

Отчёт по лабораторной работе №5

Построение графиков

Статический анализ данных

Выполнила: Коняева Марина Александровна,
НФИбд-01-21, 1032217044

Содержание

Цели лабораторной работы	4
Теоретическое введение	5
Задачи лабораторной работы	6
Выполнение лабораторной работы	7
Основные пакеты для работы с графиками в Julia	7
Опции при построении графика	11
Точечный график	14
Простой точечный график	14
Точечный график с кодированием значения размером точки	15
3-мерный точечный график с кодированием значения размером точки	16
Аппроксимация данных	17
Две оси ординат	19
Полярные координаты	21
Параметрический график	22
Параметрический график кривой на плоскости	22
Параметрический график кривой в пространстве	23
График поверхности	24
Линии уровня	28
Векторные поля	31
Анимация	34
Gif-анимация	34
Гипоцилоида	35
Errorbars	38
Использование пакета Distributions	43
Подграфики	46
Задания для самостоятельного выполнения	51
Выводы по проделанной работе	71
Вывод	71
Список литературы	72

Список иллюстраций

1	Установка пакетов	7
2	Построение графика функции	8
3	Построение графика функции	9
4	Построение графика функции	10
5	Построение графика функции	10
6	Функция	11
7	Построение графика функции	11
8	Построение графика функции	12
9	Построение графиков функций	12
10	Листинг	13
11	Построение графика функции	14
12	Сохранение	14
13	Построение графика	15
14	Построение графика	16
15	Построение графика	17
16	Аппроксимация данных	18
17	Листинг и построение графика	18
18	Листинг и построение графика	19
19	Пример отдельно построенной траектории	20
20	Пример траекторий	21
21	Листинг и построение графика	22
22	Листинг и построение графика	23
23	Листинг и построение графика	24
24	Листинг и построение графика	25
25	Листинг и построение графика	26
26	Листинг и построение графика	27
27	Листинг и построение графика	28
28	Листинг и построение графика	29
29	Листинг и построение графика	30
30	Листинг и построение графика	31
31	Построение графика	32
32	Листинг и построение графика	32
33	Построение графика	33
34	Построение графика	34
35	Листинг и построение графика	35
36	Листинг	36
37	Построение графика	37

38	Листинг и анимация	38
39	Подключение пакета	39
40	График исходных значений	39
41	График исходных значений с отклонениями	40
42	Листинг и построение графика	40
43	Листинг и построение графика	41
44	Листинг и построение графика	42
45	Листинг и построение графика	43
46	Гистограмма	44
47	Гистограмма распределения людей по возрастам	45
48	Гистограмма распределения людей по возрастам	46
49	Серия из 4-х графиков в ряд	47
50	Серия из 4-х графиков в сетке	48
51	Серия из 4-х графиков разной высоты в ряд	48
52	Листинг	49
53	Объединение нескольких графиков в одной сетке	49
54	Листинг и разнообразные варианты представления данных	50
55	Листинг и сложный макет	51
56	Задание 1	52
57	Задание 2 листинг	53
58	Задание 2 график	53
59	Задание 3 листинг и график	54
60	Задание 3 листинг	55
61	Задание 3 график	55
62	Задание 4 листинг	56
63	Задание 4 график	57
64	Задание 5 листинг и график	57
65	Задание 5 листинг и график	58
66	Задание 6 листинг и график	59
67	Задание 7 листинг и график	60
68	Задание 8 листинг и график	61
69	Задание 9 листинг и анимация	62
70	Задание 10 листинг (целые)	62
71	Задание 10 анимация (целые)	63
72	Задание 10 листинг (целые)	63
73	Задание 10 анимация (целые)	64
74	Задание 10 листинг (рацио)	64
75	Задание 10 анимация (рацио)	65
76	Задание 10 листинг (рацио)	65
77	Задание 10 анимация (рацио)	66
78	Задание 11 листинг (целые)	67
79	Задание 11 анимация (целые)	67
80	Задание 11 листинг (целые)	68
81	Задание 11 анимация (целые)	68

82	Задание 11 листинг (рацио)	69
83	Задание 11 анимация (рацио)	69
84	Задание 11 листинг (рацио)	70
85	Задание 11 анимация (рацио)	70

Цели лабораторной работы

Освоить синтаксис языка Julia для построения графиков.

Теоретическое введение

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`
[d330b81b] + PyPlot v2.11.5
Updating `C:\Users\User\.julia\environments\v1.10\Manifest.toml`
[438e738f] + PyCall v1.96.4
[d330b81b] + PyPlot v2.11.5
Building PyCall → `C:\Users\User\.julia\scratchspaces\44cfe95a-1eb2-52ea-b672-e2afdf69b78f\981
```

Рис. 1: Установка пакетов

- Повторим пример: рассмотрим построение графика функции $y(x) = (3x^2 + 6x - 9)*x^{-0,3}$ 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:
f(x) = (3x.^2 + 6x .. 9).*exp.(-0.3x)
# генерирование массива значений y 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")

```

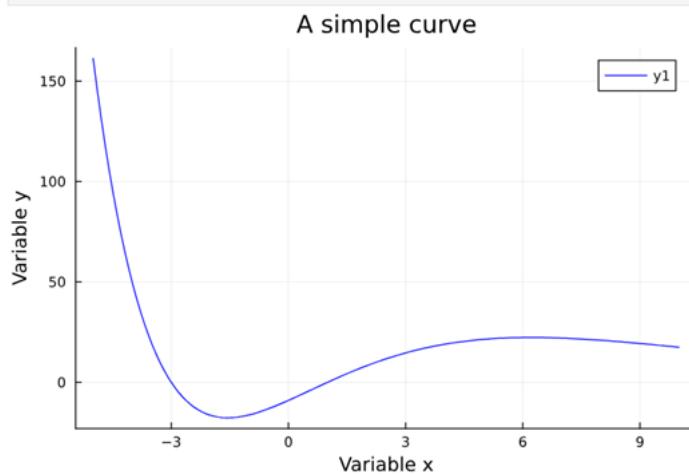


Рис. 2: Построение графика функции

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.

```

# 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")

```

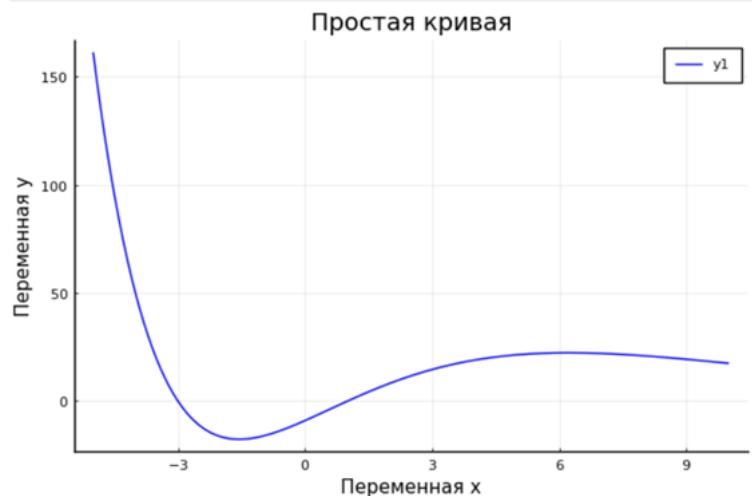


Рис. 3: Построение графика функции

4. Выполним упражнение, построим график функции $\square(\square) = (3\square^2 + 6\square - 9)*\square^{-0,3}$ при помощи plotly().

```

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

```

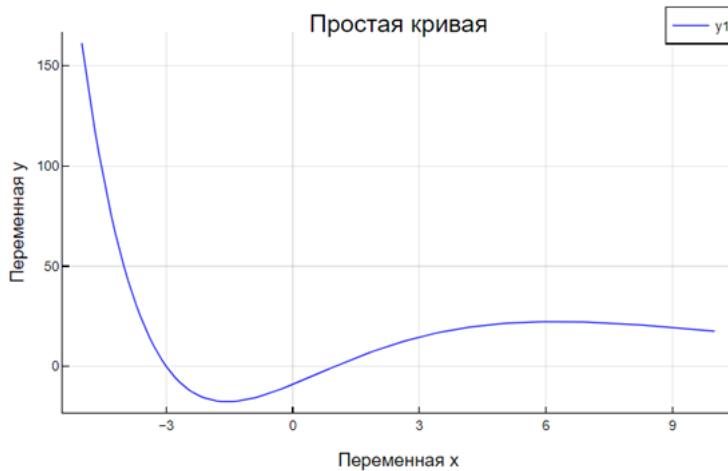


Рис. 4: Построение графика функции

5. Выполним упражнение, построим график функции $\square(\square) = (3\square^2 + 6\square - 9)*\square^{-0,3\square}$ при помощи `unicodeplots()`.

```

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

```



Рис. 5: Построение графика функции

Опции при построении графика

6. Повторим пример 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)
```

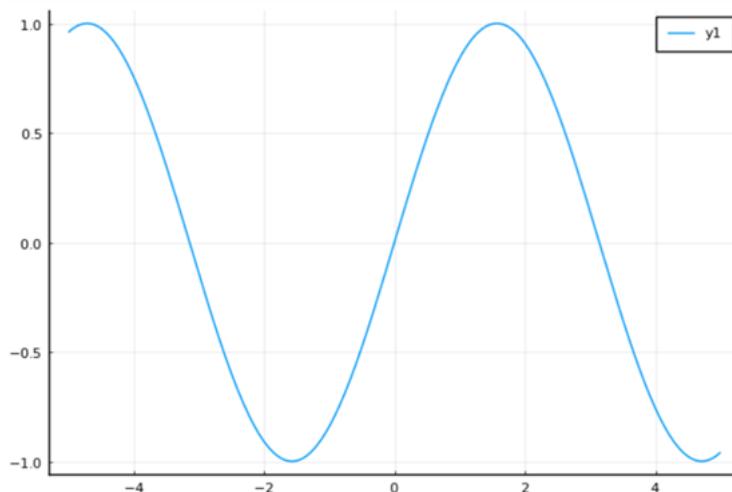


Рис. 7: Построение графика функции

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) = [(-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)
```

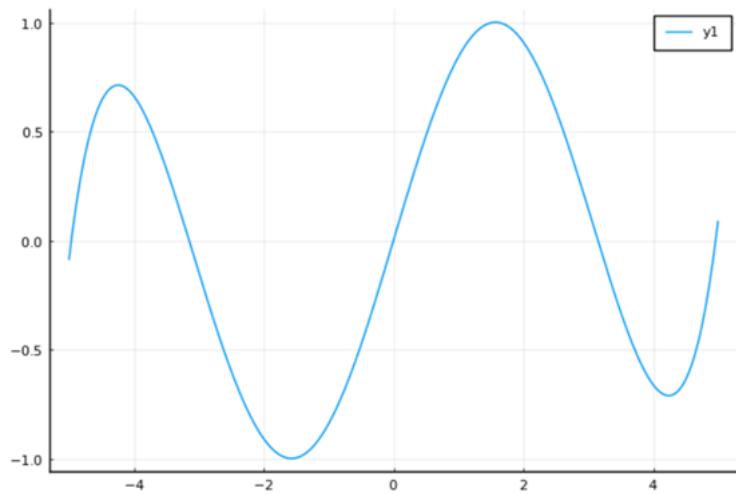


Рис. 8: Построение графика функции

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

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

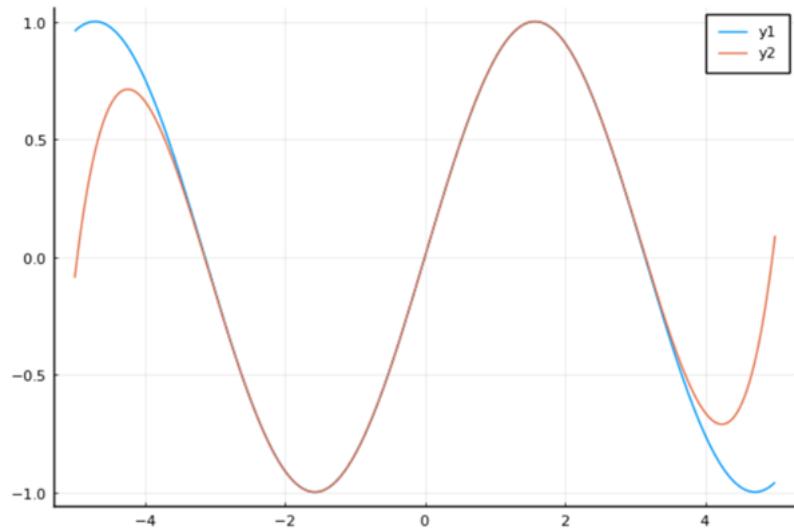


Рис. 9: Построение графиков функций

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 = (-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",  
    line=(:black, 1.0, 2, :dash))
```

Рис. 10: Листинг

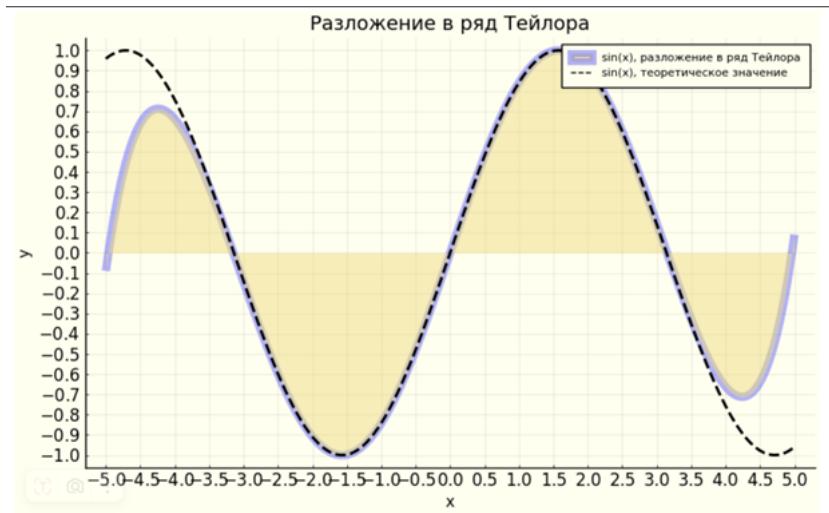


Рис. 11: Построение графика функции

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

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

Рис. 12: Сохранение

Точечный график

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 \square , посчитать или задать значения \square , задать опции построения графика, добавляя надписи осей, легенды и названия.

