Построение графиков

Гань Чжаолун

13 декабрь, 2024, Москва, Россия

Российский Университет Дружбы Народов

Цели и задачи работы

Цель лабораторной работы

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

Процесс выполнения лабораторной работы

Используя Jupyter Lab, повторите примеры из раздела 5.2

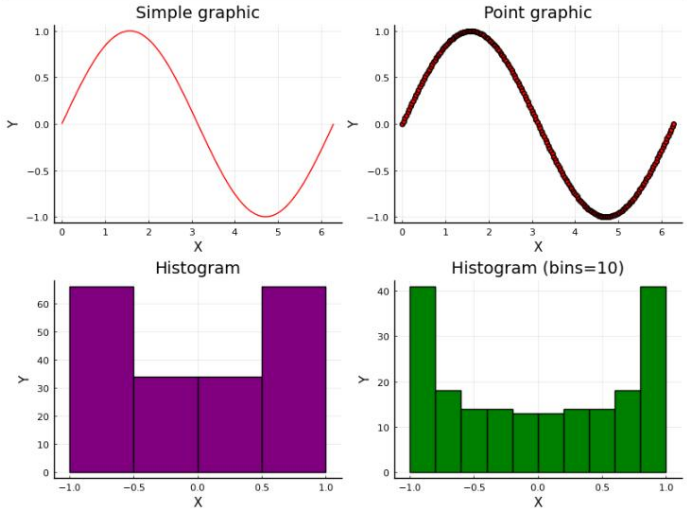
Я повторю все задание 5.2 целиком

Выполните задания для самостоятельной работы

5.4. Задания для самостоятельного выполнения

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

```
f4(x) = sin.(x)
# сгенерируем масссив значений x в диапазоне от 0 до П
x = collect(range(0, 2*pi, length=200))
# зададим функцию
y = f4(x)

fig1 = plot(x, y,
    title="Simple graphic",
    xlabel="X",
    ylabel="Y",
    color="red")
fig2 = scatter(x, y,
    title="Point graphic",
    xlabel="X",
    ylabel="Y",
    color="red")
fig3 = histogram(f4(x),
    title="Histogram",
    xlabel="X",
    ylabel="Y",
    color="purple")
fig4 = histogram(f4(x),
    bins = 10,
    title="Histogram (bins=10)",
    xlabel="X",
    ylabel="Y",
    color="green")
plot(fig1, fig2, fig3, fig4, layout=(2, 2), legends=false, size=(800, 600))
```



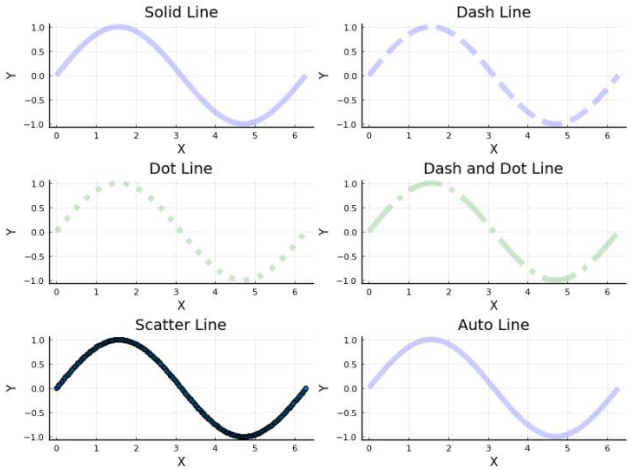
Моя основная идея заключалась в том, чтобы сначала определить набор данных для функции `y = sin(x)` в диапазоне от `0` до `2π`. Затем я последовательно построил несколько типов графиков на основе этих данных: простой линейный график (plot), точечный график (scatter) и две гистограммы (histogram) с разным количеством столбцов (bins). Наконец, я совместил все эти графики в одном общем оконном представлении, используя функцию `plot` с параметром `layout`. Такой подход позволяет наглядно сравнить различные типы визуализации для одних и тех же данных.

