Лабораторная работа No 5. (Построение графиков)

Адабор Кристофер Твум

1032225824

НКНбд-01-22

Основные пакеты для работы с графиками в

Адабор Кристофер Твум

1032225824

НКНбд-01-22

```
using Plots

# задание функции:
f(x) = (3x.^2 .+ 6x .- 9).*exp.(-0.3x)

# генерирование массива значений x в диапазоне от -5 до 10 с шагом 0,1
x = range(-5, 10, length=151)

# генерирование массива значений y:
y = f.(x)

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

# задание опций при построении графика
plot(x, y,
    title="A simple curve",
    xlabel="Variable x",
    ylabel="Variable y",
    color="blue",
    linewidth=2)
```

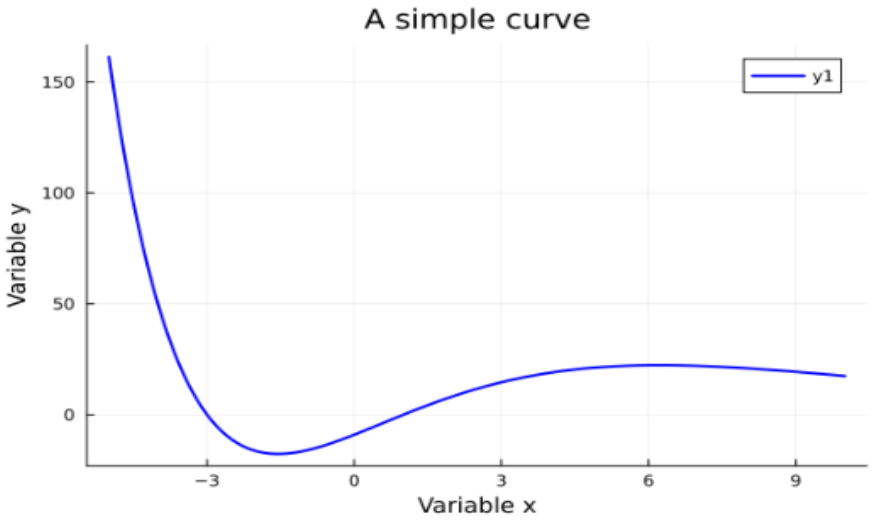


Рис. 5.2. График функции $f(x) = (3x2 + 6x - 9)e-0,3x$, построенный при помощи pyplot()

```
using Plots

# задание функции:
f(x) = (3x.^2 .+ 6x .- 9).*exp.(-0.3x)

# генерирование массива значений x в диапазоне от -5 до 10 с шагом 0,1
x = range(-5, 10, length=151)

# генерирование массива значений y:
y = f.(x)

# используем gr() вместо pyplot()
gr()

# задание опций при построении графика
plot(x, y,
    title="Простая кривая",
    xlabel="Переменная x",
    ylabel="Переменная y",
    color="blue",
    linewidth=2)
```

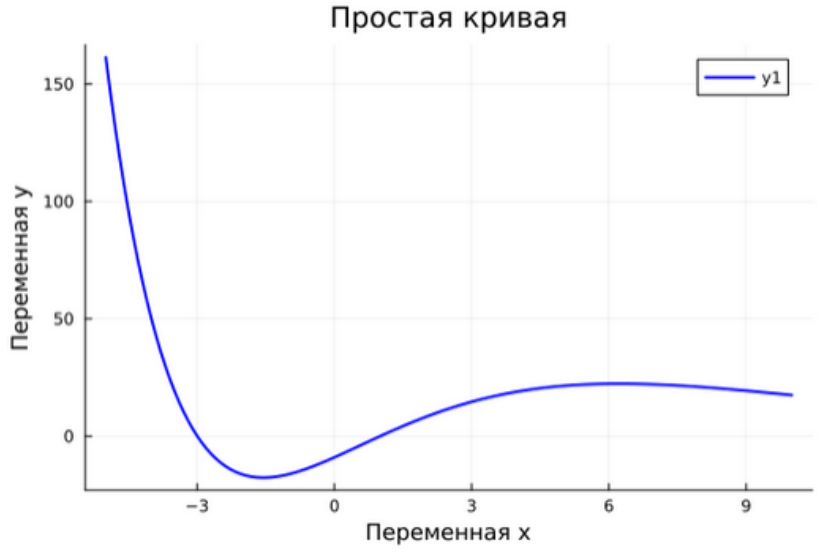


Рис. 5.3. График функции sin(*x*)

```
using Plots

# Используем gr() вместо pyplot() так как он работает стабильнее
gr()

# задание функции sin(x):
sin_theor(x) = sin(x)

# построение графика функции sin(x):
plot(sin_theor, -2π, 2π,
    title="График функции sin(x)",
    xlabel="x",
    ylabel="sin(x)",
    color="blue",
    linewidth=2,
    label="sin(x)",
    legend=:topright,
    grid=true)
```

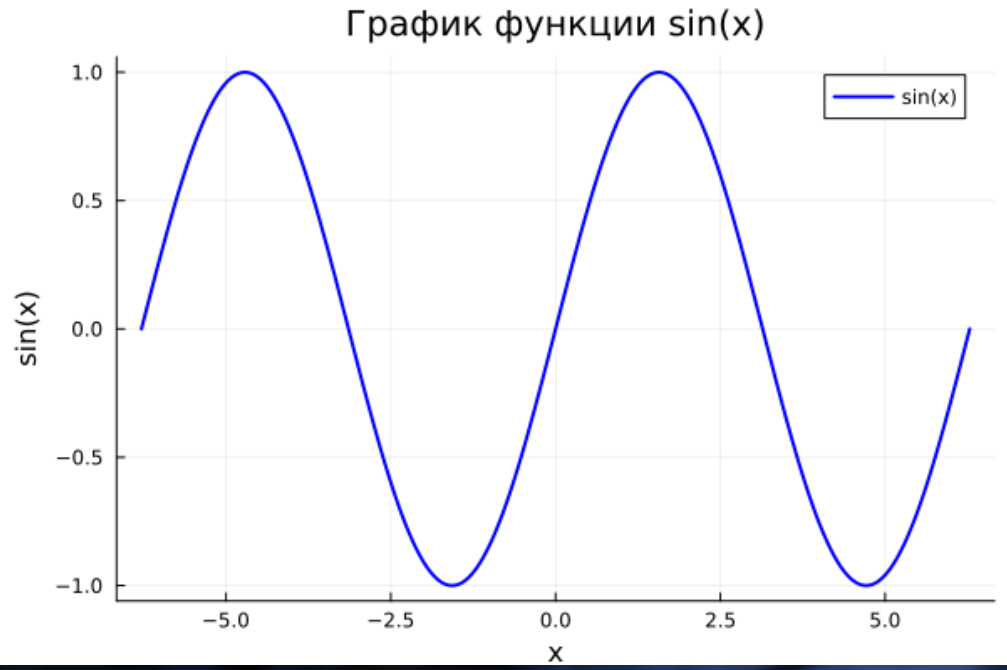


Рис. 5.4. График функции разложения исходной функции в ряд Тейлора

```
using Plots

# Используем gr() вместо pyplot()
gr()

# задание функции разложения sin(x) в ряд Тейлора (5 членов):
function sin_taylor(x)
    sum = 0.0
    for i in 0:4
        term = (-1)^i * x^(2*i+1) / factorial(2*i+1)
        sum += term
    end
    return sum
end

# построение графика функции разложения в ряд Тейлора:
plot(sin_taylor, -2π, 2π,
    title="Разложение sin(x) в ряд Тейлора (5 членов)",
    xlabel="x",
    ylabel="sin_taylor(x)",
    color="red",
    linewidth=2,
    label="sin(x) ряд Тейлора",
    legend=:topright,
    grid=true)
```

Рис. 5.5. Графики исходной функции и её разложения в ряд Тейлора ¶

```
using Plots

# Используем gr() вместо pyplot()
gr()

# задание точной функции sin(x):
sin_theor(x) = sin(x)

# задание функции разложения sin(x) в ряд Тейлора:
function sin_taylor(x)
    sum = 0.0
    for i in 0:4
        term = (-1)^i * x^(2*i+1) / factorial(2*i+1)
        sum += term
    end
    return sum
end

# построение двух функций на одном графике:
plot(sin_theor, -π, π,
    label = "sin(x), теоретическое значение",
    linewidth=2,
    color=:blue,
    title="Сравнение sin(x) и ряда Тейлора",
    xlabel="x",
    ylabel="y",
    legend=:bottomright)

plot!(sin_taylor, -π, π,
    label = "sin(x), разложение в ряд Тейлора",
    linewidth=2,
    linestyle=:dash,
    color=:red)
```

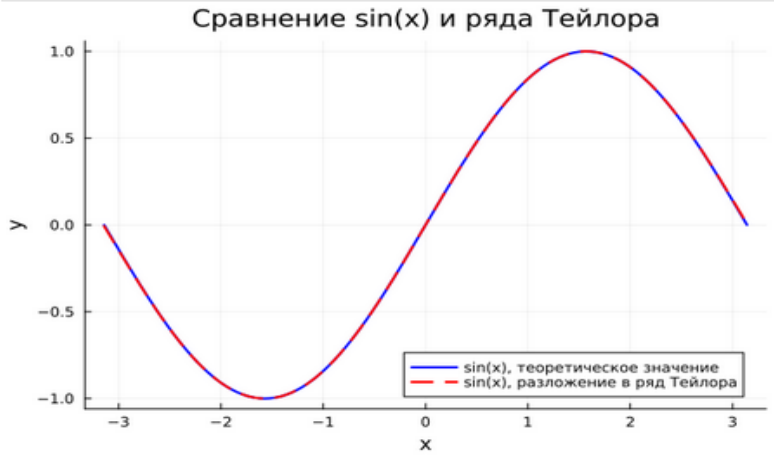


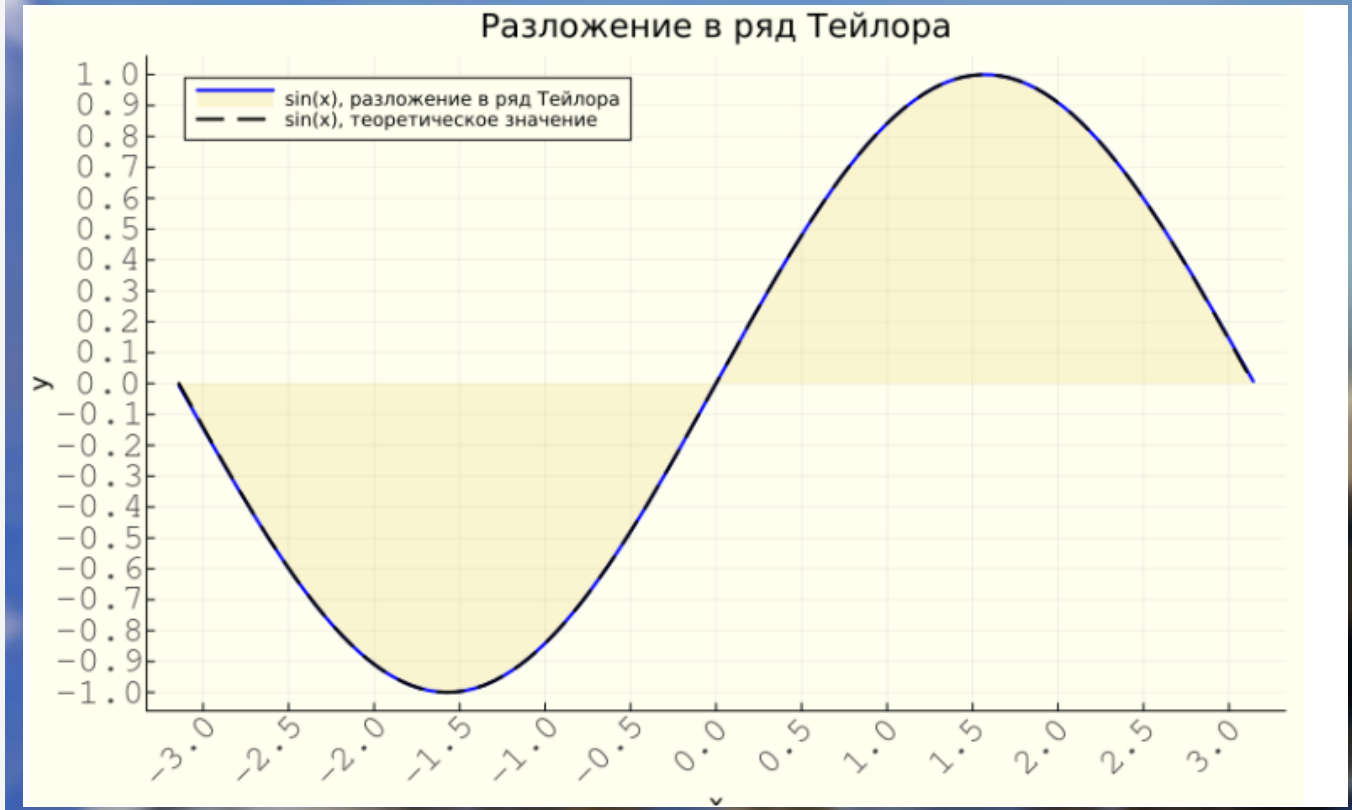
Рис. 5.6. Вид графиков из рис. 5.5 после добавления опций при их построении

```
# сохранение графика в файле в формате pdf или png:
savefig("taylor.pdf")
savefig("taylor.png")
```

```
"C:\\Users\\KRIS\\Desktop\\Pere\\taylor.png"
```

```
using Plots

# Используем gr() вместо pyplot()
gr()

# задание функции sin(x):
sin_theor(x) = sin(x)

# 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(
    # функция sin_taylor:
    sin_taylor, -π, π,

    # подпись в легенде, цвет и тип линии:
    label = "sin(x), разложение в ряд Тейлора",
    line=(:blue, 2, :solid),

    # размер графика:
    size=(800, 500),
```

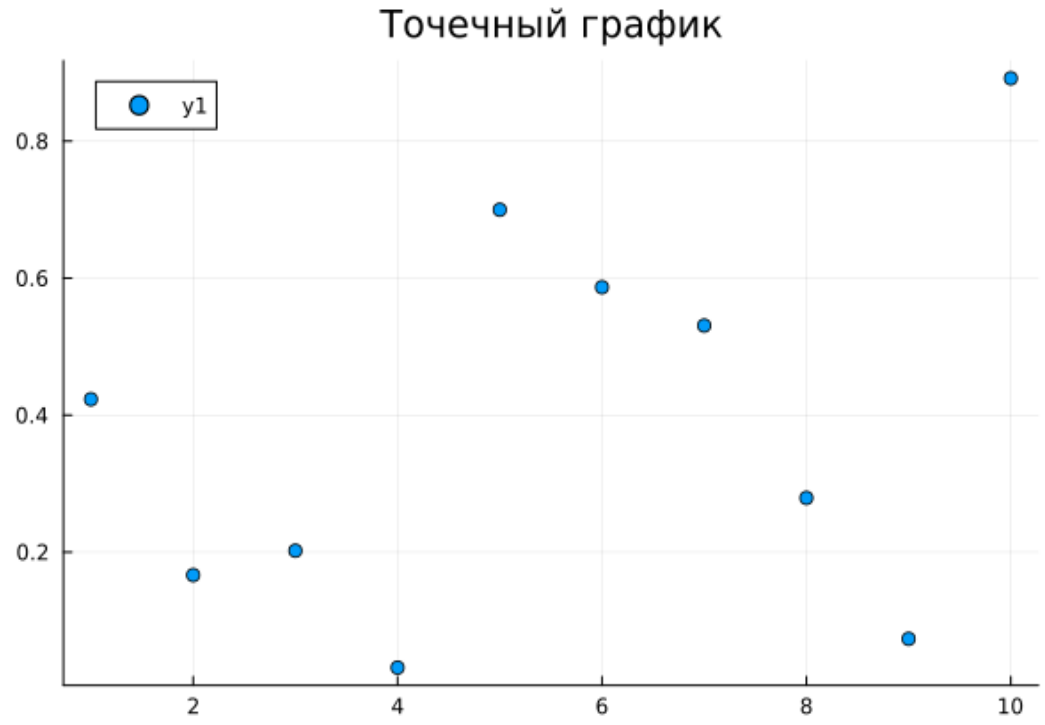
```
    # размер графика:
    size=(800, 500),

    # 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",

    # название графика:
    title = "Разложение в ряд Тейлора",

    # поворот значений, заданный по оси x:
    xrotation = 45,

    # заливка области графика цветом:
    fillrange = 0,
    fillalpha = 0.3,
    fillcolor = :lightgoldenrod,

    # задание цвета фона:
    background_color = :ivory
)

plot!(
    # функция sin_theor:
    sin_theor, -π, π,

    # подпись в легенде, цвет и тип линии:
    label = "sin(x), теоретическое значение",
    line=(:black, 2, :dash))
```



Точечный график

Рис. 5.7. График десяти случайных значений на плоскости ¶

```
using Plots

gr()

x = range(1, 10, length=10)
y = rand(10)

# параметры построения графика:
plot(x, y,
    seriestype = :scatter,
    title = "Точечный график")
```



5.8. График пятидесяти случайных значений на плоскости с различными опциями отображения

```
using Plots

gr()

# параметры распределения точек на плоскости:
n = 50
x = rand(n)
y = rand(n)
ms = rand(50) * 30

# параметры построения графика:
scatter(x, y, markersize = ms)
```

