Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мережних технологій факультету інформатики

Побудова графіків табличних функцій, параметричних функцій та явних функцій з керуванням товщиною та іншими параметрами на мові Java. Анімація при зміні одного параметра.

Текстова частина до курсової роботи

за спеціальністю «Комп'ютерні науки»

Керівник курсової роботи

Малашонок Г.І.

_____

(підпис)

"___" _____ 2024 р.

Виконав студент

Остролуцький А. Б.

"___" _____ 2024 р.

Київ 2024

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мережних технологій факультету інформатики

ЗАТВЕРДЖУЮ

Зав. кафедри мережних технологій,

Професор, доктор фіз-мат наук,

Малашонок Г. І.

_____

(підпис)

“___” _____ 2024 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу

студенту Остролуцькому Андрію Борисовичу факультету інформатики 3 курсу

ТЕМА Побудова графіків табличних функцій, параметричних функцій та явних функцій з керуванням товщиною та іншими параметрами на мові Java. Анімація при зміні одного параметра

Зміст ТЧ до курсової роботи:

Індивідуальне завдання

Вступ

РОЗДІЛ 1. Аналіз бібліотеки JFreeChart

РОЗДІЛ 2. Проєктування програми

РОЗДІЛ 3. Інструкція до використання

Висновки

Список літератури

Дата видачі “___” _____ 2024 р.

Керівник _____

(підпис)

Завдання отримав _____

(підпис)

Календарний план виконання курсової роботи

Тема: Побудова графіків табличних функцій, параметричних функцій та явних функцій з керуванням товщиною та іншими параметрами на мові Java. Анімація при зміні одного параметра

Календарний план виконання курсової роботи:

№ п/п	Назва етапу курсової роботи	Термін виконання етапу	Примітка
1.	Отримання завдання на курсову роботу	жовтень 2023 р.	
2.	Пошук та ознайомлення з літературою за темою роботи	листопад 2023 р.	
3.	Створення плану роботи	грудень 2023 р.	
4.	Написання чорнового варіанту теоретичної частини	січень 2024 р.	
5.	Розробка чорнового варіанту коду	лютий 2024 р.	
6.	Надання роботи науковому керівнику для попереднього відгуку	березень 2024 р.	
7.	Внесення правок до ТЧ та коду згідно з відгуком наукового керівника	квітень 2024 р.	
8.	Фіналізація та захист курсової роботи	травень 2024 р.	

Студент Остролуцький А.Б.

Керівник Малашонок Г. І.

“____” _____

Зміст

Анотація

Курсова робота присвячена створенню програми для побудови графіків табличних функцій, параметричних функцій та явних функцій на мові Java за допомогою відкритої програмної бібліотеки JFreeChart. Створений функціонал був інтегрований в програмне забезпечення MathPartner. Окрім цього, в курсовій роботі надане обґрунтування обрання вищезгаданої бібліотеки разом з аналізом її переваг та недоліків, а також інструкція до використання розробленого застосунку. Застосунок дозволяє кастомізацію побудованих графіків, як-от зміни товщини, розміру шрифта тощо, та анімацію графіків при зміні параметрів.

Вступ

MathPartner – це сервіс, хмарна система символьних обчислень, розробці якого сприяли студенти Національного університету “Києво-Могилянська академія” під керівництвом професора Малашонка Геннадія Івановича. MathPartner є комплексною програмою для розв’язання низки математичних проблем, починаючи від простих числових операцій, закінчуючи вирішенням складних завдань з вищої математики [1]. Однією з багатьох можливостей, яку надає MathPartner, є будування графіків в двовимірному просторі. Побудова графіків є невід’ємною частиною функціоналу MathPartner, оскільки дозволяє візуалізувати будь-які функції, а саме: явні, параметричні та табличні функції.

Втім, функціонал пов’язаний з побудовою графіків функцій був створений досить давно з використанням вбудованої бібліотеки Java під назвою Abstract Window Toolkit (AWT). Хоча правильне використання вбудованих бібліотек має свої переваги, наявний програмний код пов’язаний з будуванням 2D графіків був досить заплутаним, неструктурованим, нехтував принципами об’єктно-орієнтованого програмування, що в результаті унеможливлювало його підтримку та розширення. Натомість правильне використання відкритих бібліотек покращує читабельність коду, сприяє стандартизації, полегшує додавання нового функціоналу та розширення можливостей застосунку і має потенціал покращити ефективність, якщо бібліотека належно оптимізована. Саме тому ця курсова робота ставить на меті розв’язати вищезгадані проблеми, а саме:

1) обрати нову бібліотеку на мові програмування Java, яка б давала можливість будувати графіки в двовимірному просторі, дозволяла б в

майбутньому з легкістю додавати нові типи графіків, при необхідності легко змінювати формат зображення на виході;

2) за допомогою нової бібліотеки створити структурований код для побудови 2D графіків, послуговуючись принципами ООП, що б дозволило його подальше розширення та ефективну підтримку.

РОЗДІЛ 1. Аналіз бібліотеки JFreeChart

1.1 Переваги бібліотеки JFreeChart

Під час обговорення обрання бібліотеки для побудови графіків на мові програмування Java враховуються кілька факторів, зокрема функціональність, простота використання, підтримка спільнотою та сумісність. Серед доступних варіантів, була обрана відкрита бібліотека JFreeChart. Ця бібліотека є найкращою завдяки своєму комплексному набору функцій і широкому застосуванню в екосистемі Java.

Однією з головних переваг JFreeChart є можливості налаштування. Розробники мають точний контроль над кожним аспектом зовнішнього вигляду та поведінки діаграми/графіка, що дозволяє їм налаштовувати візуальні елементи відповідно до конкретних вимог до дизайну. Незалежно від того, чи налаштовуєте ви кольори, шрифти, позначки осей або маркери графіка, JFreeChart пропонує безліч варіантів для створення візуально привабливих і змістовних графіків та діаграм.

Ще однією ключовою перевагою JFreeChart є надійна документація та підтримка спільноти. В той же час, наявність такої детальної документації є одним з головних показників успішної бібліотеки [2]. Офіційний веб-сайт JFreeChart є всеосяжним ресурсом, який пропонує детальну документацію API, навчальні посібники та зразки коду, щоб допомогти розробникам вивчати та ефективно використовувати бібліотеку. Окрім того, спільнота активних користувачів надає цінну підтримку та допомогу завдяки форумам, де користувачі можуть шукати допомоги, ділитися думками та співпрацювати над проєктами.

