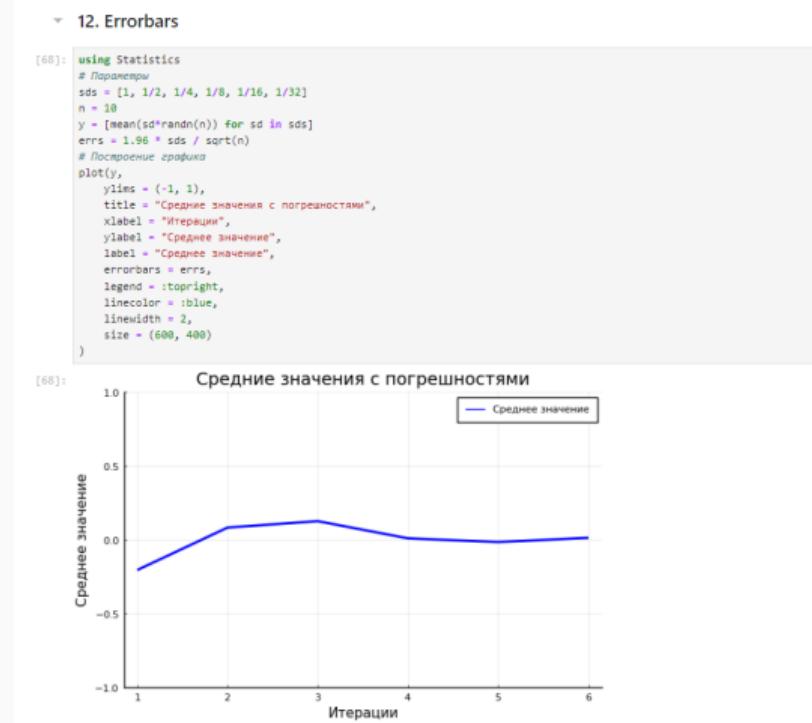


Рис. 32: График исходных значений

## Errorbars

```
[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
)
```

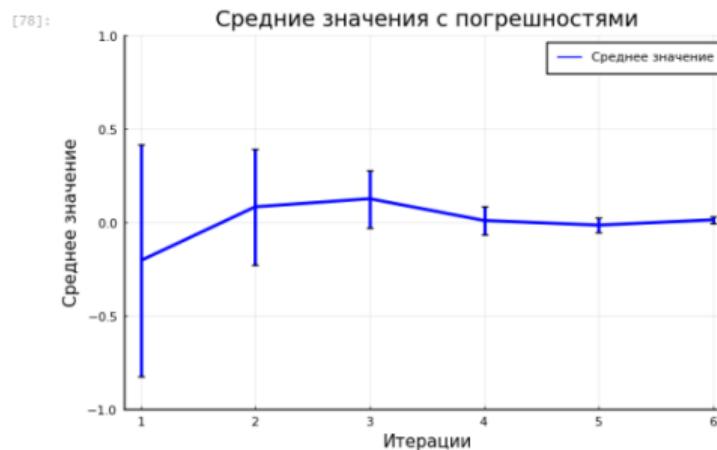


Рис. 33: График исходных значений с отклонениями

## Errorbars

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

