Отчёт по лабораторной работе №5

Построение графиков

Ким Реачна

Содержание

Список иллюстраций

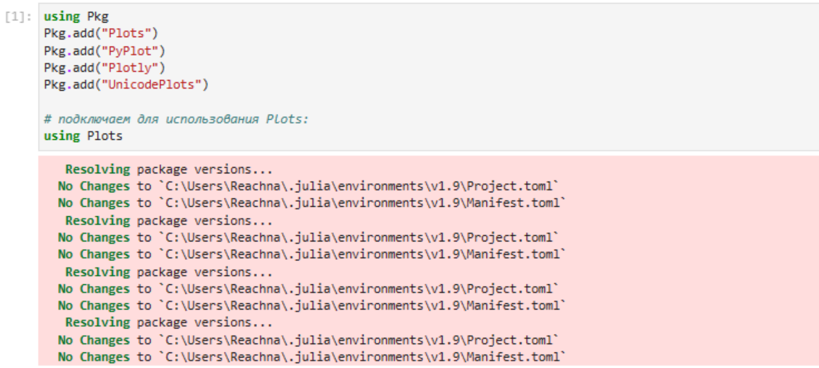
1 Цель работы

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

2 Выполнение лабораторной работы

2.1 Основные пакеты для работы с графиками в Julia

- Установила все необходимые пакеты для работы с графиками:



```
[1]: using Pkg
Pkg.add("Plots")
Pkg.add("PyPlot")
Pkg.add("Plotly")
Pkg.add("UnicodePlots")

# подключаем для использования Plots:
using Plots
```

```
  Resolving package versions...
 No Changes to `C:\Users\Reachna\.julia\environments\v1.9\Project.toml`
 No Changes to `C:\Users\Reachna\.julia\environments\v1.9\Manifest.toml`
  Resolving package versions...
 No Changes to `C:\Users\Reachna\.julia\environments\v1.9\Project.toml`
 No Changes to `C:\Users\Reachna\.julia\environments\v1.9\Manifest.toml`
  Resolving package versions...
 No Changes to `C:\Users\Reachna\.julia\environments\v1.9\Project.toml`
 No Changes to `C:\Users\Reachna\.julia\environments\v1.9\Manifest.toml`
  Resolving package versions...
 No Changes to `C:\Users\Reachna\.julia\environments\v1.9\Project.toml`
 No Changes to `C:\Users\Reachna\.julia\environments\v1.9\Manifest.toml`
```

Рис. 2.1: Установка все необходимые пакеты для работы с графиками

- Задала исходную функцию и построил первый график, перевела надписи на русский язык

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

# генерирование массива значений x в диапазоне от -5 до 10 с шагом 0,1
# (шаг задан через указание длины массива):
x = collect(range(-5,10,length=151))
# генерирование массива значений y:
y = f(x)

# указывается, что для построения графика используется gr():
gr()
# задание опций при построении графика
# (название кривой, подписи по осям, цвет графика):
plot(x,y,
    title="A simple curve",
    xlabel="Variable x",
    ylabel="Variable y",
    color="blue")
```

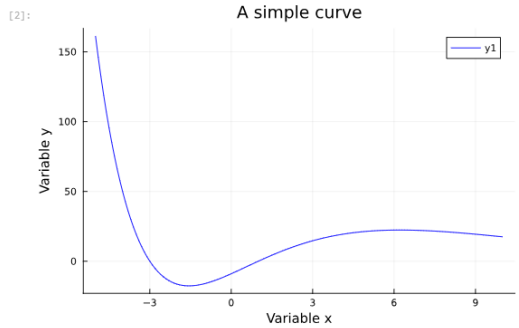


Рис. 2.2: График функции $f(x) = (3x^2 + 6x - 9)e^{-0,3x}$, при помощи gr()

```
[3]: using Plots
# указывается, что для построения графика используется pyplot():
pyplot()
# задание опций при построении графика
# (название кривой, подписи по осям, цвет графика):
plot(x,y,
    title="Простая кривая",
    xlabel="Переменная x",
    ylabel="Переменная y",
    color="blue")
```

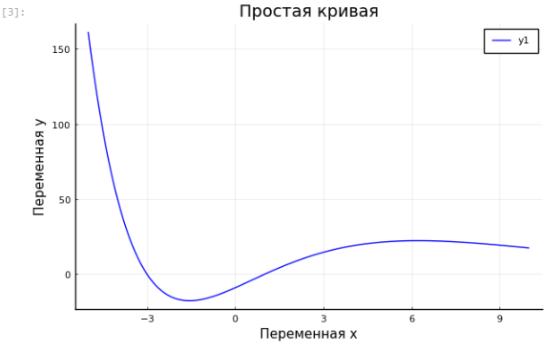


Рис. 2.3: График функции $f(x) = (3x^2 + 6x - 9)e^{-0,3x}$, при помощи pyplot()

- Построение графика при помощи plotly(). В начале вызываю необходимую библиотеку и с помощью функции plot() строю график.

```
[4]: using Plots
     # указывается, что для построения графика используется plotly():
     plotly()
     # задание опций при построении графика
     # (название кривой, подписи по осям, цвет графика):
     plot(x,y,
         title="Простая кривая",
         xlabel="Переменная x",
         ylabel="Переменная y",
         color="blue")
```

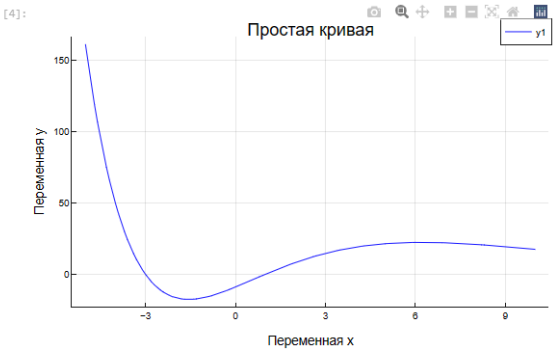


Рис. 2.4: График функции $f(x) = (3x^2 + 6x - 9)e^{-0,3x}$, при помощи plotly()

- Построение графика при помощи unicodeplots(). Так как использование просто в качестве бэкенда не сработало, вызываю функции из пакета.

```
[5]: # using Plots
     # unicodeplots()
     using UnicodePlots #т.к. использование просто в качестве бэкенда не сработало, вызываю функции собственно пакета
     # задание опций при построении графика
     # (название кривой, подписи по осям, цвет графика):
     lineplot(x, y,
         title="Простая кривая",
         xlabel="Переменная x",
         ylabel="Переменная y",
         color= :blue)
```

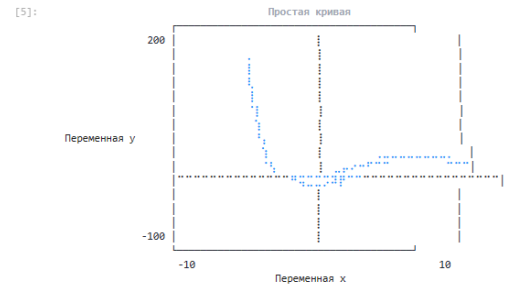


Рис. 2.5: График функции $f(x) = (3x^2+6x-9)e^{-0,3x}$, при помощи unicodeplot()

2.2 Опции при построении графика

Построила график синисуоиды, разложение ее в ряд Тейлора и сравнила результаты и красиво всё оформила:

```
[6]: # рассмотрим дополнительные возможности пакетов для работы с графиками
using Plots
# указывается, что для построения графика используется pyplot():
pyplot()
# задание функции sin(x):
sin_theor(x) = sin(x)

# построение графика функции sin(x):
plot(sin_theor)
```

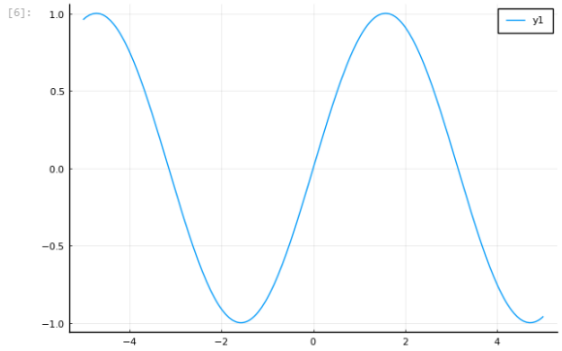


Рис. 2.6: График функции $sin(x)$

```
[7]: pyplot()
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)
```

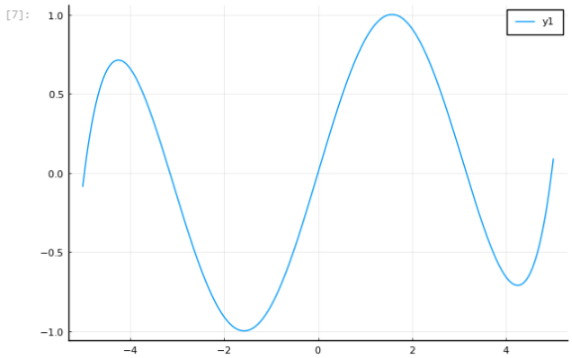


Рис. 2.7: График функции разложения исходной функции в ряд Тейлора

```
[8]: # построение двух функций на одном графике:
plot(sin_theor)
plot!(sin_taylor)
```

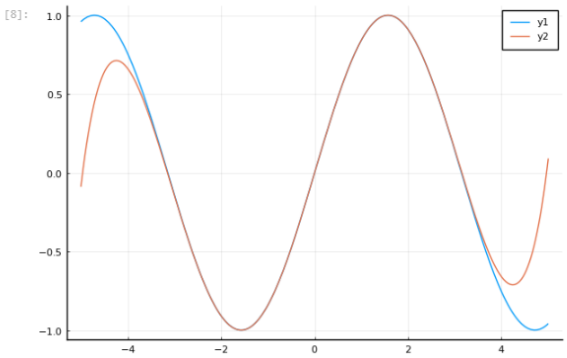


Рис. 2.8: Графики исходной функции и её разложения в ряд Тейлора

```
[9]: 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",
        # 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",
        leg=:topright,
        line=(:black, 1.0, 2, :dash)
    )
```

Рис. 2.9: Вид графиков после добавления опций при их построении

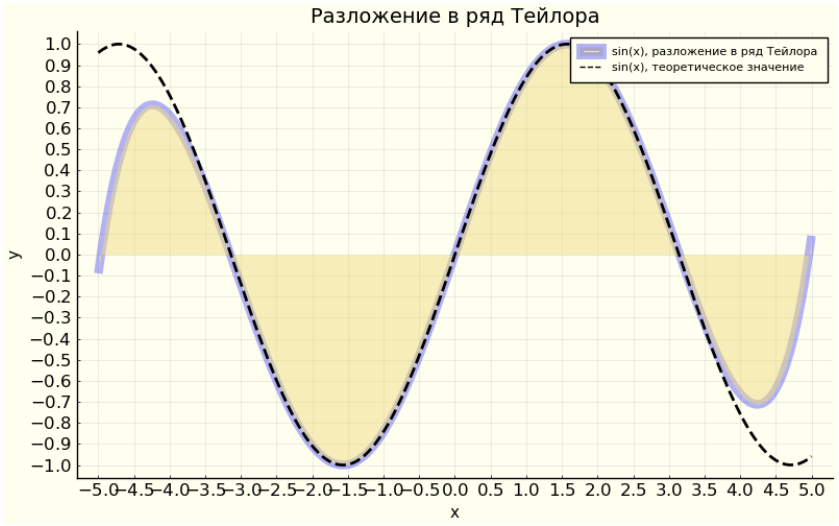


Рис. 2.10: Вид графиков после добавления опций при их построении

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

Рис. 2.11: Сохранение построенного графика

2.3 Точечный график

2.3.1 Простой точечный график

```
[11]: # параметры распределения точек на плоскости:
x = range(1,10,length=10)
y = rand(10)
# параметры построения графика:
plot(x, y,
    seriestype = :scatter,
    title = "Точечный график",
    xlabel = "x",
    ylabel = "y",
    # подпись в легенде, цвет и тип линии:
    label = "y1, теоретическое значение",
    leg = :topright
)
```

[11]:

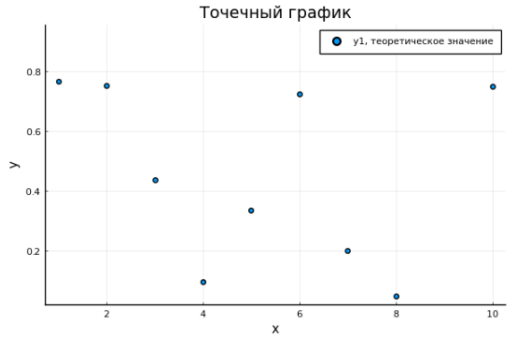


Рис. 2.12: График десяти случайных значений на плоскости

2.3.2 Точечный график с кодированием значения размером точки

