

Лабораторная работа № 5

Построение графиков

Беличева Д. М.

Российский университет дружбы народов, Москва, Россия

Информация

- Беличева Дарья Михайловна
- студентка
- Российский университет дружбы народов
- 1032216453@pfur.ru
- <https://dmbelicheva.github.io/ru/>



Цель работы

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

Задание

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

Выполнение лабораторной работы

```
[2]: using Plots
[ Info: Precompiling IJuliaExt [2f4121a4-3b3a-5ce5-9c5e-1f2673ce168a]

[3]: # задание функции:
f(x) = (x.^2 + 6x - 9).exp(-4, x)

[3]: f (generic function with 1 method)

[100]: # генерирование массива значений x в диапазоне от -5 до 10 с шагом 0,1
# (авт. аванс через указание длины массива):
x = collect(range(-5, 10, length=101));

[21]: # указывается, что для построения графика используется gr()?
gr()

[21]: Plots.GHBackend()

[7]: # 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="blue")

[7]: A simple curve

[8]: # задание опций при построение графика
# (название кривой, подписка по ось, цвет линий)

```

Рис. 1: Основные пакеты для работы с графиками в Julia

Выполнение лабораторной работы



Рис. 2: Основные пакеты для работы с графиками в Julia

Выполнение лабораторной работы

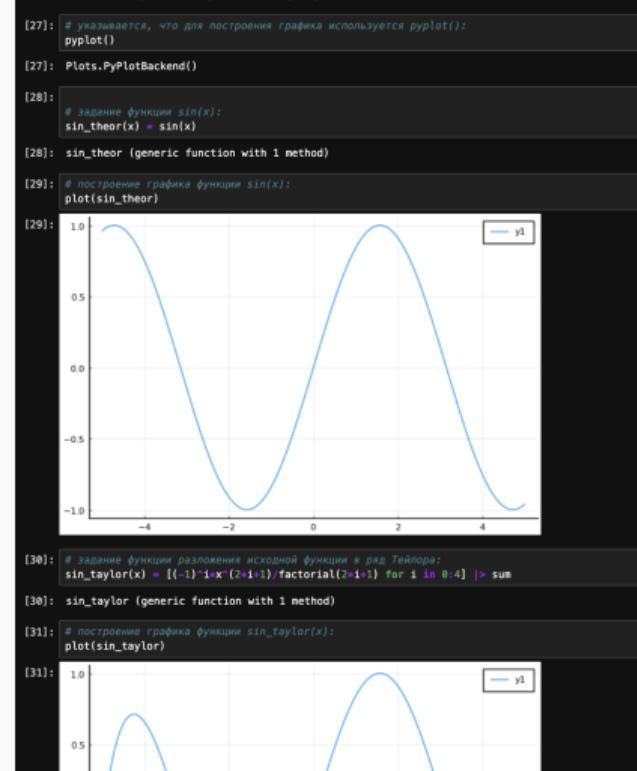


