

Лабораторная работа №5

Построение графиков

Чемоданова Ангелина Александровна

16 сентября 2025

Российский университет дружбы народов имени Патриса Лумумбы, Москва, Россия

- Чемоданова Ангелина Александровна
- Студентка НФИбд-02-22
- Российский университет дружбы народов имени Патриса Лумумбы
- 1132226443@pfur.ru
- <https://github.com/aachemodanova>



Цель работы

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

Задание

1. Используя Jupyter Lab, повторите примеры. При этом дополните графики обозначениями осей координат, легендой с названиями траекторий, названиями графиков и т.п.
2. Выполните задания для самостоятельной работы.

Основные пакеты для работы с графиками в Julia

Рассмотрим построение графика функции $f(x) = (3x^2 + 6x - 9)e^{-0.3x}$ разными способами:

```
# подключаем для использования Plots:  
using Plots  
✓ 0.0s  
  
# задание функции:  
f(x) = (3x.^2 + 6x .- 9).*exp.(-0.3x)  
✓ 0.0s  
f (generic function with 2 methods)  
  
# генерирование массива значений x в диапазоне от -5 до 10 с шагом 0.1  
# (это можно сделать через указание длины массива):  
x = collect(range(-5,10,length=151))  
✓ 0.0s  
151-element Vector{Float64}:  
-5.0  
-4.9  
-4.8  
-4.7  
-4.6  
-4.5  
-4.4  
-4.3  
-4.2  
-4.1  
⋮  
9.2  
9.3  
9.4  
9.5  
9.6  
9.7  
9.8  
9.9  
10.0
```

Рис. 1: Задание функции

Основные пакеты для работы с графиками в Julia

```
# генерирование массива значений у:  
y = f(x)
```

✓ 0.6s

Julia

```
151-element Vector{Float64}:  
161.34080653217032  
146.26477779394003  
132.19219298833204  
119.06942359634911  
106.8453557470588  
95.47128188475011  
84.9007968362764  
75.08969810741856  
65.99589024347995  
57.579293095755176  
:  
18.995125520095375  
18.811475185747092  
18.625664977689375  
18.437877278854756  
18.248288702120195  
18.057070179740368  
17.86438705526336  
17.6703991776229  
17.475260997120245
```

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

✓ 0.4s

Julia

```
Plots.GRBackend()
```

Рис. 2: Задание функции

Основные пакеты для работы с графиками в Julia

```
# задание опций при построении графика
# (название кривой, подписи по осям, цвет графика):
plot(x,y,
      title="A simple curve",
      xlabel="Variable x",
      ylabel="Variable y",
      color="pink",
      label="f(x) = (3x.^2 + 6x .. - 9).*exp.(-0.3x)")
```

✓ 6.8s

Julia

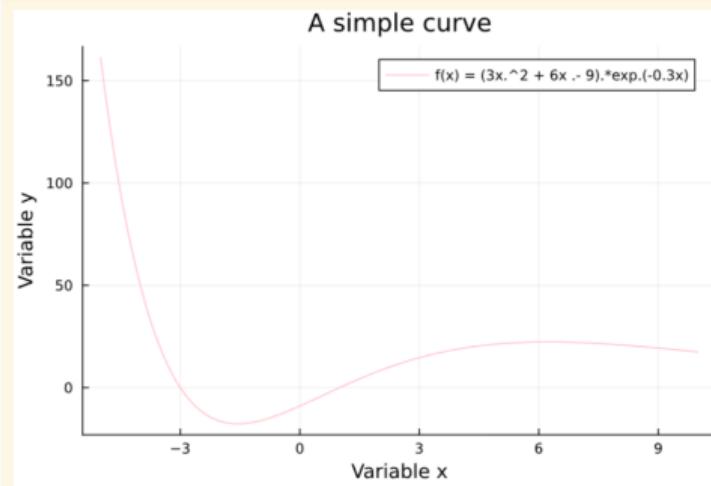


Рис. 3: График функции, построенный при помощи gr()

Основные пакеты для работы с графиками в Julia

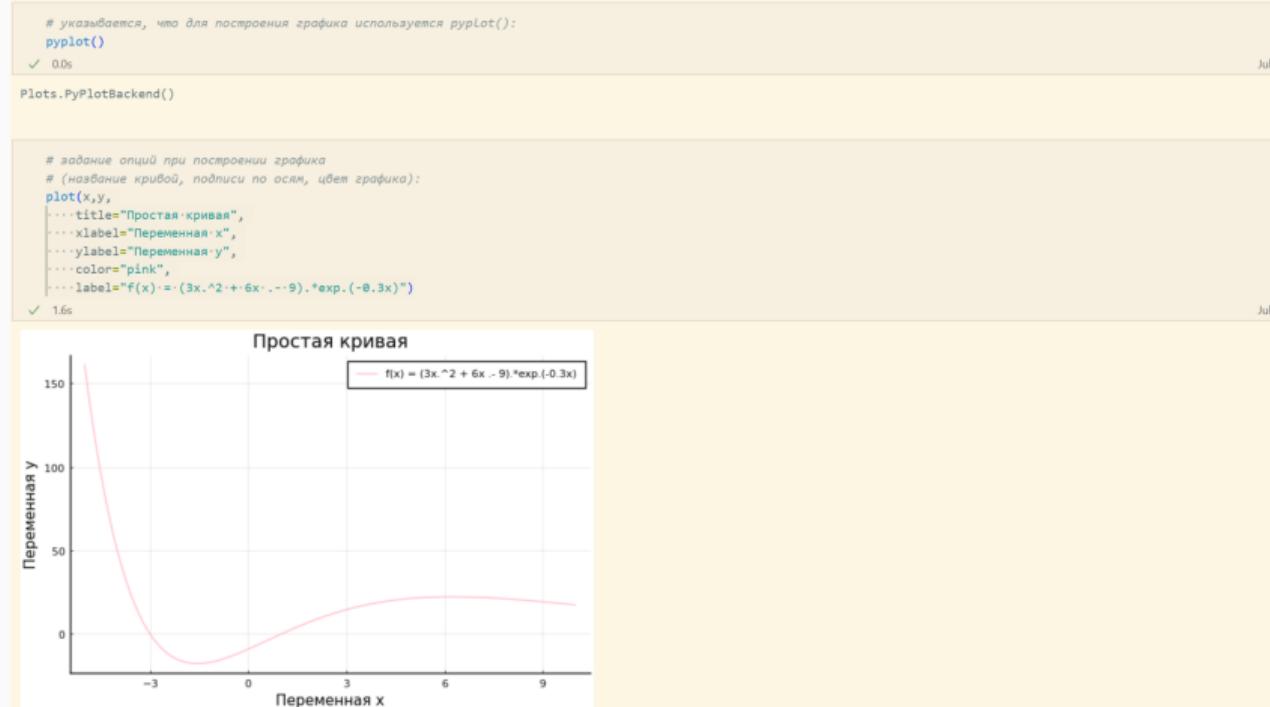
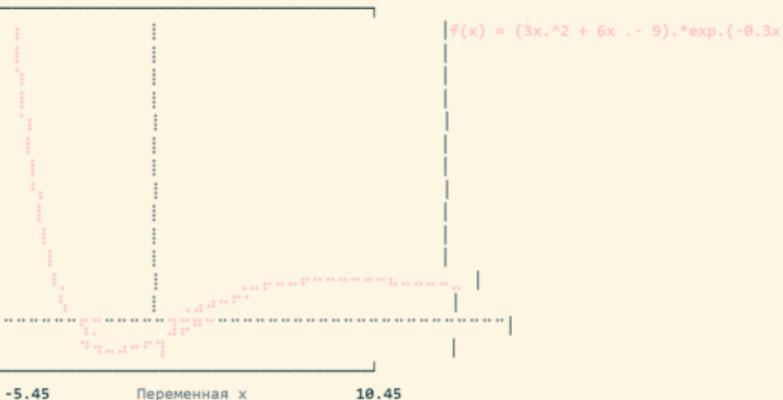


Рис. 4: График функции, построенный при помощи pyplot()

Основные пакеты для работы с графиками в Julia

```
unicodeplots()  
✓ 4.8s  
Plots.UnicodePlotsBackend()  
  
plot(x,y,  
      title="Простая кривая",  
      xlabel="Переменная x",  
      ylabel="Переменная у",  
      color="pink",  
      label="f(x) = (3x.^2 + 6x .- 9).*exp.(-0.3x)")  
✓ 14.2s
```



The figure shows a plot of a mathematical function. The x-axis is labeled "Переменная x" and has tick marks at -5.45, 0, and 10.45. The y-axis is labeled "Переменная у" and has tick marks at -23.0172 and 166.71. The plot area contains a pink parabola opening upwards, with its vertex near the bottom left. Above the plot, the function definition is given as $f(x) = (3x.^2 + 6x .- 9).*exp.(-0.3x)$. The plot is titled "Простая кривая".

Рис. 5: График функции, построенный при помощи unicodeplots()

Опции при построении графика

На примере графика функции $\sin(x)$ и графика разложения этой функции в ряд Тейлора рассмотрим дополнительные возможности пакетов для работы с графикой:

Опции при построении графика

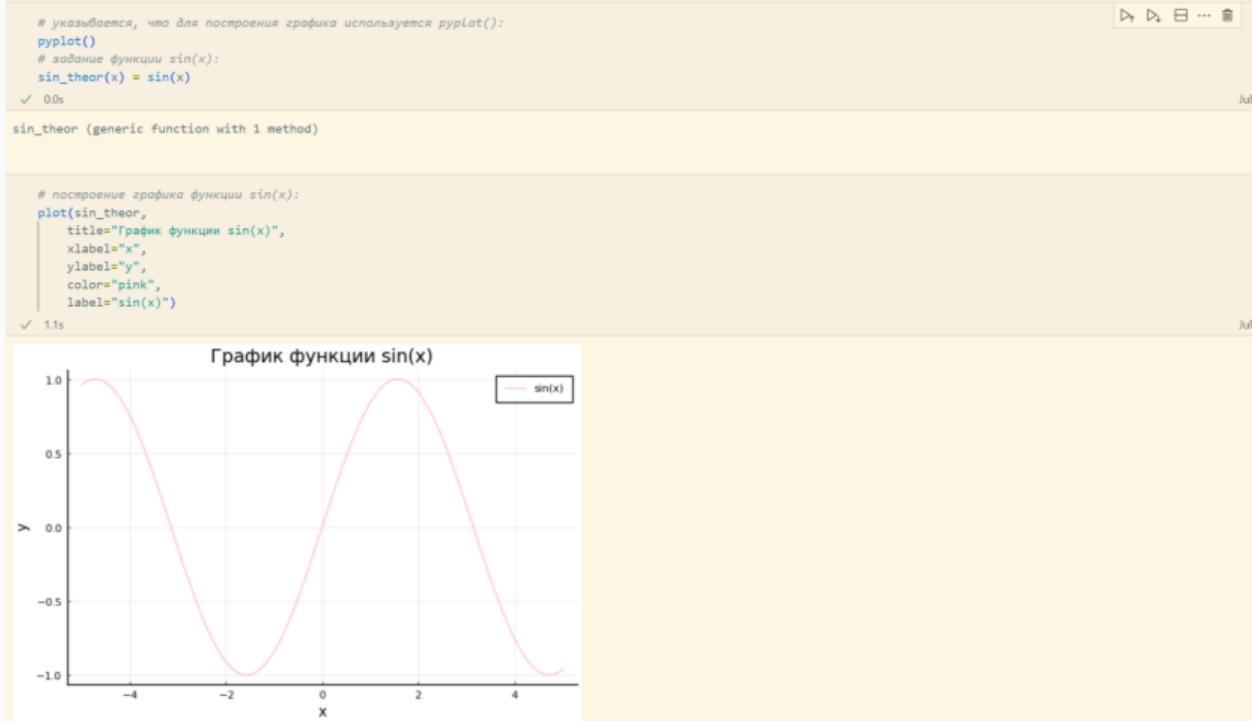


Рис. 6: График функции $\sin(x)$

Опции при построении графика

```
# 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
```

✓ 0.0s

Julia

```
sin_taylor (generic function with 1 method)
```

```
# построение графика функции sin_taylor(x):  
plot(sin_taylor,  
      title="График ф-ии разложения исходной ф-ии в ряд Тейлора",  
      xlabel="x",  
      ylabel="y",  
      color="pink",  
      label="Ряд Тейлора")
```

✓ 2.3s

Julia

График ф-ии разложения исходной ф-ии в ряд Тейлора

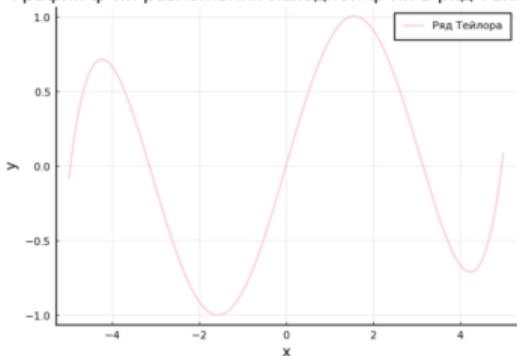


Рис. 7: График функции разложения исходной функции в ряд Тейлора

Опции при построении графика

```
# построение двух функций на одном графике:  
plot(sin_theor, label="sin(x)", xlabel="Axis x", ylabel="Axis y", title="Сравнение sin(x) и ряда Тейлора")  
plot!(sin_taylor, label="Ряд Тейлора")
```