JFreeChart витримав випробування часом, продемонструвавши свою надійність і стабільність роками. З історією, яка нараховує понад два десятиліття, бібліотека зазнавала безперервного розвитку та вдосконалення, забезпечуючи сумісність із останніми версіями Java та галузевими стандартами, що розвиваються. Ця довговічність і прагнення до досконалості дають впевненість розробникам, роблячи JFreeChart надійним вибором для реалізації графічних рішень у програмах Java. Варто додати, що JFreeChart є одним з небагатьох програмних забезпечень, в якій розробники активно оновлюють версії бібліотек, на якій JFreeChart побудована (в середньому 37%) [3]. Окрім цього, дослідження різниць між послідовними версіями, починаючи з JFreeChart-0.9.0 і закінчуючи JFreeChart-0.9.21 показали, що кожна нова версія мали значні зміни. З новими версіями змінювались інтерфейси та/або класи, видалялись інтерфейси та/або класи, або/і додавались нові пакети, інтерфейси та/або класи. Деякі версії мали дуже значні зміни [4]. Це підтверджує тезу про довговічність JFreeChart та постійного прагнення до вдосконалення розробників цієї бібліотеки.

Варто згадати, що однією з переваг JFreeChart є її універсальність. Бібліотека пропонує широкий спектр типів графіків та діаграм, від базових лінійних до складних комбінованих, що задовольняють різноманітні потреби візуалізації. Незалежно від того, чи йдеться про візуалізацію математичної функції, відображення тенденцій у часі, порівняння кількох наборів даних чи ілюстрацію ієрархічних зв’язків, JFreeChart надає інструменти, необхідні для створення переконливої та інформативної графіки. Ця широта функціональності робить його добре придатним для різних застосувань у різних галузях, від наукових досліджень до бізнес-аналітики.

Нарешті, JFreeChart забезпечує підтримку багатьох типів виводу, включаючи компоненти Swing і JavaFX, файли зображень (включаючи PNG і JPEG) і формати файлів векторної графіки (включаючи PDF, EPS і SVG) [5].

В контексті проєктування застосунку для побудови графіків табличних, параметричних та явних функцій, універсальність JFreeChart не є вирішальним фактором для вибору. Проте усі інші згадані переваги стали критичними для обрання цієї бібліотеки. Однією з вимог до проєктованого застосунку є керування візуальними аспектами графіку (товщина ліній, розмір шрифта тощо), в чому JFreeChart не має рівних. Більше того, наявність актуальної документації та підтримка спільноти полегшили створення застосунку. Варто додати, що JFreeChart є безкоштовною бібліотекою з відкритим кодом, що стало вирішальним фактором у її виборі.

1.2 Недоліки бібліотеки JFreeChart

Хоча JFreeChart пропонує багато переваг, важливо також розглядати потенційні недоліки. В цьому підрозділі наведені негативні аспекти цієї бібліотеки, які безсумнівно варто враховувати.

По-перше, незважаючи на вичерпну документацію, освоєння JFreeChart може потребувати певного часу та зусиль, особливо для розробників, котрі тільки починають знайомитися з Java або з графічною візуалізацією даних. Широкий набір функцій бібліотеки та параметри налаштування можуть бути важкими для освоєння для початківців.

По-друге, хоча JFreeChart надає широкі можливості налаштування, досягнення певних розширених налаштувань або складних макетів може вимагати глибокого розуміння внутрішніх функцій бібліотеки та механізмів

відтворення. Розробники можуть зіткнутися з проблемами, намагаючись реалізувати вузькоспеціалізовані або нетрадиційні дизайни діаграм.

По-третє, обмежена інтерактивність: інтерактивні функції JFreeChart, такі як масштабування, панорамування та підказки, дещо обмежені. Розробникам, яким потрібна високоінтерактивна та динамічна візуалізація, може бути недостатній функціонал, що надає бібліотека JFreeChart .

Наостанок, інтеграція JFreeChart у проєкт може ввести додаткові залежності та збільшити його обсяг. Розробники повинні забезпечити сумісність з іншими бібліотеками та фреймворками, що використовуються в їхніх програмах, і керувати потенційними конфліктами чи проблемами версії.

Незважаючи на ці потенційні недоліки, JFreeChart залишається потужною та широко використовуваною бібліотекою для створення високоякісних графіків та діаграм в програмах на мові програмування Java. В контексті програми для побудови графіків суто математичних функцій, ці недоліки не є суттєвими: мені цілком вдалося освоїти цю бібліотеку з нуля, обмежена інтерактивність ефективно нівелюється деякими компонентами фронтенду, а інтеграція у проєкт Mathpartner та забезпечення сумісності з наявними бібліотеками були успішними.

РОЗДІЛ 2. Проєктування програми

2.1 Аналіз існуючого функціоналу для створення графіків у MathPartner

Як було згадано у вступній частині, MathPartner вже досить давно надає можливість будувати графіки математичних функцій, в тому числі явних, параметричних та табличних. Проте існуючий код має суттєві недоліки, також згадані у вступі: насамперед, код неструкторований, ігноруються принципи об’єктно-орієнтованого програмування. Фактично весь код, пов’язаний з побудовою графіків у двовимірному просторі, був уміщений у клас Plot. Випадки побудови різних типів графіків вирішувалися просто – викликом різних конструкторів класу, які обробляли точки та створювали зображення, що потім передавалися на фронтенд. Очевидно, така імплементація є не найкращою. Виклик різних конструкторів забезпечувався за рахунок різних наданих аргументів (overloading), що призводить до певних проблем. По-перше, функціонал в рамках такого коду важко додавати та розширювати. По-друге, розробнику важко орієнтуватися в такому коді – щоб зрозуміти який конструктор відповідає за що, необхідно відлагоджувати програму [6]. Через неструктурованість цей код важко підтримувати.

2.2 Структура коду

Окрім додавання нової бібліотеки, переваги якої були описані в минулому розділі, через суттєві недоліки старої структури коду, було вирішено кардинально по-новому підійти до проєктування коду.

Нижче представлена UML діаграма нового коду.

