Отчет по лабораторной работе 5

Построение графиков

Шалыгин Георгий Эдуардович

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

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

2 Выполнение лабораторной работы

1. Повторим примеры.

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

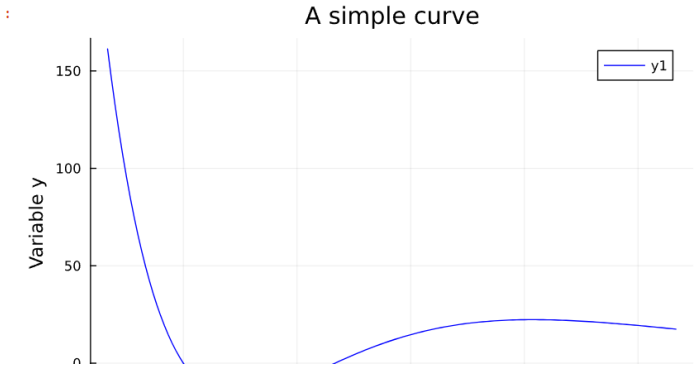


Рис. 2.1: Кривая

```
pyplot()
plot(x,y,
title="Простая кривая",
xlabel="Переменная x",
ylabel="Переменная y",
color="blue")
```

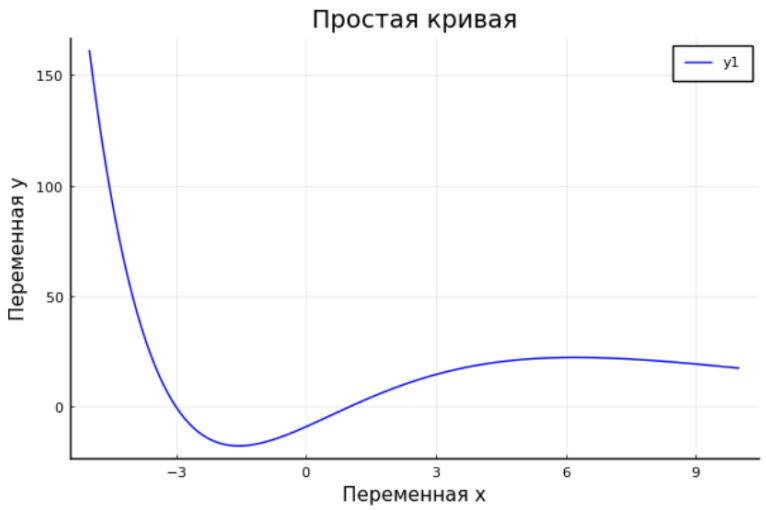


Рис. 2.2: Кривая с плотли

```
sin_theor(x) = sin(x)
plot(sin_theor)
```

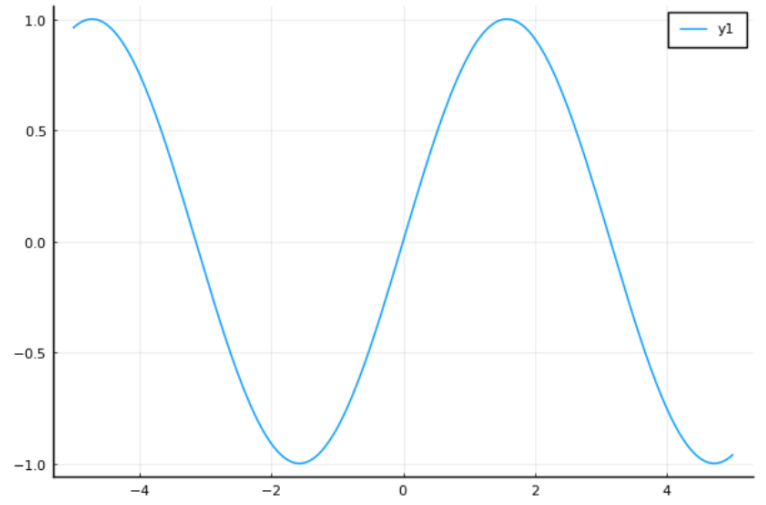


Рис. 2.3: Синус

```
sin_taylor(x) = [(-1)^i*x^(2*i+1)/factorial(2*i+1) for i in 0:4] |> sum
# построение графика функции sin_taylor(x):
plot(sin_taylor)
# построение двух функций на одном графике:
plot(sin_theor)
plot!(sin_taylor)
```

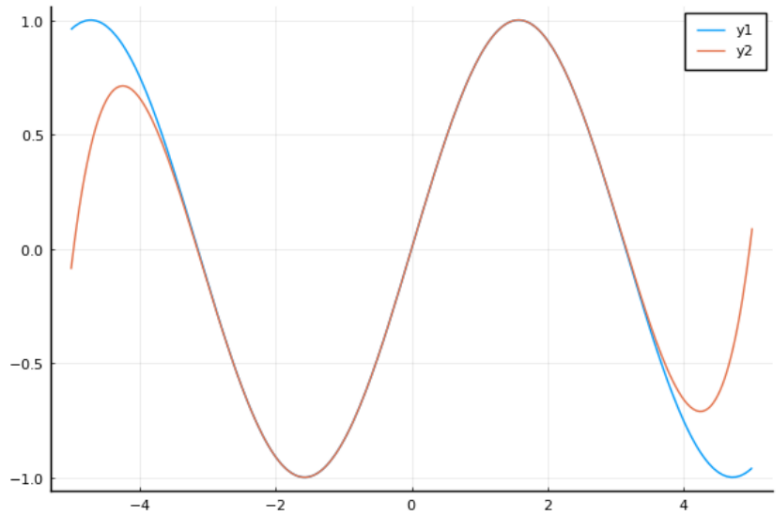


Рис. 2.4: Два графика на одном

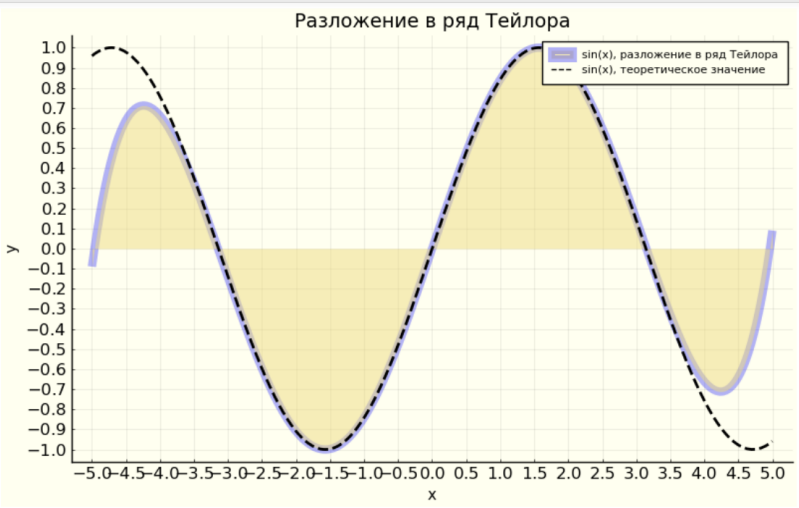


Рис. 2.5: Задание параметров графика

```
savefig("taylor.pdf")
savefig("taylor.png")
```

```
"C:\\Users\\гиоргий\\RUDN\\statan\\taylor.png"
```