```

# 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")

```

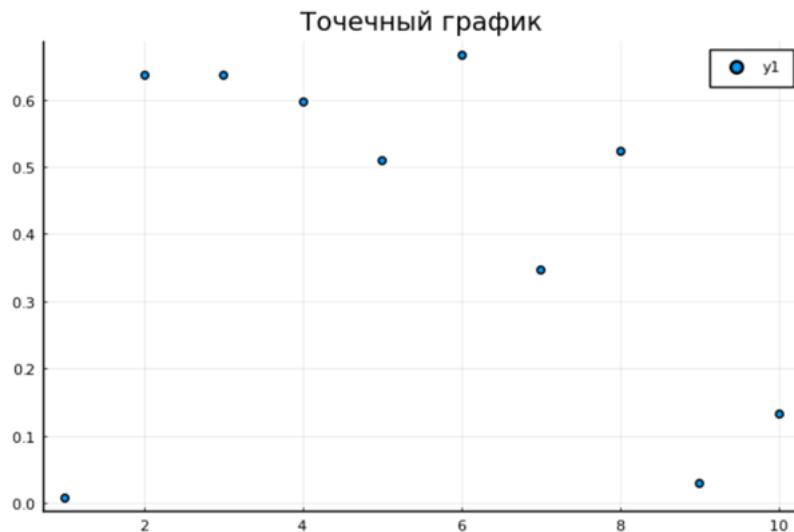


Рис. 13: Построение графика

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.

```

# параметры распределения точек на плоскости:
n = 50
x = rand(n)
y = rand(n)
ms = rand(n) * 30
# параметры построения графика:
scatter(x, y,
       label = "Случайные точки в 2D пространстве",
       leg::topright,
       ylabel = "y",
       xlabel = "x",
       title = "Случайные точки",
       markersize=ms)

```

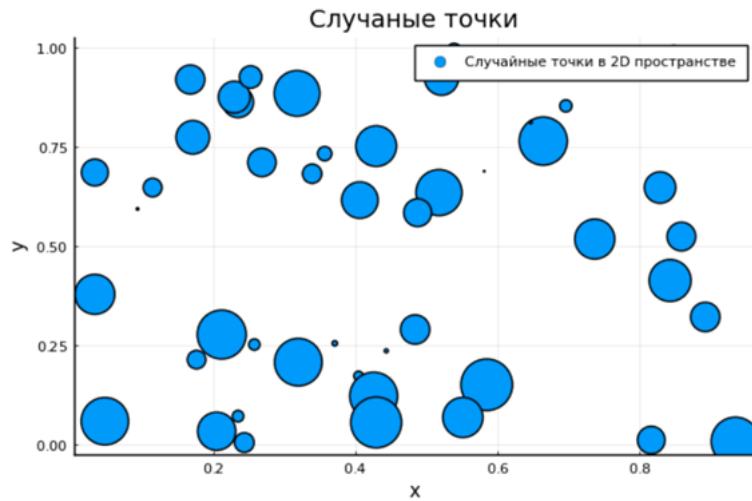


Рис. 14: Построение графика

3-мерный точечный график с кодированием значения размером точки

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

```
# параметры распределения точек в пространстве:  
n = 50  
x = rand(n)  
y = rand(n)  
z = rand(n)  
ms = rand(n) * 30  
# параметры построения графика:  
scatter(x, y, z,  
       label = "Случайные точки в 3D пространстве",  
       leg=:topleft,  
       ylabel = "y",  
       xlabel = "x",  
       zlabel = "z",  
       title = "Случайные точки",  
       markersize=ms)
```



Рис. 15: Построение графика

Аппроксимация данных

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)
```

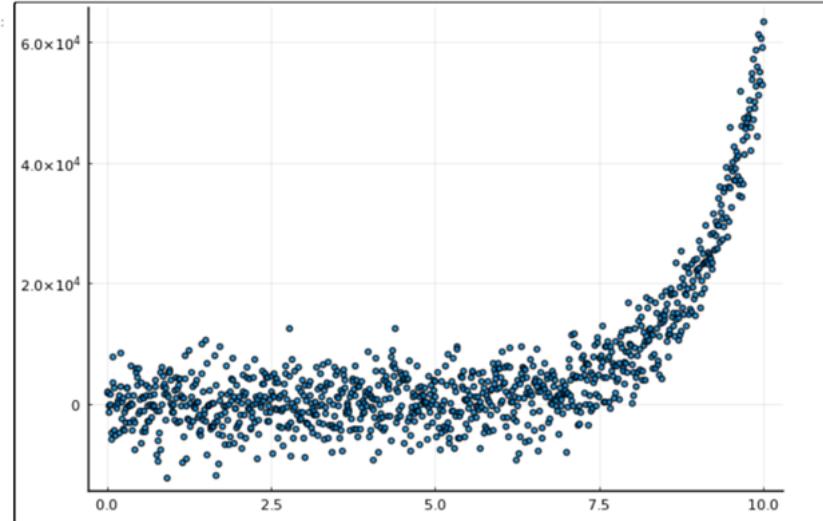


Рис. 17: Листинг и построение графика

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 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)

```

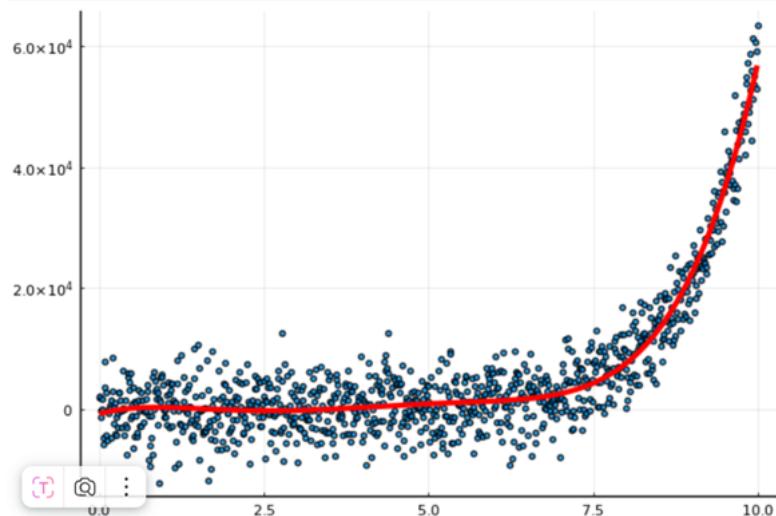


Рис. 18: Листинг и построение графика

Две оси ординат

18. Иногда требуется на один график вывести несколько траекторий с существенными различиями 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",
      leg=:topright,
      grid = :off,
      )

```

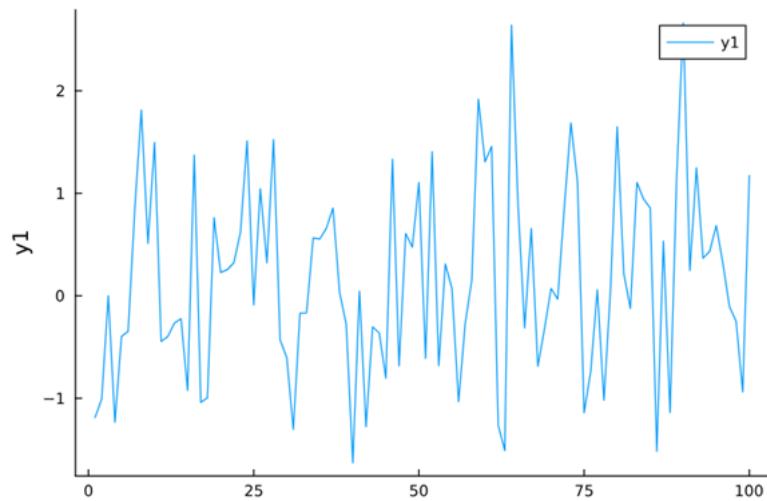


Рис. 19: Пример отдельно построенной траектории

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)
      )

```

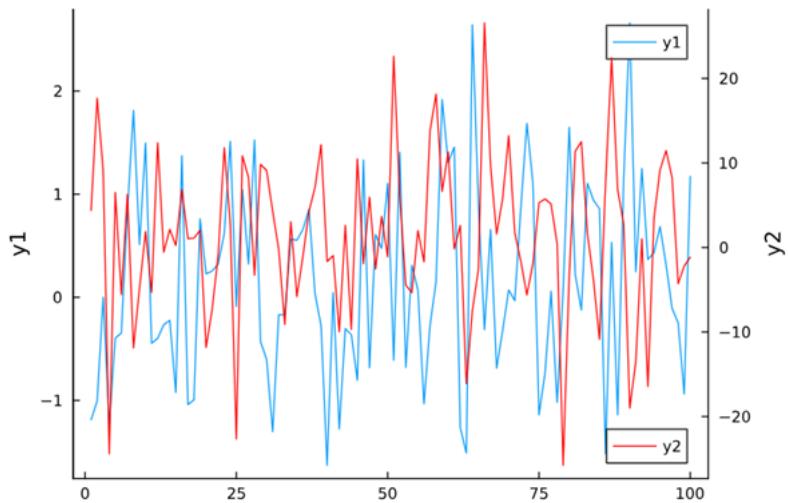


Рис. 20: Пример траекторий

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))

```

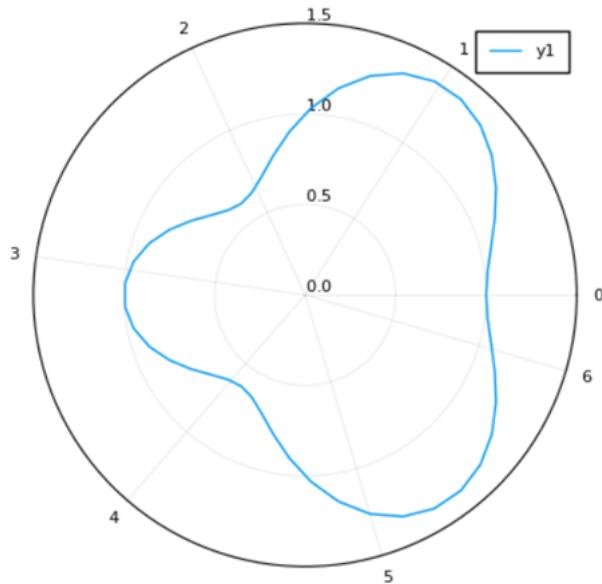


Рис. 21: Листинг и построение графика

Параметрический график

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 $x(\theta) = \sin(\theta)$, $y(\theta) = \sin(2\theta)$, $0 \leq \theta \leq 2\pi$.

```

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

```

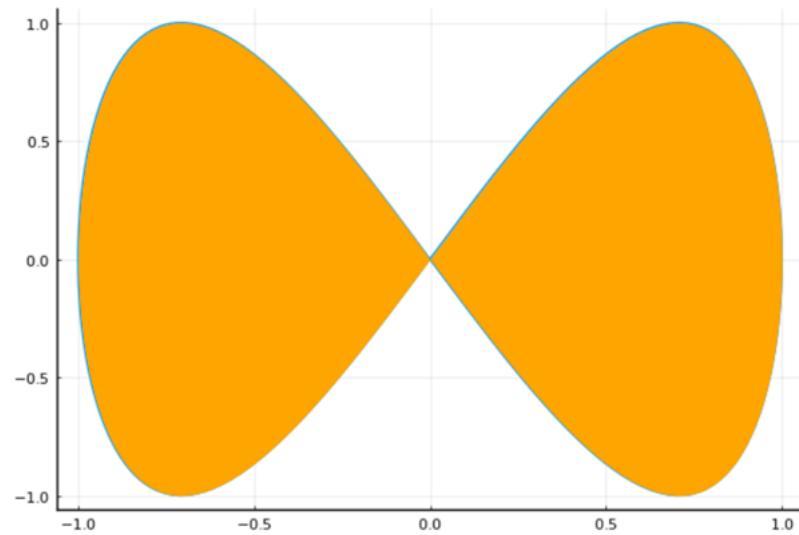


Рис. 22: Листинг и построение графика

Параметрический график кривой в пространстве

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

```

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