Рис. 3: Опции при построении графика

Выполнение лабораторной работы

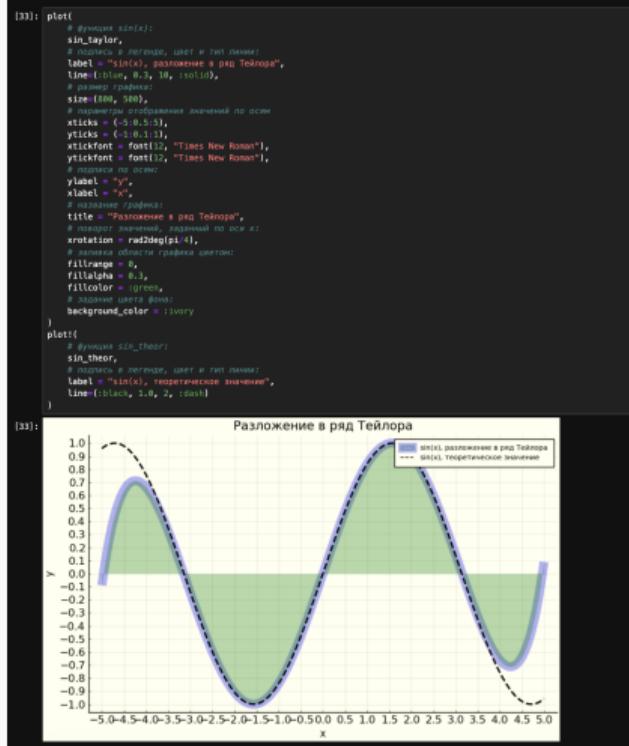


Рис. 4: Опции при построении графика

Выполнение лабораторной работы

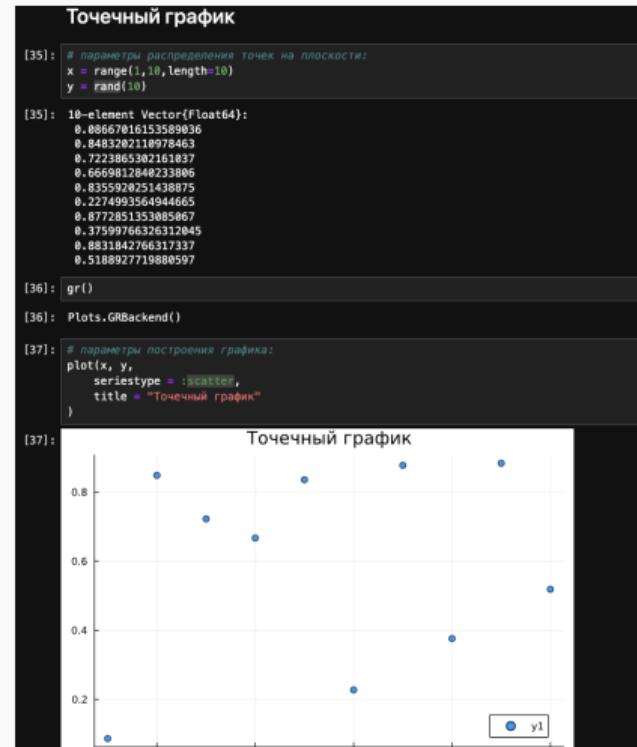


Рис. 5: Точечный график

Выполнение лабораторной работы

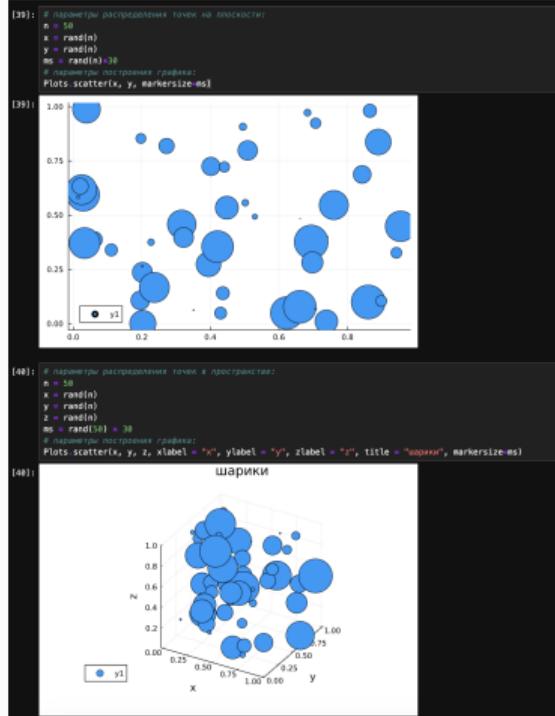


Рис. 6: Точечный график

Выполнение лабораторной работы

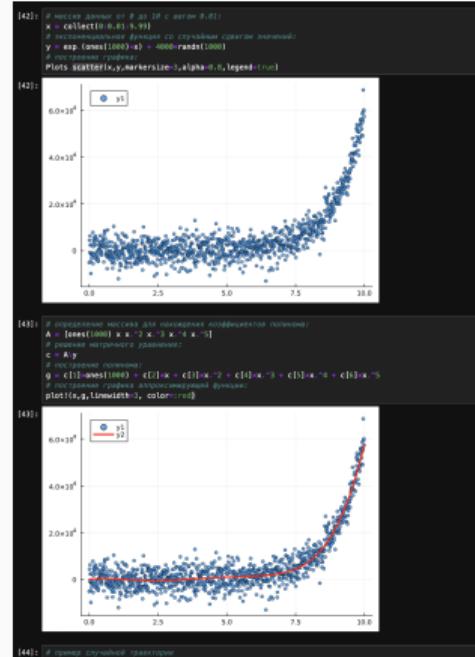


Рис. 7: Аппроксимация данных

Выполнение лабораторной работы

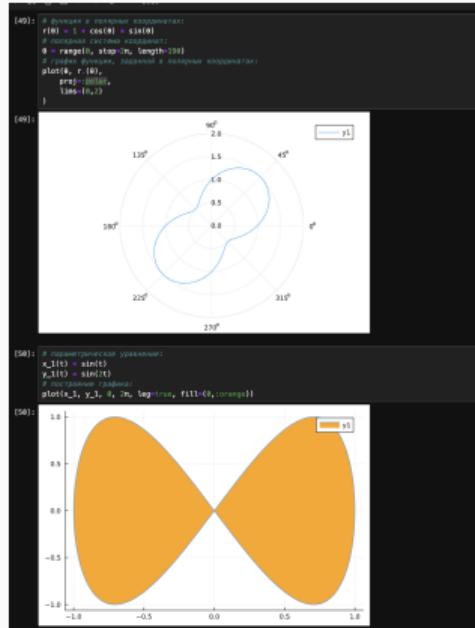


Рис. 8: Полярные координаты

Выполнение лабораторной работы

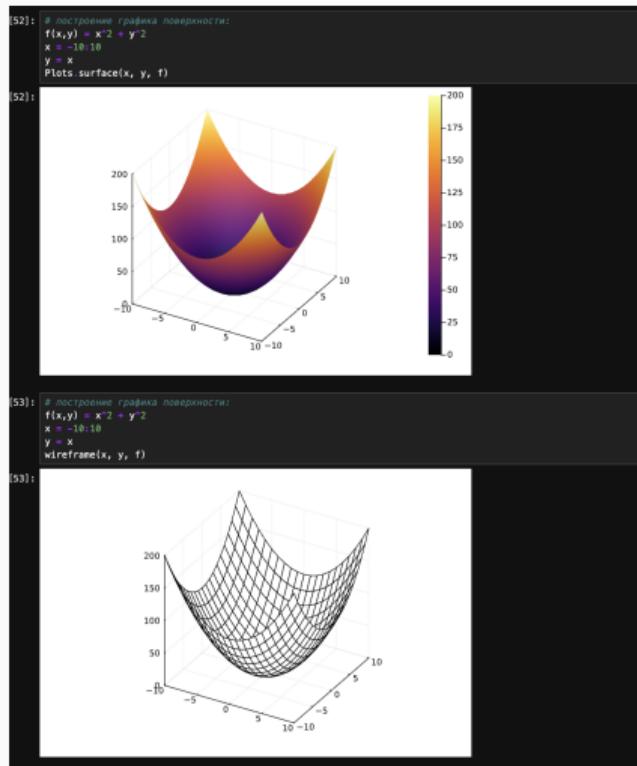


Рис. 9: График поверхности

Выполнение лабораторной работы

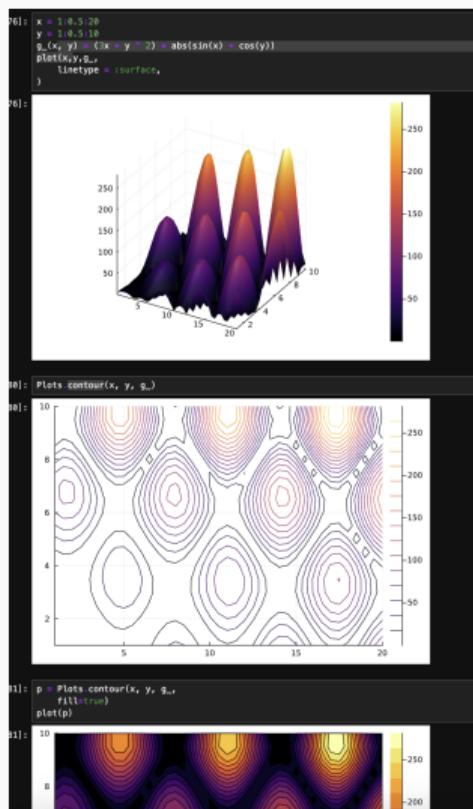


Рис. 10: Линии уровня

Выполнение лабораторной работы

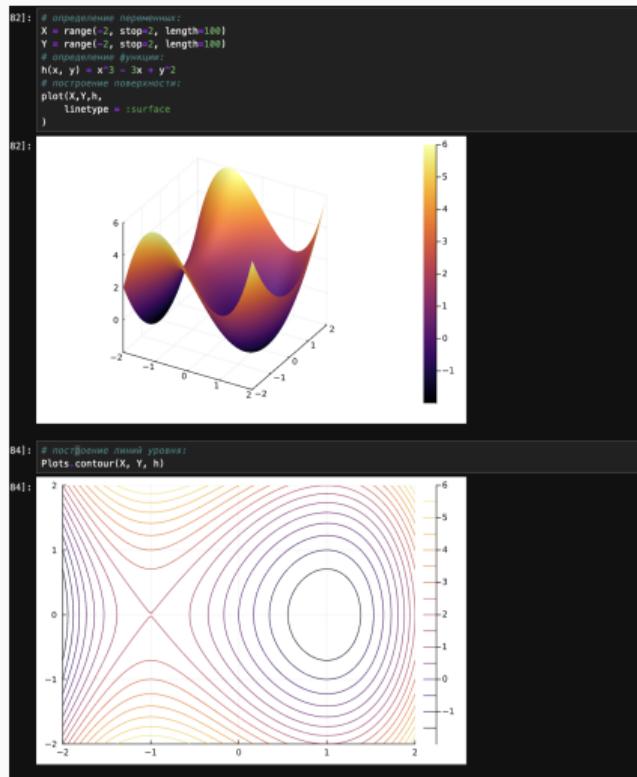


Рис. 11: Векторные поля

Выполнение лабораторной работы

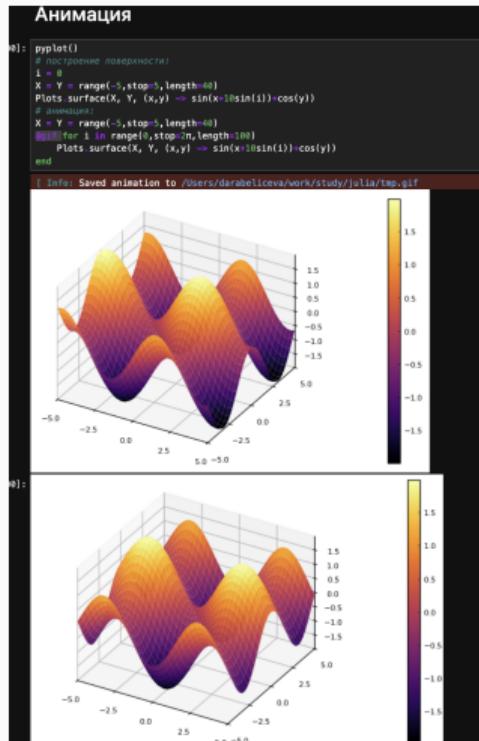


Рис. 12: Анимация

Выполнение лабораторной работы

```
!j: # радиус малой окружности
r_m = 3
# коэффициент для построения большой окружности
k = 3
# число отсчётов
n = 100
# массив значений угла θ:
# theta from θ to 2π (+ a little extra)
θ = collect(0:2π/100:2π+2π/100)
# массив значений координат
X = r_m*k*cos.(θ)
Y = r_m*k*sin.(θ)
