

**Звіт із завдання:  
«Графік функції»**

Роботу виконав:

Розумєй Максим Віталійович

Студент I курсу НаУКМА

Факультет Інформатики

Спеціальність:

Інженерія програмного забезпечення

Група 4

Викладач:

Пєчкурова Олена Миколаївна

Київ — 2024

Зміст

[Графік Функції 3](#_Toc167132774)

[Аналіз задачі 3](#_Toc167132775)

[Постановка задачі 4](#_Toc167132776)

[Структура програми 4](#_Toc167132777)

[Опис методів та класів 5](#_Toc167132778)

[Інструкція користувача 6](#_Toc167132779)

[Проблеми, що виникали 14](#_Toc167132780)

[Висновок 15](#_Toc167132781)

[Додаток 1. Програмний код графік функції 15](#_Toc167132782)

# Графік Функції

## Аналіз задачі

Побудувавши таблицю значень для початкових параметрів за допомогою Excel і створивши на основі неї графіки у декартових та полярних координатах, отримаємо такий результат:

Зображення, що містить текст, знімок екрана, схема, програмне забезпечення

Автоматично згенерований опис

Зображення, що містить текст, знімок екрана, Шрифт, схема

Автоматично згенерований опис

Було додано слайдер, прив’язаний до однієї з статичних комірок з параметром для формування певної анімації при зміні. За допомогою цього можемо впевнитись, що нам потрібна саме декартова система координат.

## Постановка задачі

Проаналізувати функцію за допомогою Excel. Побудувати таблицю початкових даних та графік Вашої функції (x = +-sqrt(A\*cos(Bt)\*cos(t) + C)

y = +-sqrt(A\*cos(Bt)\*sin(t) + C)). Написати програму яка будує графік заданої функції.

Вимоги до програми, обов'язкові функції:

1) Можливість задавати початкові дані, діапазон та крок.

2) Збереження графіка в файл.

3) У вікні програми є назва графіка, його формула та автор.

## Структура програми

Програма складається з класу, що наслідується від класу-додатку JavaFX.

Для побудови графіка використовується бібліотека JFreeChart.

Зображення, що містить текст, знімок екрана, Паралель, число

Автоматично згенерований опис

## Опис методів та класів

Клас **FunctionGraph**

Клас-програма, що наслідується від Application з бібліотеки JavaFX. У даному класі описано усе, що необхідно для поєднання компонентів, створення вікна та роботи програми.

Має поля - елементи Spinner для зберігання значень параметрів, поле графіка та кнопки збереження.

Метод **start**

Метод, який необхідно перевизначити для запуску програми на JavaFX.

Метод **getScene**

Метод, що ініціалізує та розташовує усі елементи на сцені для вікна та повертає дану сцену.

Метод **buttonClickedEvent**

Метод, що викликається при натисканні кнопки збереження. Створює новий файл, що містить поточну дату та час. Зберігає графік у форматі .PNG.  
Виводить екран-повідомлення про успішність або неуспішність операції. Також показує, куди було збережено файл.

Метод **createChart**

Метод ініціалізує початковий лінійний графік з підписом автора.

Метод **getGridPaneForLabel**

Метод створює табличку з двома комірками, щоб вирівняти напис та поле для введення.

Метод **parametersChangedEvent**

Метод, що викликається тоді, коли вміст будь-якого спінера (поля для введення параметрів) зміниться, що дозволяє динамічно перебудовувати графік. Метод тимчасово робить кнопку збереження недоступною.

Метод **buildChart**

Метод бере значення зі спінерів та обчислює набори точок. Набори точок передаються до графіку та змінюють його. Також змінюється напис, що показує, які параметри було використано.

## Інструкція користувача

Початковий екран та поля для введення параметрів A, B, C. Можна вводити дробові числа від -100 до 100. Також можна натиснути або затиснути стрілки для зміни значень. Після запуску відразу введені початкові дані. Можна помітити, що параметр B задає кількість пелюсток на кожній чверті графіка. Для того, щоб приблизити графік, наведіть курсор миші на графік на використовуйте коліщатко.

Зображення, що містить текст, схема, знімок екрана, ряд

Автоматично згенерований опис

У цих полях можна змінити з якого по яке значення t будуть обчислюватися точки. Можна вводити дробові числа від -100 до 100. Впливає на кількість ліній, якщо розділити їх за допомогою параметра B.

Зображення, що містить текст, схема, знімок екрана

Автоматично згенерований опис

Тут можна задати крок обчислення точок. Можна вводити цілі числа з 1 до 10000. Введене число буде поділено на 1000 задля отримання кроку. Впливає на точність та плавність графіка. **Низькі значення значно навантажують систему.**

Зображення, що містить текст, знімок екрана, схема

Автоматично згенерований опис

Усі обрані параметри автоматично показуватимуться над графіком.