```

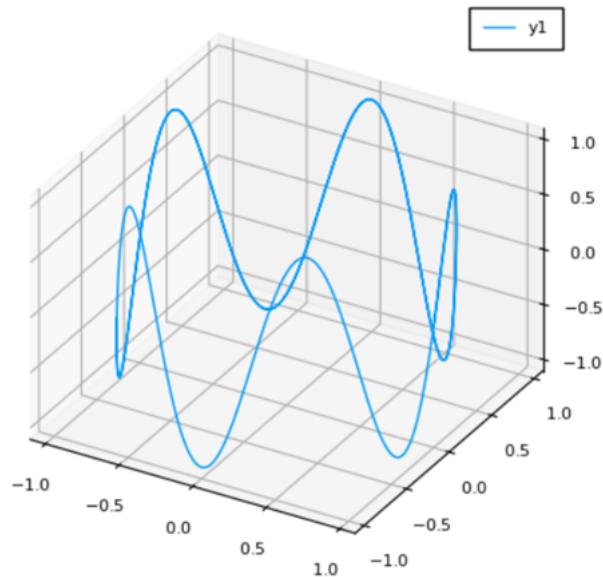


Рис. 23: Листинг и построение графика

График поверхности

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

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

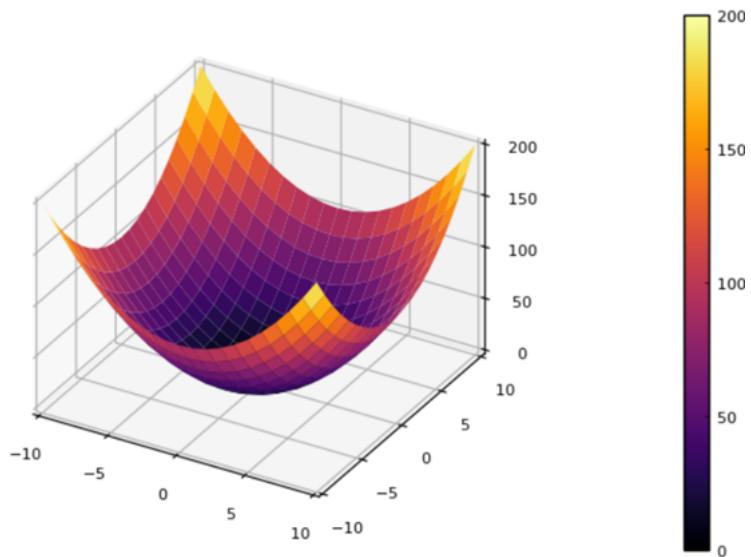


Рис. 24: Листинг и построение графика

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)
```

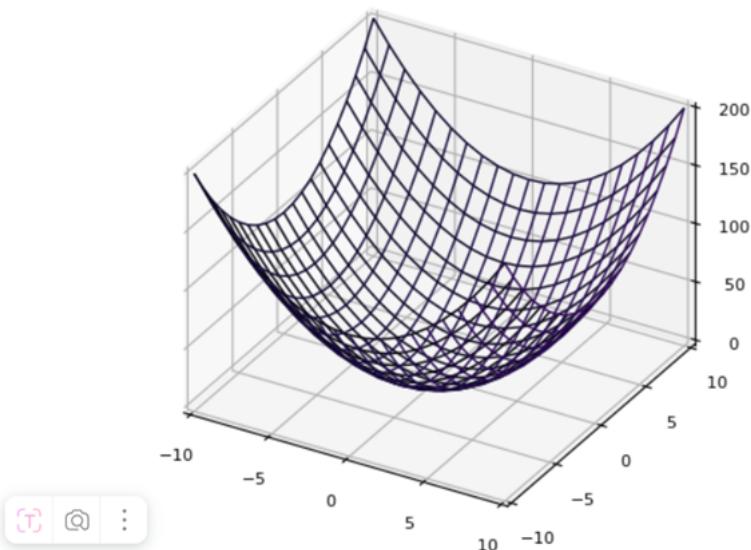


Рис. 25: Листинг и построение графика

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

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

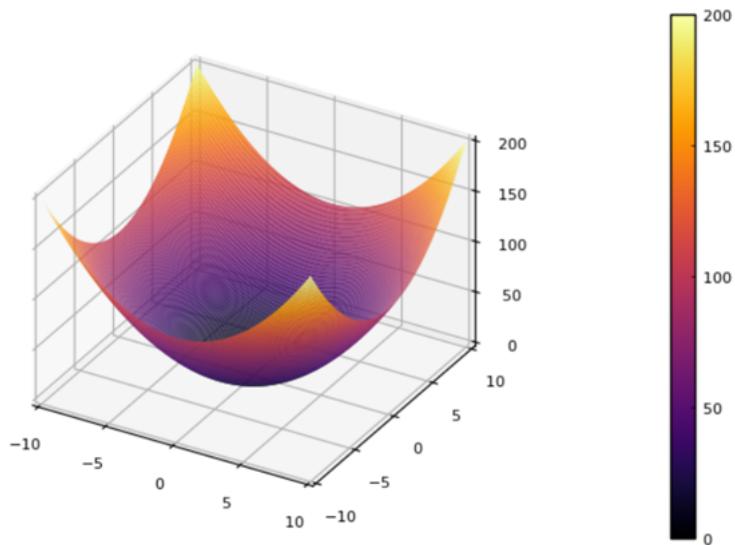


Рис. 26: Листинг и построение графика

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+1
plot(x,y,f,
    linetype = :surface,
    c=cgrad([:red,:blue]),
    camera=(-30,30),
)

```

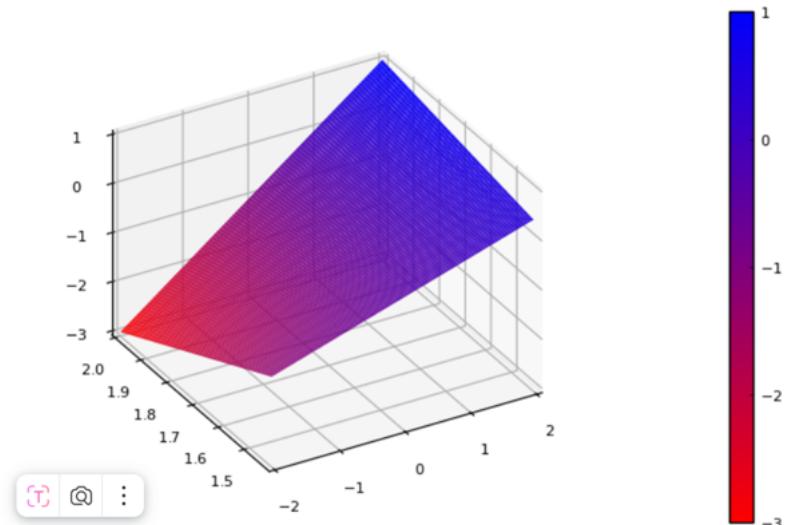


Рис. 27: Листинг и построение графика

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 $\varphi(x, y) = (3x + y^2)|\sin(x) + \cos(y)|$.

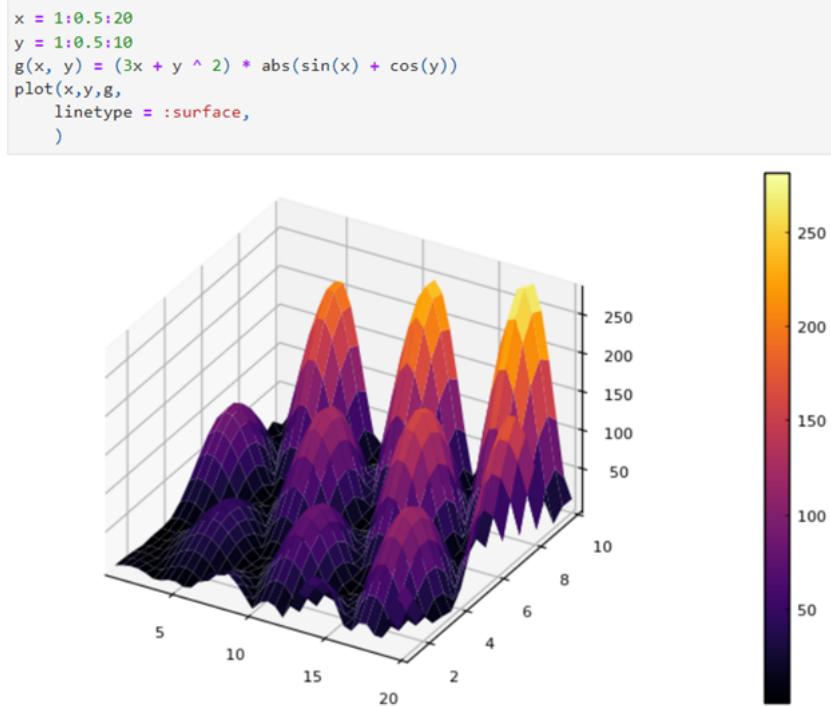


Рис. 28: Листинг и построение графика

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

```
contour(x, y, g, xlabel="x",
        ylabel="y",
        title="Линии уровня")
```

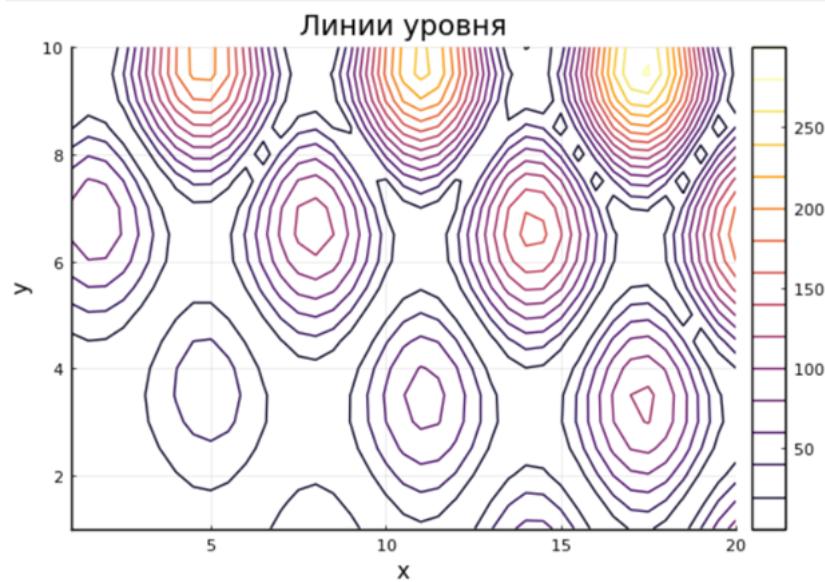


Рис. 29: Листинг и построение графика

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

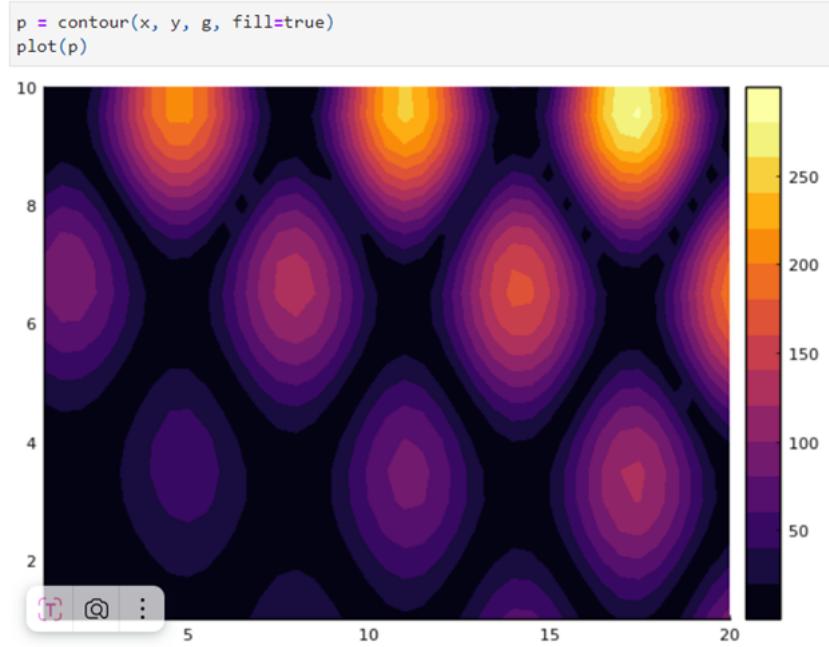


Рис. 30: Листинг и построение графика

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 $h(x, y) = x^3 - 3x + y^2$ сначала построим её график и линии уровня.

