---

## Front matter

title: "Компьютерный практикум по статистическому анализу данных"

subtitle: "Отчёт по лабораторной работе №5: Построение графиков"

author: "Ахлиддинзода Аслиддин"

## Generic otions

lang: ru-RU

toc-title: "Содержание"

## Bibliography

bibliography: bib/cite.bib

csl: pandoc/csl/gost-r-7-0-5-2008-numeric.csl

## Pdf output format

toc: true # Table of contents

toc-depth: 2

lof: true # List of figures

lot: true # List of tables

fontsize: 12pt

linestretch: 1.5

papersize: a4

documentclass: scrreprt

## I18n polyglossia

polyglossia-lang:

name: russian

options:

- spelling=modern

- babelshorthands=true

polyglossia-otherlangs:

name: english

## I18n babel

babel-lang: russian

babel-otherlangs: english

## Fonts

mainfont: PT Serif

romanfont: PT Serif

sansfont: PT Sans

monofont: PT Mono

mainfontoptions: Ligatures=TeX

romanfontoptions: Ligatures=TeX

sansfontoptions: Ligatures=TeX,Scale=MatchLowercase

monofontoptions: Scale=MatchLowercase,Scale=0.9

## Biblatex

biblatex: true

biblio-style: "gost-numeric"

biblatexoptions:

- parentracker=true

- backend=biber

- hyperref=auto

- language=auto

- autolang=other\*

- citestyle=gost-numeric

## Pandoc-crossref LaTeX customization

figureTitle: "Рис."

tableTitle: "Таблица"

listingTitle: "Листинг"

lolTitle: "Листинги"

## Misc options

indent: true

header-includes:

- \usepackage{indentfirst}

- \usepackage{float} # keep figures where there are in the text

- \floatplacement{figure}{H} # keep figures where there are in the text

- \usepackage{unicode-math}

- \setmathfont{Latin Modern Math}

---

# Цель работы

Основной целью работы освоить синтаксис языка Julia для построения графиков.

# Выполнение лабораторной работы

## 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.

Рассмотрим построение графика функции f(x) = (3x2 + 6x − 9)e−0,3x разными способами (рис. [-fig@:001] - рис. [-fig@:002]):

![График функции, построенный при помощи gr()](image/1.PNG){ #fig:001 width=100% height=100% }

![График функции, построенный при помощи pyplot()](image/2.PNG){ #fig:002 width=100% height=100% }

## Опции при построении графика

На примере графика функции sin(x) и графика разложения этой функции в ряд Тейлора рассмотрим дополнительные возможности пакетов для работы с графикой(рис. [-fig@:003] - рис. [-fig@:005] ):

![График функции sin(x)](image/3.PNG){ #fig:003 width=100% height=100% }

![График функции разложения исходной функции в ряд Тейлора](image/4.PNG){ #fig:004 width=100% height=100% }

![Графики исходной функции и её разложения в ряд Тейлора](image/5.PNG){ #fig:005 width=100% height=100% }

Затем добавим различные опции для отображения на графике (рис.[-fig@:006]):

![Вид графиков после добавления опций при их построении](image/6.PNG){ #fig:006 width=100% height=100% }

## Точечный график

Как и при построении обычного графика для точечного графика необходимо задать массив значений x, посчитать или задать значения y, задать опции построения график (рис. [-@fig:007]):

![График десяти случайных значений на плоскости (простой точечный график](image/7.PNG){ #fig:007 width=100% height=100% }

Для точечного графика можно задать различные опции, например размер маркера, его тип, цвет и и т.п. (рис. [-@fig:008]):

![График пятидесяти случайных значений на плоскости с различными опциями отображения (точечный график с кодированием значения размером точки](image/8.PNG){ #fig:008 width=100% height=100% }

Также можно строить и 3-мерные точечные графики (рис. [-@fig:009]):

![График пятидесяти случайных значений в пространстве с различными опциями отображения (3-мерный точечный график с кодированием значения размером точки)](image/9.PNG){ #fig:009 width=100% height=100% }

## Аппроксимация данных

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

Для демонстрации зададим искусственно некоторую функцию, в данном случае похожую по поведению на экспоненту (рис. [-@fig:010]):

![Пример функции](image/10.PNG){ #fig:010 width=100% height=100% }

Аппроксимируем полученную функцию полиномом 5-й степени (рис. [-@fig:011]):

![Пример аппроксимации исходной функции полиномом 5-й степени](image/11.PNG){ #fig:011 width=100% height=100% }

## Две оси ординат

Иногда требуется на один график вывести несколько траекторий с существенными отличиями в значениях по оси ординат.