Рис. 2.6: Сохранение

```
x = range(1,10,length=10)
y = rand(10)
plot(x, y,
seriestype = :scatter,
title = "Точечный график")
```

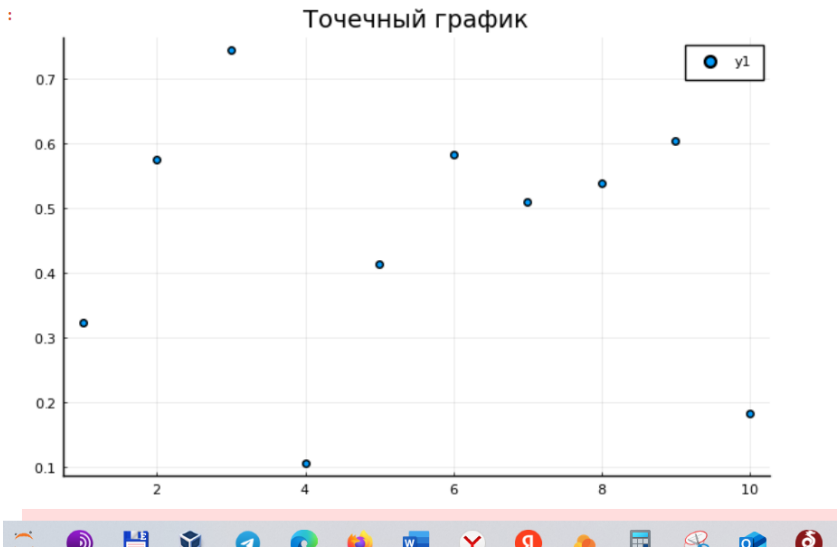


Рис. 2.7: Точечный график

```
n = 50
x = rand(n)
y = rand(n)
ms = rand(50) * 30
# параметры построения графика:
scatter(x, y, markersize=ms)
```

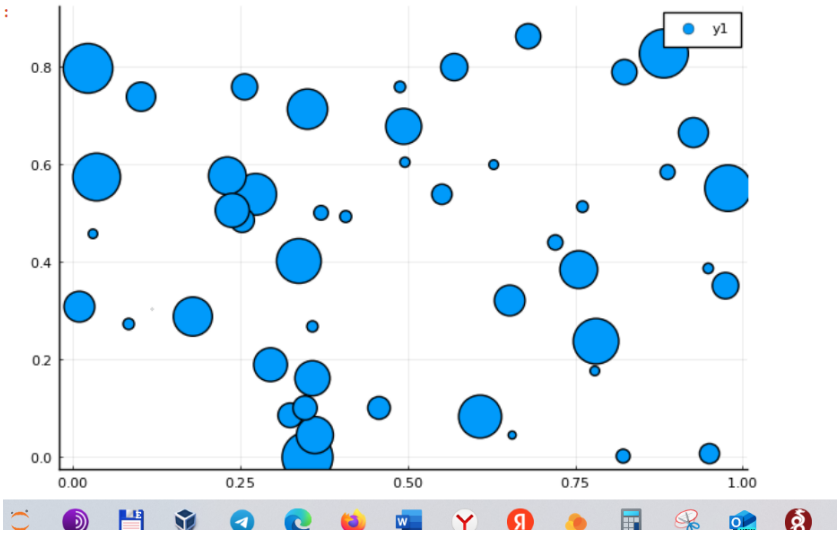


Рис. 2.8: Изменение размера вершин

```
n = 50
x = rand(n)
y = rand(n)
z = rand(n)
ms = rand(50) * 30
# параметры построения графика:
scatter(x, y, z, markersize=ms)
```

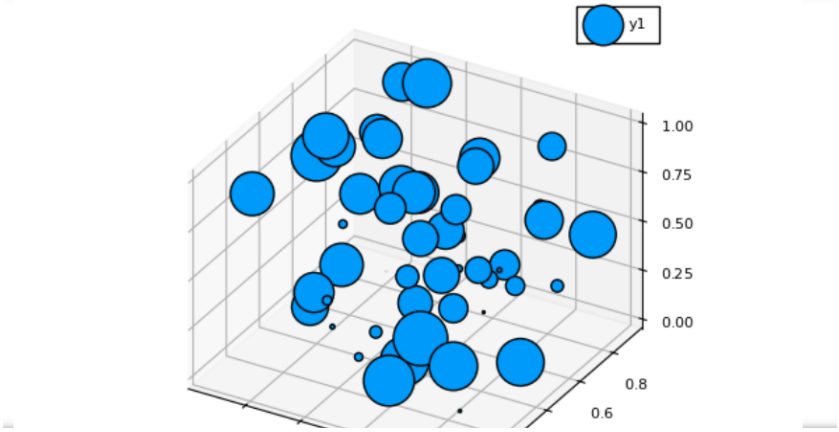


Рис. 2.9: Пространственный график

```
# 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,legend=false)
```

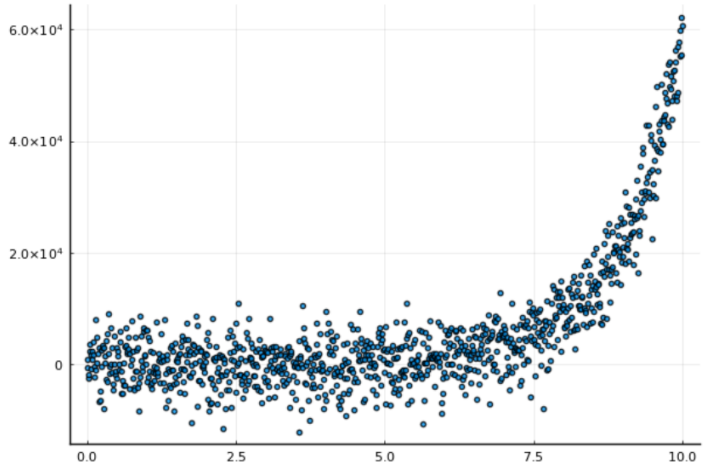


Рис. 2.10: Зашумленные данные

```
# 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_ = c[1]*ones(1000) + c[2]*x + c[3]*x.^2 + c[4]*x.^3 + c[5]*x.^4 + c[6]*x.^5
```

