Отчет по лабораторной работе №5

Построение графиков

Легиньких Галина Андреевна

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

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

2 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).

3 Выполнение лабораторной работы

1. Для начала я повторила примеры и дробавила, где это было необходимо, обозначения осей координат, легенду с названиями траекторий, названия графиков и т.п. Больше я это нигде прописывать не буду.

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.

2. Далее рассмотрела построение графика функции f(x) = (3x2 + 6x - 9)exp(-0,3x) разными способами. Фактически для построения графика функции требуется иметь массив соответствующих значений x и y.

(рис. 3.1) (рис. 3.2) (рис. 3.3)

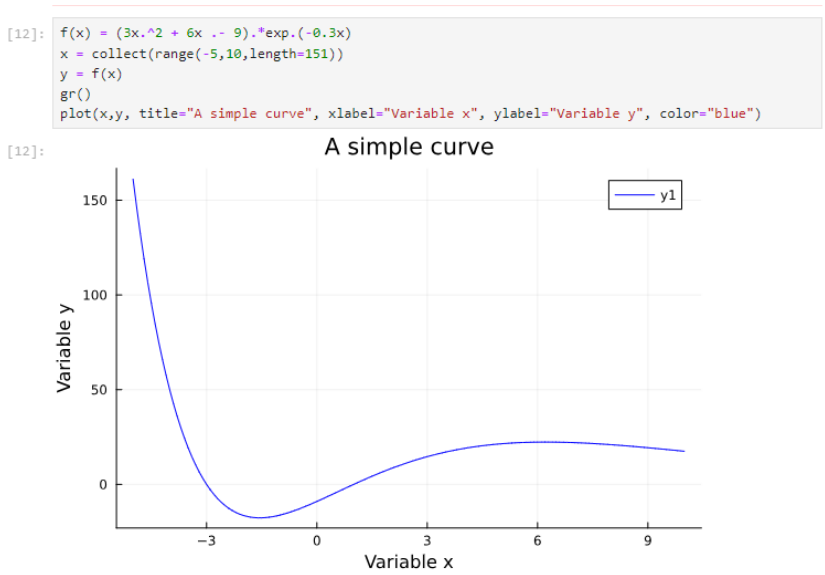

Рис. 3.1: Способ 1

```
[14]: f(x) = (3x.^2 + 6x .- 9).*exp.(-0.3x)
x = collect(range(-5,10,length=151))
y = f(x)
pyplot()
plot(x,y, title="Простая кривая", xlabel="Переменная x", ylabel="Переменная y", color="blue")
```

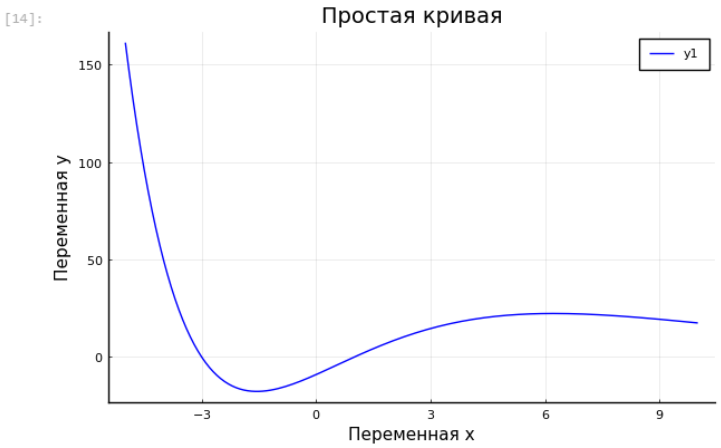

Рис. 3.2: Способ 2

```
[18]: f(x) = (3x.^2 + 6x .- 9).*exp.(-0.3x)
x = collect(range(-5,10,length=151))
y = f(x)
plotly()
plot(x,y, title="A simple curve", xlabel="Variable x", ylabel="Variable y", color="blue")
```

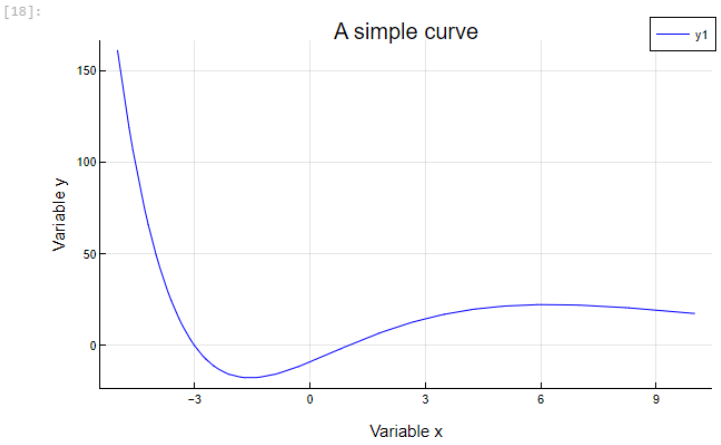

Рис. 3.3: Способ 3

3. Далее на примере графика функции sin(x) и графика разложения этой функции в ряд Тейлора рассмотрела дополнительные возможности пакетов для работы с графикой. В примерах рассматривается поэтапное улучшение графика. Добавлю скриншот конечного результата. (рис. 3.4) (рис. 3.5)

[32]:
```
pyplot()
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(sin_theor)
plot!(sin_taylor)
plot(sin_taylor,
# подпись в легенде, цвет и тип линии:
label = "sin(x), разложение в ряд Тейлора",
line=(:blue, 0.3, 6, :solid),
# размер графика:
size=(800, 500),
# параметры отображения значений по осям
xticks = (-5:0.5:5),
yticks = (-1:0.1:1),
xtickfont = font(12, "Times New Roman"),
ytickfont = font(12, "Times New Roman"),
    # подписи по осям:
ylabel = "y",
xlabel = "x",
# название графика:
title = "Разложение в ряд Тейлора",
# поворот значений, заданный по оси x:
xrotation = rad2deg(pi/4),
# заливка области графика цветом:
fillrange = 0,
fillalpha = 0.5,
fillcolor = :lightgoldenrod,
# задание цвета фона:
background_color = :ivory
)
plot!(
# функция sin_theor:
sin_theor,
# подпись в легенде, цвет и тип линии:
label = "sin(x), теоретическое значение",
line=(:black, 1.0, 2, :dash))
```

Рис. 3.4: Код для графика исходной функции и её разложения в ряд Тейлора с опциями

[32]:

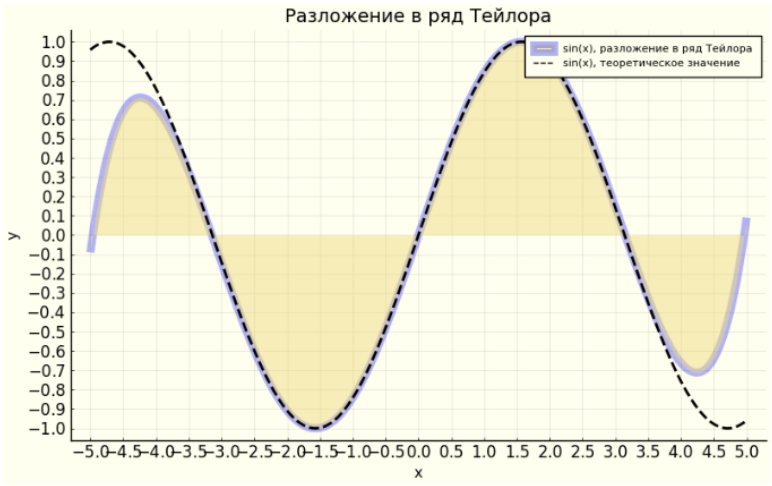

Рис. 3.5: Графики исходной функции и её разложения в ряд Тейлора с опциями

4. Как и построении обычного графика для точечного графика необходимо задать массив значений x, посчитать или задать значения ⊠, задать опции по-

строения графика. Примеры 2-мерного и 3-мерного графиков. (рис. 3.6) (рис. 3.7)

[42]:
```
# 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, seriestype = :scatter,
    title = "Точечный график",
    ylabel = "y", xlabel = "x",
    label = "Случайная десятка значений")
```

[42]:

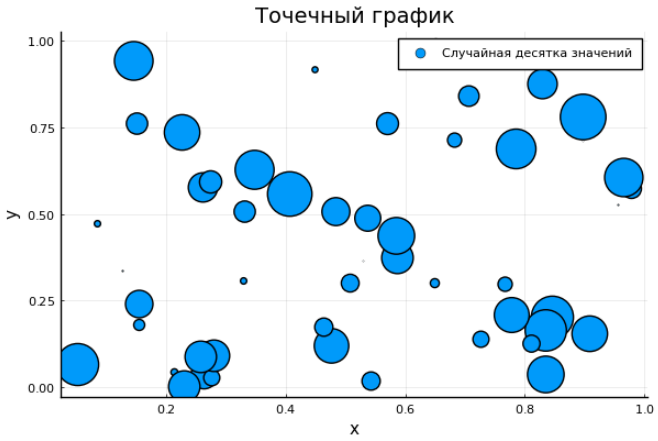

Рис. 3.6: График пятидесяти случайных значений на плоскости с различными опциями отображения

