Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра інформатики факультету інформатики

КУРСОВА РОБОТА

освітній ступінь – бакалавр
на тему**: Побудова 2D графіків табличних, параметричних та явних функцій на фронтенді у Spring проекті**

Студентки 3-го курсу
спеціальності 122 «Комп'ютерні науки»
Плахотної Д.О.
Керівник Малашонок Г.І.

Кількість балів: _____

Члени комісії:

_____ _____
(підпис) (прізвище та ініціали)

_____ _____
(підпис) (прізвище та ініціали)

_____ _____
(підпис) (прізвище та ініціали)

Київ 2024

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра інформатики факультету інформатики

ЗАТВЕРДЖУЮ
проф.
_____ Малашонок Г.І.
„_____” _____ 2024 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ
на курсову роботу

студентці Плахотній Дар’ї Олександрівні
факультету інформатики 3 курсу бакалаврської програми

ТЕМА: **Побудова 2D графіків табличних, параметричних та явних функцій на фронтенді у Spring проекті.**

Зміст ТЧ до курсової роботи:

Анотація
Вступ
Розділ 1. Теоретичні основи побудови графіків у MathPartner
Розділ 2. Реалізація побудови графіків у MathPartner
Висновки
Список використаних джерел

Дата видачі „___” _____ 2024 р.
Керівник_____
(підпис)
Завдання отримав_____
(підпис)

Календарний план виконання роботи

№	Назва етапу курсової роботи	Термін виконання етапу	Примітка
1.	Отримання завдання на курсову роботу	01.11.2023	
2.	Огляд літератури за темою роботи	05.12.2023	
3.	Проведення дослідження	01.01.2024	
4.	Написання програмного застосунку	01.03.2024	
5.	Написання текстової частини	04.04.2024	
6.	Захист курсової роботи	22.05.2024	

Студентка _____Плахотна Д.О._____
Керівник _____Малашонок Г.І._____ “_____”___ 2024

Київ 2024

ЗМІСТ

АНОТАЦІЯ

У даній курсовій роботі розглядається розширення функціоналу веб-додатку MathPartner, реалізованого на платформі Spring, шляхом оптимізації процесу відображення 2D графіків, що задаються таблично, параметрично або явно. Раніше система генерувала і ілюструвала графіки за допомогою Java-класів на сервері, що призводило до затримок у відгуку та обмеженні функціональності клієнтської частини системи. Удосконалення полягає в переміщенні процесу відображення графіків на фронтенд за допомогою бібліотеки two.js. Цей підхід значно покращує швидкість відображення графіків і зменшує навантаження на сервер. Результати демонструють поліпшену ефективність взаємодії системи побудови графіків функції з користувачами.

ВСТУП

У сучасному світі розробка програмного забезпечення для математичного моделювання відіграє важливу роль у вирішенні різноманітних наукових, інженерних та освітніх завдань, оскільки це допомагає заощадити час, зменшити кількість потенційних помилок, викликаних людським фактором, та представити інформацію у зручному форматі. Однією з ключових особливостей математичного програмного забезпечення є можливість будувати графіки різних складних функцій, тим самим візуалізуючи дані та спрощуючи процес аналізу. Веб-додаток MathPartner є потужним математичним помічником, який містить широкий спектр функціоналу обчислювальних процесів. Однією з таким можливостей є забезпечення ефективного графічного представлення математичних функцій.

Сучасний стан проблеми полягає у необхідності забезпечення високої продуктивності MathPartner та оптимізації ресурсів сервера при роботі з математичними додатками. Традиційний підхід, за яким графіки було побудовано та графічно представлено повністю на сервері, призводить до зайвого навантаження і зменшує швидкість відгуку системи. Така проблема виявляється особливо актуальною при роботі з великими даними та вимагає вдосконалення архітектури вже існуючих рішень. Крім того, графіки, згенеровані сервером, не дозволяють користувачам динамічно змінювати їх параметри, що обмежує взаємодію з візуалізацією. Підтримка інтерактивності надає можливість не тільки бачити результати своїх обчислень у реальному часі, але й миттєво адаптувати моделі під умови або параметри, що змінюються, що значно покращує процес дослідження та навчання. Включення елементів інтерактивності, таких як масштабування, переміщення за графіком, а також зміна параметрів функцій, робить процес навчання більш ефективним та цікавим для студентів усіх рівнів.

Задача даної роботи полягає в розробці оптимізованого рішення, яке дозволить перенести відображення графіків на клієнтську сторону, використовуючи JavaScript-бібліотеку two.js, зберігаючи при цьому побудову

графіків на сервері. Такий підхід знижує навантаження на сервер і забезпечує швидшу інтерактивну взаємодію з користувачами, і в результаті підвищуючи загальну продуктивність системи.

Новизна теми курсової роботи полягає у впровадженні фронтенд-компонента для відображення математичних графіків, що розширює можливості веб-додатку MathPartner і дозволяє краще використовувати сучасні технології візуалізації даних.

Важливість розробки виявляється у широкому спектрі застосувань: від освітніх інститутів, які використовують такі інструменти для навчання студентів, до наукових лабораторій, які застосовують візуалізацію для аналізу складних математичних моделей. Це робить роботу не тільки актуальною, але й важливою для подальшого розвитку в галузі програмного забезпечення для математичного моделювання.

РОЗДІЛ 1. Теоретичні основи побудови графіків у MathPartner

Огляд побудови графіків у MathPartner дозволяє зрозуміти основну концепцію програми, гнучкість та багатофункціональність системи, а також важливість правильного настроювання параметрів для досягнення бажаного візуального результату.

1.1. Переваги використання системи MathPartner

MathPartner - система математичних обчислень та візуалізації даних, якиц пропонує унікальний підхід до вирішення широкого спектру завдань, від освітніх до професійних дослідницьких проектів. MathPartner надає ряд переваг, які роблять його потужним інструментом для вирішення широкого спектру математичних завдань. У контексті побудови графіків основні переваги MathPartner включають:

1. Можливість побудови графіків різних типів функцій, від простих до складних, та функцій, що задаються різними способами.
2. Платформа підтримує одночасне відображення графіків на одній координатній площині, спрощуючи процес порівняння функцій та аналіз їх взаємодії.
3. Просунуте налаштування MathPartner дозволяє тонко налаштовувати візуальну частину графіків відповідно до вподобань користувача або вимог до публікації.
4. Команди для побудови графіків лаконічні та інтуїтивно зрозумілі, що є доступним механізмом для користувачів без глибоких знань у програмуванні або математичному моделюванні.
5. MathPartner підтримує можливість збереження графіків у різних форматах, включаючи зображення та PDF-файли.

