

# **Лабораторная работа №5 Построение графиков**

## **Статический анализ данных**

---

Коняева Марина Александровна

НФИбд-01-21

Студ. билет: 1032217044

2024

RUDN

## Цели лабораторной работы

Освоить синтаксис языка Julia для построения графиков.

Julia поддерживает несколько пакетов для работы с графиками. Использование того или иного пакета зависит от целей, преследуемых пользователем при построении. Стандартным для Julia является пакет Plots.jl. Одним из преимуществ Plots.jl является то, что он позволяет легко менять вызовы библиотек работы с графикой: (gr (), pyplot (), plotlyjs ()).

## Задачи лабораторной работы

1. Используя Jupyter Lab, повторите примеры из раздела 5.2.
2. Выполните задания для самостоятельной работы (раздел 5.4).

## **Выполнение лабораторной работы**

---

# Основные пакеты для работы с графиками в Julia

- Перед использованием графических пакетов следует их установить и подключить в Julia.

```
using Pkg
Pkg.add("Plots")
Pkg.add("PyPlot")
Pkg.add("Plotly")
Pkg.add("UnicodePlots")

Updating registry at 'C:\Users\User\.julia\registries\General.toml'
Resolving package versions...
No Changes to 'C:\Users\User\.julia\environments\v1.10\Project.toml'
No Changes to 'C:\Users\User\.julia\environments\v1.10\Manifest.toml'
Resolving package versions...
Installed PyCall - v1.96.4
Installed PyPlot - v2.11.5
Updating 'C:\Users\User\.julia\environments\v1.10\Project.toml'
[d330ff01b] + PyPlot v2.11.5
Updating 'C:\Users\User\.julia\environments\v1.10\Manifest.toml'
[438e73bf] + PyCall v1.96.4
[d330ff01b] + PyPlot v2.11.5
Building PyCall + [C:\Users\User\.julia\scratchspaces\44cfe95a-1eb2-52ea-b672-e2afdf69b78f\981]
```

Рис. 1: Установка пакетов

# Основные пакеты для работы с графиками в Julia

2. Повторим пример: рассмотрим построение графика функции  $y(x) = (3x^2 + 6x - 9) \cdot e^{-0.3x}$  разными способами. Фактически для построения графика функции требуется иметь массив соответствующих значений  $x$  и  $y$ . А именно зададим исходную функцию (используются поэлементные операции над векторами), определим массив значений  $x$ ,  $y$ . Далее показано сравнение построения графиков при использовании разных программно-аппаратных частей Plots.jl. По умолчанию используется `gr()`, стандартным образом можно задать различные опции при построении графика.

```
# подключаем для использования Plots:
using Plots
# задание функции:
y(x) = (3x.^2 + 6x .- 9).*exp.(-0.3x)
# генерирование массива значений x в диапазоне от -5 до 10 с шагом 0.1
# (здесь задан через указанное значение 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
# (надпись кривой, подписи по осям, цвет графика):
plot(x,y,
      title="A simple curve",
      xlabel="Variable x",
      ylabel="Variable y",
      color="blue")
```

A simple curve



3. Повторим примеры из пункта 2, учитывая, если требуется, чтобы на графике были надписи на русском языке, то лучше воспользоваться pyplot(), и коррекцию содержания опций при построении графика.

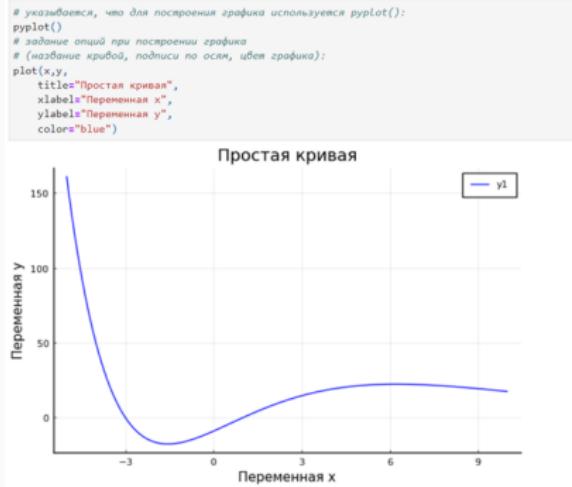
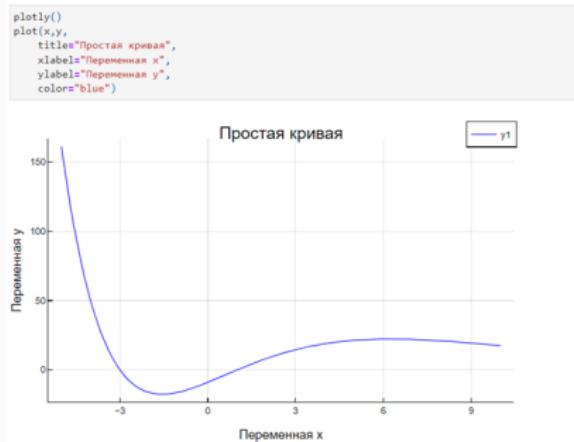


Рис. 3: Построение графика функции

4. Выполним упражнение, построим график функции  $y(x) = (3x^2 + 6x - 9)^{0.3}$  при помощи `plotly()`.



**Рис. 4:** Построение графика функции

# Основные пакеты для работы с графиками в Julia

5. Выполним упражнение, построим график функции  $y(x) = (3x^2 + 6x - 9)x^{-0.3}$  при помощи `unicodeplots()`.

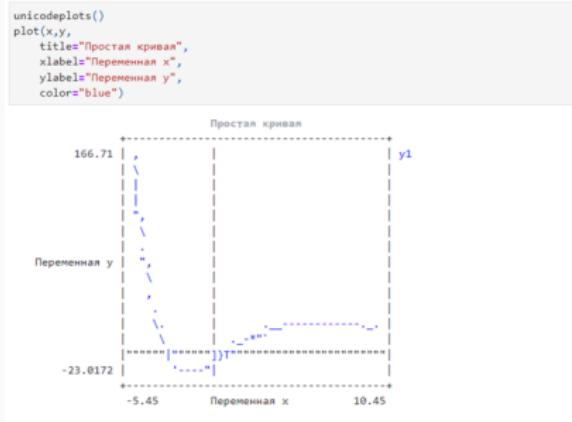


Рис. 5: Построение графика функции

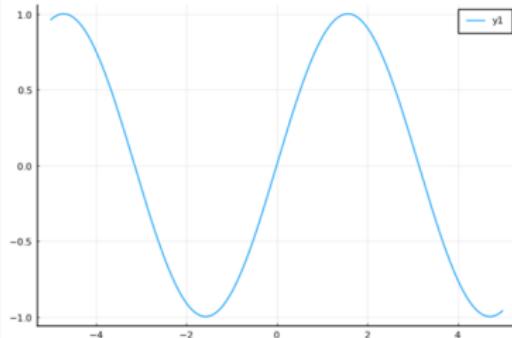
## Опции при построении графика

6. Повторим пример построения графика функции  $\sin(x)$  и графика разложения этой функции в ряд Тейлора. Рассмотрим дополнительные возможности пакетов для работы с графикой. Используем `pyplot()`, зададим функцию, построим график.

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \dots = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n+1}}{(2n+1)!}, \quad x \in \mathbb{C}$$

Рис. 6: Функция

```
# 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)
```



# Опции при построении графика

7. Повторим пример из пункта 6: задаём разложение исходной функции в ряд Тейлора, строим график разложения исходной функции в ряд Тейлора.

```
# задание функции разложения исходной функции в ряд Тейлора:  
sin_taylor(x) ≈ [(x-1)^i * x^(i+1) / factorial(2*i+1)] for i in 0:4] |> sum  
# построение графика функции sin_taylor(x):  
plot(sin_taylor)
```

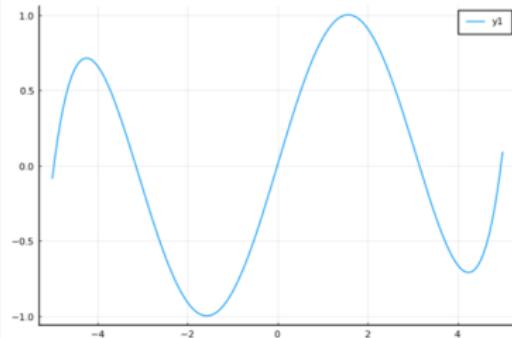


Рис. 8: Построение графика функции

8. Повторим примеры из пункта 6 и 7: построим две функции на одном графике.

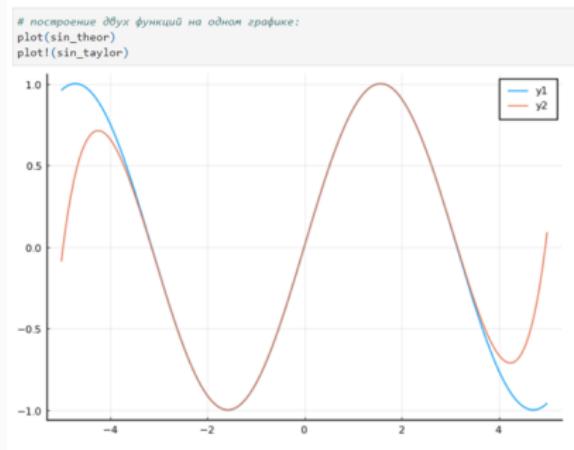


Рис. 9: Построение графиков функций

# Опции при построении графика

9. Повторим примеры с рядом Тейлора, только усовершенствуем его визуально: добавим различные опции для отображения на графике.

```
plot(
    # функция sin(x):
    sin_taylor,
    # подпись в легенде, цвет и тип линии:
    label = "sin(x), разложение в ряд Тейлора",
    line=:blue, 0.3, 6, :solid),
    # размер графика:
    size=(800, 500),
    # параметры отображения значений по осям
    xticks = (-5:0.5:5),
    yticks = (-10:1:10),
    xtickfont = font(12, "Times New Roman"),
    ytickfont = font(12, "Times New Roman"),
    # подписи по осям:
    ylabel = "y",
    xlabel = "x",
    # название графика:
    title = "Разложение в ряд Тейлора",
    # поворот значений, заданный по оси x:
    xrotation = rad2deg(pi/4),
    # заливка области графика цветом:
    fillrange = 0,
    fillalpha = 0.5,
    fillcolor = :lightgoldenrod,
    # задание цвета фона:
    background_color = :ivory
)
plot!(
    # функция sin_theor:
    sin_theor,
    # подпись в легенде, цвет и тип линии:
    label = "sin(x), теоретическое значение",
    line=:black, 1.0, 2, :dash)
```

Рис. 10: Листинг

# Опции при построении графика

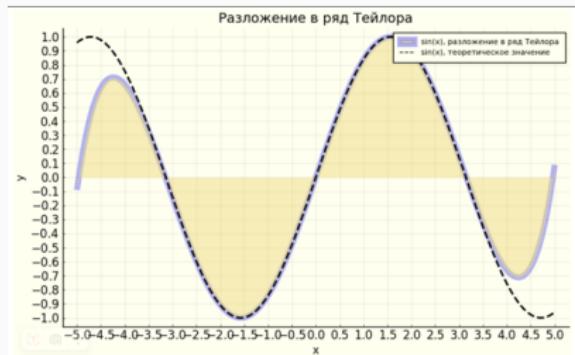


Рис. 11: Построение графика функции

10. Сохраним наш рисунок. Для сохранения построенного графика в определённом формате можно использовать функцию `savefig()`.

```
savefig("taylor.pdf")
savefig("taylor.png")  
"D:\\Education\\КомпПрактикумПоСтатМоделирование\\labs\\gitrepo\\lab5\\taylor.png"
```

Рис. 12: Сохранение

11. Графики в виде точек на плоскости или в пространстве часто используются в статистических исследованиях.