# задаем оси координат
Plots.plot(plt, X, Y, ylim=[-4,4], color=:red, aspect_ratio=1, legend=false, framestyle=:none)
# малая окружность
Plots.plot(plt, X, Y, color=:blue, legend=false)
j = 50
t = θ[1:j]
# гипоциклоида:
x = r_m*(k-1)*cos.(t) + r_m*cos.((k-1)*t)
y = r_m*(k-1)*sin.(t) - r_m*sin.((k-1)*t)
Plots.plot(x, y, c=:red)
# малая окружность:
xc = r_m*(k-1)*cos(t[end]) .+ r_m*cos.(θ)
yc = r_m*(k-1)*sin(t[end]) .+ r_m*sin.(θ)
Plots.plot!(xc, yc, c=:black)
# радиус малой окружности
x1 = transpose([r_m*(k-1)*cos(t[end]) x[end]])
y1 = transpose([r_m*(k-1)*sin(t[end]) y[end]])
Plots.plot!(x1,y1,markershape=:circle,markersize=4,color=:black)
Plots.scatter([(x[end],y[end]),(xc,yc)],c=:red, markerstrokecolor=:red)
```

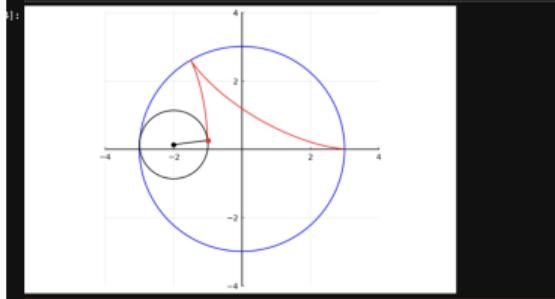


Рис. 13: Гипоциклоида

Выполнение лабораторной работы

```
# Importing the Statistics module
import Pkg
Pkg.add("Statistics")
using Statistics

# Resolving package versions...
# Warning: ~"/work/study/Project.toml"
# No changes to ~"/work/study/Manifest.toml"

#(136) sds = [1, 1/2, 1/4, 1/8, 1/16, 1/32]
#(137) x = 1d
y = [mean(sds[randn()]) for sd in sds]
err = 1.96 * sds ./ sqrt(n)
plotly,
glosses = [-1;1],
)

#(138) 1.0
#(139) plotly,
glosses = [-1;1],
err = errs
)

#(140) 1.0
#(141) plotly,
glosses = [-1;1],