✓ 1.9s

Julia

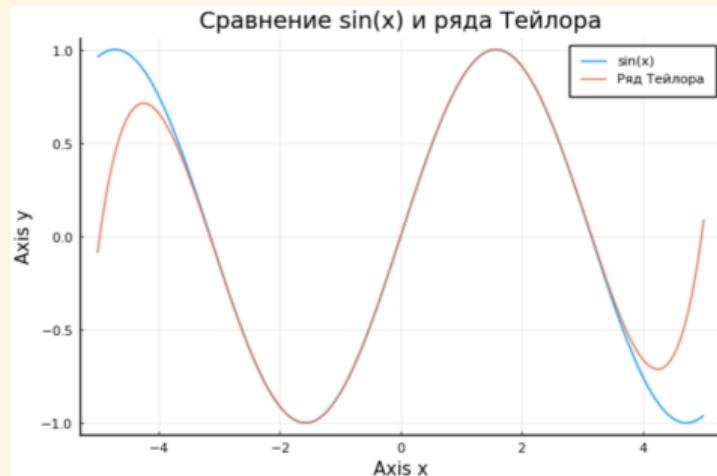


Рис. 8: Графики исходной функции и её разложения в ряд Тейлора

Опции при построении графика

Затем добавим различные опции для отображения на графике:

```
plot(
    # функция sin(x):
    sin_taylor,
    # подпись в легенде, цвет и тип линии:
    label = "sin(x), разложение в ряд Тейлора",
    line=(:blue, 0.3, 6, :solid),
    # размер графика:
    size=(800, 500),
    # параметры отображения значений по осям
    xticks = (-5:0.5:5),
    yticks = (-1:0.1:1),
    xtickfont = font(12, "Times New Roman"),
    ytickfont = font(12, "Times New Roman"),
    # подписи по осям:
    ylabel = "y",
    xlabel = "x",
    # название графика:
    title = "Разложение в ряд Тейлора",
    # побором 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)
```

✓ 7.3s

Julia

Рис. 9: Добавления опций для графика

Опции при построении графика

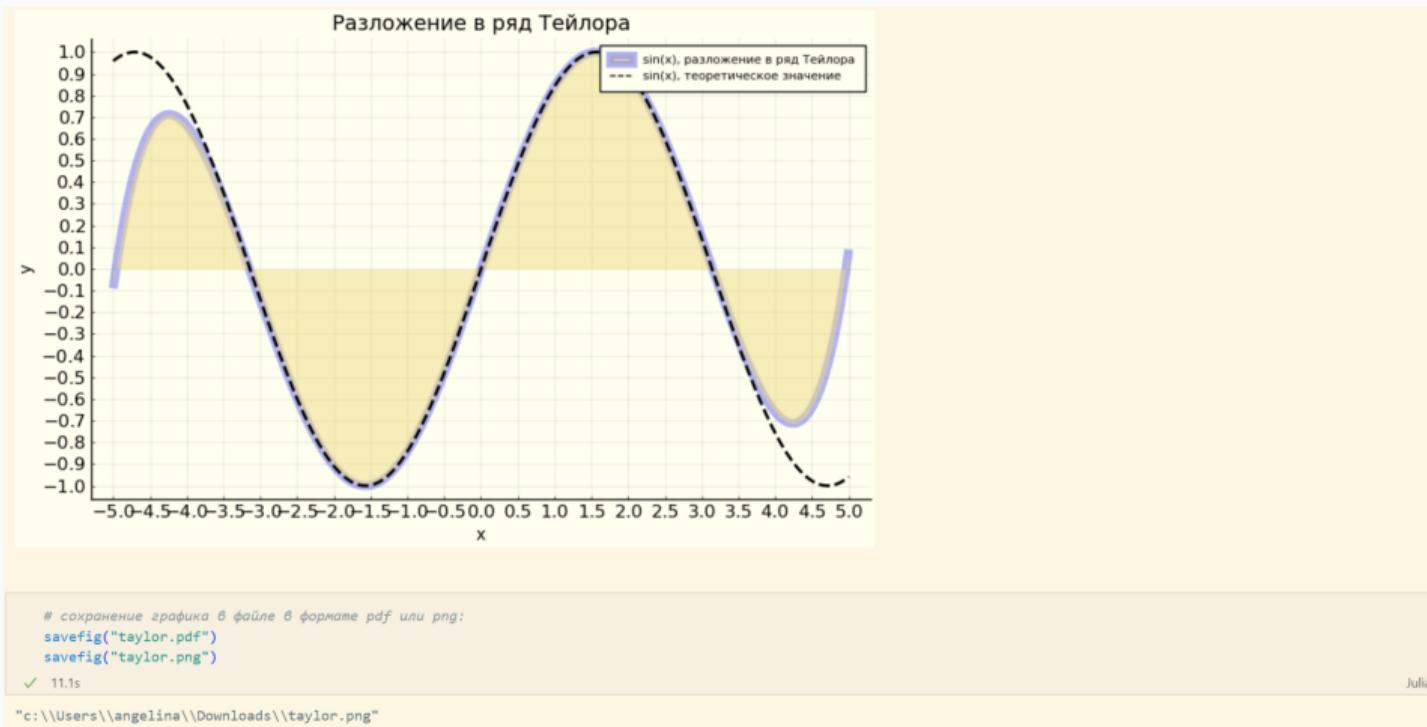


Рис. 10: Вид графиков после добавления опций при их построении

Точечный график

Как и при построении обычного графика для точечного графика необходимо задать массив значений x , посчитать или задать значения y , задать опции построения графика:

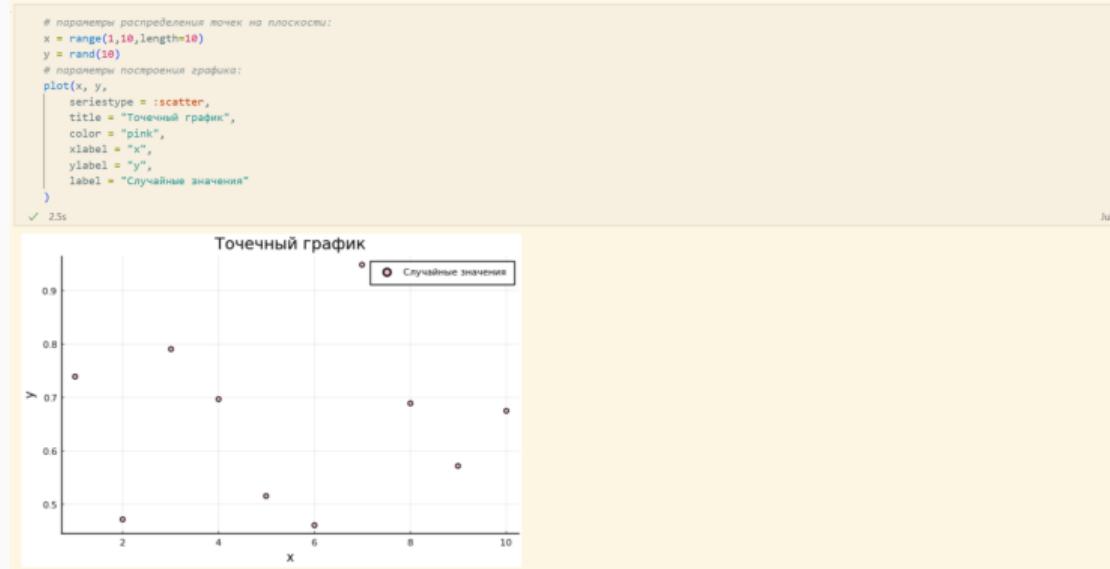


Рис. 11: График десяти случайных значений на плоскости (простой точечный график)

Точечный график

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

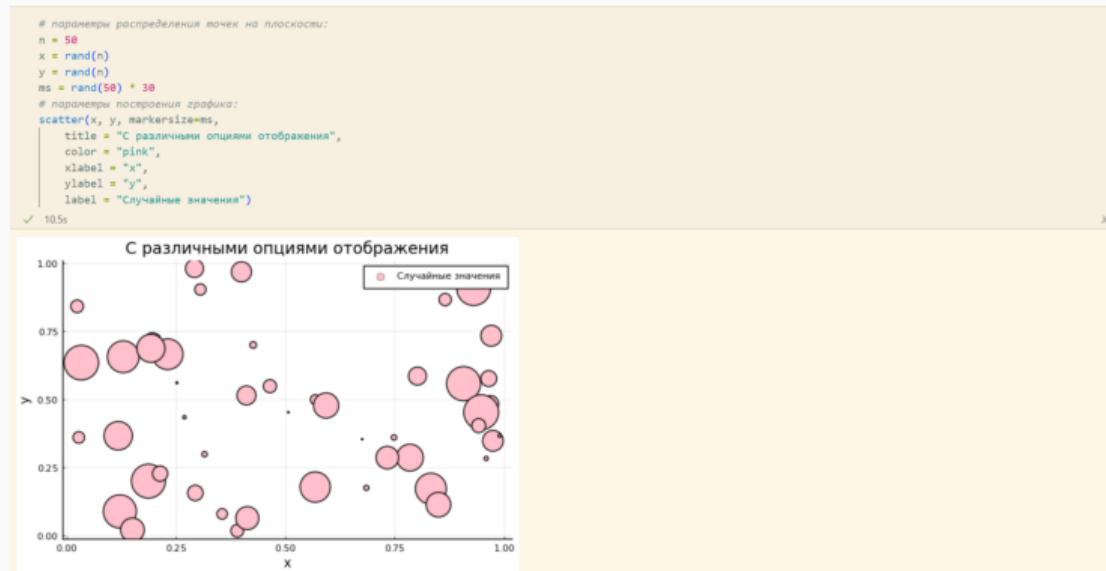


Рис. 12: График пятидесяти случайных значений на плоскости с различными опциями отображения (точечный график с кодированием значения размером точки)

Точечный график

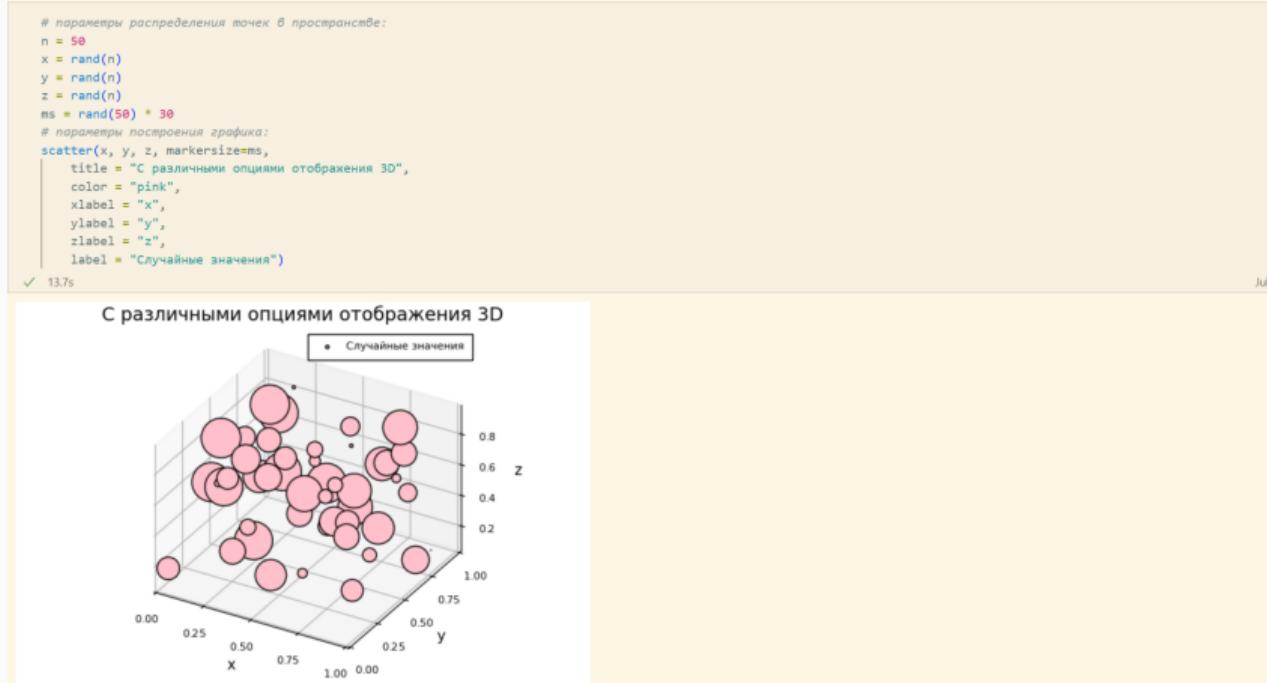


Рис. 13: График пятидесяти случайных значений в пространстве с различными опциями отображения (3-мерный точечный график с кодированием значения размером точки)

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

Для демонстрации зададим искусственно некоторую функцию, в данном случае похожую по поведению на экспоненту:

Аппроксимация данных

```
# 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, xlabel="Axis·x", ylabel="Axis·y", title="График·функции·с·шумом")
```

✓ 2.6s

Julia

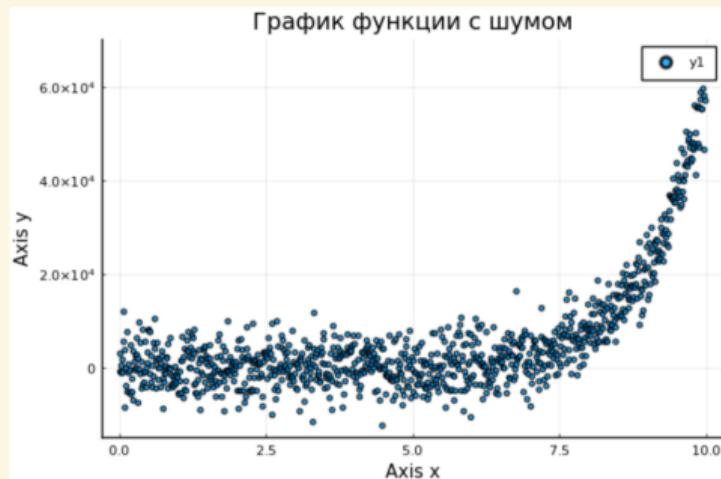


Рис. 14: Пример функции

Аппроксимация данных

Аппроксимируем полученную функцию полиномом 5-й степени:

```
# 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:  
q = 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,q,linewidth=3, color=:red, xlabel="Axis x", ylabel="Axis y", title="Аппроксимация функции  с шумом")
```