1.2. Порівняння системи MathPartner з іншими інструментами

MathPar використовує мову TeX для набору математичних формул, забезпечуючи високу точність та зручність під час роботи з математичними текстами. Можливість роботи з суперкомп'ютерами робить його придатним для складних та масштабних завдань, де потрібна висока продуктивність обчислень. Це робить MathPartner особливо цінним інструментом в освітніх та дослідницьких сферах, де часто потрібна комбінація доступності, потужності та гнучкості. Він є простий в освоєнні для початківців, особливо для тих, хто вже знайомий з TeX. Посібник користувача пропонує покрокове введення в основні та просунуті функції.

Інші інструменти математичного аналізу, такі як MATLAB, Python та R вимагають більш серйозної підготовки і розуміння програмування або статистики для ефективного використання. Таким чином MathPartner є кращим вибором для більш доступного, з можливістю простого налаштування та спеціалізованого інструменту для математичних обчислень та візуалізації даних.

1.3. Побудова графіків функції на площині

Побудова графіків - це ключовий метод візуалізації математичних функцій, що дозволяє аналізувати поведінку функцій та їх властивості. Графіки можуть бути представлені в різних формах, залежно від типу функції та цілей дослідження.

MathPartner надає функціональність для побудови графіків функцій, що задаються явно (plot), за допомогою таблиці (tablePlot) та параметрично (paramPlot). Також можливе одночасне представлення на малюнку кількох графіків в одній координатній системі, які задаються різними способами (showPlots).

Налаштування середовища для побудови графіків на площині здійснюється за допомогою команди set2D(). У разі відсутності параметрів

команди set2D() межі для графіків буде розраховано автоматично, і для функцій заданих явно встановлюється інтервал [0,1] по осі абсцис. Якщо користувач не задає команду set2D(), то функція set2D() застосовується автоматично без параметрів. Це забезпечує початкове налаштування координатної площини, гарантуючи, що користувач отримає візуальний результат без необхідності попереднього налаштування параметрів.

Також існує два формати команди set2D(): повний та скорочений. Повний формат включає вимагає три групи параметрів, кожен з яких задається у квадратних дужках, розділених комою. У перших квадратних дужках вказуються обов’язкові межі графіка по осі абсцис та ординат у вигляді [x0, x1, y0, y1]. У других квадратних дужках вказуються написи до осей координат і підпис до всього малюнку у вигляді ['xTitle','yTitle','title']. Треті останні дужки містять п’ять чисел, які являють собою установки для режиму відображення: чорно-білий режим, встановлення рівного масштабу для обох осей, розмір шрифту для підписів, товщини ліній графіків і координатних осей. Також передбачені скорочені варіанти швидкого налаштування основних параметрів: ['ES'], ['BW'], ['ESBW']. Вони відповідно встановлюють у значення або перший параметр, або другий, або обидва.

Команда set2D() має сім скорочених варіантів, що дозволяють встановити різні комбінації параметрів від простої установки інтервалу по осі абсцис до повного налаштування всіх параметрів, включаючи підписи і заголовки.

Лінії на графіках функції можуть відображатися різним чином: суцільними, пунктирними або лінії, що закінчуються стрілками. Даний функціонал задається за допомогою параметрів dash, arrow і dashAndArrow.

1.3.1. Явно задані функції

Графіки функцій, що задаються явно, є одним із основних та найпростіших способів подання математичних залежностей, де значення

однієї змінної безпосередньо виражається через значення іншої. У класичному вигляді таке відношення представлено у формі y = f(x), де x – це незалежна змінна, а y – залежна. Прикладами явно заданих функцій є поліноміальні функції високого порядку, раціональні функції, тригонометричні поліноми та експоненційно-тригонометричні функції.

Явне задання функції один із основних методів побудови графіків у системі MathPartner. Для візуального представлення функції f = f(x) використовується команда plot(f). Дана команда підтримує кілька варіантів використання, які дозволяють налаштувати відображення графіка з урахуванням специфічних вимог:

1. plot(f, [x0, x1]) - тут [x0, x1] вказує інтервал по осі OX, на якому буде побудовано графік.
2. plot(f, [x0, x1], 'options') — у цьому випадку, крім вказівки інтервалу по осі OX, можна встановити додаткові параметри відображення графіка:
   2.1. 'dash' - графік буде відображено пунктирною лінією;
   2.2. 'arrow' - на останній точці графіка буде відображена стрілка;
   2.3. 'dashAndArrow' - поєднання пунктирної лінії зі стрілкою на останній точці.
3. plot(f, 'options') — дозволяє встановити лише параметри відображення графіка без зміни інтервалу по осі OX.

Також можлива побудова графіків функцій, що містять параметри. Ці параметри мають бути попередньо визначені як змінні у налаштуваннях оточення. Параметри можуть набувати значення з інтервалу [0; 1]. За замовчанням графік будується для значень параметрів, рівних одиниці, але ці значення можна змінювати.

Дані можливості побудови та налаштування графіків забезпечують гнучкість та зручність у візуалізації математичних функцій, дозволяючи користувачам адаптувати процес під свої конкретні завдання та переваги.

1.3.2. Параметрично задані функції

Функції, що задаються параметрично, є важливим класом математичних функцій, що дозволяють описувати складні геометричні та фізичні явища. Параметрично задані функції використовують один чи кілька додаткових параметрів визначення координат точок на площині. У двовимірному випадку параметричне рівняння кривої задається парою функцій:

$$x = f(t)$$

$$y = g(t)$$

де t — параметр, який змінюється в певному заданому інтервалі $t \in [a, b]$. Функції f та g визначають положення точки на графіку для кожного значення t.

Прикладом параметрично заданої кривої може бути коло, де $x = r\cos t$ і $y = r\sin t$, а t змінюється від 0 до 2π. Це дозволяє описати положення точок на колі з радіусом r, де t відповідає куту в полярних координатах.

Для побудови графіків таких функцій у системі MathPartner використовується команда paramPlot([f, g], [t0, t1]), де f = x(t) та g = y(t) є параметрично заданими функціями, а [t0, t1] вказує інтервал значень параметра t, у якого будується графік.