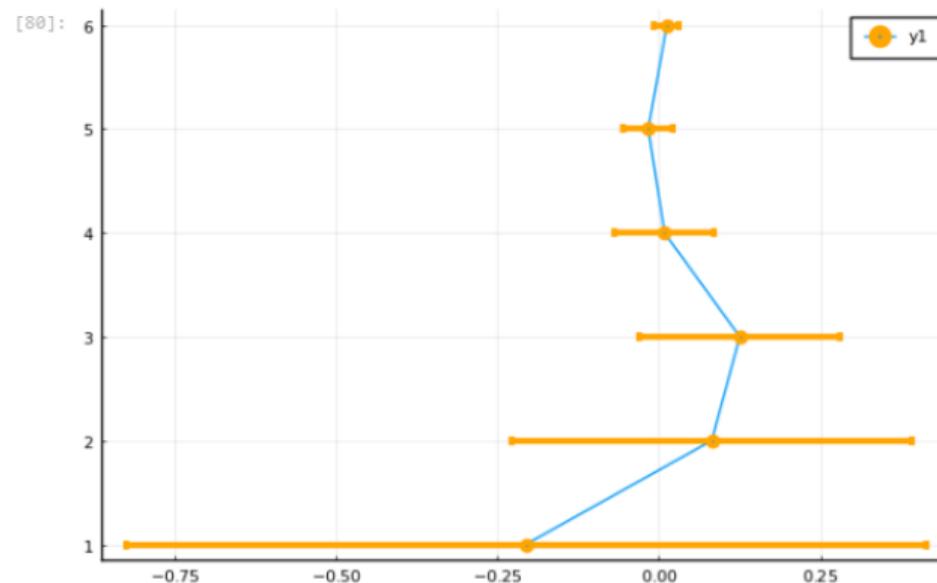


Рис. 34: Поворот графика

## Errorbars

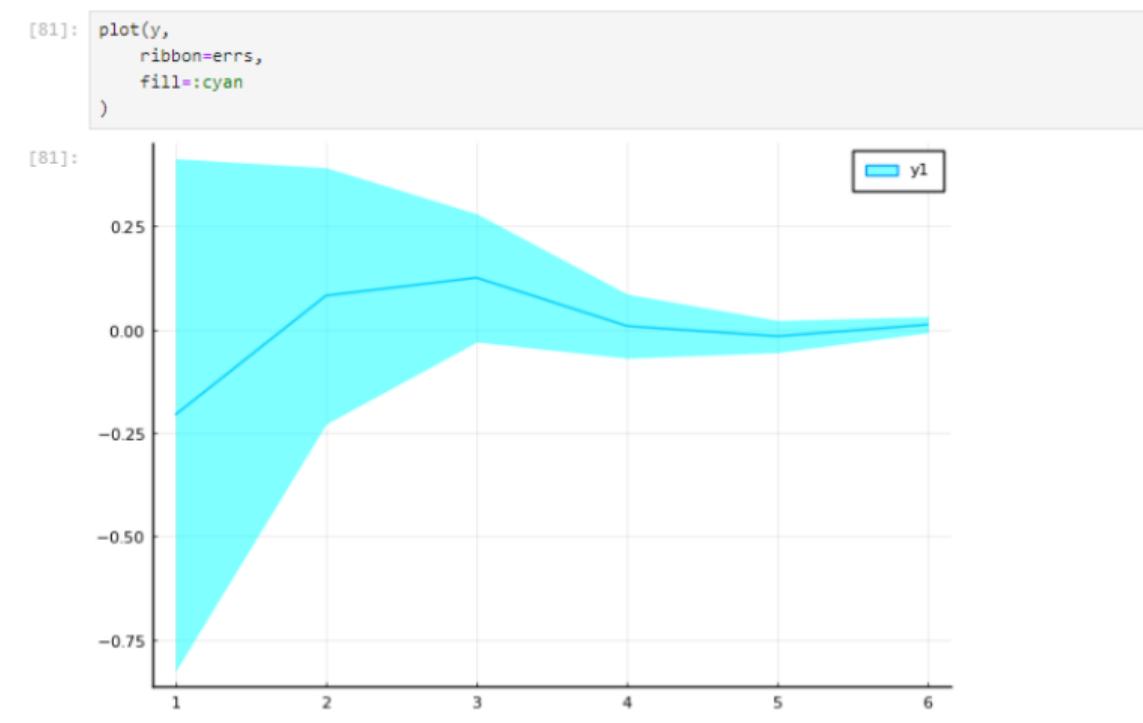


Рис. 35: Заполнение цветом

# Errorbars

```
[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) = 1.96 * std(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)
    )
```