```
[12]: # 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,
    # подписи по осям:
    xlabel = "x",
    ylabel = "y",
    # подпись в легенде, цвет и тип линии:
    label = "y1, теоретическое значение",
    title = "Точечный график")
```

[12]:

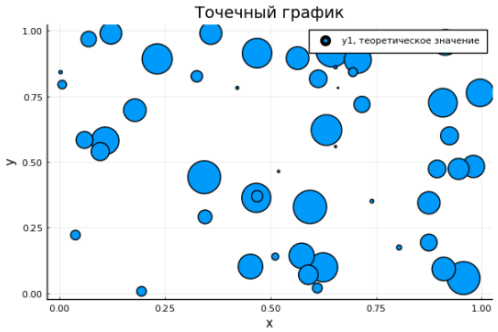


Рис. 2.13: График пятидесяти случайных значений на плоскости с различными опциями отображения

2.3.3 3-мерный точечный график с кодированием значения размером точки

```
[13]: # 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,
    xlabel = "x",
    ylabel = "y",
    zlabel = "z",
    # подпись в легенде, цвет и тип линии:
    label = "sin(x), теоретическое значение",
    title = "Точечный график")
```

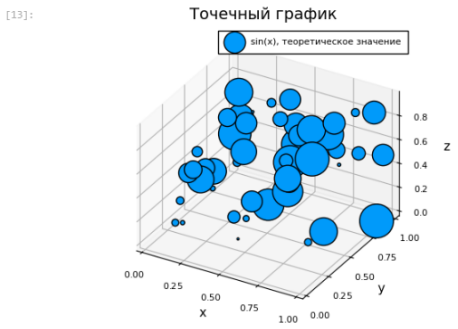


Рис. 2.14: График пятидесяти случайных значений в пространстве с различными опциями отображения

2.4 Аппроксимация данных

```
[14]: # 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)
```

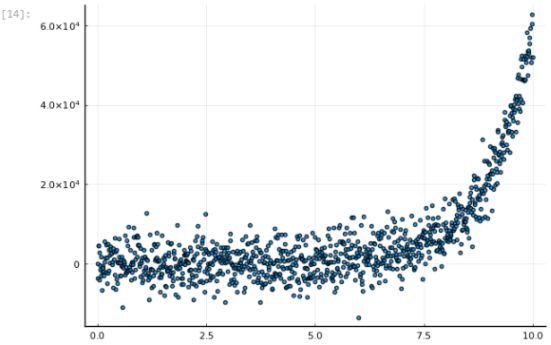


Рис. 2.15: Пример функции

```
[15]: # 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:
f1 = 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, f1, linewidth=3, color=:red)
```

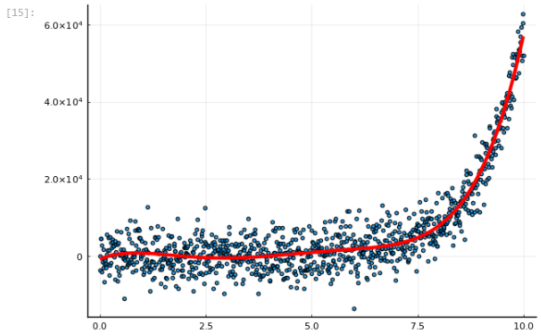


Рис. 2.16: Пример аппроксимации исходной функции полиномом 5-й степени

2.5 Две оси ординат

```
# пример случайной траектории
# (заданы обозначение траектории, легенда вверху справа, без сетки)
plot(randn(100),
    ylabel="y1",
    leg=:topright,
    grid = :off,
)
```

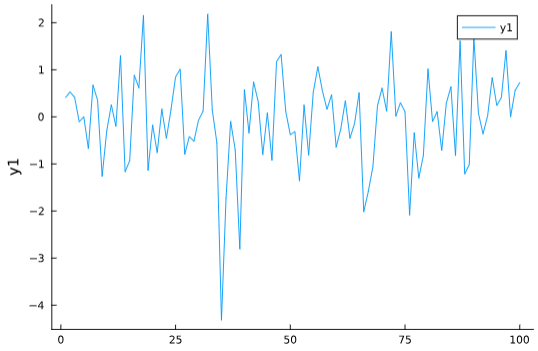


Рис. 2.17: Пример отдельно построенной траектории

```
# пример добавления на график второй случайной траектории
# (задано обозначение траектории и её цвет, легенда снизу справа, без сетки)
# задана рамка графика
plot!(twinx(), randn(100)*10,
    c=:red,
    ylabel="y2",
    leg=:bottomright,
    grid = :off,
    box = :on,
    # size=(600, 400)
)
```

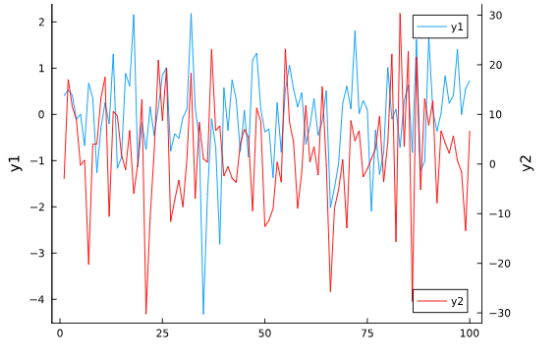


Рис. 2.18: Пример двух траекторий на одном графике с двумя осями ординат

2.6 Полярные координаты

```
[18]: # 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)
      )
```

[18]:

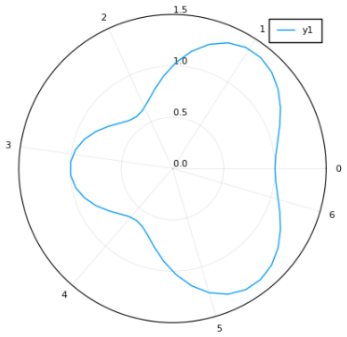


Рис. 2.19: График функции, заданной в полярных координатах

2.7 Параметрический график

2.7.1 Параметрический график кривой на плоскости

```
[19]: # параметрическое уравнение:
x1(t) = sin(t)
y1(t) = sin(2t)
# построение графика:
plot(x1, y1, 0, 2π, leg=false, fill=(0,:orange))
```

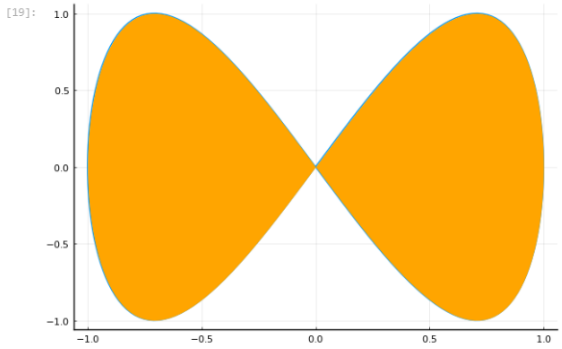


Рис. 2.20: Параметрический график кривой на плоскости

2.7.2 Параметрический график кривой в пространстве

```
[20]: # параметрическое уравнение
t = range(0, stop=10, length=1000)
x = cos.(t)
y = sin.(t)
z = sin.(5t)
# построение графика:
plot(x, y, z)
```

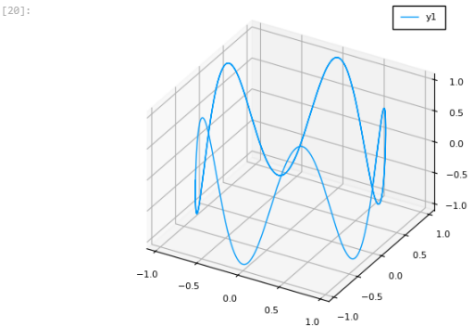


Рис. 2.21: Параметрический график кривой в пространстве

2.8 График поверхности

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

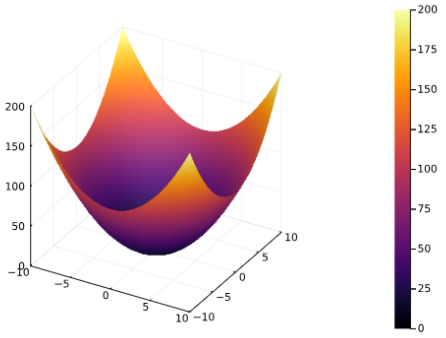


Рис. 2.22: График поверхности использована функция surface()

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

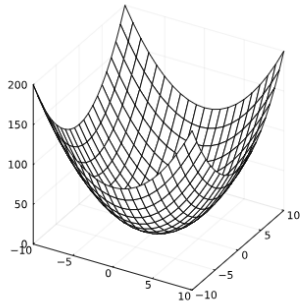


Рис. 2.23: График поверхности использована функция plot()

```
f(x,y) = x^2 + y^2
x = -10:0.1:10
y = x
plot(x, y, f,
linetype = :surface
)
```

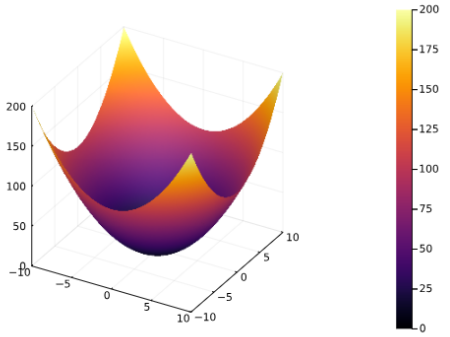


Рис. 2.24: Сглаженный график поверхности

```
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),
)
```

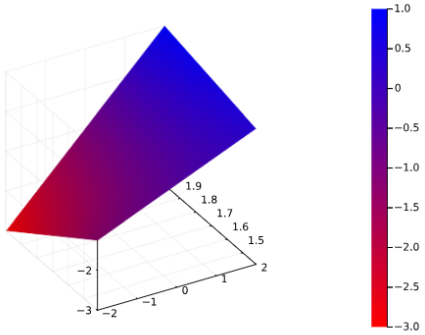


Рис. 2.25: График поверхности с изменённым углом зрения

2.9 Линии уровня

```
[25]: 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,
      )
```

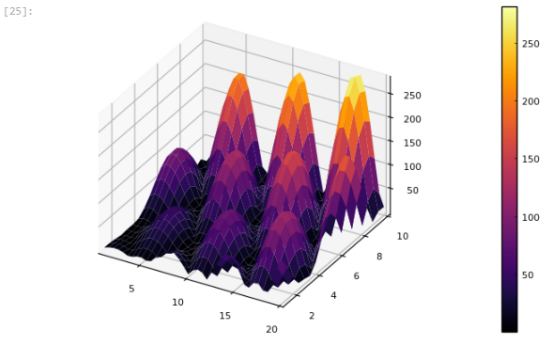


Рис. 2.26: График поверхности, заданной функцией $g(x, y) = (3x+y^2)|sin(x)+cos(y)|$

```
[26]: contour(x, y, g)
```

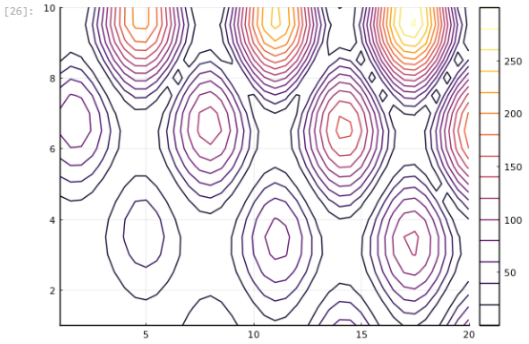


Рис. 2.27: Линии уровня

```
[27]: p = contour(x, y, g,
      fill=true)
      plot(p)
```

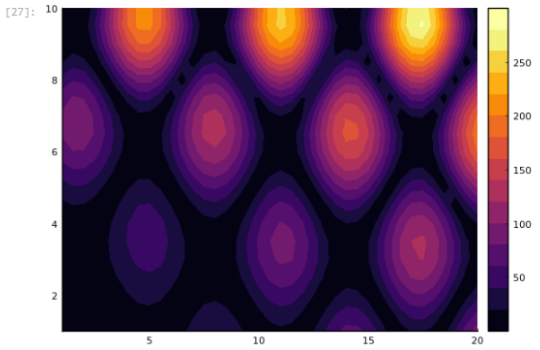


Рис. 2.28: Линии уровня с заполнением

2.10 Векторные поля

```
# определение переменных:
X0 = range(-2, stop=2, length=100)
Y0 = range(-2, stop=2, length=100)
# определение функции:
h(x, y) = x^3 - 3x + y^2
# построение поверхности:
plot(X0, Y0, h,
    linetype = :surface
)
```

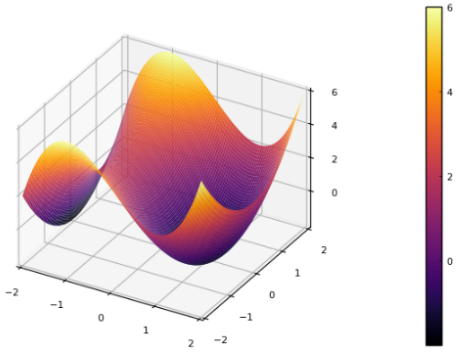


Рис. 2.29: График функции $h(x, y) = x^3 - 3x + y^2$