Крім стандартного способу побудови, доступний розширений варіант команди: paramPlot([f, g], [t0, t1], 'options'), де [t0, t1] залишається інтервалом значень для параметра t, а 'options' дозволяє налаштувати візуальне уявлення графіка. Варіанти значення 'options' включають:

1. 'dash' - графік буде відображено пунктирною лінією;
2. 'arrow' - на останній точці графіка буде відображена стрілка;
3. 'dashAndArrow' - поєднання пунктирної лінії зі стрілкою на останній точці.

Дані налаштування дозволяють користувачеві наочніше уявити динаміку змін, що описуються параметричними функціями.

1.3.3. Таблично задані функції

Графік функції, що задається таблицею, є візуалізацією даних, де кожна точка на графіку відповідає парі значень, зазначених у таблиці. Дані таблиці містять два або більше стовпців: один для значень незалежної змінної (позначається як x), і один або кілька стовпців для залежної однієї або багатьох змінних (позначаються як y). Кожен рядок таблиці являє собою одну точку на графіку, де перший стовпець задає координату горизонтальної осі, а наступні стовпці - відповідні координати вертикальної осі.

Для побудови графіків функцій на основі табличних даних MathPartner користувач може використати команду tablePlot, яка дозволяє візуалізувати дані, задані у вигляді пар значень. Основна форма команди виглядає так:

tablePlot([[x{1}, …, x{n}], [y{11}, …, y{1n}], …, [y{k1}, …, y{kn}]]))

Тут x{1}, …, x{n} є значення абсцис, а кожен список виду [y{k1}, …, y{kn}] – це значення ординат для різних наборів даних, які будуть відображені на одному графіку.

Додатково команда tablePlot може містити параметр 'options', який дозволяє налаштувати візуальне подання графіка. Розширений варіант команди представляється так:

tablePlot([[x_{1}, …, x_{n}], [y_{11}, …, y_{1n}], …, [y_{k1}, …, y_{kn}]], 'options ')

Значення для 'options' можуть бути наступними:

1. 'dash' - графік буде відображено пунктирною лінією;
2. 'arrow' — на останній точці кожного набору даних буде відображено стрілку;
3. 'dashAndArrow' - графік буде зображений пунктирною лінією, і на останній точці кожного набору даних буде відображено стрілку.

РОЗДІЛ 2. Реалізація побудови графіків у MathPartner

У Розділі 2 розглядається практична реалізація процесу побудови графіків у системі MathPartner. Цей розділ призначений для детального аналізу архітектури системи, зокрема взаємодії між серверною та клієнтською сторонами, а також механізмів передачі даних від сервера до клієнта.

Серверна частина системи MathPartner написана на мові програмування Java. Вона відповідає за виконання математичних обрахунків, необхідних для побудови графіків, та обробку вхідних даних від користувача. Основна функція сервера полягає у розрахунку точок для графіків, згідно з математичними формулами, заданими користувачем. Після визначення необхідних точок, сервер формує масив цих точок, який буде передано клієнтській стороні для візуалізації.

Клієнтська сторона розроблена з використанням бібліотеки two.js, що дозволяє ефективно візуалізувати отримані від сервера дані у вигляді графіків. Two.js забезпечує гнучкість у створенні та налаштуванні графічних представлень, таких як масштабування, зміна кольорів ліній та інші візуальні параметри, які можуть бути адаптовані під конкретні потреби користувача. Клієнтська частина відповідає за отримання масиву точок з сервера та їх відображення у веб-інтерфейсі, забезпечуючи користувачам інтуїтивно зрозумілий і візуально привабливий доступ до результатів їхніх обрахунків.

2.1. Потік інформації між функціональними частинами програми

Було побудовано архітектуру взаємодії серверної та клієнтської частин системи MathPartner з метою оптимізації процесів обчислення та візуалізації математичних графіків. На Рисунку 2.1 показано, як саме відбувається передача даних між сервером, що обробляє математичні вирази і генерує координати точок для графіків, та клієнтською стороною, яка відповідає за їх візуалізацію.

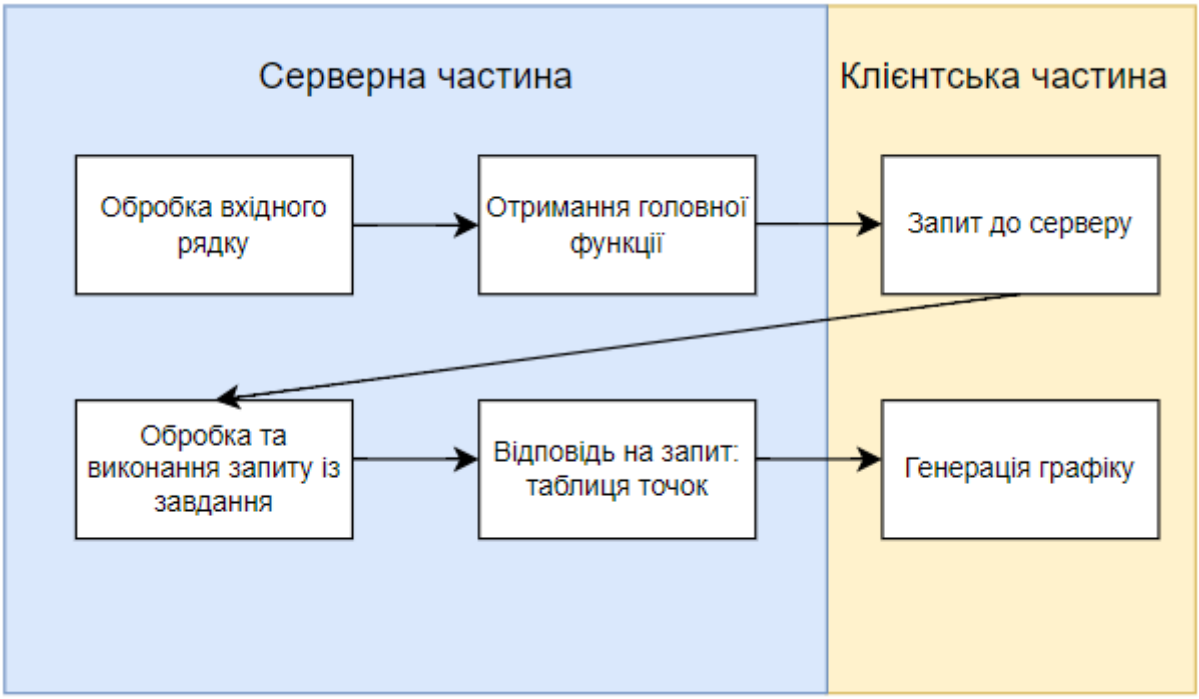

Рисунок 2.1 Потік інформації між функціональними частинами програми.