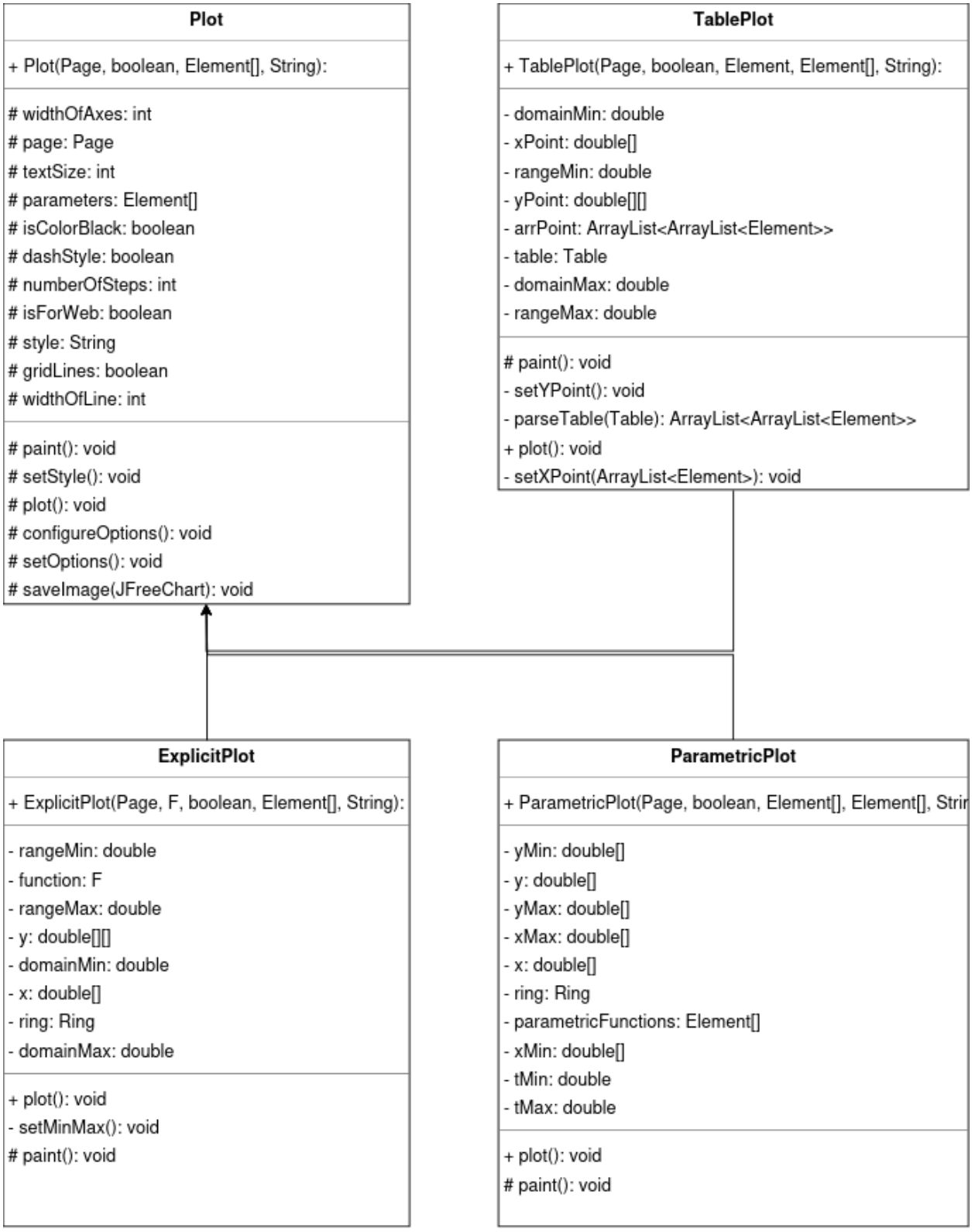


Рисунок 2.2 - Діаграма коду

2.3 Загальний опис коду

В новій версії Plot – це абстрактний клас, в той час ExplicitPlot, ParametricPlot, TablePlot – класи, що успадковують абстрактний клас Plot та відповідають за побудову відповідних двовимірних графіків. Така структура має багато плюсів. По-перше, введення абстрактного класу Plot створює абстракцію, яка визначає загальний інтерфейс для всіх типів графіків. Це забезпечує поліморфізм, що дозволяє розробникам використовувати цей загальний інтерфейс для взаємодії з будь-яким типом графіку без необхідності знати його конкретну реалізацію [7]. По-друге, така структура покращує читабельність коду. По-третє, успадкування: клас Plot сам містить декілька не абстрактних методів, які можуть використовувати класи, що успадковують абстрактний клас Plot. Це полегшує написання коду для побудови нових типів графіків. До цих методів відносяться методи:

1) setOptions() та configureOptions() – кастомізують графік, а саме: змінюють згідно з параметрами ширину ліній графіку, ширину осей, встановлюють чи графік має бути чорно-білим, чи мають бути відображені лінії сітки, а також змінюють розмір шрифта тексту;
2) setStyle() – встановлює чи необхідно малювати графік пунктирною лінією;
3) saveImage(JFreeChart) – локально зберігає зображення графіку, виконується тільки, якщо isForWeb є хибним.

2.4 Опис класу ExplicitPlot

Методи класу ExplicitPlot відповідають за створення та передачу на фронтенд зображення з графіком явної функції. ExplicitPlot використовує

методи класу Plot, згадані вище для кастомізації графіку. До безпосередніх методів класу ExplicitPlot відносяться:

1) setMinMax() – встановлює розмір полотна та межі графіку по абсцисі та ординаті;
2) plot() – відповідає за знаходження точок по осі ординаті;
3) paint() – відповідає за побудову графіка, створення зображення і передачу його на фронтенд.

З особливостей варто відмітити, що plot() переназначає глобальні змінні x та y, де x – це масив розміру numberOfSteps в межах заданого інтервалу, а y – матриця, в якій кожен рядок відповідає за значення точок по оридинаті окремих функцій у випадку, якщо користувач хоче побудувати графіки декількох функцій на одному полотні.