```
# построение линий уровня:
contour(X0, Y0, h)
```

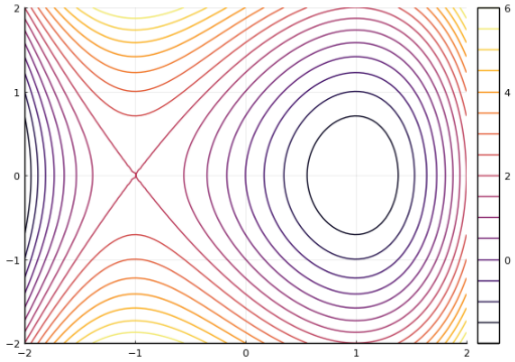


Рис. 2.30: Линии уровня для функции $h(x, y) = x^3 - 3x + y^2$

```
[30]: # градиент:
      xs = range(-2, stop=2, length=12)
      ys = range(-2, stop=2, length=12)

[30]: -2.0:0.36363636363636365:2.0

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

[31]: dh (generic function with 1 method)

[32]: xxs = [x for x in xs for y in ys]
      yys = [y for x in xs for y in ys]

      quiver!(xxs, yys, quiver=dh, c=:blue)
```

[32]:

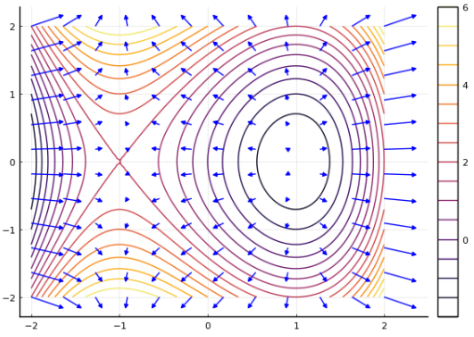


Рис. 2.31: Векторное поле функции $h(x, y) = x^3 - 3x + y^2$

```
[33]: # коррекция области видимости графика:
xlims!(-2, 2)
ylims!(-2, 2)
quiver!(xxs, yys, quiver=dh, c=:blue)
```

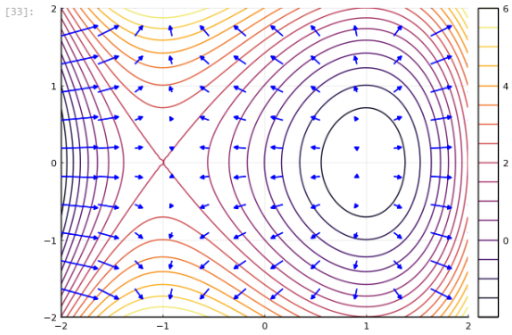


Рис. 2.32: Векторное поле функции $h(x, y) = x^3 - 3x + y^2$ скорректирована область видимости

2.11 Анимация

2.11.1 Gif-анимация

```
[34]: pyplot()
# построение поверхности:
i = 0
X = Y = range(-5,stop=5,length=40)
surface(X, Y, (x,y) -> sin(x+10sin(i))+cos(y))
```

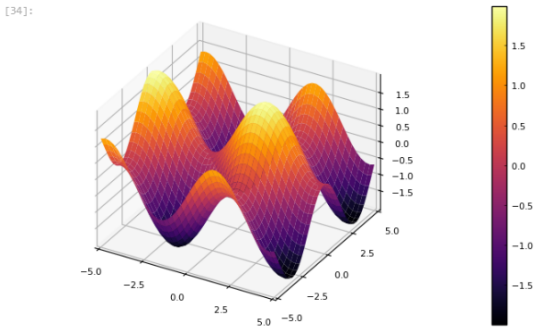
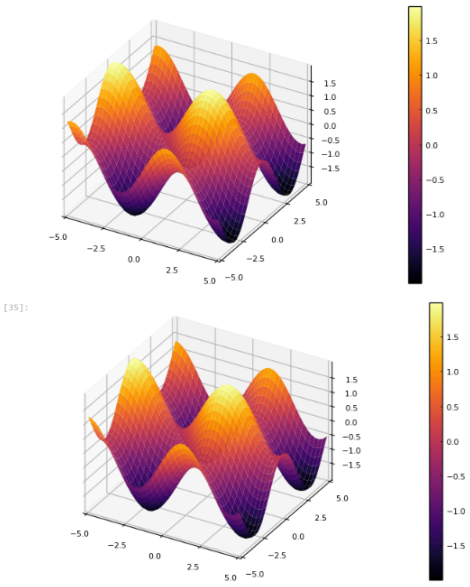


Рис. 2.33: Статичный график поверхности

```
[35]: # 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\Reachna\computer-analysis\tmp.gif
```



[35]:

Рис. 2.34: Анимированный график поверхности

2.11.2 Гипоциклоида

```
[36]: # радиус малой окружности:
r1 = 1
# коэффициент для построения большой окружности:
k = 3
# число отсчётов:
n = 100
# массив значений угла θ:
# theta from 0 to 2pi ( + a little extra)
θ = collect(0:2*π/100:2*π+2*π/100)
# массивы значений координат:
X = r1*k*cos.(θ)
Y = r1*k*sin.(θ)
# задаём оси координат:
plt = plot(5, xlim=(-4,4), ylim=(-4,4), c=:red, aspect_ratio=1, legend=false, framestyle=:origin)
# большая окружность:
plot!(plt, X,Y, c=:blue, legend=false)
```

[36]:

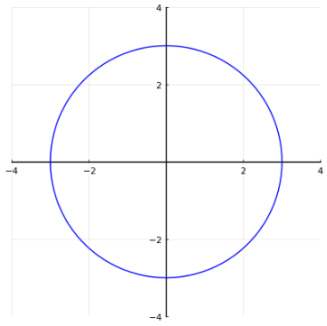


Рис. 2.35: Большая окружность гипоциклоиды

```
[37]: i = 50
      t = θ[1:i]
      # гипоциклоида:
      x = r1*(k-1)*cos.(t) + r1*cos.((k-1)*t)
      y = r1*(k-1)*sin.(t) - r1*sin.((k-1)*t)
      plot!(x,y, c=:red)
```

[37]:

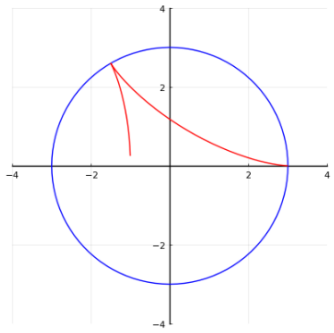


Рис. 2.36: Половина пути гипоциклоиды

```
[38]: # малая окружность:
      xc = r1*(k-1)*cos(t[end]) .+ r1*cos.(θ)
      yc = r1*(k-1)*sin(t[end]) .+ r1*sin.(θ)
      plot!(xc,yc,c=:black)
```

[38]:

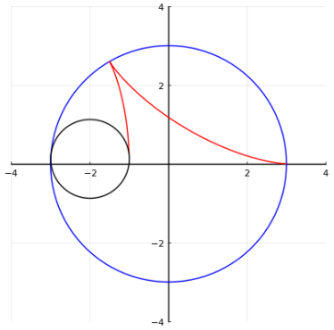


Рис. 2.37: Малая окружность гипоциклоиды

```
[39]: # радиус малой окружности:
      x1 = transpose([r1*(k-1)*cos(t[end]) x[end]])
      y1 = transpose([r1*(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)
```

[39]:

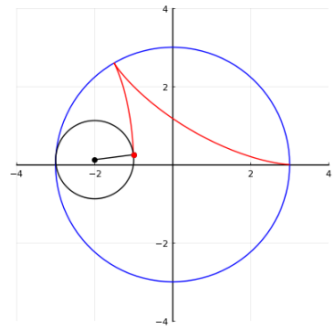


Рис. 2.38: Малая окружность гипоциклоиды с добавлением радиуса

```
[40]: anim = @animate for i in 1:n
# задаём оси координат:
plt=plot(5, xlim=(-4,4), ylim=(-4,4), c=:red, aspect_ratio=1, legend=false, framestyle=:origin)
# большая окружность:
plot!(plt, X, Y, c=:blue, legend=false)
t = θ[1:i]

# гипоциклоида:
x = r1*(k-1)*cos.(t) + r1*cos.((k-1)*t)
y = r1*(k-1)*sin.(t) - r1*sin.((k-1)*t)
plot!(x,y, c=:red)

# малая окружность:
xc = r1*(k-1)*cos(t[end]) .+ r1*cos.(θ)
yc = r1*(k-1)*sin(t[end]) .+ r1*sin.(θ)
plot!(xc, yc, c=:black)

# радиус малой окружности:
xl = transpose([r1*(k-1)*cos(t[end]) x[end]])
yl = transpose([r1*(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")
```

Рис. 2.39: Анимация движения гипоциклоиды

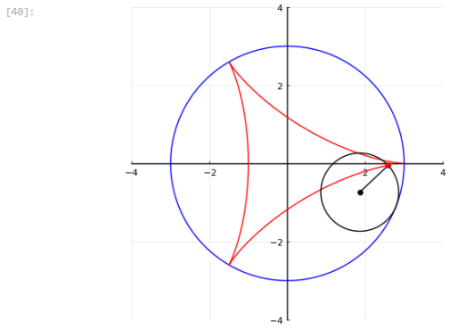


Рис. 2.40: Анимация движения гипоциклоиды

2.12 Errorbars

```
[41]: using Statistics
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),
)
```

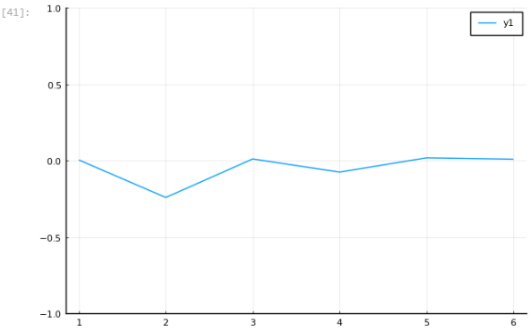


Рис. 2.41: График исходных значений

```
[42]: plot(y,
      ylims = (-1,1),
      err = errs
      )
```

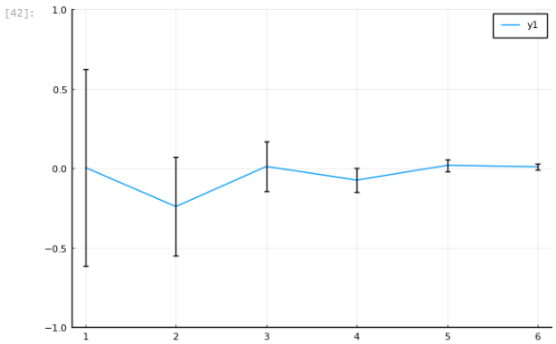


Рис. 2.42: График исходных значений с отклонениями

```
[43]: plot(y, 1:length(y),
          xerr = errs,
          marker = stroke(3,:orange)
      )
```

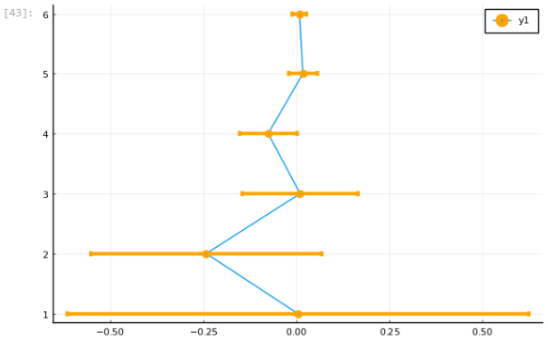


Рис. 2.43: Поворот графика

```
[44]: plot(y,
          ribbon=errs,
          fill=:cyan
      )
```

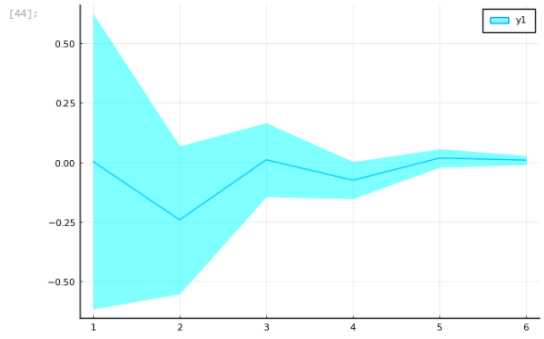


Рис. 2.44: Заполнение цветом

```
[45]: 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)
)
```

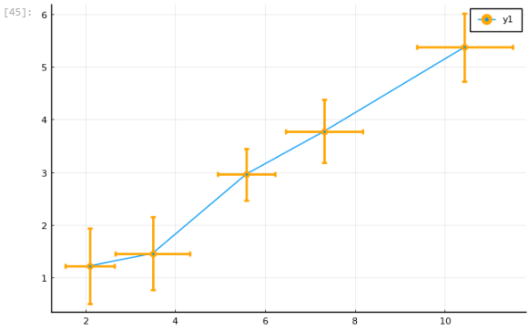


Рис. 2.45: График ошибок по двум осям