Ініціація процесу побудови графіка відбувається, коли користувач вводить математичний вираз у спеціальне текстове поле на HTML-сторінці і натискає кнопку «Побудувати». Відбувається активація збору даних для обробки сервером. Сервер приймає рядок із завданням і передає вираз у головну функцію обробки, яка відповідає за аналіз та виконання математичних операцій. Дана функція повертає результат роботи, розбиваючи завдання на окремі функції. На Рисунку 2.2 показано UML-діаграму класів для побудови параметрично заданої функції на серверній частині.

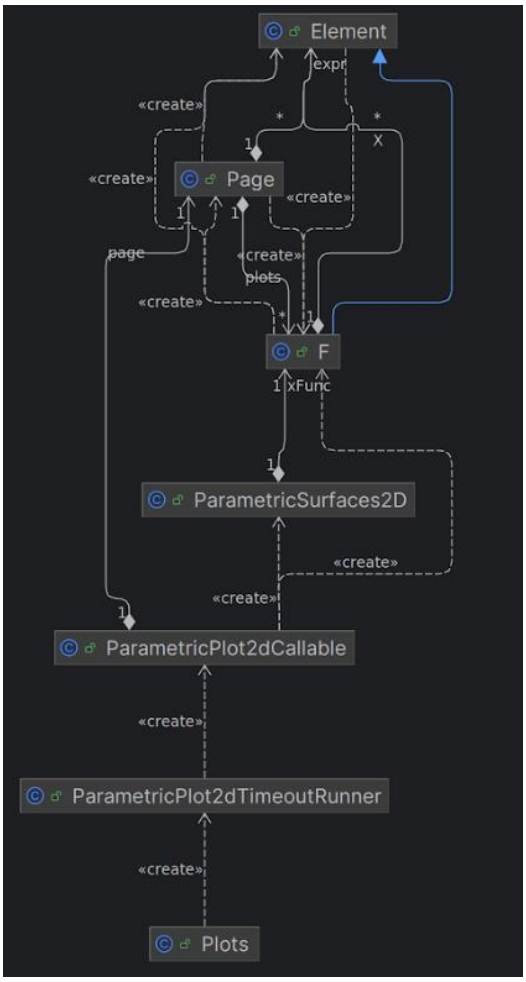

Рисунок 2.2 UML-діаграма класів

Клієнтська частина робить запит до серверу через HTTP-запит, який контролер розпізнає на стороні серверу, визначає тип завдання (наприклад, побудова графіка явно заданої, параметричної чи табличної функції) та виконує необхідні обчислення. В результаті формується масив точок, які відповідають координатам графіка.

Після обробки та генерації масиву точок сервер відправляє ці дані назад на клієнтську сторону. Це відбувається через HTTP-відповідь, який включає масив точок у форматі JSON, придатному для передачі та подальшої обробки.

Клієнтська частина системи отримує дані та починає їх візуалізацію. Бібліотека Two.js обробляє масив точок, створюючи на їх основі графік. Клієнтський скрипт керує такими параметрами відображення, як масштаб,

кольори та стилі ліній, забезпечуючи гнучкість налаштування візуального представлення графіків.

2.2. Реалізація AJAX-запитів

Для побудови графіків функції без необхідності перезавантажувати сторінку було прийнято рішення використовувати технології AJAX-запитів. Це дозволили здійснювати асинхронне отримання даних сервером тільки тієї частини документу, яка змінилася, а саме побудовані графіки за запитом користувача.

У тіло запитів /api/plot2dexplicit, /api/plot2dparametric та /api/plot2dtable передається завдання для візуалізації графіку та номер віконця (Рисунок 2.3). Ці AJAX-запити обробляються на сервері відповідними контролерами. Спочатку перевіряється, чи містить заданий користувачем вираз необхідну функцію. Якщо перевірку пройдено успішно, в окремому потоці запускається виконання завдання. Після завершення обчислень сервер формує відповідь, що містить масив пар геометричних точок необхідних для побудови графіка.

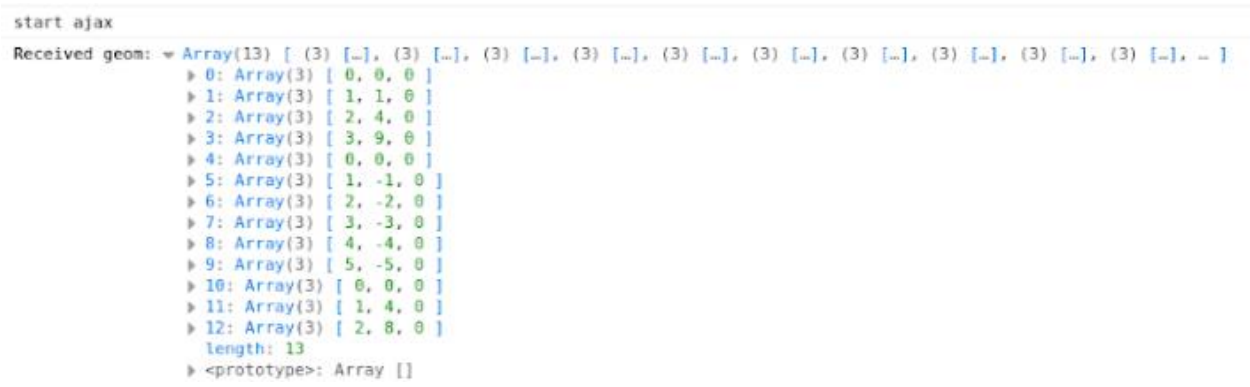

Рисунок 2.3 Приклад отриманих вершин для табличної функції

2.3. Реалізація контролерів

У проєкті MathPartner створено контролери для обробки запитів, організовані за допомогою анотацій та властивостей, які надає Spring Framework. Це забезпечує ефективне маршрутизування запитів, обробку помилок, інтеграцію з сервісами та управління залежностями.