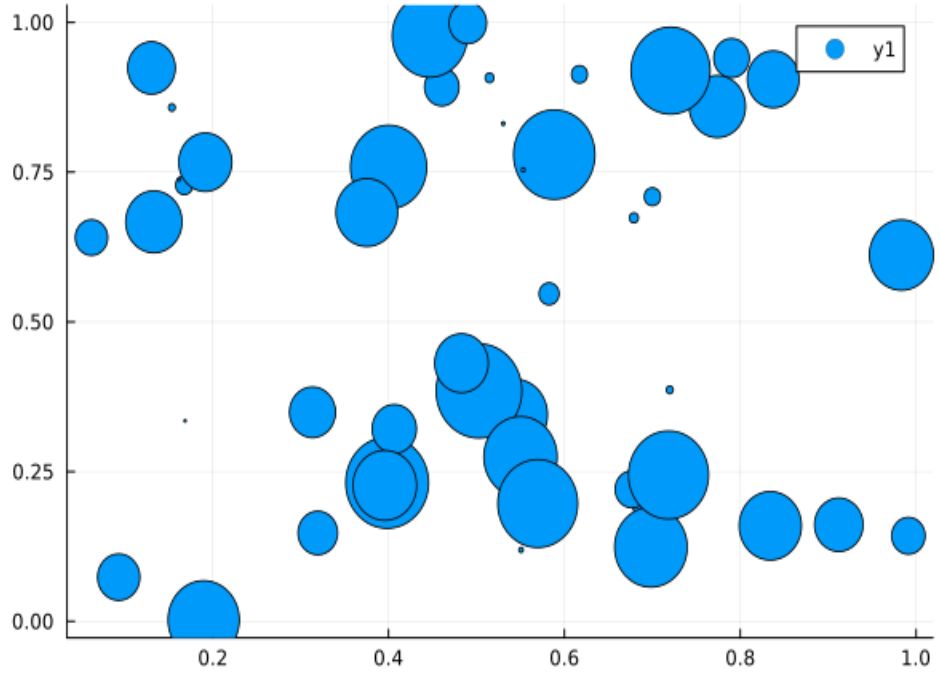


Рис. 5.9. График пятидесяти случайных значений в пространстве с различными опциями отображения ¶

```
using Plots

gr()

# 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, markersize = ms)
```

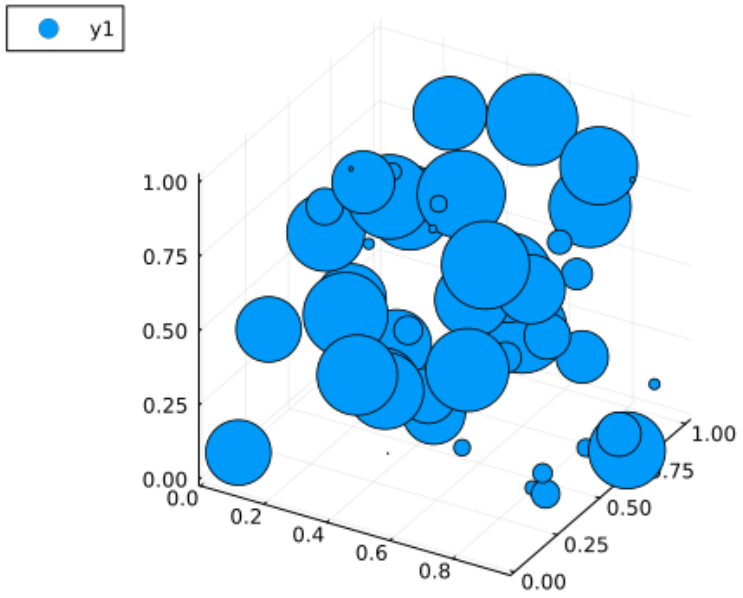
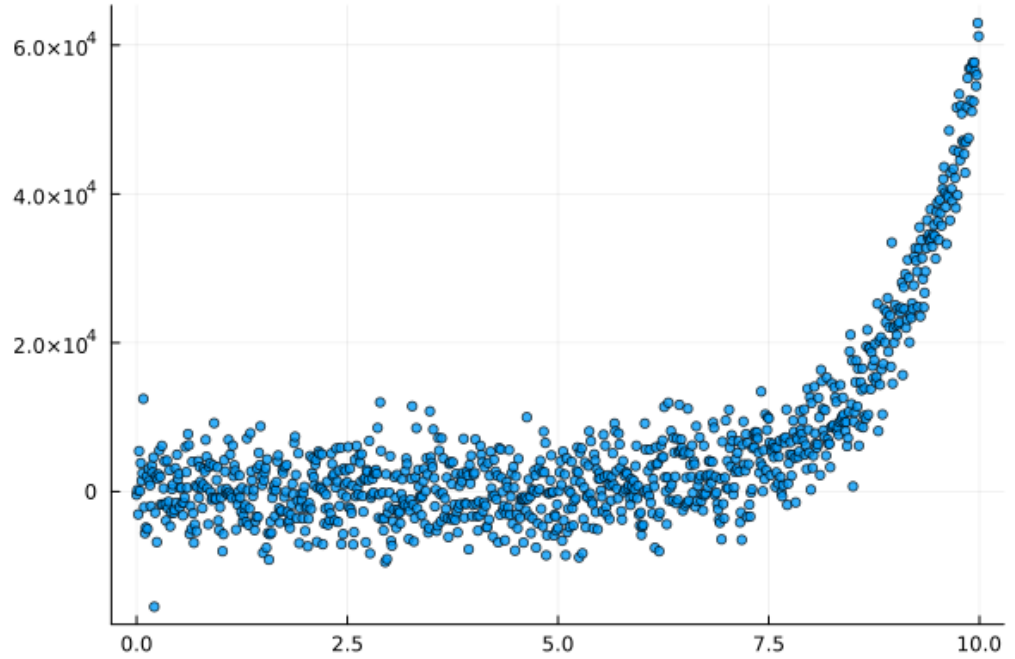


Рис. 5.10. Пример функции

```
using Plots

gr()

# 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, alpha = 0.8, legend = false)
```



Аппроксимация данных

```julia
using Plots, LinearAlgebra

gr()

# Данные
x = collect(0:0.01:9.99)
y = exp.(1 .+ x) + 4000 * randn(1000)

# График данных
scatter(x, y, markersize=3, alpha=0.8, legend=false,
    title="Аппроксимация полиномом 5-й степени",
    xlabel="x", ylabel="y")

# Аппроксимация
A = hcat(ones(1000), x, x.^2, x.^3, x.^4, x.^5)
c = A \ y
y_fit = A * c

# График аппроксимации
plot!(x, y_fit, linewidth=3, color=:red)
```

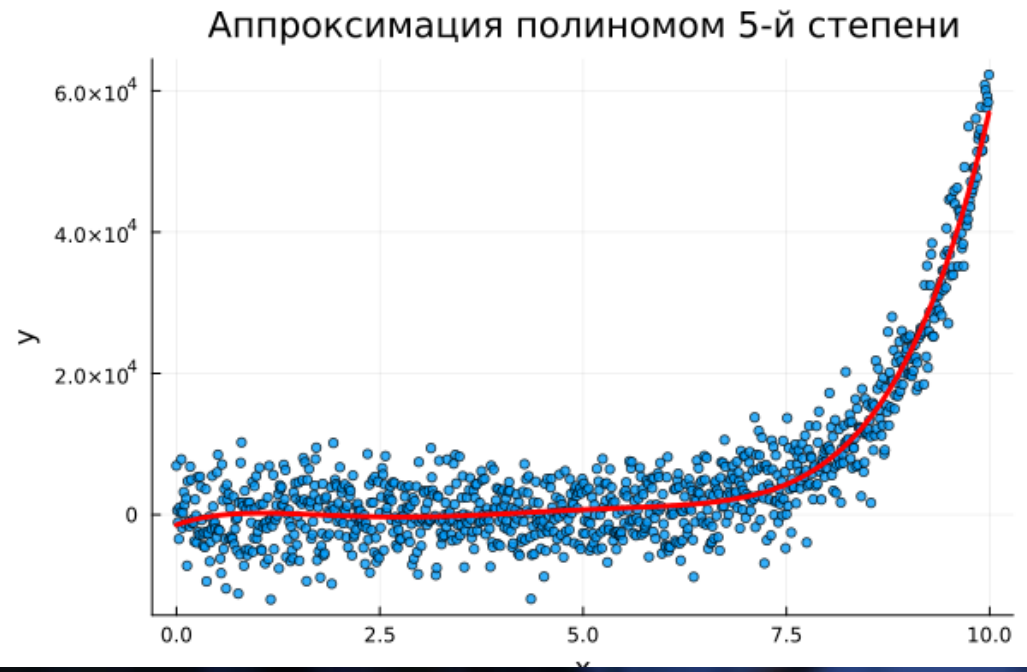


Рис. 5.12. Пример отдельно построенной траектории

```julia
using Plots

gr()

# пример случайной траектории
plot(randn(100),
    ylabel = "y1",
    leg = :topright,
    grid = false)
```

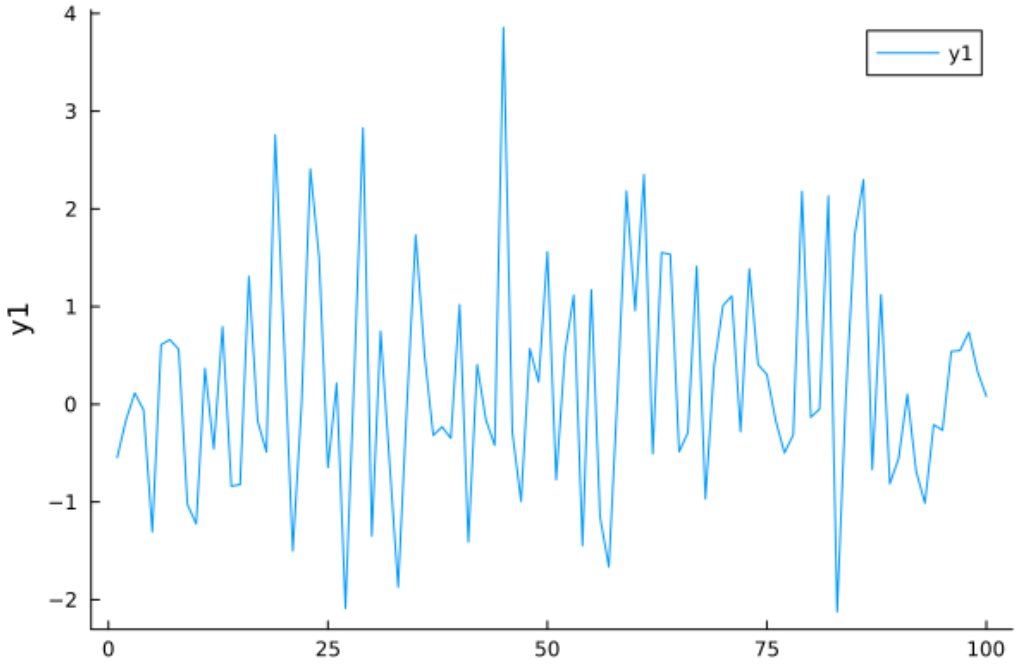
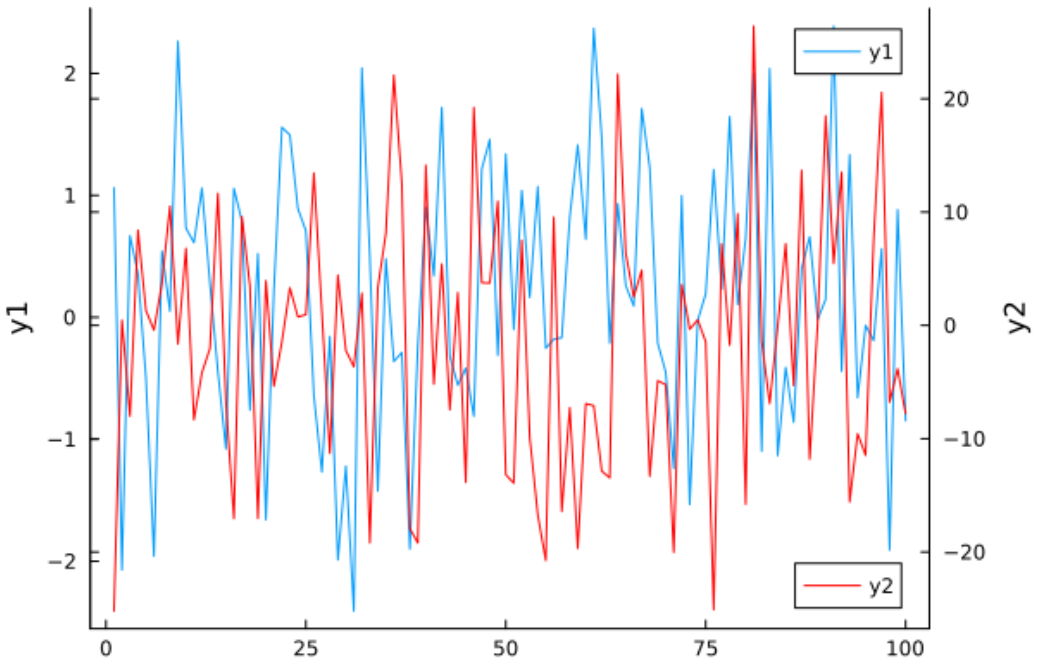


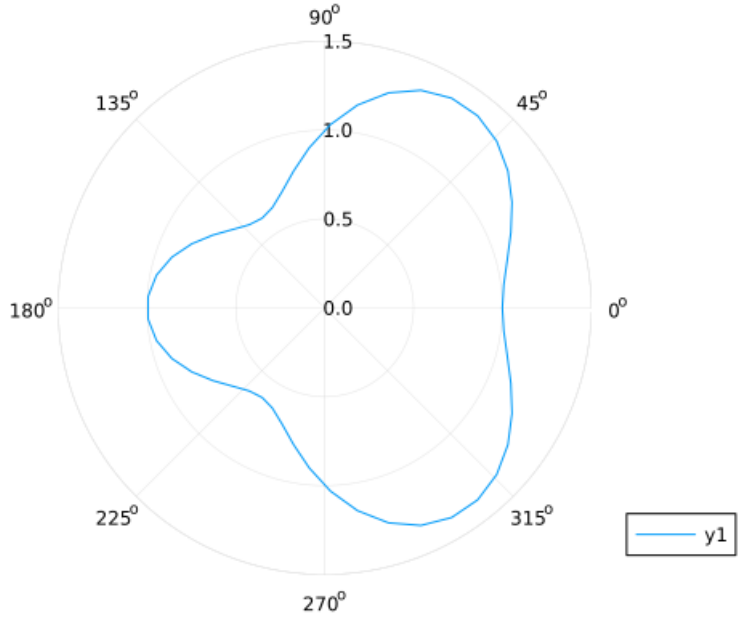
Рис. 5.13. Пример двух траекторий на одном графике с двумя осями ординат ¶

```
using Plots

gr()

# пример случайной траектории
plot(randn(100),
    ylabel = "y1",
    leg = :topright,
    grid = false)

# пример добавления на график второй случайной траектории
plot!(twinx(), randn(100) * 10,
    color = :red,
    ylabel = "y2",
    leg = :bottomright,
    grid = false,
    box = :on)
```



5.14. График функции, заданной в полярных координатах

```
using Plots

gr()

# функция в полярных координатах:
r(θ) = 1 + cos(θ) * sin(θ)^2

# полярная система координат:
θ = range(0, stop=2π, length=50)

# график функции, заданной в полярных координатах:
plot(θ, r.(θ),
    proj = :polar,
    lims = (0, 1.5))
```



Полярные координаты

Рис. 5.15. Параметрический график кривой на плоскости

```
[18]: using Plots
      gr()
      # Usar directamente las funciones sin asignar a variables
      plot(t -> sin(t), t -> sin(2t), 0, 2π, leg=false, fill=(0, :orange))
```

[18]:

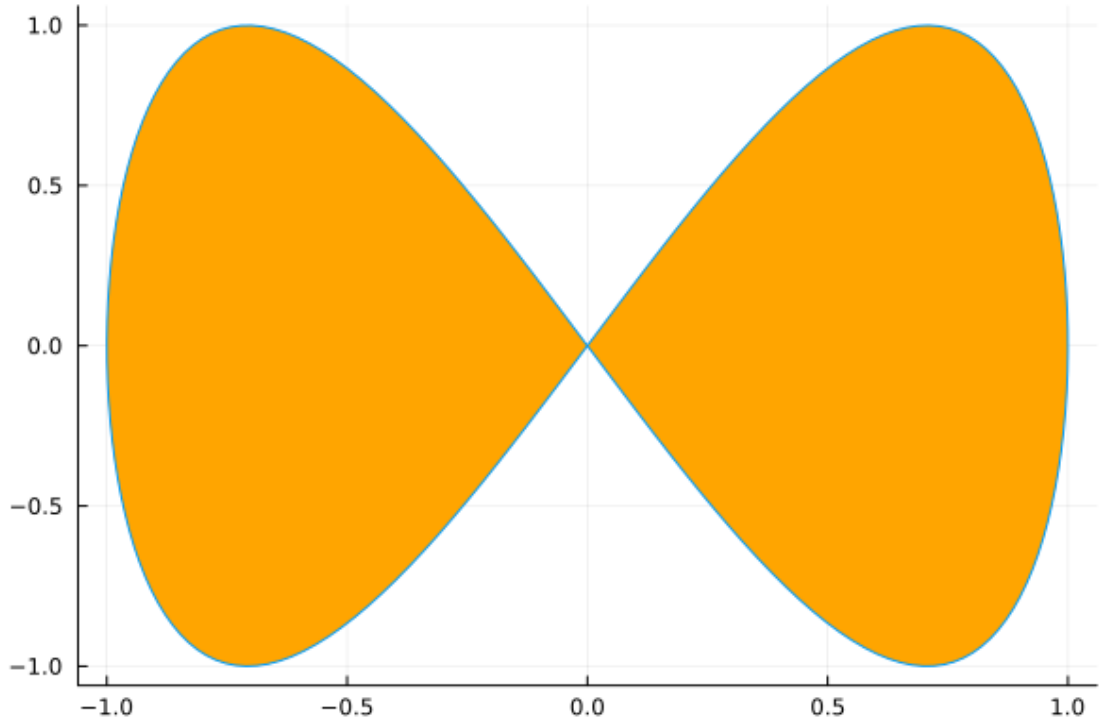


Рис. 5.16. Параметрический график кривой в пространстве ¶

```
using Plots
gr()
t = range(0, stop=10, length=1000)
x = cos.(t)
y = sin.(t)
z = sin.(5t)
plot(x, y, z)
```