Нижче наведений код імплементації методу paint() для класу ExplicitPlot. В коментарях зазначені детальні пояснення до рядків коду:

```
@Override
public void paint() {
    // Створюємо колекцію
     // Колекція складається з рядів даних (series), кожен series містить точки однієї функції
    XYSeriesCollection dataset = new XYSeriesCollection();
    // Додаємо точки до колекції
     // Якщо dashStyle не є хибою, то пропускаємо точки з певним інтервалом щоб лінії графіків були пунктирними
    // Якщо точка виходить за межі, не додаємо її
    if (dashStyle) {
        boolean isSpace = false;
        for (int i = 0; i < y.length; i++) {
```

```
            XYSeries series = new XYSeries(function.X[i]);
            for (int j = 0; j < x.length; j++) {
                 if (y[i][j] == NumberR64.POSITIVE_INFINITY_DOUBLE
|| isSpace) {
                    series.add(x[j], null);
                } else {
                    series.add(x[j], y[i][j]);
                }
                if ((j + 1) % 5 == 0) {
                    isSpace = !isSpace;
                }
            }
            dataset.addSeries(series);
        }
    } else {
        for (int i = 0; i < y.length; i++) {
            XYSeries series = new XYSeries(function.X[i]);
            for (int j = 0; j < x.length; j++) {
                                                if   (y[i][j]   ==
NumberR64.POSITIVE_INFINITY_DOUBLE) {
                    series.add(x[j], null);
                } else {
                    series.add(x[j], y[i][j]);
                }
            }
            dataset.addSeries(series);
        }
    }
    // Створюємо графік
    JFreeChart chart = ChartFactory.createXYLineChart(
            page.title,
```

```
            page.nameOX,
            page.nameOY,
            dataset
    );
    XYPlot plot = (XYPlot) chart.getPlot();
    // Встановлюємо межі полотна
    plot.getDomainAxis().setRange(domainMin, domainMax);
    plot.getRangeAxis().setRange(rangeMin, rangeMax);
    // Якщо isColorBlack не є хибою, колір графіків буде чорним
    if (isColorBlack) {
                              XYLineAndShapeRenderer    renderer    =
(XYLineAndShapeRenderer) plot.getRenderer();
        for (int i = 0; i < dataset.getSeriesCount(); i++) {
            renderer.setSeriesPaint(i, Color.BLACK);
        }
    }
    // Якщо gridLines є хибою, лінії сітки не будуть відображені
    if (!gridLines) {
        plot.setDomainGridlinesVisible(true);
        plot.setRangeGridlinesVisible(true);
        plot.setDomainMinorGridlinesVisible(true);
        plot.setRangeMinorGridlinesVisible(true);
        plot.setDomainGridlinePaint(Color.LIGHT_GRAY);
        plot.setRangeGridlinePaint(Color.LIGHT_GRAY);
        plot.setDomainMinorGridlinePaint(Color.LIGHT_GRAY);
        plot.setRangeMinorGridlinePaint(Color.LIGHT_GRAY);
    }
      // Встановлюємо розмір тексту згідно з textSize та інші
параметри
    Font titleFont = new Font("SansSerif", Font.BOLD, textSize);
    TextTitle title = chart.getTitle();
```

```
    title.setFont(titleFont);
    Font axisFont = new Font("SansSerif", Font.PLAIN, textSize);
    ValueAxis domainAxis = plot.getDomainAxis();
    domainAxis.setTickLabelFont(axisFont);
    ValueAxis rangeAxis = plot.getRangeAxis();
    rangeAxis.setTickLabelFont(axisFont);
    LegendTitle legend = chart.getLegend();
    legend.setItemFont(axisFont);
    // Встановлюємо ширину ліній графіку згідно з widthOfLine
      XYLineAndShapeRenderer renderer = (XYLineAndShapeRenderer)
plot.getRenderer();
    for (int i = 0; i < dataset.getSeriesCount(); i++) {
                                renderer.setSeriesStroke(i,     new
BasicStroke(widthOfLine));
    }
    // Встановлюємо ширину осей згідно з widthOfAxes
                                plot.setRangeZeroBaselineStroke(new
BasicStroke(widthOfAxes));
                               plot.setDomainZeroBaselineStroke(new
BasicStroke(widthOfAxes));
    // Встановлюємо сірий колір полотна
    plot.setRangeZeroBaselinePaint(Color.DARK_GRAY);
    plot.setDomainZeroBaselinePaint(Color.DARK_GRAY);
    plot.setRangeZeroBaselineVisible(true);
    plot.setDomainZeroBaselineVisible(true);
      // Якщо isForWeb не є хибою, зображення відправляється на
фронтенд
    // Інакше, зображення зберігається локально для розробника
    if (isForWeb) {
            BufferedImage image = chart.createBufferedImage(1200,
800);
        page.putImage(image);
```

```
    } else {
        saveImage(chart);
    }
}
```

2.5 Опис класу ParametricPlot

Методи класу ParametricPlot відповідають за створення та передачу на фронтенд зображення з графіком параметричної функції. ParametricPlot використовує методи класу Plot, згадані вище для кастомізації графіку. До безпосередніх методів класу ParametricPlot відносяться:

1) plot() – відповідає за знаходження точок по осі абсцис та ординаті;
2) paint() – відповідає за побудову графіка, створення зображення і передачу його на фронтенд.

З особливостей варто відмітити, що plot() переназначає глобальні змінні x та y, де x – це масив, в якому кожне значення відповідає за значення точок функції по абсцисі, а y – масив, в якому кожне значення відповідає за значення точок функції по оридинаті.

2.6 Опис класу TablePlot

Методи класу TablePlot відповідають за створення та передачу на фронтенд зображення з графіком табличної функції. TablePlot використовує методи класу Plot, згадані вище для кастомізації графіку. До безпосередніх методів класу TablePlot відносяться:

1) parseTable(Table t) – створює з таблиці на вході (Table – клас, наданий в наявному коді MathPartner) двохвимірний ArrayList;

2) setXPoint() – встановлює координати по осі абсцис, отримуючи їх з першого рядка таблиці ;
3) setYPoint() – встановлює координати по осі ординат, отримуючи їх з усіх рядків таблиці, окрім першого;
4) plot() – викликає методи parseTable(Table t), setXPoint(), setYPoint();
5) paint() – відповідає за побудову графіка, створення зображення і передачу його на фронтенд.