Запити на створення графіків обробляються асинхронно за допомогою ExecutorService. Це дозволяє управляти тривалими або ресурсомісткими завданнями без блокування основного потоку сервера. Для прикладу програмного коду контролеру див. Додаток А.

2.4. Алгоритм побудови графіків

Для побудови графіків функції на серверній частині було використано різні алгоритми, базуючись на специфіці вхідних даних. Основна логіка полягає в отриманні усієї необхідної інформації (діапазону значень, кільця, функції та опцій для налаштування) з класу, який обробляє завдання. Під час обробки виразу використовується процес підстановки функції в іншу функцію, що гарантує на виході кінцевий спрощений вираз.

Для явно заданих функцій задіяно простий підхід, де для заданого діапазону значень x обчислюється відповідне значення y. Цей метод використовується для функцій, що мають пряму залежність y від x. Результат представлено у вигляді лінійного графіка, де кожна точка з’єднана з наступною (Рисунок 2.4).

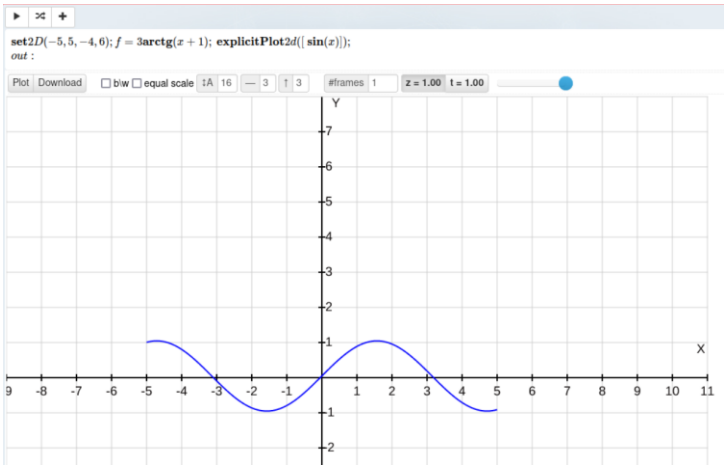

Рисунок 2.4 Побудова явно заданої функції

У даному прикладі спочатку знаходяться точки графіку з урахуванням діапазону, який задається у функції *set2D(-5, 5, -4, 6)*. Тому на клієнтській стороні немає необхідності враховувати межі графіку.

Було поставлено задачу розробити більш просунутий рівень задля розширення функціоналу MathPartner. Ідея полягала в можливості передачі сервером точок декількох функцій. Для реалізації зображення багатьох графіків на одному малюнку було задіяно логіку роздільника. Після кожного набору точок для графіку використовується порожній елемент масиву. Це гарантує розмежовування кожної групи точок для графіків, що спрощує відображення на клієнтській частині (Рисунок 2.5).

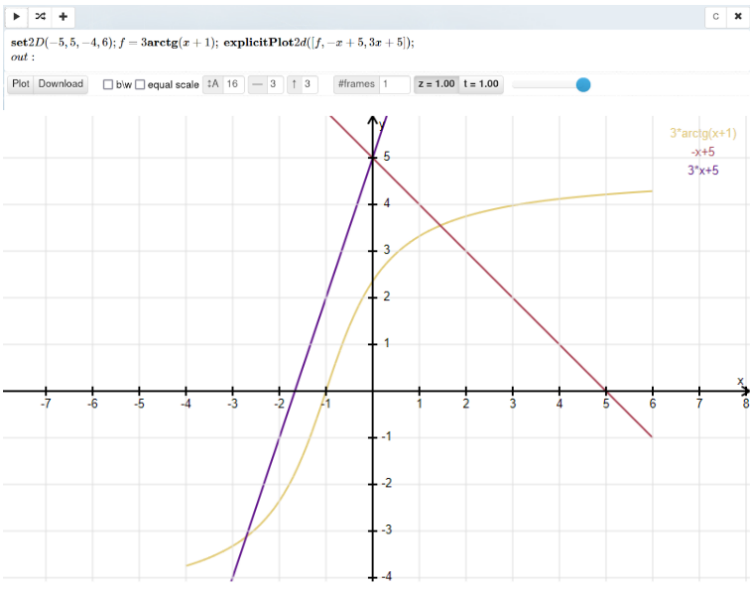

Рисунок 2.5 Побудова багатьох графіків

Використаний алгоритм для отримання точок:

Вхідні дані:

функція f(x)

x_min: мінімальне значення x

x_max: максимальне значення x

кількість_точок: бажана кількість точок на графіку

Процес:

крок = (x_max - x_min) / кількість_точок

x = x_min

Поки x <= x_max:

y = f(x) // Обчислення значення y для поточного x за допомогою функції

Додати точку (x, y) до списку точок

x = x + крок // Перехід до наступної точки x

Вивід:

Список точок (x, y), які можна використовувати для побудови графіку

Параметричну побудову графіків використовують для відображення відношення між двома або більше змінними, які залежать від одного або декількох незалежних параметрів t. Для значень параметра t з заданого діапазону обчислено відповідні координати x та y, використовуючи окремі функції для кожної координати. Графік формується шляхом з'єднання послідовних точок $(x(t), y(t))$, що дає змогу відобразити більш складні криві (Рисунок 2.6).