```
[46]: plot(x, y,
    xerr = (0.5xerr,2xerr),
    yerr = (0.5yerr,2yerr),
    marker = stroke(2, :orange)
)
```

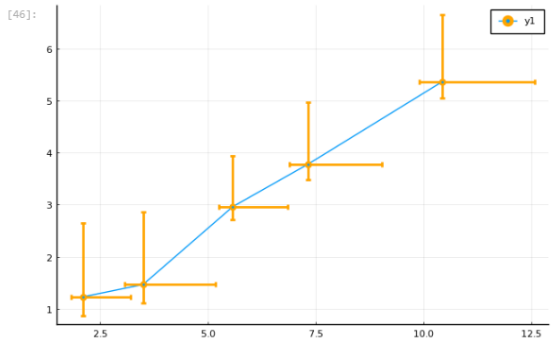


Рис. 2.46: График асимметричных ошибок по двум осям

2.13 Использование пакета Distributions

```
[47]: import Pkg
Pkg.add("Distributions")
using Distributions
```

```
  Resolving package versions...
  Installed DualNumbers ──────────── v0.6.8
  Installed Rmath_jll ────────────── v0.4.0+0
  Installed OpenSpecFun_jll ──────── v0.5.5+0
  Installed Calculus ─────────────── v0.5.1
  Installed PDMats ───────────────── v0.11.31
  Installed StatsFuns ────────────── v1.3.0
  Installed HypergeometricFunctions ─ v0.3.23
  Installed SpecialFunctions ─────── v2.3.1
  Installed Rmath ────────────────── v0.7.1
  Installed QuadGK ───────────────── v2.9.1
  Installed FillArrays ───────────── v1.9.2
  Installed Distributions ────────── v0.25.104
    Updating `C:\Users\Reachna\.julia\environments\v1.9\Project.toml`
  [31c24e10] + Distributions v0.25.104
    Updating `C:\Users\Reachna\.julia\environments\v1.9\Manifest.toml`
  [49dc2e85] + Calculus v0.5.1
  [31c24e10] + Distributions v0.25.104
  [fa6b7ba4] + DualNumbers v0.6.8
  [1a297f60] + FillArrays v1.9.2
  [34004b35] + HypergeometricFunctions v0.3.23
  [90014a1f] + PDMats v0.11.31
  [1fd47b50] + QuadGK v2.9.1
  [79098fc4] + Rmath v0.7.1
  [276daf66] + SpecialFunctions v2.3.1
  [4c63d2b9] + StatsFuns v1.3.0
  [efe28fd5] + OpenSpecFun_jll v0.5.5+0
  [f50d1b31] + Rmath_jll v0.4.0+0
```

Рис. 2.47: Установка пакета Distributions

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

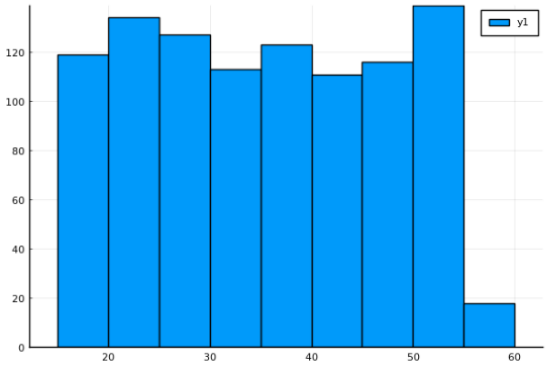


Рис. 2.48: Гистограмма, построенная по массиву случайных чисел

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

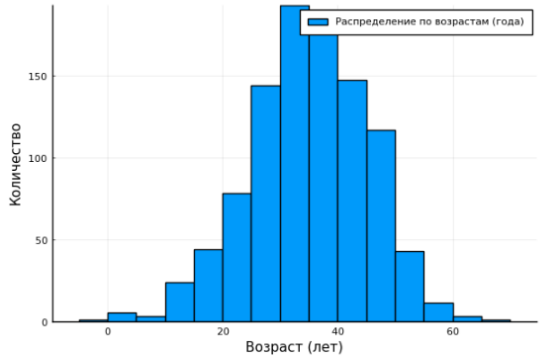


Рис. 2.49: Гистограмма нормального распределения

```
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. 2.50: Гистограмма распределения людей по возрастам

2.14 Подграфики

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

[51]:

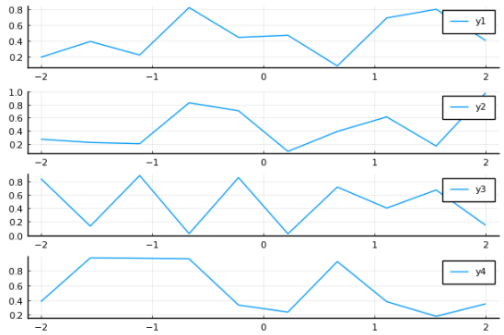


Рис. 2.51: Серия из 4-х графиков в ряд

```
[ ]: plot(x,y,
layout=4
)
```

[52]:

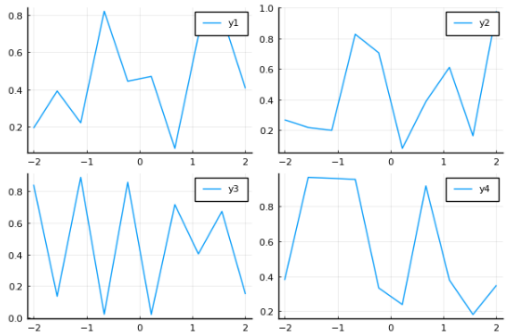


Рис. 2.52: Серия из 4-х графиков в сетке

```
[ ]: plot(x,y,
    size=(600,300),
    layout = grid(4,1,heights=[0.2,0.3,0.4,0.15])
)
```

[53]:

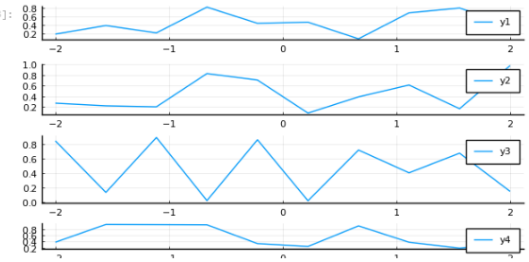


Рис. 2.53: Серия из 4-х графиков разной высоты в ряд

```
# 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
)
```

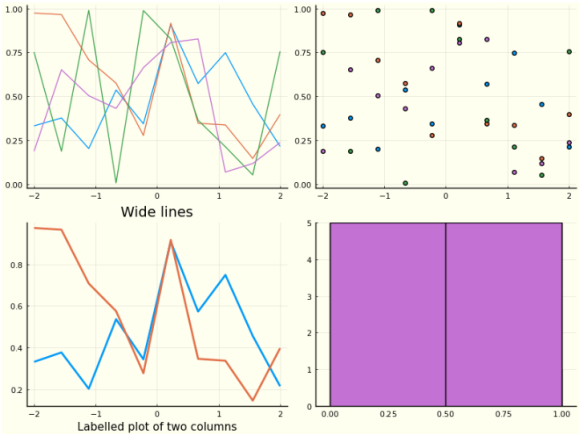


Рис. 2.54: Объединение нескольких графиков в одной сетке

```
[ ]: seriestypes = [:stephist, :sticks, :bar, :hline, :vline, :path]
titles =["stephist" "sticks" "bar" "hline" "vline" "path"]
plot(rand(20,1), st = seriestypes,
    layout = (2,3),
    ticks=nothing,
    legend=false,
    title=titles,
    m=3
)
```

[55]:

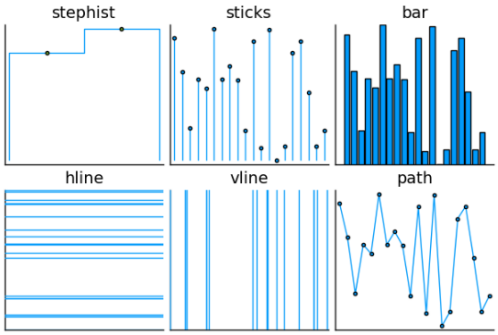


Рис. 2.55: Разнообразные варианты представления данных

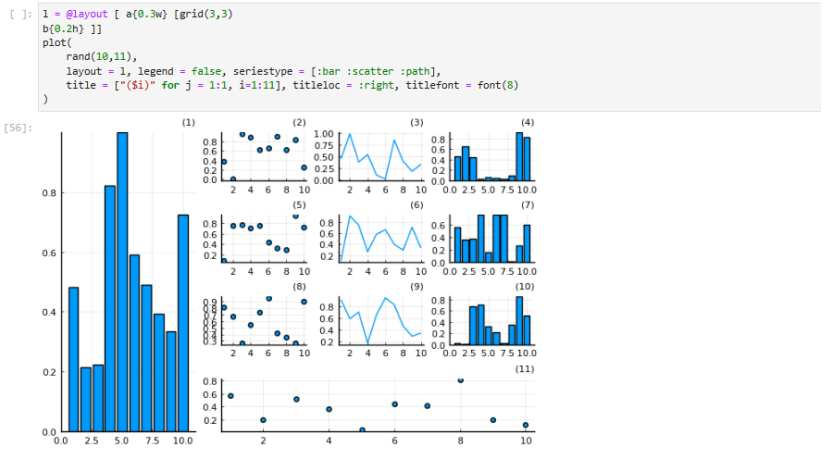


Рис. 2.56: Демонстрация применения сложного макета для построения графиков

2.15 Задания для самостоятельного выполнения

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

```
# Task1
f(x) = sin.(x)
x = collect(range(0, 2*pi, length=200))
y = f(x)

fig1 = plot(x, y,
    title = "Simple Graphic",
    xlabel = "X",
    ylabel = "Y",
    color = "green")
fig2 = scatter(x, y,
    title = "Точечные",
    xlabel = "X",
    ylabel = "Y",
    color = "blue")
fig3 = histogram(f(x),
    title = "Histogram",
    xlabel = "X",
    ylabel = "Y",
    color = "red")
fig4 = histogram(f(x),
    bins = 15,
    title = "Histogram (bins=15)",
    xlabel = "X",
    ylabel = "Y",
    color = "light blue")
plot(fig1, fig2, fig3, fig4, layout=(2, 2), legend=false, size=(800, 600))
```

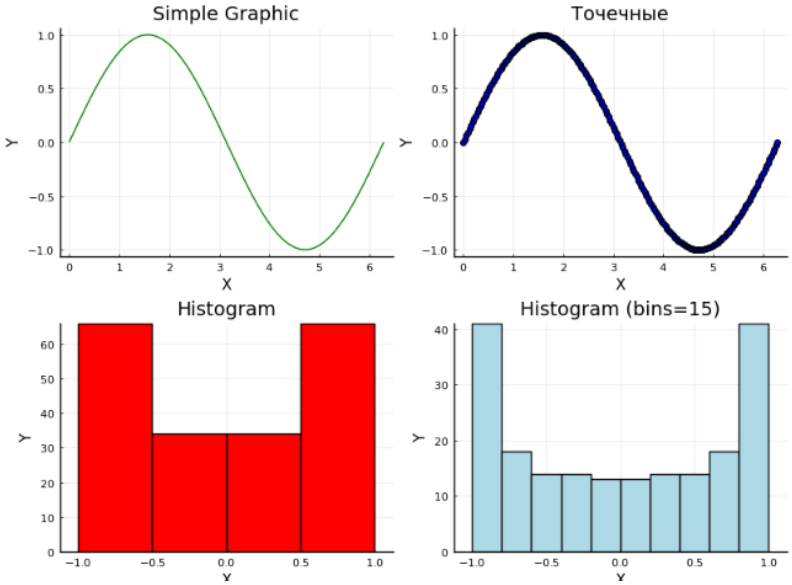


Рис. 2.57: Разные типы графиков

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

```julia
# Task2
f(x) = sin.(x)
x = collect(range(0, 2*pi, length=200))
y = f(x)

fig1 = plot(x,y,
    title = "Solid Line",
    line = (:lightblue, 0.2, 5, :solid),
    xlabel = "X",
    ylabel = "Y",)
fig2 = plot(x,y,
    title = "Dash Line",
    line = (:green, 0.2, 5, :dash),
    xlabel = "X",
    ylabel = "Y",)
fig3 = plot(x,y,
    title = "Dot Line",
    line = (:red, 0.2, 5, :dot),
    xlabel = "X",
    ylabel = "Y",)
fig4 = plot(x,y,
    title = "Dash and Dot Line",
    line = (:purple, 0.2, 5, :dashdot),
    xlabel = "X",
    ylabel = "Y",)
fig5 = plot(x,y,
    title = "Scatter Line",
    line = (:blue, 0.2, 5, :scatter),
    xlabel = "X",
    ylabel = "Y",)
fig6 = plot(x,y,
    title = "Auto Line",
    line = (:darkgreen, 0.2, 5, :auto),
    xlabel = "X",
    ylabel = "Y",)
plot(fig1, fig2, fig3, fig4, fig5, fig6, layout=(3, 2), legend=false, size=(800, 600))
```