РОЗДІЛ 3. Інструкція до використання

3.1 Початкові інструкції

Перед тим, як будувати будь-які типи графіків за допомогою графічного інтерфейсу MathPartner, необхідно дотриматись певних початкових інструкцій, а саме:

1) Встановити змінну простору SPACE = R64[x, y] – це означає, що ми будемо працювати зі дійсними змінними x та y;
2) Встановити межі двовимірного полотна:
    a) Якщо команда set2D() використовується без параметрів, границі графіків автоматично обчислюються, і для функцій вибирається інтервал [0,1] на горизонтальній вісі. Назви координатних осей будуть X та Y відповідно. Заголовок графіку залишається порожнім;
    b) Якщо користувач не вказав команду set2D(), то за замовчуванням буде використано set2D() без будь-яких аргументів на початку сесії користувача;
    c) Ця команда має 7 базових варіантів з наступними параметрами: set2D(); set2D(x0, x1); set2D(x0, x1, 'заголовок'); set2D(x0, x1, y0, y1); set2D(x0, x1, y0, y1, 'заголовок'); set2D(x0, x1, 'заголовок', 'назваOX', 'назваOY'); set2D(x0, x1, y0, y1, 'заголовок', 'назваOX', 'назваOY').

3.2 Побудова графіків явних функцій

Для отримання графіку явної функції f=f(x) використовується команда plot(f). Інші варіанти команд:

1) plot(f, [x0, x1]), де [x0, x1] — інтервал по вісі OX;
2) plot(f, [x0, x1], 'опція'), де [x0, x1] — інтервал по вісі OX, 'опція' може приймати значення 'dash', тоді графік буде пунктирною лінією;
3) plot([f, g]) – будує графіки функцій f, g (можна додати ще більше функцій).

Після початкового побудови графіку за допомогою текстового запиту, можна його ще більше кастомізувати за допомогою окремих елементів графічного інтерфейсу. Можливо зробити графік чорно-білим, зробити лінії сітки полотна графіку видимими, встановити розмір шрифта всього тексту на зображенні (назви осей, графіку, позначок чисел на осях), встановити товщину ліній графіку, змінити товщину осей. Нижче наведений приклад графіку явних функцій:

$SPACE = R64[x, y];$
$\mathbf{set2D}(-10, 10, -10, 10);$
$f = \mathbf{sin}(x);$
$p = \mathbf{plot}([f, \mathbf{tg}(x), \mathbf{cos}(x)]);$
$out:$

Plot Download B/W Grid Lines ↕A 15 — 4 ↑ 2

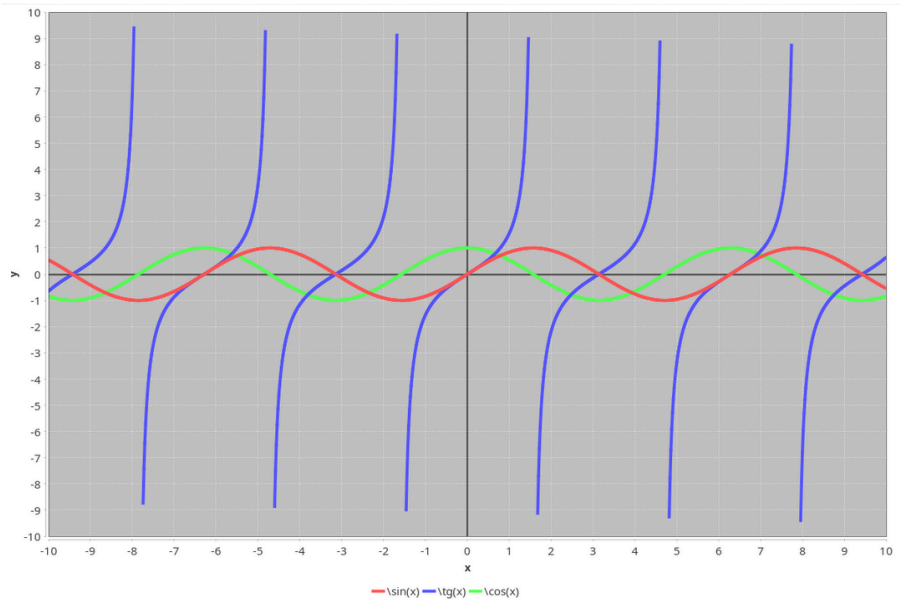


Рисунок 3.2.1 - Приклад побудови декількох явних функцій

3.3 Побудова графіків параметричних функцій

Команда paramPlot([f, g], [t0, t1]) використовується для створення графіку параметричної функції {f=x(t), g=y(t)}, де [t0,t1] - інтервал зміни параметра t. Щоб зробити лінії графіку пунктирними, можна використати варіант команди з додатковою опцією: paramPlot([f, g], [t0, t1], 'dash'). Після початкового побудови графіку за допомогою текстового запиту, можна його ще більше кастомізувати за допомогою окремих елементів графічного інтерфейсу, як вказано в минулому підрозділі. Нижче представлений приклад графіку параметричної функції:

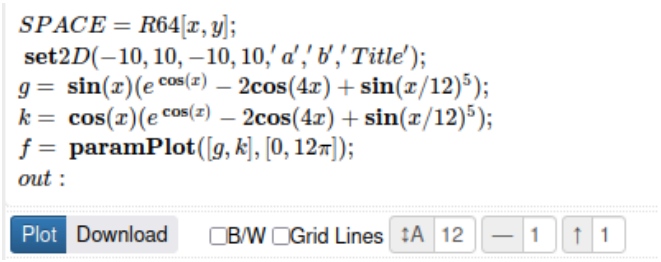


Рисунок 3.3.1 - Приклад побудови параметричної функції

3.4 Побудова графіків табличних функцій

Для побудови графіка функції, визначеної таблицею точок, ви можете використовувати команду tablePlot([[x₁,…,x□],[y₁₁,…,y₁□],…,[y□₁,…,y□□]]). Щоб зробити лінії графіку пунктирними, можна використати варіант команди з додатковою опцією: tablePlot([[x₁,…,x□],[y₁₁,…,y₁□],…,[y□₁,…,y□□]], 'dash'). Після початкового побудови графіку за допомогою текстового запиту, можна його ще більше кастомізувати за допомогою окремих елементів графічного інтерфейсу, як вказано в минулому підрозділі. Нижче наведений приклад графіку табличної функції:

$$SPACE = R64[x, y, z];$$
$$\mathbf{set2D}(-2, 6, -10, 30);$$
$$\mathbf{tablePlot}\left(\begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 \\ 0 & 1 & 4 & 9 & 16 & 25 \\ 0 & -1 & -2 & -3 & -4 & -5 \\ 0 & 4 & 8 & 12 & 16 & 20 \end{pmatrix}\right);$$
$$out:$$