Julia

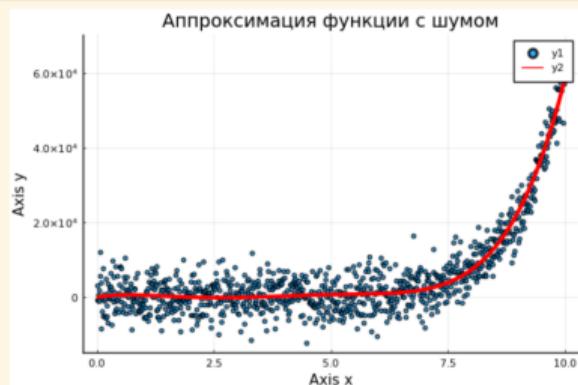


Рис. 15: Пример аппроксимации исходной функции полиномом 5-й степени

Пример траектории

```
# пример случайной траектории
# (заданы обозначение траектории, легенда вверху справа, без сетки)
plot(randn(100),
      ylabel="Axis y",
      leg=topright,
      grid = :off,
      xlabel="Axis x",
      title="Случайная траектория",
      label="Траектория"
)
```

✓ 12.8s

Julia

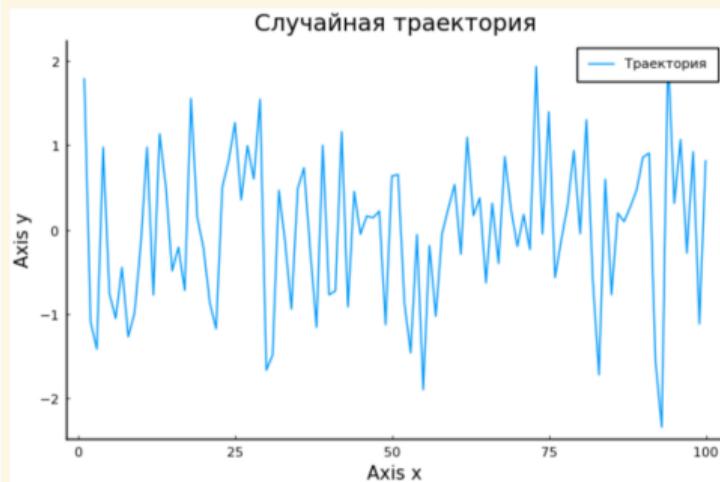


Рис. 16: Пример отдельно построенной траектории

Полярные координаты

Приведём пример построения графика функции в полярных координатах:

```
# функция в полярных координатах:  
r(θ) = 1 + cos(θ) * sin(θ)^2  
# полярная система координат:  
θ = range(θ, stop=2π, length=50)  
# график функции, заданной в полярных координатах:  
plot(θ, r(θ),  
    proj=:polar,  
    lims=(θ,1.5),  
    xlabel="Угол",  
    ylabel="Радиус",  
    title="Полярная кривая",  
    label="1 + cos(θ) * sin(θ)^2"  
)
```

✓ 2.6s

Julia

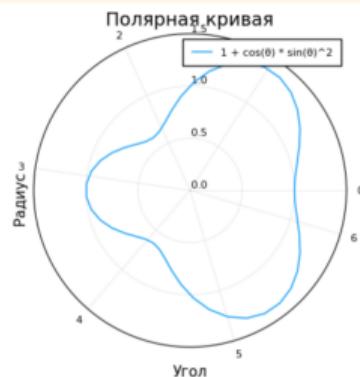


Рис. 17: График функции, заданной в полярных координатах

Параметрический график

Приведём пример построения графика параметрически заданной кривой на плоскости:

```
# параметрическое уравнение:  
x_1(t) = sin(t)  
y_1(t) = sin(2t)  
# построение графика:  
plot(x_1, y_1, 0, 2π, leg=false, fill=(0,:orange), xlabel="sin(t)", ylabel="sin(2t)", title="Параметрическая траектория", label="Траектория(x=sin(t), y=sin(2t))")  
Julia  
✓ 2.0s
```

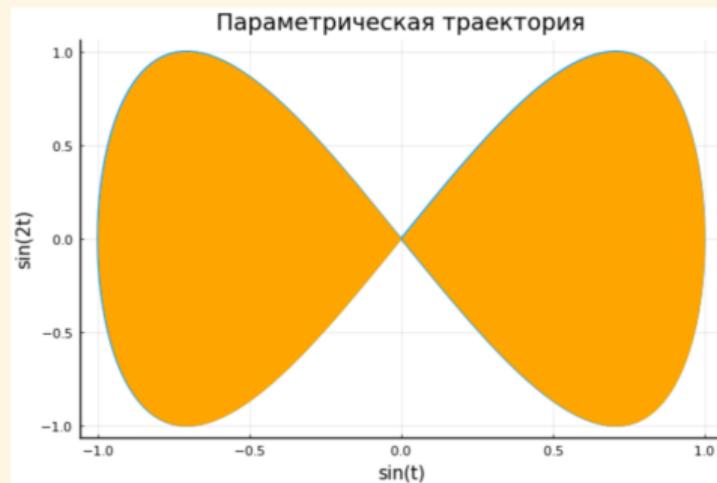


Рис. 18: Параметрический график кривой на плоскости

Параметрический график

```
# параметрическое уравнение
t = range(0, stop=10, length=1000)
x = cos.(t)
y = sin.(t)
z = sin.(5t)
# построение графика:
plot(x, y, z, xlabel="x=cos(t)", ylabel="y=sin(t)", zlabel="z=sin(5t)", title="Параметрическая 3D траектория", label="")
✓ 2.0s
```

Julia

Параметрическая 3D траектория

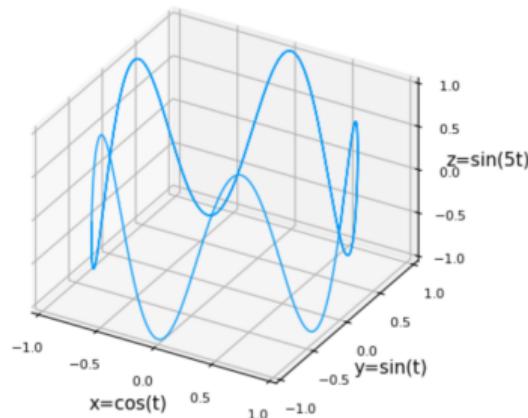


Рис. 19: Параметрический график кривой в пространстве

График поверхности

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

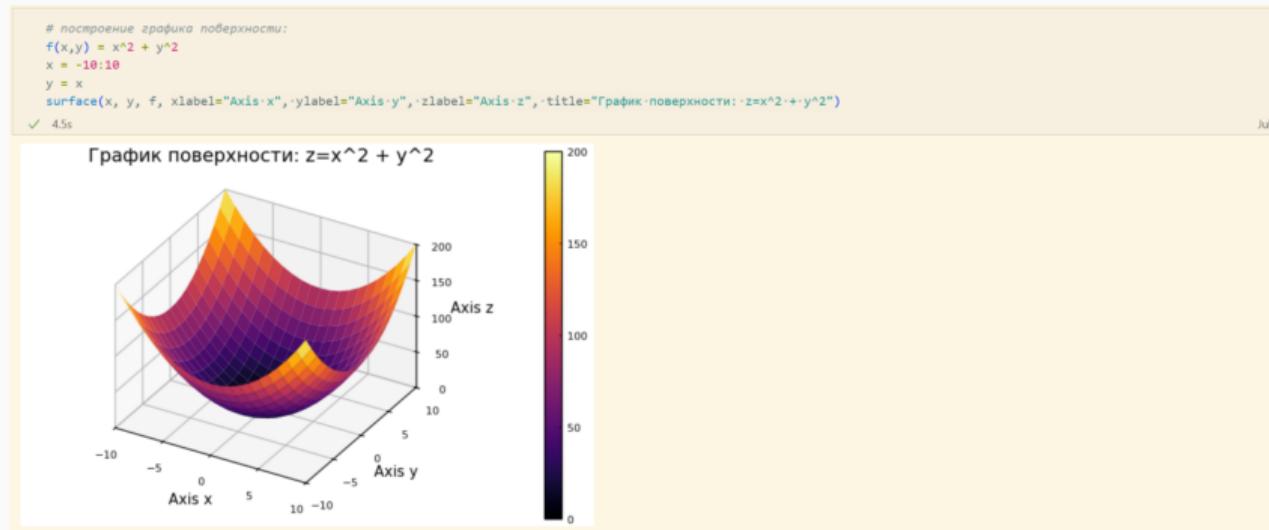


Рис. 20: График поверхности (использована функция `surface()`)

График поверхности

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

✓ 3.0s

Julia

График поверхности: $z=x^2 + y^2$

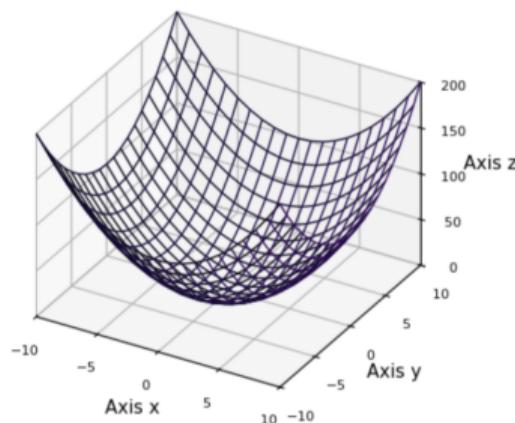


Рис. 21: График поверхности (использована функция plot())

График поверхности

```
f(x,y) = x^2 + y^2
x = -10:0.1:10
y = x
plot(x, y, f,
    linetype = :surface,
    xlabel="Axis x", ylabel="Axis y", zlabel="Axis z", title="График поверхности: z=x^2 + y^2"
)
1m 23.3s
```

Julia

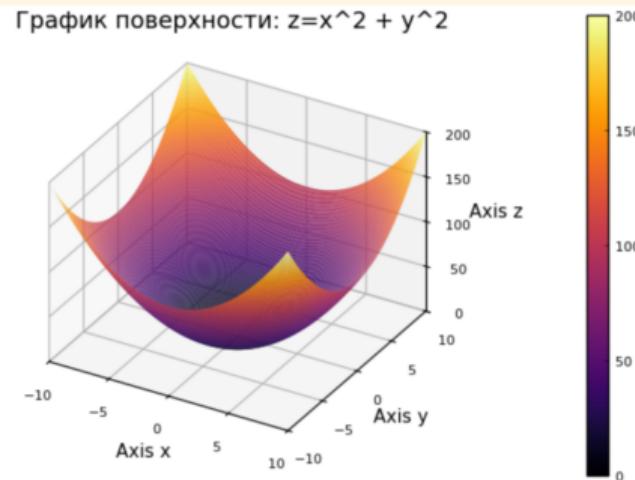


Рис. 22: Сглаженный график поверхности

График поверхности

```
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),
    xlabel="Axis·x", ylabel="Axis·y", zlabel="Axis·z", title="Поверхность: x*y-x-y+1"
)
✓ 25.7s
```

Julia

Поверхность: $x*y-x-y+1$

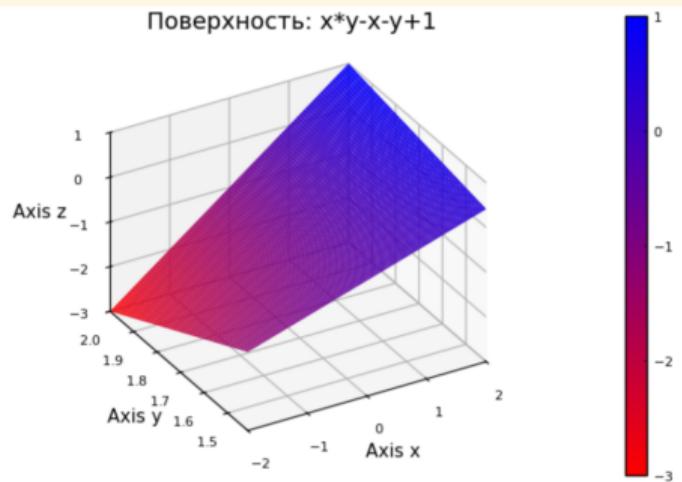


Рис. 23: График поверхности с изменённым углом зрения

Линии уровня

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

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

Поверхности уровня представляют собой непересекающиеся пространственные поверхности.