Рис. 2.58: Разные линии

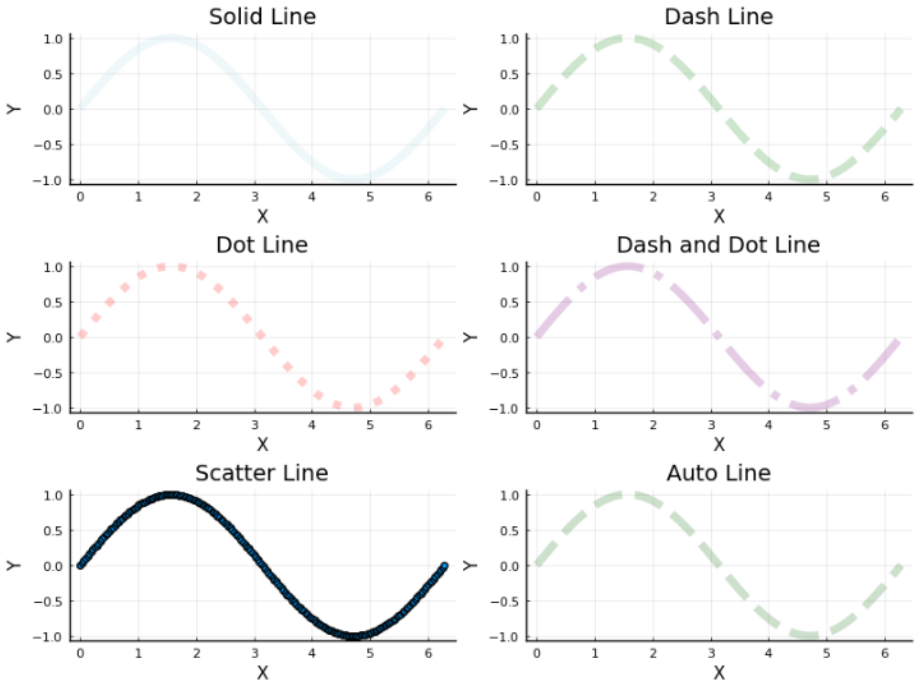


Рис. 2.59: Разные линии

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

```
# Task3
using Plots.PlotMeasures
f(x) = (pi * x.^2).*log.(x)
x = collect(range(0, 2*pi, length=200))
y = f(x)
plotly()
plot(x, y,
    color = "red",
    box = :on,
    foreground_color = "green",
    xaxis = ("Ось X", (0, 6.5), 6.5:-0.5:0),
    yaxis = ("Ось Y", (0, 250), 250:-25:0),
    leg = :right,
    title = "График функции задание №3",
    titlefont = (15, "time new roman", :black),
    top_margin = 10mm,
    right_margin = 10mm,
    legendfontsize = 8,
    guidefont = (12, "time new roman", :black),
    tickfont = (8, "time new roman", :black),
    xrotation = 90)
```

Рис. 2.60: График заданной функции

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

```
# Task4
f(x) = x.^3 - 3*x
x = [-2, -1, 0, 1, 2]
y = f(x)
pyplot()
fig1 = scatter(x, y,
    title = "Point Graphic",
    xlabel = "X",
    ylabel = "Y",
    color = "blue")
fig2 = plot(x, y,
    title="Line Graphic",
    seriestype = :sticks,
    xlabel="X",
    ylabel="Y",
    color="red")
fig3 = plot(x, y,
    title="Line and Point Graphic",
    marker = '.',
    seriestype = :step,
    xlabel="X",
    ylabel="Y",
    color="green")
fig4 = plot(x, y,
    title="Curve Graphic",
    xlabel="X",
    ylabel="Y",
    color="purple")
plot(fig1, fig2, fig3, fig4, layout=(2, 2), legends=false, size=(800, 600))
```

Рис. 2.61: График заданного вектора

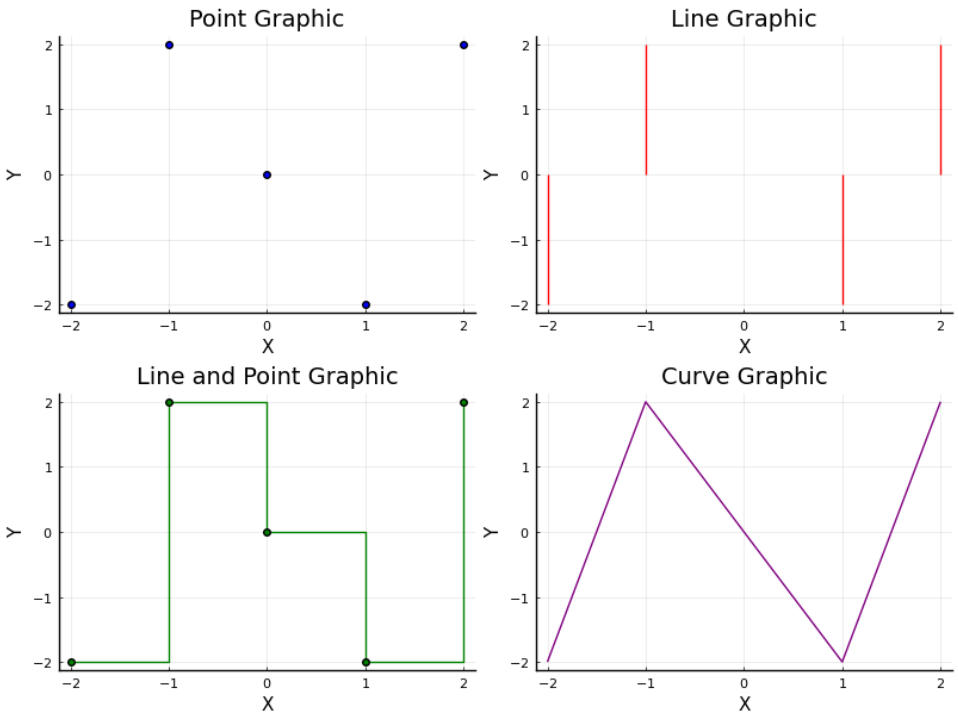


Рис. 2.62: График заданного вектора

Сохраните полученные изображения в файле `figure_familiya.png`, где вместо familiya укажите вашу фамилию.

```
savefig("figure_kim.png")
```

```
sys:1: UserWarning: No data for colormapping provided via 'c'. Parameters 'vmin', 'vmax' will be ignored
"C:\\Users\\Reachna\\computer-analysis\\figure_kim.png"
```

Рис. 2.63: Сохраните полученные изображения

5. Задайте вектор $x = (3, 3.1, 3.2, ..., , 6)$. Постройте графики функций $y_1(x) = \pi x$ и $y_2(x) = exp(x)cos(x)$ в указанном диапазоне значений аргумента x следующим образом:

```
# Task5
f1(x) = pi*x
f2(x) = exp.(x) .* cos.(x)
x = [i for i in 3:0.1:6]
y1 = f1(x)
y2 = f2(x)
pyplot()
plot(x, y1,
    title = "График функций - задание №5",
    xlabel="X",
    ylabel="Y",
    leg = :topright,
    color="blue",
    grid = (:y, :orange, :dot),
    rigth_margin = 20mm)
plot!(x, y2,
    color = "green",
    leg = :bottomright,
    grid = :on,
    box = :on)
```

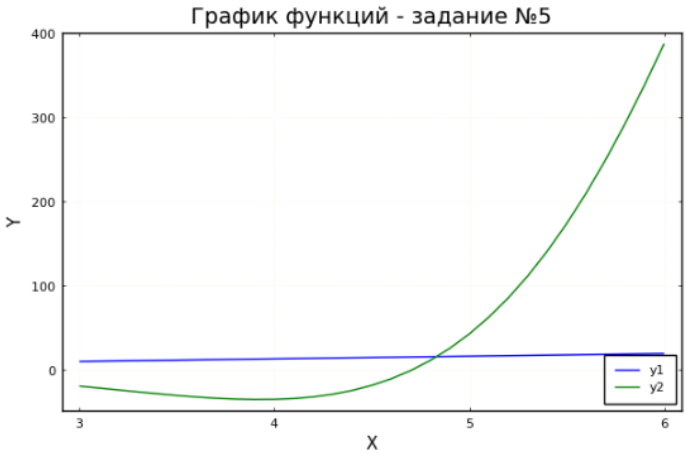


Рис. 2.64: График фунции

```
plot(x, y_1,
    title="График функций (Задание №5)",
    xlabel="X",
    ylabel="Y1",
    leg=:topleft,
    color="blue",
    grid = (:y, :orange, :dot),
    right_margin = 20mm)

plot!(twinx(), x, y_2,
    ylabel = "Y2",
    color="purple",
    leg=:bottomright,
    grid = :on,
    box = :on)
```

Результат:

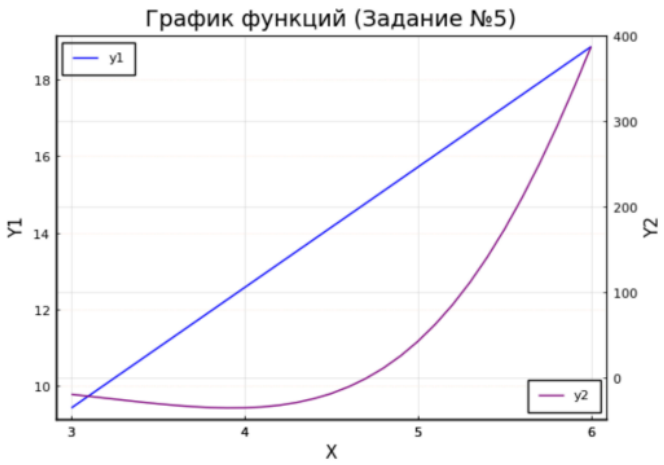


Рис. 2.65: График фунции с двумя осями координат

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

```
# Task6
f(x) = x.^2 + 2*x
x = rand(15)
y = f(x)
n = 15
error = 1.23 * y / sqrt(n)
plot(y, ylims=(-5, 5), err = error)
```

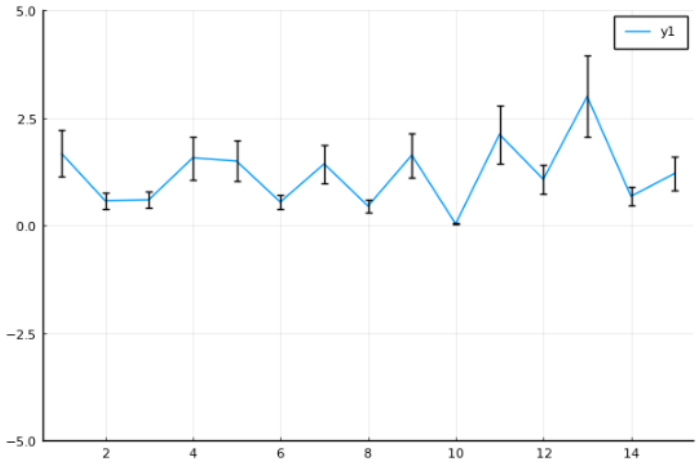


Рис. 2.66: Построение графика некоторых экспериментальных данных

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

```
# Task7
x = rand(30)
y = rand(30)
scatter(x,y,
    title = "Точечный график случайных данных",
    label = "Точки(x,y)",
    leg = :topright,
    xlabel = "X",
    ylabel = "Y")
```

Рис. 2.67: Точечный график случайных данных

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

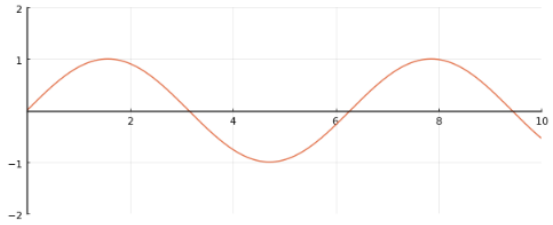
```
# Task8
x = rand(50)
y = rand(50)
z = rand(50)
scatter(x,y,z,
    title = "3-мерный точечный график случайных данных",
    label = "point(x,y,z)",
    xlabel = "X",
    ylabel = "Y",
    zlabel = "Z")
```

Рис. 2.68: 3D Точечный график случайных данных

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

[71]:
```
# Task9
n = 200
x = collect(0*pi : 2*pi/100 : 10*pi+pi/100)

anim = @animate for i in 1:n
    fig = plot(1,
               xlim=(0, 10),
               ylim=(-2, 2),
               c=:purple,
               aspect_ratio=1,
               legend=false,
               framestyle=:origin)
    step = x[1 : i]
    y = sin.(step)
    plot!(step, y)
end
#Сохранила анимацию в gif-файл
gif(anim, "sinus.gif")
```

```
[ Info: Saved animation to C:\Users\Reachna\computer-analysis\sinus.gif
```



[71]:

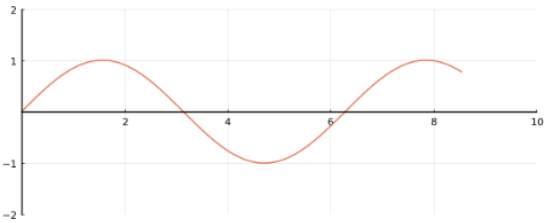


Рис. 2.69: Анимация Синусоида

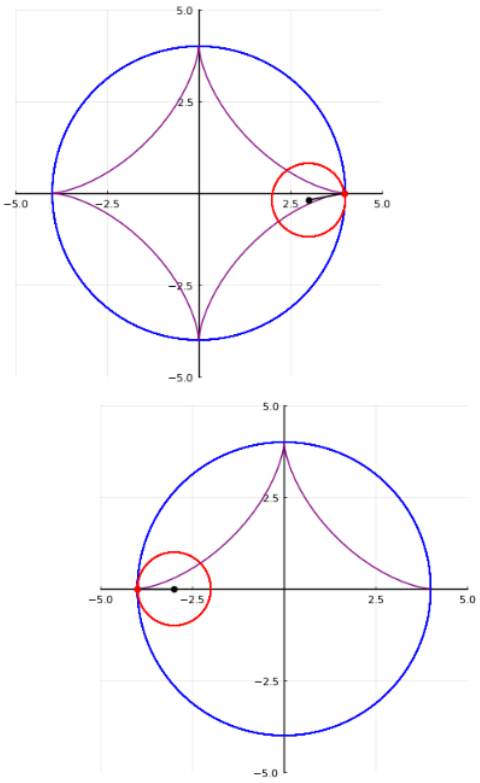
10. Постройте анимированную гипоциклоиду для 2 целых значений модуля k и 2 рациональных значений модуля k

```julia
# Task10
function hypocycloid(x, r0, n)
    # радиус малой окружности:
    r0 = r0
    # коэффициент для построения большой окружности:
    k = x
    # число отсчётов:
    count = n
    # массив значений угла θ:
    θ = collect(0: 2*π/100 : 10*π+2*π/count)

    # массивы значений координат:
    x_1 = r0 * k * cos.(θ)
    y_1 = r0 * k * sin.(θ)
    #В конце сделаем анимацию получившегося изображения
    anim = @animate for i in 1:count
        # зададём оси координат:
        plt=plot(5,
            xlim=(-k-1, k+1),
            ylim=(-k-1, k+1),
            color=:red,
            aspect_ratio=1,
            legend=false,
            framestyle=:origin)
        # большая окружность:
        plot!(plt, x_1, y_1, c=:blue, legend=false)
        t = θ[1 : i]

        # гипоциклоида:
        x = r0 * (k-1) * cos.(t) + r0 * cos.((k-1) * t)
        y = r0 * (k-1) * sin.(t) - r0 * sin.((k-1) * t)
        plot!(x,y, color=:purple)

        # малая окружность:
        x_r0 = r0*(k-1)*cos(t[end]) .+ r0*cos.(θ)
        y_r0 = r0*(k-1)*sin(t[end]) .+ r0*sin.(θ)
        plot!(x_r0, y_r0, color=:red)

        # радиус малой окружности:
        xl_r0 = transpose([r0*(k-1)*cos(t[end]) x[end]])
        yl_r0 = transpose([r0*(k-1)*sin(t[end]) y[end]])
        plot!(xl_r0, yl_r0,
            markershape=:circle,
            markersize=4,
            color=:black)
        scatter!([x[end]],
            [y[end]],
            color=:red,
            markerstrokecolor=:red)
    end
    gif(anim,"hypocycloid.gif")
end
```

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

Рис. 2.70: Гипоциклоида

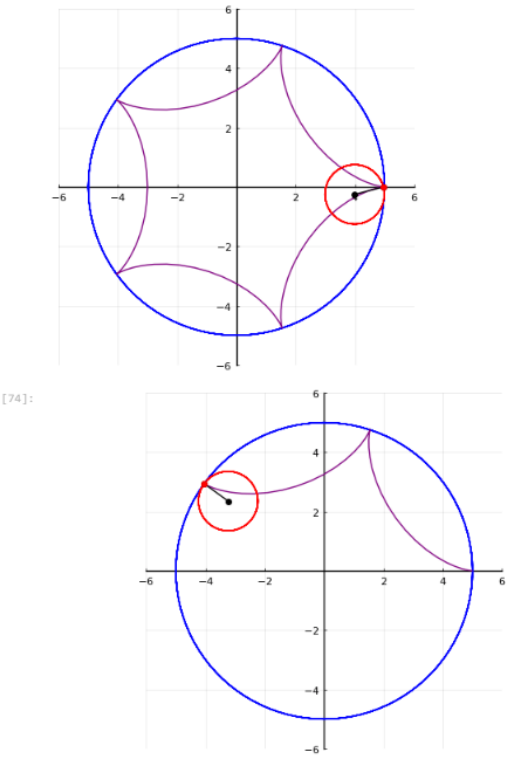
```julia
hypocycloid(4, 1, 100)
```



[73]:

Рис. 2.70: Гипоциклоидная анимация

```
hypocycloid(5, 1, 100)
```



[74]:

Рис. 2.70: Гипоциклоидная анимация

:
```
hypocycloid(1.5, 1, 200)
```

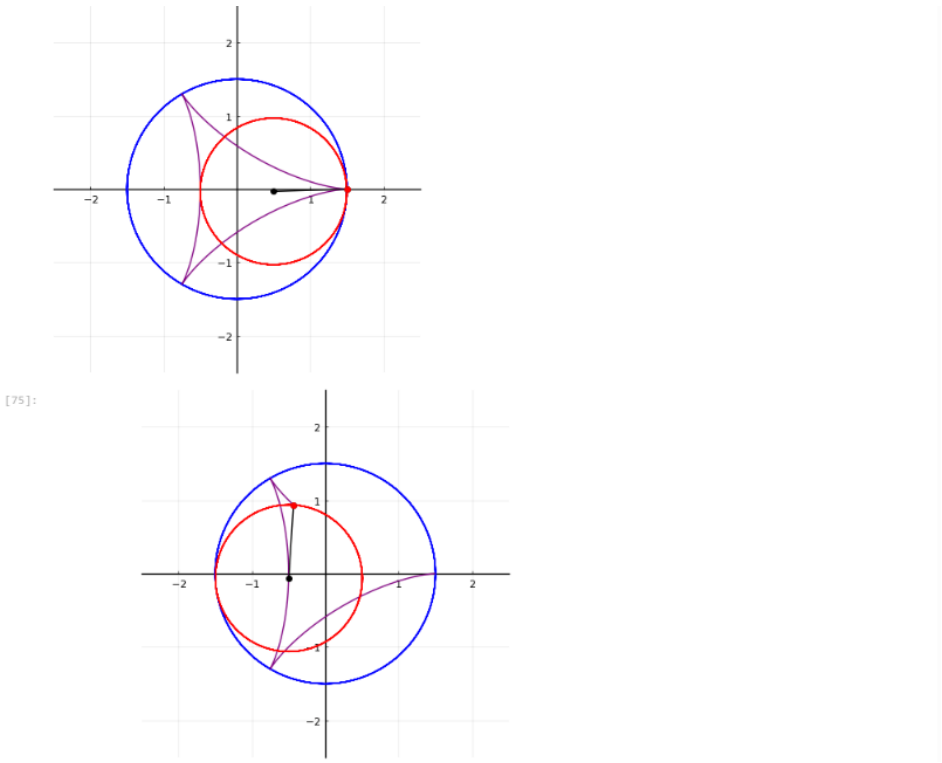


Рис. 2.70: Гипоциклоидная анимация

```
hypocycloid(5.5, 1, 200)
```

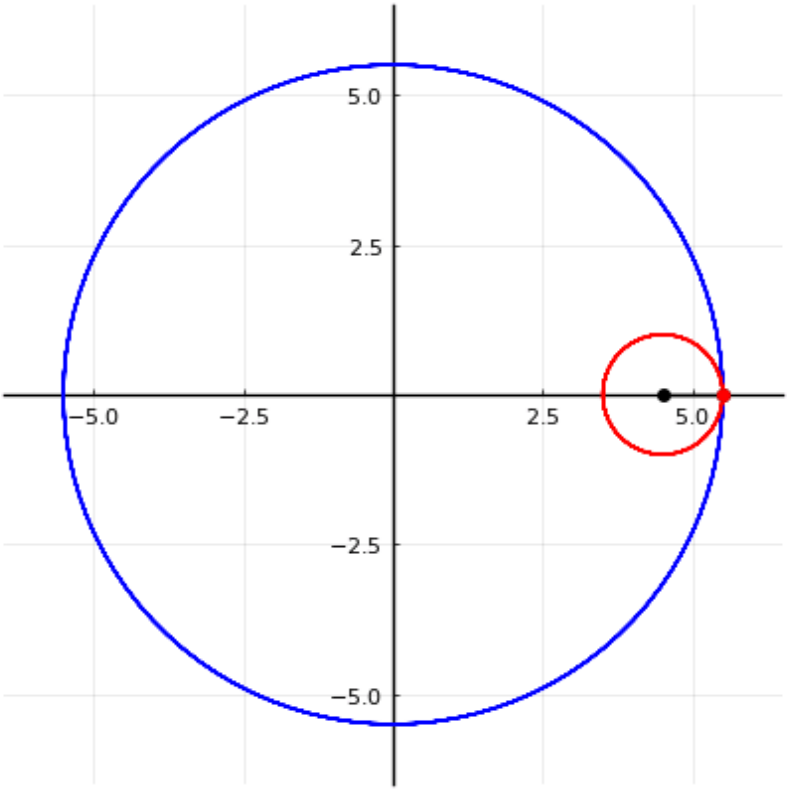


Рис. 2.70: Гипоциклоидная анимация

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

```
# Task11
function epicycloid(x, r0, n)
    # радиус малой окружности:
    r0 = r0
    # коэффициент для построения большой окружности:
    k = x
    # число отсчётов:
    count = n
    # массив значений угла θ:
    θ = collect(0: 2*π/100 : 10*π+2*π/count)

    # массивы значений координат:
    x_1 = r0 * k * cos.(θ)
    y_1 = r0 * k * sin.(θ)
    #В конце сделаем анимацию получившегося изображения
    anim = @animate for i in 1:count
        # зададём оси координат:
        plt=plot(5,
            xlim=(-2*r0*k*2, 2*r0*k*2),
            ylim=(-2*r0*k*2, 2*r0*k*2),
            color=:red,
            aspect_ratio=1,
            legend=false,
            framestyle=:origin)
        # большая окружность:
        plot!(plt, x_1, y_1, c=:blue, legend=false)
        t = θ[1 : i]

        # гипоциклоида:
        x = r0 * (k + 1) * cos.(t) - r0 * cos.((k + 1) * t)
        y = r0 * (k + 1) * sin.(t) - r0 * sin.((k + 1) * t)
        plot!(x,y, color=:purple)

        # малая окружность:
        x_r0 = r0*(k + 1)*cos(t[end]) .- r0*cos.(θ)
        y_r0 = r0*(k + 1)*sin(t[end]) .- r0*sin.(θ)
        plot!(x_r0, y_r0, color=:red)

        # радиус малой окружности:
        xl_r0 = transpose([r0*(k + 1)*cos(t[end]) x[end]])
        yl_r0 = transpose([r0*(k + 1)*sin(t[end]) y[end]])
        plot!(xl_r0, yl_r0,
            markershape=:circle,
            markersize=4,
            color=:black)
        scatter!([x[end]],
            [y[end]],
            color=:red,
            markerstrokecolor=:red)
    end
    gif(anim,"epicycloid.gif")
end
```

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

Рис. 2.71: Эпициклоида

```
# первый вариант
epicycloid(2, 1, 100)
```

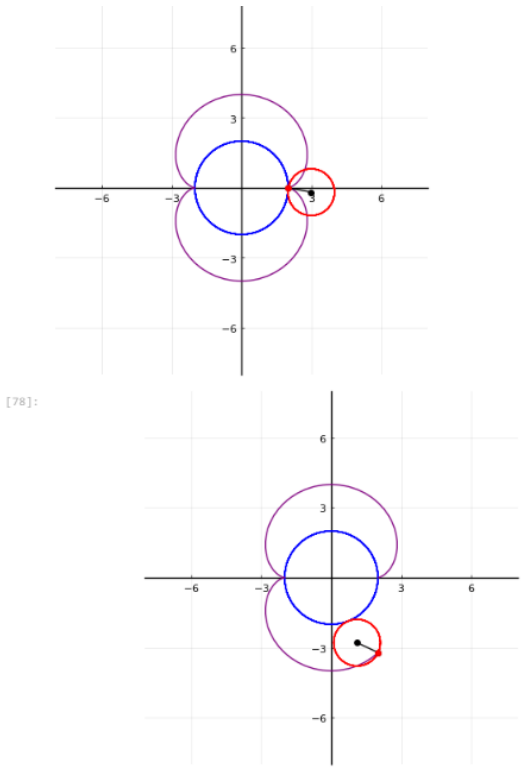


Рис. 2.71: Эпициклоидная анимация

```
# второй вариант
epicycloid(3, 1, 100)
```