Зображення, що містить текст, знімок екрана, схема

Автоматично згенерований опис

Дана кнопка зберігає поточний графік у файл формату PNG розміром 970 на 1000 px. До фото входить усе, що розташовано зліва від бічної панелі. У разі успішного збереження з’явиться вікно-повідомлення з шляхом до збереженого файлу.

Зображення, що містить текст, знімок екрана, схема

Автоматично згенерований опис

Для продовження роботи з програмою, натисніть ОК або закрийте вікно-повідомлення. Назва файлу міститиме дату та час збереження.

Зображення, що містить текст, знімок екрана, схема

Автоматично згенерований опис

Приклади різних графіків. Введіть відповідні параметри і Ви отримаєте такі ж графіки.  
Зображення, що містить текст, ескіз, малюнок, Дитяча творчість

Автоматично згенерований опис

Зображення, що містить текст, візерунок

Автоматично згенерований опис

Зображення, що містить текст, квітка, знімок екрана

Автоматично згенерований опис

Зображення, що містить текст, знімок екрана

Автоматично згенерований опис

Зображення, що містить текст, візерунок, знімок екрана, мистецтво

Автоматично згенерований опис

## Проблеми, що виникали

* Початкова функція допускала від’ємні значення під коренем.

Рішення: використати модуль для підкореневого виразу.

* Нерівне розташування параметрів та полів.

Рішення: використати GridPane для кожної пари елементів.

* Неможливо задати розташування кнопки знизу для VBox.

Рішення: використати порожній Region та задати йому параметр VBox.setVgrow(spacer, Priority.ALWAYS); що змусить його знайняти увесь вільний простір, посунувши кнопку донизу.

* Спінер для кроку повинен мати маленькі дробові значення, але навіть при встановлення маленького кроку для спінера він округлював значення так, що ним неможливо було зручно користуватися, і навіть встанолював значення 0, яке не було в діапазоні його роботи.

Рішення: допускати достатньо великі значення з цілим кроком і потім ділити значення на 1000.

* Дана нам функція складається з чотирьох однакових частин у кожній чверті. Отже потрібно якось відобразити чотири графіки на одному.

Рішення: сформувати чорити Dataset-ти множачи на -1 потрібні значення та додати їх за допомогою plot.setDataset(0, dataset);

## Висновок

Проаналізувавши функцію, я створив графічну програму для її відображення з різними параметрами. Здобув досвід у використанні нової для мене бібліотеки віконних програм JavaFX та у інтегруванні бібліотеки JFreeChart за допомогою інструменту побудови проєктів Maven. Вирішення проблем покращило мою компетентність у розробці програмного забезпечення.

# Додаток 1. Програмний код графік функції

package org.example.functiongraph\_8\_140524;

import javafx.application.Application;

import javafx.geometry.HPos;

import javafx.geometry.Insets;

import javafx.geometry.Pos;

import javafx.scene.Node;

import javafx.scene.Scene;

import javafx.scene.control.\*;

import javafx.scene.layout.\*;

import javafx.scene.paint.Color;

import javafx.scene.text.Font;

import javafx.stage.Stage;

import org.jfree.chart.ChartFactory;

import org.jfree.chart.ChartUtils;

import org.jfree.chart.JFreeChart;

import org.jfree.chart.fx.ChartViewer;

import org.jfree.chart.plot.XYPlot;

import org.jfree.chart.title.TextTitle;

import org.jfree.data.xy.XYDataset;

import org.jfree.data.xy.XYSeries;

import org.jfree.data.xy.XYSeriesCollection;

import java.io.File;

import java.io.IOException;

import java.text.SimpleDateFormat;

import java.util.Date;

/\*\*

\* JavaFX App for plotting and saving the graph of the parametric function:

\* x = +-sqrt(|A\*cos(Bt)\*cos(t) + C|)

\* y = +-sqrt(|A\*cos(Bt)\*sin(t) + C|)

\* where A, B, C are parameters, t is a parameter from the range [from, to] with a step.

\* The graph is saved as a PNG file.

\* The app uses JFreeChart library for plotting the graph.

\* The app is implemented using JavaFX.