[48]:
```
# 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, seriestype = :scatter,
    title = "Точечный график",
    ylabel = "y", xlabel = "x", zlabel = "z",
    label = "Случайные 50 значений")
```

[48]:

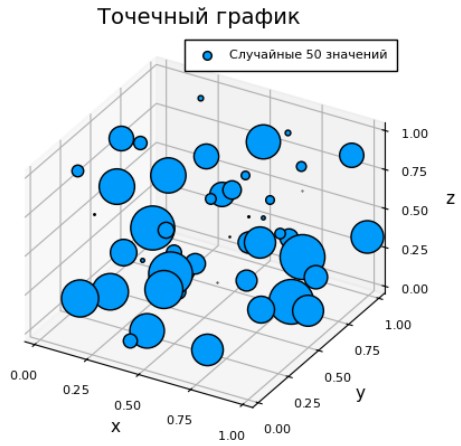

Рис. 3.7: График пятидесяти случайных значений в пространстве с различными опциями отображения

5. Аппроксимация — научный метод, состоящий в замене объектов их более простыми аналогами, сходными по своим свойствам. (рис. 3.8)

```
[80]: # 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=.8,
ylabel = "y", xlabel = "x", title = "Аппроксимация данных", label = "Экспоненциальная функция")
# 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:
f_approx = 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_approx, linewidth=3, color=:red, label = "Апроксимация")
```

[80]:

Рис. 3.8: Пример аппроксимации исходной функции полиномом 5-й степени

6. Иногда требуется на один график вывести несколько траекторий с существенными отличиями в значениях по оси ординат. (рис. 3.9)

```
[106]: p1 = plot(randn(100),
       ylabel="y1",
       title = "Пример",
       leg=:topright,
       grid = :off,
       size=(600, 400)
       )
       plot!(twinx(), randn(100)*10,
       c=:red,
       ylabel="y2",
       leg=:bottomright,
       grid = :off,
       box = :on
       )

       display(p1)
```

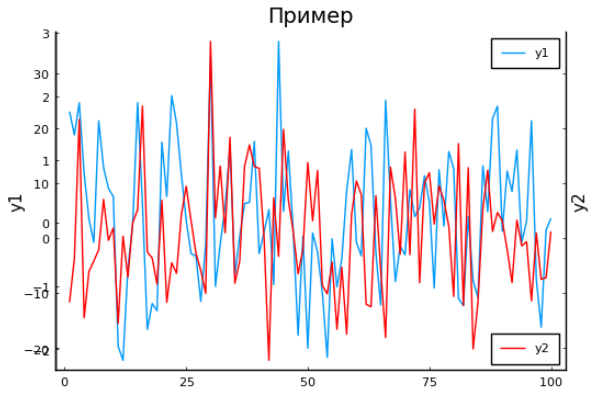

Рис. 3.9: Пример двух траекторий на одном графике с двумя осями ординат

7. Повторила пример построения графика функции в полярных координатах. (рис. 3.10)

```
[112]: # 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),
title = "Полярные координаты",
label = "y1"
)
```

Рис. 3.10: График функции, заданной в полярных координатах

8. Повторила пример построения графика параметрически заданной кривой на плоскости. (рис. 3.11)

```
[126]: # параметрическое уравнение:
x_t(t) = sin(t)
y_t(t) = sin(2t)
# построение графика:
plot(x_t, y_t, 0, 2π, fill=(0,:orange), title = "кривая на плоскости",
    ylabel = "y", xlabel = "x", label = "График")
```

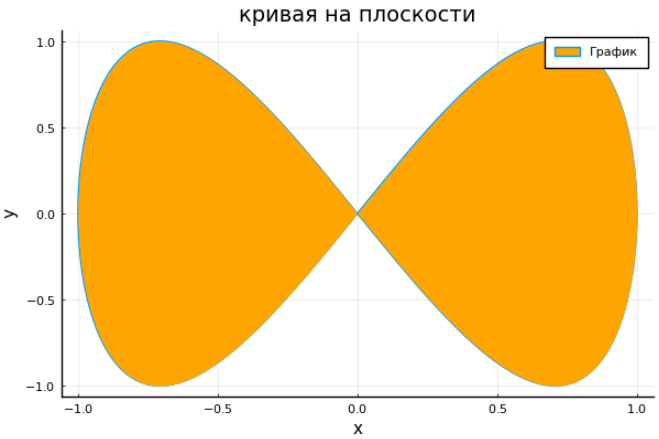

Рис. 3.11: Параметрический график кривой на плоскости

9. Приведём пример построения графика параметрически заданной кривой в

пространстве. (рис. 3.12)

```
[146]: # параметрическое уравнение
t = range(0, stop=10, length=1000)
x_e = cos.(t)
y_e = sin.(t)
z_e = sin.(5t)
# построение графика:
plot(x_e, y_e, z_e, title = "кривая в пространстве",
    ylabel = "y", xlabel = "x", zlabel = "z", label = "График")
```

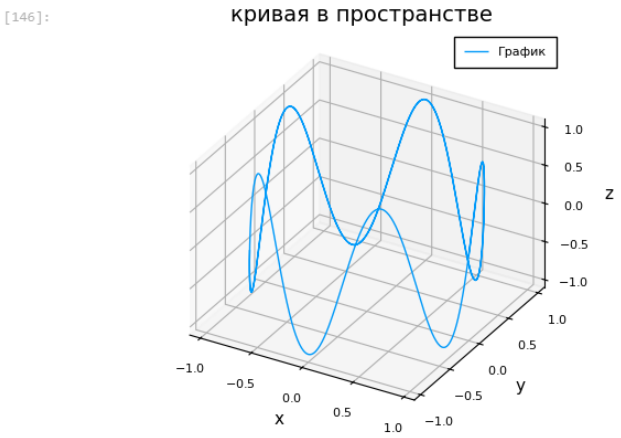

Рис. 3.12: Параметрический график кривой в пространстве

10. Перешла к графикам поверхности. (рис. 3.13) (рис. 3.14) (рис. 3.15) (рис. 3.16)

```
[152]: # построение графика поверхности:
f(x,y) = x^2 + y^2
x = -10:10
y = x
surface(x, y, f, title = "График поверхности",
    ylabel = "y", xlabel = "x", zlabel = "z")
```

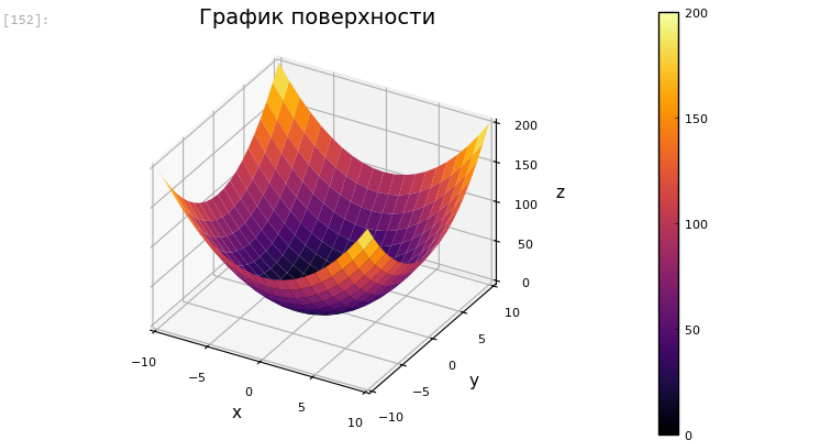

Рис. 3.13: График поверхности (использована функция surface())

```
[156]: # построение графика поверхности:
f(x,y) = x^2 + y^2
x = -10:10
y = x
plot(x, y, f,
linetype=:wireframe,  title = "График поверхности",
    ylabel = "y", xlabel = "x", zlabel = "z"
)
```

[156]:

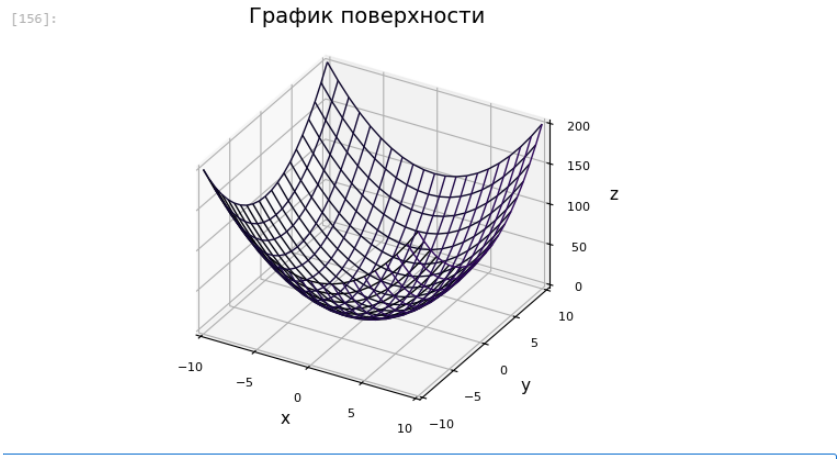

Рис. 3.14: График поверхности (использована функция plot())

```
[160]: f(x,y) = x^2 + y^2
x = -10:0.1:10
y = x
plot(x, y, f,
linetype = :surface, title = "График поверхности",
    ylabel = "y", xlabel = "x", zlabel = "z"
)
```

[160]:

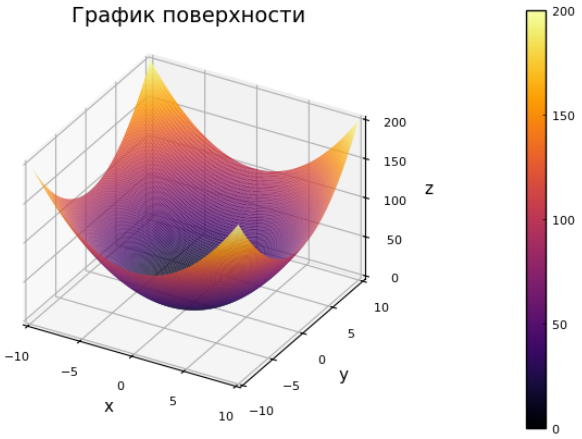

Рис. 3.15: Сглаженный график поверхности

```
[162]: x=range(-2,stop=2,length=100)
       y=range(sqrt(2),stop=2,length=100)
       f(x,y) = x*y-x-y+1
       plot(x,y,f,
       linetype = :surface,
       c=cgrad([:red,:blue]),
       camera=(-30,30),
           title = "График поверхности",
           ylabel = "y", xlabel = "x", zlabel = "z"
       )
```

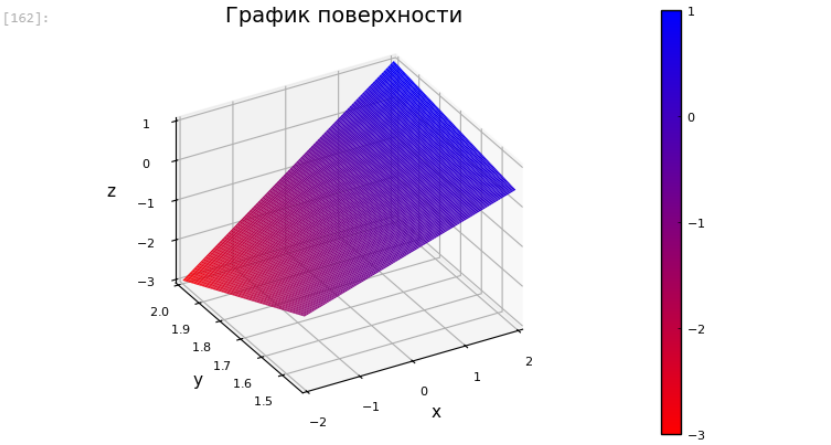

Рис. 3.16: График поверхности с изменённым углом зрения

11. 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. Рассмотрела поверхность, заданную функцией(рис. 3.17)

```
[170]: 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,  title = "График поверхности",
           ylabel = "y", xlabel = "x", zlabel = "z"
       )
```

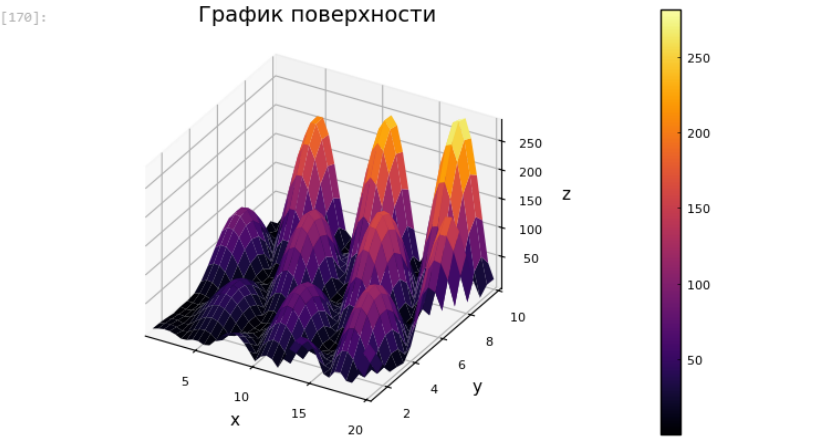

Рис. 3.17: График поверхности, заданной функцией

Линии уровня можно построить, используя проекцию значений исходной функции на плоскость. Можно дополнительно добавить заливку цветом. (рис. 3.18)

```
[180]: 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,
       )
       p = contour(x, y, g, title = "Линии уровня",
           ylabel = "y", xlabel = "x",fill=true)
       plot(p)
```

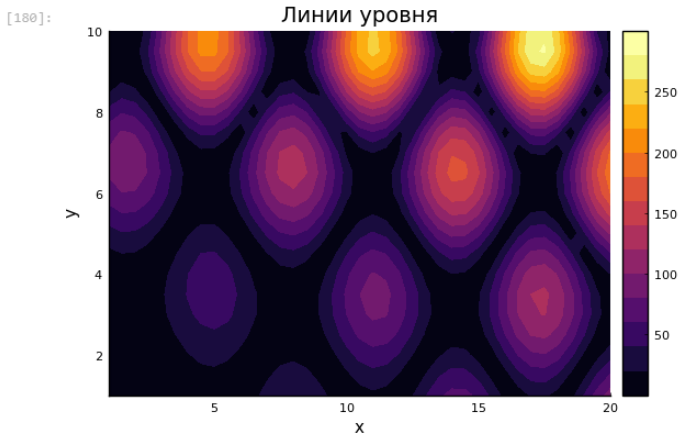

Рис. 3.18: Линии уровня с заполнением

12. Если каждой точке некоторой области пространства поставлен в соответствие вектор с началом в данной точке, то говорят, что в этой области задано векторное поле. (рис. 3.19) (рис. 3.20)

[186]:
```
# 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, title = "График функции",
    ylabel = "y", xlabel = "x", zlabel = "z"
)
```

[186]:

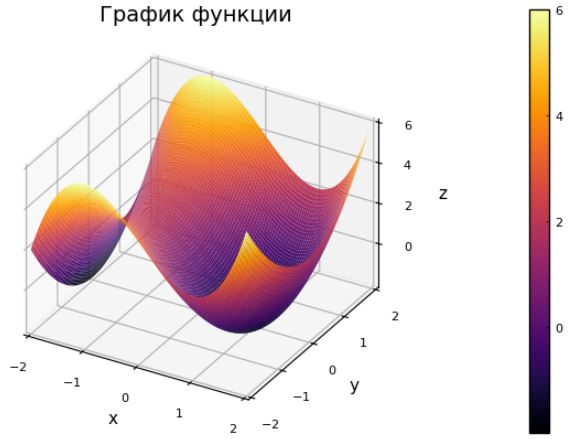

Рис. 3.19: График функции

[214]:
```
# 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
)
# построение линий уровня:
contour(X, Y, h)
# 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
# построение векторного поля:
quiver!(x, y, quiver=dh, c=:blue, title = "Линии уровня",
    ylabel = "y", xlabel = "x")
# коррекция области видимости графика:
xlims!(-2, 2)
ylims!(-2, 2)
```

[214]:

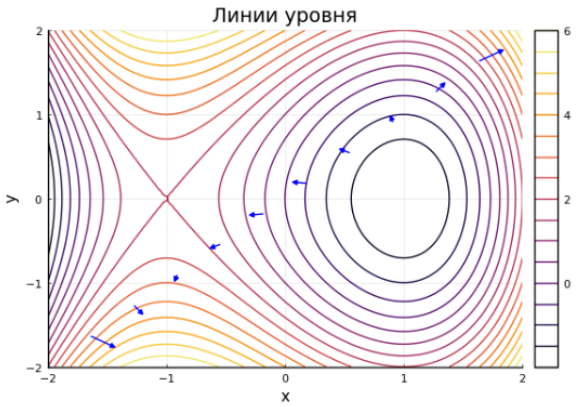

Рис. 3.20: Векторное поле функции

13. В Julia рекомендуется использовать gif-анимацию в pyplot(). (рис. 3.21) (рис. 3.22)