# Простой точечный график

12. Как и построении обычного графика для точечного графика необходимо задать массив значений  $x$ , посчитать или задать значения  $y$ , задать опции построения графика, добавляя надписи осей, легенды и названия.

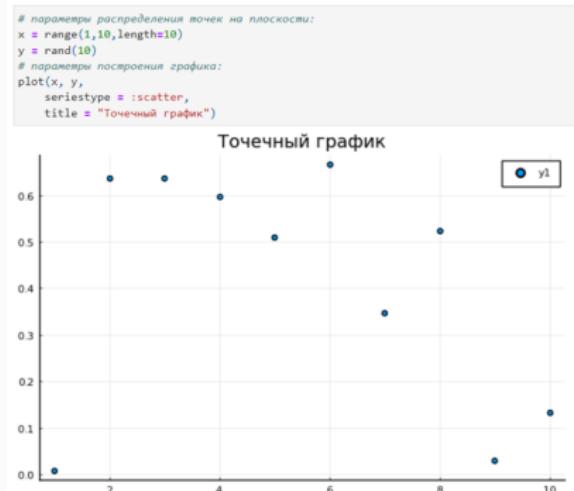


Рис. 13: Построение графика

## Точечный график с кодированием значения размером точки

13. Повторим примеры для точечного графика можно задать различные опции, например размер маркера, его тип, цвет и и т.п, добавляя надписи осей, легенды и названия.

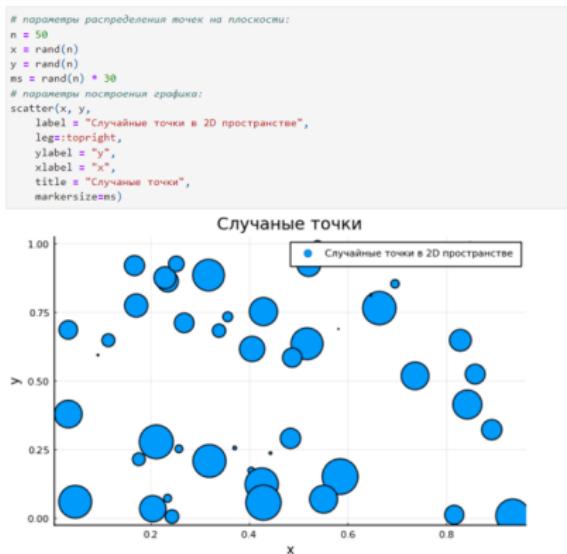


Рис. 14: Построение графика

14. Повторим примеры с построением 3-мерного точечного графика, добавляя надписи осей, легенды и названия.

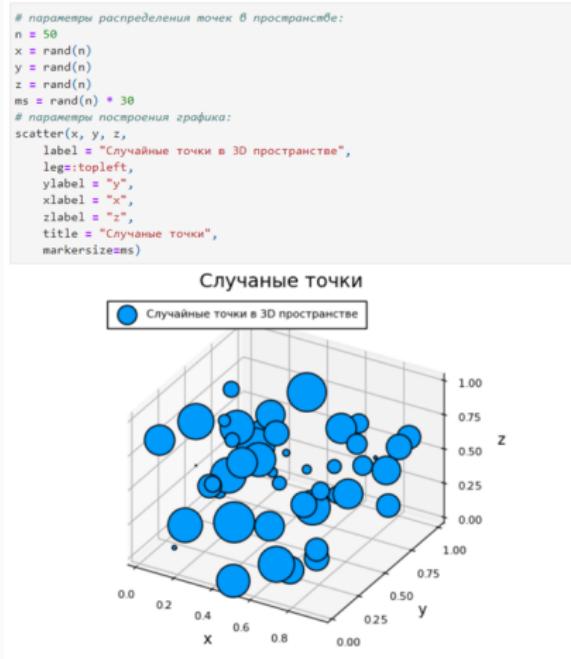


Рис. 15: Построение графика

## 15. Изучим информацию об аппроксимации данных.

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

Пусть на некотором отрезке в точках  $x_0, x_1, x_2, \dots, x_N$  известны значения некоторой функции  $f(x)$ , а именно  $y_0, y_1, y_2, \dots, y_N$ . Требуется определить параметры  $a_i$  многочлена вида  $F(x) = a_0 + a_1x + a_2x^2 + \dots + a_kx^k$ , где  $k < N$  такое, что сумма квадратов отклонений значений  $y$  от значений функции  $F(y)$  в заданных точках  $x$  минимальна, т.е.

$$S = \sum_0^N [y_i - F(x_i, a_0, a_1, \dots, a_k)]^2 \rightarrow \min$$

Геометрически это означает, что нужно найти кривую  $y = F(x)$ , полином которой проходит как можно ближе к каждой из заданных точек.

Такая задача может быть решена, если решить систему уравнений вида:  $A\vec{x} = \vec{y}$ , где  $A$  — матрица коэффициентов многочлена  $F(x)$ ,  $\vec{x}, \vec{y}$  — вектора соответствующих значений.

**Рис. 16:** Аппроксимация данных

16. Повторим пример: для демонстрации зададим искусственно некоторую функцию, в данном случае похожую по поведению на экспоненту.

```
: # 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)
```

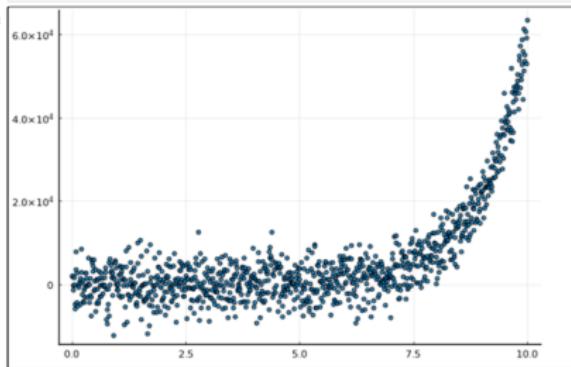


Рис. 17: Листинг и построение графика

## Аппроксимация данных

17. Повторим пример: для демонстрации зададим искусственно некоторую функцию, в данном случае похожую по поведению на экспоненту, аппроксимируем полученную функцию полиномом 5-й степени.

```
# определение масштаба 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,linewidth3, color:#red)
```

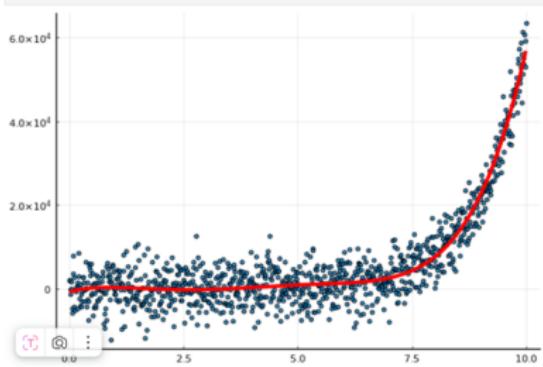


Рис. 18: Листинг и построение графика

18. Иногда требуется на один график вывести несколько траекторий с существенными отличиями в значениях по оси ординат. Повторим пример первой траектории.

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

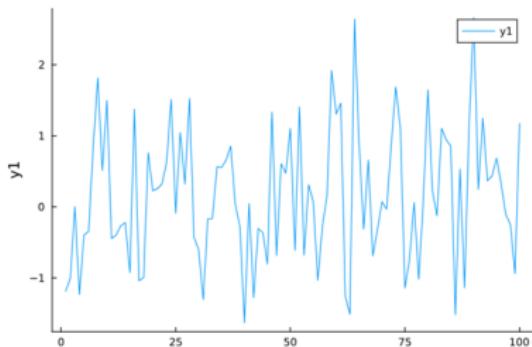


Рис. 19: Пример отдельно построенной траектории

## Две оси ординат

19. Далее на существующий график добавляется вторая траектория с дополнительными элементами для улучшения восприятия информации. Повторим пример двух траекторий на одном графике с двумя осями ординат.

```
# 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)
      )
```

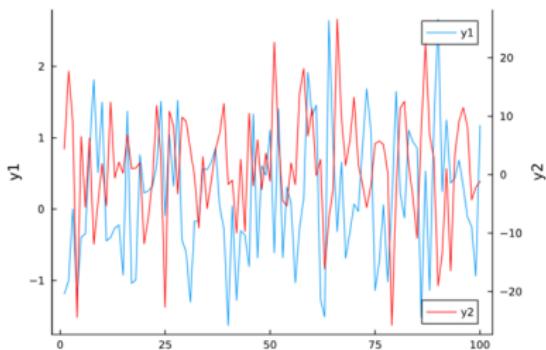


Рис. 20: Пример траекторий

# Полярные координаты

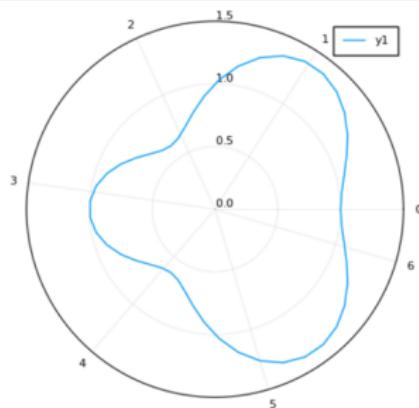
## 20. Выполним пример:

Приведём пример построения графика функции

$$r(\vartheta) = 1 + \cos \vartheta \sin^2 \vartheta$$

в полярных координатах.

```
# функция в полярных координатах:  
r(theta) = 1 + cos(theta) * sin(theta)^2  
# полярная система координат:  
theta = range(0, stop=2pi, length=50)  
# график функции, заданной в полярных координатах:  
plot(theta, r(theta),  
      proj=:polar,  
      lims=(0,1.5))
```



При параметрическом представлении графика некоторой функции координаты на графике задаются как функции от некоторого набора свободных параметров. В случае одного параметра получим параметрическое уравнение кривой. Выражая координаты точек поверхности через два свободных параметра, получим параметрическое задание поверхности.