Линии уровня

Рассмотрим поверхность, заданную функцией $g(x, y) = (3x + y^2)|\sin(x) + \cos(y)|$:

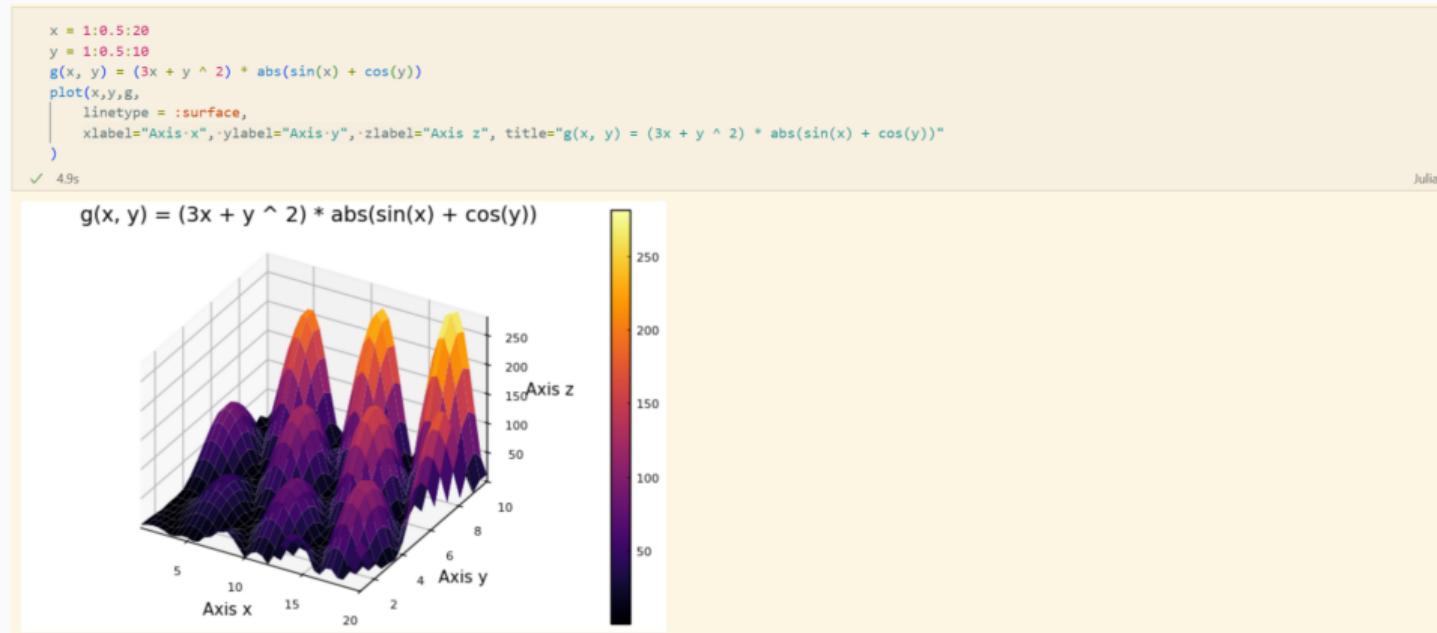


Рис. 24: График поверхности, заданной функцией $g(x, y) = (3x + y^2)|\sin(x) + \cos(y)|$

Линии уровня

Линии уровня можно построить, используя проекцию значений исходной функции на плоскость:

```
contour(x, y, g, xlabel="Axis·x", ylabel="Axis·y", title="Контурный·график·функции")  
✓ 2.3s
```

Julia

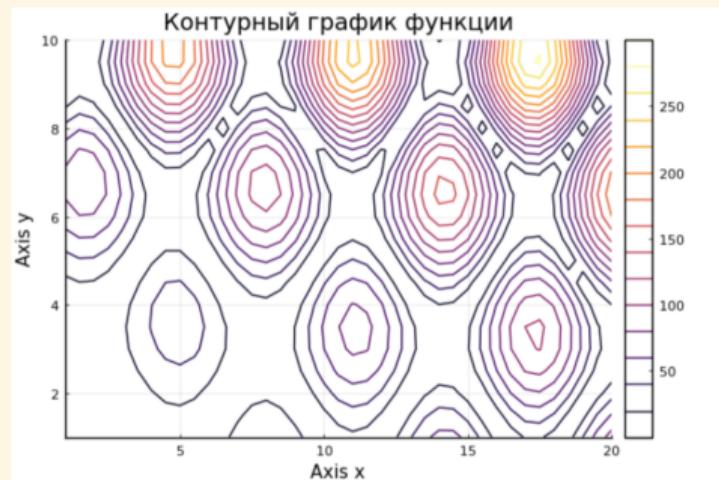


Рис. 25: Линии уровня

Линии уровня

Можно дополнительно добавить заливку цветом:

```
p = contour(x, y, g,  
            fill=true,  
            xlabel="Axis x", ylabel="Axis y", title="Заполненный контурный график функции")  
plot(p)
```

✓ 6.0s

Julia

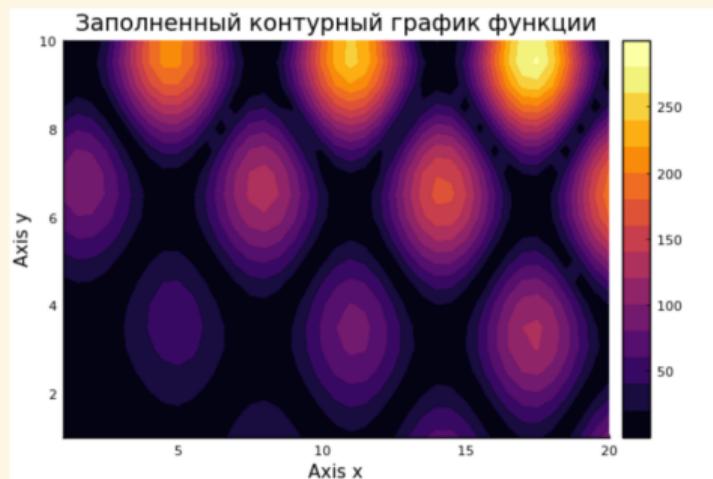


Рис. 26: Линии уровня с заполнением

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

Векторные поля задают векторными функциями.

Векторные поля

```
# 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,  
    xlabel="Axis x", ylabel="Axis y", zlabel="Axis z", title="x^3 - 3x + y^2"  
)
```

✓ 24.3s

Julia

$x^3 - 3x + y^2$

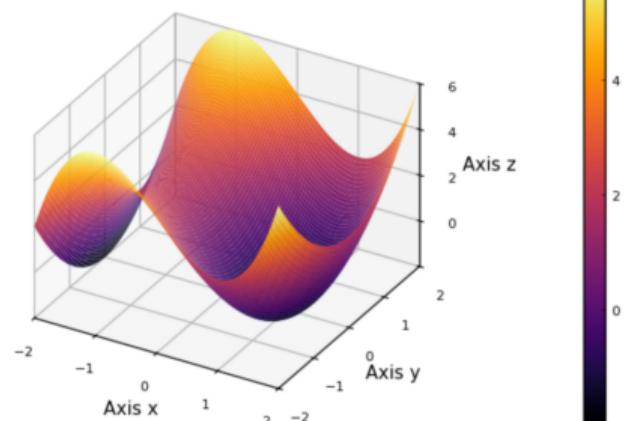


Рис. 27: График функции $h(x, y) = x^3 - 3x + y^2$

Векторные поля

```
# построение линий уровня:  
contour(X, Y, h, xlabel="Axis x", ylabel="Axis y", title="Линии уровня функции: x^3 - 3x + y^2")
```

✓ 2.0s

Julia

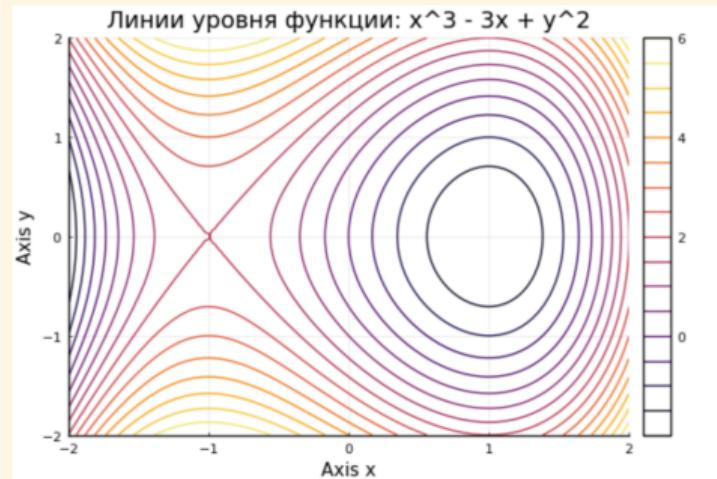


Рис. 28: Линии уровня для функции $h(x, y) = x^3 - 3x + y^2$

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

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

Анимация

```
pyplot()
```

✓ 0.0s

Julia

```
Plots.PyPlotBackend()
```

```
# построение поверхности:
```

```
i = 0
```

```
X = Y = range(-5,stop=5,length=40)
```

```
surface(X, Y, (x,y) -> sin(x+10sin(i))+cos(y))
```

✓ 11.0s

Julia

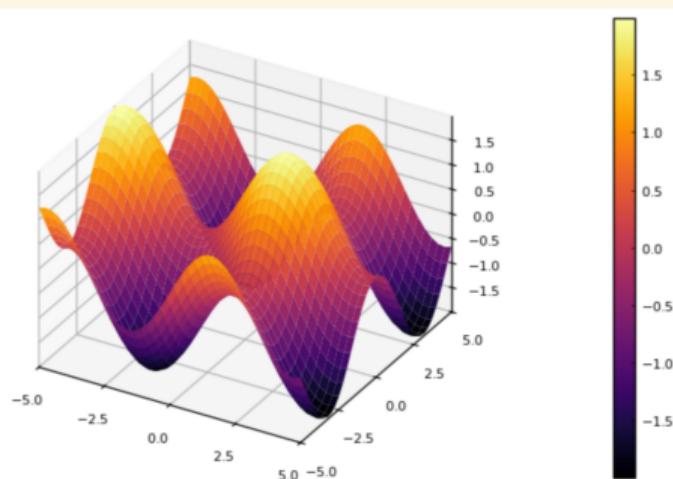


Рис. 29: Статичный график поверхности

Анимация

```
# 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
```

✓ 2m 43.5s

Julia

```
| Info: Saved animation to C:\Users\angelina\AppData\Local\Temp\jl_yjwjahC3qn.gif  
└ @ Plots C:\Users\angelina\.julia\packages\Plots\EcLl\src\animation.jl:156
```

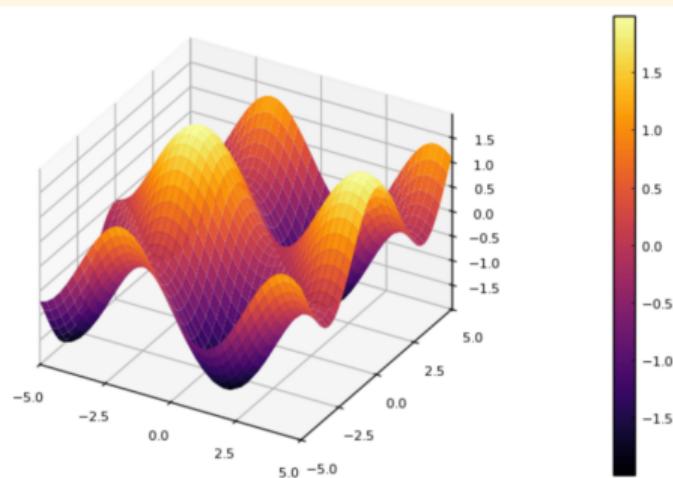


Рис. 30: Анимированный график поверхности

Гипоциклоида

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

```
# радиус малой окружности:  
r_1 = 1  
# коэффициент для построения большой окружности:  
k = 3  
# число отсчётов:  
n = 100  
✓ 0.0s
```

julia

100

```
# массив значений угла θ:  
# theta from 0 to 2pi (+ a little extra)  
θ = collect(0:2π/100:2π+2π/100)  
# массивы значений координат:  
X = r_1*k*cos.(θ)  
Y = r_1*k*sin.(θ)  
✓ 0.0s
```

julia

```
101-element Vector{Float64}:  
0.0  
0.18837155858794014  
0.3759997006929128  
0.5621439437571739  
0.74660696614945643  
0.92785098931248421  
1.104373658054034  
1.2773378746952182  
1.4452619223805146  
1.60748038493699  
:  
-1.4452610223805146  
-1.2773378746952166  
-1.1043736580540335  
-0.92785098931248404  
-0.74660696614945634  
-0.562143943757174  
-0.37599970069291133  
-0.1883715585879398  
1.929747179611964e-15
```

Рис. 31: Построение большой окружности гипоциклоиды

Гипоциклоида

```
# задаём оси координат:  
plt=plot(5,xlim=(-4,4),ylim=(-4,4), c=:red, aspect_ratio=1, legend=false, framestyle=:origin)  
plot!(plt, X,Y, c=:blue, legend=false)
```

✓ 1.4s

Julia

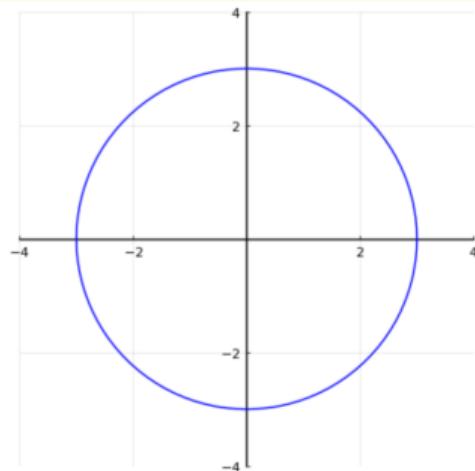


Рис. 32: Большая окружность гипоциклоиды

Гипоциклоида

Для частичного построения гипоциклоиды будем менять параметр t :

```
i = 50
t = θ[1:i]
# гипоциклоида:
x = r_1*(k-1)*cos.(t) + r_1*cos.((k-1)*t)
y = r_1*(k-1)*sin.(t) - r_1*sin.((k-1)*t)
plot!(x,y, c=:red)
```