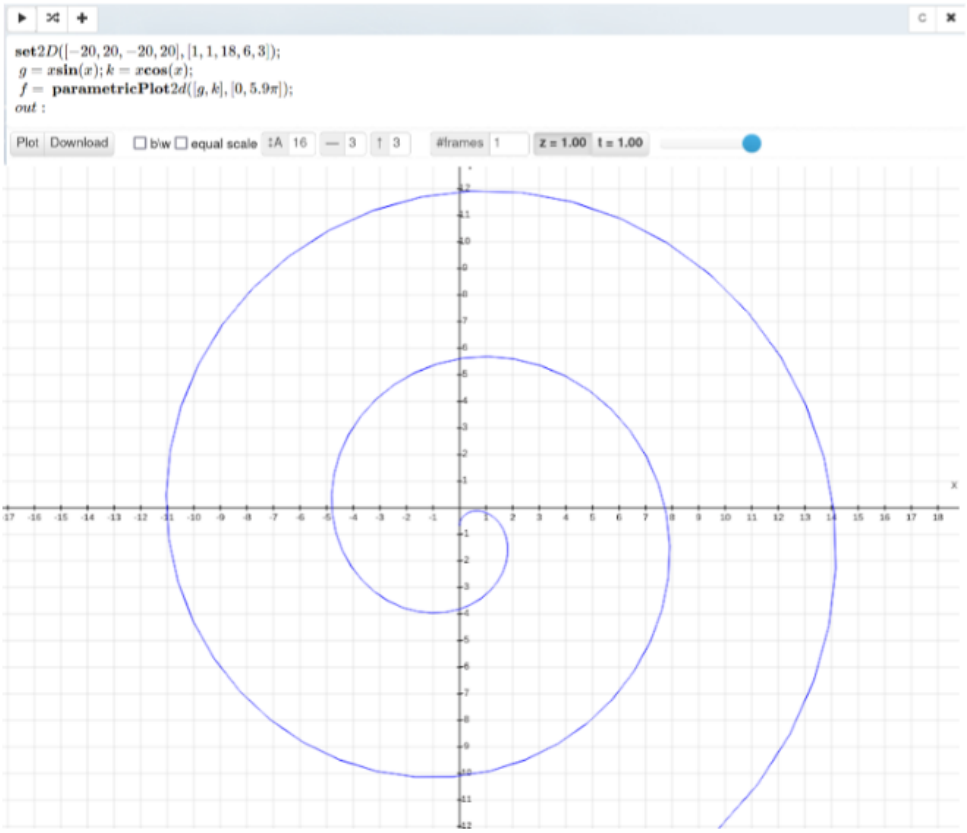

Рисунок 2.6 Побудова параметричного графіку

Використаний алгоритм для отримання точок:

Вхідні дані:

функція x(u): функція для обчислення x від параметра u
функція y(u): функція для обчислення y від того ж параметра u

u_min: мінімальне значення параметра u
u_max: максимальне значення параметра u
кількість_точок: бажана кількість точок на графіку

Процес:
крок = (u_max - u_min) / кількість_точок
u = u_min
Поки u <= u_max:
x = x(u) // Обчислення значення x для поточного u
y = y(u) // Обчислення значення y для того ж u
Додати точку (x, y) до списку точок
u = u + крок // Перехід до наступного значення параметра u

Вивід:
Список точок (x, y), які можна використовувати для побудови графіку

Табличний метод використовується для побудови графіків з дискретних наборів даних, де кожна точка задається явно (Рисунок 2.7). Під час обробки завдання програма зчитує перший масив елементів x зіставляючи відповідні значення y з кожного наступного масиву. Такий підхід дозволяє швидко та просто візуалізувати графік за вхідним набором таблиці. У вихідному списку точок міститься інформація про координату x, y та порядкового номеру графіку функції для розмежування різними кольорами кожного графіку. Командою *set2D(xMin, xMax, yMin, yMax)* задаються межі.

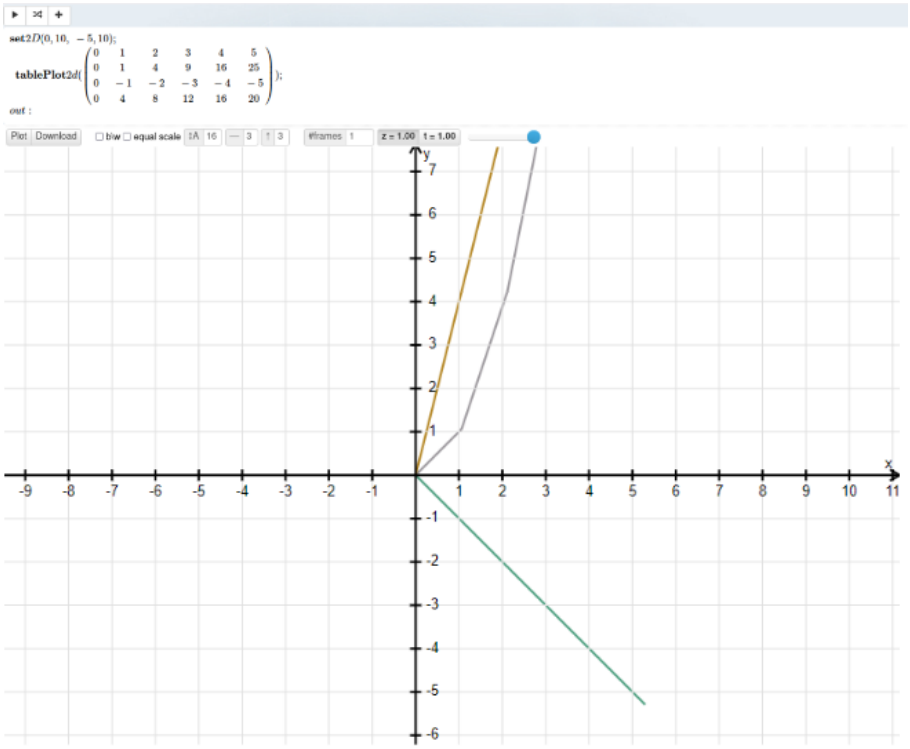

Рисунок 2.7 Побудова табличної функції

Використаний алгоритм для отримання точок:

Вхідні дані:

tablePlot2d: об'єкт, що містить табличні дані (x та y значення для кожного ряду)

x_min, x_max: межі відображення по осі X

y_min, y_max: межі відображення по осі Y

Процес:

Ініціалізація порожнього списку точки.

Для кожного набору даних у tablePlot2d:

Отримати масиви xValues і yValues з поточного набору

Якщо довжини xValues і yValues однакові:

Для і від 0 до довжини xValues:

Якщо xValues[i] в межах x_min і x_max і yValues[i] в межах y_min і y_max:

Додати точку (xValues[i], yValues[i]) до списку точки

Повернути список точки

Вивід:

Список точок (x, y), які можна використовувати для візуалізації табличних даних у вигляді графіка.

2.5. Реалізація користувацького інтерфейсу

Для реалізації користувацького інтерфейсу було використано бібліотеку Two.js, яка є двовимірним графічним інтерфейсом, розробленим для сучасних веб-браузерів. Причиною обраної бібліотеки є можливість швидкого рендерингу, що забезпечує плавність анімацій та інтерактивність інтерфейсу навіть при великих обсягах даних. Two.js має простий та інтуїтивно зрозумілий API, що істотно спрощує процес розробки. Бібліотека сумісна з більшістю сучасних браузерів, забезпечуючи однакове відображення графіків на різних платформах та пристроях.