# Параметрический график кривой на плоскости

21. Приведём пример построения графика параметрически заданной кривой на плоскости. Кривая задана выражениями  $x(t) = \sin(t)$ ,  $y(t) = \sin(2t)$ ,  $0 \leq t \leq 2\pi$ .

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

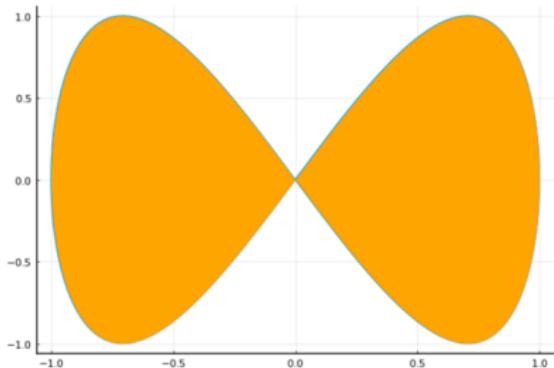


Рис. 22: Листинг и построение графика

## Параметрический график кривой в пространстве

22. Приведём пример построения графика параметрически заданной кривой в пространстве. Кривая задана выражениями  $x(t) = \cos(t)$ ,  $y(t) = \sin(t)$ ,  $z(t) = \sin(5t)$ :

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

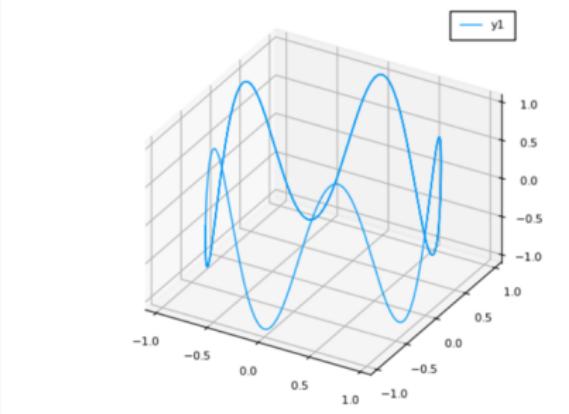


Рис. 23: Листинг и построение графика

# График поверхности

23. Повторим пример: для построения поверхности, заданной уравнением  $f(x, y) = x^2 + y^2$ , можно воспользоваться функцией `surface()`.

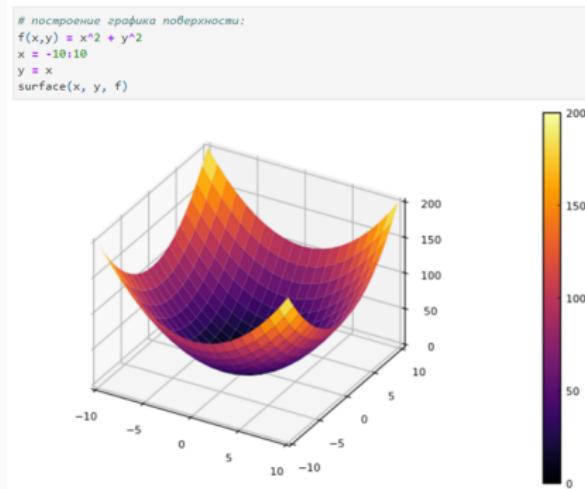


Рис. 24: Листинг и построение графика

# График поверхности

24. Выполним пример из пункта 23: также можно воспользоваться функцией `plot()` с заданными параметрами.

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

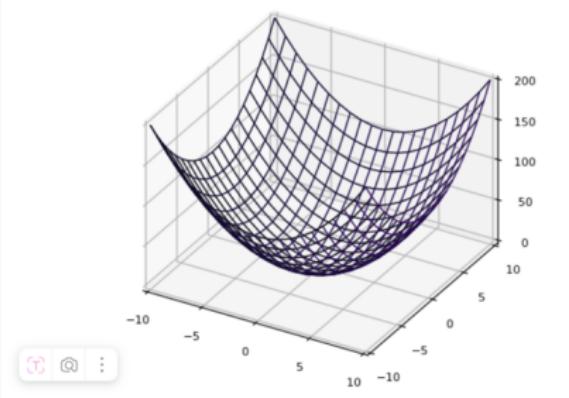


Рис. 25: Листинг и построение графика

# График поверхности

25. Выполним пример из пункта 23: можно задать параметры сглаживания.

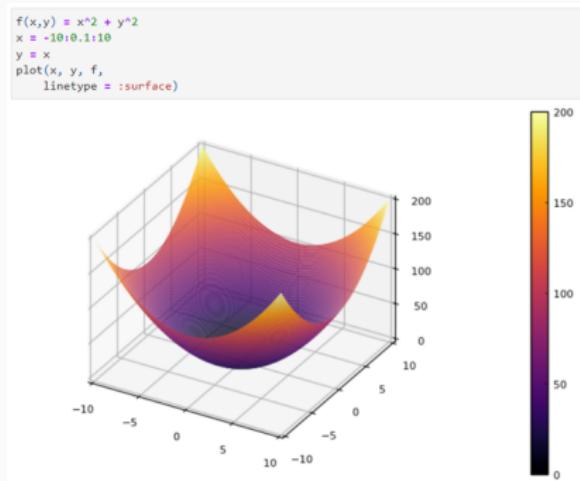


Рис. 26: Листинг и построение графика

# График поверхности

26. Выполним пример из пункта 23: можно задать определенный угол зрения.

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

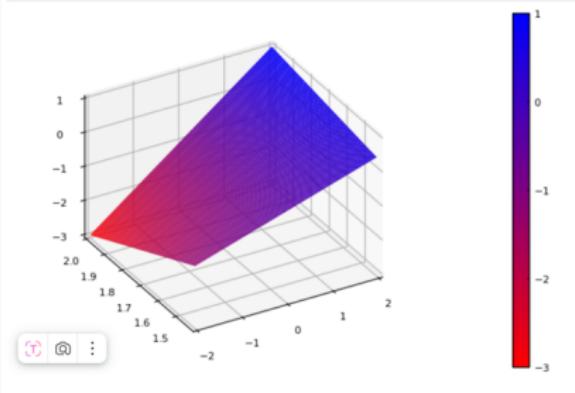
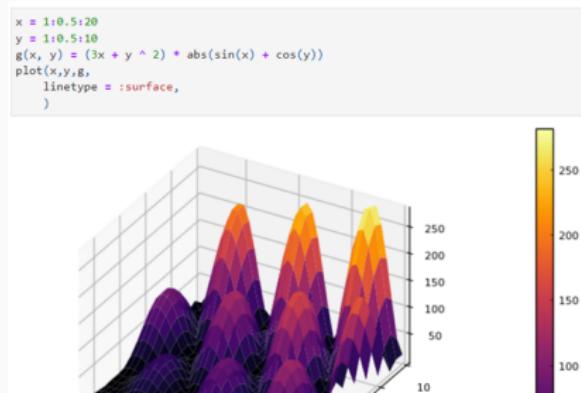


Рис. 27: Листинг и построение графика

## Линии уровня

Линией уровня некоторой функции от двух переменных называется множество точек на координатной плоскости, в которых функция принимает одинаковые значения. Линий уровня бесконечно много, и через каждую точку области определения можно провести линию уровня. С помощью линий уровня можно определить наибольшее и наименьшее значение исходной функции от двух переменных. Каждая из этих линий соответствует определённому значению высоты. Поверхности уровня представляют собой непересекающиеся пространственные поверхности.

27. Далее рассмотрим пример поверхности, заданную функцией  $\varphi(x, y) = (3x + y^2)|\sin(x) + \cos(y)|$ .



## Линии уровня

28. Выполним пример из пункта 27, учитывая, что линии уровня можно построить, используя проекцию значений исходной функции на плоскость.

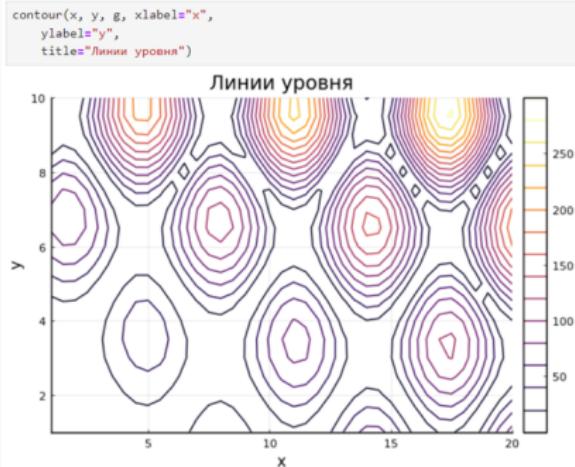


Рис. 29: Листинг и построение графика

## Линии уровня

29. Выполним пример из пункта 27, учитывая, что можно дополнительно добавить заливку цветом.

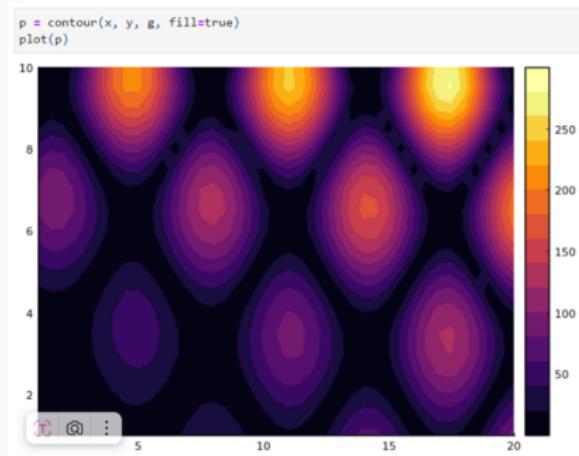


Рис. 30: Листинг и построение графика

## Векторные поля

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

30. Повторим пример: для функции  $\square(\square, \square) = \square 3 - 3\square + \square 2$  сначала построим её график и линии уровня.