Рисунок 1. Код и результат Задания 1

Выполните задания для самостоятельной работы

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

```
f4(x) = sin.(x)
# сгенерируем масссив значений x в диапазоне от 0 до П с шагом 0.1
x = collect(range(0, 2*pi, length=200))
# зададим функцию
y = f4(x)

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



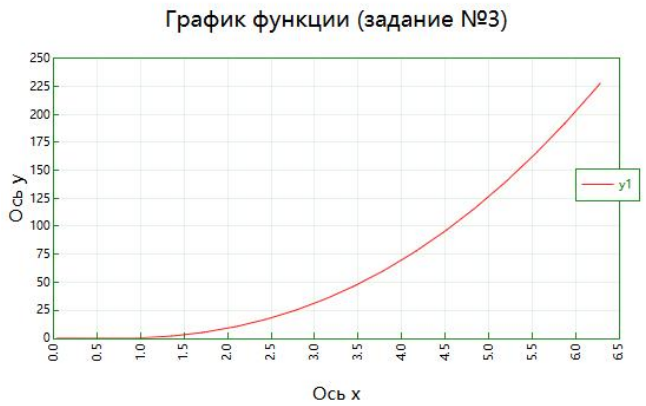
Я сначала определил массив точек для функции `y = sin(x)` в заданном диапазоне `0` до `2π`, а затем построил несколько графиков, меняя типы и стили линий — от сплошной и пунктирной до точечной и комбинированной ("dashdot"). После этого я расположил все полученные графики в одном окне, чтобы можно было наглядно сравнить различные варианты оформления линий.

Рисунок 2. Код и результат Задания 2

Выполните задания для самостоятельной работы

3. Постройте график функции $y(x) = \pi x$^2 ln(x), назовите оси соответственно. Пусть цвет рамки будет зелёным, а цвет самого графика — красным. Задайте расстояние между надписями и осями так, чтобы надписи полностью умещались в графическом окне. Задайте шрифт надписей. Задайте частоту отметок на осях координат.

```
using Plots.PlotMeasures
f5(x) = (pi*x.^2).*log.(x)
# сгенерируем масссив значений x в диапазоне от 0 до 2П
x = collect(range(0, 2*pi, length=200))
# зададим функцию
y = f5(x)
plotly()
plot(x,y, color="red", box = :on, foreground_color="green",
    xaxis = ("Ось x", (0, 6.5), 6.5:-0.5:0), yaxis = ("Ось y", (0, 250), 250:-25:0),
    leg=:right,
    title="График функции (задание №3)",
    titlefont = (15, "montserrat", :black),
    top_margin = 10mm,
    right_margin = 5mm,
    legendfontsize = 8,
    guidefont = (12, "montserrat", :black),
    tickfont = (8, "montserrat",:black),
    xrotation = 90)
```



Я сначала определил функцию и сгенерировал точки по оси в требуемом интервале. Затем построил график, придав ему все необходимые свойства: цвет линии — красный, цвет рамки (границы графической области) — зелёный, соответствующие подписи осей с заданным шрифтом, отступы и частоту отметок на осях. В итоге, все настройки визуализации были применены так, чтобы надписи и оси были аккуратно расположены и удобно читаемы.

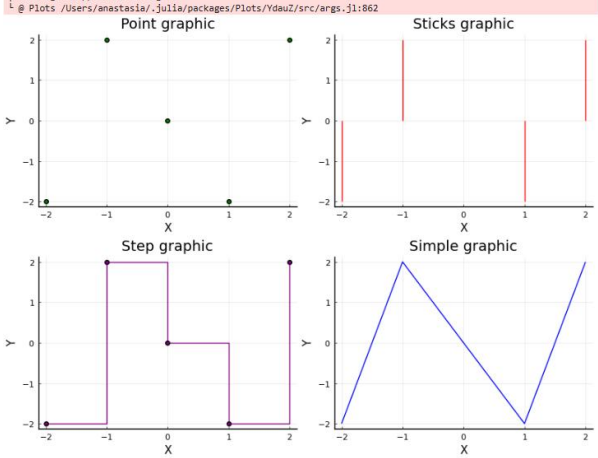
Рисунок 3. Код и результат Задания 3

Выполните задания для самостоятельной работы

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.