Також Two.js дозволяє створювати двовимірну графіку за допомогою SVG, WebGL, і Canvas. Така гнучкість робить two.js чудовим вибором для створення динамічних та інтерактивних графіків, які потрібні в сучасних освітніх та дослідницьких додатках. Це було особливо корисним для відображення параметричних графіків, які потребують динамічного рендерингу в залежності від вхідних даних від користувача.

Бібліотека Two.js надає широкий спектр можливостей для створення і маніпулювання графічними об’єктами. Було використано ключові компоненти для побудови графіку, а саме: головний контейнер, що визначається розміром та типом елементів, які буде відображено на сторінці; лінії та стрілки для побудови сітки, координатних осей та інших лінійних елементів графіку; текстові мітки для позначення одиничного підрізку та назви осей; масштабування, що впливає на розмір та розташування всіх графічних об’єктів.

Для реалізації суцільної лінії на основі списку точок у бібліотеці Two.js використовується досить простий алгоритм, який включає створення векторів

та ліній між сусідніми точками. Для кожної пари послідовних точок у списку створюється вектор Two.js, який представляє собою лінію між двома точками координат.

2.5.1 Масштабування

Користувач може змінювати масштаб графіка, що впливає на відстань між одиницями сітки та розміром елементів. Зміна масштабу відбувається за допомогою керованого повзунка, що задає відносне збільшення чи зменшення значень zoomX та zoomY. Основний крок одиниці визначає відстань між лініями сітки або мітками на осях при вазовому масштабі. Новий крок одиниці обчислюється як добуток базового кроку і відношення поточного значення масштабу до базового масштабу. Це змінює частоту сітки та розмірність текстових міток. Після встановлення нових значень масштабу та зміщень полотно перемальовується (Рисунок 2.8).

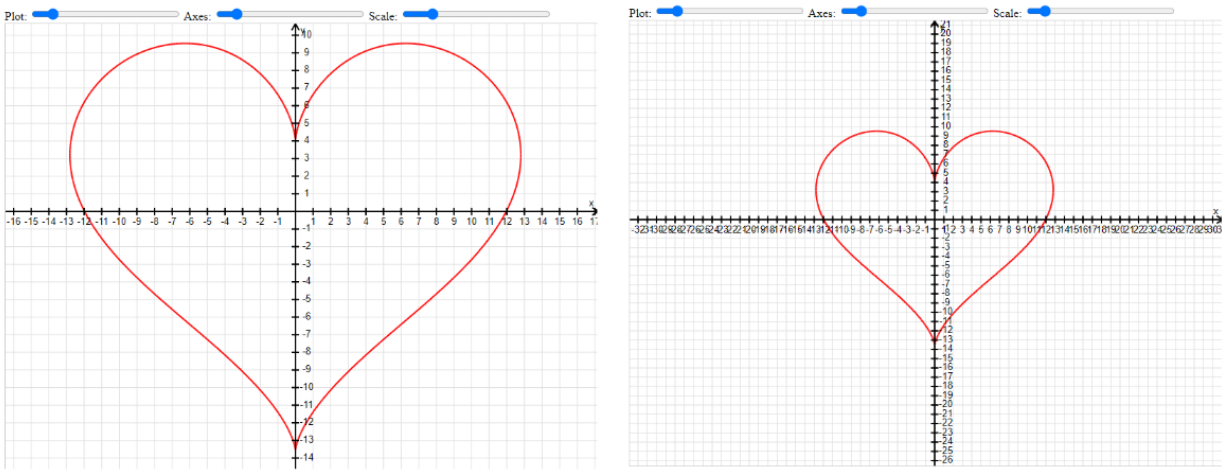

Рисунок 2.8 Масштабування параметричної функції

2.5.2 Зміна товщини ліній

Товщина ліній налаштовується користувачем для осей та функцій. Це забезпечує краще візуалізацію при різних масштабах. Атрибут *lineWidth* визначає товщину лінії в бібліотеці Two.js. Для зміни товщини ліній використовується цикл для перебору необхідних графічних об’єктів і оновленнях їхніх властивостей *lineWidth*.

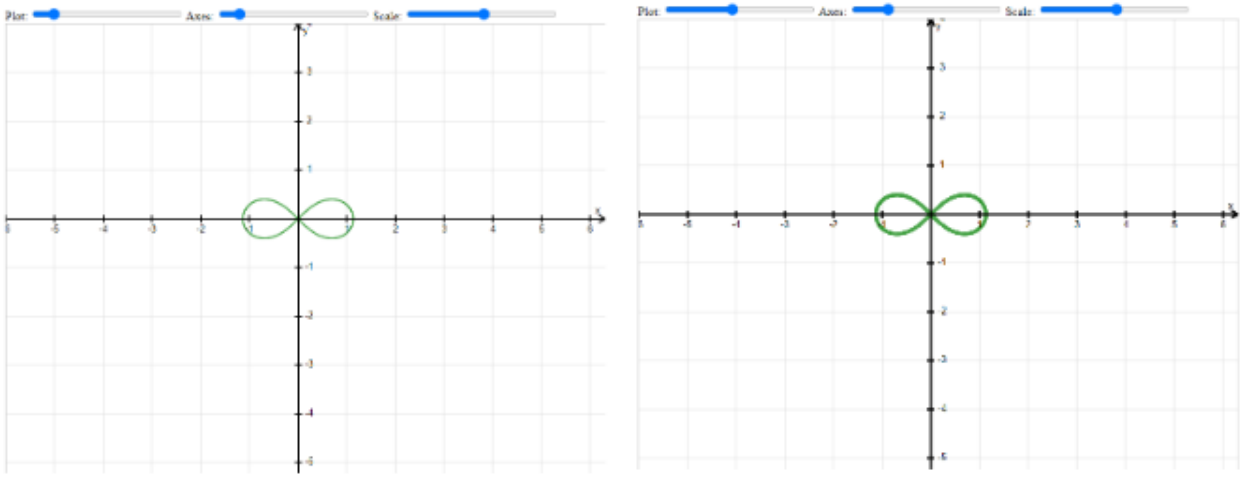

Рисунок 2.9 Зміна товщини ліній

2.5.3 Опції налаштування лінії графіку

MathPartner надає різні опції для налаштування візуального представлення графіків, включаючи відображення ліній у вигляді пунктир, зі стрілками на кінцях або комбінований стиль, який включає і пунктир, і стрілки (Рисунок 2.10).