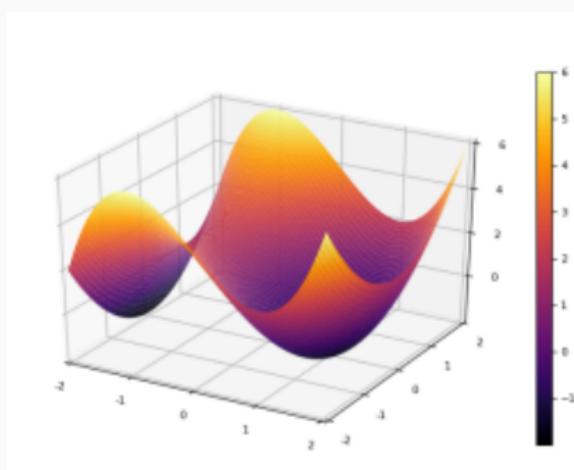


Рис. 31: Построение графика

# Векторные поля

```
# определение переменных:  
X = range(-2, stop=2, length=100)  
Y = range(-2, stop=2, length=100)  
# определение функции:  
h(x, y) = x^3 - 3x + y^2  
# построение поверхности:  
plot(X,Y,h,linetype = :surface)  
# построение линий уровня:  
contour(X, Y, h)
```

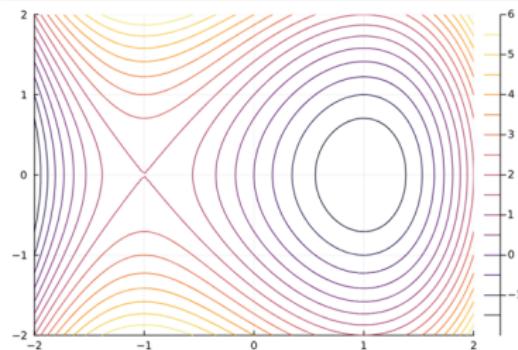
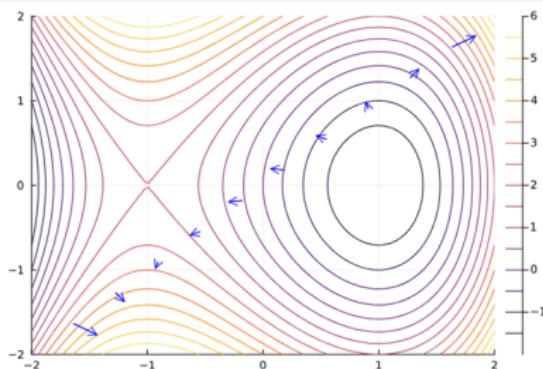


Рис. 32: Листинг и построение графика

31. Повторим пример: векторное поле можно охарактеризовать векторными линиями. Каждая точка векторной линии является началом вектора поля, который лежит на касательной в данной точке. Для нахождения векторной линии требуется решить дифференциальное уравнение. Для заданной функции построим векторное поле.

```
# градиент:
x = range(-2, stop=2, length=12)
y = range(-2, stop=2, length=12)
# производная от исходной функции:
dh(x, y) = [3x^2 - 3; 2y] / 25
# построение векторного поля:
quiver!(x, y, quiver=dh, cs:blue)
# коррекция области видимости графика:
xlim!(-2, 2)
ylim!(-2, 2)
```



# Векторные поля

```
# градиент:
pyplot()
xv = collect(range(-2, stop=2, length=12))
yv = collect(range(-2, stop=2, length=12))
# производная от исходной функции:
dh(x,y) = [3*x^2 - 3; 2y] / 25
x = vcat([xv for i in 1:12]...)
y = reshape(hcat([yv for i in 1:12]...), 1, 1)
# построение векторного поля:
quiver!(x, y, quiver=dh, c=:blue)
# коррекция области видимости графика:
xlims!(-2, 2)
ylims!(-2, 2)
```

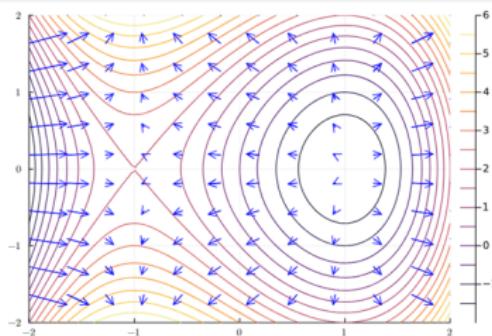


Рис. 34: Построение графика

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

## Gif-анимация

32. Выполним пример, в Julia рекомендуется использовать gif-анимацию в `pyplot()`, зададим поверхность, добавим анимацию.

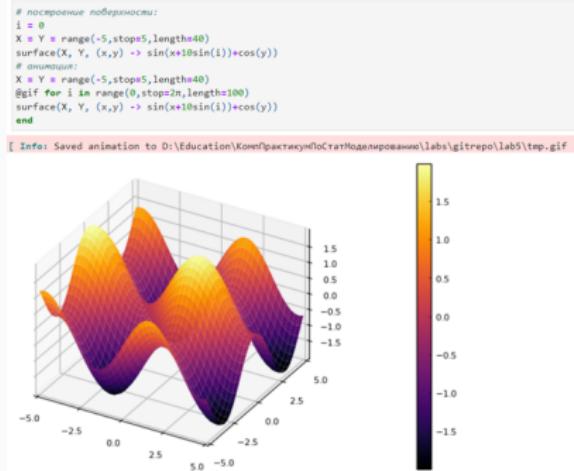


Рис. 35: Листинг и построение графика

# Гипоциклоида

## 33. Выполним пример: построим гипоциклоиду (зададим параметры, массивы, построим оси).

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

$$\begin{cases} x = r(k - 1) \left( \cos t + \frac{\cos((k - 1)t)}{k - 1} \right), \\ y = r(k - 1) \left( \sin t - \frac{\sin((k - 1)t)}{k - 1} \right), \end{cases}$$

где  $k = \frac{R}{r}$ ,  $R$  — радиус неподвижной окружности,  $r$  — радиус катящейся окружности.

Модуль величины  $k$  определяет форму гипоциклоиды.

```
# 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(0:2π/100:2π+2π/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)
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)
# 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)
```

Рис. 36: Листинг

# Гипоциклоида

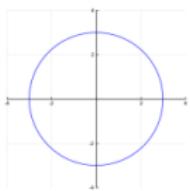


Рис. 5.30. Большая окружность гипоциклоиды

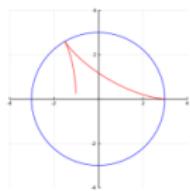


Рис. 5.31. Половина пути гипоциклоиды

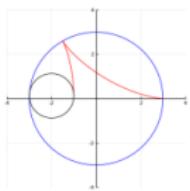


Рис. 5.32. Малая окружность гипоциклоиды  
Рис. 5.33. Малая окружность гипоциклоиды  
с добавлением радиуса

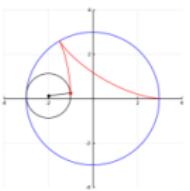


Рис. 37: Построение графика

# Гипоциклоида

34. Выполним анимацию пункта 33, анимация получившегося изображения и сохраним его.

```
anim = @animate for i in 1:n
    # зададим оси координат:
    plt=plt(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 = 0:1: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[iend]) .+ r1*cos.(θ)
    yc = r1*(k-1)*sin(t[iend]) .+ r1*sin.(θ)
    plot!(xc,yc,c=:black)
    # радиус малой окружности:
    xl = transpose((r1*(k-1)*cos(t[iend]):x[iend]))
    yl = transpose((r1*(k-1)*sin(t[iend]):y[iend]))
    plot!(xl,yl,markershape=:circle,markersize=4,c=:black)
    scatter!([(x[iend],y[iend]),c=:red, markerstrokecolor=:red])
end
gif(anim,"hypocycloid.gif")
```

[ Info: Saved animation to D:\Education\КомпПрактикумПоСтатМоделирование\labs\gitrepo\lab5\hypocycloid.gif ]

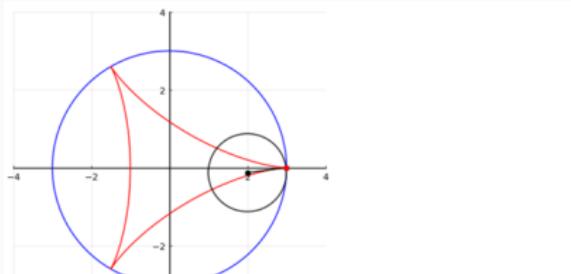


Рис. 38: Листинг и анимация

## Errorbars

35. В исследованиях часто требуется изобразить графики погрешностей измерения. Подключим пакет Statistics. Подключение пакета Statistics:  
import Pkg  
Pkg.add("Statistics") using Statistics

```
# подключение пакета Statistics:  
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)  
  
6-element Vector{Float64}:  
0.6198064213930023  
0.3099032106965012  
0.1549516053482506  
0.0774758026741253  
0.03873790133706265  
0.019368950668531323
```

Рис. 39: Подключение пакета

## Errorbars

36. Выполним пример: зададим массив значений, затем сгенерируем массив ошибок (отклонений от исходных значений), построим график.

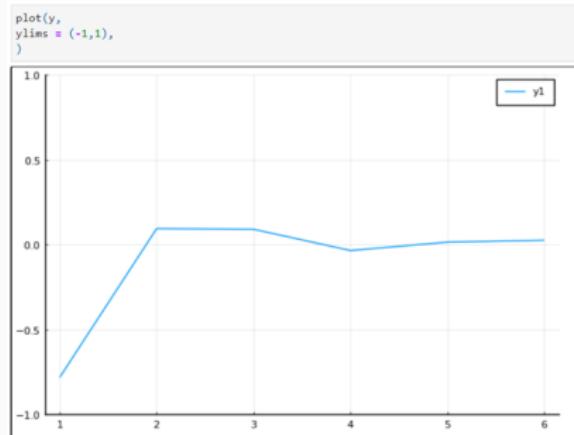
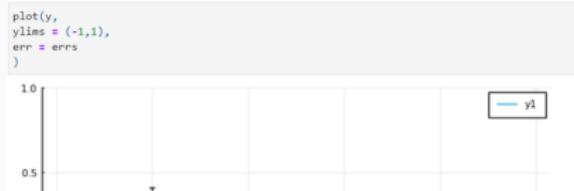


Рис. 40: График исходных значений



# Errorbars

37. Выполним пример: повернем наш график из пункта 36.