[222]:
```
pyplot()
# построение поверхности:
i = 0
X = Y = range(-5,stop=5,length=40)
surface(X, Y, (x,y) -> sin(x+10sin(i))+cos(y),  title = "График функции",
    ylabel = "y", xlabel = "x", zlabel = "z")
# 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),  title = "График функции",
    ylabel = "y", xlabel = "x", zlabel = "z")
end
```

```
[ Info: Saved animation to C:\Users\galin\tmp.gif
```

Рис. 3.21: Код для статичного графика и анимации

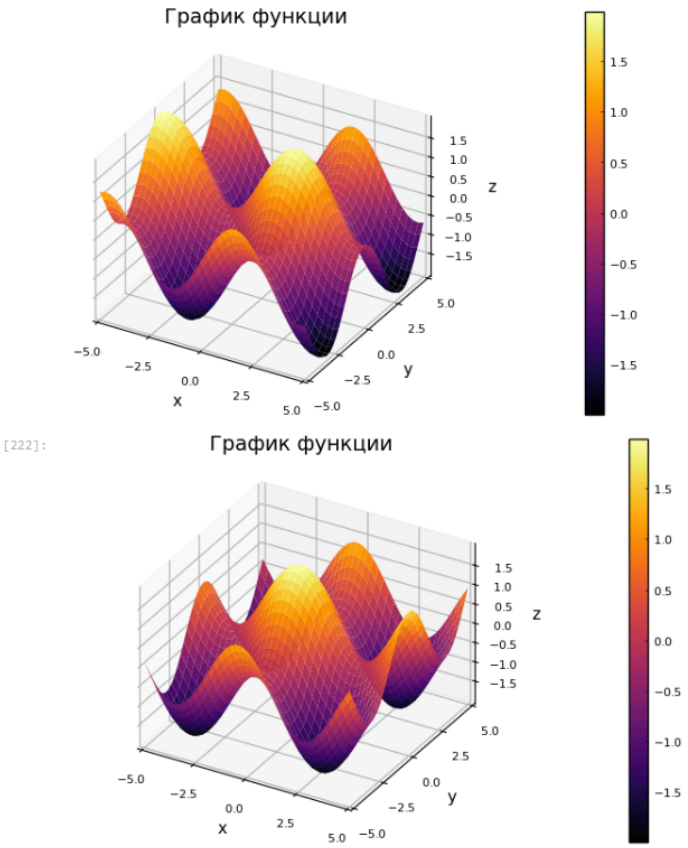

Рис. 3.22: Анимированный график поверхности

14. Гипоциклоида — плоская кривая, образуемая точкой окружности, катящейся по внутренней стороне другой окружности без скольжения. Дано поэтапное создания графика. Приложу конечный вариант. (рис. 3.23) (рис. 3.24)

[260]:
```
# радиус малой окружности
radius = 1
# коэффициент для построения большой окружности
k = 3
# число отсчётов
n = 100
# массив значений угла θ
θ = collect(0:2π/n:2π + 2π/n)
# массивы значений координат
X = radius * k * cos.(θ)
Y = radius * k * sin.(θ)
# задаём оси координат
plt = plot(xlim=(-4, 4), ylim=(-4, 4), c=:red, aspect_ratio=1,
    framestyle=:origin,  ylabel = "y", xlabel = "x")
# большая окружность
plot!(plt, X, Y, c=:blue, title = "Гипоциклоида", label = "Окружность")
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, label = "Часть гиперциклоицы")
# 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, label = "Малая окружность гиперциклоицы")

# 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)
```

Рис. 3.23: Код малая окружность гипоциклоиды с добавлением радиуса

[260]:

Рис. 3.24: Малая окружность гипоциклоиды с добавлением радиуса

В конце сделаем анимацию получившегося изображения. (рис. 3.25)

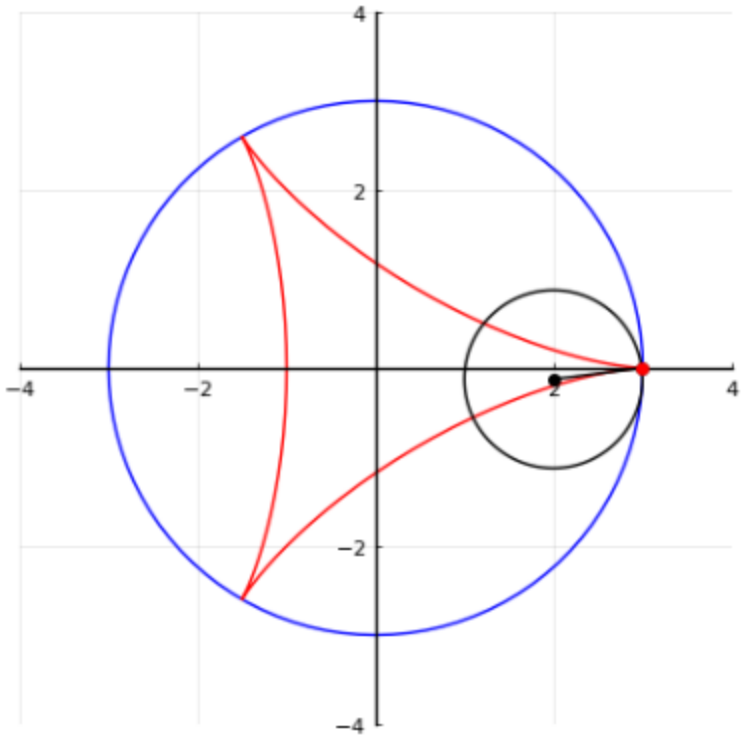

Рис. 3.25: Анимация гипоциклоиды

15. В исследованиях часто требуется изобразить графики погрешностей измерения. Подключила пакет Statistics. (рис. 3.26) (рис. 3.27) (рис. 3.28)

[276]:
```
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)
plot(y,
ylims = (-1,1),
err = errs, title = "График исходных значений с отклонениями",
    ylabel = "y", xlabel = "x", label = "Значения")
```

[276]:

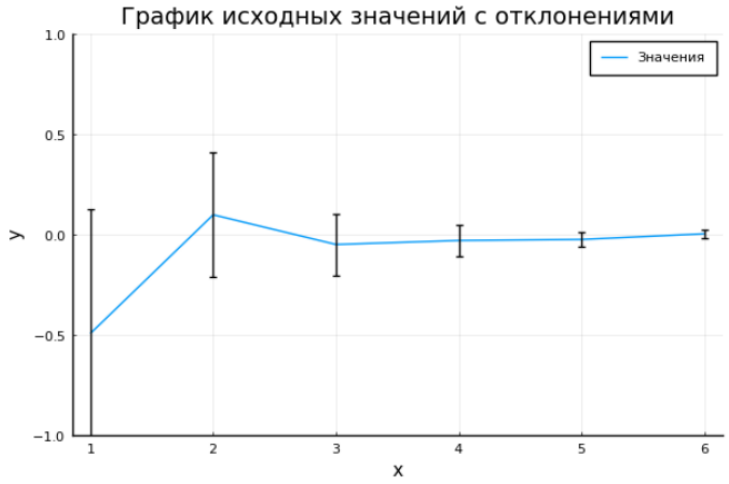

Рис. 3.26: График исходных значений с отклонениями

[278]:
```
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)
plot(y, 1:length(y),
xerr = errs,
marker = stroke(3,:orange), title = "Поворот графика",
    ylabel = "y", xlabel = "x", label = "Значения")
```

[278]:

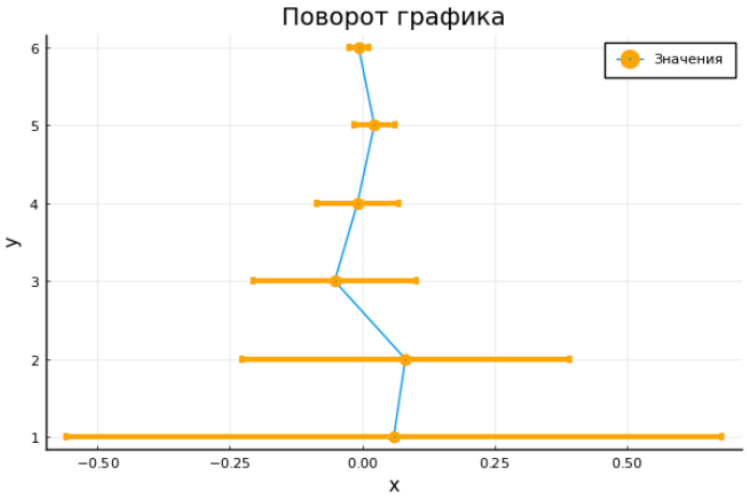

Рис. 3.27: Поворот графика

[282]:
```
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)
plot(y,
ribbon=errs,
fill=:cyan, title = "Заполнение цветом",
    ylabel = "y", xlabel = "x", label = "Значения")
```