Рис. 2.11: QR-факторизация

```
plot!(x,f_,linewidth=3, color=:red)
```

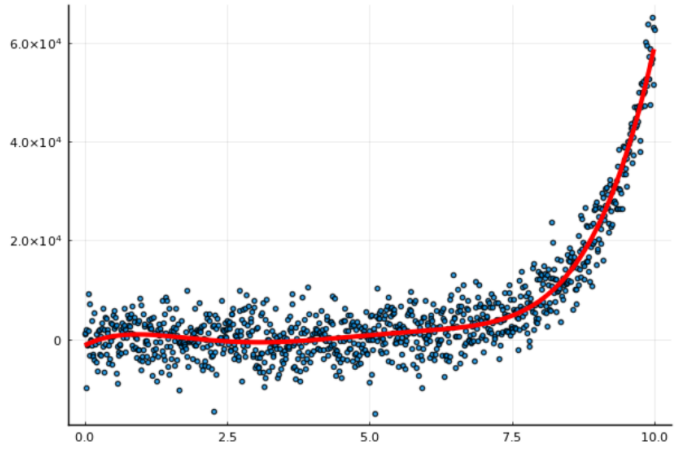


Рис. 2.12: Регрессия данных

```
plot(randn(100),
ylabel="y1",
leg=:topright,
grid = :off,
)
```

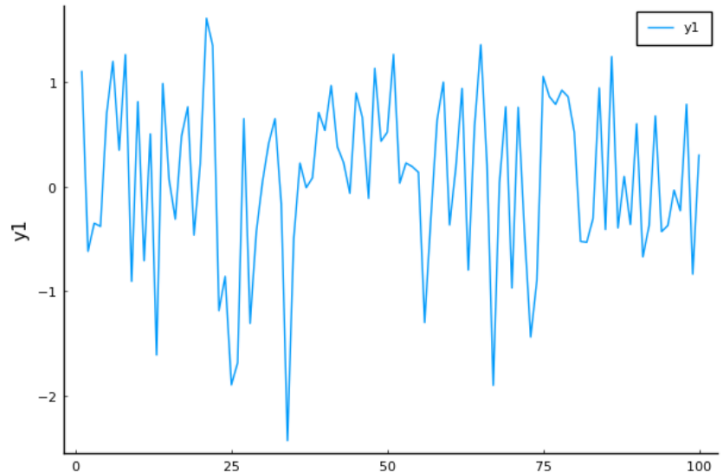


Рис. 2.13: Случайный график

```
plot!(twinx(), randn(100)*10,
c=:red,
ylabel="y2",
leg=:bottomright,
grid = :off,
box = :on,
# size=(600, 400)
)
```

```
KeyError: key "join" not found

Stacktrace:
  [1] __getproperty
    @ D:\julia\depot\packages\PyCall\gaz9g\src\PyCall.jl:313 [inlined]
  [2] getproperty(o::PyCall.PyObject, s::String)
    @ PyCall D:\julia\depot\packages\PyCall\gaz9g\src\PyCall.jl:317
  [3] _before_layout_calcs(plt::Plots.Plot{Plots.PyPlotBackend})
    @ Plots D:\julia\depot\packages\Plots\sxUvK\src\backends\deprecated\pyplot.jl:1331
```

Рис. 2.14: Вторые оси добавить не получилось

```
# 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)
)
```

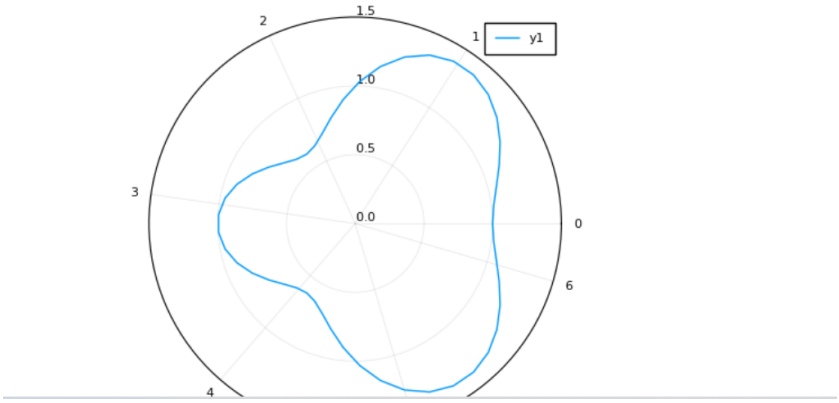


Рис. 2.15: Полярные координаты

```
# параметрическое уравнение:
x_(t) = sin(t)
y_(t) = sin(2t)
# построение графика:
plot(x_, y_, 0, 2π, leg=false, fill=(0,:orange))
```

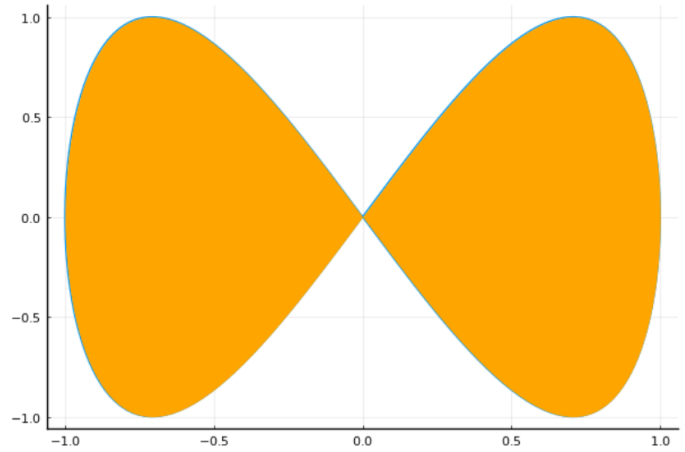


Рис. 2.16: Параметрический график

```
# параметрическое уравнение
t = range(0, stop=10, length=1000)
xx = cos.(t)
yy = sin.(t)
zz = sin.(5t)
# построение графика:
plot(xx, yy, zz)
```

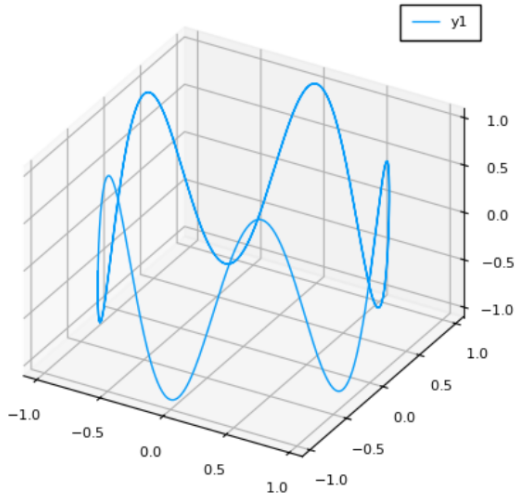


Рис. 2.17: Параметрический график в 3д

```
# построение графика поверхности:
f(x,y) = x^2 + y^2
x = -10:10
y = x
surface(x, y, f)
```