✓ 1.9s

Julia

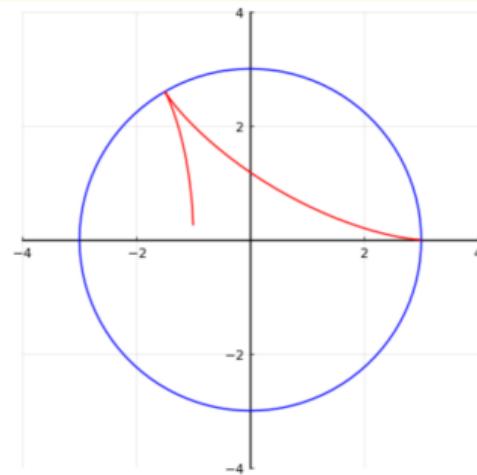


Рис. 33: Половина пути гипоциклоиды

Гипоциклоида

Добавляем малую окружность гипоциклоиды:

```
# малая окружность:  
xc = r_1*(k-1)*cos(t[end]) .+ r_1*cos.(θ)  
yc = r_1*(k-1)*sin(t[end]) .+ r_1*sin.(θ)  
plot!(xc,yc,c=:black)
```

✓ 1.6s

Julia

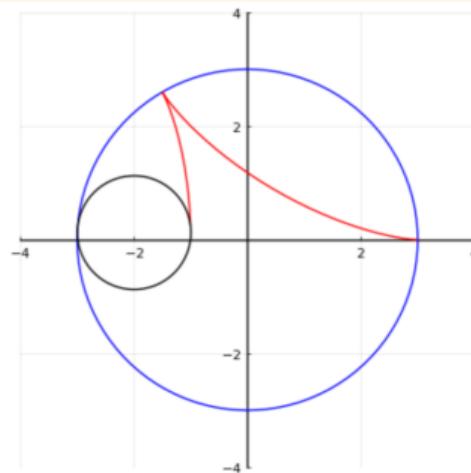


Рис. 34: Малая окружность гипоциклоиды

Гипоциклоида

Добавим радиус для малой окружности:

```
# радиус малой окружности:  
x1 = transpose([r_1*(k-1)*cos(t[end]) x[end]])  
y1 = transpose([r_1*(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)
```

✓ 2.9s

Julia

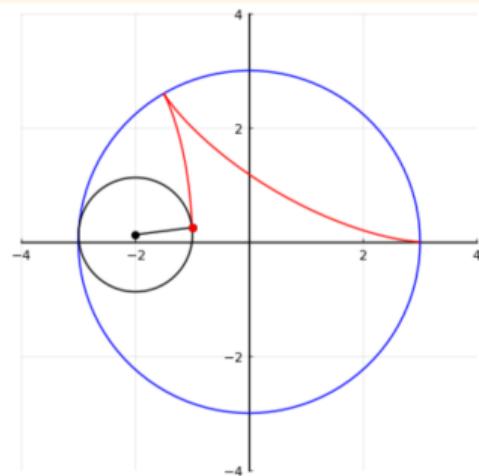


Рис. 35: Малая окружность гипоциклоиды с добавлением радиуса

Errorbars

В исследованиях часто требуется изобразить графики погрешностей измерения.

```
# подключение пакета Statistics:  
import Pkg  
Pkg.add("Statistics")  
using Statistics  
✓ 35.2s  
  
Resolving package versions...  
No Changes to `C:\Users\angelina\.julia\environments\v1.11\Project.toml'  
No Changes to `C:\Users\angelina\.julia\environments\v1.11\Manifest.toml'  
  
Julia  
  
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)  
✓ 0.1s  
  
6-element Vector{Float64}:  
0.6198064213930023  
0.30990321069658012  
0.1549516053482506  
0.0774758026741253  
0.03873790133706265  
0.019368950668531323  
  
Julia
```

Рис. 36: Зададим исходные значения

Errorbars

```
plot(y,
      ylims = (-1,1),
      xlabel="Axis·x",
      ylabel="Axis·y",
      title="График·исходных·значений",
      label="Исходные·значения"
    )
✓ 1.7s
```

Julia

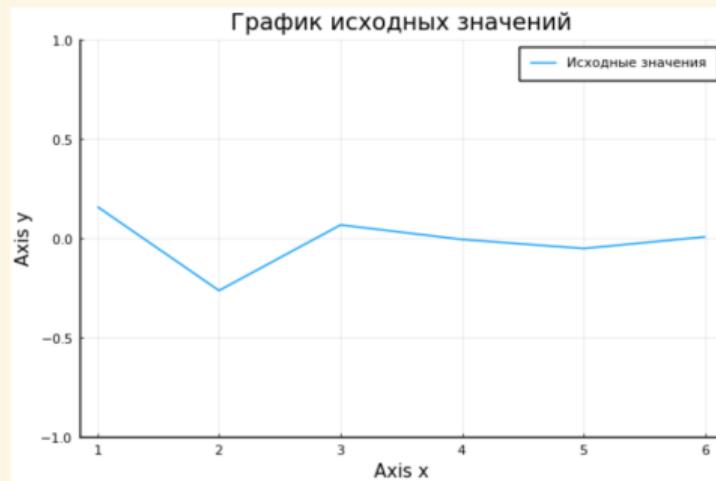


Рис. 37: График исходных значений

Errorbars

```
plot(y,
    ylims = (-1,1),
    err = errs,
    xlabel="Axis x",
    ylabel="Axis y",
    title="График исходных значений с отклонениями",
    label="Исходные значения с отклонениями"
)
✓ 8.3s
```

Julia

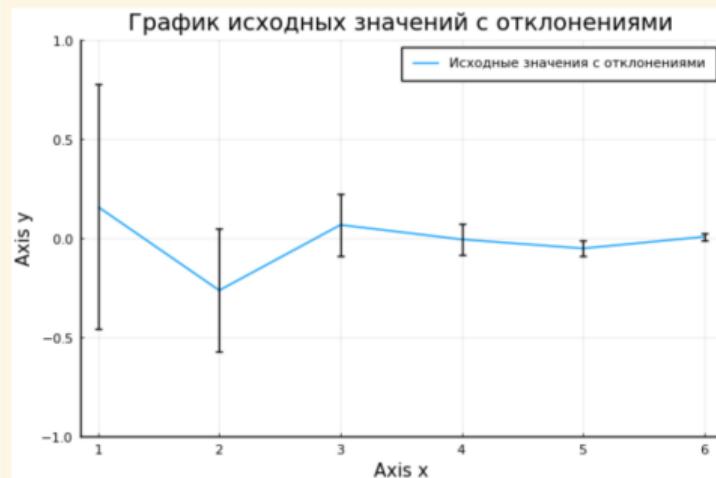


Рис. 38: График исходных значений с отклонениями

Errorbars

```
plot(y, 1:length(y),
      xerr = errs,
      marker = stroke(3,:orange),
      xlabel="Axis·x",
      ylabel="Axis·y",
      title="Поворот·графика",
      label="Исходные·значения·с·отклонениями"
    )
✓ 2.8s
```

Julia

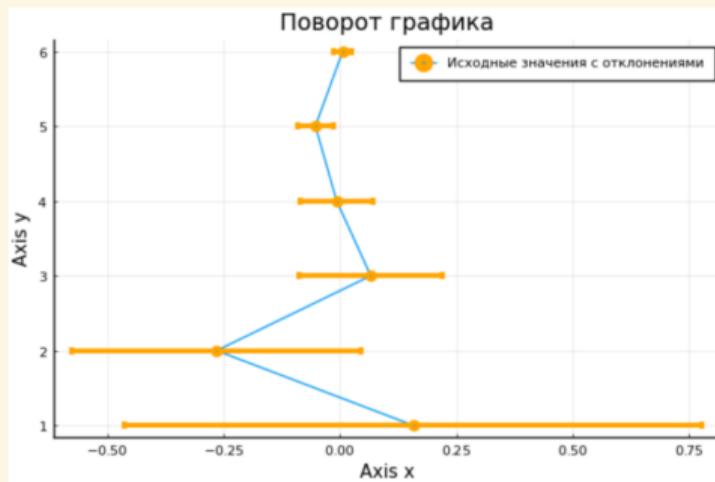


Рис. 39: Поворот графика

Errorbars

```
plot(y,
    ribbon=errs,
    fill=:cyan,
    xlabel="Axis x",
    ylabel="Axis y",
    title="Заполнение цветом",
    label="Исходные значения с отклонениями"
)
✓ 1.3s
```

Julia

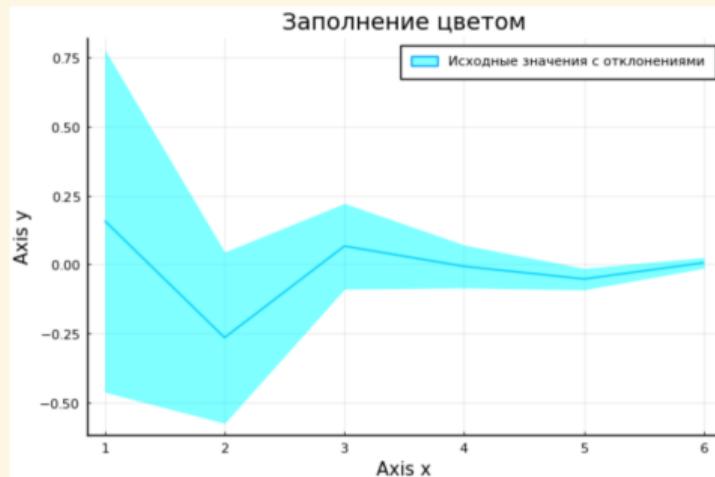


Рис. 40: Заполнение цветом

Errorbars

```
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),
      xlabel="Axis x",
      ylabel="Axis y",
      title="График ошибок по двум осям",
      label="Исходные значения с отклонениями"
)
```

✓ 2.6s

Julia

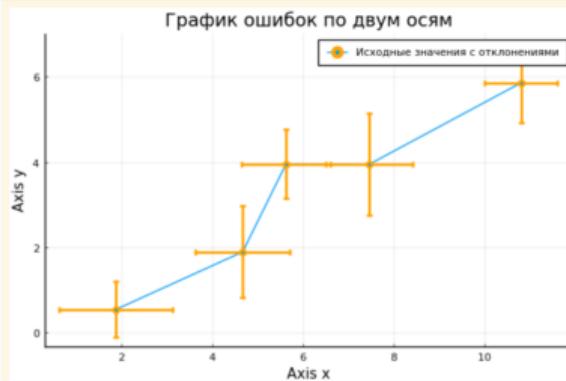


Рис. 41: График ошибок по двум осям

Errorbars

```
plot(x, y,
      xerr = (0.5xerr,2xerr),
      yerr = (0.5yerr,2yerr),
      marker = stroke(2, :orange),
      xlabel="Axis·x",
      ylabel="Axis·y",
      title="График асимметричных ошибок по двум осям",
      label="Исходные значения с отклонениями"
    )
```

✓ 1.8s

Julia

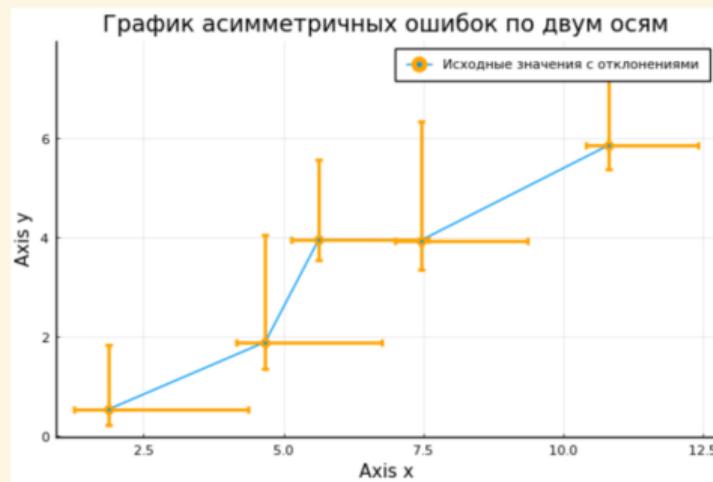


Рис. 42: График асимметричных ошибок по двум осям

Использование пакета Distributions

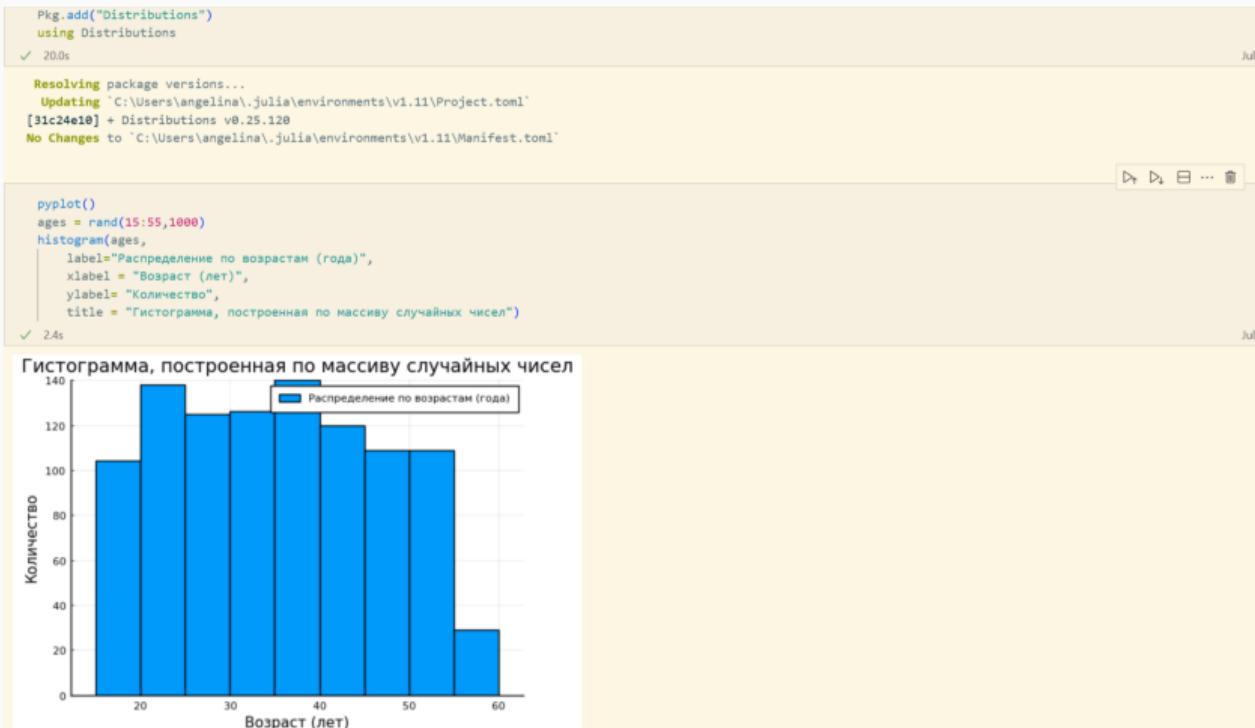


Рис. 43: Гистограмма, построенная по массиву случайных чисел

Использование пакета Distributions

```
d=Normal(35.0,10.0)
ages = rand(d,1000)
histogram(
    ages,
    label="Распределение по возрастам (года)",
    xlabel = "Возраст (лет)",
    ylabel= "Количество",
    title = " Гистограмма нормального распределения"
)
```