[282]:

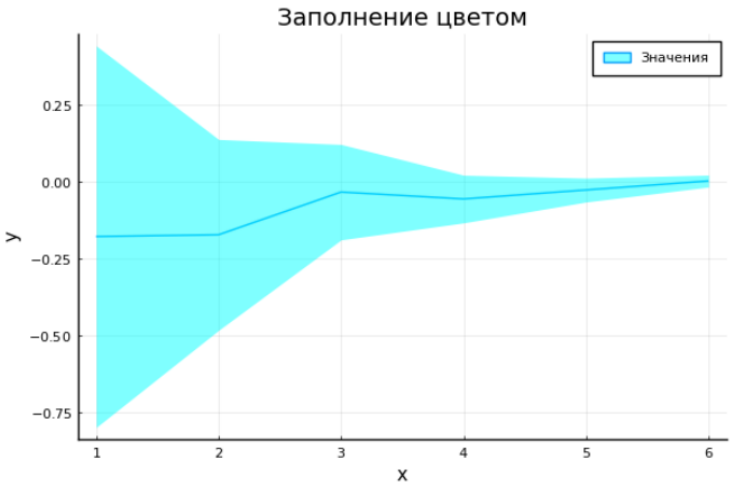

Рис. 3.28: Заполнение цветом

[284]:
```
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) = 1.96std(v) / sqrt(n)
xerr = map(f, x)
yerr = map(f, y)
x = map(mean, x)
y = map(mean, y)
plot(x, y,
xerr = xerr,
yerr = yerr,
marker = stroke(2, :orange), title = "График ошибок по двум осям",
    ylabel = "y", xlabel = "x", label = "Значения"
)
```

[284]:

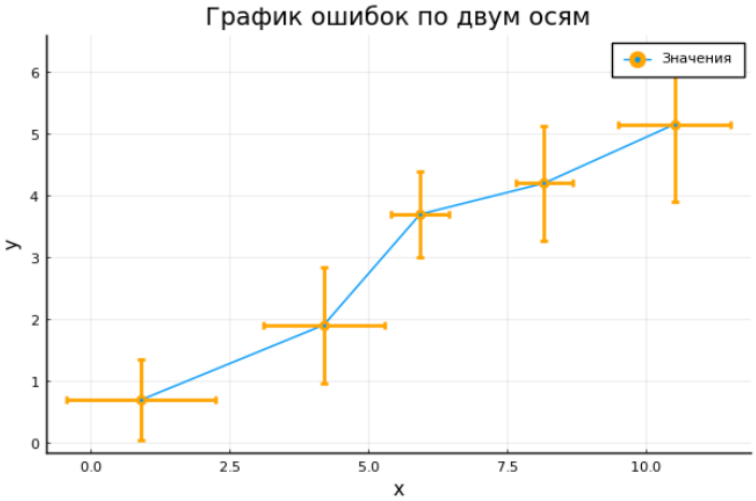

Рис. 3.29: График ошибок по двум осям

[286]:
```
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) = 1.96std(v) / sqrt(n)
xerr = map(f, x)
yerr = map(f, y)
x = map(mean, x)
y = map(mean, y)
plot(x, y,
xerr = (0.5xerr,2xerr),
yerr = (0.5yerr,2yerr),
marker = stroke(2, :orange), title = "График асимметричных ошибок по двум осям",
    ylabel = "y", xlabel = "x", label = "Значения"
)
```

[286]:

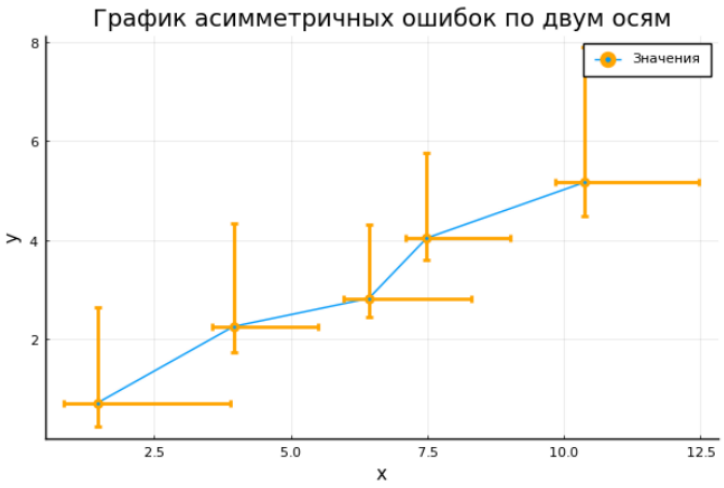

Рис. 3.30: График асимметричных ошибок по двум осям

16. Использование пакета Distributions. (рис. 3.31) (рис. 3.32) (рис. 3.33)

```
[290]: pyplot()
       ages = rand(15:55,1000)
       histogram(ages)
```

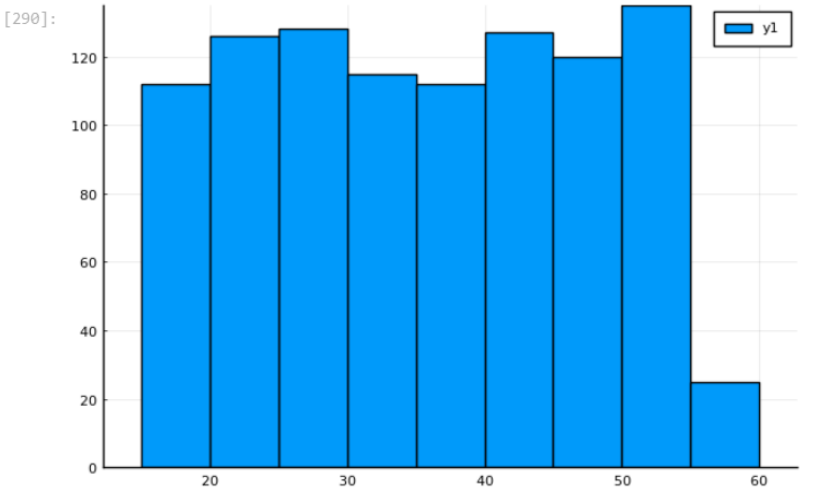

Рис. 3.31: Гистограмма, построенная по массиву случайных чисел

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

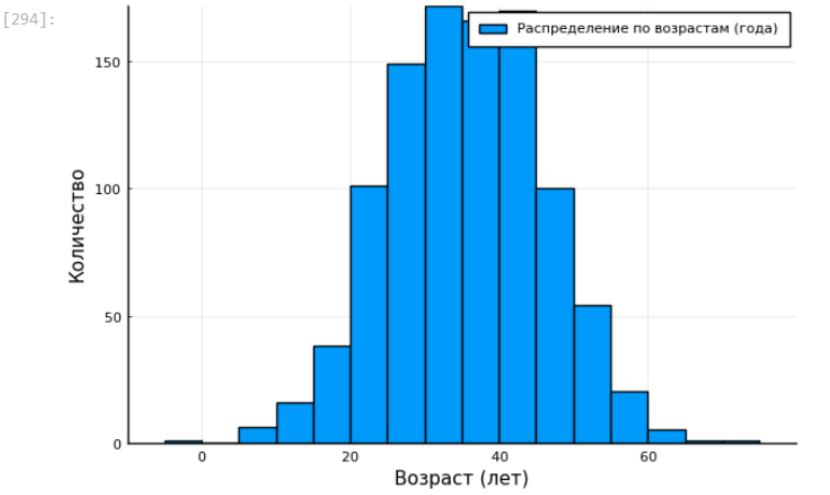

Рис. 3.32: Гистограмма нормального распределения

```
[298]: 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)"
)
```