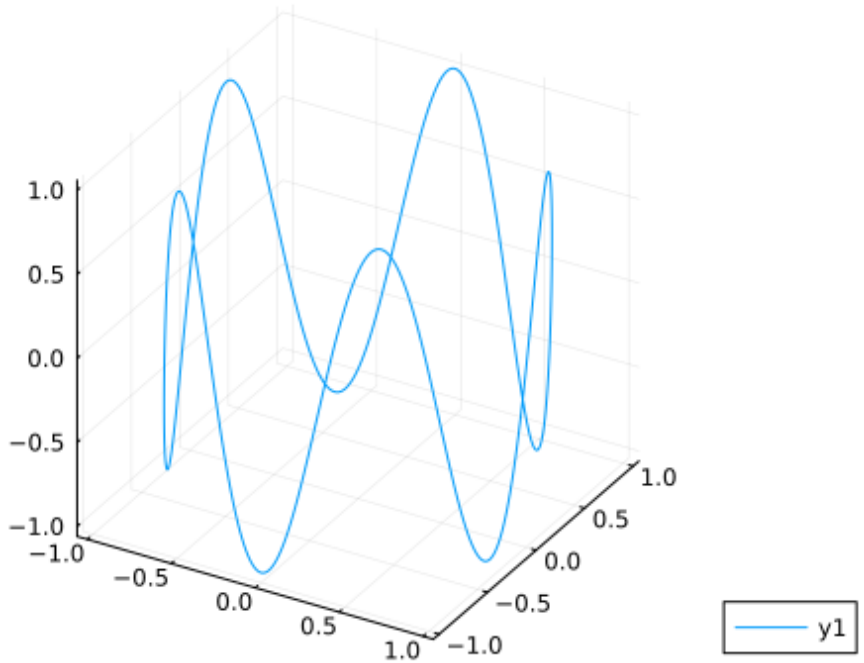


График поверхности

5.2.8. График поверхности

```julia
using Plots

# Используем gr() вместо pyplot()
gr()

# Функция поверхности
f(x, y) = x^2 + y^2

# Создаем сетку координат
x = -10:10
y = -10:10

# ===== Рис. 5.17. График поверхности (surface()) =====
p1 = surface(x, y, f,
    title = "Рис. 5.17. График поверхности (surface())",
    xlabel = "x",
    ylabel = "y",
    zlabel = "f(x,y) = x² + y²",
    color = :viridis,
    camera = (45, 45))

# ===== Рис. 5.18. График поверхности (plot() wireframe) =====
p2 = plot(x, y, f,
    linetype = :wireframe,
    title = "Рис. 5.18. График поверхности (wireframe)",
    xlabel = "x",
    ylabel = "y",
    zlabel = "f(x,y) = x² + y²",
    color = :red,
    linewidth = 1,
    camera = (45, 45))

# ===== Рис. 5.19. Сглаженный график поверхности =====
x_smooth = range(-10, 10, length=50)
y_smooth = range(-10, 10, length=50)
p3 = surface(x_smooth, y_smooth, f,
    title = "Рис. 5.19. Сглаженный график поверхности",
    xlabel = "x",
    ylabel = "y",
    zlabel = "f(x,y) = x² + y²",
    color = :heat,
    camera = (45, 45))

# ===== Рис. 5.20. График поверхности с изменённым углом зрения =====
x_angle = range(-2, 2, length=100)
y_angle = range(-2, 2, length=100)
f_angle(x, y) = x*y - x - y + 1

p4 = plot(x_angle, y_angle, f_angle,
    linetype = :surface,
    c = cgrad([:red, :blue]),
    camera = (-30, 30),
    title = "Рис. 5.20. Поверхность с изменённым углом зрения",
    xlabel = "x",
    ylabel = "y",
    zlabel = "f(x,y) = xy - x - y + 1")
```

Рис. 5.17. График поверхности (surface())

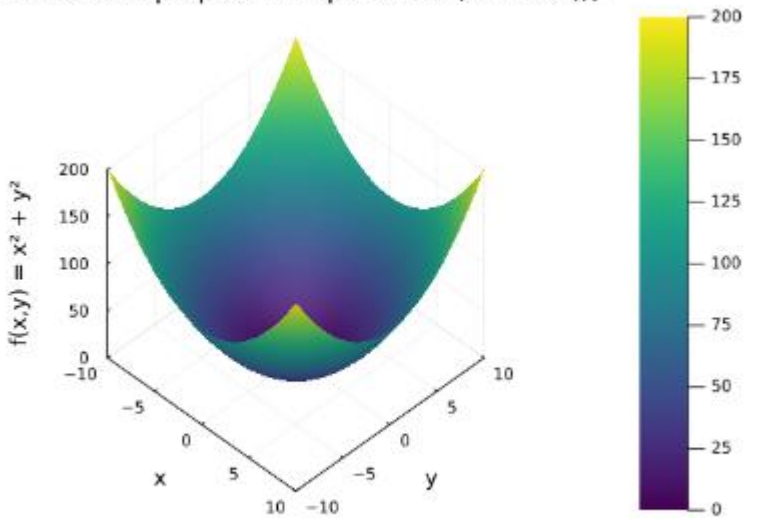


Рис. 5.18. График поверхности (wireframe)

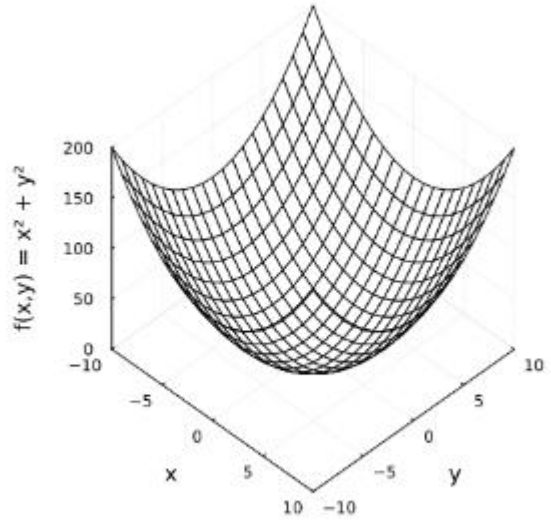


Рис. 5.19. Сглаженный график поверхности

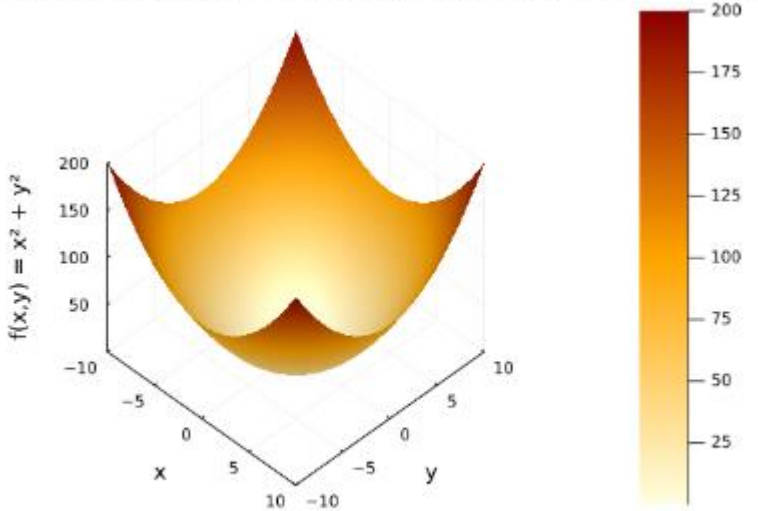


Рис. 5.20. Поверхность с изменённым углом зрения

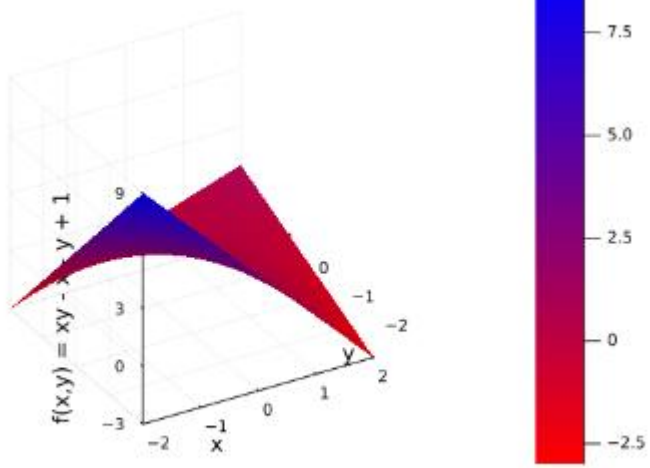
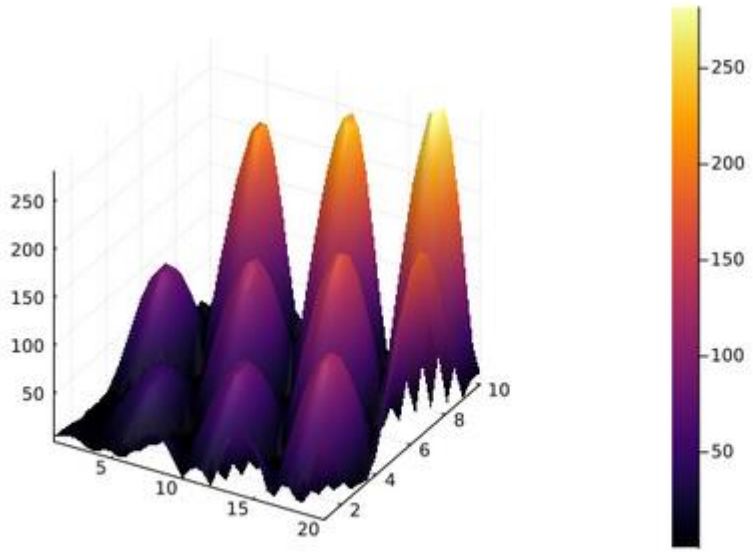
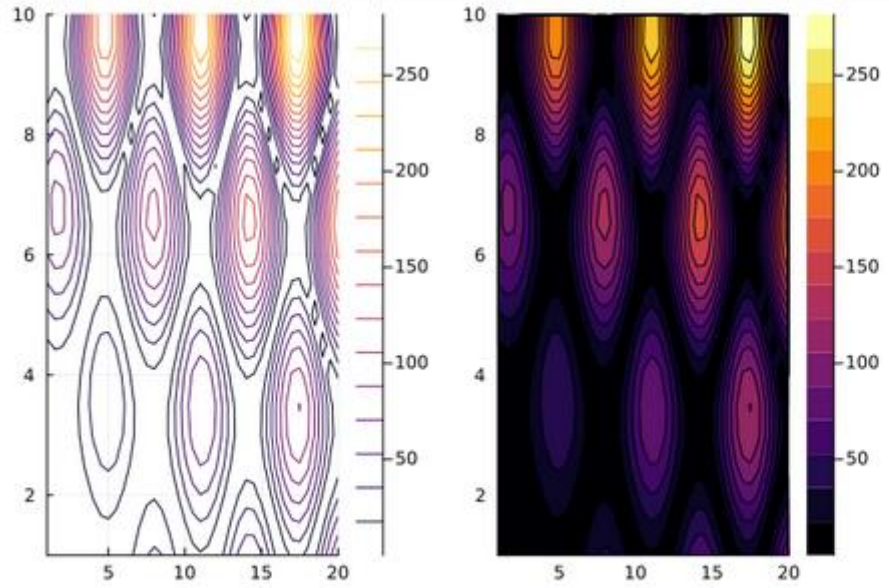


Рис. 5.21. График поверхности, заданной функцией $g(x, y) = (3x + y^2)|\sin(x) + \cos(y)|$

```
using Plots

gr()

x = 1:0.5:20
y = 1:0.5:10
g(x, y) = (3x + y^2) * abs(sin(x) + cos(y))

plot(x, y, g,
    linetype = :surface)
```

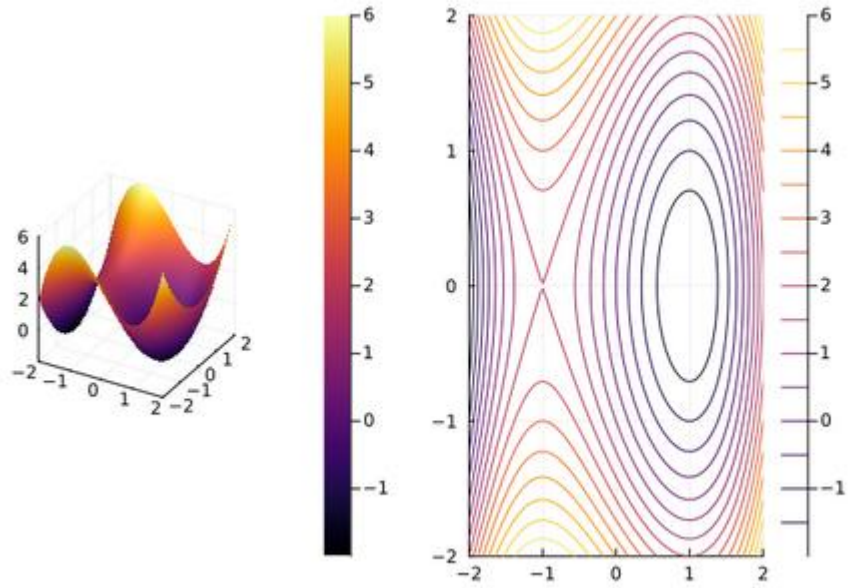



```
using Plots

gr()

x = 1:0.5:20
y = 1:0.5:10
g(x, y) = (3x + y^2) * abs(sin(x) + cos(y))

# Рис. 5.22
p1 = contour(x, y, g)

# Рис. 5.23
p2 = contour(x, y, g, fill=true)

plot(p1, p2, layout = (1, 2))
```

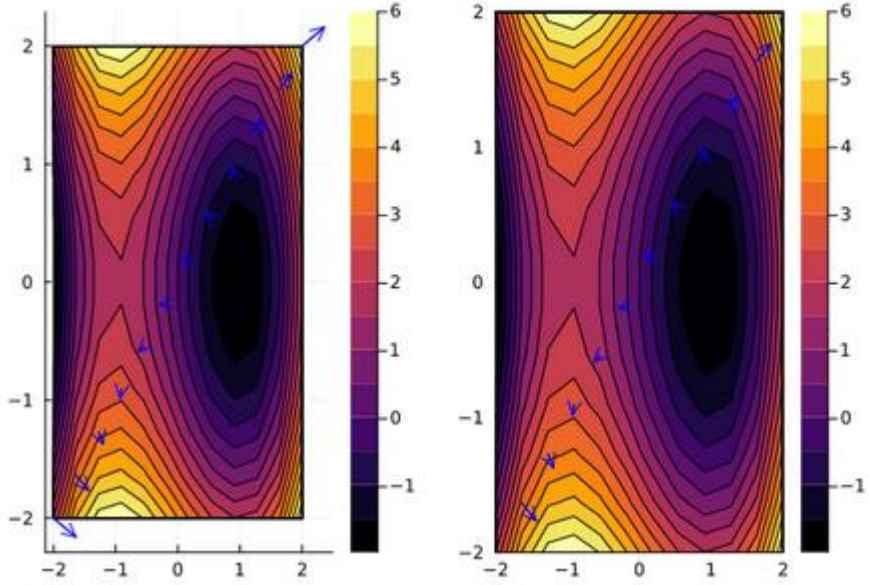


5.2.10. Векторные поля

```
using Plots

gr()

# 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:
p1 = plot(X, Y, h, linetype = :surface)

# построение линий уровня:
p2 = contour(X, Y, h)

plot(p1, p2, layout = (1, 2))
```



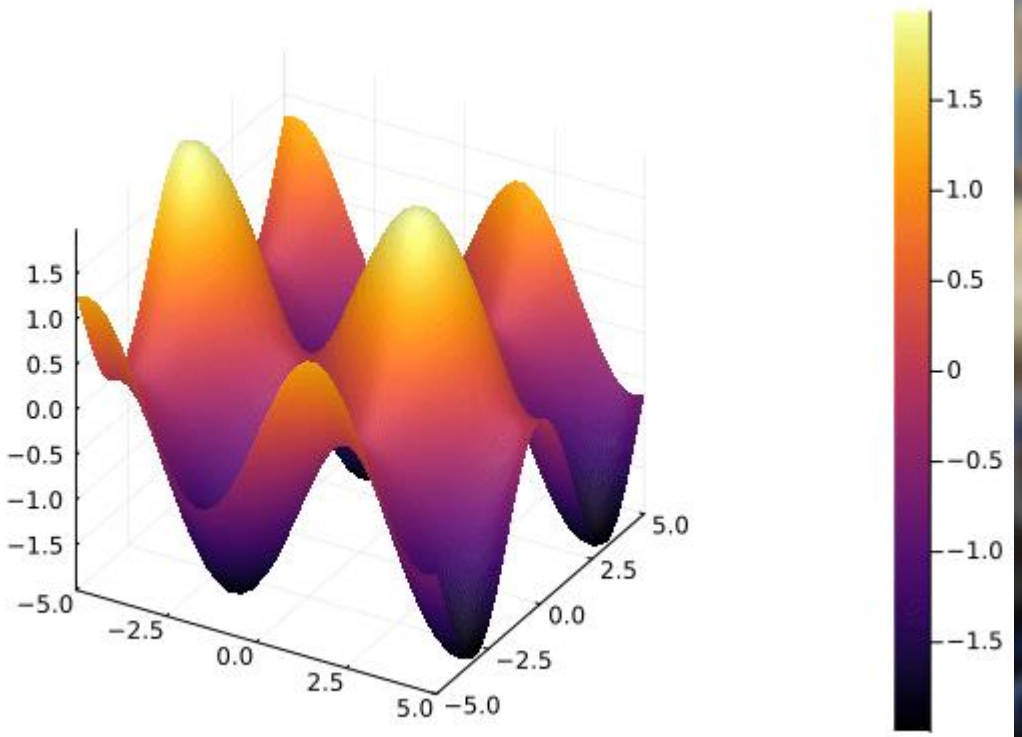
```
using Plots

gr()

# определение функции:
h(x, y) = x^3 - 3x + y^2

# градиент:
x = range(-2, stop=2, length=12)
y = range(-2, stop=2, length=12)

# производная от исходной функции:
dh(x, y) = [3x^2 - 3; 2y] / 25

# Рис. 5.26. Векторное поле функции
p1 = contour(x, y, h, fill=true)
quiver!(x, y, quiver=dh, color=:blue)

# Рис. 5.27. Векторное поле функции (скорректирована область видимости)
p2 = contour(x, y, h, fill=true)
quiver!(x, y, quiver=dh, color=:blue)
xlims!(-2, 2)
ylims!(-2, 2)

# Mostrar ambas gráficas
plot(p1, p2, layout=(1,2))
```