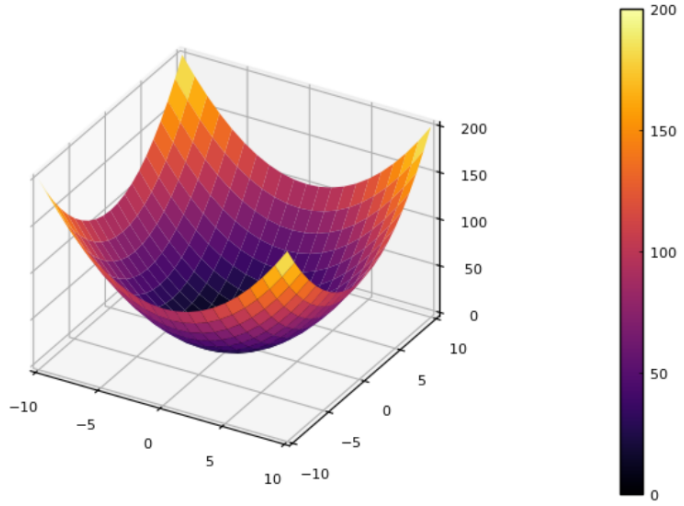


Рис. 2.18: Поверхность

```
# построение графика поверхности:
f(x,y) = x^2 + y^2
x = -10:10
y = x
plot(x, y, f,
linetype=:wireframe
)
```

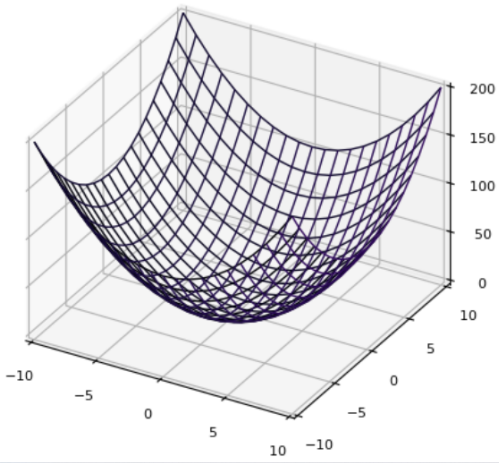


Рис. 2.19: Поверхность

```
x = 1:0.5:20
y = 1:0.5:10
g(x, y) = (3x + y ^ 2) * abs(sin(x) + cos(y))
p = contour(x, y, g,
fill=true)
plot(p)
```

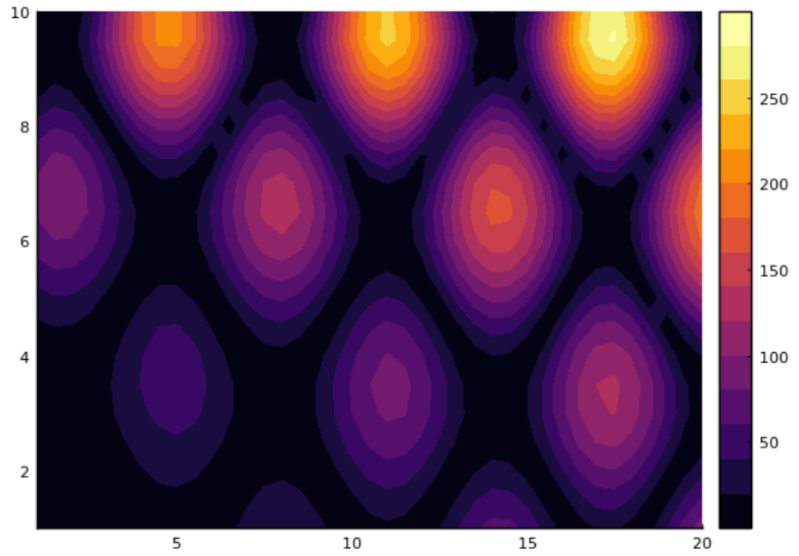


Рис. 2.20: Линии уровня

```
# 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
)
```

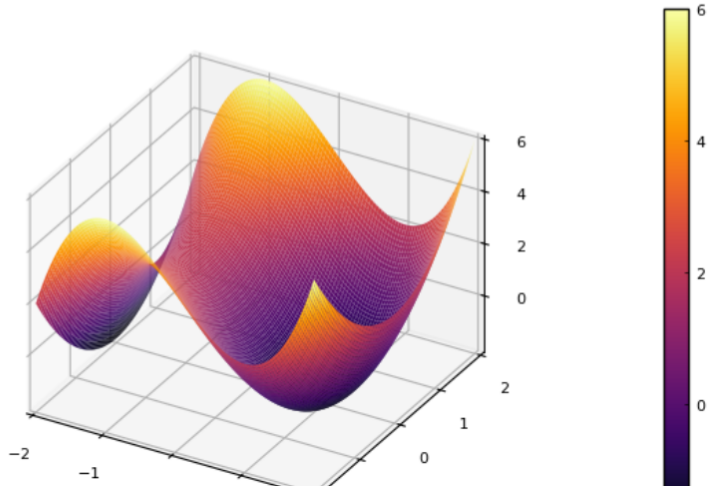


Рис. 2.21: Поверхность

```
pyplot()
i = 0
X = Y = range(-5,stop=5,length=40)
surface(X, Y, (x,y) -> sin(x+10sin(i))+cos(y))

# 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
```

```
[ Info: Saved animation to C:\Users\гиоргий\RUDN\statan\tmp.gif
```

Рис. 2.22: Анимация

```
yc = r*(k-1)*sin(t[end]) .+ r*sin.(θ)
plot!(xc,yc,c=:black)
# радиус малой окружности:
xl = transpose([r*(k-1)*cos(t[end]) x[end]])
yl = transpose([r*(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")
```

```
[ Info: Saved animation to C:\Users\гиоргий\RUDN\statan\hypocycloid.gif
```

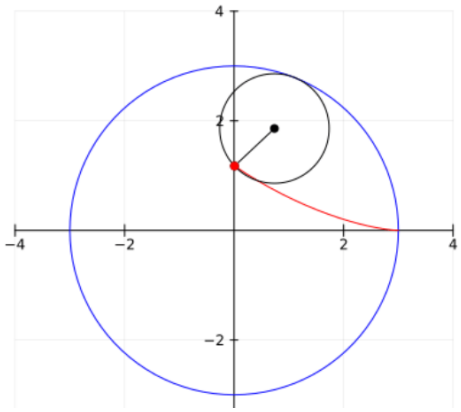


Рис. 2.23: Гипоциклоида

```
ylims = (-1,1),
err = errs
)
```

```
  Updating registry at `D:\julia\depot\registries\General.toml`
 Resolving package versions...
No Changes to `D:\julia\depot\environments\v1.8\Project.toml`
No Changes to `D:\julia\depot\environments\v1.8\Manifest.toml`
```