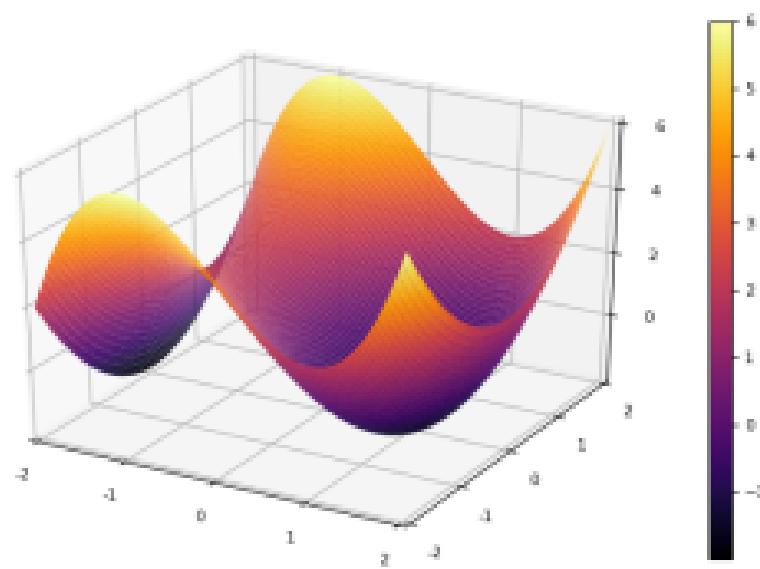


Рис. 31: Построение графика

```
# 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)
```

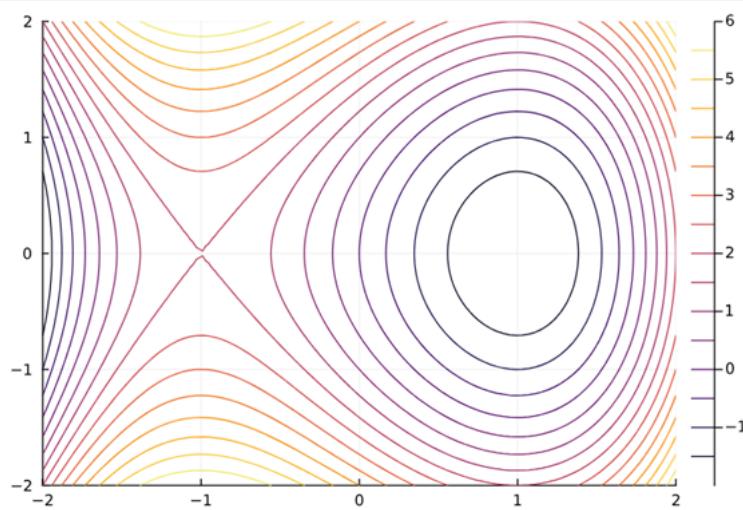


Рис. 32: Листинг и построение графика

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, c=:blue)
# коррекция области видимости графика:
xlims!(-2, 2)
ylims!(-2, 2)
```

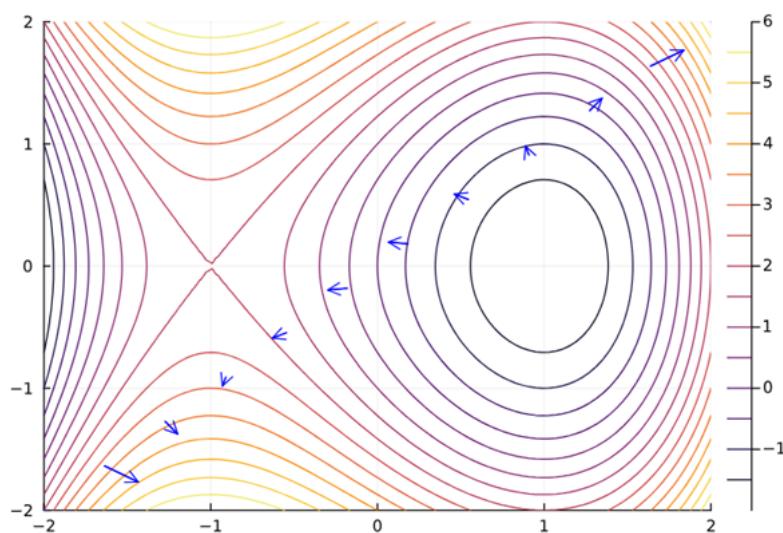


Рис. 33: Построение графика

```

# 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)
# построение векторного поля:
quiver!(x, y, quiver=dh, c=:blue)
# коррекция области видимости графика:
xlims!(-2, 2)
ylims!(-2, 2)

```

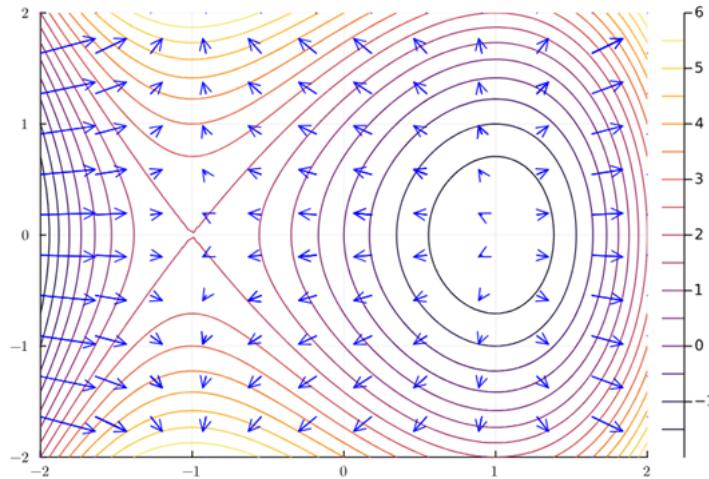


Рис. 34: Построение графика

Анимация

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.

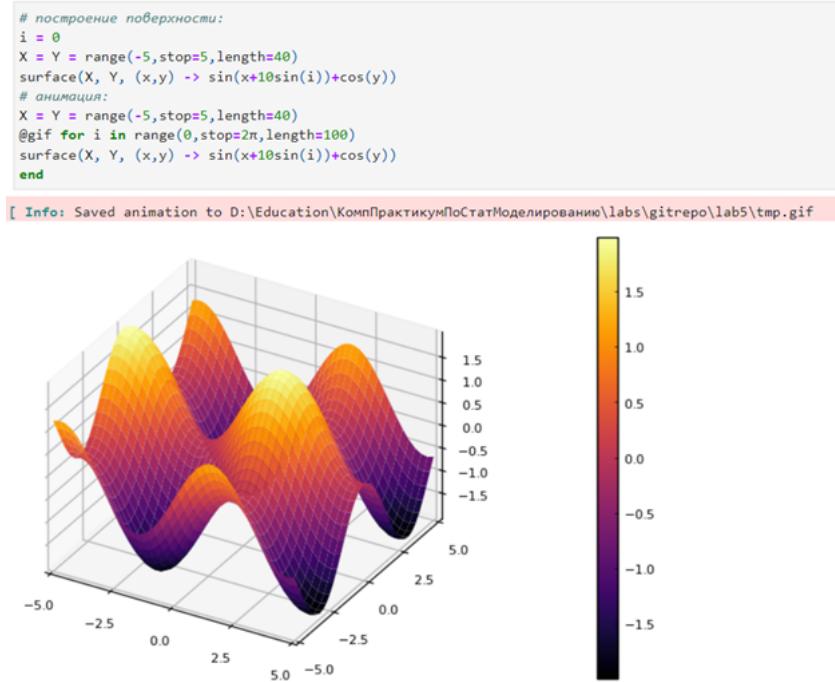


Рис. 35: Листинг и построение графика

Гипоциклоида

33. Выполним пример: потроим 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(0:2*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)
i = 50
t = θ[1:i]
# гипотенузы:
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

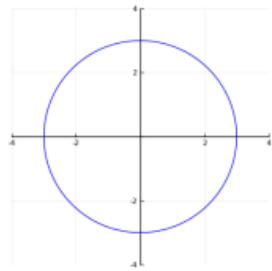


Рис. 5.30. Большая окружность гипоциклоиды

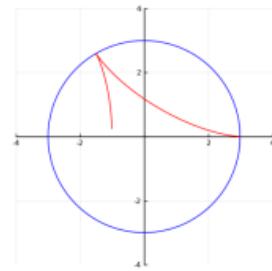


Рис. 5.31. Половина пути гипоциклоиды

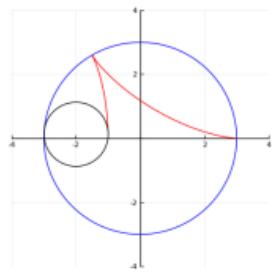


Рис. 5.32. Малая окружность гипоциклоиды Рис. 5.33. Малая окружность гипоциклоиды
с добавлением радиуса

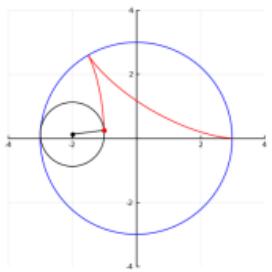


Рис. 37: Построение графика

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

```

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 = 0[1:i]
    # гипоциклоиды:
    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")

```

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

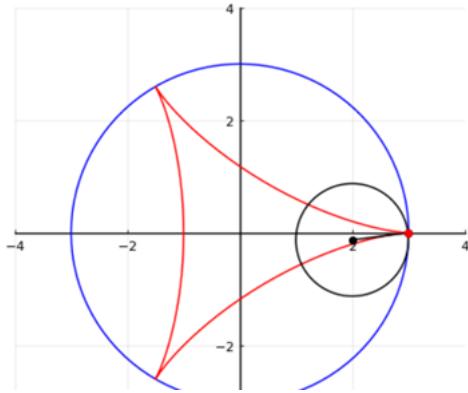


Рис. 38: Листинг и анимация

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

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

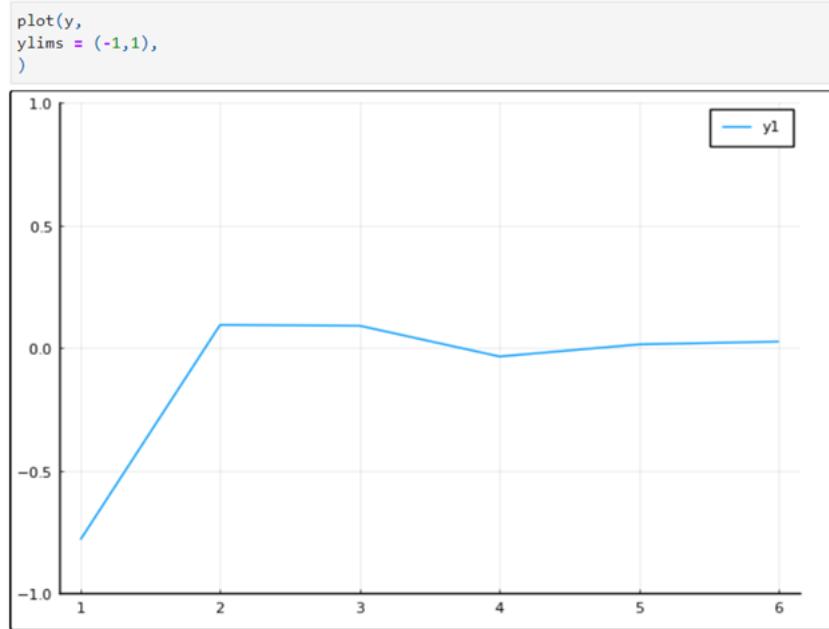


Рис. 40: График исходных значений

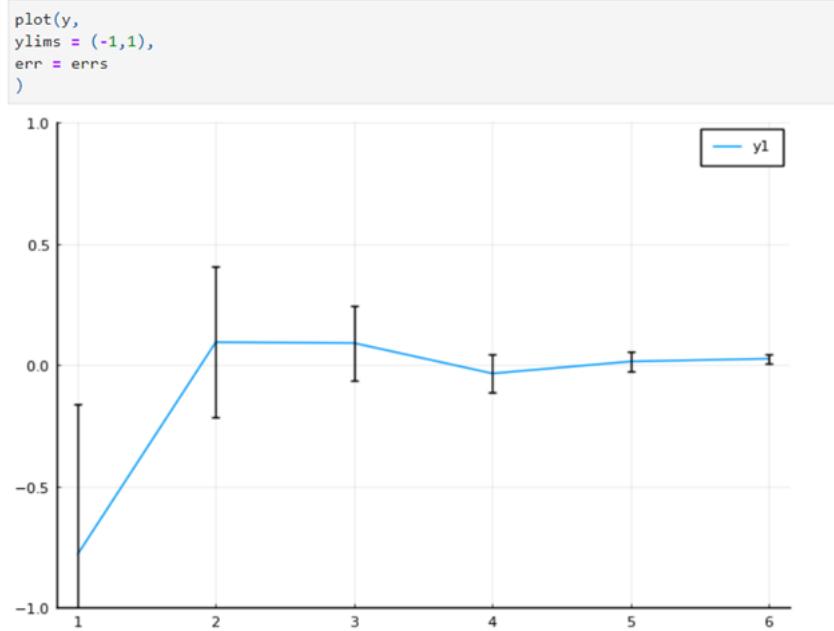


Рис. 41: График исходных значений с отклонениями

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

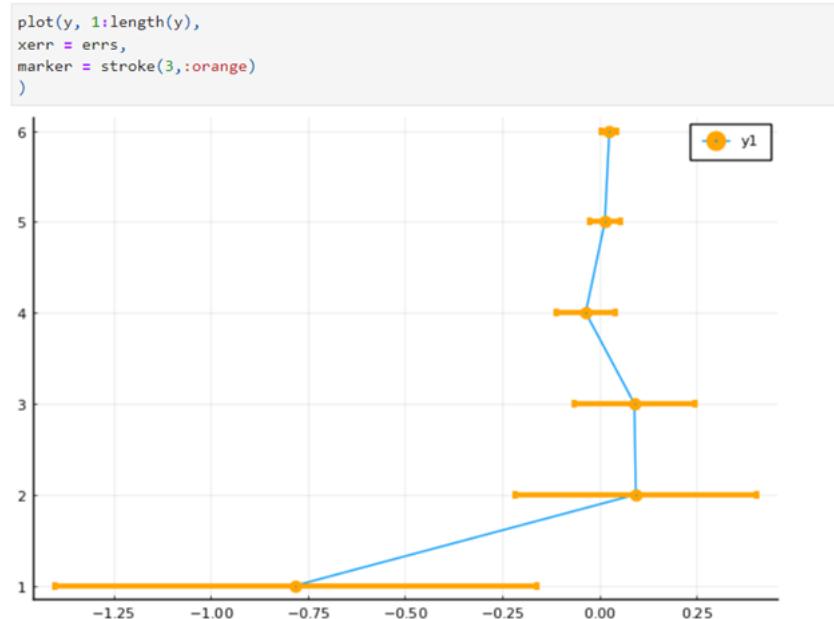


Рис. 42: Листинг и построение графика

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

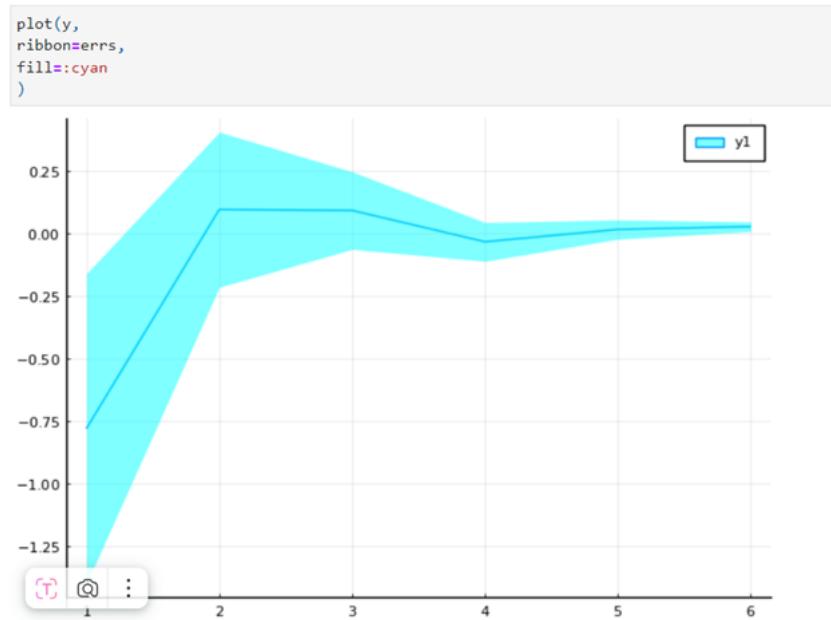


Рис. 43: Листинг и построение графика

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