err = errs
)
```

Рис. 14: Errorbars

Выполнение лабораторной работы

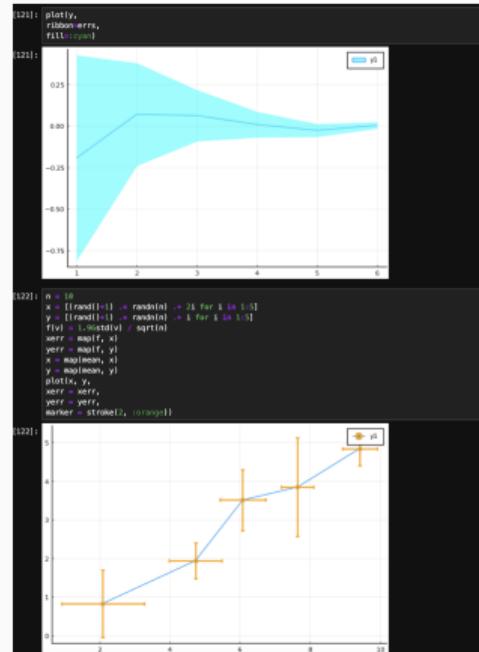
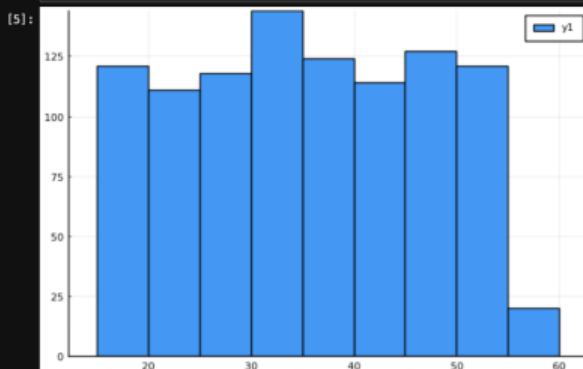


Рис. 15: Errorbars

Выполнение лабораторной работы

```
[1]: import Pkg  
Pkg.add("Distributions")  
using Distributions  
  
Resolving package versions...  
No Changes to '~/work/study/Project.toml'  
No Changes to '~/work/study/Manifest.toml'  
  
[9]: import Pkg; Pkg.add("PlotlyBase")  
  
Resolving package versions...  
Updating `~/work/study/Project.toml'  
Added [f5a5a4] + PlotlyBase v0.8.10  
No Changes to `~/work/study/Manifest.toml'  
  
[2]: using Plots  
  
[3]: using PyPlot  
  
[5]: pyplot()  
ages = rand(15:55,1000)  
Plots.histogram(ages)  
  
[5]:  


| Age Group (Center) | Frequency (y1) |
|--------------------|----------------|
| 15-20              | ~125           |
| 20-25              | ~115           |
| 25-30              | ~118           |
| 30-35              | ~135           |
| 35-40              | ~125           |
| 40-45              | ~115           |
| 45-50              | ~128           |
| 50-55              | ~120           |
| 55-60              | ~20            |

  
sys:3: UserWarning: No data for colormap mapping provided via 'c'. Parameters 'vmin', 'vmax' will be ignored.
```