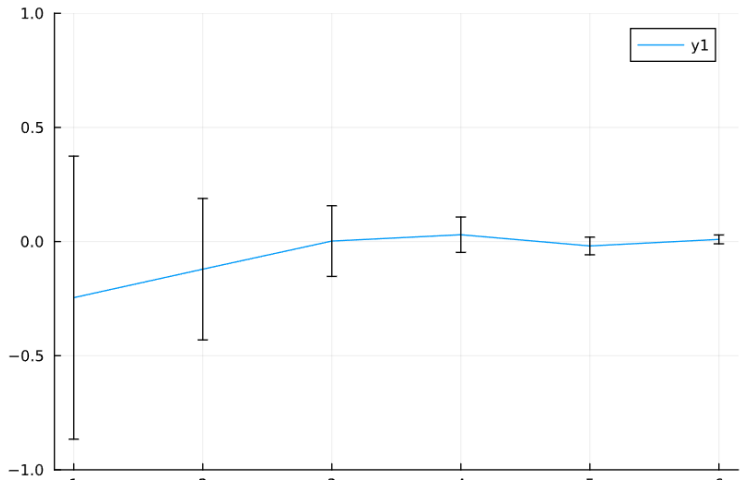


Рис. 2.24: График ошибок

```
plot(y, 1:length(y),
xerr = errs,
marker = stroke(3,:orange)
)
plot(y,
ribbon=errs,
fill=:cyan
)
```

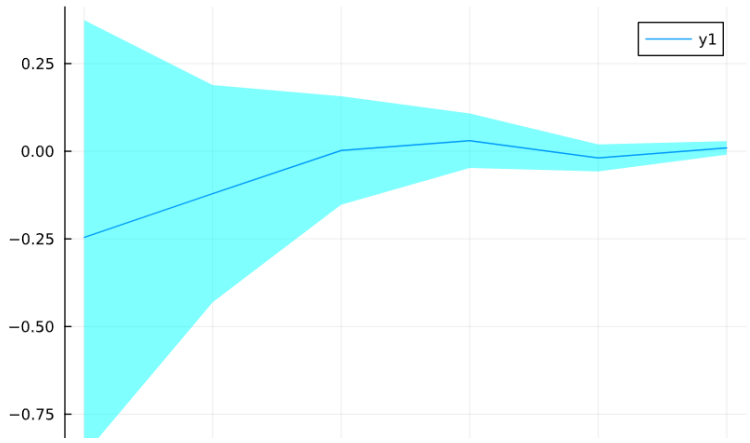


Рис. 2.25: График ошибок 2 вид

```
x = map(mean, x)
y = map(mean, y)
plot(x, y,
xerr = xerr,
yerr = yerr,
marker = stroke(2, :orange)
)
```

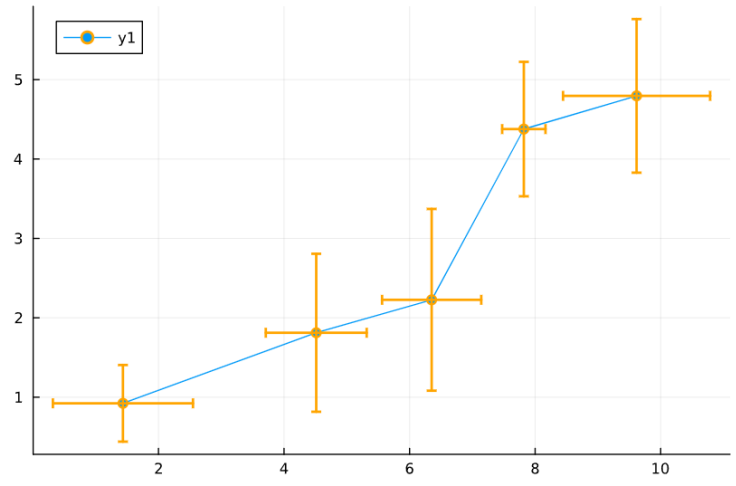


Рис. 2.26: Ошибки по осям

```
ages = rand(d,1000)
histogram(
ages,
label="Распределение по возрастам (года)",
xlabel = "Возраст (лет)",
ylabel= "Количество"
)
```

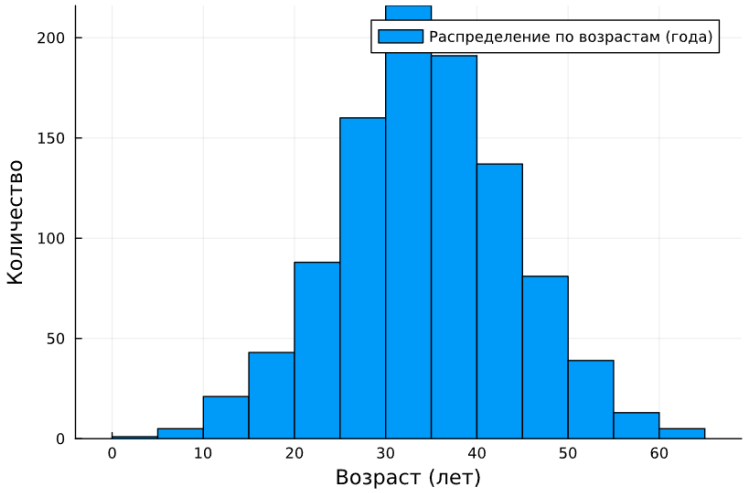


Рис. 2.27: Гистограмма

```
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)"
)
```

Out[8]:

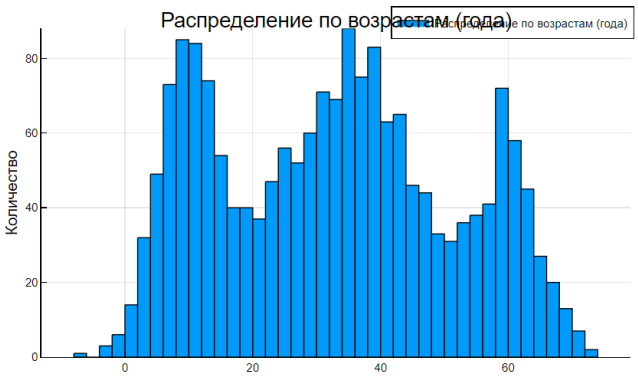


Рис. 2.28: Множественная гистограмма

```
# построение серии графиков:
x=range(-2,2,length=10)
y = rand(10,4)
plot(x,y,
layout=(4,1)
)
```