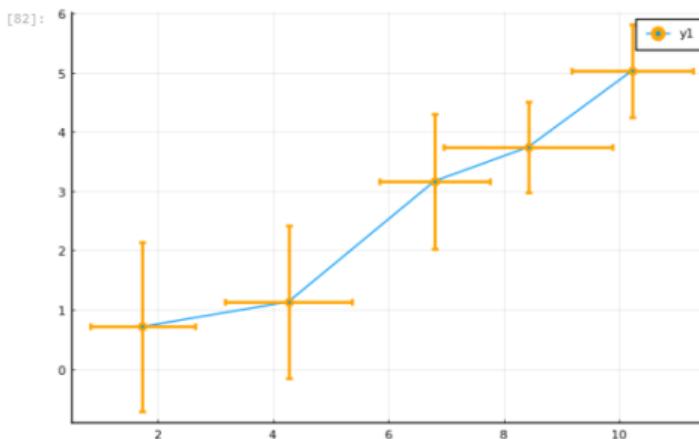


Рис. 36: График ошибок по двум осям

## Errorbars

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

```
[83]:
```

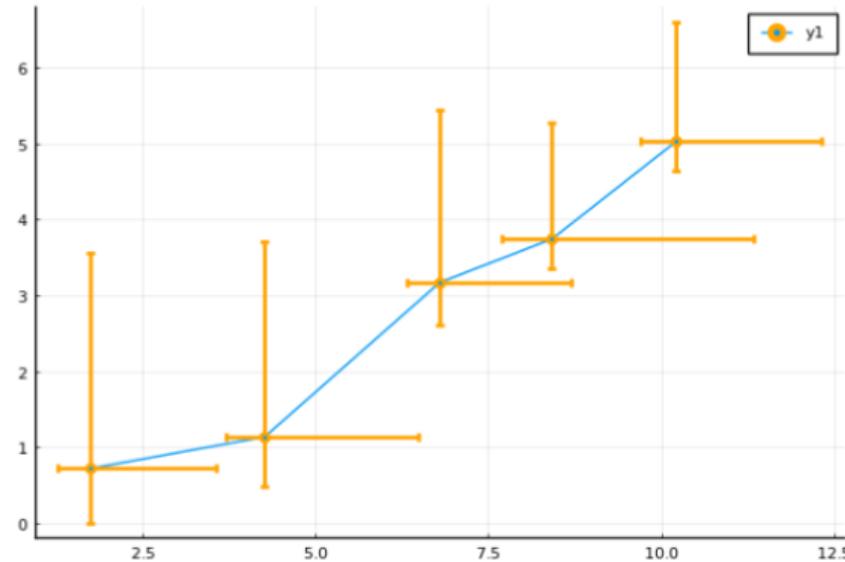


Рис. 37: График асимметричных ошибок по двум осям

# Использование пакета Distributions

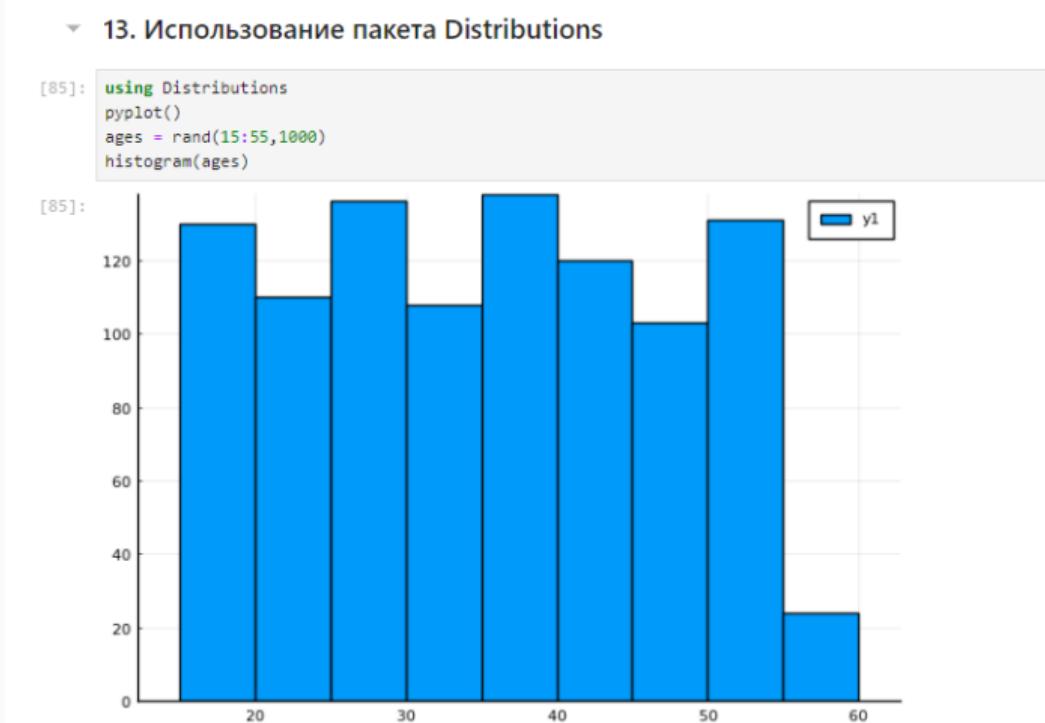


Рис. 38: Гистограмма, построенная по массиву случайных чисел

## Использование пакета Distributions

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

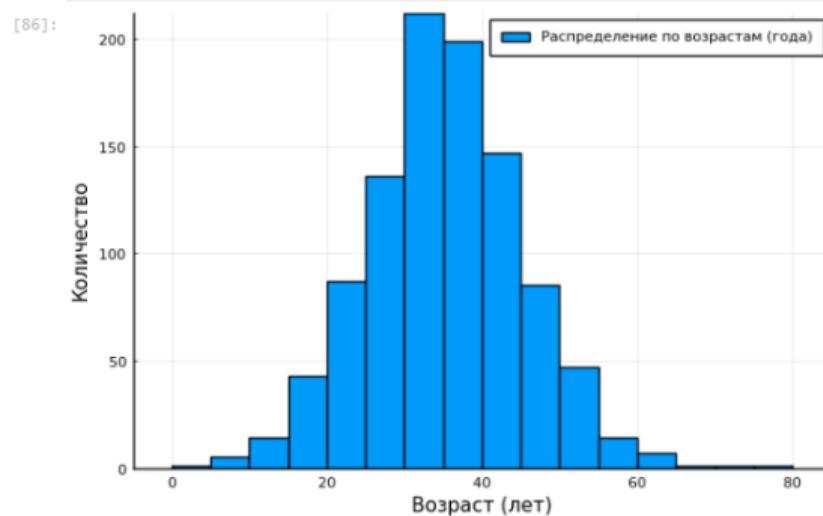


Рис. 39: Гистограмма нормального распределения

# Использование пакета Distributions

```
[89]:  
    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(floor(N/2))));  
    ages = append!(ages,rand(d2,N));  
    ages = append!(ages,rand(d3,Int64(floor(N/3))));  
    histogram(  
        ages,  
        bins=50,  
        label="Распределение по возрастам (года)",  
        xlabel = "Возраст (лет)",  
        ylabel= "Количество",  
        title = "Распределение по возрастам (года)"  
    )
```

```
[89]:
```