✓ 1.6s

Julia

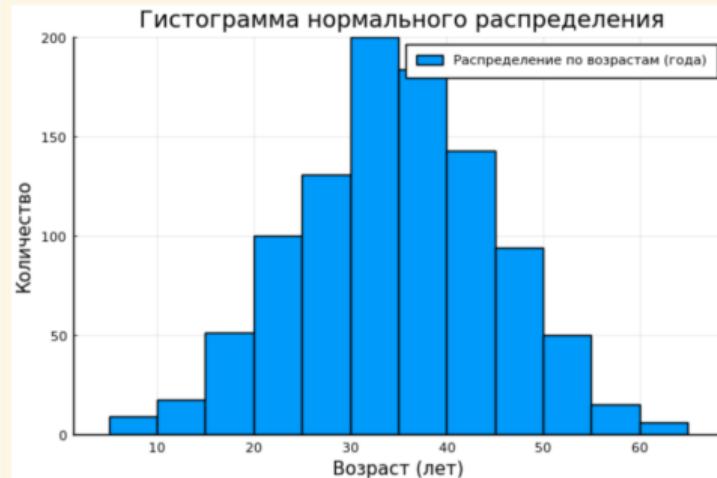


Рис. 44: Гистограмма нормального распределения

Подграфики

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

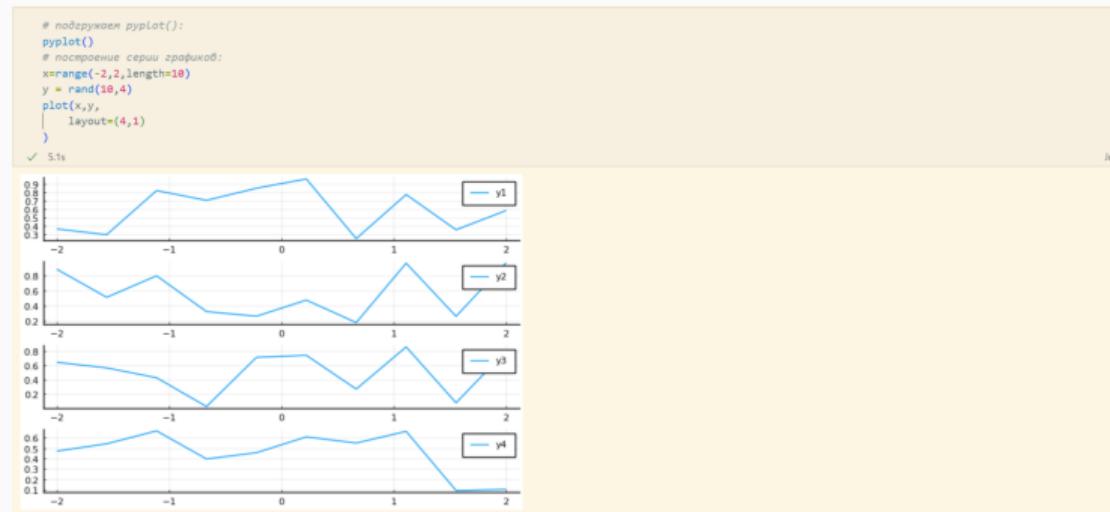


Рис. 45: Серия из 4-х графиков в ряд

Подграфики

Для автоматического вычисления сетки необходимо передать layout целое число:



Рис. 46: Серия из 4-х графиков в сетке

Подграфики

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

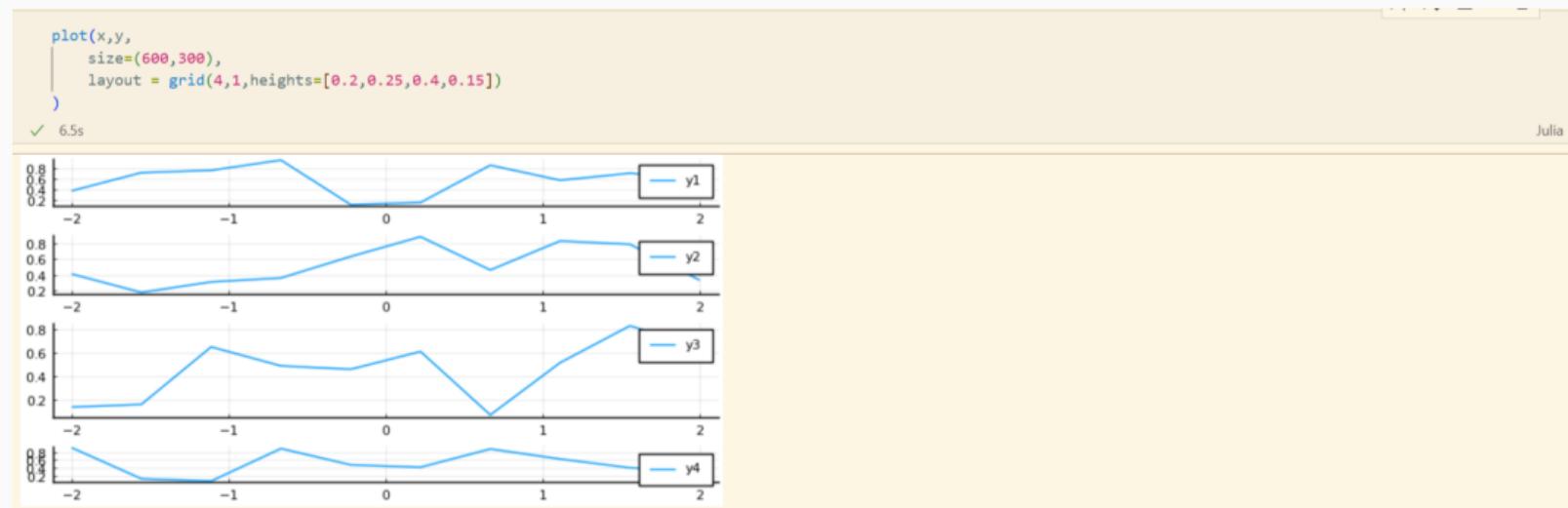


Рис. 47: Серия из 4-х графиков разной высоты в ряд

Подграфики

Можно сгенерировать отдельные графики и объединить их в один, например, в сетке 2×2 :

```
# график в виде линий:  
p1 = plot(x,y)  
# график в виде точек:  
p2 = scatter(x,y)  
# график в виде линий с оформлением:  
p3 = plot(x,y[:,1:2], xlabel="Labelled plot of two columns", lw=2, title="Wide lines")  
# 4 гистограммы:  
p4 = histogram(x,y)  
plot(  
    p1,p2,p3,p4,  
    layout=(2,2),  
    legend=false,  
    size=(800,600),  
    background_color = :ivory  
)  
✓ 16.4s
```

Julia

Рис. 48: Объединение нескольких графиков в одной сетке

Подграфики

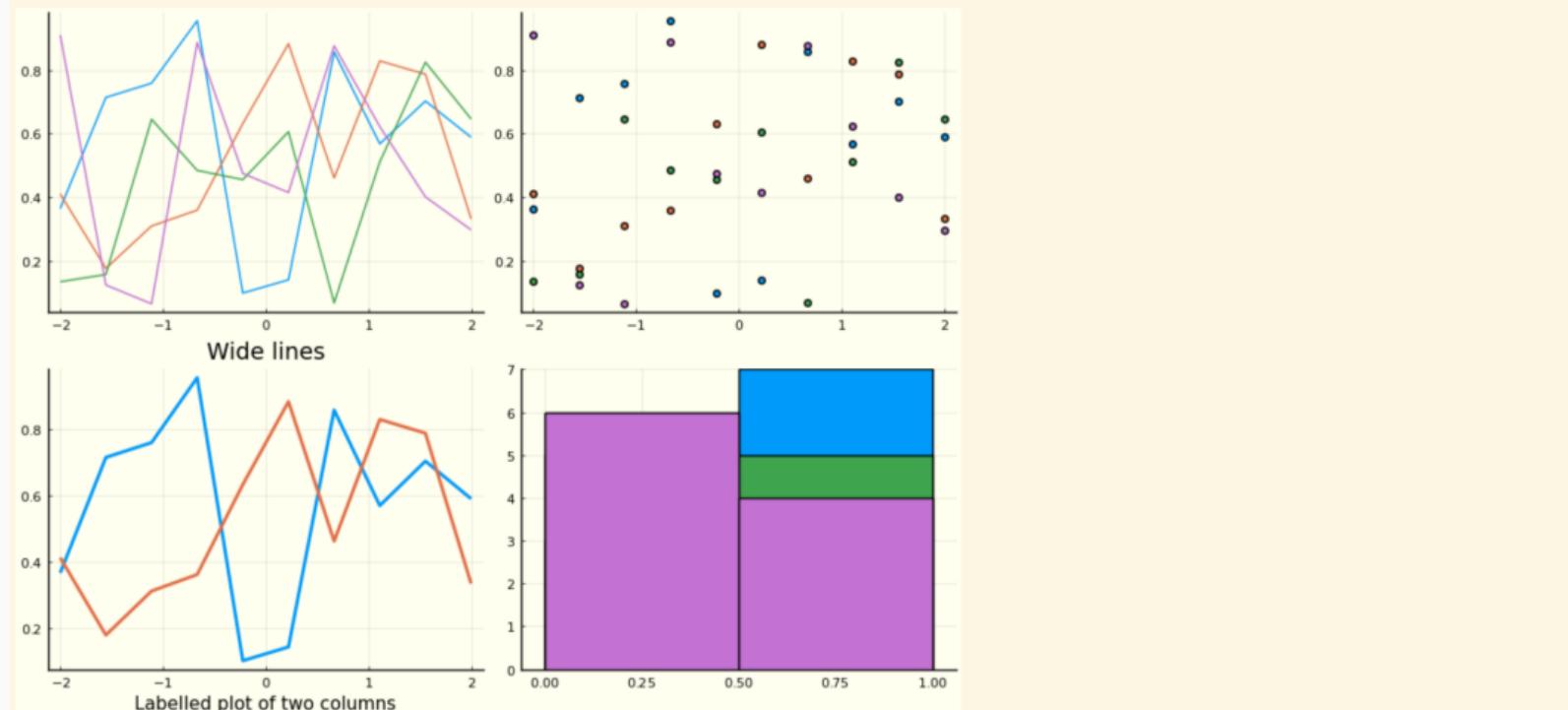


Рис. 49: Объединение нескольких графиков в одной сетке

Подграфики

```
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
    )
```

✓ 9.3s

Julia

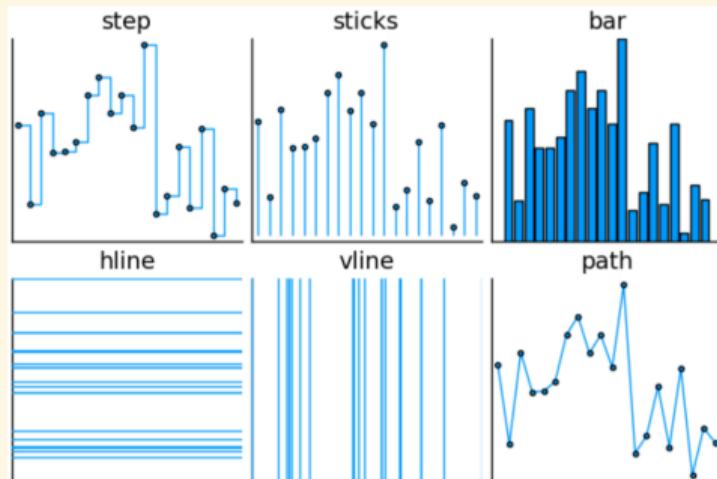


Рис. 50: Разнообразные варианты представления данных

Подграфики

```
l = @layout [ a{@.3w} [grid(3,3)
b{@.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)
)
✓ 18.0s
```