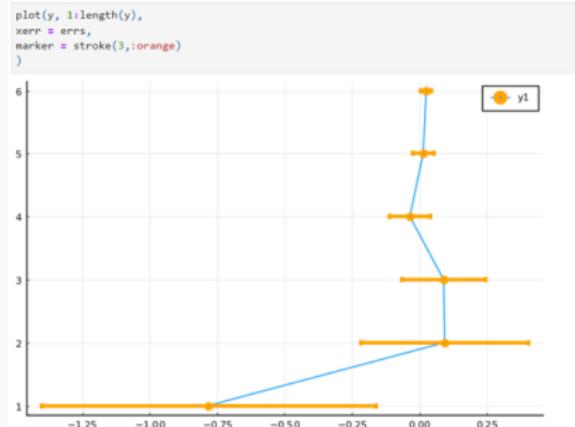
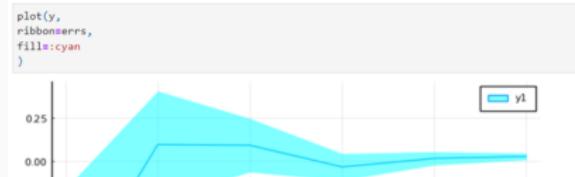


Рис. 42: Листинг и построение графика

38. Выполним пример: заполним область цветом на графике из пункта 36.



## Errorbars

39. Выполним пример: о построить график ошибок по двум осям из пункта 36.



Рис. 44: Листинг и построение графика

40. Выполним пример: построить график асимметричных ошибок по двум осям из пункта 36.

# Использование пакета Distributions

41. Подключим пакет распределений и выполним пример: зададим массив случайных чисел, построим гистограмму, также зададим нормальное распределение.

```
Подгружаем pyplot():
pyplot()
Задаём массив случайных чисел:
ages = rand(15:55,1000)
Строим гистограмму (рис. 5.41):
histogram(ages)

Задаём нормальное распределение и строим гистограмму (рис. 5.42):
d=Normal(35.0,10.0)
ages = rand(d,1000)
histogram(
    ages,
    label="Распределение по возрастам (года)",
    xlabel = "Возраст (лет)",
    ylabel= "Количество"
)
```

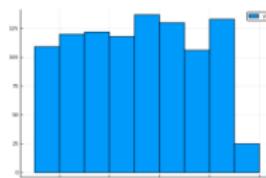


Рис. 5.41. Гистограмма, построенная по массиву случайных чисел

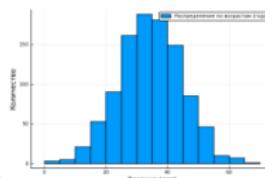


Рис. 5.42. Гистограмма нормального распределения

**Рис. 46:** Гистограмма

# Использование пакета Distributions

42. Далее применим для построения нескольких гистограмм распределения людей по возрастам на одном графике `plotly()`.

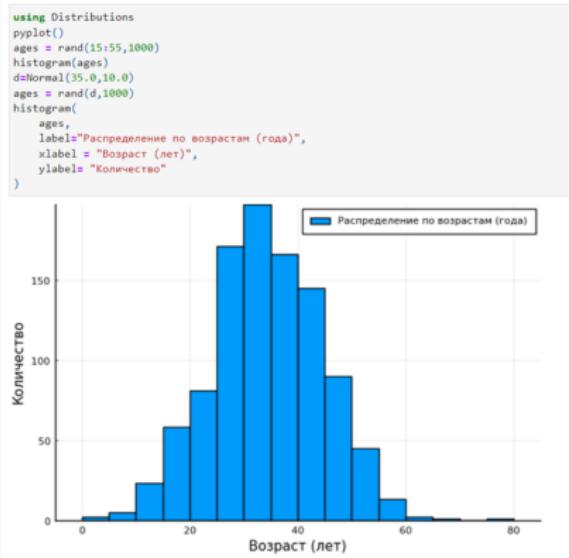


Рис. 47: Гистограмма распределения людей по возрастам

# Использование пакета Distributions

```
pyplot()
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(floor(N/2))));
ages = append(ages,rand(d2,N));
ages = append(ages,rand(d3,Int64(floor(N/3))));
histogram(
    ages,
    bins=50,
    label="Распределение по возрастам (года)",
    xlabel = "Возраст (лет)",
    ylabel = "Количество",
    title = "Распределение по возрастам (года"
)
```

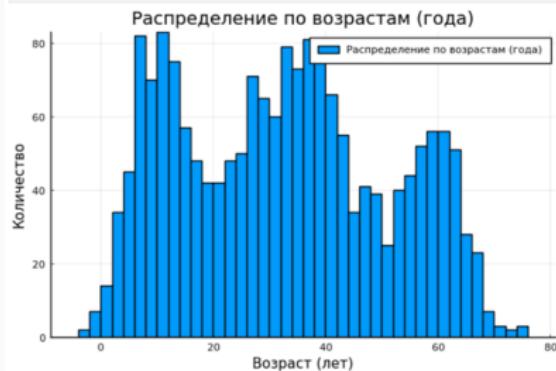


Рис. 48: Гистограмма распределения людей по возрастам

## Подграфики

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

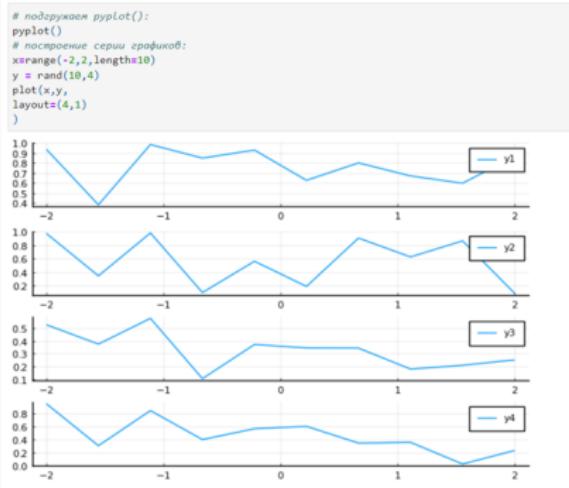


Рис. 49: Серия из 4-х графиков в ряд

## Подграфики

44. Определим макет расположения графиков. Для автоматического вычисления сетки необходимо передать layout целое число.

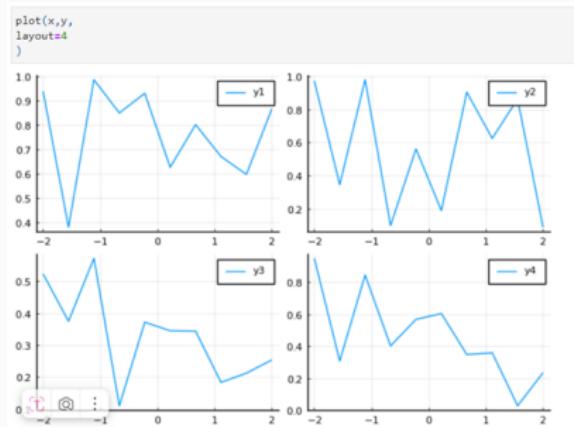


Рис. 50: Серия из 4-х графиков в сетке

45. Определим макет расположения графиков. Чуть более сложные макеты сетки могут быть созданы с помощью конструктора `grid (...)`. Например, в следующем макете создаются участки разной высоты.

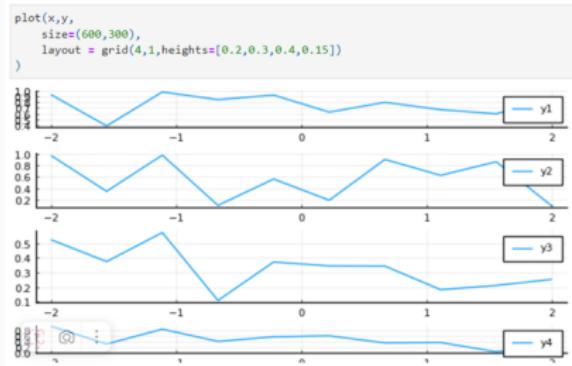


Рис. 51: Серия из 4-х графиков разной высоты в ряд

46. Аргумент `heights` принимает в качестве входных данных массив с долями желаемых высот. Если в сумме дроби не составляют 1,0, то некоторые подзаголовки могут отображаться неправильно: можно сгенерировать отдельные графики и объединить их в один, например, в сетке  $2 \times 2$ .

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

Рис. 52: Листинг

# Подграфики

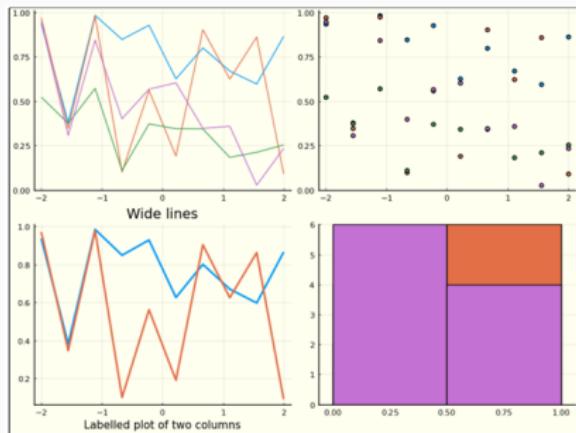


Рис. 53: Объединение нескольких графиков в одной сетке

## Подграфики

47. Обратите внимание, что атрибуты на отдельных графиках применяются к отдельным графикам, в то время как атрибуты в последнем вызове plot применяются ко всем графикам.
- Разнообразные варианты представления данных.

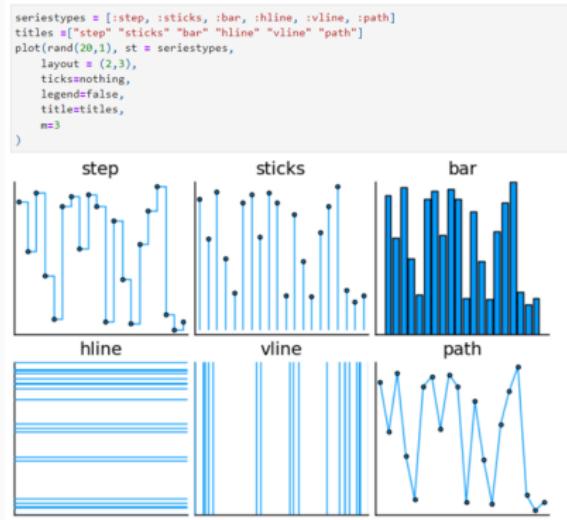


Рис. 54: Листинг и разнообразные варианты представления данных

# Подграфики

48. Повторим пример: применение макроса `@layout` наиболее простой способ определения сложных макетов. Точные размеры могут быть заданы с помощью фигурных скобок, в противном случае пространство будет поровну разделено между графиками.

```
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 = 1:11, i = 1:11], titleloc = :right, titlefont = font(8)
)
```

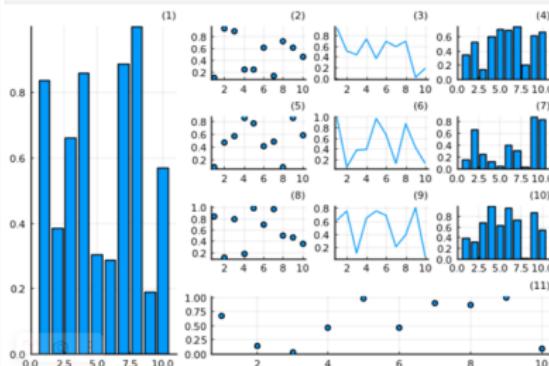


Рис. 55: Листинг и сложный макет

# Задания для самостоятельного выполнения

- 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.