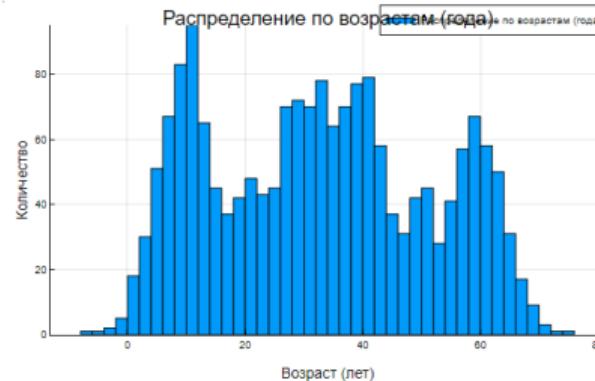


Рис. 40: Гистограмма распределения людей по возрастам

## 14. Подграфики

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

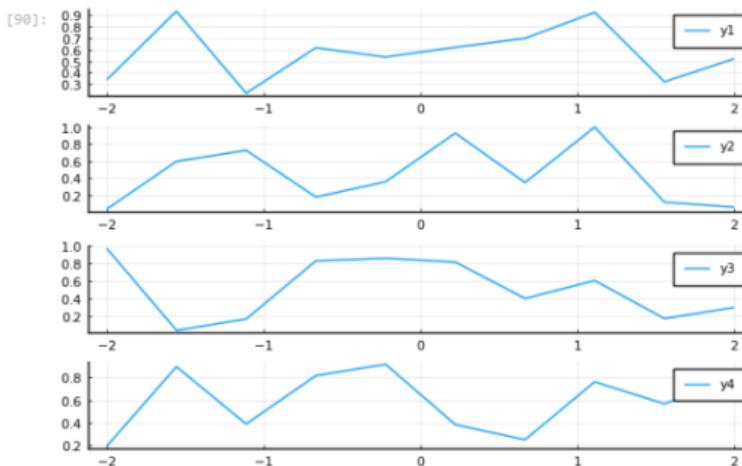


Рис. 41: Серия из 4-х графиков в ряд

## Подграфики

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

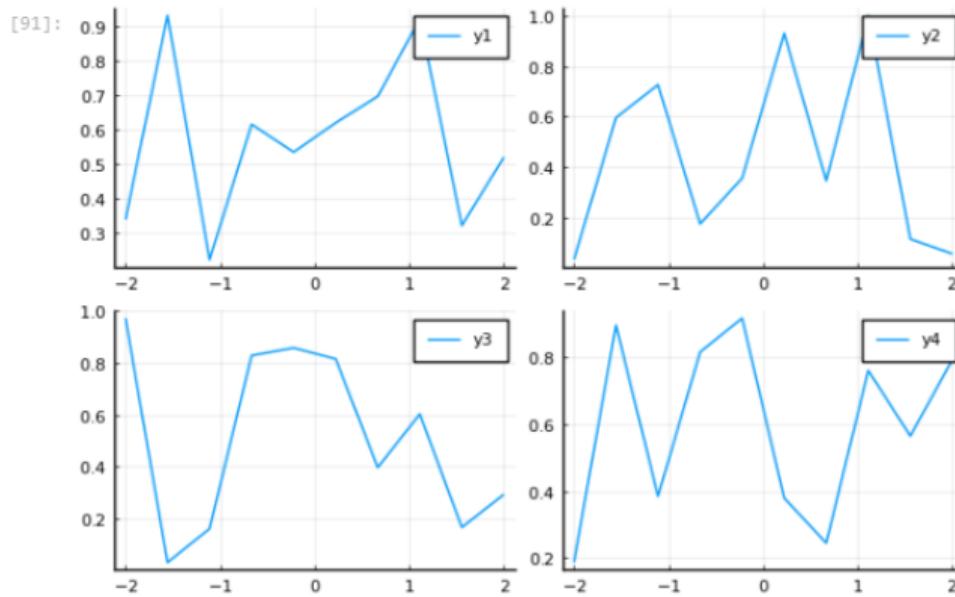


Рис. 42: Серия из 4-х графиков в сетке

# Подграфики

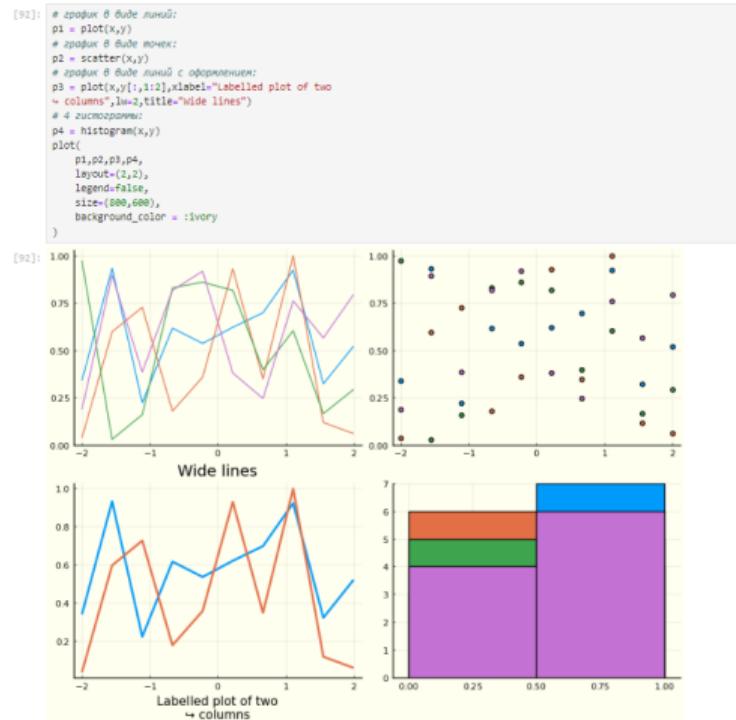


Рис. 43: Объединение нескольких графиков в одной сетке

## Подграфики

```
[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
)
```

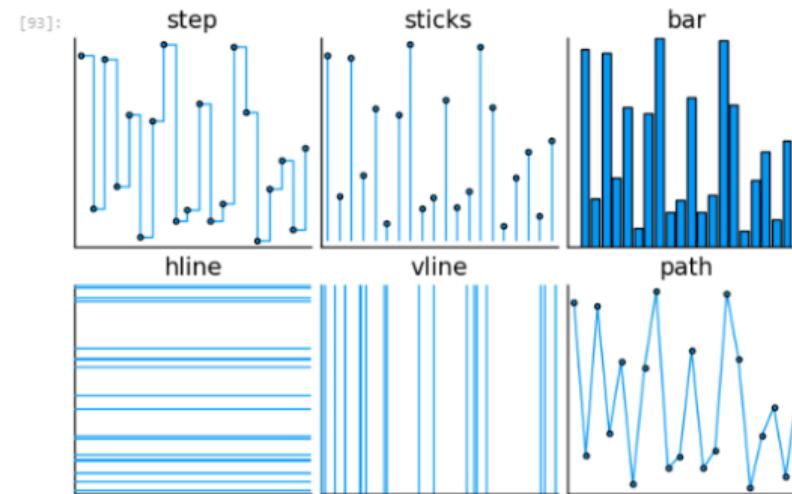


Рис. 44: Разнообразные варианты представления данных

# Подграфики

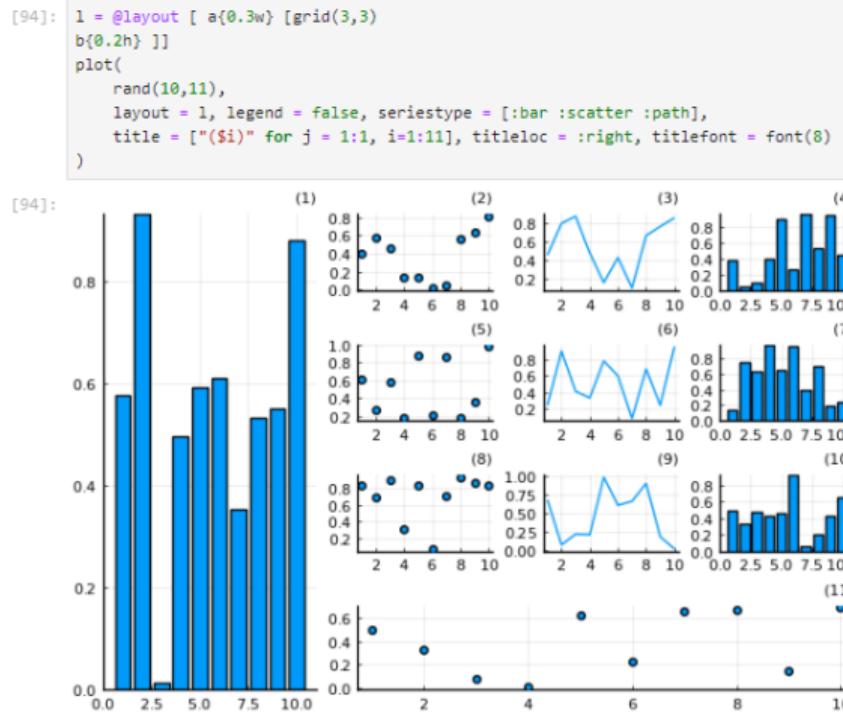


Рис. 45: Демонстрация применения сложного макета для построения графиков