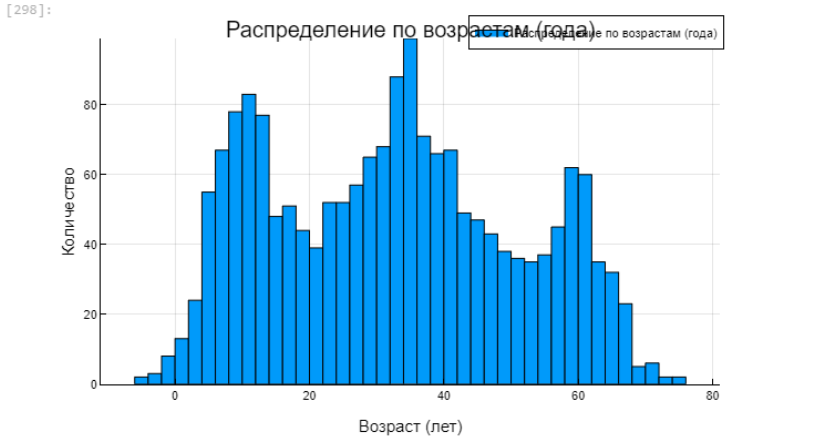

Рис. 3.33: Гистограмма распределения людей по возрастам

17. 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. 3.34) (рис. 3.35) (рис. 3.36) (рис. 3.37) (рис. 3.38)

[308]:
```
# подгружаем pyplot():
pyplot()
# построение серии графиков:
x=range(-2,2,length=10)
y = rand(10,4)
plot(x,y,
layout=(4,1), xlabel = "x",
ylabel= "y",
)
```

[308]:

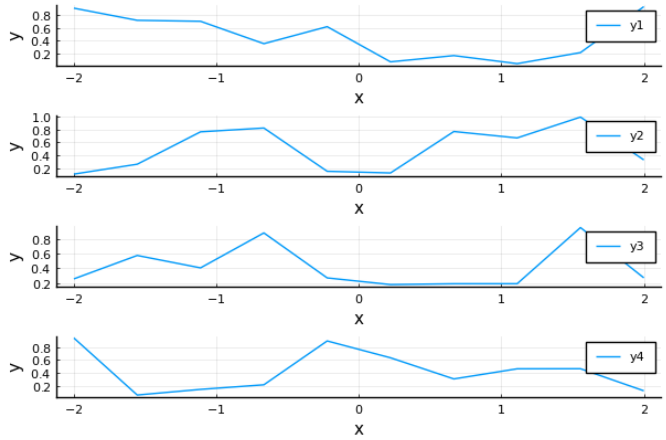

Рис. 3.34: Серия из 4-х графиков в ряд

[310]:
```
# подгружаем pyplot():
pyplot()
# построение серии графиков:
x=range(-2,2,length=10)
y = rand(10,4)
plot(x,y,
layout=4, xlabel = "x",
ylabel= "y",
)
```

[310]:

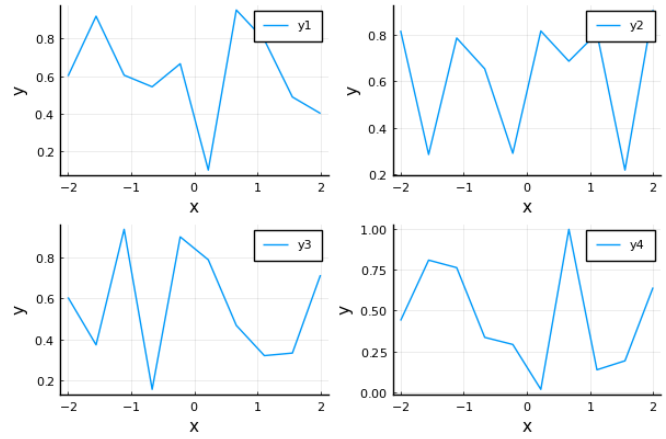

Рис. 3.35: Серия из 4-х графиков в сетке

[318]:
```
# график в виде линий:
p1 = plot(x,y, ylabel = "y", xlabel = "x",)
# график в виде точек:
p2 = scatter(x,y, ylabel = "y", xlabel = "x",)
# 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, ylabel = "y", xlabel = "x",)
plot(
p1,p2,p3,p4,
layout=(2,2),
legend=false,
size=(800,600),
background_color = :ivory
)
```

[318]:

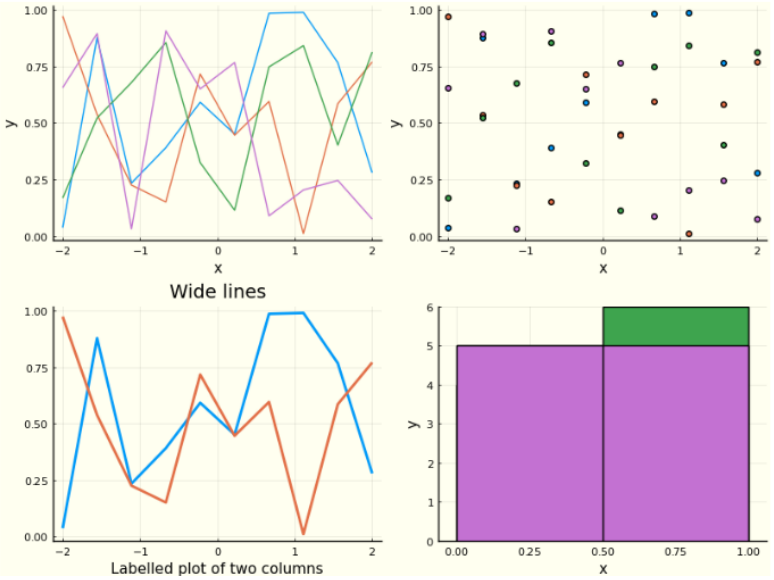

Рис. 3.36: Объединение нескольких графиков в одной сетке

```
[320]: 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
)
```

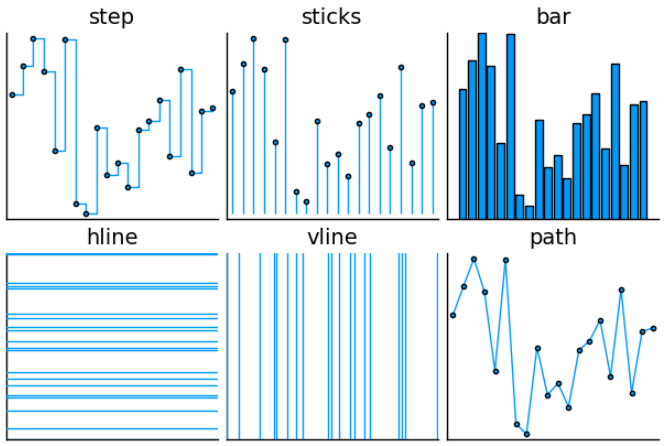

Рис. 3.37: Разнообразные варианты представления данных

```
[322]: 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)
)
```

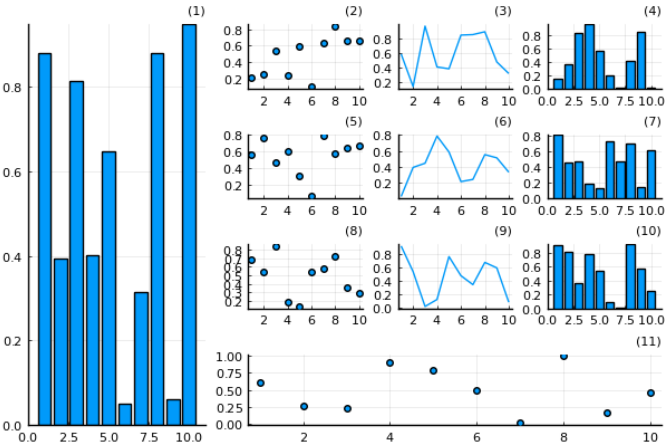

Рис. 3.38: Демонстрация применения сложного макета для построения графиков

18. Перешла к заданиям для самостоятельной работы. Нумерация соответствует.