Рис. 16: Использование пакета Distributions

Выполнение лабораторной работы

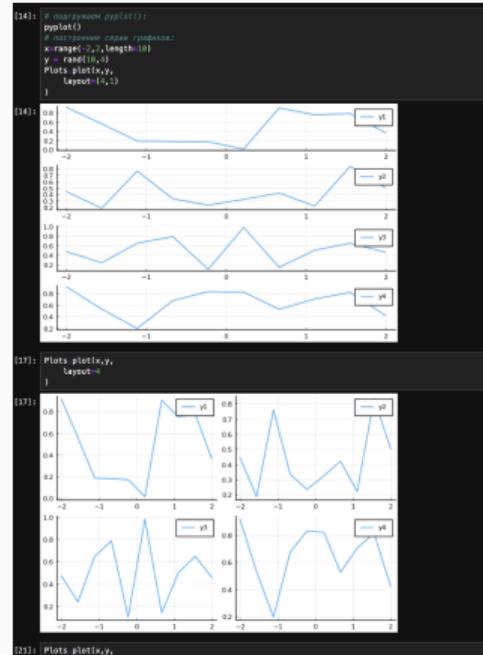


Рис. 17: Подграфики

Выполнение лабораторной работы

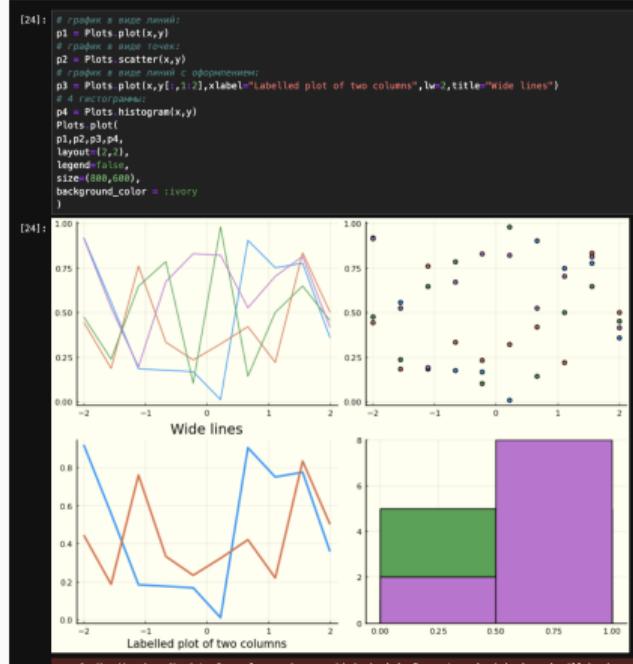


Рис. 18: Подграфики

Выполнение лабораторной работы



Рис. 19: Подграфики

Задания для самостоятельного выполнения

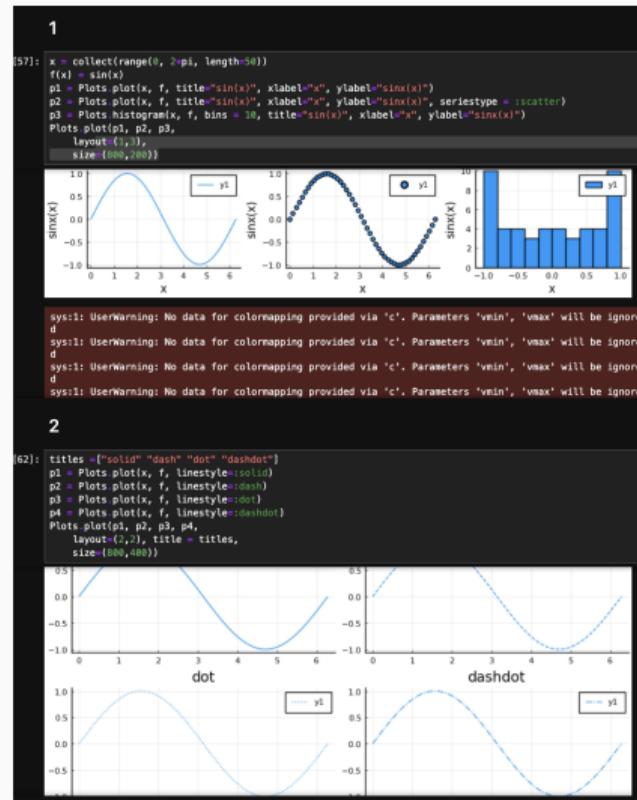


Рис. 20: Задание №1 и №2

Задания для самостоятельного выполнения



Рис. 21: Задание №3

Задания для самостоятельного выполнения

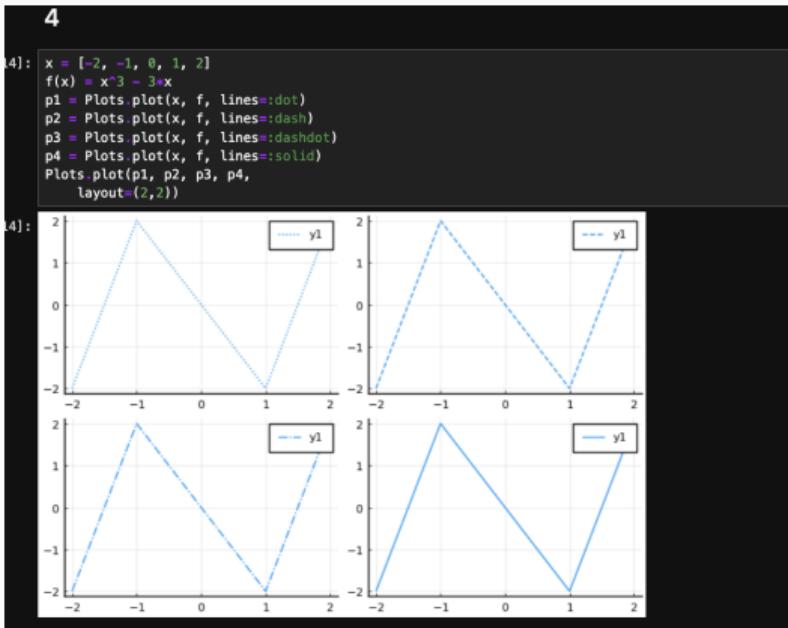


Рис. 22: Задание №4

Задания для самостоятельного выполнения

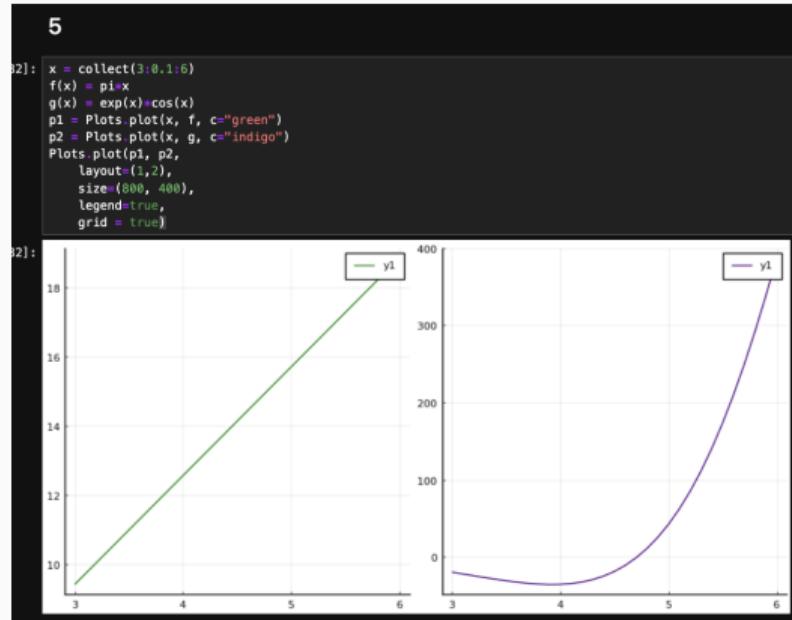


Рис. 23: Задание №5

Задания для самостоятельного выполнения

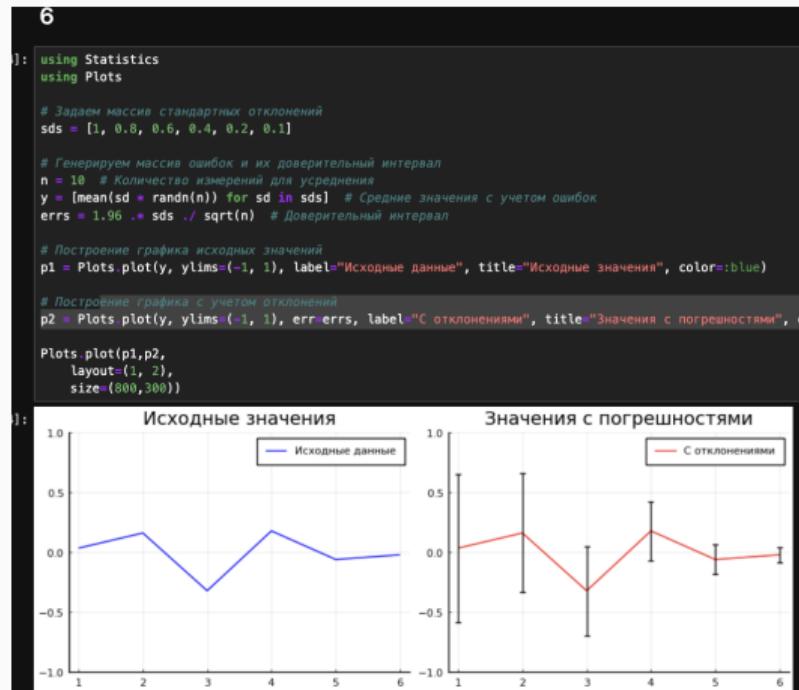


Рис. 24: Задание №6

Задания для самостоятельного выполнения

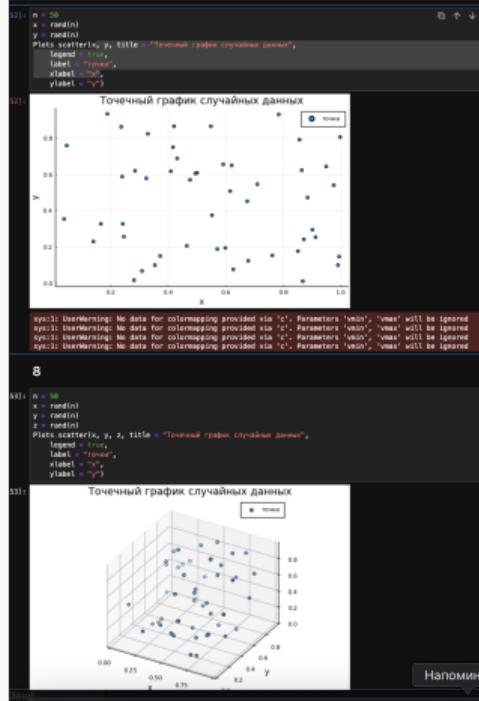


Рис. 25: Задание №7 и №8

Задания для самостоятельного выполнения

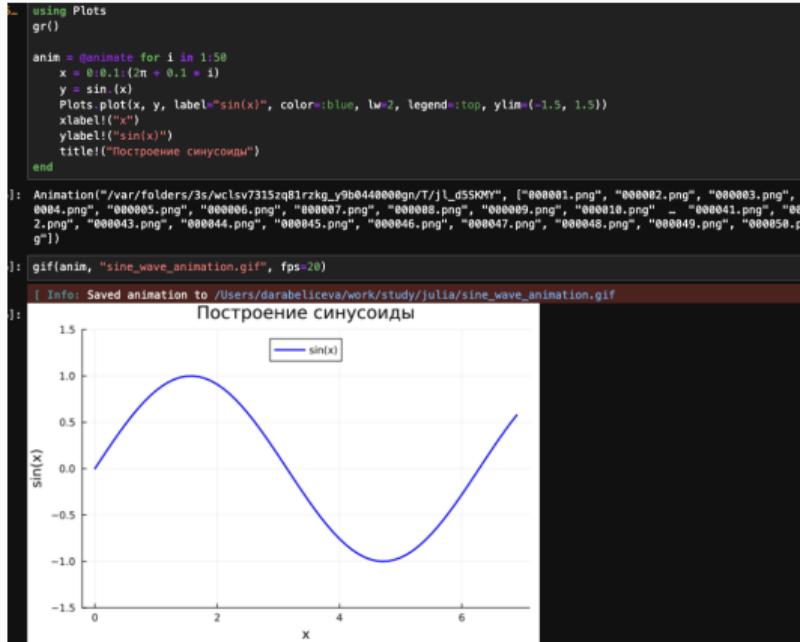


Рис. 26: Задание №9

Задания для самостоятельного выполнения

```
10
julia> using Plots

# Параметры
r_+ = 1 # Радиус малой окружности
k $\circ$  = 1 # значения модуля k (два целых и два рациональных)
n = 100 # Число кадров
θ = collect(0:2π/n:100:2π+2π/n:100) # Угловые значения для построения

anim = [animate for i in lin