# Самостоятельная работа

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)
```

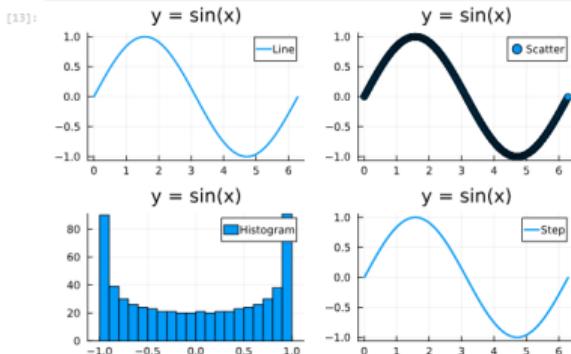


Рис. 46: Решение задания №1

# Самостоятельная работа

- 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 = plt(x, y, linestyle=:solid, label="$solid", xlabel="x", ylabel="y", title="Графики функции y = sin(x) с разными стилями линий", lin=2)
for ls in line_styles[2:end]
    plot!(plt, x, y, linestyle=ls, label="$ls", lin=2)
end
display(plt)
```

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

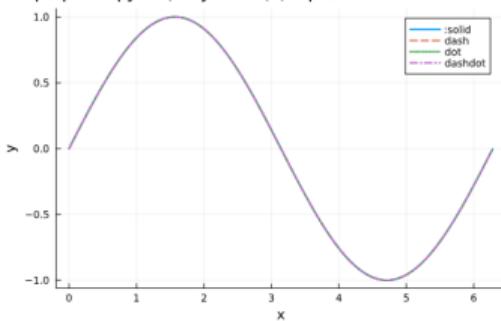


Рис. 47: Решение задания №2

# Самостоятельная работа

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: 1

```
[5]: # Определяем функция 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,           # цвет графика