- Задание 1 (рис. 3.39)

[12]:
```
x = 0:0.1:2π
sin_y = sin.(x)

plot( # Графики в одном окне
    plot(x, sin_y, title="Линия", lw=2), # Линейный график
    scatter(x, sin_y, title="Точки"), # Точечный график
    bar(x, abs.(sin_y), title="Гистограмма"), # Гистограмма (модули значений)
    histogram(sin_y, bins=10, title="Гистограмма значений"), # Гистограмма значений
    layout=(2, 2), xlabel = "x", ylabel = "y", label = "График"
)
```

[12]:

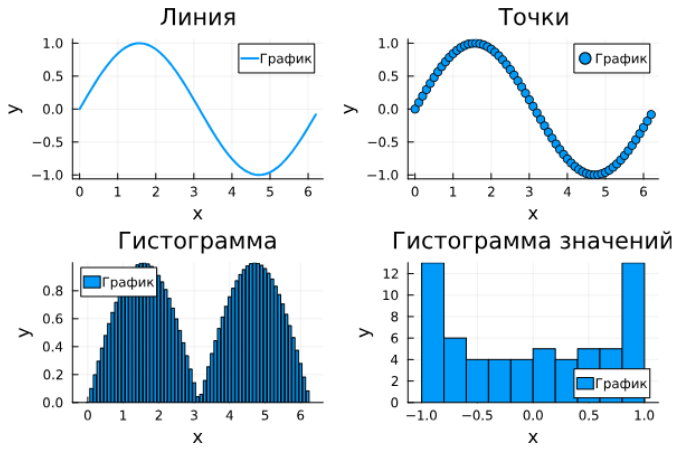

Рис. 3.39: Задание 1

- Задание 2 (рис. 3.40)

[20]:
```
x = 0:0.1:2π
y = sin.(x)

# Создание сетки с графиками разных стилей линий
plot(
    plot(x, y, label="Сплошная линия", color=:blue, linestyle=:solid, linewidth=2),  # Сплошная л
    plot(x, y, label="Пунктирная линия", color=:red, linestyle=:dot, linewidth=2),  # Пунктирная
    plot(x, y, label="Штриховая линия", color=:green, linestyle=:dash, linewidth=2),  # Штрихова
    plot(x, y, label="Линия с маркерами", color=:purple, marker=:circle, markersize=4,
        linewidth=2),  # Линия с маркерами
    layout=(2, 2),  # Макет 2x2
    xlabel="x", ylabel="y", title="y = sin(x)"
)
```

[20]:

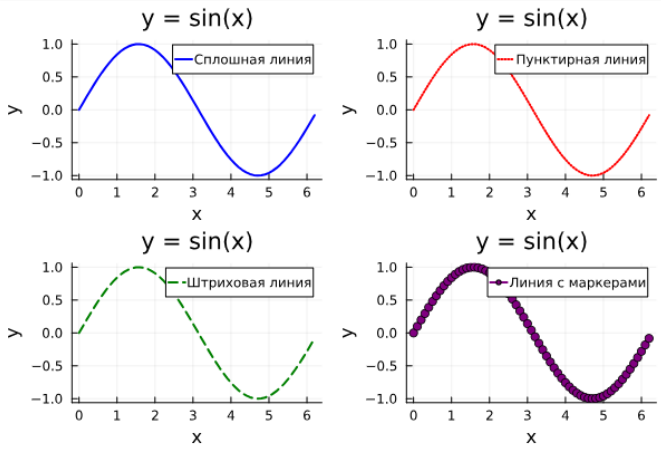

Рис. 3.40: Задание 2

- Задание 3 (рис. 3.41)

[22]:
```
x_vals = 0.1:0.1:5
y_vals = π .* x_vals.^2 .* log.(x_vals)
plot(
    x_vals, y_vals,
    label="y = πx²ln(x)",
    xlabel="x",
    ylabel="y",
    framestyle=:box,
    legend=:topright,
    grid=false,
    color=:red,
    foreground_color_border=:green # Задание зелёной рамки
)
# Настройка расстояний до осей и шрифта
xaxis!(font(12, "Arial"))
yaxis!(font(12, "Arial"))
```

[22]:

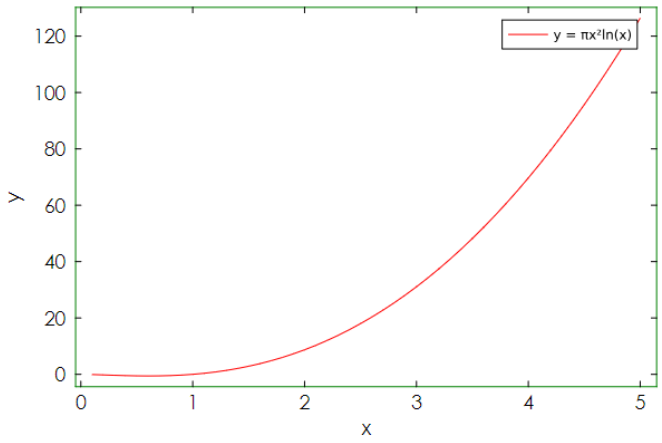

Рис. 3.41: Задание 3

- Задание 4 (рис. 3.42)

```
[30]: x_points = [-2, -1, 0, 1, 2]
y_points = x_points.^3 .- 3 .* x_points

p1 = scatter(x_points, y_points, label="Точки", title="График точек")
p2 = plot(x_points, y_points, label="Линии", title="График линий")
p3 = plot(x_points, y_points, marker=:circle, label="Линии и точки", title="График линии и точки"
p4 = plot(x_points, y_points, label="Кривая", title="Кривая")

plot(p1, p2, p3, p4, layout=(2, 2), xlabel="x",ylabel="y")
savefig("figure_leginkikh.png")
```

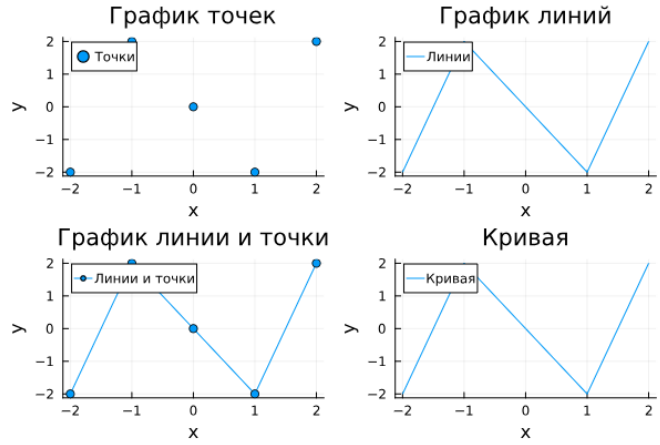

Рис. 3.42: Задание 4

- Задание 5 (рис. 3.43) (рис. 3.44)

```
[34]: x = 3:0.1:6

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

# Построение графиков на одном рисунке
plot(x, y1.(x), label="y1(x) = πx", color=:blue, xlabel="x", ylabel="y",
    title="Графики функций y1(x) и y2(x)", grid=true)
plot!(x, y2.(x), label="y2(x) = exp(x)cos(x)", color=:red)
```

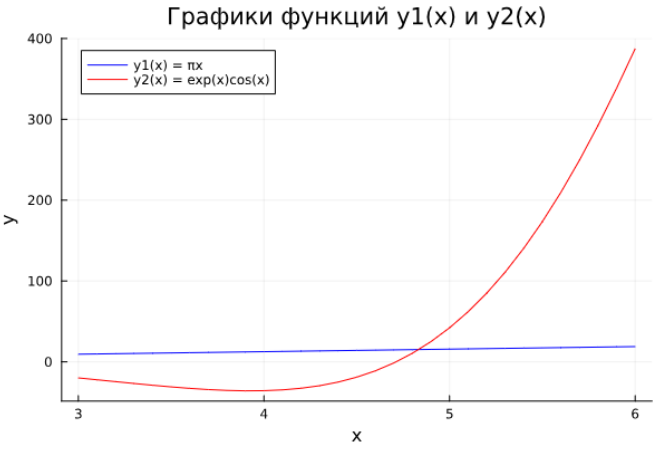

Рис. 3.43: Задание 5.1

[46]:
```
x = 3:0.1:6

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

# Построение графиков с двумя осями ординат
p1 = plot(x, y1.(x), label="y1(x) = πx", color=:blue, xlabel="x", ylabel="y1(x)",
    grid=true)
p2 = plot(x, y2.(x), label="y2(x) = exp(x)cos(x)", color=:red, xlabel="x", ylabel="y2(x)",
    secondary = true)

# Отображаем оба графика
plot(p1, p2, title="Графики")
```

[46]:

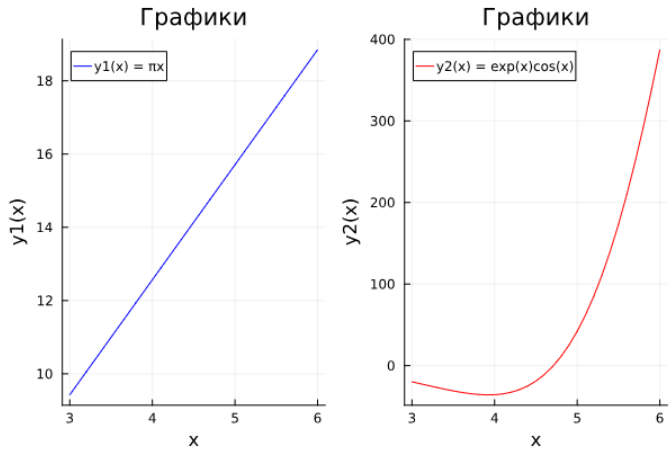

Рис. 3.44: Задание 5.2

- Задание 6 (рис. 3.45)

