Отчёт по лабораторной работе №5 по дисциплине Компьютерный практикум по статистическому анализу данных

Построение графиков

Шаповалова Диана Дмитриевна

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

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

2 Выполнение работы

2.1 Основные пакеты для работы с графиками в Julia

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.

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”)

using Plots

Стандартным образом можно задать различные опции при построении графика. В результате получим график исходной функции

```
[7]: # 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")
```

[7]:

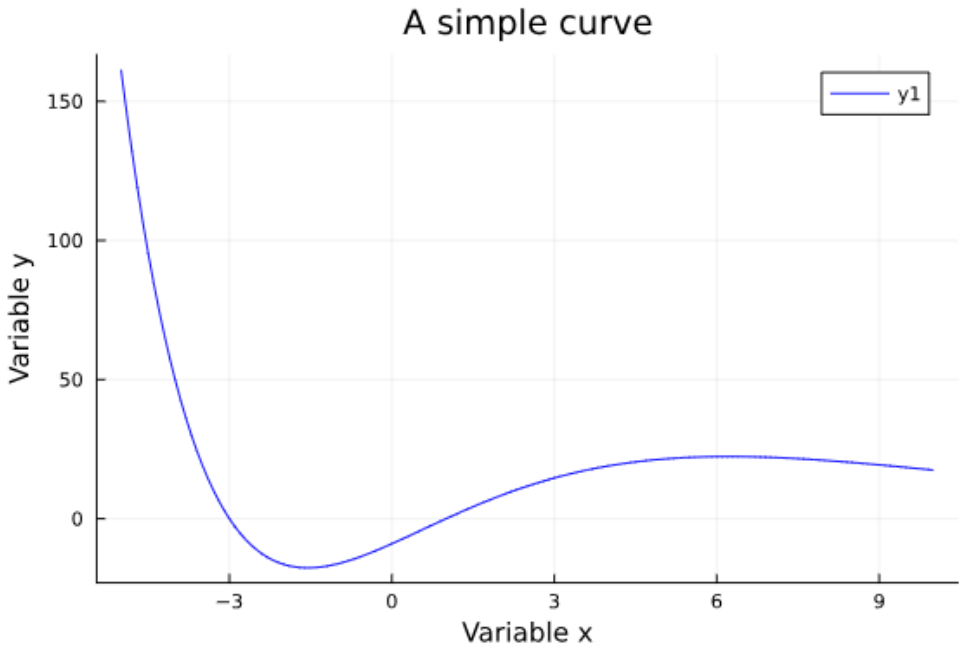


Рис. 2.1: Простая кривая

2.2 Опции при построении графика

Задаём функцию

Строим график функции (рис. 5.3)

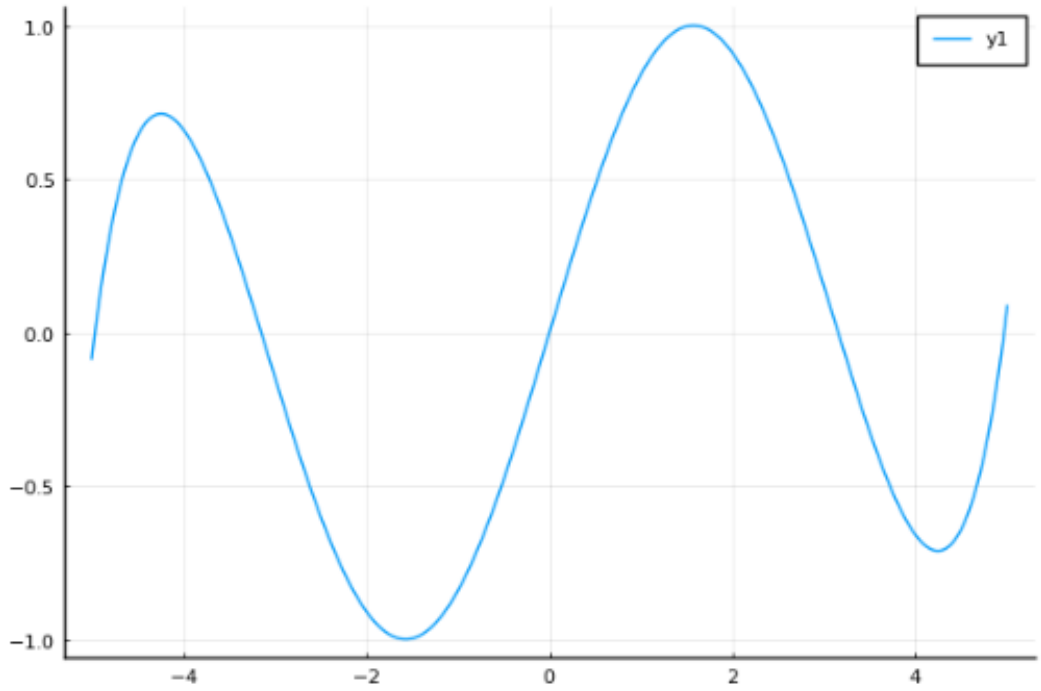
Задаём разложение исходной функции в ряд Тейлора

Строим график разложения исходной функции в ряд Тейлора

Выводим две функции на один график (рис. 5.5):

[16]:
```
# построение графика функции sin_taylor(x):
plot(sin_taylor)
```

[16]:



[17]:
```
# построение двух функций на одном графике:
plot(sin_theor)
plot!(sin_taylor)
```

[17]:

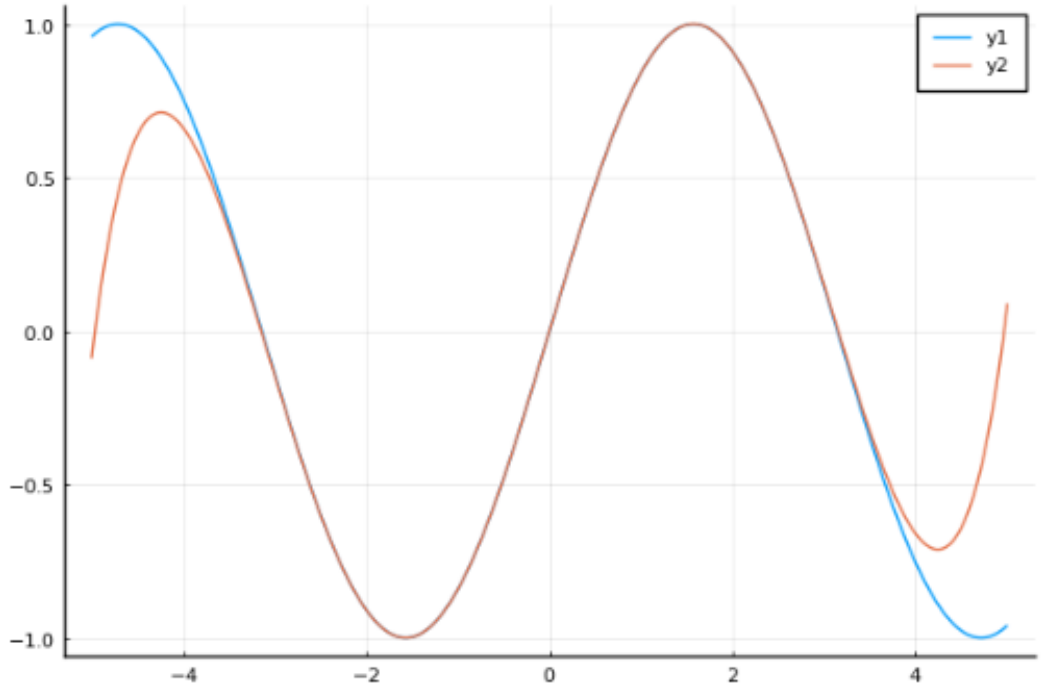
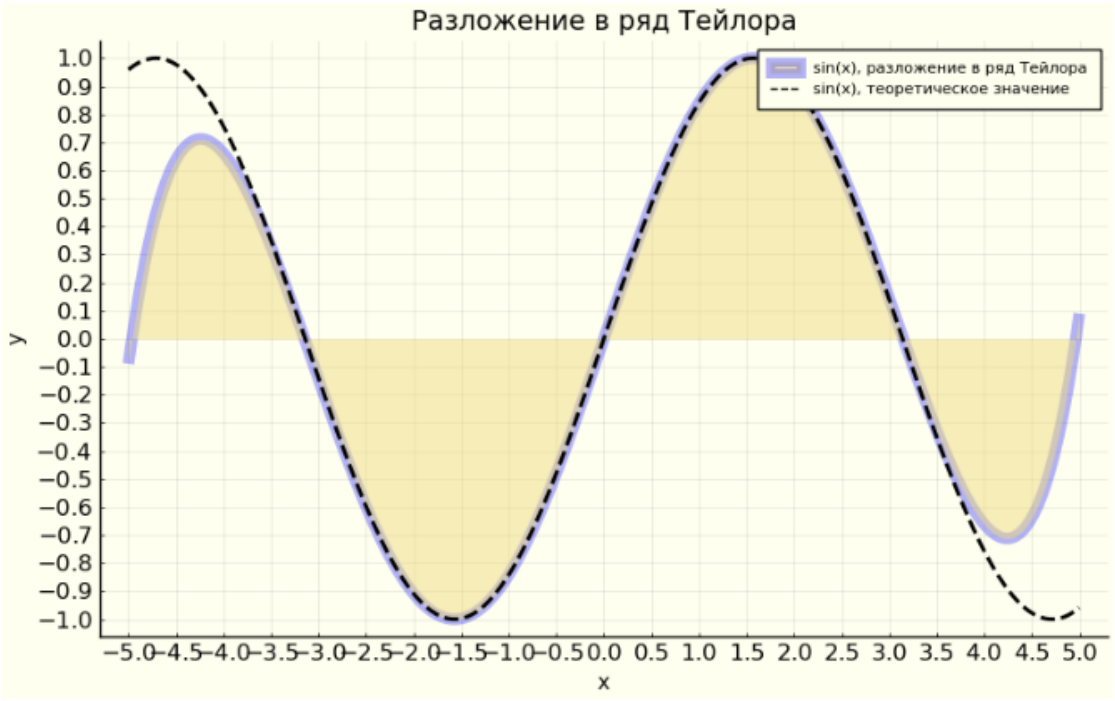