Julia

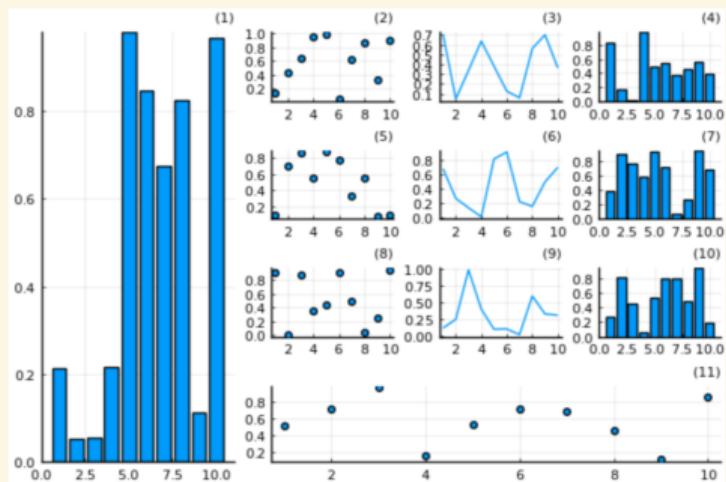


Рис. 51: Демонстрация применения сложного макета для построения графиков

Самостоятельное выполнение

Самостоятельное выполнение

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:

```
x = collect(range(0, 2*pi, length = 50))
f(x) = sin(x)
types = [:line, :scatter, :step, :sticks, :bar, :hline, :vline, :path]
titles = ["line" "scatter" "step" "sticks" "bar" "hline" "vline" "path"]
plot(x, f, st=types, title=titles, layout=(2, 4), size=(1800, 600))
```

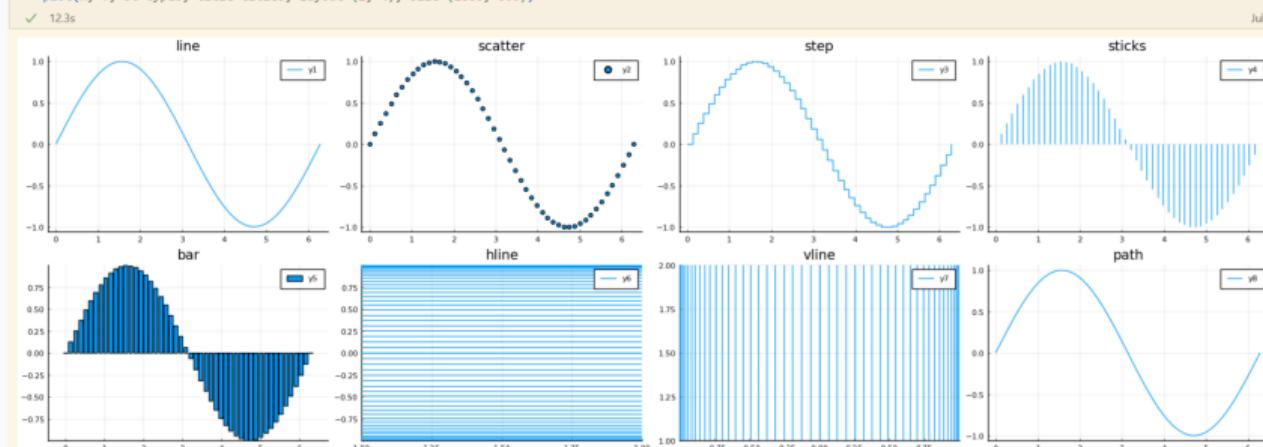


Рис. 52: Решение задания №1

Самостоятельное выполнение

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:

```
titles = ["solid" "dash" "dot" "dashdot"]
p1 = Plots.plot(x, f, linestyle=:solid)
p2 = Plots.plot(x, f, linestyle=:dash)
p3 = Plots.plot(x, f, linestyle=:dot)
p4 = Plots.plot(x, f, linestyle=:dashdot)
Plots.plot(p1, p2, p3, p4, title=titles, layout=(2, 2))
```

✓ 28.4s

Julia

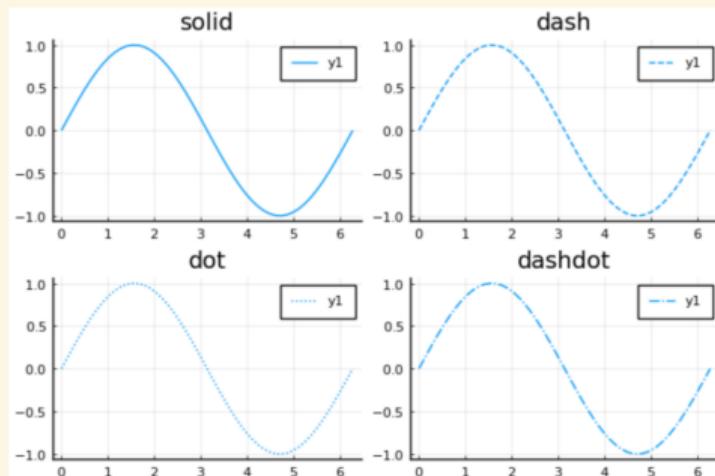


Рис. 53: Решение задания №2

Самостоятельное выполнение

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:

```
f(x) = pi * x^2 * log(x)
x = 0.1:0.01:6
plot(x, f.(x),
    color = :red,                      # Цвет графика
    label = "y(x) = pi*x^2*ln(x)",      # Легенда
    xlabel = "x",                        # Подпись оси X
    ylabel = "y(x) = pi*x^2*ln(x)",      # Подпись оси Y
    framestyle = :box,                   # Стиль рамки
    xticks = 0:0.5:6,                    # Частота отметок на оси X
    yticks = -50:50:200,                 # Частота отметок на оси Y
    bordercolor = :green)                # Цвет рамки
```

✓ 6.1s

Julia

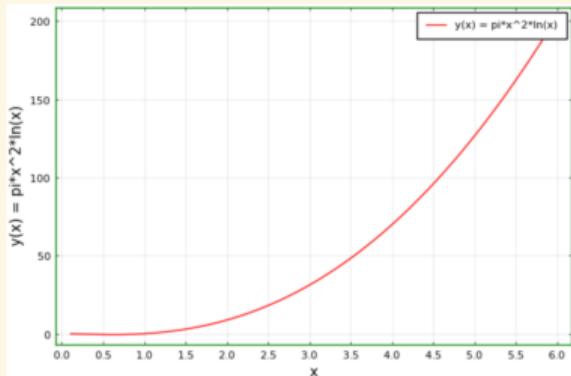


Рис. 54: Решение задания №3

Самостоятельное выполнение

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:

```
x = [-2, -1, 0, 1, 2]
f(x) = x^3-3x
p1 = plot(x, f, lines=:solid)
p2 = plot(x, f, lines=:dash)
p3 = plot(x, f, lines=:dot)
p4 = plot(x, f, lines=:dashdot)
plot(p1, p2, p3, p4, layout=(2, 2))

savefig("figure_chemanova.png")
```

Рис. 55: Решение задания №4

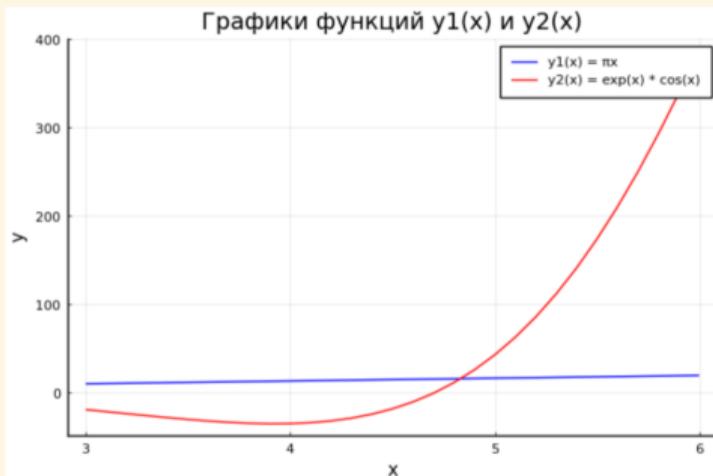
Самостоятельное выполнение

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:

```
x = 3:0.1:6
y1(x) = pi * x
y2(x) = exp(x) .* cos.(x)
plot(x, y1(x), label="y1(x) = pi x", color=:blue, xlabel="x", ylabel="y", title="Графики функций y1(x) и y2(x)")
plot!(x, y2(x), label="y2(x) = exp(x) * cos(x)", color=:red)
```

✓ 2.4s

Julia



Самостоятельное выполнение

```
p1 = plot(x, y1, color="blue")
p2 = plot(x, y2, color="red")
plot(p1, p2, layout=(1, 2))
```

✓ 11.4s

Julia

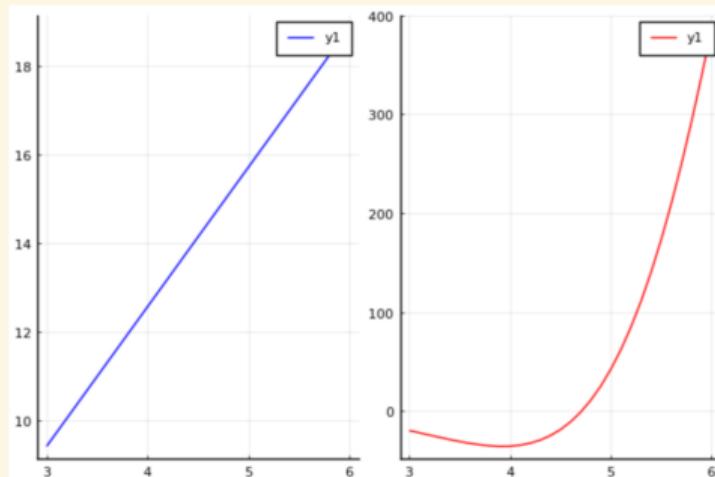


Рис. 57: Решение задания №5

Самостоятельное выполнение

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

```
time = 0:2:24
osadki = [15, 21, 40, 32, 65, 12, 67, 34, 56, 45, 23, 18]
osadki_error = [mean(sd*randn(n)) for sd in osadki]
plot(time, osadki, label="Осадки", color=:pink, yerr=osadki_error,
      marker=:o, xlabel="Время", ylabel="Осадки",
      title="Экспериментальные данные с ошибкой измерения")
```

✓ 3.3s

Julia

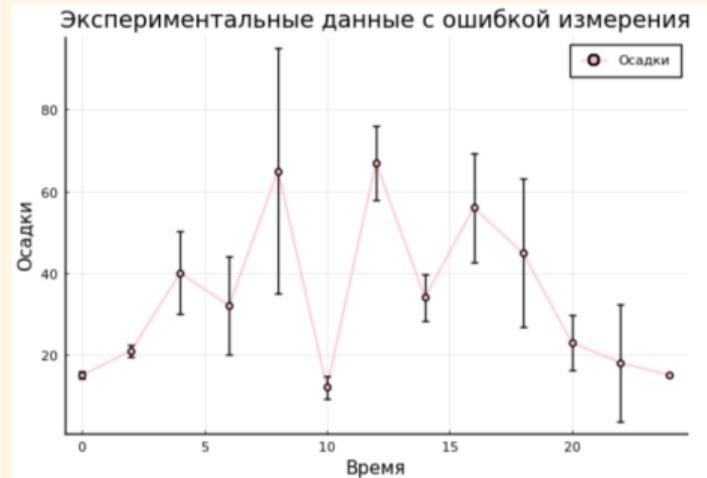


Рис. 58: Решение задания №6

Самостоятельное выполнение

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

```
x = rand(1:10, 20)
y = rand(1:10, 20)
scatter(x, y, label="Случайные данные", color="pink", marker=:o,
       xlabel="x", ylabel="y", title="Точечный график случайных данных")
✓ 10.3s
```

Julia

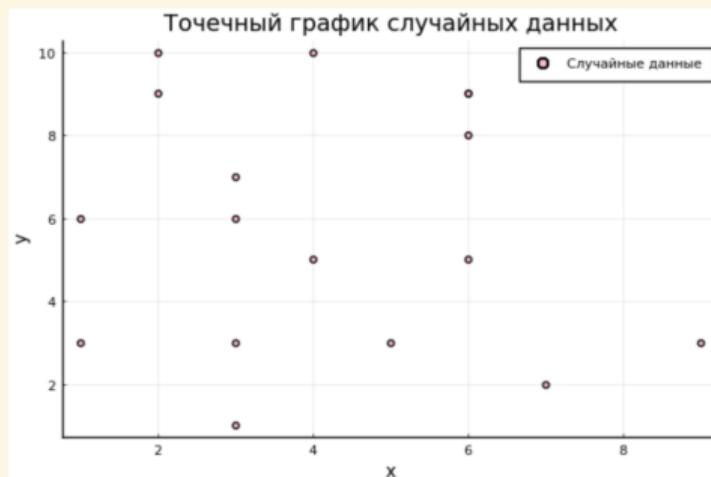


Рис. 59: Решение задания №7

Самостоятельное выполнение

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

```
x = rand(1:10, 20)
y = rand(1:10, 20)
z = rand(1:10, 20)
scatter(x, y, z, label="Случайные данные", color="red", marker=:o,
| xlabel="x", ylabel="y", zlabel="z", title="3-мерный точечный график случайных данных")
```

✓ 2.6s

Julia

3-мерный точечный график случайных данных

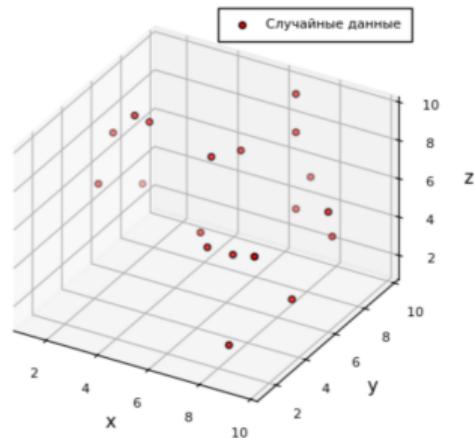


Рис. 60: Решение задания №8

Самостоятельное выполнение

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

```
x = 0:0.1:10
y_vals = sin.(x)
anim = @animate for i in 1:length(x)
    plot(x[1:i], y_vals[1:i], label="Синусоиды", color=:blue, linewidth=2)
    xlabel!("x")
    ylabel!("sin(x)")
    title!("Построение синусоиды")
end
gif(anim, "sin_wave_animation.gif", fps=10)
```