    t = θ[1:i] # Постепенное увеличение углов для анимации

    # Большая окружность
    X = r_+ * k * cos.(θ)
    Y = r_+ * k * sin.(θ)

    # Гипоциклоиды
    x = r_+ * (k - 1) * cos.(t) .+ r_- * cos.((k - 1) .* t)
    y = r_+ * (k - 1) * sin.(t) .- r_- * sin.((k - 1) .* t)

    # Малая окружность
    xc = r_+ * (k - 1) * cos(t[end]) .+ r_- * cos.(θ)
    yc = r_+ * (k - 1) * sin(t[end]) .+ r_- * sin.(θ)

    # Построение графиков
    plt = Plots.plot(S, xlim=(-6, 6), ylim=(-6, 6), color=:red, aspect_ratio=1, legend=false, framestyle=:none)
    Plots.plot!(plt, X, Y, color=:blue, legend=false) # Большая окружность
    Plots.plot!(plt, x, y, color=:red) # Гипоциклоиды
    Plots.plot!(xc, yc, color=:black) # Малая окружность
    X = transpose(r_+ * (k - 1) * cos.(θ) .+ r_- * cos.(θ))
    Y = transpose(r_+ * (k - 1) * sin.(θ) .- r_- * sin.(θ))
    Plots.plot!(X, Y, markershape=:circle, markersize=4, color=:black)
    Plots.scatter!(xc, yc, color=:red, markerstrokecolor=:red)
end

# Сохранение анимации
gif(anim, "hypocycloid.gif", fps=20)

[ Info: Saved animation to /Users/darabeliceva/work/study/julia/hypocycloid.gif
```

Рис. 27: Задание №10

Задания для самостоятельного выполнения

```
1: using Plots

# Параметры
r_m = 1 # Радиус малой окружности