Рис. 2.2: Опции при построении графика

Далее добавим различные опции для отображения на графике

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

1]:



2]:
```
# сохранение графика в файле в формате pdf или png:
savefig("taylor.pdf")
savefig("taylor.png")
```

2]:
```
"C:\\Users\\dina7\\taylor.png"
```

Рис. 2.3: Различные опции для отображения

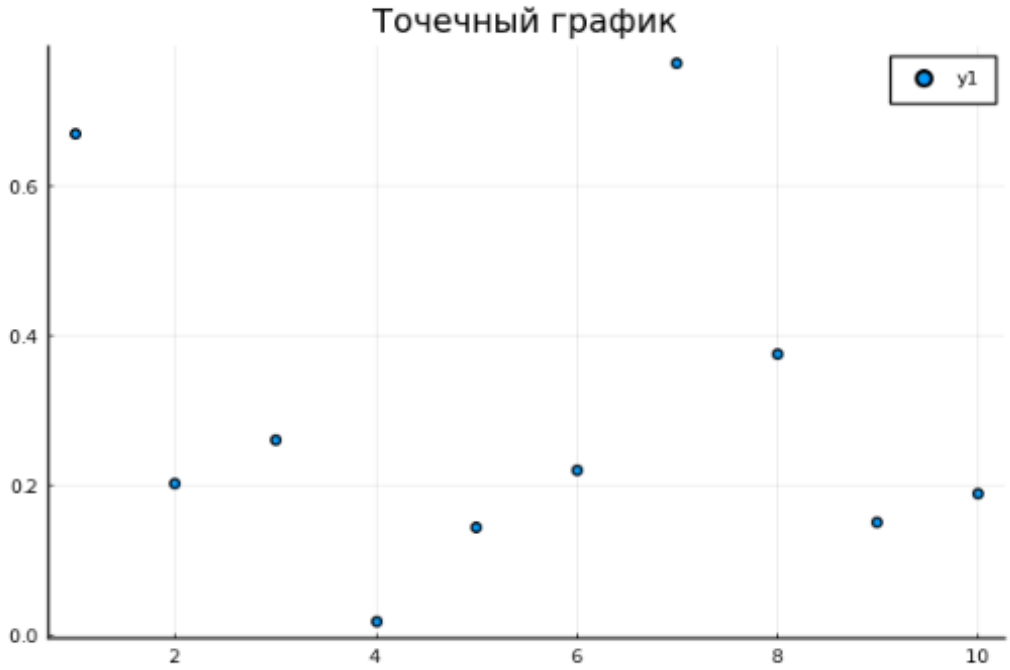
2.3 Точечный график

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

Как и построении обычного графика для точечного графика необходимо задать массив значений , посчитать или задать значения , задать опции построения графика.

Для точечного графика можно задать различные опции, например размер маркера, его тип, цвет и и т.п.

[24]:




```
sys:1: UserWarning: No data for colormapping provided via 'c'. Parameters 'vmin', 'vmax' will
sys:1: UserWarning: No data for colormapping provided via 'c'. Parameters 'vmin', 'vmax' will
sys:1: UserWarning: No data for colormapping provided via 'c'. Parameters 'vmin', 'vmax' will
sys:1: UserWarning: No data for colormapping provided via 'c'. Parameters 'vmin', 'vmax' will
```

[25]:

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

[25]:

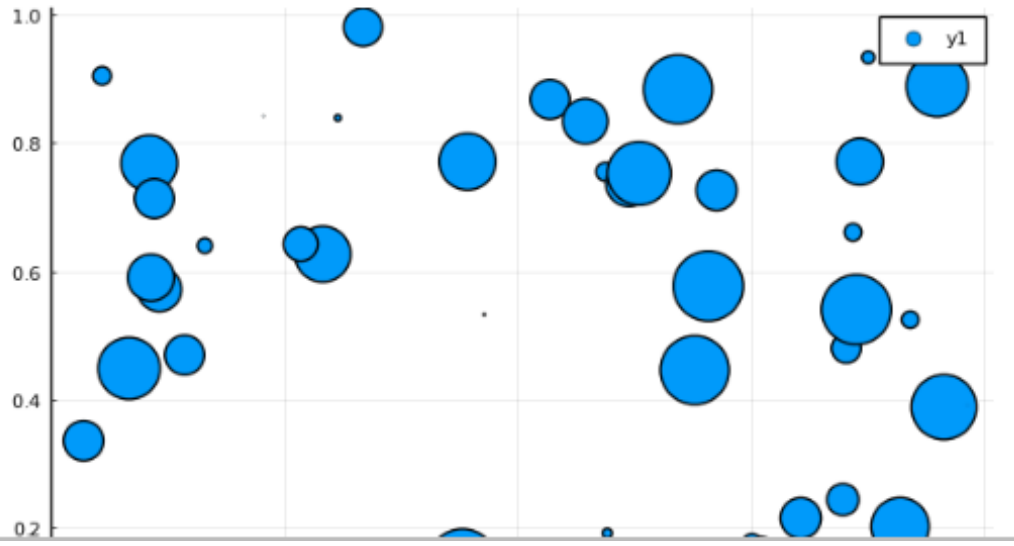


Рис. 2.4: Точечный график

Можно строить и 3-мерные точечные графики

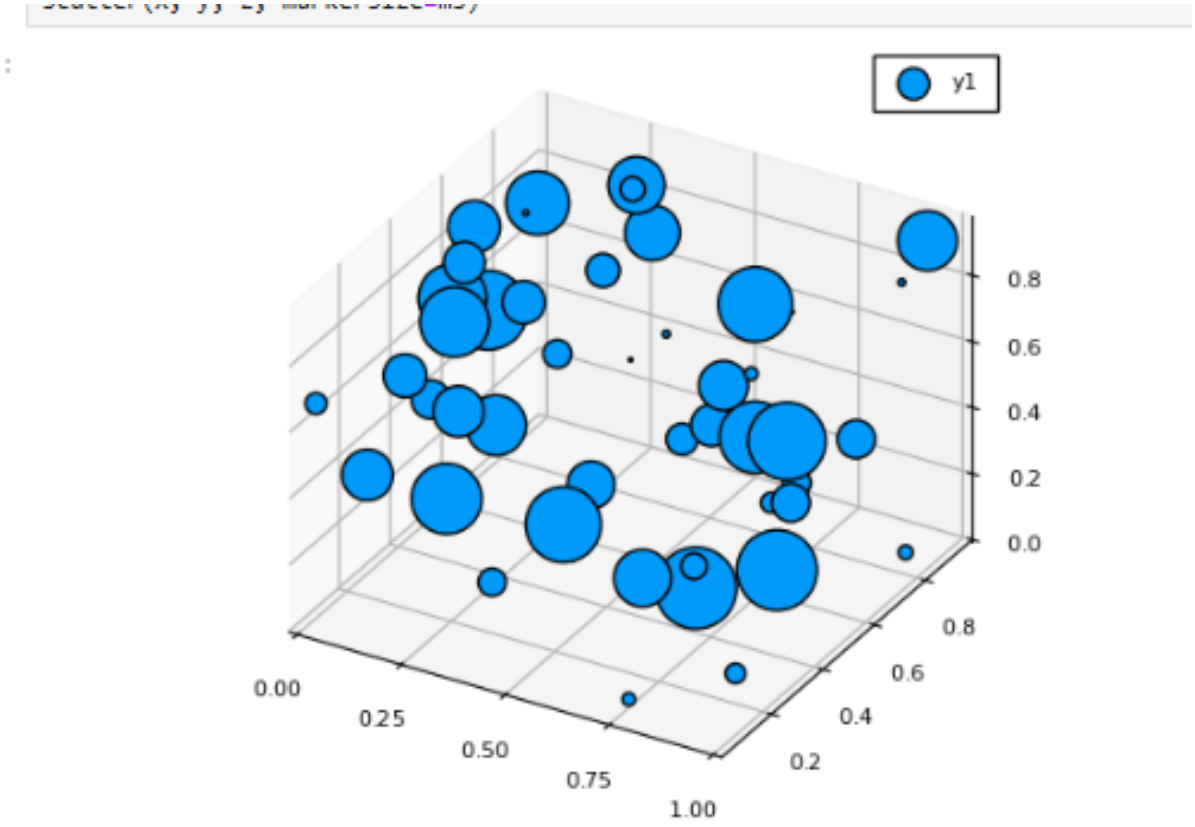


Рис. 2.5: Точечный график

2.4 Аппроксимация данных

Для демонстрации зададим искусственно некоторую функцию, в данном случае похожую по поведению на экспоненту. Аппроксимируем полученную функцию полиномом 5-й степени.

```
4]: let
        # определение массива для нахождения коэффициентов полинома:
        A = [ones(1000) x x.^2 x.^3 x.^4 x.^5]
        # решение матричного уравнения:
        c = A \ y
        # построение полинома:
        f = c[1] * ones(1000) + c[2] * x + c[3] * x.^2 + c[4] * x.^3 + c[5] * x.^4 + c[6] * x.^5
        # построение графика аппроксимирующей функции:
        plot!(x,f,linewidth=3, color=:red)
    end
```