\* Created for educational practice task v8

\*

\* @author Розумєй Максим Віталійович

\*/

public class FunctionGraph extends Application {

private Spinner<Double> spinnerA;

private Spinner<Double> spinnerB;

private Spinner<Double> spinnerC;

private Spinner<Double> spinnerFrom;

private Spinner<Double> spinnerTo;

private Spinner<Double> spinnerStep;

private JFreeChart chart;

private Button saveButton;

public static void main(String[] args) {

launch();

}

/\*\*

\* Start the JavaFX application

\* @param stage Stage

\*/

@Override

public void start(Stage stage) {

Scene scene = getScene();

stage.setTitle("Parametric Function Graph");

stage.setScene(scene);

stage.setMinHeight(200);

stage.setMinWidth(500);

parametersChangedEvent();

stage.show();

}

/\*\*

\* Create the main scene of the application

\* @return Scene

\*/

private Scene getScene() {

Label functionLabel = new Label("x = +-sqrt(|A\*cos(Bt)\*cos(t) + C|)\ny = +-sqrt(|A\*cos(Bt)\*sin(t) + C|)");

functionLabel.setFont(Font.font("Times New Roman", 20.0));

functionLabel.setWrapText(true);

functionLabel.setMaxSize(Label.USE\_PREF\_SIZE, Label.USE\_PREF\_SIZE);

Label labelParam = new Label("Parameters:");

labelParam.setFont(Font.font("Times New Roman", 17.0));

Label labelA = new Label("A");

spinnerA = new Spinner<>(-100.0, 100.0, 3.0, 0.1);

GridPane gridPaneA = getGridPaneForLabel(labelA, spinnerA);

Label labelB = new Label("B");

spinnerB = new Spinner<>(-100.0, 100.0, 2.0, 0.01);

GridPane gridPaneB = getGridPaneForLabel(labelB, spinnerB);

Label labelC = new Label("C");

spinnerC = new Spinner<>(-5.0, 5.0, 0.82, 0.03);

GridPane gridPaneC = getGridPaneForLabel(labelC, spinnerC);

Label labelRange = new Label("Range of t:");

labelRange.setFont(Font.font("Times New Roman", 17.0));

Label labelFrom = new Label("From");

spinnerFrom = new Spinner<>(-100.0, 100.0, -5.0, 0.1);

GridPane gridPaneFrom = getGridPaneForLabel(labelFrom, spinnerFrom);

Label labelTo = new Label("To");

spinnerTo = new Spinner<>(-100.0, 100.0, 5.0, 0.1);

GridPane gridPaneTo = getGridPaneForLabel(labelTo, spinnerTo);

Label labelStep = new Label("Step/1000");

spinnerStep = new Spinner<>(1, 10000.0, 10, 1);

GridPane gridPaneStep = getGridPaneForLabel(labelStep, spinnerStep);

VBox vBox = new VBox(10, functionLabel, labelParam, gridPaneA, gridPaneB, gridPaneC, labelRange,

gridPaneFrom, gridPaneTo, gridPaneStep);

for (Node child : vBox.getChildren()) {

VBox.setMargin(child, new Insets(0, 0, 0, 10));

}

saveButton = new Button("Save");

saveButton.setFont(Font.font("Times New Roman", 20.0));

HBox buttonBox = new HBox(saveButton);

buttonBox.setAlignment(Pos.CENTER);

Region spacer = new Region();

VBox.setVgrow(spacer, Priority.ALWAYS);

vBox.getChildren().add(spacer);

vBox.getChildren().add(buttonBox);

VBox.setMargin(buttonBox, new Insets(0, 0, 10, 0));

saveButton.setOnAction(event -> {

buttonClickedEvent();

});

ScrollPane scrollPane = new ScrollPane(vBox);

scrollPane.setFitToWidth(true);

scrollPane.setFitToHeight(true);

scrollPane.setBorder(new Border(new BorderStroke(Color.BLACK, BorderStrokeStyle.SOLID, CornerRadii.EMPTY,

BorderWidths.DEFAULT)));

createChart();

ChartViewer chartViewer = new ChartViewer(chart);

vBox.setBackground(Background.fill(new Color(0.5, 0.9, 0.9, 0.5)));

BorderPane borderPane1 = new BorderPane();

borderPane1.setCenter(chartViewer);

borderPane1.setRight(scrollPane);

borderPane1.setBackground(Background.fill(new Color(0.6, 0.8, 0.6, 0.5)));

return new Scene(borderPane1, 960, 700);

}

/\*\*

\* Event handler for the Save button

\* Save the chart as a PNG file

\* Show an alert dialog with the result

\* Uses JFreeChart ChartUtils to save the chart

\* Uses Date to get the current date and time for the file name

\*/

private void buttonClickedEvent() {

System.out.println("A = " + spinnerA.getValue());

System.out.println("B = " + spinnerB.getValue());

System.out.println("C = " + spinnerC.getValue());

System.out.println("From = " + spinnerFrom.getValue());