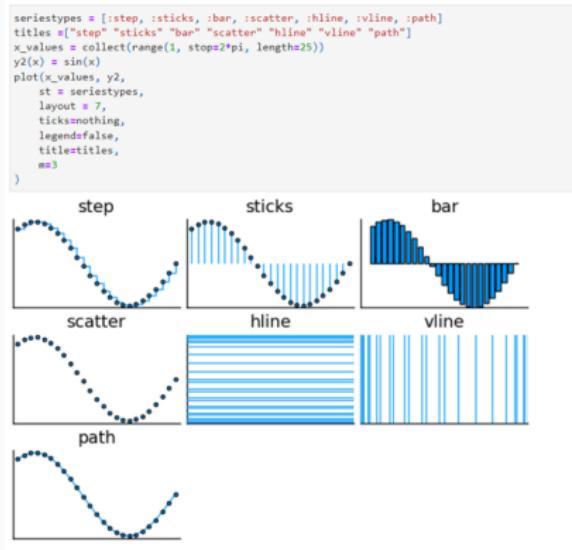


Рис. 56: Задание 1

## Задания для самостоятельного выполнения

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

```
lss = [:solid, :dash, :dot, :dashdot, :dashdotdot]
titles = ["solid" "dash" "dot" "dashdot" "dashdotdot"]
gr()
p1 = plot(x_values, y2,
           ls = lss[1],
           title = titles[1],)
p2 = plot(x_values, y2,
           ls = lss[2],
           title = titles[2],)
p3 = plot(x_values, y2,
           ls = lss[3],
           title = titles[3],)
p4 = plot(x_values, y2,
           ls = lss[4],
           title = titles[4],)
p5 = plot(x_values, y2,
           ls = lss[5],
           title = titles[5],)
plot(p1,p2,p3,p4,p5,
      layout=5,
      legend=false,
      size=(800,600),
      background_color = :ivory
)
```

Рис. 57: Задание 2 листинг

# Задания для самостоятельного выполнения

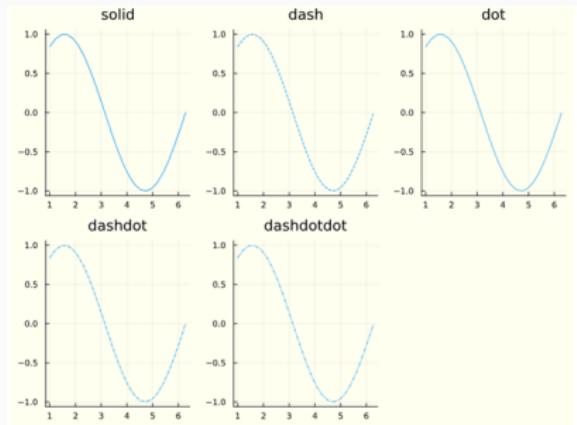
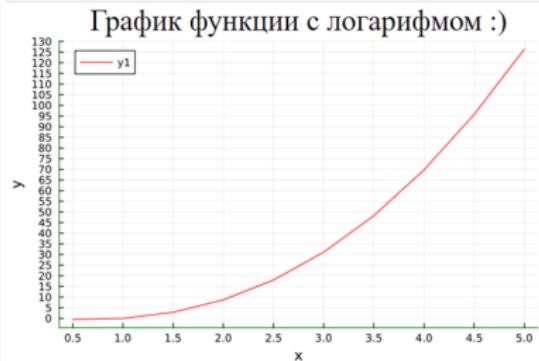


Рис. 58: Задание 2 график

## Задания для самостоятельного выполнения

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 расстояния между надписями и осями так, чтобы надписи полностью умещались в графическом окне. Задайте шрифт надписей. Задайте частоту отметок на осях координат.

```
y3(x) = pi*x^2*log(x)
x_values = collect(0.5:0.5:5)
plot(x_values, y3,
    foreground_color_border = :green,
    ylabel = "y",
    xlabel = "x",
    titlefont = (20, "times"),
    title = "График функции с логарифмом :)",
    color=:red,
    xticks = (0:0.5:5),
    yticks = (0:5:130),
```



# Задания для самостоятельного выполнения

```
x_values = [-2, -1, 0, 1, 2] # x_values = collect(-2:2)
y4(x) = x^3 + 3*x
titles = ["curved" "dash" "dot" "dashdot"]
p1 = curves(x_values, y4,
             title = titles[1],
             color = :black)
p2 = plot(x_values, y4,
           ls = :dash,
           title = titles[2],
           color = :blue)
p3 = plot(x_values, y4,
           ls = :dot,
           title = titles[3],
           color = :red)
p4 = plot(x_values, y4,
           ls = :dashdot,
           title = titles[4],
           color = :darkgreen)
plot(p3,p2,p4,p1,
      layout=4,
      legends=false,
      )
```

Рис. 60: Задание 3 листинг

# Задания для самостоятельного выполнения

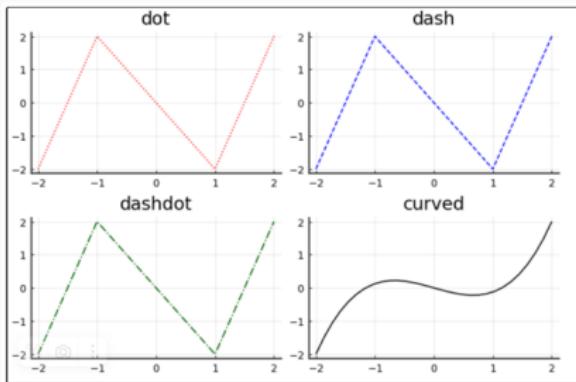


Рис. 61: Задание 3 график

## Задания для самостоятельного выполнения

4. Задайте вектор  $x = (-2, -1, 0, 1, 2)$ . В одном графическом окне (в 4-х подокнах) изобразите графически по точкам  $x$  значения функции  $y(x) = x^3 - 3x$  в виде:

- точек,
- линий,
- линий и точек,
- кривой.

Сохраните полученные изображения в файле  
figure\_familiya.png, где вместо familiya укажите вашу  
фамилию.

```
x_values = [-2, -1, 0, 1, 2] # x_values = collect(-2:2)
y4(x) = x^3 - 3*x
titles = ["curved" "dash" "dot" "dashdot"]
p1 = curves(x_values, y4,
             title = titles[1],
             color = :black)
p2 = plot(x_values, y4,
           ls = :dash,
           title = titles[2],
           color = :blue)
p3 = plot(x_values, y4,
           ls = :dot,
           title = titles[3],
           color = :red)
p4 = plot(x_values, y4,
           ls = :dashdot,
           title = titles[4],
           color = :darkgreen)
```

# Задания для самостоятельного выполнения

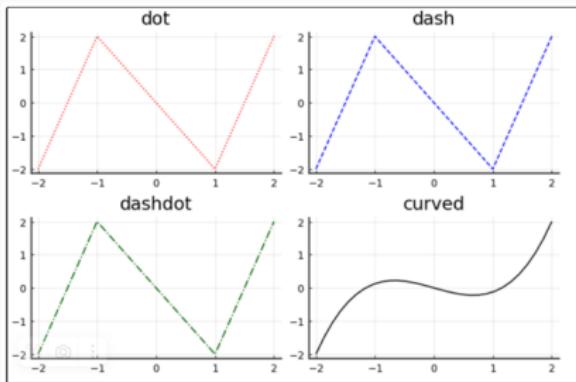


Рис. 63: Задание 4 график

## Задания для самостоятельного выполнения

5. Задайте вектор  $x = (3, 3.1, 3.2, \dots, 6)$ . Постройте графики функций  $y_1(x) = \pi x$  и  $y_2(x) = e^x \cos(x)$  в указанном диапазоне значений аргумента  $x$  двумя разными способами

```
x,y = collect(3:0.1:6)
y_1(x) = pi*x
y_2(x) = exp(x)*cos(x)
p1 = plot(x,y, y_1,
           label = "Прямая",
           color = :black)
p2 = plot(x,y, y_2,
           label = "Экспонента",
           color = :blue)
```

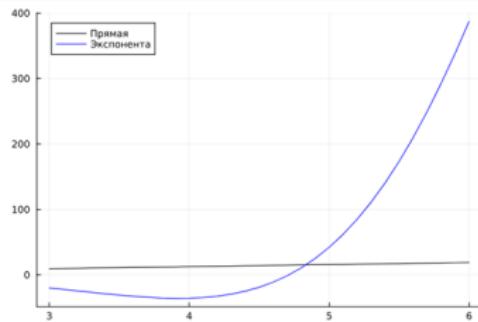


Рис. 64: Задание 5 листинг и график

# Задания для самостоятельного выполнения

```
plot(x_v, y_1,
      ylabel="\\$y_1\\$",
      legend="topleft",
      grid = :off,
      )
plot!(twinx(), x_v, y_2,
      c=:red,
      ylabel="\\$y_2\\$",
      legend="right",
      grid = :off,
      box = :on,
      )
```

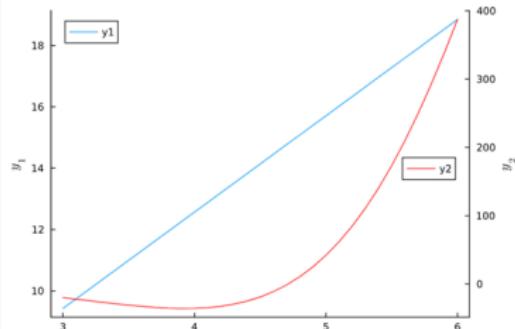


Рис. 65: Задание 5 листинг и график

# Задания для самостоятельного выполнения

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

```
sds = collect(rand(1:500, 20))
n = 10
y = [mean(sd*randn(n)) for sd in sds]
errs = 0.36 * sds / sqrt(n)
plot(y,
err = errs
)
```