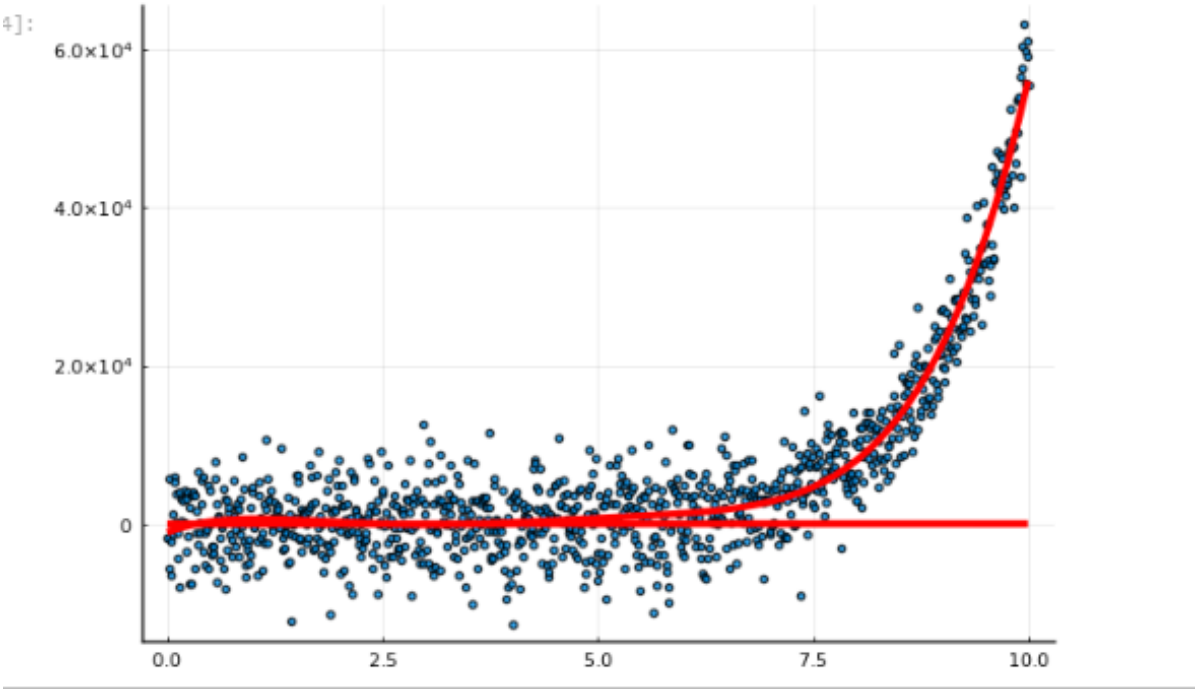


Рис. 2.6: Аппроксимация данных

2.5 Полярные координаты

Приведём пример построения графика функции □(□) = 1 + cos □ sin2 □ в полярных координатах

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

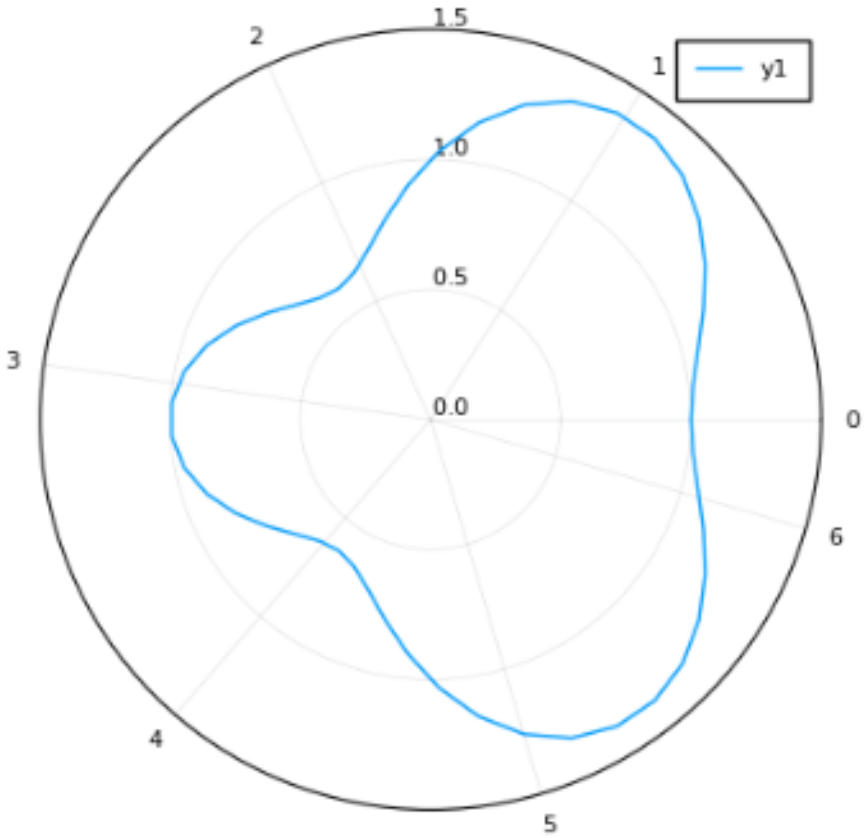


Рис. 2.7: Полярные координаты

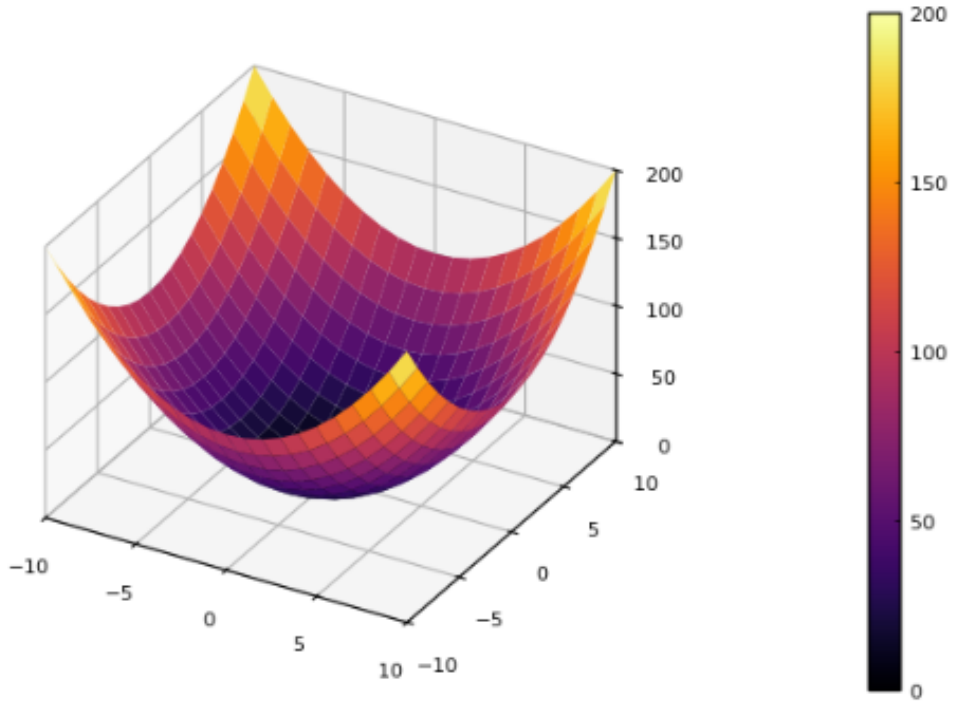
2.6 Параметрический график

Для построения поверхности, заданной уравнением □(□, □) = □^2 + □^2, можно воспользоваться функцией surface(). Также можно воспользоваться функцией

plot() с заданными параметрами. Можно задать параметры сглаживания. Можно задать определённый угол зрения.

```
surface(x, y, f)
```

[43]:



[44]:

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

[44]:

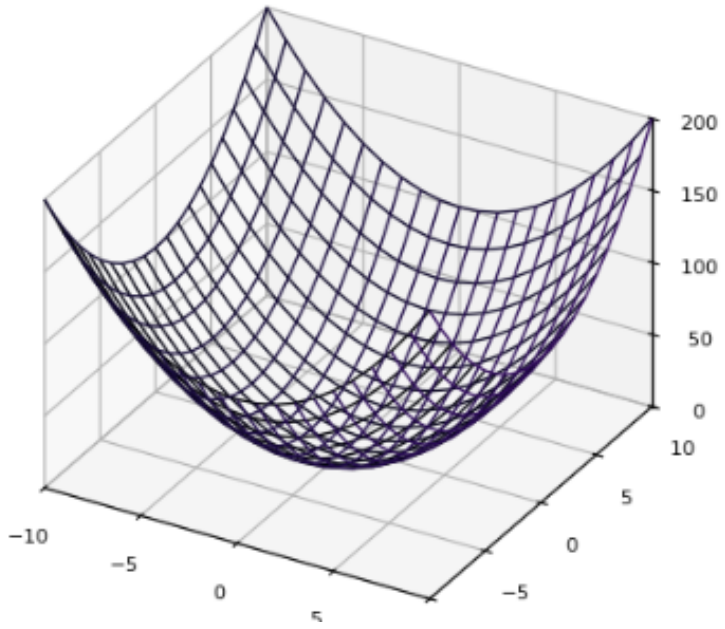
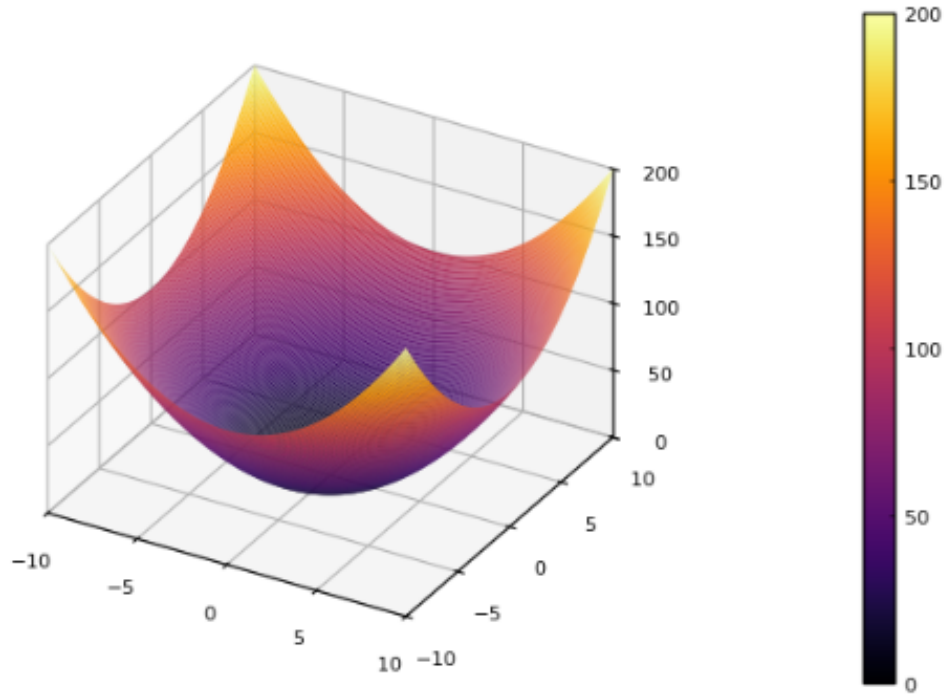


Рис. 2.8: Параметрический график

[46]:



[47]:

```
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),
)
```

[47]:

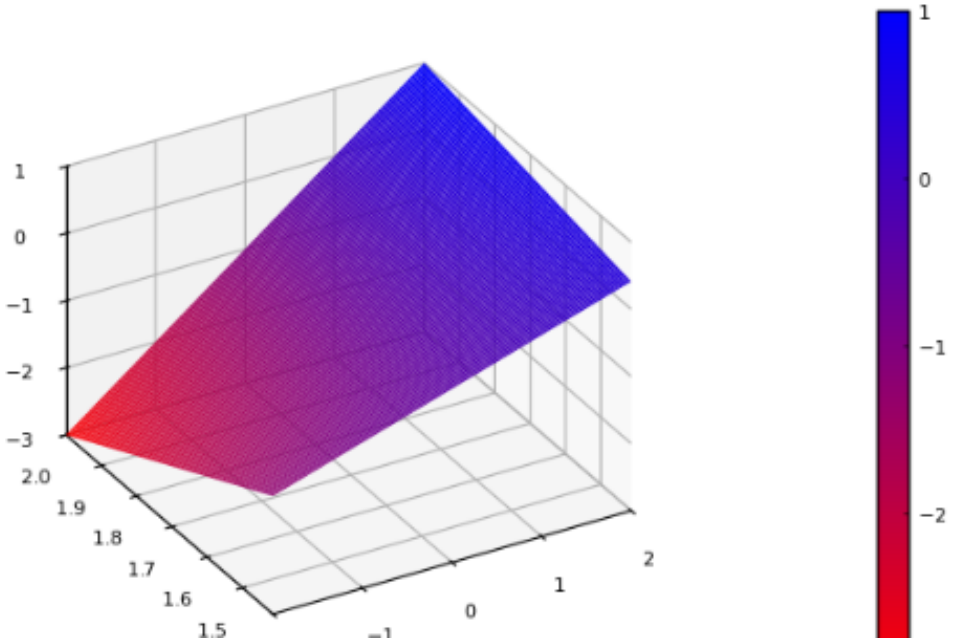
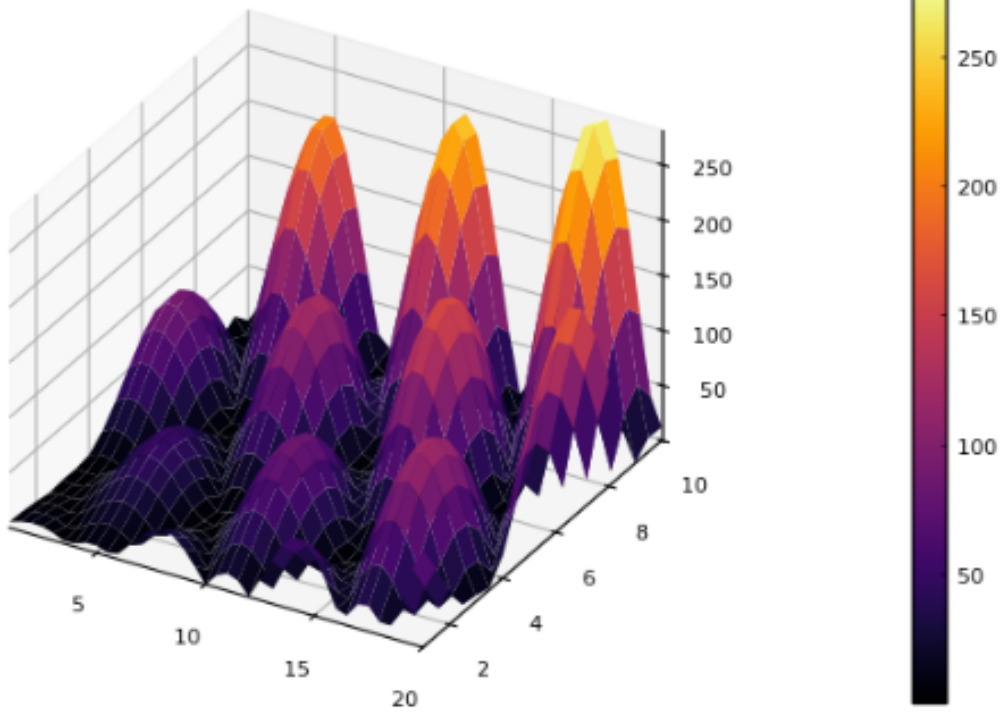


Рис. 2.9: Параметрический график

2.7 Линии 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. Рассмотрим поверхность, заданную функцией. Линии уровня можно построить, используя проекцию значений исходной функции на плоскость. Можно дополнительно добавить заливку цветом

]:



]:

```
p = contour(x, y, g,
fill=true)
plot(p)
```

]:

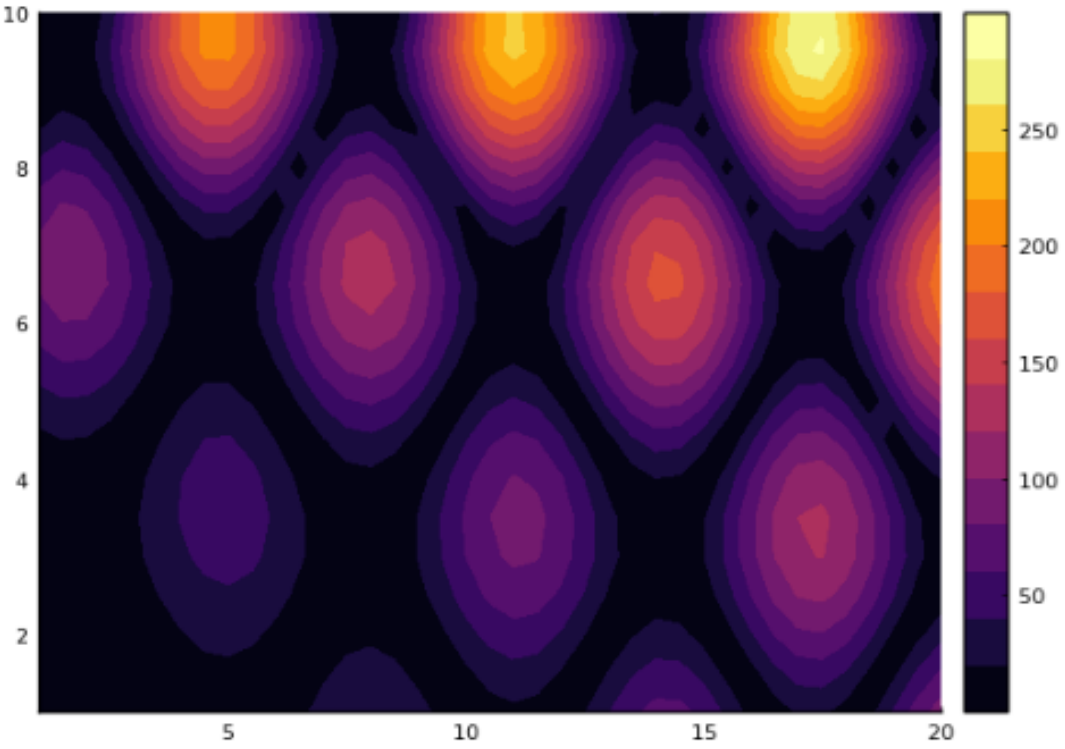
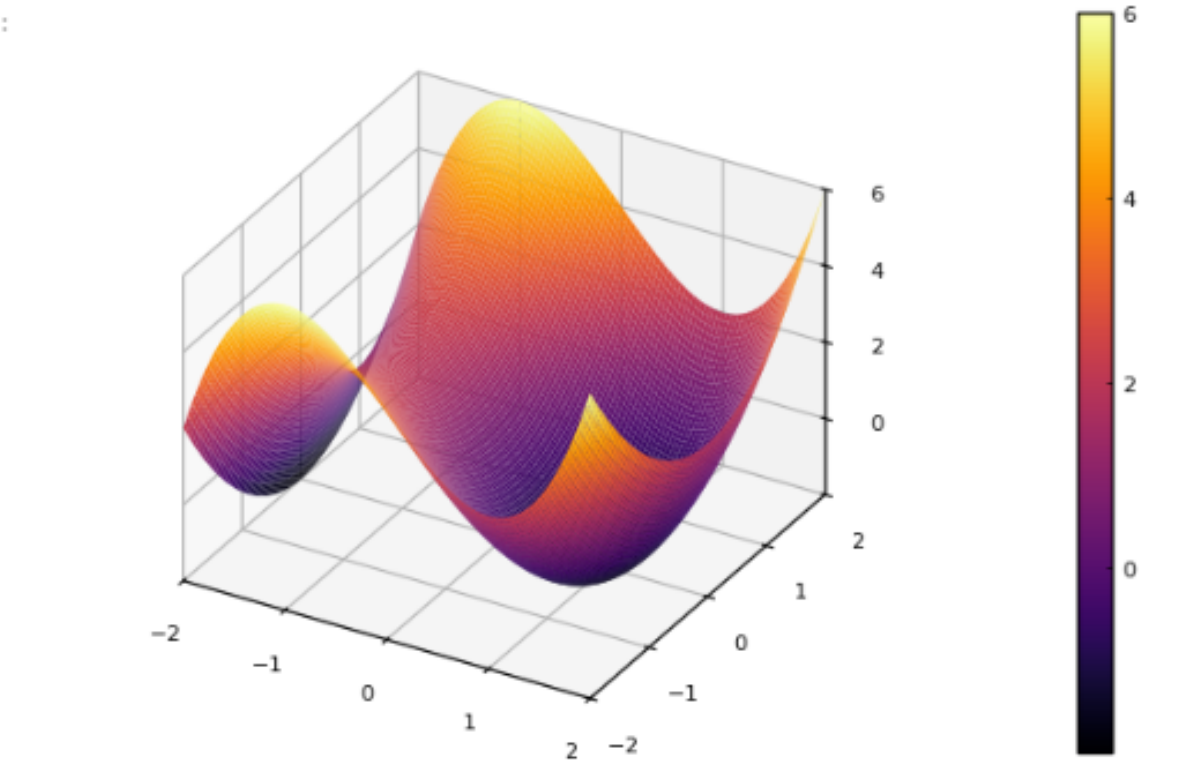


Рис. 2.10: Линии уровня

2.8 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. Для функции сначала построим её график и линии уровня.




```
# построение линий уровня:
contour(X, Y, h)
```