5.2.11.1. Gif-анимация

```
using Plots

gr()

# Рис. 5.28. Статичный график поверхности
X = range(-5, stop=5, length=40)
Y = range(-5, stop=5, length=40)
surface(X, Y, (x,y) -> sin(x) + cos(y))

# Рис. 5.29. Анимированный график поверхности
X = range(-5, stop=5, length=40)
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
```

```
[ Info: Saved animation to C:\Users\KRIS\Desktop\Pere\tmp.gif
```



Гипоциклоида

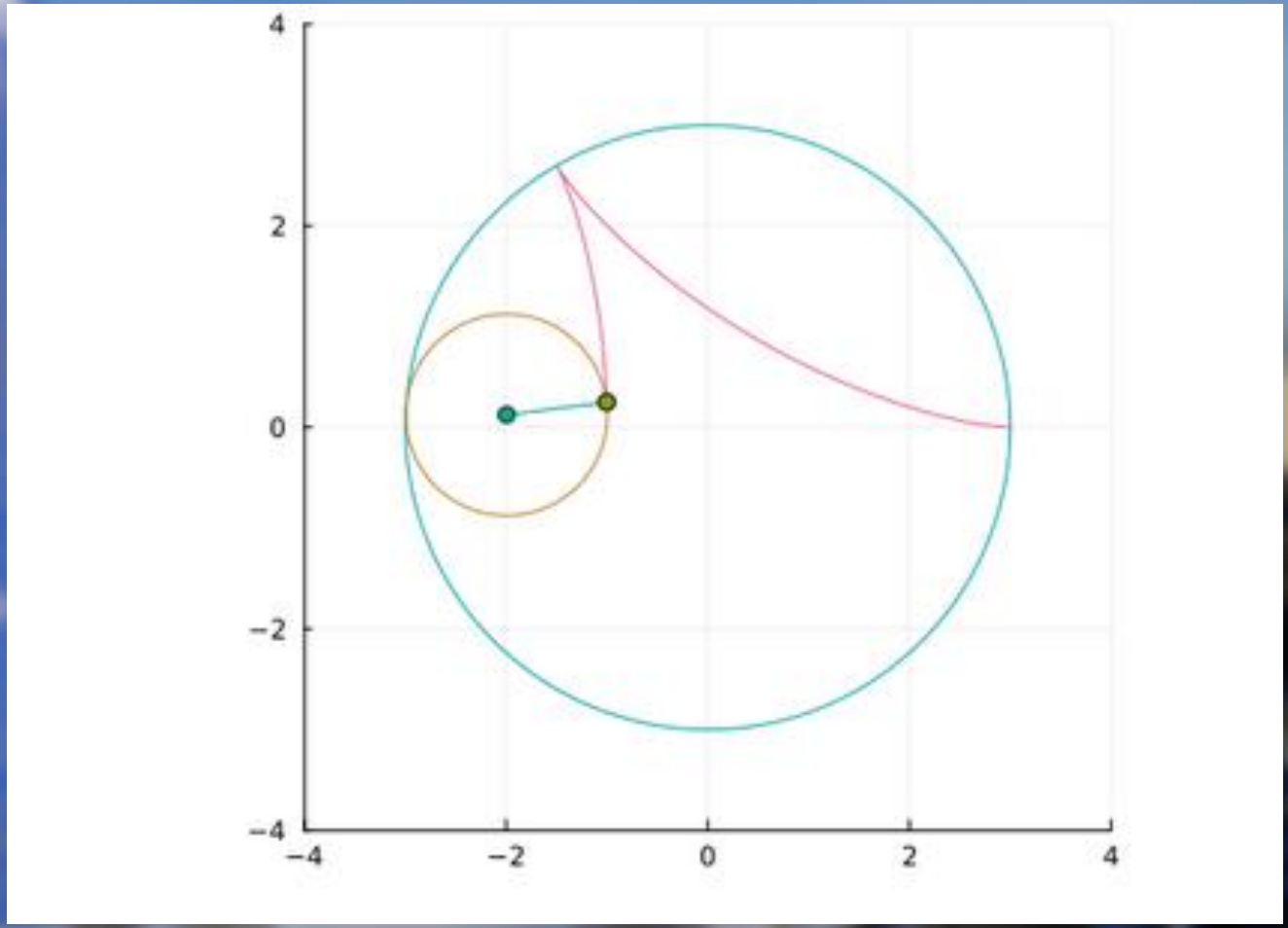
5.2.11.2. Гипоциклоида

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

$$\begin{cases} x = r(k-1)\left(\cos t + \dfrac{\cos((k-1)t)}{k-1}\right), \\ y = r(k-1)\left(\sin t - \dfrac{\sin((k-1)t)}{k-1}\right), \end{cases}$$

где $k = \dfrac{R}{r}$, R — радиус неподвижной окружности, r — радиус катящейся окружности. Модуль величины k определяет форму гипоциклоиды.

[26]:
```
using Plots

gr()

# радиус малой окружности:
r_val = 1
# коэффициент для построения большой окружности:
k = 3
# число отсчётов:
n = 100

# массив значений угла θ:
θ = collect(0:2π/100:2π+2π/100)

# массивы значений координат:
X = r_val*k*cos.(θ)
Y = r_val*k*sin.(θ)

# задаём оси координат:
plt = plot(5, xlim=(-4,4), ylim=(-4,4), aspect_ratio=1, legend=false)

# Рис. 5.30. Большая окружность гипоциклоиды:
plot!(plt, X, Y)

# Рис. 5.31. Половина пути гипоциклоиды:
i = 50
t = θ[1:i]
x = r_val*(k-1)*cos.(t) + r_val*cos.((k-1)*t)
y = r_val*(k-1)*sin.(t) - r_val*sin.((k-1)*t)
plot!(x, y)

# Рис. 5.32. Малая окружность гипоциклоиды:
xc = r_val*(k-1)*cos(t[end]) .+ r_val*cos.(θ)
yc = r_val*(k-1)*sin(t[end]) .+ r_val*sin.(θ)
plot!(xc, yc)

# Рис. 5.33. Малая окружность гипоциклоиды с добавлением радиуса:
x1 = transpose([r_val*(k-1)*cos(t[end]) x[end]])
y1 = transpose([r_val*(k-1)*sin(t[end]) y[end]])
plot!(x1, y1, markershape=:circle, markersize=4)
scatter!([x[end]], [y[end]])
```

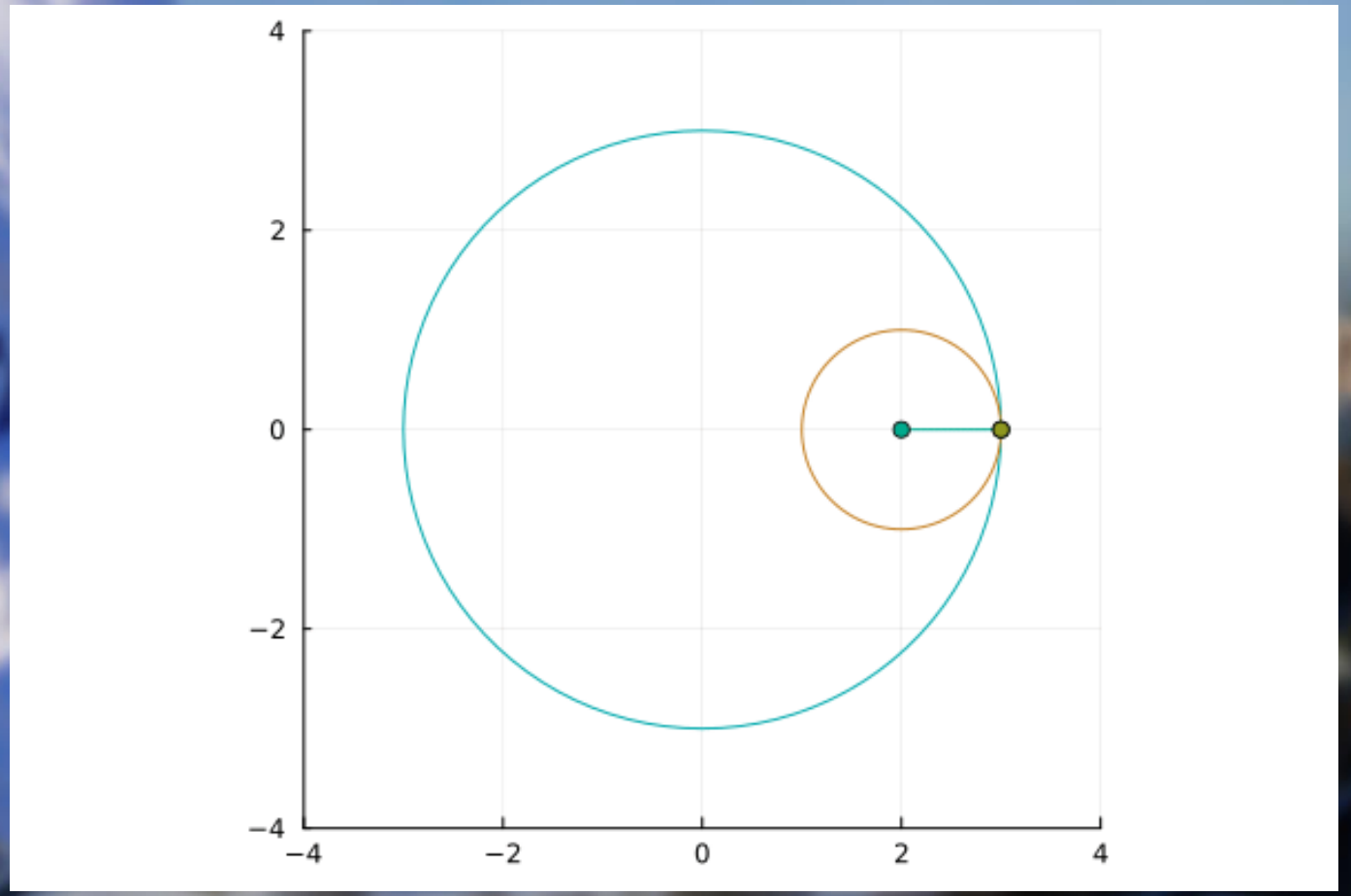


Анимация движения гипоциклоиды

Рис. 5.34. Анимация движения гипоциклоиды

```
using Plots

gr()

# радиус малой окружности:
r_val = 1
# коэффициент для построения большой окружности:
k = 3
# число отсчётов:
n = 100

# массив значений угла θ:
θ = collect(0:2π/100:2π+2π/100)

# массивы значений координат:
X = r_val*k*cos.(θ)
Y = r_val*k*sin.(θ)

# Рис. 5.34. Анимация движения гипоциклоиды
anim = @animate for i in 1:n

    # задаём оси координат:
    plt = plot(5, xlim=(-4,4), ylim=(-4,4), aspect_ratio=1, legend=false)

    # большая окружность:
    plot!(plt, X, Y)

    t = θ[1:i]

    # гипоциклоида:
    x = r_val*(k-1)*cos.(t) + r_val*cos.((k-1)*t)
    y = r_val*(k-1)*sin.(t) - r_val*sin.((k-1)*t)
    plot!(x, y)

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

    # радиус малой окружности:
    x1 = transpose([r_val*(k-1)*cos(t[end]) x[end]])
    y1 = transpose([r_val*(k-1)*sin(t[end]) y[end]])
    plot!(x1, y1, markershape=:circle, markersize=4)
    scatter!([x[end]], [y[end]])

end

# Сохраним анимацию в gif-файл:
gif(anim, "hypocycloid.gif")
```

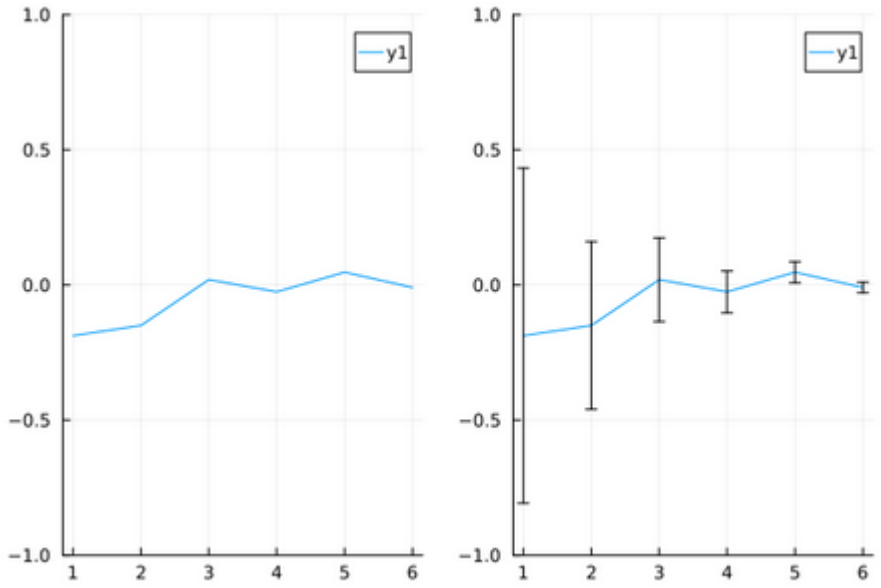
```
[ Info: Saved animation to C:\Users\KRIS\Desktop\Pere\hypocycloid.gif
```



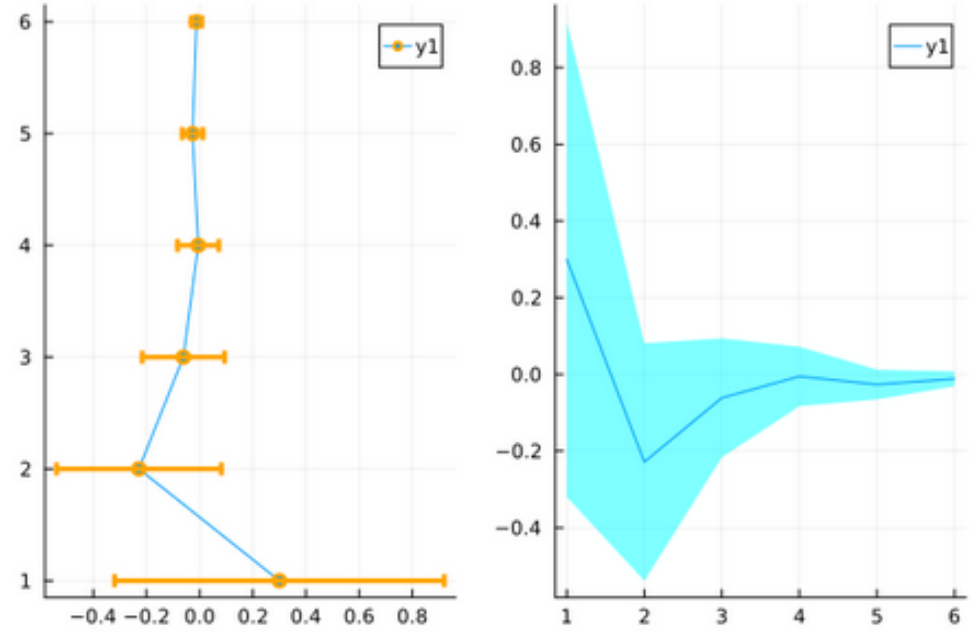
Errorbars

5.2.12. Errorbars ¶

```
using Plots, Statistics

gr()

# задание массива значений:
sds = [1, 1/2, 1/4, 1/8, 1/16, 1/32]

# сгенерируем массив ошибок:
n = 10
y = [mean(sd*randn(n)) for sd in sds]
errs = 1.96 * sds / sqrt(n)

# Рис. 5.35. График исходных значений:
p1 = plot(y, ylims=(-1,1))

# Рис. 5.36. График исходных значений с отклонениями:
p2 = plot(y, ylims=(-1,1), err=errs)

# Mostrar ambas gráficas
plot(p1, p2, layout=(1,2))
```

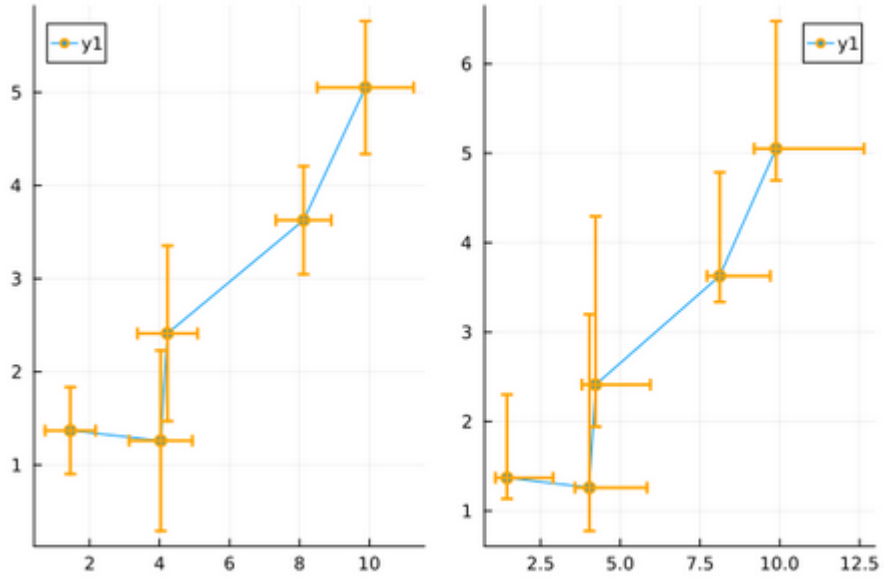



```
using Plots, Statistics

gr()

# задание массива значений:
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)

# Рис. 5.37. Поворот графика:
p1 = plot(y, 1:length(y), xerr=errs, marker=stroke(3,:orange))

# Рис. 5.38. Заполнение цветом:
p2 = plot(y, ribbon=errs, fillcolor=:cyan)

# Mostrar ambas gráficas
plot(p1, p2, layout=(1,2))
```