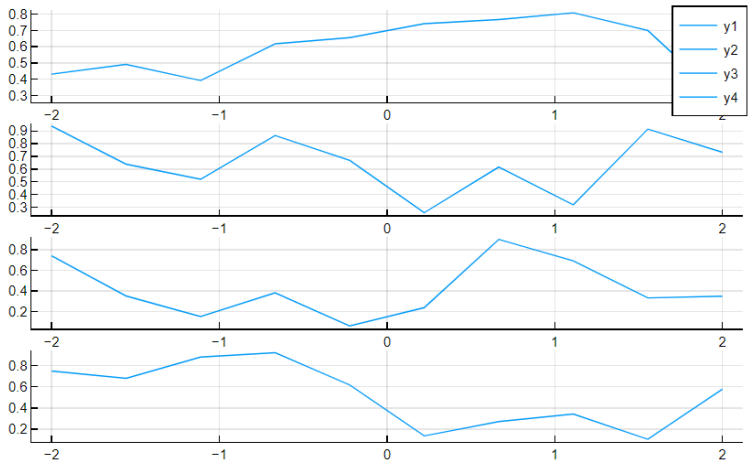


Рис. 2.29: Несколько графиков

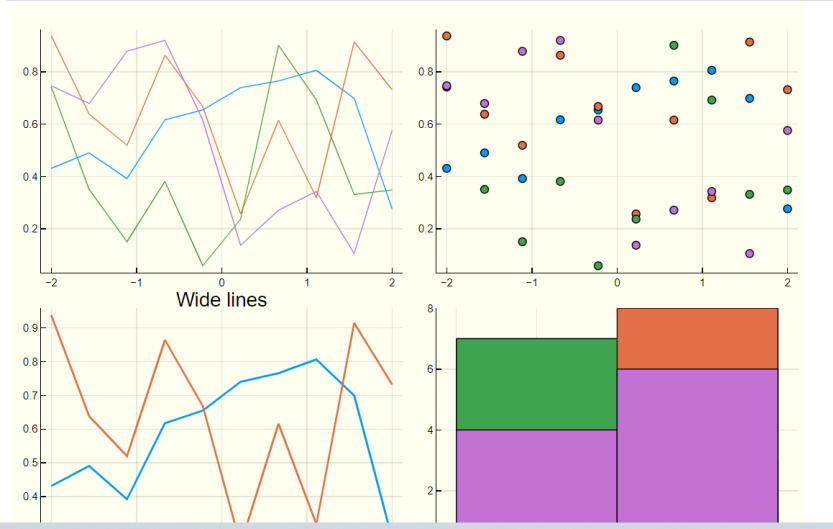


Рис. 2.30: Несколько графиков 2 вид

```
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
)
```

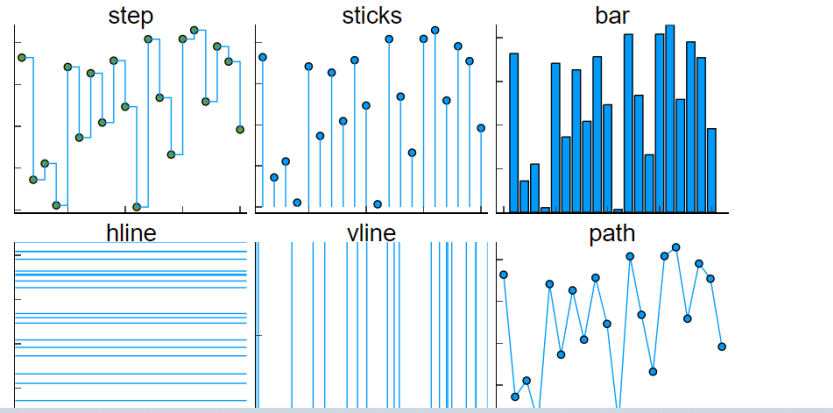


Рис. 2.31: Несколько графиков 3 вид

```
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)
)
```

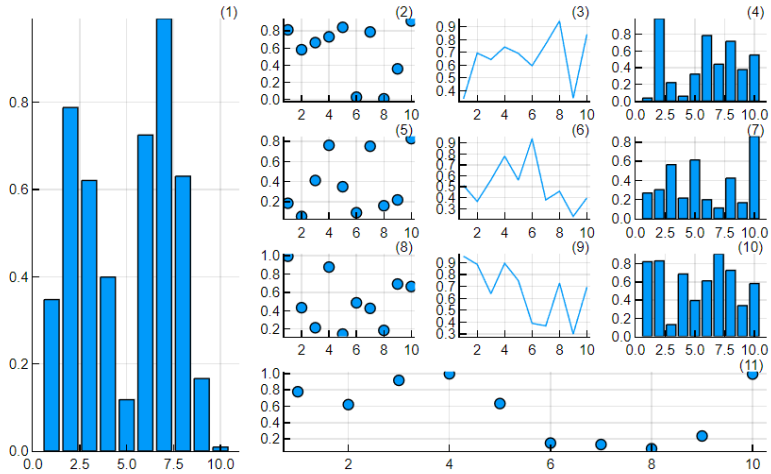


Рис. 2.32: Несколько графиков с макросом

2.1 Задания для самостоятельного выполнения

1. Постройте все возможные типы графиков (простые, точечные, гистограммы и т.д.) функции $sin(X), x \in (0..2\pi)$. Отобразите все графики в одном графическом окне.

```
x = collect(range(0,2*π,length=100))
y = map(sin, x)
p1 = plot(x,y)
p2 = scatter(x,y)
p4 = histogram(x,y)
plot(
p1,p2,p4,
layout=(2,2),
legend=false,
size=(400,300),
background_color = :ivory
)
```

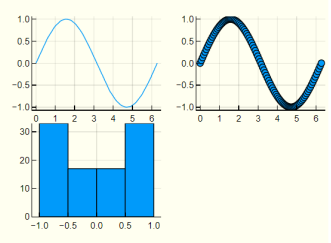


Рис. 2.33: Sinus

2. Постройте графики функции $sin(X), x \in (0..2\pi)$ со всеми возможными (сколько сможете вспомнить) типами оформления линий графика. Отобразите все графики в одном графическом окне.

```
x = collect(range(0,2*π,length=100))
y = map(sin, x)
p1 = plot(x,y, line=(:blue, 0.3, 6, :solid))
p2 = plot(x,y, line=(:black, 0.3, 3, :dash))
p3 = plot(x,y, line=(:green, 1, 6, :dot))
p4 = plot(x,y, line=(:blue, 3, 1, :dashdot))
plot(
p1,p2,p3,p4,
layout=(2,2),
legend=false,
size=(400,300),
background_color = :ivory
)
```