      label = "y(x) = \pi x^2 \ln(x)", # легенда
      xlabel = "x",            # подпись оси X
      ylabel = "y(x)",         # подпись оси Y
      framestyle = :box,       # стиль рамки
      grid = false,            # убираем сетку
      xticks = 0:2:10,          # частота отмеч на оси X
      yticks = -50:50:200,      # частота отмеч на оси Y
      bordercolor = :green)    # цвет рамки
```

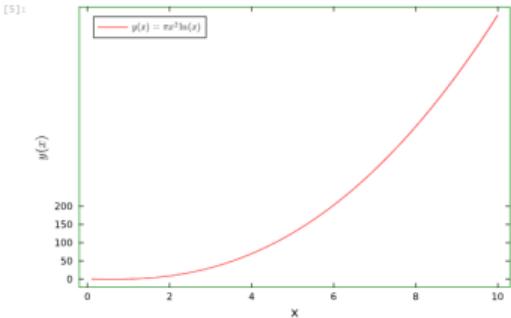


Рис. 48: Решение задания №3

# Самостоятельная работа

- ▼ 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: [1](#)

```
[10]: # Задаем вектор x
x = [-2, -1, 0, 1, 2]
# Функция y(x)
y(x) = x.^3 .- 3 .* x
# Вычисляем значения y
y_values = y.(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
```

[1]