```

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)
)

```

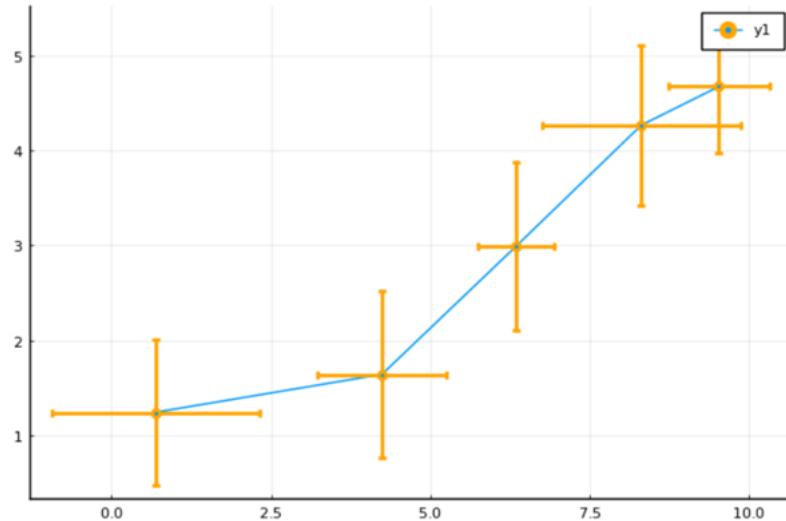


Рис. 44: Листинг и построение графика

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

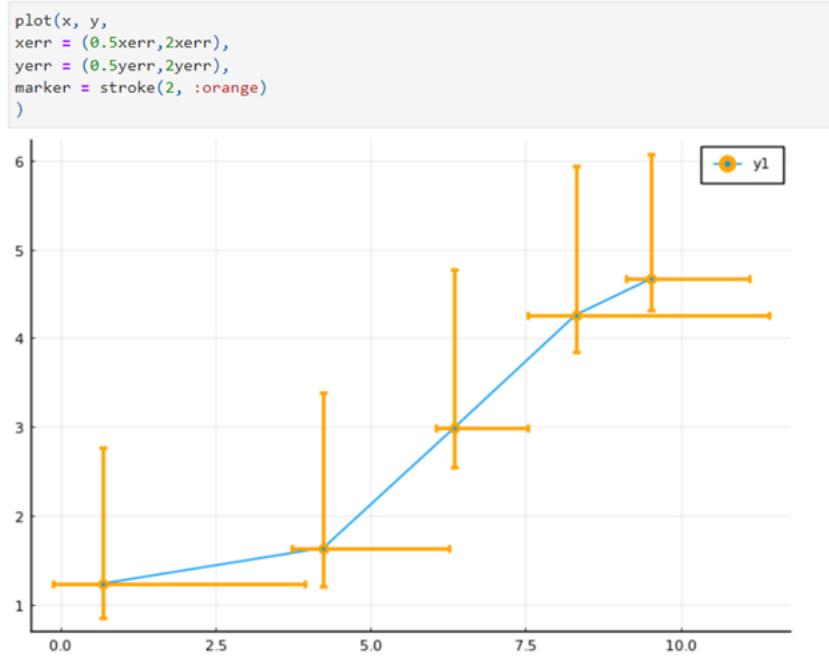


Рис. 45: Листинг и построение графика

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"
)

```

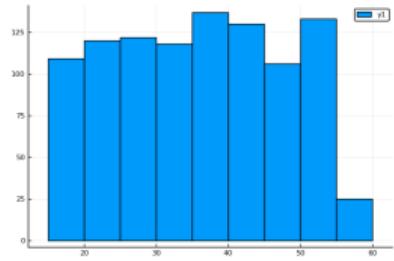


Рис. 5.41. Гистограмма, построенная по массиву случайных чисел

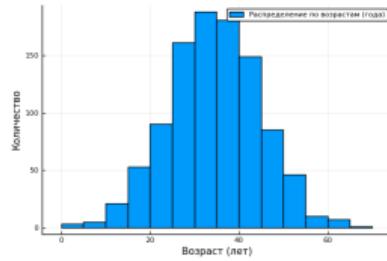


Рис. 5.42. Гистограмма нормального распределения

Рис. 46: Гистограмма

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

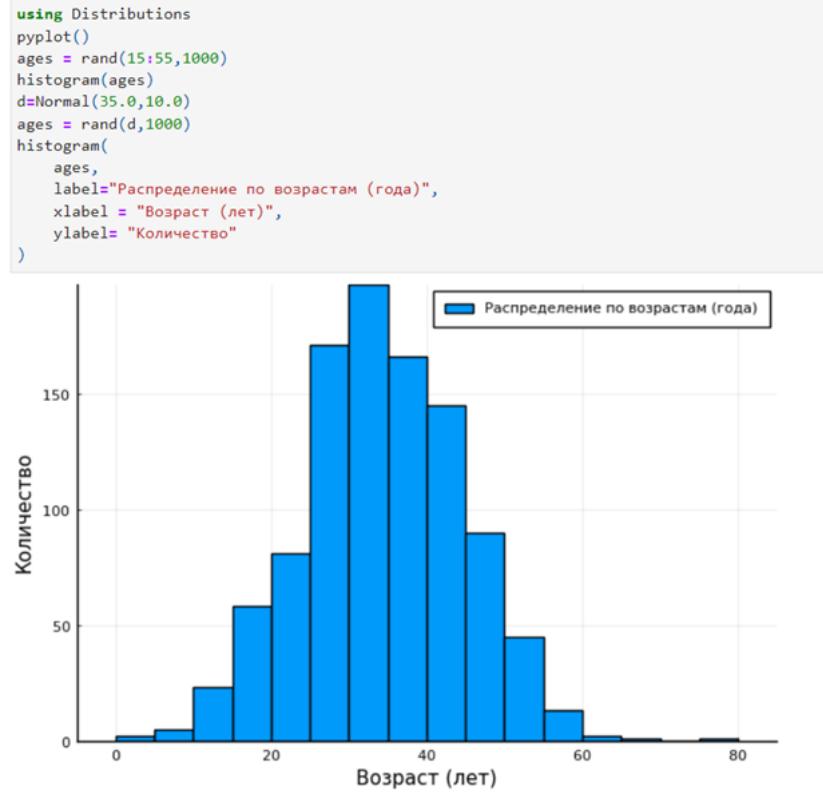


Рис. 47: Гистограмма распределения людей по возрастам

```

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(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"
)

```

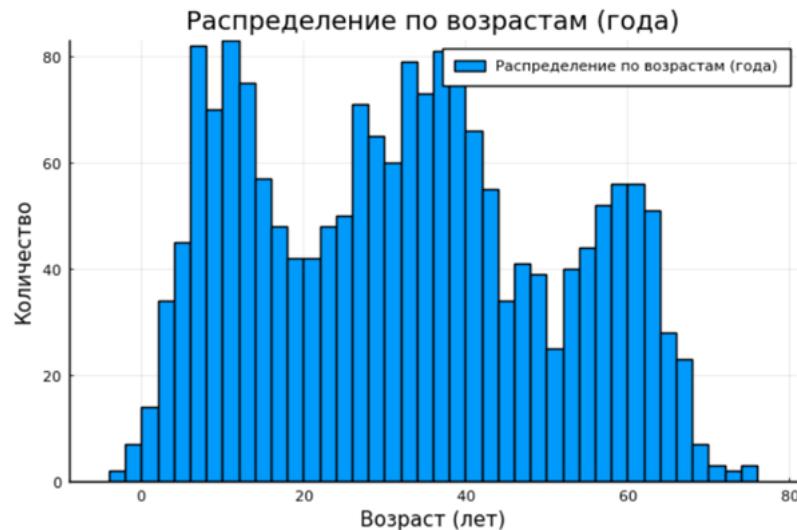


Рис. 48: Гистограмма распределения людей по возрастам

Подграфики

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

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

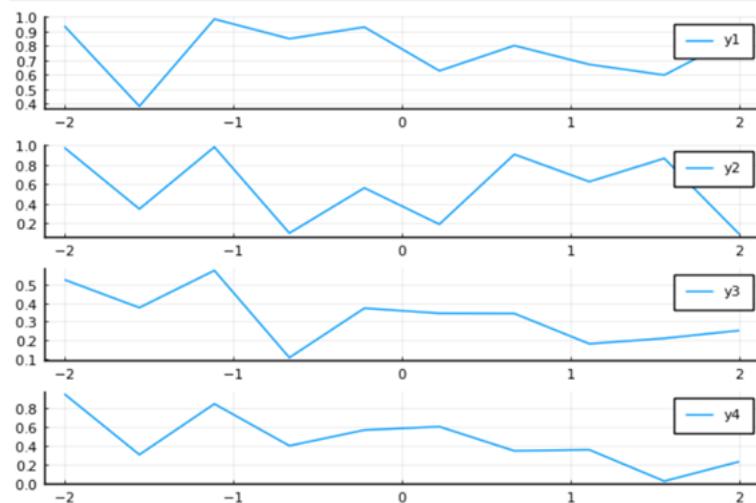


Рис. 49: Серия из 4-х графиков в ряд

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

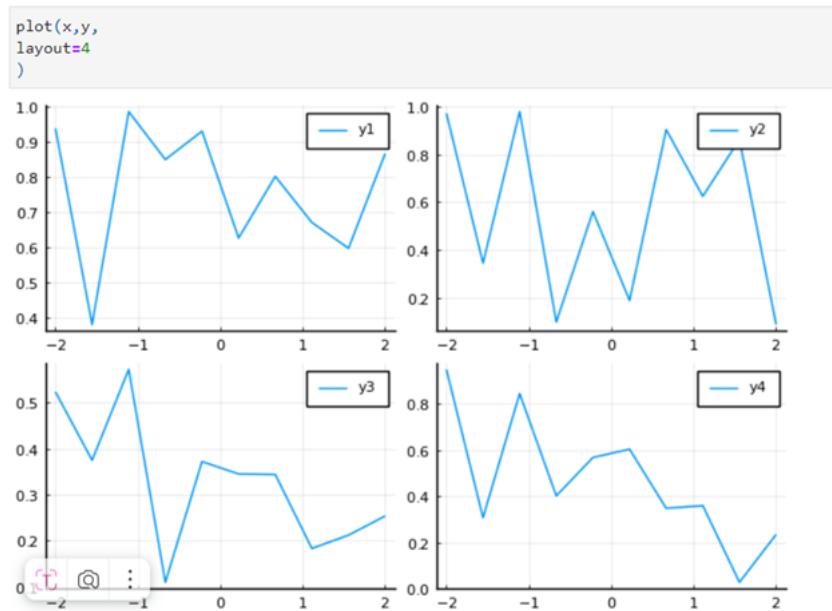


Рис. 50: Серия из 4-х графиков в сетке

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.

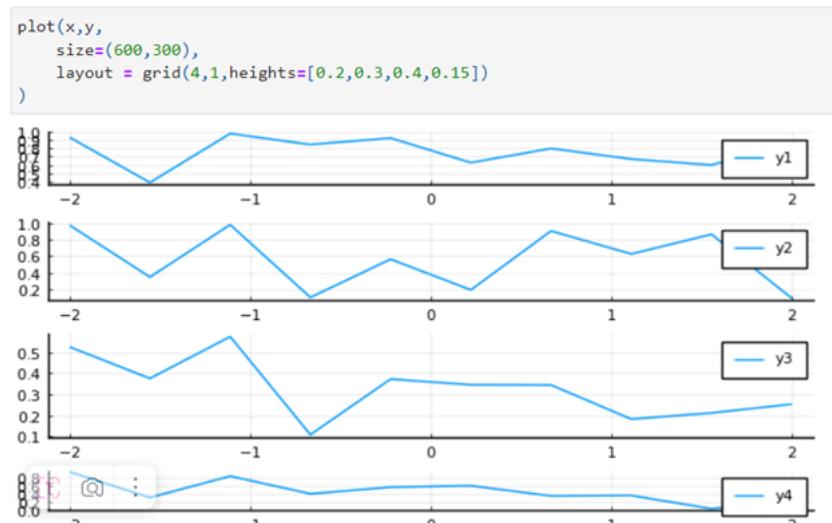


Рис. 51: Серия из 4-х графиков разной высоты в ряд

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 2×2 .