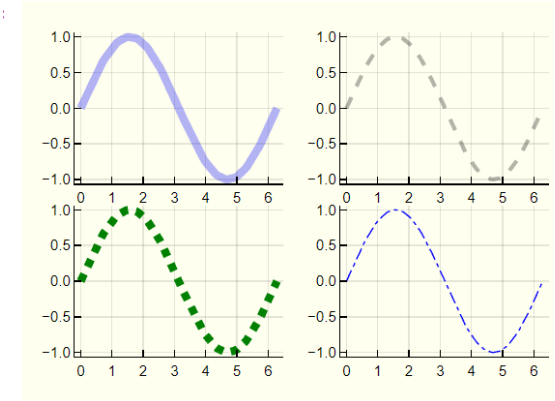


Рис. 2.34: Кастом графика синуса

3. Постройте график функции $y = \pi x^2 \ln(x)$ 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.

```
x = collect(range(0.1,10,length=100))
f(x) = π*x^2*log(x)
y = map(f, x)
plot(
# функция sin(x):
x, y,
# подпись в легенде, цвет и тип линии:
    leg=:topright,
label = "π*x^2*log(x)",
line=(:red, 1, 1, :solid),
# размер графика:
size=(800, 300),
# параметры отображения значений по осям
xticks = (0:1:10),
yticks = (0:100:10000),
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),
# задание цвета фона:
background_color = :green
)
```

Рис. 2.35: График функции

Рис. 2.36: График функции

4. Задайте вектор x = (−2, −1, 0, 1, 2). В одном графическом окне (в 4-х подокнах) изобразите графически по точкам ⊠ значения функции $x^3 - 3x$ в виде: – точек, – линий, – линий и точек, – кривой. Сохраните полученные изображения в файле figure_familiya.png, где вместо familiya укажите вашу фамилию..

```
x = collect((-2, -1, 0, 1, 2))
f(x) = x^3-3x
y = map(f, x)
p1 = plot(x,y, seriestype = :scatter)
p2 = plot(x,y, line=(:black, 1, 1, :solid))
p3 = plot(x,y, line=(:green, 1, 6, :dot))
p4 = plot(x,y, line=(:blue, 3, 1, :dashdot))
plot(
p1,p2,p3,p4,
layout=(2,2),
legend=false,
size=(400,300),
background_color = :ivory
)
```

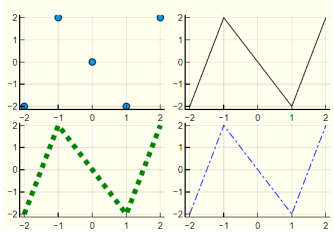
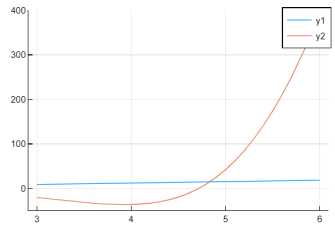


Рис. 2.37: 4 вида графика

5. Задайте вектор x = (3, 3.1, 3.2, ... , 6). Постройте графики функций $y_1 = \pi x$ и $y_2 = exp(x)cos(x)$ в указанном диапазоне значений аргумента ☒ 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 = collect(3:0.1:6)
y1 = map(x -> pi*x, x)
y2 = map(x -> exp(x)*cos(x), x)
plot(x, y1, size=(400,300))
plot!(x, y2)
```



```
x = collect(3:0.1:6)
y1 = map(x -> pi*x, x)
y2 = map(x -> exp(x)*cos(x), x)
plot(x, y1, lineline=(:black, 1, 1, :solid), size=(400,300))
plot!(twinx(), x, y2,  c=:red,
ylabel="y2",
leg=:bottomright,
grid = :off,
box = :on)
```

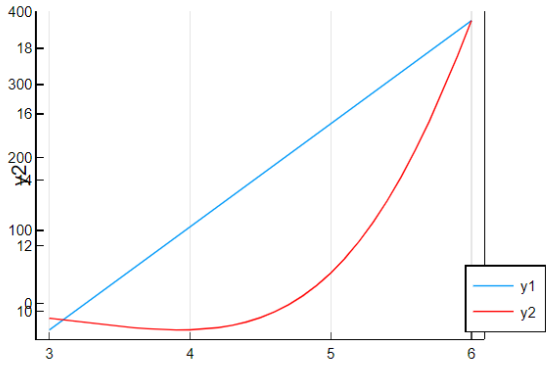


Рис. 2.38: Разные оси

Здесь, очевидно, удобнее две оси из-за разных масштабов. В первом случае график просто становится линией.

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

```
n = 10
y = [randn(n) / n^3 for xi in x]
errs = 1.96 * sds / sqrt(n)
plot(y2,
)
plot!(y2, err=errs)
```

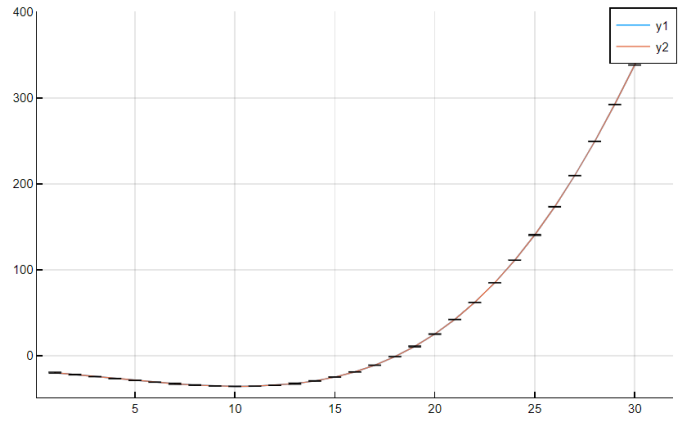


Рис. 2.39: График ошибок

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

```
n=10
x = [rand(n) for i in 1:3]
y = [rand(n) for i in 1:3]
plot(x, y, seriestype = :scatter, title = "Точечный график",
    xlabel="x", ylabel="y")
```

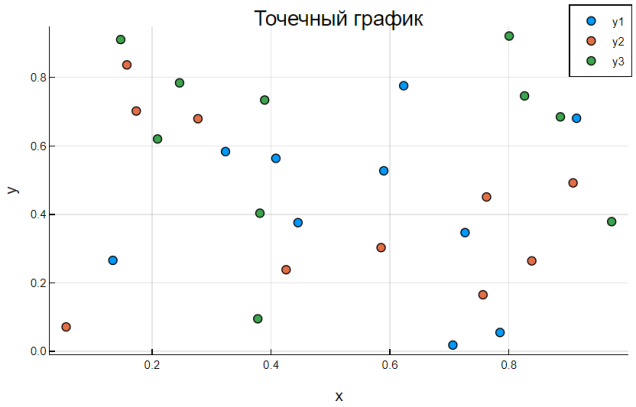


Рис. 2.40: Случайные точки трех кластеров

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

```
n = 50
x = rand(n)
y = rand(n)
z = rand(n)
ms = rand(50) * 3
# параметры построения графика:
scatter(x, y, z, markersize=ms, xlabel="x", ylabel="y", zlabel="z", title="Random data", label="random DATA")
```

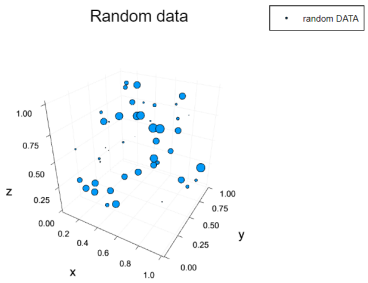


Рис. 2.41: Трехмерный случайный график

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