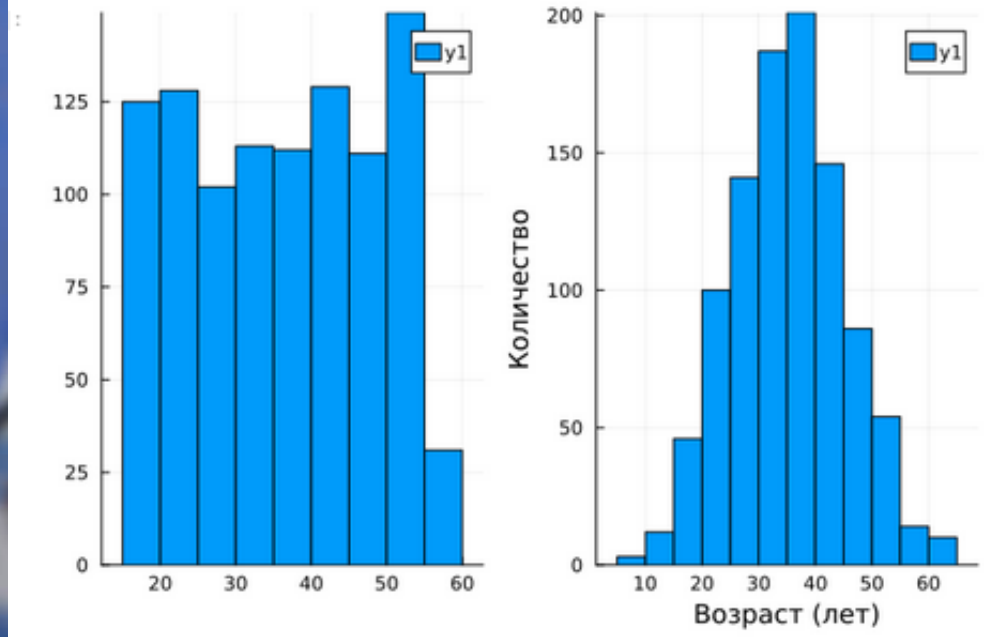
Использование пакета Distributions

```
using Plots, Statistics

gr()

# График ошибок по двум осям
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) = 1.96 * std(v) / sqrt(n)
xerr = map(f, x)
yerr = map(f, y)
x_mean = map(mean, x)
y_mean = map(mean, y)

# Рис. 5.39. График ошибок по двум осям:
p1 = plot(x_mean, y_mean, xerr=xerr, yerr=yerr, marker=stroke(2, :orange))

# Рис. 5.40. График асимметричных ошибок по двум осям:
p2 = plot(x_mean, y_mean, xerr=(0.5xerr, 2xerr), yerr=(0.5yerr, 2yerr), marker=stroke(2, :orange))

# Mostrar ambas gráficas
plot(p1, p2, layout=(1,2))
```



5.2.13. Использование пакета Distributions

```
using Plots, Statistics

gr()

# Рис. 5.41. Гистограмма, построенная по массиву случайных чисел:
ages1 = rand(15:55, 1000)
p1 = histogram(ages1)

# Рис. 5.42. Гистограмма нормального распределения (usando randn):
ages2 = 35.0 .+ 10.0 .* randn(1000)
p2 = histogram(ages2, xlabel="Возраст (лет)", ylabel="Количество")

# Mostrar ambas gráficas
plot(p1, p2, layout=(1,2))
```



Подграфики

5.2.14. Подграфики

```
using Plots

gr()

# построение серии графиков:
x = range(-2, 2, length=10)
y = rand(10, 4)

# Рис. 5.44. Серия из 4-х графиков в ряд:
p1 = plot(x, y, layout=(4, 1))

# Рис. 5.45. Серия из 4-х графиков в сетке:
p2 = plot(x, y, layout=4)

# Рис. 5.46. Серия из 4-х графиков разной высоты в ряд:
p3 = plot(x, y, layout=grid(4, 1, heights=[0.2, 0.3, 0.4, 0.1]))

# Mostrar las tres gráficas
plot(p1, p2, p3, layout=(1, 3), size=(1200, 400))
```

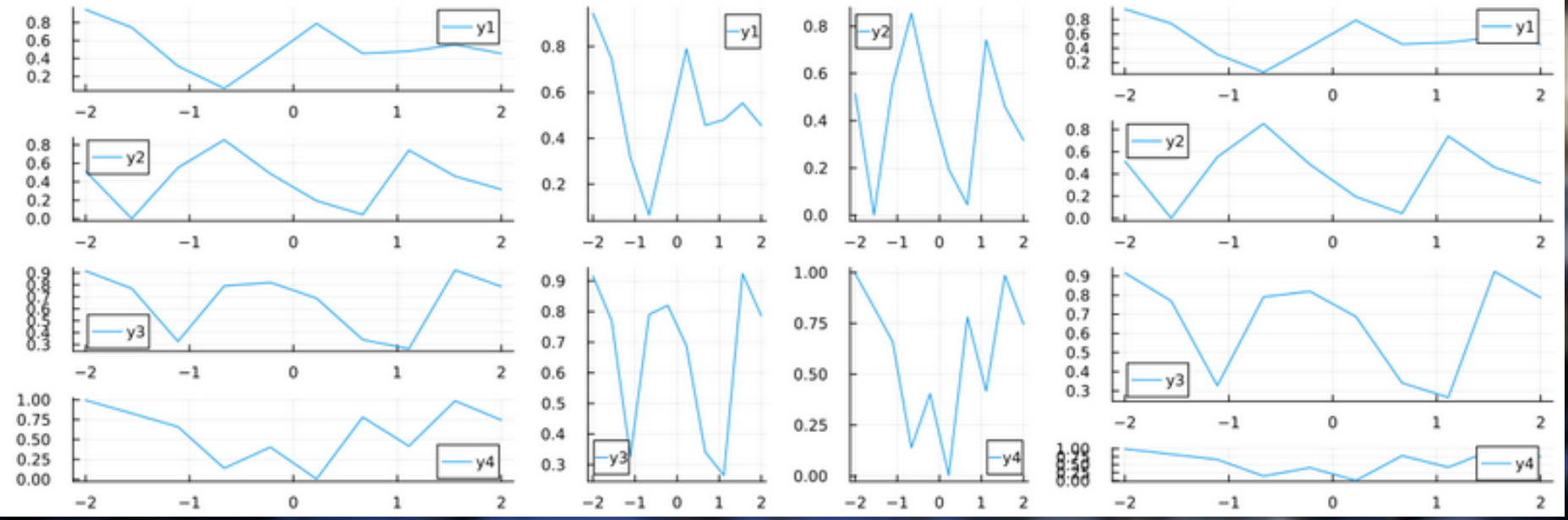


Рис. 5.47. Объединение нескольких графиков в одной сетке

```
using Plots

gr()

# график в виде линий:
p1 = plot(rand(10, 2))

# график в виде точек:
p2 = scatter(rand(10, 2))

# график в виде линий с оформлением:
p3 = plot(rand(10, 2), xlabel="Labelled plot of two columns", lw=2, title="Wide lines")

# гистограммы:
p4 = histogram(rand(10, 2))

# Рис. 5.47. Объединение нескольких графиков в одной сетке
plot(p1, p2, p3, p4, layout=(2, 2), legend=false, background_color=:ivory)
```

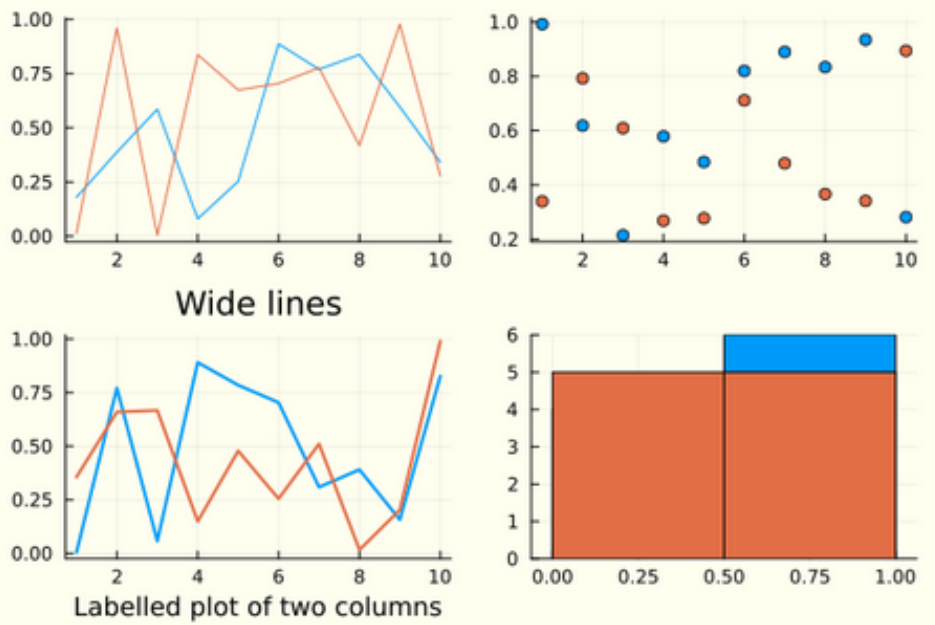


Рис. 5.48. Разнообразные варианты представления данных

```
using Plots

gr()

seriestypes = [:step, :sticks, :bar, :hline, :line, :path]
titles = ["step" "sticks" "bar" "hline" "line" "path"]

# Рис. 5.48. Разнообразные варианты представления данных
plot(rand(20, 6),
    st = seriestypes,
    layout = (2, 3),
    ticks = nothing,
    legend = false,
    title = titles,
    m = 3)
```

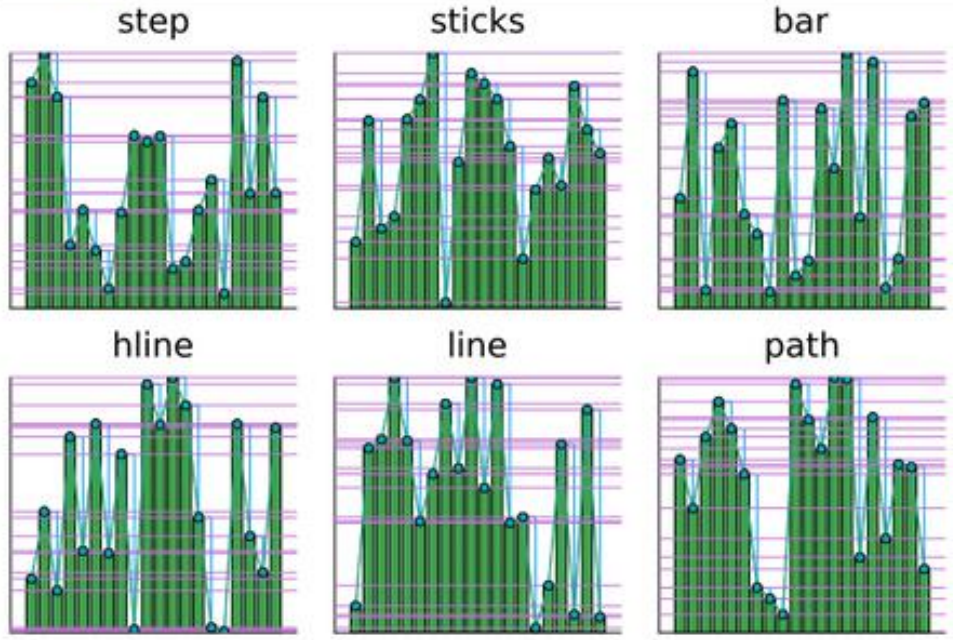
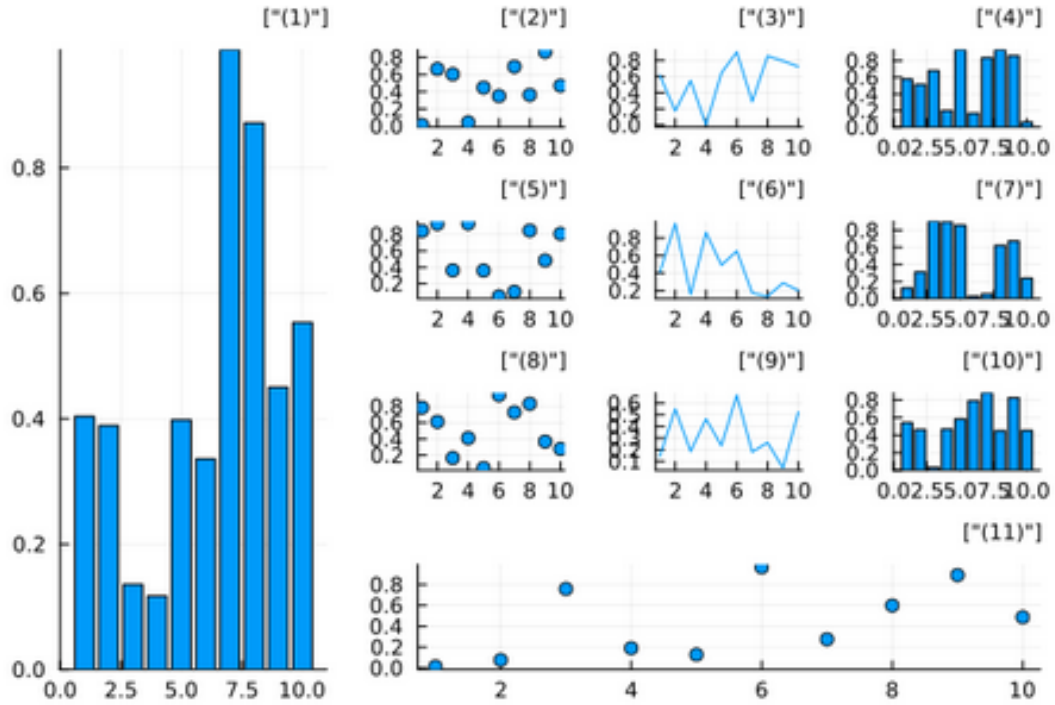


Рис. 5.49. Демонстрация применения сложного макета для построения графиков

```
using Plots

gr()

# Рис. 5.49. Демонстрация применения сложного макета для построения графиков
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))
```



Задание

5.3. Задание

1. Используя Jupyter Lab, повторите примеры из раздела 5.2. При этом дополните графики обозначениями осей координат, легендой с названиями траекторий, названиями графиков и т.п.

2. Выполните задания для самостоятельной работы (раздел 5.4).

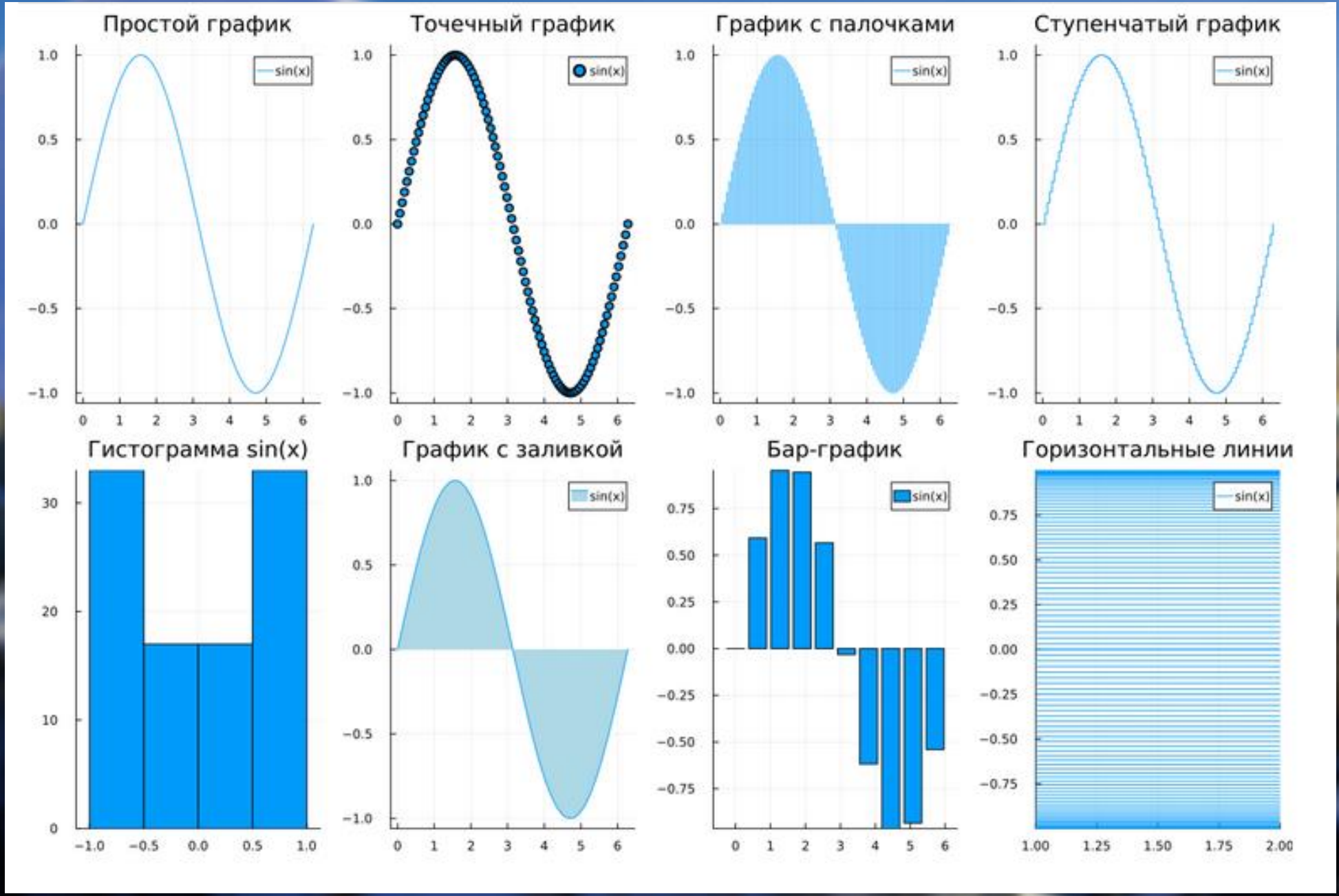
```
using Plots, Statistics, LinearAlgebra

# Используем gr() как стабильный бэкенд
gr()

# ===== ПОВТОРЕНИЕ ПРИМЕРОВ ИЗ РАЗДЕЛА 5.2 С ДОПОЛНИТЕЛЬНЫМИ ОПЦИЯМИ =====

# 1. Базовая функция с улучшенными подписями
println("1. Построение базовой функции...")
f(x) = (3x.^2 .+ 6x .- 9).*exp.(-0.3x)
x = range(-5, 10, length=151)
y = f.(x)

plot(x, y,
    title="График функции f(x) = (3x² + 6x - 9)e⁻⁰·³ˣ",
    xlabel="Переменная x",
    ylabel="Переменная y",
    color="blue",
    linewidth=2,
    label="f(x)",
    legend=:topright,
    grid=true)

# 2. Sin(x) и ряд Тейлора с улучшенной визуализацией
println("2. Сравнение sin(x) и ряда Тейлора...")
sin_theor(x) = sin(x)

function sin_taylor_improved(x)
    s = 0.0
    term = x
    for i in 0:4
        s += term
        term *= -x^2 / ((2*i+2)*(2*i+3))
    end
    return s
end
```