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

Рис. 52: Листинг

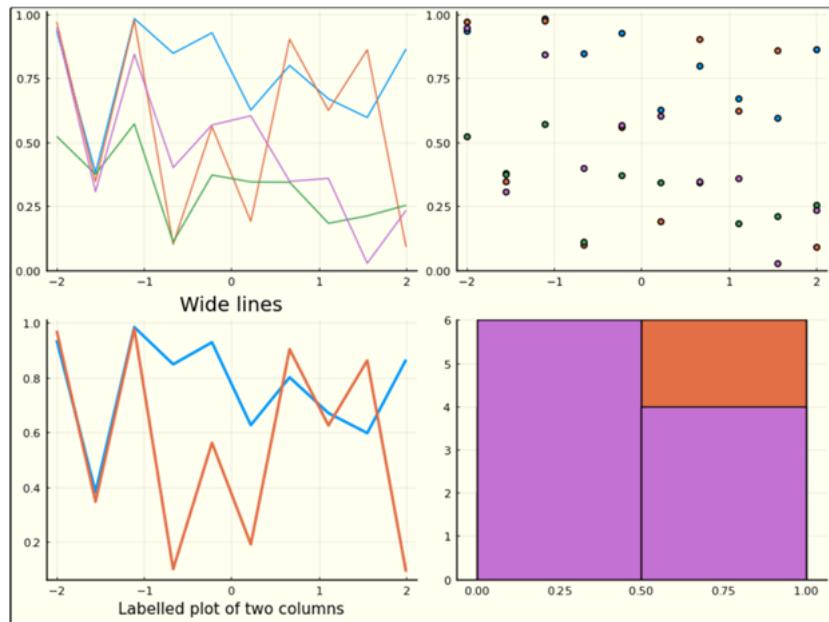


Рис. 53: Объединение нескольких графиков в одной сетке

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.

```

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

```

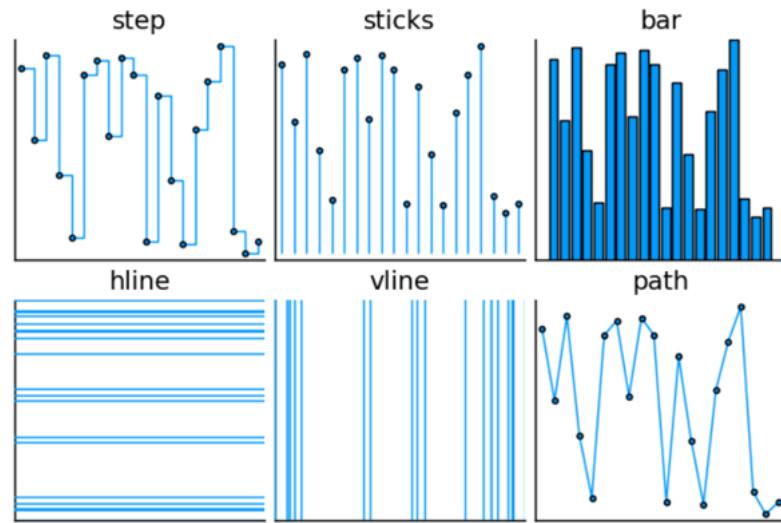


Рис. 54: Листинг и разнообразные варианты представления данных

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.

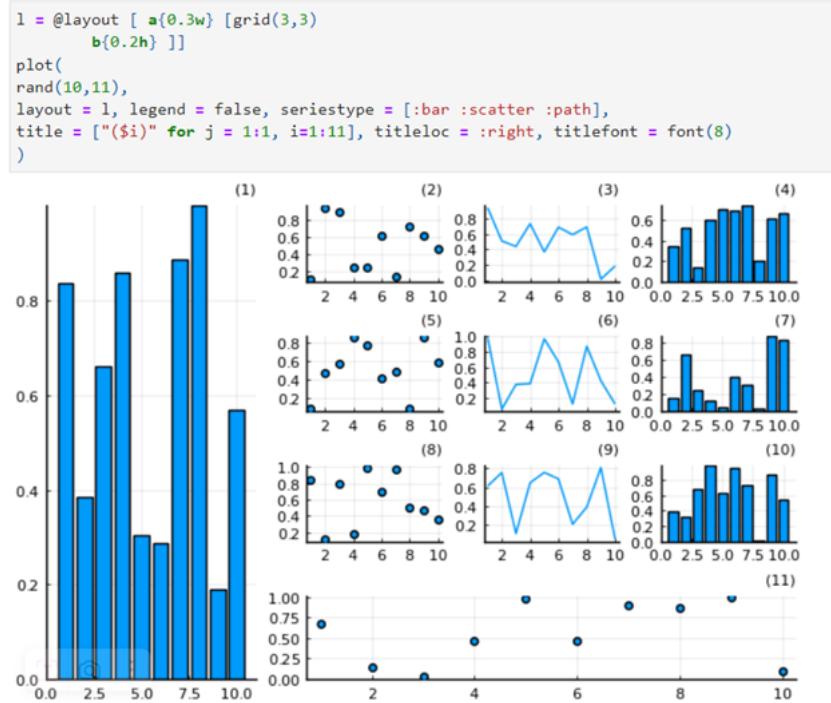


Рис. 55: Листинг и сложный макет

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.

```

seriesTypes = [:step, :sticks, :bar, :scatter, :hline, :vline, :path]
titles = ["step" "sticks" "bar" "scatter" "hline" "vline" "path"]
x_values = collect(range(1, stop=2*pi, length=25))
y2(x) = sin(x)
plot(x_values, y2,
    st = seriesTypes,
    layout = 7,
    ticks=nothing,
    legend=false,
    title=titles,
    m=3
)

```

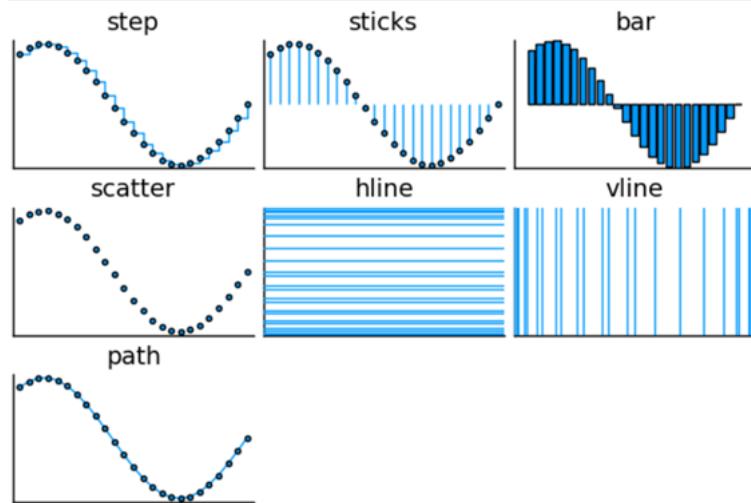


Рис. 56: Задание 1

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

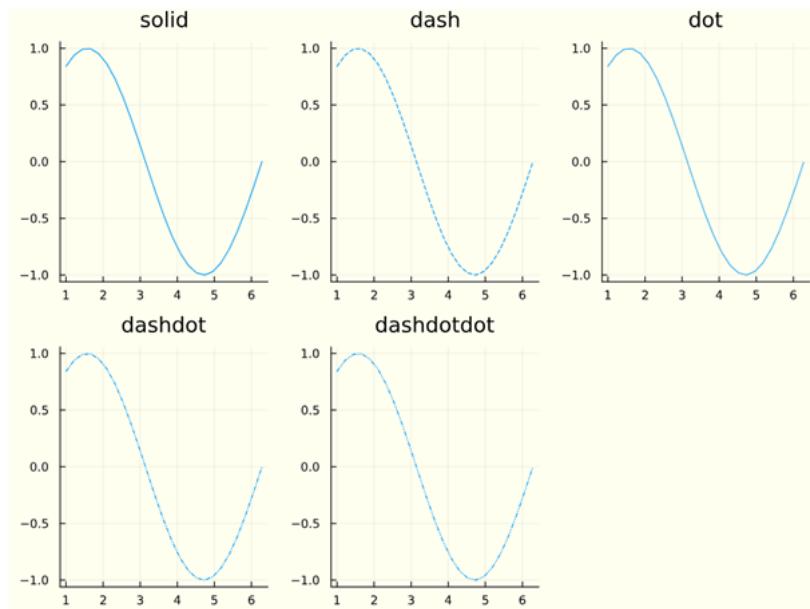


Рис. 58: Задание 2 график

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 такч 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),
)
```

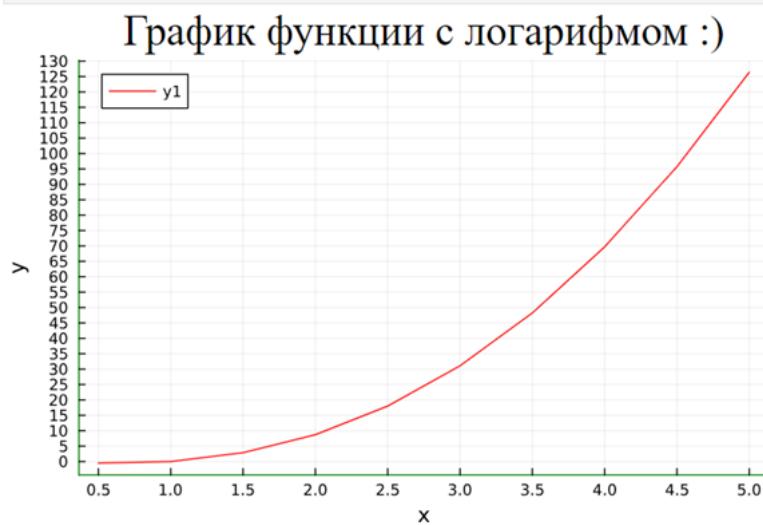


Рис. 59: Задание 3 листинг и график

```

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,
    legend=false,
)

```

Рис. 60: Задание 3 листинг

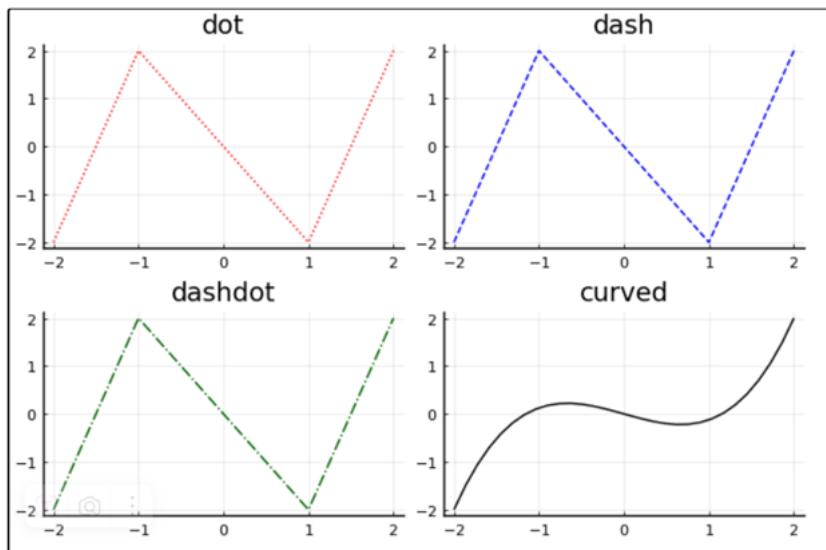


Рис. 61: Задание 3 график

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)
plot(p3,p2,p4,p1,
    layout=4,
    legend=false,
)
```

Рис. 62: Задание 4 листинг

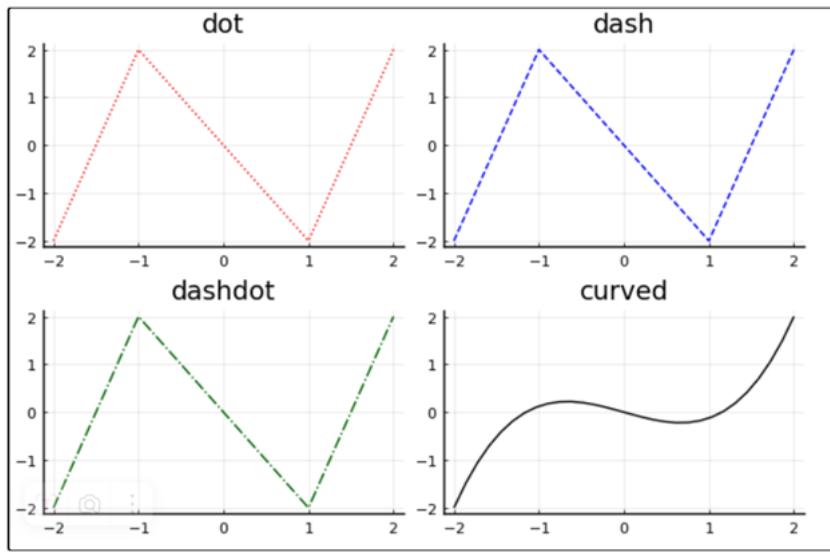


Рис. 63: Задание 4 график

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

```

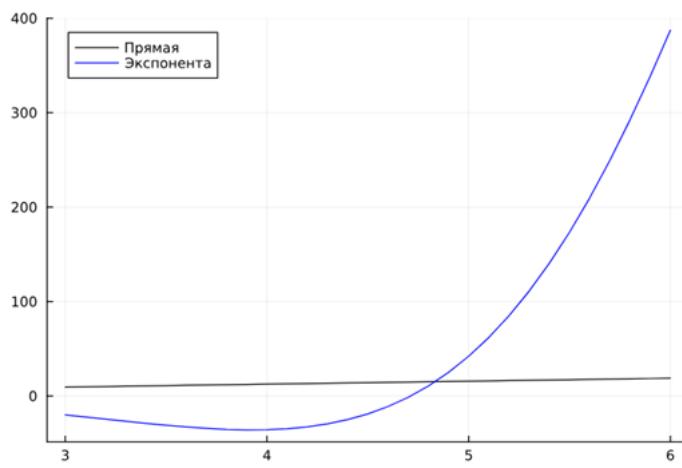


Рис. 64: Задание 5 листинг и график