Рис. 49: Решение задания №4

# Самостоятельная работа

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 заголовков
xlabel!("x")
ylabel!("y")
title!("Графики функций y1(x) и y2(x)")
```

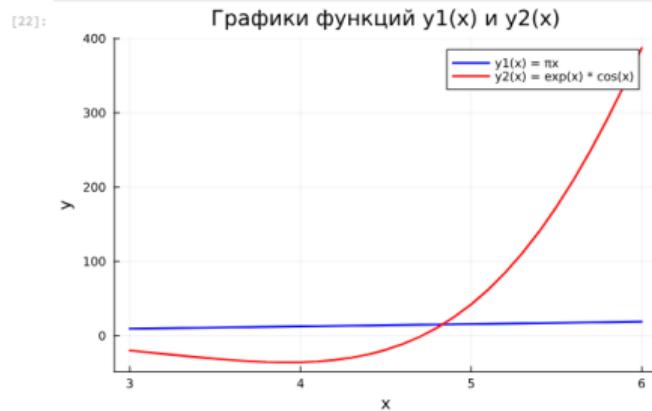


Рис. 50: Решение задания №5

# Самостоятельная работа

```
[21]: # Построение графиков с двумя осями ординат
p = plot(x, y1(x), label="y1(x) = px", 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 семка
xlabel!("x")
ylabel!("y1 (primary)", fontsize=10)
ylabel!(p, "y2 (secondary)", fontsize=10)
title!("Графики с двумя осями ординат")
```

Графики с двумя осями ординат

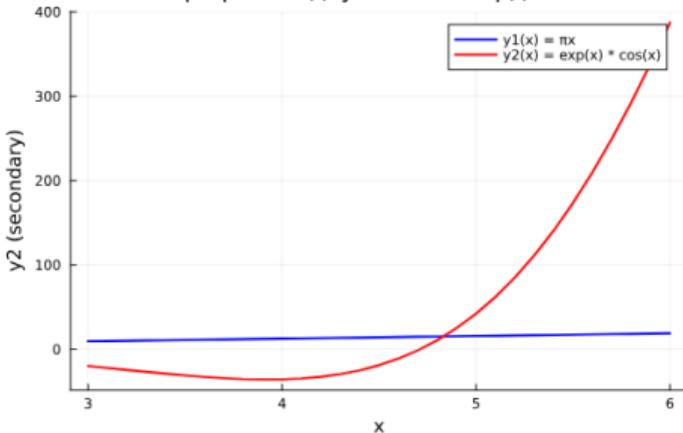


Рис. 51: Решение задания №5

# Самостоятельная работа

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 колеблется от 20 до 25 градусов)
temperature = 22 + 2 * sin(time * pi / 5) # Синусоидальное изменение температуры
# Шаг 2: Добавим ошибку измерения (например, стандартное отклонение 0.5 градуса)
# Ошибки 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")
```

Экспериментальные данные с ошибкой измерения

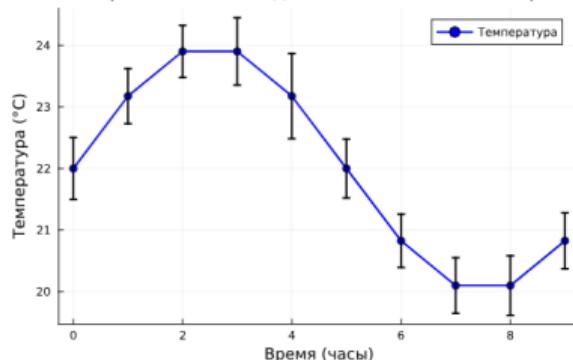


Рис. 52: Решение задания №6

# Самостоятельная работа

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

```
[31]: # 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")
```

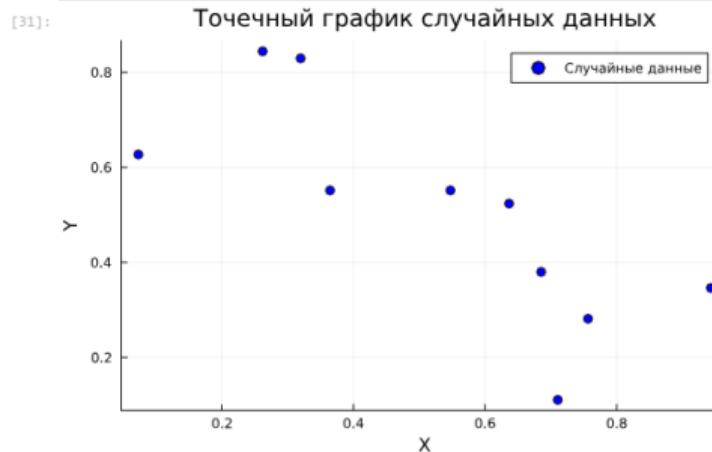


Рис. 53: Решение задания №7

# Самостоятельная работа

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

```
[34]: # 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")
```