Plot Download B/W Grid Lines ↕A 12 — 1 ↑ 1

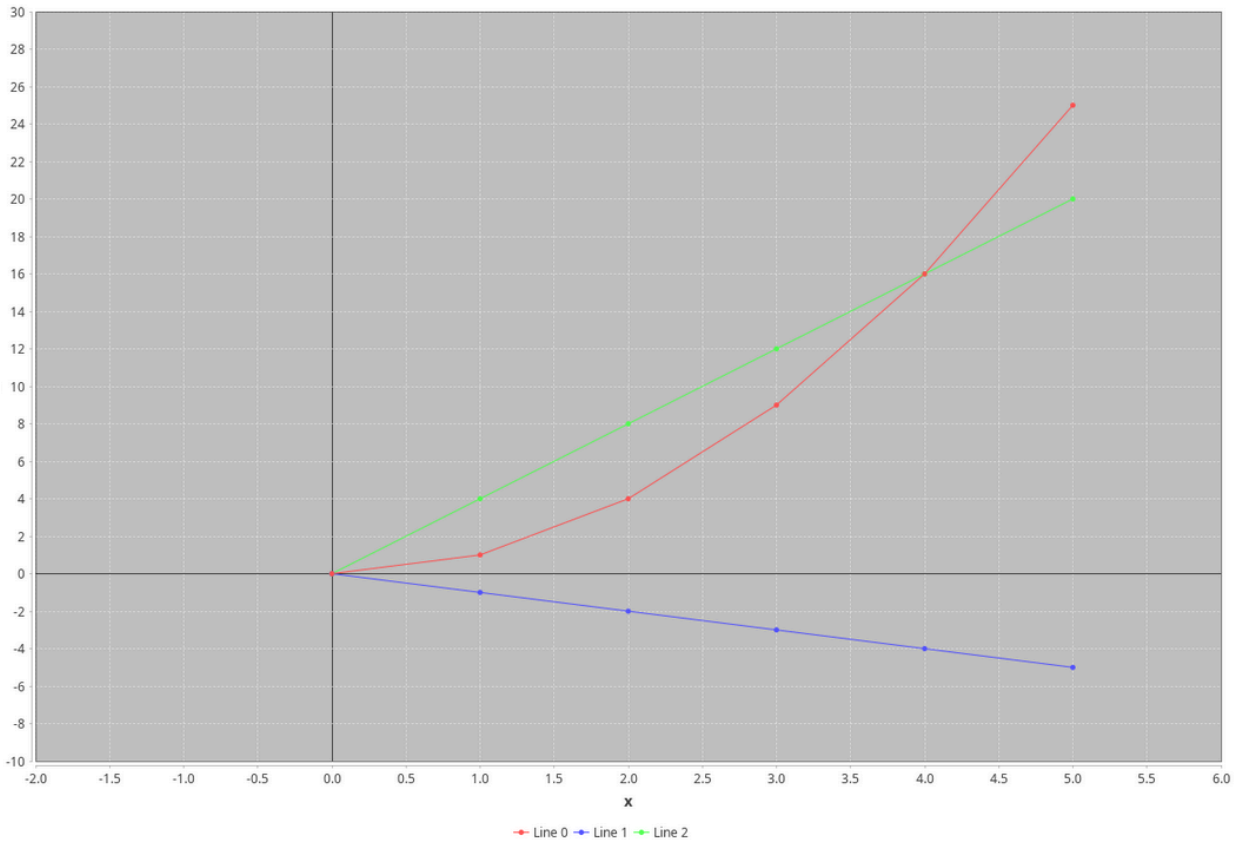


Рисунок 3.4 - Приклад побудови табличної функції

3.5 Анімація графіків при зміні параметрів

Для того, щоб зробити анімацію при зміні параметрів, необхідно відредагувати змінну простору SPACE та додати в кінець літери латинського алфавіту в алфавітному порядку, які і будуть означати параметри, наприклад

SPACE = R64[x, y, a, b, c] – x, y є дійсними змінними, в той час як a, b, c є параметрами.

Після редагування змінної простору SPACE, графічний інтерфейс користувача зміниться відповідно:

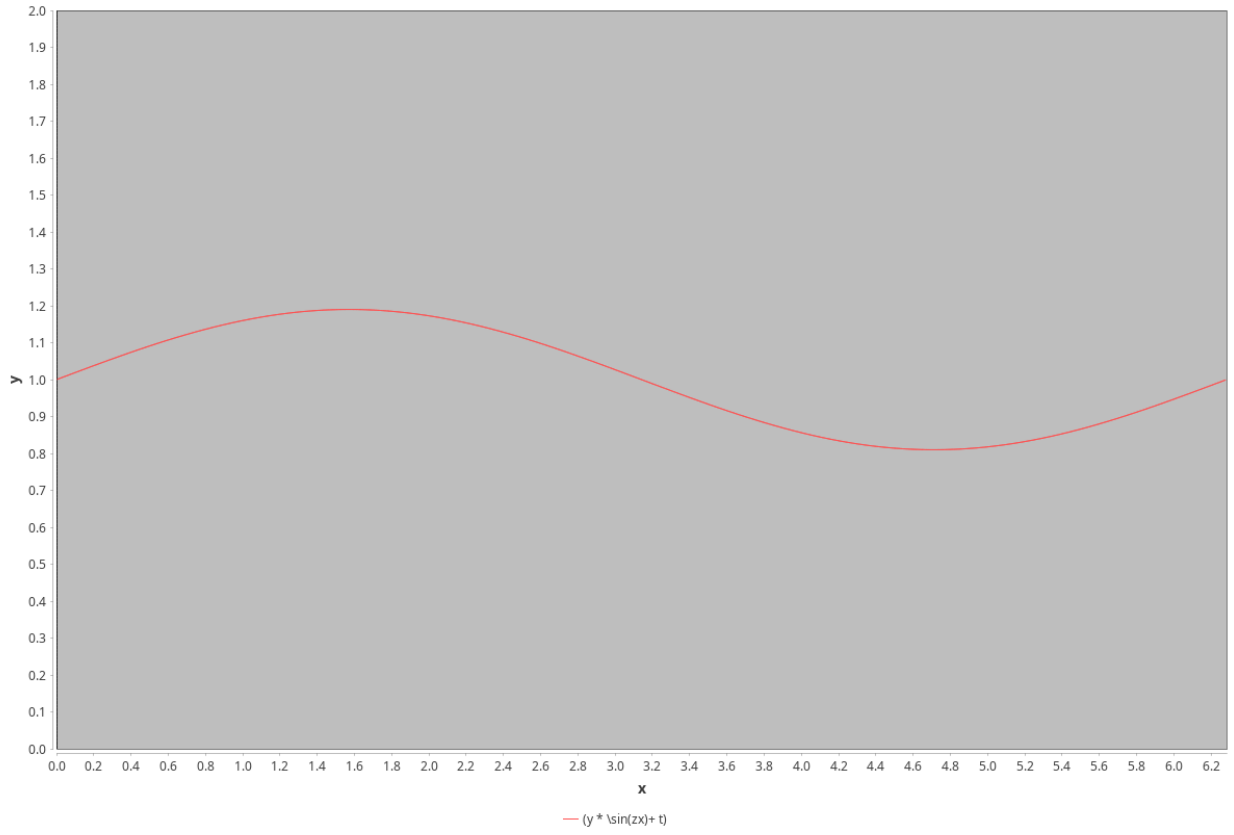
Рисунок 3.5.1 - Графічний інтерфейс користувача для керування анімацією – приклад 1