[50]:
```julia
x_data = 1:10
y_data = [2.1, 2.3, 2.7, 3.0, 3.4, 3.9, 4.2, 4.6, 5.1, 5.5]
errors = [0.2, 0.1, 0.15, 0.2, 0.1, 0.25, 0.2, 0.15, 0.1, 0.2]

plot(
    x_data, y_data,
    yerror=errors, # Ошибки измерений
    label="Экспериментальные данные",
    xlabel="Номер измерения",
    ylabel="Значение",
    legend=:topleft,
    color=:green,
    marker=:circle,
    lw=2,
    title = "Экспериментальные данные"
)
```

[50]:

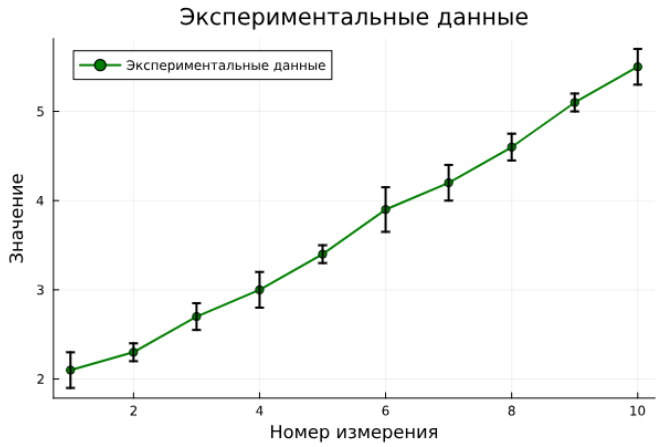

Рис. 3.45: Задание 6

- Задание 7 (рис. 3.46)

```
[52]: random_x = rand(50) * 10
random_y = rand(50) * 10
scatter(
    random_x, random_y,
    label="Случайные данные",
    xlabel="X", ylabel="Y",
    title="Точечный график случайных данных",
    legend=:topright,
    color=:blue, marker=:diamond
)
```

[52]:

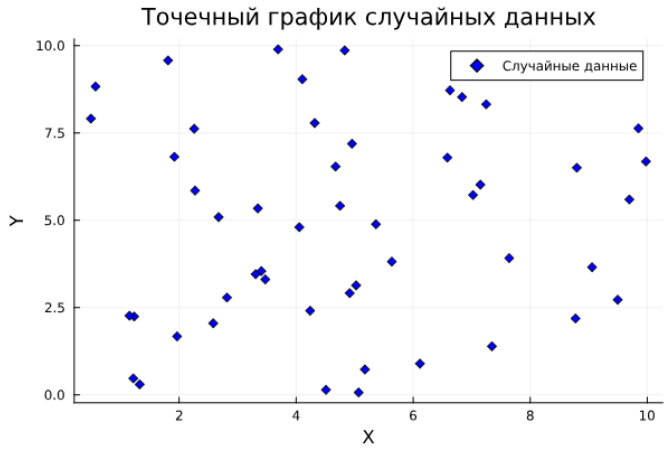

Рис. 3.46: Задание 7

- Задание 8 (рис. 3.47)

```
[56]: random_x3 = rand(50) * 10
random_y3 = rand(50) * 10
random_z3 = rand(50) * 10
plot(
    random_x3, random_y3, random_z3,
    seriestype=:scatter, marker=:circle, color=:red,
    xlabel="X", ylabel="Y", zlabel="Z",
    label="Случайные данные",
    title="3D точечный график случайных данных",
    legend=false
)
```

[56]:

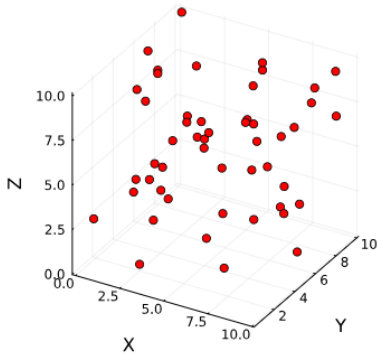

Рис. 3.47: Задание 8

- Задание 9 (рис. 3.48)

```
[58]: gr()  # Устанавливаем бэкенд GR
      @gif for i in 0:0.1:8π
          plot(sin, 0, i, label="sin(x)", title="Анимация синусоиды", xlabel="x", ylabel="y",
              legend=:topright)
      end
```

```
[ Info: Saved animation to C:\Users\galin\tmp.gif
```

[58]:

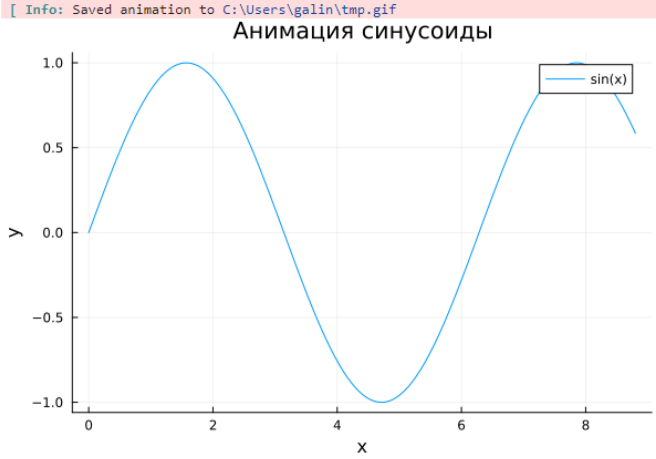

Рис. 3.48: Задание 9

- Задание 10 (рис. 3.49)

```
*[60]: @gif for k in [3, 4, 5, 1/2, 3/2] # Целые и рациональные значения для k
           θ = 0:0.1:2π
           x(t) = (3 - k) * cos(t) + k * cos((3 - k) / k * t)
           y(t) = (3 - k) * sin(t) - k * sin((3 - k) / k * t)

           plot(x.(θ), y.(θ), label="Гипоциклоид, k=$k", title="Анимация гипоциклоиды", xlabel="x",
               ylabel="y", legend=:topright)
       end
```

```
[ Info: Saved animation to C:\Users\galin\tmp.gif
```

[60]:

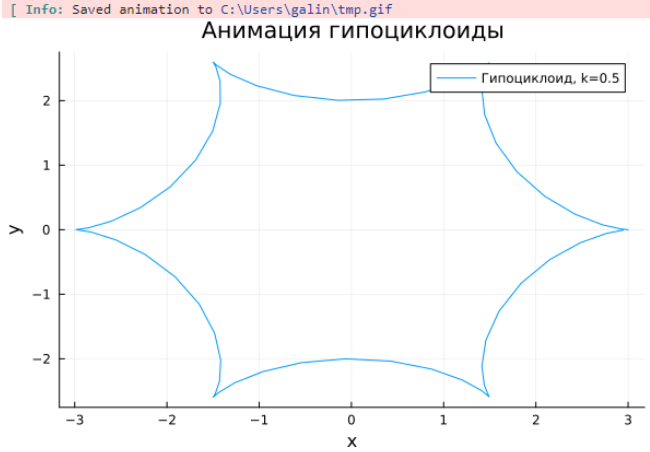

Рис. 3.49: Задание 10

- Задание 11 (рис. 3.50)

[64]:
```
gr()  # Устанавливаем бэкенд GR
@gif for k in [3, 4, 5, 1/2, 3/2] # Целые и рациональные значения для k
    θ = 0:0.1:2π
    x(t) = (3 + k) * cos(t) - k * cos((3 + k) / k * t)
    y(t) = (3 + k) * sin(t) - k * sin((3 + k) / k * t)

    plot(x.(θ), y.(θ), label="Эпициклоид, k=$k", title="Анимация эпициклоиды",
        xlabel="x", ylabel="y", legend=:topright)
end
```

```
[ Info: Saved animation to C:\Users\galin\tmp.gif
```

[64]:

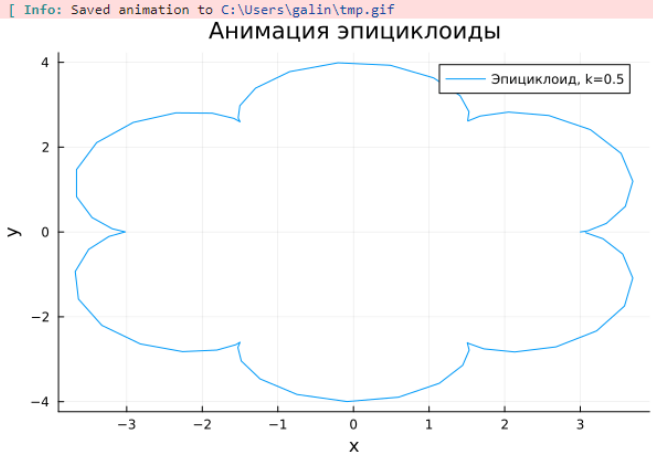

Рис. 3.50: Задание 11

4 Вывод

Освоила синтаксис языка Julia для построения графиков.