5.4. Задания для самостоятельного выполнения

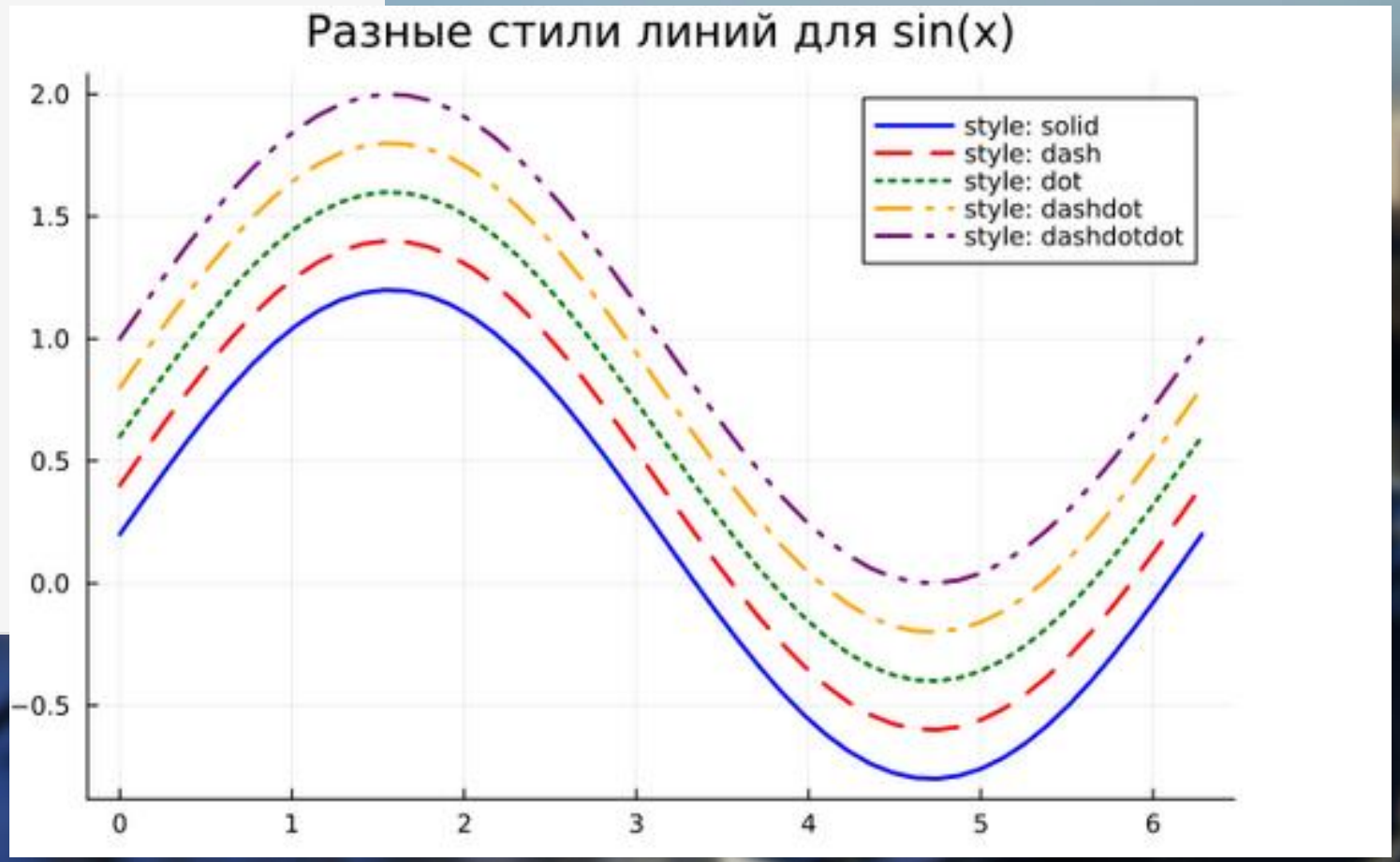
Задание 1

```
using Plots

function plot_all_sin_types()
    x = range(0, 2π, length=100)
    y = sin.(x)

    plots = [
        # 1. Простой график
        plot(x, y, title="Простой график", label="sin(x)"),

        # 2. Точечный график
        scatter(x, y, title="Точечный график", label="sin(x)"),

        # 3. График с палочками
        plot(x, y, st=:sticks, title="График с палочками", label="sin(x)"),

        # 4. Ступенчатый график
        plot(x, y, st=:step, title="Ступенчатый график", label="sin(x)"),

        # 5. Гистограмма
        histogram(y, title="Гистограмма sin(x)", label=""),

        # 6. График с областями
        plot(x, y, st=:path, fill=(0, :lightblue), title="График с заливкой", label="sin(x)"),

        # 7. График с барами
        bar(x[1:10:end], y[1:10:end], title="Бар-график", label="sin(x)"),

        # 8. График с горизонтальными линиями
        plot(x, y, st=:hline, title="Горизонтальные линии", label="sin(x)")
    ]

    plot(plots..., layout=(2,4), size=(1200, 800))
end

plot_all_sin_types()
```

Задания для самостоятельного выполнения

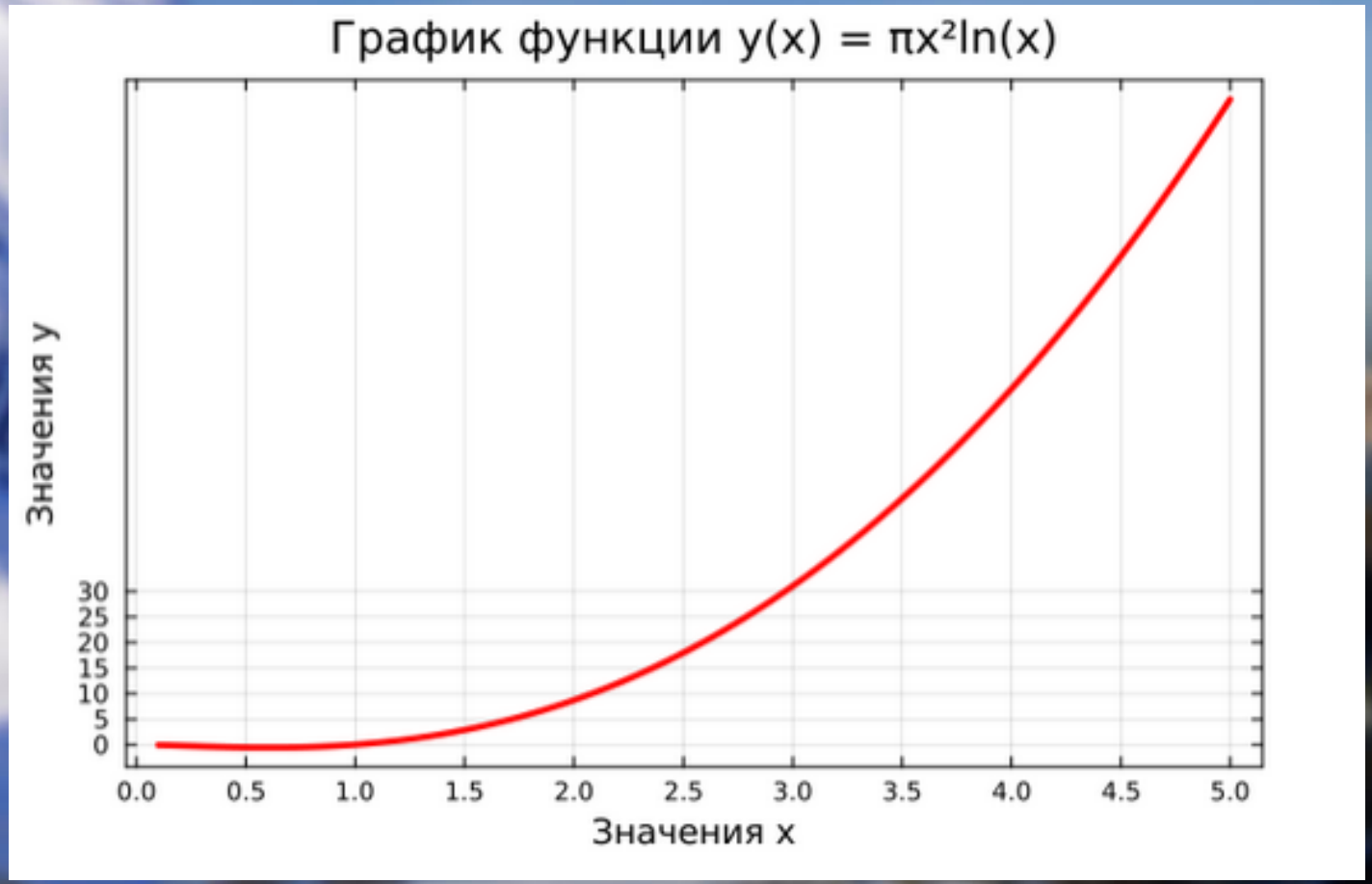


```julia
using Plots

function plot_sin_line_styles()
    x = range(0, 2π, length=50)
    y = sin.(x)

    # Все возможные стили линий
    line_styles = [:solid, :dash, :dot, :dashdot, :dashdotdot]
    line_colors = [:blue, :red, :green, :orange, :purple]

    p = plot(title="Разные стили линий для sin(x)")

    for (i, style) in enumerate(line_styles)
        plot!(x, y .+ i*0.2,
              line=style,
              color=line_colors[i],
              label="style: $style",
              linewidth=2)
    end

    return p
end

plot_sin_line_styles()
```



Задание 3

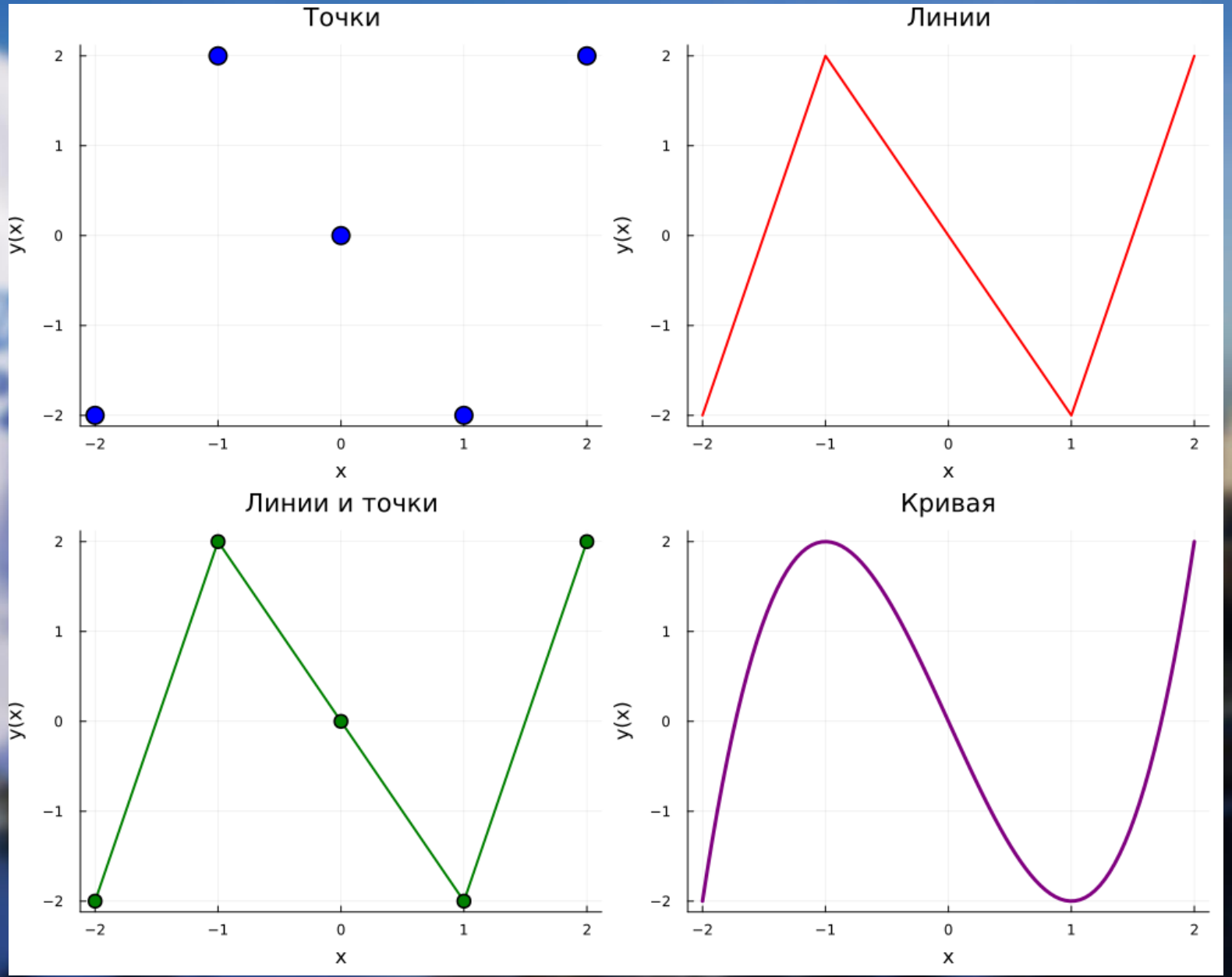
```
using Plots

function plot_custom_function()
    # Функция y(x) = πx²ln(x)
    y_func(x) = π * x^2 * log(x)

    # Область определения (x > 0 из-за логарифма)
    x = range(0.1, 5, length=100)
    y_vals = y_func.(x)

    plot(x, y_vals,
        title="График функции y(x) = πx²ln(x)",
        xlabel="Значения x",
        ylabel="Значения y",
        color=:red,
        linewidth=3,
        grid=true,
        legend=false,
        xticks=0:0.5:5,
        yticks=-10:5:30,
        framestyle=:box)
end

plot_custom_function()
```



Задание 4 ¶

```
using Plots

function plot_function_family()
    # Вектор x
    x_points = [-2, -1, 0, 1, 2]
    # Функция y(x) = x^3 - 3x
    y_func(x) = x^3 - 3x
    y_vals = y_func.(x_points)

    # 4 подграфика
    p1 = scatter(x_points, y_vals,
        title="Точки",
        xlabel="x", ylabel="y(x)",
        color=:blue, markersize=8,
        label="")

    p2 = plot(x_points, y_vals,
        title="Линии",
        xlabel="x", ylabel="y(x)",
        color=:red, linewidth=2,
        label="")

    p3 = plot(x_points, y_vals,
        title="Линии и точки",
        xlabel="x", ylabel="y(x)",
        color=:green, linewidth=2,
        marker=:circle, markersize=6,
        label="")

    # Интерполируем для гладкой кривой
    x_smooth = range(-2, 2, length=100)
    y_smooth = y_func.(x_smooth)

    p4 = plot(x_smooth, y_smooth,
        title="Кривая",
        xlabel="x", ylabel="y(x)",
        color=:purple, linewidth=3,
        label="")

    # Объединяем все графики
    plot(p1, p2, p3, p4, layout=(2,2), size=(1000, 800))

    # Сохраняем с вашей фамилией
    savefig("figure_ADABOR.png")  # Замените ADABOR на вашу фамилию
end

plot_function_family()
```

```
"C:\\Users\\KRIS\\Desktop\\Pere\\figure_ADABOR.png"
```