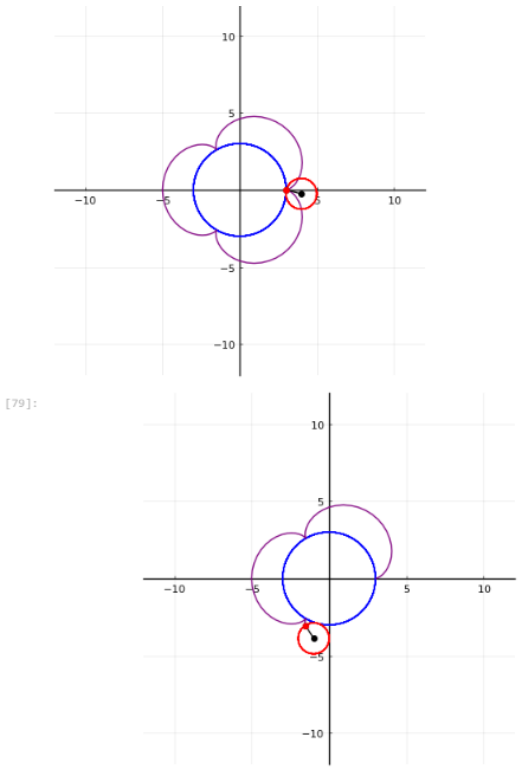


Рис. 2.71: Эпициклоидная анимация

```
# третий вариант
epicycloid(5.5, 1, 500)
```

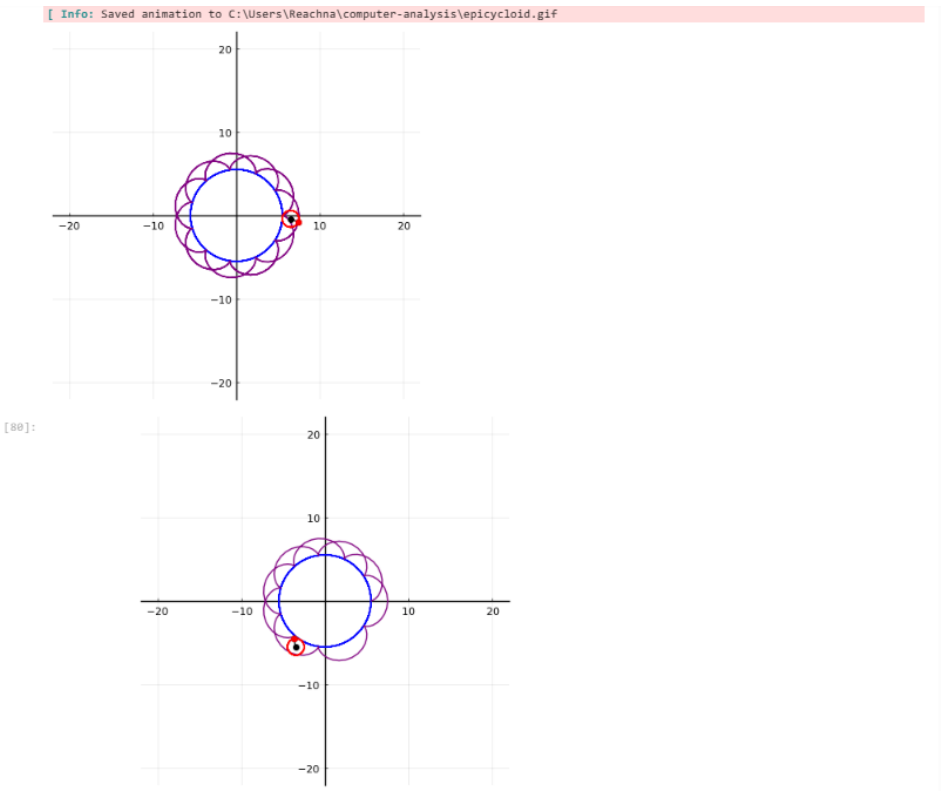


Рис. 2.71: Эпициклоидная анимация

```
# четвертый вариант
epicycloid(3.8, 1, 500)
```

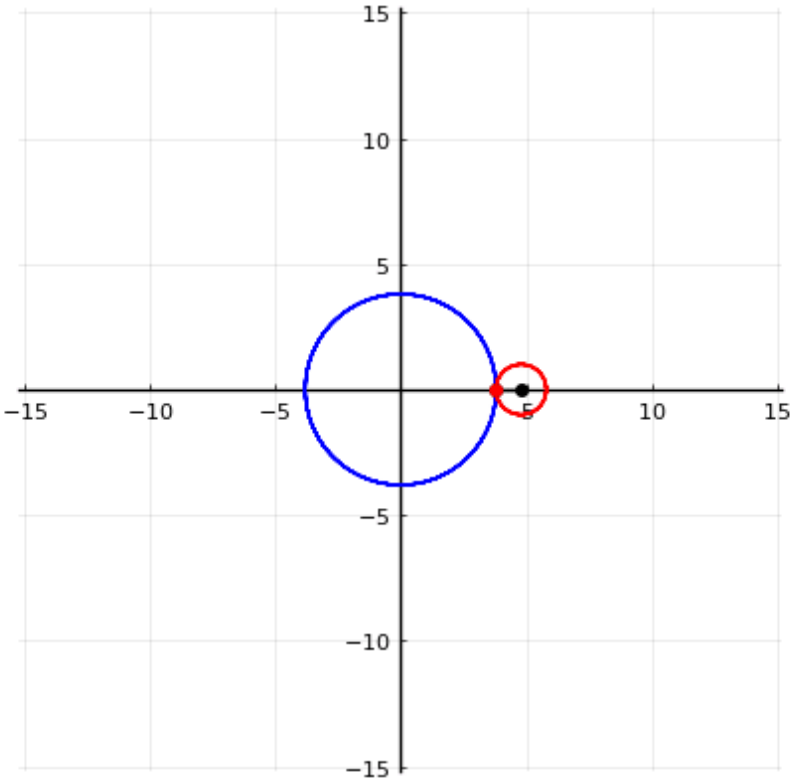


Рис. 2.71: Эпициклоидная анимация

3 Листинги программы

```
using Pkg
Pkg.add("Plots")
Pkg.add("PyPlot")
Pkg.add("Plotly")
Pkg.add("UnicodePlots")
# подключаем для использования Plots:
using Plots
Построение графика функции
# задание функции:
f(x) = (3x.^2 + 6x .- 9).*exp.(-0.3x)
# генерирование массива значений x в диапазоне от -5 до 10 с шагом 0,1
# (шаг задан через указание длины массива):
x = collect(range(-5,10,length=151))
# генерирование массива значений y:
y = f(x)
# указывается, что для построения графика используется gr():
gr()
# задание опций при построении графика
# (название кривой, подписи по осям, цвет графика):
plot(x,y,
    title="A simple curve",
    xlabel="Variable x",
```

```
    ylabel="Variable y",
    color="blue")
using Plots
# указывается, что для построения графика используется pyplot():
pyplot()
# задание опций при построении графика
# (название кривой, подписи по осям, цвет графика):
plot(x,y,
    title="Простая кривая",
    xlabel="Переменная x",
    ylabel="Переменная y",
    color="blue")
using Plots
# указывается, что для построения графика используется plotly():
plotly()
# задание опций при построении графика
# (название кривой, подписи по осям, цвет графика):
plot(x,y,
    title="Простая кривая",
    xlabel="Переменная x",
    ylabel="Переменная y",
    color="blue")
# using Plots
# unicodeplots()
using UnicodePlots #т.к. использование просто в качестве бэкенда не сработало, вы
# задание опций при построении графика
# (название кривой, подписи по осям, цвет графика):
lineplot(x, y,
    title="Простая кривая",
```

```
    xlabel="Переменная x",
    ylabel="Переменная y",
    color= :blue)
# рассмотрим дополнительные возможности пакетов для работы с графиками
using Plots
# указывается, что для построения графика используется pyplot():
pyplot()
# задание функции sin(x):
sin_theor(x) = sin(x)

# построение графика функции sin(x):
plot(sin_theor)
pyplot()
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)
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",
    # 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",
    leg=:topright,
    line=(:black, 1.0, 2, :dash)
)
# сохранение графика в файле в формате pdf или png:
Plots.savefig("taylor.pdf")
Plots.savefig("taylor.png")
# параметры распределения точек на плоскости:
```

```
x = range(1,10,length=10)
y = rand(10)
# параметры построения графика:
plot(x, y,
    seriestype = :scatter,
    title = "Точечный график",
    xlabel = "x",
    ylabel = "y",
    # подпись в легенде, цвет и тип линии:
    label = "y1, теоретическое значение",
    leg = :topright
)
# 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,
    # подписи по осям:
    xlabel = "x",
    ylabel = "y",
    # подпись в легенде, цвет и тип линии:
    label = "y1, теоретическое значение",
    title = "Точечный график")
# 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,
    xlabel = "x",
    ylabel = "y",
    zlabel = "z",
    # подпись в легенде, цвет и тип линии:
    label = "sin(x), теоретическое значение",
    title = "Точечный график")
```

5.2.4. 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,legend=false)
# 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:
f1 = 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, f1, linewidth=3, color=:red)
using Plots.PlotMeasures
```

```
# пример случайной траектории
# (заданы обозначение траектории, легенда вверху справа, без сетки)
plot(randn(100),
    xlabel = "X",
    ylabel="y1",
    label = "1st path",
    leg=:topright,
    grid = :off,
    right_margin = 20mm
)
# пример добавления на график второй случайной траектории
# (задано обозначение траектории и её цвет, легенда снизу справа, без сетки)
# задана рамка графика
plot!(twinx(), randn(100)*10,
    c=:red,
    ylabel="y2",
    label = "2nd path",
    leg=:bottomright,
    grid = :off,
    box = :on,
    size=(600, 400)
)
# функция в полярных координатах:
r(tetha) = 1 + cos(tetha) * sin(tetha)^2
# полярная система координат:
tetha = range(0, stop=2π, length=50)
# график функции, заданной в полярных координатах:
plot(tetha, r.(tetha),
    proj=:polar,
```

```
    lims=(0,1.5)
)
# параметрическое уравнение:
x1(t) = sin(t)
y1(t) = sin(2t)
# построение графика:
plot(x1, y1, 0, 2π, leg=false, fill=(0,:orange))
# параметрическое уравнение
t = range(0, stop=10, length=1000)
x = cos.(t)
y = sin.(t)
z = sin.(5t)
# построение графика:
plot(x, y, z)
# построение графика поверхности:
f(x,y) = x^2 + y^2
x = -10:10
y = x
surface(x, y, f)
# построение графика поверхности:
f(x,y) = x^2 + y^2
x = -10:10
y = x
plot(x, y, f,
    linetype=:wireframe
)
f2(x,y) = x^2 + y^2
x = -10:0.1:10
y = x
```

```
plot(x, y, f2,
linetype = :surface
)
x = range(-2,stop=2,length=100)
y = range(sqrt(2),stop=2,length=100)
f3(x,y) = x*y-x-y+1
plot(x, y, f3,
    linetype = :surface,
    c=cgrad([:red,:blue]),
    camera=(-30,30),
)
```

5.2.9. Линии уровня

```
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,
)
contour(x, y, g)
p = contour(x, y, g,
fill=true)
plot(p)
# определение переменных:
X0 = range(-2, stop=2, length=100)
Y0 = range(-2, stop=2, length=100)
# определение функции:
h(x, y) = x^3 - 3x + y^2
# построение поверхности:
plot(X0, Y0, h,
```

```
    linetype = :surface
)
# построение линий уровня:
contour(X0, Y0, h)
# градиент:
xs = range(-2, stop=2, length=12)
ys = range(-2, stop=2, length=12)
# производная от исходной функции:
dh(x, y) = [3x^2-3; 2y]/25
xxs = [x for x in xs for y in ys]
yys = [y for x in xs for y in ys]

quiver!(xxs, yys, quiver=dh, c=:blue)
# коррекция области видимости графика:
xlims!(-2, 2)
ylims!(-2, 2)
quiver!(xxs, yys, quiver=dh, c=:blue)
pyplot()
# построение поверхности:
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
# радиус малой окружности:
r1 = 1
```

```
# коэффициент для построения большой окружности:
k = 3
# число отсчётов:
n = 100
# массив значений угла tetha:
# theta from 0 to 2pi ( + a little extra)
tetha = collect(0:2*π/100:2*π+2*π/100)
# массивы значений координат:
X = r1*k*cos.(tetha)
Y = r1*k*sin.(tetha)
# задаём оси координат:
plt = plot(5, xlim=(-4,4), ylim=(-4,4), c=:red, aspect_ratio=1,
legend=false, framestyle=:origin)
# большая окружность:
plot!(plt, X,Y, c=:blue, legend=false)
i = 50
t = tetha[1:i]
# гипоциклоида:
x = r1*(k-1)*cos.(t) + r1*cos.((k-1)*t)
y = r1*(k-1)*sin.(t) - r1*sin.((k-1)*t)
plot!(x,y, c=:red)
# малая окружность:
xc = r1*(k-1)*cos(t[end]) .+ r1*cos.(tetha)
yc = r1*(k-1)*sin(t[end]) .+ r1*sin.(tetha)
plot!(xc,yc,c=:black)
# радиус малой окружности:
xl = transpose([r1*(k-1)*cos(t[end]) x[end]])
yl = transpose([r1*(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)
anim = @animate for i in 1:n
# задаём оси координат:
plt=plot(5, xlim=(-4,4), ylim=(-4,4), c=:red, aspect_ratio=1, legend=false, frame
# большая окружность:
plot!(plt, X, Y, c=:blue, legend=false)
t = tetha[1:i]