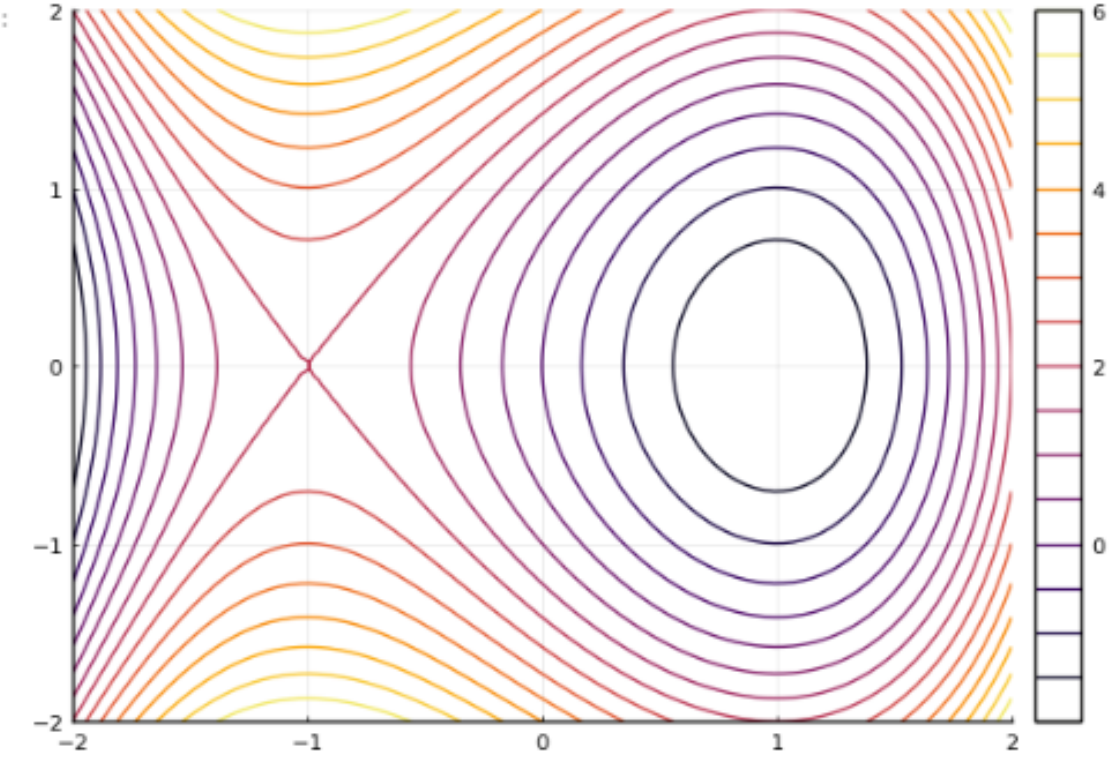


Рис. 2.11: Векторные поля

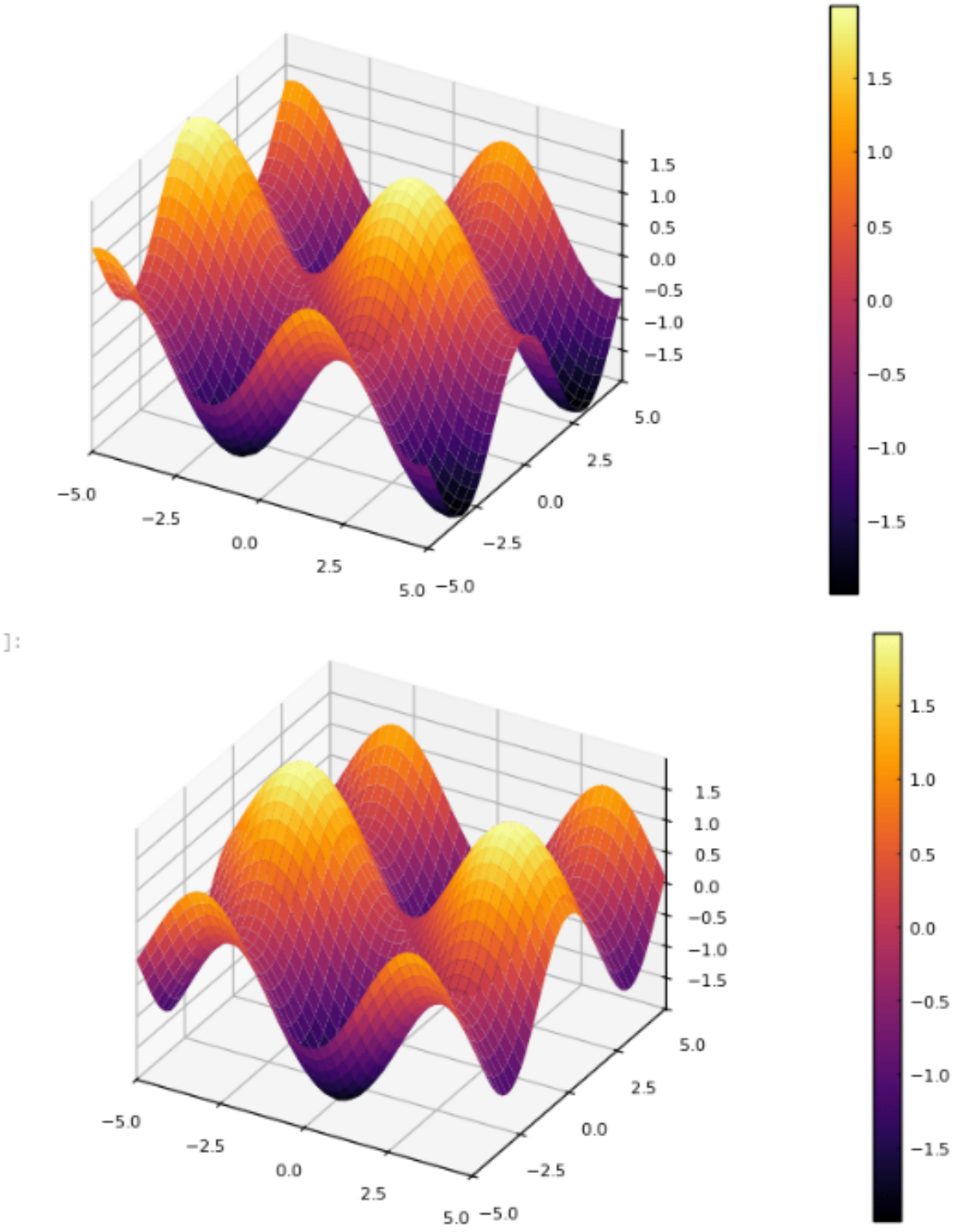
2.9 Анимация

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

Gif-анимация

В Julia рекомендуется использовать gif-анимацию в pyplot()

```
[ Info: Saved animation to C:\Users\dina7\tmp.gif
```



]:

Рис. 2.12: Gif-анимация

2.10 Гипоциклоида

Гипоциклоида — плоская кривая, образуемая точкой окружности, катящейся по внутренней стороне другой окружности без скольжения. Построим гипоциклоиду. Сначала зададим параметры. Затем зададим массивы необходимых значений. Построим оси координат. Построим большую окружность Для частичного построения гипоциклоиды будем менять параметр . Добавляем малую окружность гипоциклоиды/ Добавим радиус для малой окружности.

В конце сделаем анимацию получившегося изображения

```
[ Info: Saved animation to C:\Users\dina7\hypocycloid.gif
```

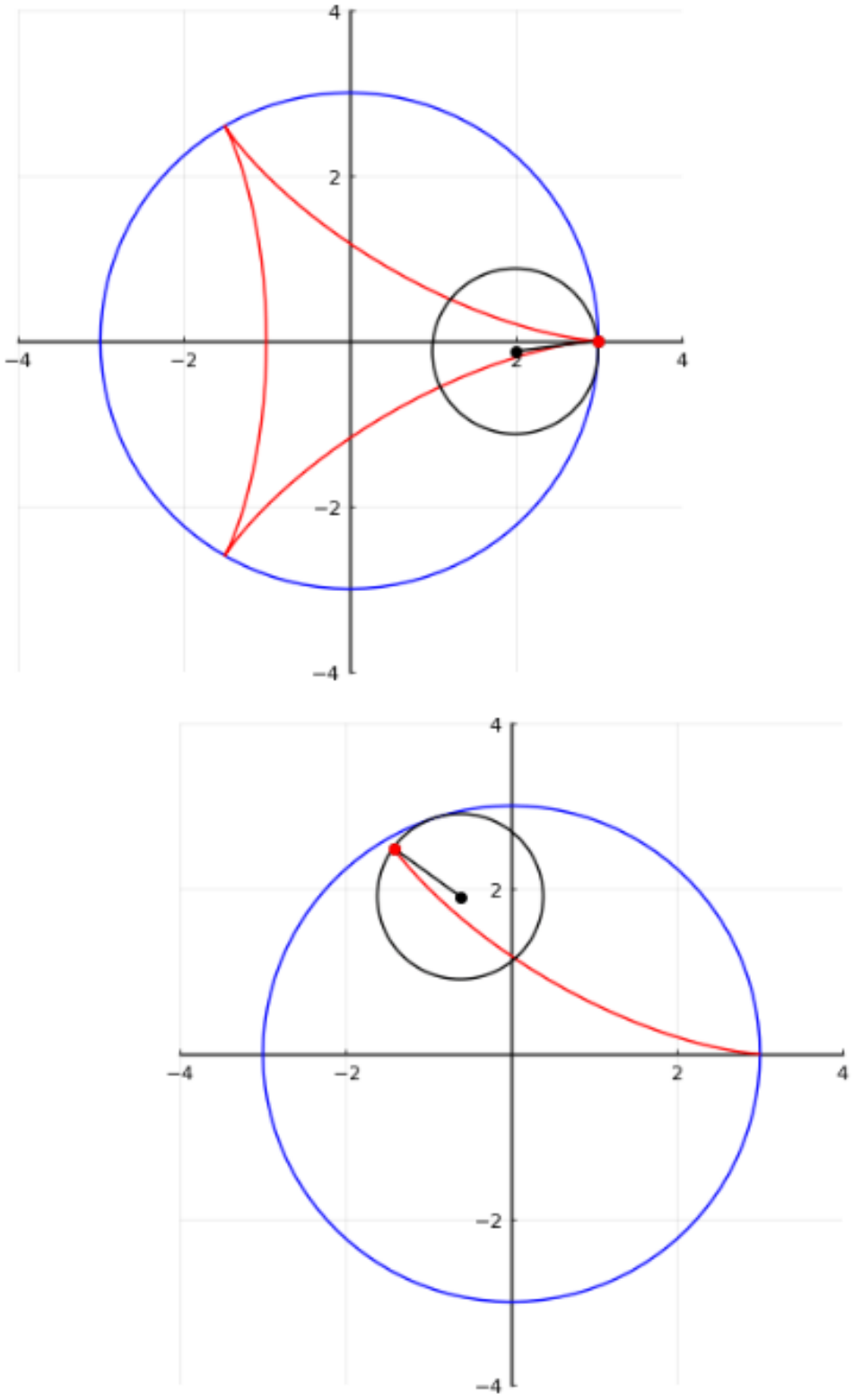
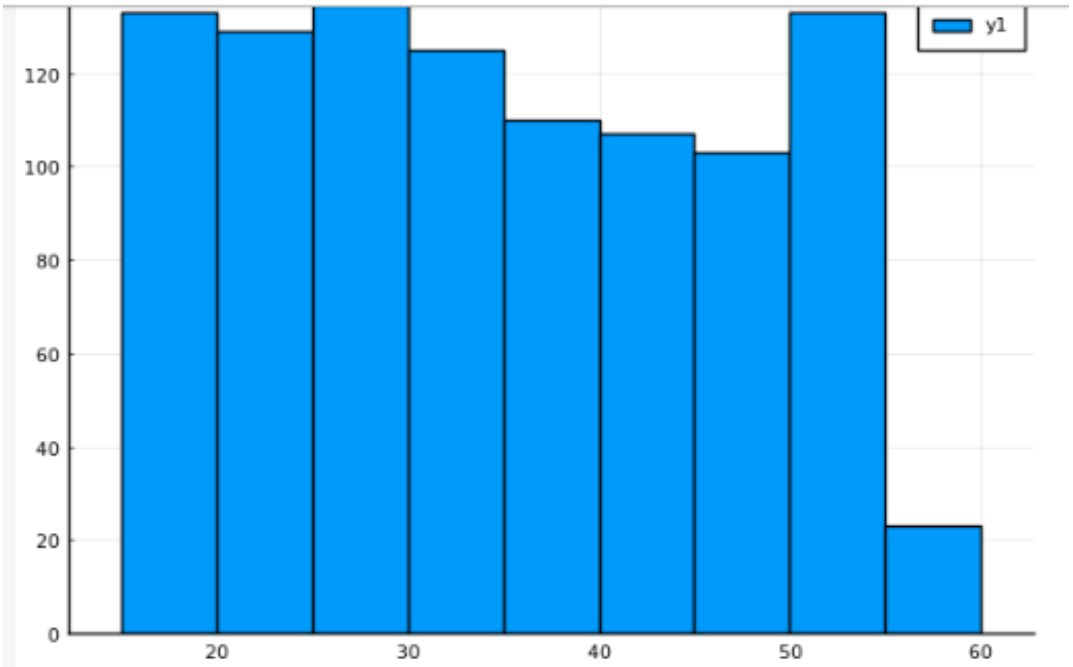


Рис. 2.13: Гипоциклоида

2.11 Использование пакета Distributions

Строим гистограмму. Задаём нормальное распределение и строим гистограмму




```
sys:1: UserWarning: No data for colormapping provided via 'c'. Parameters 'vmin', 'vm
sys:1: UserWarning: No data for colormapping provided via 'c'. Parameters 'vmin', 'vm
sys:1: UserWarning: No data for colormapping provided via 'c'. Parameters 'vmin', 'vm
sys:1: UserWarning: No data for colormapping provided via 'c'. Parameters 'vmin', 'vm
```