```

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

```

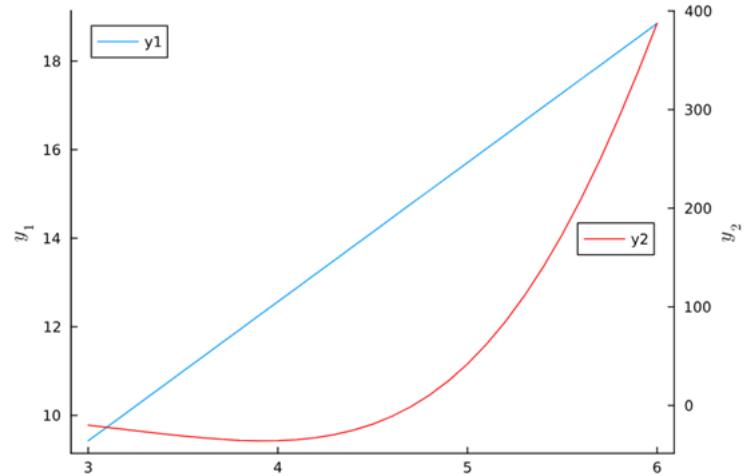


Рис. 65: Задание 5 листинг и график

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
)
```

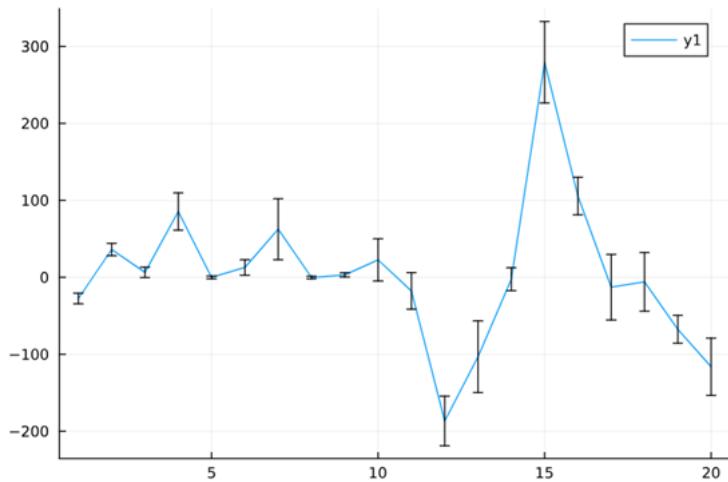


Рис. 66: Задание 6 листинг и график

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

```
n = 50
x = rand(n)
y = rand(n)
ms = rand(n) * 10
# параметры построения графика:
scatter(x, y,
       label = "Случайные точки",
       leg=:topright,
       ylabel = "y",
       xlabel = "x",
       title = "Случайные точки в 2D пространстве",
       markersize=ms)
```

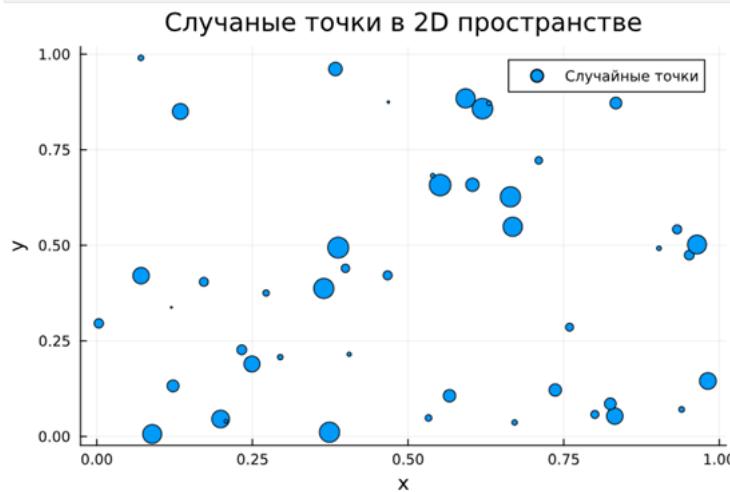


Рис. 67: Задание 7 листинг и график

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

```

n = 150
x = rand(n)
y = rand(n)
z = rand(n)
ms = rand(n) * 10
# параметры построения графика:
scatter(x, y, z,
       label = "Случайные точки",
       leg=opleft,
       ylabel = "y",
       xlabel = "x",
       zlabel = "z",
       title = "Случайные точки в 3D пространстве",
       markersize=ms,
       mc = :purple
)

```

Случайные точки в 3D пространстве

● Случайные точки

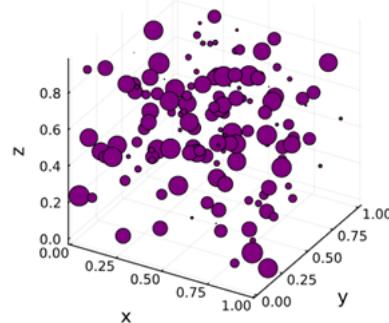


Рис. 68: Задание 8 листинг и график

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

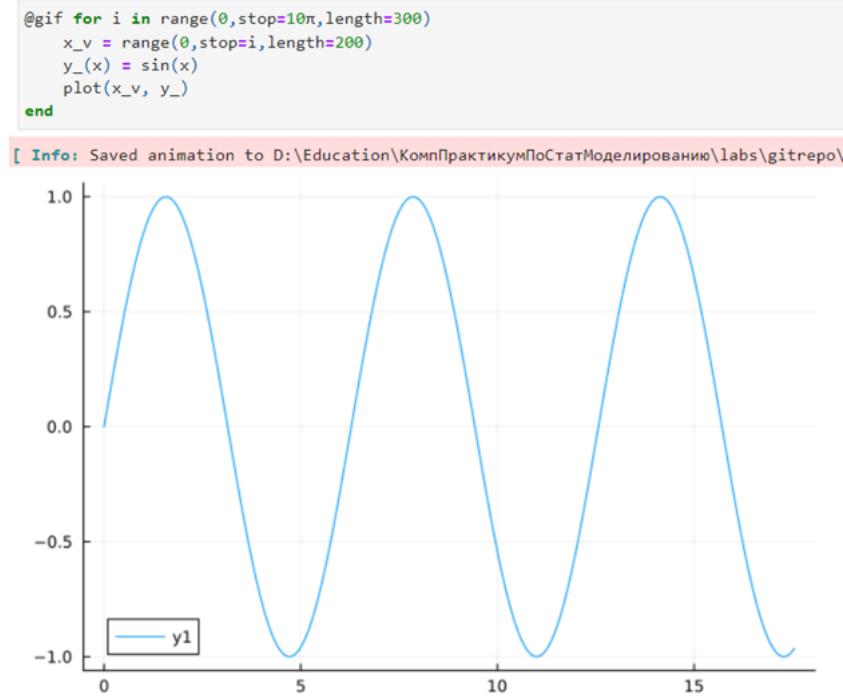


Рис. 69: Задание 9 листинг и анимация

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

```

r1 = 1
k = 6
n = 100
θ = collect(0:2π/100:2π+2*π/100)
X = r1*k*cos.(θ)
Y = r1*k*sin.(θ)
anim = @animate for i in 1:n
    # задаём оси координат:
    plt=plot(5,xlim=(-k-1,k+1), ylim=(-k-1,k+1), 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)
    # редущая 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_1.gif")

```

Рис. 70: Задание 10 листинг (целые)

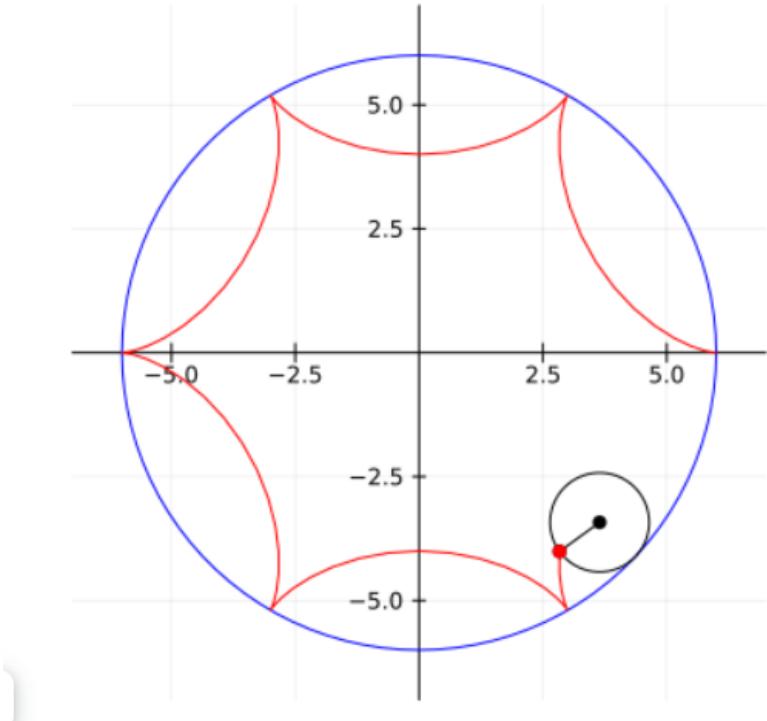


Рис. 71: Задание 10 анимация (целые)

```
r1 = 1
k = 15
n = 200
θ = collect(0:2*π/n:2*π+2*π/n)
X = r1*k*cos.(θ)
Y = r1*k*sin.(θ)
anim = @animate for i in 1:n
    # задаём оси координат:
    plt=plot(5,xlim=(-k-1,k+1), ylim=(-k-1,k+1), 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_2.gif")
```

Рис. 72: Задание 10 листинг (целые)

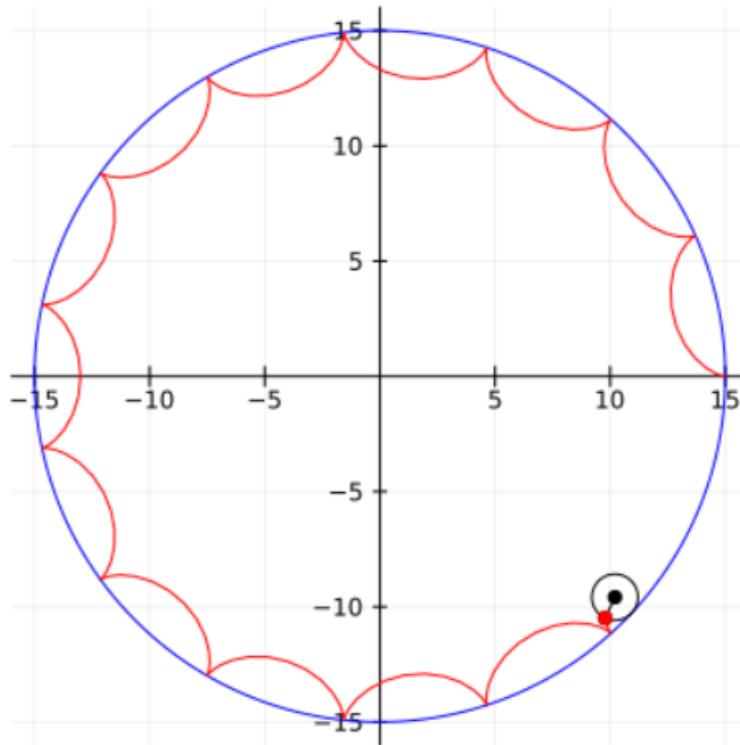


Рис. 73: Задание 10 анимация (целые)

```
r1 = 1
k = 7/3
n = 100
θ = collect(0:2*π/π:20*π+20*π/n)
X = r1*k*cos.(θ)
Y = r1*k*sin.(θ)
anim = @animate for i in 1:n
    # задаём оси координат:
    plt=plot(5,xlim=(-k-1,k+1), ylim=(-k-1,k+1), c=:red, aspect_ratio=1, legend=false, framestyle=:origin)
    # большая окружность:
    plot!(plt, X,Y, c=:blue, legend=false)
    t = θ[1:i]
    # цепочки:
    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_3.gif")
```

Рис. 74: Задание 10 листинг (рацио)

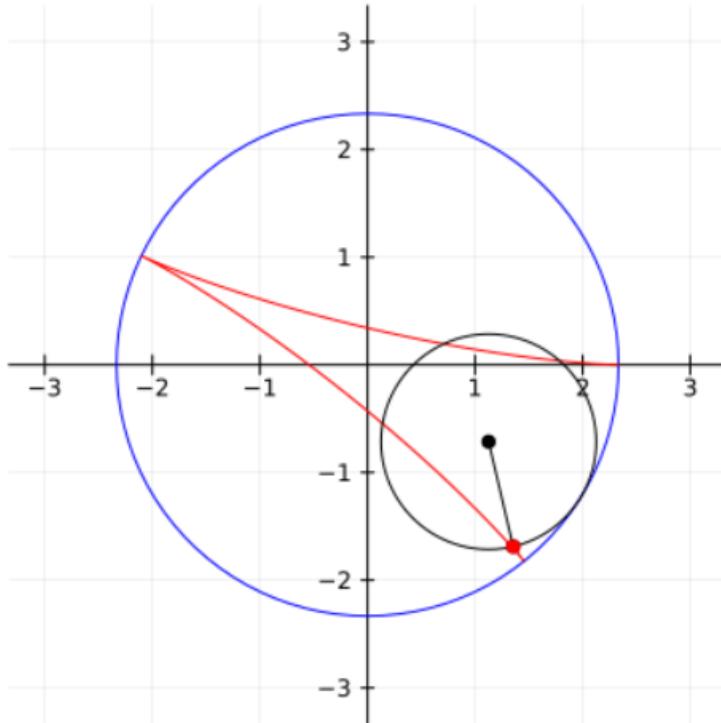


Рис. 75: Задание 10 анимация (рацио)

```

r1 = 1
k = 3/7
n = 100
θ = collect(0:2*π/n:20*π+20*π/n)
X = r1*k*cos.(θ)
Y = r1*k*sin.(θ)
anim = @animate for i in 1:n
    # задаём оси координат:
    plt=plot(5,xlim=(-k-1,k+1),ylim=(-k-1,k+1), 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_4.gif")