✓ 42.8s
Info: Saved animation to <c:\Users\angelina\Downloads\sin wave animation.gif>
└ @ Plots C:\Users\angelina\.julia\packages\Plots\EciLi\src\animation.jl:156



Самостоятельное выполнение

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

```
rr = 1
k = 10
n=100
theta = collect(0:2*pi/100:2*pi+2*pi/100)
X = rr*k*cos.(theta)
Y = rr*k*sin.(theta)
anim = @animate for i in 1:n
    plt=plot(5, xlim=(-10,10), ylim=(-10, 10), c="red", aspect_ratio=1, legend=false, framestyle="origin")
    plot!(plt, X, Y, c="blue", legend=false)
    t = theta[1:i]
    x = rr*(k-1)*cos.(t) + rr*cos.((k-1)*t)
    y = rr*(k-1)*sin.(t) - rr*sin.((k-1)*t)
    plot!(x, y, c="red")
    xc = rr*(k-1)*cos(t[end]) .+ rr*cos.(theta)
    yc = rr*(k-1)*sin(t[end]) .+ rr*sin.(theta)
    plot!(xc, yc, c="black")
    xl = transpose([rr*(k-1)*cos(t[end]) x[end]])
    yl = transpose([rr*(k-1)*sin(t[end]) y[end]])
    plot!(xl, yl, makershape="circle", markersize=4, c="black")
    scatter!([x[end]], [y[end]], c="red", markerstrokecolor="red")
end
gif(anim, "hypocycloid.gif")
```

✓ 35.8s

Julia

Рис. 62: Решение задания №10

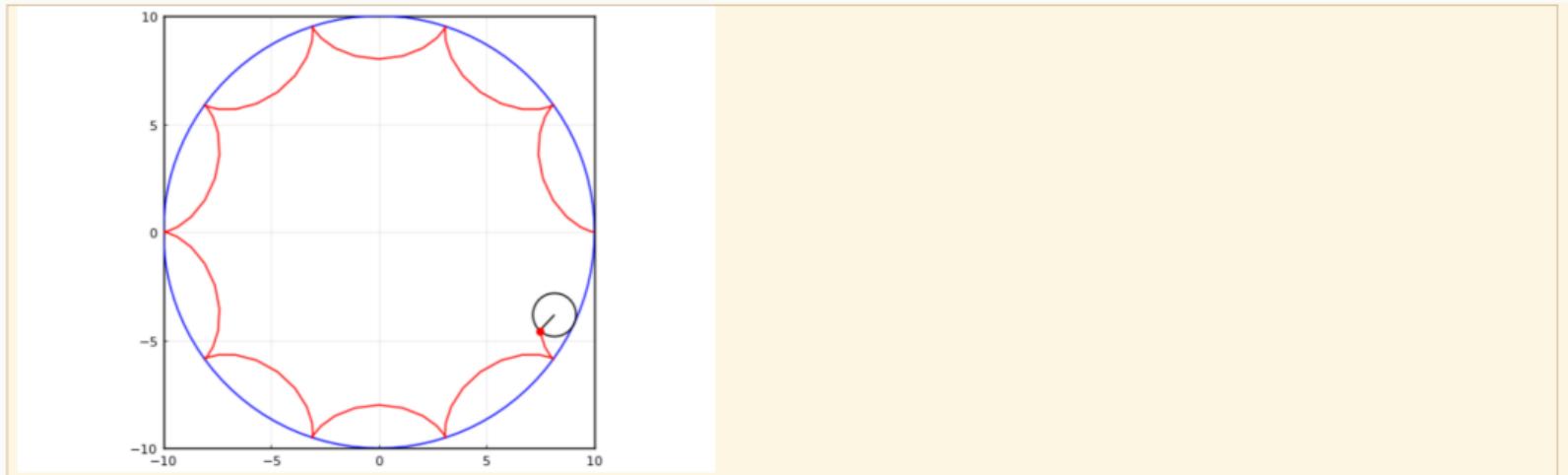


Рис. 63: Решение задания №10

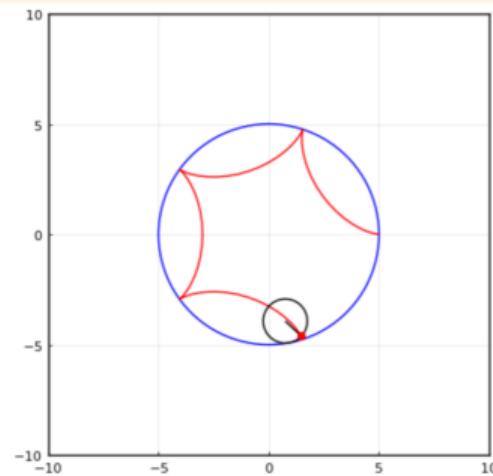


Рис. 64: Решение задания №10

Самостоятельное выполнение

```
rr = 1
k = 19//3
n=100
theta = collect(0:2*pi/100:2*pi+2*pi/100)
X = rr*k*cos.(theta)
Y = rr*k*sin.(theta)
anim = @animate for i in 1:n
    plt=plot(5, xlim=(-10,10), ylim=(-10, 10), c="red", aspect_ratio=1, legend=false, framestyle="origin")
    plot!(plt, X, Y, c="blue", legend=false)
    t = theta[1:i]
    x = rr*(k-1)*cos.(t) + rr*cos.((k-1)*t)
    y = rr*(k-1)*sin.(t) - rr*sin.((k-1)*t)
    plot!(x, y, c="red")
    xc = rr*(k-1)*cos(t[end]) .+ rr*cos.(theta)
    yc = rr*(k-1)*sin(t[end]) .+ rr*sin.(theta)
    plot!(xc, yc, c="black")
    xl = transpose([rr*(k-1)*cos(t[end]) x[end]])
    yl = transpose([rr*(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")
```

✓ 47.8s

Julia

Рис. 65: Решение задания №10

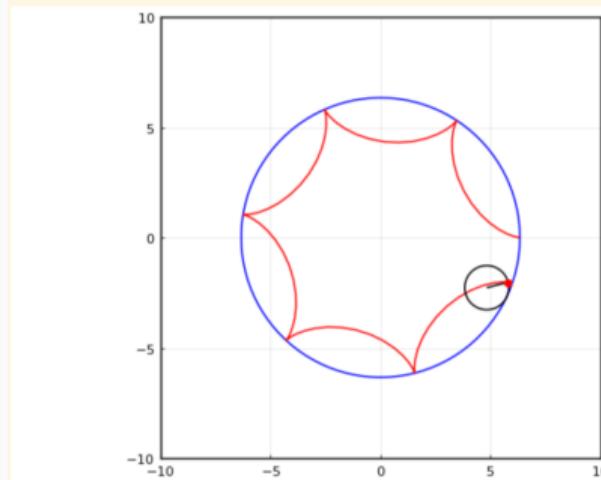


Рис. 66: Решение задания №10



Рис. 67: Решение задания №10

Самостоятельное выполнение

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

```
rr = 1
k = 10
n=100
theta = collect(0:2*pi/100:2*pi+2*pi/100)
X = rr*k*cos.(theta)
Y = rr*k*sin.(theta)
anim = @animate for i in 1:n
    plt=plot(5, xlim=(-10,10), ylim=(-10, 10), c="red", aspect_ratio=1, legend=false, framestyle="origin")
    plot!(plt, X, Y, c="blue", legend=false)
    t = theta[1:i]
    x = rr*(k-1)*cos.(t) - rr*cos.((k-1)*t)
    y = rr*(k-1)*sin.(t) - rr*sin.((k-1)*t)
    plot!(x, y, c="red")
    xc = rr*(k-1)*cos(t[end]) .+ rr*cos.(theta)
    yc = rr*(k-1)*sin(t[end]) .+ rr*sin.(theta)
    plot!(xc, yc, c="black")
    xl = transpose([rr*(k-1)*cos(t[end]) x[end]])
    yl = transpose([rr*(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")
```

✓ 43.6s

Julia

Рис. 68: Решение задания №11

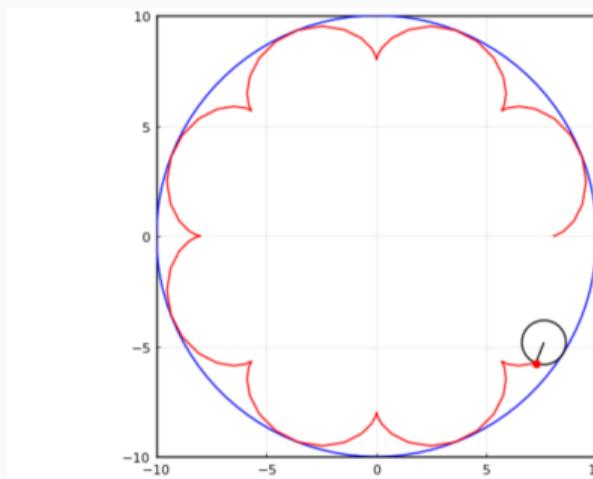


Рис. 69: Решение задания №11

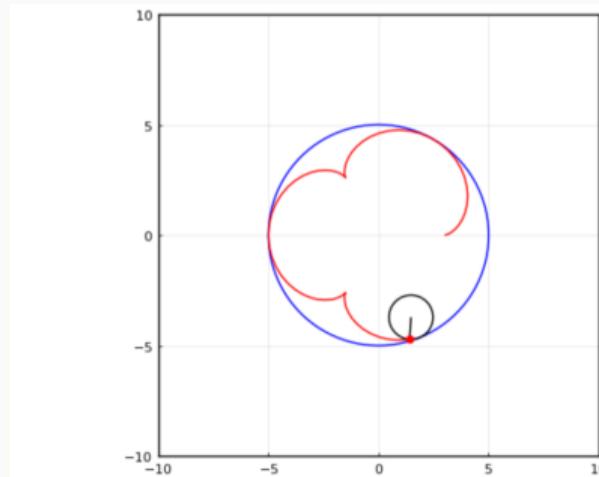


Рис. 70: Решение задания №11

Самостоятельное выполнение

```
rr = 1
k = 19//3
n=100
theta = collect(0:2*pi/100:2*pi+2*pi/100)
X = rr*k*cos.(theta)
Y = rr*k*sin.(theta)
anim = @animate for i in 1:n
    plt=plt(5, xlim=(-10,10), ylim=(-10, 10), c="red", aspect_ratio=1, legend=false, framestyle="origin")
    plot!(plt, X, Y, c="blue", legend=false)
    t = theta[1:i]
    x = rr*(k-1)*cos.(t) - rr*cos.((k-1)*t)
    y = rr*(k-1)*sin.(t) - rr*sin.((k-1)*t)
    plot!(x, y, c="red")
    xc = rr*(k-1)*cos(t[end]) .+ rr*cos.(theta)
    yc = rr*(k-1)*sin(t[end]) .+ rr*sin.(theta)
    plot!(xc, yc, c="black")
    xl = transpose([rr*(k-1)*cos(t[end]) x[end]])
    yl = transpose([rr*(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")
```

✓ 46.3s

Julia

Рис. 71: Решение задания №11

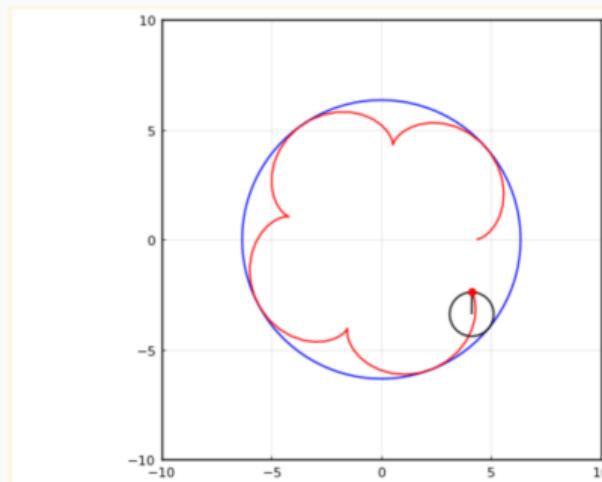


Рис. 72: Решение задания №11

Самостоятельное выполнение

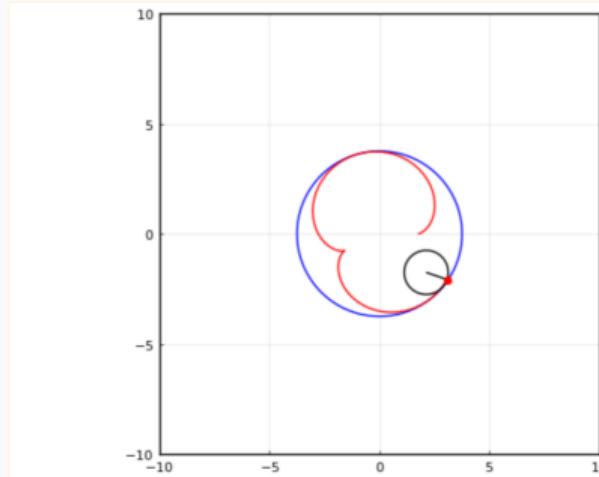


Рис. 73: Решение задания №11

Выводы

В результате выполнения данной лабораторной работы мы освоили синтаксис языка Julia для построения графиков.