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

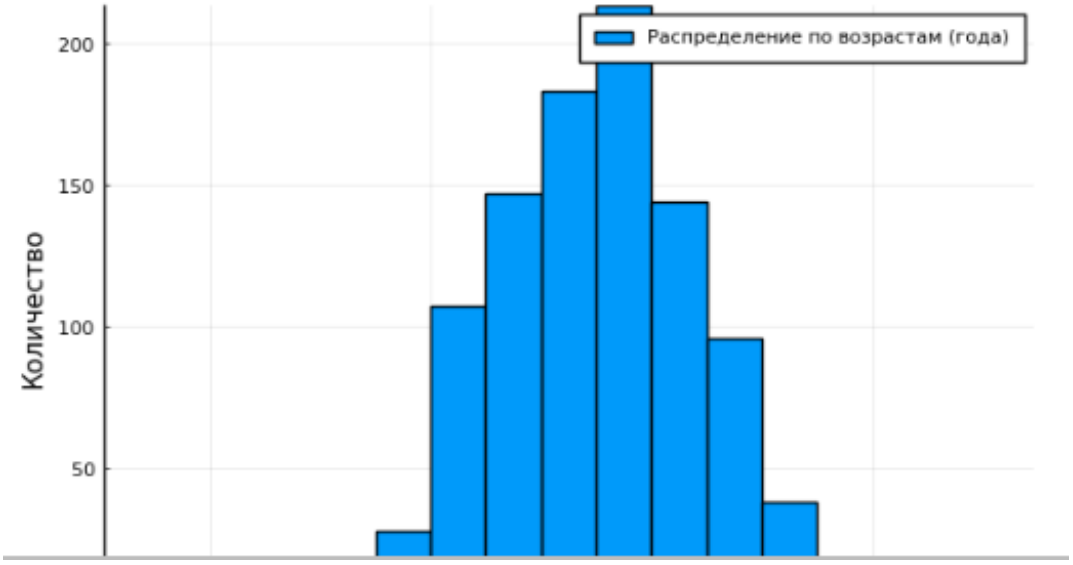


Рис. 2.14: Использование пакета Distributions

2.12 Подграфики

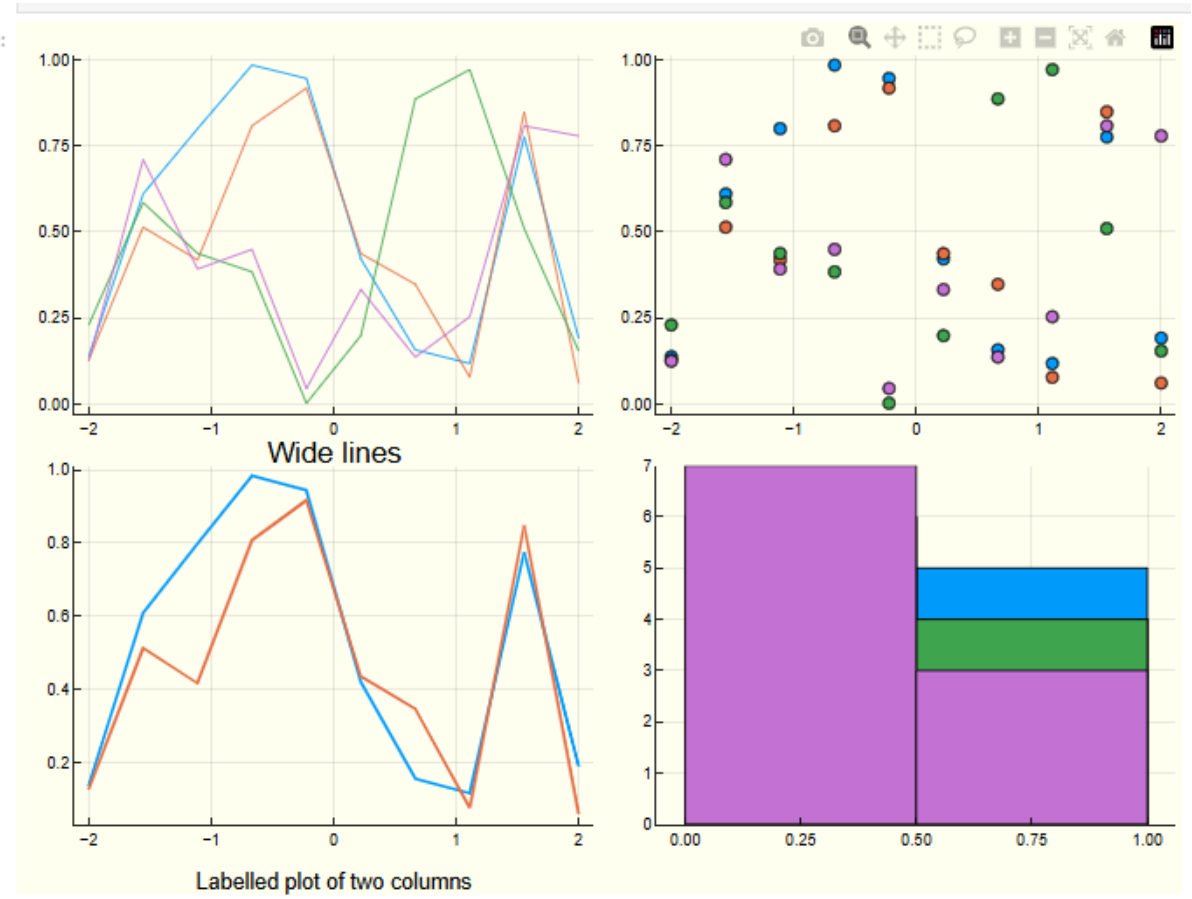


Рис. 2.15: Объединение нескольких графиков в одной сетке

2.13 Задания для самостоятельного выполнения

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

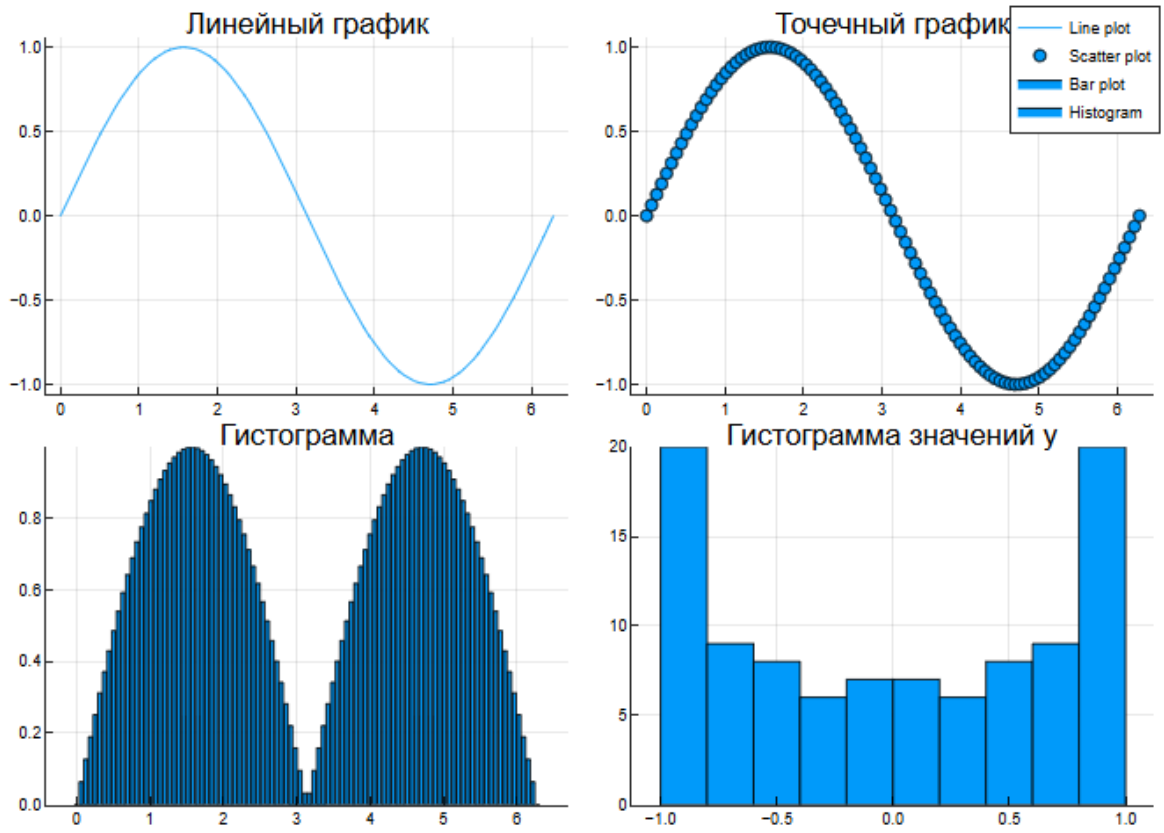


Рис. 2.16: Задание 1

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

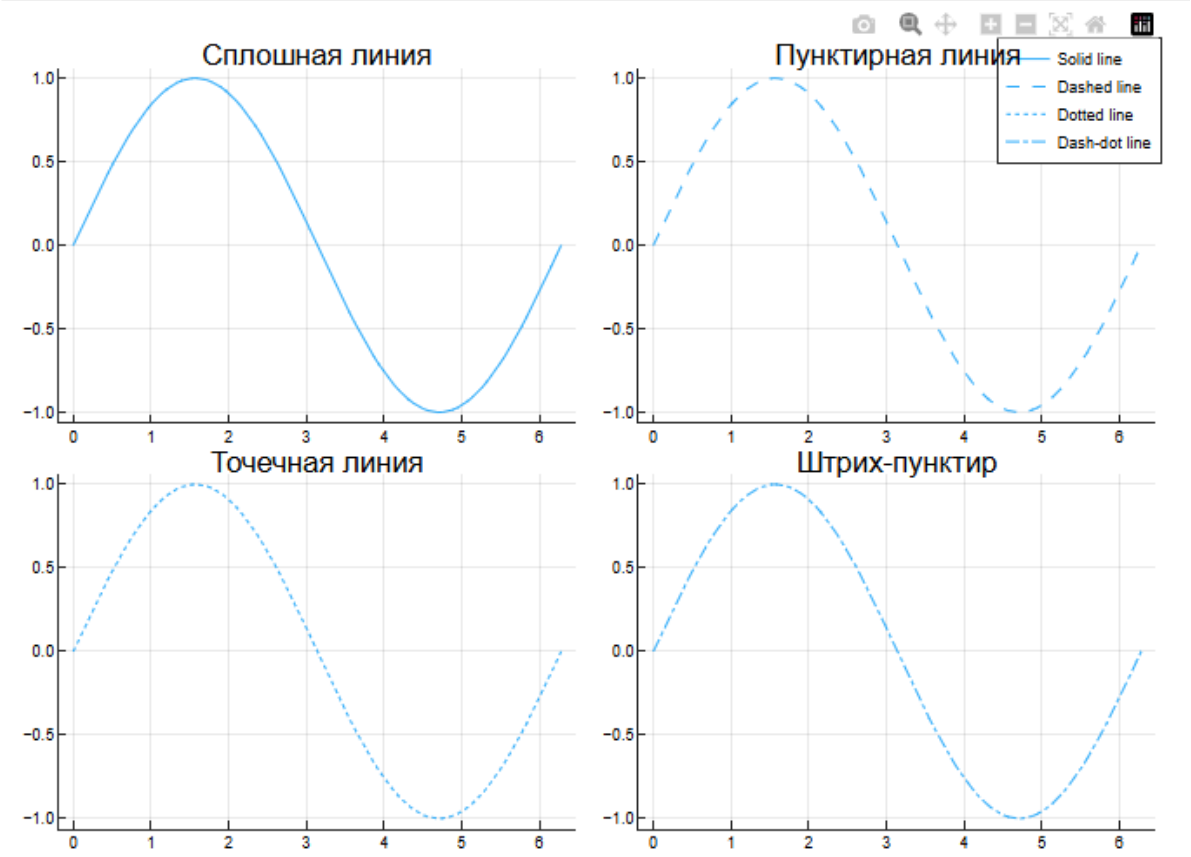


Рис. 2.17: Задание 2