```

Рис. 76: Задание 10 листинг (рацио)

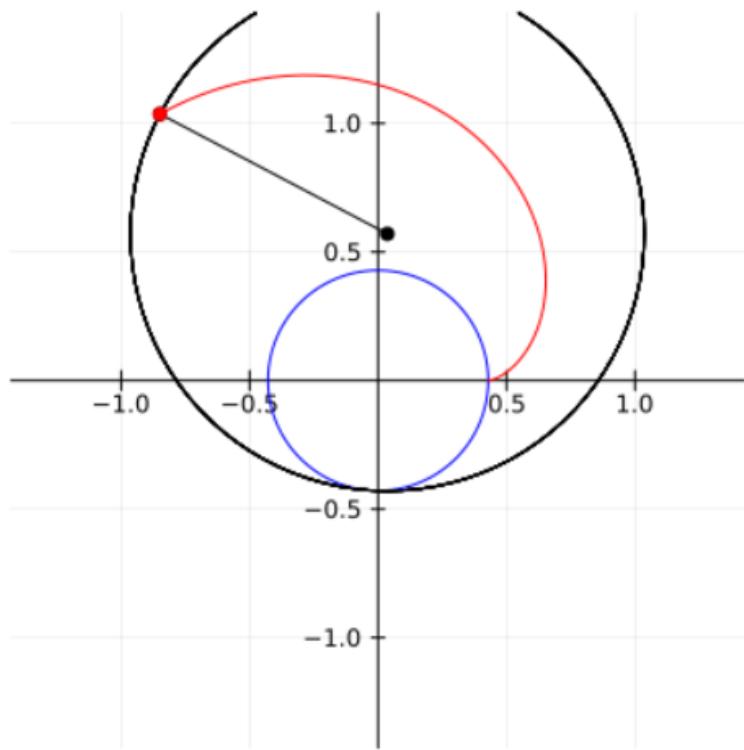


Рис. 77: Задание 10 анимация (рацио)

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(θ:2π/100:2π+2π/100)
# массивы значений координат:
X = rr*k*cos.(θ)
Y = rr*k*sin.(θ)
i = 50
t = θ[1:i]
anim = @animate for i in 1:n
    # задаём оси координат:
    plt=plot(5,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 = θ[1:i]
    # эпюноклона:
    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[end]) .- rr*cos.(θ)
    yc = rr*(k+1)*sin(t[end]) .- rr*sin.(θ)
    plot!(xc,yc,c=:black)
    # радиус малой окружности:
    xl = transpose([rr*(k+1)*cos(t[end]) x[end]])
    yl = transpose([rr*(k+1)*sin(t[end]) y[end]])
    plot!(xl,yl,markershape=:circle,markersize=4,c=:black)
    scatter!([x[end]], [y[end]], c=:red, markerstrokecolor=:red)
end

```

Рис. 78: Задание 11 листинг (целые)

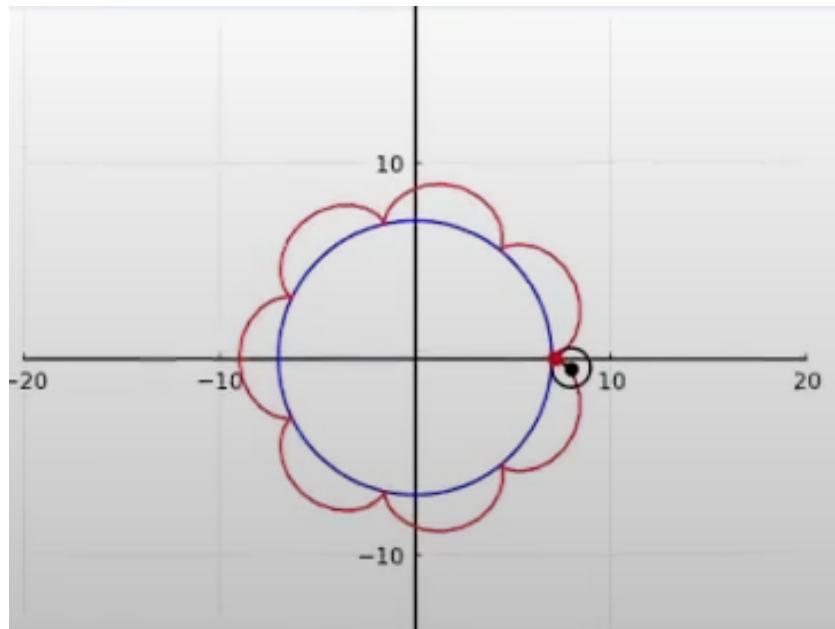


Рис. 79: Задание 11 анимация (целые)

```

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

```

Рис. 80: Задание 11 листинг (целые)

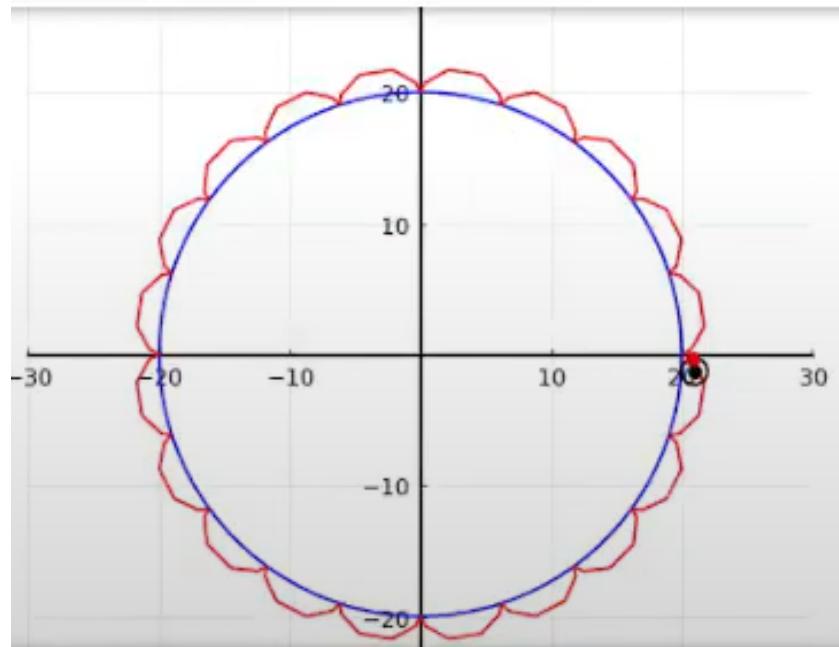


Рис. 81: Задание 11 анимация (целые)

```

rr = 1
# коэффициент для построения большой окружности:
k = 4.44
# число отсчётов:
n = 100
# массив значений угла θ:
# theta from 0 to 2pi ( + a little extra)
θ = collect(0:2*pi/100:2*pi+2*pi/100)
# массивы значений координат:
X = rr*k*cos.(θ)
Y = rr*k*sin.(θ)
i = 50
t = 0:1:i
anim = @animate for i in 1:n
    # задаём оси координат:
    plt=plot(5,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 = θ[1:i]
    # эпюциклионда:
    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[end]) - rr*cos.(θ)
    yc = rr*(k+1)*sin(t[end]) - rr*sin.(θ)
    plot!(xc,yc,c=:black)
    # радиус малой окружности:
    xl = transpose([rr*(k+1)*cos(t[end]) x[end]])
    yl = transpose([rr*(k+1)*sin(t[end]) y[end]])
    plot!((xl,yl,markershape=:circle,markersize=4,c=:red))
    scatter!([(x[end]),(y[end]),c=:red, markerstrokecolor=:red])
end

```

Рис. 82: Задание 11 листинг (рацио)

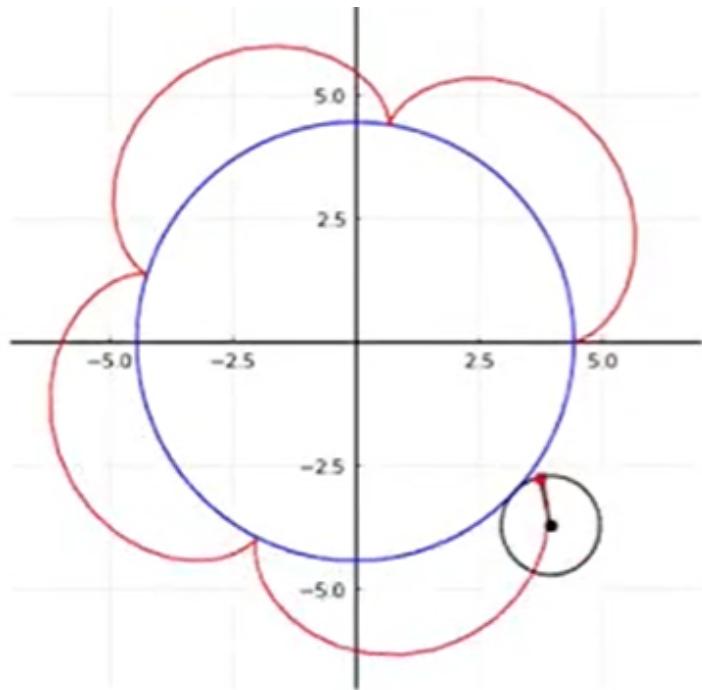


Рис. 83: Задание 11 анимация (рацио)

```

rr = 1
# коэффициент для построения большой окружности:
k = 7
# число отсчётов:
n = 100
# массивы значений угла θ:
θ = collect(θ:2*pi:2*pi+2*pi/100)
# массивы значений координат:
X = rr*k*cos.(θ)
Y = rr*k*sin.(θ)
i = 50
t = θ[1:i]
anim = @animate for i in 1:n
    # задаём оси координат:
    plt=plot(5,xlim=(-10,10),ylim=(-10,10), c=:red, aspect_ratio=1, legend=false, framestyle=:origin)
    # большая окружность:
    plot!(plt, X,Y, c=:blue, legend=false)
    t = θ[1:i]
    # эпциклоиды:
    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[end]) .- rr*cos.(θ)
    yc = rr*(k+1)*sin(t[end]) .- rr*sin.(θ)
    plot!(xc,yc,c=:black)
    # радиус малой окружности:
    xl = transpose([rr*(k+1)*cos(t[end]) x[end]])
    yl = transpose([rr*(k+1)*sin(t[end]) y[end]])
    plot!(xl,yl,markershape=:circle,markersize=4,c=:black)
    scatter!([(x[end]),(y[end])],c=:red, markerstrokecolor=:red)
end

```

Рис. 84: Задание 11 листинг (рацио)

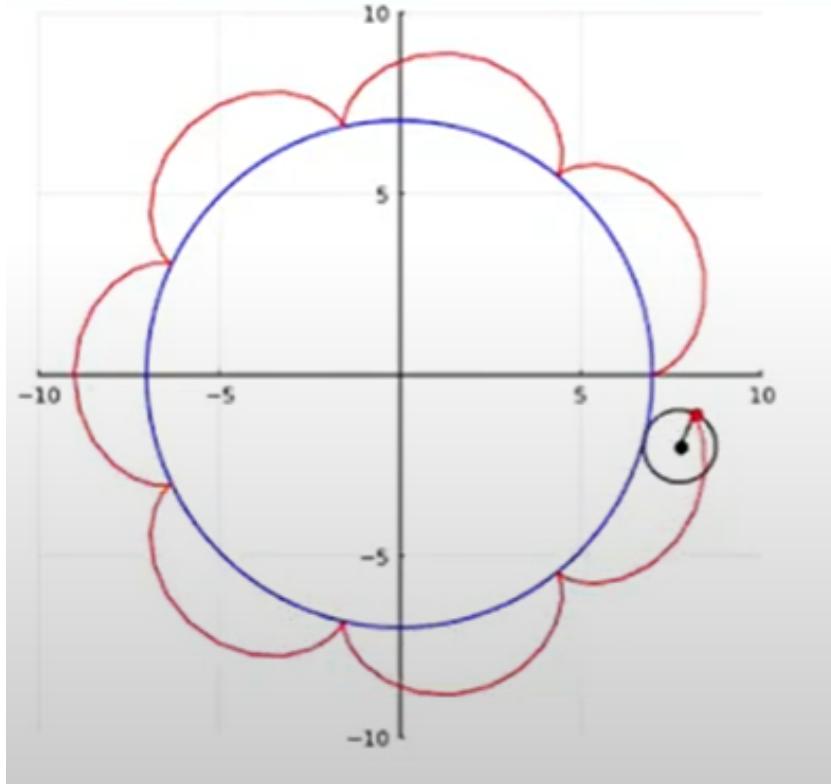


Рис. 85: Задание 11 анимация (рацио)

Выводы по проделанной работе

Вывод

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>