[34]: 3-мерный точечный график случайных данных

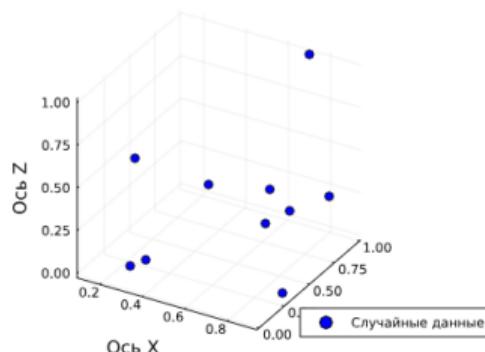


Рис. 54: Решение задания №8

# Самостоятельная работа

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

```
[37]: # Подготавливаем 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="Синусоида", 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)
```

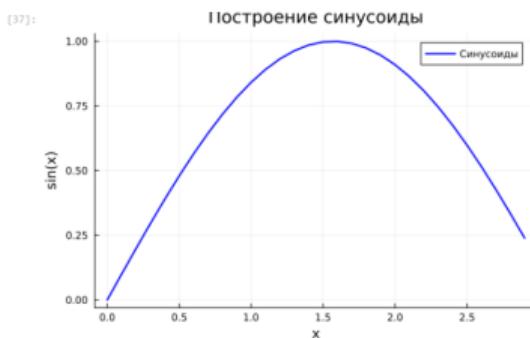


Рис. 55: Решение задания №9

# Самостоятельная работа

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

```
[49]: # функция для вычисления координат гипоциклоиды
function hypocycloid(R, r, θ)
    x = (R - r) * cos.(θ) + r * cos.((R - r) / r * θ)
    y = (R - r) * sin.(θ) - r * sin.((R - r) / r * θ)
    return x, y
end

# функция для создания анимации гипоциклоиды для целого значения k
function create_hypocycloid_animation(k)
    R = 10 # радиус большей окружности
    r = R / k # радиус меньшей окружности
    θ = 0:0.0512 * π # шаг от 0 до 2π, шаг 0.05
    x, y = hypocycloid(R, r, θ)
    anim = @animate for i in 1:length(θ) # итерация по всем углам
        plot(x[i:i], y[i:i], xlabel="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_5k.gif", fps=10)
end

# Создание анимации для целого значения k = 2
create_hypocycloid_animation(2)
```

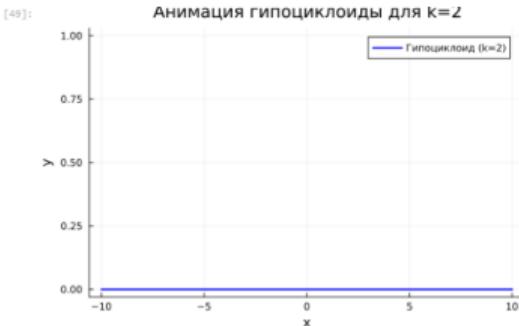


Рис. 56: Решение задания №10

# Самостоятельная работа

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

```
[51]: # функция для вычисления координат эпициклоиды
function epicycloid(R, r, θ)
    x = (R + r) * cos.(θ) - r * cos.((R + r) / r * θ)
    y = (R + r) * sin.(θ) - r * sin.((R + r) / r * θ)
    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
    θ = 0:0.0512 * π # угол от 0 до 2π, шаг 0.05
    x, y = epicycloid(R, r, θ)
    anim = @animate for i in 1:length(θ) # 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_5k.gif", fps=10)
end

# Создание анимации для целого значения k = 2
create_epicycloid_animation(2)
```

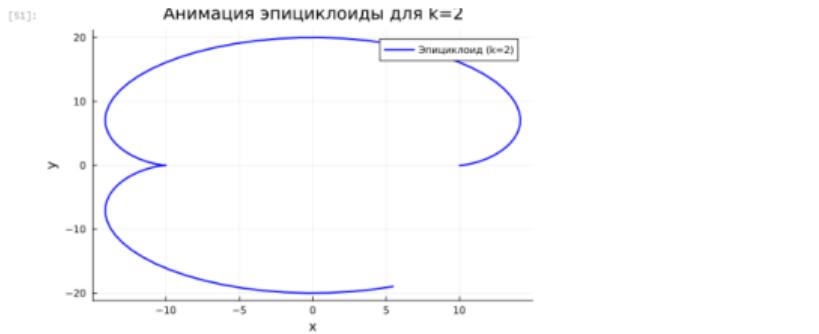


Рис. 57: Решение задания №11

## Вывод

---

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

## Список литературы. Библиография

---

## Список литературы. Библиография

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