```
# зададим функцию
f(x) = x.^3 - 3*x
x = [-2, -1, 0, 1, 2]
y = f(x)
using Plots.PlotMeasures
pyplot()
fig1 = scatter(x, y,
    title="Point graphic",
    xlabel="X",
    ylabel="Y",
    color="green")
fig2 = plot(x, y,
    title="Sticks graphic",
    seriestype = :sticks,
    xlabel="X",
    ylabel="Y",
    color="red")
fig3 = plot(x, y,
    title="Step graphic",
    marker = '.',
    seriestype = :step,
    xlabel="X",
    ylabel="Y",
    color="purple")
fig4 = plot(x, y,
    title="Simple graphic",
    xlabel="X",
    ylabel="Y",
    color="blue")
plot(fig1, fig2, fig3, fig4, layout=(2, 2), legends=false, size=(800, 600))
```

```
┌ Warning: Skipped marker arg ..
└ @ Plots /Users/anastasia/.julia/packages/Plots/YdauZ/src/args.jl:862
```




```
# Сохраню полученное изображение
savefig("figure_solomko.png")
```

Я сначала определил функцию и выбрал набор точек для оценки этой функции. Затем я построил несколько вариантов графиков, используя один и тот же набор данных: в одном подграфике я изобразил точки (scatter), в другом — линии (sticks), в третьем — сочетание линий и точек (step с маркерами), и, наконец, в четвёртом — простую кривую (line). После этого я расположил все четыре вида графических представлений в одном общем окне в виде сетки из четырёх подграфиков. Такой подход позволил мне наглядно сравнить различные способы визуализации одних и тех же данных.

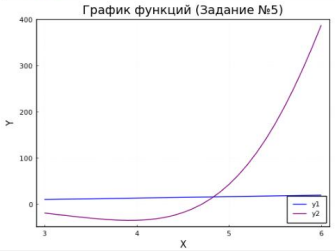
Рисунок 4. Код и результат Задания 4

Выполните задания для самостоятельной работы

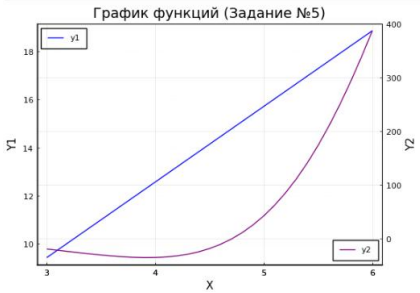
5. Задайте вектор x = (3, 3.1, 3.2, ... , 6). Постройте графики функций $y1(x) = \pi x$ и $y2(x) = \exp(x)\cos(x)$ в указанном диапазоне значений аргумента x следующим образом:

- постройте оба графика разного цвета на одном рисунке, добавьте легенду и сетку для каждого графика; укажите недостатки у данного построения;
- постройте аналогичный график с двумя осями ординат.

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

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

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




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

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



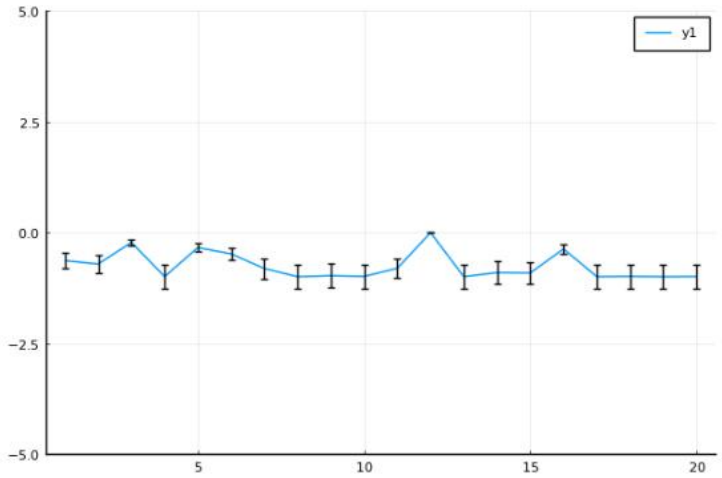
Мой основной подход заключался в том, чтобы сначала определить функции , а затем построить их на одном наборе точек . Я создал два графика. В первом варианте оба графика были отображены на одной координатной сетке с общей осью ординат, использовал разный цвет и легенду для каждой функции, а также включил сетку. Во втором варианте я воспользовался двумя осями ординат: одну оставил для (y_1), другую добавил с помощью `twinx()` для (y_2). Таким образом, каждый график имел свою вертикальную шкалу, что позволило более наглядно сравнивать их.

Рисунок 5. Код и результат Задания 5

Выполните задания для самостоятельной работы

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