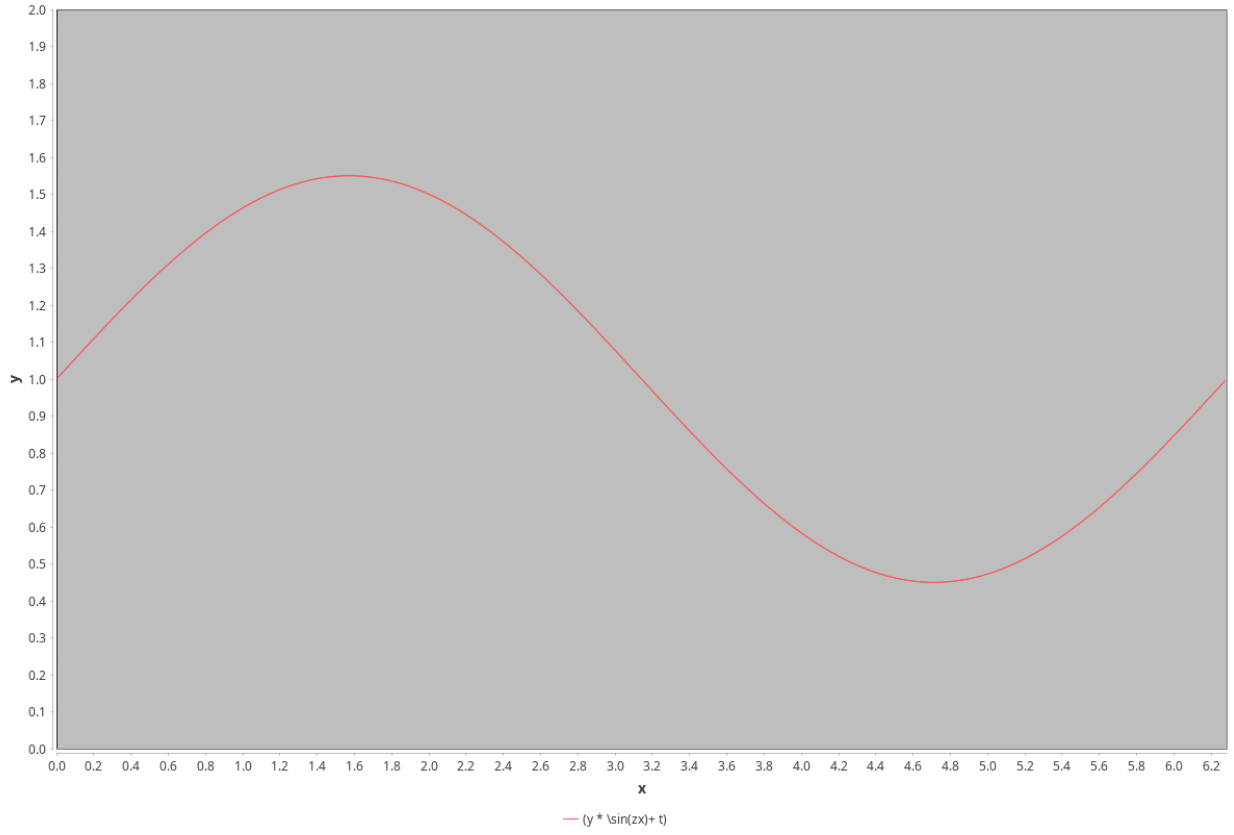
Бачимо, що з'явилось поле для визначення кількості кадрів вихідної анімації, а також кнопки для вибору одного з параметрів і повзунок, який визначає мінімальне значення параметра.