Пример первой траектории (рис. [-@fig:012]):

![Примеры отдельно построенной траектории](image/12.PNG){ #fig:012 width=100% height=100% }

## Полярные координаты

Приведём пример построения графика функции в полярных координатах (рис. [-@fig:013]):

![График функции, заданной в полярных координатах](image/13.PNG){ #fig:013 width=100% height=100% }

## Параметрический график

Приведём пример построения графика параметрически заданной кривой на плоскости (рис. [-@fig:014]):

![Параметрический график кривой на плоскости](image/14.PNG){ #fig:014 width=100% height=100% }

Далее приведём пример построения графика параметрически заданной кривой в пространстве (рис. [-@fig:015]):

![Параметрический график кривой в пространстве](image/15.PNG){ #fig:015 width=100% height=100% }

## График поверхности

Для построения поверхности, заданной уравнением f(x, y) = x2 + y2, можно воспользоваться функцией surface() (рис. [-@fig:016]):

![График поверхности (использована функция surface())](image/16.PNG){ #fig:016 width=100% height=100% }

Также можно воспользоваться функцией plot() с заданными параметрами (рис. [-@fig:017]):

![График поверхности (использована функция plot())](image/17.PNG){ #fig:017 width=100% height=100% }

Можно задать параметры сглаживания (рис. [-@fig:018]):

![Сглаженный график поверхности](image/18.PNG){ #fig:018 width=100% height=100% }

Можно задать определённый угол зрения (рис. [-@fig:019]):

![График поверхности с изменённым углом зрения](image/19.PNG){ #fig:019 width=100% height=100% }

## 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.

Рассмотрим поверхность, заданную функцией g(x, y) = (3x + y2)| sin(x) + cos(y)| (рис.[-fig@:020]):

![График поверхность, заданную функцией g(x, y) = (3x + y2)| sin(x) + cos(y)|](image/20.PNG){ #fig:020 width=100% height=100% }

Линии уровня можно построить, используя проекцию значений исходной функции на плоскость (рис.[-fig@:021]):

![Линии уровня](image/21.PNG){ #fig:021 width=100% height=100% }

Можно дополнительно добавить заливку цветом (рис.[-fig@:022]):

![Линии уровня с заполнением](image/22.PNG){ #fig:022 width=100% height=100% }

## 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 h(x, y) = x3 − 3x + y2 сначала построим её график ( рис.[-fig@:023]) и линии уровня ( рис.[-fig@:024]):

![График функции h(x, y) = x3 − 3x + y2](image/23.PNG){ #fig:023 width=100% height=100% }

![Линии уровня функции h(x, y) = x3 − 3x + y2](image/24.PNG){ #fig:024 width=100% height=100% }

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.

## Анимация

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

В Julia рекомендуется использовать gif-анимацию в pyplot().

Строим поверхность (рис.[-fig@:025])

![Статичный график поверхности](image/25.PNG){ #fig:025 width=100% height=100% }

Добавляем анимацию (рис.[-fig@:026])

![Анимированный график поверхности](image/26.PNG){ #fig:026 width=100% height=100% }

## Гипоциклоида

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

Построим большую окружность (рис.[-fig@:027]):

![Большая окржуность гипоциклоида](image/27.PNG){ #fig:027 width=100% height=100% }

Для частичного построения гипоциклоиды будем менять параметр t (рис.[-fig@:028]):

![Половина пути гипоциклоиды](image/28.PNG){ #fig:028 width=100% height=100% }

Добавляем малую окружность гипоциклодиы (рис.[-fig@:029]):

![Малая окружность гипоциклоиды ](image/29.PNG){ #fig:029 width=100% height=100% }

Добавим радиус для малой окружности (рис.[-fig@:030]):

![Малая окружность гипоциклоиды с добавлением радиуса](image/30.PNG){ #fig:030 width=100% height=100% }

В конце сделаем анимацию получившегося изображения (рис.[-fig@:031]):

![Малая окружность гипоциклоиды с добавлением радиуса(анимация)](image/31.PNG){ #fig:031 width=100% height=100% }

## Errorbars

В исследованиях часто требуется изобразить графики погрешностей измерения.