```
f_3(x) = x.^2 - 2*x
x = rand(20)
y_3 = f_3(x)
n = 20

error = 1.23 * y_3 / sqrt(n)

plot(y_3, ylims=(-5, 5), err = error)
```



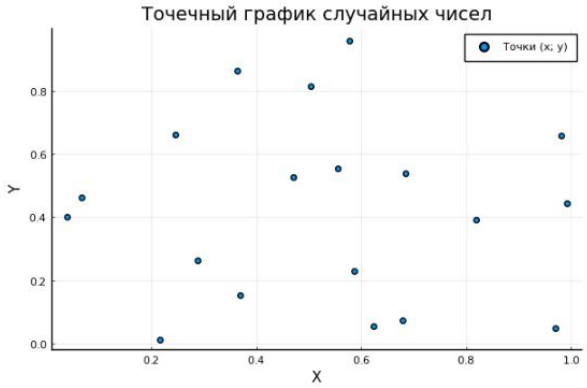
Я решил сгенерировать набор экспериментальных данных, основываясь на простой функции , и использовать случайные значения (x) для придания имитации неопределённости. Затем я вычислил значения (y_3) и задал ошибку измерения как некоторую пропорциональную величину от значения функции, делённую на корень из количества точек. В конце я построил график с использованием ошибок, чтобы визуализировать разброс данных и показать, как может выглядеть экспериментальная неопределённость измерений.

Рисунок 6. Код и результат Задания 6

Выполните задания для самостоятельной работы

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

```
x = rand(20)
y = rand(20)

scatter(x, y,
    title = "Точечный график случайных чисел",
    label = "Точки (x; y)",
    leg = :topright,
    xlabel="X",
    ylabel="Y")
```



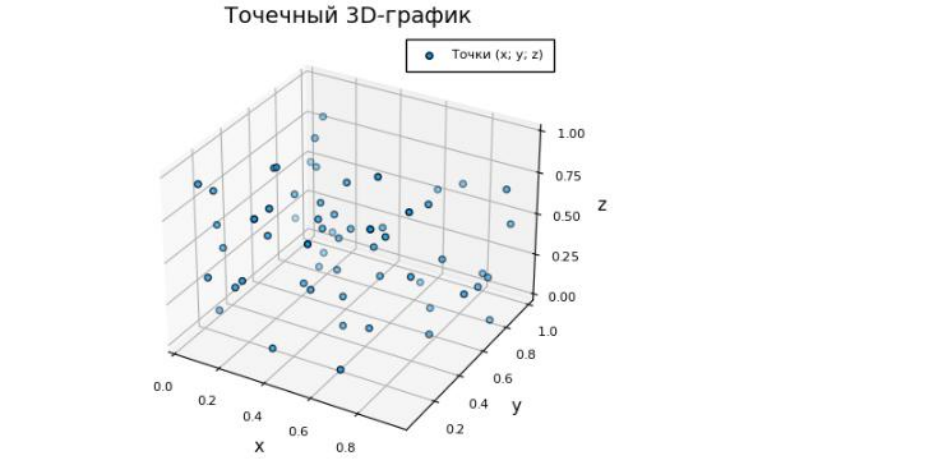
Я сгенерировал два набора случайных чисел для осей (x) и (y), затем построил точечный график, отобразив каждую пару (x_i, y_i) в виде отдельной точки. Добавил подписи к осям, легенду и заголовок, чтобы график был понятен и информативен. Такой подход даёт быстрое визуальное представление о распределении случайных данных.

Рисунок 7. Код и результат Задания 7

Выполните задания для самостоятельной работы

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