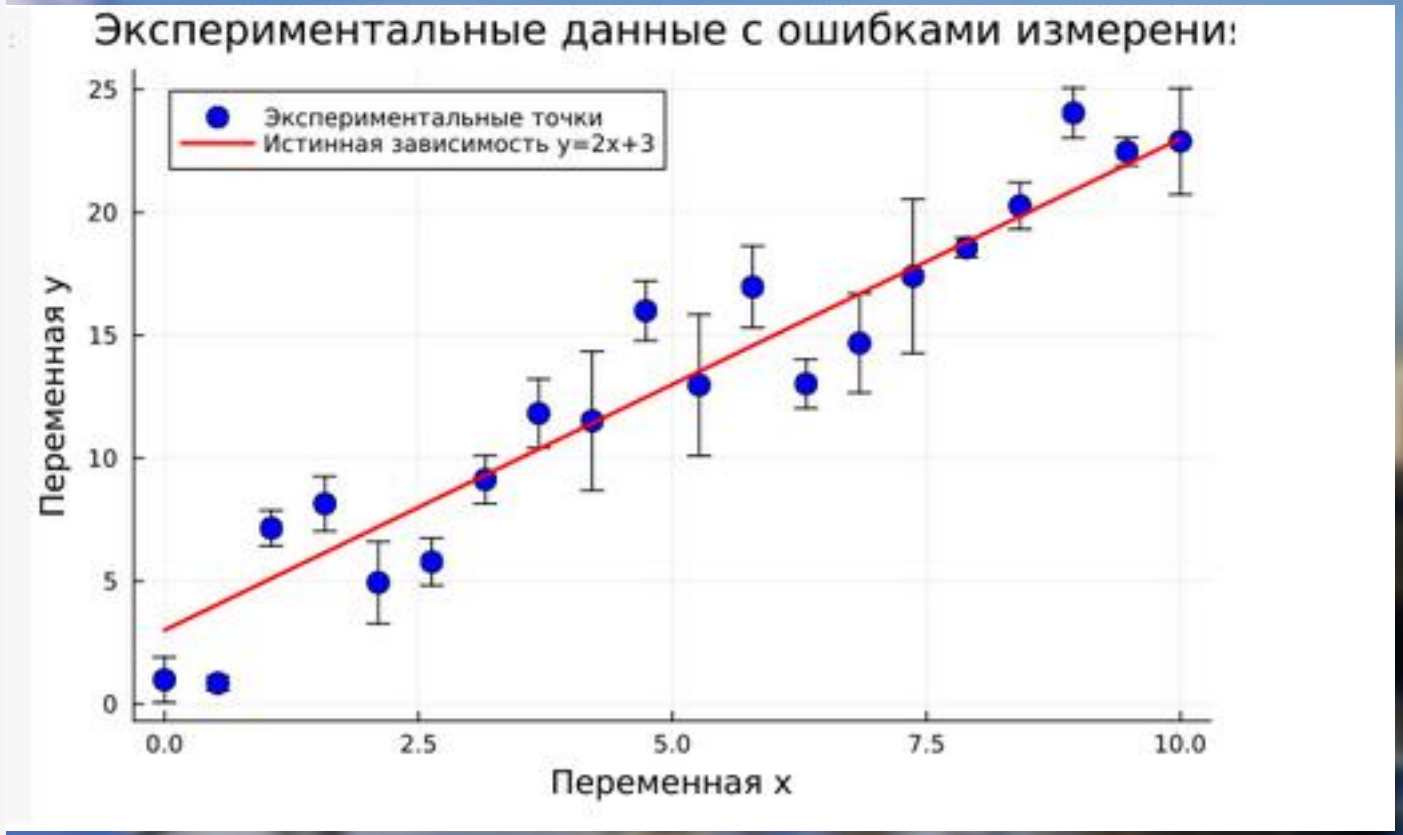
Задание 5

```
using Plots

function plot_two_functions_comparison()
    # Вектор x
    x = range(3, 6, step=0.1)

    # Функции
    y1(x) = π * x
    y2(x) = exp(x) * cos(x)

    y1_vals = y1.(x)
    y2_vals = y2.(x)

    # График 1: Обе функции на одном рисунке
    p1 = plot(x, y1_vals,
        label="y₁(x) = πx",
        color=:blue,
        linewidth=2,
        xlabel="x",
        ylabel="y",
        title="Две функции на одном графике",
        legend=:topright,
        grid=true)

    plot!(x, y2_vals,
        label="y₂(x) = e^cos(x)",
        color=:red,
        linewidth=2)

    # График 2: С двумя осями ординат
    p2 = plot(x, y1_vals,
        label="y₁(x) = πx",
        color=:blue,
        linewidth=2,
        xlabel="x",
        ylabel="y₁",
        title="Две функции с разными осями Y",
        legend=:topright,
        grid=true)

    plot!(twinx(), x, y2_vals,
        label="y₂(x) = e^cos(x)",
        color=:red,
        linewidth=2,
        ylabel="y₂",
        legend=:bottomright)

    plot(p1, p2, layout=(2,1), size=(800, 600))

    println("Недостатки первого графика:")
    println("1. Масштабы функций сильно отличаются")
    println("2. Функция y₂(x) плохо видна из-за большого диапазона значений")
    println("3. Трудно сравнивать поведение функций")
end

plot_two_functions_comparison()
```

```
Недостатки первого графика:
1. Масштабы функций сильно отличаются
2. Функция y₂(x) плохо видна из-за большого диапазона значений
```

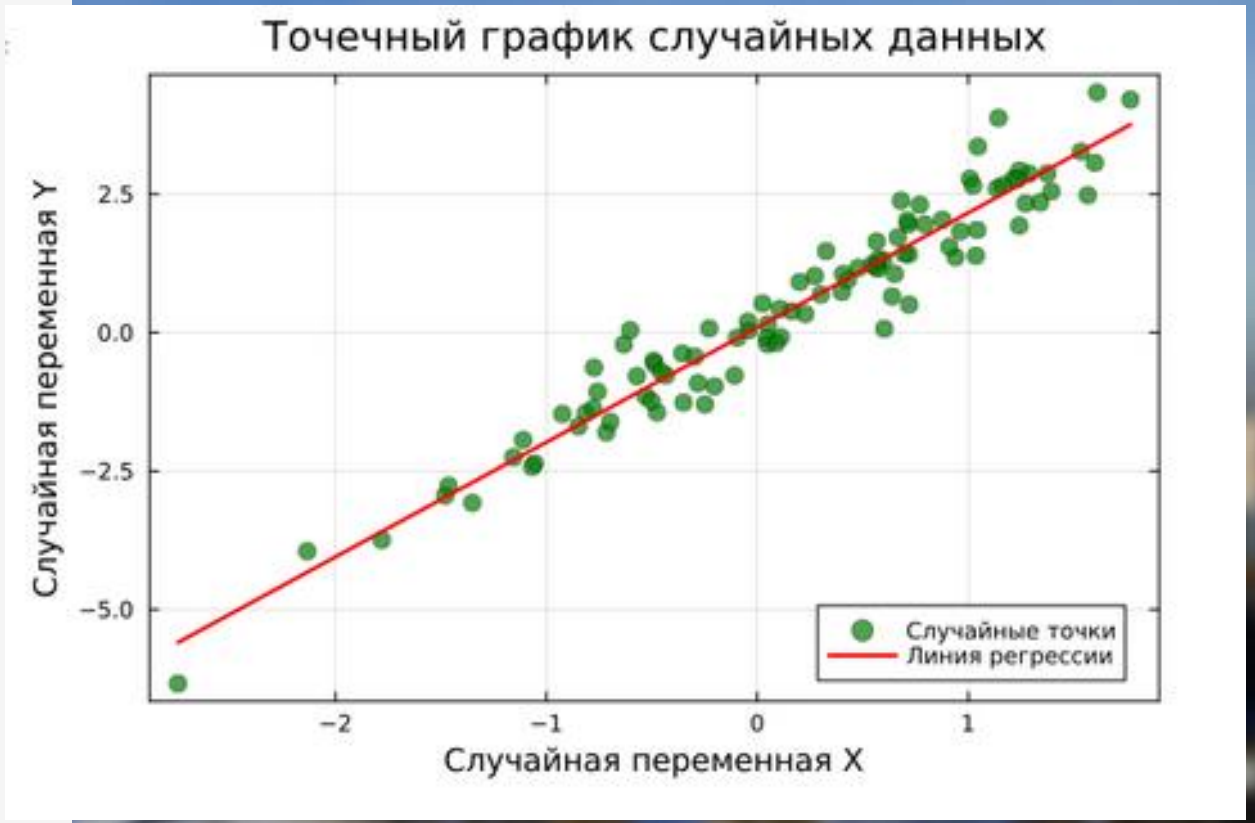
Задание 6

```julia
using Plots

function plot_experimental_data()
    # Генерируем экспериментальные данные с ошибкой измерения
    x_exp = range(0, 10, length=20)
    # Истинная функция: y = 2x + 3 + шум
    y_true = 2 .* x_exp .+ 3
    # Добавляем случайную ошибку измерения
    error = randn(20) .* 2
    y_exp = y_true + error
    # Ошибки для каждого измерения
    errors = abs.(randn(20) .* 1.5)

    scatter(x_exp, y_exp,
        title="Экспериментальные данные с ошибками измерения",
        xlabel="Переменная x",
        ylabel="Переменная y",
        label="Экспериментальные точки",
        yerror=errors,
        markersize=6,
        color=:blue,
        legend=:topleft)

    # Добавляем линию тренда
    plot!(x_exp, y_true,
        label="Истинная зависимость y=2x+3",
        color=:red,
        linewidth=2)
end

plot_experimental_data()
```



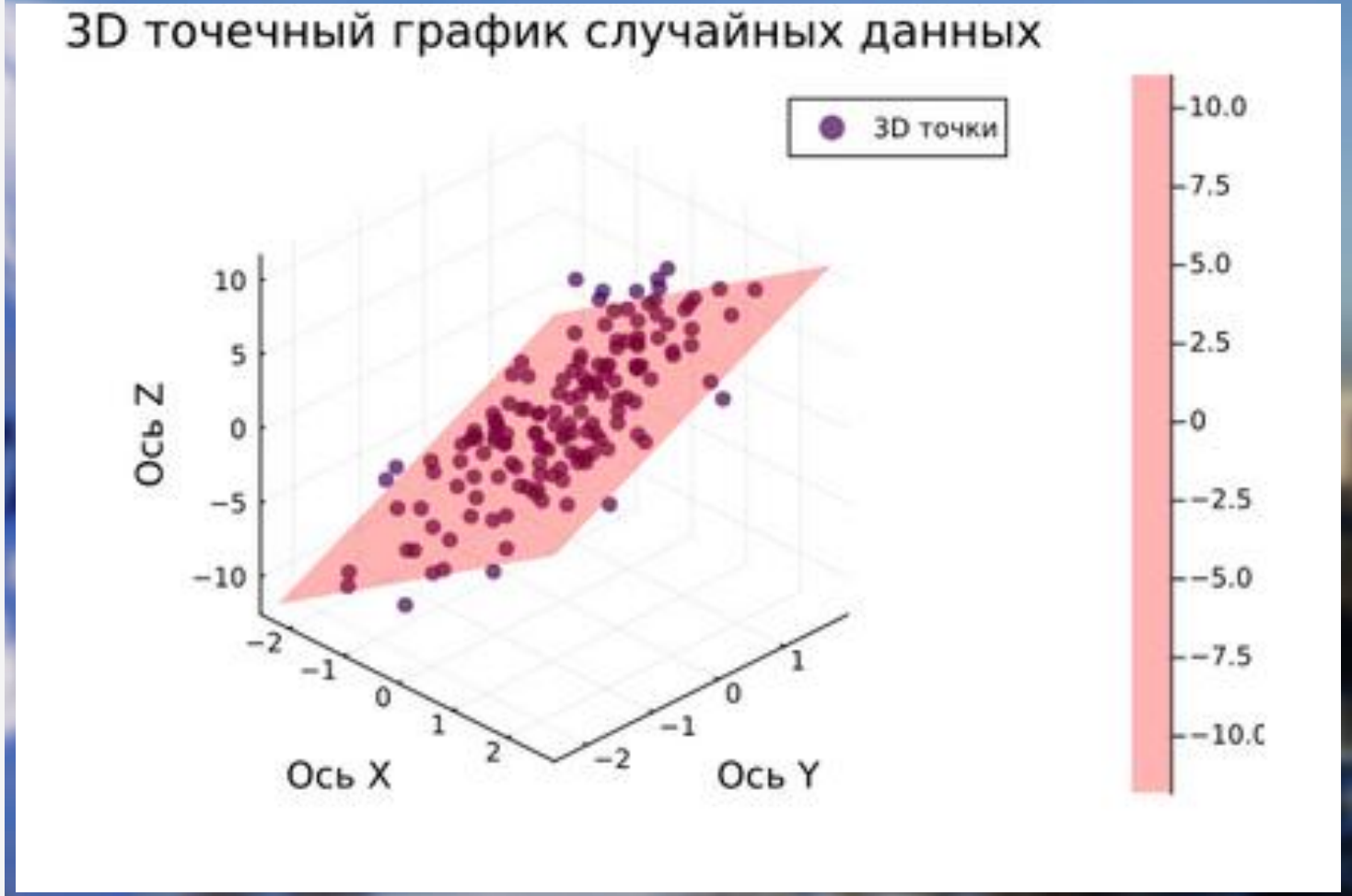
Задание 7 ¶

```
using Plots, Statistics

function plot_random_scatter()
    # Генерируем случайные данные
    n = 100
    x_rand = randn(n)
    y_rand = 2 .* x_rand .+ randn(n) .* 0.5

    scatter(x_rand, y_rand,
        title="Точечный график случайных данных",
        xlabel="Случайная переменная X",
        ylabel="Случайная переменная Y",
        label="Случайные точки",
        markersize=5,
        color=:green,
        markerstrokewidth=0.5,
        alpha=0.7,
        legend=:bottomright,
        grid=true,
        framestyle=:box)

    # Добавляем линию регрессии
    slope = cor(x_rand, y_rand) * std(y_rand) / std(x_rand)
    intercept = mean(y_rand) - slope * mean(x_rand)
    x_line = [minimum(x_rand), maximum(x_rand)]
    y_line = slope .* x_line .+ intercept

    plot!(x_line, y_line,
        label="Линия регрессии",
        color=:red,
        linewidth=2)
end

plot_random_scatter()
```



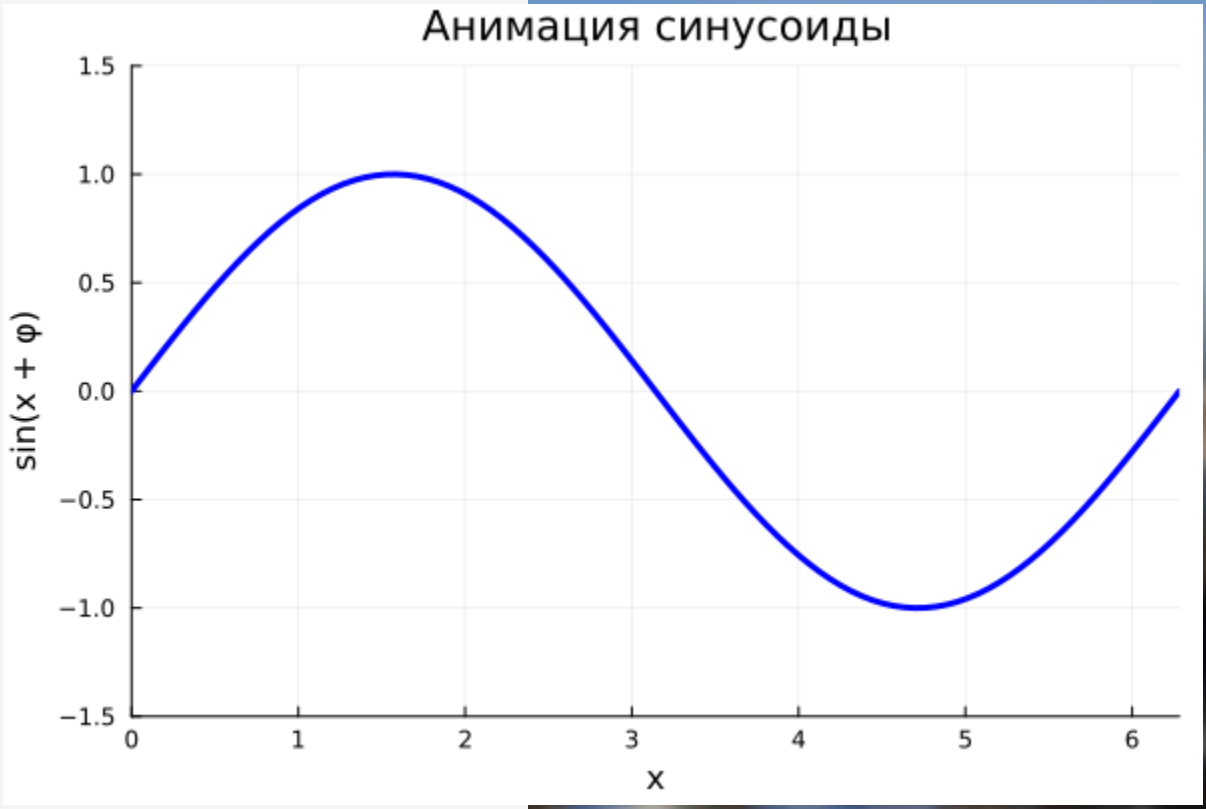
Задание 8

```
using Plots, Statistics, LinearAlgebra

function plot_3d_random_scatter()
    # Генерируем 3D случайные данные
    n = 150
    x_3d = randn(n)
    y_3d = randn(n)
    z_3d = 2 .* x_3d .+ 3 .* y_3d .+ randn(n) .* 0.5

    scatter(x_3d, y_3d, z_3d,
        title="3D точечный график случайных данных",
        xlabel="Ось X",
        ylabel="Ось Y",
        zlabel="Ось Z",
        label="3D точки",
        markersize=4,
        color=:viridis,
        markerstrokewidth=0,
        alpha=0.7,
        legend=:topright,
        camera=(45, 30))

    # Добавляем плоскость регрессии
    x_plane = range(minimum(x_3d), maximum(x_3d), length=10)
    y_plane = range(minimum(y_3d), maximum(y_3d), length=10)
    A = [x_3d y_3d ones(n)] \ z_3d
    z_plane = [A[1]*x + A[2]*y + A[3] for x in x_plane, y in y_plane]

    surface!(x_plane, y_plane, z_plane,
        alpha=0.3,
        color=:red,
        label="Плоскость регрессии")
end

plot_3d_random_scatter()
```