Построим график исходных значений (рис.[-fig@:032]):

![График исходных значений](image/32.PNG){ #fig:032 width=100% height=100% }

Построим график отклонений от исходных значений (рис.[-fig@:033]):

![График отклонений от исходных значений](image/33.PNG){ #fig:033 width=100% height=100% }

Повернем график (рис.[-fig@:034]):

![Поворот графика](image/34.PNG){ #fig:034 width=100% height=100% }

Заполним область цветом (рис.[-fig@:035]):

![Заполнение цветом](image/35.PNG){ #fig:035 width=100% height=100% }

Можно построить график ошибок по двум осям (рис.[-fig@:036]):

![График ошибок по двум осям](image/36.PNG){ #fig:036 width=100% height=100% }

Можно построить график ассиметричных ошибок по двум осям (рис.[-fig@:037]):

![График ассиметричных ошибок по двум осям](image/37.PNG){ #fig:037 width=100% height=100% }

## Использование пакета Distributions

Строим гистограмму (рис. [-@fig:038]):

![Гистограмма, построенная по массиву случайных чисел](image/38.PNG){ #fig:038 width=100% height=100% }

Задаём нормальное распределение и строим гистограмму (рис. [-@fig:039]):

![Гистограмма нормального распределения](image/39.PNG){ #fig:039 width=100% height=100% }

Далее применим для построения нескольких гистограмм распределения людей по возрастам на одном графике plotly() (рис. [-@fig:040]):

![Гистограмма распределения людей по возрастам](image/40.PNG){ #fig:040 width=100% height=100% }

## Подграфики

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 (рис. [-@fig:041]):

![Серия из 4-х графиков в ряд](image/41.PNG){ #fig:041 width=100% height=100% }

Для автоматического вычисления сетки необходимо передать layout целое число (рис. [-@fig:042]):

![Серия из 4-х графиков в сетке](image/42.PNG){ #fig:042 width=100% height=100% }

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 (рис. [-@fig:043]):

![Объединение нескольких графиков в одной сетке](image/43.PNG){ #fig:043 width=100% height=100% }

Обратите внимание, что атрибуты на отдельных графиках применяются к отдельным графикам, в то время как атрибуты в последнем вызове plot применяются ко всем графикам.

Разнообразные варианты представления данных (рис. [-@fig:044]):

![Разнообразные варианты представления данных](image/44.PNG){ #fig:044 width=100% height=100% }

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. [-@fig:045]):

![Демонстрация применения сложного макета для построения графиков](image/45.PNG){ #fig:045 width=100% height=100% }

## Самостоятельное выполнение

Выполнение задания №1 (рис. [-@fig:046]):

![Выполнение задания №1](image/46.PNG){ #fig:046 width=100% height=100% }

Выполнение задания №2 (рис. [-@fig:047]):

![Выполнение задания №2](image/47.PNG){ #fig:047 width=100% height=100% }

Выполнение задания №3 (рис. [-@fig:048]):

![Выполнение задания №3](image/48.PNG){ #fig:048 width=100% height=100% }

Выполнение задания №4 (рис. [-@fig:049]):

![Выполнение задания №4](image/49.PNG){ #fig:049 width=100% height=100% }

Выполнение задания №5. Часть 1 (рис. [-@fig:050]):

![Выполнение задания №5. Часть 1](image/50.PNG){ #fig:050 width=100% height=100% }

Выполнение задания №5. Часть 2 (рис. [-@fig:051]):

![Выполнение задания №5. Часть 2](image/51.PNG){ #fig:051 width=100% height=100% }

Выполнение задания №6 (рис. [-@fig:052]):

![Выполнение задания №6](image/52.PNG){ #fig:052 width=100% height=100% }

Выполнение задания №7 (рис. [-@fig:053]):

![Выполнение задания №7](image/53.PNG){ #fig:053 width=100% height=100% }

Выполнение задания №8 (рис. [-@fig:054]):

![Выполнение задания №8](image/54.PNG){ #fig:054 width=100% height=100% }

Выполнение задания №9 (рис. [-@fig:055]):

![Выполнение задания №9](image/55.PNG){ #fig:055 width=100% height=100% }

Выполнение задания №10 (рис. [-@fig:056]):

![Выполнение задания №10](image/56.PNG){ #fig:056 width=100% height=100% }

Выполнение задания №11 (рис. [-@fig:057]):

![Выполнение задания №11](image/57.PNG){ #fig:057 width=100% height=100% }

# Вывод

В ходе выполнения лабораторной работы был освоен синтаксис языка Julia для построения графиков

# Список литературы. Библиография

[1] Mininet: https://mininet.org/