k = 33/4 # Значение модуля k (два целых и два дробных)
n = 180 # Число кадров
θ = collect(0:2π/180:(n-2)π/180) # Угловые значения для построения

anim = [animate for i in 1:n
        t = θ[i:i] # Постепенное увеличение углов для анимации

        # Большая окружность
        X = r_m * k * cos.(t)
        Y = r_m * k * sin.(t)

        # Гипотрохоиды
        x_h = r_m * (k - 1) * cos.(t) .+ r_m * cos.((k - 1) * t)
        y_h = r_m * (k - 1) * sin.(t) .- r_m * sin.((k - 1) * t)

        # Малая окружность
        xc = r_m * (k - 1) * cos(t[1end]) .+ r_m * cos.(t)
        yc = r_m * (k - 1) * sin(t[1end]) .+ r_m * sin.(t)

        # Построение графиков
        plot(X, Y, xlim=(-10, 10), ylim=(-10, 10), color=:red, aspect_ratio=1, legend=false, framestyle=:origin)
        Plots.plot(xc, yc, color=:red) # Большая окружность
        Plots.plot(xc, yc, color=:black) # Малая окружность
        Plots.plot(xc, yc, color=:black) # Гипотрохоиды
        Plots.plot(transpose([r_m * (k - 1) * cos(t[1end]) x[1end]]))
        yl = transpose([r_m * (k - 1) * sin(t[1end]) y[1end]])
        Plots.plot(xl, yl, markershape=:circle, markersize=4, color=:black)
        Plots.scatter!(xl, yl, color=:red, markerstrokecolor=:red)

    end

# Создание анимации
gif(anim, "hypocycloid.gif", fps=20)
```