Задание 9

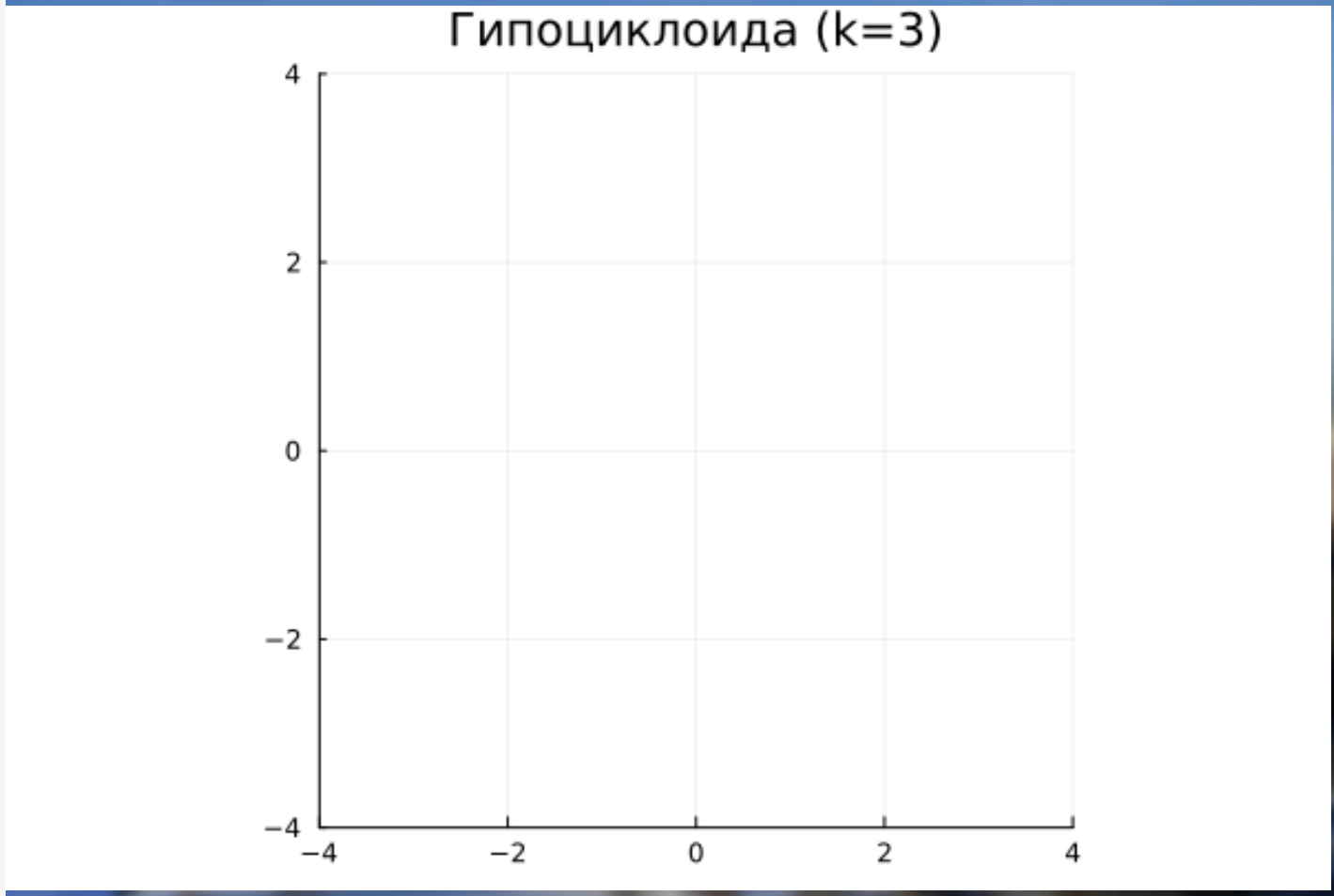
```
using Plots

function create_sine_animation()
    # Анимация синусоиды
    anim = @animate for phase in range(0, 4π, length=50)
        x_anim = range(0, 2π, length=100)
        y_anim = sin.(x_anim .+ phase)

        plot(x_anim, y_anim,
            title="Анимация синусоиды",
            xlabel="x",
            ylabel="sin(x + ϕ)",
            xlim=(0, 2π),
            ylim=(-1.5, 1.5),
            color=:blue,
            linewidth=3,
            legend=false)
    end

    # Сохраняем анимацию
    gif(anim, "sine_animation.gif", fps=10)
    println("Анимация сохранена в файл sine_animation.gif")
end

create_sine_animation()
```




```
Анимация сохранена в файл sine_animation.gif
[ Info: Saved animation to C:\Users\KRIS\Desktop\Pere\sine_animation.gif
```

Задание 10

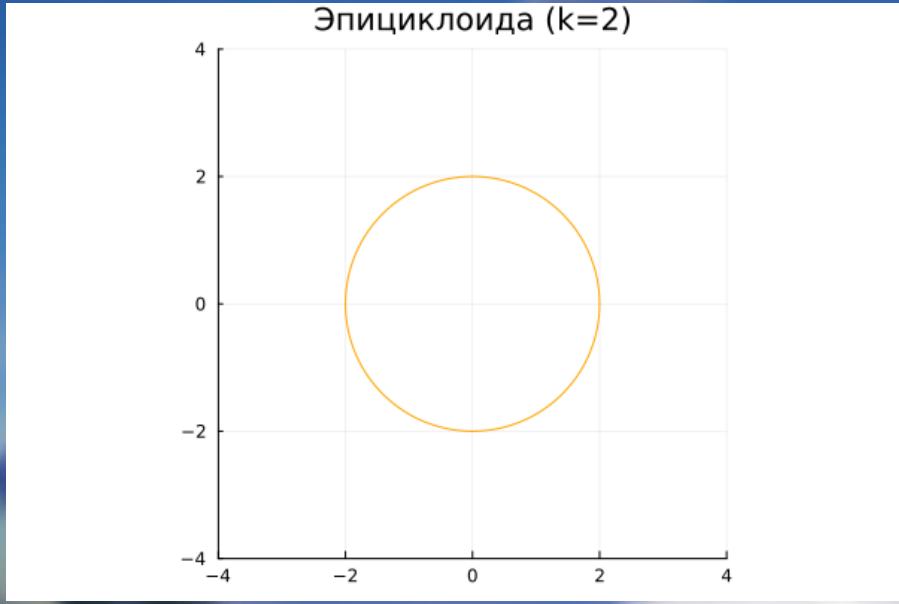
```julia
using Plots

function create_hypocycloid_animation()
    # Простая гипоциклоида с k=3
    r = 1
    k = 3
    θ = range(0, 2π, length=100)

    anim = @animate for i in 1:length(θ)
        t = θ[1:i]
        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=3)",
            xlim=(-4, 4),
            ylim=(-4, 4),
            aspect_ratio=1,
            color=:red,
            linewidth=2,
            legend=false)
    end

    gif(anim, "hypocycloid_simple.gif", fps=15)
    println("Анимация гипоциклоиды сохранена")
end

create_hypocycloid_animation()
```

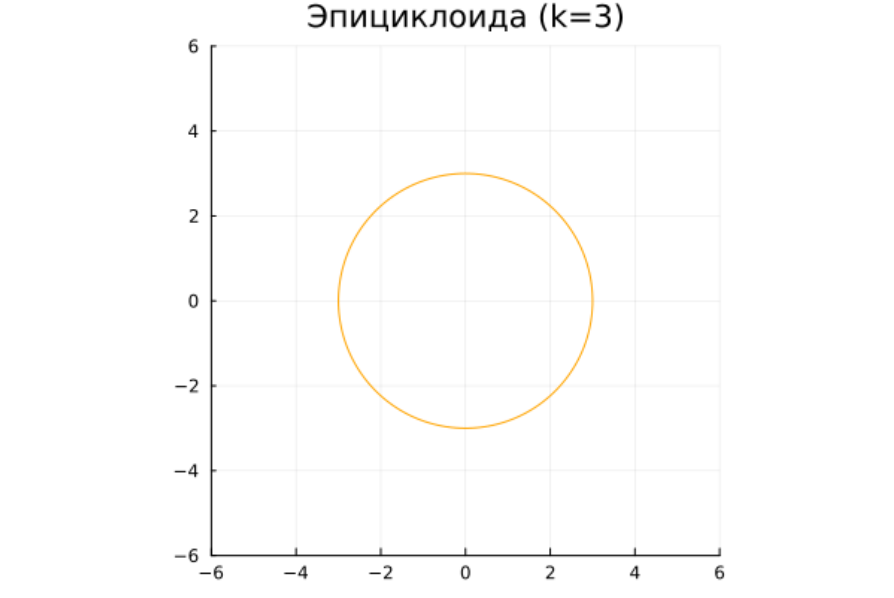


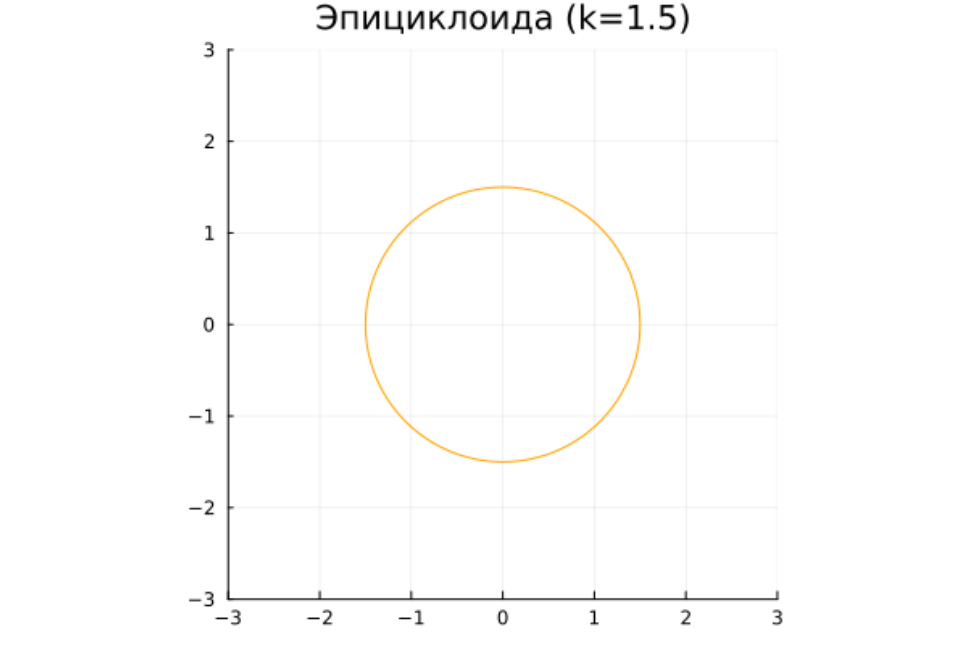
Задание 11

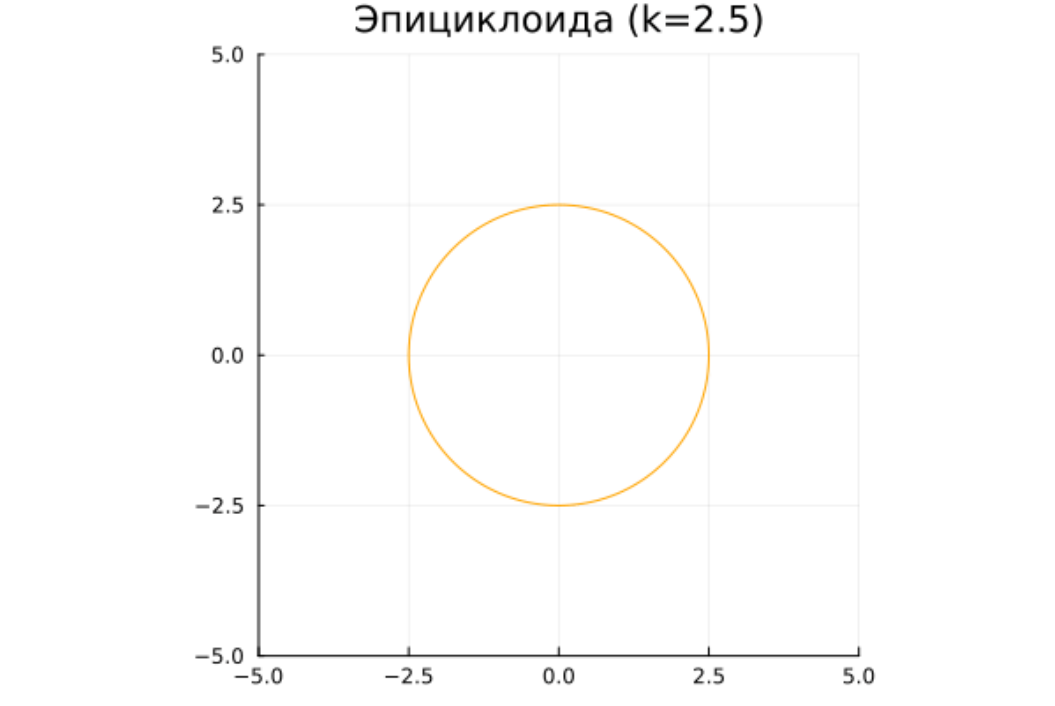
```
using Plots

function create_epicycloid_animations()
    # Функция для эпициклоиды
    # x = r(k+1)cos(t) - rcos((k+1)t)
    # y = r(k+1)sin(t) - rsin((k+1)t)

    # Целые значения k
    k_epi_int = [2, 3]

    # Рациональные значения k
    k_epi_rat = [3/2, 5/2]

    # Создаем анимации для целых k
    for k in k_epi_int
        r = 1
        θ = range(0, 2π, length=200)

        anim = @animate for i in 1:length(θ)
            t = θ[1:i]
            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="Эпициклоида (k=$k)",
                xlim=(-2k, 2k),
                ylim=(-2k, 2k),
                aspect_ratio=1,
                color=:purple,
                linewidth=2,
                label="")

            # Большая окружность
            X_big = r*k*cos.(θ)
            Y_big = r*k*sin.(θ)
            plot!(X_big, Y_big, color=:orange, linewidth=1, label="")
        end

        gif(anim, "epicycloid_int_$(k).gif", fps=20)
    end

    # Создаем анимации для рациональных k
    for k_rat in k_epi_rat
        r = 1
        θ = range(0, 4π, length=400)

        anim = @animate for i in 1:length(θ)
            t = θ[1:i]
            x = r*(k_rat+1)*cos.(t) - r*cos.((k_rat+1)*t)
            y = r*(k_rat+1)*sin.(t) - r*sin.((k_rat+1)*t)

            plot(x, y,
                title="Эпициклоида (k=$k_rat)",
                xlim=(-2k_rat, 2k_rat),
                ylim=(-2k_rat, 2k_rat),
                aspect_ratio=1,
                color=:brown,
                linewidth=2,
                label="")

            X_big = r*k_rat*cos.(θ)
            Y_big = r*k_rat*sin.(θ)
            plot!(X_big, Y_big, color=:orange, linewidth=1, label="")
        end

        gif(anim, "epicycloid_rat_$(k_rat).gif", fps=20)
    end

    println("Анимации эпициклоид сохранены в файлы")
end

create_epicycloid_animations()
```

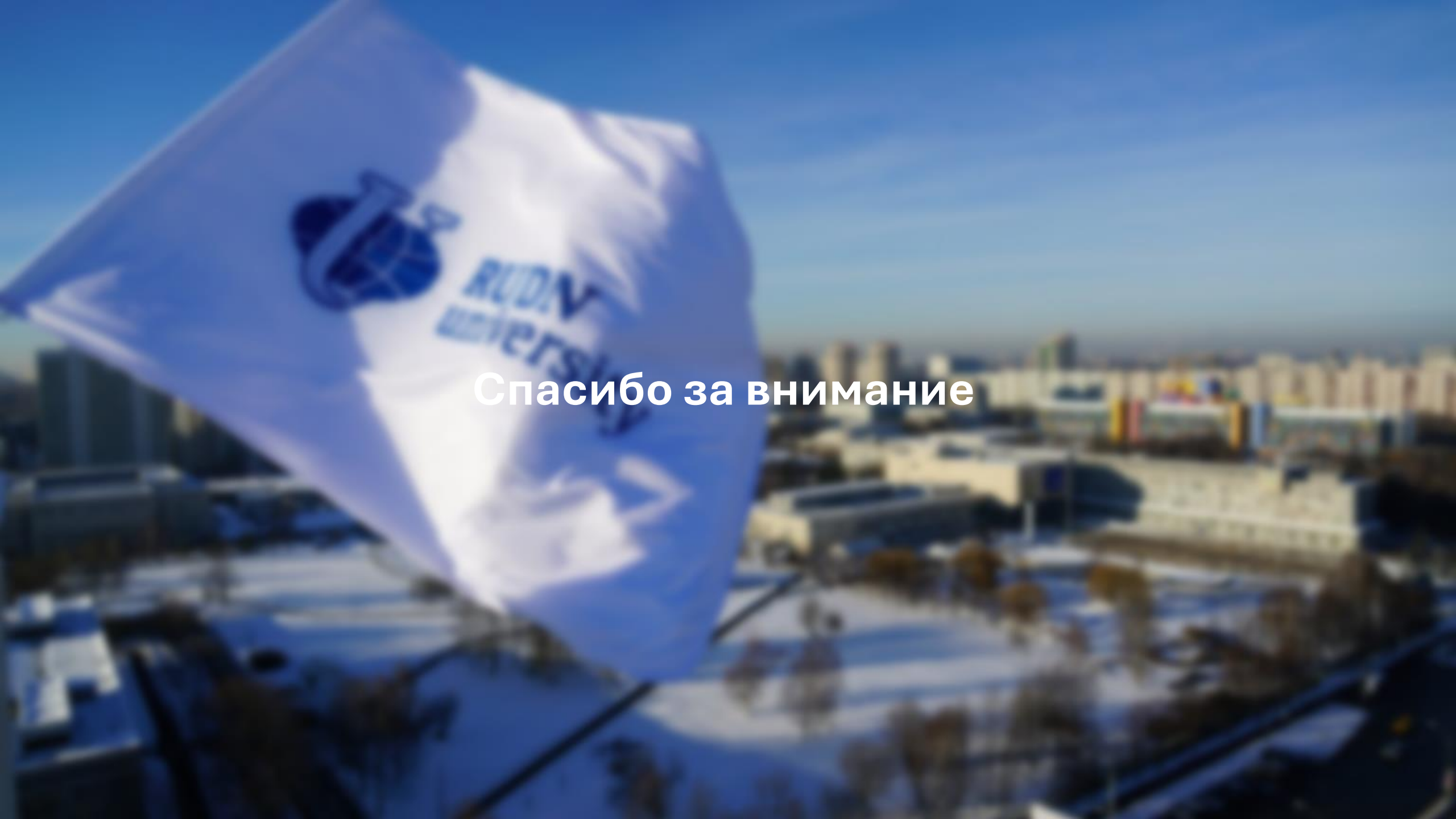
```
[ Info: Saved animation to C:\Users\KRIS\Desktop\Pere\epicycloid_int_2.gif
[ Info: Saved animation to C:\Users\KRIS\Desktop\Pere\epicycloid_int_3.gif
[ Info: Saved animation to C:\Users\KRIS\Desktop\Pere\epicycloid_rat_1.5.gif
Анимации эпициклоид сохранены в файлы
[ Info: Saved animation to C:\Users\KRIS\Desktop\Pere\epicycloid_rat_2.5.gif
```


Эпициклоида (k=2)
4
2
0
−2
−4
−4
−2
0
2
4


Эпициклоида (k=3)
6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6


Эпициклоида (k=1.5)
3
2
1
0
−1
−2
−3
−3
−2
−1
0
1
2
3


Эпициклоида (k=2.5)
5.0
2.5
0.0
−2.5
−5.0
−5.0
−2.5
0.0
2.5
5.0


Спасибо за внимание