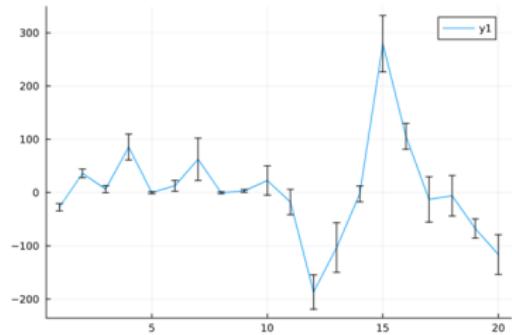


Рис. 66: Задание 6 листинг и график

# Задания для самостоятельного выполнения

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

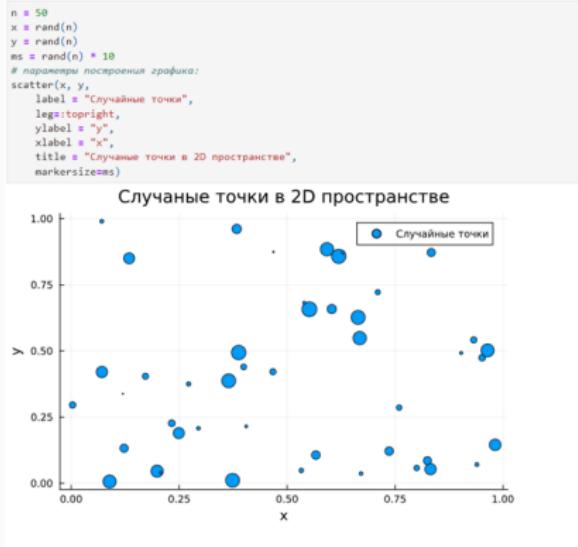


Рис. 67: Задание 7 листинг и график

# Задания для самостоятельного выполнения

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

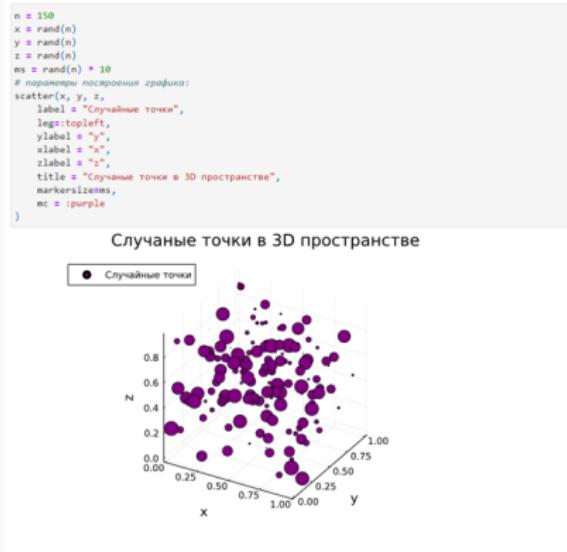


Рис. 68: Задание 8 листинг и график

## Задания для самостоятельного выполнения

9. Создайте анимацию с построением синусоиды. То есть, постройте последовательность графиков синусоиды, постепенно увеличивая значение аргумента, после чего соединить их в анимацию.

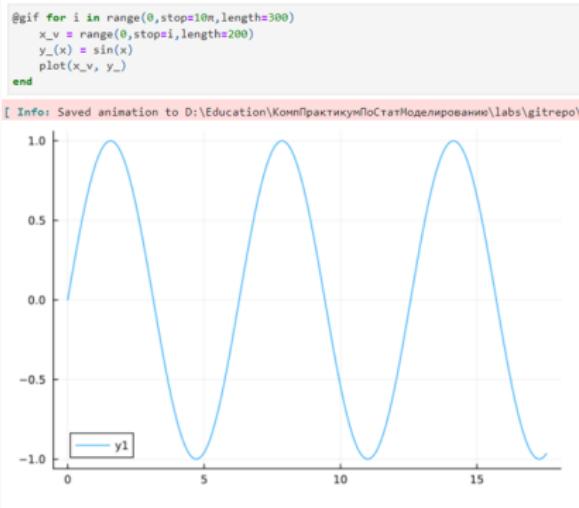


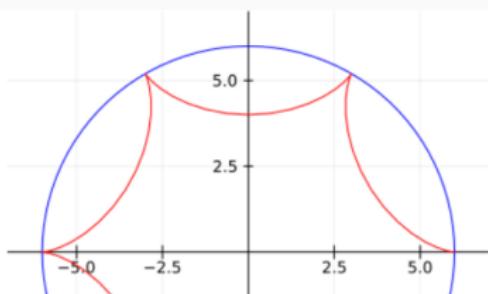
Рис. 69: Задание 9 листинг и анимация

# Задания для самостоятельного выполнения

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

```
r1 = 1
k = 6
n = 100
θ = collect(0.2*n/100+2*pi*k/100)
X = r1*k*cos.(θ)
Y = r1*k*sin.(θ)
anim = @animate for i in lin
    # зондирует оси координат:
    plt=plot(5,xlim=(-k+1,k+1),ylim=(-k+1,k+1), c:red, aspect_ratio1,legends=false, framestyle:origin)
    # далее отрисовка:
    plot!(plt, X,Y, c:blue, legend=false)
    t = 0[i]:1
    # зондирование:
    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)
    # малое окружение:
    xc = r1*(k-1)*cos(t|end)) + r1*cos.(θ)
    yc = r1*(k-1)*sin(t|end)) + r1*sin.(θ)
    plot!(xc,yc, c:blue)
    # далее отрисовка:
    xl = transpose([r1*(k-1)*cos(t|end)) + end])
    yl = transpose([r1*(k-1)*sin(t|end)) + end])
    plot!(xl,yl,markershape:circle,markerized:c:black)
    scatter!([x|end]], [y|end]], c:red, markerstrokecolors:red)
end
gif(anim,"hypocycloid_1.gif")
```

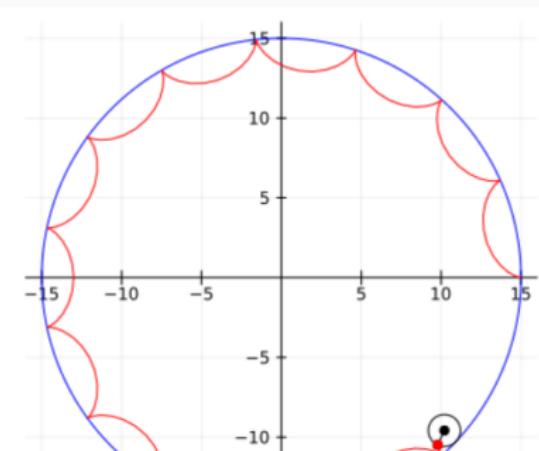
Рис. 70: Задание 10 листинг (целые)



# Задания для самостоятельного выполнения

```
r1 = 1
k = 15
n = 200
θ = collect(θ/2*π/n/2*n+2*π/n)
X = r1*k*cos.(θ)
Y = r1*k*sin.(θ)
anim = @animate for i in 1:n
    # создание окна
    plt = plt(xlim=(-k-1,k+1), ylim=(-k-1,k+1), cn:red, aspect_ratio=1, legendstyle=:origin)
    # добавление окружности
    plot!(plt, X, Y, c:black, legend=false)
    t = θ[i]
    # суммируем радиусы
    x = r1*(k-1)*cos.(t) + r1*k*cos.((k-1)*t)
    y = r1*(k-1)*sin.(t) + r1*k*sin.((k-1)*t)
    plot!(x, y, c:red)
    # наращиваем радиусы
    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)
    # подготавливаем данные
    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, markersizes=4, cn:black)
    scatter!([(x[end]), (y[end])], cx:red, markerstrokecolor:red)
end
gif(anim, "hypocycloid_2.gif")
```

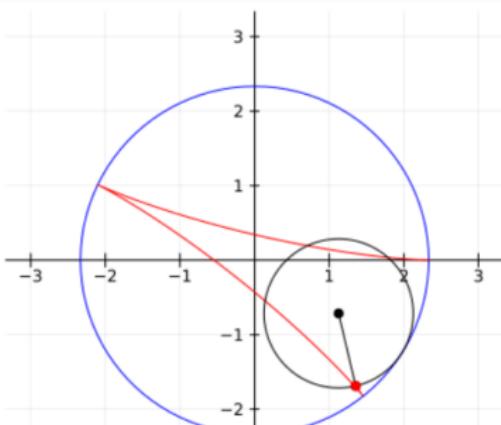
Рис. 72: Задание 10 листинг (целые)



# Задания для самостоятельного выполнения

```
r1 = 1
k = 7/3
n = 100
θ = collect(θ+2πn/n2θ+n+2θ)n/n)
X = r1*k*cos.(θ)
Y = r1*k*sin.(θ)
anim = Genimate for i in 1:n
    # another osc coordinate:
    pltplot(S,xlim=(-k-1,k+1),ylim=(-k-1,k+1), c:red, aspect_ratio=1,framestyle:origin)
    # большая окружность:
    plot(p1t,X,Y, c:blue, legend=false)
    t = θ[1:i]
    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)
    # подпись 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[and]],y[and]),c:red, markerstrokecolor:red)
end
gif(anim,"hypocycloid_3.gif")
```

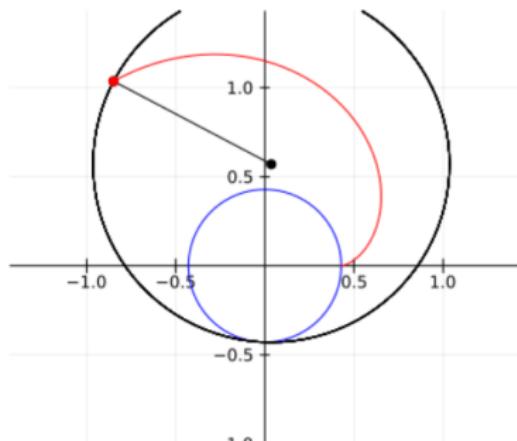
Рис. 74: Задание 10 листинг (рацио)