[Info: Saved animation to /Users/darabeliceva/work/study/julia/hypocycloid.gif

Рис. 28: Задание №10

Задания для самостоятельного выполнения

```
julia> using Plots

# Параметры
r_0 = 1 # Радиус малой окружности
K = 30 # Значение модуля K (тогда целик и дао рациональны)
n = 100 # Число кадров
θ = collect(0:2π/100:2π+2π/100) # Угловые значения для построения

anim = @animate for i in 1:n
    t = 6i/11 # Постепенное увеличение углов для анимации

    # Большая окружность
    X = r_0 * k * cos.(θ)
    Y = r_0 * k * sin.(θ)

    # Эпциклоида
    x = r_0 * x + (k + 1) * cos.(t) - r_0 * x * cos.((k + 1) .* t)
    y = r_0 * y + (k + 1) * sin.(t) - r_0 * y * sin.((k + 1) .* t)

    # Малая окружность
    xc = r_0 * x + (k + 1) * cos(t[end]) - r_0 * x * cos.(θ)
    yc = r_0 * y + (k + 1) * sin(t[end]) - r_0 * y * sin.(θ)

    # Построение графиков
    plt = Plots.plot(x, y, xlim=(-7, 7), ylim=(-7, 7), color=:red, aspect_ratio=:equal, legend=false, framestyle=:origin)
    Plots.plot!(plt, X, Y, color=:blue, legend=false) # Большая окружность
    Plots.plot!(plt, x, y, color=:red) # Эпциклоида
    Plots.plot!(plt, xc, yc, color=:black) # Малая окружность
    xl = transpose([r_0 * x + (k + 1) * cos(t[n]) x[n]])
    yl = transpose([r_0 * y + (k + 1) * sin(t[n]) y[n]])
    Plots.plot!(xl, yl, markershape=:circle, markersize=4, color=:black)
    Plots.scatter!([(x[n]), (y[n])], color=:red, markerstrokecolor=:red)
    end

# Сохранение анимации
gif(anim, "epicycloid_K_3.gif", fps=20)

[ Info: Saved animation to /Users/daraboticeva/work/study/julia/epicycloid_K_3.gif
```

Рис. 29: Задание №11

Задания для самостоятельного выполнения

```
# coding: utf-8

# Параметры
r_m = 1 # Радиус малой окружности
k = 7/3 # Значение модуля k (две целых и два рациональных)
n = 100 # Число кадров
theta = collect(0.2*pi:100.2*pi:2*pi/100) # Угловые значения для построения

@animate for i in 1:n
    t = theta[i] # Постепенное увеличение углов для анимации

    # Большая окружность
    X = r_m + k * cos(theta)
    Y = r_m + k * sin(theta)

    # Эпциклоиды
    x = r_m + (k + 1) * cos(t) - r_m * cos((k + 1) * t)
    y = r_m + (k + 1) * sin(t) - r_m * sin((k + 1) * t)

    # Малая окружность
    xc = r_m + (k + 1) * cos(tend) - r_m * cos(theta)
    yc = r_m + (k + 1) * sin(tend) - r_m * sin(theta)

    # Построение графика
    plt = Plots.plot(x, ylim=(-10, 10), color=:red, aspect_ratio=1, legend=false, framestyle:none)
    Plots.plot!(plt, X, Y, color=:blue, legend=false) # Большая окружность
    Plots.plot!(plt, x, y, color=:red) # Эпциклоида
    Plots.plot!(xc, yc, color=:black) # Малая окружность
    xl = transpose([r_m + (k + 1) * cos(tend) xend])
    yl = transpose([r_m + (k + 1) * sin(tend) yend])
    Plots.plot!(xl, yl, markershape=:circle, markersize=4, color=:black)
    Plots.scatter!([xend], [yend], color=:red, markerstrokecolor=:red)
end

# Сохранение анимации
gifanim, "epicycloid.gif", fps=20

179: 1 Info: Saved animation to /Users/darbeliceva/work/study/julia/epicycloid.gif
```

Рис. 30: Задание №11

Выводы

В результате выполнения данной лабораторной работы я освоила синтаксис языка Julia для построения графиков.

Список литературы

1. JuliaLang [Электронный ресурс]. 2024 JuliaLang.org contributors.
URL:<https://julialang.org/>(дата обращения: 11.10.2024).
2. Julia 1.11 Documentation [Электронный ресурс]. 2024 JuliaLang.orgcontributors.
URL:<https://docs.julialang.org/en/v1/>(дата обращения:11.10.2024).