# гипоциклоида:
x = r1*(k-1)*cos.(t) + r1*cos.((k-1)*t)
y = r1*(k-1)*sin.(t) - r1*sin.((k-1)*t)
plot!(x,y, c=:red)

# малая окружность:
xc = r1*(k-1)*cos(t[end]) .+ r1*cos.(tetha)
yc = r1*(k-1)*sin(t[end]) .+ r1*sin.(tetha)
plot!(xc, yc, c=:black)

# радиус малой окружности:
xl = transpose([r1*(k-1)*cos(t[end]) x[end]])
yl = transpose([r1*(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")
using Statistics
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),
)
plot(y,
ylims = (-1,1),
err = errs
)
plot(y, 1:length(y),
    xerr = errs,
    marker = stroke(3,:orange)
)
plot(y,
    ribbon=errs,
    fill=:cyan
)
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)
```

```
)
plot(x, y,
    xerr = (0.5xerr,2xerr),
    yerr = (0.5yerr,2yerr),
    marker = stroke(2, :orange)
)
import Pkg
Pkg.add("Distributions")
using Distributions
pyplot()
ages = rand(15:55, 1000)
histogram(ages)
d = Normal(35.0, 10.0)
ages = rand(d, 1000)
histogram(
    ages,
    label="Распределение по возрастам (года)",
    xlabel = "Возраст (лет)",
    ylabel= "Количество"
)
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)"
)
```

5.2.14. Подграфики

```
# подгружаем pyplot():
pyplot()
# построение серии графиков:
x=range(-2,2,length=10)
y = rand(10,4)
plot(x,y,
    layout=(4,1)
)
plot(x,y,
layout=4
)
plot(x,y,
    size=(600,300),
    layout = grid(4,1,heights=[0.2,0.3,0.4,0.15])
)
# 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
)
seriestypes = [:stephist, :sticks, :bar, :hline, :vline, :path]
titles =["stephist" "sticks" "bar" "hline" "vline" "path"]
plot(rand(20,1), st = seriestypes,
    layout = (2,3),
    ticks=nothing,
    legend=false,
    title=titles,
    m=3
)
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)
)
f4(x) = sin.(x)
# сгенерируем масссив значений x в диапазоне от 0 до П
```

```
x = collect(range(0, 2*pi, length=200))
# зададим функцию
y = f4(x)

fig1 = plot(x, y,
    title="Simple graphic",
    xlabel="X",
    ylabel="Y",
    color="red")
fig2 = scatter(x, y,
    title="Point graphic",
    xlabel="X",
    ylabel="Y",
    color="red")
fig3 = histogram(f4(x),
    title="Histogram",
    xlabel="X",
    ylabel="Y",
    color="purple")
fig4 = histogram(f4(x),
    bins = 10,
    title="Histogram (bins=10)",
    xlabel="X",
    ylabel="Y",
    color="green")
plot(fig1, fig2, fig3, fig4, layout=(2, 2), legends=false, size=(800, 600))
f4(x) = sin.(x)
# сгенерируем масссив значений x в диапазоне от 0 до П с шагом 0.1
x = collect(range(0, 2*pi, length=200))
```

```
# зададим функцию
y = f4(x)

fig1 = plot(x, y,
    title="Solid Line",
    line = (:blue, 0.2, 5, :solid),
    xlabel="X",
    ylabel="Y")
fig2 = plot(x, y,
    title="Dash Line",
    line = (:blue, 0.2, 5, :dash),
    xlabel="X",
    ylabel="Y")
fig3 = plot(x, y,
    line = (:green, 0.2, 5, :dot),
    title="Dot Line",
    xlabel="X",
    ylabel="Y")
fig4 = plot(x, y,
    line = (:green, 0.2, 5, :dashdot),
    title="Dash and Dot Line",
    xlabel="X",
    ylabel="Y")
fig5 = plot(x, y,
    line = (:blue, 0.2, 5, :scatter),
    title="Scatter Line",
    xlabel="X",
    ylabel="Y")
fig6 = plot(x, y,
```

```
    line = (:blue, 0.2, 5, :auto),
    title="Auto Line",
    xlabel="X",
    ylabel="Y")
plot(fig1, fig2, fig3, fig4, fig5, fig6, layout=(3, 2),
legends=false, size=(800, 600))
using Plots.PlotMeasures
f5(x) = (pi*x.^2).*log.(x)
# сгенерируем масссив значений x в диапазоне от 0 до 2П
x = collect(range(0, 2*pi, length=200))
# зададим функцию
y = f5(x)
plotly()
plot(x,y, color="red", box = :on, foreground_color="green",
    xaxis = ("Ось x", (0, 6.5), 6.5:-0.5:0), yaxis = ("Ось y",
    (0, 250), 250:-25:0),
    leg=:right,
    title="График функции (задание №3)",
    titlefont = (15, "montserrat", :black),
    top_margin = 10mm,
    right_margin = 5mm,
    legendfontsize = 8,
    guidefont = (12, "montserrat", :black),
    tickfont = (8, "montserrat",:black),
    xrotation = 90)
# зададим функцию
f(x) = x.^3 - 3*x
x = [-2, -1, 0, 1, 2]
y = f(x)
```

```
using Plots.PlotMeasures
pyplot()
fig1 = scatter(x, y,
    title="Point graphic",
    xlabel="X",
    ylabel="Y",
    color="green")
fig2 = plot(x, y,
    title="Sticks graphic",
    seriestype = :sticks,
    xlabel="X",
    ylabel="Y",
    color="red")
fig3 = plot(x, y,
    title="Step graphic",
    marker = '.',
    seriestype = :step,
    xlabel="X",
    ylabel="Y",
    color="purple")
fig4 = plot(x, y,
    title="Simple graphic",
    xlabel="X",
    ylabel="Y",
    color="blue")
plot(fig1, fig2, fig3, fig4, layout=(2, 2), legends=false,
size=(800, 600))
# Сохраню полученное изображение
savefig("figure_solomko.png")
```

```
f_1(x) = pi*x
f_2(x) = exp.(x).*cos.(x)
x = [i for i in 3:0.1:6]
y_1 = f_1(x)
y_2 = f_2(x)

pyplot()
plot(x, y_1,
    title="График функций (Задание №5)",
    xlabel="X",
    ylabel="Y",
    leg=:topleft,
    color="blue",
    grid = (:y, :orange, :dot),
    right_margin = 20mm)

plot!(x, y_2,
    color="purple",
    leg=:bottomright,
    grid = :on,
    box = :on)
plot(x, y_1,
    title="График функций (Задание №5)",
    xlabel="X",
    ylabel="Y1",
    leg=:topleft,
    color="blue",
    grid = (:y, :orange, :dot),
```

```
    right_margin = 20mm)

plot!(twinx(), x, y_2,
    ylabel = "Y2",
    color="purple",
    leg=:bottomright,
    grid = :on,
    box = :on)

f_3(x) = x.^2 - 2*x
x = rand(20)
y_3 = f_3(x)
n = 20
error = 1.23 * y_3 / sqrt(n)
plot(y_3, ylims=(-5, 5), err = error)
x = rand(20)
y = rand(20)
scatter(x, y,
    title = "Точечный график случайных чисел",
    label = "Точки (x; y)",
    leg = :topright,
    xlabel="X",
    ylabel="Y")
x = rand(50)
y = rand(50)
z = rand(50)
scatter(x, y, z,
    xlabel = "x",
    ylabel = "y",
```

```
    zlabel = "z",
    label = "Точки (x; y; z)",
    title = "Точечный 3D-график")
n = 300
x = collect(0*pi : 2*pi/100 : 10*pi+pi/100)
anim = @animate for i in 1:n
    fig = plot(1,
                xlim=(0, 10),
                ylim=(-2, 2),
                c=:purple,
                aspect_ratio=1,
                legend=false,
                framestyle=:origin)
    step = x[1 : i]
    y = sin.(step)
    plot!(step, y)
end
#Сохранила анимацию в gif-файл
gif(anim, "sinusoida.gif")

function hypocycloid(x, r0, n)
    # радиус малой окружности:
    r0 = r0
    # коэффициент для построения большой окружности:
    k = x
    # число отсчётов:
    count = n
    # массив значений угла tetha:
    tetha = collect(0: 2*π/100 : 10*π+2*π/count)
```

```
# массивы значений координат:
x_1 = r0 * k * cos.(tetha)
y_1 = r0 * k * sin.(tetha)
#В конце сделаем анимацию получившегося изображения
anim = @animate for i in 1:count
    # задаём оси координат:
    plt=plot(5,
        xlim=(-k-1, k+1),
        ylim=(-k-1, k+1),
        color=:red,
        aspect_ratio=1,
        legend=false,
        framestyle=:origin)
    # большая окружность:
    plot!(plt, x_1, y_1, c=:blue, legend=false)
    t = tetha[1 : i]

    # гипоциклоида:
    x = r0 * (k-1) * cos.(t) + r0 * cos.((k-1) * t)
    y = r0 * (k-1) * sin.(t) - r0 * sin.((k-1) * t)
    plot!(x,y, color=:purple)

    # малая окружность:
    x_r0 = r0*(k-1)*cos(t[end]) .+ r0*cos.(tetha)
    y_r0 = r0*(k-1)*sin(t[end]) .+ r0*sin.(tetha)
    plot!(x_r0, y_r0, color=:red)

    # радиус малой окружности:
```

```
        xl_r0 = transpose([r0*(k-1)*cos(t[end]) x[end]])
        yl_r0 = transpose([r0*(k-1)*sin(t[end]) y[end]])
        plot!(xl_r0, yl_r0,
            markershape=:circle,
            markersize=4,
            color=:black)
        scatter!([x[end]],
            [y[end]],
            color=:red,
            markerstrokecolor=:red)
    end
    gif(anim,"hypocycloid.gif")
end
# первый вариант
hypocycloid(4, 1, 100)
# второй вариант
hypocycloid(5, 1, 100)
# третий вариант
hypocycloid(1.5, 1, 200)
# четвертый вариант
hypocycloid(5.5, 1, 200)

function epicycloid(x, r0, n)
    # радиус малой окружности:
    r0 = r0
    # коэффициент для построения большой окружности:
    k = x
    # число отсчётов:
    count = n
```

```
# массив значений угла tetha:
tetha = collect(0: 2*π/100 : 10*π+2*π/count)

# массивы значений координат:
x_1 = r0 * k * cos.(tetha)
y_1 = r0 * k * sin.(tetha)
#В конце сделаем анимацию получившегося изображения
anim = @animate for i in 1:count
    # задаём оси координат:
    plt=plot(5,
        xlim=(-2*r0*k*2, 2*r0*k*2),
        ylim=(-2*r0*k*2, 2*r0*k*2),
        color=:red,
        aspect_ratio=1,
        legend=false,
        framestyle=:origin)
    # большая окружность:
    plot!(plt, x_1, y_1, c=:blue, legend=false)
    t = tetha[1 : i]

    # гипоциклоида:
    x = r0 * (k + 1) * cos.(t) - r0 * cos.((k + 1) * t)
    y = r0 * (k + 1) * sin.(t) - r0 * sin.((k + 1) * t)
    plot!(x,y, color=:purple)

    # малая окружность:
    x_r0 = r0*(k + 1)*cos(t[end]) .- r0*cos.(tetha)
    y_r0 = r0*(k + 1)*sin(t[end]) .- r0*sin.(tetha)
    plot!(x_r0, y_r0, color=:red)
```

```
        # радиус малой окружности:
        xl_r0 = transpose([r0*(k + 1)*cos(t[end]) x[end]])
        yl_r0 = transpose([r0*(k + 1)*sin(t[end]) y[end]])
        plot!(xl_r0, yl_r0,
            markershape=:circle,
            markersize=4,
            color=:black)
        scatter!([x[end]],
            [y[end]],
            color=:red,
            markerstrokecolor=:red)
    end
    gif(anim,"epicycloid.gif")
end
# первый вариант
epicycloid(2, 1, 100)
# второй вариант
epicycloid(3, 1, 100)
# третий вариант
epicycloid(5.5, 1, 500)
# четвертый вариант
epicycloid(3.8, 1, 500)
# четвертый вариант
epicycloid(3.8, 1, 500)
```

4 Вывод

Освоила синтаксис языка Julia для построения графиков.