# Задания для самостоятельного выполнения

```
r1 := 1
k := 3/7
n := 100
θ := collect(θ+2*π/n;20*n+20*π/n)
X := r1*k*cos.(θ)
Y := r1*k*sin.(θ)
anim :=animate for i in 1..n
    do
        # синусоиды
        plot1:=plot(xl,yl,clim(-k-1,k+1),ylim(-k-1,k+1), c:red, aspect_ratio=false, framestyle:origin)
        # оканчивающиеся кривые
        plot2:=plot(X,Y, c:blue, legend=false)
        t := θ[i]:1
        # синусоиды:
        x := r1*(k-1)*cos.(t) + r1*cos.((k-1)*t)
        y := r1*(k-1)*sin.(t) + r1*sin.((k-1)*t)
        plot3:=plot(x,y, c:red)
        # оканчивающиеся кривые
        xc := r1*(k-1)*cos(t[and]) + r1*cos.(θ)
        yc := r1*(k-1)*sin(t[and]) + r1*sin.(θ)
        plot4:=plot(xc,yc, c:black)
        # радиус малой окружности:
        xl := transpose([r1*(k-1)*cos(t[and]):x[and]])
        yl := transpose([r1*(k-1)*sin(t[and]):y[and]])
        plot5:=plot(xl,yl,markershape=circle,markersize=8,c:black)
        scatter([x[and],y[and]],c:red, markerstrokecolor:red)
        and
    gif(anim,"hypocycloid_4.gif")
```

Рис. 76: Задание 10 листинг (рацио)

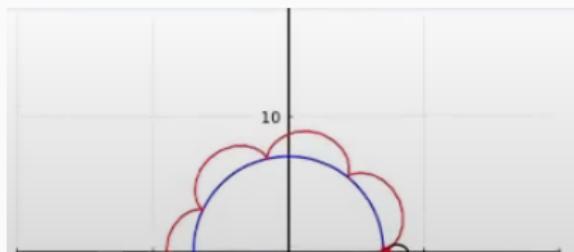


# Задания для самостоятельного выполнения

11. Постройте анимированную эпициклоиду для 2 целых значений модуля  $\square$  и 2 рациональных значений модуля  $\square$ .

```
rr = 1
# коэффициент для построения большой окружности
k = 7
# число отсчётов:
n = 100
# массив значений угла θ:
# theta from 0 to 2pi (+ a little extra)
θ = collect(0:2π/n:100+2π/n/100)
# массив значений координат:
X = rr*k*cos.(θ)
Y = rr*k*sin.(θ)
l = 50
t = 0:l:l
# @onstate for i in l:m
# зададим оси координат:
plt=plot(lf, xlim=(-20,20), ylim=(-20,20), c=:red, aspect_ratio=1, legend=false, framestyle=:origin)
# большая окружность:
plot!(plt, X,Y, c=:blue, legend=false)
t = 0:l:l
# эпиклиоды:
x = rr*(k+l)*cos.(t) - rr*cos.((k+l)*t)
y = rr*(k+l)*sin.(t) - rr*sin.((k+l)*t)
plot!(x,y, c=:red)
# наше изображение:
xc = rr*(k+l)*cos(t[end]) - rr*cos.(θ)
yc = rr*(k+l)*sin(t[end]) - rr*sin.(θ)
plot!(xc,yc,c=:black)
# радиус малой окружности:
x1 = transpose([rr*(k+l)*cos(t[end]) x1[end]])
y1 = transpose([rr*(k+l)*sin(t[end]) y1[end]])
plot!(x1,y1,markershape=:circle,markerstroke=:black)
scatter!(x1,y1, c=:red, markerstrokecolor=:red)
end
```

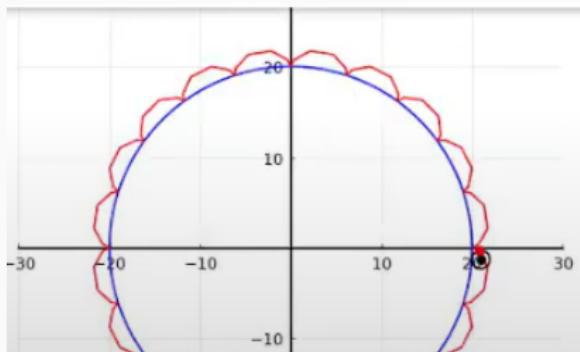
Рис. 78: Задание 11 листинг (целые)



# Задания для самостоятельного выполнения

```
r=1  
# коэффициент для построения большой окружности:  
k = 20  
# число отсчетов:  
n = 100  
# массив значений угла theta:  
# theta from theta to 2pi (+ a little extra)  
theta = collect(0:2*pi/100:2*pi+2*pi/100)  
# массивы значений координат:  
X = r*cos(theta)  
Y = r*sin(theta)  
t = 50  
t = 0[1:t]  
anim = generate for i in t  
    # зададим оси координат:  
    plot!(5,xlim=(-30,30), ylim=(-30,30), c=:red, aspect_ratio=1, legend=false, framestyle=:origin)  
    # больших окружности:  
    plot!(X,Y, c=:blue, legend=false)  
    t = t+1  
    # эллипса:  
    x = r*(k+1)*cos.(t) .- r*cos.((k+1)*t)  
    y = r*(k+1)*sin.(t) .- r*sin.((k+1)*t)  
    plot!(x,y, c=:red)  
    # малых окружности:  
    xc = r*(k+1)*cos(t[1:end]) .- r*cos.(theta)  
    yc = r*(k+1)*sin(t[1:end]) .- r*sin.(theta)  
    plot!(xc,yc, c=:black)  
    # радиус малой окружности:  
    xl = transpose([r*(k+1)*cos(t[1:end]) xiend])  
    yl = transpose([r*(k+1)*sin(t[1:end]) yiend])  
    plot!(xl,yl, markershape=:circle, markersize=4, c=:black)  
    scatter!([x[1],y[1]],c=:red, markerstrokecolor=:red)  
end
```

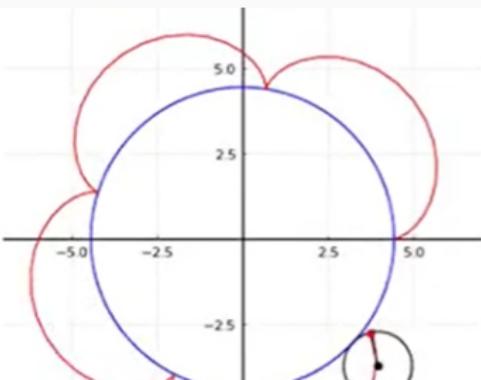
Рис. 80: Задание 11 листинг (целые)



# Задания для самостоятельного выполнения

```
rr = 1
# коэффициент для построения большой окружности:
k = 4.44
# число отсчётов:
n = 50
# массив значений угла theta:
theta from 0 to 2pi (+ a little extra)
theta = collect(theta/2pi/100+2pi*2*pi/100)
# массивы значений координат:
X = rr*cos(theta)
Y = rr*sin(theta)
l = 50
t = theta[1:l]
anim = animate for i in l:n
    # задать оси координат:
    plt=plot(X,Y,xlim=(-7,7),ylim=(-7,7), c=:red, aspect_ratio=1, legend=false, framestyle=:origin)
    # большая окружность:
    plot!(plt, X,Y, c=:blue, legend=false)
    t = theta[1:i]
    # эпиклипода:
    x = rr*(k+1)*cos(t) - rr*cos((k+1)*t)
    y = rr*(k+1)*sin(t) - rr*sin((k+1)*t)
    plot!(plt, x,y, c=:red)
    # малая окружность:
    xc = rr*(k+1)*cos(t[lend]) - rr*cos(theta[lend])
    yc = rr*(k+1)*sin(t[lend]) - rr*sin(theta[lend])
    plot!(plt, xc,yc, c=:black)
    # радиус малой окружности:
    xl = transpose([rr*(k+1)*cos(theta[lend]), x[lend]])
    yl = transpose([rr*(k+1)*sin(theta[lend]), y[lend]])
    plot!(xl,yl, markershape=:circle, markersize=4, c=:black)
    scatter!([x[lend]], [y[lend]], c=:red, markerstrokecolor=:red)
end
```

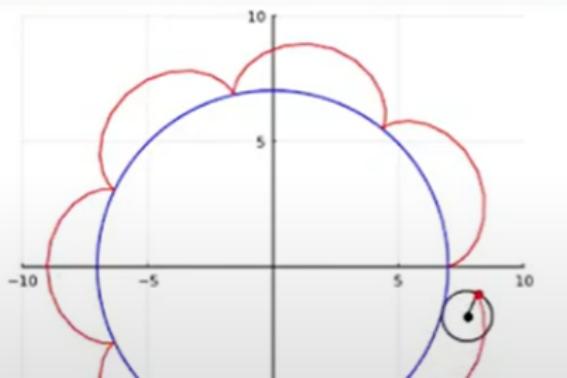
Рис. 82: Задание 11 листинг (рацио)



# Задания для самостоятельного выполнения

```
rr = 1
# коэффициент для построения большой окружности:
k = 1
# число отсчётов:
n = 100
# массив значений угла θ:
theta = [theta from 0 to 2pi {+ a little extra}]
θ = collect(θ[2π/n:100:2π+2π/n*100])
# массивы значений координат:
X = rr*k*cos(θ)
Y = rr*k*sin(θ)
t = 50
t = 1:t:n
while t < n+1:
    # задаём оси координат:
    plt=plt()
    plt.plot(t,xlim=(-10,10),ylim=(-10,10),c:red,aspect_ratio=1,legend=False, framestyle:origin)
    # большая окружность:
    plot(t,pt,X,Y, c:blue, legend=False)
    t = t[1:t]
    # эллипсы:
    x = rr*(k-1)*cos(t) - rr*cos((k-1)*t)
    y = rr*(k-1)*sin(t) - rr*sin((k-1)*t)
    plot(x,y, c:red)
    # малая окружность:
    xc = rr*(k-1)*cos(t[n]) - rr*cos(θ)
    yc = rr*(k-1)*sin(t[n]) - rr*sin(θ)
    plot(xc,yc,c:black)
    # радиус малой окружности:
    x1 = transpose([rr*(k-1)*cos(t[n]), x[n]])
    y1 = transpose([rr*(k-1)*sin(t[n]), y[n]])
    plot(x1,y1,narkershape:circle,markersize=6,c:black)
    scatter([x[n]], [y[n]],c:red, markerstrokecolor:red)
end
```

Рис. 84: Задание 11 листинг (рацио)



В результате выполнения работы мы освоили синтаксис языка Julia для построения графиков. Были записаны скринкасты выполнения , создания отчета, презентации и защиты лабораторной работы.

## Список литературы

- Julia: <https://ru.wikipedia.org/wiki/Julia>
- <https://julialang.org/packages/>
- <https://juliahub.com/ui/Home>
- <https://juliaobserver.com/>
- <https://github.com/svaksha/Julia.jl>