```
pyplot()
x = collect(0:0.1:1.5*pi)
plot(x, x -> sin(x))

@gif for i in range(0,stop=2π,length=100)
    plot(x, x -> sin(x*i))
end
```

```
[ Info: Saved animation to C:\Users\гиоргий\RUDN\statan\tmp.gif
```

Рис. 2.42: Анимация синуса

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

```
plot!(xl,yl,markershape=:circle,markersize=4,c=:black)
scatter!([x[end]],[y[end]],c=:red, markerstrokecolor=:red)
end
gif(anim,"hypocycloid.gif")
```

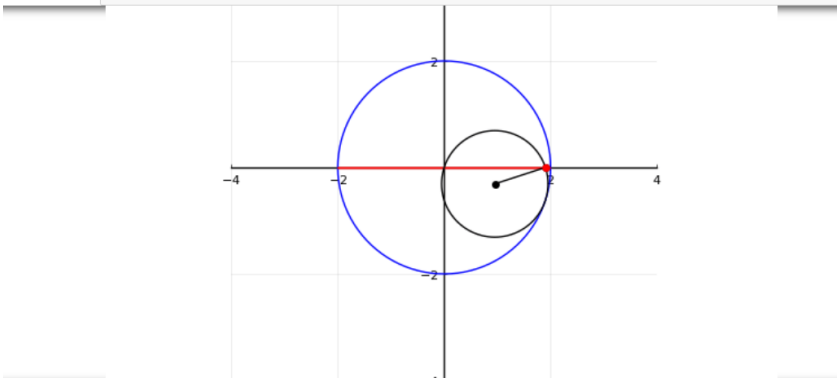


Рис. 2.43: Целое значение

```
end
gif(anim,"hypocycloid.gif")
```

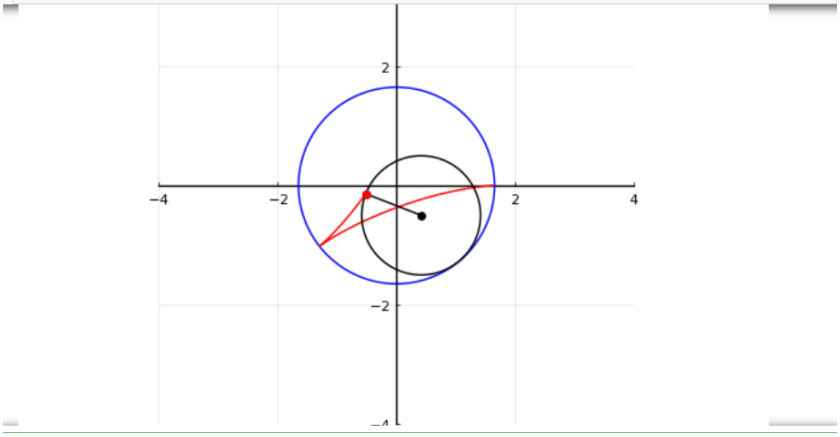


Рис. 2.44: Нецелое значение

В итоге для всех нецелых значений циклоида расходилась, а для целых возвращалась в исходную точку.

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

Out[126]:

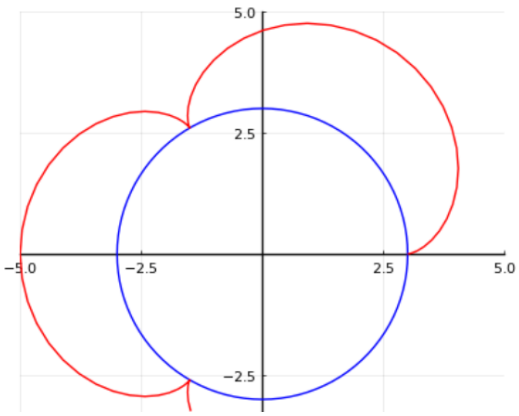


Рис. 2.45: Целое значение

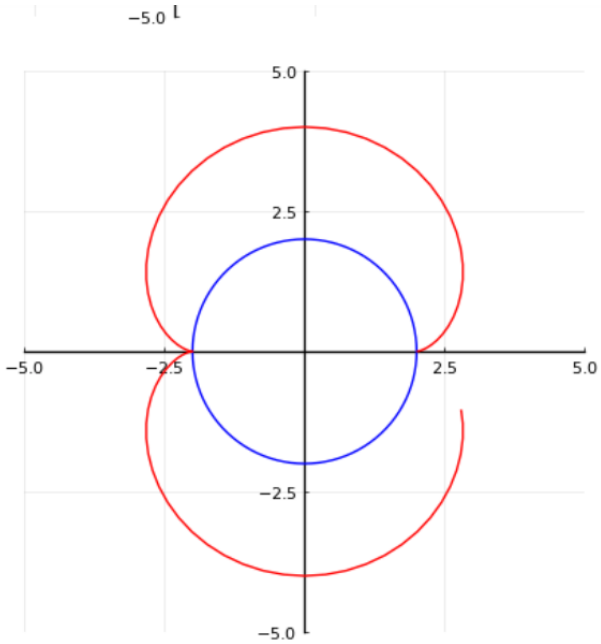


Рис. 2.46: Целое значение

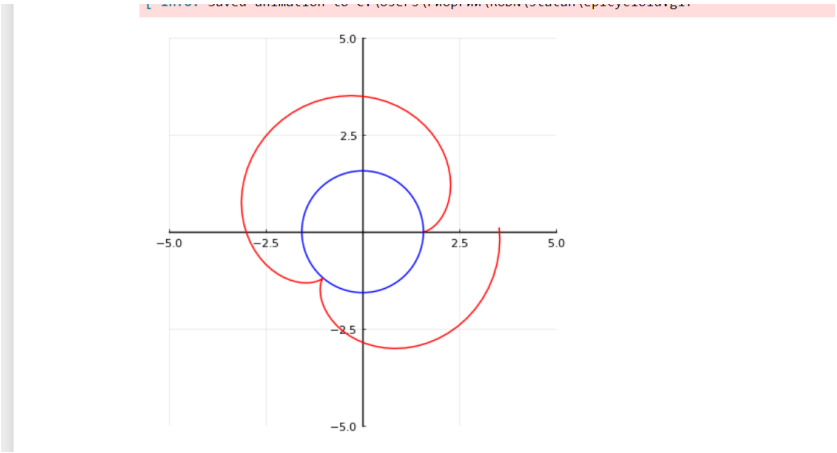


Рис. 2.47: Нецелое значение

Аналогично, циклоида замыкалась для целых значений.

3 Выводы

В ходе работы был освоен синтаксис языка Julia для построения графиков

Список литературы