System.out.println("To = " + spinnerTo.getValue());

System.out.println("Step = " + spinnerStep.getValue());

try {

Date currentDate = new Date();

SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd\_HH.mm.ss");

String currentDateTime = dateFormat.format(currentDate);

File outputFile = new File("graph8\_"+ currentDateTime + ".png");

if(outputFile.createNewFile()){

ChartUtils.saveChartAsPNG(outputFile, chart, 960, 1000);

System.out.println("Chart saved to " + outputFile.getAbsolutePath());

Alert alert = new Alert(Alert.AlertType.INFORMATION);

alert.setTitle("Information Dialog");

alert.setHeaderText("Success!");

alert.setContentText("Chart saved to " + outputFile.getAbsolutePath());

alert.showAndWait();

} else {

System.out.println("File already exists.");

Alert alert = new Alert(Alert.AlertType.ERROR);

alert.setTitle("Error Dialog");

alert.setHeaderText("Error!");

alert.setContentText("File already exists.");

alert.showAndWait();

}

} catch (IOException e) {

e.printStackTrace();

Alert alert = new Alert(Alert.AlertType.ERROR);

alert.setTitle("Error Dialog");

alert.setHeaderText("Error!");

alert.setContentText(e.getMessage());

alert.showAndWait();

}

}

/\*\*

\* Create the line chart with the default dataset and author subtitle

\*/

private void createChart() {

XYSeries series = new XYSeries("Example Series");

XYDataset dataset = new XYSeriesCollection(series);

chart = ChartFactory.createXYLineChart("Example Chart", "X-Axis", "Y-Axis", dataset);

chart.addSubtitle(new TextTitle("Author: Maksym Rozumiei", new java.awt.Font("Arial",

java.awt.Font.PLAIN, 12)));

}

/\*\*

\* Create a GridPane with a Label and a Spinner

\* @param label Label

\* @param spinner Spinner

\* @return GridPane

\*/

private GridPane getGridPaneForLabel(Label label, Spinner<Double> spinner) {

label.setFont(Font.font("Times New Roman", 15.0));

GridPane.setHalignment(label, HPos.CENTER);

label.setLabelFor(spinner);

spinner.setEditable(true);

spinner.valueProperty().addListener((observable, oldValue, newValue) -> {

System.out.println("Spinner value changed from " + oldValue + " to " + newValue);

parametersChangedEvent();

});

GridPane gridPane = new GridPane();

gridPane.add(label, 0, 0);

gridPane.add(spinner, 1, 0);

ColumnConstraints column1 = new ColumnConstraints();

column1.setPercentWidth(50);

gridPane.getColumnConstraints().addAll(column1, column1);

return gridPane;

}

/\*\*

\* Event handler for the Spinner value change

\* Rebuild the chart with the new parameters

\* Disable the Save button while the chart is being built

\*/

private void parametersChangedEvent() {

saveButton.setDisable(true);

buildChart();

saveButton.setDisable(false);

}

/\*\*

\* Build the chart with the current parameters

\* Set the title of the chart with the current parameters

\*/

private void buildChart() {

double a = spinnerA.getValue();

double b = spinnerB.getValue();

double c = spinnerC.getValue();

double from = spinnerFrom.getValue();

double to = spinnerTo.getValue();

double step = spinnerStep.getValue()/1000.0;

chart.setTitle("x = +-sqrt(|"+a+"\*cos("+b+"t)\*cos(t) + ("+c+")|)\n" +

"y = +-sqrt(|"+a+"\*cos("+b+"t)\*sin(t) + ("+c+")|)\n" +

"t from " + from + " to " + to + " with step " + step);

XYSeries series1 = new XYSeries("Generated Series 1", false, true);

XYSeries series2 = new XYSeries("Generated Series 2", false, true);

XYSeries series3 = new XYSeries("Generated Series 3", false, true);

XYSeries series4 = new XYSeries("Generated Series 4", false, true);

for (double t = from; t <= to; t += step) {

double x = Math.sqrt(Math.abs(a \* Math.cos(b \* t) \* Math.cos(t) + c));

double y = Math.sqrt(Math.abs(a \* Math.cos(b \* t) \* Math.sin(t) + c));

series1.add(x, y);

series2.add(-x, y);

series3.add(x, -y);

series4.add(-x, -y);

}

XYDataset dataset = new XYSeriesCollection(series1);

XYPlot plot = (XYPlot) chart.getPlot();

plot.setDataset(0, dataset);

plot.setDataset(1, new XYSeriesCollection(series2));

plot.setDataset(2, new XYSeriesCollection(series3));

plot.setDataset(3, new XYSeriesCollection(series4));

}

}