3. Постройте график функции □(□) = □□2 ln(□), 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.

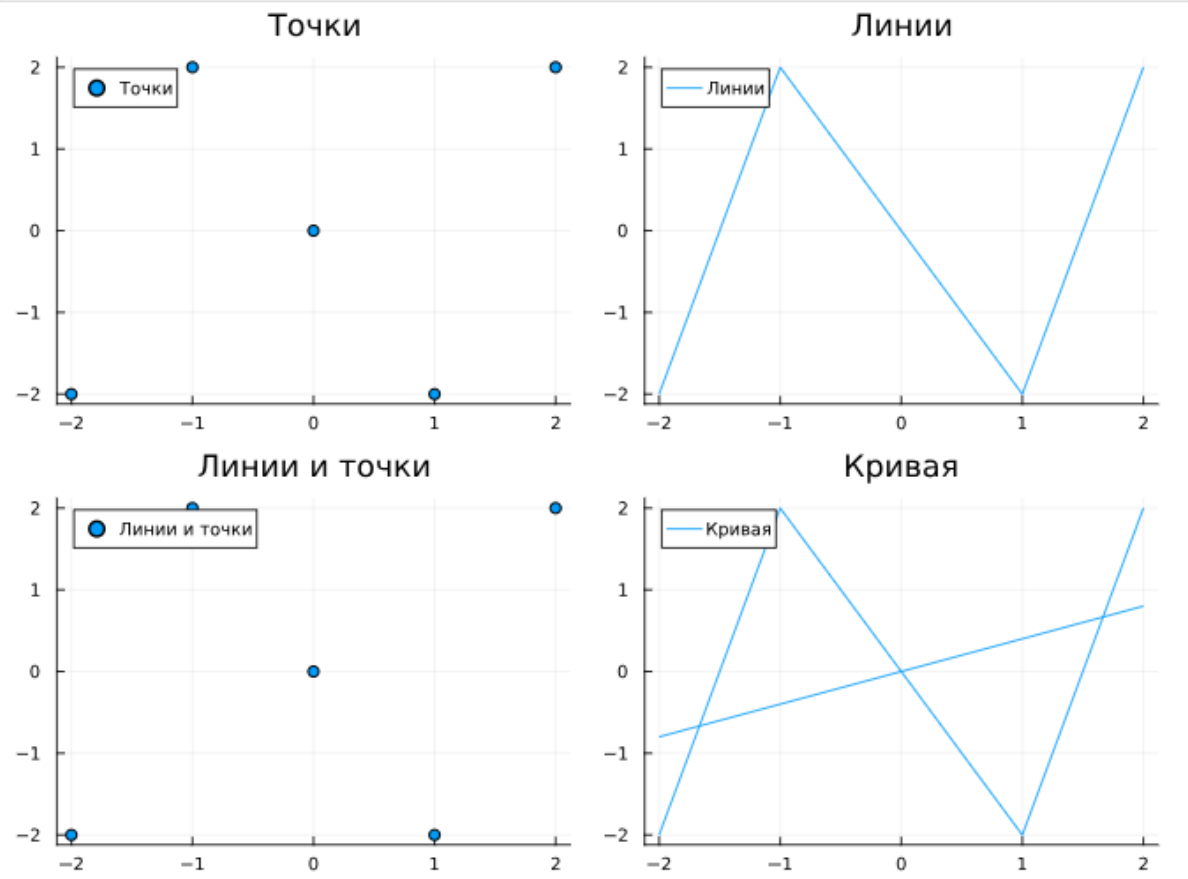


Рис. 2.18: Задание 3

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

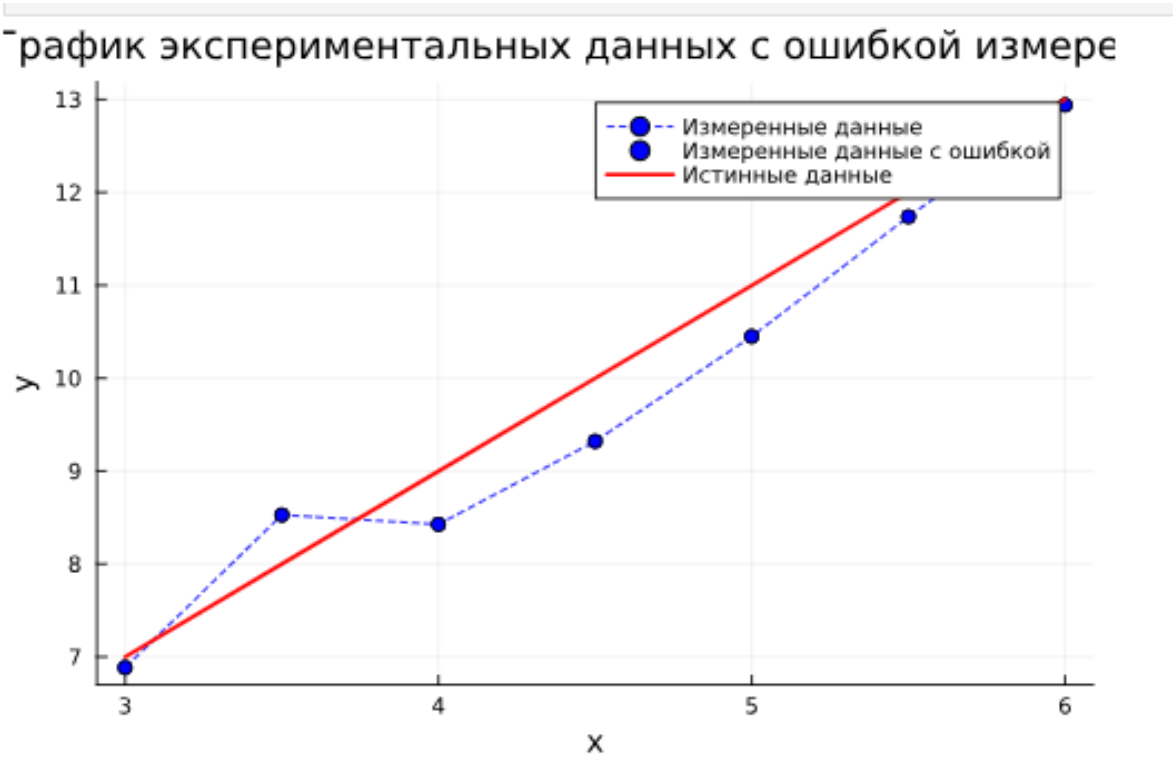


Рис. 2.19: Задание 6

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

Рис. 2.20: Задание 8

(В отчете не все отображены задания, т.к. их очень много. А сам файл я не могу загрузить, т.к. он больше 25мб)

3 Выводы

Мы освоиили синтаксис языка Julia для построения графиков.