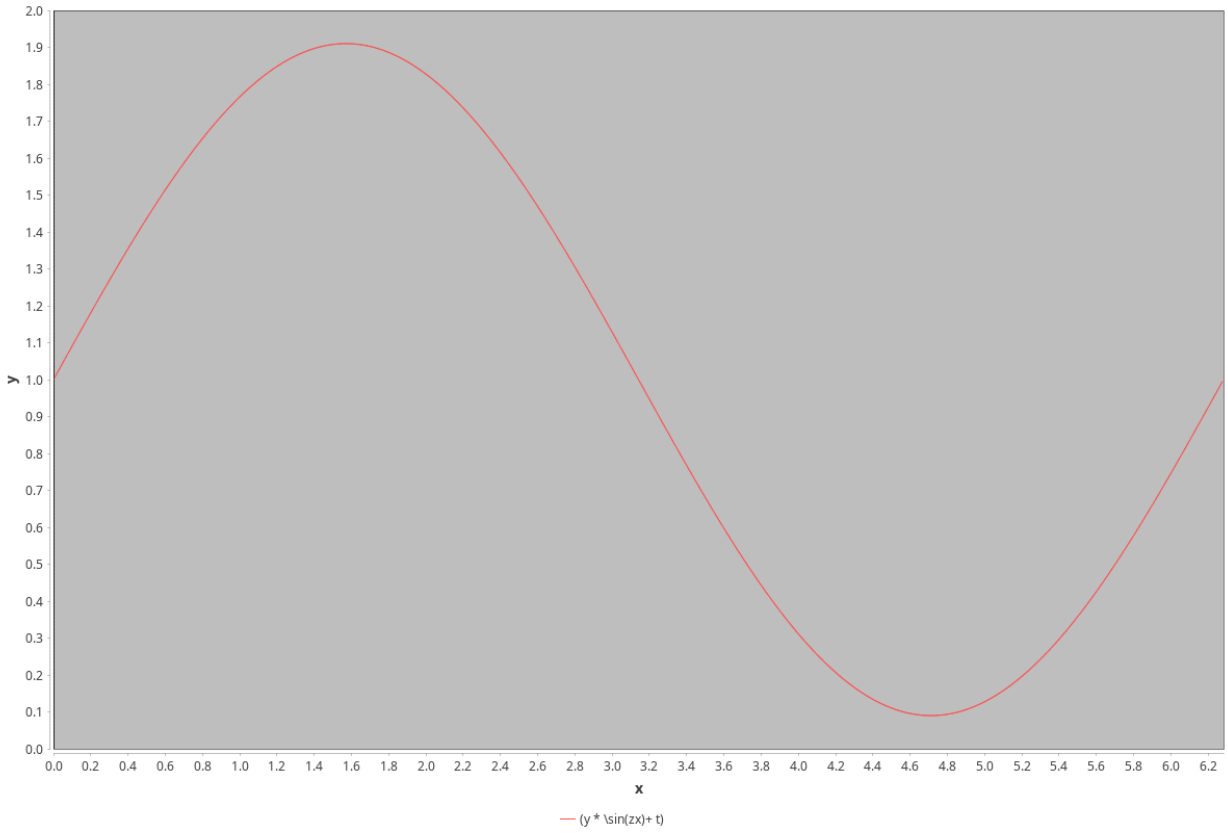
Наприклад, наступні задані опції створять анімацію, яка складатиметься з 30 кадрів, де на першому кадрі b матиме значення 0.4, з кожним кадром b буде поступово зростати і на останньому кадрі b буде дорівнювати 1.0:

Рисунок 3.5.2 - Графічний інтерфейс користувача для керування анімацією – приклад 2

Також нижче наведено приклад анімації лише з трьома кадрами, в якій значення параметра a змінюється з 0.1 до 1 впродовж цих трьох кадрів в функції *f(x) = a * sin (bx) + c*:


2.0
1.9
1.8
1.7
1.6
1.5
1.4
1.3
1.2
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
y
0.0 0.2 0.4 0.6 0.8 1.0 1.2 1.4 1.6 1.8 2.0 2.2 2.4 2.6 2.8 3.0 3.2 3.4 3.6 3.8 4.0 4.2 4.4 4.6 4.8 5.0 5.2 5.4 5.6 5.8 6.0 6.2
x
(y * \sin(zx)+ t)


2.0
1.9
1.8
1.7
1.6
1.5
1.4
1.3
1.2
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
y
0.0 0.2 0.4 0.6 0.8 1.0 1.2 1.4 1.6 1.8 2.0 2.2 2.4 2.6 2.8 3.0 3.2 3.4 3.6 3.8 4.0 4.2 4.4 4.6 4.8 5.0 5.2 5.4 5.6 5.8 6.0 6.2
x
(y * \sin(zx)+ t)



Рисунки 3.5.3 - Приклад трьох кадрів анімації

Висновки

У ході виконання курсової роботи було розроблено програму для побудови графіків табличних, параметричних та явних функцій на мові Java, використовуючи відкриту програмну бібліотеку JFreeChart. Цей функціонал був успішно інтегрований в програмне забезпечення MathPartner, що розробляється студентами Національного університету "Києво-Могилянська академія" під керівництвом професора Малашонка Геннадія Івановича.

Аналіз обраної бібліотеки дозволив визначити переваги використання JFreeChart. До цих переваг належать велика кількість можливостей налаштування та кастомізації графіків та діаграм, надійна документація та підтримка спільноти, надійність, актуальність, стабільність та універсальність. Розроблена програма дозволяє кастомізацію графіків та їх анімацію при зміні параметрів, що підвищує його користувацьку привабливість та функціональність.

Однією з цілей курсової роботи було виправлення недоліків попереднього функціоналу MathPartner, пов'язаного з побудовою графіків, шляхом використання структурованого коду на основі принципів об'єктно-орієнтованого програмування. Це дозволить полегшити подальше розширення та підтримку програмного забезпечення.

Отже, результатом курсової роботи є успішне впровадження нового функціоналу для побудови графіків у програмне забезпечення MathPartner з використанням відкритої бібліотеки JFreeChart, що покращує користувацький досвід та забезпечує більш ефективну підтримку та розвиток програми.

Список літератури

1. Malaschonok, G.I. "MathPartner Computer Algebra." Programming and Computer Software 43, no. 2 (2017): 112–118.

2. Bangerth, Wolfgang, and Timo Heister. "Comput. Sci. Discov." 6, no. 015010 (2013): 5-6.

3. Aggarwal, D., and Naveeta, M. "Software Reuse: A Compendium." International Journal of Research in IT & Management 2, no. 2 (February 2012): 93–100.

4. Lee, Y., J. Yang, and K. H. Chang. "Metrics and Evolution in Open Source Software." In Seventh International Conference on Quality Software (QSIC 2007), 191-197. Portland, OR, USA: IEEE, 2007. doi: 10.1109/QSIC.2007.4385495.

5. "JFreeChart." Accessed May 5, 2024. https://www.jfree.org/jfreechart/.

6. Höher, Alexander (Sascha). "Overloading Considered Harmful." Java Specialists' Newsletter, June 1, 2003. Accessed May 5, 2024. https://www.javaspecialists.eu/archive/Issue071-Overloading-Considered-Harmful.html.

7. GeeksforGeeks. "Polymorphism in Java." GeeksforGeeks, November 1, 2023. Accessed May 5, 2024. https://www.geeksforgeeks.org/polymorphism-in-java/.