```
x = rand(60)
y = rand(60)
z = rand(60)

scatter(x, y, z,
    xlabel = "x",
    ylabel = "y",
    zlabel = "z",
    label = "Точки (x; y; z)",
    title = "Точечный 3D-график")
```



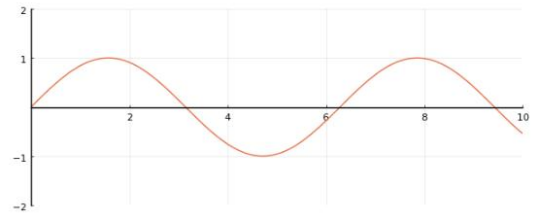
Я сгенерировал три набора случайных чисел для координат (x), (y) и (z), а затем построил трёхмерный точечный график. Для удобства интерпретации я добавил подписи к осям, легенду и заголовок графика. Таким образом, я получил наглядное 3D-представление случайных данных, показывающее их распределение в пространстве.

Рисунок 8. Код и результат Задания 8

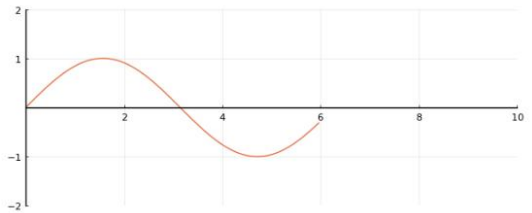
Выполните задания для самостоятельной работы

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

```
n = 300
x = collect(0*pi : 2*pi/100 : 10*pi+pi/100)

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



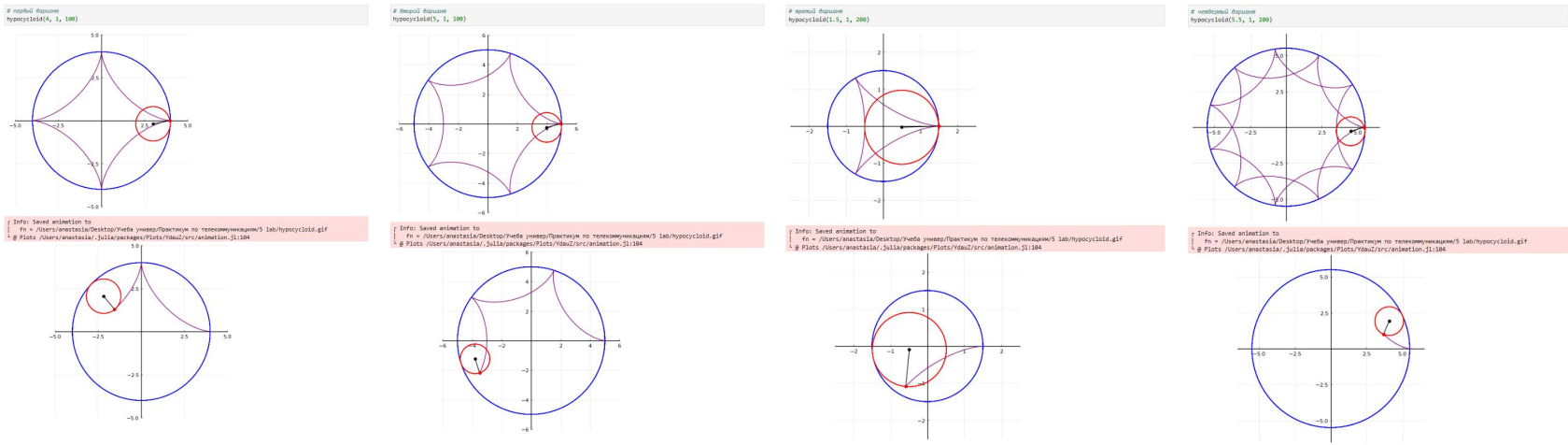

```
┌ Info: Saved animation to
│   fn = /Users/anastasia/Desktop/Учеба универ/Практикум по телекоммуникациям/5 lab/sinusoida.gif
└ @ Plots /Users/anastasia/.julia/packages/Plots/YdauZ/src/animation.jl:104
```



Я захотел создать анимацию, показывающую, как синусоида «нарастает» с увеличением аргумента. Для этого я сначала сформировал вектор значений (x), затем пошагово — от кадра к кадру — добавлял всё больше точек на график, вычисляя для них значение sin(x). С помощью цикла и макроса `@animate` я генерировал последовательность кадров, а затем превратил её в GIF-анимацию. Таким образом, по мере продвижения по циклу синусоида постепенно удлиняется, наглядно демонстрируя процесс построения кривой.