Пунктирні лінії налаштовано через властивість dashes, що приймає масив із двома числами: довжиною штриха та довжиною проміжку. Для додавання стрілок на кінці ліній можна використовувати кастомні форми або попередньо визначені маркери SVG. Two.js не надає вбудованої підтримки стрілок, тому їх було створено.

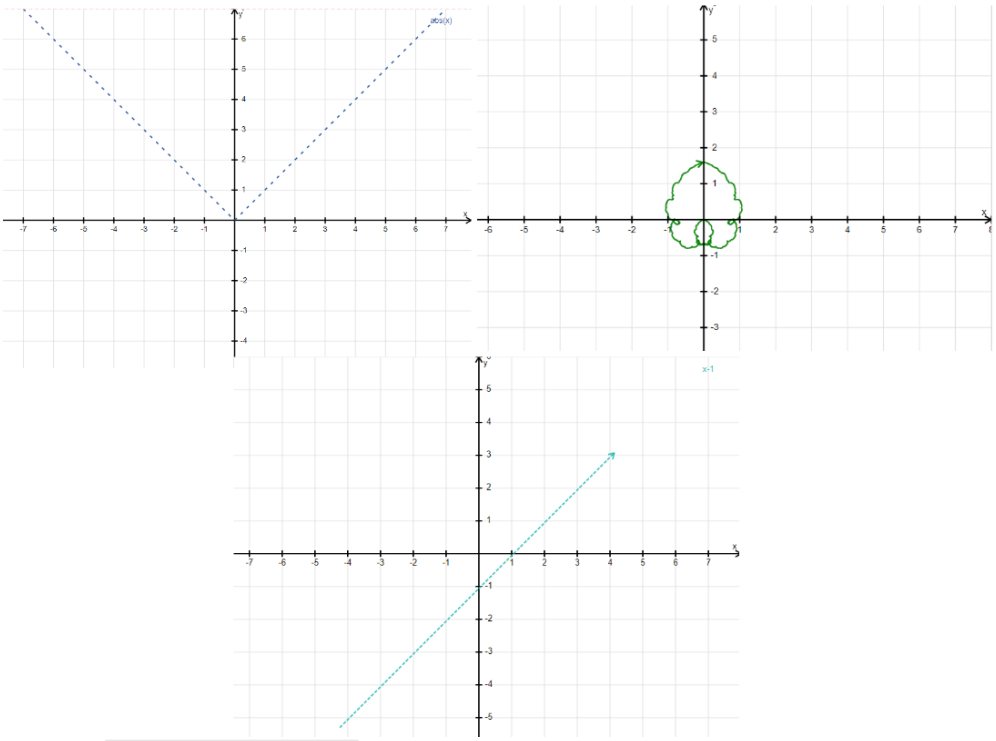

Рисунок 2.10 Опції налаштування лінії графіку

ВИСНОВКИ

У результаті виконання даної курсової роботи було досягнуто значних успіхів у вдосконаленні функціоналу веб-додатку MathPartner. Впровадженням клієнтського створення графіків за допомогою бібліотеки two.js, при збереженні логіки побудови графіків на сервері, вдалося значно підвищити швидкість відгуку системи та зменшити навантаження на серверну частину. Дане покращення дозволило не лише забезпечити більш ефективну взаємодію з користувачами завдяки покращенню якості візуалізації графіків, але й посприяло підвищенню загальної продуктивності платформи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Malaschonok G.I. Mathpar Language Guide : Study Guide / Malaschonok G.I.; Ministry of Education and Science of RF, FSBEI HPE ”Tambov State University named after G.R. Derzhavin”. Tambov: the Publishing House of TSU named after G.R. Derzhavin, 2013. 125 pp.
2. Гатцук, А.С. Побудова графіка функції, заданої параметрично / А.С. Гатцук, І.М. Беда // Сучасні технології в промисловому виробництві : матеріали Всеукраїнської міжвузівської науково-технічної конференції (Суми, 19 - 23 квітня 2010 року) / Редкол.: О.Г.Гусак, В.Г.Євтухов. — Суми : СумДУ, 2010. — Ч.ІІ. — С. 96.
3. Демчишин О. І. Вища математика : навч. посіб. / О. І. Демчишин, Б. Г. Шелестовський. – Тернопіль : Навчальна книга - Богдан, 2010. –592 с.
4. Огурцов А. П. Вища математика для підготовки бакалаврів з інженерії : навч. посіб. : у 3 ч. / А. П. Огурцов, Т. В. Наконечна, О. В. Нікулін; зазаг. ред. А. П. Огурцова. – Дніпродзержинськ : ДДТУ, 2008. – Ч. 1. – 428 с.; Ч. 2. – 340 с.; Ч. 3. – 320 с.
5. Лозовий Б. Л. Практикум з вищої математики : навч. посіб. / Б. Л. Лозовий, Я. С. Пушак, О. Є. Шабат. – Львів : «Магнолія – 2006», 2007. – 285 с.
6. jQuery API Documentation [Електронний ресурс]. – URL: https://api.jquery.com/
7. Веб-застосунок MathPartner [Електронний ресурс]. – URL: https://mathpar.com/
8. Документація бібліотеки Two.js [Електронний ресурс]. – URL: https://two.js.org/docs/two/
9. Документація обробки запитів SpringFramework [Електронний ресурс]. – URL: https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/web/bind/annotation/PostMapping.html

ДОДАТОК А

```
@RequestMapping(value = "/api/plot2dexplicit", method = RequestMethod.POST)
public ResponseEntity<List<double[]>> getMeshForExplicit2dPlot(
        @RequestBody MathparRequest mpReq, @PageParam Page page) {
    final String task = mpReq.getTask();
    if (!task.contains(F.FUNC_NAMES[F.EXPLICIT_PLOT2D])) {
                task, F.FUNC_NAMES[F.EXPLICIT_PLOT2D]);
        return new ResponseEntity<>(HttpStatus.BAD_REQUEST);
    }
    final List<double[]> data = new ExplicitPlot3dTimeoutRunner()
            .run(page, task, mpReq.getSectionId());
    if (data == null) {
        return new ResponseEntity<>(HttpStatus.INTERNAL_SERVER_ERROR);
    }
    return new ResponseEntity<>(data, HttpStatus.OK);
}
```