Рисунок 9. Код и результат Задания 9

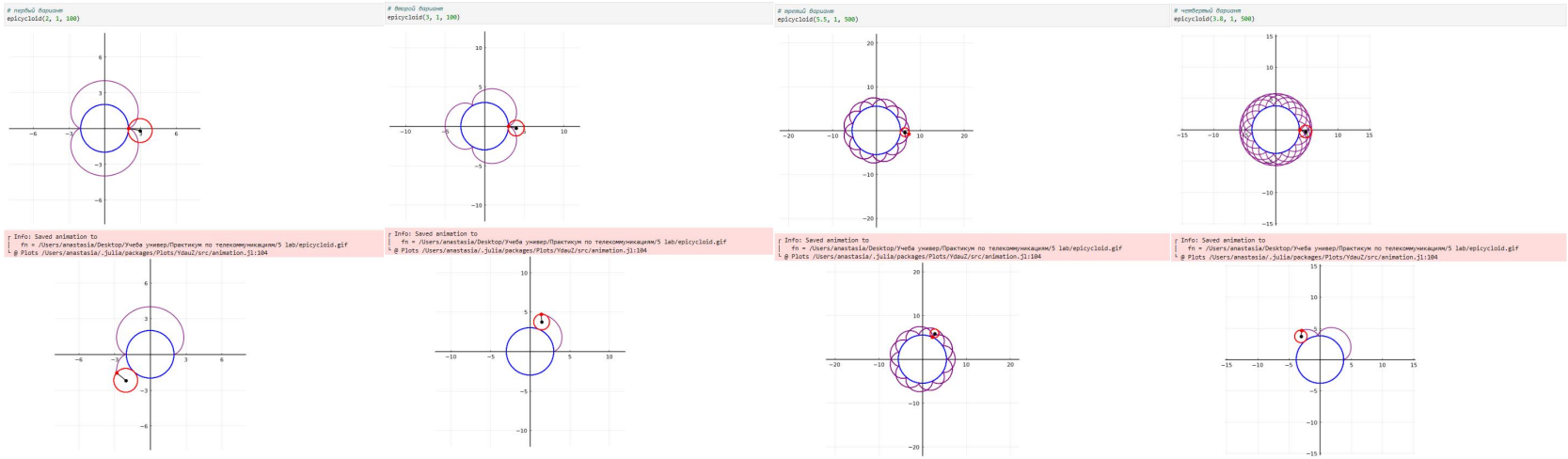
Выполните задания для самостоятельной работы



Я решил визуализировать гипоциклоиду, постепенно «прорисовывая» её траекторию и показывая процесс вращения маленькой окружности по внутренней стороне большой. Для этого я создал функцию, которая на заданном интервале угла строит координаты большой окружности, малой окружности и самой гипоциклоиды. Затем я использовал цикл анимации (`@animate`), который по шагам добавляет новые точки на траектории, позволяя увидеть, как форма развивается во времени. Поменяв параметр (k) (отношение радиусов окружностей), я получил разные варианты гипоциклоид. В итоге, каждая анимация сохраняется в GIF-файл, наглядно демонстрируя процесс построения фигуры для разных значений (k).

Рисунок 10. Код и результат Задания 10

Выполните задания для самостоятельной работы



Я захотел наглядно показать построение эпициклоиды при различных значениях параметра (k). Для этого я написал функцию, которая для заданных (k), радиуса (r_0) и количества шагов (n) генерирует точки большой и малой окружностей, а также координаты эпициклоиды на каждом шаге, постепенно «прорисовывая» её траекторию. В цикле анимации на каждом кадре я добавляю всё больше точек, показывая, как фигура формируется во времени. Изменяя значение (k), я получил различные характерные формы эпициклоиды. После завершения построения всех кадров я сохранил анимацию в виде GIF-файла.

Рисунок 11. Код и результат Задания 11

Выводы по проделанной работе

Вывод

Я освоил синтаксис языка Julia для построения графиков.